601.220 Intermediate Programming

Git

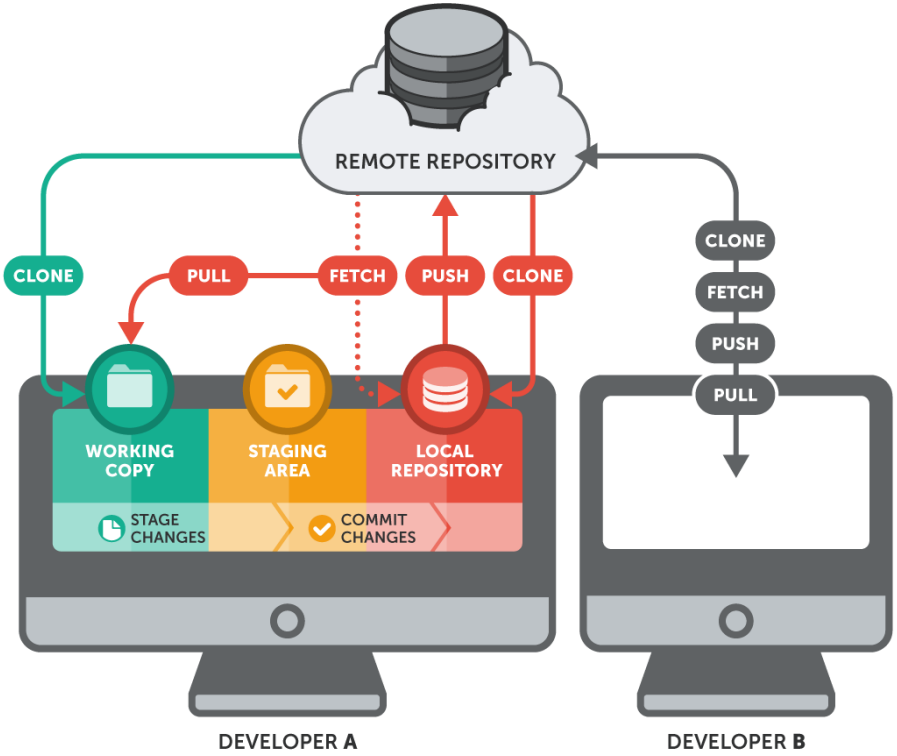
Plan for today

- Git
- Cloning our class public repositories using git

Git

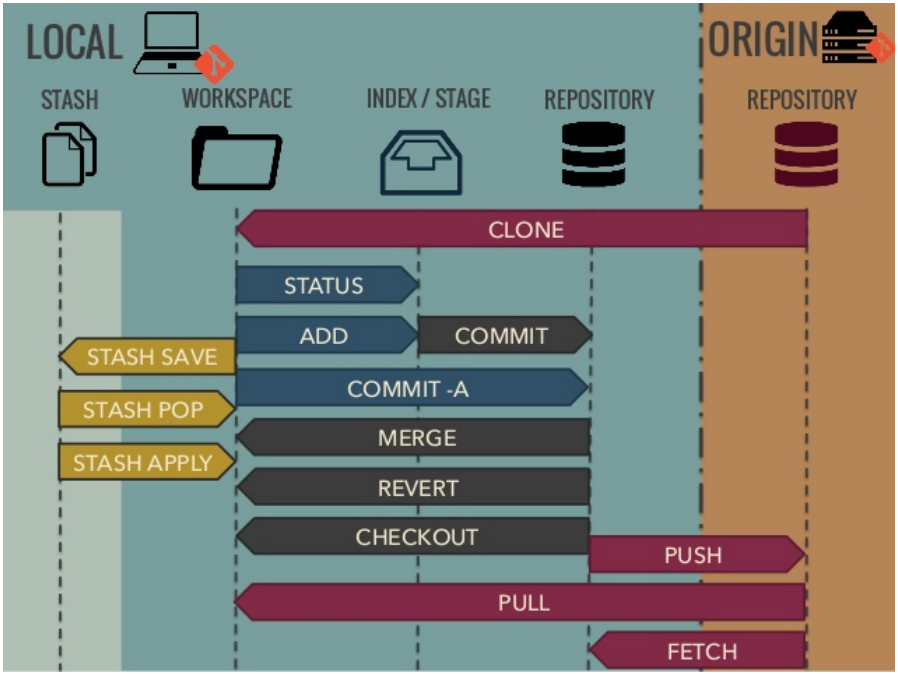
- Git is a way of sharing files; like DropBox or GoogleDrive, only much more powerful (and great for sharing code)
- Distributed version control
- Facilitates collaboration, snapshots, sharing
- Basic software skill, along with programming
- Works with any programming language; really, any project that consists of mostly text files

Git



- From `www.git-tower.com/learn/git/ebook/command-line/remote-repositories/introduction`

Git



- From `www.slideshare.net/origamiaddict/git-get-ready-to-use-it`

Git

- In your working copy, you can go about your usual business:
  - Editing files (with `emacs`, `vim`, etc)
  - Compiling and executing files
- But you'll also perform some repo-related tasks
  - `git add <file>`: add to project ("stage a file")
  - `git commit -m "commit message"`: update local repo to include changes since last commit ("take a local snapshot")
  - `git push`: send changes up to remote repo (on github)
  - `git status`: check what's been modified or staged, etc.
- Can't modify a repo directly using plain-old `mv` or `rm`; all interactions are via `git` command
  - `git mv <file> <file>`: rename a file
  - `git rm <file>`: remove a file (delete it)

- Full list: https://education.github.com/git-cheat-sheet-education.pdf

Git

- Files that are part of your project (you `git add`'ed them) are called *tracked*
- Tracked files can be in one of a few states
  - Unmodified (same as copy in local repo)
  - Modified (different from copy in local repo but not yet staged)
  - Staged (next `git commit` will update repo)
- editing files: Unmodified -> Modified
- `git add`: Modified -> Staged
- `git commit`: Staged -> Unmodified
- Information about changes in a copy of the repo is stored across several non-human-readable files in a subdirectory called `.git`
  - This subdirectory gets created for you when you clone a repo

Git

- Files that are *not* yet part of your project ("unstaged") are called *untracked*
  - When you create a new file; it's *unstaged* until you `git add` it
  - But git will notice it, and it will appear as unstaged if you check your `git status`
- Some untracked files are are files that we want git to "ignore", because we'll never want to include them in the remote repo
  - Tell `git` to ignore a file by adding it to `.gitignore` file
  - Good candidates for ignoring might be `a.out`, `gitlog.txt`
  - **Anything generated by the compiler (executables, .o files) should be in .gitignore**
  - We'll discuss this again soon

Git

- After `git clone` occurs, syncing between local and remote repos accomplished via `git pull` and `git push`
  - `git pull`: local repo asks for most updated copy from remote repo
  - `git push`: local repo sends all recent *commits* up to remote repo

Git

- Workflow Suggestions
  - Start each work session with `git pull`, to ensure your local copy is up-to-date
  - After you complete work on a small task, `commit` it
  - Include a message with every commit to explain what changes you committed (use -m, or you might be forced into an editor to create one!)
  - Make sure you commit and push before the end of each work session
  - To see a record of your latest commits displayed on the screen, you can type `git log`

Git

- Common `git` command orders
  - Step 1: Before you start working
    `git pull`
  - Step 2: After you've finished your edit
    `git add <files you edited>`
  - Step 3: Commit your changes with *comments*
    `git commit -m <comments>`
  - Step 4: Pull it one more time to sync with new updates if any
    `git pull`
  - Step 5: Solve conflicts if it happends (between your edit and new updates) and repeat step 2-5
  - Step 6: Push back to the repo
    `git push`

Git

- Don't be discouraged if `git` concepts are elusive at first
- You can get by with just a few key ideas
- `commit` early, `commit` often
- Tutorials and explanations linked from Resources section of Piazza (go to General Resources area, then click on Tools Reference)
- Lots of help available from CAs, instructors, Google, ...

Git

- Today, we want everyone to have access to class resources for this section
  - our class repository (repo) is hosted by github.com
  - can view the shared files in a web browser, but we want *local* copies to work with
  - today you'll *clone* the class repo into your ugrad account
  - when instructors add more to the repo, you can *pull* down updates
    - unlike Dropbox, git doesn't auto-sync the files in the repo

Our public file repository for this course

`https://github.com/jhu-ip/cs220-s22-public`

- contains files shared with you for use in this course
- open a web browser and view this repo

Getting a local copy of the repo

On ugrad, get into your home directory:

`cd ~` or just `cd`

Now clone the repo:

```
git clone https://github.com/jhu-ip/cs220-s22-public.git
```