CI/CD and DevOps in 3 Weeks

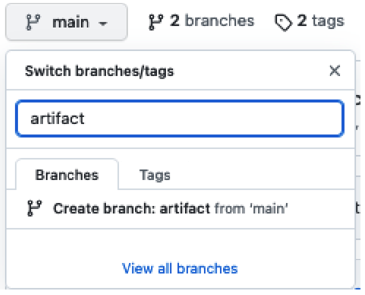
Week 2
Revision 1.2 – 08/14/22
Tech Skills Transformations LLC / Brent Laster

Lab 6 – Managing Artifacts

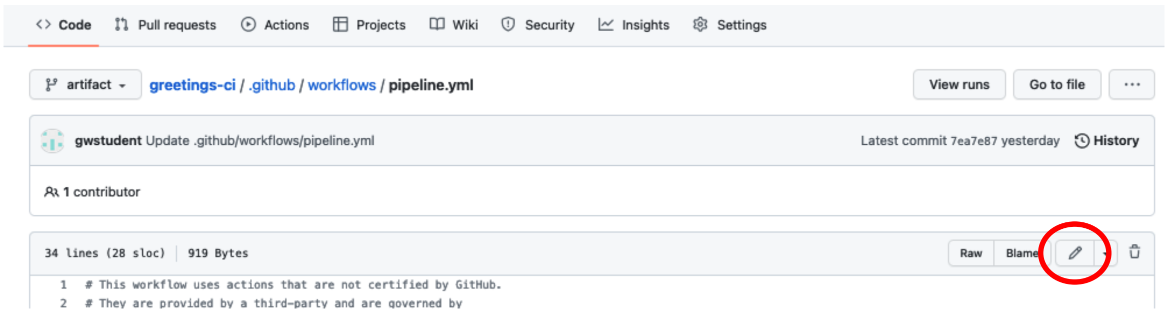
Purpose: In this lab, we'll look at how to do simple artifact management – an important part of Continuous Delivery.

1. Log in to GitHub with your primary GitHub id and go to your "greetings-ci" project that we used in week 1.

2. As a best practice for building out the pipeline as a larger project, let's create a separate branch to work in for managing the versioning and storage of the artifact. We'll call it "artifact". In the "Code" tab, click on the branch dropdown that says "main". Then in the text area that says "Find or create a branch…", enter the text "artifact". Then click on the **"Create branch: artifact from 'main'"** link.

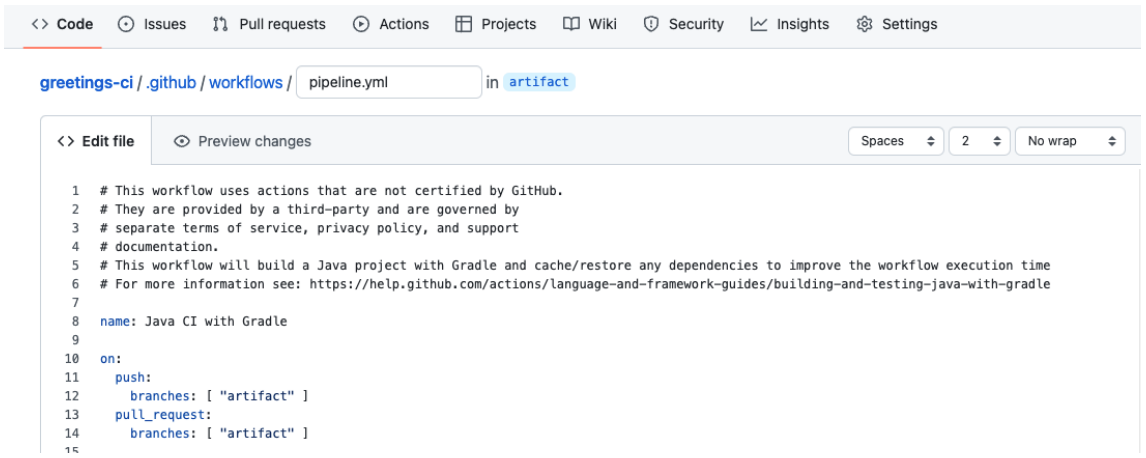


3. Now you should be on the "artifact" branch. We're going to first add the code to persist the artifact that we built in our build step. We want to persist this for use with other jobs in our pipeline such as ones that might test it. Open the .github/workflows/pipeline.yaml file (click on the name) and edit it by clicking on the pencil icon.


Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
artifact
greetings-ci / .github / workflows / pipeline.yml
View runs
Go to file
gwstudent Update .github/workflows/pipeline.yml
Latest commit 7ea7e87 yesterday
History
1 contributor
34 lines (28 sloc) | 919 Bytes
Raw
Blame

```
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
```

4. Change the references in the "on:" clause to be just the "artifact" branch so we don't trigger action runs on the other branches while we are working on this one. Make sure you are on the *artifact* branch before you proceed.

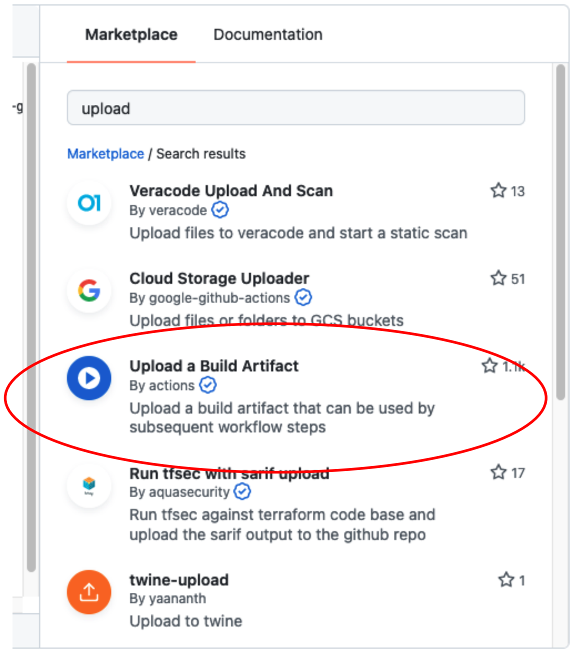


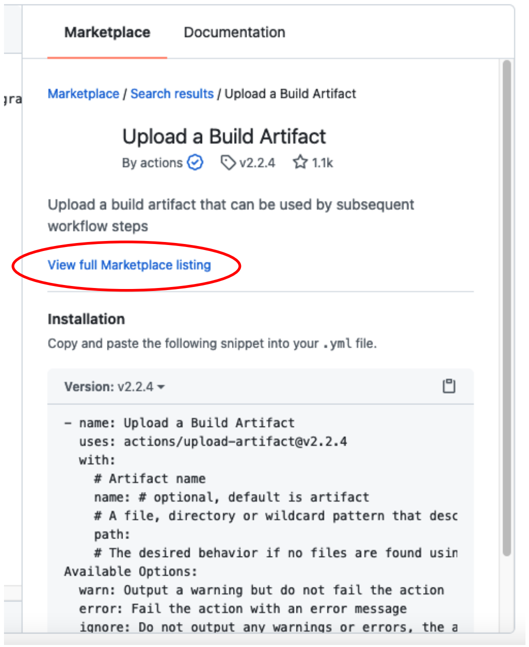
```
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.
# This workflow will build a Java project with Gradle and cache/restore any dependencies to improve the workflow execution time
# For more information see: https://help.github.com/actions/language-and-framework-guides/building-and-testing-java-with-gradle

name: Java CI with Gradle

on:
  push:
    branches: [ "artifact" ]
  pull_request:
    branches: [ "artifact" ]
```

5. As before, to the right, you should see a pane with references to GitHub actions. We're going to add a job to our workflow to upload an artifact. Let's find actions related to uploading.

   In the "Search Marketplace for Actions" box on the upper right, enter "Upload" and see what's returned.

   Next, click on the "Upload a Build Artifact" item. Take a look at the page that comes up from that. Let's look at the full listing on the Actions Marketplace. Click on the "View full Marketplace listing".

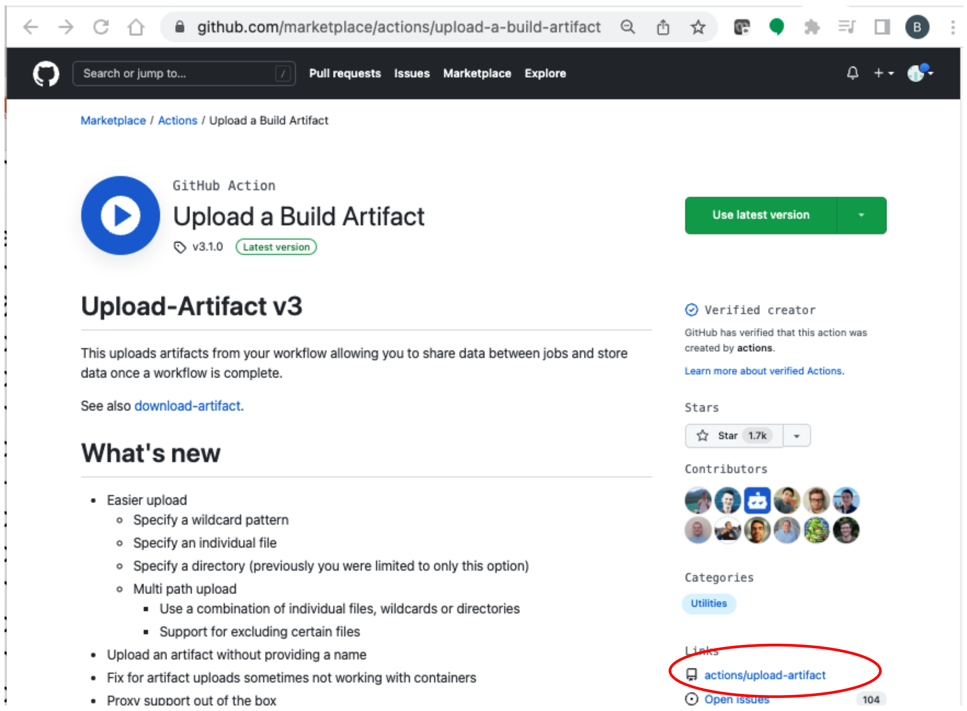

Marketplace
Documentation
upload
Marketplace / Search results
Veracode Upload And Scan
By veracode
Upload files to veracode and start a static scan
13
Cloud Storage Uploader
By google-github-actions
Upload files or folders to GCS buckets
51
Upload a Build Artifact
By actions
Upload a build artifact that can be used by subsequent workflow steps
1.1k
Run tfsec with sarif upload
By aquasecurity
Run tfsec against terraform code base and upload the sarif output to the github repo
17
twine-upload
By yaananth
Upload to twine
1


Marketplace
Documentation
Marketplace / Search results / Upload a Build Artifact
Upload a Build Artifact
By actions
v2.2.4
1.1k
Upload a build artifact that can be used by subsequent workflow steps
View full Marketplace listing
Installation
Copy and paste the following snippet into your .yml file.
Version: v2.2.4

```
- name: Upload a Build Artifact
  uses: actions/upload-artifact@v2.2.4
  with:
    # Artifact name
    name: # optional, default is artifact
    # A file, directory or wildcard pattern that desc
    path:
    # The desired behavior if no files are found usin
Available Options:
  warn: Output a warning but do not fail the action
  error: Fail the action with an error message
  ignore: Do not output any warnings or errors, the a
```

6. This should open up the full GitHub Actions Marketplace listing for this action. Notice the URL at the top - https://github.com/marketplace/actions/upload-a-build-artifact.

   Then click on the "actions/upload-artifact" link under "Links" in the lower right.


github.com/marketplace/actions/upload-a-build-artifact

Marketplace / Actions / Upload a Build Artifact

GitHub Action
Upload a Build Artifact
v3.1.0 Latest version

Use latest version

Upload-Artifact v3

This uploads artifacts from your workflow allowing you to share data between jobs and store data once a workflow is complete.

See also download-artifact.

What's new

- Easier upload
  - Specify a wildcard pattern
  - Specify an individual file
  - Specify a directory (previously you were limited to only this option)
  - Multi path upload
    - Use a combination of individual files, wildcards or directories
    - Support for excluding certain files
- Upload an artifact without providing a name
- Fix for artifact uploads sometimes not working with containers
- Proxy support out of the box

Verified creator
GitHub has verified that this action was created by **actions**.
Learn more about verified Actions.

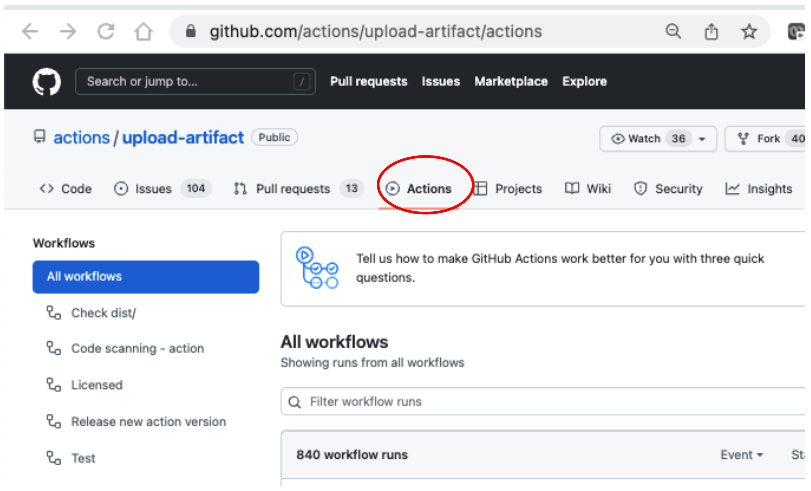
Stars
Star 1.7k

Contributors

Categories
Utilities

Links
actions/upload-artifact
Open issues 104

7. This will put you on the screen for the source code for this GitHub Action. Notice there is also an Actions button here. GitHub Actions use CI/CD themselves via GitHub Actions. Click on the Actions button to see the workflows that are in use/available


github.com/actions/upload-artifact/actions

actions / upload-artifact Public

Watch 36 Fork 40

Code Issues 104 Pull requests 13 Actions Projects Wiki Security Insights

Workflows
- All workflows
- Check dist/
- Code scanning - action
- Licensed
- Release new action version
- Test

Tell us how to make GitHub Actions work better for you with three quick questions.

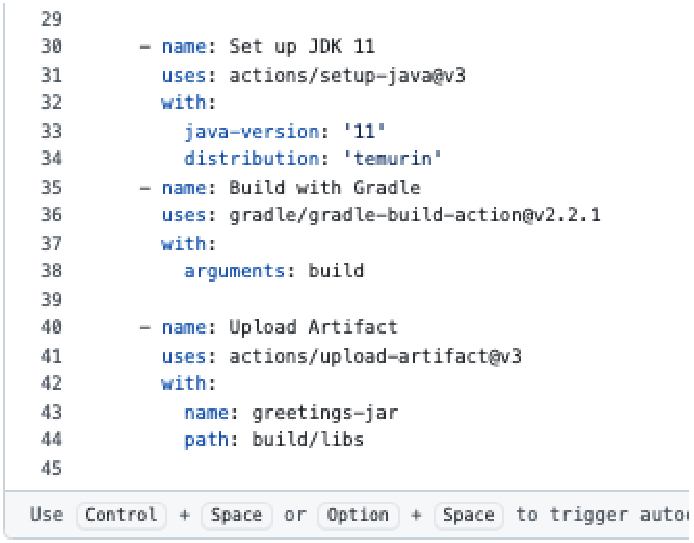
All workflows
Showing runs from all workflows

Filter workflow runs

840 workflow runs Event

8. Switch back to the browser tab where you are editing the workflow for greetings-actions. Update the build job to include a new step to use the "upload-artifact" action to upload the jar the build job creates. To do this, add the following lines inline with the build job steps. **Pay attention to the indenting.** See the screenshot (lines 40-44) for how this should look afterwards. (Your line numbers may be different.)

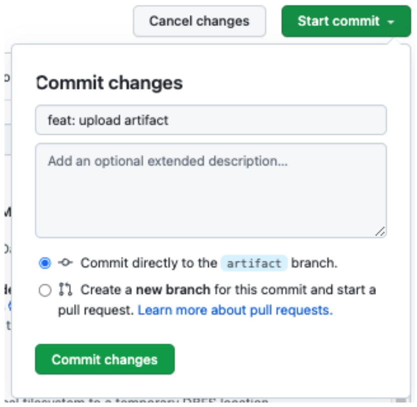
```
- name: Upload Artifact
  uses: actions/upload-artifact@v3
  with:
    name: greetings-jar
    path: build/libs
```



```
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: '11'
          distribution: 'temurin'
      - name: Build with Gradle
        uses: gradle/gradle-build-action@v2.2.1
        with:
          arguments: build

      - name: Upload Artifact
        uses: actions/upload-artifact@v3
        with:
          name: greetings-jar
          path: build/libs

```

Use Control + Space or Option + Space to trigger auto

9. Click on the green "Start commit" button in the upper right. In the dialog that comes up, add a commit message like "feat: upload artifact", then click the green "Commit changes" button to make the commit.



Commit changes

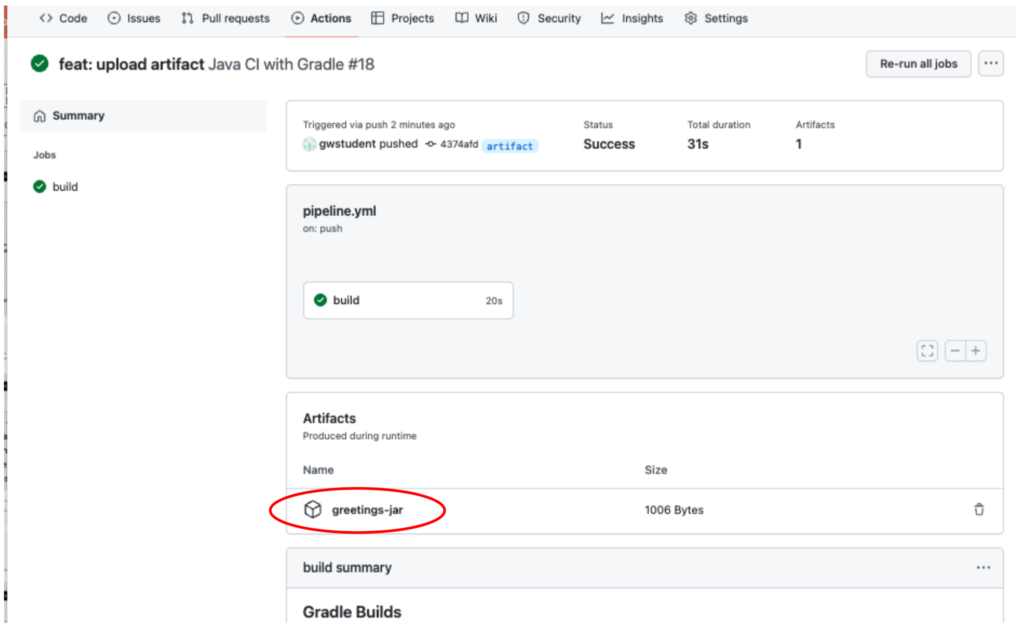
feat: upload artifact

Add an optional extended description...

(●) Commit directly to the `artifact` branch.

( ) Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

10. Switch to the "Actions" tab in your repository to see the workflow run. After a few moments, you should see that the run was successful. Click on the title of that run "feat: upload artifact". On the next screen, in addition to the graph, there will be a new section called "Artifacts" around the middle of the page. You can download the artifact from there. Click on the name of the artifact to try this.



feat: upload artifact Java CI with Gradle #18

Triggered via push 2 minutes ago
gwstudent pushed 4374afd artifact
Status Success
Total duration 31s
Artifacts 1

pipeline.yml
on: push
build 20s

Artifacts
Produced during runtime

Name	Size
greetings-jar	1006 Bytes

build summary

Gradle Builds

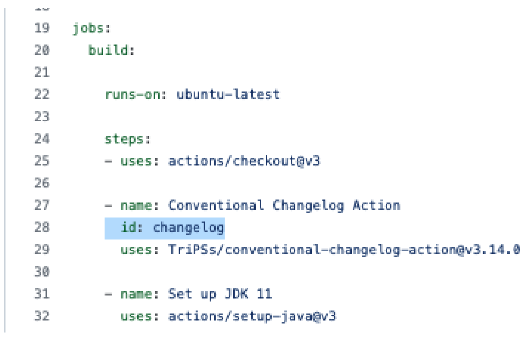
Lab 7 – Versioning Artifacts

Purpose: In this lab, we'll look at how to do simple artifact versioning to better keep track of what we produce and can use in our CI/CD processes.

1. Our artifact is being uploaded now, but we need to have each instance from a run of our pipeline clearly versioned so we can easily track changes and get back to specific versions if we need. For simplicity, we'll use the same semantic versioning scheme and value provided by our changelog generation process - specifically we'll use the "version" output from the changelog step.

2. To reference the output from a step, we need to assign the step an "id". We can't just use the name. So, edit the pipeline.yaml file again in the usual way and add the line in **bold** below in the "Conventional Changelog Action" step. See the screen capture below for a reference of where to add this. **Make sure you are on the "artifact" branch again!**

```
- name: Conventional Changelog Action
  id: changelog
  uses: TriPSs/conventional-changelog-action@v3.14.0
```

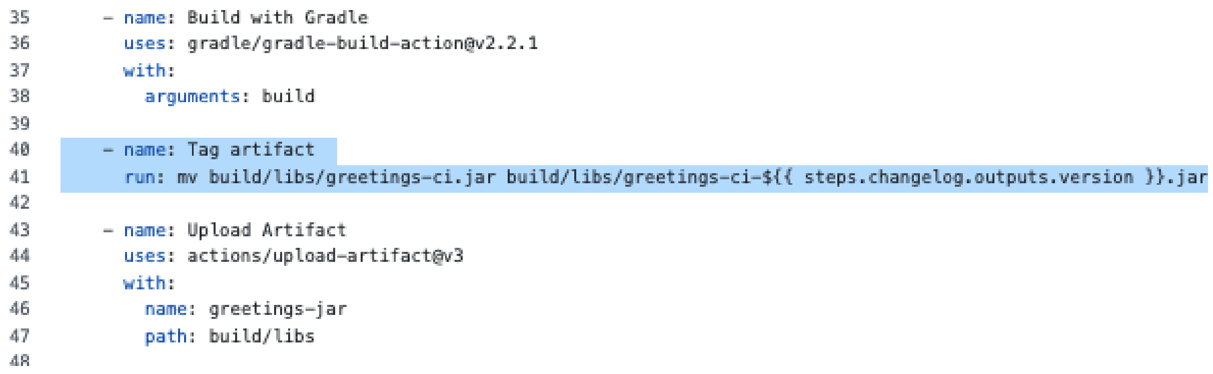

```
jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Conventional Changelog Action
      id: changelog
      uses: TriPSs/conventional-changelog-action@v3.14.0

    - name: Set up JDK 11
      uses: actions/setup-java@v3
```

3. Now, we need to construct the reference to get the version output from the changelog step. This is pretty straightforward. The reference looks like this (where *changelog* is the id we added in the previous step): (We're just looking at code here, we'll make the actual changes in the next step)

```
${{ steps.changelog.outputs.version }}.jar
```

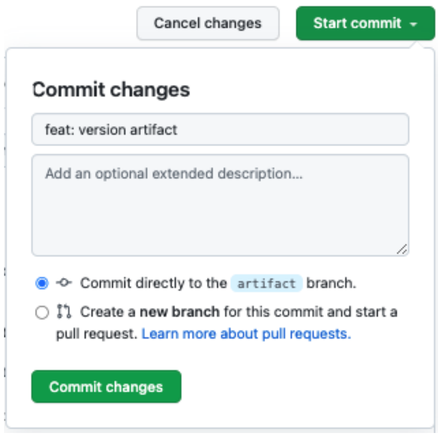
4. To make this simple, we'll just add a step to tag the artifact with the version by renaming it to include the version number. We'll want to add this after the build has produced the artifact and before we upload it. Add the two lines for a new step in the code after the build step and before the upload step. (Note the second line does not need to be split in your code - it just displays that way because of the length.)

```
  - name: Tag artifact
      run: mv build/libs/greetings-ci.jar build/libs/greetings-ci-${{
steps.changelog.outputs.version }}.jar
```



```
      - name: Build with Gradle
        uses: gradle/gradle-build-action@v2.2.1
        with:
          arguments: build

      - name: Tag artifact
        run: mv build/libs/greetings-ci.jar build/libs/greetings-ci-${{ steps.changelog.outputs.version }}.jar

      - name: Upload Artifact
        uses: actions/upload-artifact@v3
        with:
          name: greetings-jar
          path: build/libs
```

5. Now we can commit this with a commit message that will trigger a new version, for example: "feat: version artifact". Go ahead and do the commit.



Cancel changes | Start commit

Commit changes

feat: version artifact

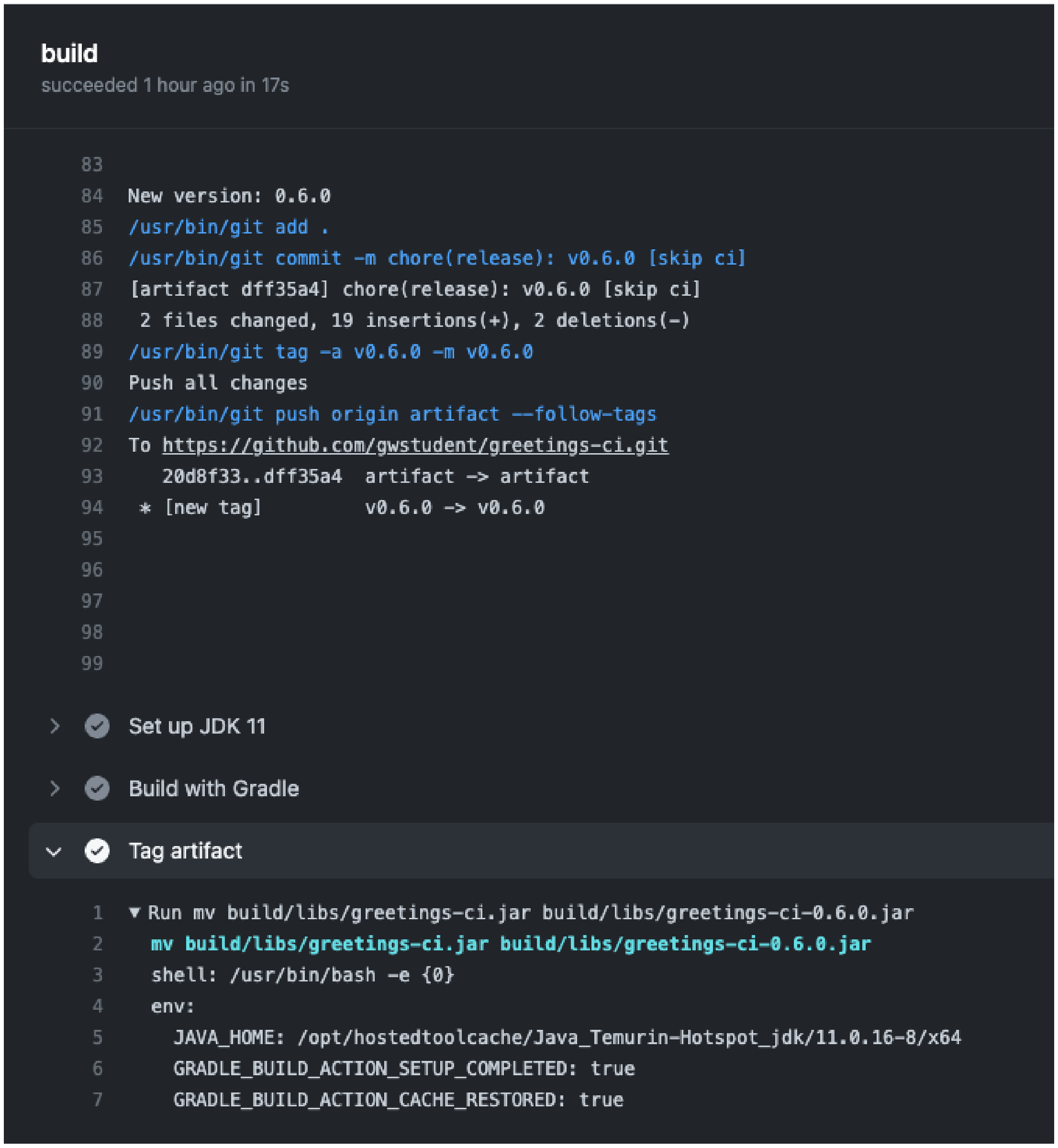
Add an optional extended description...

(●) Commit directly to the `artifact` branch.

( ) Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Commit changes

6. After this commit, there will be a new run of the workflow and the build job. If you select the build job in the workflow and expand the conventional changelog steps and the "Tag artifact" step, you'll be able to see the newly generated version and the rename that occurs.



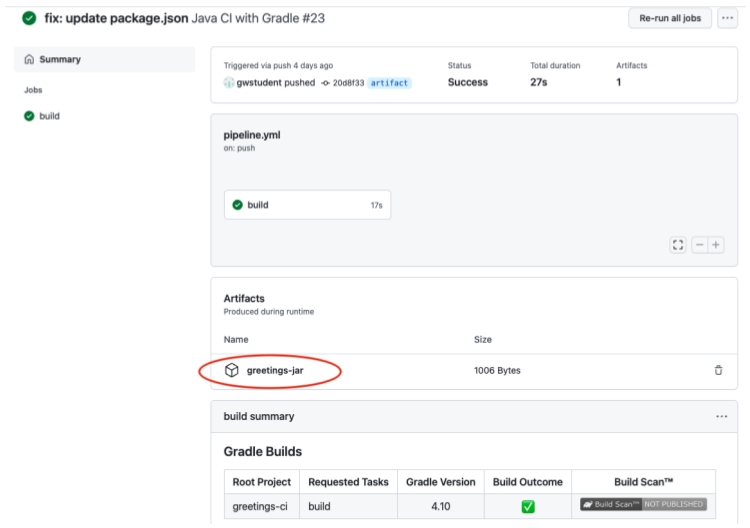
build
succeeded 1 hour ago in 17s

```

New version: 0.6.0
/usr/bin/git add .
/usr/bin/git commit -m chore(release): v0.6.0 [skip ci]
[artifact dff35a4] chore(release): v0.6.0 [skip ci]
 2 files changed, 19 insertions(+), 2 deletions(-)
/usr/bin/git tag -a v0.6.0 -m v0.6.0
Push all changes
/usr/bin/git push origin artifact --follow-tags
To https://github.com/gwstudent/greetings-ci.git
   20d8f33..dff35a4  artifact -> artifact
 * [new tag]         v0.6.0 -> v0.6.0
```

```
▼ Run mv build/libs/greetings-ci.jar build/libs/greetings-ci-0.6.0.jar
  mv build/libs/greetings-ci.jar build/libs/greetings-ci-0.6.0.jar
  shell: /usr/bin/bash -e {0}
  env:
    JAVA_HOME: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/11.0.16-8/x64
    GRADLE_BUILD_ACTION_SETUP_COMPLETED: true
    GRADLE_BUILD_ACTION_CACHE_RESTORED: true
```

(Note: If you run into an issue where a new tag is not generated, you can try editing the package.json file and updating the version in it to be higher than the hightest current tag on the repo.)

7. If you go to the page for the run of the action, you can see the new jar in the *Artifacts* section. You can click on it and download it and extract it to see the actual versioned artifact that was created.



fix: update package.json Java CI with Gradle #23

Triggered via push 4 days ago — gwstudent pushed 20d8f33 artifact

Status	Total duration	Artifacts
Success	27s	1

pipeline.yml
on: push

build 17s

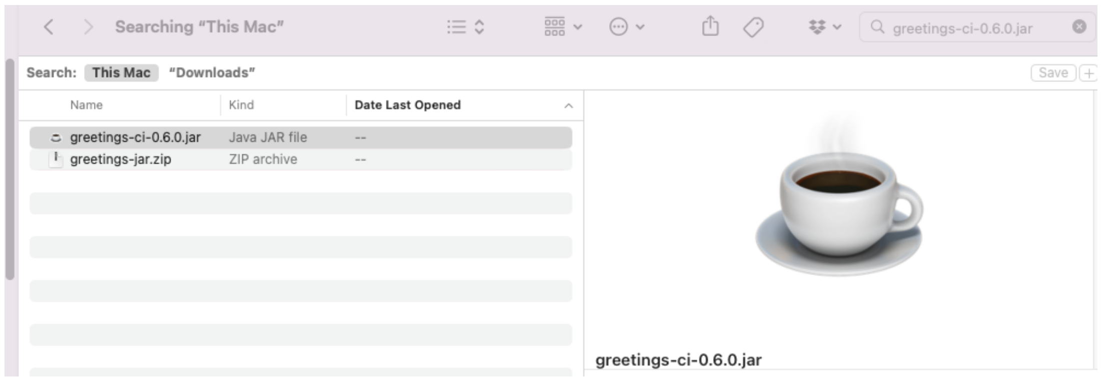
Artifacts
Produced during runtime

Name	Size
greetings-jar	1006 Bytes

build summary

Gradle Builds

Root Project	Requested Tasks	Gradle Version	Build Outcome	Build Scan™
greetings-ci	build	4.10	✓	Build Scan™ NOT PUBLISHED



Searching "This Mac"

Search: This Mac "Downloads"

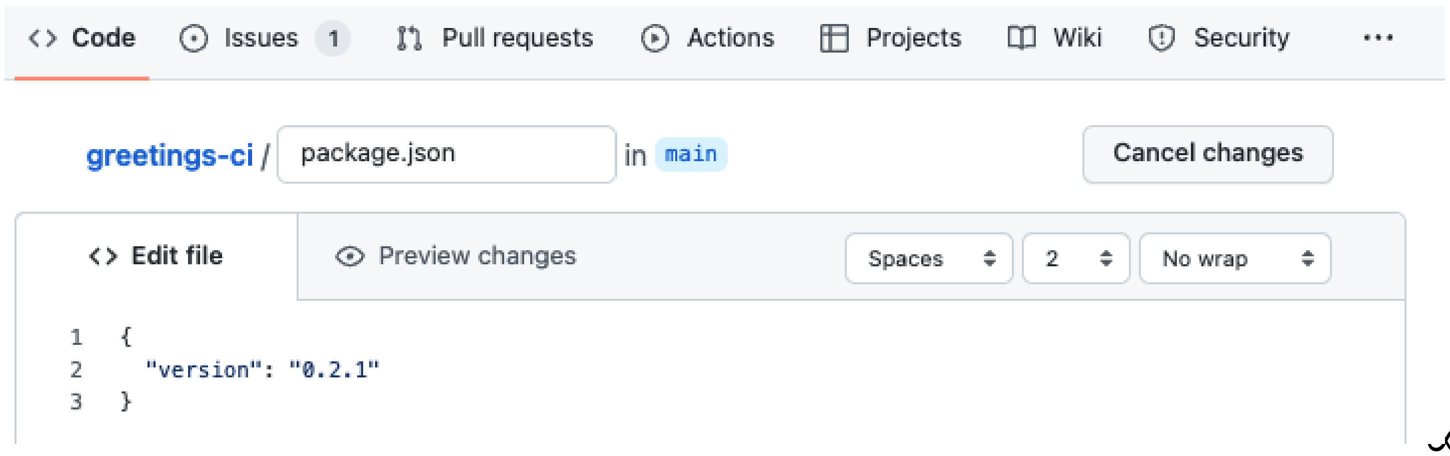
Name	Kind	Date Last Opened
greetings-ci-0.6.0.jar	Java JAR file	--
greetings-jar.zip	ZIP archive	--

greetings-ci-0.6.0.jar

Lab 8 – Merging branches

Purpose: In this lab, we'll look at how to do a Pull Request against branches, deal with conflicts and how to use the Visual Studio web editor.

1. Let's force an update of a file in main to make sure we have a merge conflict. For simplicity, we'll just update the "package.json" file in the main branch. Switch to the branch "main" and **edit the package.json file and bump one of the revision numbers and then commit the change.**

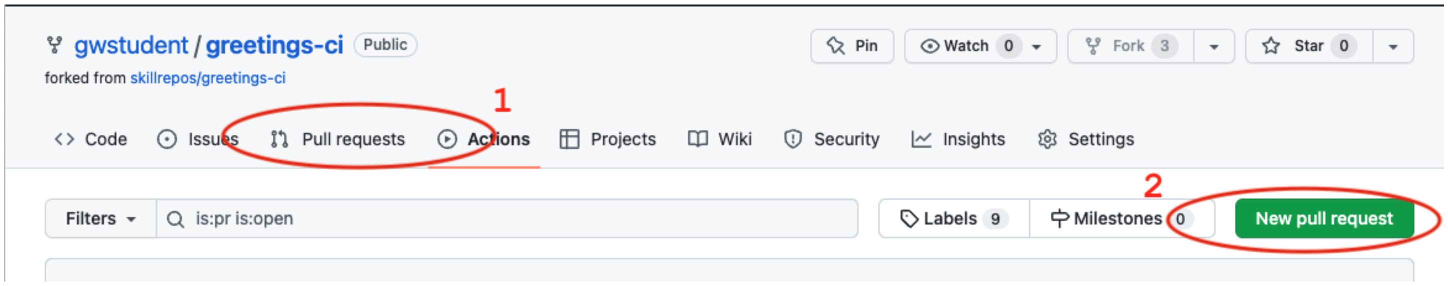


greetings-ci / package.json in main

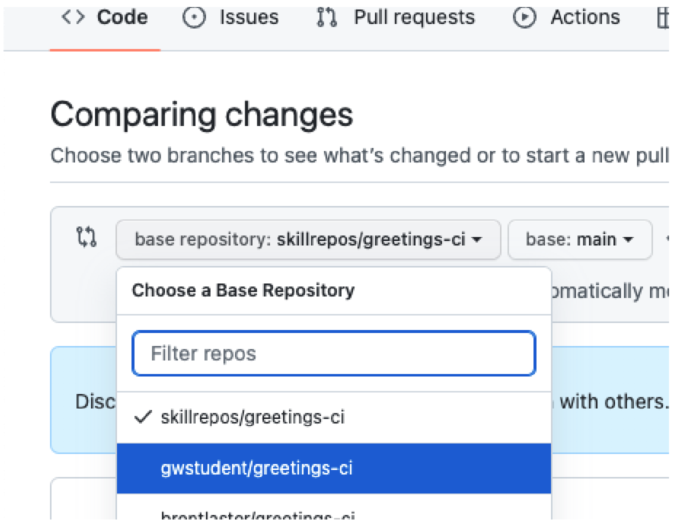
Spaces | 2 | No wrap

```
{
  "version": "0.2.1"
}
```

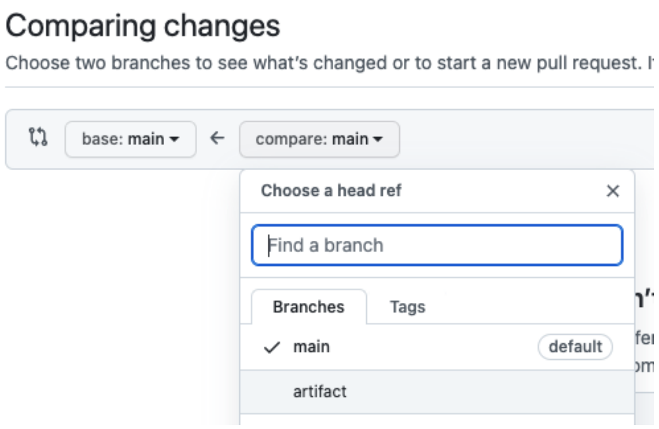
2. Going back to our enhancements for artifact versioning, now that we've proven that the change works in our *artifact* branch, let's merge that branch back into our primary workflow in the *main* branch. In the greetings-ci repo, click on the top-level "Pull requests" menu and then click on "New Pull Request".



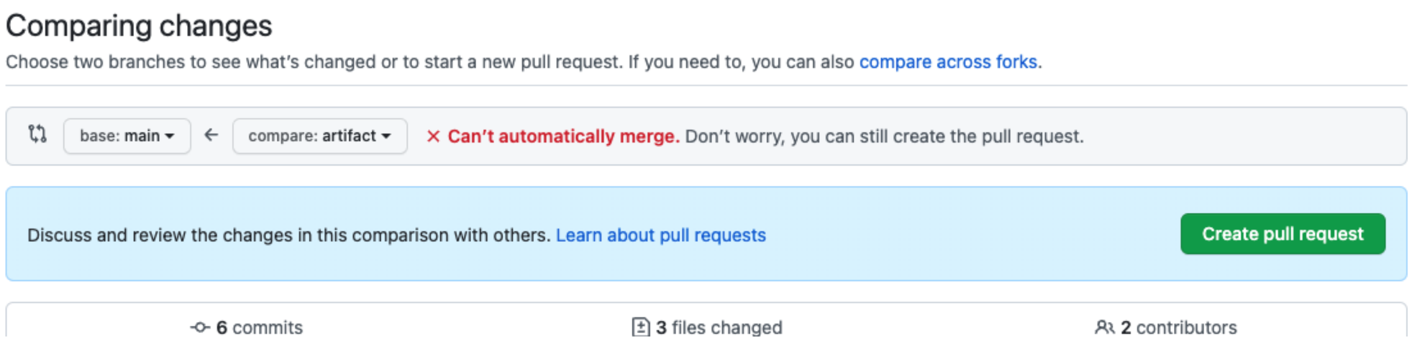
3. This will default to creating a PR to merge back to the original project that this was forked from. We need to adjust the Base Repository back to this one. Click on the "base repository" dropdown and select the "<your primary github userid>/greetings-ci" entry.



4. Then the comparison will switch to ones for branches within the repository. On the "compare:" side, select the "artifact" branch.



5. You'll get a prompt that says "Can't automatically merge". We'll fix that shortly. Go ahead and click the green "Create pull request" button. Enter a comment if you want and then just click the second "Create pull request" button.



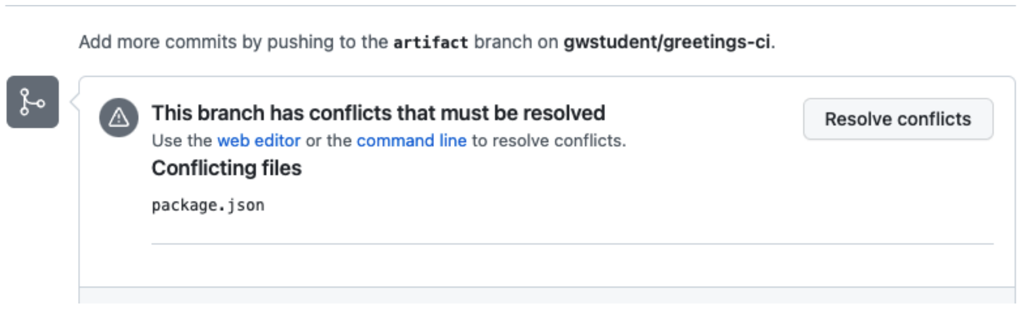
Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

✕ Can't automatically merge. Don't worry, you can still create the pull request.

Discuss and review the changes in this comparison with others. Learn about pull requests

6. On the next screen, click on the "Resolve conflicts" button.



Add more commits by pushing to the **artifact** branch on **gwstudent/greetings-ci**.

This branch has conflicts that must be resolved

Use the web editor or the command line to resolve conflicts.

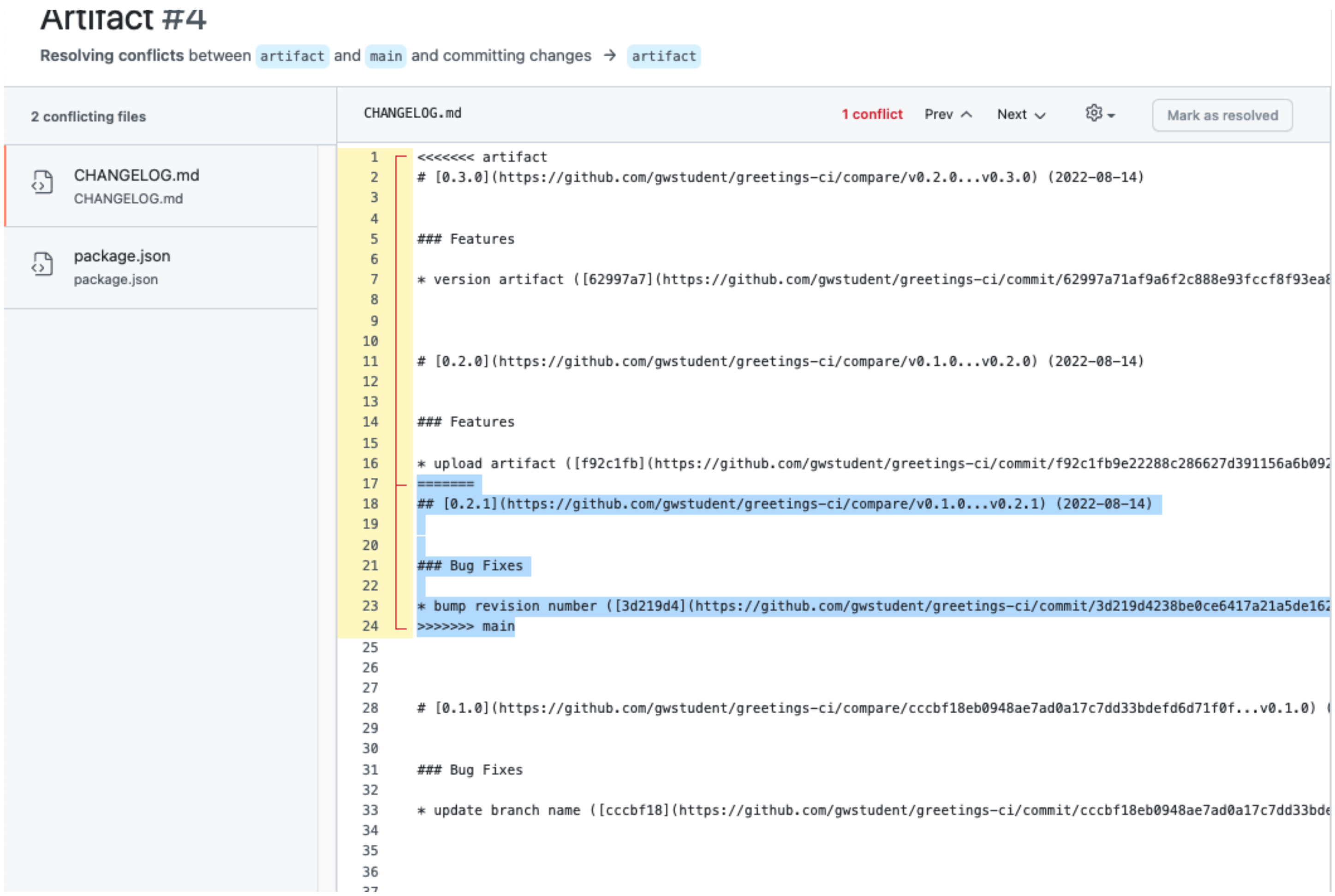
Conflicting files

`package.json`

7. The next screen will be the conflict for CHANGELOG.md and package.json. You will already be in the edit screen for this conflict. Just click on each file and highlight the lines you want to remove and Delete them. You can highlight the lines between and including the "========" and ">>>>>>>>> main" lines and then hit the "Delete" key. Also you can delete the line labeled "<<<<<< artifact" at the top.

   After that, you can click on the "Mark as resolved" button.

   Then just click on the package.json selection on the left and repeat the process.


Artifact #4

Resolving conflicts between `artifact` and `main` and committing changes → `artifact`

2 conflicting files

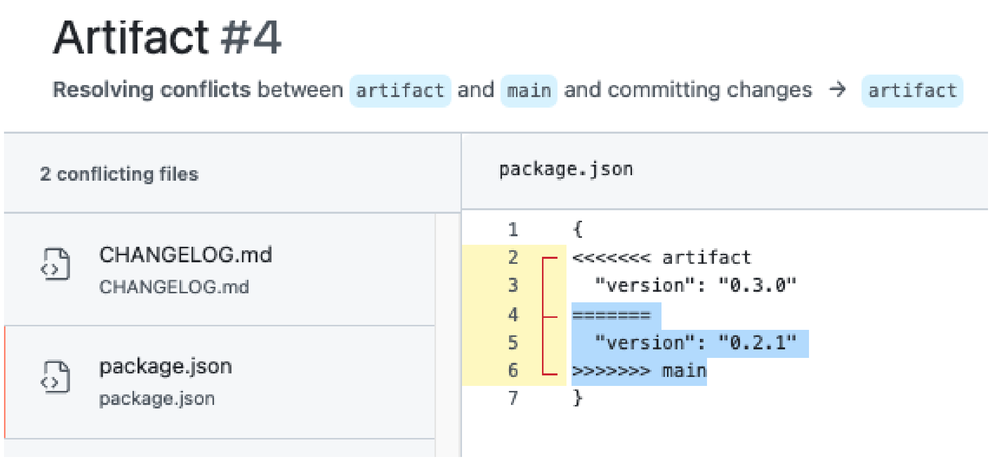
CHANGELOG.md
CHANGELOG.md

package.json
package.json

CHANGELOG.md

1 conflict

```
<<<<<<< artifact
# [0.3.0](https://github.com/gwstudent/greetings-ci/compare/v0.2.0...v0.3.0) (2022-08-14)


### Features

* version artifact ([62997a7](https://github.com/gwstudent/greetings-ci/commit/62997a71af9a6f2c888e93fccf8f93ea8



# [0.2.0](https://github.com/gwstudent/greetings-ci/compare/v0.1.0...v0.2.0) (2022-08-14)


### Features

* upload artifact ([f92c1fb](https://github.com/gwstudent/greetings-ci/commit/f92c1fb9e22288c286627d391156a6b092
=======
## [0.2.1](https://github.com/gwstudent/greetings-ci/compare/v0.1.0...v0.2.1) (2022-08-14)


### Bug Fixes

* bump revision number ([3d219d4](https://github.com/gwstudent/greetings-ci/commit/3d219d4238be0ce6417a21a5de162
>>>>>>> main



# [0.1.0](https://github.com/gwstudent/greetings-ci/compare/cccbf18eb0948ae7ad0a17c7dd33bdefd6d71f0f...v0.1.0) (


### Bug Fixes

* update branch name ([cccbf18](https://github.com/gwstudent/greetings-ci/commit/cccbf18eb0948ae7ad0a17c7dd33bde
```


Artifact #4

Resolving conflicts between `artifact` and `main` and committing changes → `artifact`

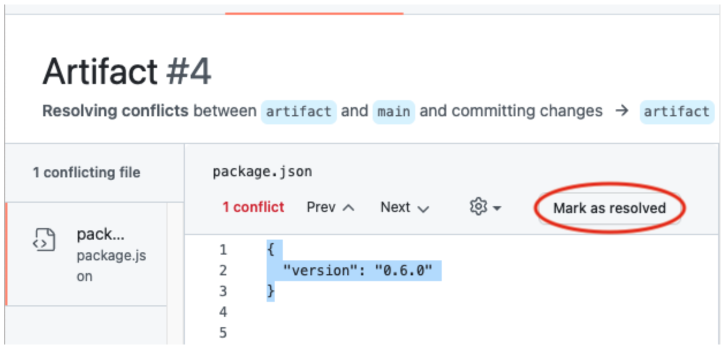
2 conflicting files

CHANGELOG.md
CHANGELOG.md

package.json
package.json

package.json

```
{
<<<<<<< artifact
  "version": "0.3.0"
=======
  "version": "0.2.1"
>>>>>>> main
}
```


Artifact #4

Resolving conflicts between `artifact` and `main` and committing changes → `artifact`

1 conflicting file

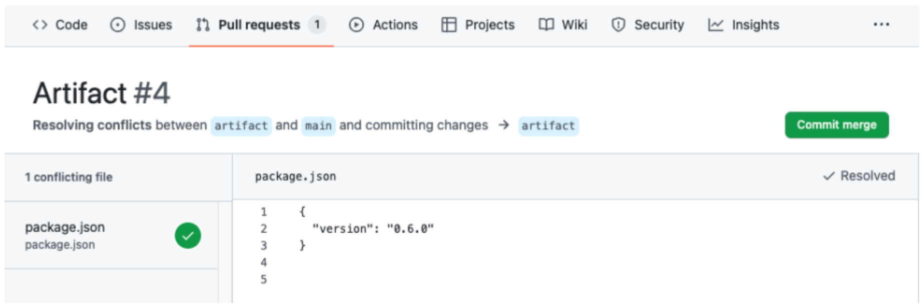
pack...
package.json

package.json

1 conflict

```
{
  "version": "0.6.0"
}
```

8. Now click on the green "Commit merge" button to the right.



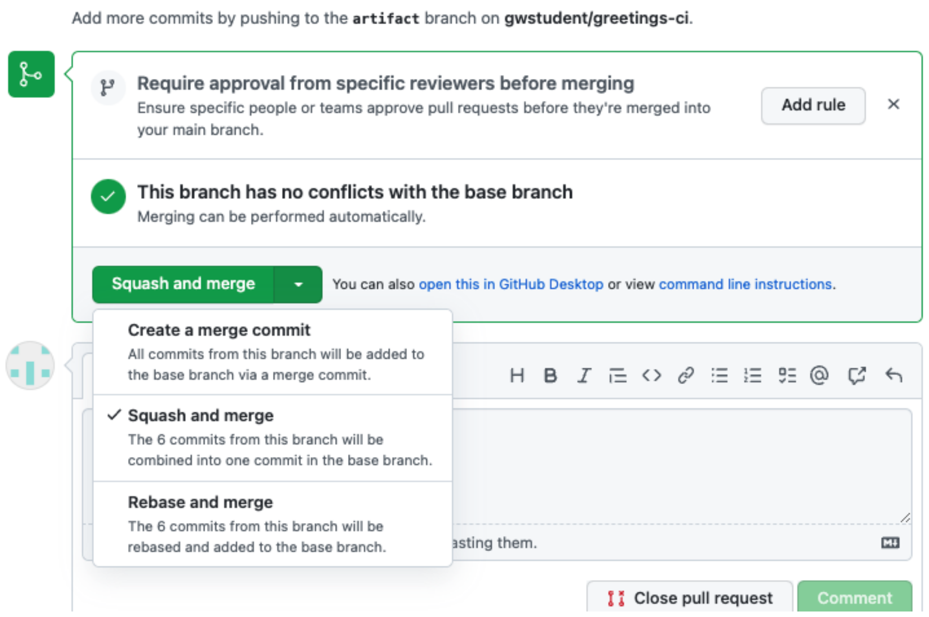
Artifact #4

Resolving conflicts between `artifact` and `main` and committing changes → `artifact`

package.json
package.json

```
{
  "version": "0.6.0"
}
```

9. After a few moments, the pre-check should complete. Before we merge the pull request, let's tell GitHub to squash all our commits into one. Click the down arrow next to the "Merge pull request" button and select "Squash and merge". Afterwards, the button should say "Squash and merge" - now click on that.



Add more commits by pushing to the **artifact** branch on **gwstudent/greetings-ci**.

Require approval from specific reviewers before merging
Ensure specific people or teams approve pull requests before they're merged into your main branch.

This branch has no conflicts with the base branch
Merging can be performed automatically.

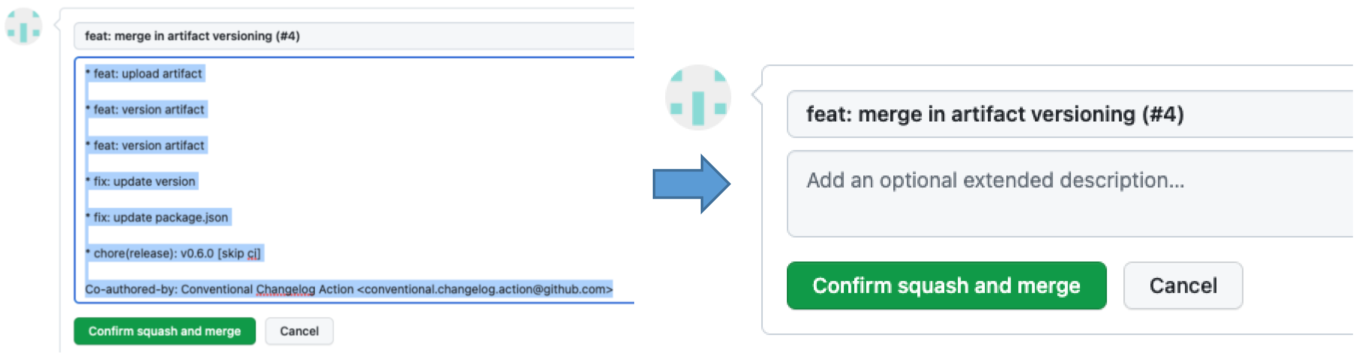
You can also open this in GitHub Desktop or view command line instructions.

Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

✓ **Squash and merge**
The 6 commits from this branch will be combined into one commit in the base branch.

Rebase and merge
The 6 commits from this branch will be rebased and added to the base branch.

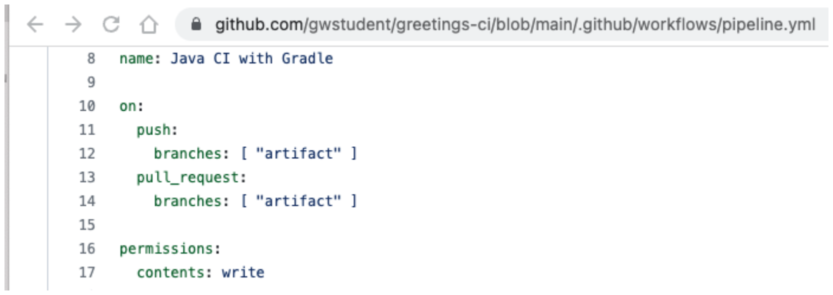
10. We don't need all the history from the previous commits, so you can just clear out the comment window except for the most recent commit message specifically for the PR and then click on the "Confirm squash and merge" button.



```
feat: merge in artifact versioning (#4)

* feat: upload artifact

* feat: version artifact

* feat: version artifact

* fix: update version

* fix: update package.json

* chore(release): v0.6.0 [skip ci]

Co-authored-by: Conventional Changelog Action <conventional.changelog.action@github.com>
```

```
feat: merge in artifact versioning (#4)

Add an optional extended description...
```

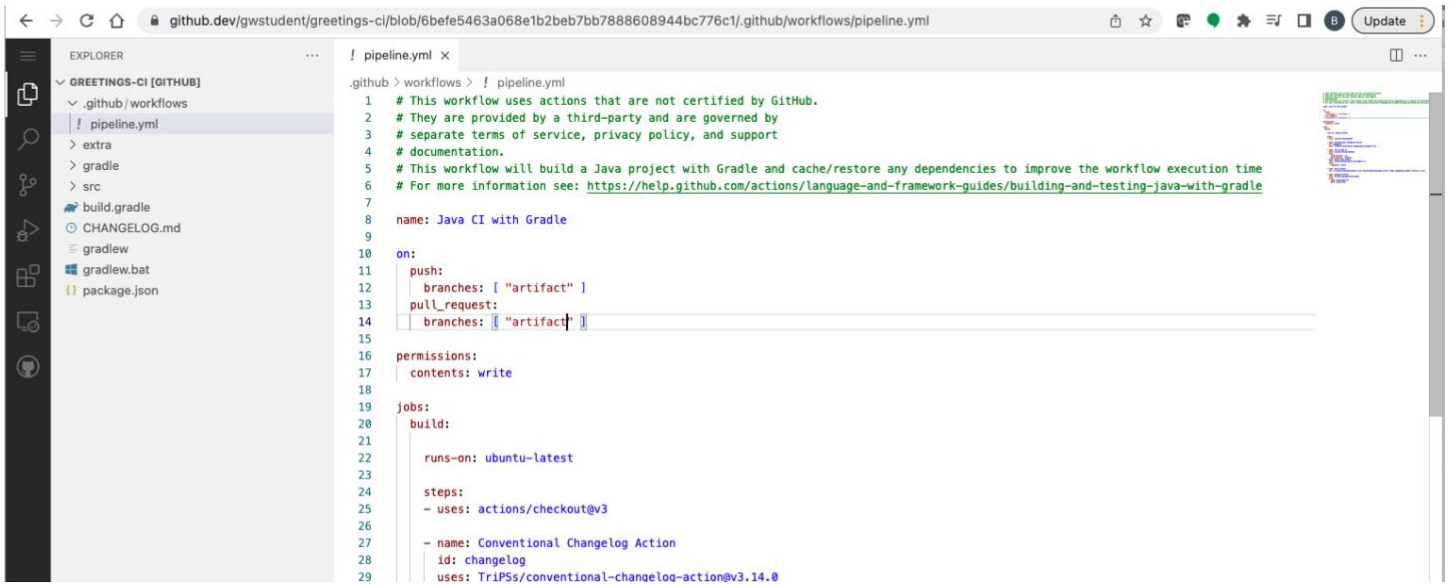
11. We might have expected that the merge would have kicked off another run of our workflow. Why didn't it? Take a look at the updated pipeline.yaml file that is now in main. Notice that the branch that it is triggered on was set to "artifact" when we just did the merge.



github.com/gwstudent/greetings-ci/blob/main/.github/workflows/pipeline.yml

```
name: Java CI with Gradle

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

permissions:
  contents: write
```

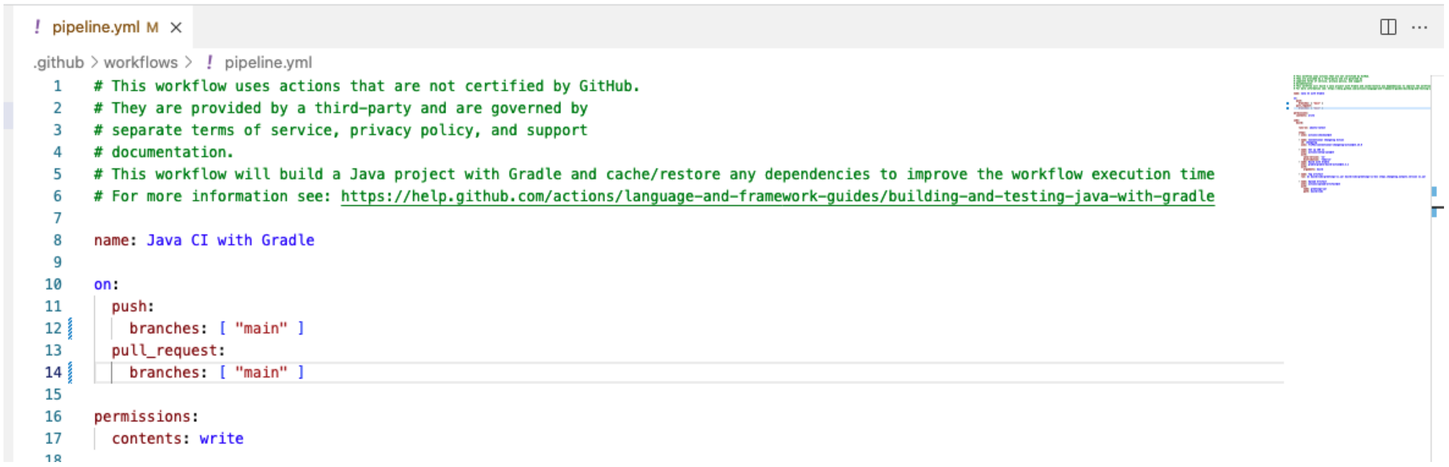
12. We need to correct that so it is set to trigger on "main" again. We'll edit the pipeline.yaml file in main and change the branch "artifact" references to be "main" instead. Instead of clicking on the pencil icon to edit, change to the file and just hit "." on your keyboard. This will open the VSCode editor in the browser.



github.dev/gwstudent/greetings-ci/blob/6befe5463a068e1b2beb7bb7888608944bc776c1/.github/workflows/pipeline.yml

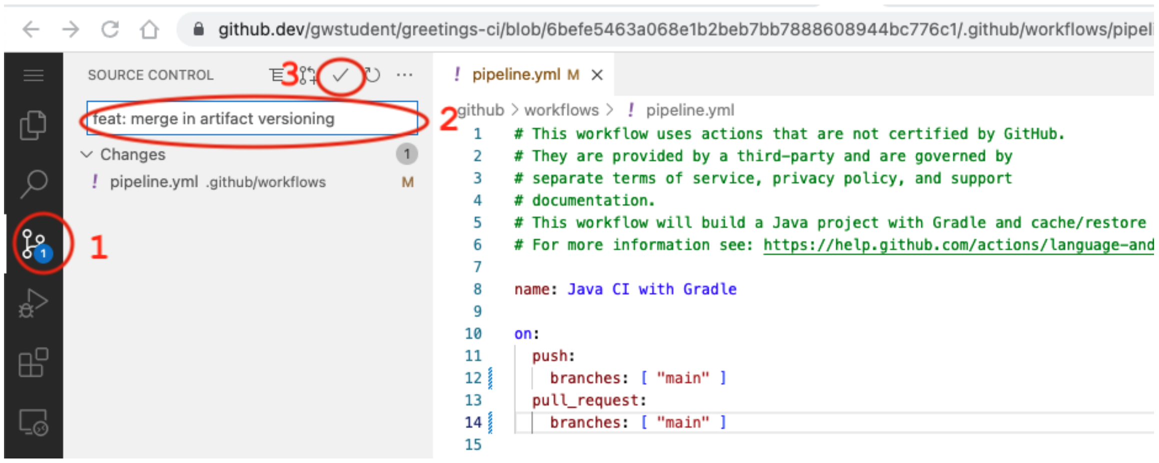
```
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.
# This workflow will build a Java project with Gradle and cache/restore any dependencies to improve the workflow execution time
# For more information see: https://help.github.com/actions/language-and-framework-guides/building-and-testing-java-with-gradle

name: Java CI with Gradle

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

permissions:
  contents: write

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Conventional Changelog Action
      id: changelog
      uses: TriPSs/conventional-changelog-action@v3.14.0
```

13. Change the references for branches to be "main" instead of "artifact" in the editor.



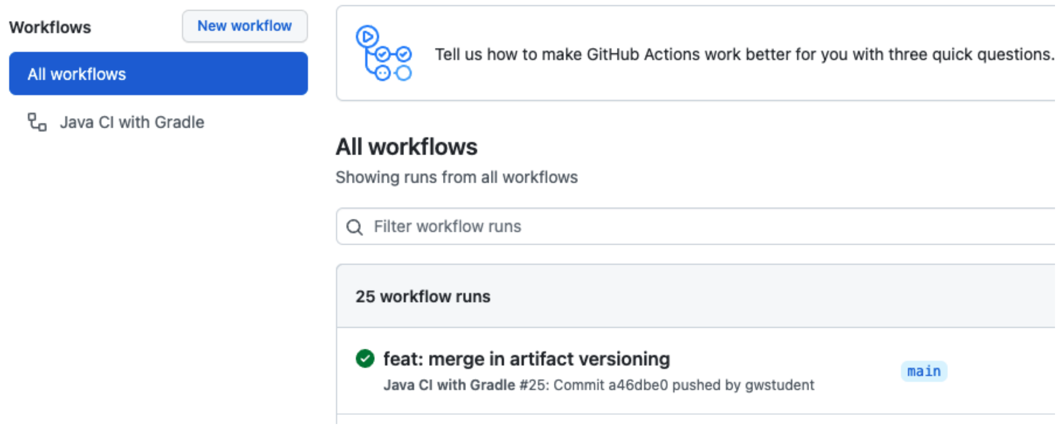
```yaml
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.
# This workflow will build a Java project with Gradle and cache/restore any dependencies to improve the workflow execution time
# For more information see: https://help.github.com/actions/language-and-framework-guides/building-and-testing-java-with-gradle

name: Java CI with Gradle

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

permissions:
  contents: write
```

14. In the lefthand column, select the branching symbol to change to the source control menu. Click in the commit message area and enter "feat: merge in artifact versioning" and then click the checkmark above to commit and push the changes.

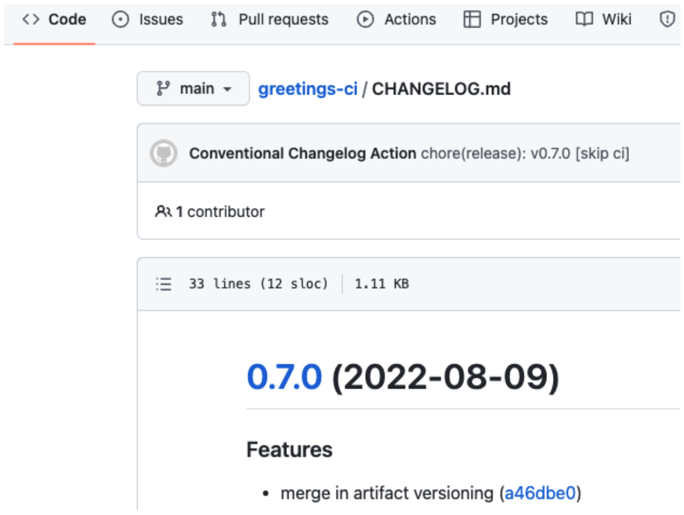

github.dev/gwstudent/greetings-ci/blob/6befe5463a068e1b2beb7bb7888608944bc776c1/.github/workflows/pipeli
SOURCE CONTROL
3
feat: merge in artifact versioning
2
Changes
1
! pipeline.yml .github/workflows
M
1
! pipeline.yml M
github > workflows > ! pipeline.yml

```yaml
# 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.
# This workflow will build a Java project with Gradle and cache/restore
# For more information see: https://help.github.com/actions/language-and

name: Java CI with Gradle

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

15. After this, your change will be committed. You can just go back in the browser to get out of the editor. When you go to the Actions menu, a new run of the workflow should have completed and if you look at the CHANGELOG file, you should see a new minor version update.


Workflows
New workflow
All workflows
Java CI with Gradle
Tell us how to make GitHub Actions work better for you with three quick questions.
All workflows
Showing runs from all workflows
Filter workflow runs
25 workflow runs
main

feat: merge in artifact versioning
Java CI with Gradle #25: Commit a46dbe0 pushed by gwstudent



greetings-ci / CHANGELOG.md

Conventional Changelog Action chore(release): v0.7.0 [skip ci]

1 contributor

33 lines (12 sloc) | 1.11 KB

0.7.0 (2022-08-09)

Features

- merge in artifact versioning (a46dbe0)

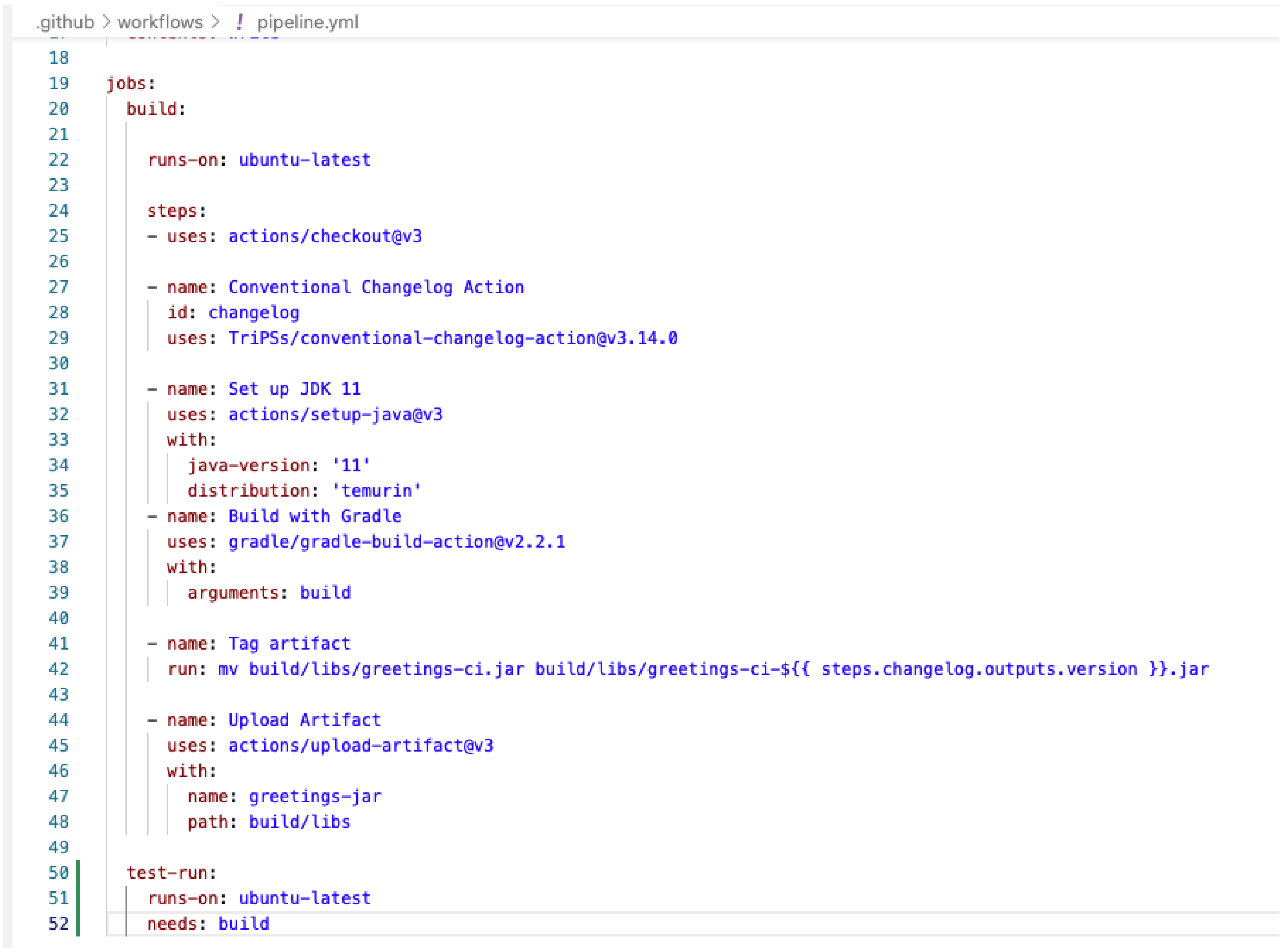
Lab 9 – Adding in a test case

Purpose: In this lab, we'll add a simple test case to download the artifact and verify it

1. Now let's add a second job to our workflow (in pipeline.yml) to do a simple "test". To start, add the job definition for a job called "test-run" that runs on ubuntu-latest. Since we want to test what we built, it will need to wait for the build job to be completed. That's what the "needs: build" part does.

   You can use either the built-in editor or the vscode editor to edit the file and add the code below. For simplicity, we'll just make the changes in the "main" branch. The screenshot shows where it should go. Pay attention to indentation - "test-run:" should line up with "build:".

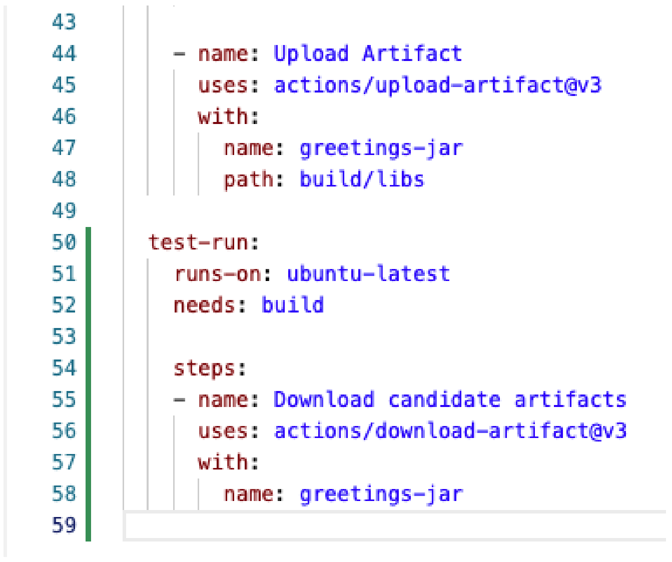
```
test-run:

  runs-on: ubuntu-latest
  needs: build
```


.github > workflows > ! pipeline.yml

```yaml

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Conventional Changelog Action
      id: changelog
      uses: TriPSs/conventional-changelog-action@v3.14.0

    - name: Set up JDK 11
      uses: actions/setup-java@v3
      with:
        java-version: '11'
        distribution: 'temurin'
    - name: Build with Gradle
      uses: gradle/gradle-build-action@v2.2.1
      with:
        arguments: build

    - name: Tag artifact
      run: mv build/libs/greetings-ci.jar build/libs/greetings-ci-${{ steps.changelog.outputs.version }}.jar

    - name: Upload Artifact
      uses: actions/upload-artifact@v3
      with:
        name: greetings-jar
        path: build/libs

  test-run:
    runs-on: ubuntu-latest
    needs: build
```

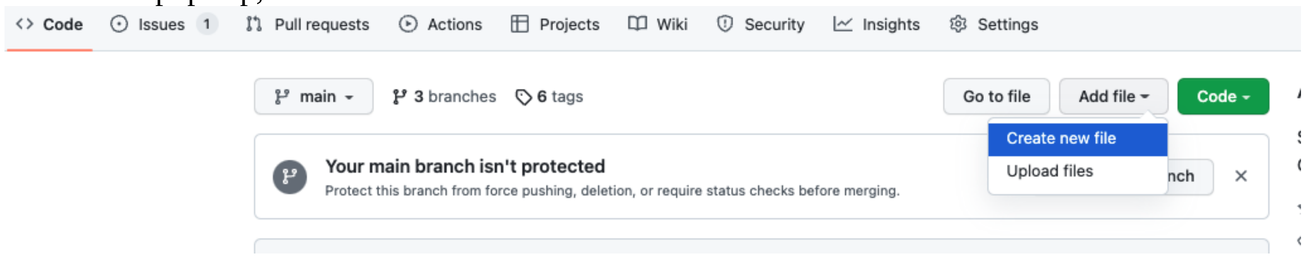
2. Now add a step to the job to download our artifact that was uploaded by the build job. Just as there was for the upload function, there is also an action to download the artifact. You can look up more about this if you want at *https://github.com/actions/download-artifact*. For the parameter, we'll just pass the name we called our artifact when we uploaded it.

```
    steps:
        - name: Download candidate artifacts
          uses: actions/download-artifact@v3
          with:
            name: greetings-jar
```



```
      - name: Upload Artifact
        uses: actions/upload-artifact@v3
        with:
          name: greetings-jar
          path: build/libs

  test-run:
    runs-on: ubuntu-latest
    needs: build

    steps:
    - name: Download candidate artifacts
      uses: actions/download-artifact@v3
      with:
        name: greetings-jar
```

3. Next, let's create a new script to test our code. To create a new file via the browser, go back to the "Code" tab at the top and then click on the "Add file" button next to the green "Code" button. From the list that pops up, select "Create new file".

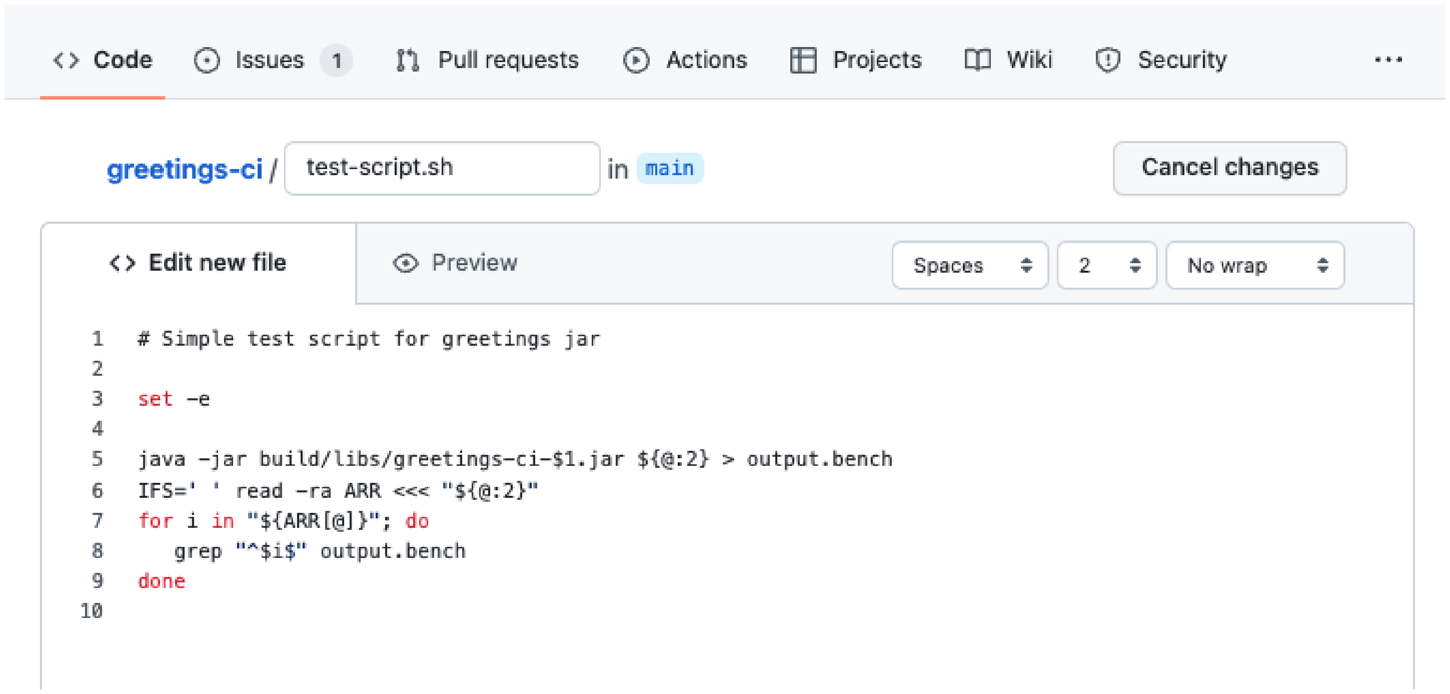


Your main branch isn't protected

Protect this branch from force pushing, deletion, or require status checks before merging.

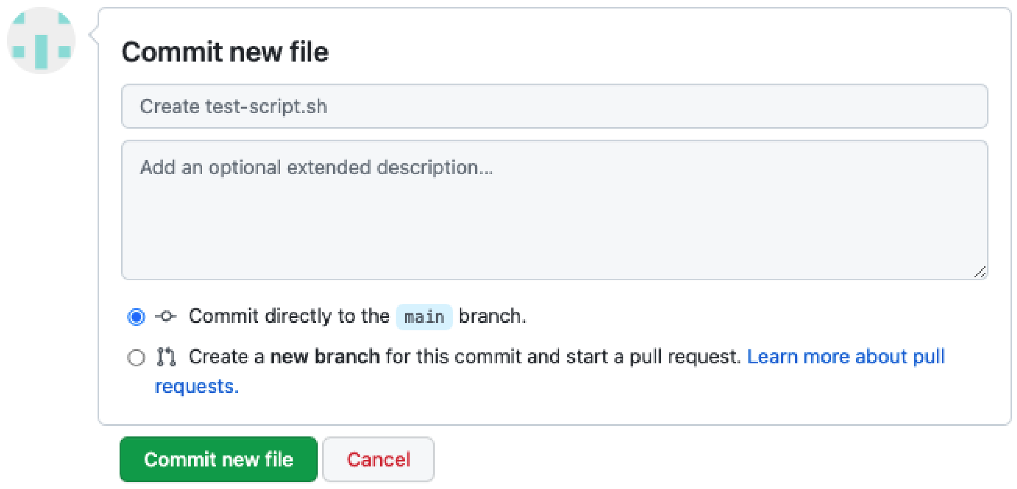
4. In the new editor that pops up, you'll be at the location to type in a name. You can name this "test-script.sh". Then copy and paste the following code into the new file.

```
# Simple test script for greetings jar

set -e

java -jar build/libs/greetings-ci-$1.jar ${@:2} > output.bench
IFS=' ' read -ra ARR <<< "${@:2}"
for i in "${ARR[@]}"; do
   grep "^$i$" output.bench
done
```



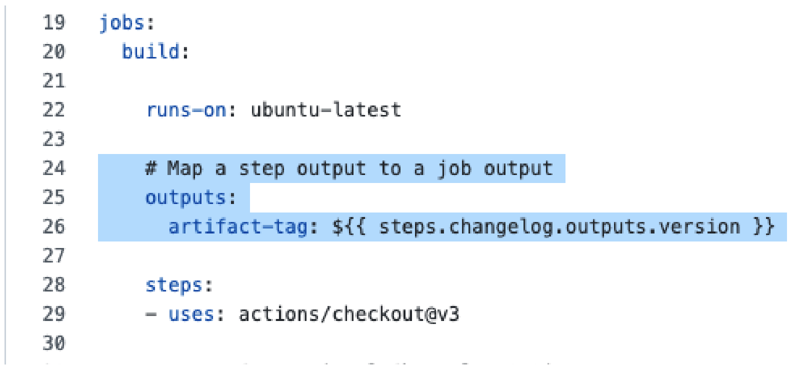
```
# Simple test script for greetings jar

set -e

java -jar build/libs/greetings-ci-$1.jar ${@:2} > output.bench
IFS=' ' read -ra ARR <<< "${@:2}"
for i in "${ARR[@]}"; do
   grep "^$i$" output.bench
done

```

5. This script takes the version of the jar to run as its first parameter and the remaining values passed in as the rest of the parameters. Then it simply cycles through all but the first parameter checking to see if they print out on a line by themselves.
   Go ahead and commit this file into the repository on the *main* branch.


Commit new file
Create test-script.sh
Add an optional extended description...
Commit directly to the main branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit new file
Cancel

6. There are a couple of "housekeeping" tasks we need to take care of before we call our script. First, since we want to be able to identify a specific version of the script, we need to capture the version from the "build" job into a "job output" that can be accessed from another job.
   Edit the pipeline.yaml file in main and add the lines below in the "build:" job definition after the "runs-on" and before the "steps". This will setup a new output from the job named artifact-tag.

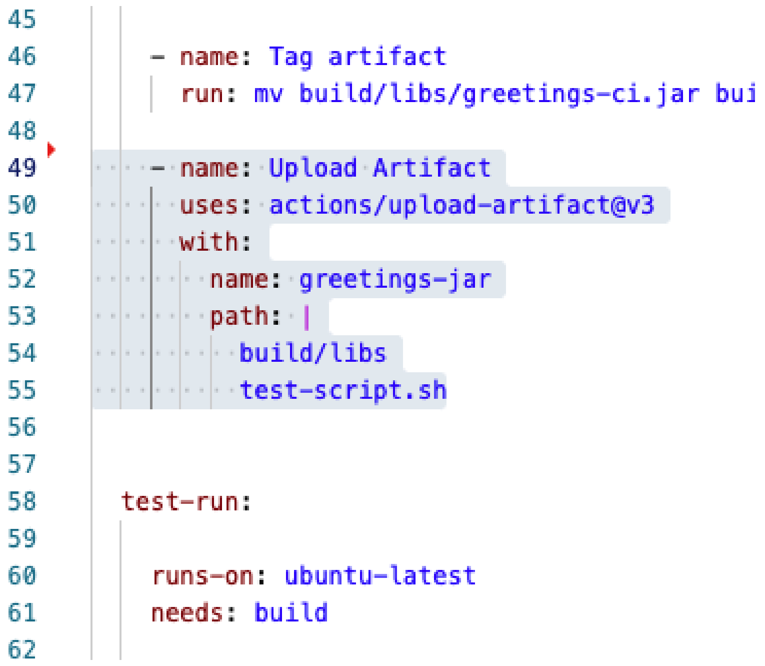
```
# Map a step output to a job output
outputs:
  artifact-tag: ${{ steps.changelog.outputs.version }}
```



```
jobs:
  build:

    runs-on: ubuntu-latest

    # Map a step output to a job output
    outputs:
      artifact-tag: ${{ steps.changelog.outputs.version }}

    steps:
    - uses: actions/checkout@v3

```

7. Next, since each job executes on a separate runner system, we need to make sure our new test script is available on the runner that will be executing the tests. For simplicity, we can just add it to the list of items that are included in the uploading of artifacts. Modify the path section of the "Upload Artifact" step in the "build" job to look like below.

```
path: |
  build/libs
  test-script.sh
```



```

      - name: Tag artifact
        run: mv build/libs/greetings-ci.jar bui

      - name: Upload Artifact
        uses: actions/upload-artifact@v3
        with:
          name: greetings-jar
          path: |
            build/libs
            test-script.sh


  test-run:

    runs-on: ubuntu-latest
    needs: build

```

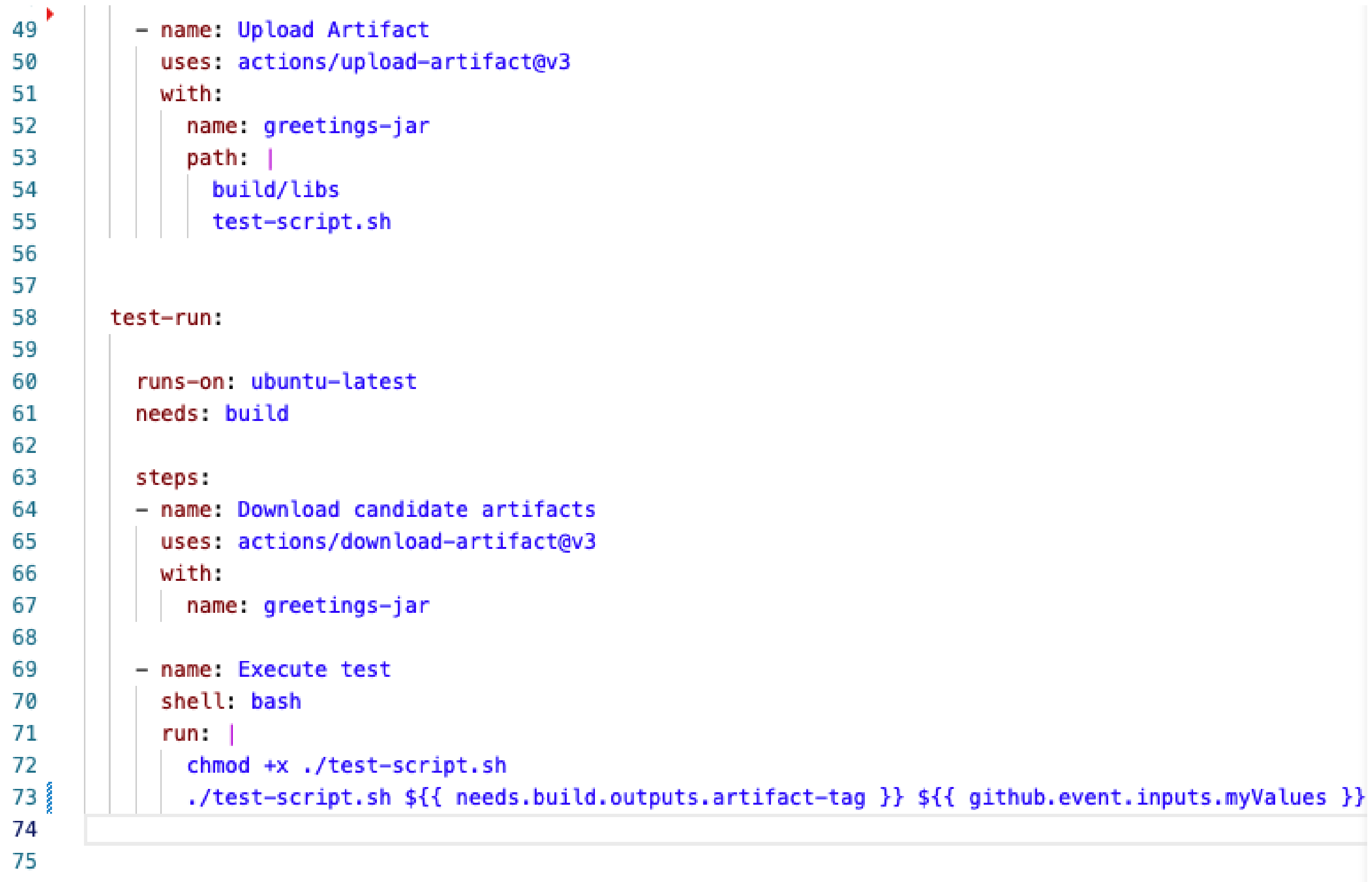
8. Finally, we can add a step to our test job to invoke the script. This step in the job will just be a shell command, so we'll use the "run" invocation.

   We first need to make the file executable on the runner, so we'll use the chmod command.

   For testing, our code echoes out the message that we input, so we'll just use the value that is input through our event. Notice that we pass the version value from the previous job as the first parameter so the correct versioned deliverable can be found and the remaining parameters passed in after that. That the "needs.build.outputs.artifact-tag" using the output we setup in step 6.

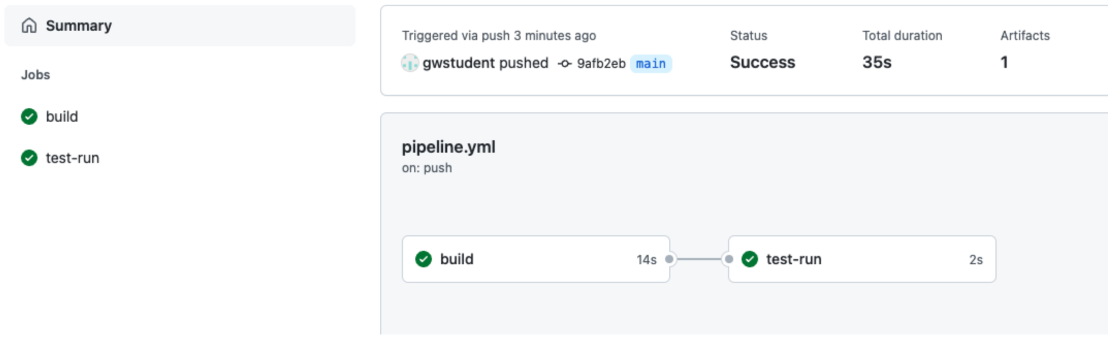
Edit the pipeline.yml file in the main branch and add the code below as the next step in the test-run job, paying attention to the alignment again.

```
    - name: Execute test
      shell: bash
      run: |
        chmod +x ./test-script.sh
       ./test-script.sh ${{ needs.build.outputs.artifact-tag }} ${{
github.event.inputs.myValues }}
```

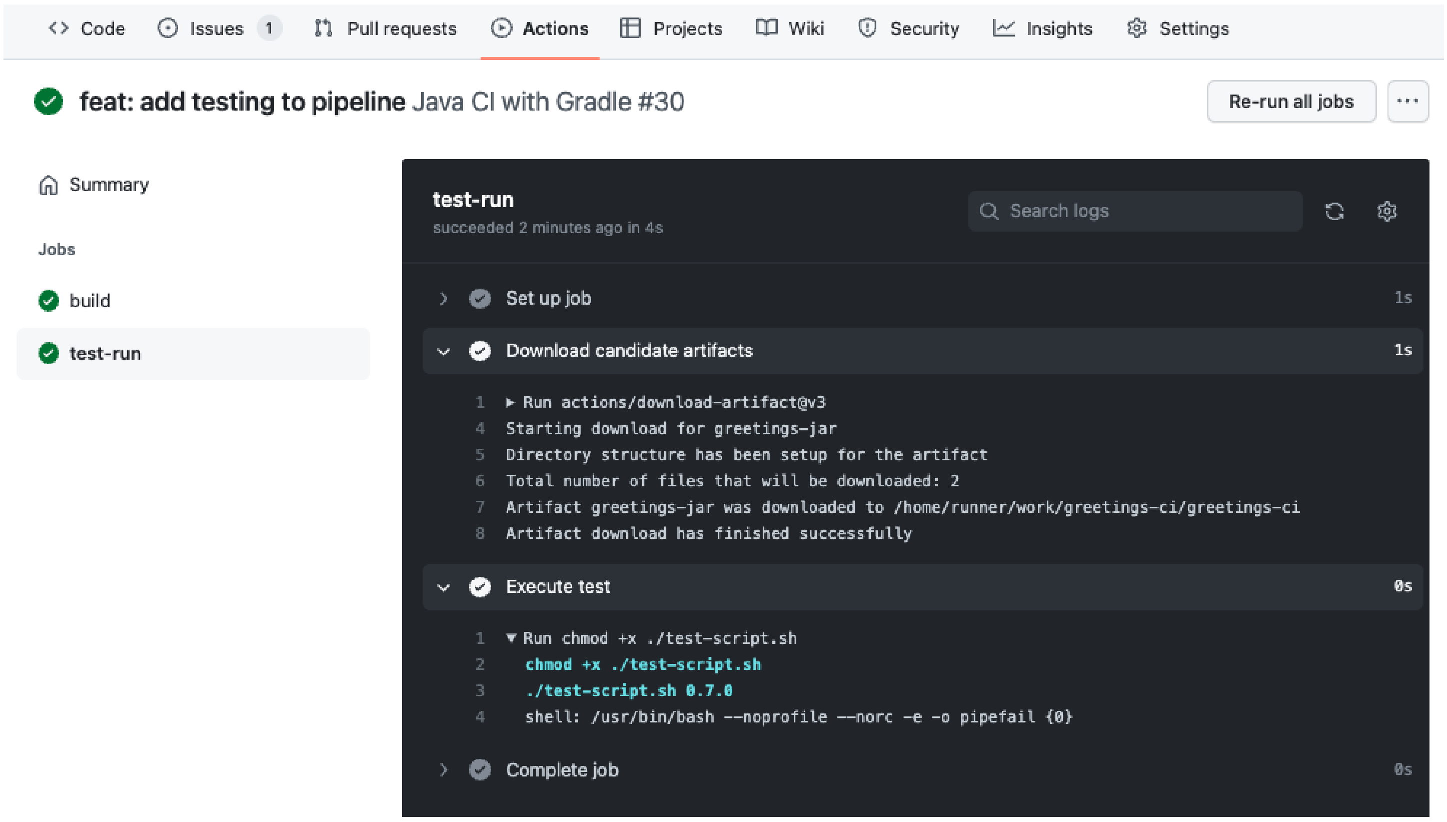


```
      - name: Upload Artifact
        uses: actions/upload-artifact@v3
        with:
          name: greetings-jar
          path: |
            build/libs
            test-script.sh


  test-run:

    runs-on: ubuntu-latest
    needs: build

    steps:
    - name: Download candidate artifacts
      uses: actions/download-artifact@v3
      with:
        name: greetings-jar

    - name: Execute test
      shell: bash
      run: |
        chmod +x ./test-script.sh
        ./test-script.sh ${{ needs.build.outputs.artifact-tag }} ${{ github.event.inputs.myValues }}

```

9. Now, you can just commit the pipeline changes with a simple message like "feat: add testing to pipeline". Afterwards, you should see a new run of the action showing multiple jobs in the action run detail. Notice that we can select and drill into each job separately.



10. You can look at the logs from the test-run job if you want to see the downloaded script and execution. Note which version got passed in the "./test-script.sh" line. You'll need this later.

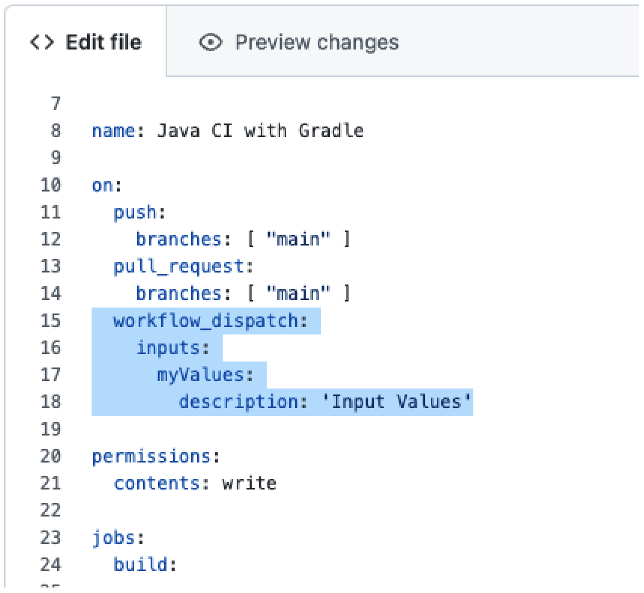


```
▶ Run actions/download-artifact@v3
Starting download for greetings-jar
Directory structure has been setup for the artifact
Total number of files that will be downloaded: 2
Artifact greetings-jar was downloaded to /home/runner/work/greetings-ci/greetings-ci
Artifact download has finished successfully
```

```
▼ Run chmod +x ./test-script.sh
  chmod +x ./test-script.sh
  ./test-script.sh 0.7.0
  shell: /usr/bin/bash --noprofile --norc -e -o pipefail {0}
```

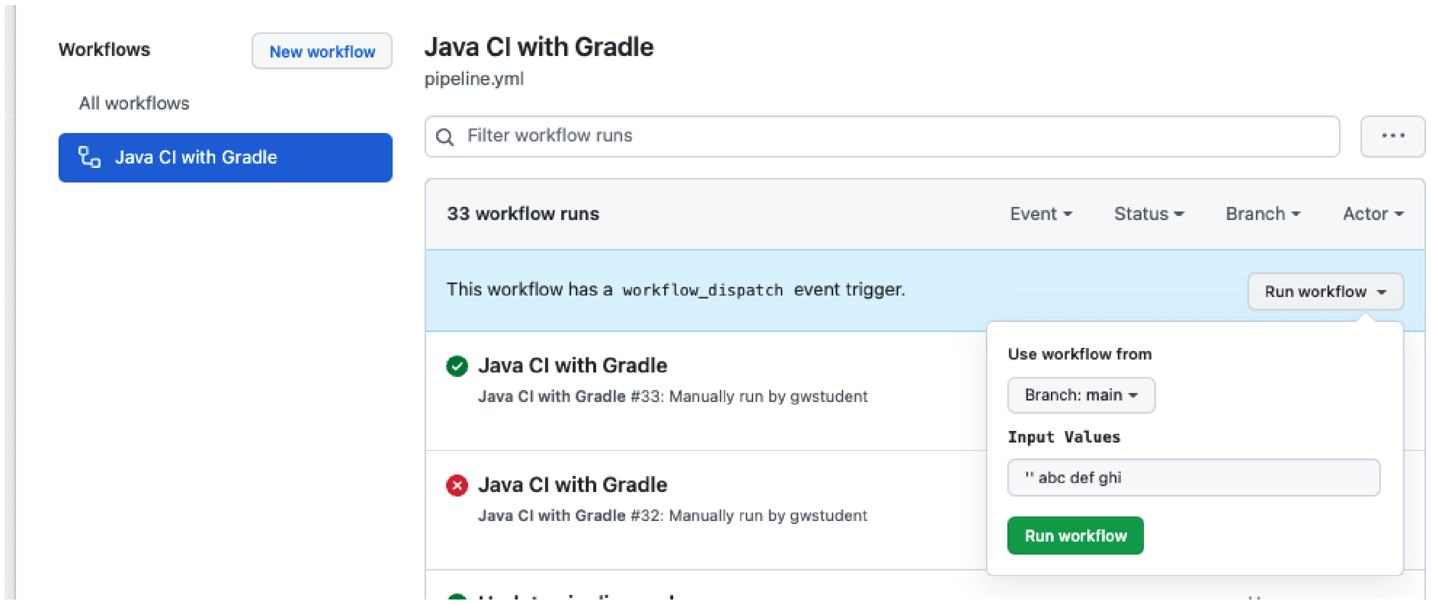
11. Let's make one more change to make it easier to run our workflow manually to try things out, start runs, etc. Edit the pipeline.yaml file again and add the code below at the bottom of the "on" section. ("workflow_dispatch" should line up with "pull" and "push") and then commit the changes.

```
workflow_dispatch:
  inputs:
    myValues:
      description: 'Input Values'
```



```
name: Java CI with Gradle

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]
  workflow_dispatch:
    inputs:
      myValues:
        description: 'Input Values'

permissions:
  contents: write

jobs:
  build:
```

12. Since we added the workflow_dispatch event trigger in our last set of code changes, if you select the workflow, you should see a blue bar that has a button that allows you to manually run the workflow. Click on "Run workflow". Make sure "main" is selected. Since we can't force a version through from the manual start, just enter an empty string '' (**that's two single quotes together - not a double quote**) for the first argument then enter whatever you want for arguments. Click on the green "Run workflow" button. You can look at the output of the test-run job again to see what occurred.



Java CI with Gradle
pipeline.yml

33 workflow runs

This workflow has a `workflow_dispatch` event trigger.

Java CI with Gradle
Java CI with Gradle #33: Manually run by gwstudent

Java CI with Gradle
Java CI with Gradle #32: Manually run by gwstudent

Input Values
'' abc def ghi

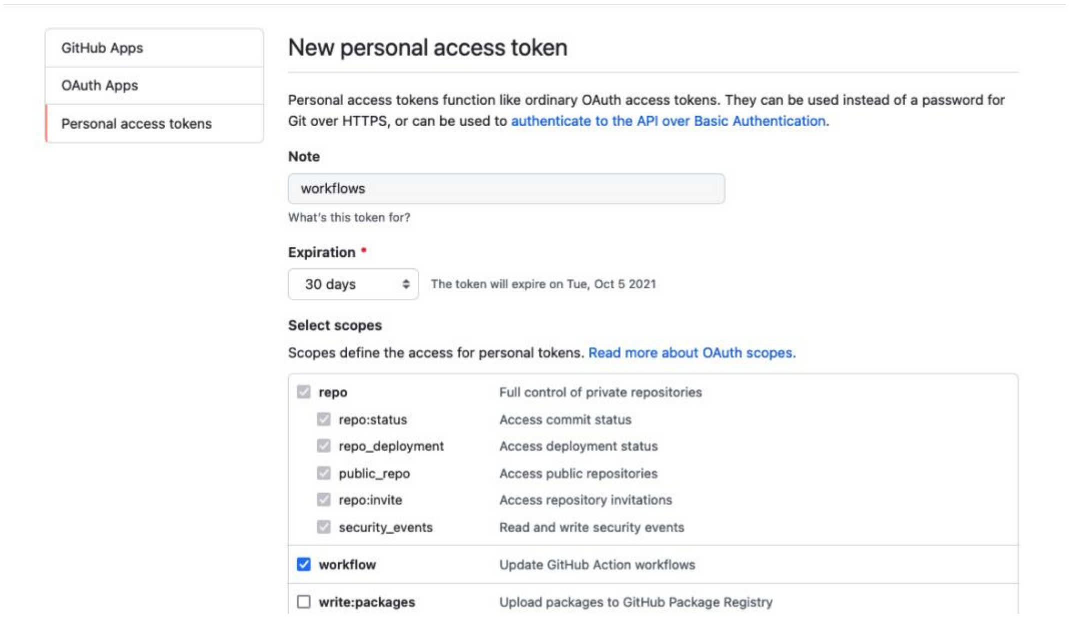
Lab 10: Working with fast feedback and automatically reporting issues

Purpose: Learning how to get fast feedback and automatic failure reporting in our pipeline

1. For this lab, we need to prepare a Personal Access Token (PAT) and add it to a secret that our workflow can reference. If you already have a PAT, you may be able to use it if it has access to the project. If not, you'll need to create a new one. Go to https://github.com/settings/tokens.

   (Alternatively, on the GitHub repo screen, click on your profile picture in the upper right, then select "Settings" from the drop-down menu. You should be on the https://github.com/settings/profile screen. On this page on the left-hand side, select "Developer settings" near the bottom. On the next page, select "Personal access tokens".)

2. Click on "Generate new token". Confirm your password if asked. In the "Note" section enter some text, such as "workflows". You can set the "Expiration" time as desired or leave it as-is. Under "Select scopes", assuming your repository is public, you can just check the boxes for "repo" and "workflow". Then click on the green "Generate token" at the bottom.



New personal access token

Personal access tokens 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.

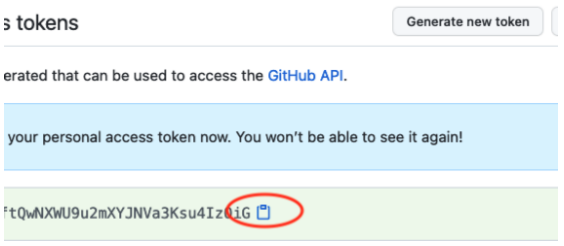
Note: workflows
What's this token for?

Expiration *: 30 days — The token will expire on Tue, Oct 5 2021

Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events
✓ workflow	Update GitHub Action workflows
☐ write:packages	Upload packages to GitHub Package Registry

3. After the screen comes up that shows your new token, make sure to copy it and store it somewhere you can get to it.

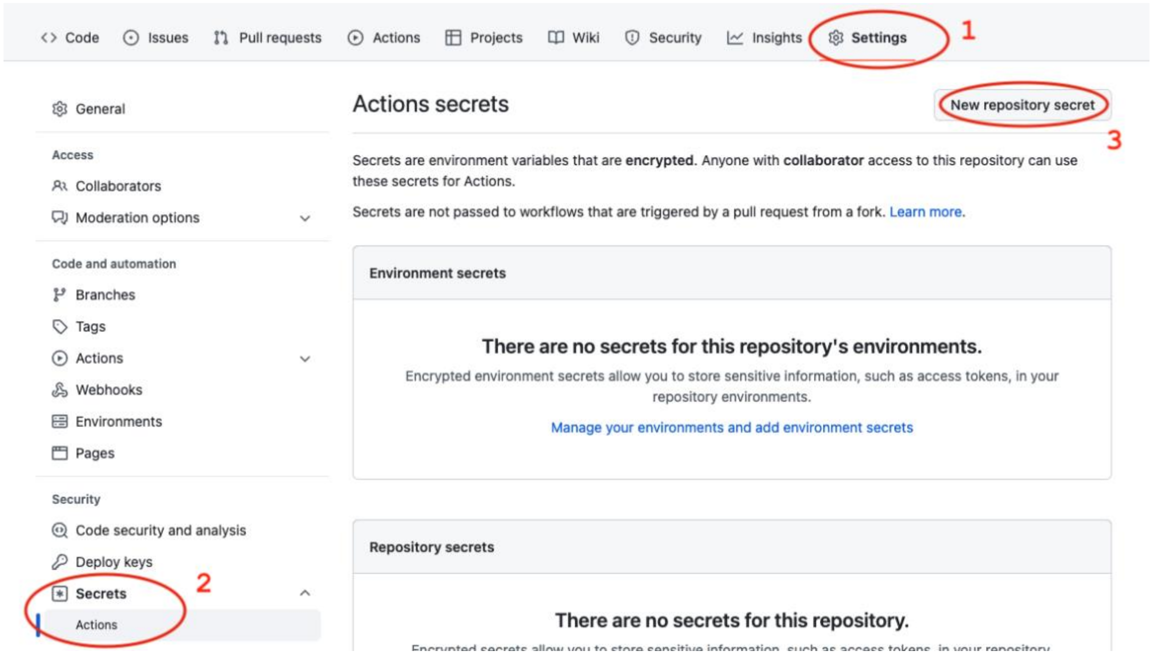


erated that can be used to access the GitHub API.

your personal access token now. You won't be able to see it again!

```
ftQwNXWU9u2mXYJNVa3Ksu4IzdiG
```

4. Now we'll create a new secret and store the PAT value in it. Go to the repository and in the top menu select "Settings". Then on the left-hand side, select "Secrets" and select "Actions". Now, click on the "New repository secret in the upper right to create a new secret for the action to use.



Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

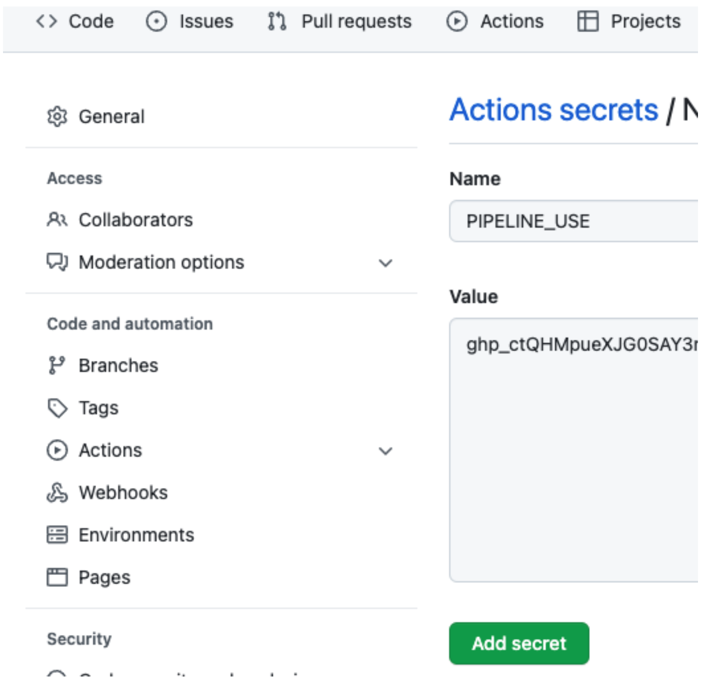
Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

There are no secrets for this repository's environments.

Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository environments.

There are no secrets for this repository.

5. For the Name of the new secret, use PIPELINE_USE. Paste the value from the PAT into the Value section. Then click on the "Add secret" button at the bottom. After this, the new secret should show up at the bottom.

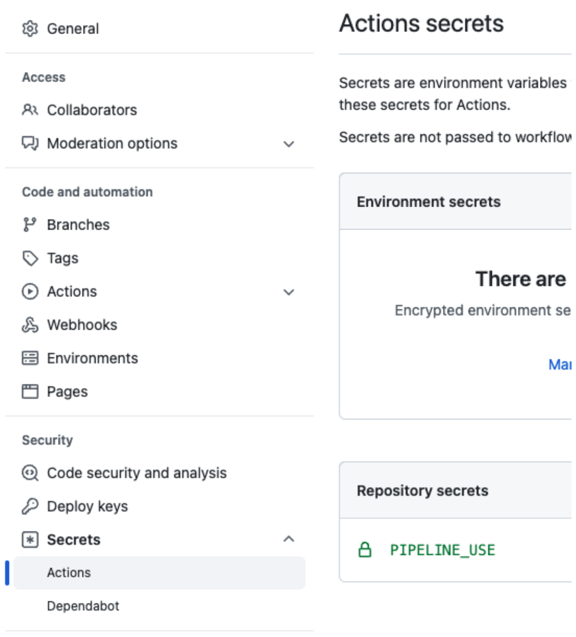


Name

PIPELINE_USE

Value

ghp_ctQHMpueXJG0SAY3r



Secrets are environment variables

these secrets for Actions.

Secrets are not passed to workflow

There are

Encrypted environment se

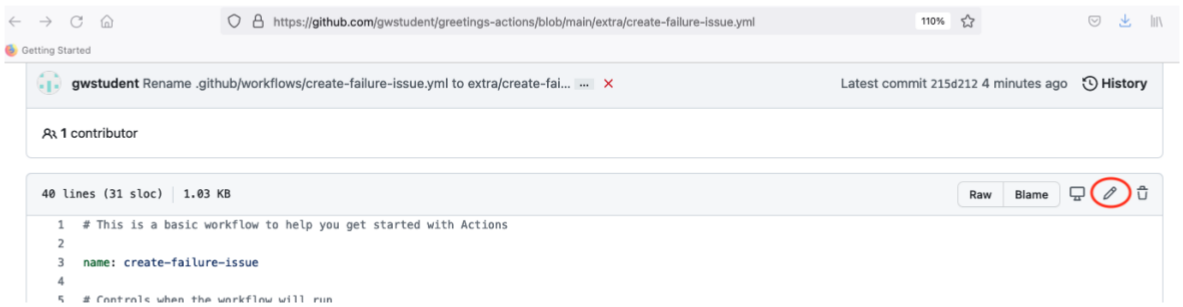
Mar

PIPELINE_USE

6. We're going to create a new workflow that will be able to automatically create a GitHub issue in our repository. And then we will invoke that workflow from our current workflow. The workflow to create

the issue using a REST API call is already written to save time. It is in the main project under "extra/create-failure-issue.yml". You need to get this file in the .github/workflows directory. To do that, you can clone and move it. Or you can just do it via GitHub with the following steps.

a. In the repository, browse to the "extra" folder and to the "create-failure-issue.yml" file.
b. Take a few moments to look over the file and see what it does. Notice that:
    i. it has a workflow_dispatch section in the "on" area, which means it can be run manually.
    ii. It has two inputs - a title and body for the issue.
    iii. The primary part of the body is simply a REST call (using the GITHUB_TOKEN) to create a new issue.
c. Click the pencil icon to edit it.


https://github.com/gwstudent/greetings-actions/blob/main/extra/create-failure-issue.yml

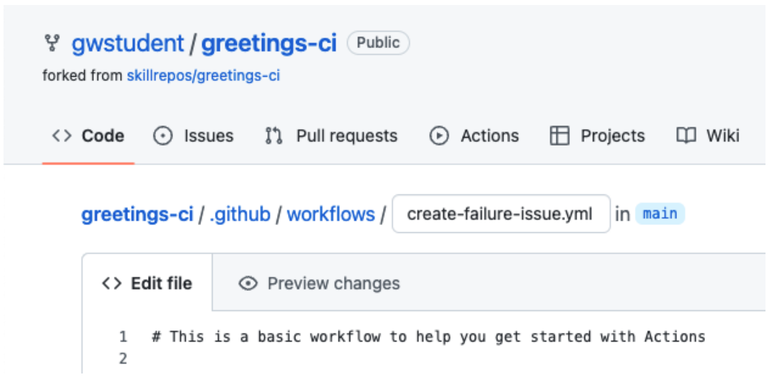
gwstudent Rename .github/workflows/create-failure-issue.yml to extra/create-fai... Latest commit 215d212 4 minutes ago History

1 contributor

40 lines (31 sloc) | 1.03 KB

```
# This is a basic workflow to help you get started with Actions

name: create-failure-issue

# Controls when the workflow will run
```

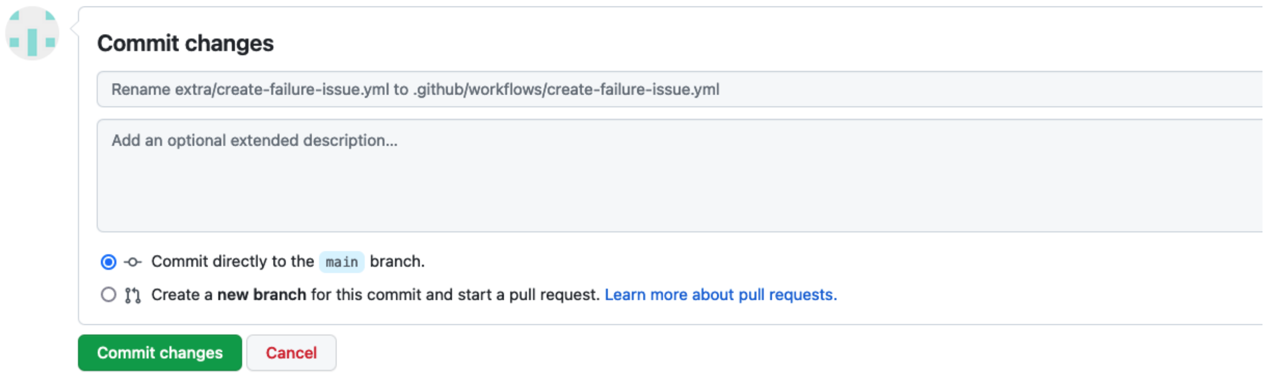
d. In the filename field, change the name of the file. Use the backspace key to backspace over "extra/" making sure to backspace over the word. Then type in the path to put it in the workflows ".github/workflows/create-failure-issue.yml".


gwstudent / greetings-ci Public
forked from skillrepos/greetings-ci

greetings-ci / .github / workflows / create-failure-issue.yml in main

```
# This is a basic workflow to help you get started with Actions

```

e. To complete the change, scroll to the bottom of the page, and click on the green "Commit changes" button.


Commit changes

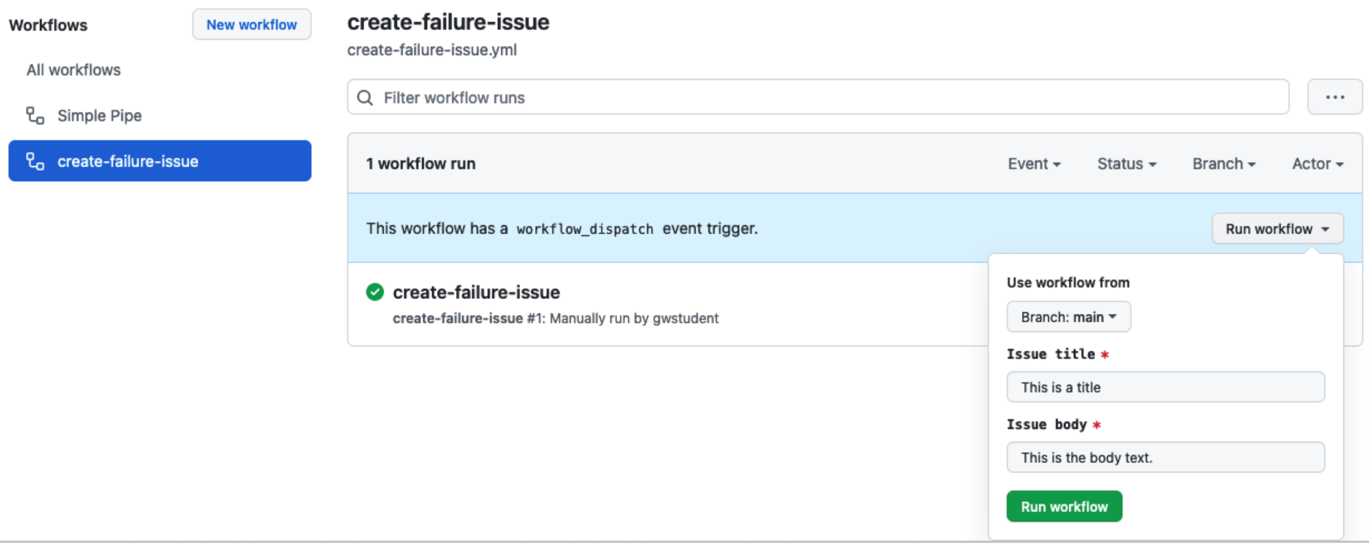
Rename extra/create-failure-issue.yml to .github/workflows/create-failure-issue.yml

Add an optional extended description...

Commit directly to the `main` branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

7. Go back to the Actions tab. You'll see a new workflow execution due to the rename. Also, in the Workflows section on the left, you should now see a new workflow titled "create-failure-issue". Click on that. Since it has a workflow_dispatch event trigger available, we can try it out. Click on the "Run workflow" button and enter in some text for the "title" and "body" fields. Then click "Run workflow".


create-failure-issue
create-failure-issue.yml

1 workflow run

This workflow has a `workflow_dispatch` event trigger.

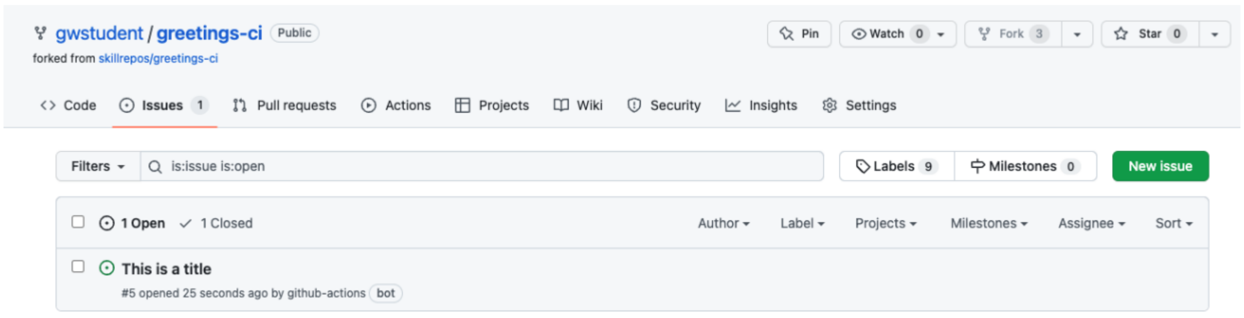
create-failure-issue
create-failure-issue #1: Manually run by gwstudent

Use workflow from
Branch: main

Issue title *
This is a title

Issue body *
This is the body text.

8. After a moment, you should see the workflow run start and then complete. If you now click on the Issues tab at the top, you should see your new issue there.


gwstudent / greetings-ci Public
forked from skillrepos/greetings-ci

is:issue is:open

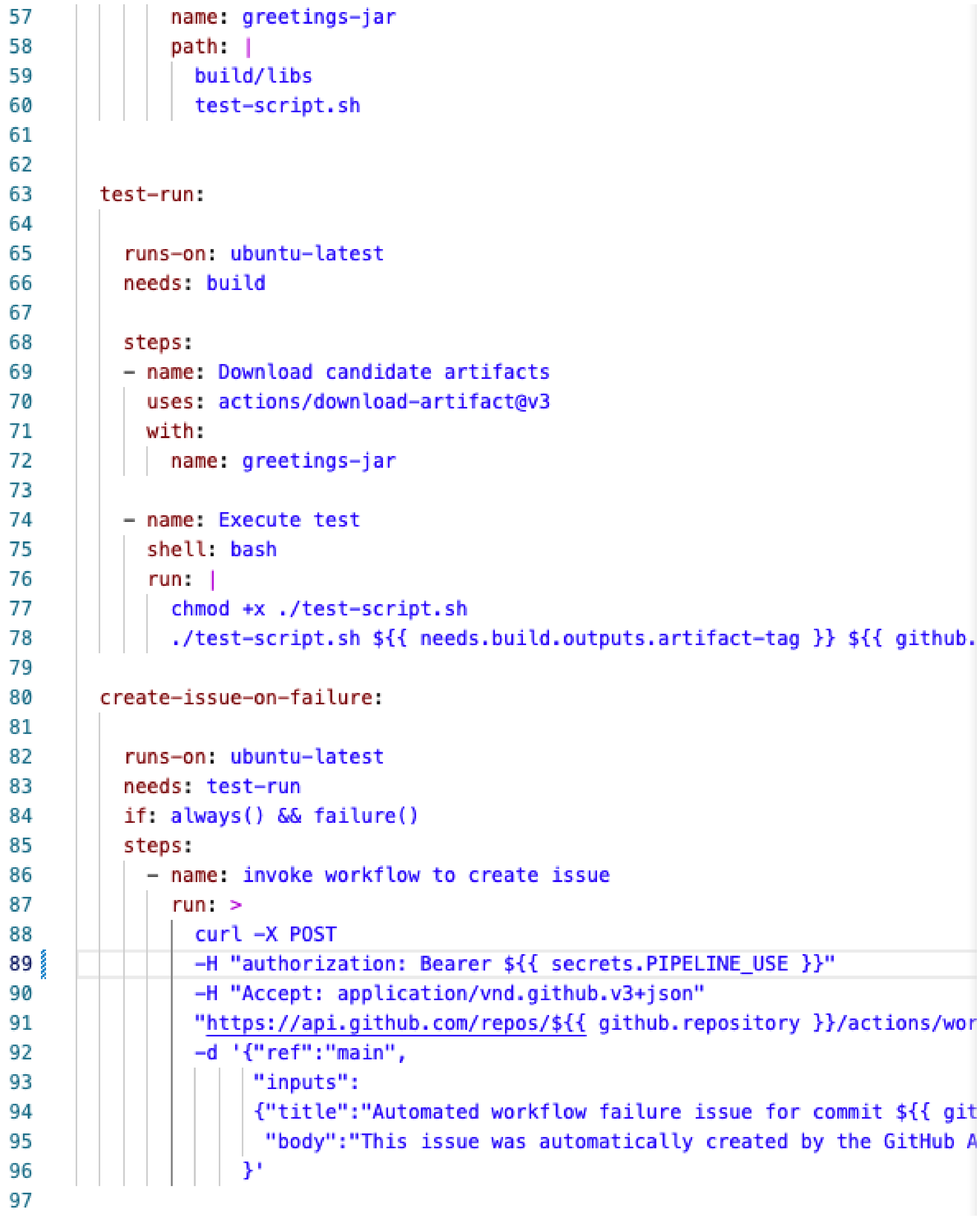
1 Open ✓ 1 Closed

This is a title
#5 opened 25 seconds ago by github-actions bot

9. Now that we know that our new workflow works as expected, we can make the changes to the previous workflow to "call" this if we fail. Edit the pipeline.yml file and add the following lines as a new

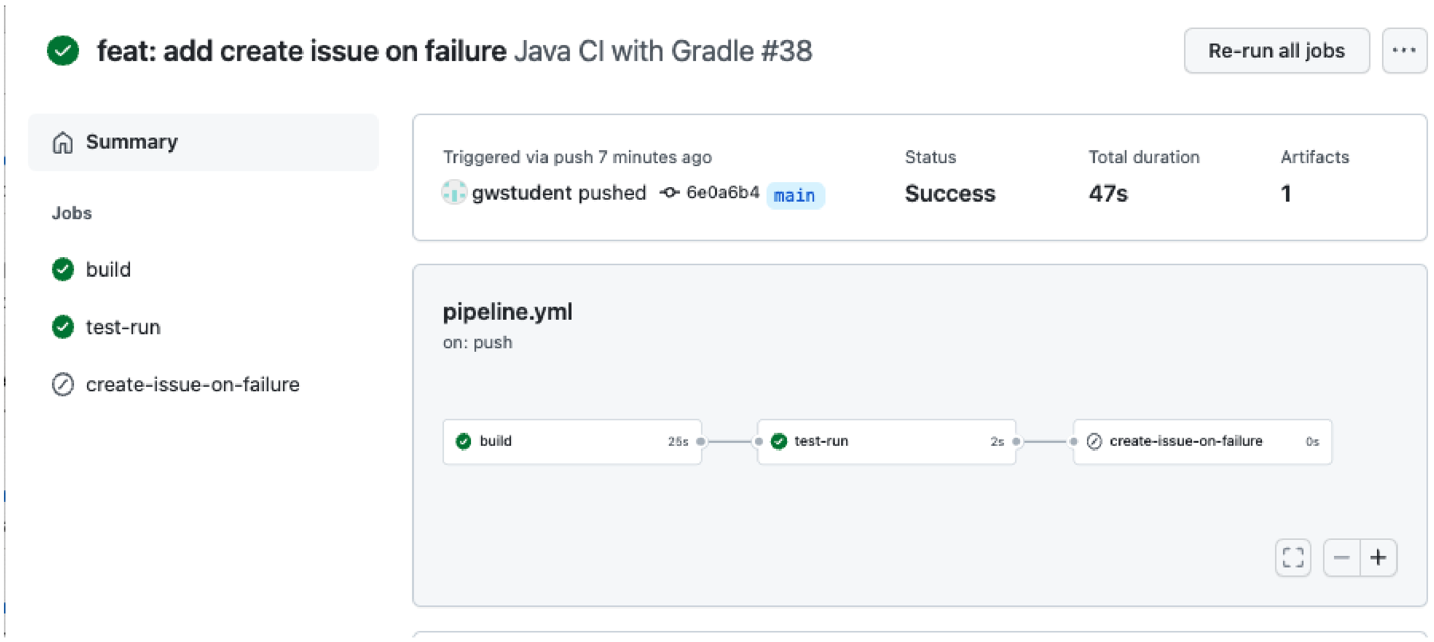
job and set of steps at the end of the workflow. (For convenience, these lines are also in the file "extra/create-issue-on-failure.txt" if you want to copy and paste from there.) The "create-issue-on-failure" job name should align with the "test-run" job name. See screenshot further down.

```
  create-issue-on-failure:

    runs-on: ubuntu-latest
    needs: test-run
    if: always() && failure()
    steps:
      - name: invoke workflow to create issue
        run: >
          curl -X POST
          -H "authorization: Bearer ${{ secrets.PIPELINE_USE }}"
          -H "Accept: application/vnd.github.v3+json"
          "https://api.github.com/repos/${{ github.repository }}/actions/workflows/create-
failure-issue.yml/dispatches"
          -d '{"ref":"main",
               "inputs":
               {"title":"Automated workflow failure issue for commit ${{ github.sha }}",
                "body":"This issue was automatically created by the GitHub Action workflow
** ${{ github.workflow }} **"}
              }'
```



```
        name: greetings-jar
        path: |
          build/libs
          test-script.sh


  test-run:

    runs-on: ubuntu-latest
    needs: build

    steps:
    - name: Download candidate artifacts
      uses: actions/download-artifact@v3
      with:
        name: greetings-jar

    - name: Execute test
      shell: bash
      run: |
        chmod +x ./test-script.sh
        ./test-script.sh ${{ needs.build.outputs.artifact-tag }} ${{ github.

  create-issue-on-failure:

    runs-on: ubuntu-latest
    needs: test-run
    if: always() && failure()
    steps:
      - name: invoke workflow to create issue
        run: >
          curl -X POST
          -H "authorization: Bearer ${{ secrets.PIPELINE_USE }}"
          -H "Accept: application/vnd.github.v3+json"
          "https://api.github.com/repos/${{ github.repository }}/actions/wor
          -d '{"ref":"main",
               "inputs":
               {"title":"Automated workflow failure issue for commit ${{ git
                "body":"This issue was automatically created by the GitHub A
              }'

```

10. After this is committed and the workflow runs, you can look at the output for the run and you'll see that the "create-issue-on-failure" job was skipped. That makes sense because we have the checks in the code and there was no failure on previous jobs.


feat: add create issue on failure Java CI with Gradle #38

Re-run all jobs

Summary

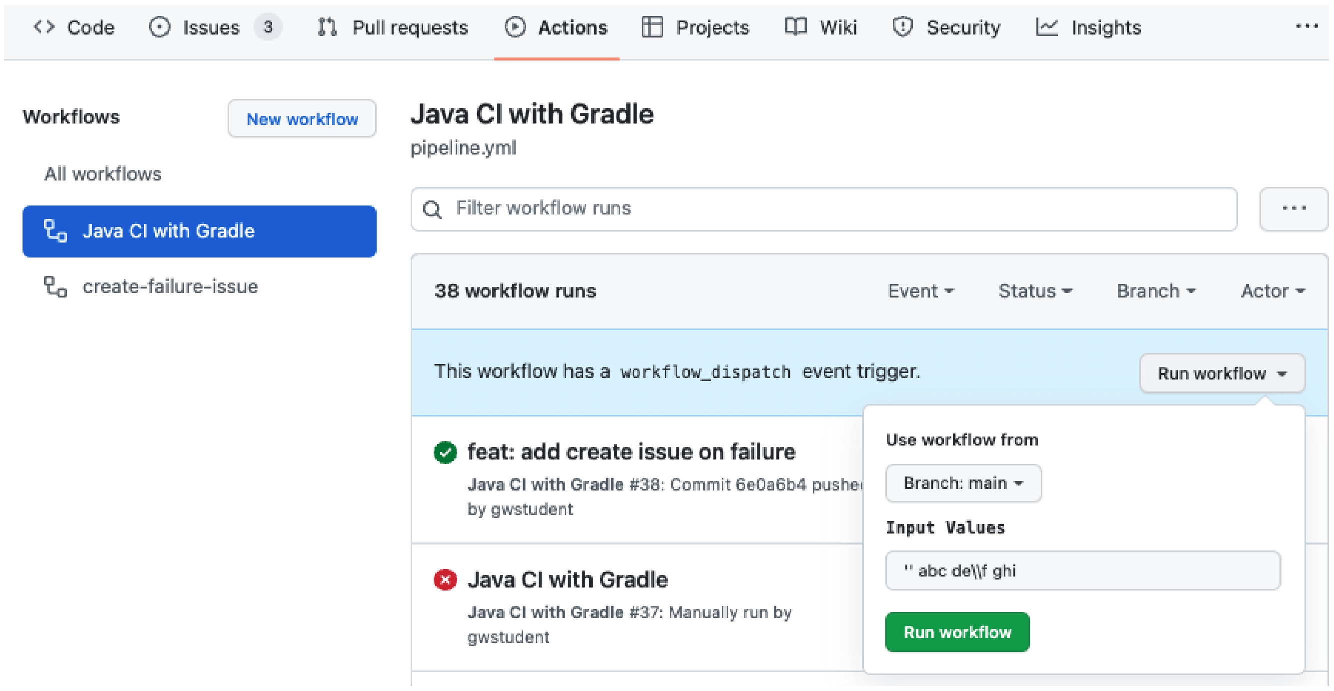
Jobs
- build
- test-run
- create-issue-on-failure

Triggered via push 7 minutes ago	Status	Total duration	Artifacts
gwstudent pushed 6e0a6b4 main	Success	47s	1

pipeline.yml
on: push

11. To have this executed via the "if" statement, we need to have a failure. Let's try some different input with special characters that may not print out as expected. Go to the Actions menu, and then select our main "Java CI with Gradle" workflow. Click on the "Run workflow" button and enter text like below: (that's two single quotes together on the front and two backslashes between the "de" and "f"). As long as you have the empty string at the start and two backslashes somewhere, this should fail.

    '' abc. de\\f ghi



Java CI with Gradle
pipeline.yml

Filter workflow runs

38 workflow runs | Event | Status | Branch | Actor

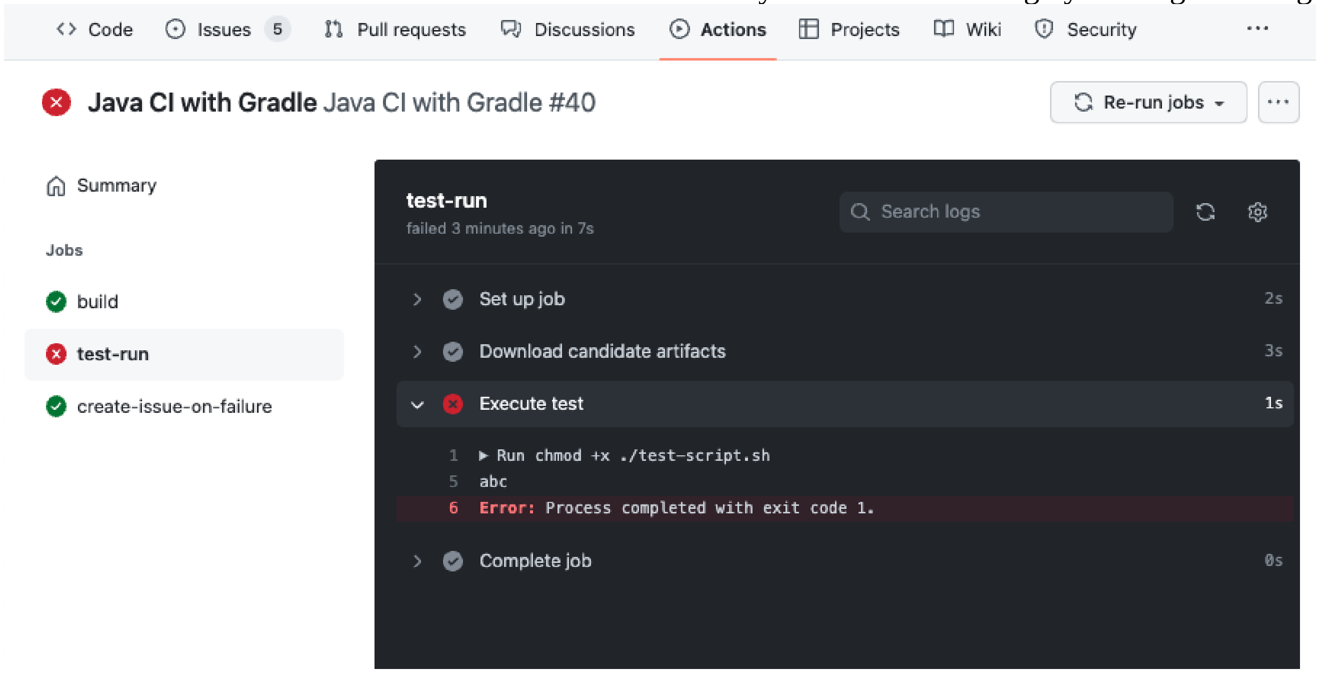
This workflow has a `workflow_dispatch` event trigger. Run workflow

feat: add create issue on failure
Java CI with Gradle #38: Commit 6e0a6b4 pushed by gwstudent

Java CI with Gradle
Java CI with Gradle #37: Manually run by gwstudent

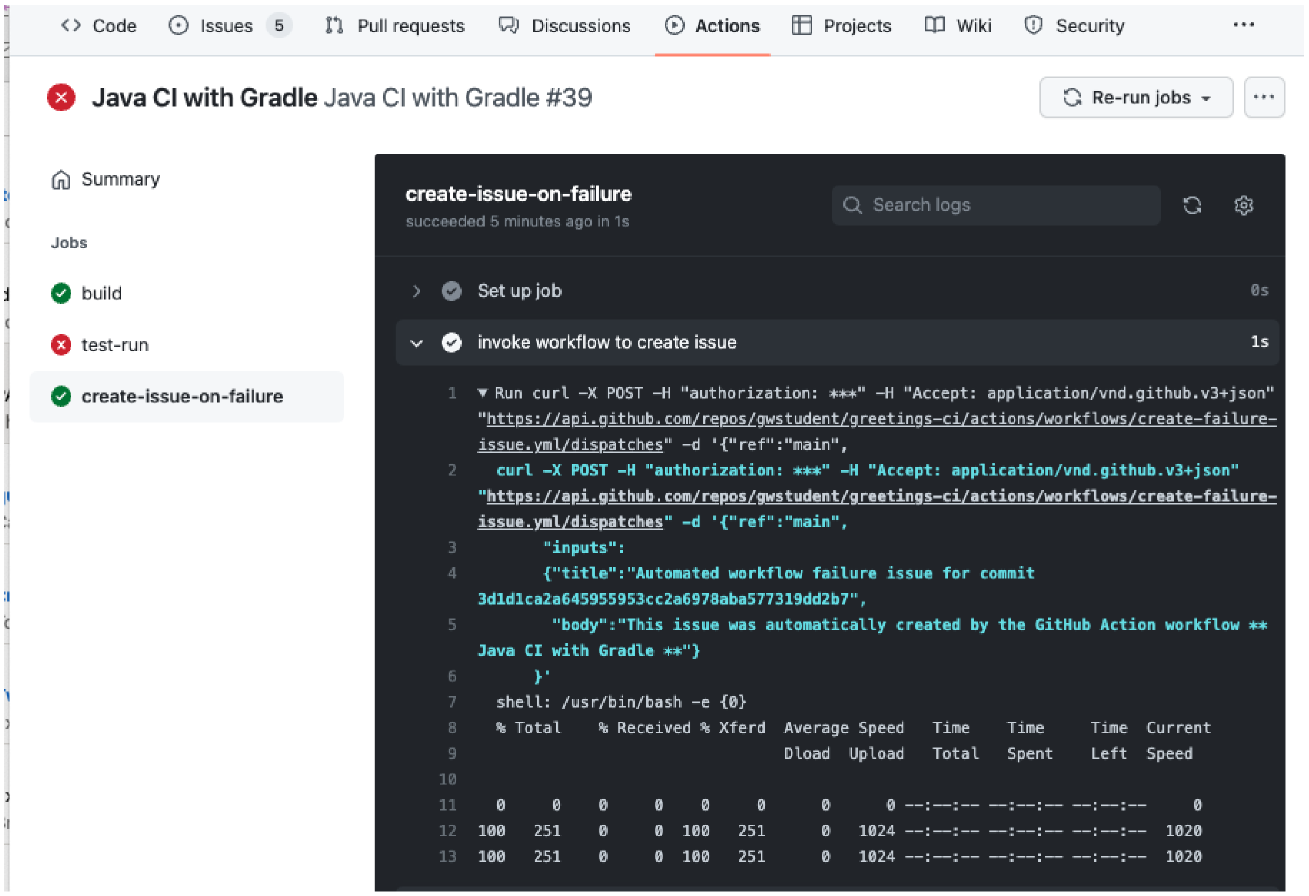
Use workflow from
Branch: main
Input Values
'' abc de\\f ghi
Run workflow

12. After the workflow run completes for this, there should be a failure in our testing. This will in turn, cause our other workflow to create an issue. You can verify the failure in testing by looking at the logs.

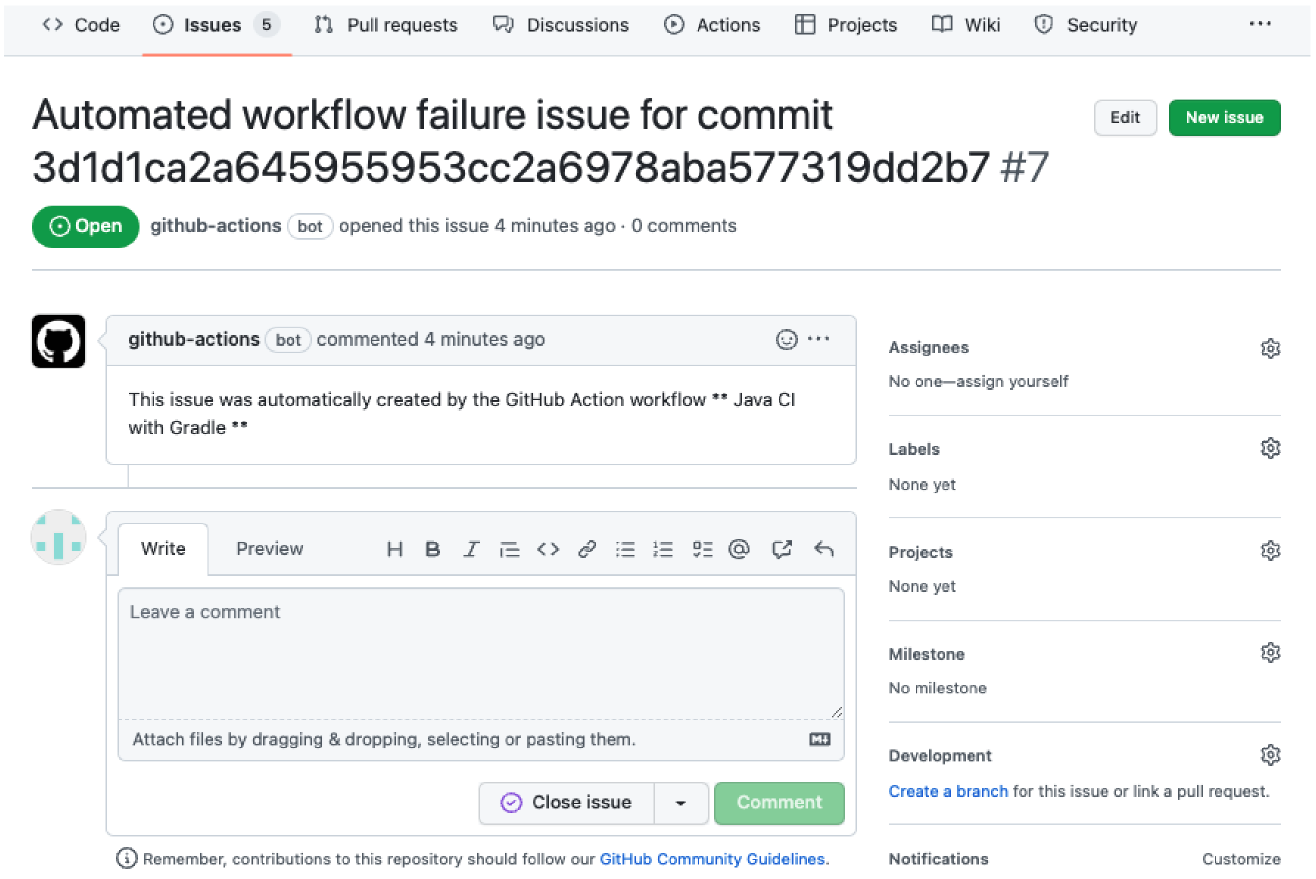


```
▶ Run chmod +x ./test-script.sh
abc
Error: Process completed with exit code 1.
```

You can also verify the new issue got created as a result of the failure through the logs of that job and by looking in the Issues menu at the top.



```
▼ Run curl -X POST -H "authorization: ***" -H "Accept: application/vnd.github.v3+json"
"https://api.github.com/repos/gwstudent/greetings-ci/actions/workflows/create-failure-
issue.yml/dispatches" -d '{"ref":"main",
  curl -X POST -H "authorization: ***" -H "Accept: application/vnd.github.v3+json"
"https://api.github.com/repos/gwstudent/greetings-ci/actions/workflows/create-failure-
issue.yml/dispatches" -d '{"ref":"main",
       "inputs":
       {"title":"Automated workflow failure issue for commit
3d1d1ca2a645955953cc2a6978aba577319dd2b7",
        "body":"This issue was automatically created by the GitHub Action workflow **
Java CI with Gradle **"}
      }'
  shell: /usr/bin/bash -e {0}
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100   251    0     0  100   251      0   1024 --:--:-- --:--:-- --:--:--  1020
100   251    0     0  100   251      0   1024 --:--:-- --:--:-- --:--:--  1020
```



Automated workflow failure issue for commit 3d1d1ca2a645955953cc2a6978aba577319dd2b7 #7

Open github-actions bot opened this issue 4 minutes ago · 0 comments

github-actions bot commented 4 minutes ago

This issue was automatically created by the GitHub Action workflow ** Java CI with Gradle **

Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Close issue
Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.