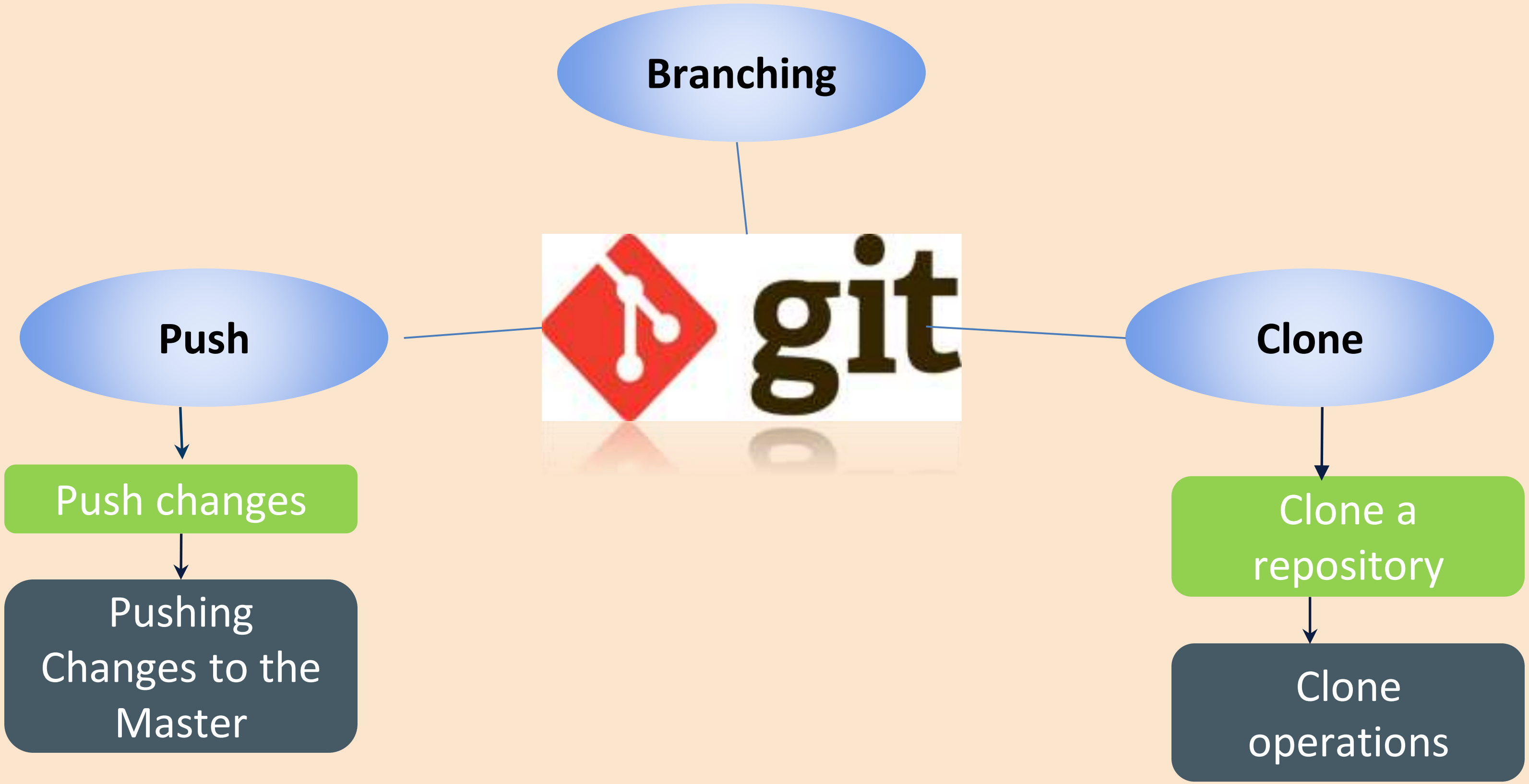
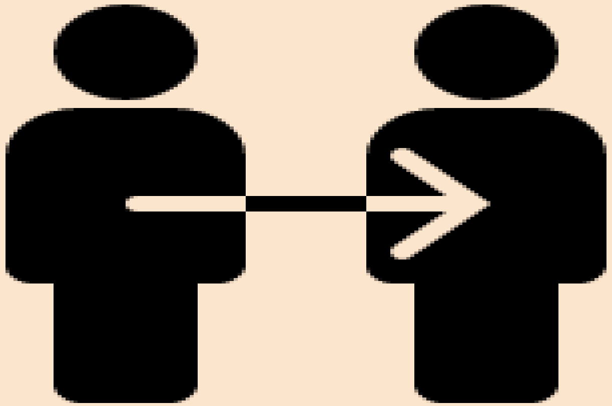
INTERACTION WITH GITHUB

CONTENTS

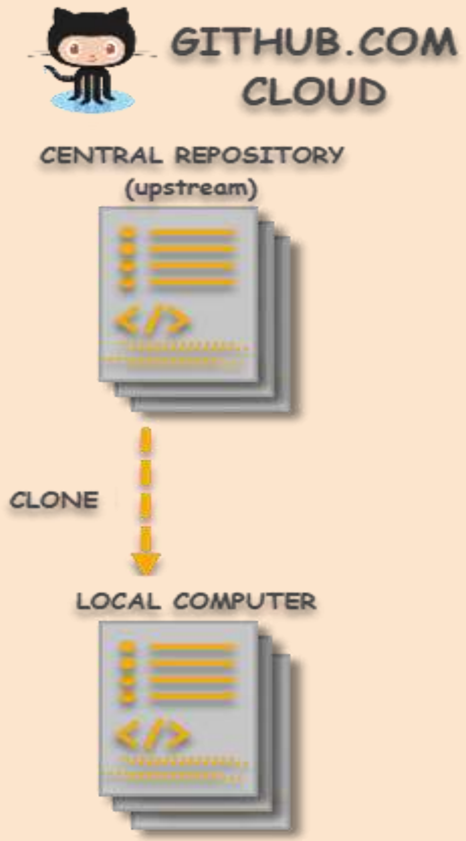

Branching
Push
Clone
Push changes
Pushing Changes to the Master
Clone a repository
Clone operations

CLONE A REPOSITORY



WHAT IS CLONING ?

Cloning is a process of creating an identical copy of a Git Remote Repository to the local machine.



Why clone a repository ?

- ❖ **Contribute to Organizational Projects:**
- *A centralized system is required for organizations where multiple people work on the same code base. Cloning helps us achieve this motive.*
- *By cloning, people can edit the project code to either fix some issue or provide some modifications i.e. an extra or extended feature.*
- ❖ **Make use of Open Source Repositories:**

If someone wants to use some functionality which has already been developed by someone else, then why to code it from scratch and waste time & resources? For e.g. there are unlimited open source repositories are available, which can directly fit into the projects.

Cloning an existing repository : git clone

Git clone is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository.

- **git clone**

```
git  clone  <repo>
```

Clone the repository <repo >on local machine.

- **Cloning a specific folder**

```
git clone <repo> <directory>
```

Clone the repository <repo> into the folder called <directory> on the local machine.

- **git clone –branch**

The -branch argument lets you specify a specific branch to clone instead of the branch the remote HEAD is pointing to, usually the master branch.

```
git  clone -branch  new_feature  git://remoterepository.git
```

This above example would clone only the new_feature branch from the remote Git repository

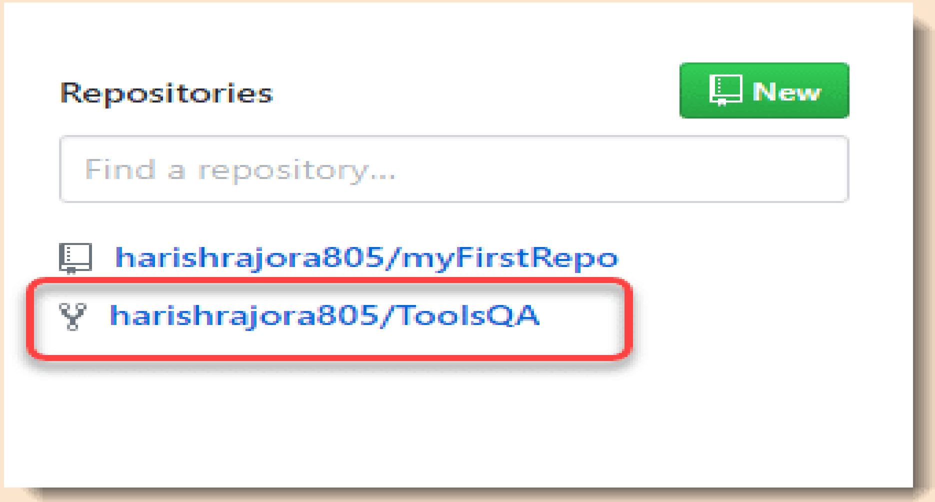
git clone --recurse-submodules

The --recurse-submodules with git clone command,will automatically initialize and update each submodule in the repository,including nested submodules.

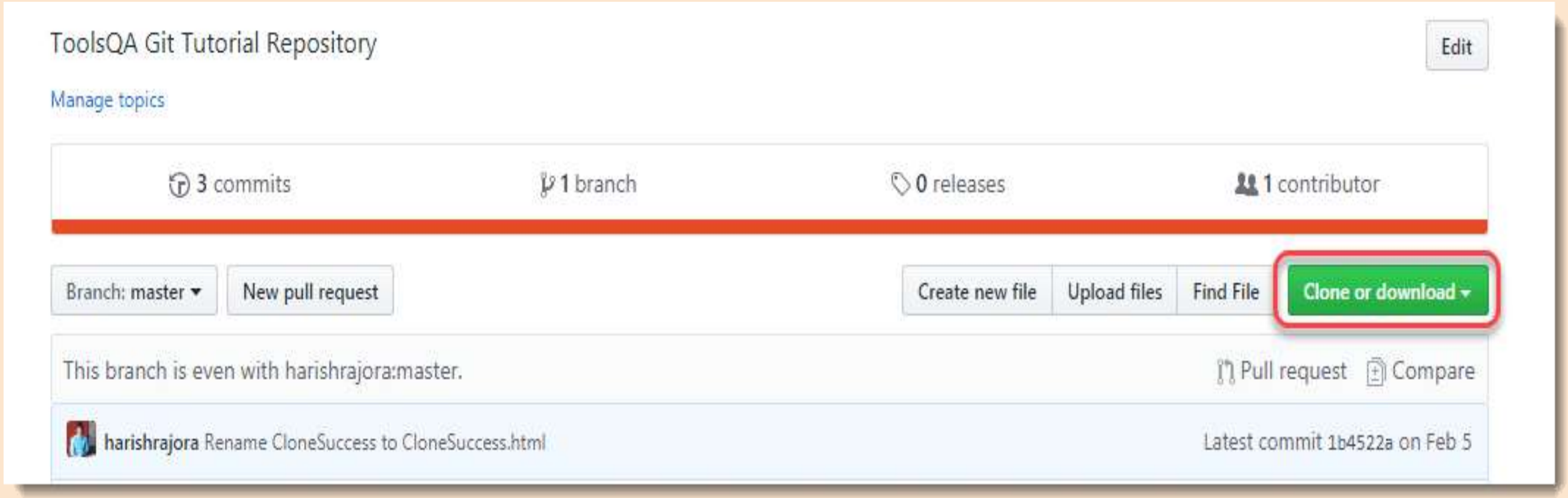
```
git  clone --recurse-submodules  https://github.com/ch/MainProject
```

How to Clone a Repository or use Git Clone Command?

- *To clone a repository, go to the repository page which you want to clone. This can be done through the side column on your dashboard.*



- *Press Clone or download button.*

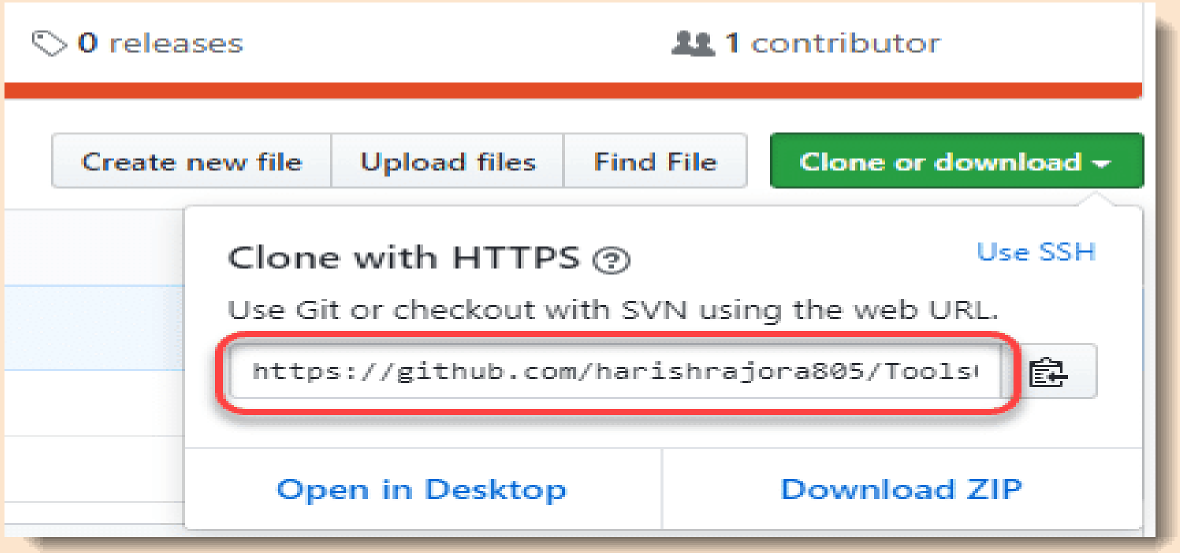


This branch is even with harishrajora:master.

harishrajora Rename CloneSuccess to CloneSuccess.html

Latest commit 1b4522a on Feb 5

- *Copy the code that appears after pressing the button.*

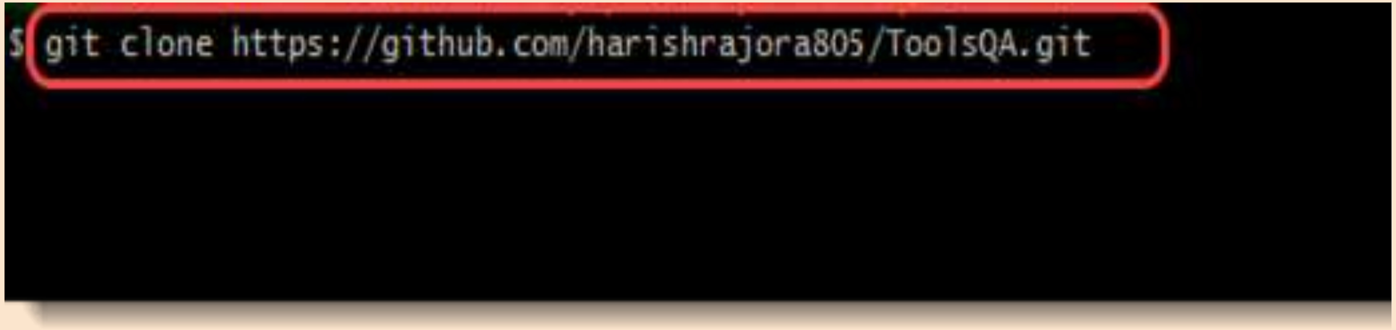


Use Git or checkout with SVN using the web URL.

```
https://github.com/harishrajora805/Tools
```

- *Once done, open Command prompt/Git bash on your system and press the following command to clone the repository:*

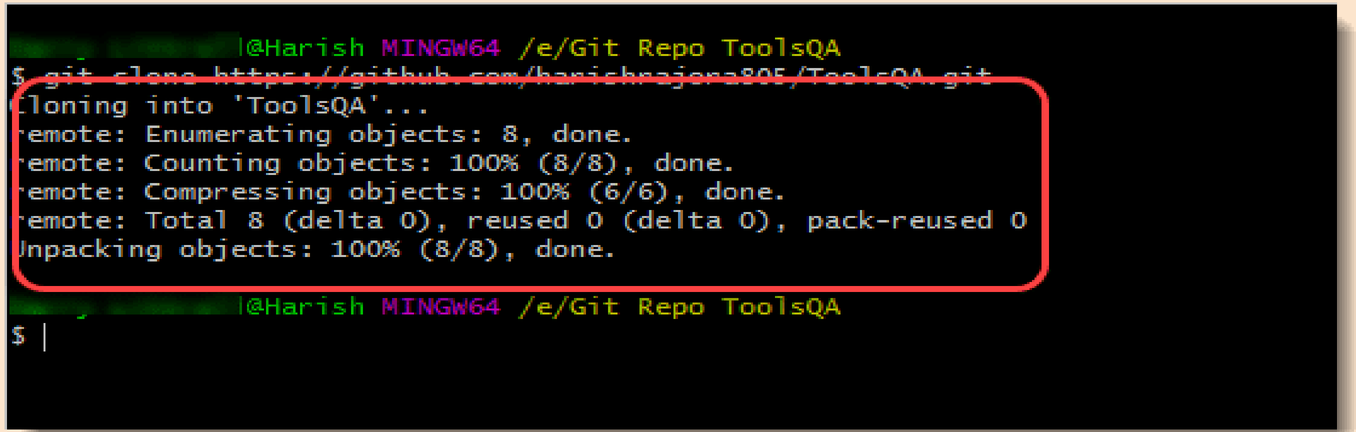
git clone <repo>



```
$ git clone https://github.com/harishrajora805/ToolsQA.git
```

Note: repo is the link of the repository over GitHub Cloud.

- *The following message will appear as you press enter.*

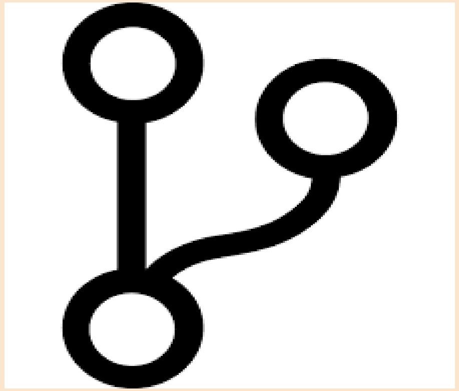


```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA
$ git clone https://github.com/harishrajora805/ToolsQA.git
Cloning into 'ToolsQA'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA
$ |
```

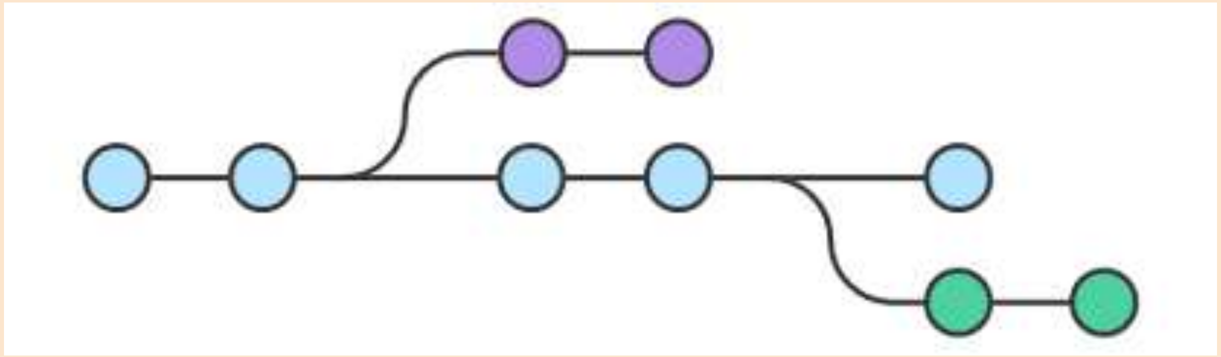
- *Confirm the cloning by listing the directories using the ls command which lists all the files and folder or check in the local drive by navigating to it manually.*

BRANCHING

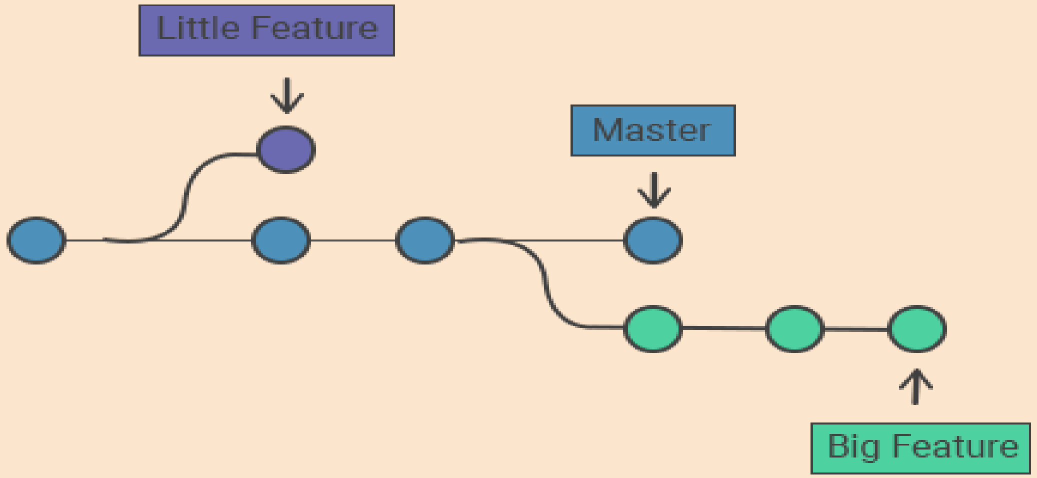


What is a Git Branch?

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process.



- *Branching enables you to isolate your work from others.*
- *Changes in the primary branch or other branches will not affect your branch, unless you decide to pull the latest changes from those branches.*



- *It is a common practice to create a new branch for each task (i.e., a branch for bug fixing, a branch for new features, etc.). This method allows others to easily identify what changes to expect and also makes backtracking simple.*

Why do we need a Branch in Git and Why Branches Are Important?

- *Git branches come to the rescue at many different places during the development of a project.*

- *Branches create another line of development that is entirely different or isolated from the main stable master branch.*

For example:

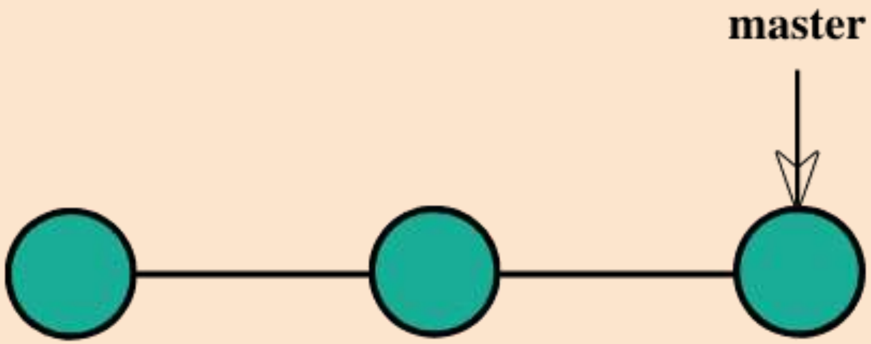
Consider that you are developing a project with your team, and you finish a feature. You contact the client to request them to see the feature, but they are too busy, so you send them the link to have a look at the project.

We can do the project in two ways:

1. Project Development through linear development

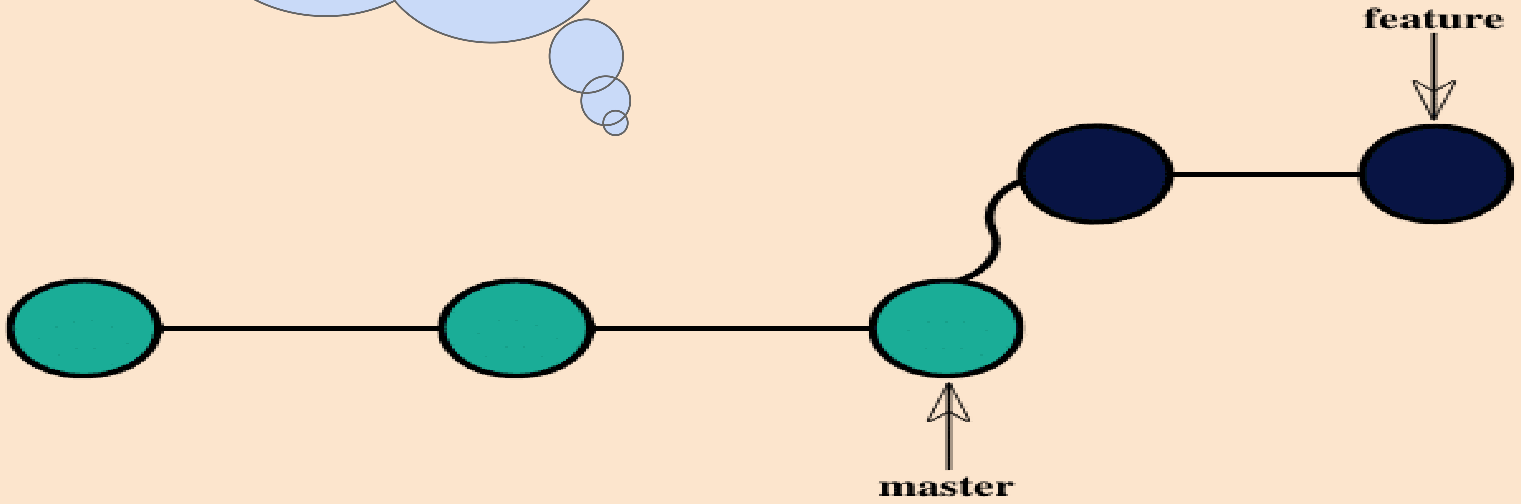
1. **Development through Branching :**
   - Branches give you the freedom to independently work on different modules (not necessarily though) and merge the modules when you finish developing them.
   - Git branches are swift to be created and destroy
   - Branches in Git help the team, which are in different parts of the world, work independently on independent features that would ultimately combine to produce a great project.

Developing the project through branching.



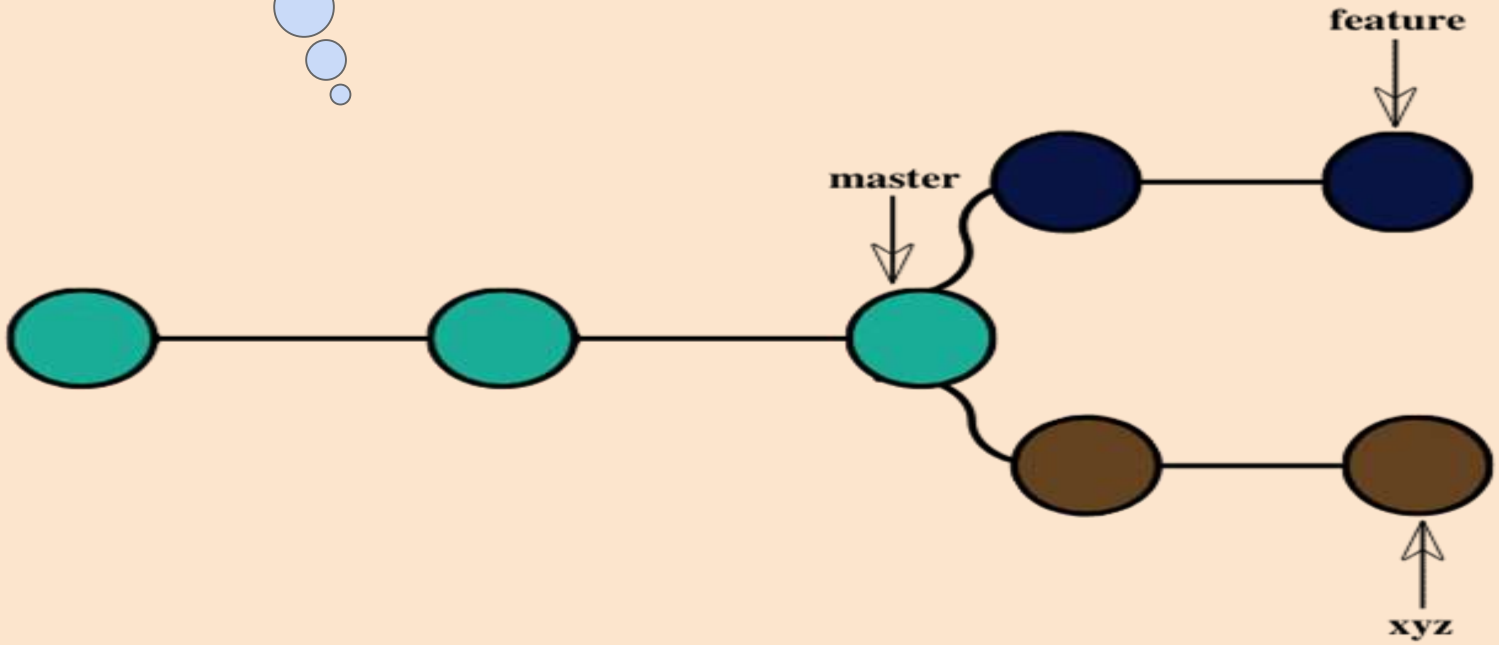
You have been working on a project with the client being happy until this point.

After that, you decide to develop a feature and create a new branch called feature for the same purpose and start working on it.



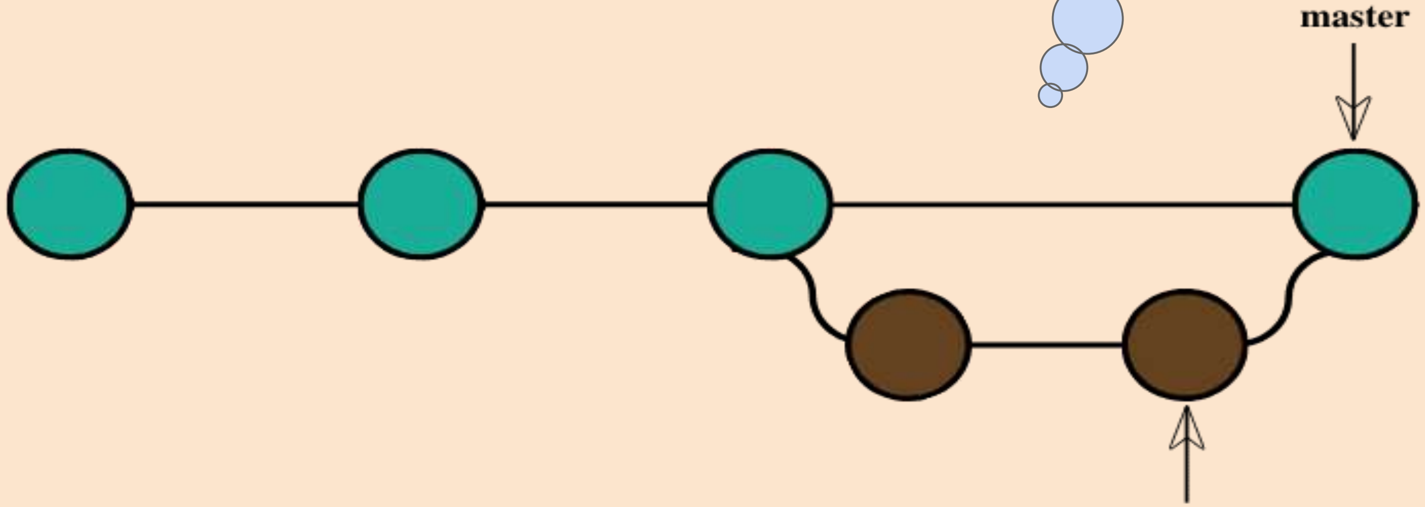
You show this to your client.

In the meantime, you decide to develop another feature and wait for the client's approval.



The client disapproves of the feature and requests to delete it.

Now, since you were following the branched strategy, you need to remove the branch, and all the remaining code remains as it is. The new feature can be easily added to the master branch to achieve the following.

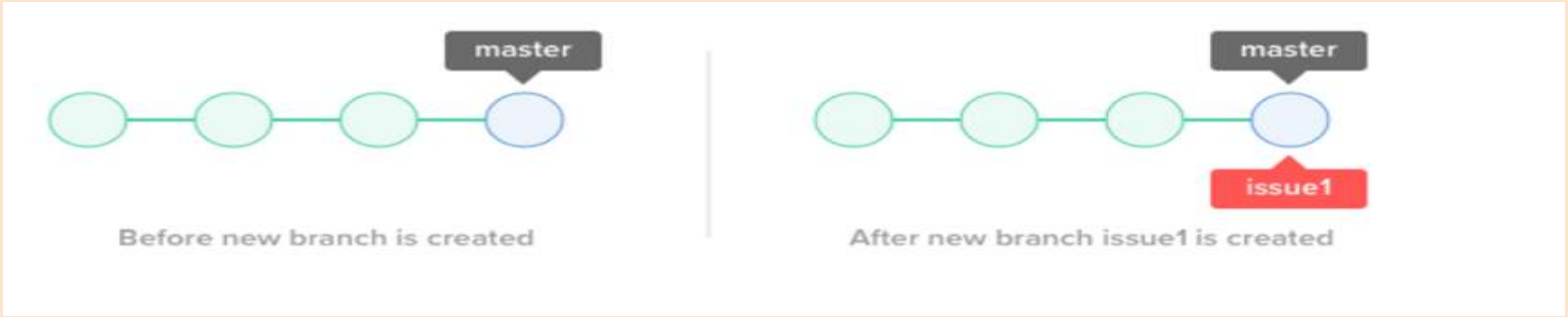


Create a Branch

- *Creating a new branch does not change the repository; it simply points out the commit.*
- *For example, let's create a branch called "issue1" using the command git branch.*

```
git branch issue1
```

The illustration below provides a visual on what happens when the branch is created. The repository is the same, but a new pointer is added to the current commit.


master
Before new branch is created
master
issue1
After new branch issue1 is created

Add Files

```
git add --        .
```

This adds your new files for git to track in the new branch.

Commit Changes

The parameter -a tells git to automatically stage files that have been modified and/or deleted. and -m tells git that you provide a commit message.

```
git commit -a -m 'Adding my files'
```

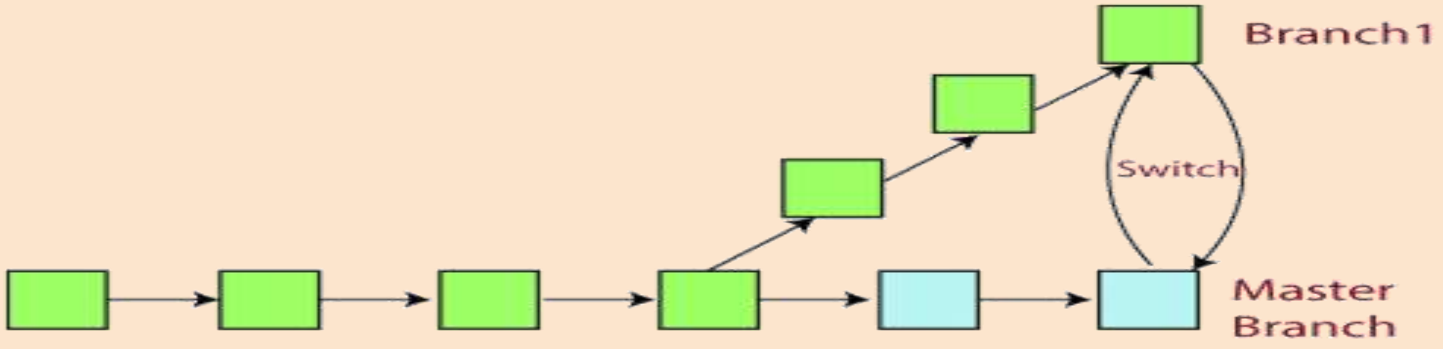
This commits the files.

Git Checkout

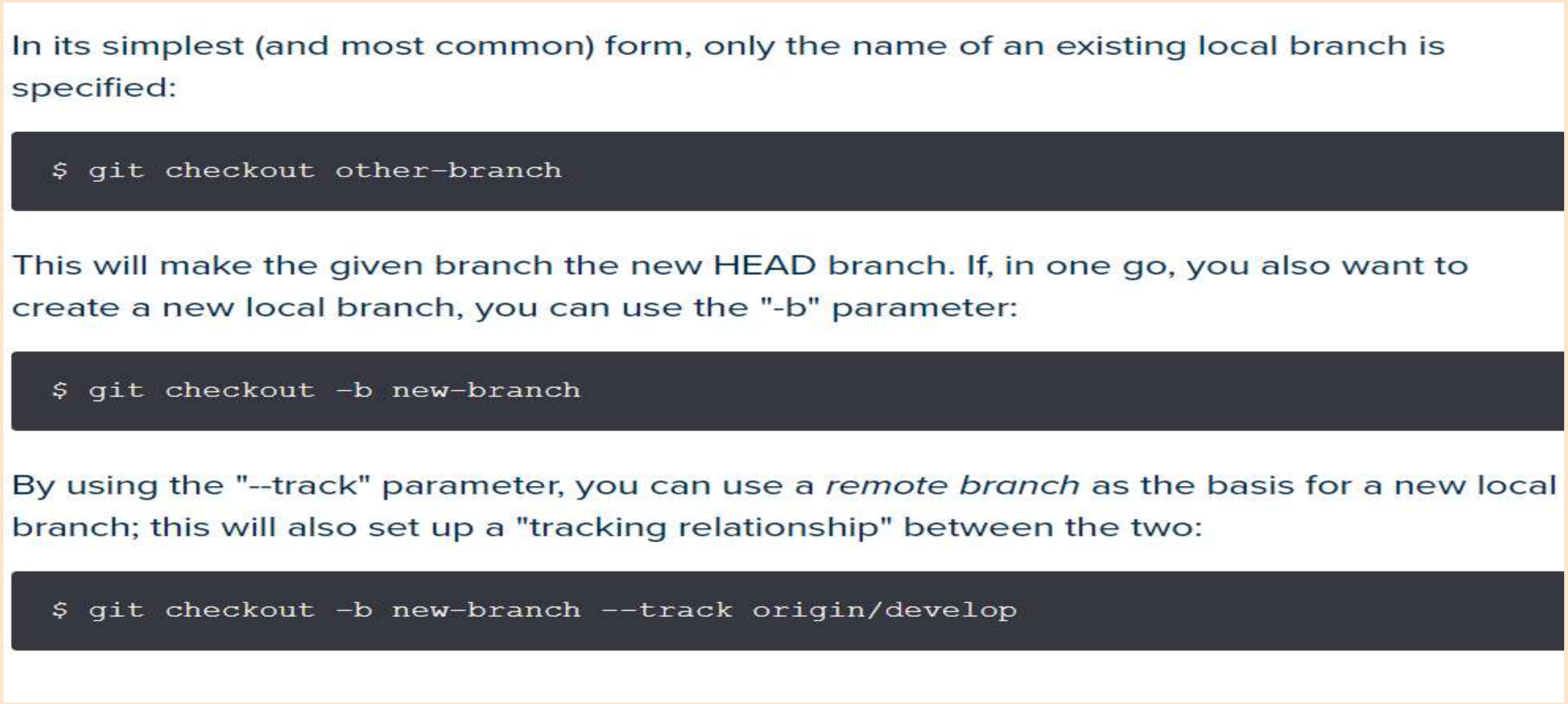
The "checkout" command can switch the currently active branch - but it can also be used to restore files.

The most common use case for "checkout" is when you want to switch to a different branch, making it the new HEAD branch.

```
git checkout -b
```



Usages



In its simplest (and most common) form, only the name of an existing local branch is specified:

```
$ git checkout other-branch
```

This will make the given branch the new HEAD branch. If, in one go, you also want to create a new local branch, you can use the "-b" parameter:

```
$ git checkout -b new-branch
```

By using the "--track" parameter, you can use a *remote branch* as the basis for a new local branch; this will also set up a "tracking relationship" between the two:

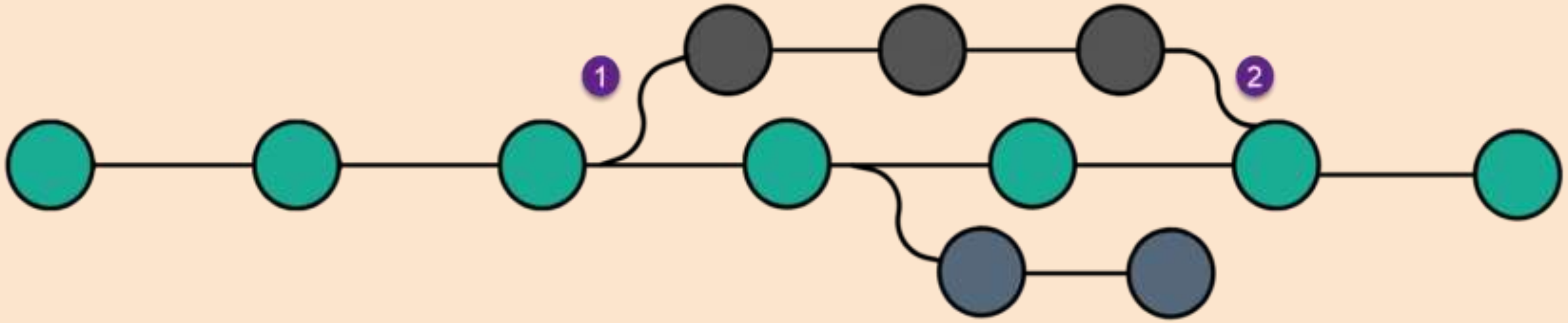
```
$ git checkout -b new-branch --track origin/develop
```

Merge a branch

- *A single repository generally contains multiple branches in Git.*
- *Git merge is used to combine two branches.*

A merge operation can be executed by typing the command

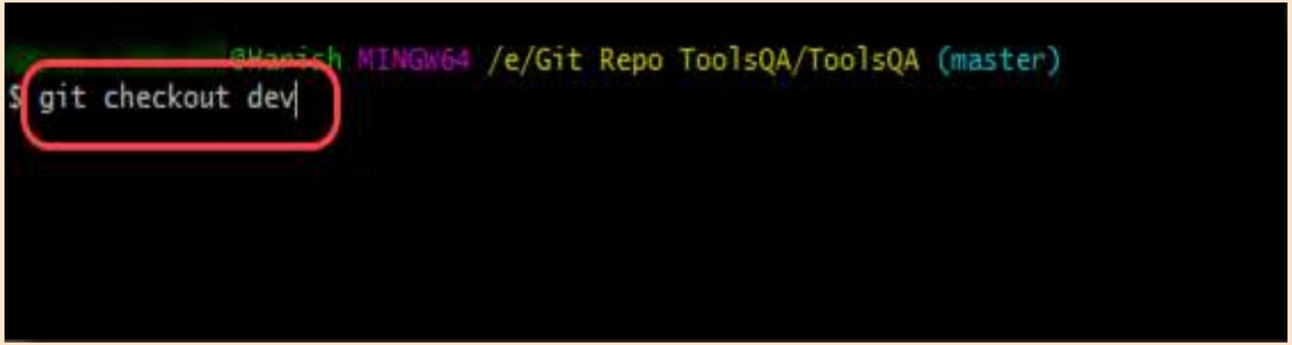
```
git merge <branch_name>
```



The top branch merges into the master branch after three commits.

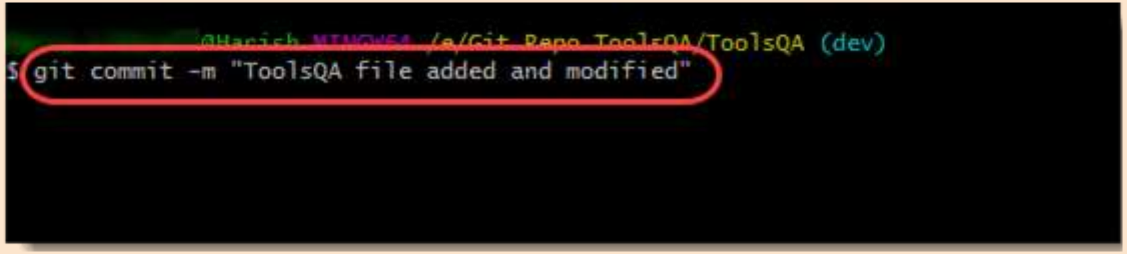
To merge branch in Git to another branch

- *To merge a branch called “**dev**” into another branch called the “**master**” branch.*
- *Before merging, we need to log some changes into the branch. For this, let’s switch to the branch that we will merge by the following command:*



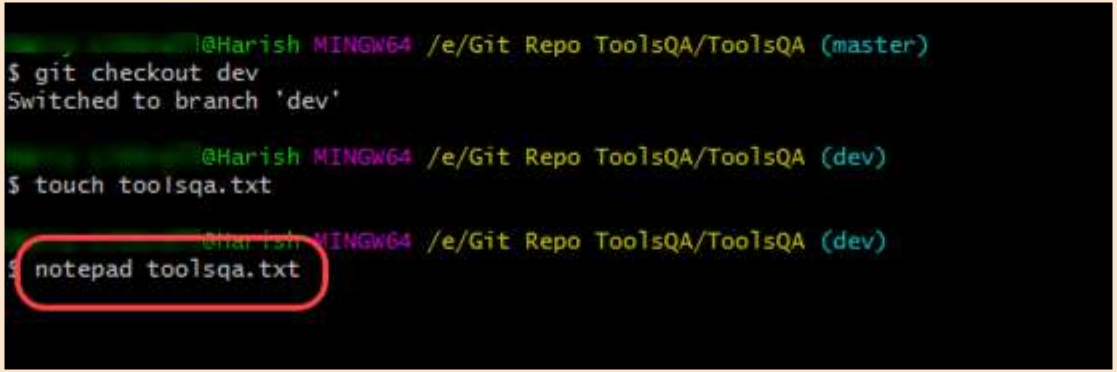
```
@Manish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git checkout dev
```

- *Now open a file using notepad and make some changes to the file. (Create a file if there is none).*



```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (dev)
$ git commit -m "ToolsQA file added and modified"
```

- *Close the file and commit these changes (After adding the file to the staging area).*

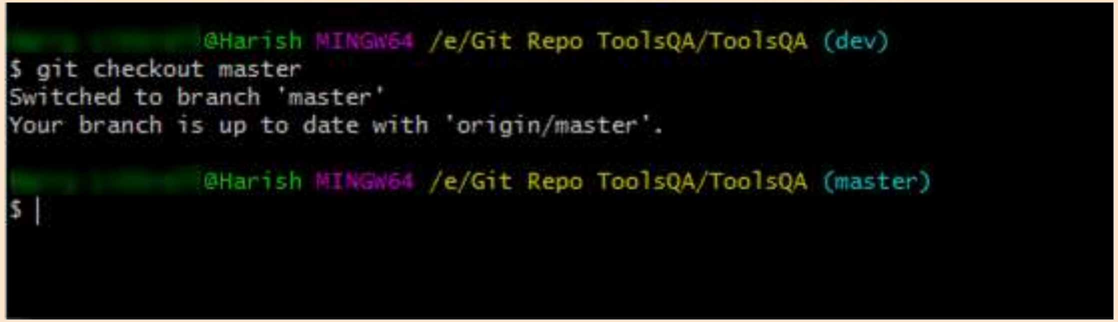


```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git checkout dev
Switched to branch 'dev'

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (dev)
$ touch toolsqa.txt

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (dev)
$ notepad toolsqa.txt
```

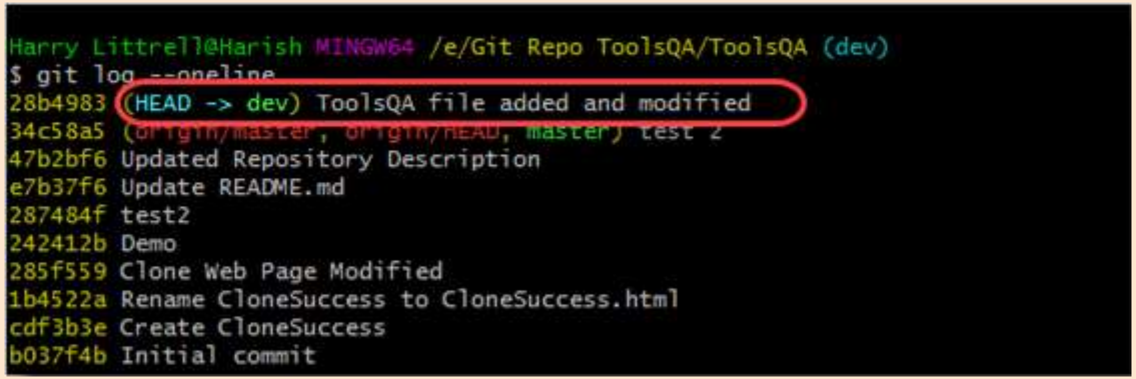
- *Perform Git Log operation to check the commit using oneline flag.*



```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (dev)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ |
```

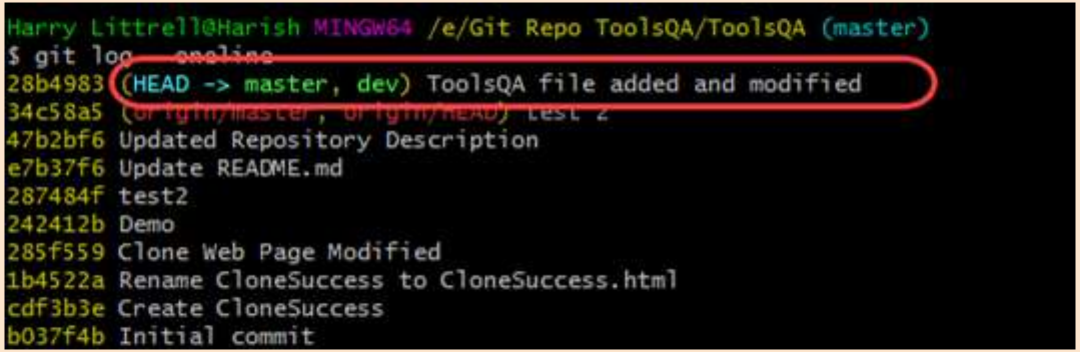
- *Now switch to the branch to which we will merge the changes (master in this example).*



```
Harry Littrell@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (dev)
$ git log --oneline
28b4983 (HEAD -> dev) ToolsQA file added and modified
34c58a5 (origin/master, origin/HEAD, master) test 2
47b2bf6 Updated Repository Description
e7b37f6 Update README.md
287484f test2
242412b Demo
285f559 Clone Web Page Modified
1b4522a Rename CloneSuccess to CloneSuccess.html
cdf3b3e Create CloneSuccess
b037f4b Initial commit
```

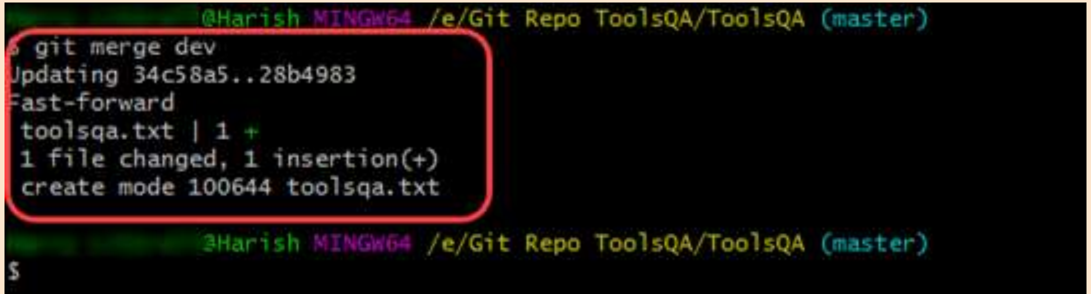
- *Execute the following command to merge the branch dev to the branch master.*

git merge <branch_name>



```
Harry Littrell@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git log --oneline
28b4983 (HEAD -> master, dev) ToolsQA file added and modified
34c58a5 (origin/master, origin/HEAD) test 2
47b2bf6 Updated Repository Description
e7b37f6 Update README.md
287484f test2
242412b Demo
285f559 Clone Web Page Modified
1b4522a Rename CloneSuccess to CloneSuccess.html
cdf3b3e Create CloneSuccess
b037f4b Initial commit
```

- *The output will show a successful merge along with the file name, i.e.,* ***toolsqa.txt.*** *Recheck the log to see the merge.*



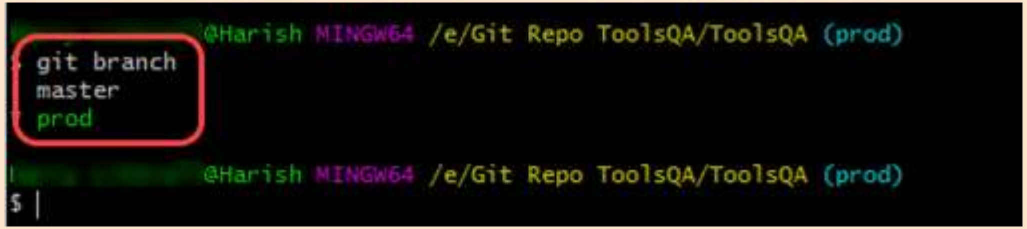
```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git merge dev
pdating 34c58a5..28b4983
ast-forward
 toolsqa.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 toolsqa.txt

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$
```

Delete a Branch

- *First list out all the branches in the local repository with the command:*

git branch



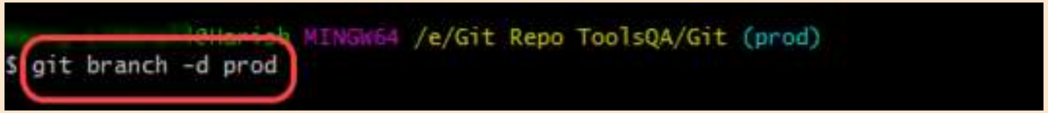
```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (prod)
$ git branch
  master
* prod

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (prod)
$ 
```

- *To delete a branch on your local system, follow these simple steps:*

  Type in the following command:

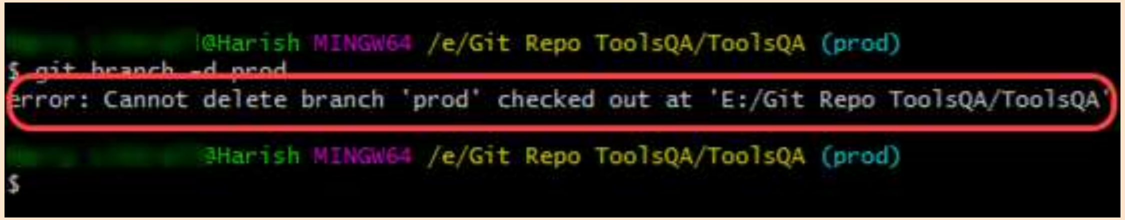
_git branch -d <branch_name>_



```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/Git (prod)
$ git branch -d prod
```

The “d” flag used here specifies that we intend to delete a branch.

Note: We can't delete a branch we are currently working on.



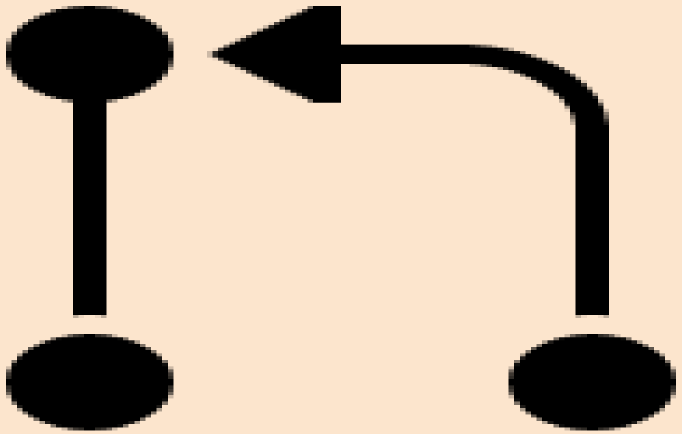
```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (prod)
$ git branch -d prod
error: Cannot delete branch 'prod' checked out at 'E:/Git Repo ToolsQA/ToolsQA'

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (prod)
$
```

- *Git suggests us to "check out" from this branch and try deleting again.*

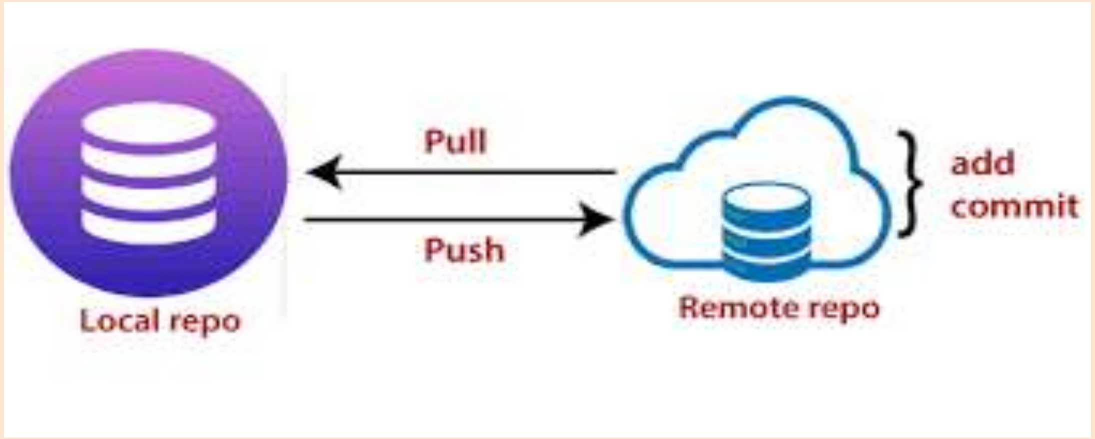
- *We can use* ***git branch -d <branch_name>*** *to force delete a branch without checking the merged status of the branch.*

PUSH AND PULL



PUSHING

- *Pushing sends your changes up to GitHub, so they can be shared with the rest of the world*
- *It also serves as a hedge against data loss.*



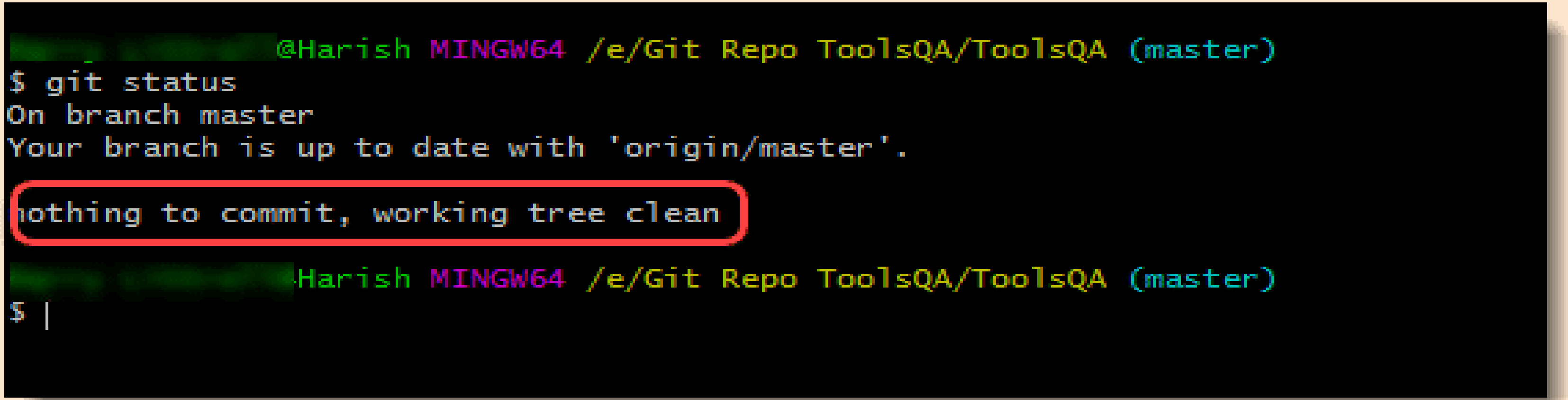
The simple command to PUSH from a branch is:

```
git push 'remote_name' 'branch_name'
```

- *`git push` command is used to transfer or push the commit.*
- *`remote_name` is the name of the remote repository to which we are pushing the changes.*
- *`branch_name` is the branch the user is pushing to the remote repository.*

HOW TO PUSH ?

- *First, check that you have a clean repository through git status command (no pending changes to commit).*



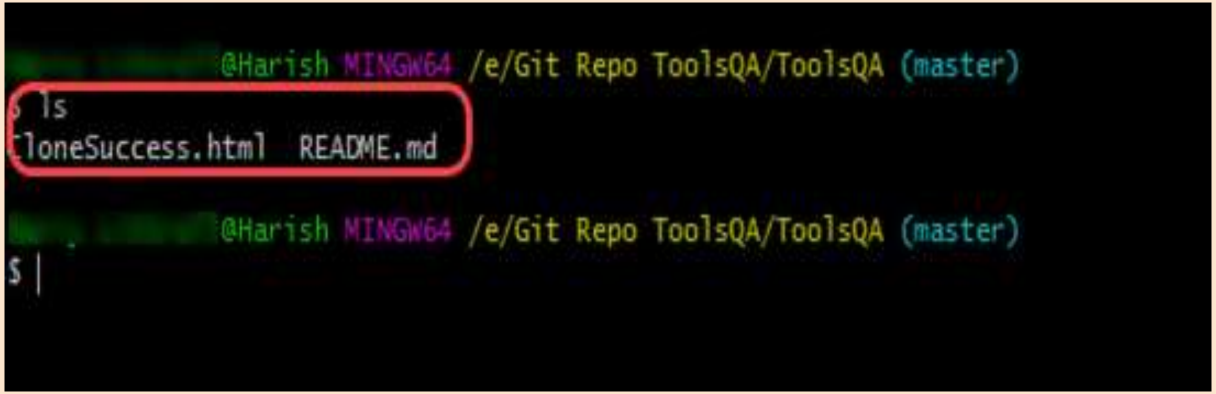
```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

othing to commit, working tree clean

[redacted]Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ |
```

- ***On branch master :*** *Denotes that we are currently on the 'master' branch. Since there are no other branches yet, we are on the master branch by default.*
- ***Your branch is up to date with origin/master :*** *Origin is the name of the remote repository that we gave while connecting the local repository with the remote repository.*

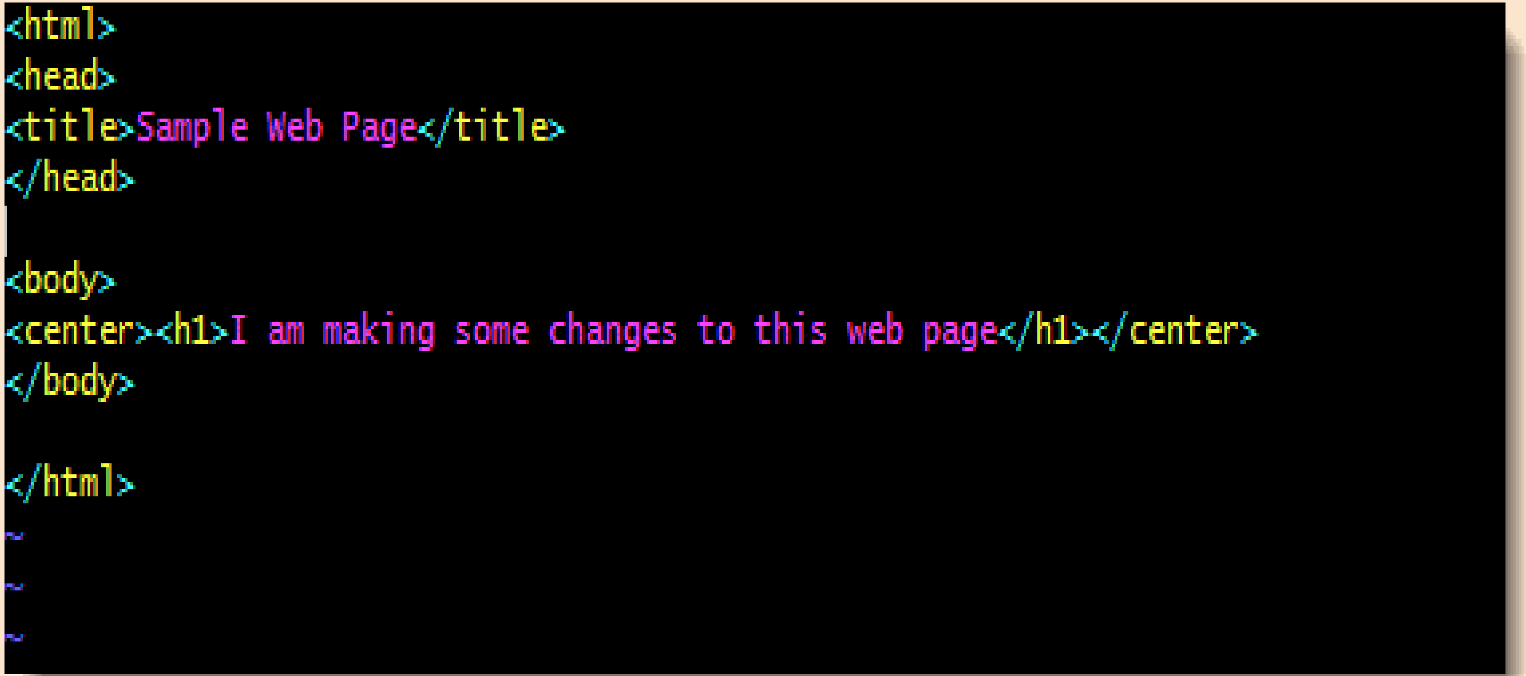
1. *List all the files with the ls command in the repository.*



```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ ls
CloneSuccess.html  README.md

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ |
```

- *Since there is only one file (README.md i). let's make some changes to its content.*

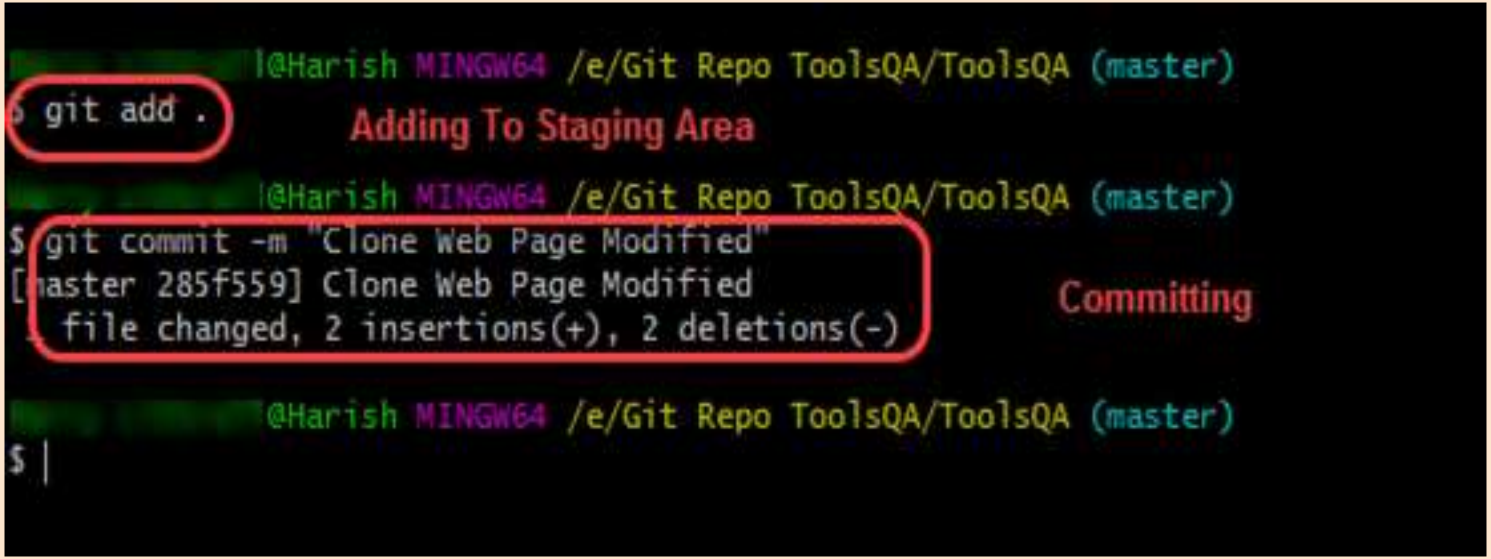


```
<html>
<head>
<title>Sample Web Page</title>
</head>

<body>
<center><h1>I am making some changes to this web page</h1></center>
</body>

</html>
~
~
~
```

2. *Add the changes made to the staging area and commit these changes.*



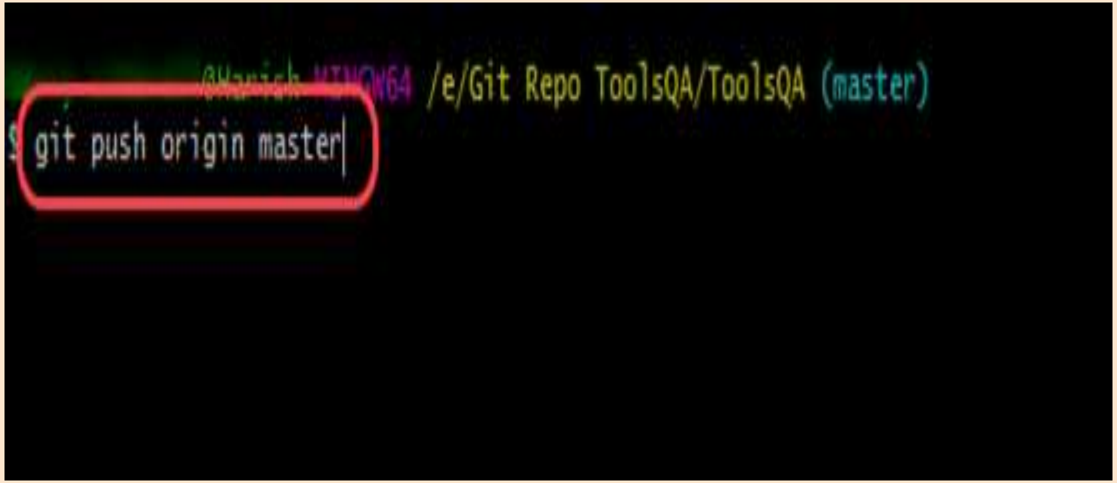
```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git add .

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git commit -m "Clone Web Page Modified"
[master 285f559] Clone Web Page Modified
 1 file changed, 2 insertions(+), 2 deletions(-)

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ |
```

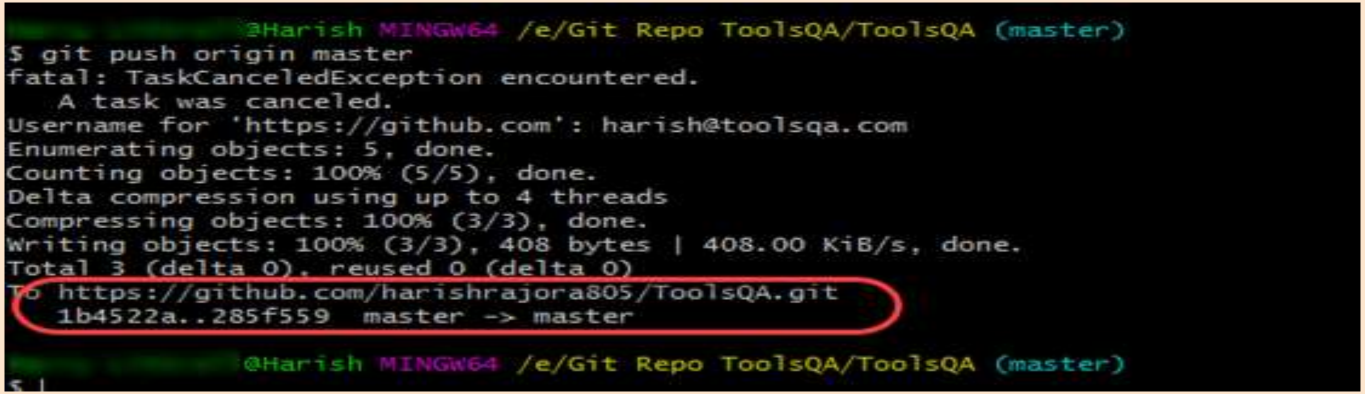
3. *Type the following command to push these changes into your GitHub repository and press enter.*

git push origin master



```
@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git push origin master
```

4. *User will get the following message in Git Bash.*



```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git push origin master
fatal: TaskCanceledException encountered.
   A task was canceled.
Username for 'https://github.com': harish@toolsqa.com
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 408 bytes | 408.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/harishrajora805/ToolsQA.git
   1b4522a..285f559  master -> master

[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ |
```

- *master -> master: The line master-> master shows the source branch from which merging happens to the destination branch. In the above scenario, both are master branches.*
- *Writing Objects: 100% all the changes have been successfully pushed onto the cloud.*

git status

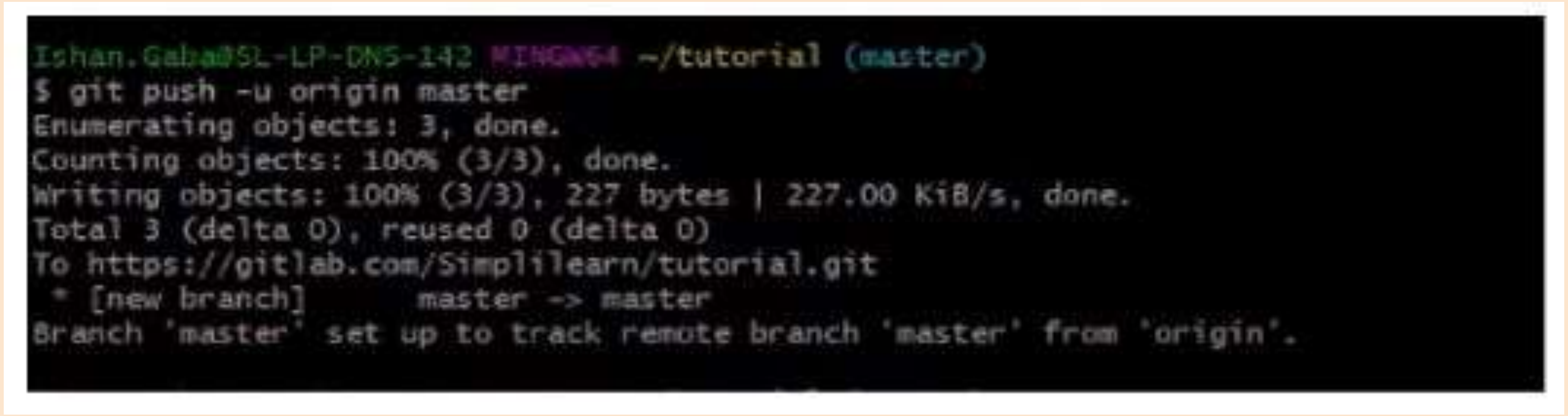
- *Status tells us we have no local changes pending,but that the remote is one commit behind.*
- *This isn't actually checking against the remote repository, just letting us know how many   times we've committed since our last push.*

git remote -v

- *Check out where our remote target is.We will revisit this in a bit,when we start working on syncing to the original repository.*

git push -u origin master

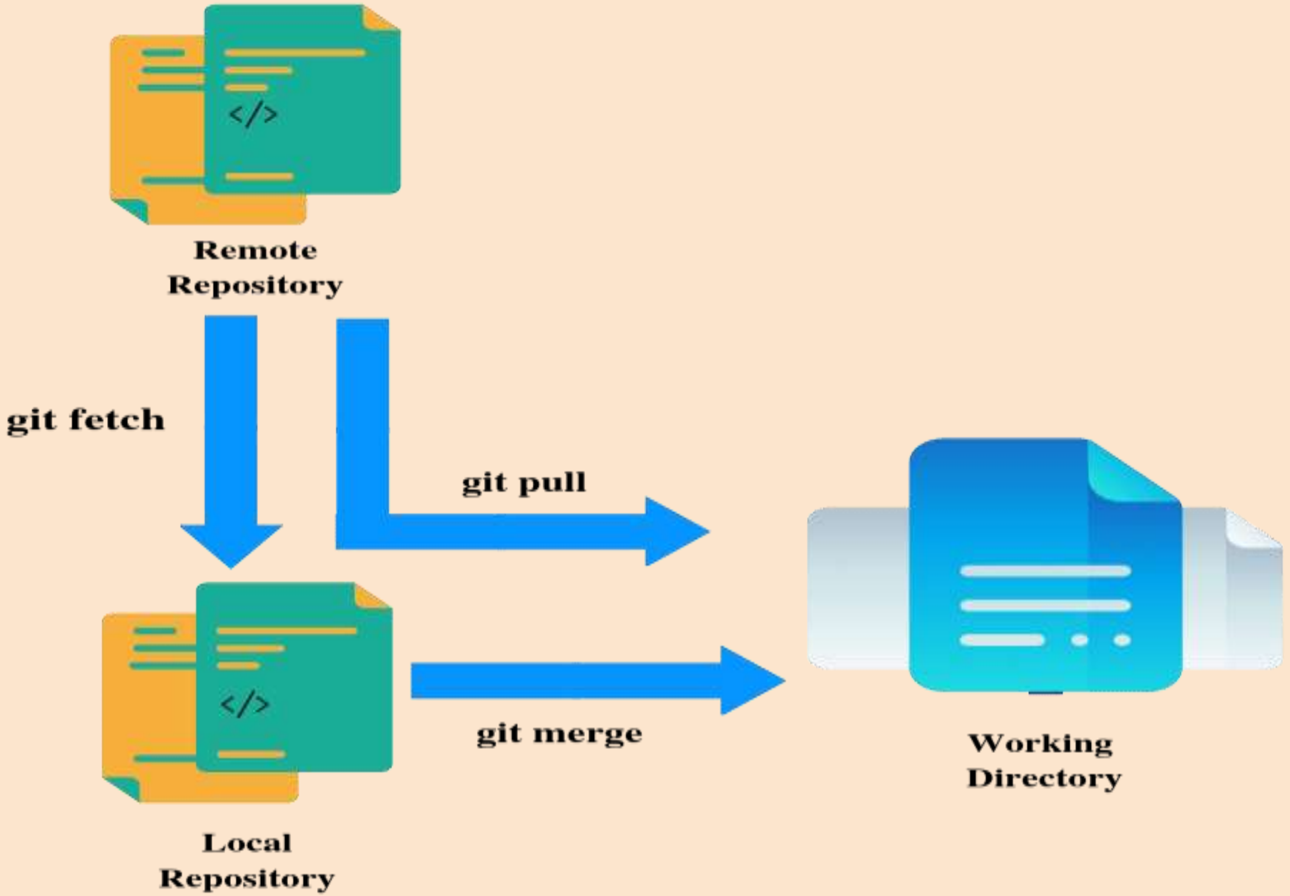
- *push the changes in branch master to remote location origin, and remember the settings (-u).*



```
Ishan.Gaba@SL-LP-DNS-142 MINGW64 ~/tutorial (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 227 bytes | 227.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://gitlab.com/Simplilearn/tutorial.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

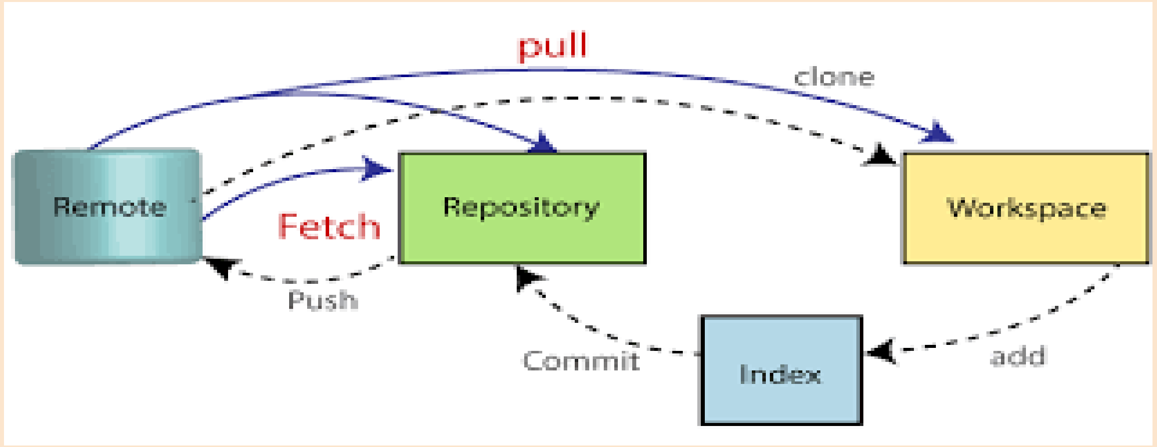
Git Fetch Command

- *git fetch command helps the user download commits, refs, and files from the remote repository to the local repository.*
- *Git fetch is a great way to stand at a place from where you can see the changes and decide if you want to keep them or discard them.*

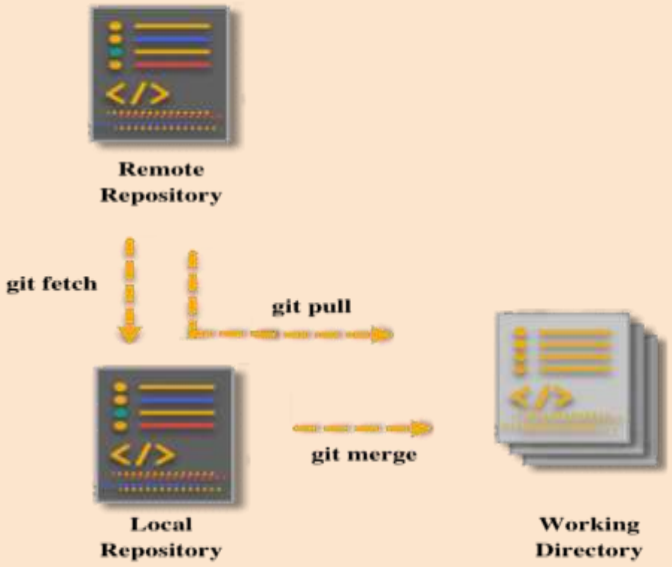

Remote Repository
git fetch
git pull
Local Repository
git merge
Working Directory

PULLING

- *opposite of pushing*
- *retrieves changes from the remote location and applies them to your local repository*
- *more useful in a group environment where more than one person is submitting changes to a single repository*



Git pull: *a magical way to perform a combined operation of git-fetch & git-merge with a single command*



The simple command to PULL from a branch is :

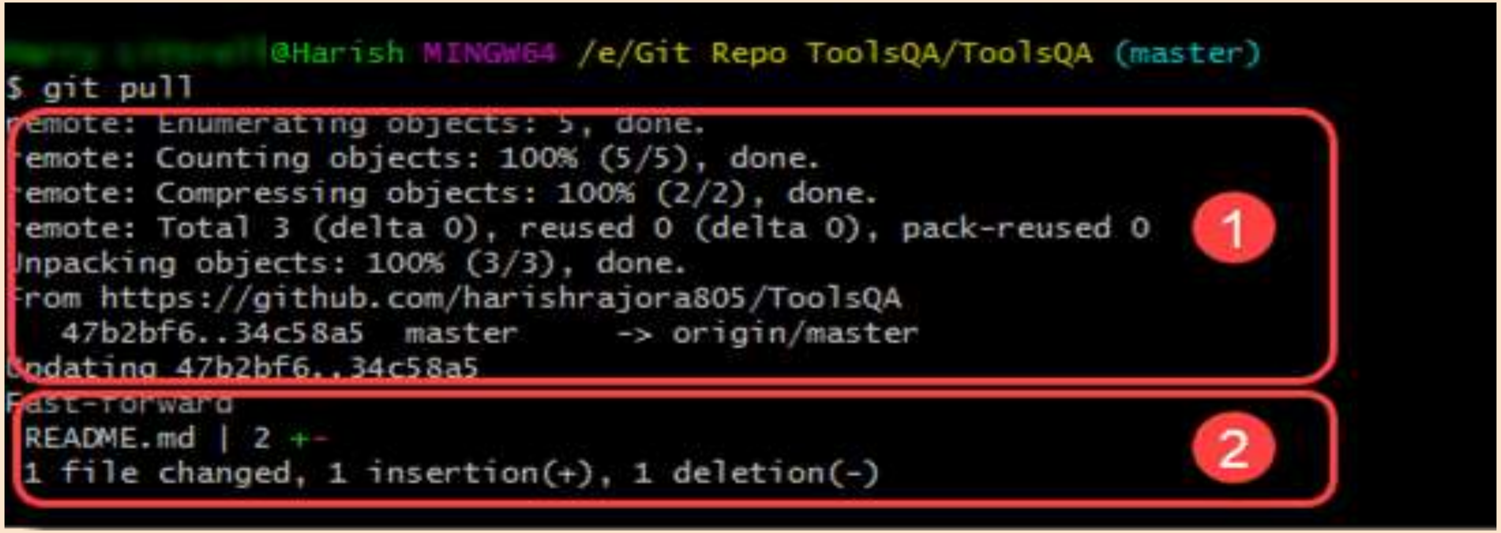
```
git pull 'remote_name' 'branch_name'.
```

- The git pull command is a combination of git fetch
- 'remote_name' is the repository name and 'branch_name' is the name of the specific branch.

HOW TO PULL?

We can use the git pull command by typing the following command in the Git Bash.

git pull



```
[redacted]@Harish MINGW64 /e/Git Repo ToolsQA/ToolsQA (master)
$ git pull
remote: Enumerating objects: 5, done.
emote: Counting objects: 100% (5/5), done.
emote: Compressing objects: 100% (2/2), done.
emote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Jnpacking objects: 100% (3/3), done.
rom https://github.com/harishrajora805/ToolsQA
   47b2bf6..34c58a5  master     -> origin/master
Jpdating 47b2bf6..34c58a5
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Cont.

The first section has the same output as the git fetch command whereas the second section has the same output as the git merge command. It proves git pull is an amalgam of git fetch and git merge command.

THANK YOU