GitHub Brief Instruction

# Base Concepts of Gits – Difference from Subversion

Everyone owns local repository and can be version controlled on local. Once the change is done, local repository can be updated from Github and push local change to Github.

# Git setup on Windows Desktop

## Install Git on local machine

<https://git-scm.com>

After Git installation, you are able to open a command line window by running “Git Bash”.

### Configure Git on local machine

1. Configure username

$git config –global user.name “Adam Wang”

1. Configure user’s email

$ git config –global user.name “awang@sanjel.com”

## Install favorite text editor on local machine

We use Notepad++ as example.

<https://notepad-plus-plus.org>

### Configure Notepad++ as default editor

1. Right-click “This PC”, and then click “Properties” to open the “System” window.
2. Click “Advanced system settings” to open the “System Properties” window.
3. Click “Environment Variables” to open the “Environment Variables” window.
4. Click “Path” variable entry to highlight it within the “User variables for [username]” list.
5. Click “Edit” button to open “Edit environment variable” window.
6. Click “Browse” button to open “Browse For Folder” window.
7. Locate the text editor installation folder. E.g. C:\Program Files (x86)\Notepad++. Then click “OK”. The installation folder will be added in the environment variable list.
8. Click “OK” on all opened windows to save the change.
9. Test text editor. Type in “notepad++” in Git Bash command line, notepad++ app will be opened.
10. Specify Notepad++ as the editor for Git.

$ git config core.editor “notepad++ -multiInst -nosession”

# Set up a Git repository on local

1. Create a folder as the home of your Git repositories. E.g. D:\code.

$ cd /d/

$ mkdir code

1. Get into “code” home folder

$ cd code

1. Initialize a repository for your application. E.g. MyApp

$ git init MyApp

A new folder “MyApp” is created under code folder, and a folder .git is created underneath which is local Git database.

1. Get into “MyApp” folder. In comman line window the folder path will show as “/d/code/MyApp (master)”. That means the folder is created as master branch as default.
2. View the repository status.

$ git status

# Move existing code from SVN to a Git repository

1. Export existing code files from SVN to local repository folder.
2. Check Git repository status.

$ git status

You will see the imported folders/files are listed under “Untracked files” in red color.

1. Add imported folders/files to git repository to be tracked.

$ git add .

1. View the repository status, all added files are displayed under “Changes to be committed:” in green color. That means the files are staged.
2. Commit the changes to local repository.

$git commit -m “Import code from SVN”

1. Check Git repository status.

$ git status

You will see “Nothing to commit, working tree clean”. That means all files are committed.

# Link local repository to GitHub repository

## Create Repository on GitHub

1. Login into your organization account on GitHub, create a private repository named “MyApp”.
2. You may find some information for Git operation under “Code” tab page.

## Link Local Repository to GitHub Repository

1. Link local repository to GitHub repository

$ git remote add origin https://github.com/Your-Company/MyApp.git

1. Check remote links

$ git remote -v

It will find remote references which show like following:

Origin https://github.com/Your-Company/MyApp.git (fetch)

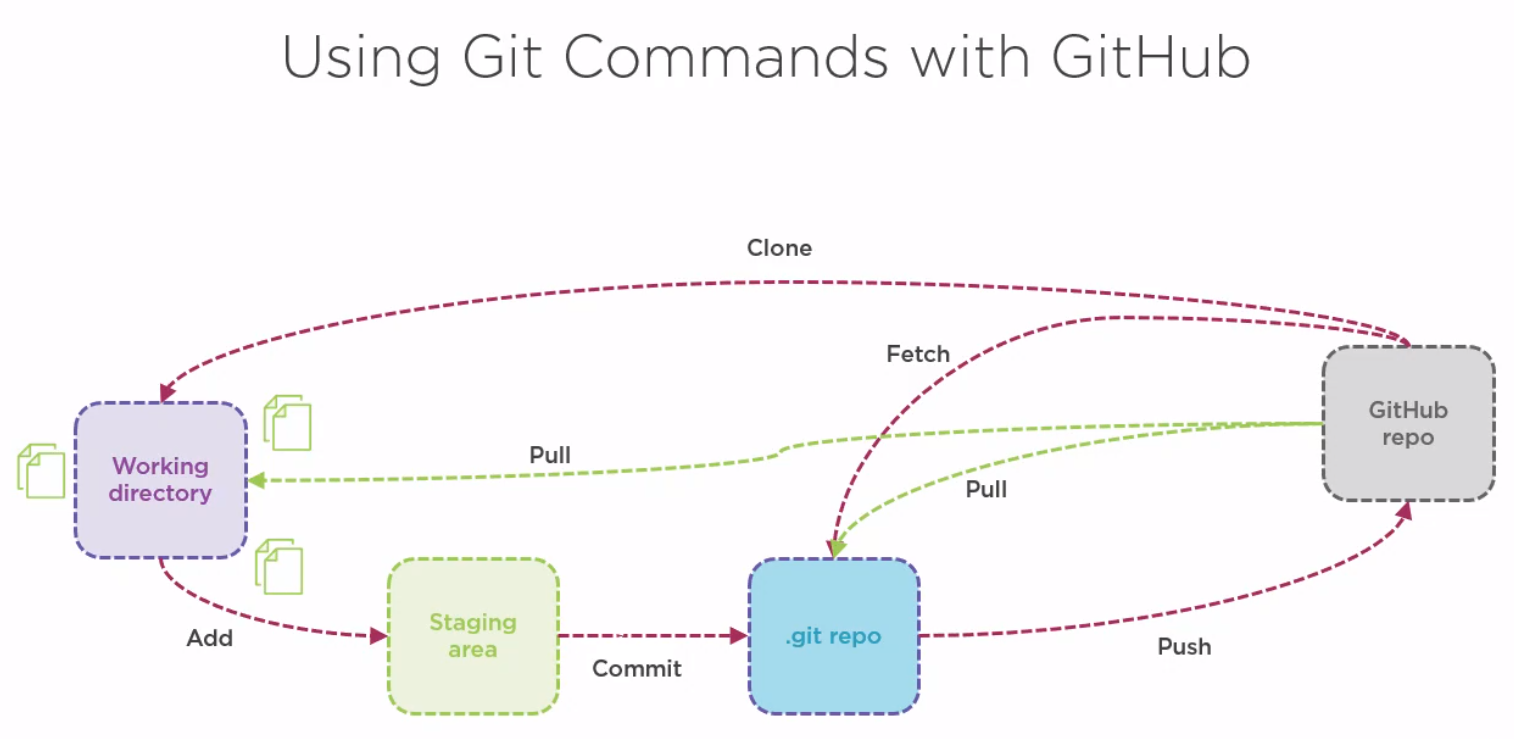
Origin https://github.com/Your-Company/MyApp.git (push)

1. Push local repository to GitHub repository

$ git push -u origin master

Command line console will show the progress local code being pushed to GitGub.

# Working with GitHub Locally



# Git vs Subversion

Git maintains a local repository which allows user to commit changes to local. So, it is convenient to monitor changes and roll back changes before push to server.

When Git is working with GitHub copy, it is very similar with Subversion. Local repository needs to pull server copy to allow local changes merged with other changes committed to GitHub. Once the merge is done, the conflicts are solved, the local copy needs to be pushed to GitHub.

We can consider Git is two step Subversion operations. To take the advantage of this feature. We would encourage developers to make commitment to local more often, basically on task level.

Let’s make some mapping between Git vs Subversion to help easy learning. But you have to pay attention that the concepts might not be exactly same.

## Git: Pull vs SVN:Update

Before you commit your working copy to GitHub, you must merge the changes made by other people. Git will merge the changes automatically and mark the conflict if exists. The conflict is marked in the files. However, Git doesn’t mark conflict files explicitly as Subversion does. It doesn’t have separate files of original version, local version and merge version.

### Git: Push vs SVN: Commit

They are very similar.

### Steps to commit local changes

1. Stage local changes

$ git add .

1. Commit changes to local repository

$ git commit -m “Description of the change”

1. Pull the change from GitGub

$ git pull

1. Fix conflicts. Once you have pulled changes from GitHub, if there are conflicts generated in files, you may find them out with command line.

$git diff –name-only –diff-fliter=U

1. Stage the changes. Remember, Git doesn’t hold your change staging against the unsolved conflicts. Once the files are staged, you are not able to find them out by using command line in step 4.

$ git add .

1. Commit changes to local repository

$ git commit -m “Description of the conflict fix”

1. Push changes to GitHub

$ git push.

Tips: To find all unresolved conflicts under your folder, following command may be easier and reliable.

$ grep -r “<<< HEAD” \*

### Git: clean/reset vs SVN: cleanup/revert

1. Revert all changes in tracked files

$ git reset – hard

1. Remove new added untracked files

$ git clean -f -d