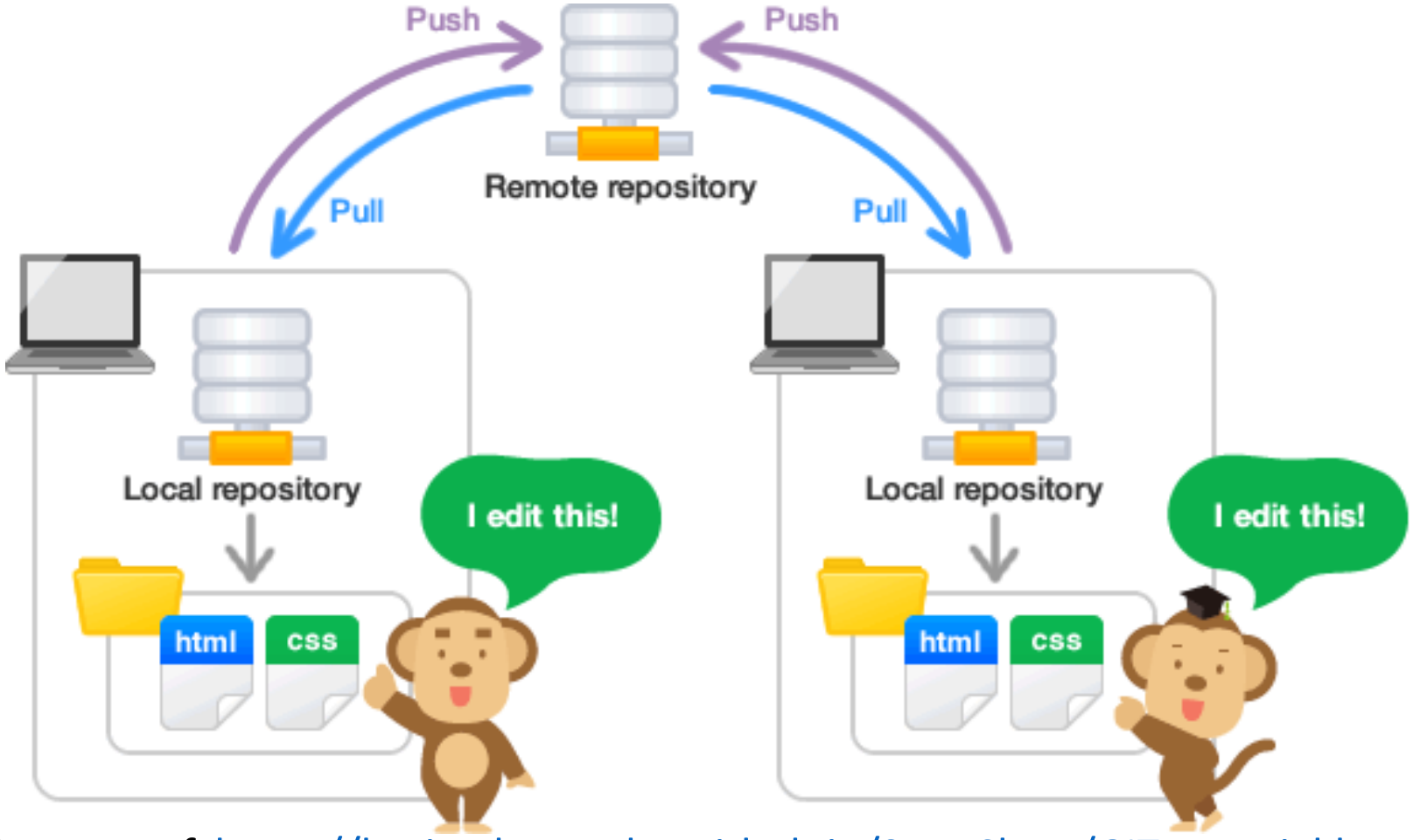
Using Git

Wonsun Ahn

Git Basics

- A means for software versioning and collaboration



Courtesy of: https://kevintshoemaker.github.io/StatsChats/GIT_tutorial.html

Git Etiquette 1: Push As Soon As Possible

- Your group member may be waiting …
  - For a feature to be implemented
  - For a bug to be debugged

- Push as soon as you have made a change that improves the project
  - If you delay pushing that change, the entire project will be delayed!
  - Pushing most of your code 2 days before the deadline is unacceptable

Git Etiquette 2: Leave a Descriptive Message

- You are required to leave a message whenever you commit
- Leave something descriptive so that your partner knows what you changed and what you are still working on

Git Etiquette 3: Do Not Push Bugs

- Worst thing you can do is to push a compile error
  - That means the project can no longer compile and no longer run
  - Entire project will be delayed until the error is fixed

- Do not push defects either
  - Do regression testing before pushing (run all unit tests written so far)
  - Make sure it doesn’t break something that used to run well

Git Etiquette 4: Pull / Push Frequently

- Before doing code changes → always pull the most recent version
  - Ensures that you work on the most up-to-date version
- After doing code changes → always push the committed changes
  - Ensures that your group members are work on the most up-to-date version
- If you do this, vast majority of changes will be sequentially ordered
  - Example of two ordered changes (blue by member 1, red by member 2):
    pull version 1 → update to version 2 → push version 2 →
    pull version 2 → update to version 3 → push version 3
    ☛ version 3 is applied on top of version 2
  - Example of two unordered changes:
    pull version 1 → pull version 1 → update to version 2 → update to version 2' →
    push version 2 → push version 2'
    ☛ version 2' is not applied on top of version 2. What to do? Have to merge!

Merging Two Changes

- When two changes are unordered, Git will attempt to merge automatically
  - When the two changes are to different sets of files → success!
  - When the two changes are to the same file but different methods → success!
  - When the two changes are to the same source code line or very close → merge conflict!
- Example of a merge conflict in source code:

```
<<<<<<< HEAD
x++; (local version)
=======
y++; (remote version)
>>>>>>> 542582954eae8845ba6b0498d7d04ac09a6a63c6
```

- Must resolve by replacing above with whatever makes sense. Options:
  - `x++;`
  - `y++;`
  - `x++; y++;`
  - Or any other code that correctly merges the two changes

Merging Two Changes

- After resolving all conflicts, “commit merge” on GitHut Desktop

- Using GitHub Desktop to merge conflicts (recommended): https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-on-github

- Using Git command line mergetool: https://gist.github.com/karenyyng/f19ff75c60f18b4b8149