**Contents:**

1. Installation guide
2. The Basics

**Installation Guide:**

Step 1: Go to <https://git-scm.com/> and click on download for windows.

Step2: Click on the installer link based on your system and download.

Step 3: Double click on the installer and accept the agreement. Click NEXT and go with the default values.

Step 4: Once installation is done, do minimum configuration:

In Git Bash:

* git config --global user.name “Huski-commando”
* git config --global user.email “gyan.mu@gmail.com”

Step 5: Download and install any Text Editor like ‘VS code, Notepad++ …’

Step6: Download the P4MERGE (<https://www.perforce.com/products/helix-core-apps/merge-diff-tool-p4merge>) tool for visual differences and click on Download. Download it according to your system configuration.

In Git Bash:

* git config --global diff.tool p4merge
* git config –global difftool.p4merge.path “c:/Program Files/Peforce/p4merge.exe”
* git config --global difftool.prompt false
* git config --global merge.tool p4merge
* git config –global mergetool.p4merge.path “c:/Program Files/Peforce/p4merge.exe”
* git config --global mergetool.prompt false

**The Basics:**

1. **Initialization**:

git init project\_name – To initialize a folder with git

1. **Git States:**
2. **Working Directory:** It holds all files and folders of working applications that may or may not be managed by git.
3. **Staging Area:** Stating area is used to prepare for the next commit. Files are moved from modifying directory to the git staging area and finally committed to the git repository.
4. **Repository (.**git folder): Files are placed in this folder after commit.
5. **First Commit:**

* Create a file with the name README.md and type some inputs.
* Run “**git status,**” command and git tells that it is an untracked file.
* Run the “**git add README.md**” command to add it to the staging area.
* Run “**git commit -m “First file in demo repo”** to commit a file from the staging area.

1. **Starting with Existing Project:**

* A project without git in it.
* To initialize your current project folder with git, use the command “**git init** .”
* Check with “**git status”**.

1. **Commits and Messages:**

* Create another file in the same directory.
* Now “**git add .**” to move all new files and modified files in the staging area at once.

1. **Commit Details with log and show:**

* Use the “**git log”** command to show log messages.
* Use the “**git show**”command to show last commits and shows containing last changes.
* To get out of the show command press “**q”.**

1. **Express Commits:**

* When a file is not added to the staging area for the first time and if you use the git status command to check the status, it shows **“Untracked file”.**
* Use the “**git ls-files**” command to see which files are being tracked by git.
* If you are updating committed files in git, you’ll have to add it again to the staging area and then commit it. Instead of doing it separately, you can do it with a single command. “**git commit -am “Updating README”**. Here ‘-a’ is for add to the staging area and ‘m’ is for commit.

1. **Backing out changes:**

* If you have updated your file and added it to the staging area as well but then you realize that you need to change it back as it was before adding it to the staging area, then use the command given below:

1. **Remove it from the staging area first**

**git restore --staged <file name>**

**b. Change it back using the command:**

**git checkout -- <file name>**

1. **Rename and Delete files using git**

* When files are in the git staging area then you can rename that file without unstaging it.

**git mv example.txt demo.txt**

* Then commit it.
* To remove a file use command:

**git rm demo.txt –** it will unstage the file them it needs to commit it again.

1. git status -A ls -al
2. git status -u
3. **Excluding unwanted files:**

* If you want git to ignore some files to staging you can put those files in .gitignore like a log file.

**Advanced: Beyond the basics**

1. **Comparing Differences:**

* Use “**git diff”** to compare files
* Use “**git difftool**” to compare in the P4Merge diff tool

1. **Branching and Merging Types:**

* **git checkout -b updates**
* It will create a branch with updates as name
* **git checkout master -** Switching back to the master branch
* **git merge updates(filename)-** To merge branch file with the master file
* Then **commit changes** to the master file.
* **git branch -d updates(filename) –** Once the branch file has been merged to the master file then delete the branch file.

1. **Conflict Resolution:**

* Create a branch “**very-bad**” update it and commit the changes.
* Now switch back to the “**master”** branch, modify the same file before merging both branches and commit the changes for the master branch.
* Let’s merge both branch “**very-bad”** and “**master”** branches. It’ll fail to merge both branches.
* So, open up git merge tool “**git mergetool”**, it’ll open the merge tool and let you choose the best option before merging it.
* Now, commit changes that you decide in the merge tool
* There will be another file with **.orig,** put this file in the .gitignore file and then add & commit the branch.

1. **Git Stash:**

* Let’s assume you have made some changes in a file but then you realized you need to work on something else, so you can save the changes at the last commit of the current branch by using the “**git stash”** command and you can check it by using “**git stash list”** command.
* Now work on your other project. Once finished getting back to your stashed project.
* **“git stash pop”** to remove stashing from your file and it’ll put your file back where you have edited.
* **Commit** the changes made by git stash pop and it’ll save what you have worked on.

1. **git reset <**3da6b20> and **git reflog**

**WELCOME TO GITHUB:**

1. If you have an account on GitHub, sign in but if you don’t have, create one account.
2. Create a repository with a demo name.
3. Link it with your local repository. Below is the command

**git remote add origin** [**https://github.com/gokugyan/demo.git**](https://github.com/gokugyan/demo.git) **-(directory location)**

1. git push -u origin master --tags
2. To push changes on GitHub account.

**SSH Authentication:**

**ssh -T git@github.com**

**GitHub Repository:**

**Fetch & Pull:** When you have updated a file on GitHub directly and committed it. Then you have updated a file in local and committed it again.

When you try to push the changes to GitHub, it will throw an error.

So, then first fetch it again from GitHub using **“git fetch”** and then pull it using **“git pull”**.

Push the changes to GitHub using **“git push”**.