Source Code Management with GIT

By Satya

What is Source Code Management

- Source code management (SCM) is used to track modification of Projects.

Some features of SCM as follows.

- Track file versions.
- Document History.
- Manage Branches.
- Provide merging solutions.
- Identify conflicts when merging and help resolve them.

What is Version Control System

- A **version control system** or **VCS**, also know as revision control or source control system, is a software utility that tracks and manages changes to a filesystem.

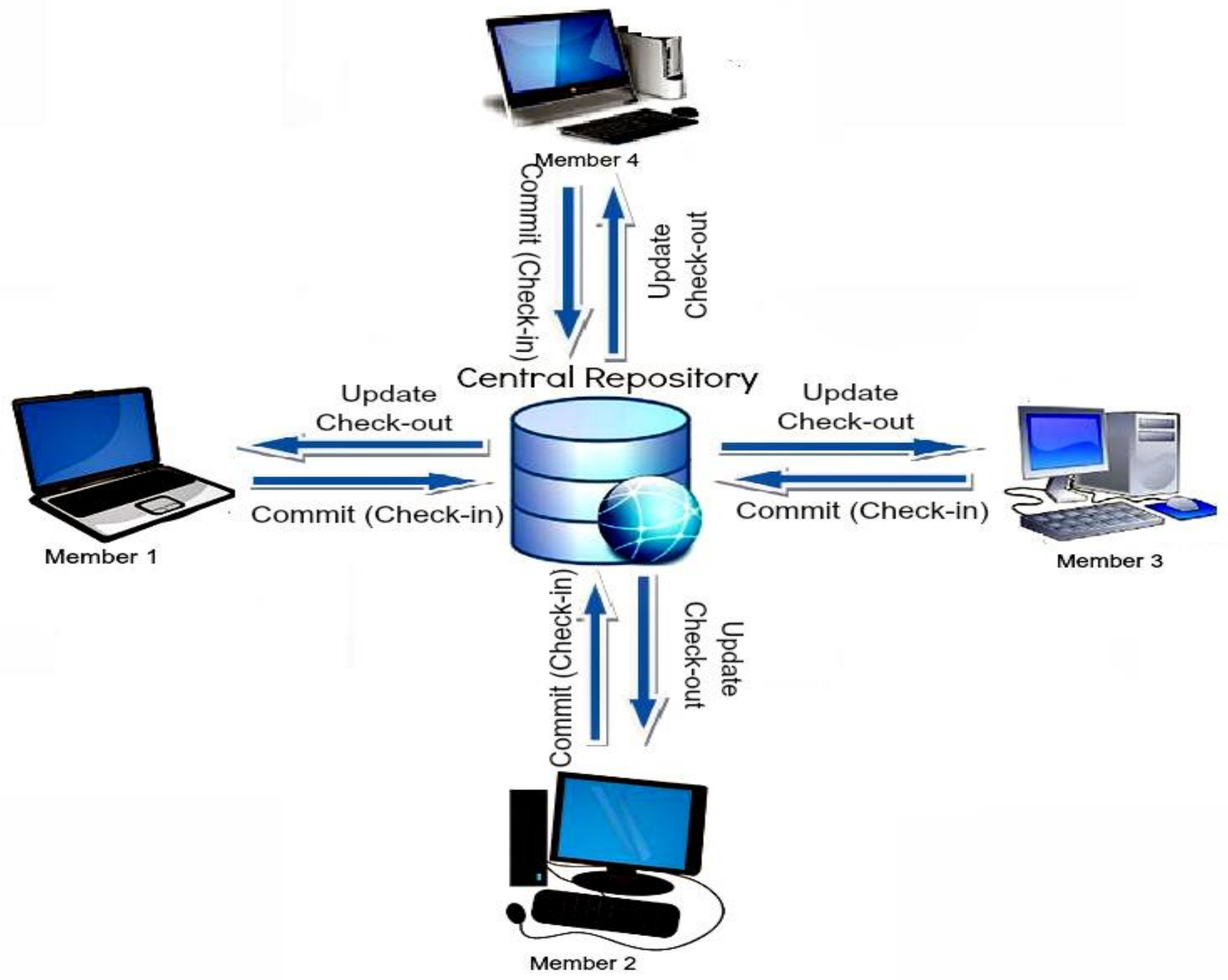
  Types of Version Control Systems

  Centralized and Distributed.

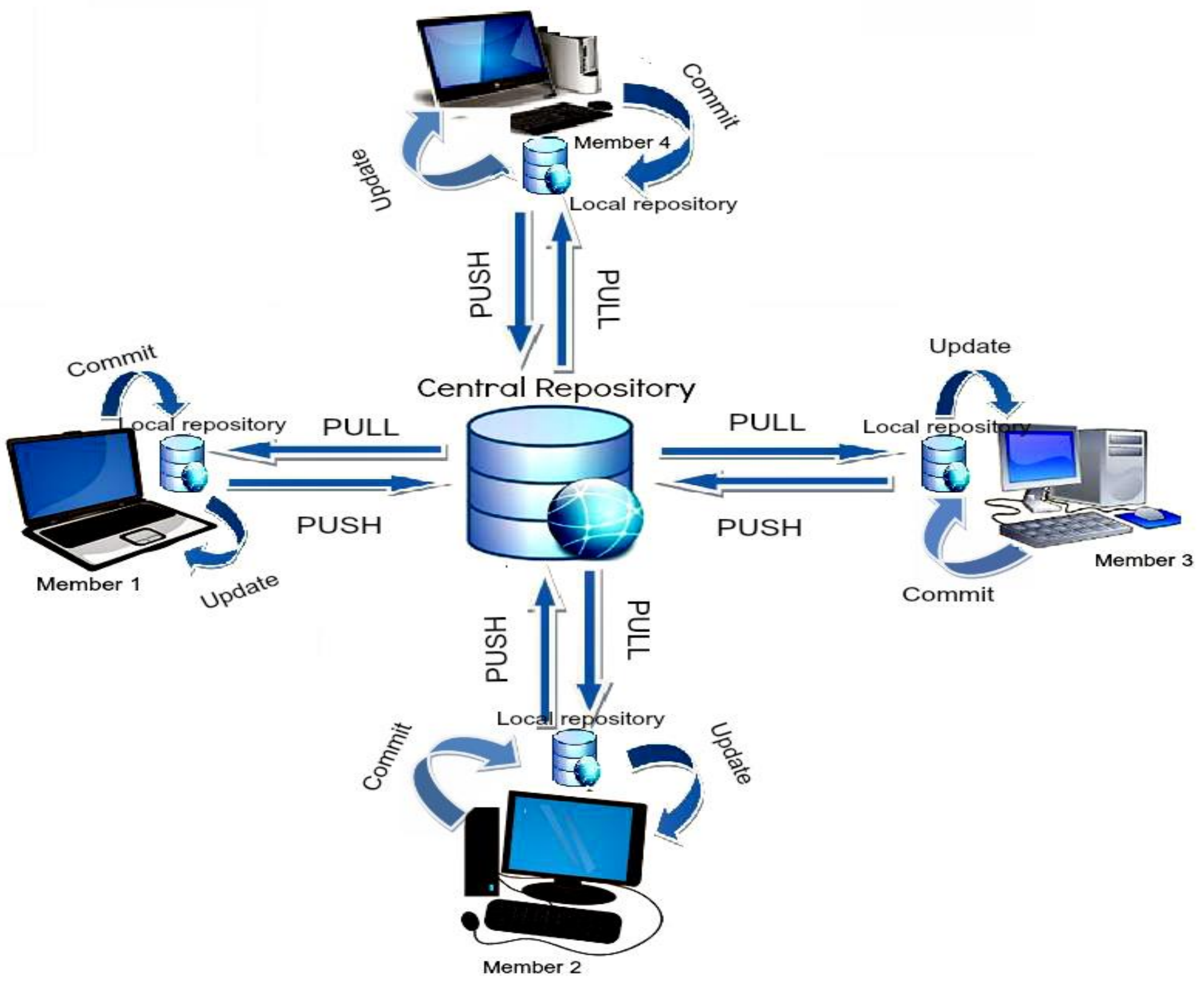
- The concept of a centralized system is that it works on a Client-Server relationship. The repository is located at one place and provides access to many clients.

- Whereas, in a **Distributed System**, every user has a local copy of the repository in addition to the central repo on the server side.

Centralized Version Control System



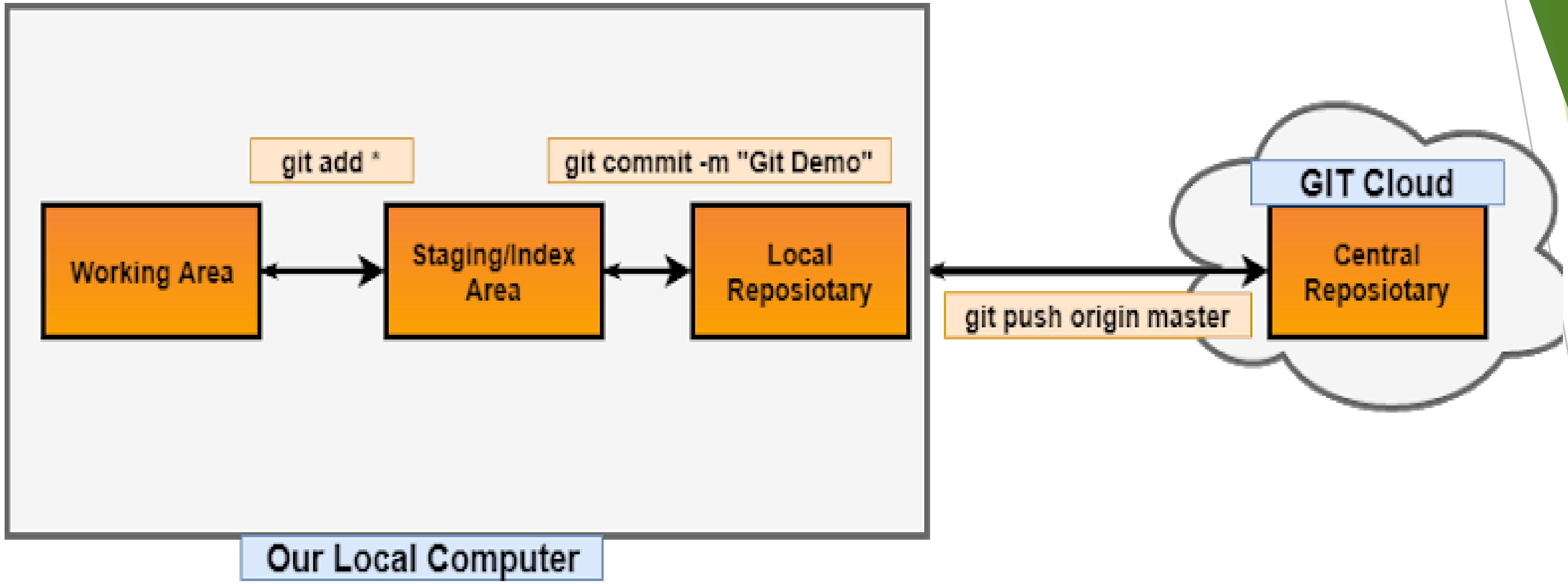
Distributed Version Control System



What is Git

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- GIT is called SCM (Source Code Management) tool, that means where we maintain all our project source code.
- GIT is a distributed version controlling system , that means GIT maintains all modifications(or versions) happening to a specific file made by different developers.
- GIT also records who modified when modified and why modified, because of those features we can easily track the project source code.
- Some other SCM tools:

  SVN(SubVersion)/Clear Case/TFS/Perforce/CVS



Git Architecture

Creating GitHub account:

From https://github.com/ → Sign Up → Choose Unique username → Enter your Email → Choose alphanumeric password → Sign Up for GitHub.

Once you sign up git will send a confirmation email to you, once you confirm it, git creates account for you.

Installing GIT client: To interact or communicate to central git server from our laptop we want mediator, for that so we need git client as mediator. Availabne GIT clients in market are

GIT Bash

Tortoise GIT.

ATOM

Creating GIT project: There are two ways to create your project.

Create locally and initialize it, and upload to Central.

Create in central and Clone it.

Installing Git

- On different OS we have different methods for installing Git.

- Refer the following document on how to install git on various OS.
  - https://www.atlassian.com/git/tutorials/install-git#linux
- For this training we will use yum to install git.
  - sudo su - # Switch to root user.
  - yum install git –y # command to install Git

GIT Cheat Sheet: https://education.github.com/git-cheat-sheet-education.pdf

- **Configuring GIT Client with user details:**

  git config --global user.name "john"

  git config --global user.email "john123@gmail.com"

  git config -l → To list the author username and email

This information is used by git to record our commits.

Note: Without configuring the user details we can't commit the files from staging area to local repository.

- **Commands to add the files to the staging or index area:**

git add one.txt

*git add ** → To add all files at a time

*git add *.py* → To add all files with extension .py

git add 1.txt 2.txt 3.txt → To add more file by using space separated.

*git add p** → to add all files with file name starts with "p"

- **Command to Commit the files to the local repository:**

git commit -m "Commit message"

Where m is "message"

- **Command to check the status of the git repository:**

git status

This command will give the present status of the git repository.

- **Command to push the committed files to the central repository:**

git push origin master

Where origin → alias name for remote repository url Master → Default main branch

- **Command to check commit history of this branch:**

git log → To check Commit history of all files

git log 1.txt → To check commit history of particular file

- **GIT Branching:**

  -Branch is used to work on a specific task.

  -Branch provides isolation, that means separation of work.

Master branch :

Every git repository comes with a default branch which is called as master branch. Master must contain only well tested code and no one should be directly work on master. If any work assigned to you, you can create a new branch and work on the branch and finally merge that branch to master branch.

Command to create a new branch:

  git branch branch-name

Command to Switching a new branch:

  git checkout branch-name

  git checkout -b development → To create a new branch and switch to new branch

- **Merging changes in new branch to main branch:**

We can do this in two ways

-By using merge command

-By creating a pull request

git branch <u>development</u> → To Create new branch

git checkout <u>development</u> → To switch “development” branch

-----DO ALL CHANGES -----

git checkout master → To switch “Master” branch

git merge development → Finally to Merge

Note: In real world all people will not have permissions to merge directly to master, so this approach is not good, Instead of we will create a ”pull request”

Q&A