

Git

Git is a piece of software that makes it simpler to manage different versions of your own software in a software repository.

GitHub

GitHub is a place to share software repositories and collaborate with other developers. Github is a website where you host all of your Git repositories.

These tools make it easier for multiple people to work on the same software project.

To create bullet points in Markdown on GitHub, start each line with an asterisk (*), hyphen (-), or plus sign (+), followed by a space, and then the text for the bullet point.

In this course you will learn about:

- common terms
- installing Git
- creating a Git repository
- cloning a Git repository
- adding and removing from a Git repository
- using GitHub
- creating pull requests
- branching repositories
- merging repositories
- and more

What is Version Control?

What is Git?

A: Free and open source version control system.

What is Version Control?

A: The management of changes to documents, computer programs, large web sites, and other collections of information.

VC is basically a way **programmers track their code changes**. They can look back at **all of their code changes** they have **made over time**. It helped them

- to see and understand what they did
- Track down bugs
- Go back to previous versions of the code if they need to

Terms

- Directory -> **Folder**
- Terminal or Command Line -> **Interface for Text Commands**
- CLI -> **Command Line Interface**
- cd -> **Change Directory**
- Code Editor -> **Word Processor for Writing Code**
- Repository -> **Project, or the folder/place where your project is kept**
- Github -> **A website to host your repositories online**

Git Commands

Git Commands

- Clone -> Bring a repository that is hosted somewhere like Github into a folder on your local machine
- add -> Track your files and changes in Git
- commit -> Save your files in Git
- push -> Upload Git commits to a remote repo, like Github
- pull -> Download changes from remote repo to your local machine, the opposite of push

How to find Git version in command prompt?

```
git Command Prompt  
Microsoft Windows [Version 10.0.19045.5737]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\kiruthikard>git --version  
git version 2.40.0.windows.1  
  
C:\Users\kiruthikard>
```

Link to install Git

<https://www.atlassian.com/git/tutorials/install-git>

Link to install Visual Studio

<https://www.code.visualstudio.com>

Git Clone

git clone https://github.com/kirthi2005/Git_GitHub_Freecodecamp_Course.git -
Clone the remote repository local to your machine

Git Status

git status - command shows all the files that were created or updated or deleted but not saved in commit yet.

Git Add

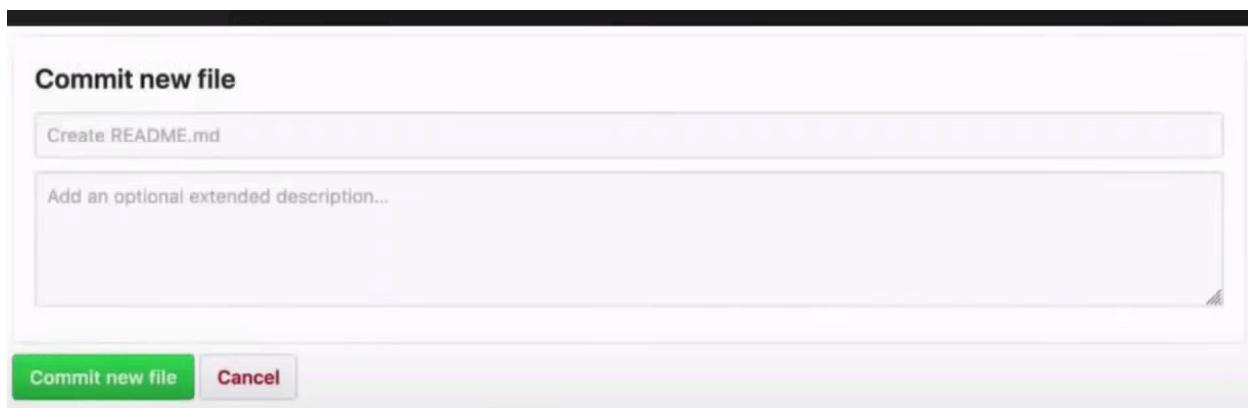
git add . -> command to tell git to track all of the files the modified and the intracked files (new files) (stage the changes)

git add filename - to add specific file

Git Commit

git commit -m “Added Readme” -m “you can add description here”

-m is for message



Git Push

git push origin master or git push origin main - push our changes in local machine to remote repository. Origin - location of our github repository. master/main are the branch where we want to put our code

SSH Keys

Used to connect your local machine and remote repository safely. These keys are used to identify you are owner of the repository.

```
→ ~ ssh-keygen -t rsa -b 4096 -C "email@example.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/gwen/.ssh/id_rsa): testkey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in testkey.
Your public key has been saved in testkey.pub.
The key fingerprint is:
SHA256:gHmbQn8S/hAhl2GMFzAap1moz2HYzD0wf6LkisCW9J0 email@example.com
The key's randomart image is:
+---[RSA 4096]---+
| ..***+ |
| +B.B+. |
| *+=+=.= |
| o.B.=+.* |
| o*ooo+0 S |
| .+=. E = |
| +. . |
| o |
+---[SHA256]---+
→ ~ █
```

ssh-keygen - generate ssh key

-t rsa - algorithm name

-b 4096 strength of the key

Email id

```
→ ~ ls | grep testkey
testkey
testkey.pub
→ ~ █
```

testkey is a private key. You have keep secured in your local machine

testkey.pub - indicates the key is available to the public. This is the one we are going to upload in github

The public will look like below

```
→ ~ cat testkey.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQACQC0UvmTlMBPo2Le0xboAwIs0Ms7BKYtlc1oKlXl0kHnTDPl
L+byKM+A8FFvavp90ktwmp+1Bkb9FP4UP39XM6+TQDjb+RPu9ic+f8wuzHeMHVu77AYLGAQKBCxU6cYVsbeF
q+bplyqb6vmIDFH6++NFe35R2bexXE4ojE/RXCMDx4xvEUvuy4l33v6bA5HcIWGPyusYpfwYPB6GKStWy2Rf
wBjnD/l00XRwzcmJu6EJ3ab1wQPGHBEj8BAucZlm0ZpWIdnNg7k0fhv+/tNVYfc3GE+qXpAay486C/z5SU01
F2nnHnsA3WI/L0yJSTc3XKo8zYfWdTmRMnL5MYbS5ifNJpy2YYFWF86kvw14rkxnAX5/VA/BJ/VOTXUh/CRb
gB0o+DdSXZ008Wtl3a6QMkqprEaShf/vzAvVmbz+Qco0jZ9jUMFeg5wMYkFKycTWIbcse3tBbvzPICBkmGx8
NCGxlv/nLxr28AwPUGkIPxZhCYVjYZj8n0oCa8b/E70Lod94Csyp5B50aXu60ZR93y2mGB0y49muTCA6vKUi
HNT/KT4072e5ocCyuMw1Fi2Gg+F0e43nYY0MdwnCXL0pvFcQzXs9T0BuaeP9G1FZ5M+GBUFImLt31aRkPn
VrKVCyfJSbmqeGWkMnukFfmmmp25jxEr4n3GM2gzhQbkJK6cIGQ== email@example.com
→ ~
```

Git Init

git init - initialize git repository in the folder in your local machine

```
PS C:\Kiruthika\Git\demo_repo> git init
Initialized empty Git repository in C:/Kiruthika/Git/demo_repo/.git/
PS C:\Kiruthika\Git\demo_repo>
```

Git Remote

Create remote repository first and then push our local code.

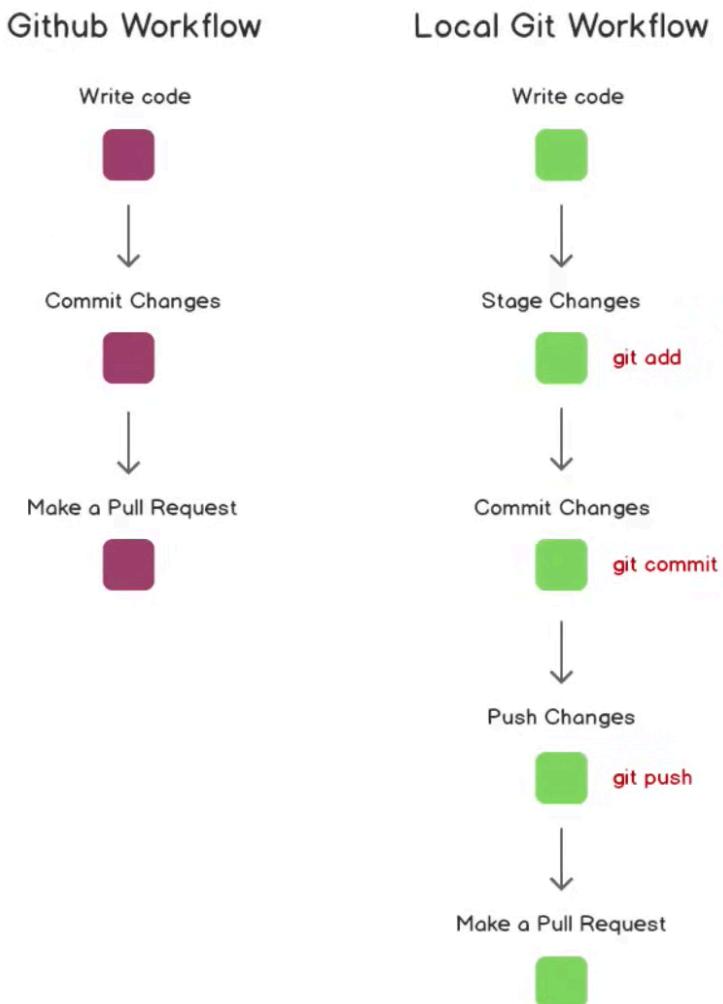
git remote add origin “your remote repository url” - add reference to the remote repository on github.

git remote -v - show any remote repositories that I've connected to this repo.

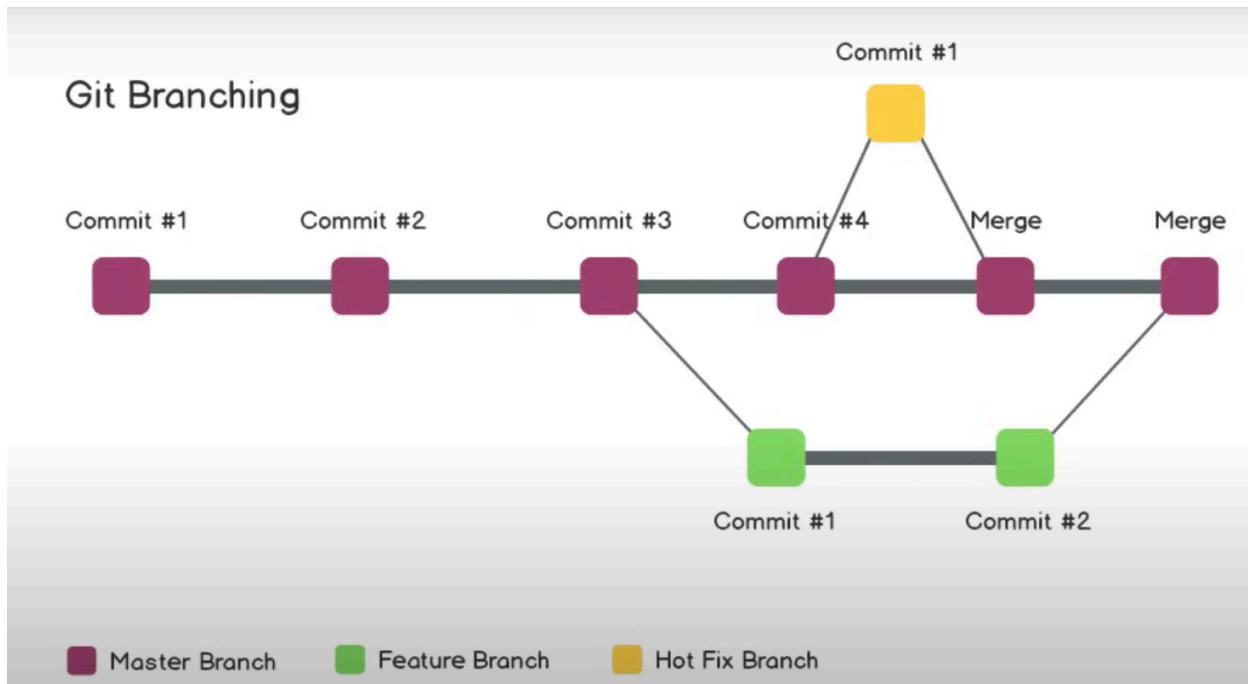
git push origin master

git push -u origin master (-u is upstream. No need to type origin master everytime)

Git and Local workflow



Git Branching



Code in master branch and feature branch will be the same. But if you do changes in feature branch it won't get reflected in master branch.

```
→ demo-repo git:(master) git branch
→ demo-repo git:(master) git checkout -b feature-readme-instructions
Switched to a new branch 'feature-readme-instructions'
→ demo-repo git:(feature-readme-instructions) git branch
→ demo-repo git:(feature-readme-instructions) git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
→ demo-repo git:(master) git branch
→ demo-repo git:(master) git checkout feature-readme-instructions █
```

git checkout -b branchname -> -b will create new branch

git checkout master - switch to master branch

git checkout feature - switch to feature branch

```
→ demo-repo git:(master) git checkout feature-readme-instructions
Switched to branch 'feature-readme-instructions'
→ demo-repo git:(feature-readme-instructions) git status
On branch feature-readme-instructions
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

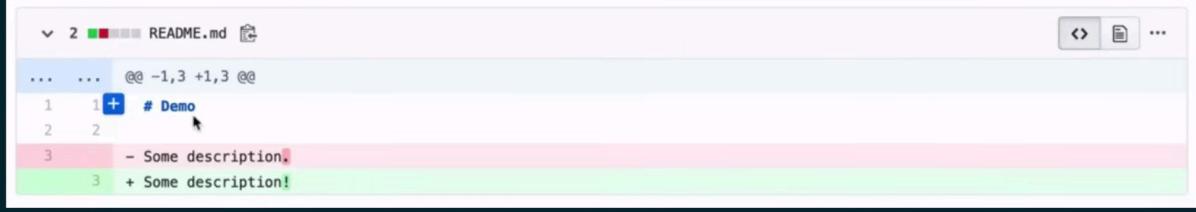
no changes added to commit (use "git add" and/or "git commit -a")
→ demo-repo git:(feature-readme-instructions) x MORE VIDEOS
```

```
→ demo-repo git:(feature-readme-instructions) x git add README.md
→ demo-repo git:(feature-readme-instructions) x git commit -m "updated readme"
[feature-readme-instructions cec9aeb] updated readme
 1 file changed, 4 insertions(+)
→ demo-repo git:(feature-readme-instructions) git checkout mast
```

```
1 file changed, 4 insertions(+)
→ demo-repo git:(feature-readme-instructions) git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
→ demo-repo git:(master)
```

Git diff

```
→ demo-repo git:(master) git diff
```



The screenshot shows a terminal window with the command 'git diff' executed. Below the terminal, a graphical diff viewer displays the changes made to 'README.md'. The viewer shows a context diff with three lines of code: '# Demo', '- Some description.', and '+ Some description!'. The '+' sign indicates a new line, while the '-' sign indicates a deleted line.

git diff feature-readme-instructions

```
diff --git a/README.md b/README.md
index ac186b2..2b23fdd 100644
--- a/README.md
+++ b/README.md
@@ -5,7 +5,3 @@ Some description!
## Subheader

Watch tutorial on YouTube.

-
-## Local Development
-
-1. Open index.html in your browser.
(END)
```

Git Merge

```
demo-repo git:(master) git diff feature-readme-instructions
demo-repo git:(master) git merge feature-readme-instructions
→ demo-repo git:(master) git diff feature-readme-instructions
→ demo-repo git:(master) git checkout feature-readme-instructions
Switched to branch 'feature-readme-instructions'
→ demo-repo git:(feature-readme-instructions) git status
On branch feature-readme-instructions
nothing to commit, working tree clean
→ demo-repo git:(feature-readme-instructions) git push
fatal: The current branch feature-readme-instructions has no upstream branch.
To push the current branch and set the remote as upstream, use
    git push --set-upstream origin feature-readme-instructions
→ demo-repo git:(feature-readme-instructions)
```

```

→ demo-repo git:(feature-readme-instructions) git push -u origin feature-readme-instructions
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 380 bytes | 380.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'feature-readme-instructions' on GitHub by visiting:
remote:     https://github.com/gwenf/demo-repo/pull/new/feature-readme-instructions
remote:
To github.com:gwenf/demo-repo.git
 * [new branch]      feature-readme-instructions -> feature-readme-instructions
Branch 'feature-readme-instructions' set up to track remote branch 'feature-readme-instructions' from 'origin'.
→ demo-repo git:(feature-readme-instructions)

```

Pull request

It is basically a to have your code pulled into another branch. In the example, we are making PR from the feature branch to the master branch.once we have made a PR anyone can review our code ,comment on it, ask us to make changes or updates.After you make a PR you can also update the code just by making additional commits and pushing them upto Github, as long as u r on the branch where u made the PR.Once PR is merged delete your feature branch and switch back to master branch.

The screenshot shows a GitHub repository interface. At the top, there are summary statistics: 3 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. Below this, a section titled "Your recently pushed branches:" lists a single branch: "feature-readme-instructions" (less than a minute ago). To the right of this list is a green button labeled "Compare & pull request". Further down, there are buttons for "Branch: master" (with a dropdown arrow), "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". At the bottom of the page, a commit history shows a single entry: "gwenf Added index.html" with a timestamp of "Latest commit 3a1d48e on Mar 18".

Open a pull request

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

The screenshot shows the GitHub interface for opening a pull request. At the top, it says "base: master" and "compare: feature-readme-instructions". A green checkmark indicates "Able to merge. These branches can be automatically merged." Below this, there's a preview window titled "updated readme" with "Write" and "Preview" tabs. The preview tab shows a rich text editor with a "Leave a comment" area. To the right, there are sections for "Reviewers" (No reviews), "Assignees" (No one—assign yourself), "Labels" (None yet), and "Projects" (None yet). Below these, a box highlights "Continuous integration has not been set up" and mentions GitHub Actions and several other apps. Another section shows a green checkmark indicating "This branch has no conflicts with the base branch" and notes that merging can be performed automatically. At the bottom, a green button says "Merge pull request" with a dropdown arrow, and a note says "You can also open this in GitHub Desktop or view command line instructions."

Pull code to local master branch

```
→ demo-repo git:(feature-readme-instructions) git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
→ demo-repo git:(master) git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From github.com:gwenf/demo-repo
  3a1d48e..cbf1341  master    -> origin/master
Updating 3a1d48e..cbf1341
Fast-forward
 README.md | 4 ++++
 1 file changed, 4 insertions(+)
→ demo-repo git:(master)
```

Git branch delete

```
→ demo-repo git:(master) git branch
→ demo-repo git:(master) git branch -d feature-readme-instructions
Deleted branch feature-readme-instructions (was cec9aeb).
→ demo-repo git:(master) }
```

Merge Conflicts

Different people working on the same file and merging their own code at same time in master branch. So sometime Git doesn't know which code to keep or which code is redundant or which code you want to get rid of.

Commit and add at the same time

It works only for modified files.

```
git commit -am "commit message"
```

Undoing in Git

Changes are staged

```
→ demo-repo git:(quick-test) git status
On branch quick-test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
→ demo-repo git:(quick-test) x git add README.md
→ demo-repo git:(quick-test) x git status
On branch quick-test
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
→ demo-repo git:(quick-test) }
```

Undo the changes

```
git reset or git reset filename (name of the file to unstage)
git reset --hard id# (not only instaged but completely removed)
```

```
→ demo-repo git:(quick-test) ✘ git reset  
Unstaged changes after reset:  
M README.md  
→ demo-repo git:(quick-test) ✘ █
```

```
→ demo-repo git:(quick-test) ✘ git status  
On branch quick-test  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git checkout -- <file>..." to discard changes in working directory)  
  
        modified:   README.md  
  
no changes added to commit (use "git add" and/or "git commit -a")  
→ demo-repo git:(quick-test) ✘ █
```

```
no changes added to commit (use "git add" and/or "git commit -a")  
→ demo-repo git:(quick-test) ✘ git add README.md  
→ demo-repo git:(quick-test) ✘ git commit -m "added install step"  
[quick-test 74367da] added install step  
 1 file changed, 1 insertion(+)  
→ demo-repo git:(quick-test) git status  
On branch quick-test  
nothing to commit, working tree clean  
→ demo-repo git:(quick-test) git reset HEAD~1  
Unstaged changes after reset:  
M README.md  
→ demo-repo git:(quick-test) ✘ g█  
MORE VIDEOS
```

HEAD points to last commit and ~1 is go back before one step of head

Git log

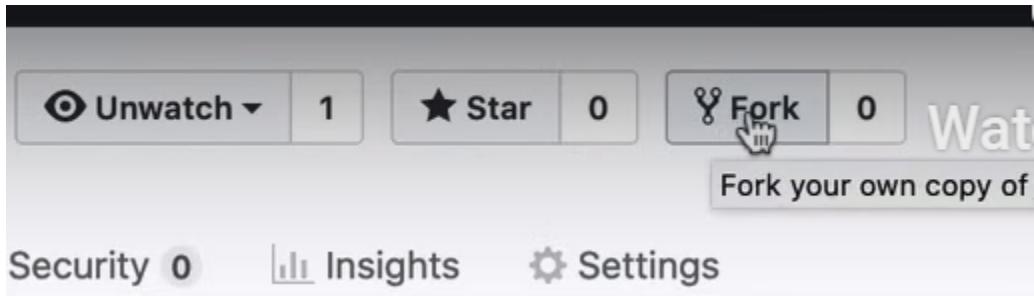
git log - see a log of all your commits. Logs will be arranged in reverse chronological order.

Git Reset Hard

```
→ demo-repo git:(quick-test) ✘ git reset --hard cec9aebc03f09f1b8619a3f9ac39a576196  
7f0cf  
HEAD is now at cec9aeb updated readme  
→ demo-repo git:(quick-test) █
```

Fork

To make a complete copy of the repository



YouTube Video Link

<https://www.freecodecamp.org/news/git-and-github-crash-course/>

GitBash Command Review

Git bash basic commands review : <https://www.youtube.com/watch?v=oQc-2gsjgDg>

Top 20 Git Commands

<https://dzone.com/articles/top-20-git-commands-with-examples>

How to understand Git: an intro to basic commands, tips, and tricks

<https://www.freecodecamp.org/news/understanding-git-basics-commands-tips-tricks/>