PUSHING GIT & GITHUB IN UNDERGRADUATE COMPUTER SCIENCE CLASSES*

Esmail Bonakdarian
Computer Science Department
Dominican University
River Forest, IL 60305
ebonakdarian@dom.edu

ABSTRACT

The ability to use version control is a valuable skill for computer science graduates to possess. This paper describes our experience introducing undergraduate computer science students to the widely used distributed version control system Git in conjunction with GitHub as part of an existing class. The class need not to be built around Git/GitHub; it is possible to introduce each of our four instructional components in sequence at any time they fit. Placing them earlier in the course schedule will give students the benefit of being able to use these tools throughout their class. We choose to focus on the use of the Git command line since this provided a consistent interface across all major platforms and therefore yielded the best return for time invested learning the software, while also offering greater flexibility and power to the user. A few of our students had some previous experience working with the Windows command line, while others were new to it, far fewer knew about version control. We describe the structure of our approach along with some of the instructional resources made available to the class which we believe provided an effective and flexible way to introduce students to these tools. We conclude by outlining successes and challenges encountered with our methods, what we've learned, and what might change for the next iteration of this approach.

INTRODUCTION

Git is one of the most widely used version control systems in the software industry; Google, Microsoft, Apple, Facebook count among its many users. It provides a valuable in-demand skill for our students to possess by the time they graduate or apply for internships. It has permeated software development to such a degree that most major IDEs (e.g., Visual Studio, Eclipse, NetBeans) now offer support for Git/GitHub integration. It is also useful to students still in school as they can benefit from version control while working on programming assignments. A number of faculty using version control with Git in the classroom have shared their experiences [3, 5], for instance as a course platform for distributing and collecting exercises from students [4]. We chose to use Git with GitHub, which facilitates team-based software development efforts, including those that are geographically dispersed and occur asynchronously over time. Using GitHub has the added benefit of freeing us from having to set up and maintain our own server. Many of our current students are not familiar with these important types of tools, as they are not ordinarily covered in our curriculum. To make this a more realistic and helpful introduction, we selected a class that had a semester-long team project as one of its core components and where the sustained software development effort could benefit from both Git and GitHub's use.

Background

The class selected for evaluating our approach was CPSC 434 Introduction to Unix class, a popular elective. Most of our 17 students were second-semester sophomores, and few (12%) had prior familiarity with version control systems. Teams of 4-5 students worked on a semester-long software development project using a number of Raspberry Pis running Raspian (a hardware-specific Linux distribution), requiring students to become comfortable with the Linux environment and the bash shell.

Goals

One of our goals was to expose students to the concepts of version control in the context of software developed as a team, while using Git and GitHub to demonstrate the specific and practical benefits of these tools during the semester-long group project. This is in contrast to the individual programs students are usually assigned in our other classes, aside from the final two-semester capstone sequence during their senior year. Additionally, as many open source projects use both Git and GitHub, these skills enable our students to contribute to such projects in the future, something we would like to encourage since it offers additional software development experience. Finally, more and more employers ask job candidates for examples of their work, and having a GitHub repository can serve as an effective showcase for the student's ability to write, document and organize code.

METHODOLOGY

Our approach consisted of a sequence of four incremental stages; each depended on the one preceding it (see Table 1). The goal of stage 1 was to get all students comfortable working in a command line environment. The second and third stages introduced Git *for local use only* (setting up repositories, staging, committing files, etc.). In the fourth stage, and only after students had become sufficiently comfortable with the basic operations and concepts, assessed via quizzes and in-class exercises, did we introduce GitHub which added another, shared/networked layer above the local repository. We describe each of these in detail in the subsections below.

Stage 1 – CLI	**Stage 2 – Git**	**Stage 3 – Git**	**Stage 4 - GitHub**
Command Line	*Local Repository*		*Remote (GitHub) Repository*
Basic file and directory manipulation, command line arguments, running program under Windows and the bash shell	git init, config, status, log, diff, git add, commit, unstaging, resetting, .gitignore (local and global) setting global preferences, e.g., id, editor, etc.	Creating git aliases Managing branches: creating, checking out, deleting branches, merging branches. Moving, deleting files in Git, HEAD, tags, topic branches, workflow	Cloning remote repositories, pushing, pulling, setting up remotes, fetching, seeing remote branches, forking, issues, MD markup, GitHub navigation

Table 1*- Examples of Concepts, Topics and Commands Covered*

Our classroom was equipped with networked PCs running Windows 7 for every student, and an instructor station able to project onto student screens and onto a screen at the front of the room. A shared Linux system, a virtual machine running Ubuntu 14.04 LTS - configured and administered by the instructor and procured via a Microsoft Azure educational grant [6] - was available for class use. Students logged into this system using putty or ssh through the Git bash shell to practice using Git in a Linux environment. Git version 2 was used on all platforms.

The Command Line

To bring all students up to speed with the command line we did a quick review (or introduction for some) of the Windows command line along with basic file and directory operations and use of wildcards. Our department is solidly Windows-based, so all of our students are familiar with the operating system and its file system. They could observe and compare the effects of command line actions via Window's Explorer GUI interface. At the same time, equivalent Linux command names were also introduced (e.g., ls in place of dir) and distributed for later use.

Even before we had students sign on to a Linux system for the first time, we used the Git bash shell to familiarize them with the Linux commands for manipulating files, directories and running programs. Since this was initially done on a Windows platform,

they could again verify their operations through the Windows File Explorer. This also greatly eased their subsequent transition to the Linux environment. The availability of the Git bash shell enabled students to learn about bash without even needing a Linux system, making this a very cost effective and logistically easy introduction to this environment.

Introduction to Git

Git is a free open source distributed version control system available for all major personal computing platforms. One of its appealing features is that it can store all files and history locally, eliminating the need for a central sever. Many of its other benefits are enumerated in [5]. It has a somewhat steep learning curve, and a few of its operations can initially look quite complex and intimidating, especially undergraduate students new to version control and the command line. To minimize student overload with new environments and concepts, the initial introduction and exercises to Git and GitHub occurred under Windows. While graphical interfaces for Git are available, we deliberately chose the command line interface because of the greater control and power it offers, and because it is consistent between different platforms. Kelleher reports that the GUI interface was not well received by his students [3]. The fact that most of our students hadn't used a version control system before was seen as a benefit, as this meant they had no preconceived notions about how it should work. Git's operation is different from other popular systems such as subversions, so new users often have to "unlearn" certain habits and mental models to use Git effectively.

After students worked through Windows based exercises for basic command line operations and Git, they were asked to repeat the same under bash and eventually our shared Linux system, and all further instruction took place under Linux (either on the shared Linux system, or on their own personal Raspberry Pis).

To ease the transition, plenty of free tutorial materials (both text-based and videos) were made available on the course webpages, along with required readings from the free online Pro Git text (chapters 1-3) [1]. The large number of available tutorials online attests to the popularity of this platform, and perhaps also to the need for additional help, beyond the standard documentation, to understand how to effectively use these tools. These supplemental materials supported class lectures introducing the basic concepts of version control and Git in particular, followed by a series of guided lab exercises to help solidify understanding of the material.

Git Exercises

After an initial lecture on Git, all students were required to work through the interactive "Got 15 minutes and want to learn Git?" tutorial [7]. To help students learn Git and GitHub in a systematic fashion, several instructor-generated guided lab exercises were provided. These were started during class after a suitable introduction and demonstration of the material, and were to be completed outside of class. For two of three exercises, students worked with their own local repository. The third exercise covering GitHub was a team exercise and is described in the section below.

The first Git exercise introduced the conceptual model and workflow for Git by covering basics such as the working directory, the staging area and the local repository. Students were given instructions about how to initialize their local repository and configure necessary options such as user identities, editor preferences, and global git-ignore files. A series of tasks with existing sample files and directories provided practice for modifying, adding and committing files; reviewing the status, log, and history of the repository, and comparing different versions of files.

The focus of the second guided Git exercise was branching, an essential feature of version control. Students learned how to create and manage branches, including switching to topic branches, merging, and successfully resolving possible conflicts during merging, and eventually how to delete no longer needed branches. These topics also allowed us to introduce some of the internals of Git, such as the HEAD pointer. With an understanding of branching, we were able to describe a few common workflow and branching models for teams.

GitHub

We decided to pair up Git with GitHub, a major software collaboration site boasting over 14 million registered users and 35 million software repositories [2], allowing for social computing. By incorporating Git and GitHub in a class that had a team-based semester-long software project, we hoped that students would realize the real tangible benefits this technology could offer.

We set up an organization for the class on GitHub and four project teams within it. To sidestep any potential FERPA issues, free private repositories were requested from GitHub via its Education site, https://education.github.com. Students used their university e-mail addresses to sign up for GitHub accounts, which greatly simplified invitations to their respective teams. The GitHub repositories made it easy to monitor the progress of the group overall and the contributions of individual team members. A separate set of four private training repositories on GitHub for practice and instructional purposes were used for the introductory exercises (not graded), designed to sufficiently familiarize students with basic GitHub operations. A separate private repository within the organization was created for instructor use to house Git and GitHub related exercises and files for distribution to students.

In the third guided exercise introducing GitHub students worked in project teams. They copied a training repository on GitHub by cloning it and each student created a separate branch to fix bugs in the sample source files. They merged their topic branches back to the master branch and resolved any resulting conflicts, more likely to occur when working as a team. Finally, they pushed their local changes to their shared training repository on GitHub and pulled updates. We also explored the GitHub pages and the various activity and status reports they provided.

Workflow

After discussing a number of different workflow scenarios for Git and GitHub, students were given a set of concrete best practices and a commonly used workflow was

used for the actual project. A "production" quality master branch would be maintained by the team. Any development done by team members would be on topic branches, where new features or bug fixes would be implemented and tested, and then merged back to the master branch. This enabled parallel development while maintaining a stable core version of the software at all times. Students were encouraged to commit early and frequently.

RESULTS

Our anonymous surveys of students indicate that our approach was successful in providing an effective introduction of Git and GitHub via the command line. Our students (initially 17, then 16) were surveyed with a 5-point Likert scale prior to introducing the new material and at the end of the semester to help assess a given subject.

Response	Prior	Post
1 – No Confidence	24%	6%
2	35%	0%
3 – Moderate Confidence	35%	13%
4	6%	56%
5 – A lot of Confidence	0%	25%

***Table 2** - How confident are you using Git/GitHub?*

Response	Prior	Post
1 – No	0%	0%
2	0%	0%
3 – Maybe	12%	6%
4	24%	25%
5 – Yes	65%	69%

***Table 3** - Do you think knowing the basics of Git/GitHub will be helpful for your future work (in school/career)?*

Table 2 shows the level of confidence students gained with Git and GitHub by the end of the semester, and a significant shift can be observed. One student, for reasons known only to them, never used Git and GitHub regularly, which explains the 6% "No Confidence" outlier at the Post level – which represents a single student.

Table 3 indicates whether students consider Git and GitHub potentially useful in their future work (in school or their career) and can be interpreted as a marker if students see value in learning about these topics. It turns out that students were already excited to learn about Git and GitHub, and their initial enthusiasm that this was time spent usefully increased once they learned more about these tools.

CONCLUSIONS

Our students finished the course with a solid understanding of the command line, and what version control systems can do. Using Git and GitHub for the bulk of the semester left them with much more confidence and a deeper and practical level of knowledge, something we feel is essential for anyone planning to work in the software industry. This knowledge was earned through the authentic use of these tools in a semester-long team development effort.

We believe that having students first practice using Git locally on their own machines in Stages 2 and 3 eliminates a potential source of confusion if others are involved working on the same shared repository.

Our two-semester capstone sequence requires students to develop software as a team and in the past has relied on simple DIY approaches (using shared storage, such as Dropbox or Microsoft's OneDrive) without any systematic naming or recovery features. It would be interesting to know if students coming out of this class would use Git/GitHub for the capstone. We are equally curious to find out if any of our students will later contribute to any open source software projects.

Future work involves creating new self-contained lab exercises (including pull request, covered this term, but not practiced), perhaps exposure to some GUI tools, and possible use in CS I.

One encouraging sign: Even though our class just ended, a group of students from this course have already set up a GitHub project of their own and are using Git to manage the development of a game under the Windows platform, attesting that these tools are largely platform agnostic.

Our experience shows that it is possible to effectively introduce students to these somewhat complex, in-demand tools for the software industry as part of an existing class and thereby equip them with valuable skills that will serve them well beyond their time in our classrooms.

REFERENCES

[1] Chacon, S., Straub, B., Pro Git, 2nd ed, https://git-scm.com/book/en/v2, retrieved January 4, 2016

[2] GitHub, https://github.com/about/press, retrieved April 29, 2016

[3] Kelleher, J., Employing git in the classroom, *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, pp. 1-4.

[4] Haaranen, L., Lehtinen, T., Teaching Git on the Side: Version Control System as a Course Platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE '15), pp. 87-92.

[5] Lawrance, J., Jung, S., Wiseman, C., Git on the cloud in the classroom. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13), pp. 639-644.

[6] Microsoft Azure in education, https://azure.microsoft.com/en-us/community/education/, retrieved April 20, 2016

[7] https://try.github.io/levels/1/challenges/1, retrieved March 10, 2016