## ****Git Tutorial – Operations & Commands****

Some of the basic operations in Git are:

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

Some advanced Git operations are:

1. Branching
2. Merging
3. Rebasing



**Installation of Git Client**

If you are using Debian base GNU/Linux distribution, then **apt-get** command will do the needful.

[ubuntu ~]$ sudo apt-get install git-core

[ubuntu ~]$ git --version

git version 1.8.1.2

And if you are using RPM based GNU/Linux distribution, then use **yum** command as given.

[CentOS ~]# yum -y install git-core

[CentOS ~]# git --version

git version 1.7.1

## Customize Git Environment

Git stores all global configurations in **.gitconfig** file, which is located in your home directory. To set these configuration values as global, add the **--global** option, and if you omit **--global** option, then your configurations are specific for the current Git repository.

Git stores these values in the **/etc/gitconfig** file, which contains the configuration for every user and repository on the system.

### **Setting username**

This information is used by Git for each commit.

[jerry@CentOS project]$ git config --global user.name "Jerry Mouse"

### **Setting email id**

This information is used by Git for each commit.

[jerry@CentOS project]$ git config --global user.email [jerry@abc.com](mailto:jerry@abc.com)

### **Avoid merge commits for pulling**

You pull the latest changes from a remote repository, and if these changes are divergent, then by default Git creates merge commits. We can avoid this via following settings.

jerry@CentOS project]$ git config --global branch.autosetuprebase always

### **Color highlighting**

The following commands enable color highlighting for Git in the console.

[jerry@CentOS project]$ git config --global color.ui true

[jerry@CentOS project]$ git config --global color.status auto

[jerry@CentOS project]$ git config --global color.branch auto

### **Listing Git settings**

To verify your Git settings of the local repository, use **git config –list** command as given below.

[jerry@CentOS ~]$ git config –list

## ****Initialize****

In order to do that, we use the command **git init.**

**git init** creates an empty Git repository or re-initializes an existing one.

Now that my repository is initialized, let me create some files in the directory/repository. For e.g. I have created two text files namely test1.txt and test2.txt.

**Git status**

The **git status**command lists all the modified files which are ready to be added to the local repository.

**Add**

This command updates the index using the current content found in the working tree and then prepares the content in the staging area for the next commit.

**git add <directory>**

or

**git add <file>**

Let us add the files using the command **git add -A**. This command will add all the files to the index which are in the directory but not updated in the index yet.

**Commit**

You can commit by using the command below:

**git commit**

Or you can use:

**git commit -m “<message>”**

## ****Pull****

The **git pull** command fetches changes from a remote repository to a local repository.

**git remote add origin <link of your central repository>**

**git pull origin master**

**Push**

This command transfers commits from your local repository to your remote repository. It is the opposite of pull operation.

**git push <remote>**

**Note** : This remote refers to the remote repository which had been set before using the pull command.

## ****Branching****

You can check what your current branch is by using the command:

**git branch**

To create a new branch we use the following command:

**git branch <branch-name>**

**git checkout <branch\_name>** and then

**git commit**

Now while we are in the branch, add and commit the text file test3.txt using the following commands:

**git add test3.txt**

**git commit -m”adding test3.txt”**

## ****Merging****

Merging is the way to combine the work of different branches together.

Now let us merge the two branches with the command below:

**git merge <branch\_name>**