Continuous Integration (CI) with MATLAB and GitHub® Actions Workshop

CI provides developers with a consistent, automated way to regularly test and merge their code changes into a shared repository. CI makes it easier to detect and resolve conflicts early, resulting in a more consistently stable codebase and faster development time. CI is a DevOps best practice and a key part of Agile development workflows.

This workshop provides a step-by-step guide to getting started with CI using MATLAB and GitHub Actions.

In this workshop, you will:

- fork and modify your own copy of the MathWorks CI configuration examples repository on GitHub
- use GitHub Actions directly from within GitHub
- create and use GitHub personal access tokens
- automatically run a GitHub Action by pushing code changes to GitHub from MATLAB
- identify test failures directly within GitHub Actions

The workshop also includes a separate homework document with additional tasks to grow your skills further.

The homework tasks include:

- adding new files to your repository
- finding and fixing a bug in one of the files
- getting a passing GitHub Actions result

Table of Contents

Part 1: Requirements for the workshop

The following steps cover all of the things you will need to successfully complete the workshop.

1A) Create your MathWorks account

- Go to: **https://www.mathworks.com/mwaccount/register**

1B) Create your GitHub account

- Go to: **https://github.com/signup**

1C) Access to MATLAB

- You can use MATLAB Online (**https://matlab.mathworks.com**) or a desktop installation of MATLAB for the workshop
- ***Note:*** The majority of the workshop and screenshots depict the MATLAB Online interface

1D) Google Chrome web browser *(recommended if using MATLAB Online)*

- Google Chrome is the recommended web browser when using MATLAB Online
- Go to: **https://www.google.com/chrome/downloads**

Part 2: Fork the "CI Configuration Examples" repository to your GitHub Account

To help you get started using CI with MATLAB, MathWorks provides a repository of CI configuration examples for various CI platforms (e.g., Azure® DevOps, CircleCI®, Github Actions, Travis CI™...). The various CI platforms will automatically pick up the configuration files and add the relevant options or connections for you.

First, we'll start by forking the "CI Configuration Examples" repository to your GitHub account.

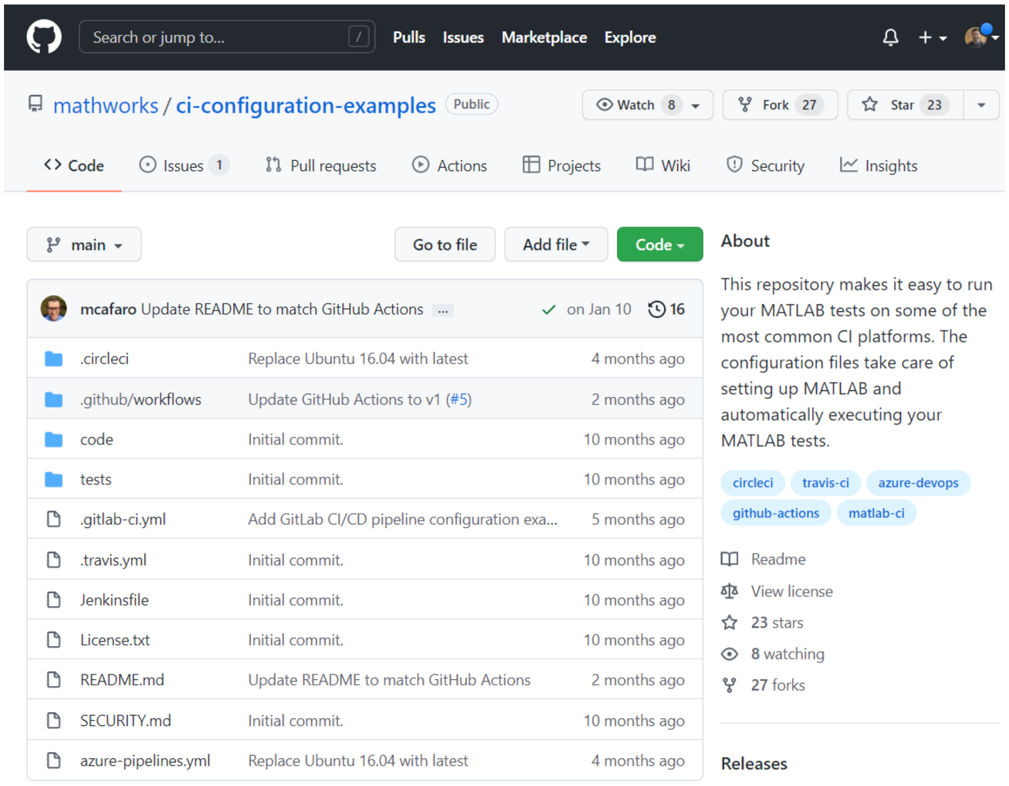
Wait... What does it mean to "fork a repository" and why should I do it?

Forking a repository allows you to freely experiment with changes to a project without affecting the original project.

GitHub provides some great information about "why" and "how" to fork a repository: https://docs.github.com/en/github-ae@latest/get-started/quickstart/fork-a-repo

Let's get started!

2A) Go to: https://github.com/mathworks/ci-configuration-examples



mcafaro Update README to match GitHub Actions ... ✓ on Jan 10 16

.circleci	Replace Ubuntu 16.04 with latest	4 months ago
.github/workflows	Update GitHub Actions to v1 (#5)	2 months ago
code	Initial commit.	10 months ago
tests	Initial commit.	10 months ago
.gitlab-ci.yml	Add GitLab CI/CD pipeline configuration exa...	5 months ago
.travis.yml	Initial commit.	10 months ago
Jenkinsfile	Initial commit.	10 months ago
License.txt	Initial commit.	10 months ago
README.md	Update README to match GitHub Actions	2 months ago
SECURITY.md	Initial commit.	10 months ago
azure-pipelines.yml	Replace Ubuntu 16.04 with latest	4 months ago

About

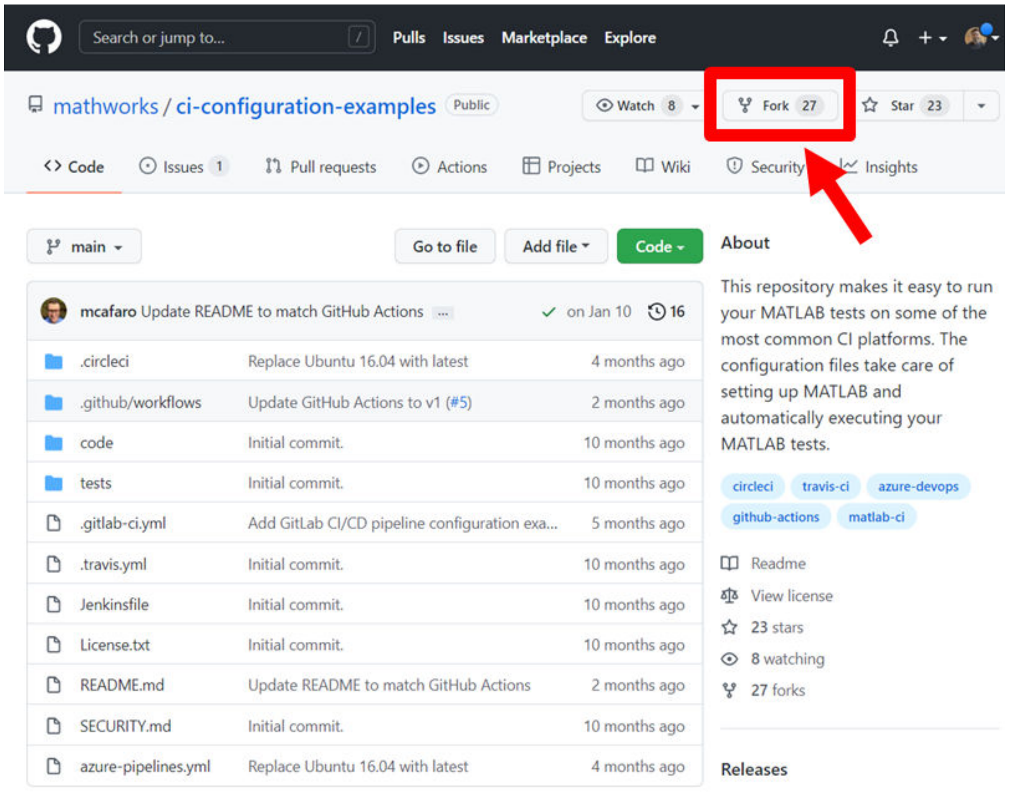
This repository makes it easy to run your MATLAB tests on some of the most common CI platforms. The configuration files take care of setting up MATLAB and automatically executing your MATLAB tests.

circleci travis-ci azure-devops github-actions matlab-ci

Readme
View license
23 stars
8 watching
27 forks

Releases

2B) Press the "Fork" button (top right)



mcafaro Update README to match GitHub Actions ... ✓ on Jan 10 16

.circleci	Replace Ubuntu 16.04 with latest	4 months ago
.github/workflows	Update GitHub Actions to v1 (#5)	2 months ago
code	Initial commit.	10 months ago
tests	Initial commit.	10 months ago
.gitlab-ci.yml	Add GitLab CI/CD pipeline configuration exa...	5 months ago
.travis.yml	Initial commit.	10 months ago
Jenkinsfile	Initial commit.	10 months ago
License.txt	Initial commit.	10 months ago
README.md	Update README to match GitHub Actions	2 months ago
SECURITY.md	Initial commit.	10 months ago
azure-pipelines.yml	Replace Ubuntu 16.04 with latest	4 months ago

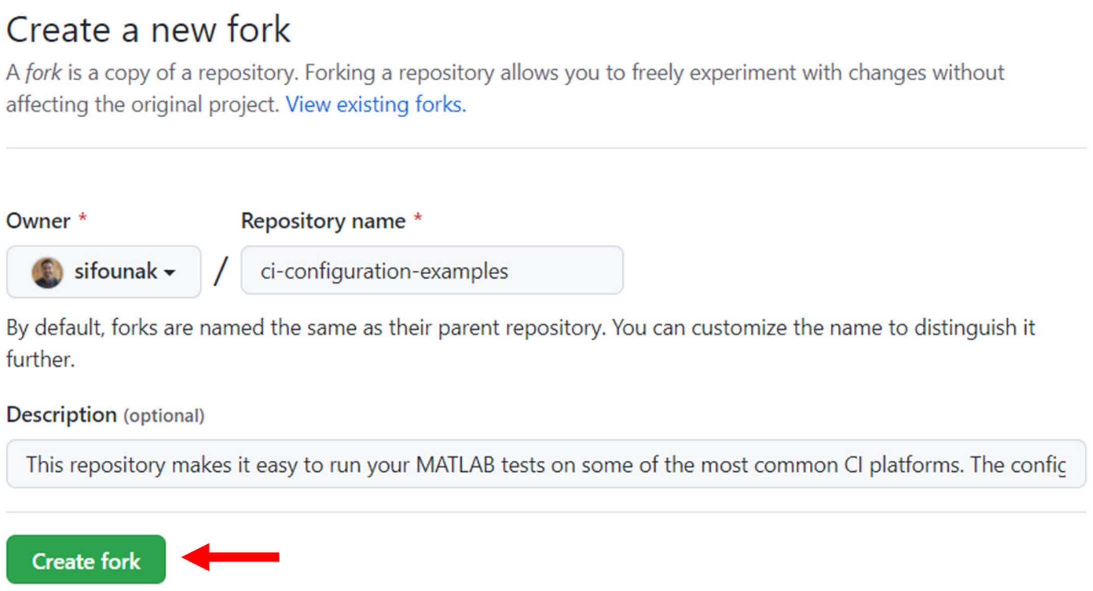
About

This repository makes it easy to run your MATLAB tests on some of the most common CI platforms. The configuration files take care of setting up MATLAB and automatically executing your MATLAB tests.

2C) Tell GitHub what to name your copy of the repository

GitHub offers you the opportunity to rename your forked copy of the repository.

For now, let's just stick with the default name and press the "Create fork" button.



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.

Owner * sifounak / Repository name * ci-configuration-examples

By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

This repository makes it easy to run your MATLAB tests on some of the most common CI platforms. The config

Notes:

- You may need to log into your GitHub account again during this step
- There's no problem with renaming your copy of the repository, but the rest of the workshop guide and screenshots assume the default repository name

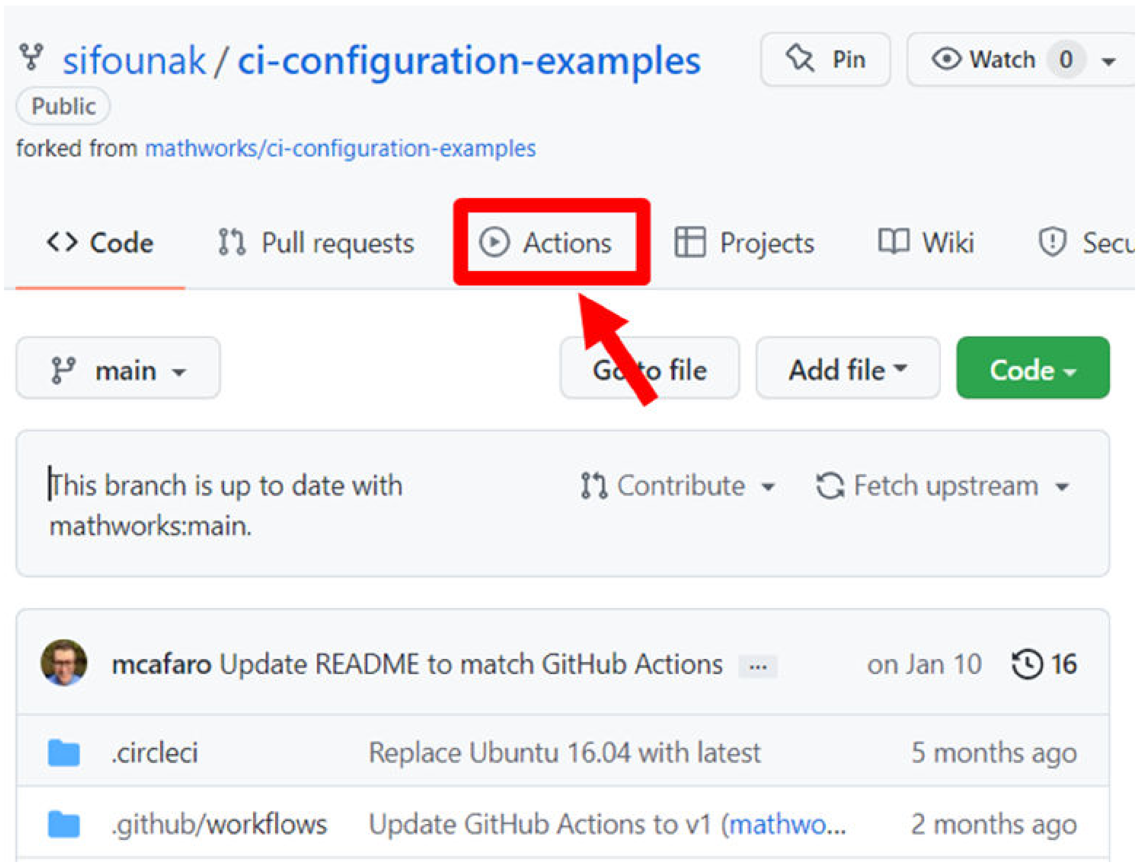
At this point, you will have your own copy of the CI Configuration Examples repository available directly in your GitHub account. You are now ready to start working on your repository!

Part 3: Enable and run GitHub Actions

GitHub offers built-in CI services via their "GitHub Actions" feature. By default, GitHub Actions is disabled for new repositories, so we'll have to enable Actions before we can use them for the workshop.

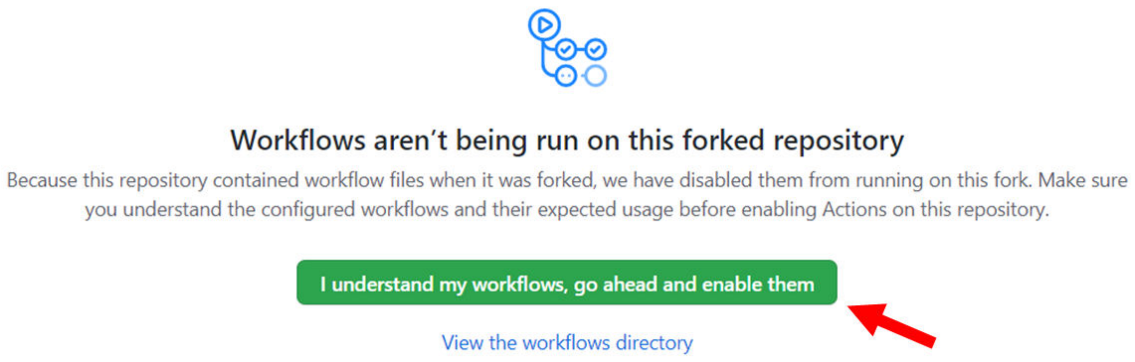
Let's enable GitHub Actions and run our first Action!

3A) Select the "Actions" tab at the top of your repository


sifounak / ci-configuration-examples
Pin
Watch 0
Public
forked from mathworks/ci-configuration-examples
Code
Pull requests
Actions
Projects
Wiki
Secu
main
Go to file
Add file
Code
This branch is up to date with mathworks:main.
Contribute
Fetch upstream

mcafaro Update README to match GitHub Actions ...		on Jan 10 · 16
.circleci	Replace Ubuntu 16.04 with latest	5 months ago
.github/workflows	Update GitHub Actions to v1 (mathwo...	2 months ago

3B) Enable GitHub Actions by clicking on the green button



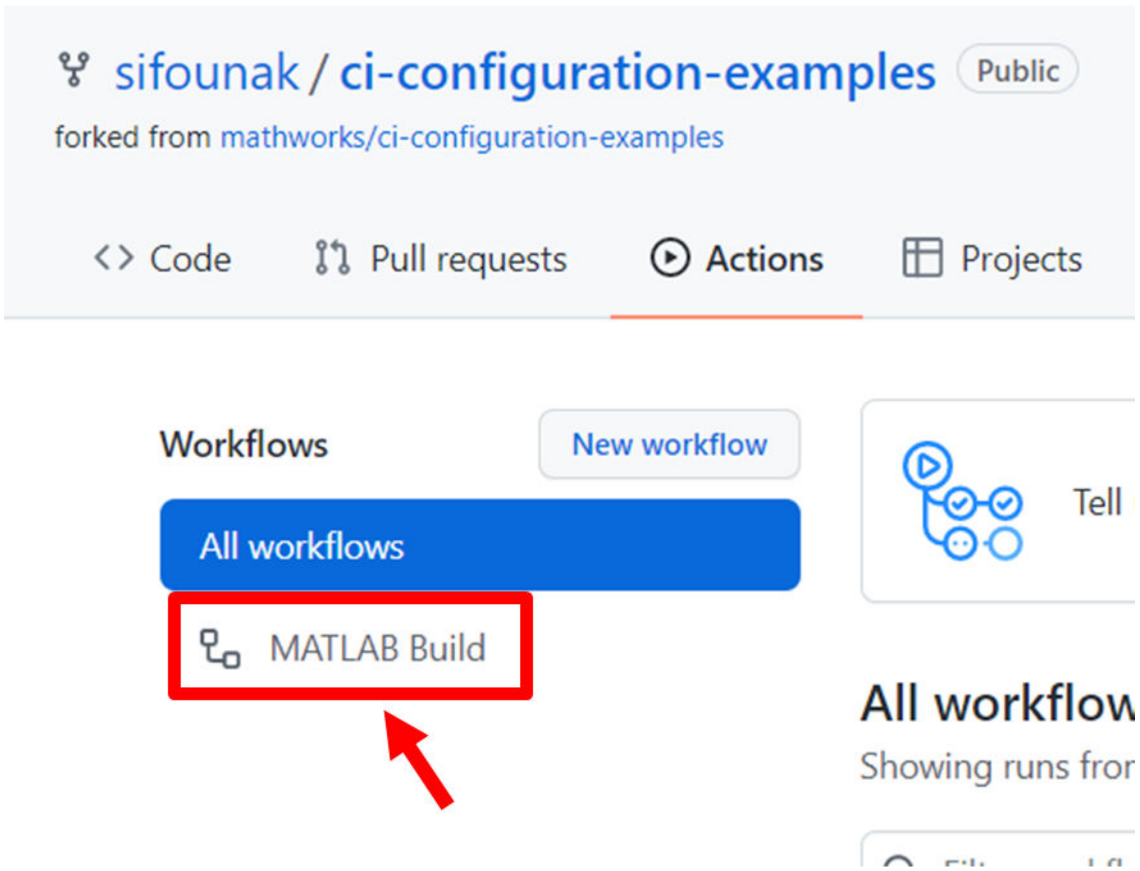
Workflows aren't being run on this forked repository

Because this repository contained workflow files when it was forked, we have disabled them from running on this fork. Make sure you understand the configured workflows and their expected usage before enabling Actions on this repository.

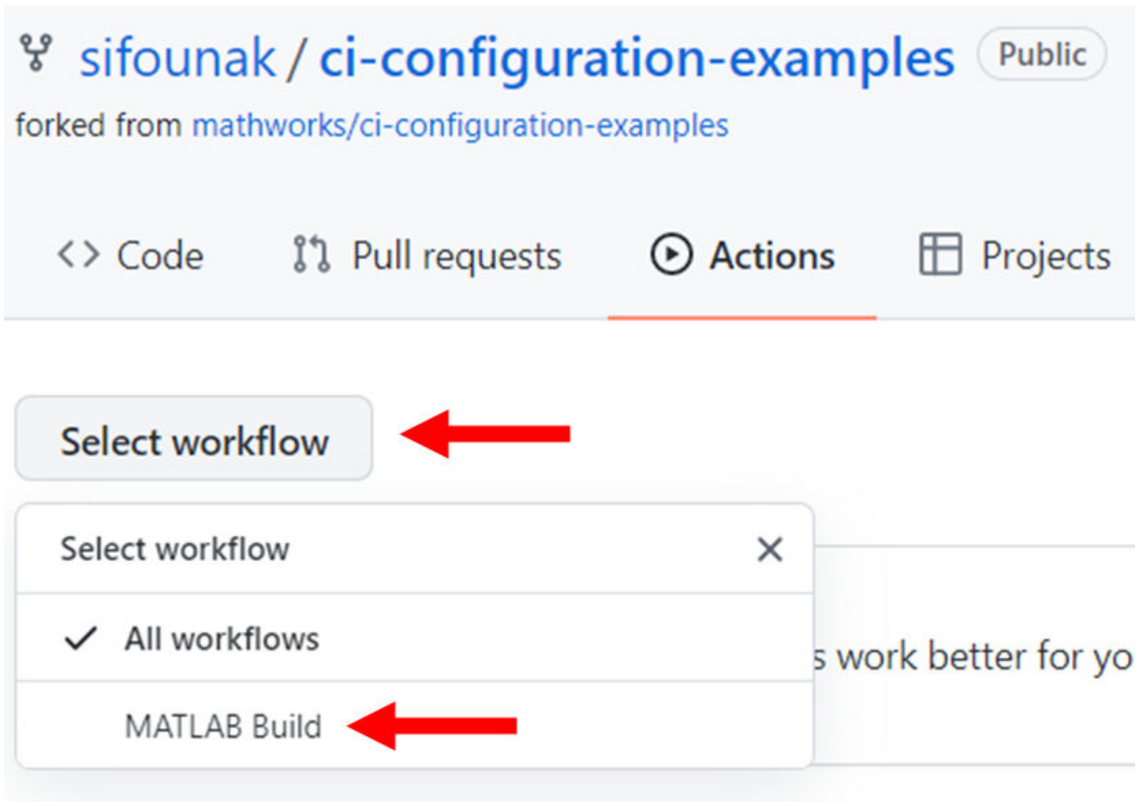
I understand my workflows, go ahead and enable them
View the workflows directory

3C) Select the available "MATLAB Build" action

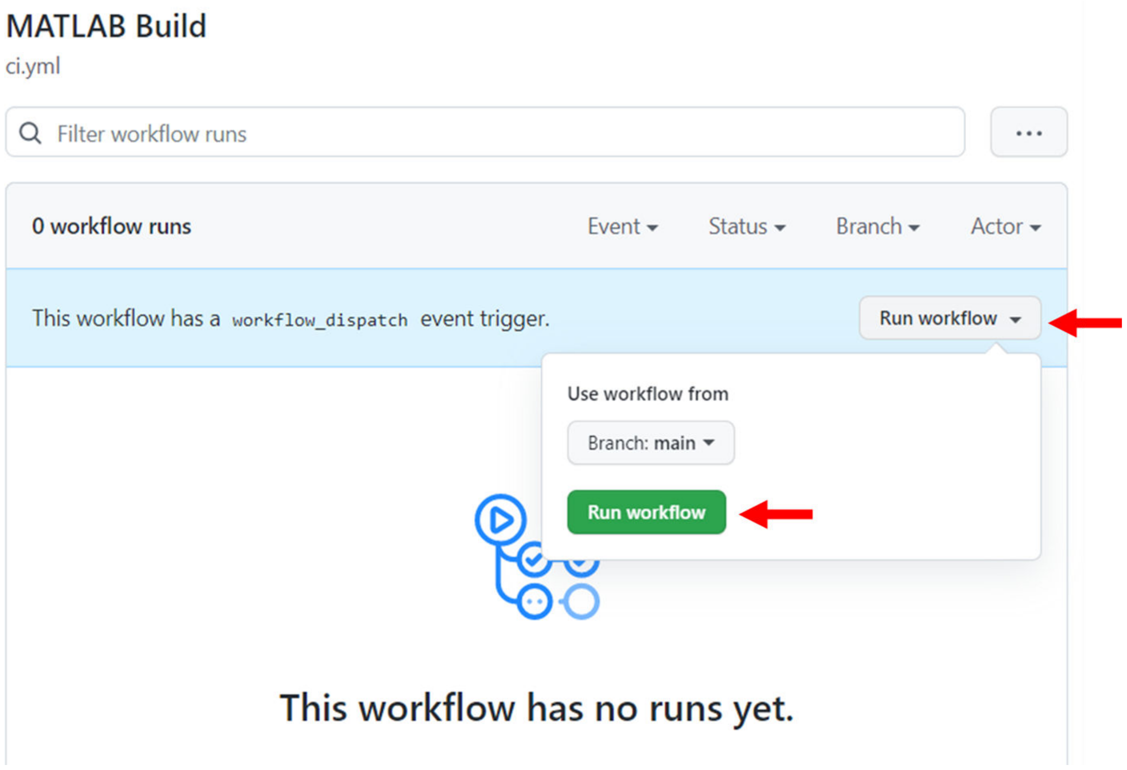
The "CI Configuration Examples" repository we are using provides a built-in "MATLAB Build" action for us to use. It's a really easy way to tell GitHub to run all the MATLAB tests in your repository.



__Note:__ If the width of your browser is too narrow, GitHub's responsive web layout will show you a button that opens a dropdown menu instead.



3D) Run the "MATLAB Build" action by selecting "Run workflow (grey button) > Run workflow (green button)"

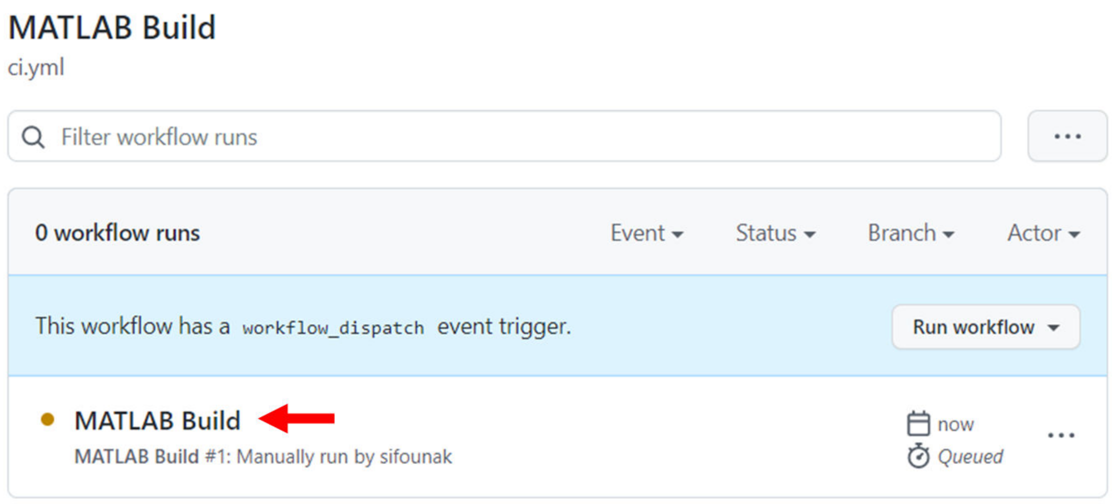


Notes:

- It usually takes between 5-30 seconds for the build to start, but it can take up to several minutes if GitHub's servers are very busy.
- If your workflow gets stuck in the "Queued" state for over two minutes, you may want to cancel and restart the workflow.

3E) Explore the MATLAB Build as it runs

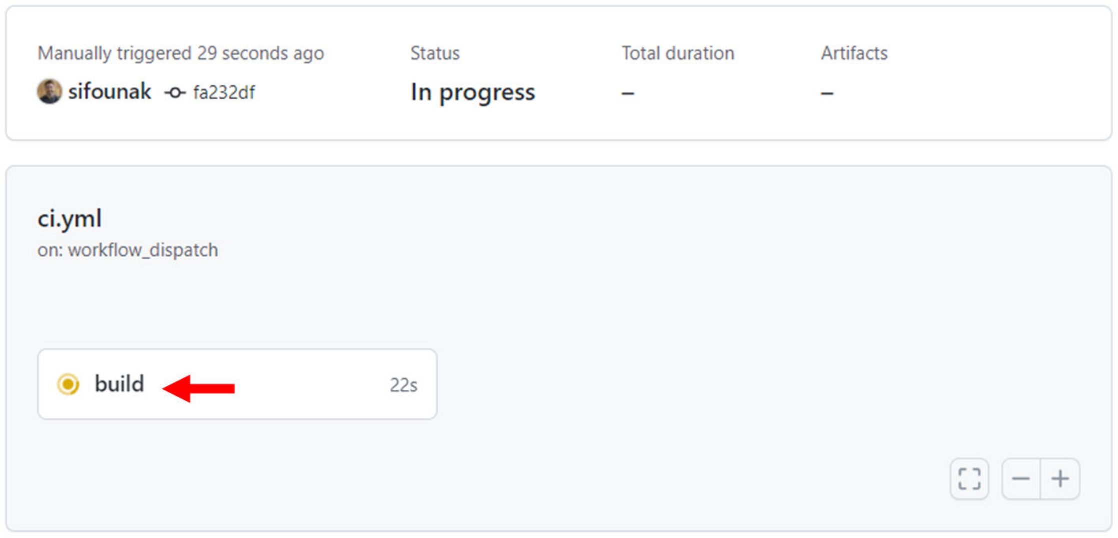
If you'd like to see all the details as the build is running, click on the "MATLAB Build" item.



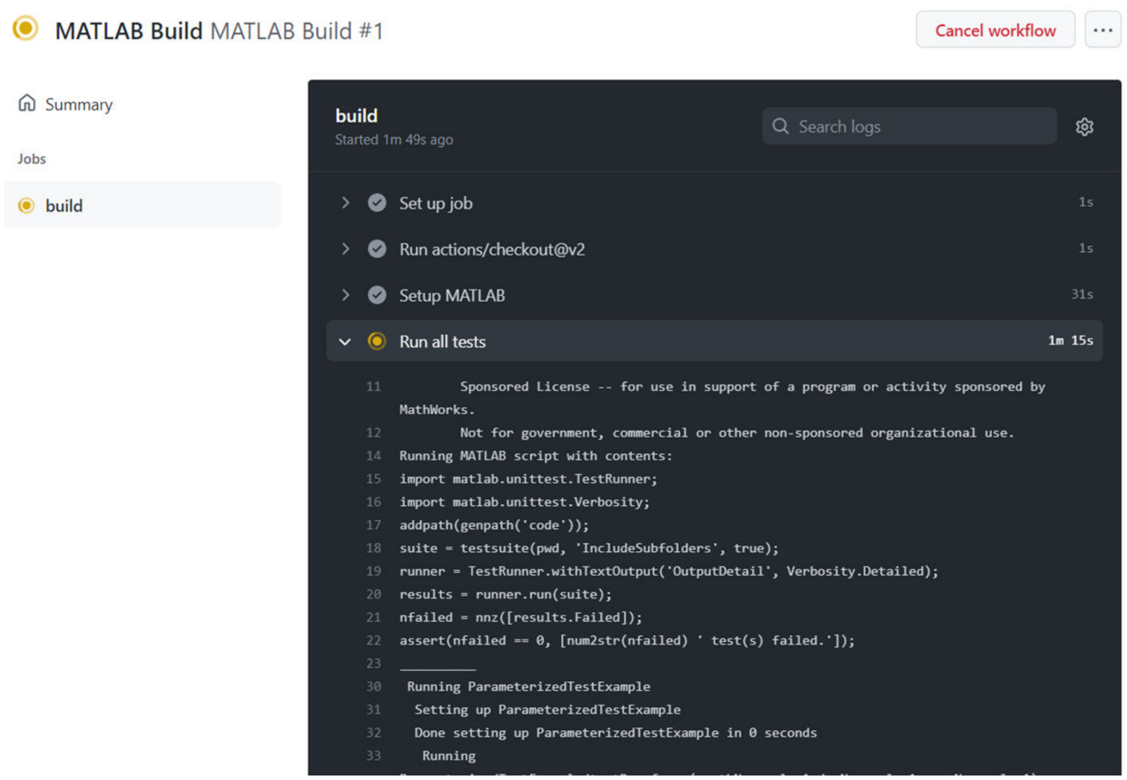
This workflow has a `workflow_dispatch` event trigger.

MATLAB Build
MATLAB Build #1: Manually run by sifounak

Click on the "build" item.



Take a look at the details as the build is running.

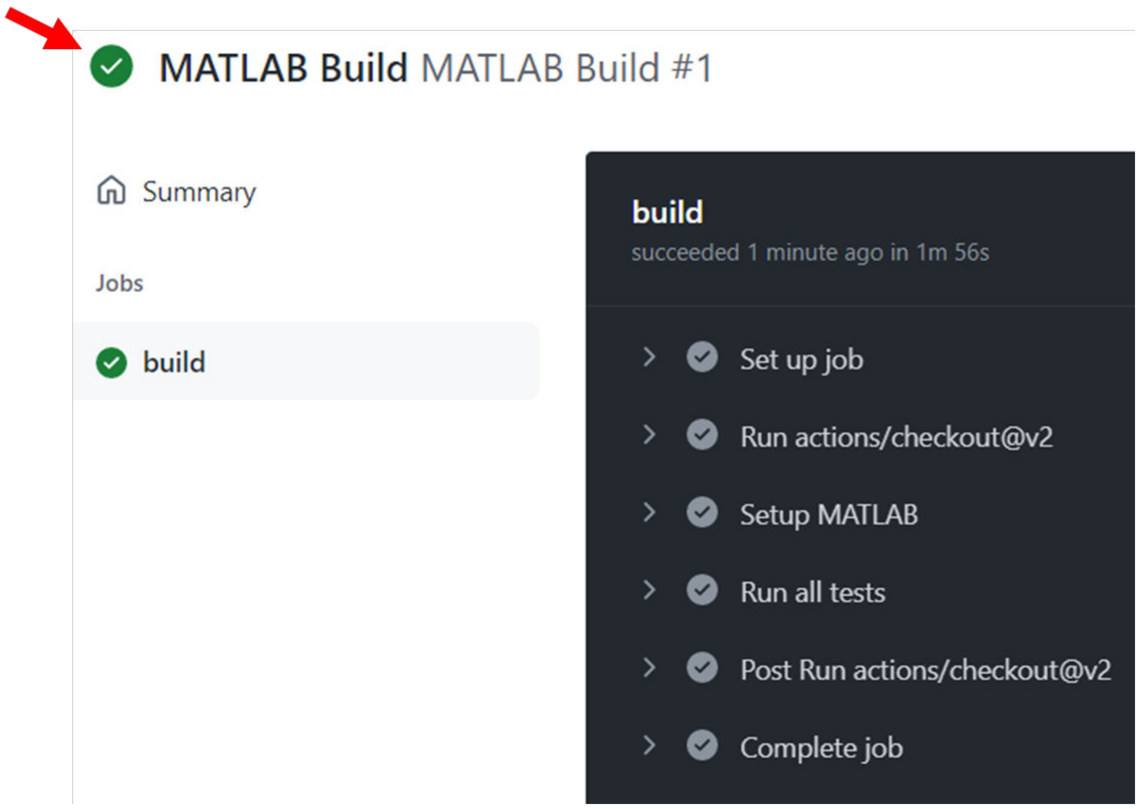


```
        Sponsored License -- for use in support of a program or activity sponsored by
MathWorks.
        Not for government, commercial or other non-sponsored organizational use.
Running MATLAB script with contents:
import matlab.unittest.TestRunner;
import matlab.unittest.Verbosity;
addpath(genpath('code'));
suite = testsuite(pwd, 'IncludeSubfolders', true);
runner = TestRunner.withTextOutput('OutputDetail', Verbosity.Detailed);
results = runner.run(suite);
nfailed = nnz([results.Failed]);
assert(nfailed == 0, [num2str(nfailed) ' test(s) failed.']);
__________
 Running ParameterizedTestExample
  Setting up ParameterizedTestExample
  Done setting up ParameterizedTestExample in 0 seconds
   Running
```

3F) Seeing your first successful GitHub Actions build

The icon next to the build reflects the result of the build. A green check mark means the build ran successfully.

It's awesome to be able to verify that a piece of MATLAB code works, even without launching MATLAB yourself!


MATLAB Build MATLAB Build #1

Summary

Jobs

build

build
succeeded 1 minute ago in 1m 56s

- Set up job
- Run actions/checkout@v2
- Setup MATLAB
- Run all tests
- Post Run actions/checkout@v2
- Complete job

Part 4: Log into MATLAB Online

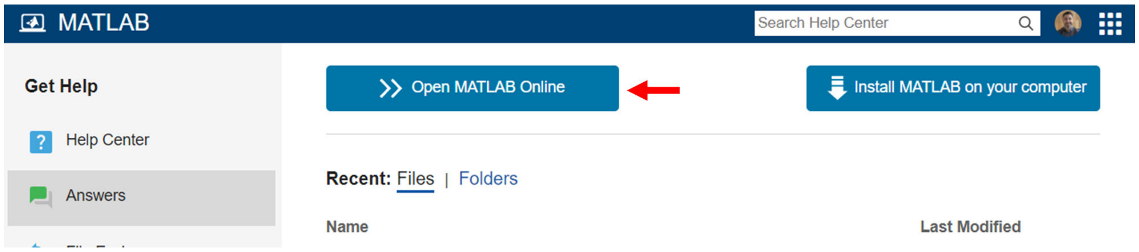
Now that we've forked the repository and verified that everything is working as expected, let's go to MATLAB Online and get ready to start making changes to our code!

Note: If you are using a desktop installation of MATLAB, you can simply launch your desktop MATLAB go directly to **Step 5**

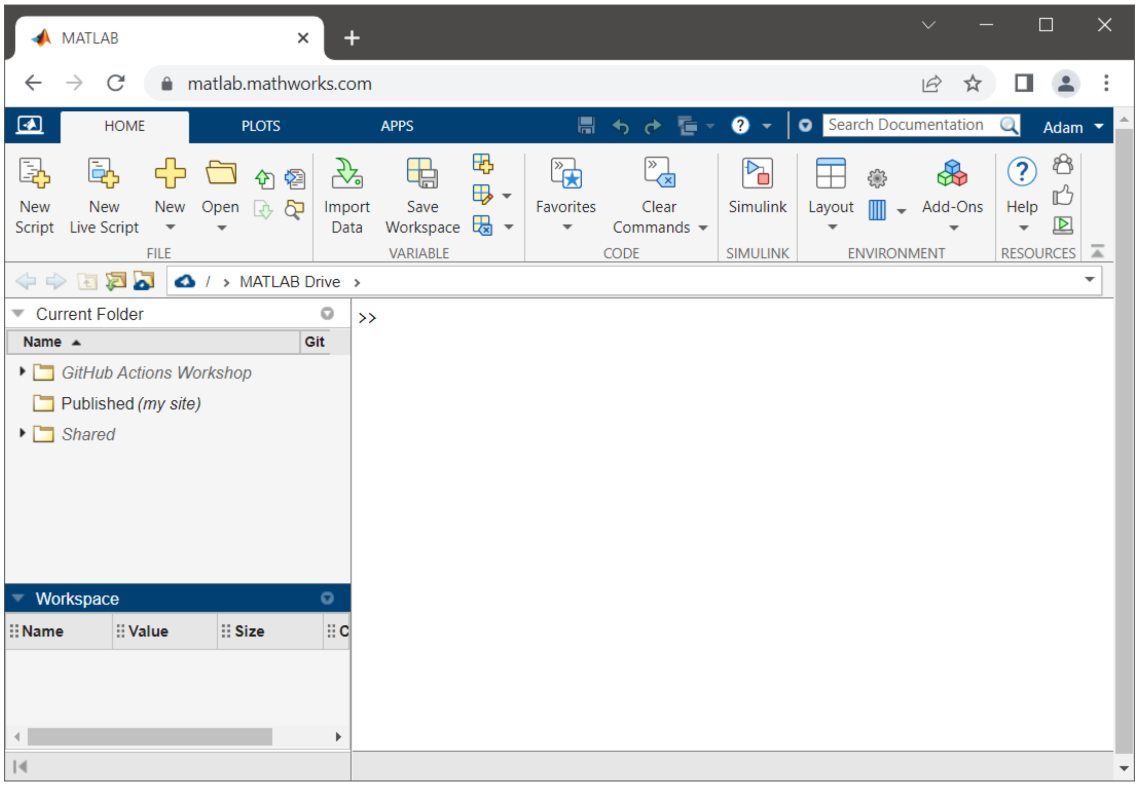
4A) Go to https://matlab.mathworks.com and log in using your MathWorks account credentials

4B) Open MATLAB Online

Use the button at the top of the screen to open MATLAB Online.



The MATLAB Online interface looks very similar to the desktop version of MATLAB.

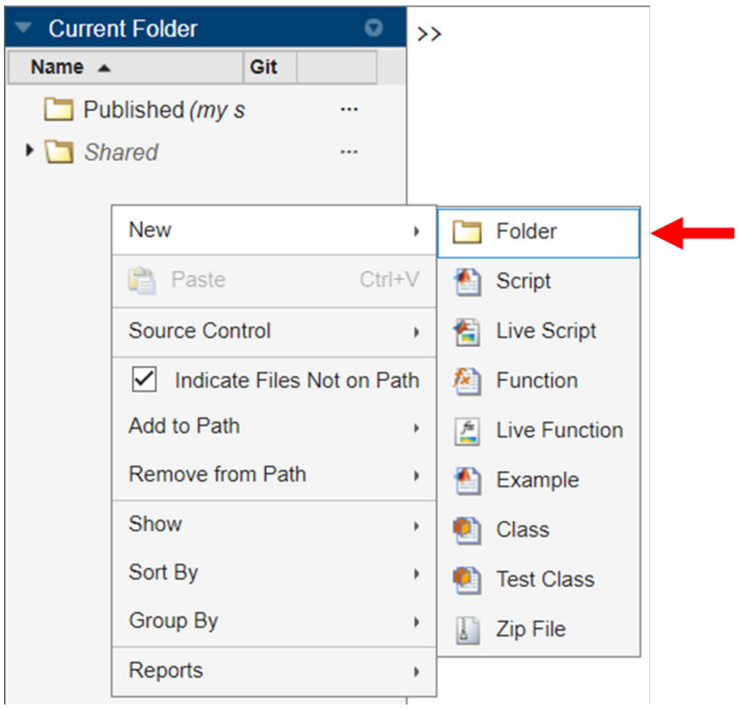


Part 5: Clone your repo to MATLAB Online

Now that we're in MATLAB Online, let's grab a copy of your repository and start working on it.

5A) Create a new folder

Right-click an empty space in the Current Folder Browser > New > Folder



Note: You can name the directory anything you like, but we recommend using the same name as your repository (in this case, "ci-configuration-examples")

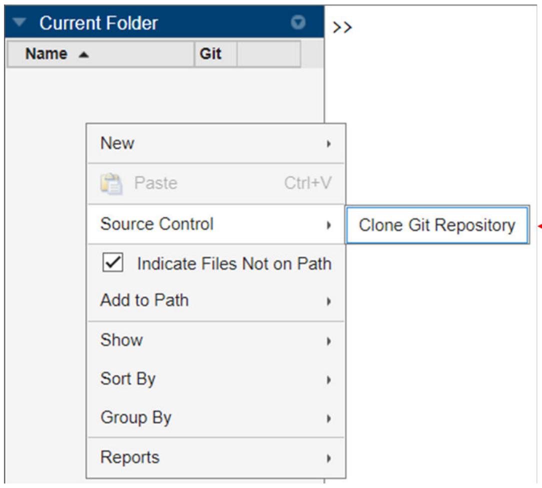
5B) Go into the new folder

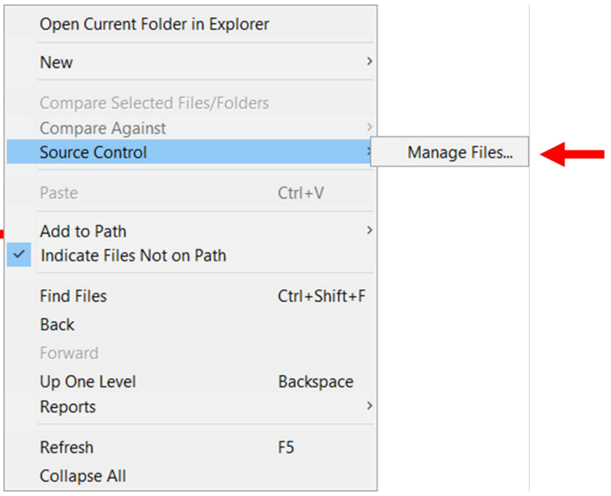
Double-click the folder

5C) Clone the repository into this folder

Right-click in the Current Folder Browser > Source Control > Clone repo

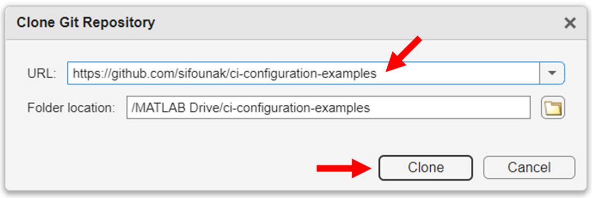
MATLAB Online: Desktop MATLAB:





5D) Paste the URL to your forked repository and click "Clone"

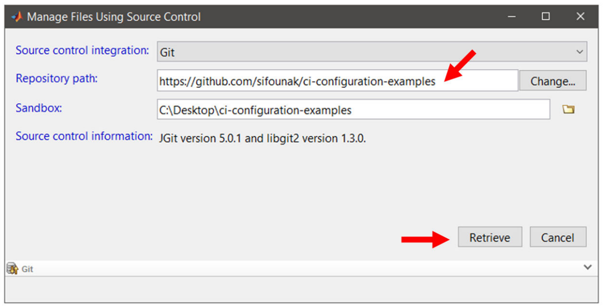
<u>MATLAB Online:</u>


Clone Git Repository

URL: https://github.com/sifounak/ci-configuration-examples

Folder location: /MATLAB Drive/ci-configuration-examples

<u>Desktop MATLAB:</u>


Manage Files Using Source Control

Source control integration: Git

Repository path: https://github.com/sifounak/ci-configuration-examples

Sandbox: C:\Desktop\ci-configuration-examples

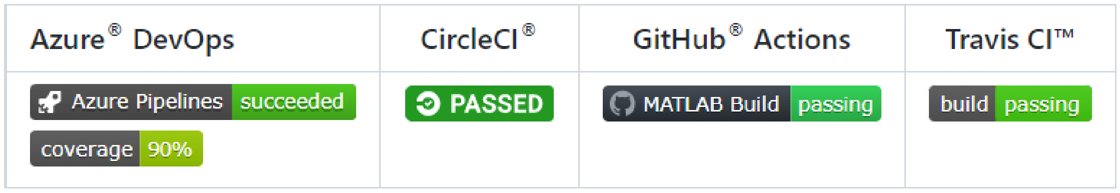
Source control information: JGit version 5.0.1 and libgit2 version 1.3.0.

After this step, your current directory should now contain the code from your GitHub repository.

<u>Notes:</u> It's always a great idea to run the tests in a repository before you start working with the code in that repository

- Since we already ran the tests via GitHub Actions, we're going to skip this step for now
- **Workshop Homework Task H3C** walks you through the process of running the tests in your repository

Part 6: Update the badge in the `README.md` file



Azure® DevOps	CircleCI®	GitHub® Actions	Travis CI™
Azure Pipelines succeeded coverage 90%	PASSED	MATLAB Build passing	build passing

Repository badges are defined in the `README.md` file, and they are always hard-coded to a specific repository. Since we copied this badge from another repository, we will need to update the badge code to make the badge reflect the status of our current repository.

6A) Open the `README.md` file

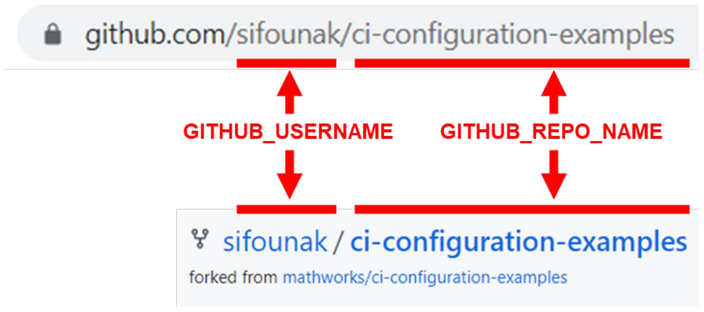
Note: The `README.md` file is located in the root of the repository

6B) Replace the top badge table (first 3 lines of the `README.md` file) with the following code and customize the GitHub username and repository name pieces:

- `[![MATLAB](https://github.com/`**`GITHUB_USERNAME`**`/`**`GITHUB_REPO_NAME`**`/actions/workflows/ci.yml/badge.svg)](https://github.com/`**`GITHUB_USERNAME`**`/`**`GITHUB_REPO_NAME`**`/actions/workflows/ci.yml)`

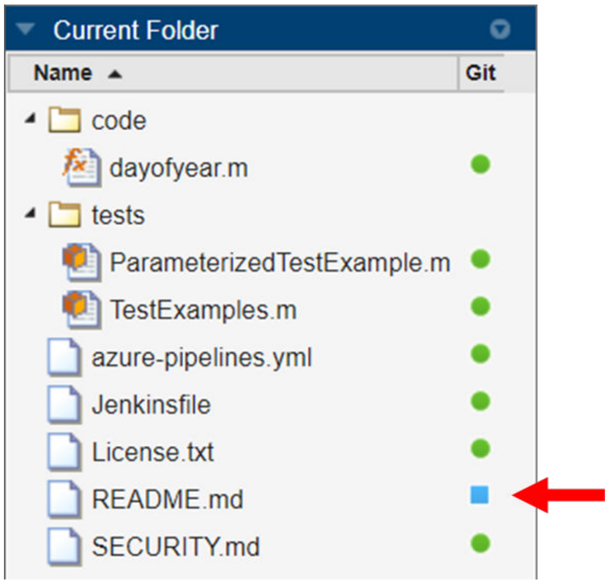
Notes:

- Replace both instances of **GITHUB_USERNAME** with your GitHub username
- Replace both instances of **GITHUB_REPO_NAME** with your GitHub repository name
- Both of these can be easily found in the URL or the main page of your forked GitHub repository


github.com/sifounak/ci-configuration-examples
GITHUB_USERNAME
GITHUB_REPO_NAME
sifounak / ci-configuration-examples
forked from mathworks/ci-configuration-examples

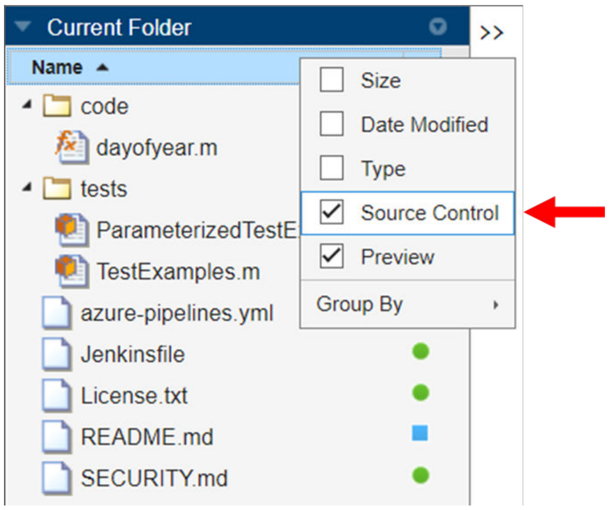
6C) Save and close the file

The Git status column of the `README.md` file should now be a blue square, indicating that the file has been modified.



Notes:

- If the Git status column isn't visible, you can show it by right-clicking the Current Folder Browser header > Check the "Source Control" item
- The Git status icons may sometimes take a few minutes to update



Part 7: Let's introduce a bug in our code

One of the biggest values of continuous integration systems is that they can help you catch bugs that you might have missed during development.

To quickly see this in action, we're going to artificially introduce a bug in our code, and then submit our changes without first running our tests. (Submitting code without first running your tests is one of the most common development mistakes.)

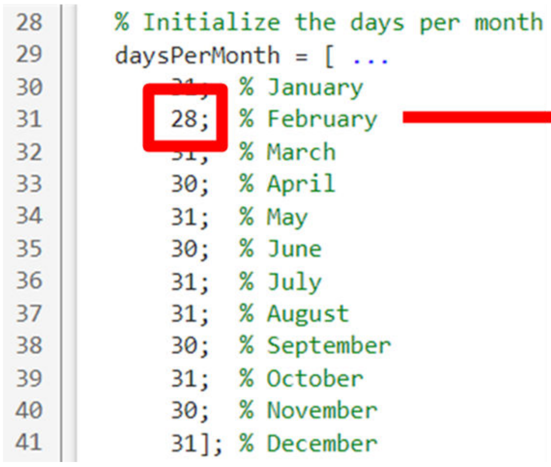
7A) Open `dayofyear.m`

Note: `dayofyear.m` is located in the "`code`" folder

7B) Change one of the values in the daysPerMonth array

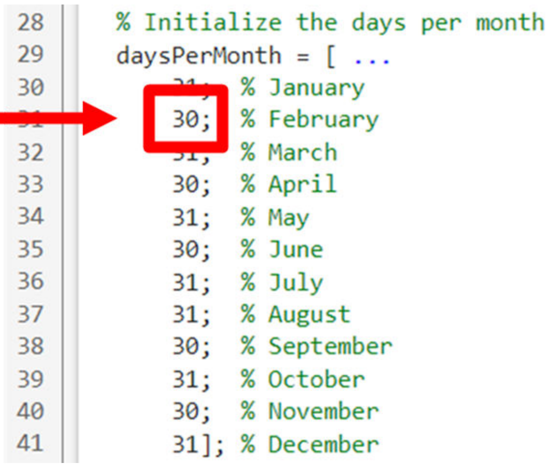
In this case, we're going to change the second element (corresponding to the month of February) from 28 days per month, to 30 days per month.

Before:



```
% Initialize the days per month
daysPerMonth = [ ...
    31;  % January
    28;  % February
    31;  % March
    30;  % April
    31;  % May
    30;  % June
    31;  % July
    31;  % August
    30;  % September
    31;  % October
    30;  % November
    31]; % December
```

After:



```
% Initialize the days per month
daysPerMonth = [ ...
    31;  % January
    30;  % February
    31;  % March
    30;  % April
    31;  % May
    30;  % June
    31;  % July
    31;  % August
    30;  % September
    31;  % October
    30;  % November
    31]; % December
```

7C) Save and close the file

Part 8: Commit your changes

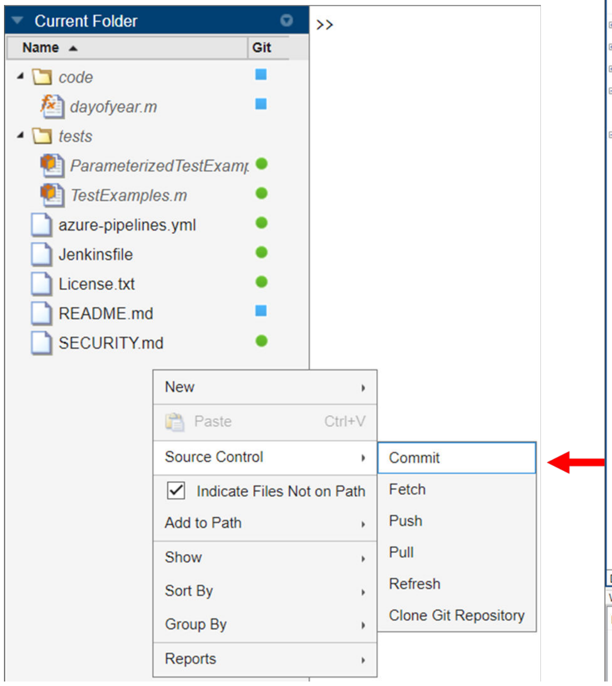
Now that we've updated our badge and introduced a bug, let's commit our changes.

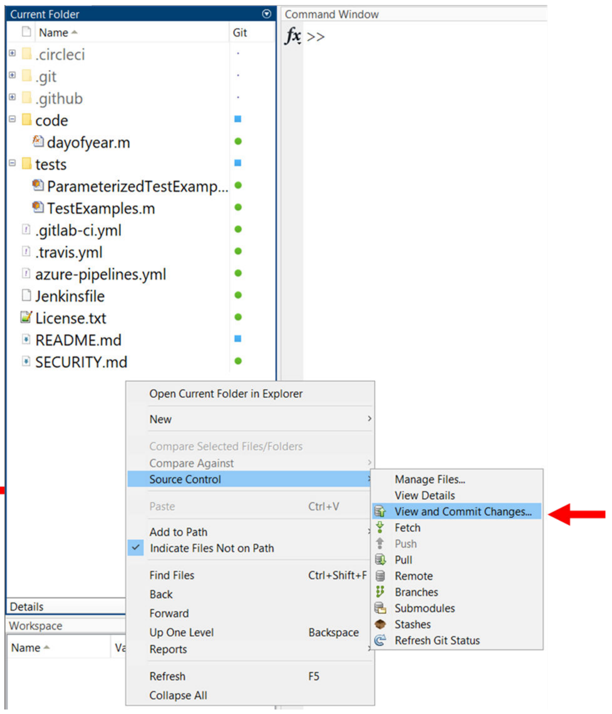
8A) Open the commit dialog

Right-click in the whitespace of the Current Folder Browser > Source Control > Commit

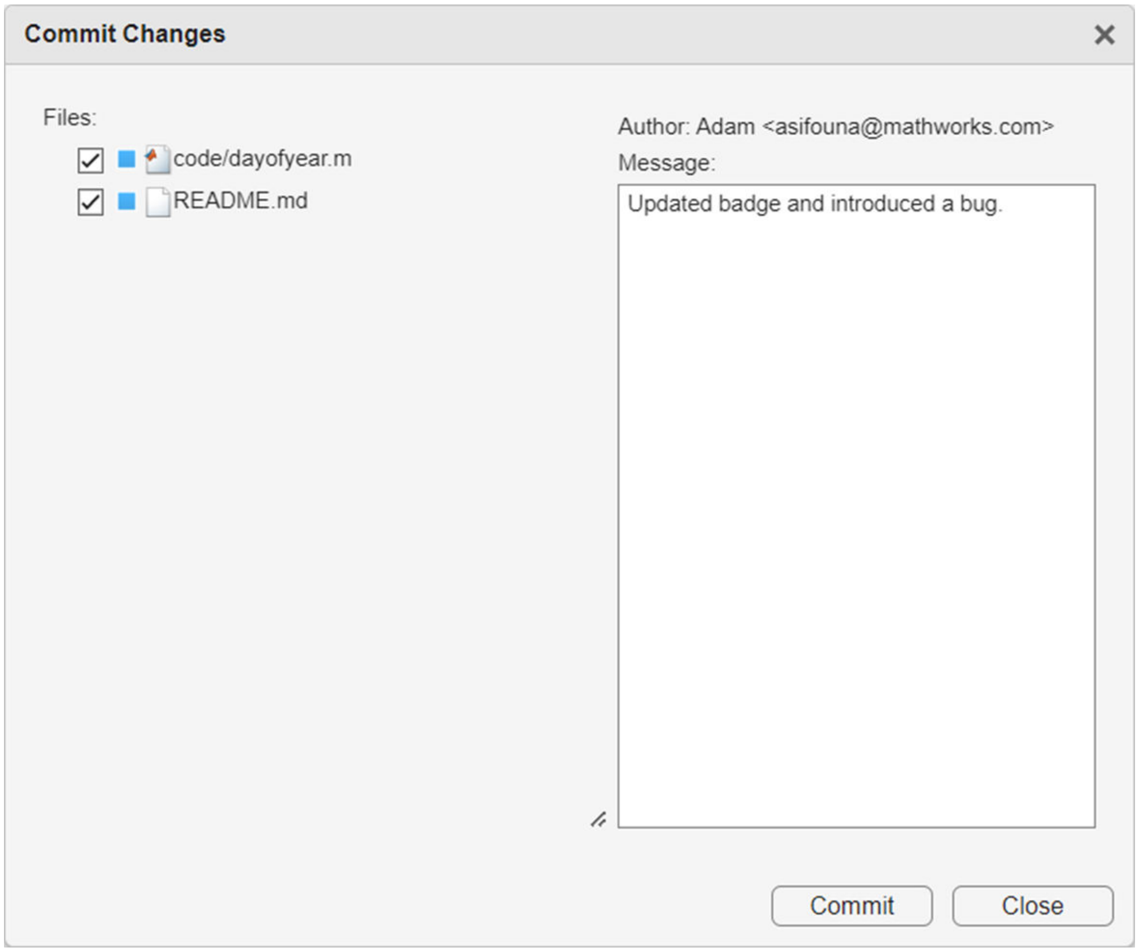
MATLAB Online

Desktop MATLAB





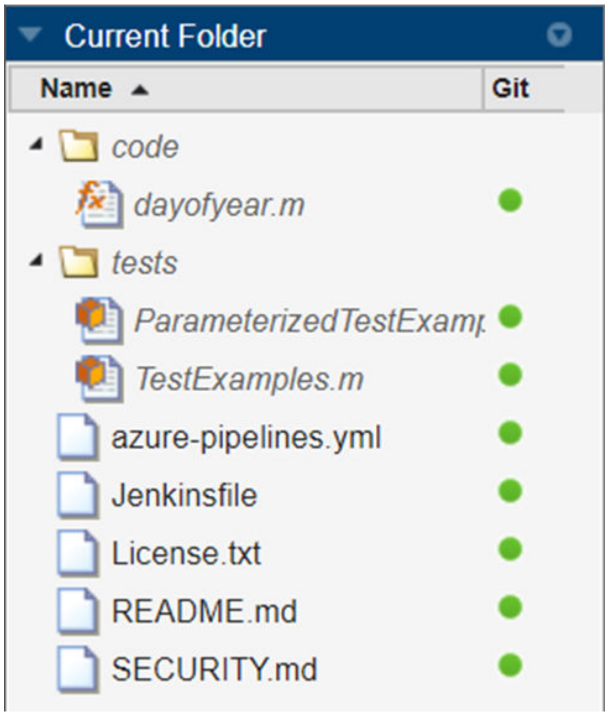
8B) Add a commit message and press "Commit"



code/dayofyear.m
README.md

Updated badge and introduced a bug.

All of your changes are now recorded in source control, and all of the Git status icons should be green.



Name	Git
code	
dayofyear.m	●
tests	
ParameterizedTestExam	●
TestExamples.m	●
azure-pipelines.yml	●
Jenkinsfile	●
License.txt	●
README.md	●
SECURITY.md	●

Part 9: Pushing your changes to GitHub from MATLAB Online the very first time

Before we begin: A note about GitHub's move away from passwords to personal access tokens (PATs)

Over the last few years, GitHub has been increasing its focus on security. One of the major user-facing ways GitHub is enforcing some of these better security practices is by moving away from the use of passwords, and recommending that people use personal access tokens instead.

When you install Git on your desktop, it comes with the GitHub Credential Manager (GCM). The GCM will accept a username/password combination, and it will securely get and store a personal access token for you.

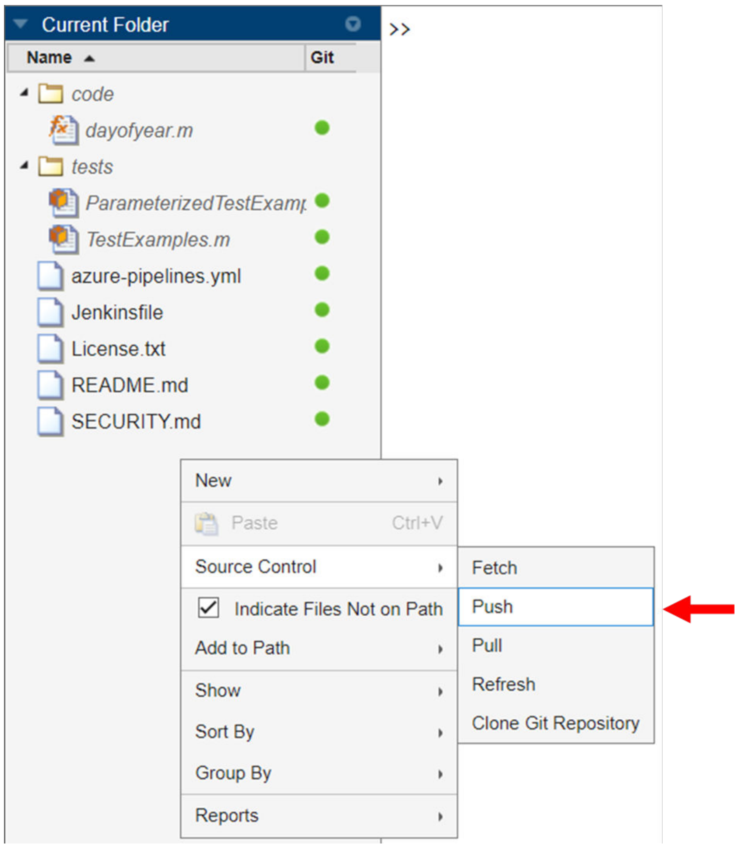
For this MATLAB Online session, we will need to manually create a personal access token for MATLAB Online to use when you push to GitHub.

GitHub provides directions for how to create a personal access token, but we have also included the directions below for convenience.

In this section, we will:

- start the "push" operation
- switch to GitHub to generate the personal access token
- copy the token to your clipboard
- paste the token into the login dialog
- complete the "push" operation

9A) Right-click a blank area in the Current Folder Browser > Source Control > Push

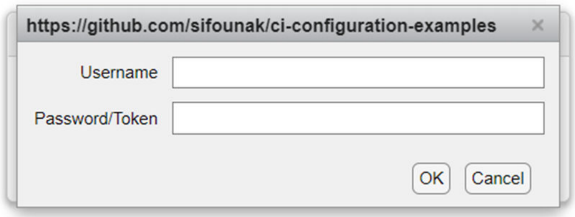


The push operation will bring up a different dialog, depending on whether you are doing the workshop in MATLAB Online or using a desktop installation of MATLAB.

<u>MATLAB Online:</u>

MATLAB Online will prompt you for credentials, but **<u>do not fill out the credentials yet</u>**.

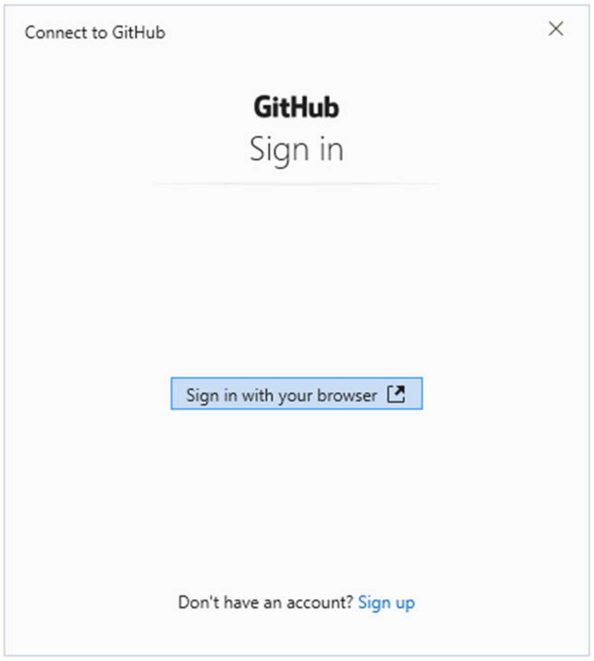
Steps 9B-9E will guide you through the process of creating the personal access token you will need for this dialog.



<u>Desktop MATLAB:</u>

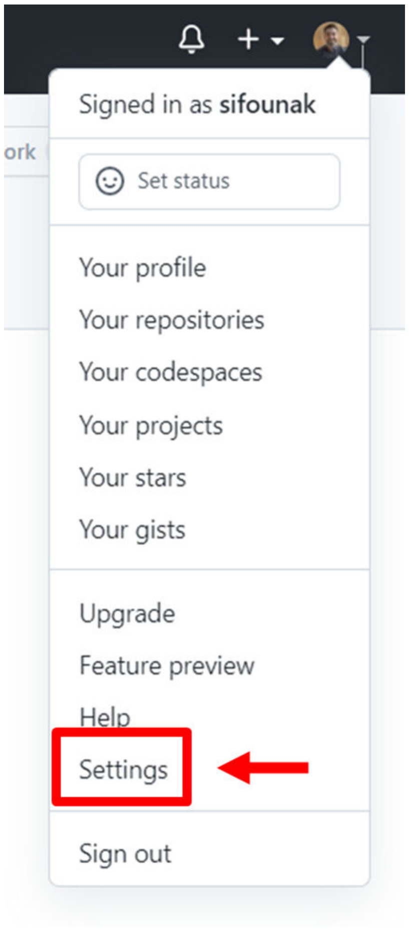
When using desktop MATLAB:

- the GitHub interactions will be handled by the GitHub Credential Manager
- you can safely skip the rest of Step 9 and move on to **Step 10**

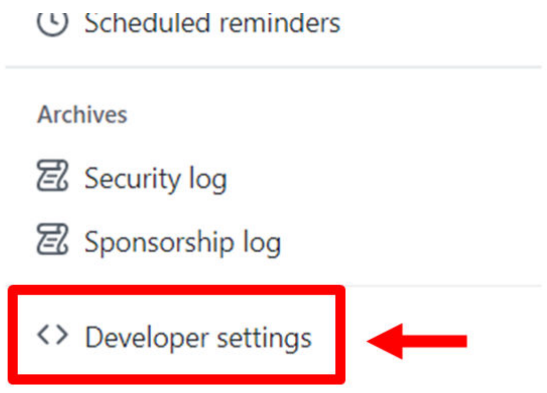


9B) Go to your GitHub account Settings > Developer settings > Personal access tokens

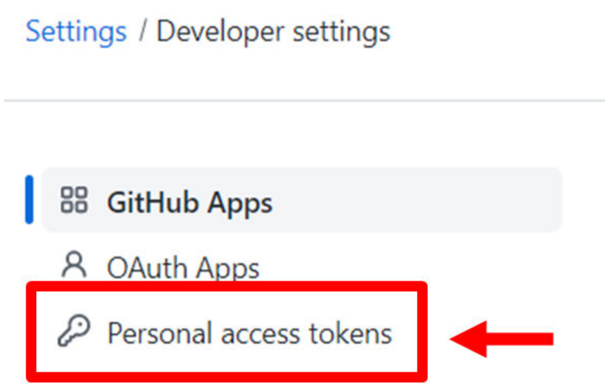
Click on your profile icon in the top right corner of your GitHub page and select "Settings"



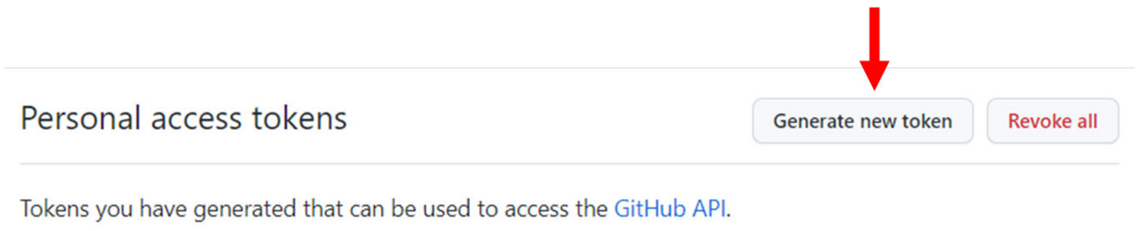
Select "Developer settings" at the bottom of the left-hand navigation menu



Select "Personal access tokens"



9C) Press the "Generate new token" button

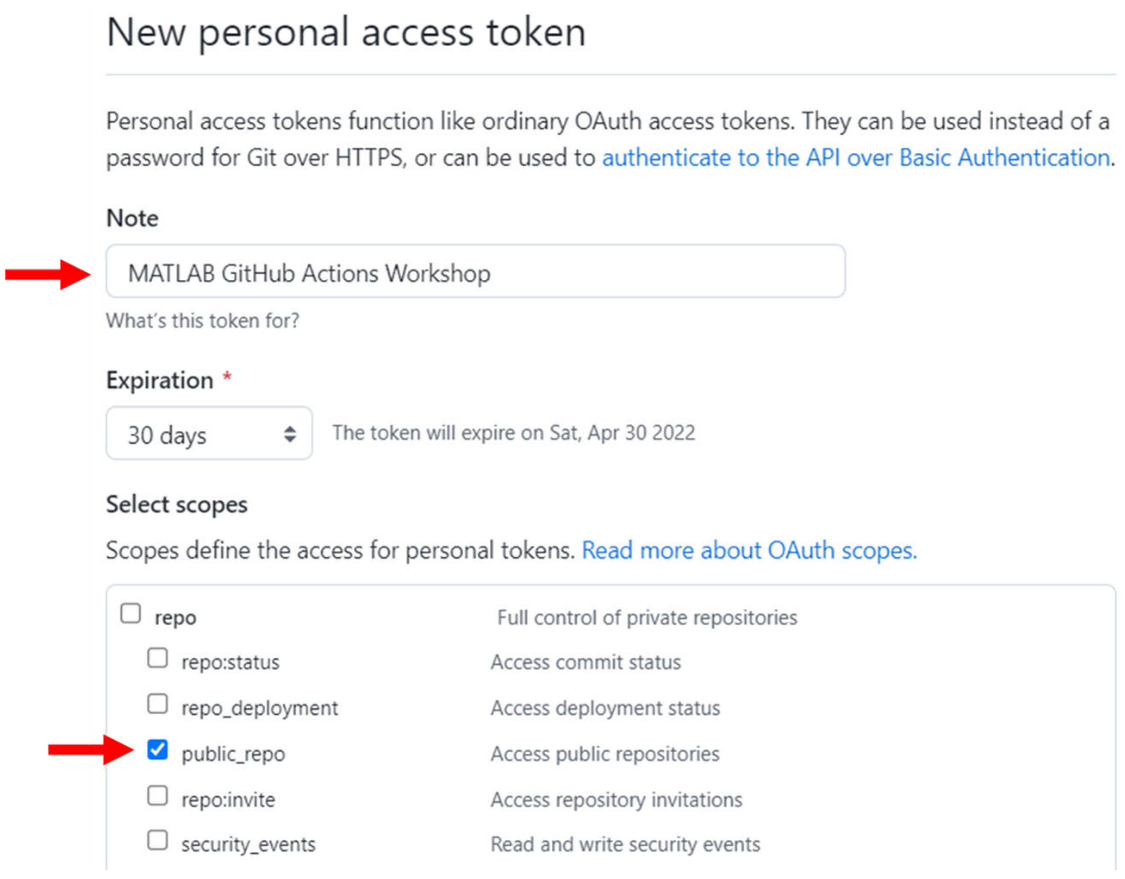


Tokens you have generated that can be used to access the GitHub API.

Note: You may need to log into GitHub again to complete this action.

9D) Customize your personal access token

- Add a note to make it easy for you identify which service/application this token is for
- Select the "repo / public_repo" scope



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

MATLAB GitHub Actions Workshop

What's this token for?

Expiration *

30 days

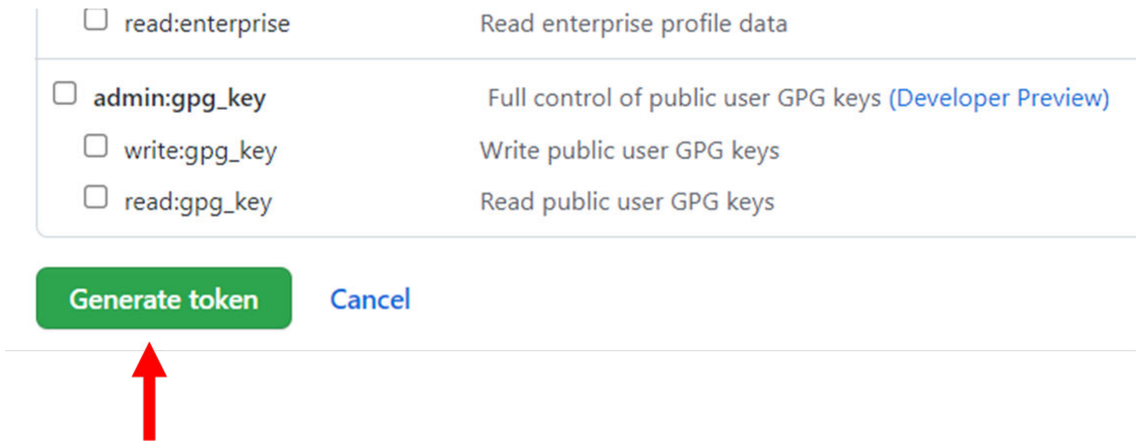
The token will expire on Sat, Apr 30 2022

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

9E) Press the "Generate token" button at the bottom of the page



☐ read:enterprise	Read enterprise profile data
☐ admin:gpg_key	Full control of public user GPG keys (Developer Preview)
☐ write:gpg_key	Write public user GPG keys
☐ read:gpg_key	Read public user GPG keys

9F) Copy your personal access token to your clipboard

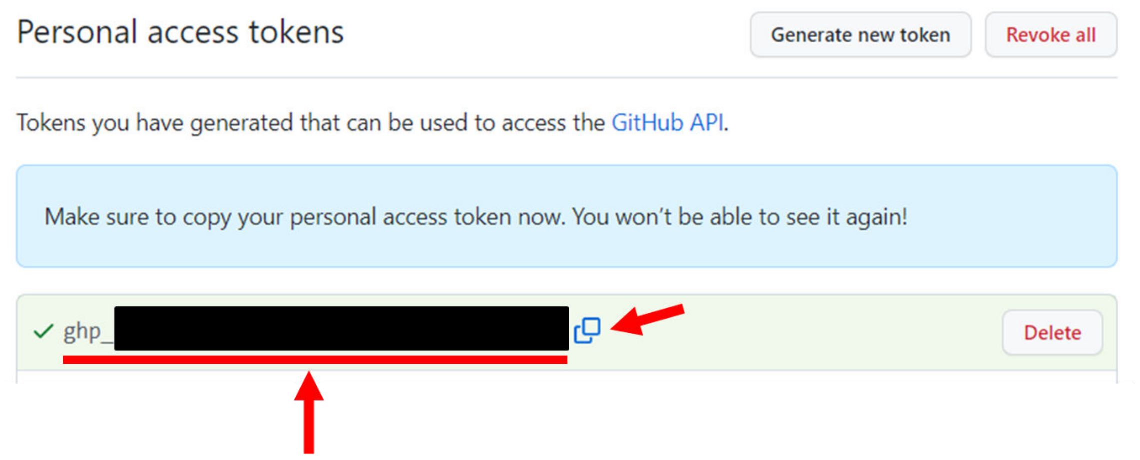
Token strings are meant to be treated like passwords:

- They are supposed to be difficult to memorize or guess
- They are not meant to be written down

- They are not meant to be shared with others

In addition, any service that generates token strings will only allow you to see or copy the token string the moment the token is created. As soon as you refresh the page or navigate away, the token string will be hidden forever. This protects you from someone breaking into your account, stealing all of your tokens, and using them to access all of your services.

If you forget to copy your token string or you need to give another service access to your repository at a later time, you will need to create a new token.


Personal access tokens

Generate new token | Revoke all

Tokens you have generated that can be used to access the GitHub API.

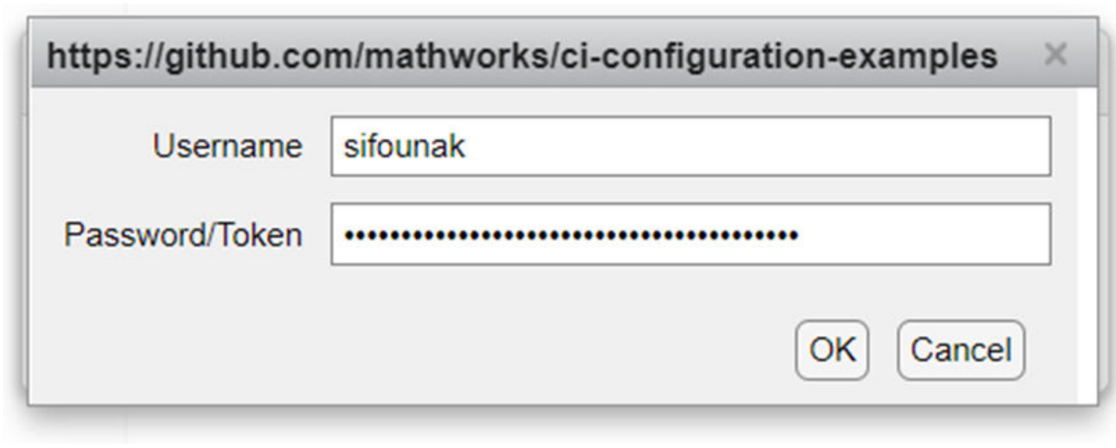
Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_[redacted]

Delete

Note: Your personal access token should start with "ghp_" followed by 36 alphanumeric characters.

9F) Paste the personal access token into the MATLAB Online Git dialog


https://github.com/mathworks/ci-configuration-examples

Username: sifounak

Password/Token: ••

9G) Press the "OK" button

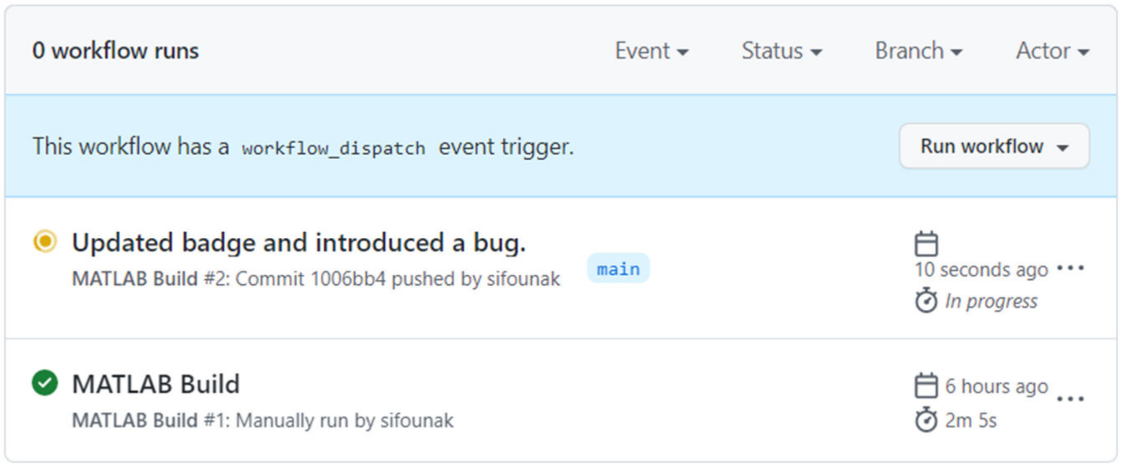
At this point, all of your changes will be pushed to GitHub.

Part 10: View the GitHub Actions test results

One of the best parts of continuous integration systems is that the CI systems can be configured to automatically run all of your tests whenever a new change is pushed to the repository. The default GitHub Actions configuration file we've provided will automatically run all of your tests every time a new commit is pushed to the main branch of your repository.

10A) In your GitHub repository, go to the "Actions" section to see your automated testing in action

You can see your latest code changes are automatically being tested by GitHub Actions.

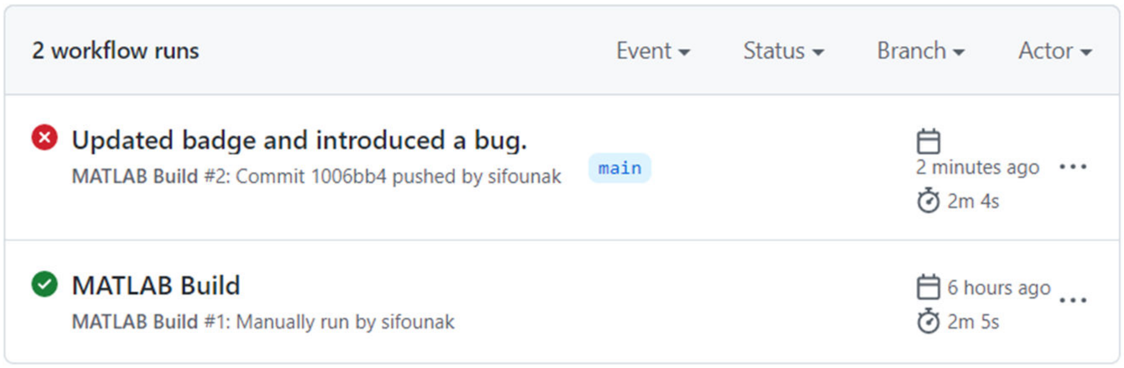


This workflow has a `workflow_dispatch` event trigger.

Updated badge and introduced a bug.
MATLAB Build #2: Commit 1006bb4 pushed by sifounak
10 seconds ago
In progress

MATLAB Build
MATLAB Build #1: Manually run by sifounak
6 hours ago
2m 5s

After about 1-2 minutes, you should see:

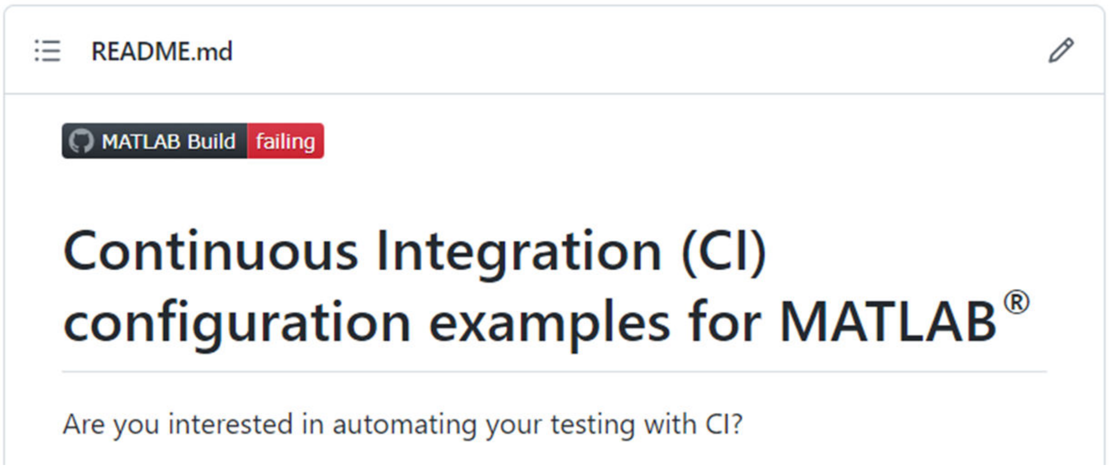


Updated badge and introduced a bug.
MATLAB Build #2: Commit 1006bb4 pushed by sifounak
2 minutes ago
2m 4s

MATLAB Build
MATLAB Build #1: Manually run by sifounak
6 hours ago
2m 5s

Oh no! It looks like at least one of our tests failed!

The badge on your repository main page should also reflect the test failure.



Continuous Integration (CI) configuration examples for MATLAB®

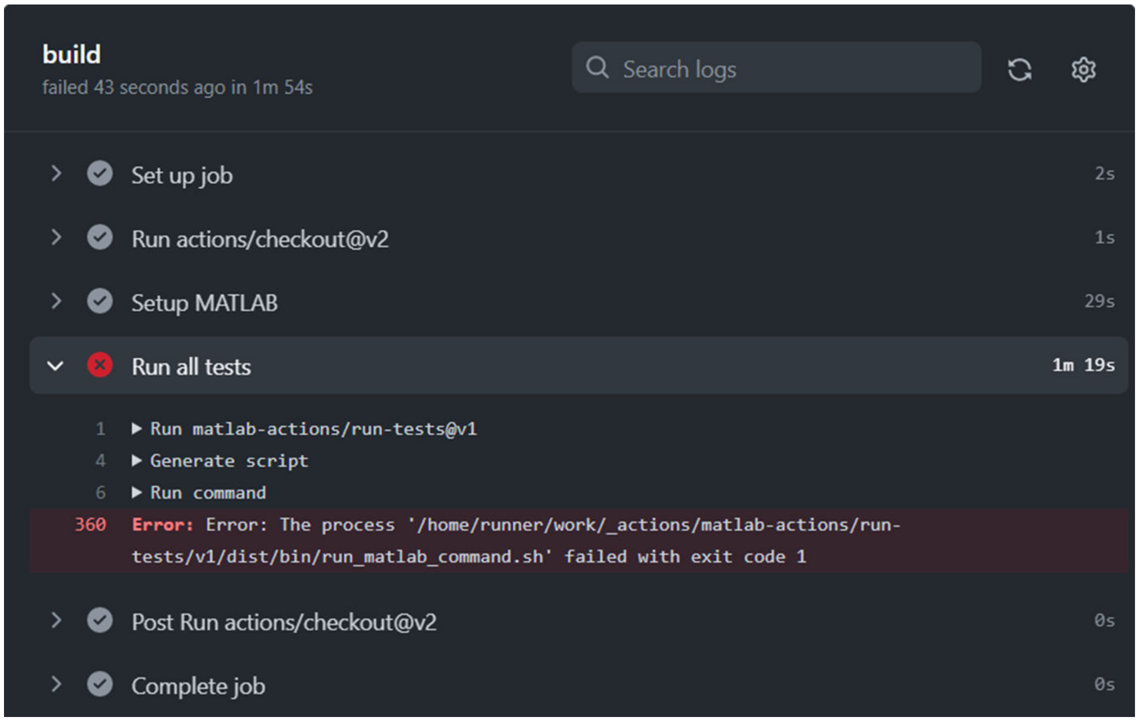
Are you interested in automating your testing with CI?

Note: The badge on your repository's main page may not update right away due to web browser image caching. To force the badge image to be reloaded, you may do either of the following actions:

- Hold the "`Shift`" key as you press the refresh icon
- Press "`Shift + F5`"

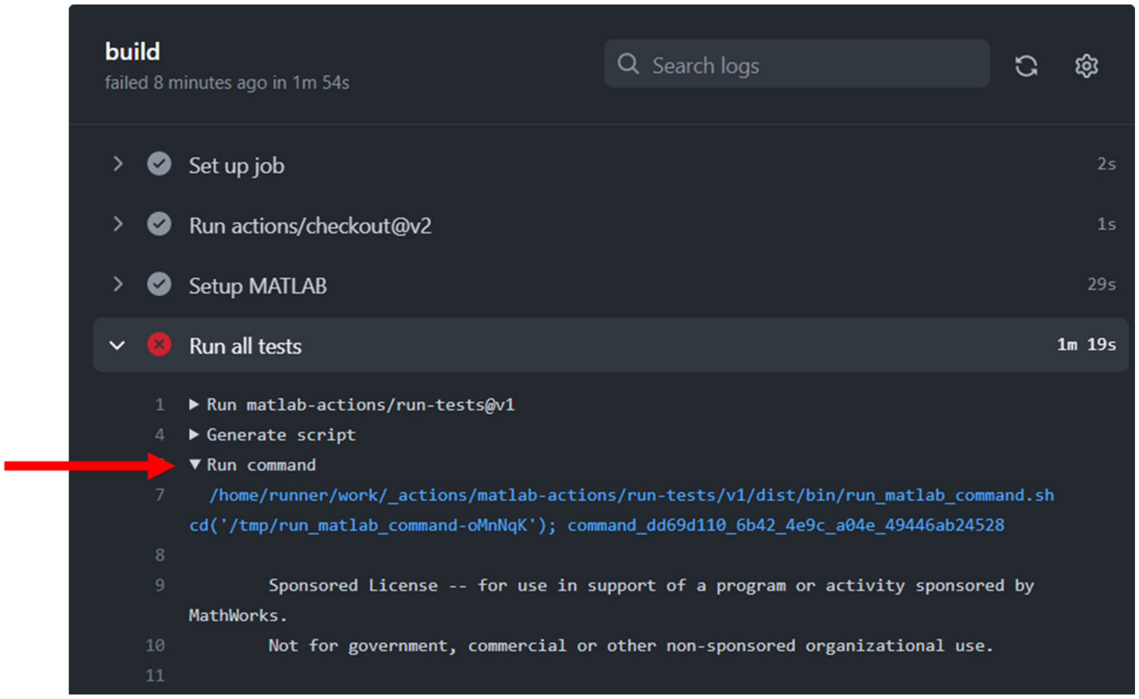
10B) Investigate the errors

Click through the build items in the "Actions" area until you reach the build log.



```
Set up job                                   2s
Run actions/checkout@v2                      1s
Setup MATLAB                                 29s
Run all tests                                1m 19s
    ▶ Run matlab-actions/run-tests@v1
    ▶ Generate script
    ▶ Run command
360 Error: Error: The process '/home/runner/work/_actions/matlab-actions/run-
    tests/v1/dist/bin/run_matlab_command.sh' failed with exit code 1
Post Run actions/checkout@v2                 0s
Complete job                                 0s
```

Expand the "Run all tests > Run command" section to examine the testing diagnostics.

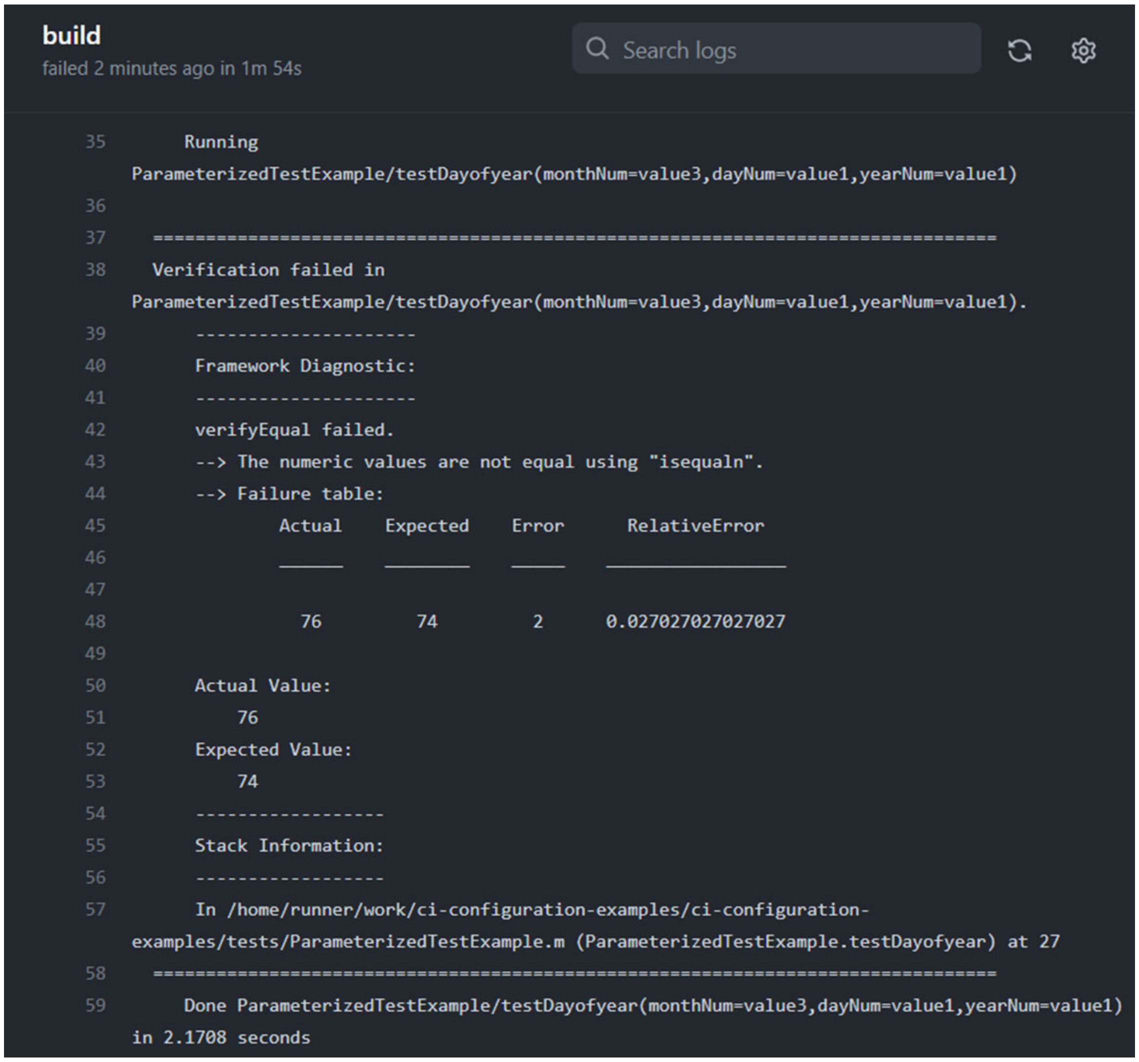


```
▶ Run matlab-actions/run-tests@v1
▶ Generate script
▼ Run command
  /home/runner/work/_actions/matlab-actions/run-tests/v1/dist/bin/run_matlab_command.sh
cd('/tmp/run_matlab_command-oMnNqK'); command_dd69d110_6b42_4e9c_a04e_49446ab24528

        Sponsored License -- for use in support of a program or activity sponsored by
MathWorks.
        Not for government, commercial or other non-sponsored organizational use.

```

As you look through the test diagnostics, you will notice the following failure from a test in `ParameterizedTestExample.m`.


build
failed 2 minutes ago in 1m 54s

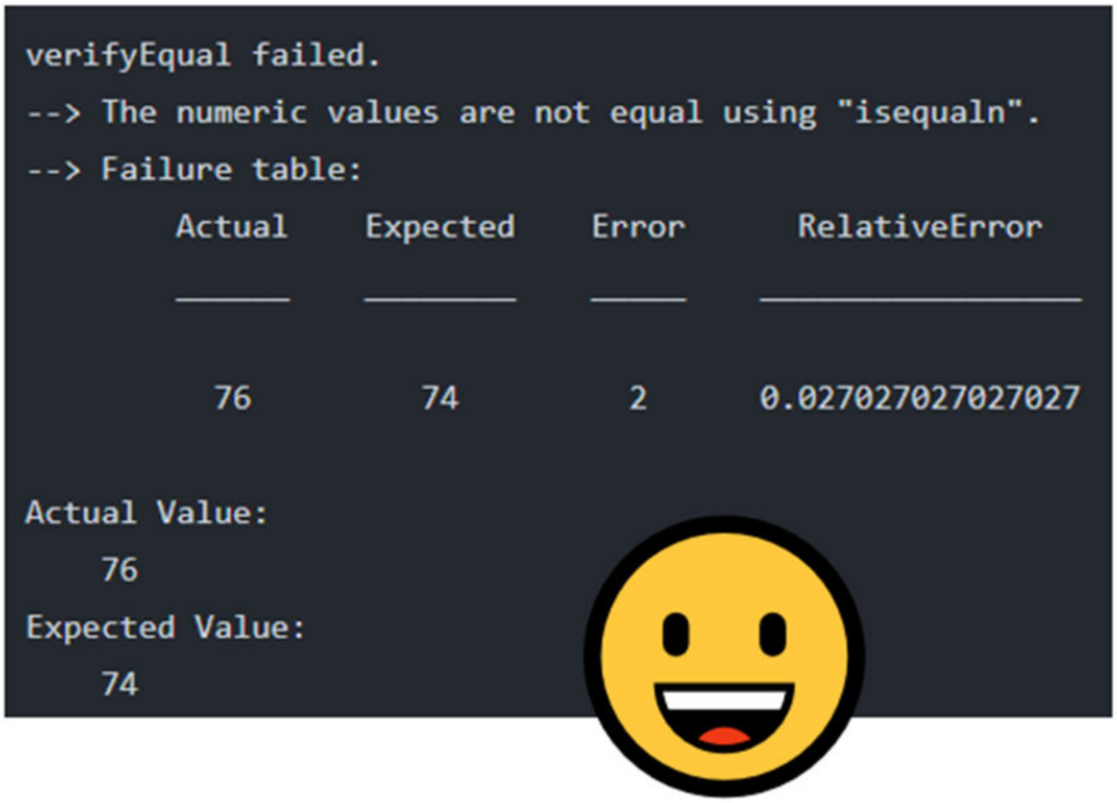
```
    Running
ParameterizedTestExample/testDayofyear(monthNum=value3,dayNum=value1,yearNum=value1)

  ================================================================================
  Verification failed in
ParameterizedTestExample/testDayofyear(monthNum=value3,dayNum=value1,yearNum=value1).
      --------------------
      Framework Diagnostic:
      --------------------
      verifyEqual failed.
      --> The numeric values are not equal using "isequaln".
      --> Failure table:
              Actual    Expected    Error      RelativeError
              ______    ________    _____    _________________

                76         74         2      0.027027027027027

      Actual Value:
          76
      Expected Value:
          74
      ------------------
      Stack Information:
      ------------------
      In /home/runner/work/ci-configuration-examples/ci-configuration-
examples/tests/ParameterizedTestExample.m (ParameterizedTestExample.testDayofyear) at 27
  ================================================================================
    Done ParameterizedTestExample/testDayofyear(monthNum=value3,dayNum=value1,yearNum=value1)
in 2.1708 seconds
```

Looking closer, we notice that our actual answer was larger than our expected answer by 2. That was exactly the number of days we added to February!



```
verifyEqual failed.
--> The numeric values are not equal using "isequaln".
--> Failure table:
        Actual    Expected    Error      RelativeError
        ______    ________    _____    _________________

          76         74         2      0.027027027027027

Actual Value:
    76
Expected Value:
    74
```

GitHub Actions automatically detected the bug we introduced in our code, and we were able to see the problem directly in GitHub!

From now on, let's try not to intentionally introduce bugs, and let's make sure to always run our tests before pushing our code changes!

Wrap-up and homework tasks

Congratulations on completing the Continuous Integration with MATLAB and GitHub Actions Workshop!

During this workshop, you have successfully:

- forked a repository to your GitHub account
- cloned a repository to MATLAB Online
- added and updated files in your repository
- automatically run tests via GitHub Actions

- used GitHub Actions to identify a bug in your repository

For the purpose of the interactive workshop, we will be stopping here, but take a look at the workshop homework below to take your skills even further!

Your workshop homework:

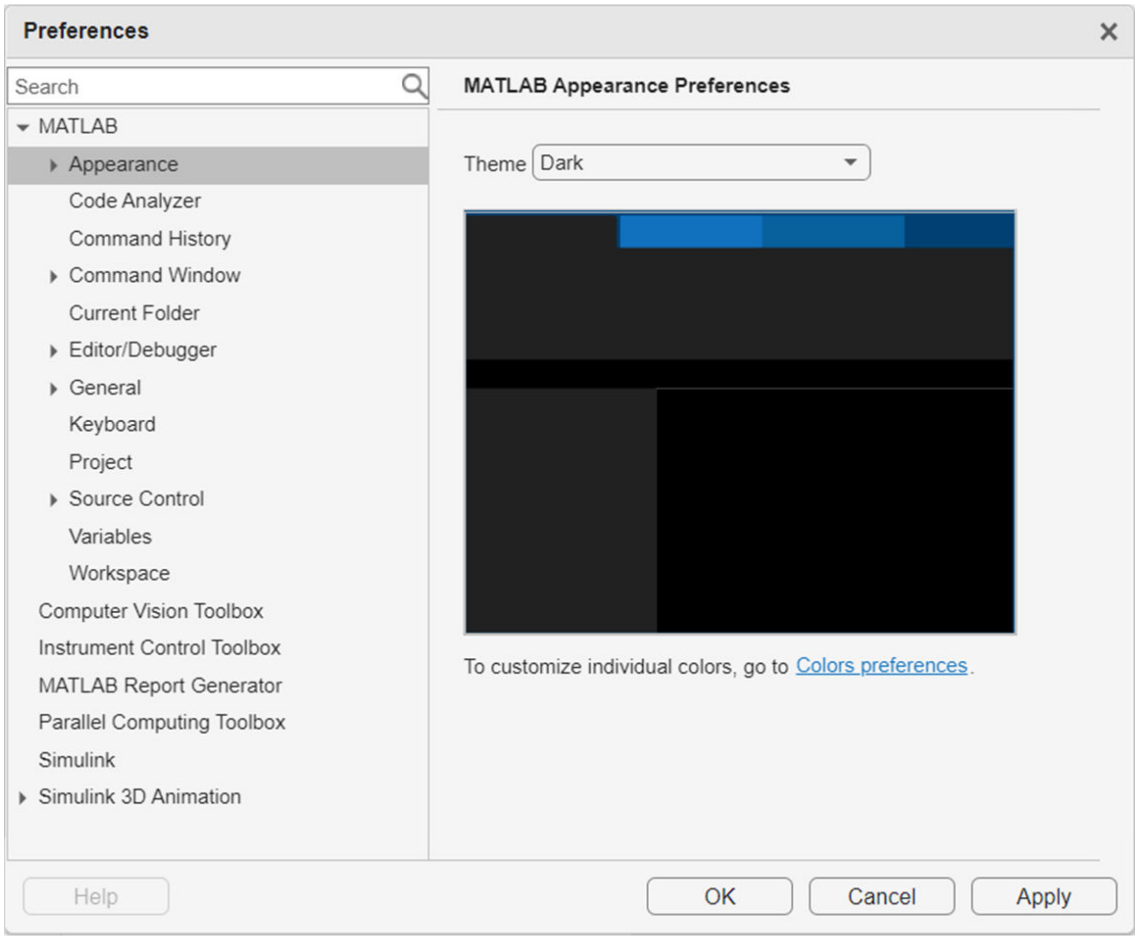
Use everything you've learned today to add new files to your repository, find and fix the bug in one of the files, and get a passing GitHub Actions result and badge.

Please take a look at the **Workshop Homework** document for a walkthrough of the homework tasks.

Bonus: Dark Mode in MATLAB Online

MATLAB Online has Dark Mode, and you can turn it on from:

- Home > Preferences (gear icon in "Environment" section) > MATLAB > Appearance > Theme: Dark

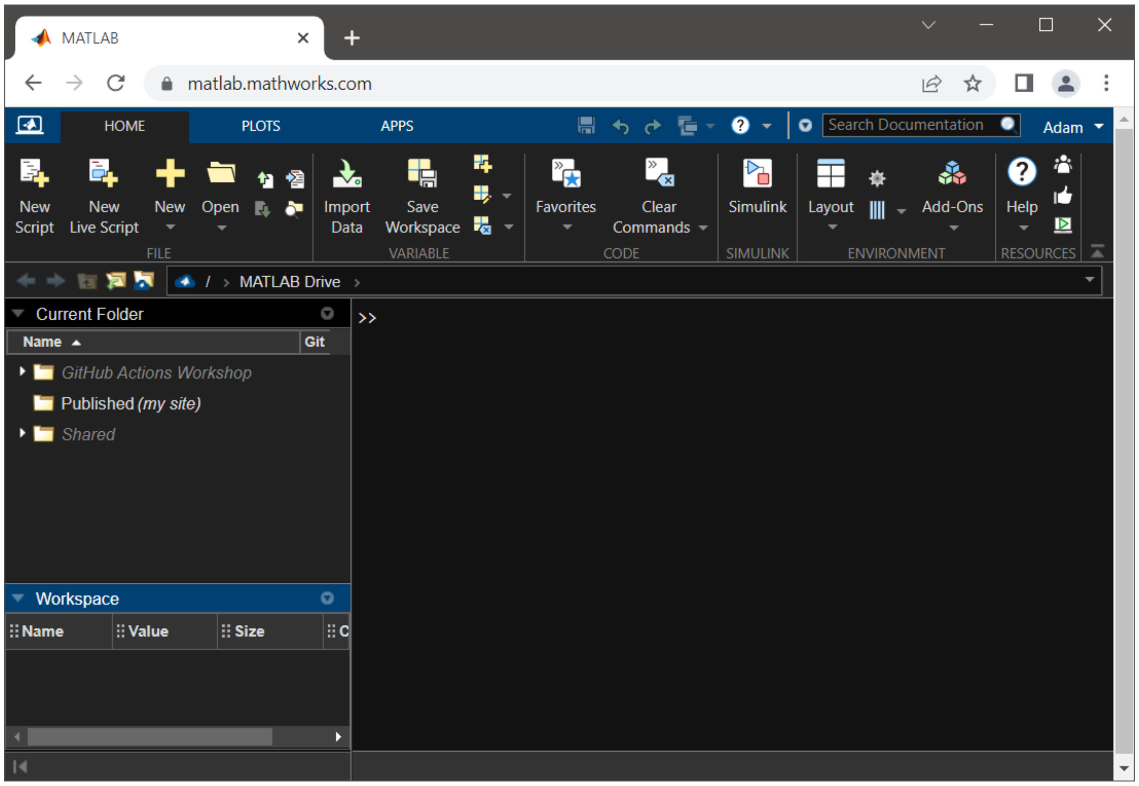


MATLAB Appearance Preferences

Theme: Dark

To customize individual colors, go to Colors preferences.

Here's what MATLAB Online with Dark Mode looks like!



Copyright 2022 The MathWorks, Inc.