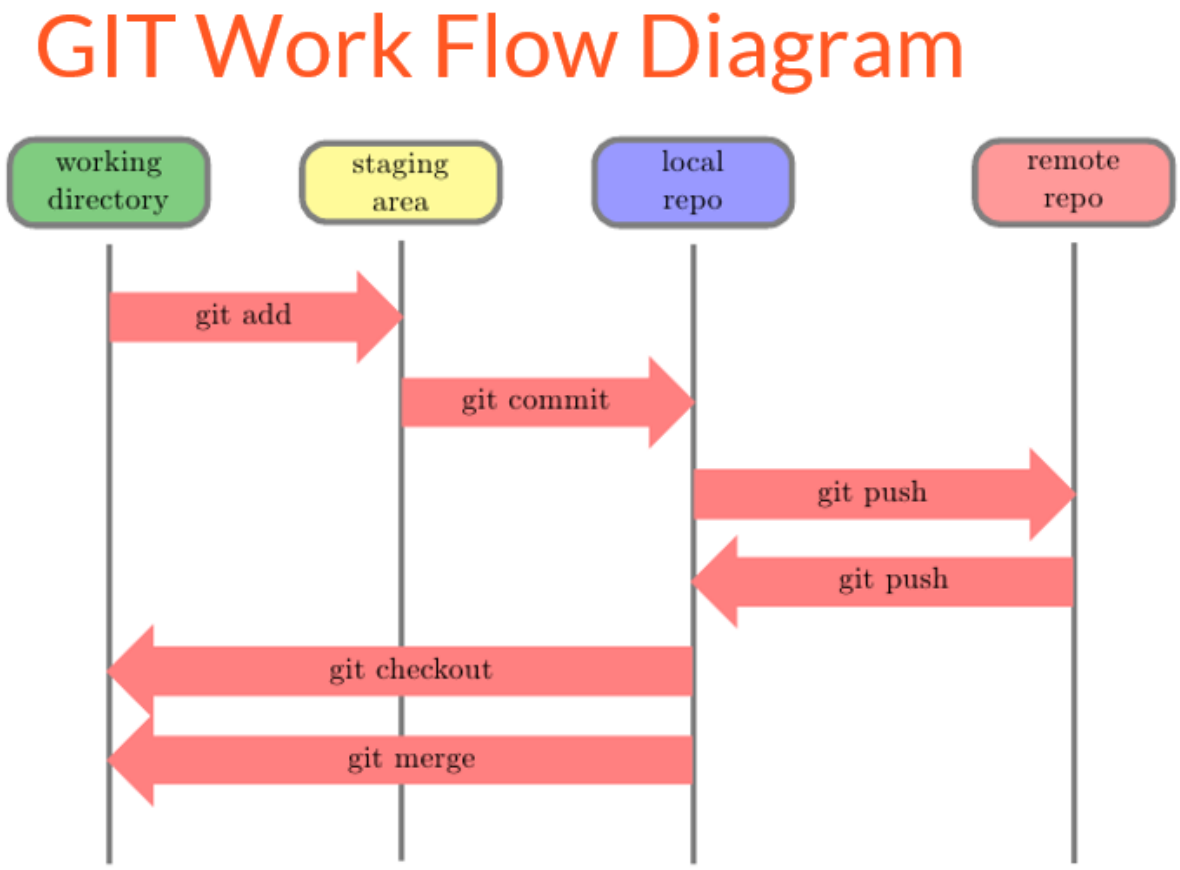
Viranchi Badheka

15IT111

Version Control Tools

1. Git

- Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.
- It is primarily used for source code management in software development,[8] but it can be used to keep track of changes in any set of files.
- Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history.
- In Git, a core assumption is that a change will be merged more often than it is written, as it is passed around to various reviewers.
- In Git, branches are very lightweight: a branch is only a reference to one commit. With its parental commits, the full branch structure can be constructed.



2. Concurrent Versions System (CVS)

- CVS has been around since the 80s, and has been very popular with both commercial and open source developers.

- It is released under the GNU license, and uses a system to let users "check out" the code they are going to work on and "check in" their changes

- Originally, CVS handled conflicts between two programmers by only allowing for the latest version of the code to be worked on and updated.

- Now, CVS can handle branching projects so the developed software can diverge into different products with unique features and will be reconciled at a later time.

- The CVS server runs on Unix-like systems with client software that runs on multiple operating systems.

3.Mercurial

- Mercurial is a distributed revision-control tool for software developers.

- Mercurial's major design goals include high performance and scalability, decentralized, fully distributed collaborative development, robust handling of both plain text and binary files, and advanced branching and merging capabilities, while remaining conceptually simple.

- Mercurial uses SHA-1 hashes to identify revisions. For repository access via a network, Mercurial uses an HTTP-based protocol that seeks to reduce round-trip requests, new connections and data transferred.

- Although Mercurial was not selected to manage the Linux kernel sources, it has been adopted by several organizations, including Facebook,[10] the W3C, and Mozilla.

Comparison between Git and CVS

- **Setting up repository**. Git stores repository in .git directory in top directory of your project; CVS require setting up CVSROOT, a central place for storing version control info for different projects (modules).

- **Changesets**. Changes in CVS are per file, while changes (commits) in Git they always refer to the whole project.

- **Naming revisions / version numbers**. There is another issue connected with the fact that in CVS changes are per files: version numbers like 1.4 reflects how many time given file has been changed. In Git each version of a project as a whole (each commit) has its unique name given by SHA-1 id; usually first 7-8 characters are enough to identify a commit.

- **Rename (and copy) tracking**. File renames are not supported in CVS, and manual renaming might break history in two, or lead to invalid history where you cannot correctly recover the state of a project before rename. Git uses heuristic rename detection, based on similarity of contents and filename.

- **Branching Operation.** In CVS, branch operations are expensive as it is not designed for long-term branching while in Git branch operations are cheap.

- **Keyword expansion**. Git offers a very, very limited set of keywords as compared to CVS. This is because of two facts: changes in Git are per repository and not per file, and Git avoids modifying files that did not change when switching to other branch or rewinding to other point in history

- **More tools**. Git offers much more tools than CVS. One of more important is "git bisect" that can be used to find a commit (revision) that introduced a bug.

Comparison between Git and Mercurial

- Git is a platform, Mercurial is “just” an application. Git is a versioned filesystem platform that happens to ship with a DVCS app in the box, but as normal for platform apps, it is more complex and has rougher edges than focused apps do

- Both systems are very similar. Additionally, many of the features Git provides out-of-the-box are brought into Mercurial as plugins.

- The feature branch workflow can be implemented easily in both Git and Mercurial.

- Mercurial comes as a monolith app, whereas Git, following the linux ideology, comes as a number of small binaries that often remain hidden to the developer. Having the fine tools to be able to tweak the system means Git is extremely flexible.

- Git has a strong focus on mutating this history graph whereas Mercurial does not encourage history rewriting.