**GIT**

**What is Git**

Simply git is a tool which is used in web development but not just a simple tool its one of the most popular git revision control and source code management system which keeps all the old and new versions of code and track who changed the code when changed the code and what part of code developer changed and also can see commit log to know why he changed and why he made that changes in code. Git was initially made by Linux Torvalds for Linux Kernel development.

**Why to Use Git**

There are many tools available in market right now like Git to revision control and SCM (source code management) but Git is popular because of following reasons:

* Git tracks state, history and integrate of the source tree
* Git keeps old versions for you if some developer did any mistake in code then you’ll always have previous version to fix it
* Multiple developers can work together, once they write code in their local machine and commit it then other developers can pull it easily.
* Large developer’s community and online websites to upload your source codes or get others source codes to make your work easier
* Lots of software available for both who comfortable with command line and for others GUI tools
* Easy and clear documentation to get started with
* Git will not use much bandwidth you don’t have to connect with your server always you just need to connect to push code when you are done

**GIT commands:**

1. **Git config**

The git config command is used to set git configuration values.

Example:

**git config –global user.name “rajendra”**

**git config –global user.email “nagabhoinarajendra@gmail.com”**

There are three configuration levels in git they are Global, local, system.

1. **Git init**

The git init command is used to create new git repository.

After executing the **“git init”** command it creates .git subdirectory in the current working directory which contains all the necessary git metadata for the new repository.

.git directory consists of following sub-directories

/HEAD

/branches

/config

/description

/hooks

/info

/objects

/refs

1. **Git add**

The git add command adds the changes in the current directory to the staging area.

**git add .**

**git add <file>**

The above command stage all changes in the specified file.

**git add <directory>**

The above command stages all the changes in the specified directory.

1. **Git commit**

Git commit takes the snapshot and saves it.

**git commit**

The above command simply commits with out any message.

**git commit –a**

Commit a snapshot of all changes in the working directory. This only includes modifications to tracked files (those that have been added with git add at some point in their history).

**Git commit –m “message”**

A shortcut command that immediately creates a commit with a passed commit message. By default, git commit will open up the locally configured text editor, and prompt for a commit message to be entered. Passing the -m option will forgo the text editor prompt in-favor of an inline message.

1. **Git status**

The git status command displays the state of the working directory and the stating area. It lets you see which changes that have been staged and which are not staged and files which are not being tracked by git.

1. **Git log**

The git log command displays the committed snapshots.

**Branching commands**

1. **git branch <name>**

Create a new branch and switch to it

1. **git checkout -b <branchname>**

Switch from one branch to another.

1. **git checkout <branchname>**

List all the branches in your repo, and also tell you what branch you're currently in.

1. **git branch**

Delete the feature branch.

1. **git branch -d <branchname>**

Push the branch to your remote repository, so others can use it.

1. **git push origin <branchname>**

Push all branches to your remote repository.

1. **git push --all origin**

Delete a branch on your remote repository.