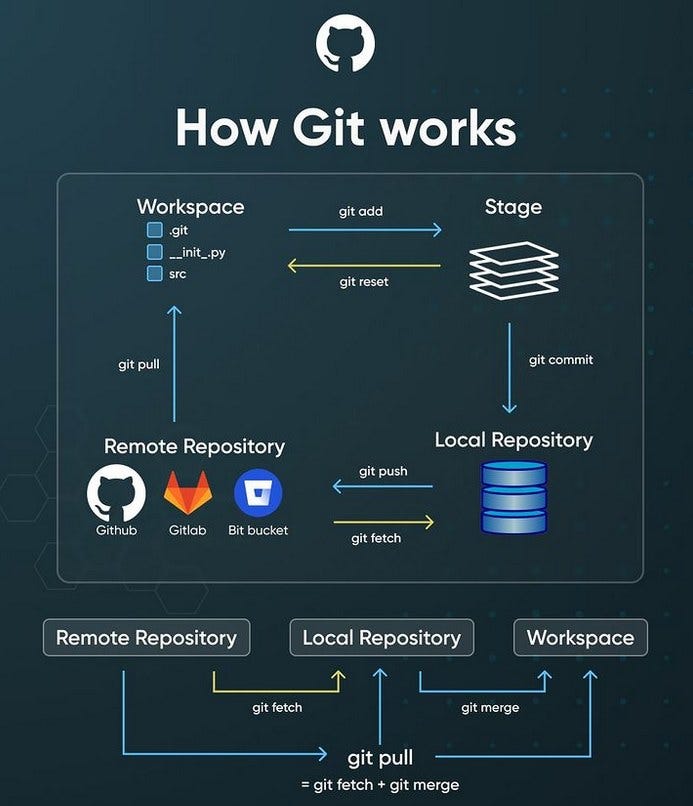
Git is a distributed version control system that tracks versions of files.  
 It is often used to control source code by programmers collaboratively developing software.

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

A workspace is the directory that git-workspace will manage for you, filling it with projects cloned from your providers.



**Installing on Linux**

If you want to install the basic Git tools on Linux via a binary installer, you can generally do so through the package management tool that comes with your distribution. If you’re on Fedora (or any closely-related RPM-based distribution, such as RHEL or CentOS), you can use dnf:

$ sudo dnf install git-all

If you’re on a Debian-based distribution, such as Ubuntu, try apt:

$ sudo apt install git-all

For more options, there are instructions for installing on several different Unix distributions on the Git website, at <https://git-scm.com/download/linux>.

$ git --version

output: git version 2.43.0



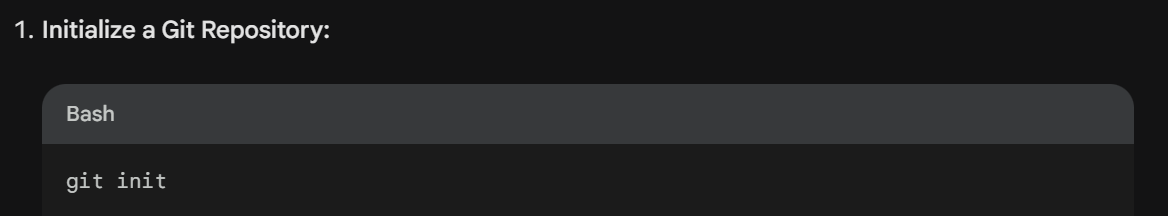
$which git

OUTPUT :/usr/bin/git

$ git --version

output: git version 2.43.0

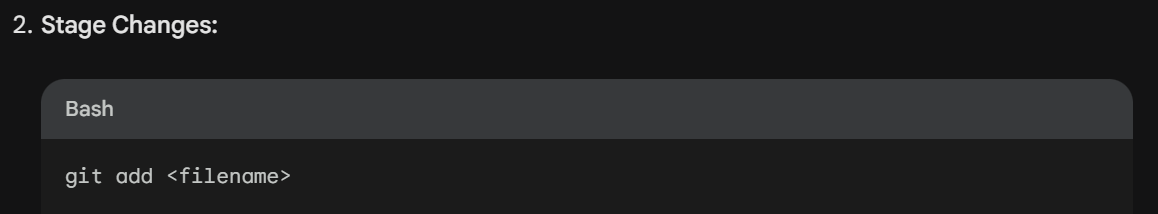
This command creates a new Git repository in the current directory, establishing a local workspace for tracking changes.



OUTPUT :

/home/Movit/desktop/.git/

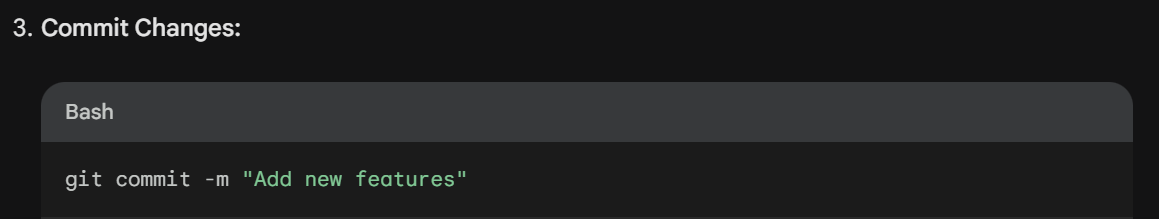
Staging marks files or changes for inclusion in the next commit. You can stage individual files or use wildcards to stage multiple files



# git add cloudguru

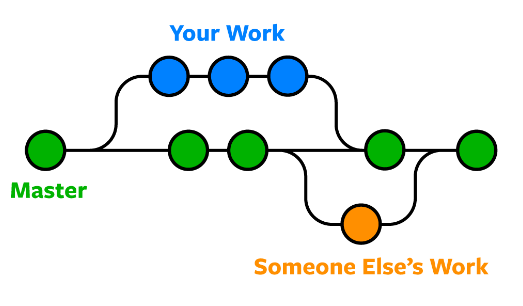
OUTPUT:

Committing saves the staged changes to the Git repository along with a descriptive message. The message explains the modifications made in the commit.



What is git branch in Git?

In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes



What is Git branch used for?

Branches allow you to work on different parts of a project without impacting the main branch.

When the work is complete, a branch can be merged with the main project. You can even switch between branches and work on different projects without them interfering with each other.

## For All the Commands Below

* Run this command:
  + **git status**

### List All Branches

**NOTE:**The current local branch will be marked with an asterisk (\*).

* To see **local branches**, run this command:
  + **git branch**
* To see **remote branches**, run this command:
  + **git branch -r**
* To see **all local and remote branches**, run this command:
  + **git branch -a**

### Create a New Branch

* Run this command (replacing **my-branch-name**with whatever name you want):
  + **git checkout -b my-branch-name**
* You're now ready to commit to this branch.

### Switch to a Branch In Your Local Repo

* Run this command:
  + **git checkout my-branch-name**

### Switch to a Branch That Came From a Remote Repo

1. To get a list of all branches from the remote, run this command:
   * **git pull**
2. Run this command to switch to the branch:
   * **git checkout --track origin/my-branch-name**

### Push to a Branch

* If your local branch **does not exist**on the remote, run either of these commands:
  + **git push -u origin my-branch-name**
  + **git push -u origin HEAD**

**NOTE:**HEAD is a reference to the top of the current branch, so it's an easy way to push to a branch of the same name on the remote. This saves you from having to type out the exact name of the branch!

* If your local branch **already exists**on the remote, run this command:
  + **git push**

### Merge a Branch

1. You'll want to make sure your working tree is clean and see what branch you're on. Run this command:
   * **git status**
2. First, you must check out the branch that you want to merge another branch into (changes will be merged into this branch). If you're not already on the desired branch, run this command:
   * **git checkout master**
   * **NOTE:** Replace **master**with another branch name as needed.
3. Now you can merge another branch into the current branch. Run this command:
   * **git merge my-branch-name**
   * **NOTE:** When you merge, there may be a conflict. Refer to **Handling Merge Conflicts**(the next exercise) to learn what to do.

### Delete Branches

* To delete a **remote branch**, run this command:
  + **git push origin --delete my-branch-name**
* To delete a **local branch**, run either of these commands:
  + **git branch -d my-branch-name**
  + **git branch -D my-branch-name**
* **NOTE:** The -d option only deletes the branch if it has already been merged. The -D option is a shortcut for --delete --force, which deletes the branch irrespective of its merged status.

**How to Create a Git Branch?**

* Creating a git branch is a concept used to work on the latest updates without changing existing code.
* Nowadays, git and many other different versions of control tools use Branching for a more effective software development process.
* Before starting the process of creating any new branch in our code, we want to take a pull of code to have the latest updated code.
* Then one can see a list of all existing branches in git by calling the command as:

git branch

* It will show all the branches present in the git repository.
* HEAD is used to show the status of on which branch currently user is working.
* So, it is totally different from HEAD. This works as a pointer to your local branch on which you are currently working.
* Git log is a simple command used to check on which branch currently the programmer is working.
* If a programmer wants to move from one branch to another, then one can hit a command like:

git checkout (branch-name)

* After all this process is done, then you want to push your code on the Git repository so you can call a command like:

git push origin <newly-created-branch-name>

* While you want to commit your latest updated code, you must first switch on the master branch from any other branch and then only commit your latest code [**using git checkout master**](https://www.educba.com/git-checkout-tag/).

**https://www.educba.com/what-is-git-branch/**

**How to Create a New Git Branch?**

There are multiple ways to create a new branch using Git.

We will see one by one, as mentioned below:

1. A most basic way to create a new branch using the following command:

git checkout –b <branch-name>

This is the most commonly used method for creating a branch to you from your recent branch, and it will change to your branch by using a single command itself.

2. Another way to create a branch is by specifying a particular branch name in command as follows:

git checkout –b new branch <branch-name>switched to branch <new-branch-name>

In the above scenario, you can define different branches through which another branch will be created.

3. One of the simple way to create a branch is as shown below:

git branch <branch-name>

By using this simplest command, one can easily create a new branch in git. In this case, the checkout will be done in a background process.

From the above steps, branch can’t automatically allow us to move on to a newly created branch. Initially, it puts us on our main branch like :

git branch

\* master

git branch Branch\_1

git branch

\* master

Branch\_1

If you want to update manually on the newly created branch, switch on this branch using the checkout command.

git checkout Branch\_1

Switched to branch Branch\_1.

4. Creating a branch through a Commit: This is another way to create a branch in git by defining a commit with its hash:

$ git branch <branch-name><hash>

With the help of hash git, it specifies some of the characters among them.

git branch

\* master

git branch commit-branch 635d5b3

git branch

commit-branch

\*master

This one command does both tasks of creating and checking out of the branch.

5. Creating a branch from a Tag: As we know, one creates a branch using a commit, the same as creating a branch using a tag.

* Its use for pointing something in the project’s code history.
* Here tags are working as an identifier.

Here is the syntax for creating a branch using the tag as follows:

git checkout –b <branch-name> <tax>

git branch tag-branch v0.2.23

git branch

tag-branch

\*master

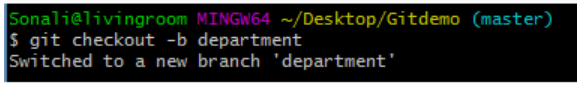
**Examples**

Let’s consider we are working on some projects with an updated commit, and we are on the master branch. Suppose we want to do changes in our latest code as per requirements, so we will not directly change the master branch.

We will create one new branch and switch directly to the newly created branch:

git branch department

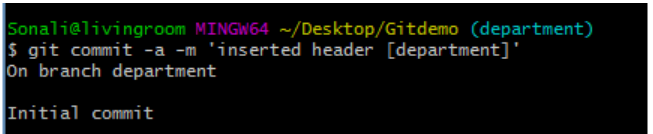
git checkout department



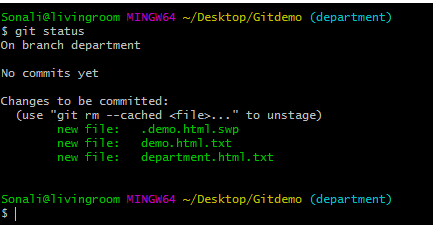
You worked on the code and saved the file to the department branch.



$ git commit –a –m 'inserted header [department]'

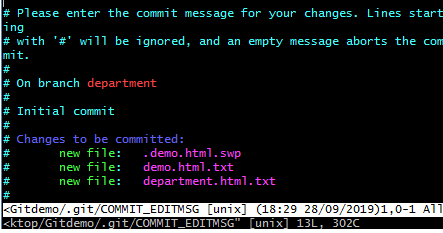


$ git status



* All changes are saved successfully. Next, let’s commit the code.

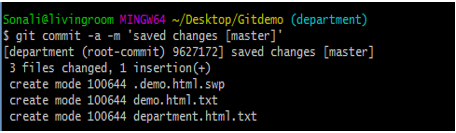
$ git commit



* As shown above code, we committed all the changes in our newly created branch department.
* Once it gets clear, everything is fine, now its time to commit all the code on the master branch.

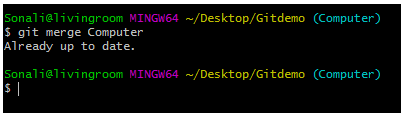
$ git commit –a –m 'saved changes [master]'

* By executing the above command, we can save all our latest code to the master branch of the git repository.

**Output:**

* Merging changes of branch code: It will show the latest updates.

Git merge [branch-name] = git merge Computer



* Suppose we have some unnecessary branches present in our project, and we want to remove those branches from the git repository.
* So this can happen with the help of the delete branch option as follows:

git branch –d [branch-name]

git branch –d Civil

