Git:

For version control system.

Git was created in 2005. Distributed version control., open source and free software. Compatible with Unix-like systems and also windows.

Distributed version control:

Maintain own repositories instead of working from a central repository. Track changes, not versions(different from CVS).

Imagine changes to document as sets A,B,C,D,E,F.

Repo: A,B,C,D,E,F

Repo: A,B,C,D

Repo: A,B,C,E

Not right or wrong.

No central server:

Faster;

No network access;

No single failure point;

All repos are considered equally.

Who use Git?

Track edits;

Share changes;

Not afraid of command-line tools;

Not useful for tracking non-text files.

Download Address:

<http://git-scm.com/downloads>

Command:

Which git: Find address

Git version: git –version

Configuration:

System Level: /etc/gitconfig

User Level: ~/.gitconfig (~:Home directory)

Project Level: my\_project/.git/config

Command:

Git config --system

Git config –global

E.G.

git config --global user.name "alexliu0809@hotmail.com"

git config --global user.name "AlexLiu"

git config –-global core.editor “notpad.exe” //set text editor

git config -–global color.ui true

Git auto-completion(not for windows):

Download from GitHub:

Cd ~

Curl –OL <https://github.com/git/git/raw/master/contrib/completion/git-completion.bash>

Rename file:

Mv ~/git-completion.bash ~/.git-completion.bash

Edit

~/.bash\_profile or ~/.bashrc

Git initializing a repository.

Cd ~

Cd ./MyProject/

Git init

Ls –la //check if repository created successfully.

First Commit:

//make changes

//add

//commit

//make changes

Git add . //add everything

Git commit –m “initial commit”

Writing commit messages.

Provide enough information for updates.

“Add:xxx”

“Change:xxx”

See commit Messages:

Git log

Git log –grep=”Init” //message have “init”

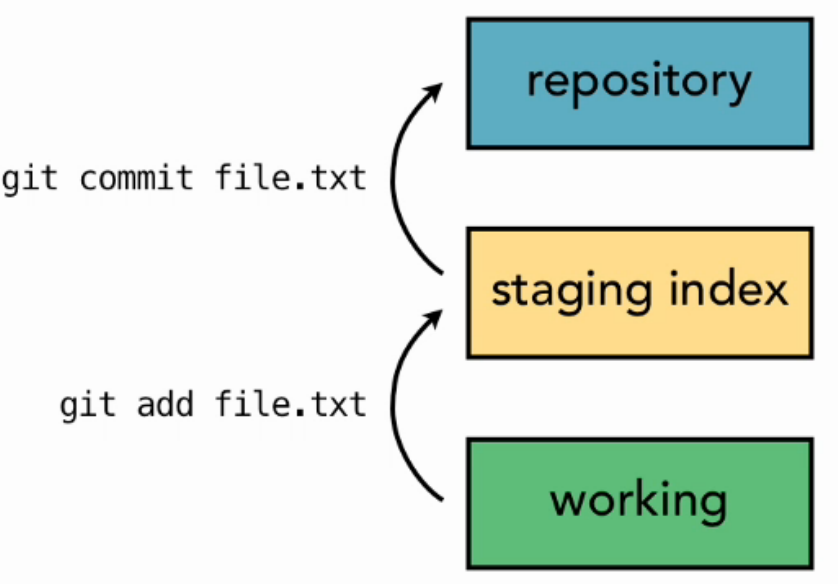
Git concepts:

Three-tree architecture.

Repository

Staging Index

Working Space.



Make changes of 10 files. You could add 5 of them to stage, then commit. Five of them on repository and working space. Another 5 changes stay on the working space. Not committed.

Git Workflow

New file:

Add the file to working space.

Use add command, the file exist in staging index now.

Commit, the file is pushed to repository.

Edit:

Change in working space. (v2)

Add push v2 to staging index.

Commit v2, to repository.

The repository will save all the history versions, rather than the newest one.

Referring to Commits.

V1,V2,V3(could represent many files, not only single file).

Generates a checksum for each change set. (SHA-1 Hash Algorithm)

So check checksum(Commit ID in git log) if the data has changed or not.

Git uses HEAD to reference the commit. HEAD points to current branch in repository (points to parent of next commit). It’s related to branches. HEAD always points to the newest commit or the check-out branch.

Making Changes to Files.

Use “git status” to check status. Untracked files(cannot tell changes).

ADD: //Add to staging index.

“Git add .” “Git add second\_file.txt”

Commit Changes: //after change something

Use add to add to staging index.

Commit –m “Message”

See changes of Files.

“git diff” //compare the changes to every changed file in working spac with comparison to the newest ones in repository and staging index.

“git diff --staged” //stage means cached, compare staging and repo

Delete files. //first tracked in our repo

//commit first and delete, git status to track

1.Drag file to trash, and use “git rm first\_file” + “git commit” //delete first file. And this will be cached. Then commit

Notice: Cannot delete on the disk and then commit.

2.easier. “git rm file\_to\_delete” + “Commit –m “abc””

//not in the trash, vanished

//the op is staged. Then commit

Renaming. //deleted+new untracked

1. change name + “git add file\_name” + “git rm file\_name” +”commit”

//after these, git can find that it’s renamed.

2.”git mv file\_name new\_file\_name” + “commit”

//easier. Move=Rename