Basic Git Workshop

TechJI

University of Michigan - Shanghai Jiaotong University

Joint Institute

2023.10.8

Table of Contents

Before we start

- This is **not** a Linux workshop (although I encourage you to use it).
- This is **not** a Vim workshop (although I encourage you to use it).
- This is **not** a Bash workshop either.
- We are organizing this workshop primarily because many students encountered difficulties when using Git in courses ENGR151 and SilverFOCS(VG100).
- The most important part in this workshop is *lazygit*.
- The target audience for this workshop is students who are unfamiliar with Git or do not understand the working principles of Git.

Table of Contents

What is Git?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

When to use Git?

- Writing code
- Managing docs
- Managing projects
- Version control
- Team corporation

Git Installation

You can go to the official website to download the installation package and refer to *Installation_git* file in the repo.

Where to use Git?

We first introduce shell commands, then we will use Git in the shell.

What is Shell?

Shell is a program that takes commands from the keyboard and gives them to the operating system to perform. It is also referred to as a command-line interpreter or CLI. Common shells:

- Bash
- Zsh
- etc

A brief history of bash

- Born: 1989
- Probably played Pokémon on the Game Boy
- Is an umbrella term for zsh, fish, ...
- Runs on Unix-like environments

A brief history of Unix

- Born: 1969
- Probably listened to Michael Jackson
- Gave rise to Linux, BSD, and Mac OS
- We call them "Unix-like"

Unix: The Good Part

The Unix philosophy (paraphrased):

- Store data in plain text
- Hierarchical file system
- Everything is a file
- One tool does one thing
- Tools together strong

Quote

The power of a system comes more from the relationships among programs than from the programs themselves.

— Brian Kernighan and Rob Pike [1]

[1] The UNIX Programming Environment. 1984. viii

How to open a Shell?

- Windows: cmd, powershell, Git Bash
- Mac: Terminal
- Linux: Terminal

Table of Contents

Files

Each of these is a different **file**:

- `a`
- `.a (Hidden)`
- `a.txt`
- `A.txt`
- `A.TXT`

Note

The dot and suffix are part of the filename. Windows users please turn on **show file extensions**.

Avoid spaces and special characters (except `._-`). If you have to, surround filename in quotes: ‘`Lab Report (3) final FINAL-1.docx`’

Directories

Each of these is a **directory** ("dir" for short):

- `hteam-10086/`
- `hteam-10086/h1/`
- `hteam-10086/.gitea/ (Hidden dir)`

Convention

For clarity, we add a slash (`/`) to the end of a directory in the slides. However, in reality it often makes no difference.

cd, pwd: Changing directory

- `cd hteam-10086/`
- `pwd`

Explanation

- `cd:` "change directory"
- `pwd:` "print working directory"
- `../` means "parent directory"
- `./` means "current directory"
- `~` means "home directory"

Paths

File $\cup$ directory $=$ **path**. [2]

No paths under the same directory can bear the same name. These **cannot** coexist:

- `hteam-10086/h1/, a directory`
- `hteam-10086/h/ex1.m, a regular file`

[2] At least in the scope of this workshop.

Absolute & relative paths

- Paths beginning with `/` are absolute: `/usr/bin/cat`
- Otherwise it is relative: `hteam-10086`

If you know where you are, you can convert a relative path to an absolute one.

ls: Listing directories

- ls
- ls -a
- ls -l
- ls -la

Explanation

- `ls: ‘‘list’’`
- `-a is short for --all`
- `-l enables long listing format`
- `-la = -l + -a`

More...

More usage of Bash and Shell will be explored in the spring semester's Bash Workshop. You can refer to the cheatsheet in this repository.

JOINT INSTITUTE
交大密西根学院

Table of Contents

Some basic git commands

Tell Git who you are:

- git config –global user.name "Your Name"
- git config –global user.email "Your Email"

Some basic git commands

How to use "git add":

- git add *file*
- git add *
- git add .
- git add -A

use "git status" to check the status of your repo.

Some basic git commands

If you want some files never be tracked by git, you can create a file named ".gitignore" in your repo, and write the file names in it.

Some basic git commands

How to use "git commit":

- git commit -m "commit message"

If you only type:

- git commit

then you will enter vim /other default editor to write your commit message.
How to write commit message?

Some basic git commands

How to use "git push":

- git push origin *branch*
- git push origin master
- git push

Some basic git commands

How to use "git rm":

- git rm –cached *file*

Then, git will stop tracking this file, but the file still exists in your repo.

Some basic git commands

How to delte a file in your remote repo:

- git rm *file*
- git commit -m "remove file"
- git push

Some basic git commands

How to use "git pull":

- git pull

Some basic git commands

How to use branch in git:

- create new branch(based on current branch): git checkout -b *branch*
- go to other branch: git checkout *branch*
- delete branch: git branch -d *branch*

Some basic git commands

How to use "git merge": For example, we are now on branch "master", and we want to merge branch "dev" to "master":

- git checkout dev
- git pull
- git checkout master
- git merge dev
- git push

If you encounter conflicts, you need to solve them manually.
In most cases, Git will attempt to auto-merge first. If a merge conflict occurs, Git will inform you which file has a conflict, and you'll have to manually edit that file.
Suggestions: You can use "lazygit" or search for "Git merge tool" online.

Some basic git commands

How to use "git revert":

- git revert *commit* – *SHA*
- git revert HEAD

Table of Contents

Lazygit

Lazygit is a powerful Git frontend that integrates many git commands. It is written in Go and is cross-platform.

Lazygit

Lazygit is a powerful Git frontend that integrates many git commands. It is written in Go and is cross-platform.
Why lazygit?

- Fast, TUI
- Easy to use
- Powerful

Other alternatives are Gitui which is written in Rust.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.

[3] Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.

[3] Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal

[3] Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal
- Navigate the interface (h/j/k/l/[/]/arrow keys).

[3] Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal
- Navigate the interface (h/j/k/l/[/]/arrow keys).
- Stage/Unstage files (a/A).

[3] Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal
- Navigate the interface (h/j/k/l/[/]/arrow keys).
- Stage/Unstage files (a/A).
- Commit changes (c).

[3]Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal
- Navigate the interface (h/j/k/l/[/]/arrow keys).
- Stage/Unstage files (a/A).
- Commit changes (c).
- Push files (P).

[3]Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal
- Navigate the interface (h/j/k/l/[/]/arrow keys).
- Stage/Unstage files (a/A).
- Commit changes (c).
- Push files (P).
- Create/delete branches (n/d).

[3]Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal
- Navigate the interface (h/j/k/l/[/]/arrow keys).
- Stage/Unstage files (a/A).
- Commit changes (c).
- Push files (P).
- Create/delete branches (n/d).
- Checkout branches (space).

[3]Operations with * will rewrite history.

Tasks[3]

- Install lazygit. You may refer to *Installation_Lazygit* in repo.
- Clone your repository.
- cd into your repo and type "lazygit" in terminal
- Navigate the interface (h/j/k/l/[/]/arrow keys).
- Stage/Unstage files (a/A).
- Commit changes (c).
- Push files (P).
- Create/delete branches (n/d).
- Checkout branches (space).
- Merge branches (M).

[3]Operations with * will rewrite history.

More...

More usage of Git will be explored in the next year's Advanced Git Workshop. You can refer to the cheatsheet in this repository.

References

- Pro Git. Git - Book
- fakefred/bash-workshop
- linsyking/git-wksp

Reading Materials

- (Highly recommend)TheCW-Git *Bilibili*. BV1Yx411f7Cu
- (Highly recommend)TheCW-Lazygit *Bilibili*. BV1gV411k7fC
- Pro Git. Git - Book
- MIT Course
- Learn git online
- Liao Xuefeng's Git Tutorial(Chinese)

Thanks for listening!