GIT

* Introduction
* GIT work flow
* Nonlinear development
* Installation
* GIT commands
* GIT repositories
* GIT Checkout and Branching

Introduction:

GIT is an open source distributed version control system to handle

small to large projects with efficiency.

What is distributed version control system ?

It is a form of version control where the complete codebase including

its full history is mirrored on every developers computer.

What are benefits of version control systems ?

Helps a software team to manage changes to source code over time.

Keeps track of every modification to the code in special kind of

database.

Who invented GIT ?

Linus Torvalds.

What is the purpose of GIT ?

The purpose of GIT is to manage a project or set of files .

Why do we use GIT ?

GIT allows a team of people to work together, all using the same files. And it helps the team cope with the confusion that tends to happen when multiple people are editing the same files.

GIT workflow

* Clone the remote repository
* $ git checkout –b My Branch
* Work and commit some stuff
* GIT rebase master
* Work and commit some stuff
* GIT rebase master
* Finish the feature , commit
* GIT rebase master
* GIT checkout master
* GIT merge the My Branch

GIT for non linear development

GIT supports rapid branching and merging , and includes specific tools for visualizing and navigating a non-linear development history.

A **branch in GIT** is simply a lightweight movable pointer to one of these commits. The default **branch** name in **GIT** is master . As you start making commits, you're given a master **branch** that points to the last commit you made. Every time you commit, the **master branch** pointer moves forward automatically.

Strong support for non-linear development (thousands of parallel branches).

Installation:

Install GIT on **Windows**

1. Download the latest **GIT for Windows installer**.

2. When you've successfully started the installer, you should see the GIT **Setup wizard** screen. Follow the **Next** and Finish prompts to complete the installation. The **default** options are pretty sensible for most users.

3. Open **GIT Bash** .

4. Run the following commands to configure your **username**, **email** .

$ git config --global user.name "sumanth333"

$ git config --global user.email "polesumanth333@gmail.com"

Install GIT on **Linux**

1. Install GIT using **apt-get**

$ sudo apt-get update

$ sudo apt-get install git

2. Verify **installation**

$ git --version

git version 2.17.1

3. Configure your GIT **username** and **email**

$ git config --global user.name "sumanth333"

$ git config --global user.email "polesumanth333@gmail.com"

GIT Commands

$ git init

Initializes a new GIT repository. Until you run this command

your working directory considered as a regular folder.

$ git config

This is the most useful command when you are setting up GIT for the

first time.

$ git status

Check the status of your repository like changes need to be committed.

$ git add

It brings new files to GIT attention then you can directly commit

changes.

$ git commit -m "Message "

GIT's most important command . You should use this command to

save any sort of change and it takes a snapshot.

$ git branch

This command will let you build a new branch.

$ git merge

When you're done working on a branch, you can merge your changes

back to the master branch, which is visible to all collaborators.

$ git push

If you want your commits to be visible online on GitHub as well,

you can use this command.

$ git pull

If you want the updated version of the GIT repository to work with

you can pull the changes down from GitHub with this command.

$ git help

Brings up the details of most common GIT commands .

$ git log

Shows the listing of commits on a branch including the corresponding details.

$ git reset

Resets your index and working directory to the state of your last commit.

$ git remote

Shows all the remote versions of your repository.

$ git diff

Generates patch files or statistics of differences between paths or files in your GIT repository , or your index or your working directory.

$ git archive

Creates a tar or zip file including the contents of a single tree from your repository.

$ git prune

Removes objects that are no longer pointed to by any object in any reachable branch.

GIT Repositories

GIT stores the information in a data structure called a repository.

A **GIT** **repository** contains a set of commit objects.

**Clone the repository using command line:**

1. Navigate to your repository’s **Code** tab.

2. Click **Clone or download**.

3. Copy the **URL** provided.

4. Open your command line or Terminal application and enter the directory where you would like to copy the repository.

cd ~/

5. Clone the repository by replacing <URL> with clone URL you copied in the previous step. The repository will be cloned into a new directory in this location.

git clone <URL>

6. Navigate into the directory of the repository you just created. Replace with your own repository's name.

cd <REPOSITORY-NAME>

7. Type:

$ git status

GIT Checkout and Branching

Branching:

It is creating copies of object under revision control so that the

modifications can happen parallel.

Creating a branch:

$ git checkout -b Branch Name

Deleting a branch:

$ git branch -D Branch Name

Why Branches in GIT ?

This enables you with unlimited amount of different contexts.

You will have at least one context for your main production

state and another contexts for different features like

bug fix, experiment .

GIT Checkout:

It is to checkout your desired status of your repository.

checkout is switching between branches you already have.

How to check out a remote branch in GIT ?

$ git fetch

$ git checkout test