Git & GitHub

Introduction
Git

- Git is a popular DVCS
- Initially, written by Linus Torvalds
- Becoming the industry standard
- Open-source
Git

- Different than SVN (and similar systems)
  - More complex and powerful

- No need to be overwhelmed
  - Get the basic concepts
  - Learn new features as new needs arise
  - Don’t get scared by terminology

Let’s start with the basics ...
Git Commits = Snapshots

- Need to think differently about git
  - Forget about revisions to individual files

- Each commit is a snapshot of the full directory tree
  - That’s the abstraction. Under the hood, Git stores differences (to optimize space usage)

- Git repo is a graph of commits
  - A version of the code is a node in the graph
  - History is described by paths in the graph
Each *commit* represents a version of the code.
A path of commits represents its history.
A new commit (i.e. snapshot) is created and added to the graph.

Commit changes to the repo …
Working Locally

1. Working directory
   The actual files on your machine.

2. Staging Area (aka index)
   Intermediate storage for code changes.

3. Repository (aka history)
   The graph of commits.
Local Operations

- working directory
- staging area
- git directory (repository)

1. Checkout the project
2. Stage files
3. Commit
Why the extra step?

- The extra step (i.e. *staging*, before committing) gives us more granularity
  - Choose which changes you want to commit
    - Ex: Do not commit temporary changes made only for the purpose of local testing
  - When saving changes, can break them into multiple commits.
    - Each commit with its own concise, meaningful message.
  - Goal: Work in a traceable manner
Demo Time

Let’s use some basic Git commands ...

- init
- status
- add
- commit
- log
Think Outside Of The (One) Box

- The demo we just saw is local.
  - That is, it was done entirely on my machine

- A more common scenario involves remote repos:
  - Clone some remote repo to your machine
  - Commit changes locally
  - When ready, push changes from your machine to the remote repo

Q: Where do we store remote repositories?
GitHub

- GitHub is a hosting service for Git repos
- Website, social layer and a rich toolset on top of Git
- Free for public projects
- Industry standard for OSS development
Demo Time

Let’s see a few more basic Git commands ...

- clone
- push
- pull
Commit ≠ Push

- It is important to understand the distinction between commit and push
  - `git commit` creates a node in the commit graph of your local repo
  - `git push` creates node(s) in the commit graphs of some remote repo

- Allows for more granularity
  - Make small frequent commits while working locally on your code
  - When your work is ready (and tested) push all commits to a remote repo
GitHub, Fork

• Forking = Cloning directly on GitHub
  ○ No need to clone anything to your local machine
  ○ The fork is a separate GitHub repo, associated with your GitHub account (i.e. You can read/write to it)

• More than just a clone
  ○ The fork inherits access permissions
  ○ Forks create an “implicit social layer”
  ○ And more ...
How to do real work?

- How do you contribute work to a repo you have no write permission for?
  - Create a pull request, and let someone who has write permission merge.
    - This is how open-source software works.
    - This is how you will submit your individual coding assignments in this course.
Common Workflow

Public repo, hosted on GitHub.
Readable, but not writable, by you.
How can you contribute changes?

The fork, also on github, is readable and writable by you.

However, you can’t run/test your java code up on GitHub...
Common Workflow

Create a local clone, so you can use your favorite IDE (e.g. Eclipse), run the code to test your changes, etc.
Commit some changes (locally) ...
Common Workflow

Push the changes from your local repo to your GitHub-hosted fork …
Common Workflow

Create a *pull-request* …
One of the original project’s maintainers merges your pull-request.
GitHub - Pull Request

- Request project maintainer(s) to pull changes from your fork into their repo.
  - An easy way to follow the Linux project’s workflow (emailing patch files)
  - Discussion is part of the pull-request
  - Automatically warn about conflicts
  - Can merge pull-requests directly from GitHub
GitHub - Pull Request

- GitHub didn’t invent pull-requests
  - Git has built-in support
  - GitHub just simplified the process and added convenient web UI on top of it
  - Not everybody likes this simplification

- For CSC301, the simplification works great.
  - You will submit your homework by submitting pull-requests
Demo Time

Common workflow

1. **Fork** a repo on GitHub
2. Make a local **clone** of the fork
3. **Commit** changes locally
4. **Push** them to the fork
5. Submit a **pull request** (form the fork to the original repo we forked from)
Resources

Many great resources for learning Git and GitHub

- A Simple Guide
- Training page on GitHub and BitBucket
- An interactive tutorial
- Pro Git - A whole book on Git
- Git for Computer Scientists
That’s it for today

Don’t forget to join the course GitHub organization by Wednesday at 10 pm