Git Instructions

In order to use the git instructions you need to install git in your local machines.

For Mac:

First install “brew” package manager for mac.

From: “http://brew.sh/”

Then execute the command in your terminal:

“brew install git” without the quotes

Tt will install git for you. You can run the git instructions in the terminal. If you are a sudoer add sudo at the start of the command.

For Windows:

Open the cmd prompt and install “chocolatey” package manager for windows.

From: “https://chocolatey.org/”

Then execute the command in your cmd prompt:

“choco install git” without the quotes

It will install git for you. It will install git bash for you too.

Go to search in the start menu and search for git bash and open it. Run your git instructions there.

First create a file in a directory and then give the path to it in your terminal or git bash. Then you can perform the following instructions.

Initializing a Repository in an Existing Directory

- To know what you are accessing in github, you need to do the following steps:

$ git config –global user.name "your username"

$ git config –global user.email "your email address"

To make sure you can access this account enter these commands

Ex: $ git config –global user.name "hakuva"

Ex: $ git config –global user.email "**********@gmail.com"

- $ git init

This creates a new subdirectory named .git that contains all of your necessary repository files — a Git repository skeleton. At this point, nothing in your project is tracked yet.

- If you want to start version-controlling existing files you should probably begin tracking those files and do so in an initial commit. You can accomplish that with a few "git add" commands that specify the files you want to track, followed by a commit:

$ git add *.c (or) "$ git add –A ." to add all

$ git add README to add a specific file.

$ git commit -m "initial project version" to commit the changes performed

If you want to get a copy of an existing Git repository — for example, a project you'd like to contribute to — the command you need is git clone.

You clone a repository with git **clone** [url]. For example, if you want to clone the Ruby Git library called Grit, you can do so like this:

$ git clone **https://github.com/tjohnson1965/MSU2U-iOS.git**

- Checking the **Status** of Your Files

The main tool you use to determine which files are in which state is the git status command. If you run this command directly after a clone, you should see something like this:

$ git status

o/p:

On branch master

nothing to commit (working directory clean)

If the file didn’t exist before, and you run git status, you see your untracked file like so:

$ git status

o/p:

On branch master

Untracked files:

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

README

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

- **Log:**

When you run git log in this project, you should get output that looks something like this:

$ git log

o/p:

commit ca82a6dff817ec66f44342007202690a93763949

Author: blah

Date: blah

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7

Author: blah

Date: blah

    removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6

Author: blah

Date: blah

    first commit

- **Revert**:

$ git revert “some huge commit address from git status”

Reverts back the commit by one step.

Before performing git revert, perform git status to see all the commits performed. Copy one of the commit addresses at the side of commit in git status.

- **Reset:**

$ git reset “some huge commit address from git status”

Reset the whole repository to the commit we select.

Same instructions as git revert

- **Tags:**

Listing the available tags (used for marking release points) in Git is straightforward. Just type git tag:

$ git tag

v0.1

v1.3

- **Help:**

If you ever need help while using Git, there are three ways to get the manual page (manpage) help for any of the Git commands:

$ git help <verb>

$ git <verb> --help

$ man git-<verb>

- **Branches:**

Let’s say you create a new branch called testing. You do this with the git branch command:

$ git branch testing

To switch to an existing branch, you run the git checkout command. Let’s switch to the new testing branch:

$ git checkout testing

- **Push commits**

Once you've made some commits to a forked repository and want to push the changes to your forked project, is done the same way you would with a regular repository:

$ git push origin master

Pushes commits to your remote repository stored on GitHub

- **Pull in upstream changes**

If the original repository you forked your project from gets updated, you can add those updates to your fork by running the following code:

$ git remote add upstream “your repository link”

$ git fetch upstream

Fetches any new changes from the original repository

$ git merge upstream/master

Merges any changes fetched into your working files

(or)

Even $ git pull “ur repo ssh” does same but the above method is preferable.