GIT- A Simple Guide

What is GIT?

Git is the most widely used version control system in the world.
Now what's a version control system you'd ask.
Well, Google defines it as

"*Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are **software tools that help software teams manage changes to source code over time.***"

It's simply saving various instances of the changes that you make to your code in various branches so in case an error arises, it happens only in the the particular instance and you can always revert back to the original instance.

Why GIT?

- **Performance**

  The raw performance characteristics of Git are very strong when compared to many alternatives. Committing new changes, branching, merging and comparing past versions are all optimised for performance.

- **Security**

  Git has been designed with the integrity of managed source code as a top priority. The content of the files as well as the true relationships between files and directories, versions, tags and commits, all of these objects in the Git repository are secured with a cryptographically secure hashing algorithm called SHA1. This protects the code and the change history against both accidental and malicious change and ensures that the history is fully traceable.

- **Flexibility**

  One of Git's key design objectives is flexibility. Git is flexible in several respects: in support for various kinds of nonlinear development workflows, in its efficiency in both small and large projects and in its compatibility with many existing systems and protocols.

Installing GIT

https://www.atlassian.com/git/tutorials/install-git#windows
https://www.atlassian.com/git/tutorials/install-git#mac-os-x
https://www.atlassian.com/git/tutorials/install-git#linux

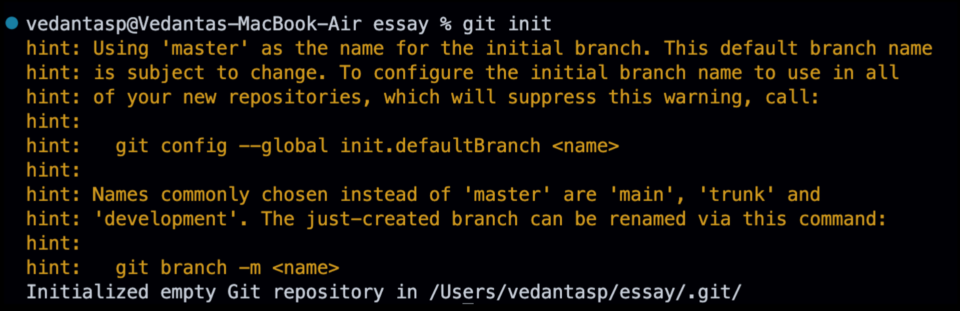
GIT Commands

So now that you've GIT installed, let's git started.

For this guide i am using visual studio code
Open a folder and create a text file eg: text.txt

To initiate git basically activate it we open the terminal in vsc
This creates a *repository- a file which can be tracked by git*

git init



```
● vedantasp@Vedantas-MacBook-Air essay % 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/vedantasp/essay/.git/
```

Fig 1.

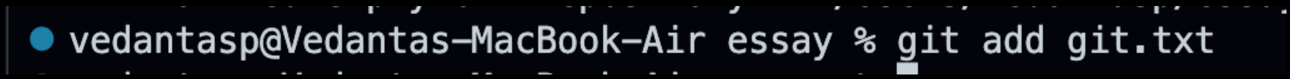
Now that we have a repository we will add files in it

To add files

git add <filename>

or to add all files , we simply use asterisk * in the place of file name.

git add *

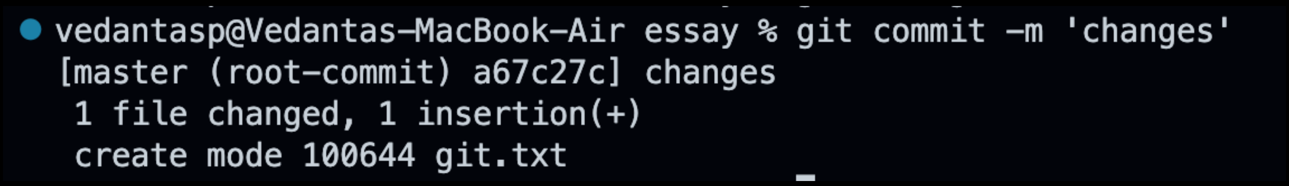


```
● vedantasp@Vedantas-MacBook-Air essay % git add git.txt
```

To commit or to save the changes you made in your repository like adding the text file

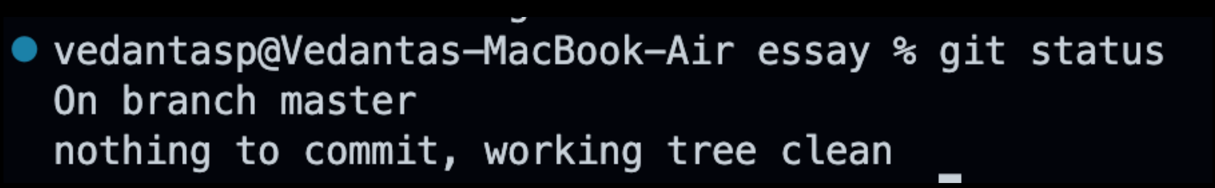
git commit -m "this is a message"

-m stands for message
And for every commit it is necessary not only for the clarity of your peers but as a good coding practice so you can know what changes you made.



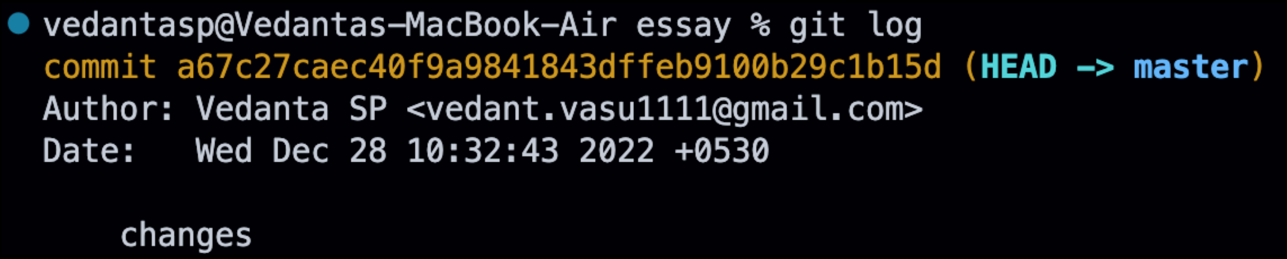
```
● vedantasp@Vedantas-MacBook-Air essay % git commit -m 'changes'
  [master (root-commit) a67c27c] changes
   1 file changed, 1 insertion(+)
   create mode 100644 git.txt
```

to check if any commits remain to be made
git status



```
● vedantasp@Vedantas-MacBook-Air essay % git status
  On branch master
  nothing to commit, working tree clean
```

If you want to see a log of all the changes you made
git log



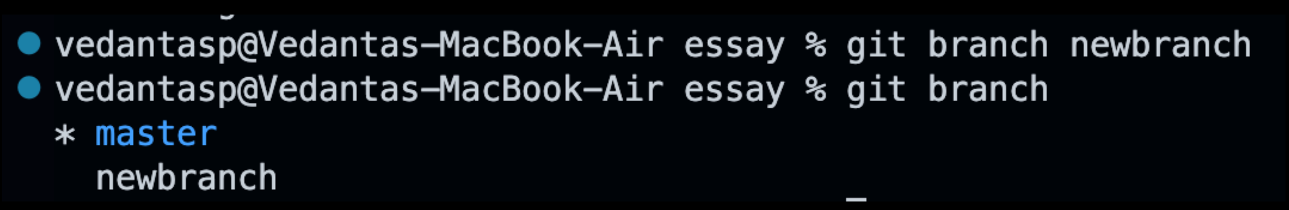
```
● vedantasp@Vedantas-MacBook-Air essay % git log
  commit a67c27caec40f9a9841843dffeb9100b29c1b15d (HEAD -> master)
  Author: Vedanta SP <vedant.vasu1111@gmail.com>
  Date:   Wed Dec 28 10:32:43 2022 +0530

      changes
```

Now we are done with the basic steps of git
We move to branching

Branches like their name are just like the real life branches stemming out of a main. branches are used to develop features independent from each other which can be merged into the main branch.

to create a new branch

git branch <branchname>

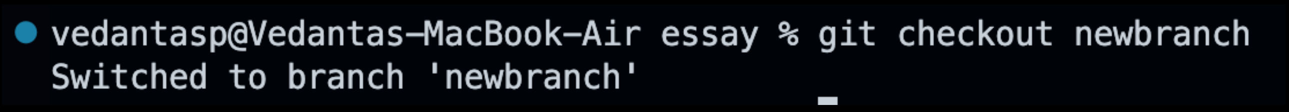


```
● vedantasp@Vedantas-MacBook-Air essay % git branch newbranch
● vedantasp@Vedantas-MacBook-Air essay % git branch
  * master
    newbranch
```

Every branch should have a unique name

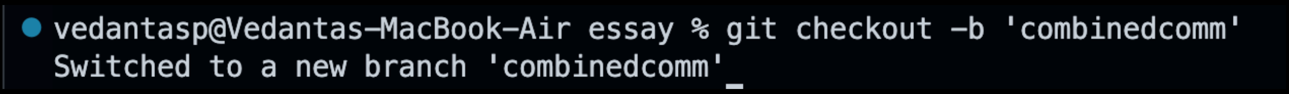
now we have created the branch but we still are on the main branch,

to move to a specific branch
git checkout <branchname>



```
● vedantasp@Vedantas-MacBook-Air essay % git checkout newbranch
  Switched to branch 'newbranch'
```

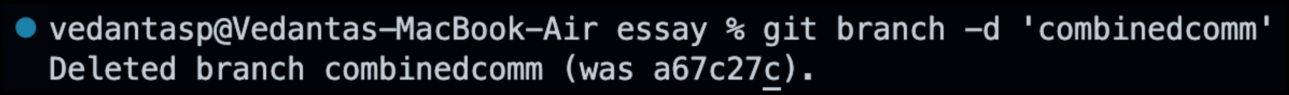
or we can Simply do both of these commands together
git checkout -b < branchname>



```
vedantasp@Vedantas-MacBook-Air essay % git checkout -b 'combinedcomm'
Switched to a new branch 'combinedcomm'
```

to delete a branch
It is important to be not on the branch that you want to delete

git branch -d < branchname>



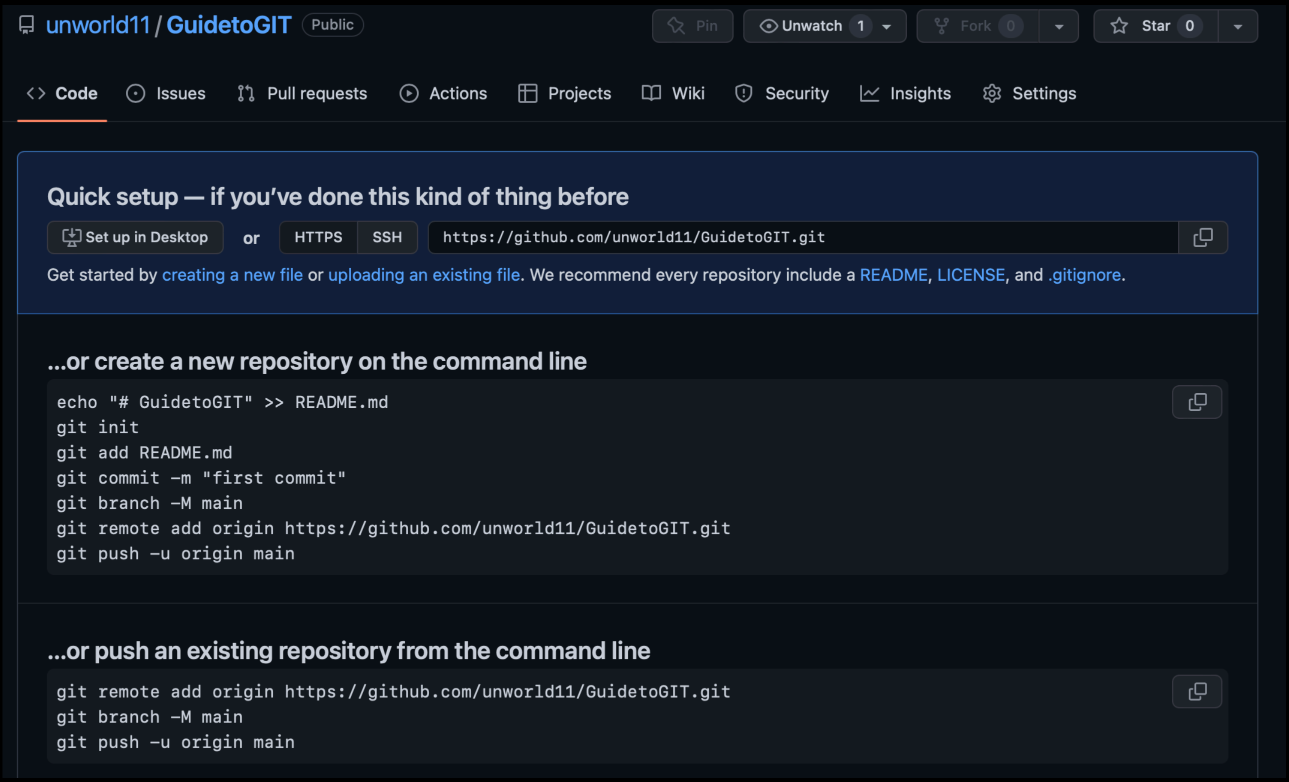
```
vedantasp@Vedantas-MacBook-Air essay % git branch -d 'combinedcomm'
Deleted branch combinedcomm (was a67c27c).
```

This will do most of the functions that you'll require locally on your machine.

but if you've to host

We use
GITHUB

We create a repository on Github and copy the url
git remote add origin <server>



Quick setup — if you've done this kind of thing before

https://github.com/unworld11/GuidetoGIT.git

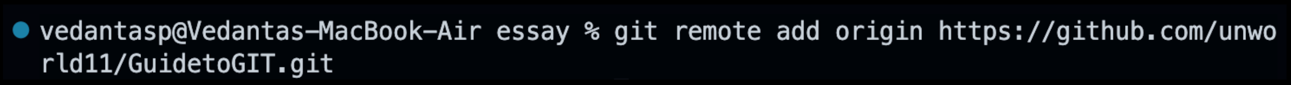
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# GuidetoGIT" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/unworld11/GuidetoGIT.git
git push -u origin main
```

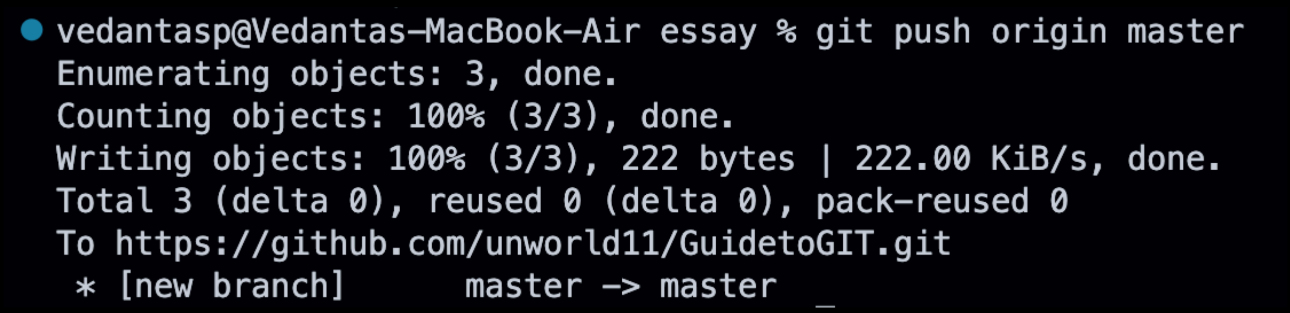
...or push an existing repository from the command line

```
git remote add origin https://github.com/unworld11/GuidetoGIT.git
git branch -M main
git push -u origin main
```



```
● vedantasp@Vedantas-MacBook-Air essay % git remote add origin https://github.com/unwo
  rld11/GuidetoGIT.git
```

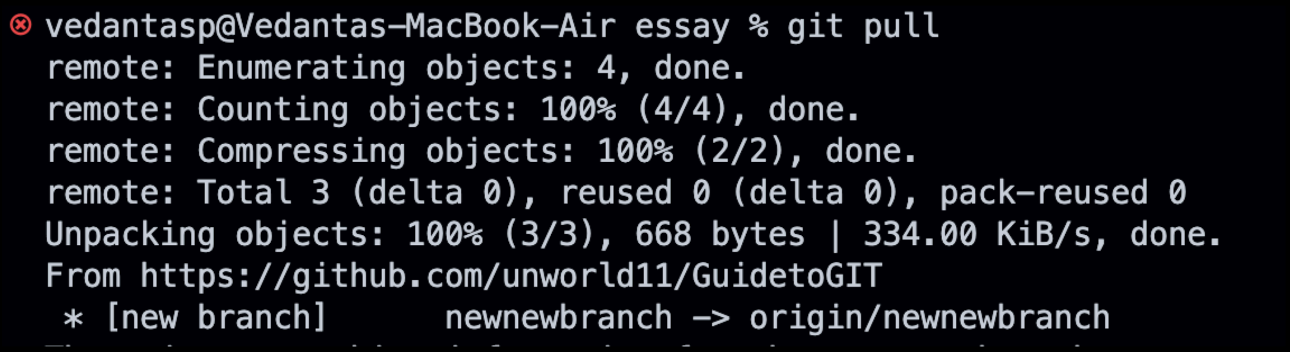
to make changes or to push changes on the remote repo
git push origin <branch changes you want to push >



```
● vedantasp@Vedantas-MacBook-Air essay % git push origin master
  Enumerating objects: 3, done.
  Counting objects: 100% (3/3), done.
  Writing objects: 100% (3/3), 222 bytes | 222.00 KiB/s, done.
  Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
  To https://github.com/unworld11/GuidetoGIT.git
   * [new branch]      master -> master
```

Now suppose your friend is also your collaborator and he makes some changes on the remote repository from his machine

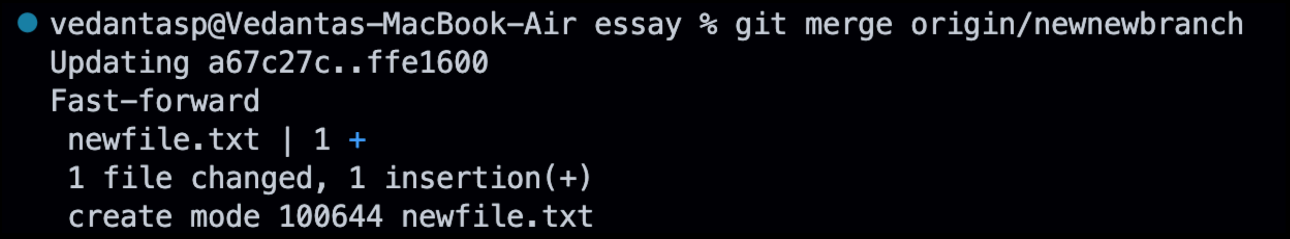
to update your local repo with new commits made by him
git pull



```
⊗ vedantasp@Vedantas-MacBook-Air essay % git pull
  remote: Enumerating objects: 4, done.
  remote: Counting objects: 100% (4/4), done.
  remote: Compressing objects: 100% (2/2), done.
  remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
  Unpacking objects: 100% (3/3), 668 bytes | 334.00 KiB/s, done.
  From https://github.com/unworld11/GuidetoGIT
   * [new branch]      newnewbranch -> origin/newnewbranch
```

Now you have added an feature and it works perfectly
You want to add it to the main branch

To merge branches
git merge <branch>



```
● vedantasp@Vedantas-MacBook-Air essay % git merge origin/newnewbranch
  Updating a67c27c..ffe1600
  Fast-forward
   newfile.txt | 1 +
   1 file changed, 1 insertion(+)
   create mode 100644 newfile.txt
```

And that’s all folks
git isn’t as complicated as it looks.