GIT

Version Control System

Version control systems

- Version control (or revision control, or source control) is all about managing multiple versions of documents, programs, web sites, etc.
  - Almost all “real” projects use some kind of version control
  - Essential for team projects, but also very useful for individual projects
- Some well-known version control systems are CVS, Subversion, Mercurial, and Git
  - CVS and Subversion use a “central” repository; users “check out” files, work on them, and “check them in”
  - Mercurial and Git treat all repositories as equal
- Distributed systems like Mercurial and Git are newer and are gradually replacing centralized systems like CVS and Subversion

Why version control?

- For working by yourself:
  - Gives you a “time machine” for going back to earlier versions
  - Gives you great support for different versions (standalone, web app, etc.) of the same basic project
- For working with others:
  - Greatly simplifies concurrent work, merging changes

Why Git?

- Git has many advantages over earlier systems such as CVS and Subversion
  - More efficient, better workflow, etc.
  - See the literature for an extensive list of reasons
  - Of course, there are always those who disagree
- Best competitor: Mercurial
  - Same concepts, slightly simpler to use
  - Much less popular than Git

Download and install Git

- Here's the standard one: http://git-scm.com/downloads
- Note: Git is primarily a command-line tool
- But it can be used with GUI.

Introduce yourself to Git

- Enter these lines (with appropriate changes):
  - `git config --global user.name "Alper Uysal"`
  - `git config --global user.email alper.uysal@alanya.edu.tr`
- You only need to do this once

- If you want to use a different name/email address for a particular project, you can change it for just that project
  - `cd` to the project directory
  - Use the above commands, but leave out the `--global`

Create and fill a repository

1. `cd` to the project directory you want to use
2. Type in `git init`
   - This creates the repository (a directory named `.git`)
   - You seldom (if ever) need to look inside this directory
3. Type in `git add .`
   - The period at the end is part of this command!
     - Period means “this directory”
   - This adds all your current files to the repository
4. Type in `git commit -m "Initial commit"`
   - You can use a different commit message, if you like

Clone a repository from elsewhere

- `git clone URL`
- `git clone URL mypath`
  - These make an exact copy of the repository at the given URL
- `git clone git://github.com/rest_of_path/file.git`
  - Github is the most popular (free) public repository
- All repositories are equal
  - But you can treat some particular repository (such as one on Github) as the “master” directory
- Typically, each team member works in his/her own repository, and “merges” with other repositories as appropriate

The repository

- Your top-level **working directory** contains everything about your project
  - The working directory probably contains many subdirectories—source code, binaries, documentation, data files, etc.
  - One of these subdirectories, named .git, is your repository
- At any time, you can take a "snapshot" of everything (or selected things) in your project directory, and put it in your repository
  - This "snapshot" is called a commit object
  - The commit object contains (1) a set of files, (2) references to the "parents" of the commit object, and (3) a unique "SHA1" name

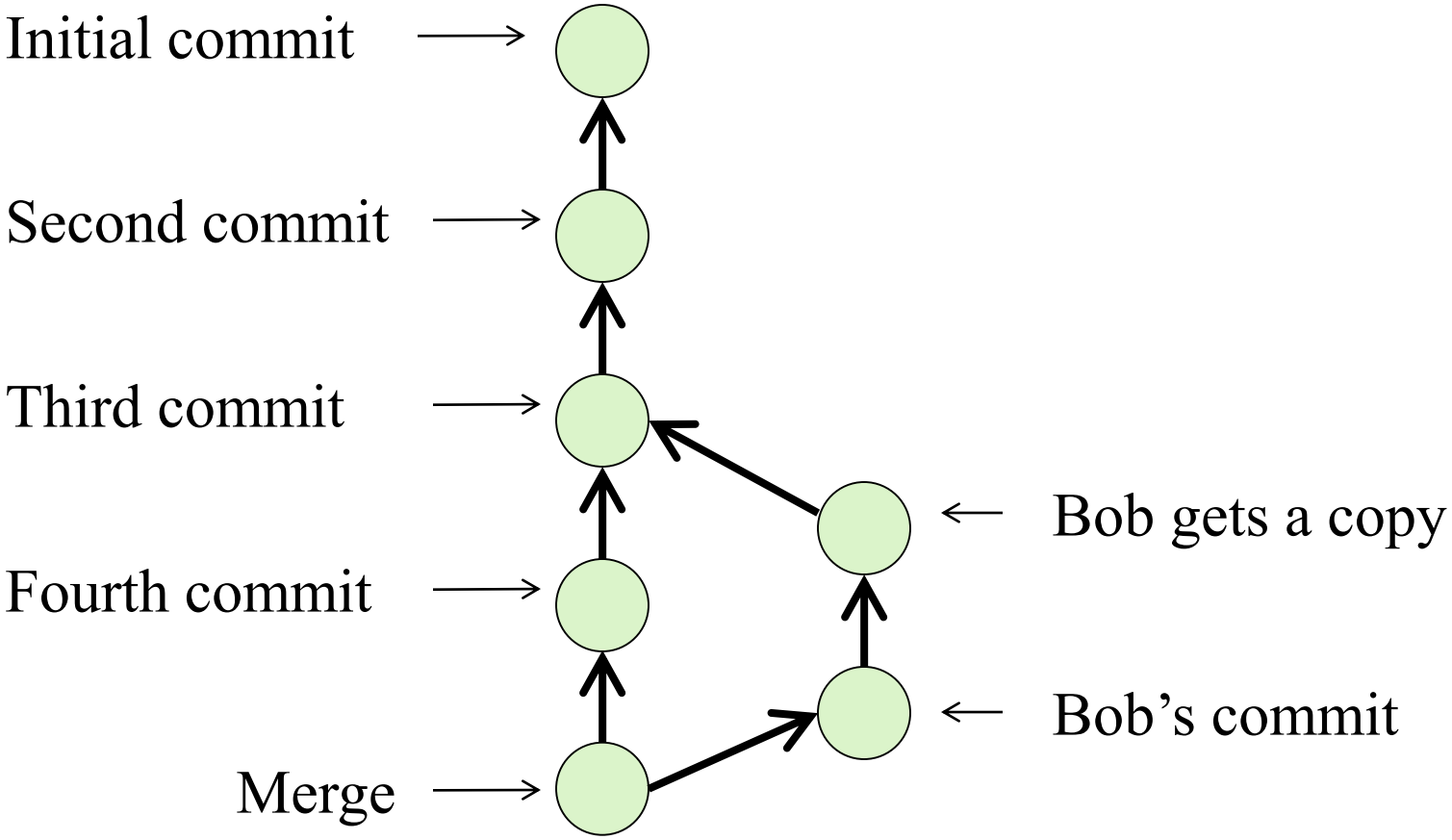
init and the .git repository

- When you said `git init` in your project directory, or when you cloned an existing project, you created a repository
  - The repository is a subdirectory named .git containing various files
  - The dot indicates a “hidden” directory
  - You do *not* work directly with the contents of that directory; various git commands do that for you
  - You *do* need a basic understanding of what is in the repository

Making commits

- You do your work in your project directory, as usual
- If you create new files and/or folders, they are *not tracked* by Git unless you ask it to do so
  - `git add newFile1 newFolder1 newFolder2 newFile2`
- Committing makes a “snapshot” of everything being tracked into your repository
  - A message telling what you have done is required
  - `git commit –m “Uncrevulated the conundrum bar”`
  - `git commit`
    - This version opens an editor for you the enter the message
    - To finish, save and quit the editor
- Format of the commit message
  - One line containing the complete summary
  - If more than one line, the second line must be blank

Multiple versions



Github with Pycharm

- PyCharm lets you manage Git projects hosted on GitHub directly from the IDE: clone repositories, **share your projects**, create forks, share code through gists, create pull requests and review incoming pull requests.

- https://www.jetbrains.com/help/pycharm/github.html

References

- https://www.cis.upenn.edu/~matuszek/cit591-2012/Lectures/git.ppt