

# Assignment on the Git Training Session

Task 1 : Create a git repo and host it on any public git hosting service [github / gitlab / bitbucket ] Clone that repo , create a branch , add a readme file and push the changes to the branch and get it merged by your mentor.

Creating a git repo and host it on any public git hosting service :

Step 1: Go to <https://github.com>

Step 2: Sign in to your account and create a new repository. Choose according to the preferences whether to make the repository public or private.

Clone that repo , create a branch , add a readme file and push the changes to the branch.

Step 1 : Clone that repo on the terminal with the command `git clone <url-to-the repo>`

Step 2 : Initialize git in that directory created by cloning this repository

Step 3 : Use `git init` to Initialize the directory.

Step 4 : Create a branch using the command `git checkout -m <branch-name>`

Step 5 : Now create the readme file. Do some changes in it.

Step 6 : Add it to the staging area using `git add` .

Step 7 : Commit the changes using `git commit -m "message"`

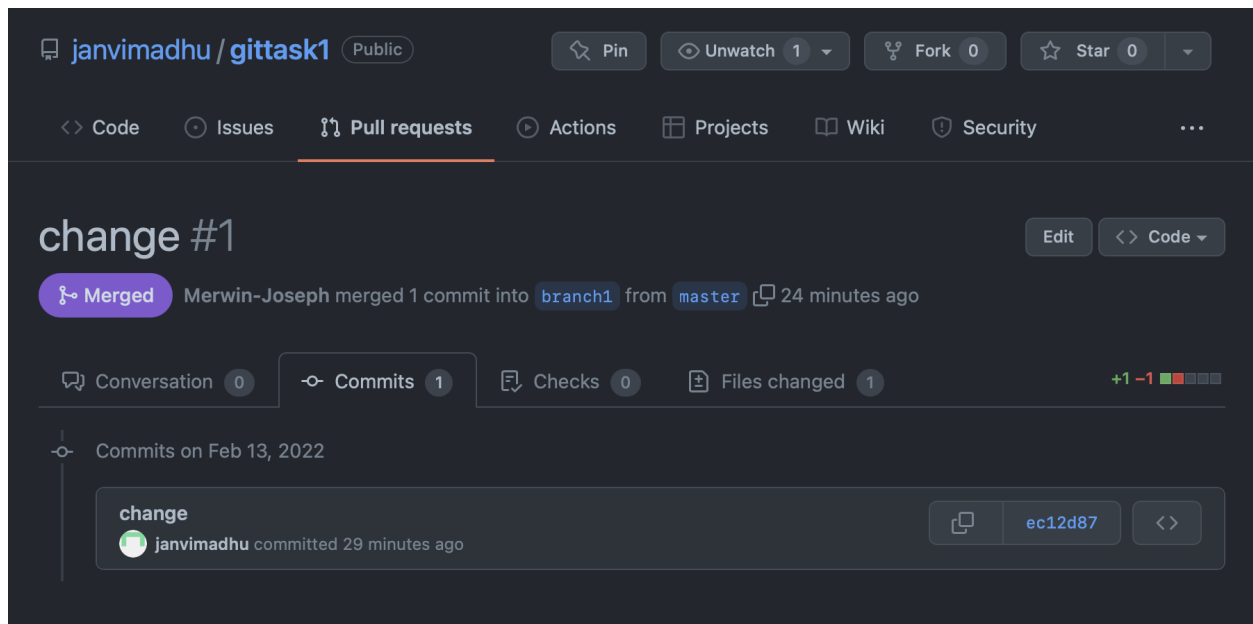
Step 8 : Now push the changes to github repository using `git push origin <branch-name>`

Now the changes will be reflected on the github UI.

Once the branch is pushed we should share the pull request url to our mentor for merging it.

To get it merged with the mentor we are supposed to add our mentor as a collaborator which can be done by going to settings and under collaborators send your mentor the request to collaborate to your repository.

After adding them as a collaborator, create a pull request for the branch and share the remote link to the collaborator and they will be able to merge it. Or they will see it under the pull request and be able to merge it.



## Task 2 : Simulate a merge conflict and try to resolve it.

Mostly git automatically figures out how to integrate new changes. Conflict generally arises when two people have changed the same line in a file , or if one developer deleted a file while another developer was modifying it.

So , to stimulate this , let's just take the example of line change merge conflict.

Step 1 : Create a new directory using the mkdir command and cd into that directory.

Step 2 : Initialize it as git repo with the git init command.

Step 3 : Create a text file using the touch command.

Step 4 : Add some text in the text file and add it in the repo and commit it.

Step 5 : Create a new branch which will be used to create the conflicting merge by using git checkout command.

Step 6 : Overwrite the existing text file.

Step 7 : Add the changes to the git repo and commit it.

Step 8 : Checkout to the master branch and append the text file.

Step 9: Again , add those changes and commit them.

Step 10: Merge the new branch to the master branch now and you will encounter the merge conflict.

```
merge_conflict_test — -zsh — 125x67

Last login: Fri Feb 11 00:00:06 on ttys001
phonepe@PP-C02FX1QZMD6M ~ % cd Desktop
phonepe@PP-C02FX1QZMD6M Desktop % mkdir merge_conflict_test
phonepe@PP-C02FX1QZMD6M Desktop % cd merge_conflict_test
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/phonepe/Desktop/merge_conflict_test/.git/
phonepe@PP-C02FX1QZMD6M merge_conflict_test % touch test_file.txt
phonepe@PP-C02FX1QZMD6M merge_conflict_test % echo "Add some content so that you can mess with it" > test_file.txt
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git add test_file.txt
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git commit -m "Initial commit"
[master (root-commit) 8a68487] Initial commit
1 file changed, 1 insertion(+)
 create mode 100644 test_file.txt
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git checkout -b merge_conflict_branch
Switched to a new branch 'merge_conflict_branch'
phonepe@PP-C02FX1QZMD6M merge_conflict_test % echo "Changing the contents of the text file from the branch" > test_file.txt
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git add .
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git commit -m "edited the text file to cause a conflict"
[merge_conflict_branch 7d9b2c4] edited the text file to cause a conflict
1 file changed, 1 insertion(+), 1 deletion(-)
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git checkout master
Switched to branch 'master'
phonepe@PP-C02FX1QZMD6M merge_conflict_test % echo "Append this to the initial commit" >> test_file.txt
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git add .

phonepe@PP-C02FX1QZMD6M merge_conflict_test % git commit -m "appended some text to initial commit"
[master 1f041ae] appended some text to initial commit
1 file changed, 1 insertion(+)
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git merge merge_conflict_branch
Auto-merging test_file.txt
CONFLICT (content): Merge conflict in test_file.txt
Automatic merge failed; fix conflicts and then commit the result.
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

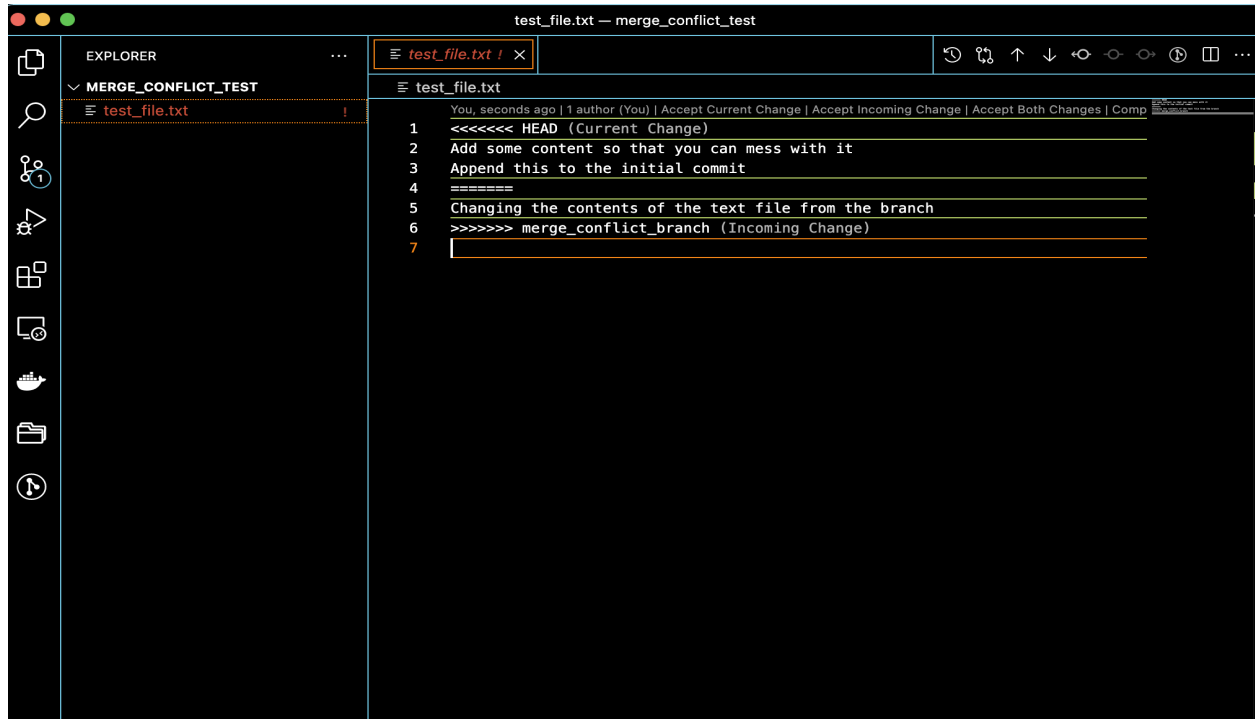
Unmerged paths:
  (use "git add <file>..." to mark resolution)
[
    both modified:   test_file.txt
```

After the merge conflict is encountered , with the git status command.

Now since the conflict is identified by using the git status command so now comes the part where we need to resolve the issue.

Resolving Part:

when we open the text file , we will see some conflict dividers as shown in the below picture.



```
===== -center conflict
<<<<<<<<<HEAD -current master branch
>>>>>>>>>branch_name -merging branch
```

To resolve the merge conflict we have to manually remove the unnecessary information and change the file as per the requirement , also we have to remove these conflict dividers from our file. Once this is done we just have to use git add command to add te changes and commit the changes.

```
[phonepe@PP-C02FX1QZMD6M merge_conflict_test % git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
[
    both modified:   test_file.txt

no changes added to commit (use "git add" and/or "git commit -a")
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git add .
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git commit -m "Resolved the merge conflict"
[master b5b2765] Resolved the merge conflict
phonepe@PP-C02FX1QZMD6M merge_conflict_test % git merge merge_conflict_branch
Already up to date.
phonepe@PP-C02FX1QZMD6M merge_conflict_test %
```

After that our text will look like :

```
test_file.txt
You, seconds ago | 1 author (You)
1
2 Add some content so that you can mess with it
3 Append this to the initial commit
4 Changing the contents of the text file from the branch | You, seconds ago
5
```




Conflict is Resolved and the branches are merged now.

Git Graph — merge\_conflict\_test

EXPLORER

- MERGE\_CONFLICT\_TEST
  - test\_file.txt

Branches: Show All ☒ Show Remote Branches

Graph	Description	Date	Author	Commit
	 master Resolved the merge conflict	11 Feb 2022 00:31	Janvi	b5b27650
	appended some text to initial commit	11 Feb 2022 00:24	Janvi	1f041ae2
	 merge_conflict_branch edited the text file to cause a conflict	11 Feb 2022 00:21	Janvi	7d9b2c4f
	Initial commit	11 Feb 2022 00:20	Janvi	8a68487e

Task 3 : Create 2 git repositories , Project-A with Readme file and Project-B with empty repository. Write a script to pull the latest changes from Project-A/Readme file and push it to Project-B/Readme with a proper commit message each time there is a change .

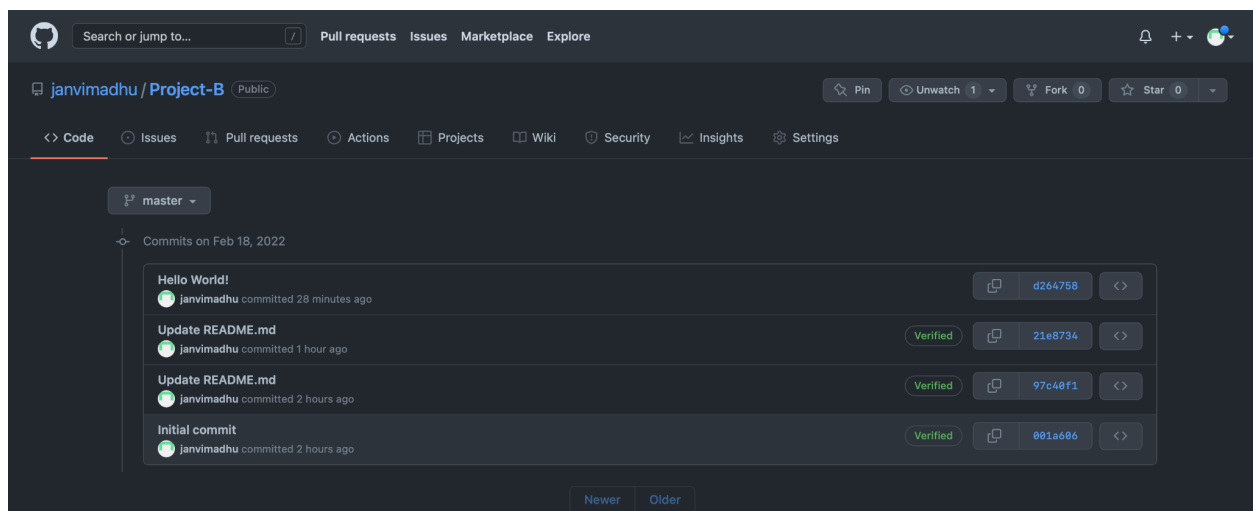
Step 1: Create two git repositories Project-A and Project-B .

Step 2: Initialize Readme.md file in Project-A and an empty directory in Project-B.

Step 3: Create a Script which will fetch the latest changes from Project-A and and push it to Project-B.

```
gitautomate.sh — Project-A
$ gitautomate.sh
1 #!/bin/bash
2 git pull https://github.com/janvimadhu/Project-A.git
3
4 git remote set-url origin https://github.com/janvimadhu/Project-B.git
5
6 git push origin master --force
7
8
```

Step 4: Run the Script using bash gitautomate.sh command and you will see the latest changes in Project-A will be updated to Project-B.



Created By - Janvi

