Git

Git init

```
git init
```

Git clone

```
git clone `link`
```

Git add

```
git add .

adds a change in the working directory to the staging area.
```

Git commit

```
git commit -m "Message"

create a snapshot of the staged changes along a timeline of a
Git projects history.
```

Git stash

```


temporarily shelves (or stashes) changes you've made to your wo
you can work on something else, and then come back and re-apply

git stash   => will stash the changes made to some temporary ad
                                perform git fetch


```

.gitignore

```
put the file names that are not requited in this file with name

we can also put the names of folders like node_modules
```

Git fetch

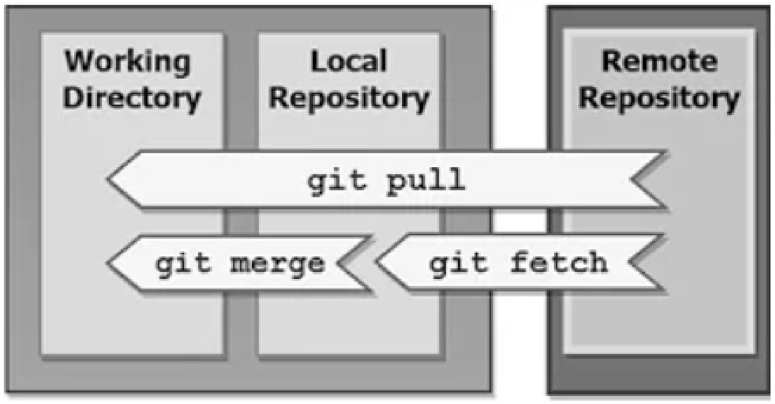
```
git fetch => will fecth from remote directory to the local direc
                            not working directory

git fetch <remote> <branch>  => Same as the above command, but o

```

Git pull

```
git pull or git pull origin main =>
1. is combination of git fetch and git merge.
2. to fetch and download content from a remote repository and
     immediately update the local repository to match that conte
```



Git push

```
git push => to upload local repository content to a remote repos

git push origin main => push to remote origin branch from the lo
```

Git tag

```
git tag => A tag is an object referencing a specific commit witl
                    project history


git tag v1.4-lw
```

GIt status:

```
git status => shows the tract of the file / directories.
```

Git branch

```
git branch => List all of the branches in your repository

git branch <branch> => Create a new branch called <branch>

git branch -d <branch> => Delete the specified branch
```

Git checkout

```
git checkout <branch> => switches to the branch
```

Git merge

```
## Standard way of doing.
# Start a new feature
git checkout -b new-feature main
# Edit some files
git add <file>
git commit -m "Start a feature"
# Edit some files
git add <file>
git commit -m "Finish a feature"
# Merge in the new-feature branch
```

```
git checkout main
git merge new-feature
git branch -d new-feature
```

Git revert | GIt reset

```
git revert <hash_of_commit> => revert the commit with the hash

git reset <hash_of_commit>  => reset the current HEAD to a previ
                                                       discard
```

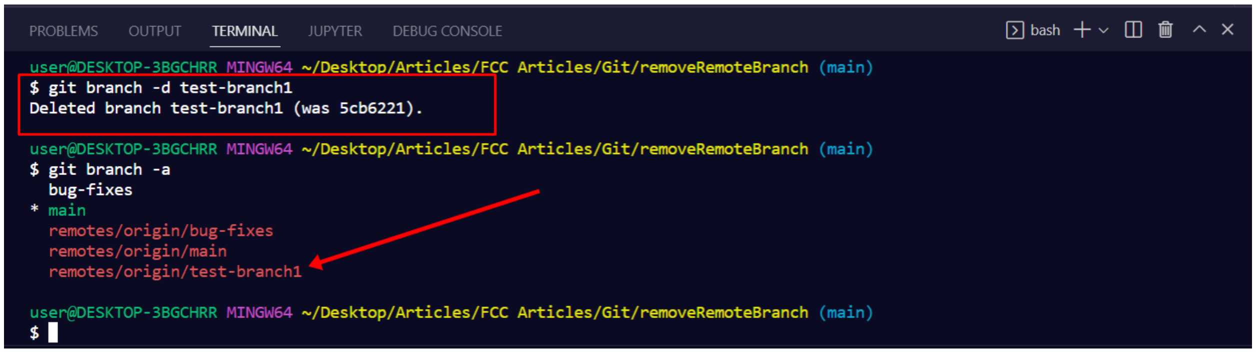
Git rebase | Git rebase -i

```
git rebase <branch>
```

https://learngitbranching.js.org/

use this link.

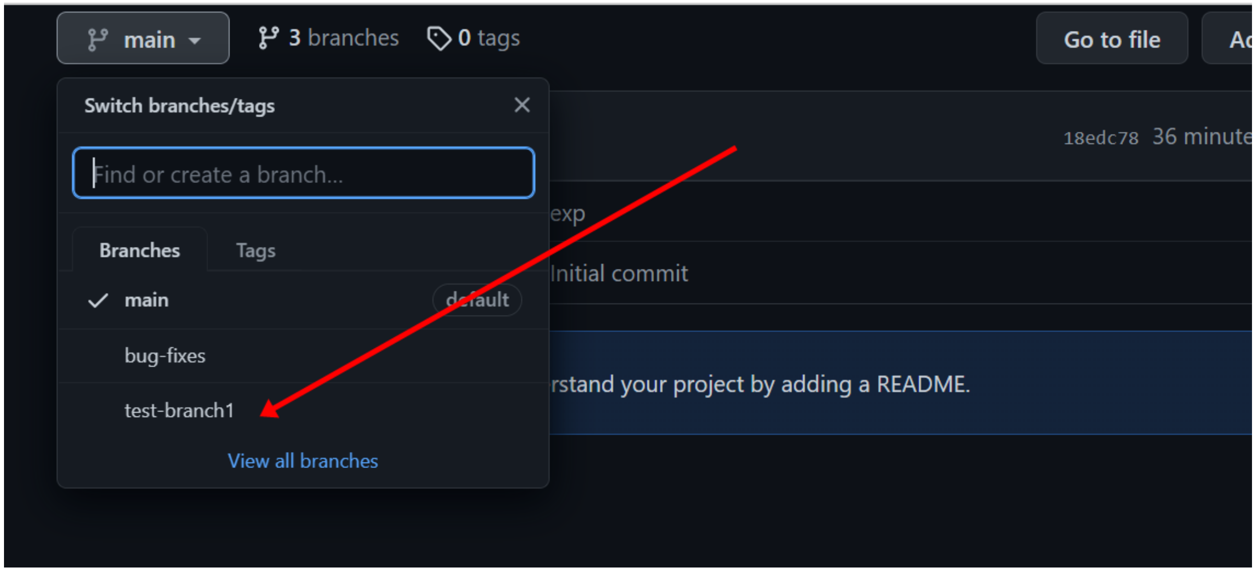
Problem: 1



```
user@DESKTOP-3BGCHRR MINGW64 ~/Desktop/Articles/FCC Articles/Git/removeRemoteBranch (main)
$ git branch -d test-branch1
Deleted branch test-branch1 (was 5cb6221).

user@DESKTOP-3BGCHRR MINGW64 ~/Desktop/Articles/FCC Articles/Git/removeRemoteBranch (main)
$ git branch -a
  bug-fixes
* main
  remotes/origin/bug-fixes
  remotes/origin/main
  remotes/origin/test-branch1

user@DESKTOP-3BGCHRR MINGW64 ~/Desktop/Articles/FCC Articles/Git/removeRemoteBranch (main)
$
```



then `git push origin -d branch-name`

Doubts

1.

Git checkout a remote branch

When collaborating with a team it is common to utilize remote repositories. These repositories may be hosted and shared or they may be another colleague's local copy. Each remote repository will contain its own set of branches. In order to checkout a remote branch you have to first fetch the contents of the branch.

```
git fetch --all
```

In modern versions of Git, you can then checkout the remote branch like a local branch.

```
git checkout <remotebranch>
```

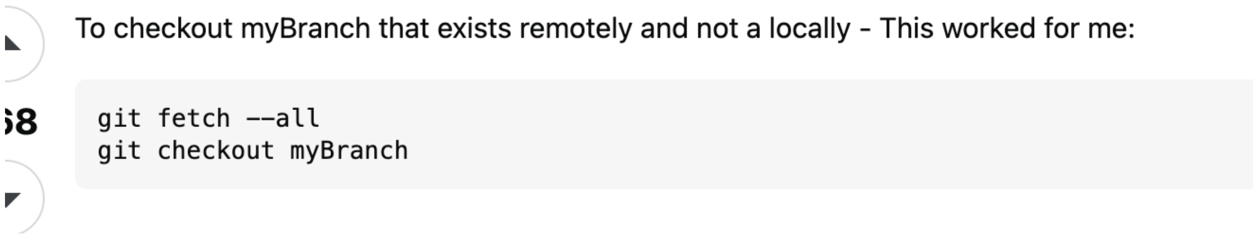
Older versions of Git require the creation of a new branch based on the `remote`.

we are just fetching but not pulling, then how will this work

2.

1 commit behind, 3 commits ahead,

commit local changes → pull → merge → commit after merging → push.



To checkout myBranch that exists remotely and not a locally - This worked for me:

```
git fetch --all
git checkout myBranch
```

Didn’t understand error handling in Link