Microsoft Power Platform

Microsoft Power Platform

Dev in a day

Lab 05 Application lifecycle management/ May 2022

Table of Contents

Lab Scenario

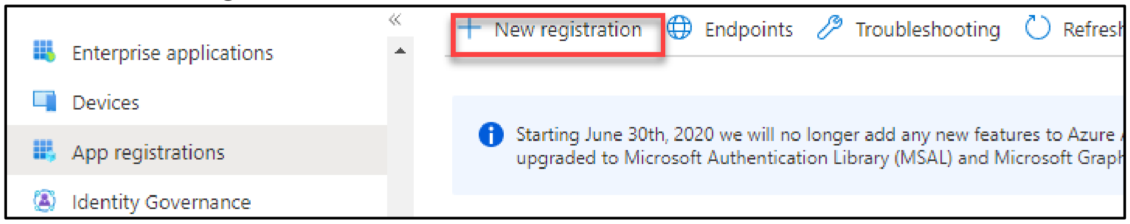
Working as part of the PrioritZ fusion team you will be configuring GitHub Actions using the Power Platform Build Tools to automate the team's deployments.

Exercise 1 – Configure a Service Principal

In this exercise, you will create service principal. The service principal will be used by the workflow actions, so they don't execute under your individual user identity.

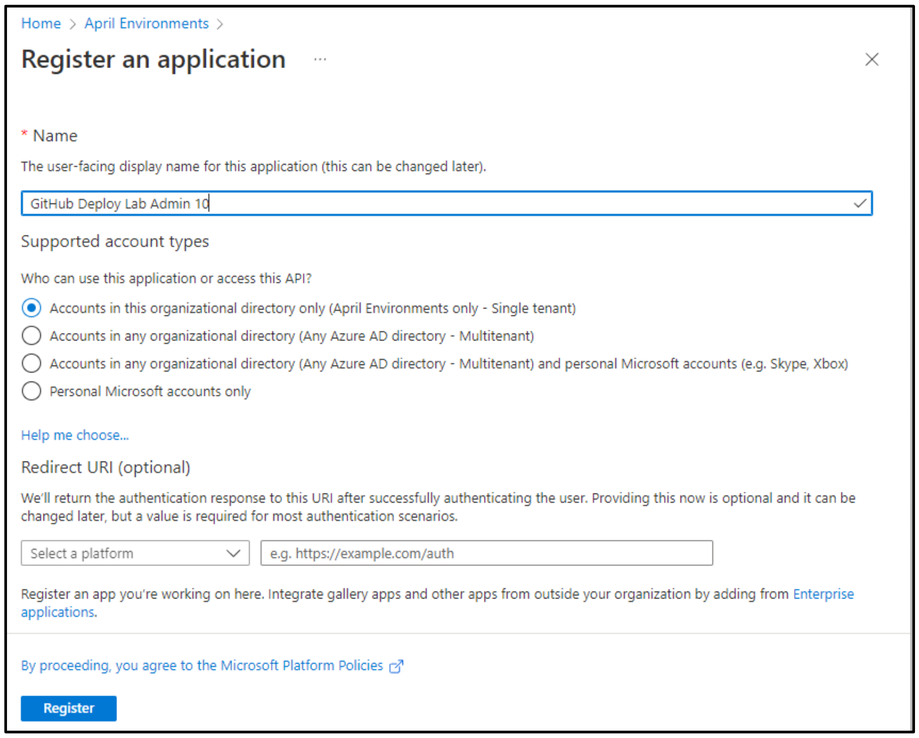
Task 1: Create app registration

1. Navigate to https://portal.azure.com/
2. Select **Azure active directory**.
3. Select **App registrations**.
4. Click **+ New registration**.



Starting June 30th, 2020 we will no longer add any new features to Azure
upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph

5. Enter **GitHub Deploy <Your Name>** e.g., **GitHub Deploy Lab Admin 1** for Name, select **Accounts in this organizational directory only** for Supported account types, and click **Register**.



Home > April Environments >

Register an application ...

* Name

The user-facing display name for this application (this can be changed later).

GitHub Deploy Lab Admin 10

Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only (April Environments only - Single tenant)]
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

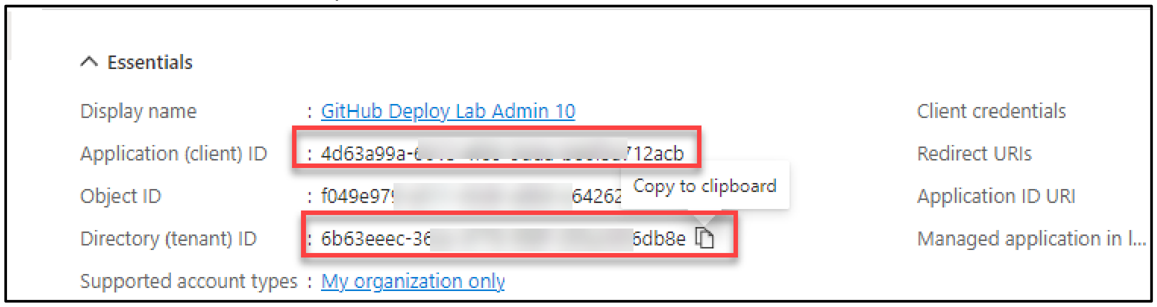
Select a platform | e.g. https://example.com/auth

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

By proceeding, you agree to the Microsoft Platform Policies

Register

6. Copy the **Application (client) ID** and the Directory (tenant) ID. Keep the ids in a notepad, you will need them in future steps.

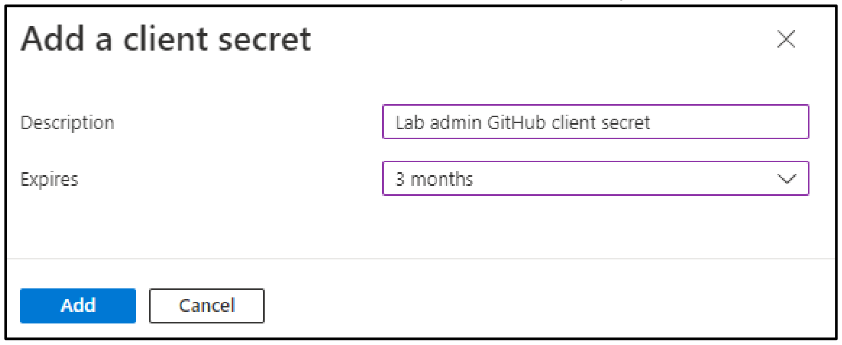


Essentials

Display name	: GitHub Deploy Lab Admin 10	Client credentials
Application (client) ID	: 4d63a99a-[redacted]712acb	Redirect URIs
Object ID	: f049e97[redacted]64262	Application ID URI
Directory (tenant) ID	: 6b63eeec-36[redacted]6db8e	Managed application in l...
Supported account types	: My organization only	

Copy to clipboard

7. Select **Certificates and secrets**.
8. Click **+ New client secret**.
9. Enter **Lab admin GitHub client secret** for Description, select **3 months** for Expires, and click **Add**.



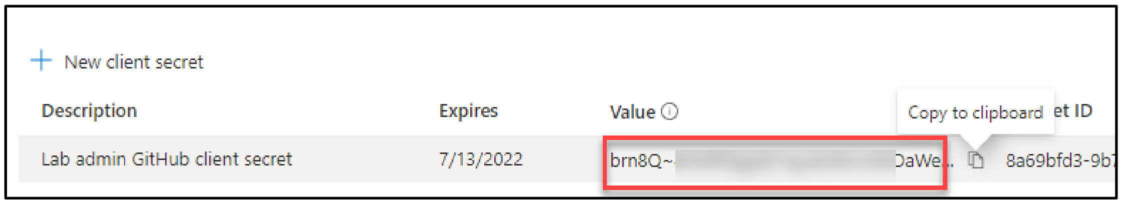
Add a client secret

Description: Lab admin GitHub client secret

Expires: 3 months

Add | Cancel

10. Copy the **Value** and save it on a notepad, you will need this value in future step. The value will not show again after you leave this page.



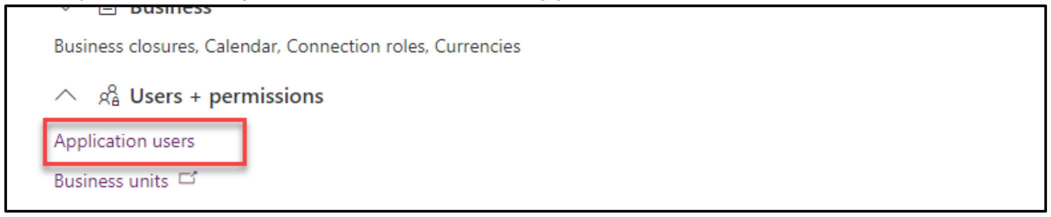
+ New client secret

Description	Expires	Value ⓘ	Copy to clipboard et ID
Lab admin GitHub client secret	7/13/2022	brn8Q~[redacted]DaWe...	8a69bfd3-9b

Task 2: Create app user in Dataverse

In this task, you will be registering the app you created in Azure Active Directory into the dev and test Dataverse environments. You will also be assigning a security role that will allow the service principal to deploy solutions.

1. Navigate to [Power Platform admin center]()
2. Select **Environments**.
3. Select your **Dev** environment and click **Settings**.
4. Expand **Users + permissions** and select **Application users**.



Business

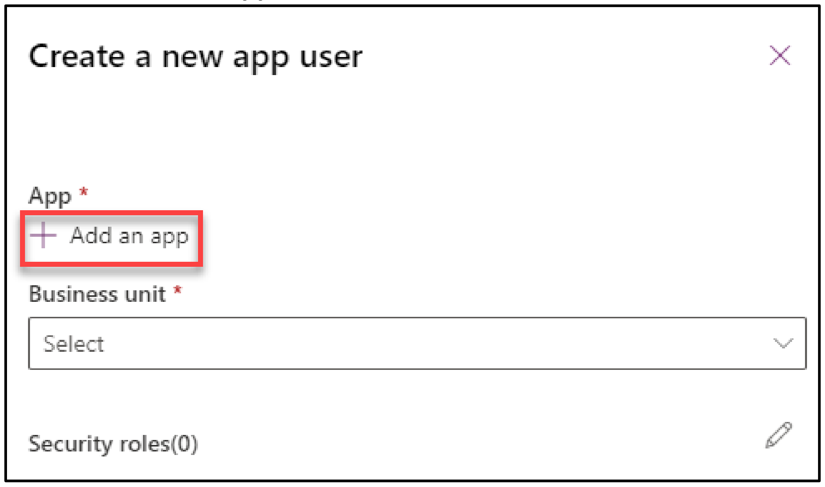
Business closures, Calendar, Connection roles, Currencies

Users + permissions

Application users

Business units

5. Click **+ New app user**.
6. Click **+ Add an app**.



Create a new app user

App *

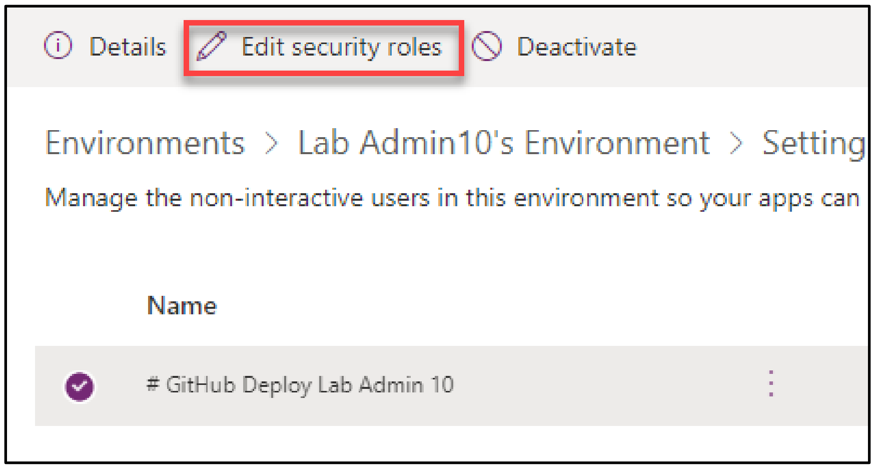
+ Add an app

Business unit *

Select

Security roles(0)

7. Select application registration you created and click **Add**.
8. Select your Business unit and click **Create**. There should only be one unless you created more.
9. Select the application user you just created and click **Edit security roles**.

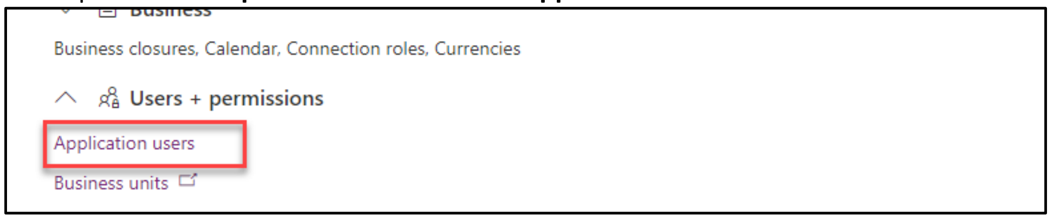


Environments > Lab Admin10's Environment > Setting

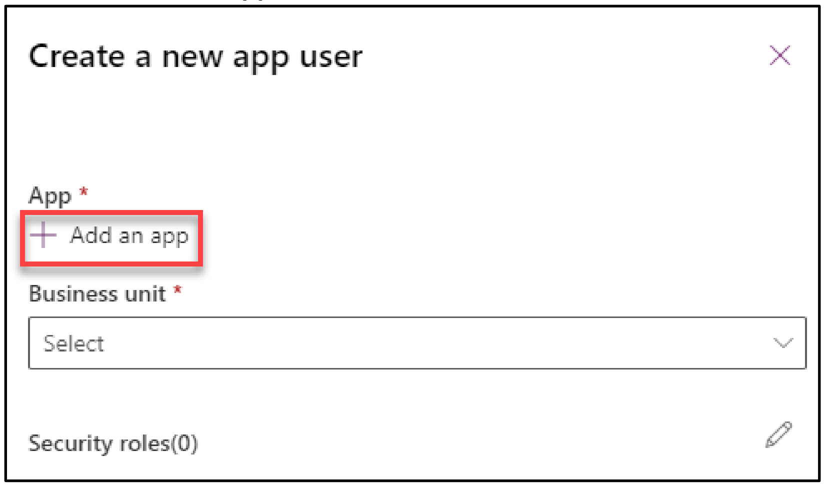
Manage the non-interactive users in this environment so your apps can

Name
GitHub Deploy Lab Admin 10

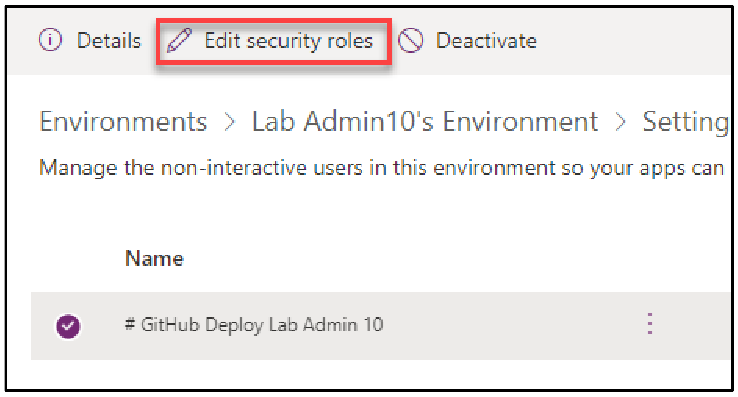
10. Select **System administrator** and click **Save**.
11. Select **Environments** again.
12. Select your **Test** environment and click **Settings**.
13. Expand **Users + permissions** and select **Application users**.



14. Click **+ New app user**.
15. Click **+ Add an app**.



16. Select application registration you created and click **Add**.
17. Select your Business unit and click **Create**.
18. Select the application user you just created and click **Edit security roles**.



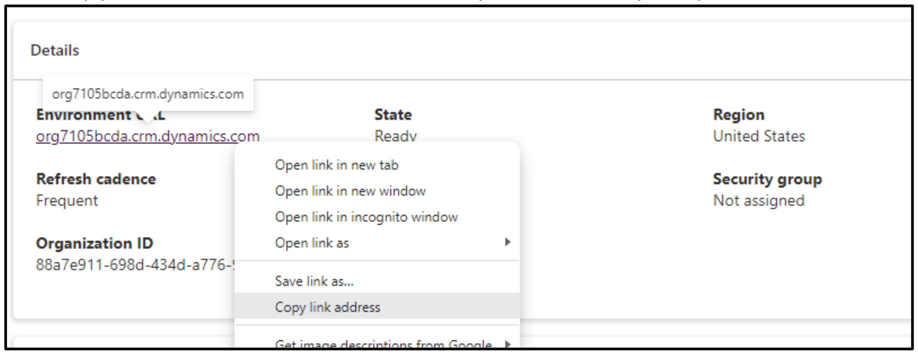
Environments > Lab Admin10's Environment > Setting

Manage the non-interactive users in this environment so your apps can

Name

GitHub Deploy Lab Admin 10

19. Select **System administrator** and click **Save**.
20. Select **Environments** and click to open the **Dev** environment.
21. Copy the **Environment URL** and keep it on a notepad, you will use this URL in future steps.



Details

org7105bcda.crm.dynamics.com

Environment URL	State	Region
org7105bcda.crm.dynamics.com	Ready	United States
Refresh cadence		Security group
Frequent		Not assigned
Organization ID		
88a7e911-698d-434d-a776-		

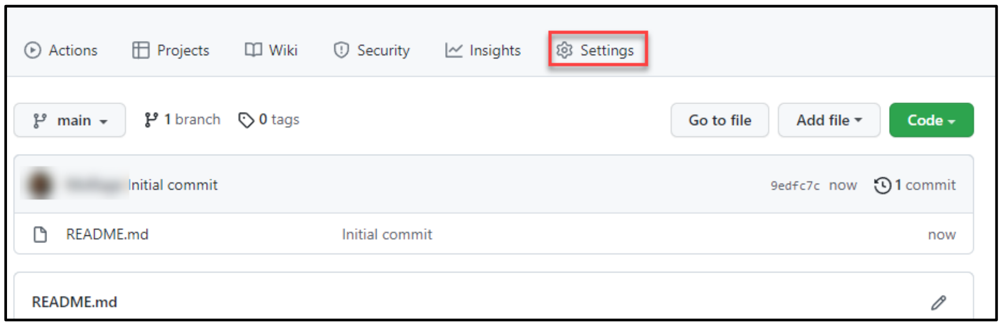
22. Select **Environments** and click to open the **Test** environment.
23. Copy the **Environment URL** and keep it on a notepad, you will this URL in future steps.

Exercise 2 – Create GitHub Repo

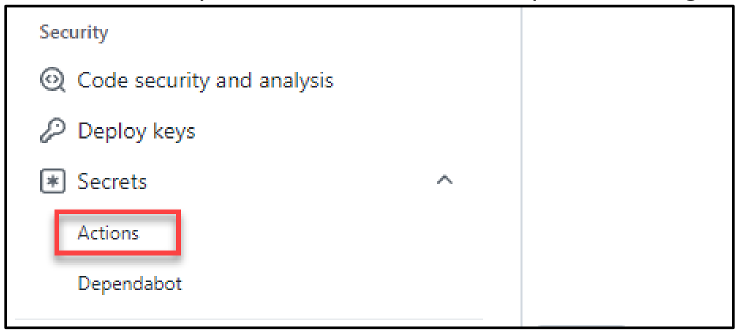
In this exercise, you will create a GitHub repository and add repository secrets.

Task 1: Create repository

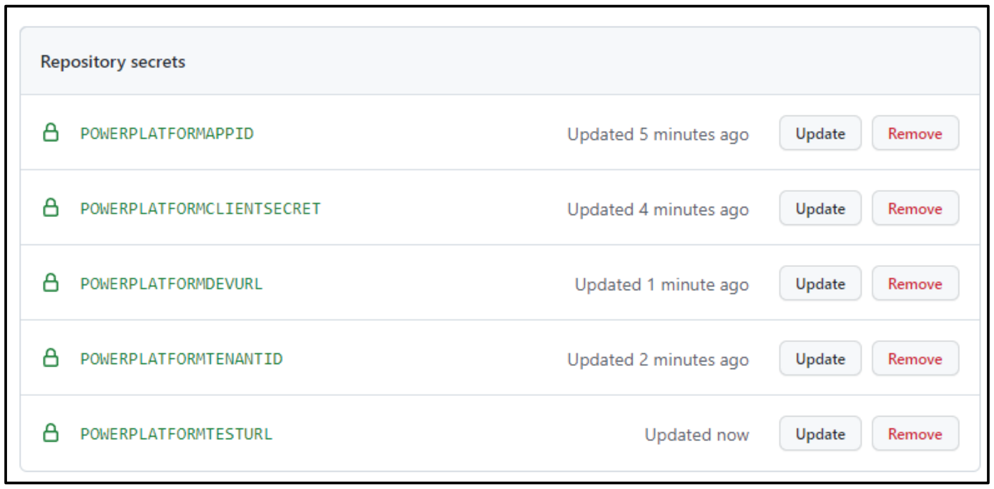
1. Navigate to https://github.com/
2. Sign in or signup for GitHub.
3. Click **New repository**.
4. Enter **PrioritZ** for Repository name, select **Public**, check the **Add a README file**, and click **Create repository**.
5. Click **Settings**.



6. Go to the **Security** section, expand **Secrets** and select **Actions**. The values you provide will not be visible after you create the item so take your time to get the values correct.



7. Click **New repository secret**.
8. Enter **PowerPlatformAppID** for Name and paste the **Application (client) ID** from your notepad in the **Value** field and click **Add secret**.
9. Click **New repository secret** again.
10. Enter **PowerPlatformClientSecret** for Name and paste the secret **Value** from your notepad in the **Value** field and click **Add secret**.
11. Click **New repository secret** again.
12. Enter **PowerPlatformTenantID** for Name and paste the secret **Tenant ID** from your notepad in the **Value** field and click **Add secret**.
13. Click **New repository secret** again.
14. Enter **PowerPlatformDevUrl** for Name and paste the secret **Dev environment URL** from your notepad in the **Value** field and click **Add secret**.
15. Click **New repository secret** one more time.
16. Enter **PowerPlatformTestUrl** for Name and paste the secret **Test environment URL** from your notepad in the **Value** field and click **Add secret**.
17. You should now have **5** repository secrets.


Repository secrets

POWERPLATFORMAPPID	Updated 5 minutes ago	Update	Remove
POWERPLATFORMCLIENTSECRET	Updated 4 minutes ago	Update	Remove
POWERPLATFORMDEVURL	Updated 1 minute ago	Update	Remove
POWERPLATFORMTENANTID	Updated 2 minutes ago	Update	Remove
POWERPLATFORMTESTURL	Updated now	Update	Remove

18. Do not navigate away from this page.

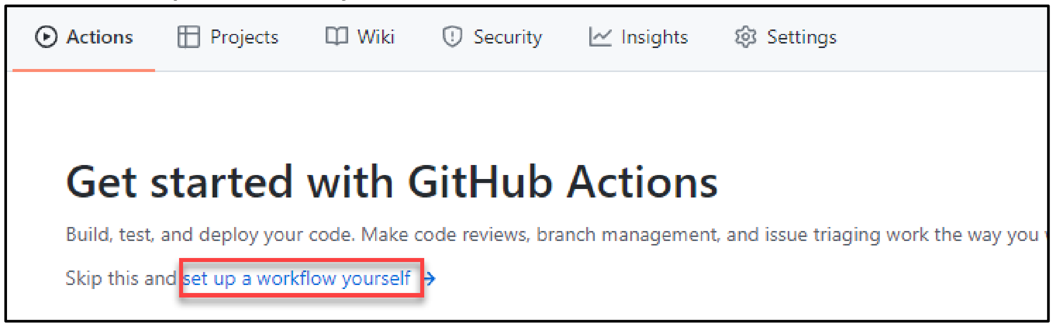
Exercise 3 – Export and Branch

In this exercise, you will setup a workflow action, and add steps that will export the solution from dev environment and create a new branch.

Task 1: Export and branch

In this task you will create the workflow definition using the YAML provided. The action YAML uses two space indentation so follow that carefully as you build the workflow definition. If in doubt, review the indentation shown in the images.

1. Select the **Actions** tab.
2. Click **set up a workflow yourself**.

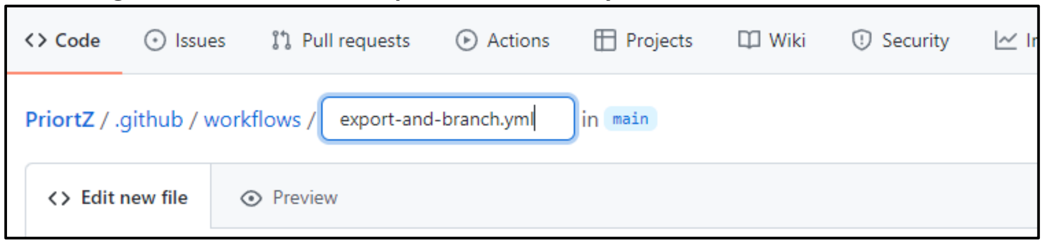


Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you

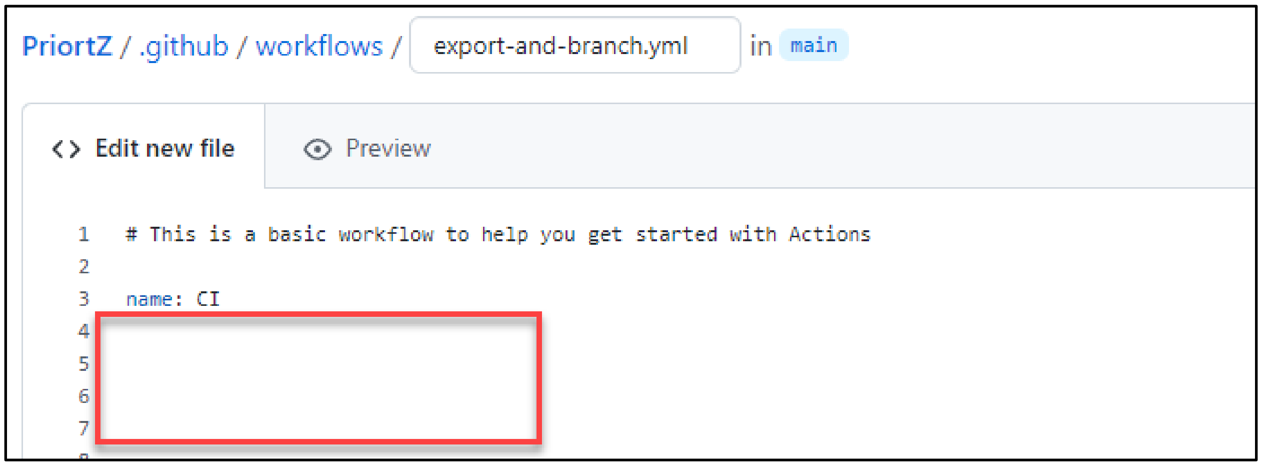
Skip this and set up a workflow yourself →

3. Change the file name to it **export-and-branch.yml**.



PriortZ / .github / workflows / export-and-branch.yml in main

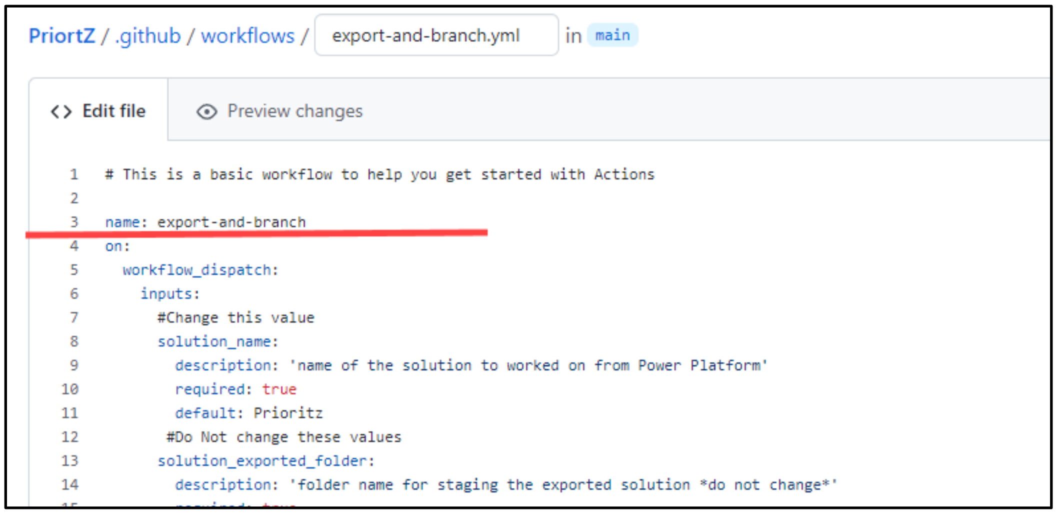
4. Remove everything below the name line.


PriortZ / .github / workflows / export-and-branch.yml in main

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

name: CI
```

5. Change the name to **export-and-branch**
6. Add the below YAML snippet after the name line. This defines the action trigger and some input parameters that could be changed when manually running the action.

```
on:
  workflow_dispatch:
    inputs:
      #Change this value
      solution_name:
        description: 'name of the solution to worked on from Power Platform'
        required: true
        default: Prioritz
       #Do Not change these values
      solution_exported_folder:
        description: 'folder name for staging the exported solution *do not
change*'
        required: true
        default: out/exported/
      solution_folder:
        description: 'staging the unpacked solution folder before check-in *do
not change*'
        required: true
        default: out/solutions/
      solution_target_folder:
       description: 'folder name to be created and checked in *do not change*'
       required: true
       default: solutions/
```


PriortZ / .github / workflows / export-and-branch.yml in main

Edit file | Preview changes

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

name: export-and-branch
on:
  workflow_dispatch:
    inputs:
      #Change this value
      solution_name:
        description: 'name of the solution to worked on from Power Platform'
        required: true
        default: Prioritz
       #Do Not change these values
      solution_exported_folder:
        description: 'folder name for staging the exported solution *do not change*'
```

7. Setup the workflow. Add below YAML snippet after the last snippet. This sets up the jobs and identifies the first job as export-from-dev. This also defines the steps with the first one checking out the current main branch content.

```
jobs:
  export-from-dev:
    runs-on: windows-latest
    steps:
    - uses: actions/checkout@v2
      with:
        lfs: true
```

8. Next you will export both an unmanaged and managed solution file from your dev environment.
9. Export the unmanaged solution. Add below snippet after the last snippet.

```
    - name: export-solution action
      uses: microsoft/powerplatform-actions/export-solution@0.4.0
      with:
        environment-url: ${{secrets.PowerPlatformDevUrl}}
        app-id: ${{secrets.PowerPlatformAppID}}
        client-secret: ${{ secrets.PowerPlatformClientSecret }}
        tenant-id: ${{secrets.PowerPlatformTenantID}}
        solution-name: ${{ github.event.inputs.solution_name }}
        solution-output-file: ${{
github.event.inputs.solution_exported_folder}}/${{
github.event.inputs.solution_name }}.zip
```

10. Export the managed solution. Add below snippet after the last snippet.

```
    - name: export-managed-solution action
```

```
      uses: microsoft/powerplatform-actions/export-solution@0.4.0
      with:
        environment-url: ${{secrets.PowerPlatformDevUrl}}
        app-id: ${{secrets.PowerPlatformAppID}}
        client-secret: ${{ secrets.PowerPlatformClientSecret }}
        tenant-id: ${{secrets.PowerPlatformTenantID}}
        solution-name: ${{ github.event.inputs.solution_name }}
        solution-output-file: ${{
github.event.inputs.solution_exported_folder}}/${{
github.event.inputs.solution_name }}_managed.zip
        managed: true
```

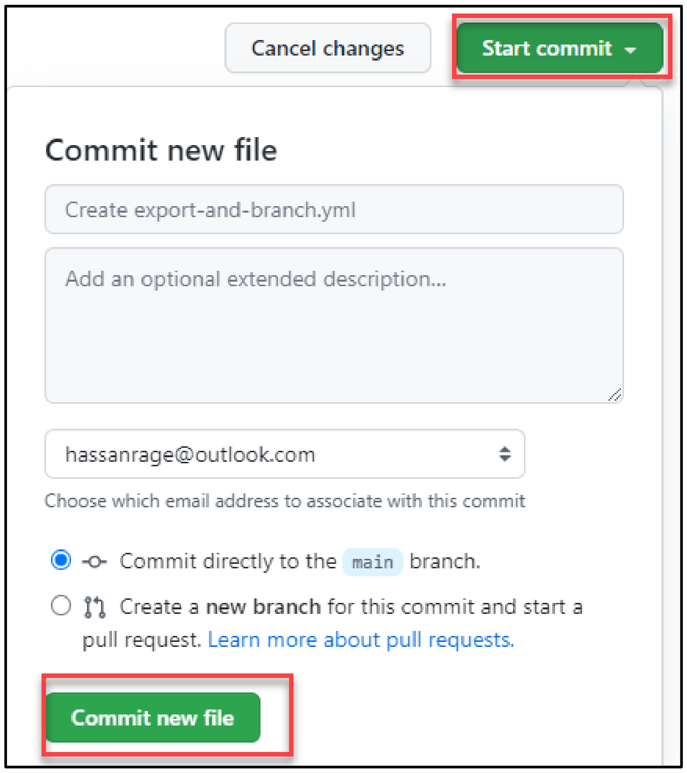
11. The solution files are compressed files and don't version control well. Using unpack you will expand the solution files into a set of files that can be easily checked into the repo.
12. Unpack solution file. Add below snippet after the last snippet.

```
    - name: unpack-solution action
      uses: microsoft/powerplatform-actions/unpack-solution@0.4.0
      with:
        solution-file: ${{ github.event.inputs.solution_exported_folder}}/${{
github.event.inputs.solution_name }}.zip
        solution-folder: ${{ github.event.inputs.solution_folder}}/${{
github.event.inputs.solution_name }}
        solution-type: 'Both'
```

13. Branch and prepare for pull. Add below snippet after the last snippet.

```
    - name: branch-solution, prepare it for a PullRequest
      uses: microsoft/powerplatform-actions/branch-solution@v0
      with:
        solution-folder: ${{ github.event.inputs.solution_folder}}/${{
github.event.inputs.solution_name }}
        solution-target-folder: ${{
github.event.inputs.solution_target_folder}}/${{
github.event.inputs.solution_name }}
        repo-token: ${{ secrets.GITHUB_TOKEN }}
        allow-empty-commit: true
```

14. Click **Start commit** and then click **Commit new file**.



Commit new file

Create export-and-branch.yml

Add an optional extended description...

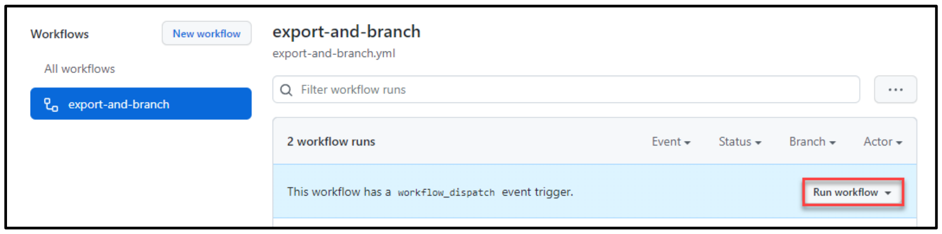
hassanrage@outlook.com

Choose which email address to associate with this commit

[✓ Commit directly to the `main` branch.]

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

15. Select the **Actions** tab and select the workflow you created.
16. Click **Run workflow.**



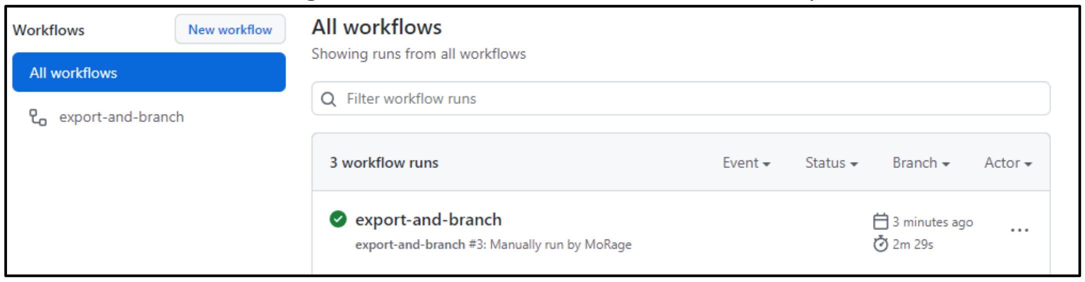
export-and-branch
export-and-branch.yml

Filter workflow runs

2 workflow runs

This workflow has a `workflow_dispatch` event trigger.

17. Click **Run workflow** again and wait for the workflow run to complete.



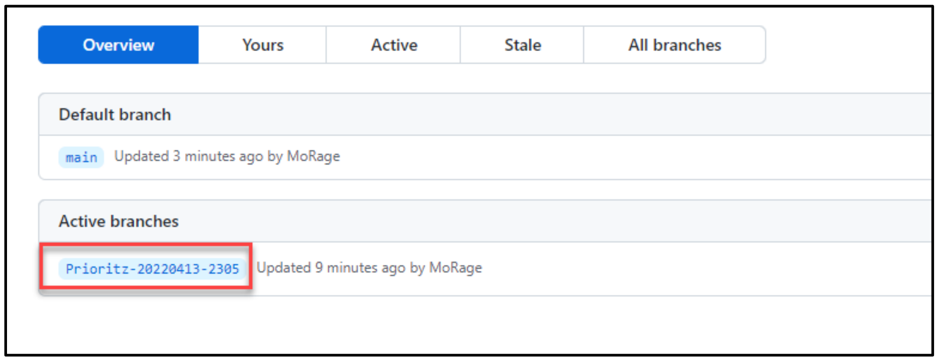
All workflows
Showing runs from all workflows

Filter workflow runs

3 workflow runs

export-and-branch
export-and-branch #3: Manually run by MoRage
3 minutes ago
2m 29s

18. Select the **Code** tab.
19. Select **Branches**. You should see two branches.
20. Click to open the branch that was created by the workflow action.



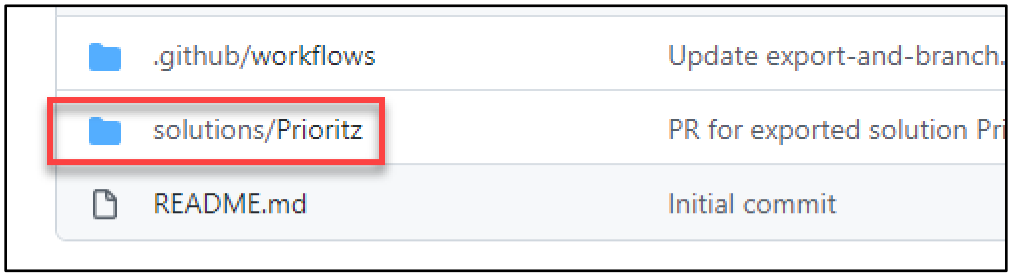
Default branch

main Updated 3 minutes ago by MoRage

Active branches

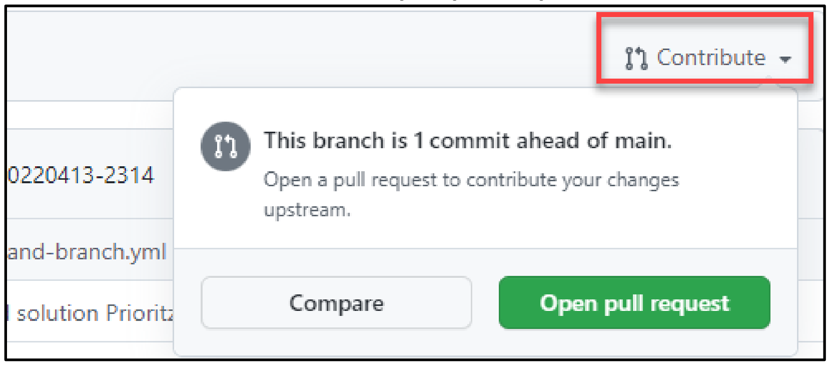
Prioritz-20220413-2305 Updated 9 minutes ago by MoRage

21. You should see solution folder.



.github/workflows	Update export-and-branch.
solutions/Prioritz	PR for exported solution Pri
README.md	Initial commit

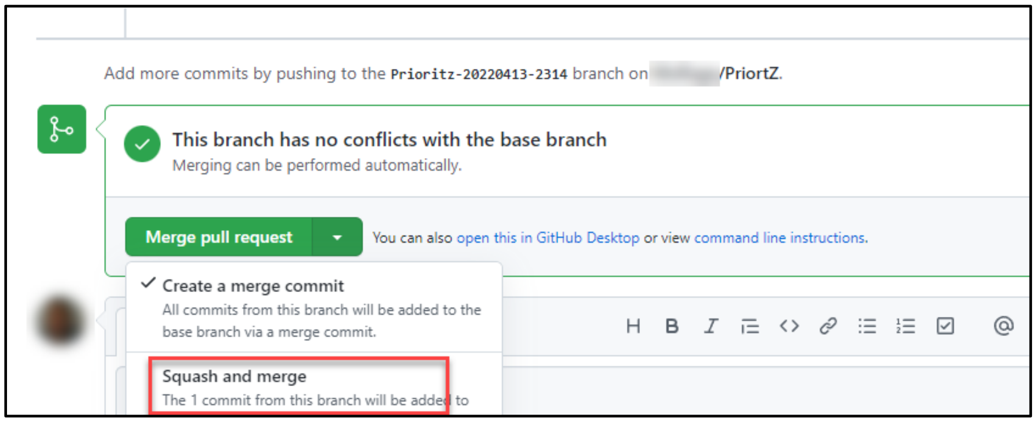
22. Click **Contribute** and select **Open pull request**.



This branch is 1 commit ahead of main.

Open a pull request to contribute your changes upstream.

23. Add description if you like and then click **Create pull request**.
24. You should now see the pull request summary. Confirm that the branch has no conflicts with the main branch and that the changes can be merged into the main branch automatically.
25. Click on the chevron button next to the **Merge pull request** button and select **Squash and merge**.


Add more commits by pushing to the `Prioritz-20220413-2314` branch on [redacted]/PriortZ.

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

Merge pull request
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 1 commit from this branch will be added to

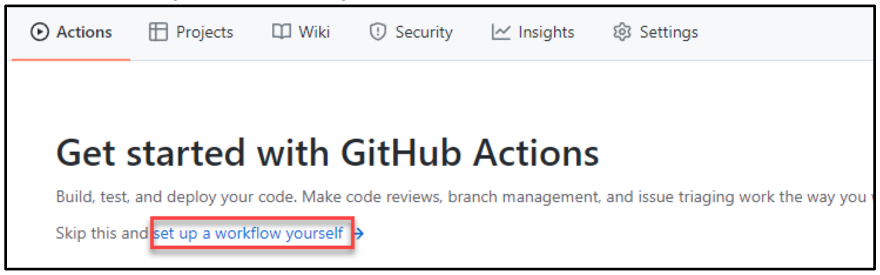
26. Click **Squash and merge**.
27. Click **Confirm squash and merge**.
28. The pull request should get merged successfully.
29. Do not navigate away from this page.

Exercise 4 – Release to Test

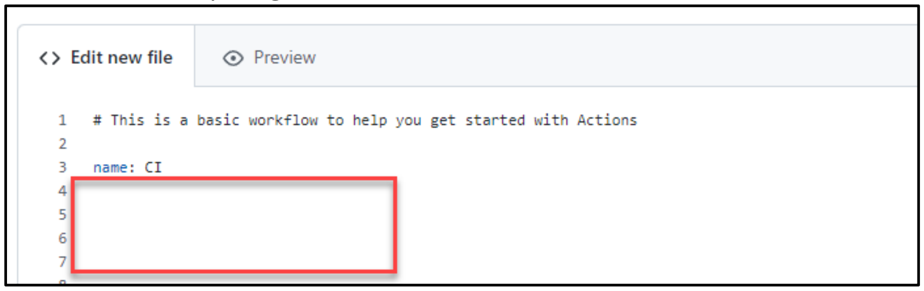
In this exercise, you will create a workflow action and add steps that will release the solution you exported to the test environment.

Task 1: Create workflow

1. Select the **Actions** tab.
2. Click **New workflow**.
3. Click **set up a workflow yourself**.


Actions
Projects
Wiki
Security
Insights
Settings
Get started with GitHub Actions
Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you
Skip this and set up a workflow yourself

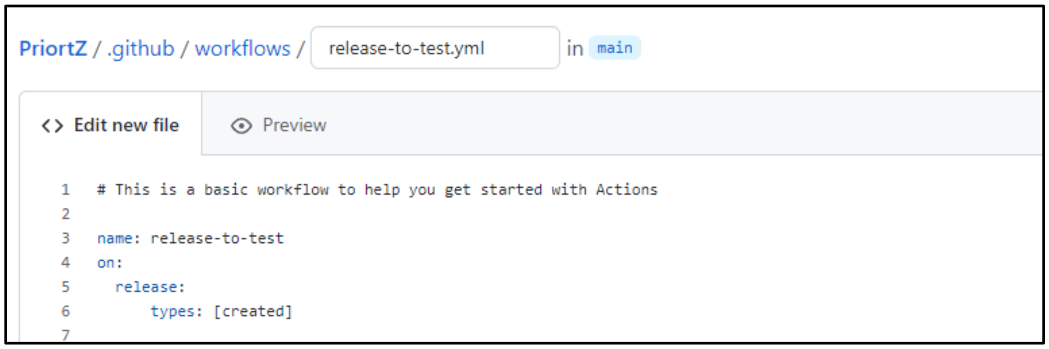
4. Change the file name to it **release-to-test.yml**.
5. Remove everything below the name line.



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

name: CI
```

6. Change the name to **release-to-test**
7. Add the following trigger. This will trigger on creation of a new release.

```
on:
  release:
      types: [created]
```



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

name: release-to-test
on:
  release:
      types: [created]
```

8. Define constants. Add the below YAML snippet.

```
env:
  solution_name: Prioritz
  solution_source_folder: solutions
  solution_outbound_folder: out/solutions
  solution_release_folder: out/release
```

9. Add job and steps. Add the below YAML snippet.

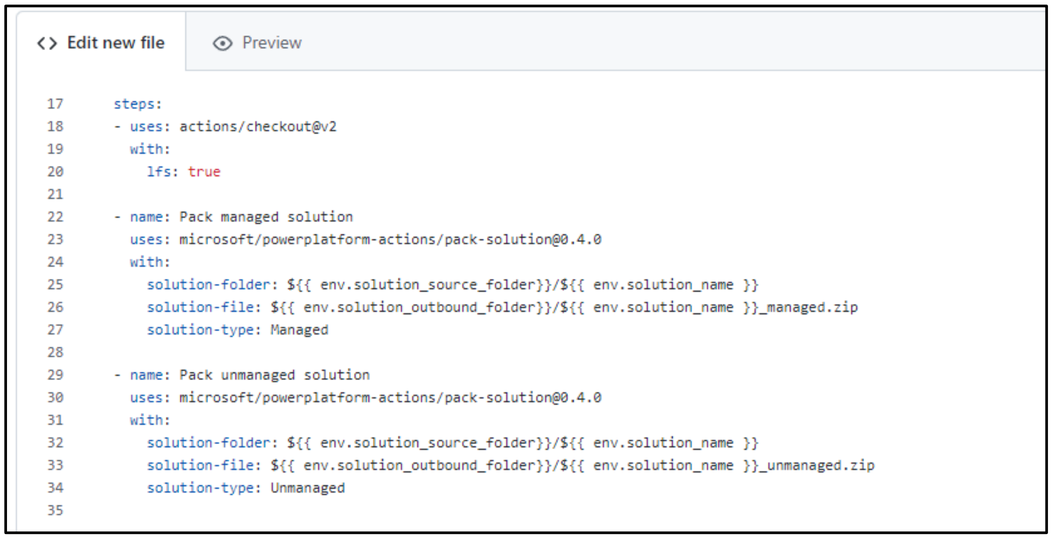
```
jobs:
  convert-to-managed:
    runs-on: windows-latest


    steps:
    - uses: actions/checkout@v2
      with:
        lfs: true
```

10. Package managed solution. Add the below YAML snippet. This will take the individual files and put them in a compressed file that can be deployed.

```
    - name: Pack managed solution
      uses: microsoft/powerplatform-actions/pack-solution@0.4.0
      with:
        solution-folder: ${{ env.solution_source_folder}}/${{ env.solution_name
}}
        solution-file: ${{ env.solution_outbound_folder}}/${{ env.solution_name
}}_managed.zip
        solution-type: Managed
```

11. Package unmanaged solution. Add the below YAML snippet.

```
    - name: Pack unmanaged solution
      uses: microsoft/powerplatform-actions/pack-solution@0.4.0
      with:
        solution-folder: ${{ env.solution_source_folder}}/${{ env.solution_name
}}
        solution-file: ${{ env.solution_outbound_folder}}/${{ env.solution_name
}}_unmanaged.zip
        solution-type: Unmanaged
```



```
    steps:
    - uses: actions/checkout@v2
      with:
        lfs: true

    - name: Pack managed solution
      uses: microsoft/powerplatform-actions/pack-solution@0.4.0
      with:
        solution-folder: ${{ env.solution_source_folder}}/${{ env.solution_name }}
        solution-file: ${{ env.solution_outbound_folder}}/${{ env.solution_name }}_managed.zip
        solution-type: Managed

    - name: Pack unmanaged solution
      uses: microsoft/powerplatform-actions/pack-solution@0.4.0
      with:
        solution-folder: ${{ env.solution_source_folder}}/${{ env.solution_name }}
        solution-file: ${{ env.solution_outbound_folder}}/${{ env.solution_name }}_unmanaged.zip
        solution-type: Unmanaged
```

12. Upload solution artifacts. Add the below YAML snippet.

```
    - name: Upload the unmanaged solution to GH artifact store
      uses: actions/upload-artifact@v2
      with:
        name: unmanagedSolutions
        path: ${{ env.solution_outbound_folder}}/${{ env.solution_name
}}_unmanaged.zip

    - name: Upload the managed solution to GH artifact store
      uses: actions/upload-artifact@v2
      with:
        name: managedSolutions
        path: ${{ env.solution_outbound_folder}}/${{ env.solution_name
}}_managed.zip
```

13. Release to staging. Add the below YAML snippet. This defines a second job to deploy.

```
  release-to-staging:
    needs: [ convert-to-managed ]
    runs-on: windows-latest

    steps:
    - uses: actions/checkout@v2
      with:
        lfs: true
```

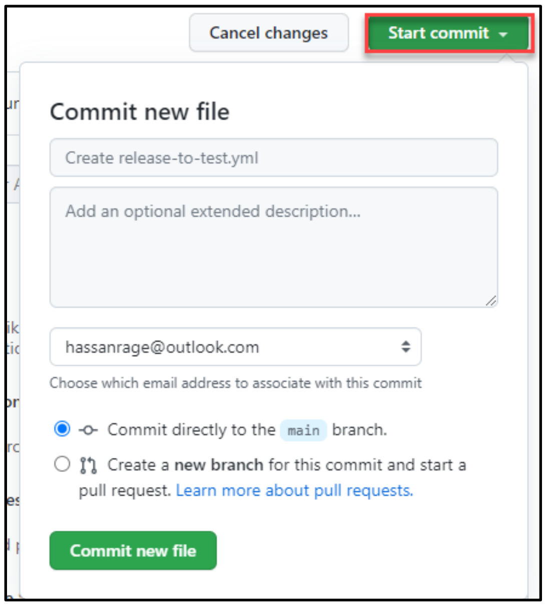
14. Download artifacts. Add the below YAML snippet.

```
    - name: Fetch the ready to ship solution from GH artifact store
      uses: actions/download-artifact@v2
      with:
        name: managedSolutions
        path: ${{ env.solution_release_folder}}
```

15. Import the managed solution to the test environment. Add the below YANL snippet.

```
    - name: Import solution to prod env
      uses: microsoft/powerplatform-actions/import-solution@0.4.0
      with:
        environment-url: ${{secrets.PowerPlatformTestUrl}}
        app-id: ${{secrets.PowerPlatformAppID}}
        client-secret: ${{ secrets.PowerPlatformClientSecret }}
        tenant-id: ${{secrets.PowerPlatformTenantID}}
        solution-file: ${{ env.solution_release_folder}}/${{ env.solution_name
}}_managed.zip
        run-asynchronously: true
```

16. Click **Start commit** and then click **Commit new file**.


Cancel changes

Start commit

Commit new file

Create release-to-test.yml

Add an optional extended description...

hassanrage@outlook.com

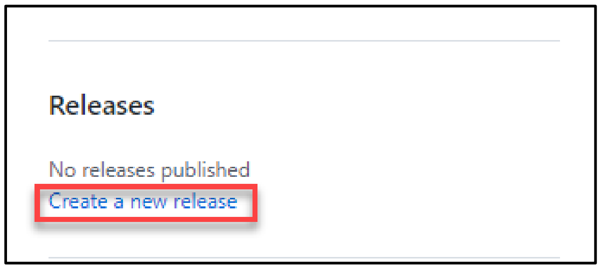
Choose which email address to associate with this commit

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

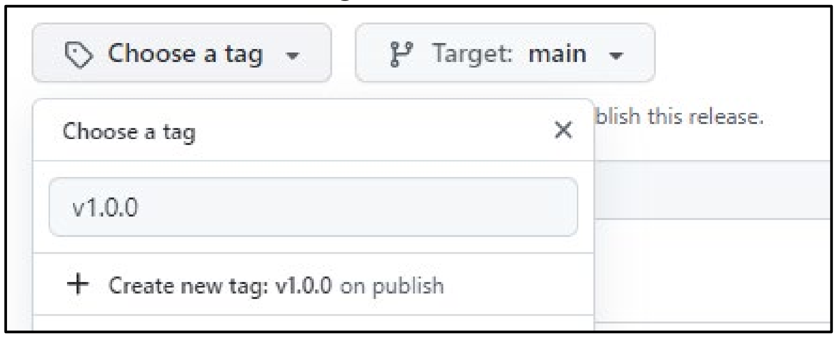
17. Select the **Code** tab.
18. Go to the **Releases** section and click **Create new release**.


Releases

No releases published

Create a new release

19. Click on the **Choose a tag** button, enter **v1.0.0**, and select **+ Create new tag on publish**.


Choose a tag

Target: main

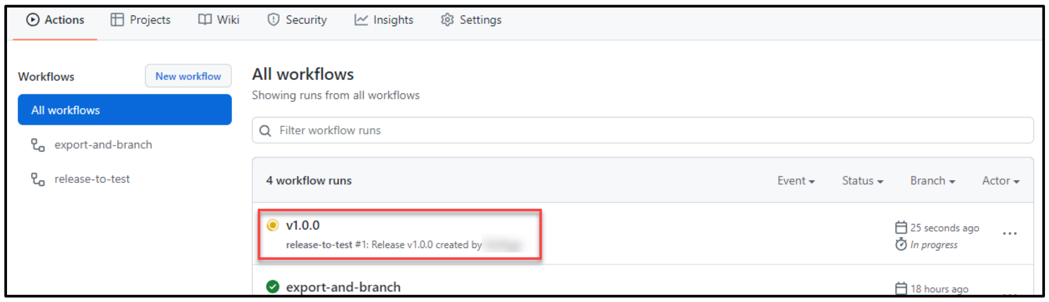
Choose a tag

v1.0.0

+ Create new tag: v1.0.0 on publish

blish this release.

20. Click **Publish release**.
21. Select the **Actions** tab and monitor the workflow.



All workflows
Showing runs from all workflows

4 workflow runs

v1.0.0
release-to-test #1: Release v1.0.0 created by [redacted]
25 seconds ago
In progress

export-and-branch
18 hours ago

22. The release should complete successfully.
23. Check your test environment and you should see the solution deployed.