ICCS207: Term I/2018-19

Lecture 5: Version Control and Git

Sunsern Cheamanunkul (sunsern.che@mahidol.edu)

Mahidol University
International College

Why version control?

- "I swear my code worked yesterday".
- "I shouldn't have deleted and rewritten my code"
- "My computer crashed. I lost all of my work"
- "I work with my friend. He accidentally deleted my code"

Version Control System (VCS)

- Tracks every change
  - Allows going back in time
- Facilitates experimenting
- Collaboration
  - Each can work independently
- Tracks individual contributions
  - and blames…

VCS

Distributed VCS

- Unlike centralized VCS, users don't need to constantly talk to a central server.
- Full local access to every file and branch, while keeping full history of all changes.
- Changes are periodically synced to a central server.
- A good balance of collaborating and working independently.

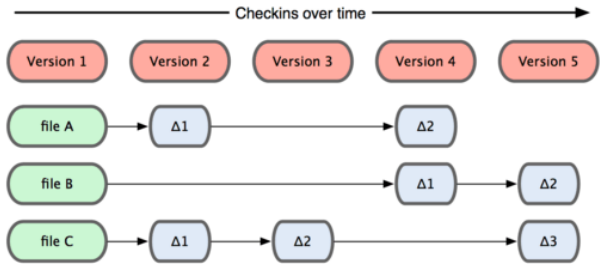
Git

- (Most) popular distributed VCS
- Known for
  - Easy branching and merging
  - Flexible workflow
  - Data integrity
  - Performance

Repository

- Contains collection of files and folders, along with each file's revision history.
- File history appears as snapshots in time called ***commits***.
- A repository can have multiple ***branches*** where each branch represents a line of development

Versions



Creating a new repo

- Initializes a brand new Git repository and begins tracking an existing directory.
- Creates a hidden .git directory

```
git init
```

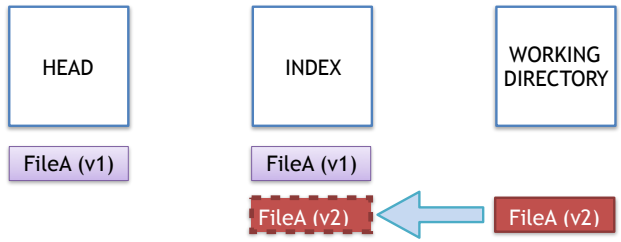

Cloning an existing repo

- Creates a local copy of a project that already exists remotely

```
git clone <repo>
```

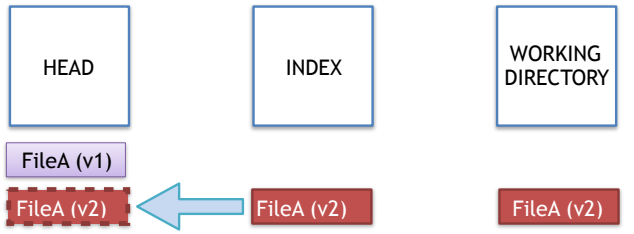

Understand Git

Stage a change



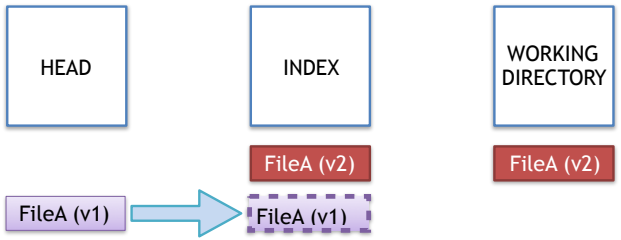

```
git add <files>
```

Commit a change



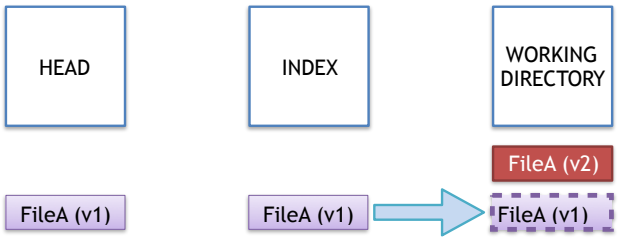

```
git commit -m <message>
```


Unstage a file




```
git reset <files>
```


Discard changes

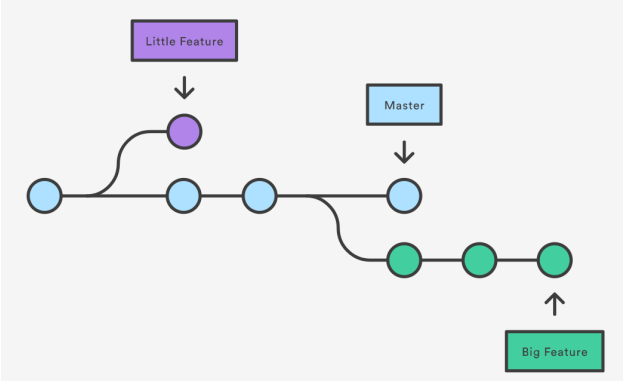



```
git reset --hard
```

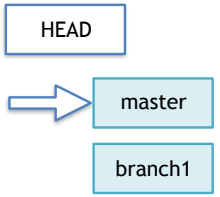

Repo Status

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   lect/05/05-git.key

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   lect/05/05-git.key

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

    assn/01/code/guest.csv
```

Branching

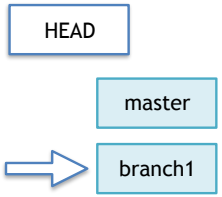


Switching branches



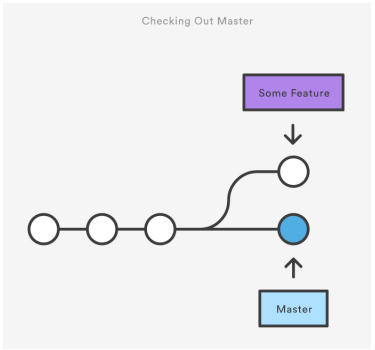

```
git checkout master
```

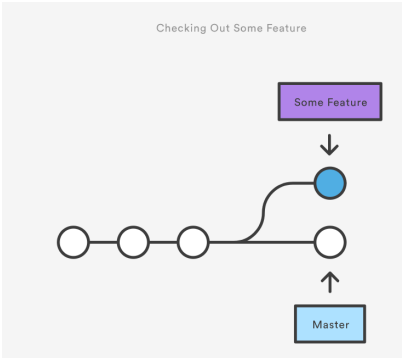

Switching branches



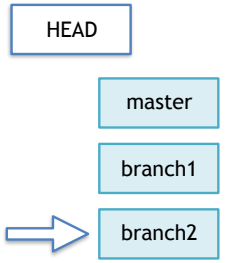

```
git checkout branch1
```


git checkout



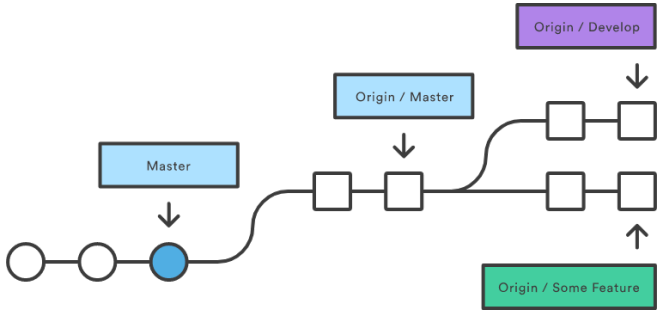


Create a new branch

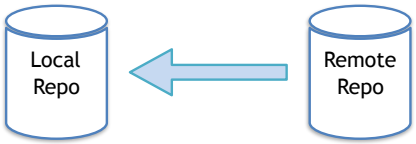



```
git checkout -b branch2
```


Remote Git repo

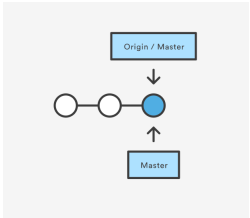


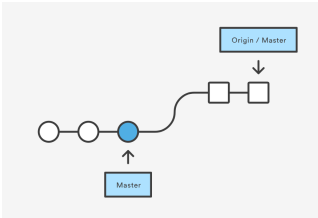
Working with remote repo

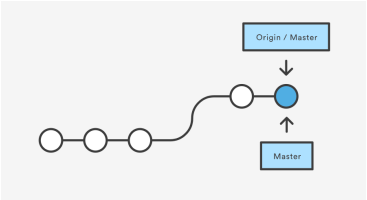



```
git pull
```

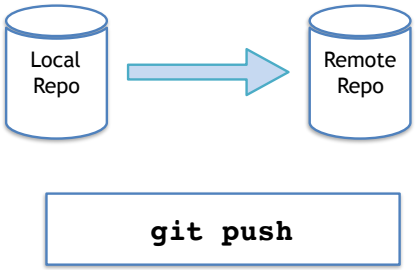

git pull



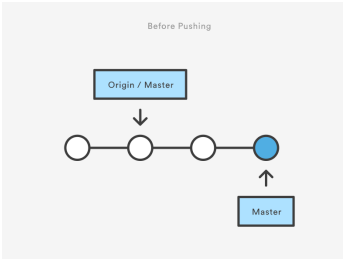


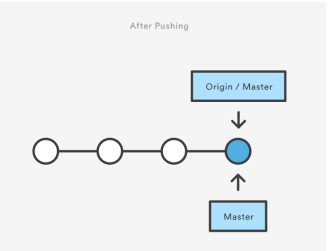


Working with remote repo

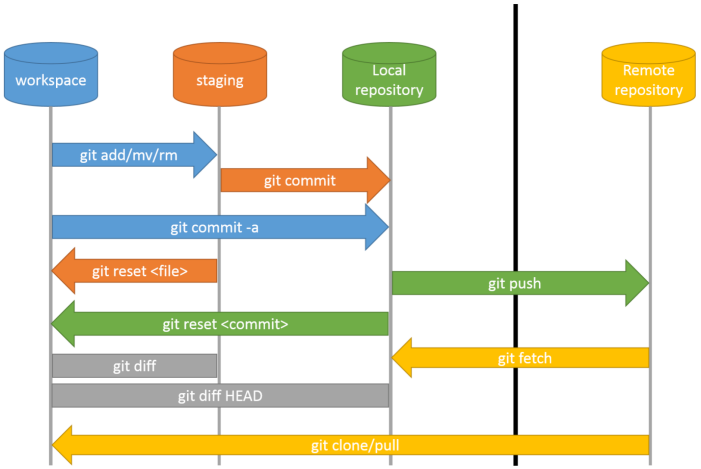


git push





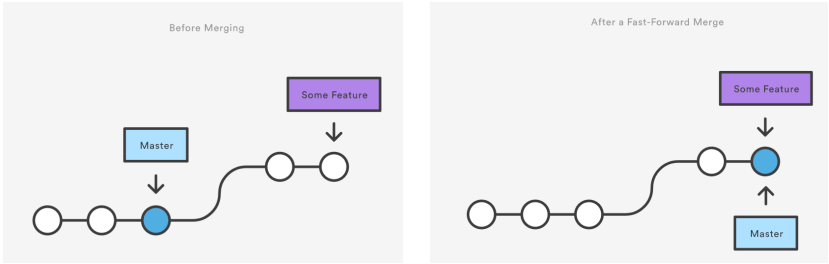
Summary of Git Workflow



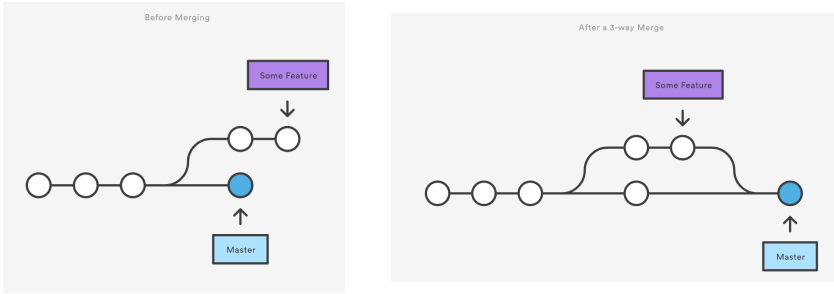
Resolving conflicts

- Local and remote repo could be in a conflicting state
  - e.g. two people updating the same file in the remote repo
- The person who commits later has to resolve the conflict by merging the changes.

git merge (fast-forward)



git merge (3-way)



Git commands

- init — initialize a repository
- clone — clone a repository
- status — get information about a repository
- log — view the history and commit messages
- add — add a file to the staging area.
- commit — commit your changes to local repository
- push — push changes to a remote repository
- pull — pull changes from a remote repository
- checkout — retrieve a specific version of a file