IMPORTANCE AND BASICS OF GIT

K VIJETH

Co-author SUMA S

MASTER OF COMPUTER APPLICATION

DAYANANDA SAGAR COLLEGE OF ENGINEERING

BENGALURU

Abstract: This paper describes the importance of Git in the IT fields and how Git used in the IT industries. According to a recent Global Developer Survey reveals the need for a more collaborative workflow. About 92% of the developers say distributed version control systems (Git repositories) are essential for their everyday work. This shows that Git is important and college students should deploy their project work on Git so that lectures can keep track of students' work, and they can add Git experience in their resume.

Index term: Git, GitHub, GitLab, Bitbucket, Distributed workflow.

1. Introduction

Git is the distributed version control system used in almost every software company around the world to keep track of changes in the source code during software development.

It is designed for coordinate workflow among developers and alose can be used to keep track of changes in any set of files.

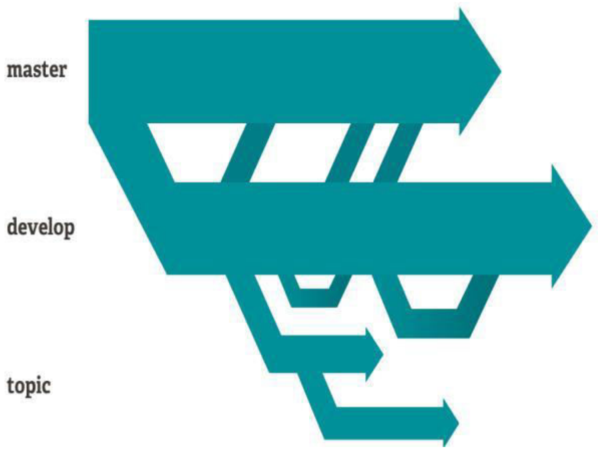
The main intention of the Git is to provide data integrity and support for distributed, speed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, unlike most client-server systems, every Git directory on every computer is a full-fledged repository with a complete history and full version-tracking abilities, independent of network access or a central server

2. Branching and Merging

Git stands out from every other SCM (Supply chain management) by its feature branching model

Git allows developers to create branches and also encourages to have local multiple branches and every branch independent of each other. Every operation like creating, merging and deleting of software development takes seconds.



[fig 1 Repository with multiple branches] [source: git-scm.com/about]

Having multiple branches can be useful like:

Frictionless Context Switching: Users can create experimental branches try new things and .commite few times and switch back to work branch and merge the code tried in the experimental branch

Role-based Codelines: Developers can have one main branch for the production line and another for the testing and other small branches for the day to day work

Feature-based workflow: Developer can create a new feature and commit in different branches and the developer can use those branches whenever required and he can delete branch after the feature is merged to the main code line.

Disposable experimentation: Create a new branch to experiment with, if it doesn't work just delete the branch and nobody else will ever see it.

3. Distributed

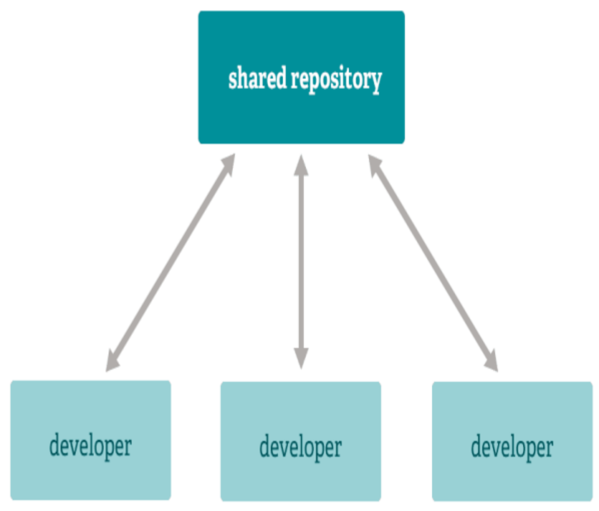
Git is a distributed SCM. This means instead of doing a "checkout" of the current tip of the source code, and you can "clone" the entire repository.

Multiple Backups: If a company using a centralized workflow, then every user will have the bits of backup of the central server. So in the event of the crash or corruption, then each of these users' copies could be pushed up to replace the central server.

Workflows: Because of Git's distributed nature and superb branching, an almost endless number of the workflow can be implemented with ease.

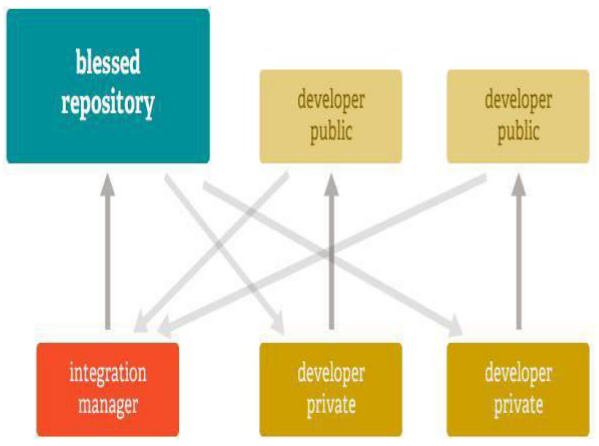
Some of the workflows are:

Subversion-style workflow: A centralized workflow is very common in most of the companies, especially from people transitioning from a centralized system. Git won't permit you to push if there is any conflict with the file you push or if someone has pushed since the last time you fetched. In a centralized system, every developer will push to the same server, and it will work just fine.



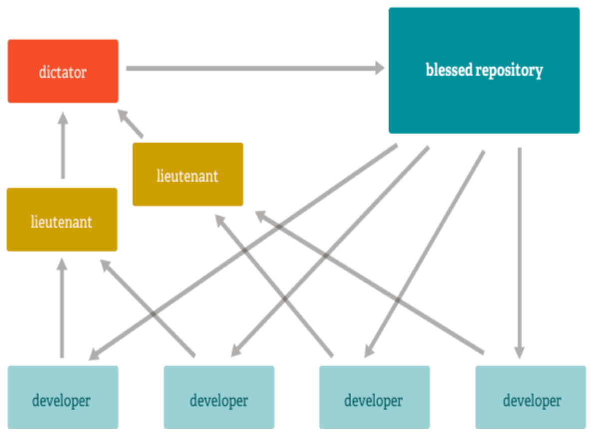
[fig 1 Subversion-style workflow]
[source: git-scm.com/about]

Integration manager workflow: Integration manager workflow is another typical workflow of the Git workflow. A developer who commits to the "blessed" repository and many developers can clone that repository and push their independent repository, and ask the integrator to pull in their changes. This type of development model seen mostly in open source or GitHub repository.



[fig 1 Integration manager workflow]
[source: git-scm.com/about]

Director and lieutenant workflow: Director and lieutenant workflow are mainly used in large projects like development workflow Linux kernel. In this model, lieutenants are in charge of a specific subsystem of the project and the check code and perform merge operation. A director can only pull changes from his/her lieutenant and then push to a repository that everyone else can clone that again.



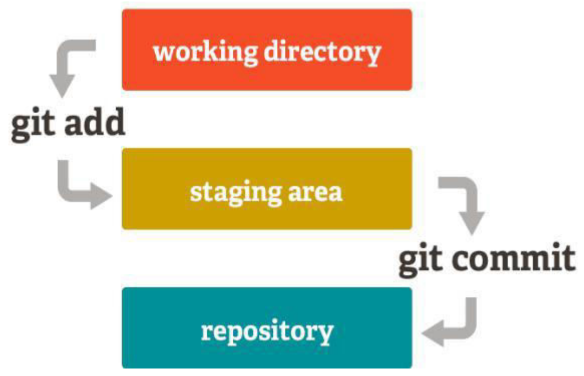
[fig 1 Director and lieutenant workflow] [source: git-scm.com/about]

4. Staging area

The staging area, or also called an index area, is an intermediate area between the working directory and the repository where a commit can be formatted and reviewed before completing the commit.

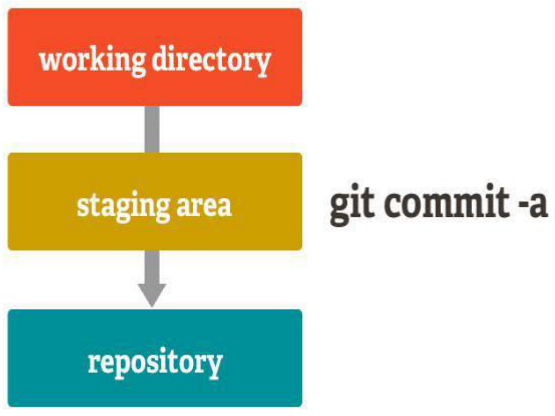
One thing unique about the Git while compared to the other tools is its ability to quickly stage some of the files and commit them without committing all other modified files in the working directory having to list them during the commit.

The staging area allows you to stage only portions of a modified file. Gone are the days of making two logically unrelated modifications to a file before you realized that you forgot to commit one of them. Now you can stage the change you need for the current commit and stage the other change for the next commit. This feature scales up to as many different variations to your file as needed.



[fig 1 Two-step commiting]
[source: git-scm.com/about]

Git also provides another easy way to ignore this feature if don’t want this kind of controls. In order to do so, you can just add “-a” to the commit command to allow all the changes to all files to the staging area.



[fig 1 Single step commiting]
[source: git-scm.com/about]

5. Commands used in git.

git config: Set user name and email in the configuration file.

git init: Initialize the new or existing projects git repository.

git clone: It is used to get an existing repository and pull that repository again and again.

git status: Show all the modified files since the last commit.

git add: Add changes to the staging area of the working directory.

git commit: Commits the changes.

git push/pull: User can push their changes to the repository or the user can pull from the repository.

git branch: Display all the branches.

git checkout: Change current branch or create a new branch using “-b”.

git stash: Save changes and not commit immediately.

git merge: Used to merge two branches.

git reset: Used to abort the latest commit so that the developer can further work on it.

git remote: Check present remote/source or add a new remote.

6. Conclusion.

In conclusion git is the most used software in the IT industry and git is not taught in colleges not even as an extra curricular subject. My approach is colleges must teach git at least two or three classes. And students must upload their projects to the github so that they can show their project companies or interviewers. And it adds weight to the resume.

7. Reference

1. git-scm.com

2. wikipedia.org

3. Gitlab.com

4. www.cs.toronto.edu

5.dev.to/dhruv/essential-git-commands-every-developer-should-know-2fl