GitHub Desktop Tutorial – Collaborate With GitHub From Your Desktop

Updated: June 20, 2023

This Tutorial Explains how to Download and use the GitHub Desktop to Collaborate With GitHub From Your Desktop for Efficient Version Control:

As we all know, GitHub provides a website to host Git repositories. In our previous tutorials on GitHub, we have seen the developer's activities on versioning files mostly on GitHub.

There is also a Git Client wherein the developers can work on the repository offline on their local machines using git commands from the command prompt or git bash, make changes and push it back to the remote repository on GitHub.

=> Visit Here To Learn GitHub From Scratch.

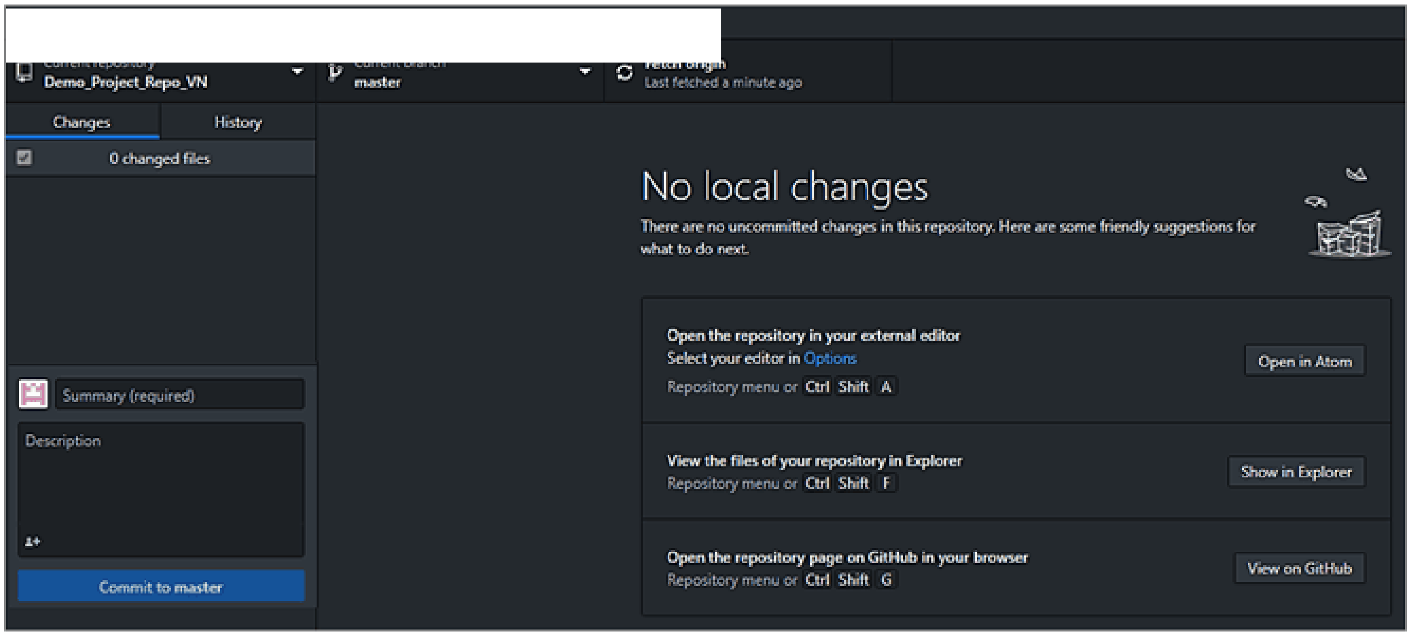
WORKING WITH GITHUB DESKTOP
A tool to collaborate with GitHub from your Desktop
© www.SoftwareTestingHelp.com

GitHub Desktop

Though the Git commands executed from the command line are great from a learning point of view, there is a good user interface to work on the local repositories i.e. **GitHub Desktop.**

GitHub Desktop for Windows can be downloaded and installed from the following URL

Launch GitHub Desktop



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A
Open in Atom

View the files of your repository in Explorer
Repository menu or Ctrl Shift F
Show in Explorer

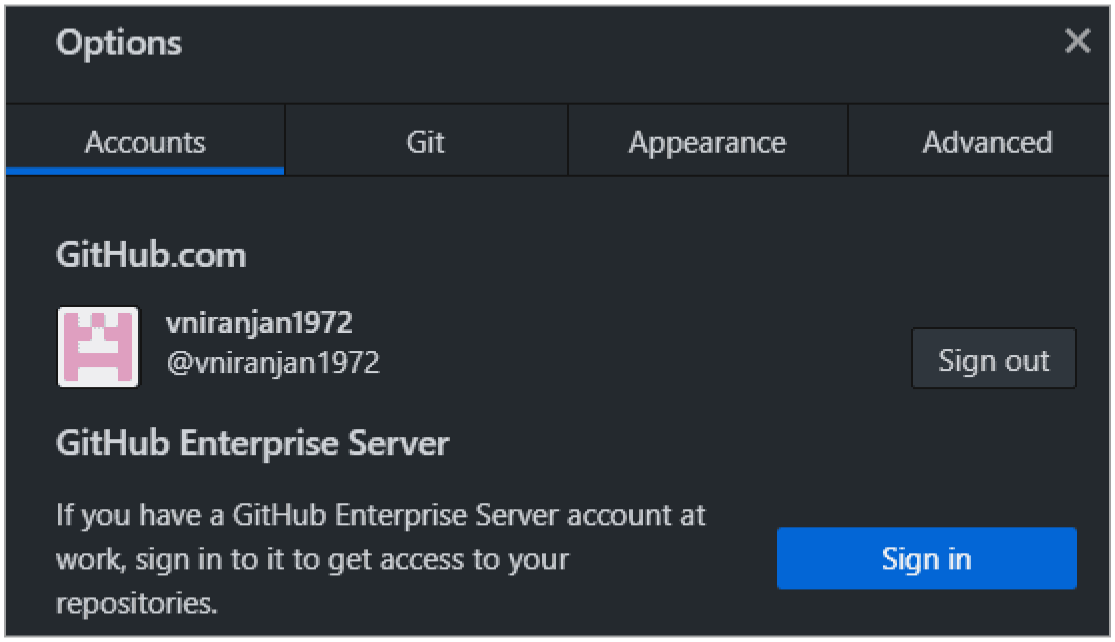
Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G
View on GitHub

Work With The Remote Repository

Once the GitHub desktop is launched, we can start by cloning the remote repository to the local machine, make changes and push it back to the remote repository.

Account Settings

In GitHub Desktop, go to **File => Options** and ensure your GitHub account is setup.



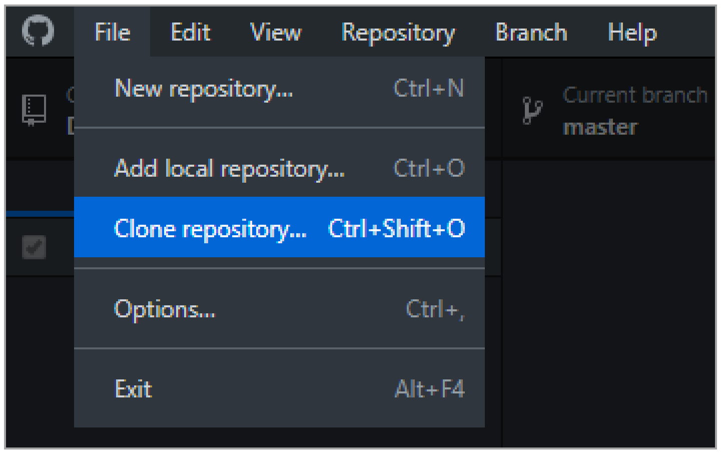
GitHub.com

vniranjan1972
@vniranjan1972

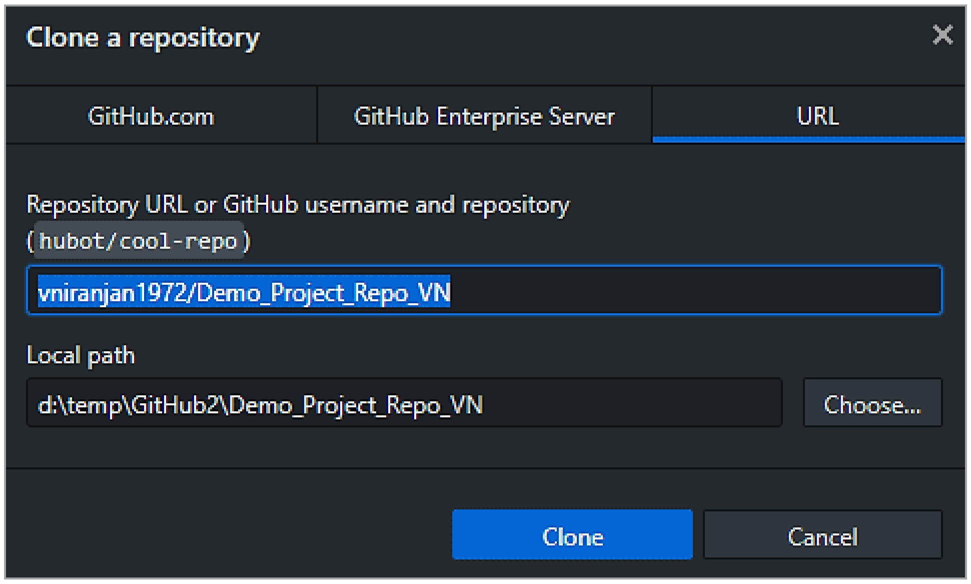
GitHub Enterprise Server

If you have a GitHub Enterprise Server account at work, sign in to it to get access to your repositories.

ct **File => Clone Repository**



Go to the URL tab and enter the remote repository details in the form of the **GitHub Username/repository.** Click on **Clone**.



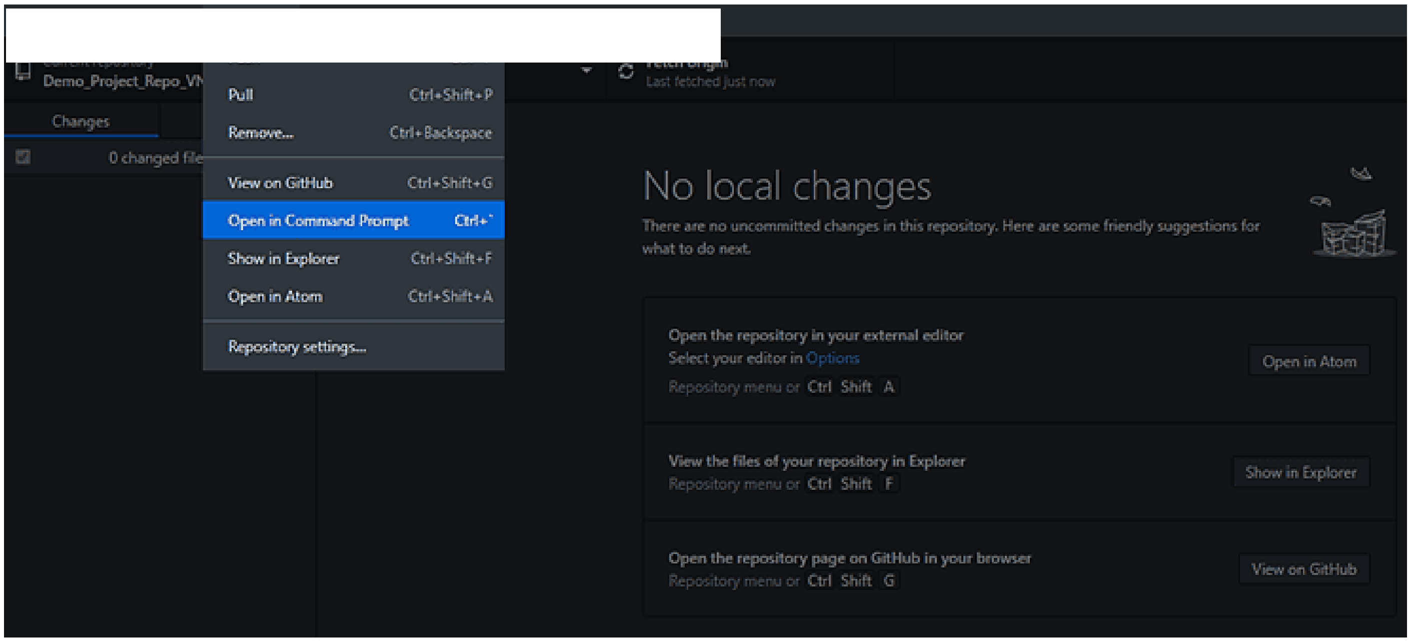
Repository URL or GitHub username and repository (`hubot/cool-repo`)

vniranjan1972/Demo_Project_Repo_VN

Local path

d:\temp\GitHub2\Demo_Project_Repo_VN

Now as the repository is cloned to the local machine, we can open the local repository contents using command prompt or explorer or even Atom editor if installed and make changes to the files.



No local changes

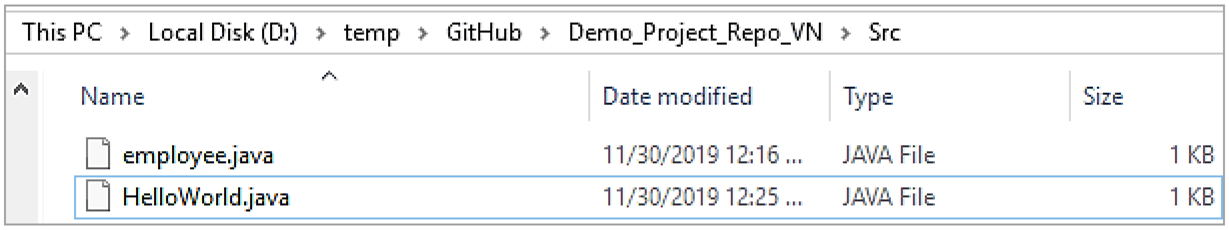
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A
Open in Atom

View the files of your repository in Explorer
Repository menu or Ctrl Shift F
Show in Explorer

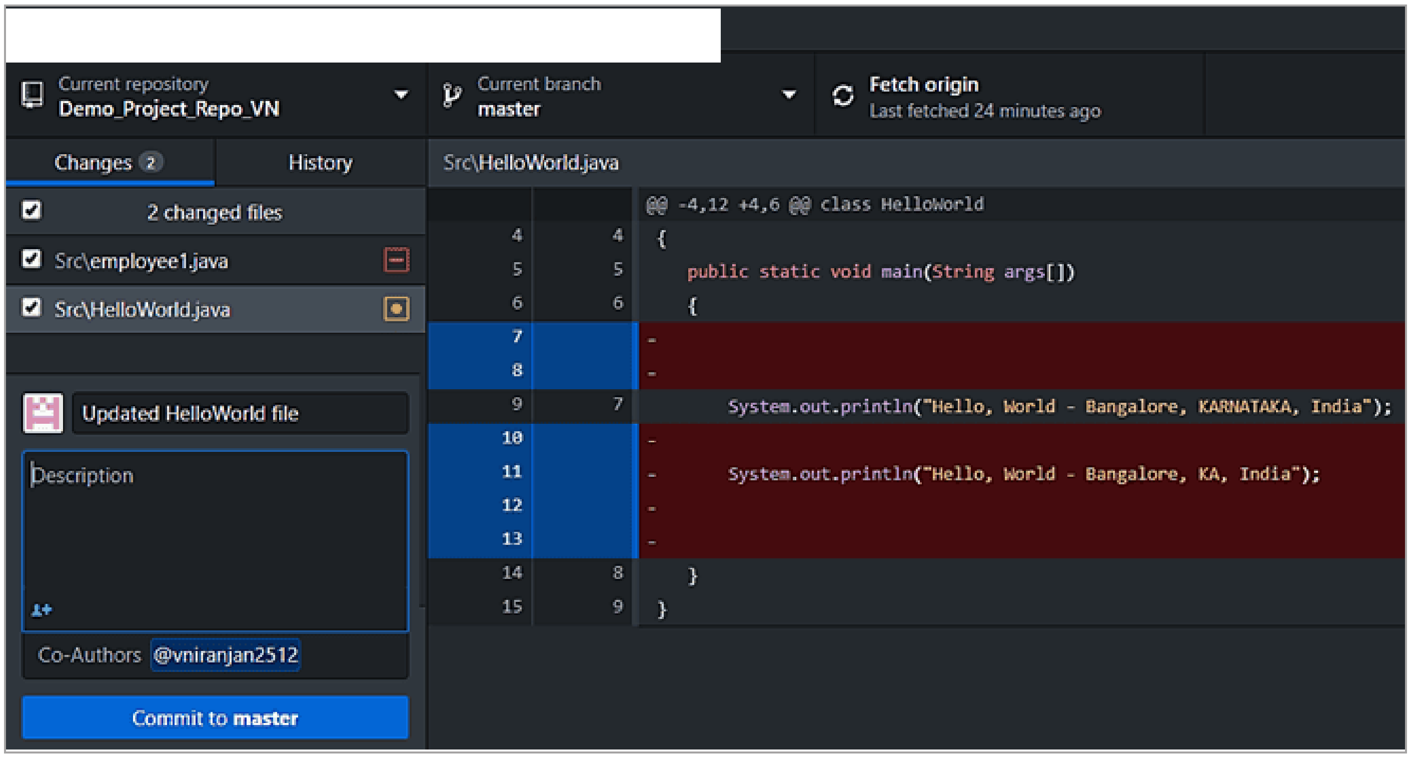
Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G
View on GitHub

Make changes to the files and save the same.


This PC > Local Disk (D:) > temp > GitHub > Demo_Project_Repo_VN > Src

Name	Date modified	Type	Size
employee.java	11/30/2019 12:16 ...	JAVA File	1 KB
HelloWorld.java	11/30/2019 12:25 ...	JAVA File	1 KB

Back in the GitHub Desktop, you can see the **RED** marking which specifies if the lines were added or were deleted.

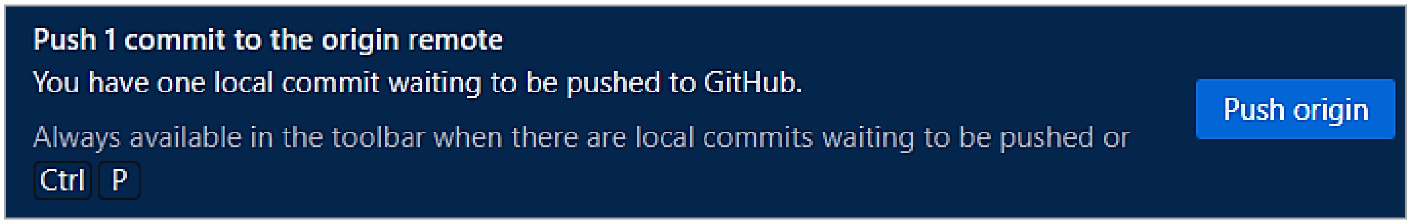


```
@@ -4,12 +4,6 @@ class HelloWorld
 {
    public static void main(String args[])
    {
-
-
        System.out.println("Hello, World - Bangalore, KARNATAKA, India");
-
-       System.out.println("Hello, World - Bangalore, KA, India");
-
-
    }
 }
```

Add a Summary and Co-authors if needed and click on **Commit to master** at the bottom.

You will notice that most of the git commands that you execute from the command prompt have been done through the user interface.

We can now push the changes to the remote repository in GitHub. Click on **Push origin.**



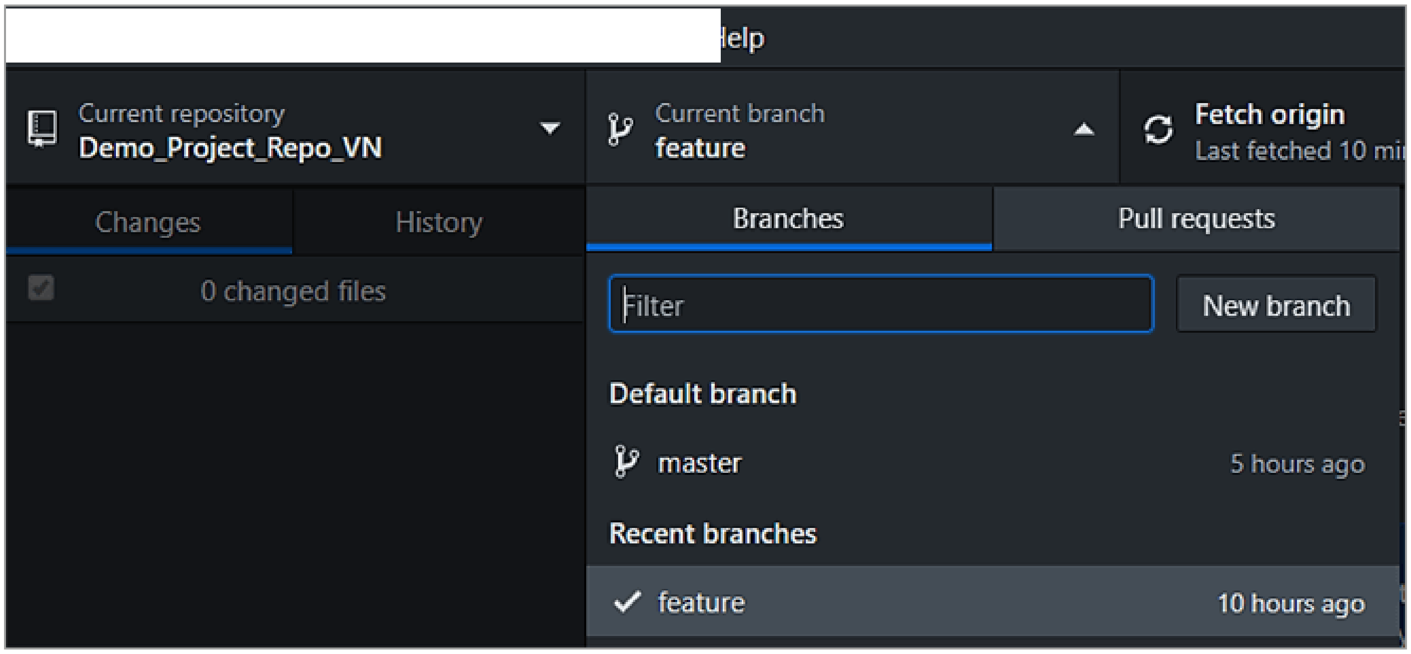
Push 1 commit to the origin remote

You have one local commit waiting to be pushed to GitHub.

Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Now the changes are visible in the master branch. To ensure that the changes are merged to the feature branch we will need to create a **Pull Request.**

Switch to the **feature** branch and create a **Pull Request.**


[redacted]elp

Current repository
Demo_Project_Repo_VN

Current branch
feature

Fetch origin
Last fetched 10 mi

Changes | History

0 changed files

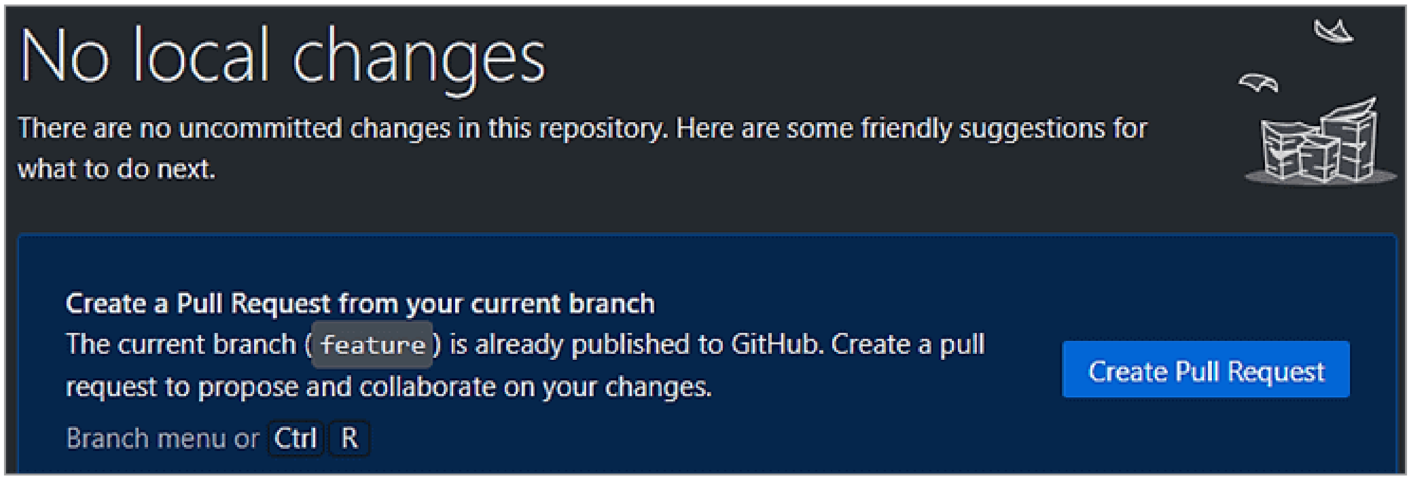
Branches | Pull requests

Filter | New branch

Default branch
master 5 hours ago

Recent branches
✓ feature 10 hours ago

Click on **Create Pull Request.**

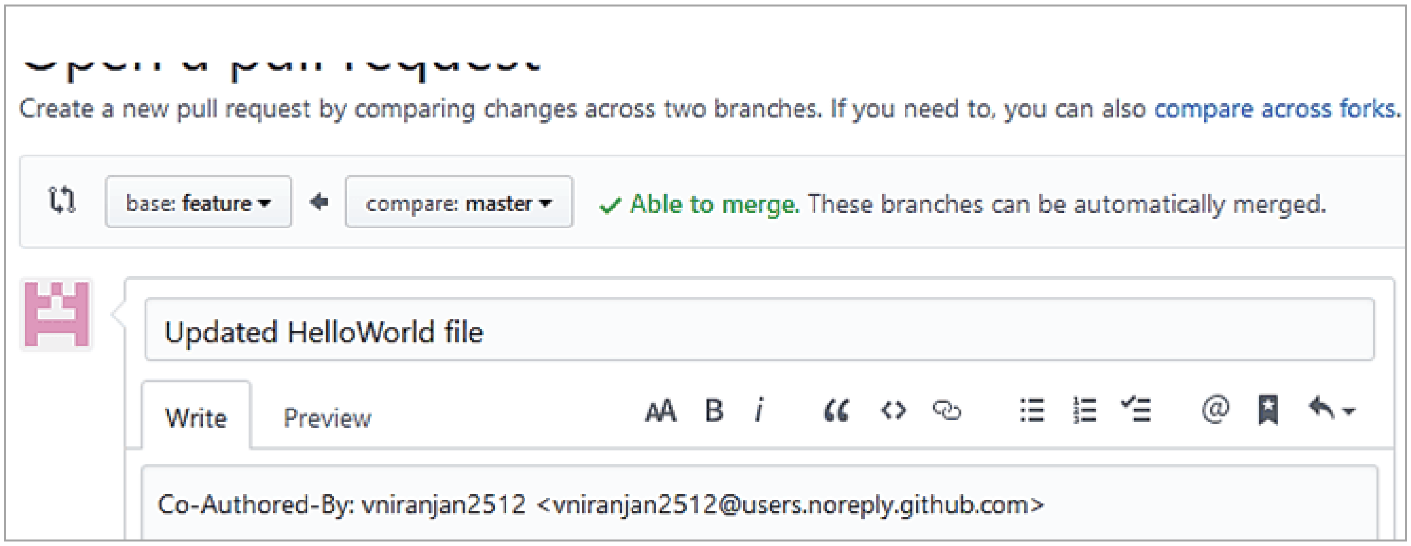

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Create a Pull Request from your current branch
The current branch (`feature`) is already published to GitHub. Create a pull request to propose and collaborate on your changes.
Branch menu or Ctrl R

Create Pull Request

You are then re-directed to GitHub to create the Pull Request.


Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

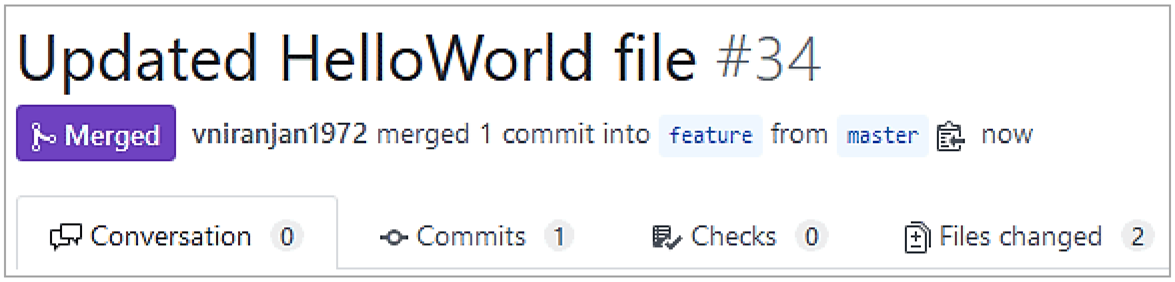
base: feature ← compare: master ✓ Able to merge. These branches can be automatically merged.

Updated HelloWorld file

Write Preview

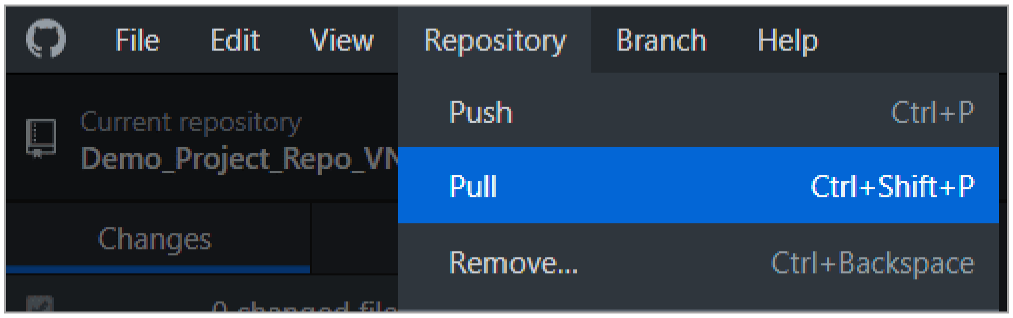
Co-Authored-By: vniranjan2512 <vniranjan2512@users.noreply.github.com>

Proceed to create and merge the Pull Request and then finally **pull**(sync) the changes to your local repository.


Updated HelloWorld file #34

Merged vniranjan1972 merged 1 commit into `feature` from `master` now

From the Repository, the menu selects the Pull option.

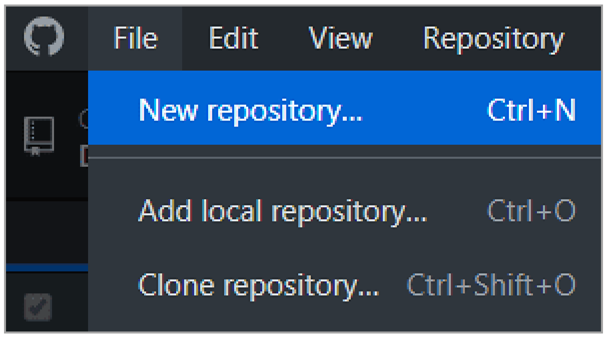


Now the local repository would be in sync with the remote repository.

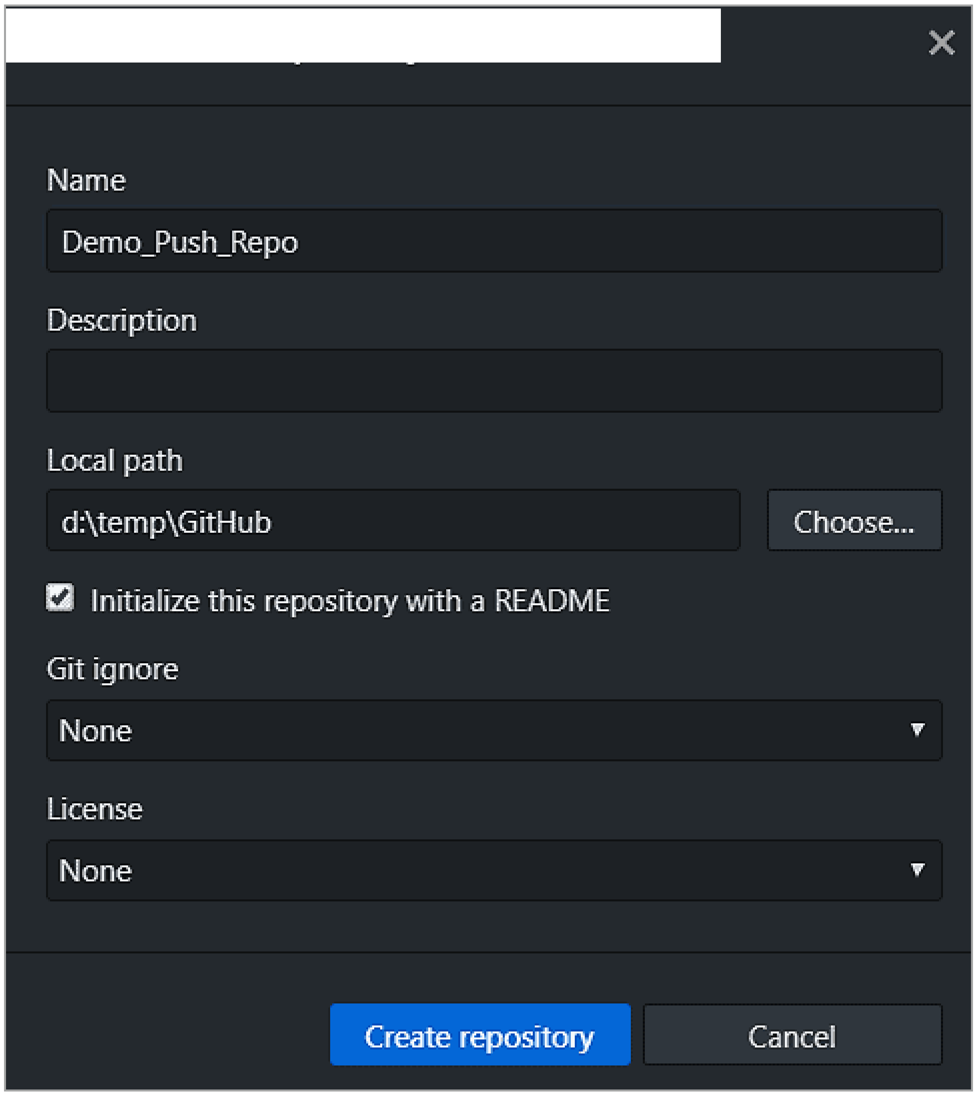
Create A New Local Repository And Branch

rking with the remote repository by cloning it. Using GitHub desktop, we can also create a new local repository and push or publish the same to GitHub.

Click on **File =>New Repository**


File
Edit
View
Repository
New repository... Ctrl+N
Add local repository... Ctrl+O
Clone repository... Ctrl+Shift+O

Enter the name of the repository and the local path. Click on **Create Repository.**



Name
Demo_Push_Repo
Description

Local path
d:\temp\GitHub

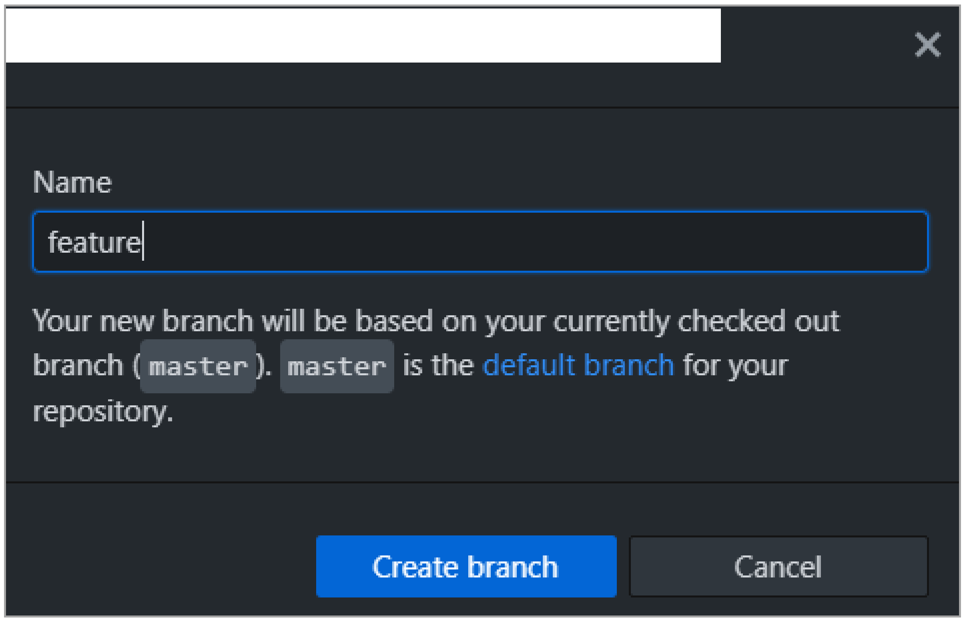
☑ Initialize this repository with a README

Git ignore
None

License
None

As the repository is created, you can also create a branch before you publish/push the changes to GitHub.

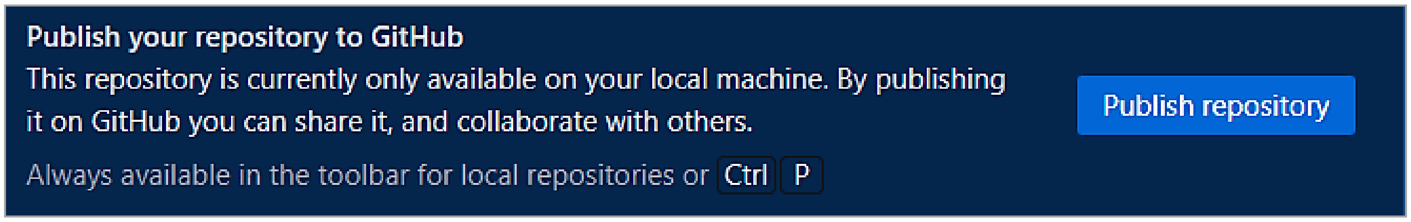
Select **New branch** from the **Branch** menu**.** Call it to **feature** and click on **Create branch**.


Name

feature

Your new branch will be based on your currently checked out branch (`master`). `master` is the default branch for your repository.

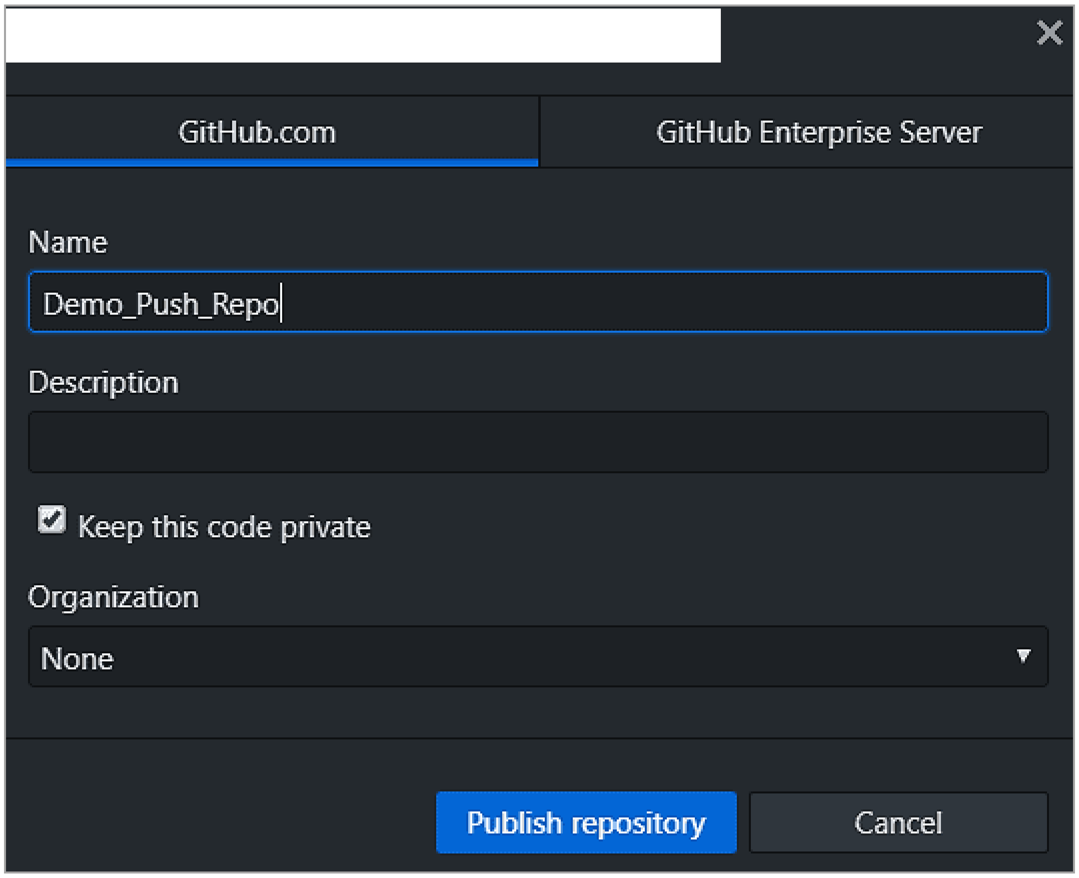
Now we do have 2 branches and we can proceed to Publish / Push the changes to GitHub. Click on **Publish repository.**


Publish your repository to GitHub

This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.

Always available in the toolbar for local repositories or Ctrl P

Click on **Publish Repository.**



Name

Demo_Push_Repo

Description

[✓ Keep this code private]

Organization

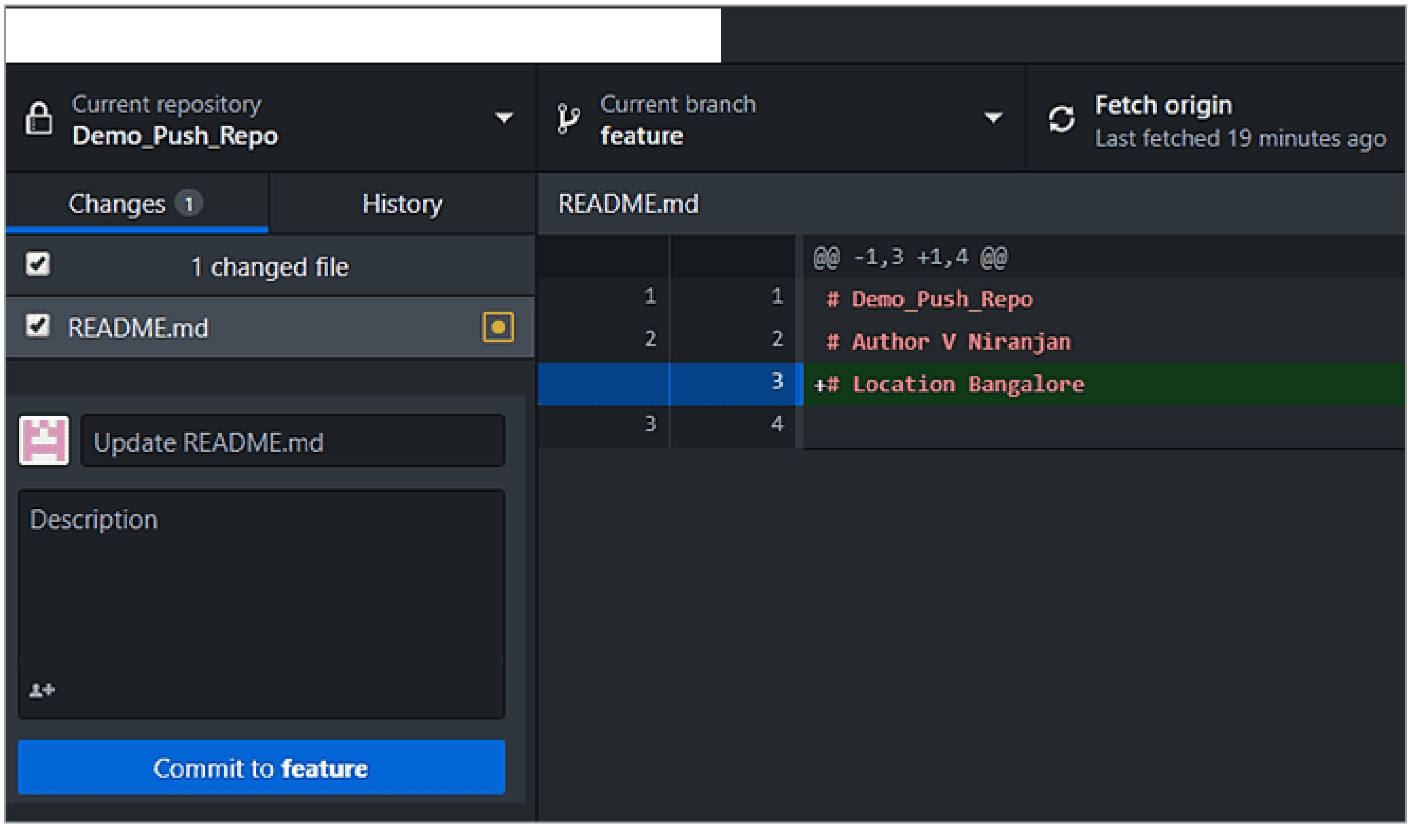
None

As there is also a Feature branch, you must publish the feature branch as well. Now the changes can be made to the files locally and then push the changes to the remote repository. Changes in the remote repository should also be in sync with the local repository.

Merge Changes In Local Repository

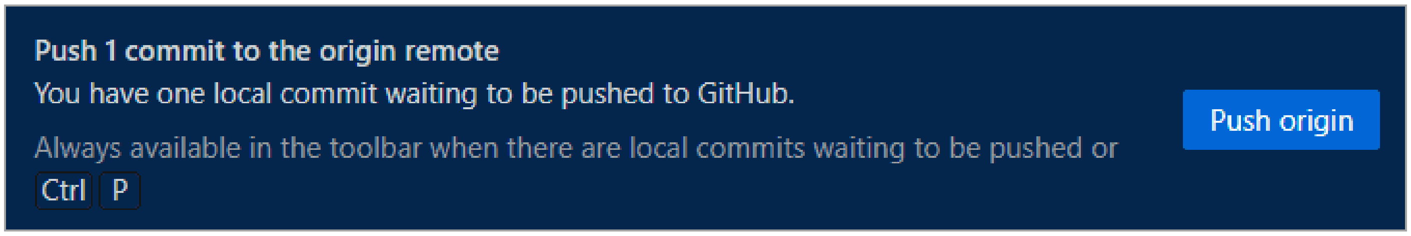
Assume that there are changes in the feature branch in the local repository. We can merge the changes to the master branch. Post this we should push the changes of the master and feature branch to GitHub.

Make a change to a file in the feature branch and commit the same.



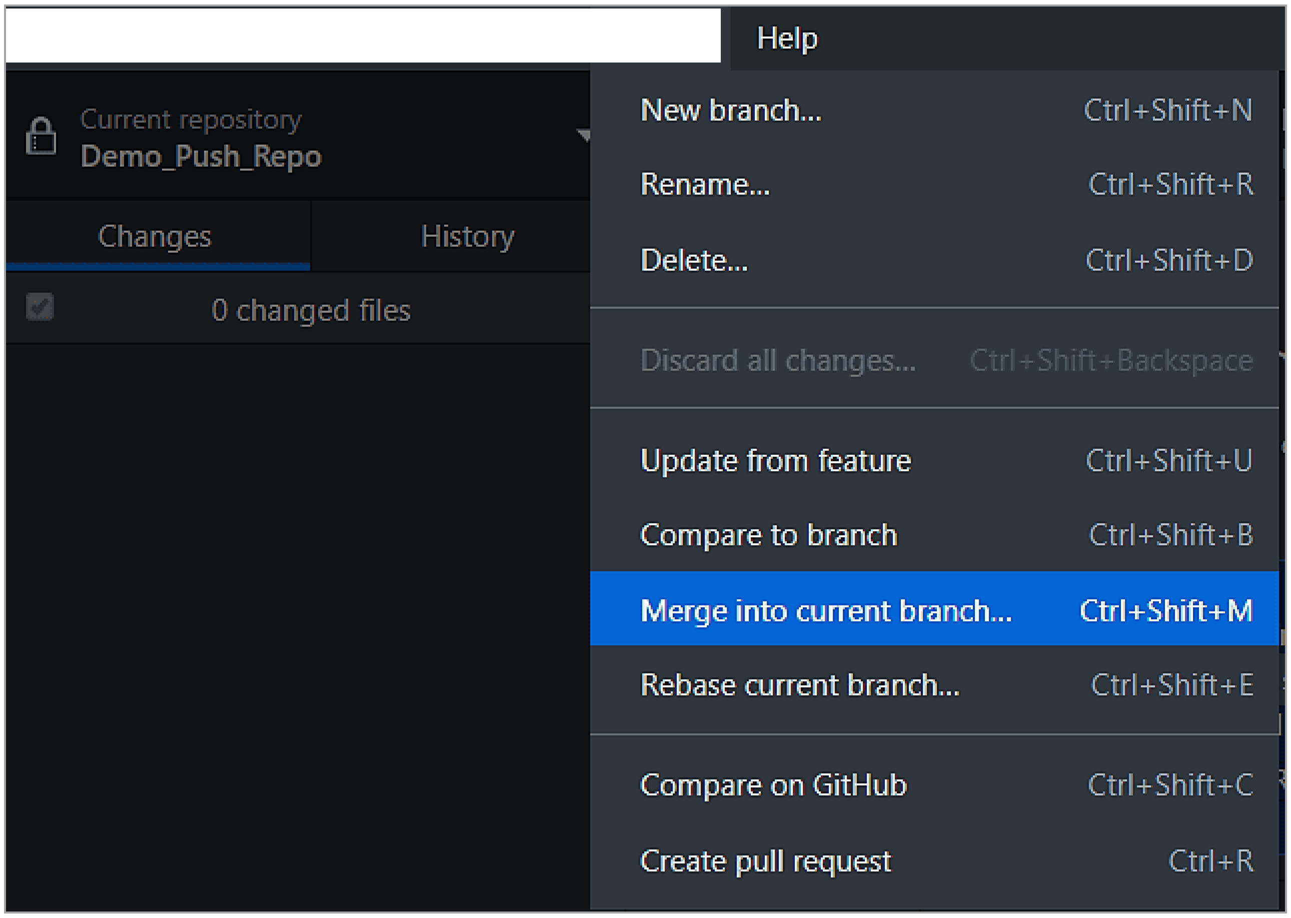
```
@@ -1,3 +1,4 @@
1  1   # Demo_Push_Repo
2  2   # Author V Niranjan
   3  +# Location Bangalore
3  4
```

Push the changes to the remote repository.

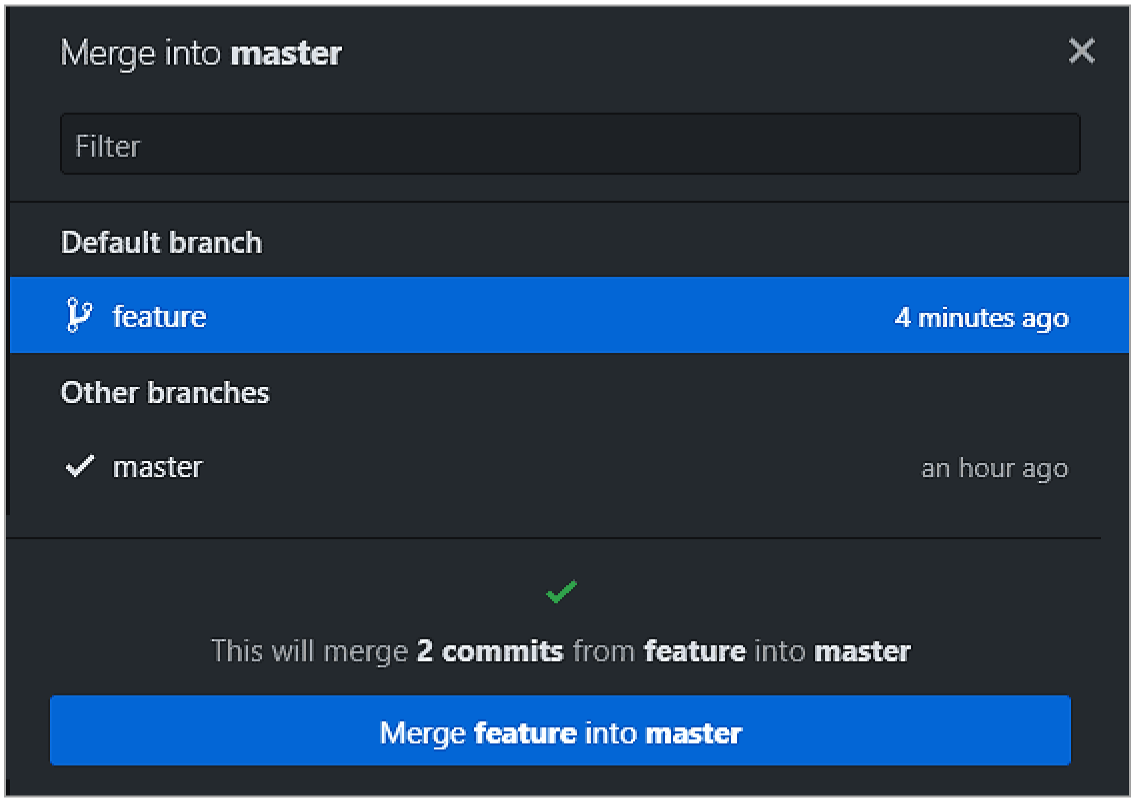


Push 1 commit to the origin remote
You have one local commit waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Switch to the Master branch and click on **Branch =>Merge into the current branch.**

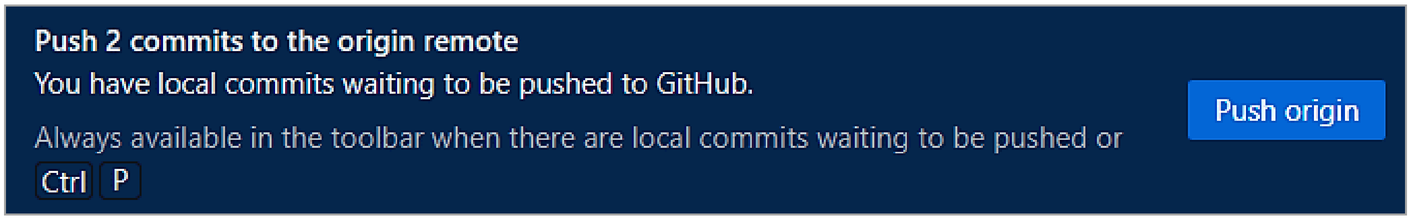

Help
Current repository
Demo_Push_Repo
Changes
History
0 changed files
New branch... Ctrl+Shift+N
Rename... Ctrl+Shift+R
Delete... Ctrl+Shift+D
Discard all changes... Ctrl+Shift+Backspace
Update from feature Ctrl+Shift+U
Compare to branch Ctrl+Shift+B
Merge into current branch... Ctrl+Shift+M
Rebase current branch... Ctrl+Shift+E
Compare on GitHub Ctrl+Shift+C
Create pull request Ctrl+R

Select the **Feature branch** which is the source branch. Click on the **Merge button**.



This will merge **2 commits** from **feature** into **master** r branch, you can then push the changes to the remote repository to be in sync.

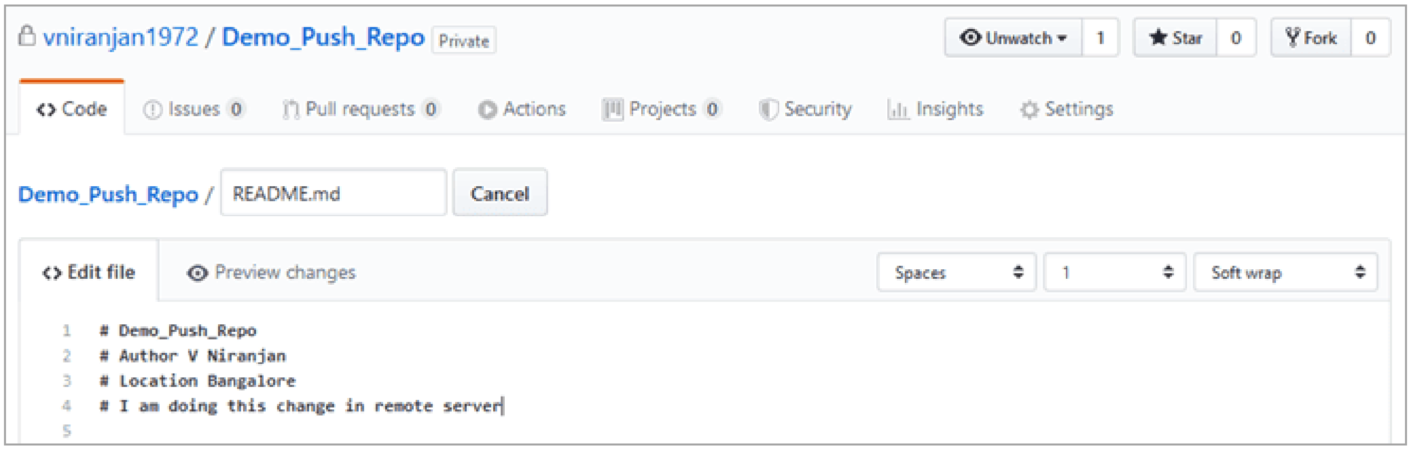
All the changes committed to branches in the local repository can be merged and pushed to the remote repository to be in sync.


Push 2 commits to the origin remote
You have local commits waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Resolving Conflicts

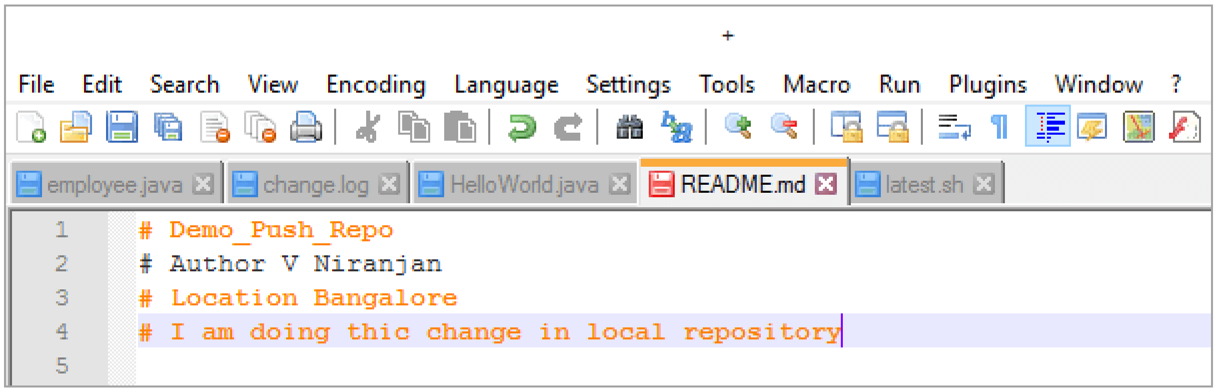
There could also be a scenario where the changes have been committed to a file in the remote repository and also a change to the same file locally. In this case, the conflicts would be seen and would need to be resolved to have both the remote and local repository to be in sync.

Remote repository changes committed in the Master branch



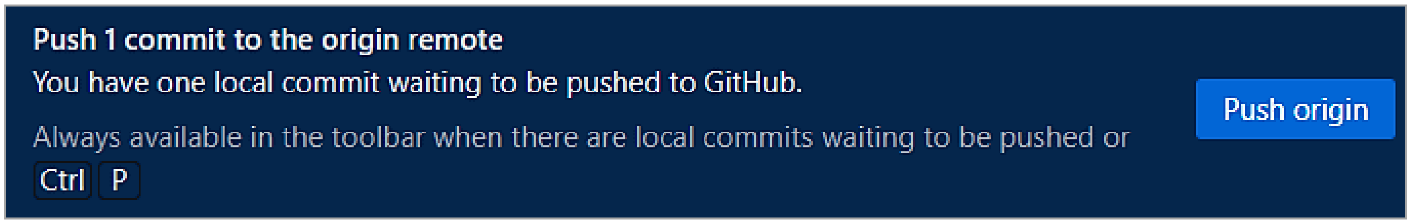
```
# Demo_Push_Repo
# Author V Niranjan
# Location Bangalore
# I am doing this change in remote server
```

Local repository changes committed in the master branch


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
employee.java
change.log
HelloWorld.java
README.md
latest.sh

```
# Demo_Push_Repo
# Author V Niranjan
# Location Bangalore
# I am doing thic change in local repository
```

As the changes are committed to the local repository, you can now push the changes to the remote repository. The conflicts will be seen while doing this. Click on **Push origin.**

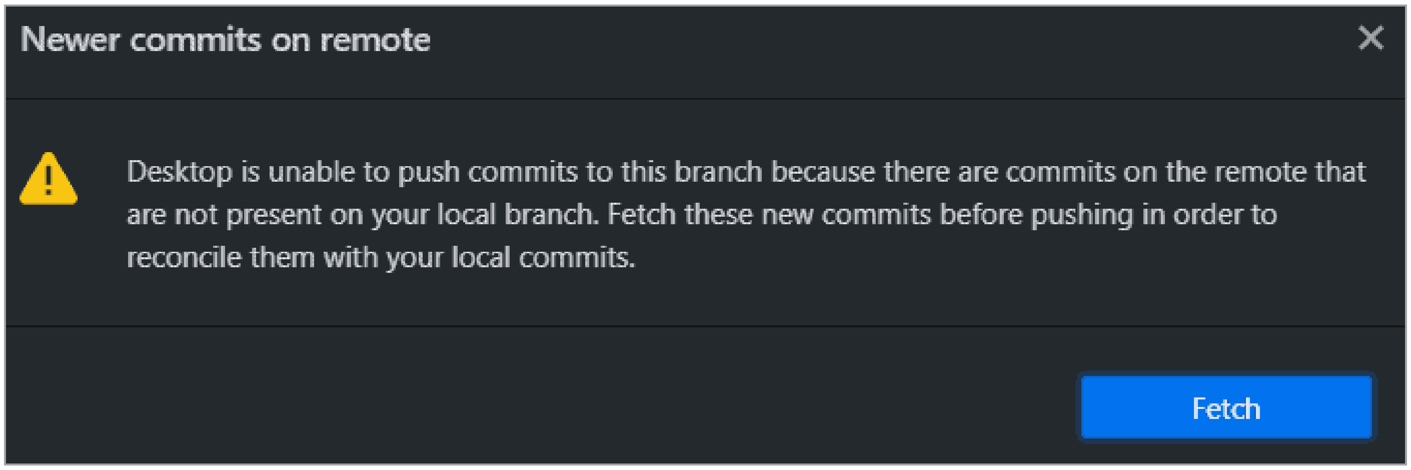


Push 1 commit to the origin remote

You have one local commit waiting to be pushed to GitHub.

Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

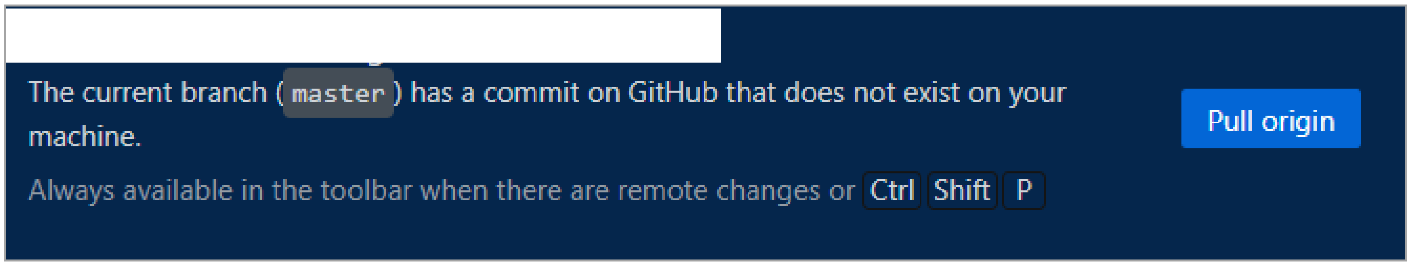
The following message would appear as there are changes in the remote repository to the same file. Click on **Fetch.**



Newer commits on remote

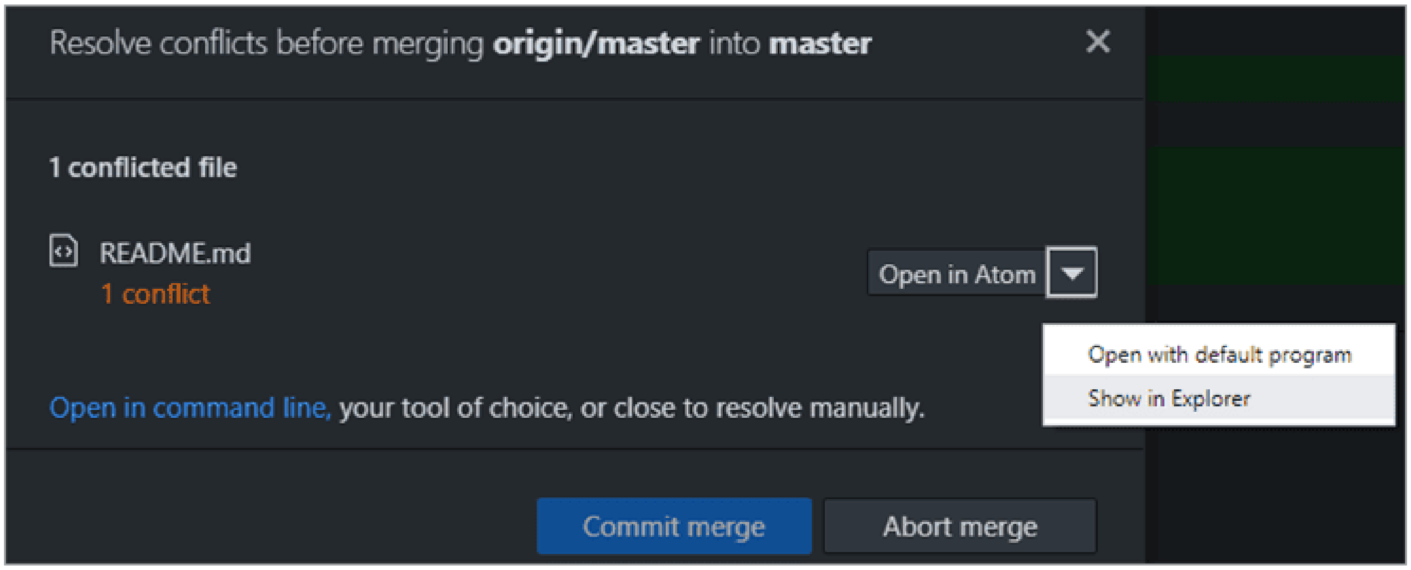
Desktop is unable to push commits to this branch because there are commits on the remote that are not present on your local branch. Fetch these new commits before pushing in order to reconcile them with your local commits.

Now click on **Pull origin.**


The current branch (`master`) has a commit on GitHub that does not exist on your machine.

Always available in the toolbar when there are remote changes or Ctrl Shift P

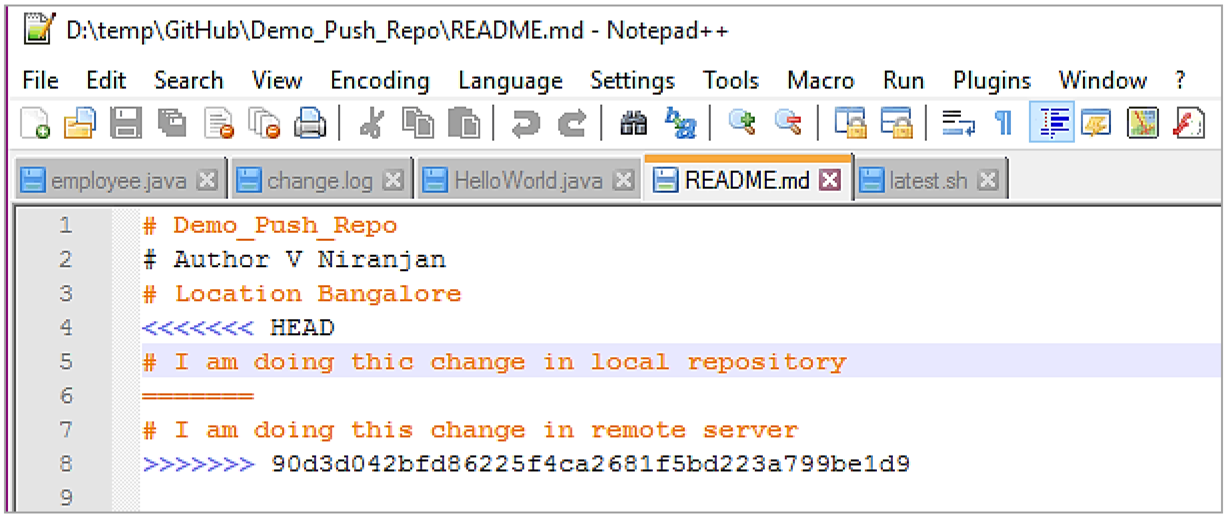
In the screen that comes up, you can open the file in your editor and resolve the conflicts. In this case, we are opening the file in explorer and resolving the conflicts.


Resolve conflicts before merging **origin/master** into **master**

1 conflicted file

README.md
1 conflict

Open in command line, your tool of choice, or close to resolve manually.

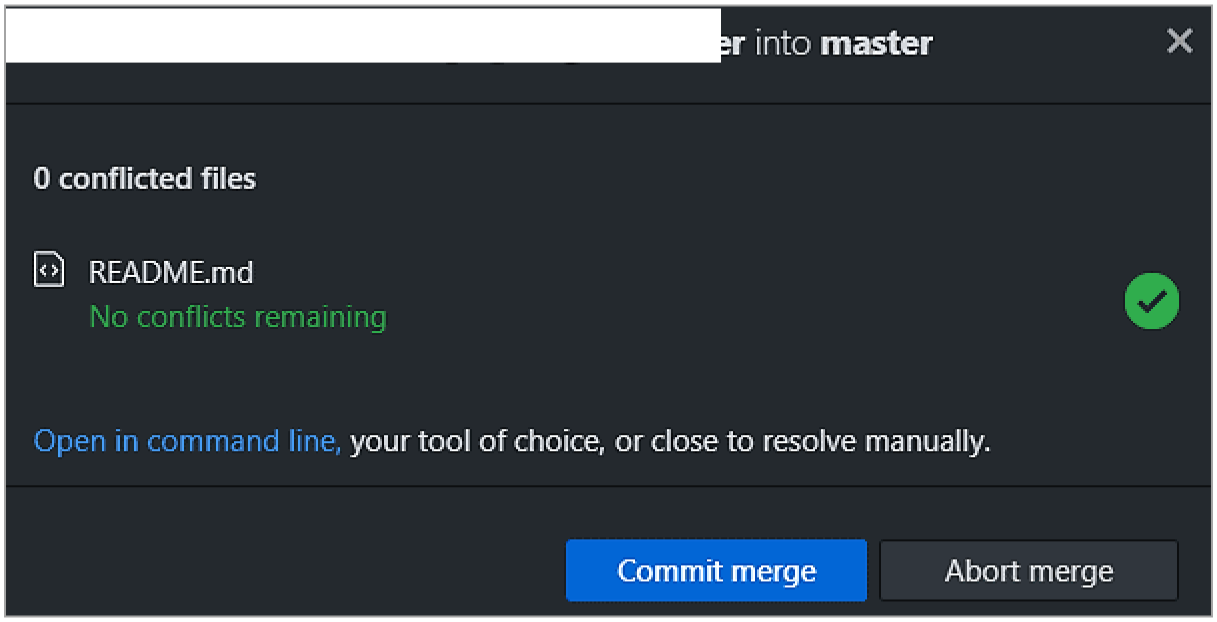

D:\temp\GitHub\Demo_Push_Repo\README.md - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
employee.java
change.log
HelloWorld.java
README.md
latest.sh

```
# Demo_Push_Repo
# Author V Niranjan
# Location Bangalore
<<<<<<< HEAD
# I am doing thic change in local repository
=======
# I am doing this change in remote server
>>>>>>> 90d3d042bfd86225f4ca2681f5bd223a799be1d9

```

Fix all of the conflicts by retaining the appropriate content and removing the others with markers. Once the conflicts are resolved, you can commit the merge.

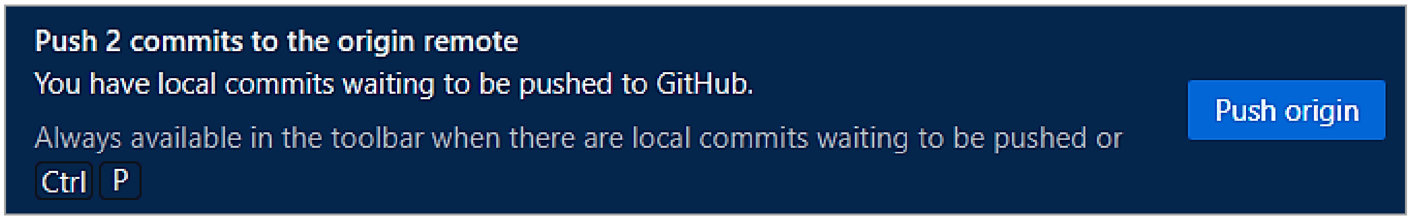

[redacted] er into **master**

0 conflicted files

README.md
No conflicts remaining

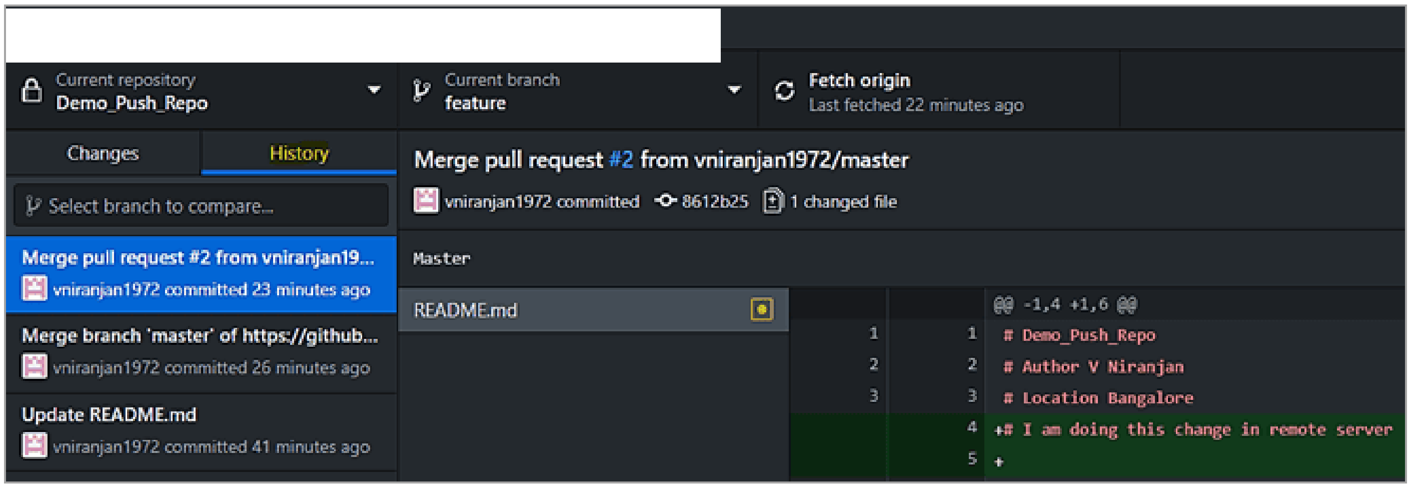
Open in command line, your tool of choice, or close to resolve manually.

Now push the changes back to the remote repository. The local and remote repository is now in sync. As the changes have been done on one branch you can then create a Pull Request to merge the changes to the other branches.


Push 2 commits to the origin remote
You have local commits waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Looking At History

You can also look at the history of changes done so far to the repository. Toggle to the **History tab**.



Merge pull request #2 from vniranjan19...
vniranjan1972 committed 23 minutes ago
Merge branch 'master' of https://github...
vniranjan1972 committed 26 minutes ago
Update README.md
vniranjan1972 committed 41 minutes ago

Merge pull request #2 from vniranjan1972/master
vniranjan1972 committed 8612b25 1 changed file

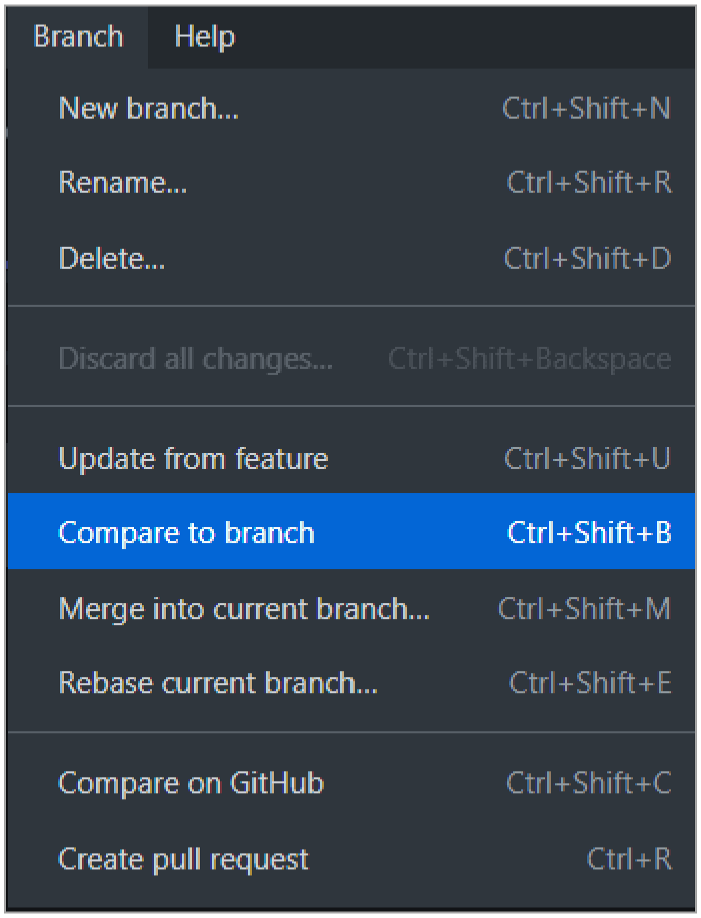
Master

README.md

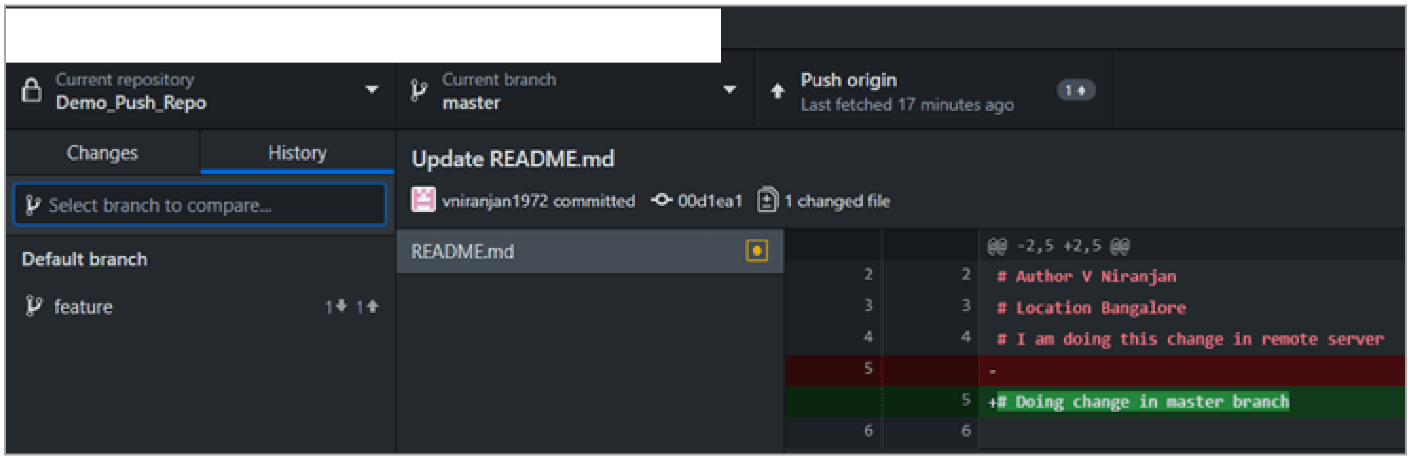
```
@@ -1,4 +1,6 @@
 # Demo_Push_Repo
 # Author V Niranjan
 # Location Bangalore
+# I am doing this change in remote server
+
```

Comparing Branches

Suppose you have done changes to a file in the master branch, you can then compare it with any of the other branches. Select **Branch => Compare to branch**.


Branch
Help
New branch... Ctrl+Shift+N
Rename... Ctrl+Shift+R
Delete... Ctrl+Shift+D
Discard all changes... Ctrl+Shift+Backspace
Update from feature Ctrl+Shift+U
Compare to branch Ctrl+Shift+B
Merge into current branch... Ctrl+Shift+M
Rebase current branch... Ctrl+Shift+E
Compare on GitHub Ctrl+Shift+C
Create pull request Ctrl+R

Select the feature branch to look at the changes.



```
@@ -2,5 +2,5 @@
 # Author V Niranjan
 # Location Bangalore
 # I am doing this change in remote server
-
+# Doing change in master branch
```

Conclusion

Though the use of Git commands from the command line is great, we saw in this GitHub Desktop tutorial, how a great Git Client like GitHub Desktop with a good user interface can ease the developer's work while working with the local and remote repositories.

In the upcoming tutorial, we will look at another Git client interface Tortoise Git that integrates with the Windows Explorer Shell.

=> **Watch Out The Simple GitHub Training Series Here.**

Recommended Reading

- GitHub Tutorial For Developers | How To Use GitHub
- PowerShell UIAutomation Tutorial: UI Automation of Desktop Applications
- GitHub REST API Tutorial - REST API Support In GitHub
- Advanced Git Commands And GitHub Integration Tutorial
- Tortoise SVN Tutorial: Revisions In Code Repository
- How To Delete Content From SVN Repository
- GitLab Jira Integration Tutorial
- How To Use GitHub Extension For Microsoft Visual Studio?