The Git config command

**Setting username**

The username is used by the Git for each commit.

$ git config --global user.name "Himanshu Dubey"

**Setting email id**

The Git uses this email id for each commit.

$ git config --global user.email  "himanshudubey481@gmail.com"

### **Git Init command**

This command is used to create a local repository.

**Syntax**

$ git init Demo

### **Git clone command**

This command is used to make a copy of a repository from an existing URL. If I want a local copy of my repository from GitHub, this command allows creating a local copy of that repository on your local directory from the repository URL.

**Syntax**

$ git clone https://github.com/rajat73125/git1.git

### **Git add command**

This command is used to add one or more files to staging (Index) area.

**Syntax**

To add one file

$ git add Filename

To add more than one file

$ git add\*

### **Git commit command**

Commit command is used in two scenarios. They are as follows.

**Git commit -m**

This command changes the head. It records or snapshots the file permanently in the version history with a message.

**Syntax**

$ git commit -m " Commit Message"

**Git commit -a**

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

**Syntax**

$ git commit -a

### **Git status command**

The status command is used to display the state of the working directory and the staging area. It allows you to see which changes have been staged, which haven't, and which files aren?t being tracked by Git. It does not show you any information about the committed project history. For this, you need to use the git log. It also lists the files that you've changed and those you still need to add or commit.

**Syntax**

$ git status

### **Git push Command**

It is used to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repo. It's the complement to git fetch, but whereas fetching imports commits to local branches on comparatively pushing exports commits to remote branches. Remote branches are configured by using the git remote command. Pushing is capable of overwriting changes, and caution should be taken when pushing.

Git push command can be used as follows.

**Git push origin master**

This command sends the changes made on the master branch, to your remote repository.

**Syntax**

$ git push [variable name] master

## Git Add All

$ git add -A

Or

$ git add .

## Add all New and Updated Files Only

1. $ git add --ignore-removal .

Add all Modified and Deleted Files

1. $ git add -u

Add Files by Wildcard

$ git add \*.java

## Git Undo Add

git reset <filename>

## The git commit command

1. $ git commit -a

git commit -m "Commit message."

git commit -amend

## Git Clone Command

git clone **<repository** URL**>**

git clone https://github.com/ImDwivedi1/Git-Example.git "new folder(2)"

## Git Clone Branch

1. $ git clone -b **<Branch** name**><Repository** URL**>**

 git clone -b master https://github.com/ImDwivedi1/Git-Example.git "new folder(2)"

# **Git Stash**

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don't want to make a commit of half-done work. Git stashing allows you to do so. The **git stash command** enables you to switch branches without committing the current branch.

* **Git stash**

**Git stash save**

**i**n Git, the changes can be stashed with a message. To stash a change with a message, run the below command:

**Syntax:**

$ git stash save "**<Stashing** Message**>**"

* **Git stash list**
* **Git stash apply**

In case of more than one stash, you can use "git stash apply" command followed by stash index id to apply the particular commit. It is used as:

**Syntax:**

$ git stash apply **<stash** id**>**

* **Git stash changes**

We can track the stashes and their changes. To see the changes in the file before stash and after stash operation, run the below command:

**Syntax:**

$ git stash show

We can exactly track what changes are made on the file. To display the changed content of the file, perform the below command:

**Syntax:**

$ git stash show -p

**Git stash pop**

The git stash pop command is quite similar to git stash apply. The main difference between both of these commands is stash pop command that deletes the stash from the stack after it is applied.

**Syntax:**

$ git stash pop

* **Git stash drop**

The **git stash drop** command is used to delete a stash from the queue.

$ git stash drop or git stash drop **<stash** id**>**

* **Git stash clear**

The **git stash clear** command allows deleting all the available stashes at once.

* **Git stash branch**

he git stash branch command allows the user to stash work on a separate branch to avoid conflicts.

 git stash branch **<Branch** Name**>**