**Version Control Tools:**Svn,Git,CVS are the popular VCS tools which provides a way to maintain a history of changes during your software development process. It allows you to go back in the history and see the whole changes in the codebase.

**GitHub** **:**It is a website for online hosting your git repository.The main idea behind hosting is collaboration between various communities of developer.

**Build Tools**: Maven,Gradle,Ant,SBT etc.. are the called build tools which help you to manage all your dependencies and simplify your building process which involves compile, executing,deploying etc.. Jenkins is an open source automation tool which lets you simplify your build process by automating it.

**A continuous integration (CI)** system helps you build, or test or both, the source code of your project to see if it works without any errors, every time you make a meaningful set of changes. The CI system reports whether a build was successful or not on its own, but can also report back to whoever triggered the building process.

s**Git is a revision control system**, a tool to manage your source code history.

**GitHub is a hosting service for Git repositories**.

**Git the tool, GitHub the service for projects that use Git.**

Github allows you to:

* Share your repositories with others.
* Access other user's repositories.
* Store remote copies of your repositories (github servers) as backup of your local copies.
* Commit - committing is the process which records changes in the repository. Think of it as a snapshot of the current status of the project. Commits are done locally.
* Push - pushing sends the recent commit history from your local repository up to GitHub. If you’re the only one working on a repository, pushing is fairly simple. If there are others accessing the repository, you may need to pull before you can push.
* Pull - a pull grabs any changes from the GitHub repository and merges them into your local repository.
* For each git directory/repository the setting will remain unique. Even on same system

<https://learn.sparkfun.com/tutorials/using-github-to-share-with-sparkfun/committing-pushing-and-pulling>

## Fetching a remote

git clone grab a complete copy of another user's repository, if going to work on already existing repository then first clone it.

If we are cloning any repo first from gitHub-

1. Need not to do git init, it is auto created
2. need not to set **remote origin** , it is auto set
3. No need to mention parameters in pull and push in cloned repo. But can mention if need specific update.

git clone <URL> <RepoGItName>

e.g. git clone <https://github.com/JagseerSinghNadwal/Selenium.git> InJags

Cloning into 'Injags'..

pull is a combination of fetch and merge.

Pulling is the opposite of pushing–it retrieves changes from the remote location and applies them to your local repository. You probably won’t do it that much; it’s more useful in a group environment where more than one person is submitting changes to a single repository.

git pull remotenameGitHub branchNameToBeMergeWith

* git pull https://github.com/JagseerSinghNadwal/Selenium.git master
* git pull origin next (Merge into the current branch the remote branch next)
* git pull (Update the remote-tracking branches for the repository you cloned from)

Note: Because pull performs a merge on the retrieved changes, you should ensure that your local work is committed before running the pull command. If you run into [a merge conflict](https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line) you cannot resolve, or if you decide to quit the merge, you can use git merge --abort to take the branch back to where it was in before you pulled.

fetch to retrieve new work done by other people. Fetching from a repository grabs all the new remote-tracking branches and tags without merging those changes into your own branches

git fetch remotename

merge is used to merge different people's work together with yours, and

git merge remotename/branchname

## Creates a new local repository with the specified name -

git init //.git will be created in same direc

git init jags //A git folder with name jags will be created and inside that .git be there

## Lists all new or modified files to be commited:

git status

## Lists version history for the current branch -

git log

## git add:

git add .

## git commit:

git commit –m “my first commit”

## Adding a remote's URL

git remote add origin <REMOTE\_URL>// Sets the new remote

git remote –v // Verifies the new remote URL ---it will show all remote URLs

git remote add jagseRepo https://github.com/JagseerSinghNadwal/Selenium.git

//origin can be any user defined name.

## Changing a remote's URL

git remote -v

git remote set-url origin <https://github.com/USERNAME/REPOSITORY.git>

1. The next time you git fetch, git pull, or git push to the remote repository, you'll be asked for your GitHub username and password
2. We can give either HTTPS or SSH url to remote github repository. Same way we can switch URL from HTTPS to SSH or vice versa.

## Adding an existing project to GitHub using the command line

## Push: Pushing sends your changes up to GitHub, so they can be shared with the rest of the world. It also serves as a hedge against data loss.

git push origin <your-branch > // e.g. Master

//will ask for usename and password

### [**git reflog**](https://git-scm.com/docs/git-reflog)

display any change which updated the HEAD and checking out the desired reflog entry will set the HEAD back to this commit

### **git reset**

git reset 0d1d7fc32

**git branch - list all branches**

**git branch jags- create branch jags**

**git checkout jags- switched to jags branch**

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

**git config –global user.email** [**jags.nas@gma.com**](mailto:jags.nas@gma.com)

### **Push variations:**

git push

Works like git push <remote>, where <remote> is the current branch’s remote

git push origin master

Find a ref that matches master in the source repository (most likely, it would find refs/heads/master), and update the same ref (e.g. refs/heads/master) in origin repository with it. If master did not exist remotely, it would be created.

git push origin HEAD

A handy way to push the current branch to the same name on the remote.

**git push origin HEAD: <any name e.g. master >**

Push the current branch to the remote ref matching master in the origin repository.

git push origin :experimental

Find a ref that matches experimental in the origin repository (e.g. refs/heads/experimental), and delete it.

Jagseer@Jagseer-PC MINGW64 ~/Downloads

$ git status

fatal: Not a git repository (or any of the parent directories): .git

Jagseer@Jagseer-PC MINGW64 ~/Downloads

$ git init

Initialized empty Git repository in C:/Users/Jagseer/Downloads/.git/

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git status

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

Comm.docx

desktop.ini

jags/

nothing added to commit but untracked files present (use "git add" to track)

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git log

fatal: your current branch 'master' does not have any commits yet

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git add .

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: Comm.docx

new file: desktop.ini

new file: jags

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git commit -m "first commit"

[master (root-commit) a6f70f5] first commit

3 files changed, 1 insertion(+)

create mode 100644 Comm.docx

create mode 100644 desktop.ini

create mode 160000 jags

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git status

On branch master

nothing to commit, working tree clean

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git log

commit a6f70f55ddda80485d6e255a32568519f379ae52 (HEAD -> master)

Author: jags <jagseer.nadwal@gmail.com>

Date: Sun Aug 6 13:44:23 2017 +0530

first commit

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git remote -v

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git push

fatal: No configured push destination.

Either specify the URL from the command-line or configure a remote repository using

git remote add <name> <url>

and then push using the remote name

git push <name>

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git remote add jagseRepo https://github.com/JagseerSinghNadwal/Selenium.git

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git remote -v

jagseRepo https://github.com/JagseerSinghNadwal/Selenium.git (fetch)

jagseRepo https://github.com/JagseerSinghNadwal/Selenium.git (push)

Jagseer@Jagseer-PC MINGW64 ~/Downloads (master)

$ git push jagseRepo master

Username for 'https://github.com': jagseer.nadwal@gmail.com

To https://github.com/JagseerSinghNadwal/Selenium.git

! [rejected] master -> master (fetch first)

error: failed to push some refs to 'https://github.com/JagseerSinghNadwal/Selenium.git'

hint: Updates were rejected because the remote contains work that you do

hint: not have locally. This is usually caused by another repository pushing

hint: to the same ref. You may want to first integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Jagseer@Jagseer-PC MINGW64 /c/New folder/jags (master)

$ git pull

From https://github.com/JagseerSinghNadwal/Selenium

\* [new branch] temp -> origin/temp

Already up-to-date.

Jagseer@Jagseer-PC MINGW64 /c/New folder/jags (master)

$ git push origin temp

error: src refspec temp does not match any.

error: failed to push some refs to 'https://github.com/JagseerSinghNadwal/Selenium.git'

Jagseer@Jagseer-PC MINGW64 /c/New folder/jags (master)

$ **git push origin HEAD:temp**

Username for 'https://github.com': jagseer.nadwal@gmail.com

Everything up-to-date

Jagseer@Jagseer-PC MINGW64 /c/New folder/jags (master)

$ git pull

Already up-to-date.