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 The Purpose of This Course

The purpose of this course is to help you understand and more effectively use one of the less well-understood tools that the software engineers here use every day. ## Who This Course Is Intended For As implied above, this course is intended for software engineers who use Git in their daily lives. I will assume some familiarity with the mechanics of basic Git usage (i.e. how to turn your changes into a commit, how to interact with a remote Git server, how to create and merge branches). I will *not* assume that anyone taking this course has much of an understanding of these mechanics, or a very good internal model of what they actually do. I will also assume some familiarity with **graphs** and **trees** as they're defined in graph theory or computer science.

For people who aren't familiar with Git or graphs and trees, it is my hope that

you get something out of this course. If this relative lack of experience becomes a barrier to understanding, please let me know so that I can arrange additional materials for you.

3 Course Materials

3.1 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

3.2 Supplemental Texts

- The Thing About Git
  - Explains the Staging Area's utility through a practical example
- Oh Shit, Git!?!
  - Solutions to common Git problems

3.3 Software

- Git
- A terminal environment (where you'll run Git for all exercises)
- The `git-graph` visualizer, which we will use to view repositories and commits as graphs
  - Clone my fork here, it's just a Python script
- The Git TUI `gitui`, which we will use to visualize the staging area
  - Installation instructions can be found here

4 Lesson Plan

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

4.1 Understanding Git

4.1.1 Intro / What is a Repository?

Lecture

- What is this course?

  - Go over the Course Overview (this document)
- The three states
  - Working tree/working directory
    - Your local directory (commonly referred to as "the repository")
  - Staging Area
    - The middleman
    - Also known as the "index"
    - Discuss more later once we understand commits better
  - Repository
    - A graph representing an append-only filesystem 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 `gitui`
    - Make a new file
    - Add it to the staging area
    - Commit it
  - View graph of history via `git-graph`
    - Make another commit
    - Make a *branch* from the first commit, and commit there
    - Merge the branch back into the starting branch

4.1.2 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

4.1.3 What is the Staging Area?

Lecture

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

Lab/Homework

4.1.4 What is a Branch?

Lecture

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

Lab/Homework

4.2 Applying Your Understanding of Git to Common Problems

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