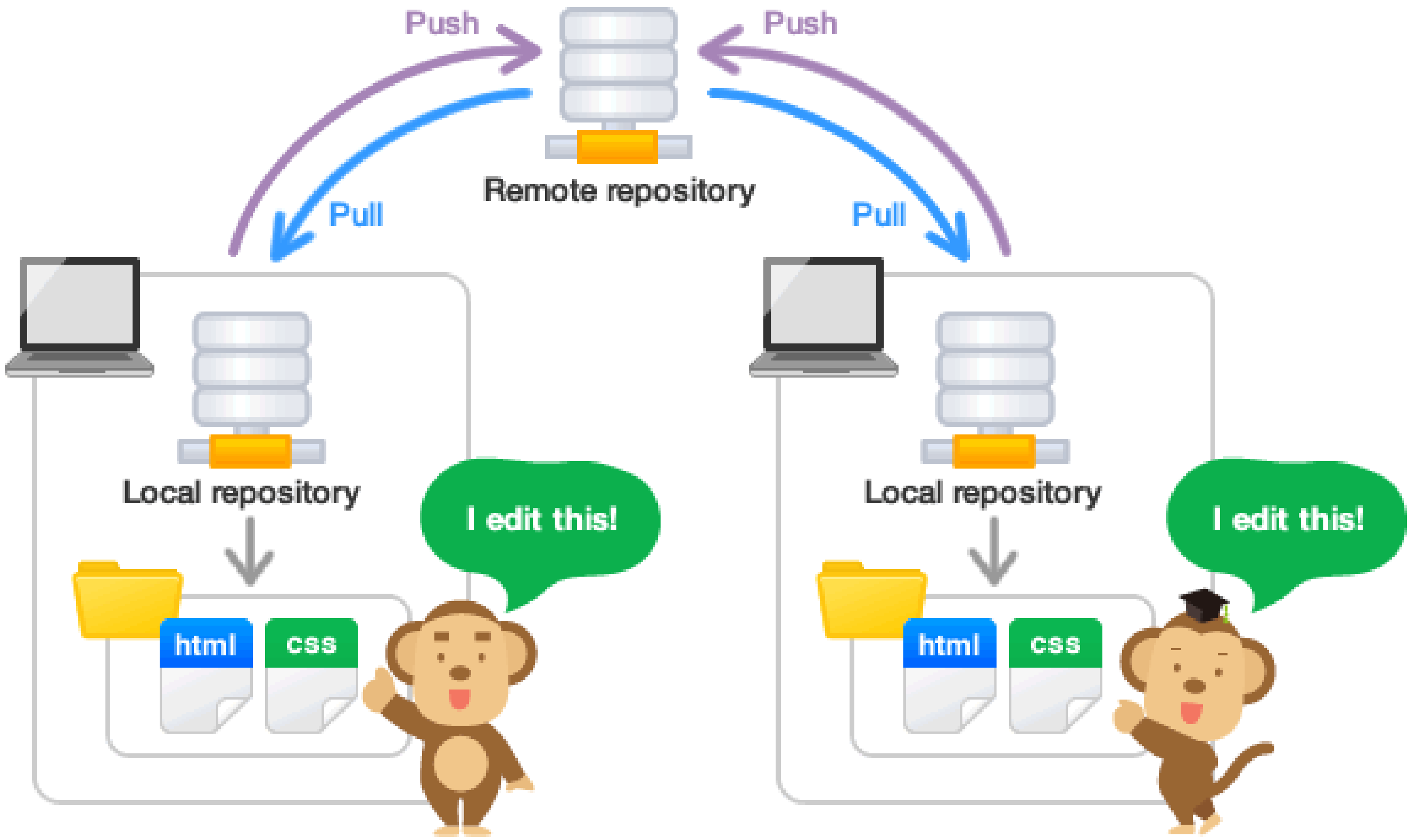
Using Git

Wonsun Ahn

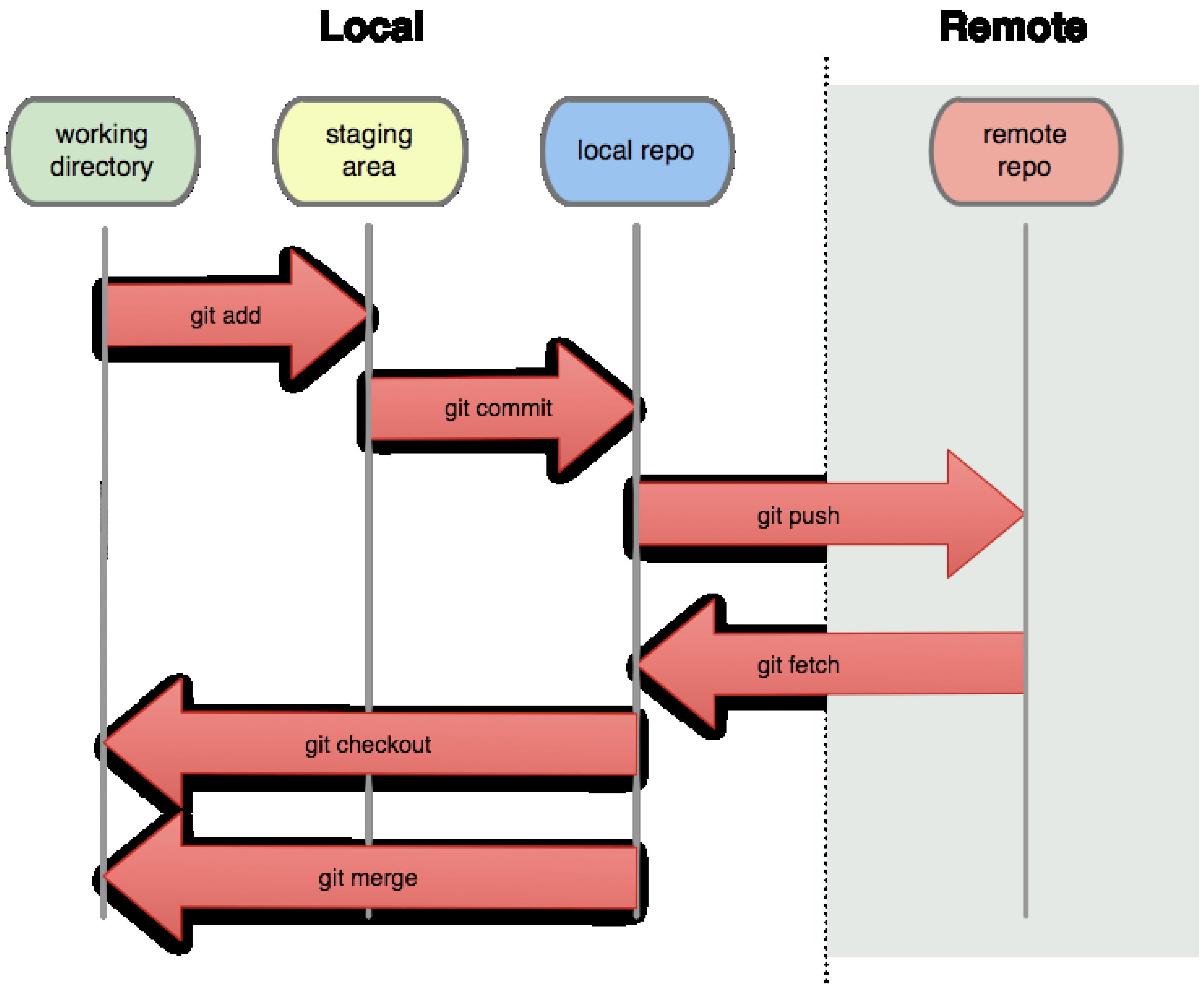
Courtesy of:
https://kevintshoemaker.github.io/StatsChats/GIT_tutorial.html

Git Basics

- A means for software versioning and collaboration



Git Basi


Local
Remote
working directory
staging area
local repo
remote repo
git add
git commit
git push
git fetch
git checkout
git merge

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
- If you delay pushing that change, the entire project will be delayed!
- Push as soon as you have made a change that improves the project
- 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

- We learned TDD; apply the lessons
  - 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 Frequently

- Always, before doing code changes pull the most recent version
- Ensures that you are working on the most up-to-date version
- Ensures that you don't get any merge conflicts
  - Merge conflict: when Git has trouble merging two simultaneous changes
  - Simultaneous change: two changes that are not ordered as follows:
    pull → change by partner 1 → push → pull → change by partner 2 → push
    but instead, look like this:
    pull → pull → change by partner 1 → change by partner 2 → push → push
    now change 2 is not built on top of change 1, so they have to be merged
  - Pushing and pulling frequently is how you avoid merge conflicts

What to do on merge conflicts

- It’s best to avoid them as much as possible

- When the two changes are to two different files
  - Merge conflicts will never happen

- When the two changes are to two different methods
  - Git usually finds a way to merge them correctly

- When the two changes are to the same method
  - Git may throw up its hands and punt to you
  - How to deal with it: https://gist.github.com/karenyyng/f19ff75c60f18b4b8149