Git Good: Course Overview

Sam Kagan

The course will be divided into 3-4 sections:

1. A theoretical framework for understanding Git
2. How one can interpret everyday Git activities in light of that framework
3. How one can best approach and optimize Git activities that aren't everyday but aren't all that rare either
4. (optional) Tools that can make Git easier to use than vanilla bash, and how to effectively use Git from IntelliJ

The first two sections will be interleaved so that the theory can be more easily understood through familiar examples.

1 Structure of Each Lesson

The plan for the first part of the course (sections 1 and 2 above) is **hour-long lessons divided evenly between lecture and lab.**

There are a couple of ideas for what *lab* might mean: 1. Like a lab in school, where assignments are done in-class after lecture so I can answer questions as they arise. 1. More like a traditional lecture class, where assignments are done outside of class and questions can be raised in the first half of class.

2 Course Materials

2.1 Primary Texts

- Think Like a Git
  - Great for how to think about branching
  - Repository as a graph
- Git from the Bottom Up
  - Great for how to think about commits & the three states
  - Repository as an append-only filesystem
- Pro Git (or The Git Book)
  - Great for how to think about commits & the three states
  - Official documentation

2.2 Supplemental Texts

- Some texts to help understand (directed, acyclic, unweighted) **graphs** in the CS/graph-theory sense at a conceptual level:
    - Graphs in Computer Science *Portland State University* (only through *Kinds of Graphs*)
    - Explained: Graphs *MIT News* (only first couple paragraphs)
    - Graph (discrete mathematics) *Wikipedia* (for the algebraically-minded)
- Some texts to help understand **trees** in the CS/graph-theory sense at a conceptual level:
    - Tree (data structure) *Wikipedia* (only *Intro*, *Terminology*, *Examples of trees and non-trees*, and *Mathematical Terminology*)

2.3 Software

- Git
- A terminal environment (where you'll run Git for all exercises)

3 Lesson Plan

Now, I'll break each section outlined in the intro. into lessons.

3.1 Understanding Git

3.1.1 Intro / What is a Repository?

Lecture

- The three states
    - Working tree/working directory
        * Your local directory (commonly referred to as "the repository")
    - Index/staging area
        * The middleman
        * Discuss more later once we understand commits better
    - Repository
        * An append-only filesystem of the working tree's history
        * A graph of the working tree's history

Lab/Homework

- Discuss how to structure lab portion of each meeting
- Discuss how often we want to meet
- Individually:
    - Create repository for class
    - View the three states via `status` and `log`
        * Make a new file

     * Add it to the index
     * Commit it
   – View graph of history via `log`
     * Make another commit
     * Make a *branch* from the first commit, and commit there
     * Merge the branch back into the starting branch

3.1.2 What is a Commit?

Lecture

- Commits are the fundamental unit of a repository
  - Nodes in the graph
  - Snapshots of history
  - Pointers to filesystem trees plus references to parent commits
    * Trees are like your working tree, but made of blobs instead of files
    * Blobs are just content, no metadata (i.e. mtime, ctime, name, permissions)
      · Trees own the metadata because different trees may associate different metadata with different blobs
    * Trees give you content addressibility

Lab/Homework

3.1.3 What is the Index?

Lecture

- The index lets you control which changes on your working tree go into your next commit

Lab/Homework

3.1.4 What is a Branch?

Lecture

- Branches are just references to commits.
  - They keep up with your work, unlike tags.

Lab/Homework

3.2   Applying Your Understanding of Git to Common Problems

3.3   (Optional) Configuring your Environment to make Git Easier to Use