Working in one branch:

YouTube tutorial: https://www.youtube.com/watch?v=ygqx50-JHEE

C:/Desktop or c:/downloads (stay in the main dir)

Git clone <repo_link> <folder_name_for_project>

Ex: git clone https://github.com/udyancinntra/rms.git rms_yudiudyan1

Make changes or lalalala

When you are ready to make changes:

git add .

git commit –m "sensible messages"

git push –u origin <branch_name_of_repo>

Ex: git push –u origin main

check the current repo link and branch for github:

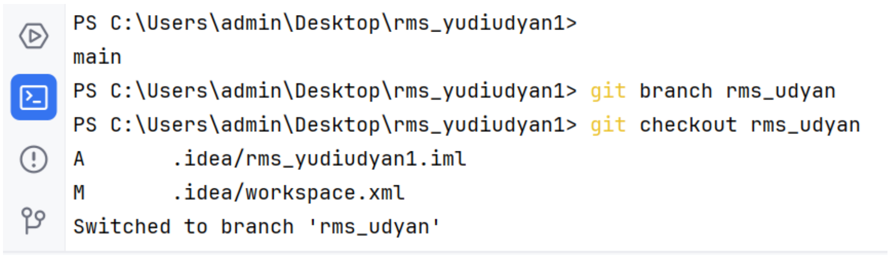
git remote –v (helpful when we want to verify the existing repository, optional)

git branch --show-current (helpful when we want to verify the existing repository, optional)

Creating a new branch and working in that:

git branch rms_udyan

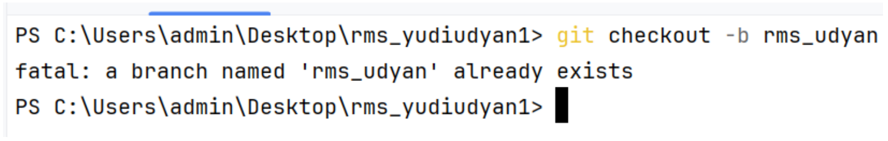
git checkout rms_udyan



```
PS C:\Users\admin\Desktop\rms_yudiudyan1>
main
PS C:\Users\admin\Desktop\rms_yudiudyan1> git branch rms_udyan
PS C:\Users\admin\Desktop\rms_yudiudyan1> git checkout rms_udyan
A       .idea/rms_yudiudyan1.iml
M       .idea/workspace.xml
Switched to branch 'rms_udyan'
```

OR, we can alternatively combine both the commands to achieve it in one line.

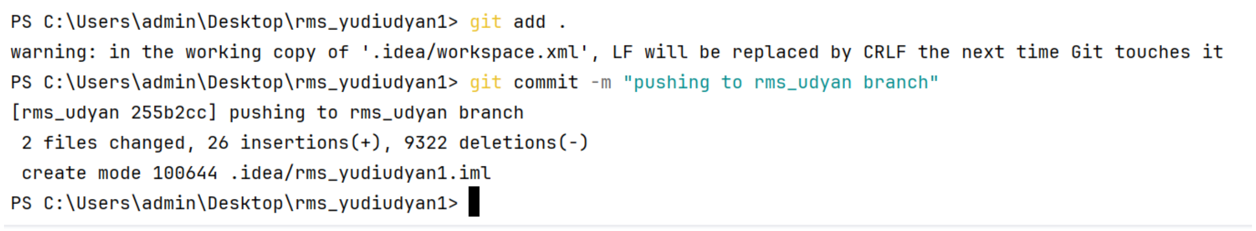
git checkout -b rms_udyan



```
PS C:\Users\admin\Desktop\rms_yudiudyan1> git checkout -b rms_udyan
fatal: a branch named 'rms_udyan' already exists
PS C:\Users\admin\Desktop\rms_yudiudyan1>
```

git add .

git commit –m "pushing to rms_udyan branch"

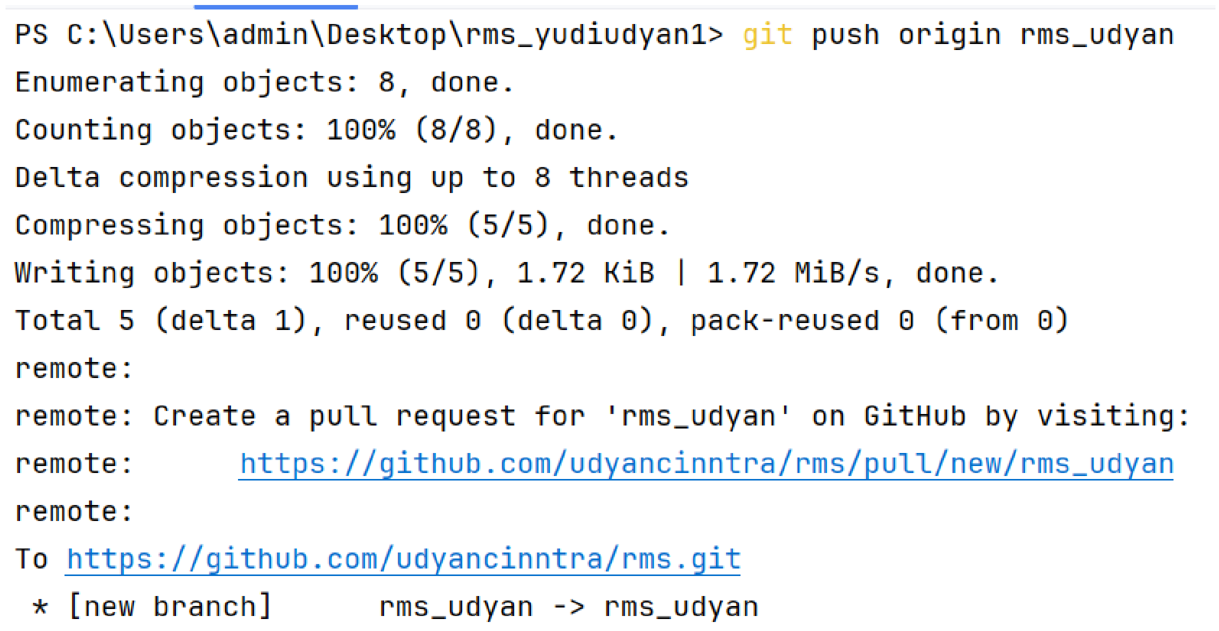


```
PS C:\Users\admin\Desktop\rms_yudiudyan1> git add .
warning: in the working copy of '.idea/workspace.xml', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\admin\Desktop\rms_yudiudyan1> git commit -m "pushing to rms_udyan branch"
[rms_udyan 255b2cc] pushing to rms_udyan branch
 2 files changed, 26 insertions(+), 9322 deletions(-)
 create mode 100644 .idea/rms_yudiudyan1.iml
PS C:\Users\admin\Desktop\rms_yudiudyan1>
```

git push –u origin <second_branch_name_of_repo>

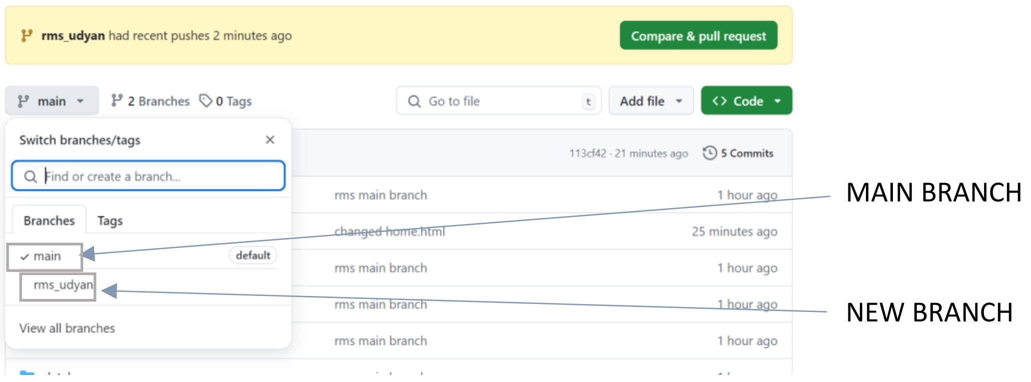
Ex: git push –u origin rms_udyan

You should get messages as below:



```
PS C:\Users\admin\Desktop\rms_yudiudyan1> git push origin rms_udyan
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.72 KiB | 1.72 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rms_udyan' on GitHub by visiting:
remote:      https://github.com/udyancinntra/rms/pull/new/rms_udyan
remote:
To https://github.com/udyancinntra/rms.git
 * [new branch]      rms_udyan -> rms_udyan
```

Now to go your repository to verify if there are two branches or not.



rms_udyan had recent pushes 2 minutes ago

113cf42 · 21 minutes ago · 5 Commits

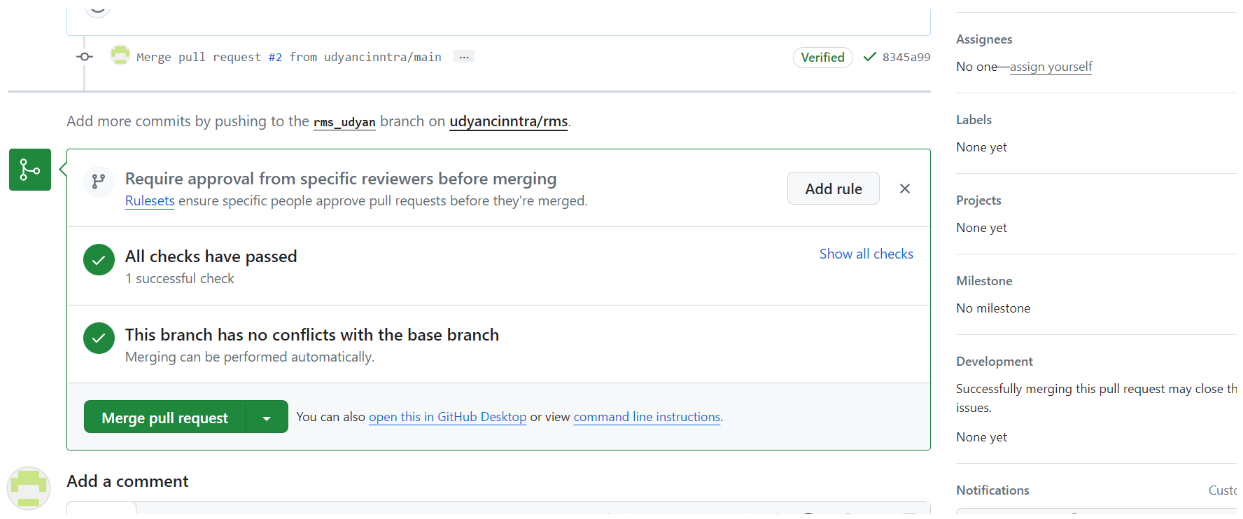
rms main branch · 1 hour ago
changed home.html · 25 minutes ago
rms main branch · 1 hour ago
rms main branch · 1 hour ago
rms main branch · 1 hour ago

Merging Conflicts:

If Git detects conflicts, it will notify you. Conflicts occur when the same lines of a file have been modified differently in both branches.

- Open the conflicted files in your code editor. Git will mark the conflicting sections in the files with conflict markers (<<<<<<<, =======, >>>>>>>).
- Edit the files to resolve conflicts, keeping the changes you want to keep.
- Remove the conflict markers once you've resolved the conflicts.

A successful merge and conflict resolution would look like:



Merge pull request #2 from udyancinntra/main … Verified ✓ 8345a99

Add more commits by pushing to the rms_udyan branch on udyancinntra/rms.

Require approval from specific reviewers before merging
Rulesets ensure specific people approve pull requests before they're merged.

All checks have passed
1 successful check

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

You can also open this in GitHub Desktop or view command line instructions.

Add a comment

Assignees
No one—assign yourself

Labels
None yet

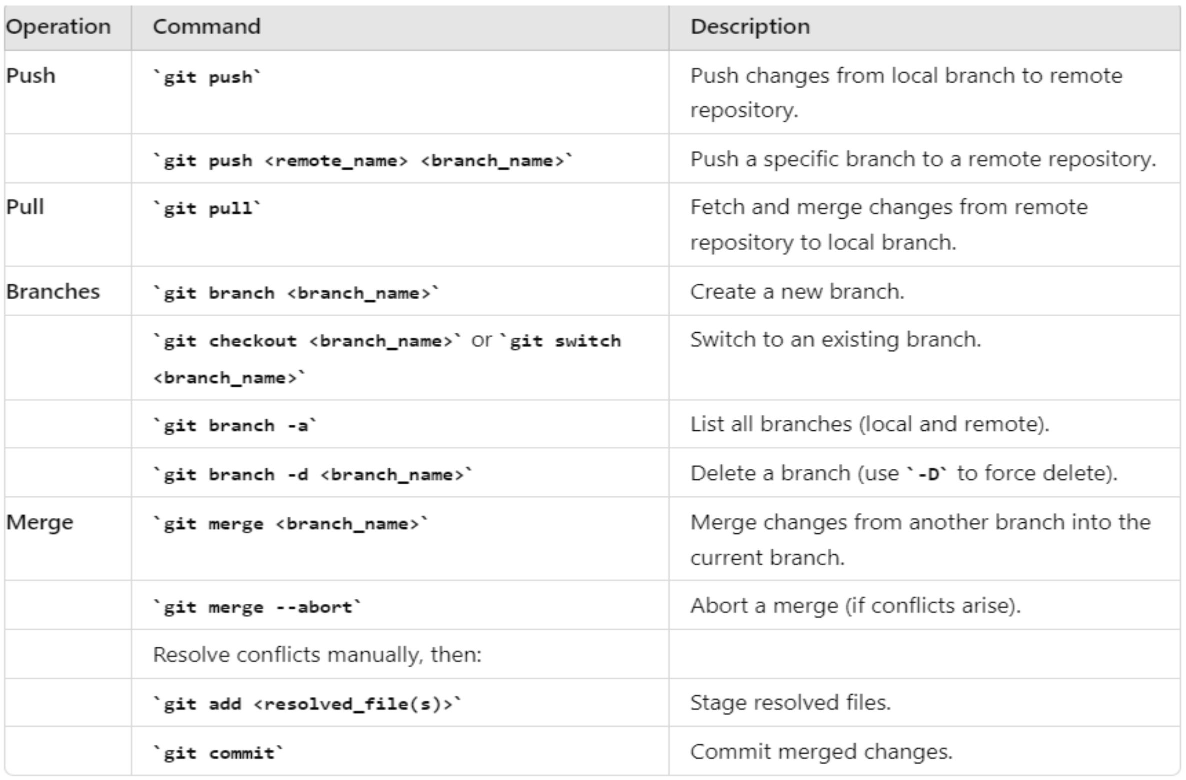
Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close th issues.
None yet

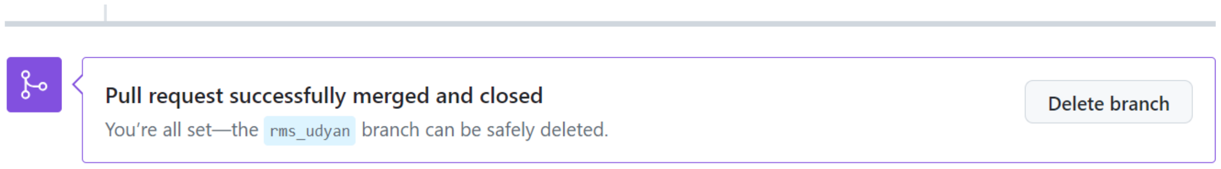
Notifications

Some useful git commands:



Operation	Command	Description
Push	`git push`	Push changes from local branch to remote repository.
	`git push <remote_name> <branch_name>`	Push a specific branch to a remote repository.
Pull	`git pull`	Fetch and merge changes from remote repository to local branch.
Branches	`git branch <branch_name>`	Create a new branch.
	`git checkout <branch_name>` or `git switch <branch_name>`	Switch to an existing branch.
	`git branch -a`	List all branches (local and remote).
	`git branch -d <branch_name>`	Delete a branch (use `-D` to force delete).
Merge	`git merge <branch_name>`	Merge changes from another branch into the current branch.
	`git merge --abort`	Abort a merge (if conflicts arise).
	Resolve conflicts manually, then:	
	`git add <resolved_file(s)>`	Stage resolved files.
	`git commit`	Commit merged changes.

Additionally, you can also delete the other branches, after the merge. It would look something like:



Pull request successfully merged and closed

You're all set—the `rms_udyan` branch can be safely deleted.

FAQs:

1. **What is a branch in GitHub?**

   - A branch in GitHub is a parallel version of a repository's main codebase. It allows developers to work on features or fixes without affecting the main code until they are ready to merge their changes.

2. **What is a pull request (PR) in GitHub?**

   - A pull request is a GitHub feature that allows developers to propose changes to a repository. It enables collaboration by letting others review and discuss the proposed changes before they are merged into the main branch.

3. **How do you create a pull request in GitHub?**

   - To create a pull request, you typically:

     1. Fork the repository or create a branch from the main repository.

     2. Make your changes in your branch.

3. Push your branch to GitHub.

4. Open a pull request from your branch to the main repository's branch you want to merge into.

4. **What is the difference between merge and rebase in Git?**

- Merge: Integrates changes from one branch into another, creating a merge commit.

- Rebase: Moves the entire feature branch onto the base branch, replaying each commit on top of the base branch.

5. **How do you merge branches in GitHub?**

- To merge branches in GitHub, you can:

1. Open a pull request from the branch you want to merge (`feature-branch`) into the target branch (`main`).

2. After review and approval, click "Merge pull request" to merge the changes into the target branch.

6. **What are conflicts in GitHub pull requests?**

- Conflicts occur when changes from one branch cannot be automatically merged with the base branch. This usually happens when both branches modify the same lines of code differently.

7. **How do you resolve conflicts in a GitHub pull request?**

- To resolve conflicts, you need to:

1. Identify conflicting files and lines in the pull request.

2. Edit the conflicting files directly on GitHub or locally on your machine.

3. Commit the resolved changes and push them to GitHub.

8. **What is a squash merge in GitHub?**

- A squash merge combines all commits from a feature branch into a single commit before merging into the base branch. It helps maintain a cleaner commit history.

9. **Can you delete a branch after merging it in GitHub?**

- Yes, after merging a branch into another (e.g., `main`), you can safely delete the merged branch through GitHub's interface.

10. **What is the difference between pull requests and push requests in GitHub?**

- Pull requests: Used to propose changes and request review and merging into another branch.

- Push requests: Generally not a specific GitHub term; pushing refers to pushing commits from your local repository to a remote repository.

11. **How do you review a pull request on GitHub?**

- To review a pull request, you:

1. Open the pull request.

2. Examine the changes, add comments, and suggest modifications if needed.

3. Approve or request changes before merging.

12. **What is a code review in GitHub?**

- A code review is the process of examining another developer's code changes to ensure quality, adherence to coding standards, and potential improvements before merging them into the main codebase.

13. **How do you enforce branch protection rules in GitHub?**

- Branch protection rules in GitHub prevent direct pushes to specified branches and enforce status checks and required reviews before merging pull requests into those branches. You can set these rules in the repository settings.

14. **What are GitHub Actions?**

- GitHub Actions automate workflows, such as testing, building, and deploying code directly from GitHub repositories. They are triggered by events like push requests, pull requests, or scheduled tasks.

15. **How do you revert a commit in GitHub?**

- To revert a commit in GitHub, you:

1. Identify the commit you want to revert using its SHA hash.

2. Create a new branch or work directly on an existing branch.

3. Use `git revert <commit-hash>` to create a new commit that undoes the changes introduced by the specified commit.