GIT: **Problems without using code version tool:**

**Problem 1:** If you machine crashes, then complete code will be lost.

**Solution:** You can write data to cloud.

Graphical user interface

Description automatically generated with medium confidence Diagram

Description automatically generated

**Problem 2:** Only one person can work on same code.

**Solution:** Both person can update same code.

Diagram

Description automatically generated Diagram

Description automatically generated

**Problem 3:** Difficult to track the changes.

**Solution:** Different versions of code is available.

Diagram, text

Description automatically generatedDiagram

Description automatically generated

**Pull request concept:** One can copy code from git hub code repo and will do changes. Person can send request to pull changes for review. After review, pull request will be accepted and code will be merged.

Diagram

Description automatically generated

**Git Basics:**

**Create new repository**: Login to GitHub and create new repository.

**Copy repository means clone the repository.**

git clone https://github.com/ashukla2019/test.git

**Check status of repository: git status**

Text

Description automatically generated

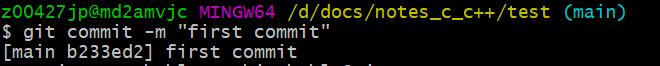
**Create new file to reposirtory:** Add new file to repository.

Now do git status, it will show you untracked file.

Text

Description automatically generated

When we add file, we require to add it in **Staging Area (means file is ready for commit now) and can commit file. Now file is committed to local version control system.**



**Git log:** This command will show you log of commit.

Diagram

Description automatically generated

Changes can be reverted. We check the log and then we will get commit id. Use git revert -n “**commit\_id**”. Then commit changes.

**Branches:** Create branch, merge branch, and delete branch

**List all the branches**: git branch

**Create new branch:** git branch xyz

Text

Description automatically generated with medium confidence

**Here main branch is active and if you want to perform operation to xyz branch, should make it active. For making branch active:**

**Git checkout xyz -> will switch to xyz branch**

Text

Description automatically generated

**Now modify into new xyz branch:**

Do changes in file and then add and commit it.

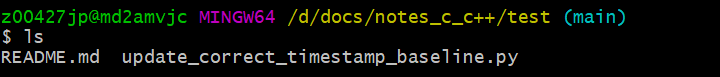
Text

Description automatically generated

Text

Description automatically generated

When switched back to main branch, no modification is present:



**Merge xyz branch to main branch:**

Switch to main branch and run merge:

Text

Description automatically generated

Diagram

Description automatically generated

**Delete branch:** git branch -d <branch-name>

Here are the Git commands which are being covered:

* **git config**
* **git init**
* **git clone**
* **git add**
* **git commit**
* **git diff**
* **git reset**
* **git status**
* **git rm**
* **git log**
* **git show**
* **git tag**
* **git branch**
* **git checkout**
* **git merge**
* **git remote**
* **git push**
* **git pull**
* **git stash**

So, let's get started!

**Git Commands**

**git config**

Usage: git config –global user.name “[name]”

Usage: git config –global user.email “[email address]”

This command sets the author name and email address respectively to be used with your commits.

Git Config Command - Git Commands - Edureka

**git init**

Usage: git init [repository name]

This command is used to start a new repository.

GitInit Command - Git Commands - Edureka

**git clone**

Usage: git clone [url]

This command is used to obtain a repository from an existing URL.



**git add**

Usage: git add [file]

This command adds a file to the staging area.

Git Add Command - Git Commands - Edureka

Usage: git add \*

This command adds one or more to the staging area.

Git Add Command - Git Commands - Edureka

**git commit**

Usage: git commit -m “[ Type in the commit message]”

This command records or snapshots the file permanently in the version history.



Usage: git commit -a

This command commits any files you’ve added with the git add command and also commits any files you’ve changed since then.

Git Commit Command - Git Commands - Edureka

**git diff**

Usage: git diff

This command shows the file differences which are not yet staged.



 Usage: git diff –staged

This command shows the differences between the files in the staging area and the latest version present.



Usage: git diff [first branch] [second branch]

This command shows the differences between the two branches mentioned.



**git reset**

Usage: git reset [file]

This command unstages the file, but it preserves the file contents.



Usage: git reset [commit]

This command undoes all the commits after the specified commit and preserves the changes locally.

Git Reset Command - Git Commands - Edureka

Usage: git reset –hard [commit]  This command discards all history and goes back to the specified commit.

Git Reset Command - Git Commands - Edureka

**git status**

Usage: git status

This command lists all the files that have to be committed.



**git rm**

Usage: git rm [file]

This command deletes the file from your working directory and stages the deletion.

Git Rm Command - Git Commands - Edureka

**git log**

Usage: git log

This command is used to list the version history for the current branch.



Usage: git log –follow[file]

This command lists version history for a file, including the renaming of files also.



**git show**

Usage: git show [commit]

This command shows the metadata and content changes of the specified commit.



**git tag**

Usage: git tag [commitID]

This command is used to give tags to the specified commit.



**git branch**

Usage: git branch

This command lists all the local branches in the current repository.

Git Branch Command - Git Commands - Edureka

Usage: git branch [branch name]

This command creates a new branch.

Git Branch Command - Git Commands - Edureka

Usage: git branch -d [branch name]

This command deletes the feature branch.

Git Branch Command - Git Commands - Edureka

**git checkout**

Usage: git checkout [branch name]

This command is used to switch from one branch to another.

Git Checkout Command - Git Commands - Edureka

Usage: git checkout -b [branch name]

This command creates a new branch and also switches to it.

Git Checkout Command - Git Commands - Edureka

**git merge**

Usage: git merge [branch name]

This command merges the specified branch’s history into the current branch.

Git Merge Command - Git Commands - Edureka

**git remote**

Usage: git remote add [variable name] [Remote Server Link]

This command is used to connect your local repository to the remote server.

Git Remote Command - Git Commands - Edureka

**git push**

Usage: git push [variable name] master

This command sends the committed changes of master branch to your remote repository.



Usage: git push [variable name] [branch]

This command sends the branch commits to your remote repository.



Usage: git push –all [variable name]

This command pushes all branches to your remote repository.



Usage: git push [variable name] :[branch name]

This command deletes a branch on your remote repository.



**git pull**

Usage: git pull [Repository Link]

This command fetches and merges changes on the remote server to your working directory.



**git stash**

Usage: git stash save

This command temporarily stores all the modified tracked files.

Git Stash Command - Git Commands - Edureka

Usage: git stash pop

This command restores the most recently stashed files.



Usage: git stash list

This command lists all stashed changesets.

Git Stash Command - Git Commands - Edureka

Usage: git stash drop

This command discards the most recently stashed changeset.

Git Stash Command - Git Commands - Edureka

# [**Add tag to release**](https://confluence.veritas.com/pages/viewpage.action?pageId=389894109)

1) git checkout release/a.b.c branch

2) git -a <tag\_name> -m message( git -a a.b.c\_x -m "created tag a.b.c\_x")

3) git push origin : <tag\_name>( git push origin a.b.c\_x)

# [**Rebase release a.b.c to master**](https://confluence.veritas.com/display/~Ankit.Shukla/Rebase+release+3.7.7+to+master)

1) Git clone master.

2) git checkout release/a.b.c

3) Create new branch from a.b.c

4) git rebase master

# [**Squash commits from my branch**](https://confluence.veritas.com/display/~Ankit.Shukla/Squash+commits+from+my+branch)

 1) Checkout your branch

2) checkout target branch to local system

3) git rebase -i target\_branch(release\x.y.z)

4) will show change log in VI/VIM editor

5) Change n-1 pick to s or squah, then save it

6) return back from VI editor screen

7) git push -f origin my\_branch