Table of Contents

[Steps to setup Self Hosted Runner in Linux: 2](#_Toc142921030)

[Downloading/cloning MigrationScript repository 9](#_Toc142921031)

[SINGLE REPO MIGRATION USING MIGRATION SCRIPT 17](#_Toc142921032)

[Sequence of Events for Migration using Runner 20](#_Toc142921033)

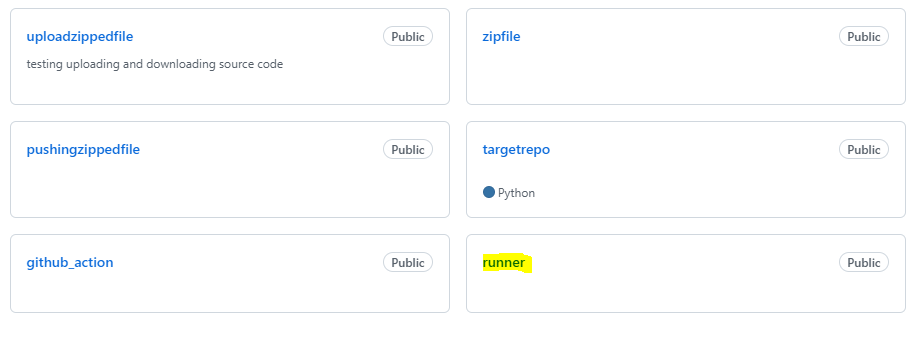
[Deleting an issue 24](#_Toc142921034)

[LOCKING AND UNLOCKING/DETING SOURCE REPOSITORIES AFTER PRODUCTION MIGRATION 28](#_Toc142921035)

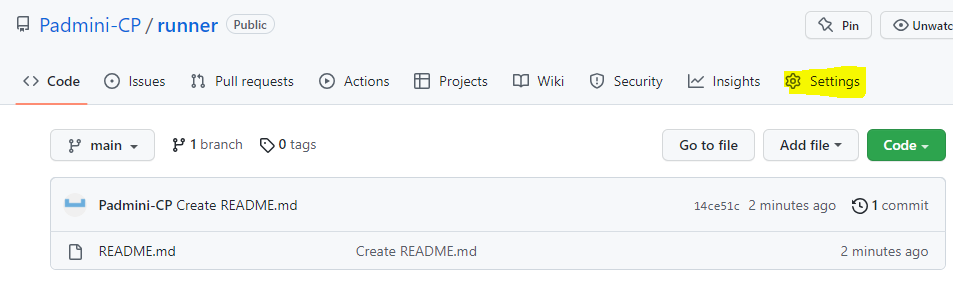
[User Mapping 32](#_Toc142921036)

# Steps to setup Self Hosted Runner in Linux:

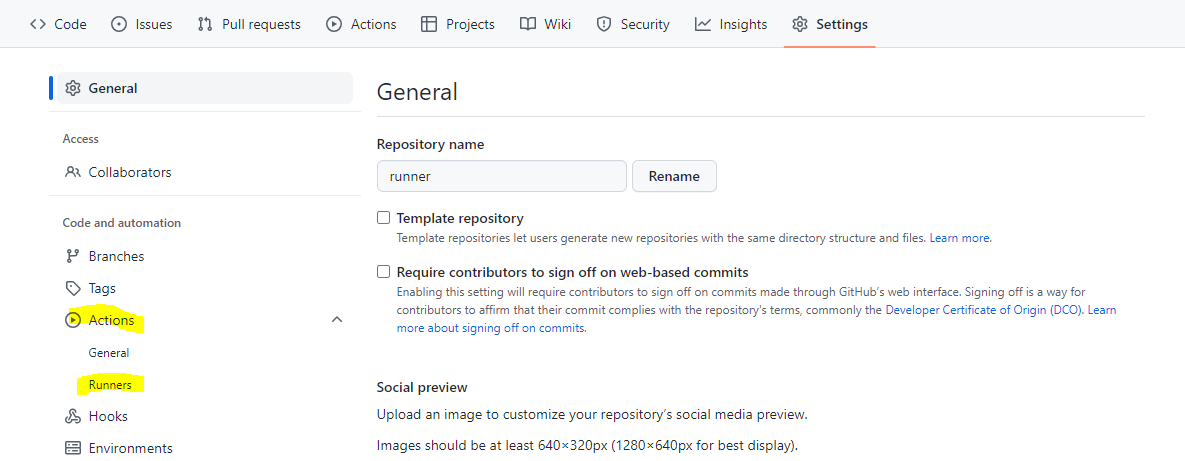
**Step 1:** Login to <https://githubdev.gehealthcare.com> and click on existing repository (for e.g., Runner repo) where self-hosted runner needs to be set up



1.1 Under your repository name, click Settings.

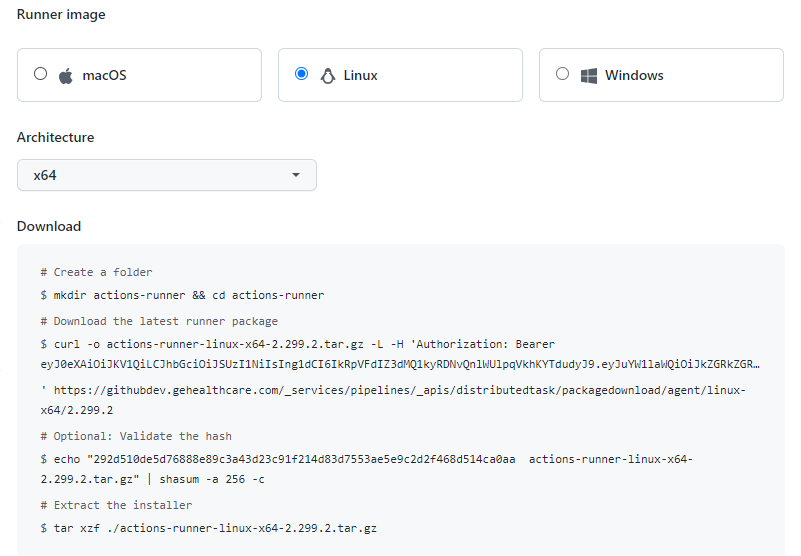


1.2 In the left sidebar, click **Actions**, then click **Runners** then click on **New self-hosted runner button**.





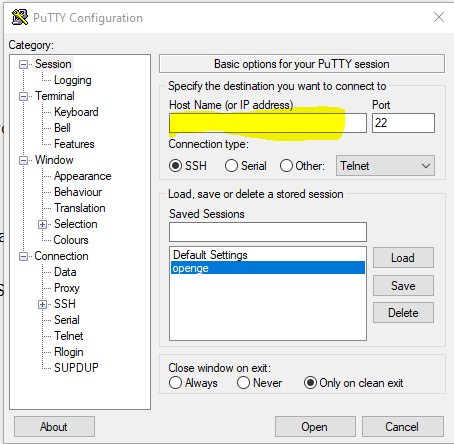
**Step 2:** Select Linux and follow the given instructions to download the runner application.



**Step 3:** Login to Linux server:

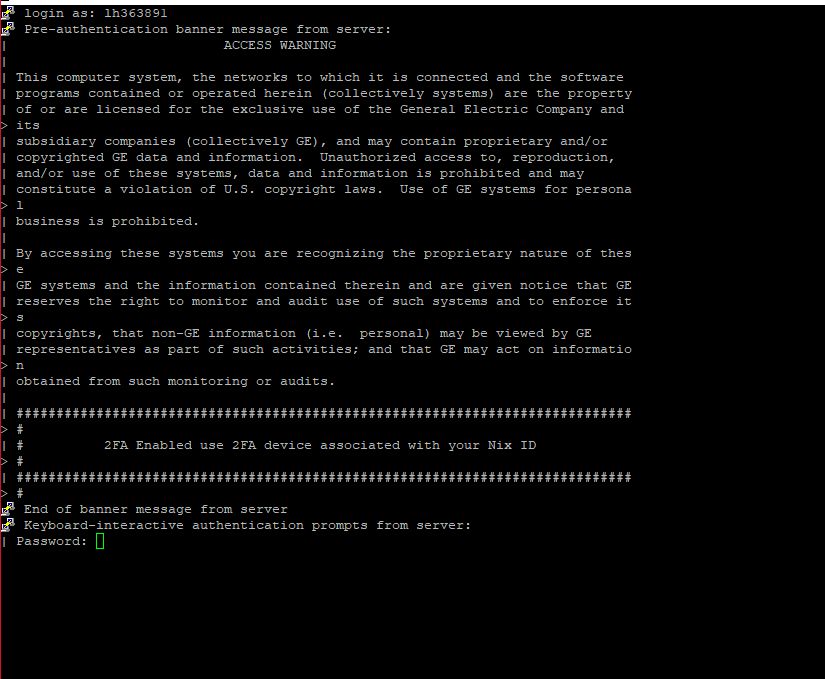
Open the putty software, if not installed download putty software online (<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> )and install it.

Enter IP address of Linux server in highlighted area (for e.g.: 10.155.11.164) and click on open



Command line interface pops up where you can enter Unix id and SSO password and ping id





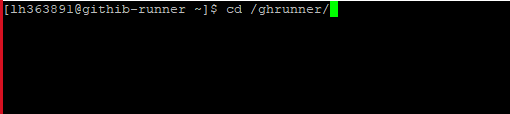
After we logged in to server, it looks like below screen



3.1: Go to the path of the directory where you want to setup runner configuration files.

(we used ghrunner directory)

cd /ghrunner --🡪get inside ghrunner directory



**Note: Follow the steps mentioned below for download and configuration by referring the screenshot of 2nd step**

3.2 mkdir actions-runner --🡪creates a new directory with the name “actions-runner”

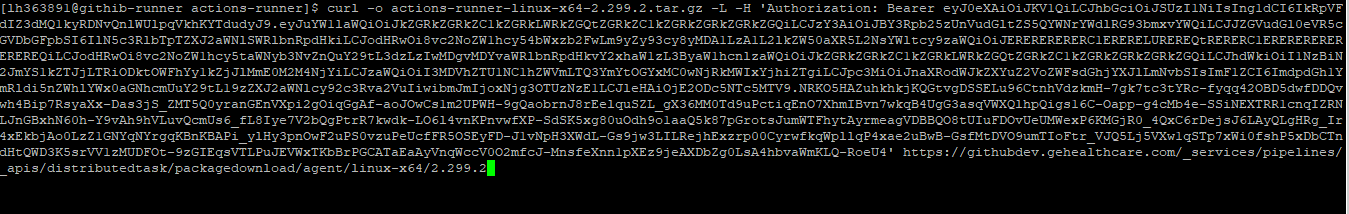


3.3 cd actions-runner -🡪get inside actions-runner



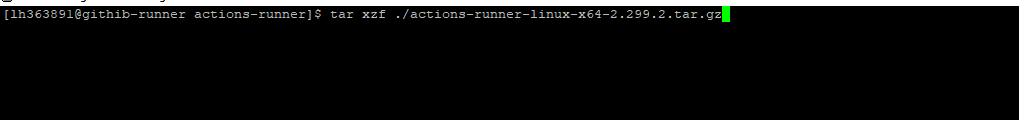
**Step 4:** Download the latest runner package (You will see instructions showing you how to download the runner application and install it on your self-hosted runner machine.)

curl -o actions-runner-linux-x64-2.299.2.tar.gz -L -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IkRpVFdIZ3dMQ1kyRDNvQnlWUlpqVkhKYTdudyJ9..d247XCmZhHnq-ZHNyd0vMFXbGqMxwG6TTxTYpzXcEdUwjlE9yaBRN38W9GCa1a\_wEeTGU01usM35Kff1rRdPGHI3o20HuODw9tA9Qf0j6KbFBjghlko57jiSD2xjilZZe8\_f7UG7Mzpqvk-ThgXcgc-1OCNeOLwe7gbYJVq-3ErVh9K0YDZgBh62c7RThAE34ibnfAsn29qaxYaiA7GGIJBvUzELbAQAFrUjrW9D84F5ctElK\_5UFivPRSVhXLh9wwfcENiES2Xwu6K3gMJW36XHX1KT-l3Ks9HBt0RNO889HBnV1QE-8sJ1eY8aBTaZlyp58z3W9U0zT63BFowz\_RJridAa9f7xmUwW\_KTi5bMkNqQ6wVOzivUzOVD0jjvZDZOV\_GgU2qjVVKNoiaNBA68XyxFIOd7Vre\_OCWZJE80lJcGBQF6GUFCo-qVVc46RjClYBazHTd3UbaJxJQdKO0V4EEkT8eKCBPNa8aRtHw88izvbnzmTAoxuG-9-T-HZhzE9t3AUtMHSfoTO42amMdd87ygz5-Nc-uFbC4wsHHVE\_loC9kcosFHMattMREkdGHfRvXart93u8JwglwkUhILcEAFdiGLo-3pQC9vp\_1-hQk4ZYcl4zFi67OtSTM2\_3UloxS6JYEIIYoNPgEL7NwkNHtzOnOleooMXVV1\_LRs' <https://githubdev.gehealthcare.com/_services/pipelines/_apis/distributedtask/packagedownload/agent/linux-x64/2.299.2>



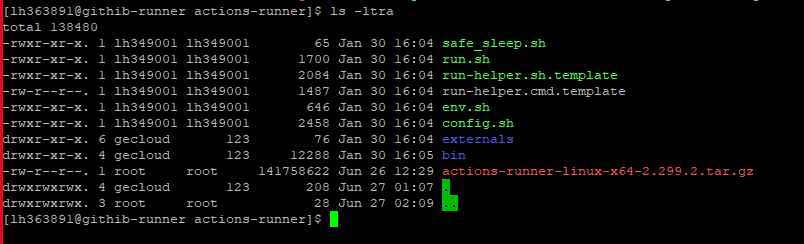
**Step 5:** # Extract the installer

tar xzf ./actions-runner-linux-x64-2.299.2.tar.gz



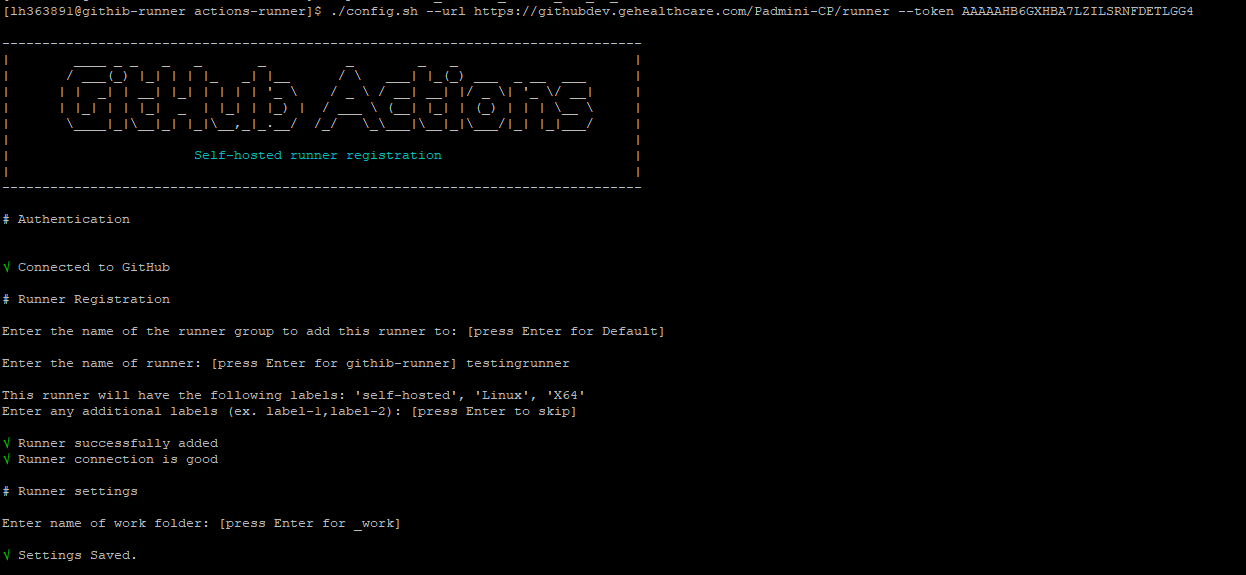
5.1 Once latest runner zipfile is extracted, all configuration files will exist inside actions-runner folder

ls -ltra -🡪list all files including the hidden ones



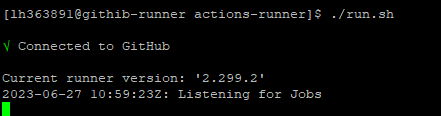
**Step 6:** # Start the configuration

./config.sh --url https://github.build.ge.com/503363891/testing1 --token AAA4XINTU7AO2OQKYOIIUQLESPYGS

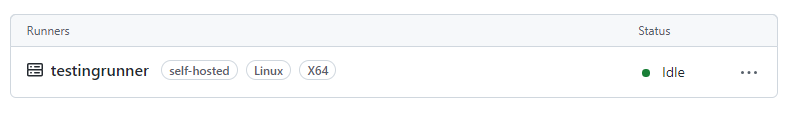


**Step 7:** Bring Runner Online

./run.sh



Runner comes online

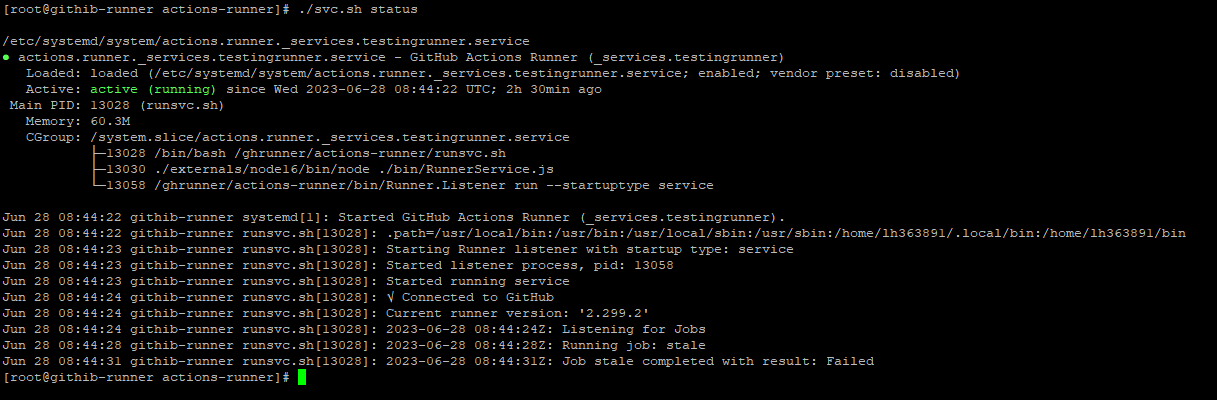


**Step 8:** To make runner as a service execute below commands in Linux server as a **root user**

# sudo ./svc.sh install

# sudo ./svc.sh start

# sudo ./svc.sh status



# Downloading/cloning MigrationScript repository

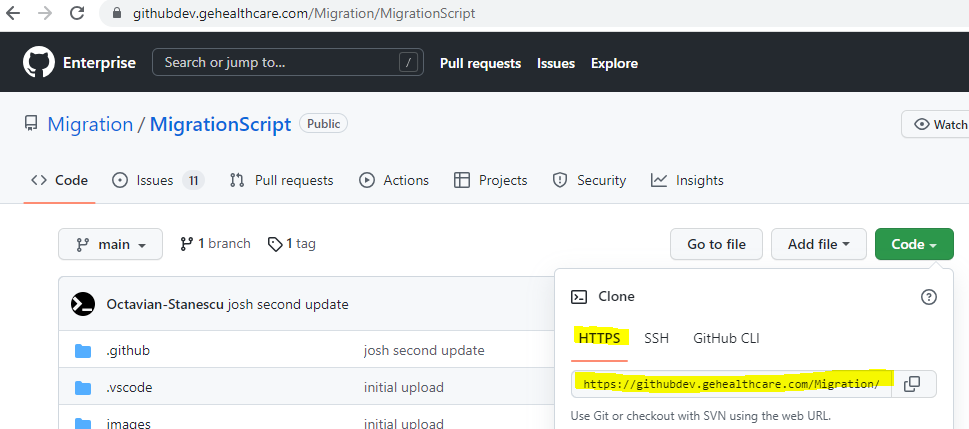
Approach 1:

Step 1: How to get Migration script

Open the Gitbash, if not installed download Gitbash software online (<https://git-scm.com/downloads> )and install it.

Clone the MigrationScript repository to your local system using gitbash

1.2.1 Copy the HTTPS URL from the MigrationScript repository



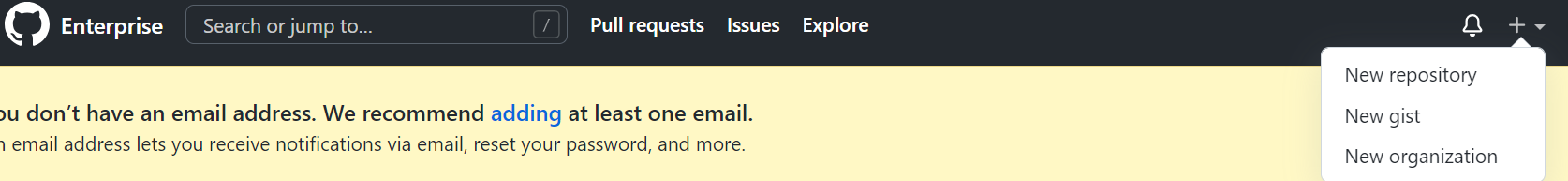
1.2.2 In Gitbash clone the above repository to your local system

$ git clone <https://githubdev.gehealthcare.com/Migration/MigrationScript.git>

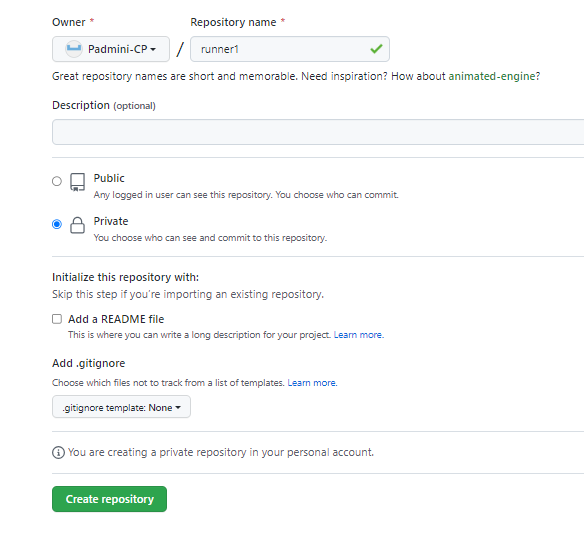


Create a new repo with any name in your github account( <https://githubdev.gehealthcare.com> )for eg :runner

1.3.1 Click on + symbol on the right side of the github account and select new repository in dropdown



1.3.2 Enter a repo name without selecting README file and click on create repository



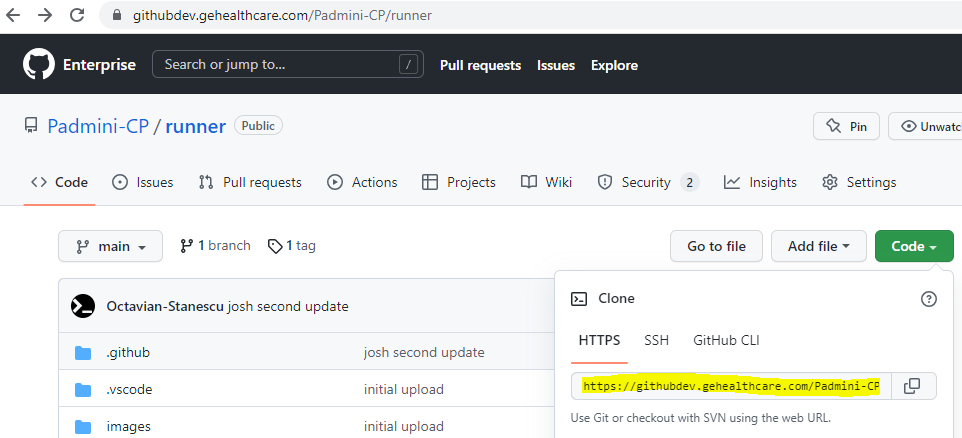
Using git bash push the migration script repository to the newly created repo using commands below

$cd MigrationScript



1.4.1 newrepourl can be taken from <https://githubdev.gehealthcare.com> account: refer below screenshot

newrepourl: <https://githubdev.gehealthcare.com/Padmini-CP/runner.git>

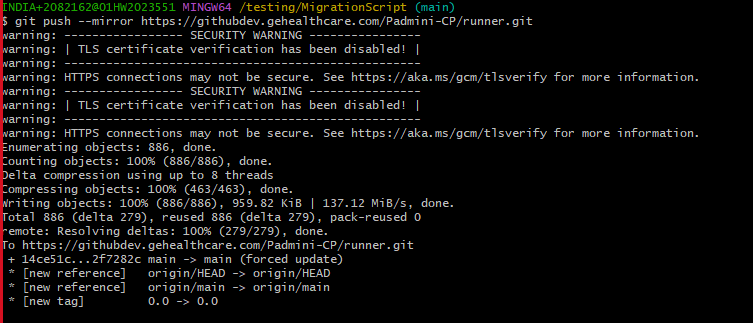


1.4.2 in gitbash run the below commands to push existed repository (MigrationScript)

$ cd MigrationScript

$ git push –mirror <newrepUrl>

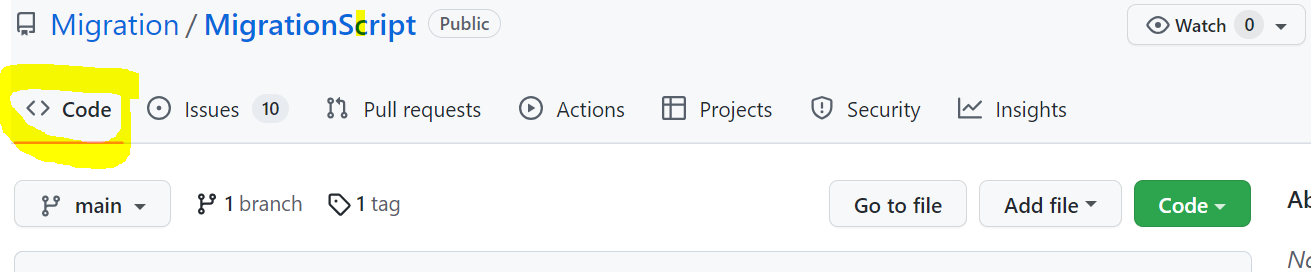
For eg : git push –mirror <https://githubdev.gehealthcare.com/Padmini-CP/runner.git>



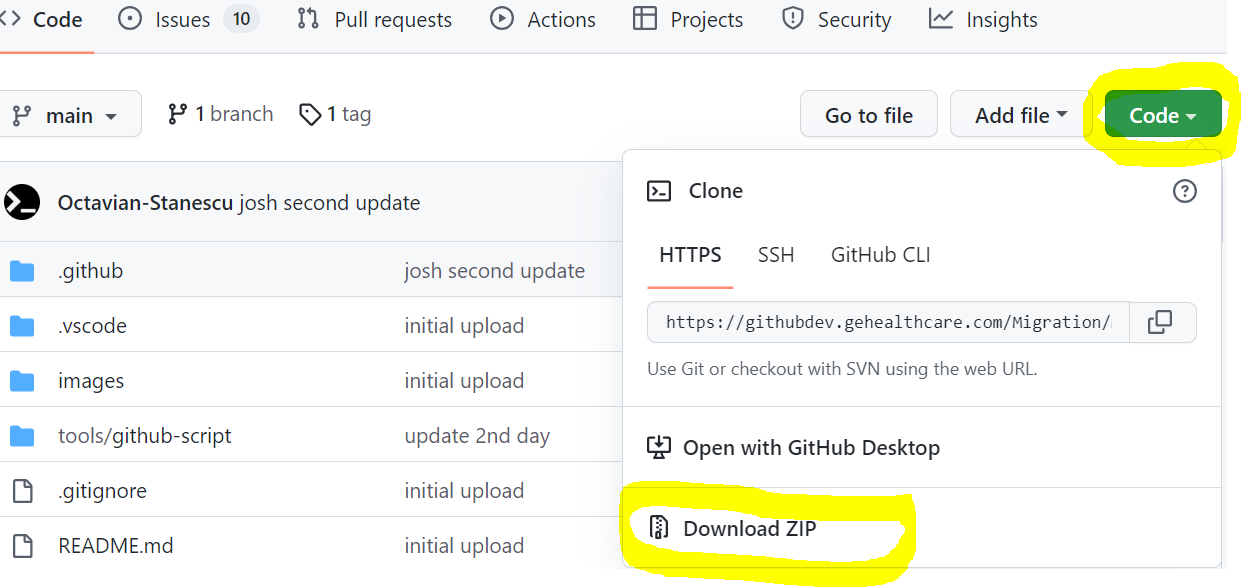
Approach 2:

Step 1 : Download a zip file from MigrationScript repo go to this link (<https://githubdev.gehealthcare.com/Migration/MigrationScript.git>)

Click on the code to display codes within the repo

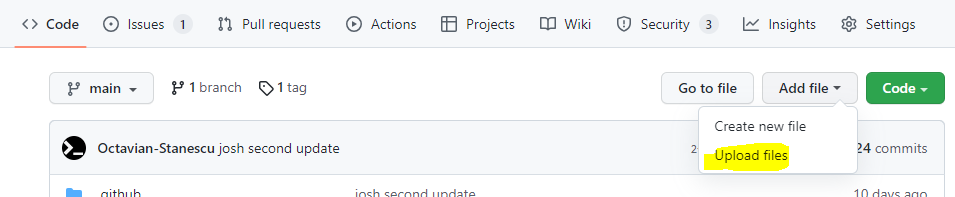


Click on the code dropdown and click on Download Zip , Extract the downloaded zip file in your local system



Step 2: Uploading extracted file to a newly created repository as shown in below screenshot

Refer 1.3.2 to check how to create new repository

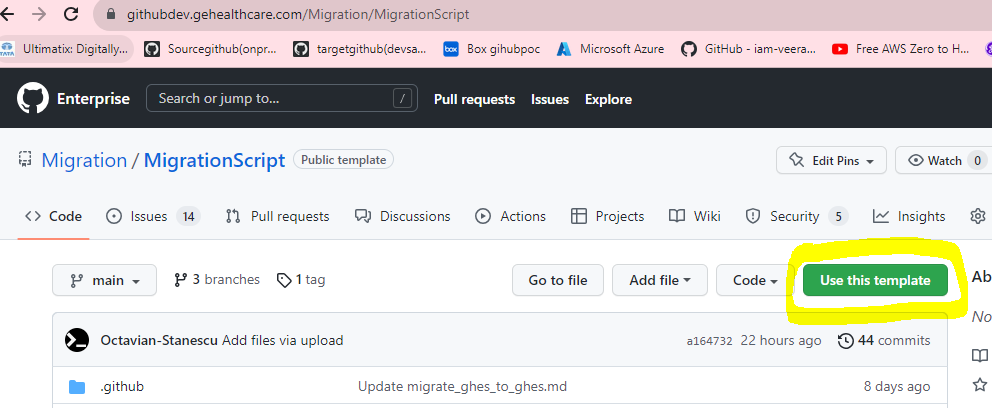


Approach 3:

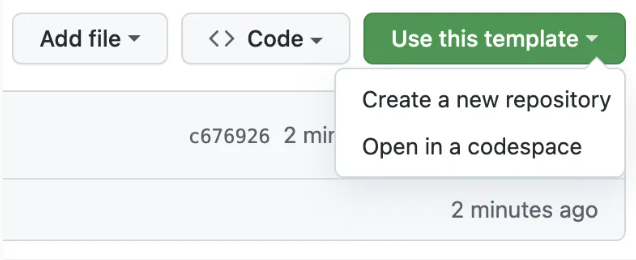
Creating a repository from MigrationScript repo template

Step 1: Login to MigrationScript repository using below link, then click on **Use this template** as shown in screenshot.

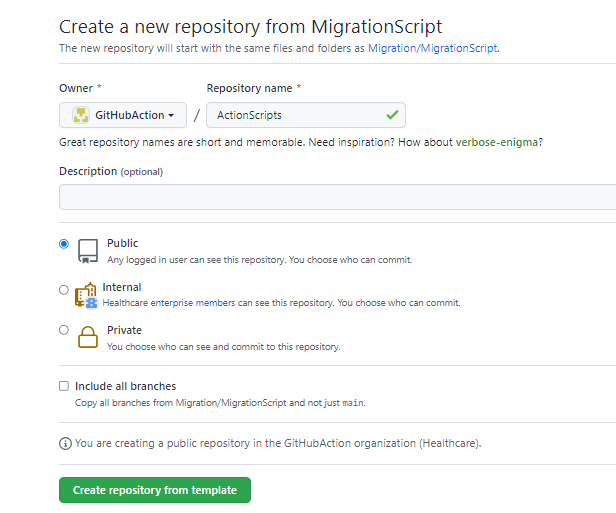
<https://githubdev.gehealthcare.com/Migration/MigrationScript>



Step 2: Select **Create a new repository.**



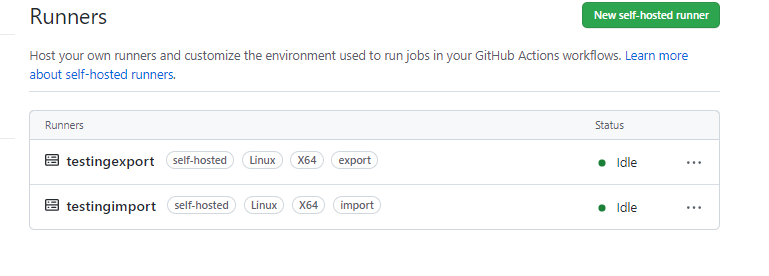
Step 3: Type a name for your repository, and an optional description and click on **Create repository from template.**



[Creating two runners in ActionScript repository](https://docs.github.com/en/actions/security-guides/encrypted-secrets#creating-encrypted-secrets-for-a-repository)

Refer Steps to setup Self Hosted Runner in Linux: page no 1

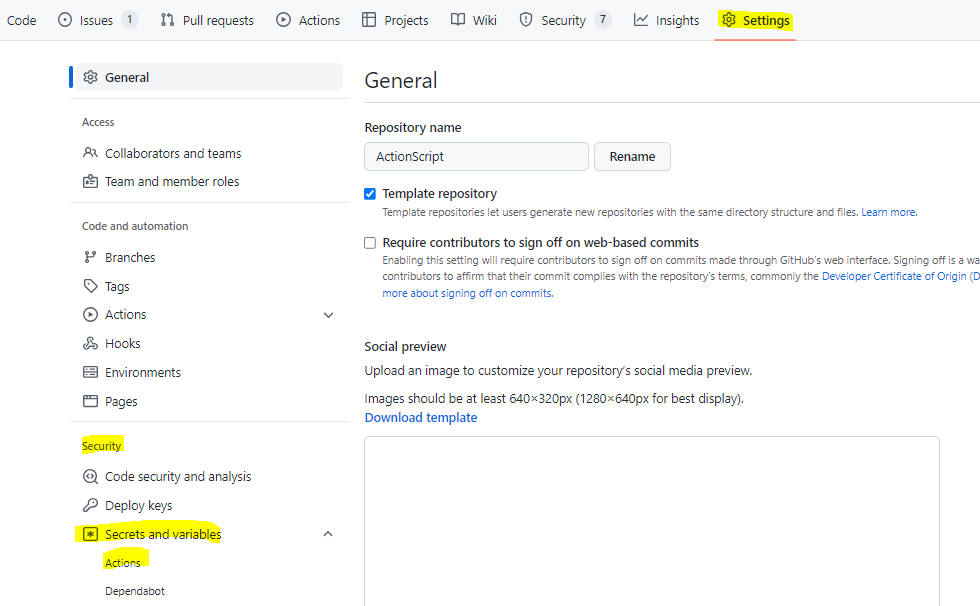
Note: Created two runners for testing new GitHub Migration in new environment



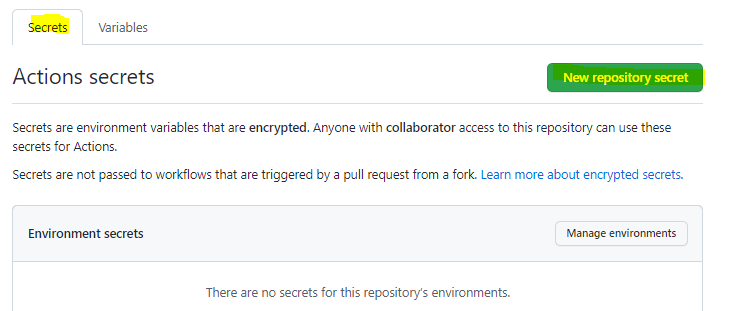
[Creating encrypted secrets and Variables for a ActionScript repository](https://docs.github.com/en/actions/security-guides/encrypted-secrets#creating-encrypted-secrets-for-a-repository)

Step 1: Click on Settings under ActionScript repository

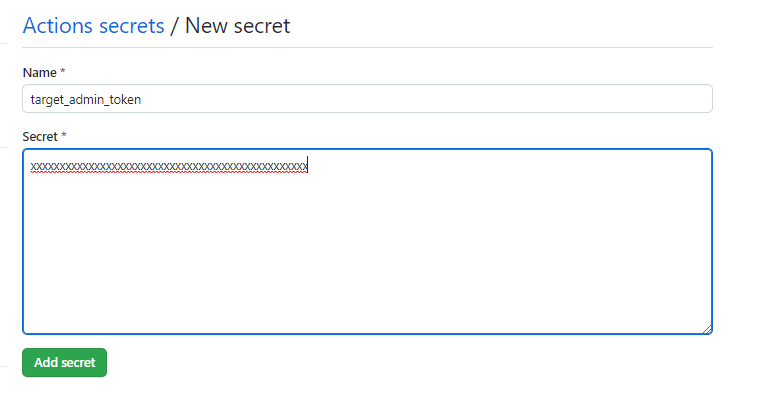
Step 2: In the "Security" section of the sidebar, select Secrets **and variables**, then click **Actions** as shown in screenshot below



Step 3: Click the **Secrets** tab then click on **New repository secret**



3.1 In the **Secret** field, enter the value for your secret then Click on **Add secret.**



**Required Values**

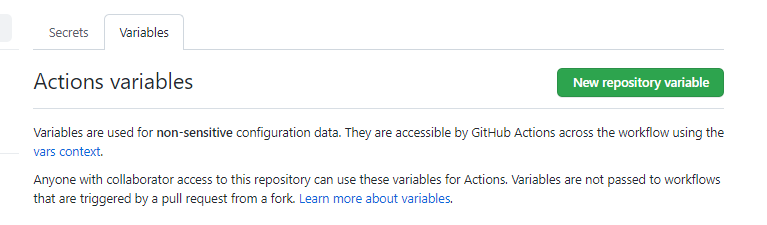
SOURCE\_ADMIN\_TOKEN

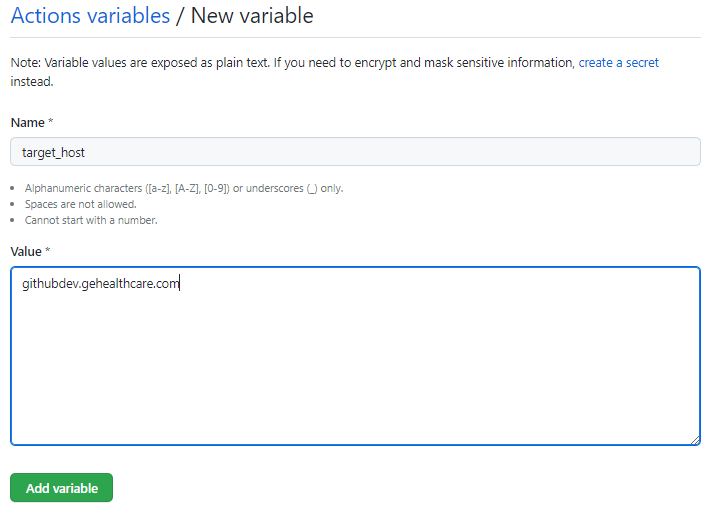
TARGET\_ADMIN\_TOKEN

TARGET\_GHES\_ADMIN\_SSH\_PRIVATE\_KEY

**3.2** Click the **Variables** tab then click on **new repository variable.**

In the **Variable** field, enter the value for your variable then Click on **Add Variable.**





**Required Values**

SOURCE\_HOST github.build.ge.com

TARGET\_HOST githubdev.gehealthcare.com

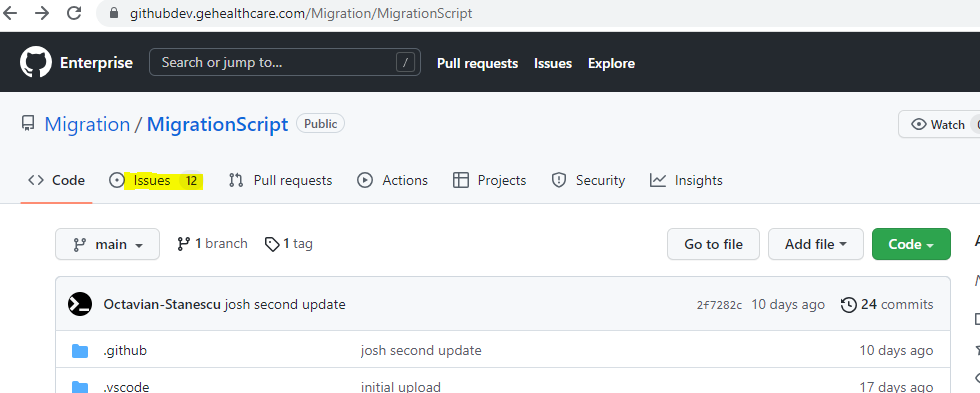
TARGET\_ADMIN\_USERNAME ghadmin

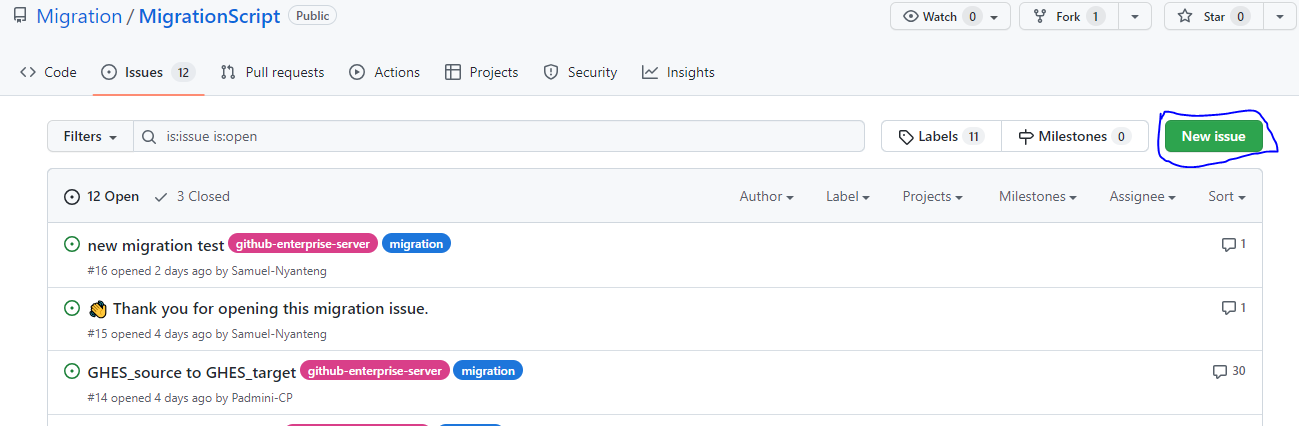
# SINGLE REPO MIGRATION USING MIGRATION SCRIPT

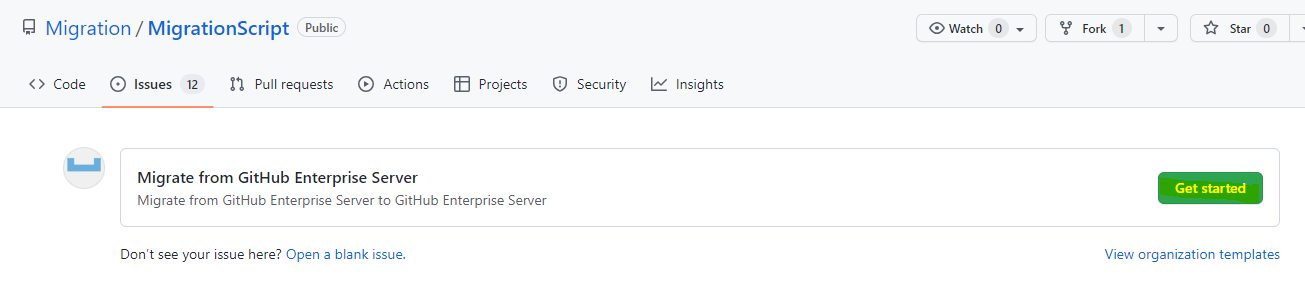
Step 1: Login to GitHub target dev URL <https://githubdev.gehealthcare.com/Migration/MigrationScript>.

You are in Organization “Migration” and Repo “MigrationScript”.

Click on Issues and select New issue and click on Get started as shown in below screenshots

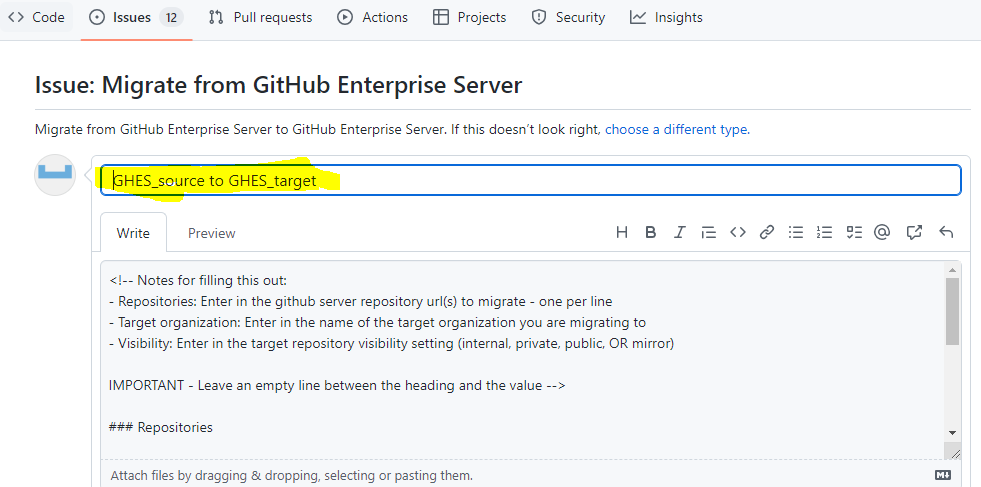




**Note**: This repository contains GitHub Actions workflows to migrate repositories from various sources to GitHub.com

Migration script: <https://githubdev.gehealthcare.com/Migration/MigrationScript>

**Step 2:** Name the issue in highlighted area (for example: **GHES\_source to GHES\_target**)

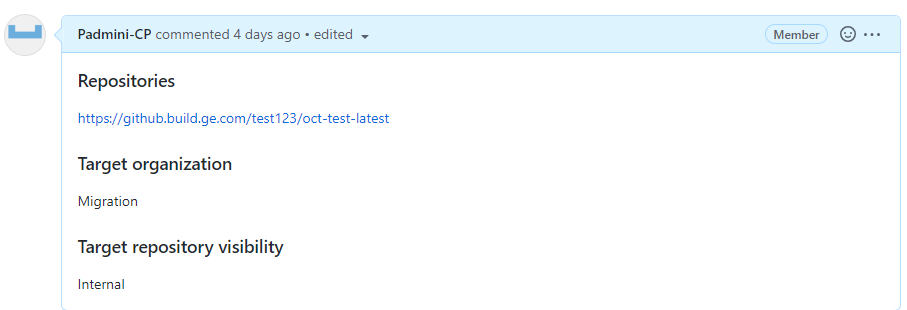


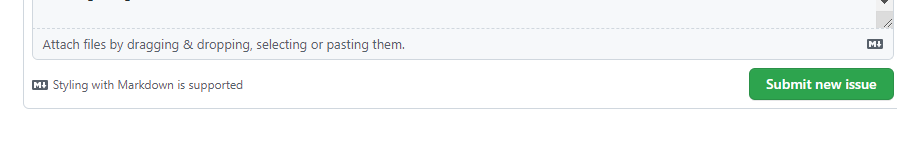
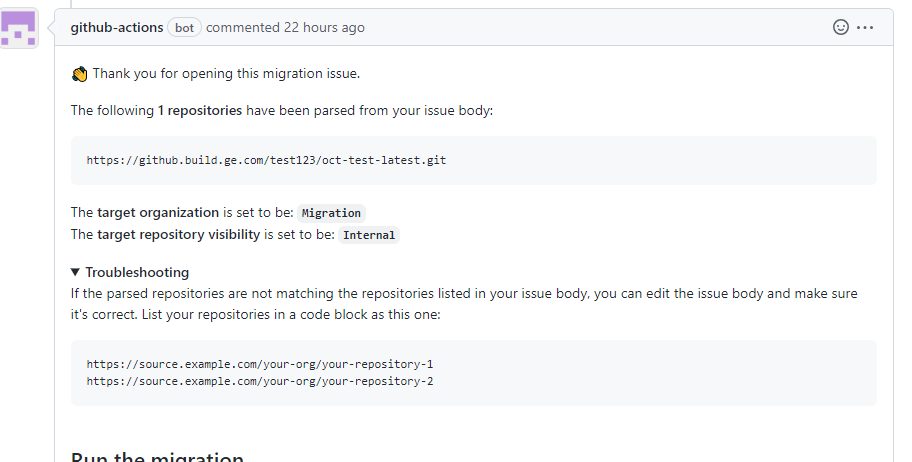
2.1 Specify **source repository**, **Target organization** and **Target repository visibility** in issue's body and click on **Submit new issue**

Screenshot repo is configured as follows:

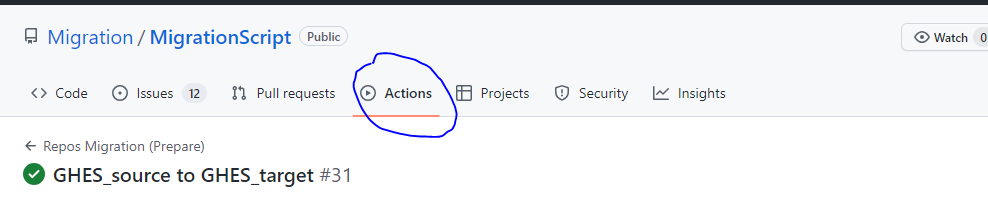
**Repository:** <https://github.build.ge.com/test123/oct-test-latest>   
 **Target organization: Migration**   
 **Target repository visibility :** Internal

**Note**: Make sure your Source repo name should be **oct-test-latest** which is correct, if you define as oct-test-latest.git it is incorrect.

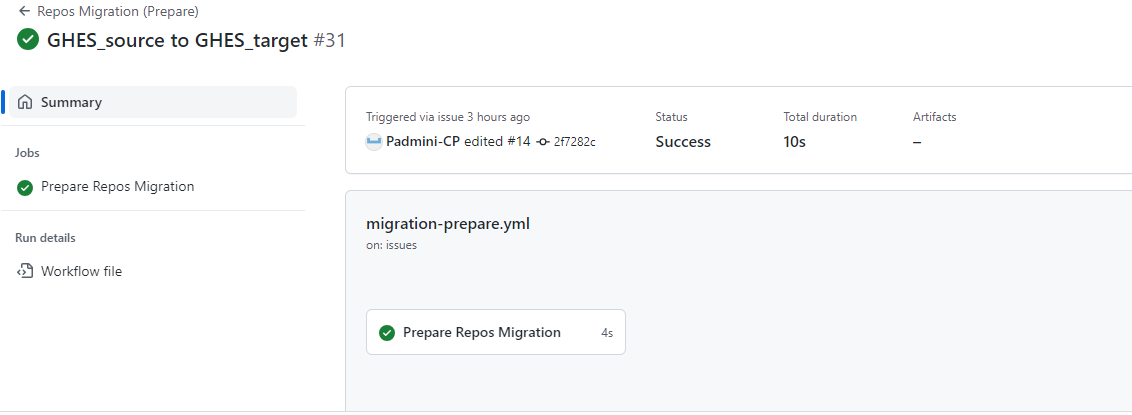


 2.2 Once you click on **Submit new issue**,you will see the response below.  
 

2.3 Click on **Actions** as shown in below screenshot to check the status of issue which is ready for migration (prepare repos migration)



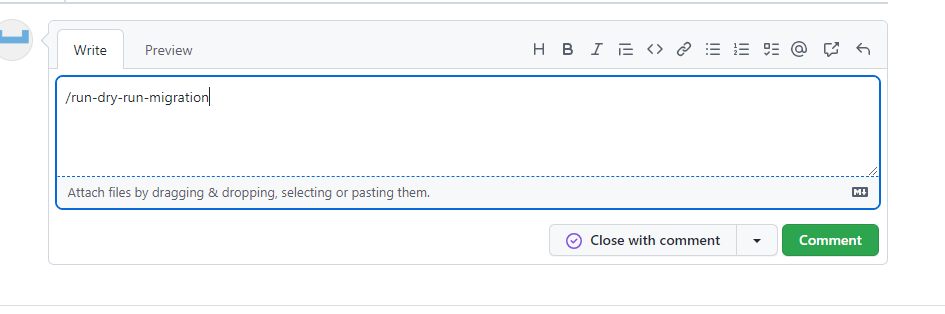
Once you click on **Actions**, you will see below response.



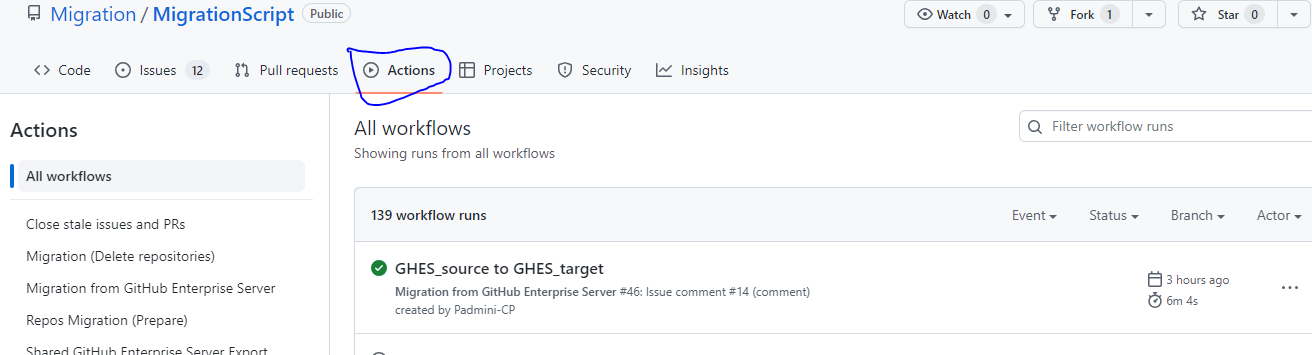
**Step 3:** Click on **Issues** as shown in Step1 to perform the dry-run Migration.

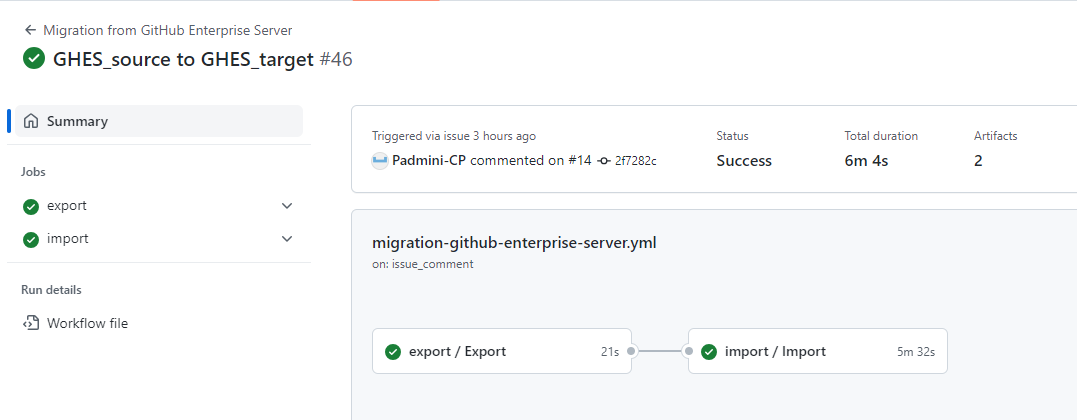
Add the following comment to the issue to perform a dry-run migration and click on **Comment**

/run-dry-run-migration

**Note: Dry-run migration will not lock your source repository and therefore does not block your users from continuing to work on the source repository.**

**Step 4:** Click on Actions to check the status of issue Workflow





# Sequence of Events for Migration using Runner

**Step 1:** Login to GitHub target URL followed by Migration organization and MigrationScript repository

Please find the link below for MigrationScript repository

<https://githubdev.gehealthcare.com/Migration/MigrationScript>

**Step 2**: Click on **Issues** and select existing **Issue** or you can create **New** **Issue**

**2.1**: Select existing issue then edit **source repository**, **Target organization** and **Target repository visibility** in issue's body and click on **update comment**

**Or**

**2.2:** if we are creating new issue then name the issue and specify **source repository**, **Target organization** and **Target repository visibility** and then click on **submit new issue**

**Step 3:** Click on **Actions** and select your **issue** to check the status of workflow that is **Prepare Repos Migration** status must be in green

**Step 4:** Add a comment to the existing issue with one of the following two commands in order to run a migration:

/run-dry-run-migration

**Or**

/run-production-migration

**Note:** dry-run-migration will not lock your source repository

Production-migration will lock your source repository and make it unaccessible for users.

**Step 5:** Click on **Actions** to check the status of migration

There are mainly two jobs we can observe

**Export**

**Import**

**5.1** export is running in githib-runner vm whereas import is running in ghrunner VM

export and import jobs are running in two runners and their names are same as VM name

i.e., export is running in githib-runner and import is running in ghrunner

both export and import jobs should reflect in green color for successful migration completion

Stages in Export:

Set up job

This stage defines where the export job is running(githib-runner) and in which machine(githib-runner)

Checkout scripts

This action check-out the script(download) the script which is defined in Migration/MigrationScript repository

Parse issue body

This action is to convert issues into a unified JSON structure

Set up Node

setup-node downloads distributions from actions/node-versions

Run the export

This action downloads the gh extension andyfeller/gh-repo-export to generate [GitHub repository migrations](https://docs.github.com/en/enterprise-cloud@latest/rest/migrations/orgs) and starts downloading migration archive

Upload migration archive to artifacts

This action will start uploading artifacts with the provided path and noted all timings like size of the file and time, storage.

Note: An artifact is a file or collection of files produced during a workflow run.

Generate user mapping file

Whatever modification done in user-mappings.csv, According to that input this step will generate user mapping file

Update issue with export finished

This action will notify us in issue that export is completed (Completed **dry-run** export) and reports failed Migration.

Post Setup Node

This action will clean up the Workspace after export job is implemented

Post checkout Script

This action will temporarily overriding HOME='/ghrunner/actions-runner/\_work/\_temp/fcfdc637-c8b4-41fa-96f8-d76323d04369' before making global git config changes

Completed Job

This action will store the unique values and it is input to the import.

Stages in Import:

Set up job

This stage defines where the import job is running(ghrunner) and in which machine(ghrunner)

Checkout scripts

This action check-out the script(download) the script which is defined in Migration/MigrationScript repository

Parse issue body

This action will extract the archive file which is output of the export job

print orgs

this action will print the source and target organization name in logs

Setup SSH Key

This action will request/make access to [admin@githubdev.gehealthcare.com](mailto:admin@githubdev.gehealthcare.com)

check if repo(s) exist

This action will check whether the repos is existed in source organization

Create Workspace

This action will Create a workspace to perform migration activity

Run actions/download-artifact@v3

This action will download the archive file which is stored in artifacts

Prepare migration

This action will upload the archive file to GHES(githubdev) and prepares for migration by running command ghe-migrator

update issue with status

we will get notified in the issue like Prepared **dry-run** import if we are doing dry run migration

Map users and organization

This action will read input which is modified in usermapping.csv and maps users according to organization.

Note: if no mapping found for eg:212413938, a local github account will be created for them'

Resolve any other conflicts

This action will resolve the conflicts if anything occur during migration

update issue with status

we will get notified in the issue like Resolved Conflicts for **dry-run** import if we are doing dry run migration

Start Migration

Migration will start and imports source data to target organization with help of ghe-migrator

update issue with status

we will get notified in the issue like Completed **dry-run** import successfully if we are doing dry run migration

Migration unlocks

This action will unlock the target repo which exists in target organization

Resolve any other conflicts

This action will resolve the conflicts if anything occur during migration

update issue with status

we will get notified in the issue like Resolved Conflicts for **dry-run** import if we are doing dry run migration

Start Migration

Migration will start and imports source data to target organization with help of ghe-migrator

Add completed comment

If condition is described in this step like After migration complete! In order to delete the migrated repositories of the dry-run, add a comment with the following slash command in the issue comment

/delete-repositories

orelse

Define as “migration completed”

complete job

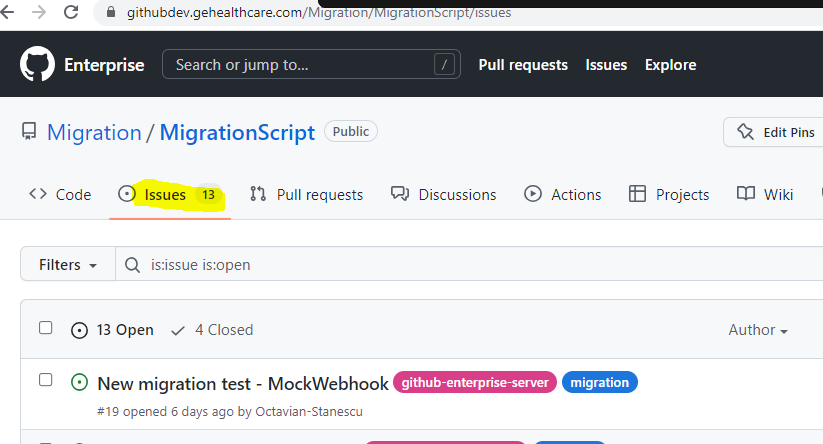
Cleaning up orphan processes

# Deleting an issue

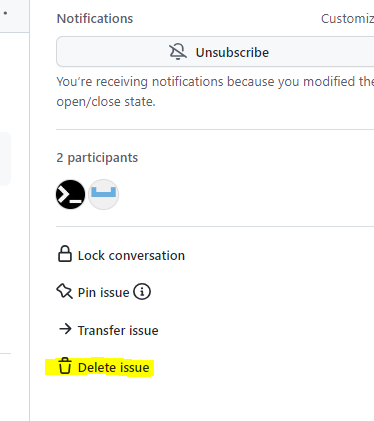
People with admin permissions in a repository can permanently delete an issue from a repository

**Step 1:** Navigate to the main page of the repository. Under your repository name, click Issues. (For e.g.: we are deleting issue in MigrationScript repo)

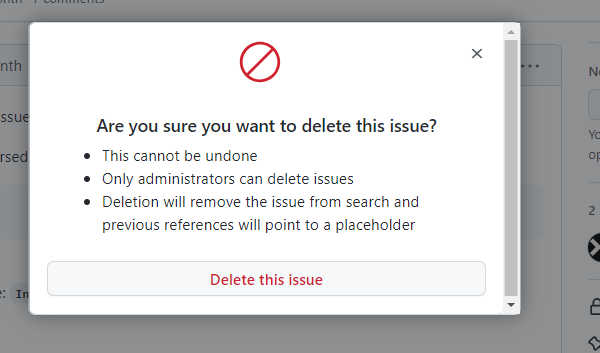
Select the issue you want to delete.



**Step 2:** On the right-side bar, under "Notifications", click Delete **issue**



**Step 3:** You will get notification prompt like below screenshot, to confirm deletion, click Delete this issue.



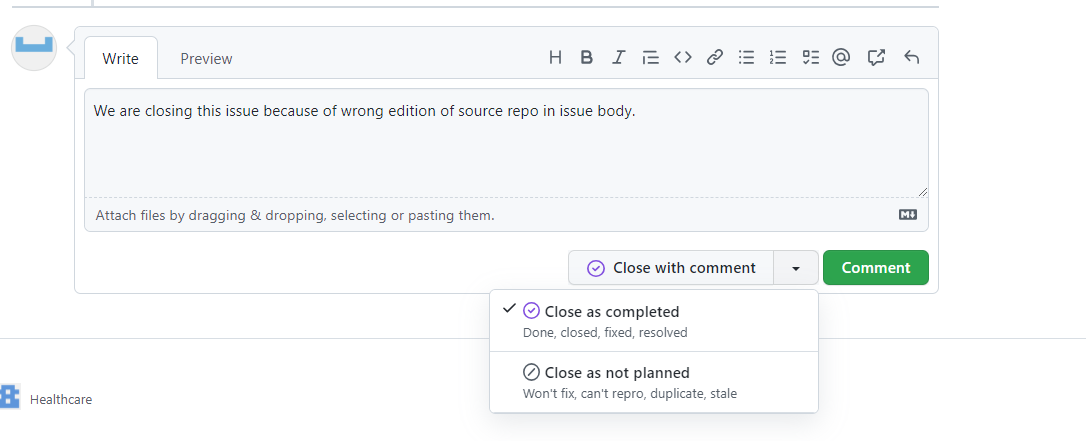
Closing an issue

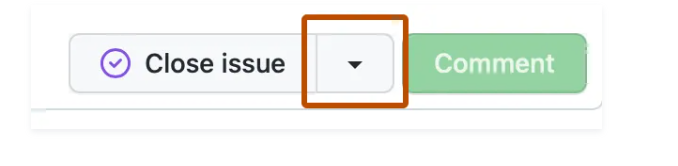
You can close an issue when bugs are fixed, feedback is acted on, or to show that work is not planned

**Step 1:** Navigate to the main page of the repository. Under your repository name, click Issues.

**Step 2:** In the list of issues, click the issue you'd like to close

**Step 3:** Optionally, to change your reason for closing the issue, next to "Close issue," select, then click a reason and click on close issue





Reopening a closed Issue

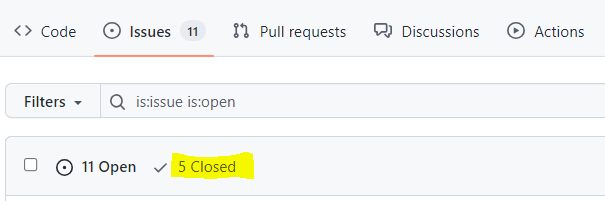
you can re-open your own issues if you closed them yourself

you cannot close or re-open issues opened by someone else

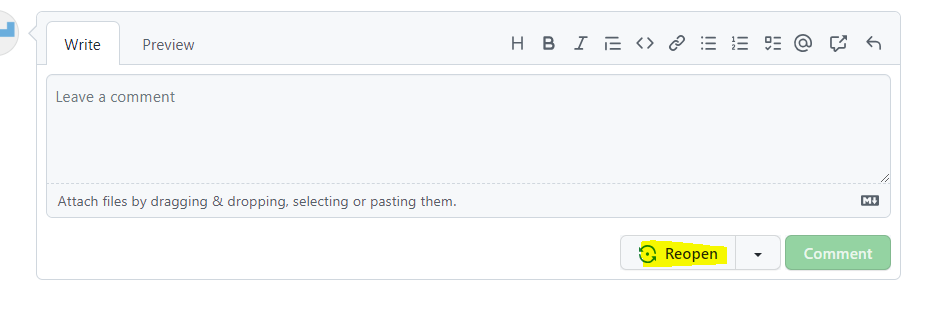
you cannot re-open your own issues if a repo collaborator closed them

**Step 1:** Navigate to the main page of the repository, under your repository name, click Issues.

**Step 2:** click on closed state as show in below screenshot and select the issue which you want to reopen



**Step 3:** Click on reopen-issue which you will find in the bottom of the screen



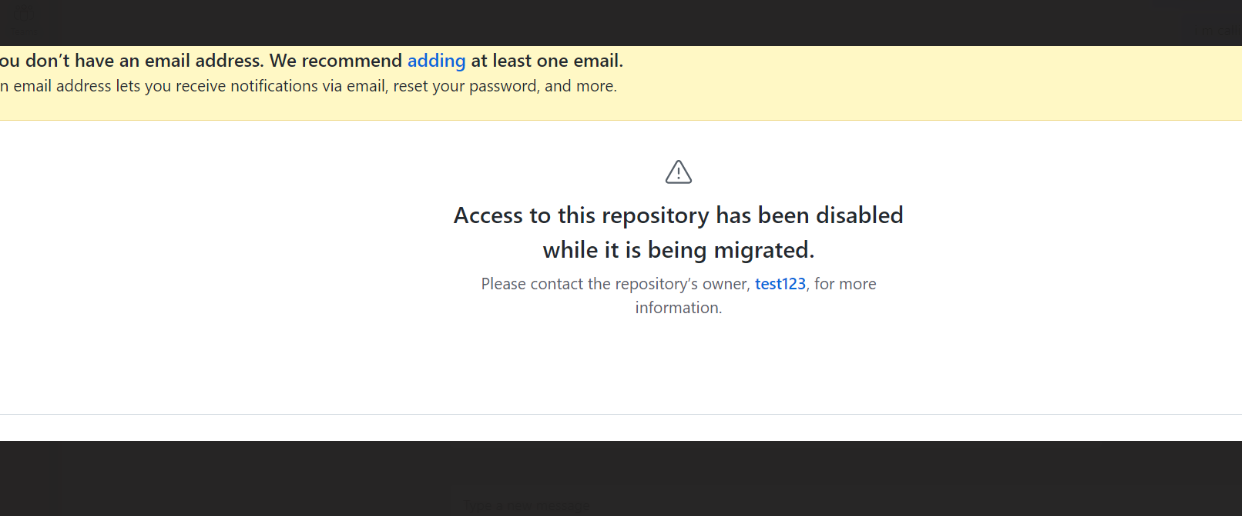
Production Migration using script(observation)

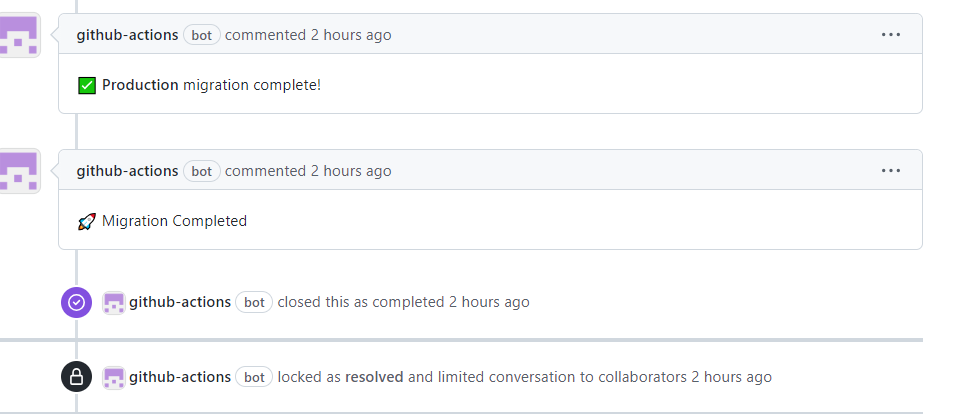
**/run-production-migration**

Observation:

Production-migration will lock your source repository and make it unaccessible for users

After Production migration complete! Issues automatically get closed (resolved) due to bot activity at the backend

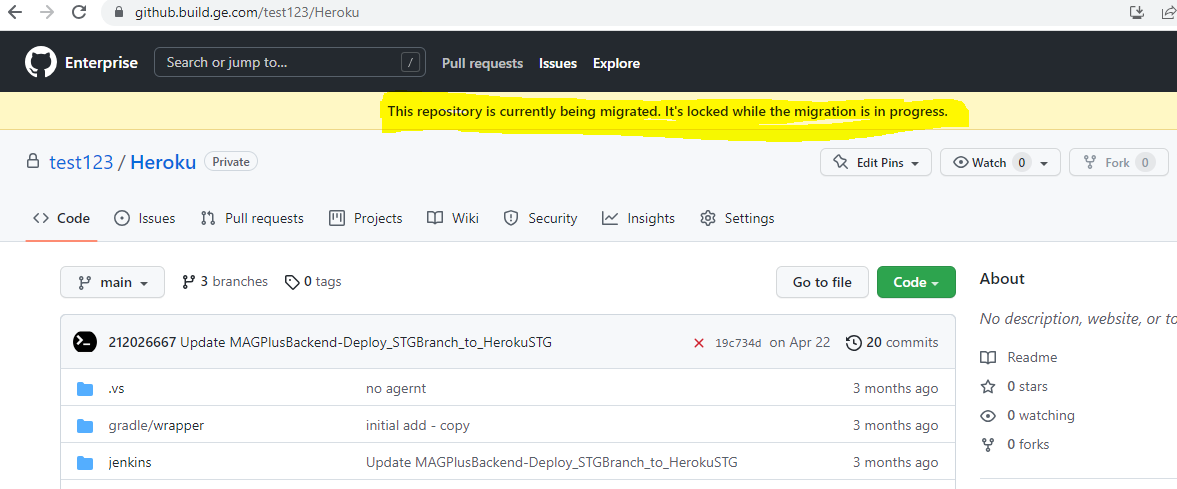




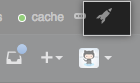
Note: dry-run-migration will not lock your source repository but during migration any changes is making in source repository will not get modify in target

# LOCKING AND UNLOCKING/DETING SOURCE REPOSITORIES AFTER PRODUCTION MIGRATION

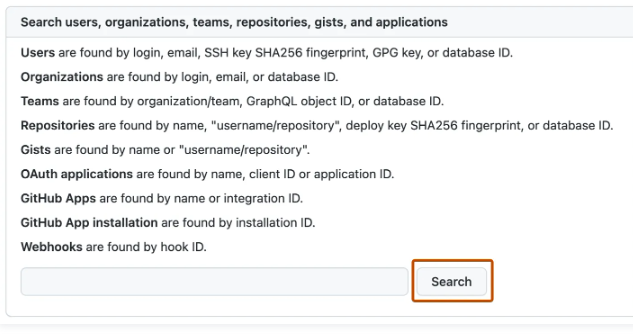
**Information** : Once we ran production data migration the source repo is locked forever - we can delete it, but we cannot unlock (only Corp can unlock with "staff tools")



Step 1: From an administrative account on GitHub Enterprise Server, in the upper-right corner of any page, click enterprise settings.

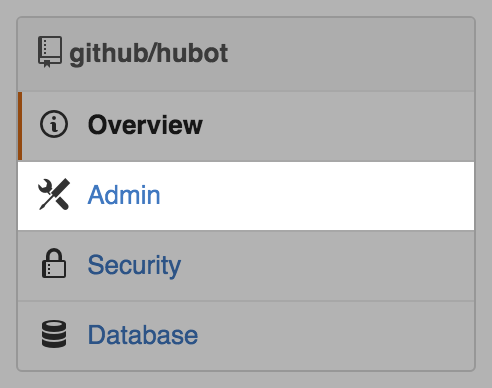


Step 2: Under "Search users, organizations, teams, repositories, gists, and applications", type the name of the repository in the text field. Then to the right of the field, click **Search**

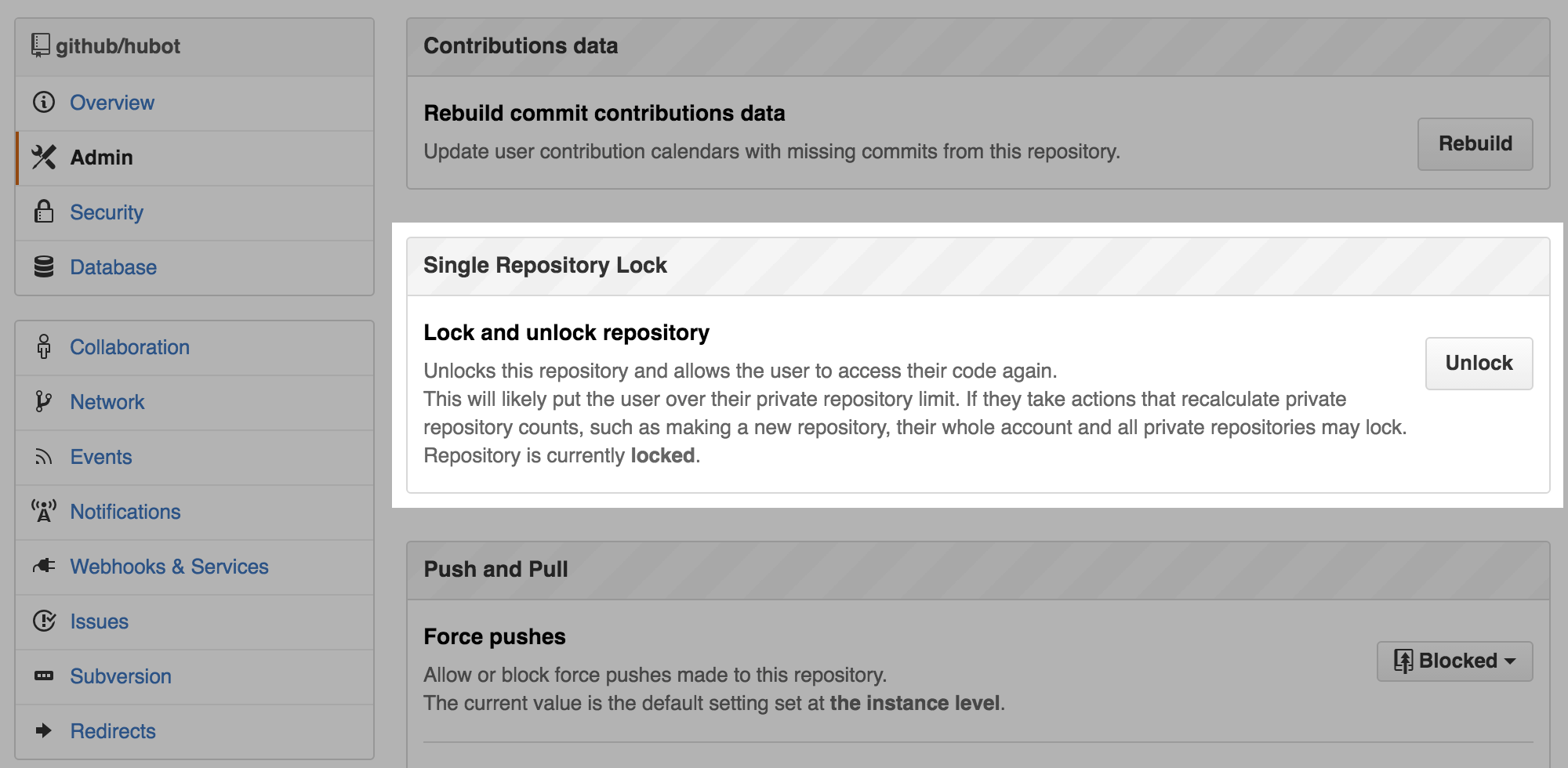


Step 3: Under "Search results – Repositories", click the name of the repository.

Step 4: Click on **Admin** in the left sidebar.



Step 5: Click **Unlock** in the Single Repository Lock area*.*



**Once a repo is migrated with success and user accepts it, is there a curl command to delete the repo from source completely?**

We have three ways to delete Source repository after Migration was successful

**1.Using Curl**

curl -L \

-X DELETE \

-H "Accept: application/vnd.github+json" \

-H "Authorization: Bearer <YOUR-TOKEN>" \

-H "X-GitHub-Api-Version: 2022-11-28" \

<https://api.github.com/repos/OWNER/REPO>

**2.GitHub CLI RESTAPI**

gh api \

--method DELETE \

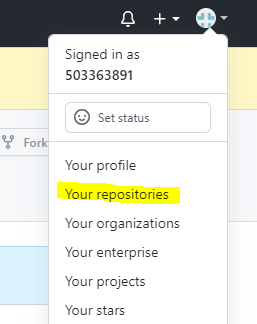
-H "Accept: application/vnd.github+json" \

-H "X-GitHub-Api-Version: 2022-11-28" \

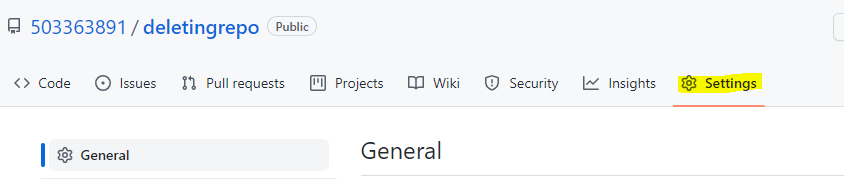
/repos/OWNER/REPO

**3.UserInterface (GUI)**

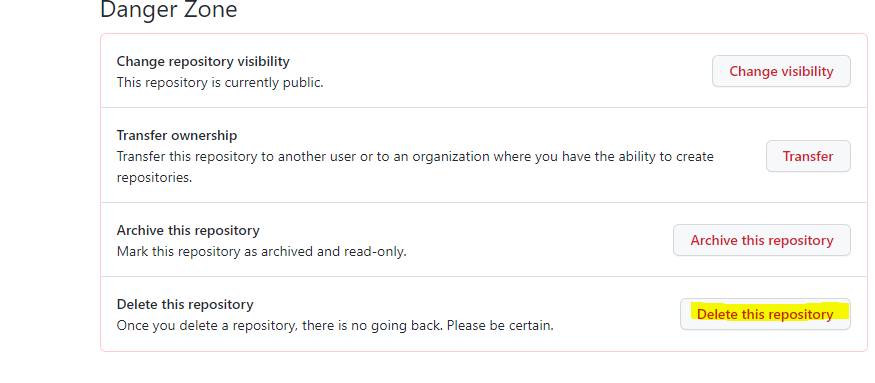
Step1: Click on your profile icon from the top-right and select Your repositories!



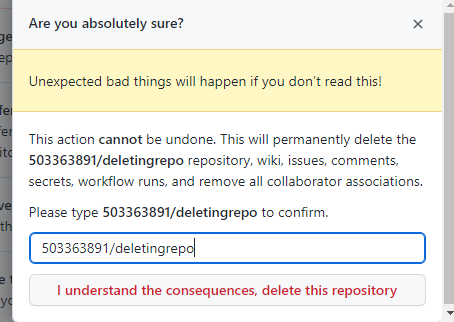
Step 2: From this page, find the repository you want to delete, and click on the title. Then, locate the toolbar at the top and click the Settings tab.



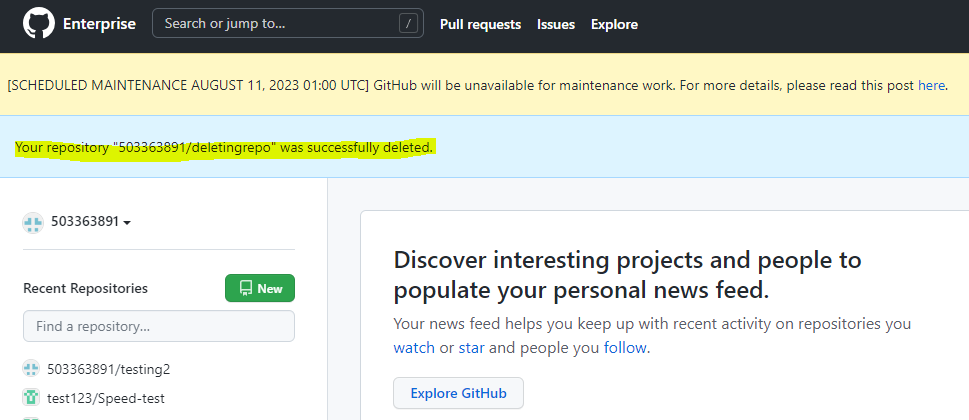
Step 3: Scroll all the way to the bottom of the page until you see the Danger Zone section. Here, click Delete this repository.



Step 4: We will now see a pop up asking if you're sure you want to delete the repository. Enter the name of the repository in the text box and click the I understand the consequences, delete this repository button.



Step 5: Once you delete the repository, you'll be back on the GitHub home page, and a banner at the top will inform you that the repository was successfully deleted.



Propose a title for the issue that is easy to figure out what it does?

Org name is identical hence we can keep.

Migrating <sourceorgname> to GEHC (eg: Migrating test123 to GEHC) or

Migrating from <sourceorgname> build.ge to <targetorgname> dev.gehealthcare

(eg: Migrating from test123 builg.ge to orgone devhealthcare )

<no. of repos migration> from <sourceorgname> (eg: 50repos from test123)

issue to migrate <sourceorgname> to GEHC  
 issue to migrate <sourceorgname>\_part1 to GEHC  
 issue to migrate <sourceorgname>\_part2 to GEHC

**if /dry run failed or user doesn't like it, can we do another dry run again after some time? Or do we need to "delete" the repo from target server first?**

**Yes!** We can do again dry-run after deleting the repo from target server

/run-dry-run-migration

"dry-run" migration is recommended for the first time Migration which **will not lock your source repository** and therefore does not block your users from continuing to work on the repository

If User doesn’t like it or dry-run migration complete! In order to delete the migrated repositories, add a comment with the following slash command then they can continue migration

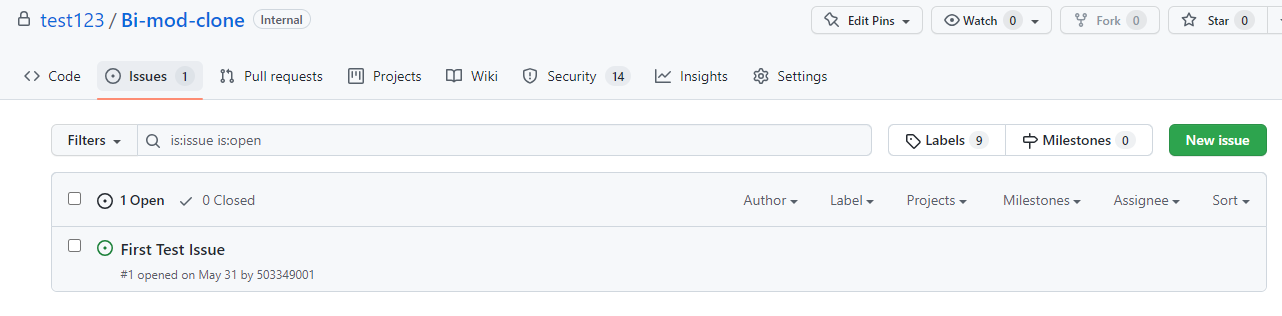
**/delete-repositories**

# User Mapping

By default we have [user-mappings.csv](https://githubdev.gehealthcare.com/Migration/MigrationScript/blob/main/user-mappings.csv) file in repo , Suppose username is not Mapped in particular repo a new account will be created for user (who created issue)after Migration.

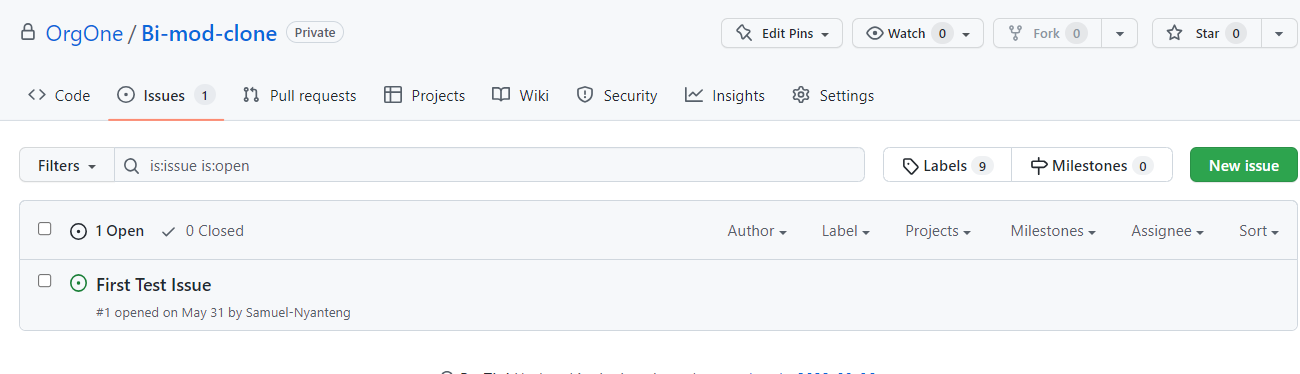
**Username is Mapped**

Samuel Created this issue in Bi-Mod-Clone repo , its showing his sso id in source account which is his username



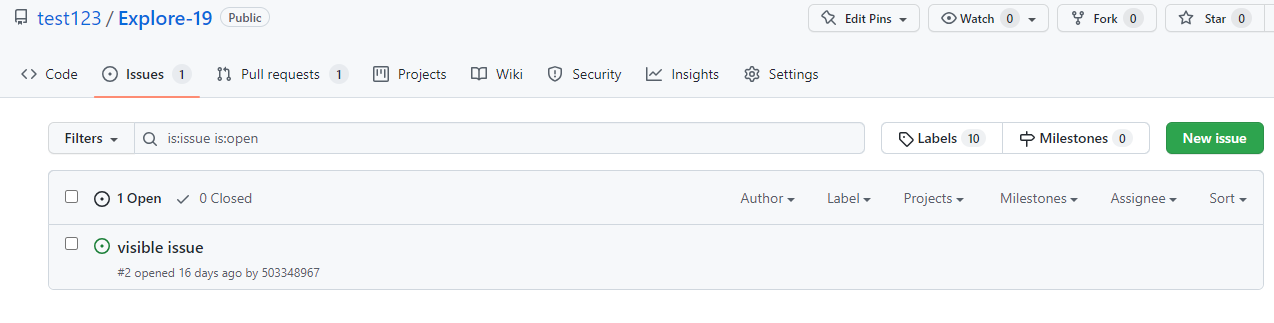
**After Migration**

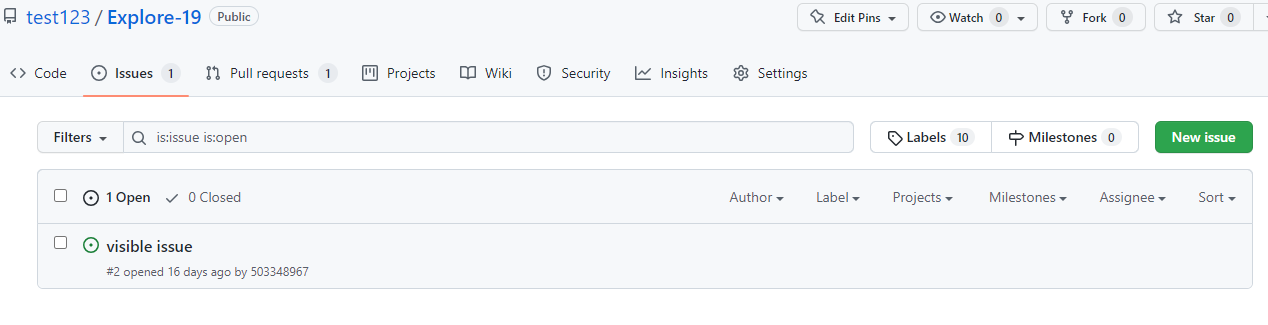
We noticed in issue his name is reflecting instead of SSO-ID



**Username is not Mapped.**

Gangappa created one issue his username can be seen in Source Account after Migration SSO-ID is reflecting not username.





**Conclusion:** If users is not Mapped a new account will be created for user (whocreatedissue)after Migration.

We can observe that in logs at import stage(Map Users and Organization)

