

Introduction to Git

Linghui Wu

BFI

References:

- <https://git-scm.com/>
- Rahaman MPC5-51042

Contents

1 Git Concepts

2 Using Git

What is version control system (VCS)?

- A version control system (VCS) records changes to a set of files
 - All the recorded changes are called **history**
 - The VCS can retrieve specific, previous versions of files
- Why is VCS?
 - If a bug is detected after deployment, you can easily revert to a working version
 - Can help discover which specific changes caused the bug
 - Can discover who was responsible for that change

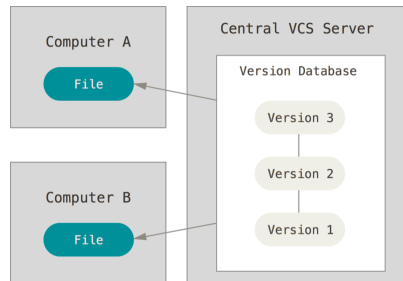
Centralized VCS

- Pros

- Individuals can see everyone's activity.
- Admin has fine-grained control over database permissions.

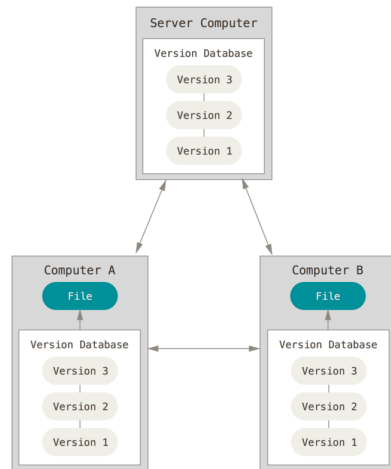
- Cons

- Developers has restricted access to the repository and cannot update if their connection is offline.
- Hopefully database is backed-up elsewhere if central system is corrupted.



Distributed VCS - Git

- Everyone has entire database
 - Arbitrarily-many remote servers
 - Arbitrarily-many local clients
- Pros
 - Any server or client can recover the database.
 - Users can work offline.
 - Support non-linear workflows
- Cons
 - Learning curve



Getting and setting-up Git

- Many choices of clients
 - Highly-recommended command line tools (<https://git-scm.com/downloads>)
 - GUI clients (<https://git-scm.com/downloads/guis>)
 - IDE-integrated clients
- First-time set-up
 - Check version

```
$ git --version
```
 - Set config variables: your CNetID and @uchicago.edu email.

```
$ git config --global user.name "Linghui Wu"
$ git config --global user.email "linghuiwu@uchicago.edu"
$ git config --list
```
 - Need help?

```
$ git help <verb>
$ git <verb> --help
```

Creating or downloading project

- Initialize a repository from existing code

```
$ mkdir git-demo  
$ cd git-demo  
$ git init
```

- Downloading an existing repository using SSH

```
$ git clone git@github.com:linghui-wu/RP-orientation-2022.git
```

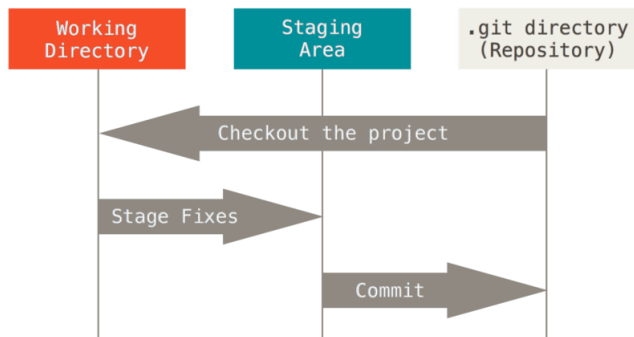
Before first commit

- Check file status

```
$ git status .
```

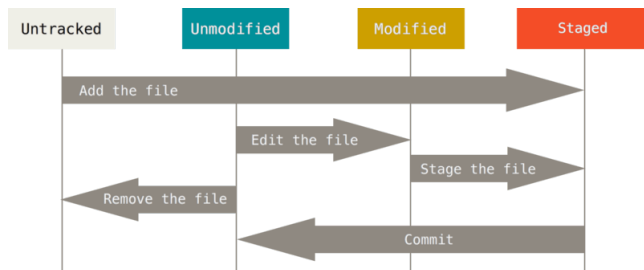
- Create .gitignore file to intentionally ignore untracked/modifeid files

Three file states



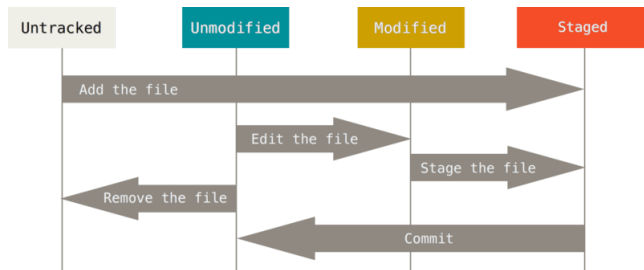
- Working directory stores untracked and modified files
- Staging area organizes changes we want to commit to the repository

Adding local changes



- For new files
 - Untracked → tracked: `git add <file_name>`
- For existing files
 - Unmodified → modified: any text editor
 - Modified → staged: `git add <file_name>`
 - Staged unmodified: `git commit -m "<commit message>"`

Undoing local changes



- Staged → modified: `git reset -- <file_name>`
- Modified unmodified:
 - Lose changes forever: `git checkout -- <file_name>`
 - Keep changes for later: `git stash`
- Unmodified untracked: `git rm --cached <file_name>`

Showing and adding remote databases

- Every remote that is known to the local repository has a short name and a URL
- To show existing remote repositories

```
$ git remote -v
```

- To add a new remote

```
$ git remote add <remote_name> <url>
```

Getting commits to and from remotes

- Get the latest changes from a remote

```
$ git pull <remote_name> <branch_name>
```

```
linghuin@8f1-14893L:RP-orientation-2022 % git pull upstream main
hint: Pulling without specifying how to reconcile divergent branches is
hint: discouraged. You can squelch this message by running one of the following
hint: commands sometime before your next pull:
hint:
hint:   git config pull.rebase false  # merge (the default strategy)
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
From github.com:energy-policy-institute-uchicago/RP-orientation-2022
 * branch      main
Updating 0890bb1..017e0fe
Fast-forward
 Intro to Python/Basics.ipynb      | 1246 +++++
 Intro to Python/NumPy.ipynb      | 103  ++
 Intro to Python/Pandas.ipynb     | 1017 +++++
 Intro to MATLAB/Intro to MATLAB__Exercises.pdf | 81n 0 -> 13248 bytes
 Intro to MATLAB/Matlab_Orientation.m | 497  +++++
 Intro to MATLAB/TA_session.m     | 29   ++
 Intro to MATLAB/datafile.mat      | 81n 0 -> 244 bytes
 Intro to MATLAB/depends/simulate_loan_lifetime.m | 61  +++
 Intro to MATLAB/element.csv       | 8     +-
 Intro to MATLAB/filename.mat      | 81n 0 -> 244 bytes
 18 files changed, 2447 insertions(+), 604 deletions(-)
 create mode 180644 Intro to Python/Basics.ipynb
 create mode 180644 Intro to MATLAB/Intro to MATLAB__Exercises.pdf
 create mode 180644 Intro to MATLAB/Matlab_Orientation.m
 create mode 180644 Intro to MATLAB/TA_session.m
 create mode 180644 Intro to MATLAB/datafile.mat
 create mode 180644 Intro to MATLAB/depends/simulate_loan_lifetime.m
 create mode 180644 Intro to MATLAB/element.csv
 create mode 180644 Intro to MATLAB/filename.mat
```

- Send committed local changes to a remote

```
$ git push <remote_name> <branch_name>
```

- ALWAYS pull before you push

Branching

- Create a branch for desired feature

```
$ git checkout -b <branch_name>
```

It is a shorthand for

```
$ git branch <branch_name>
```

```
$ git checkout <branch_name>
```

- After committing the changes, push the branch to remote repo

```
$ git push <remote_name> <branch_name>:<branch_name>
```

- Create a pull request and merge your branch into main branch
- Delete the branch

```
$ git branch -d <branch_name>
```