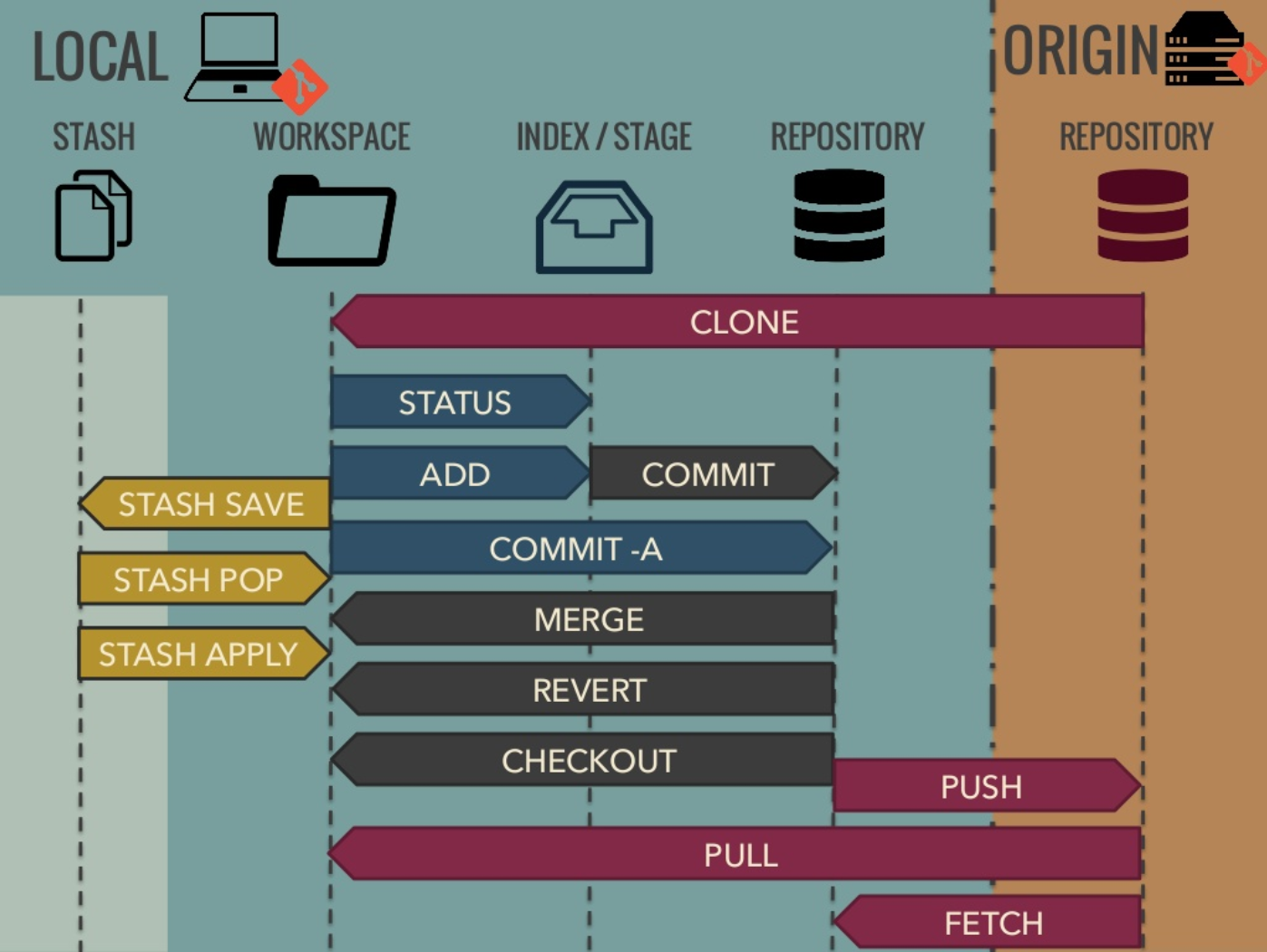
GIT

* Introduction

For everyone who is used to SVN, GIT might be a bit confusing and complex at first, however once you understand and get used to it, then it’s the simplest content manager.



From right to left:

- **Repository** the local copy of the whole version history

- **Index** (or Staging Area) is the place where changes are collected for a commit ... this allows to only checkin parts of the local changes

- **Workspace** which is just the editable files in the normal filesystem (like Working copy in SVN)

- **Stash** is an extra stack for storing backups of changes ( this allows to remove changes from Workspace without loosing them)  
... there are only a few cases where you have to use the Stash, e.g. when Pulling changes that conflict with your workspace

* GIT vs Subversion(e.g SVN)

The key difference is that it is decentralized. Imagine you are a developer and you develop on your laptop and you want to have source control so that you can go back few hours.

With Subversion, you have a Problem: The SVN Repository may be in a location you can't reach (in your company, and you don't have internet at the moment), you cannot commit. If you want to make a copy of your code, you have to literally copy/paste it.

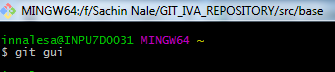
With Git, you do not have this problem. Your local copy is a repository, and you can commit to it and get all benefits of source control. When you regain connectivity to the main repository, you can commit against it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** |  | **Git** | **SVN** |
| Commit Files (A) | Add, commit and push 113 modified files (2164+, 2259-) | 0.64 | 2.60 | 4x |
| Commit Images (B) | Add, commit and push 1000 1k images | 1.53 | 24.70 | 16x |
| Diff Current | Diff 187 changed files (1664+, 4859-) against last commit | 0.25 | 1.09 | 4x |
| Diff Recent | Diff against 4 commits back (269 changed/3609+,6898-) | 0.25 | 3.99 | 16x |
| Diff Tags | Diff two tags against each other (v1.9.1.0/v1.9.3.0 ) | 1.17 | 83.57 | 71x |
| Log (50) | Log of the last 50 commits (19k of output) | 0.01 | 0.38 | 31x |
| Log (All) | Log of all commits (26,056 commits - 9.4M of output) | 0.52 | 169.20 | 325x |
| Log (File) | Log of the history of a single file (array.c - 483 revs) | 0.60 | 82.84 | 138x |
| Update | Pull of Commit A scenario (113 files changed, 2164+, 2259-) | 0.90 | 2.82 | 3x |
| Blame | Line annotation of a single file (array.c) | 1.91 | 3.04 | 1x |

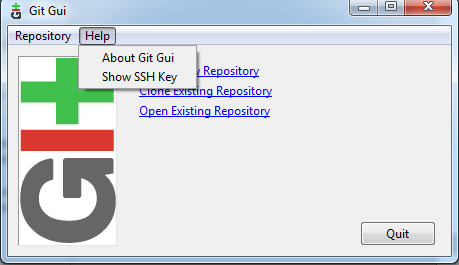
* GIT Setup
  1. Git Client : as a starting point it’s probably best to install the official windows client( git bash ) <https://git-scm.com/download/win> , there are other open source Gui also available
  2. Merge tool: Merge tool is mainly used for merging the code and its important stuff when it comes to working in a team and single set of files. It seems that git for windows doesn’t contain the merge tool, so you can install it separately One option is to install "Meld": <http://meldmerge.org/> , there are many other merge tool available. You need to do one more step to make it working.

**git config --global merge.tool meld  
git config --global mergetool.meld.path "C:\Program Files (x86)\Meld\Meld.exe"**

* 1. Setup SSH-Key: The access handling is controlled via SSH-Keys, that's why you have to setup a key in your client, and the way to generate and setup in git client is “git gui”



After this below dialog will open and “Show SSH key” allows to generate key.



* GIT Operations:
  + Clone : “git clone repository.git destFolderName”
  + Pull : “git pull master”
  + Commit: Single file-> “git commit filename”, All changes -> “git commit -a”
  + Push: “git push origin branch-name”
  + Checkout: “git checkout branch-name”
  + Branch: “git branch branchName” git checkout –b branchname
  + Stash: “git stash”
  + Status: “git status”
  + Diff : “git diff master”
  + Fetch : git fetch
  + Merge: git merge
  + Add:
* GIT tricks and tips:
  + Rename Branch :-
    - Step 1:
      * if you are on the branch you want to change
        + “git branch –m new-name”
      * if you are on different branch
        + “git branch –m oldname newname”
    - Step 2:
      * Delete the old-name remote branch and push the new-name local branch.
      * “git push origin :oldname newname”
    - Step 3:
      * Reset the upstream branch for the new-name local branch.
      * “git push origin -u newname”