**Ctrl + Click on under-lined headings and words opens the related hyperlink in browser**

**Get remote Name** 🡺 git remote



**To get only the remote URL** 🡺 git config --get remote.origin.url



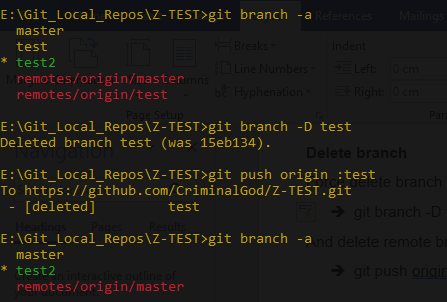
# Delete branch

Force delete branch locally 🡪 git branch -D {branch-name}

* git branch -D test

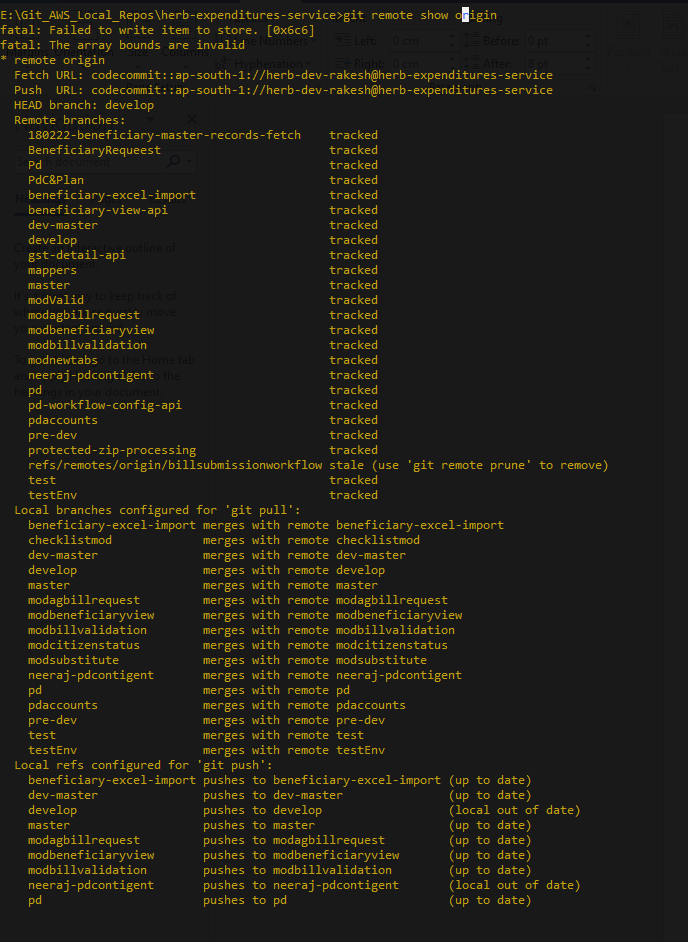
And delete remote branch 🡪 git push {remote-name} :{branch-name}

* git push origin :test

****

# To get more details about a particular remote

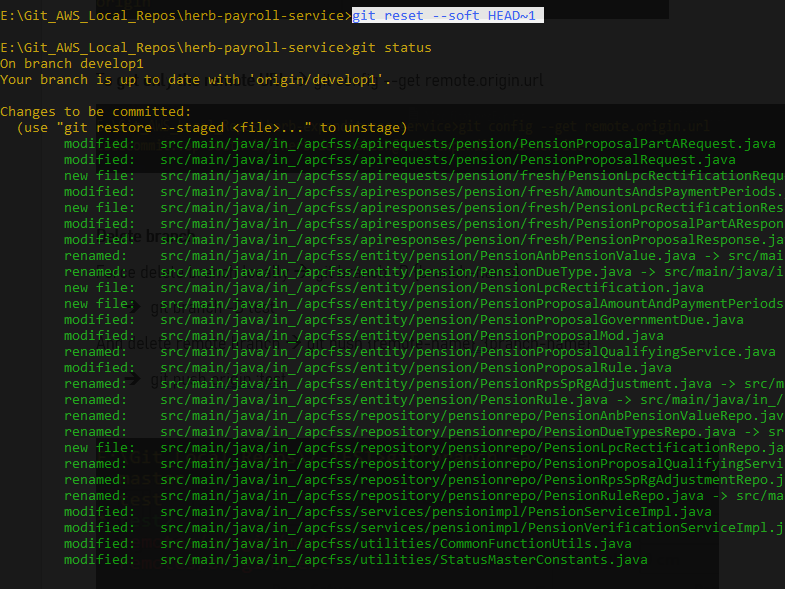
* git remote show {remote-name}



# [Move a commit to the staging area](https://stackoverflow.com/questions/7214039/how-do-you-move-a-commit-to-the-staging-area-in-git)

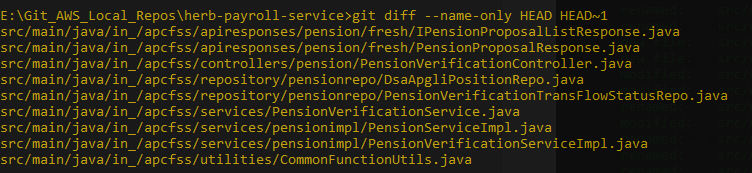
**For windows**

* git reset --soft HEAD~1



# [List only the names of files that changed between two commits](https://stackoverflow.com/questions/1552340/how-to-list-only-the-names-of-files-that-changed-between-two-commits)

* git diff --name-only HEAD~10 HEAD~5



# Make a new branch as default branch

When cloning a repo from GitHub, the default branch gets stored in the HEAD file:

* $ cat .git/refs/remotes/origin/HEAD
* ref: refs/remotes/origin/master

If the default branch is changed on GitHub after the repo has been cloned, this is not updated automatically, but can easily be fixed locally:

* git remote set-head origin -a

“-a” will set refs/remotes/<name>/HEAD according to remote

To set explicitly to a named branch:

* git remote set-head origin develop

Now, HEAD points to the new location:

* $ cat .git/refs/remotes/origin/HEAD
* ref: refs/remotes/origin/develop

# .gitignore file

A gitignore file specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected.

The purpose of gitignore files is to ensure that certain files not tracked by Git remain untracked.

To stop tracking a file that is currently tracked, use [git rm --cached](#_Remove_Files_in).

**Example:**

HELP.md

target/

\*/target/

\*target/

/target/

src/test/

!.mvn/wrapper/maven-wrapper.jar

!\*\*/src/main/\*\*/target/

!\*\*/src/test/\*\*/target/

logs/

bin/

LOGS/

### STS ###comment

.apt\_generated

.classpath

.factorypath

.project

.settings

.springBeans

.sts4-cache

# Few reasons why Git might be ignoring the .gitignore file

There are a few reasons why Git might be ignoring the .gitignore file and still tracking files that you want to ignore.

1. Git caching: Sometimes, Git might have already cached the files before you added them to the .gitignore file. In this case, you'll need to remove the files from the Git cache and then commit the changes to the .gitignore file.
2. Incorrect syntax in .gitignore: Make sure that the syntax in the .gitignore file is correct. For example, each file or directory should be on a new line and you should not use spaces in front of the line.
3. File was committed before .gitignore was updated: If the file was already committed to the repository before the .gitignore file was updated, Git will continue to track it. You'll need to remove the file from the repository and then add it to the .gitignore file.

If you've tried all these steps and the files are still being tracked, it might be worth checking if there are any other .gitignore files in the repository that are overriding the one you're trying to use.

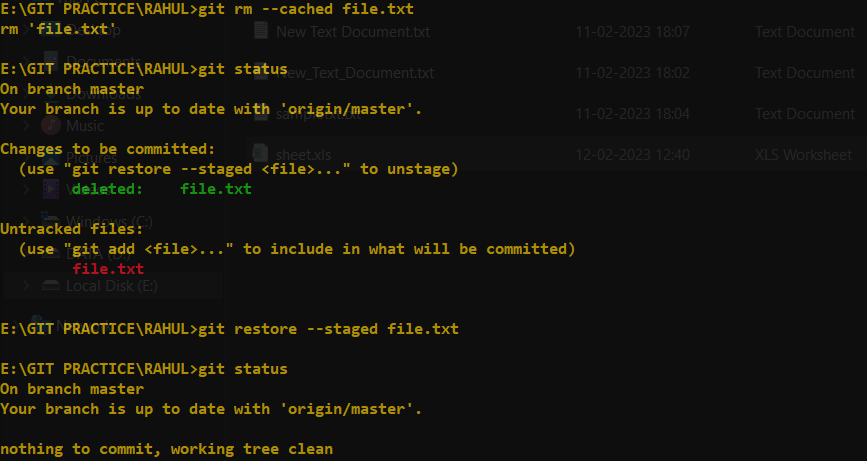
You can also try using the git check-ignore command to see if a specific file is being ignored, and if so, why. For example, git check-ignore -v path/to/file.

# [Remove Files in Git Repository](https://www.git-tower.com/learn/git/commands/git-rm" \l ":~:text=By%20default%2C%20the%20git%20rm,disk%20will%20not%20be%20deleted.)

The git rm command removes a file from a Git repository. This command removes a file from your file system and then removes it from the list of files tracked by a Git repository.

The --cached flag lets you delete a file from a Git repository without deleting it on your file system.

Using --dry-run flag, **No files are actually removed.** With this option (or its shorthand -n notation), you will only see an output of the files that Git would remove - but no files are actually deleted.



# [Create a new repository](https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository)

1. [Create a new repository](https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository) on GitHub.com. To avoid errors, do not initialize the new repository with README, license, or .gitignore files. You can add these files after your project has been pushed to GitHub.
2. Use the init command to initialize the local directory as a Git repository. It creates new local repository which contains default branch.
3. Add the files in the local repository and stages them for commit
4. Commit the tracked changes and prepares them to be pushed to a remote repository. To remove the commit which is not pushed and modify the file, use 'git reset --soft HEAD~1' and again add and commit the file.
5. At the top of your repository on GitHub.com's Quick Setup page, click copy the remote repository URL. In the Command prompt, [add the URL of the remote repository](https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories) where your local repository will be pushed.
6. Verify the new remote URL.
7. Push the changes of your local repository to the remote repository you specified as the origin.

* git init -b main
* git add .
* git commit -m "First commit"
* Create a new repository in GitHub and copy the remote URL.
* git remote add origin <REMOTE\_URL>
* git remote –v
* git push origin main