Employing Git in the Classroom

John Kelleher
Dept. Computing & Creative Design
School of Engineering & Design,
Institute of Technology,
Sligo, Rep. Ireland
kelleher.john@itsligo.ie

Abstract - ***Git is a well established, well received source version control system for the software development community and beyond. In an effort to better expose my students to the purpose and process of version control, I introduced Git as a mechanism to both disseminate in-class exercises and worked solutions as well as facilitate the submission of continuous assessment. I document in this paper my experience of the experiment and describe the steps required to marshall free third party services to support the tasks undertaken by students and lecturer.***

Keywords: course management, version control, learning management system

I. INTRODUCTION

Source version control has become a key industry tool to manage increasingly asynchronous and distributed software development efforts. Industry demand for those experienced with Version Control Systems (VCS) has increased significantly in recent years[1] reflecting a growing intolerance for those less experienced with such industry standard tooling. For my students who undertake internships (c. 4-6 months) at end of Year 3, this limits their ability to fully engage in the software development role of their placement.

Before their internship, the students work in teams of 4 to complete a significant software project from conception to deployment. As software development tools improve to accommodate disparate elements (e.g. design, database, testing, client/server, user interface), the use of VCS is increasingly mandated to enable all team members to work together effectively towards project milestones. This need is further underscored by the progression of online study leading to a more geographically disparate student cohort.

Additionally, within the classroom, there is increasing pressure on contact time with students prompting a refocus on more efficient means of communication from lecturer to student - and from student to lecturer. In parallel, a move towards the 'flipped model'[2] of student learning benefits from a quick, efficient mechanism to round-trip student and lecturer exercises back and forth. Traditional set exercises delivered in-class or as homework were often unidirectional. Though students could individual engage with an exercise in their own time, the methods available to begin a conversation with the lecturer through the medium of the code were fragmented and ultimately served only the individual student.

Selection Criteria	**Online Service**		
	Github	***Bitbucket***	***Team Foundation Service***
Pricing model	Students: 5 free private repositories, unlimited collaborators Lecturer: unlimited private repositories	Students: unlimited private repositories, max. 5 collaborators Lecturer: unlimited user account	free, private repositories, max 5 users, no distinction student/lecturer
Usability/ Integration	Rich Git standalone tooling	Rich Git standalone tooling	Git tooling integrated with Visual Studio but problematic in operation (beta)
Industry Prevalence	Class leader	Not as popular as Github but very similar in operation	Limited industry uptake and favoured Microsoft TFS (Team Foundation Server) support over Git repositories

Table 1- Selection of online hosting

II. IDENTIFICATION OF NEED

The message from industry identifying the need for graduates with VCS expertise was clear[1]. Anecdotal reporting from students seeking internships and in post-placement interviewing confirmed the deficit. Orthogonally, the need to drive better engagement with and by students demanded a richer channel of communication[6]. On a practical level, the business of distributing in-class worked exercises offered an opportunity to involve VCS. VCS supported the class scenario whereby an exercise begun in class by the lecturer could be branched from the core branch to expound upon some spontaneously identified feature. Both (or multiple) branches of code would be neatly packaged, documented and delivered to students.

978-1-4799-3351-8/14/$31.00 ©2014 IEEE

III. GIT MARKETPLACE OFFERINGS

The success of VCS in the software industry (and in particular the popularity of distributed *version* control system (DCVS) over centralised systems) has led to the development of a competitive market for cloud-based hosting services. These services often offer free educational provision and present an attractive alternative to the chore of funding, deploying and maintaining an in-house version control server. Github[1] (owned by Github Inc.), Bitbucket[2] (owned by Atlassian) and Team Foundation Service[3] (owned by Microsoft were examined as candidates for the experiment. A simple analysis matrix led to Github as the chosen cloud platform.

All options offered private repositories. However, Github and Bitbucket provided very clear mechanisms to administer access to repositories. Crucially, all services offered unlimited storage thereby easing the administrative burden of local storage.

IV. DEPLOYING GIT

The target audiences for Git were second and third year CS undergraduate students. Both groups were familiar with Visual Studio from Microsoft. A recently released extension for Visual Studio - Visual Studio Tools for Windows Git Extension - provided a limited feature GUI for manipulating the Git repositories. All students signed up for a free Github user account. Github offer a free educational account, termed an 'organisation', which provided unlimited private repositories for the lecturer. This account could invite unlimited collaborators (students). Thus all repositories could be enabled as private and specific groups of students or individuals could have access granted or revoked readily. For example, I may grant access to one group of students who may be 'ahead' of another while not revealing anything prematurely to another.

Students had no prior knowledge of Git, many never having heard of VCS. Though the third year students had a more demanding schedule, the second year students were already encountering many new concepts without the benefit of extensive past experience. Hence Git was minimally deployed. The third year students were more experienced and their appreciation of the subject matter's relevance to their careers was evident (Table 2).

Extent of Teaching	***Year 2 Students***	***Year 3 Students***
Tools used	GUI Extension only	Git extension & Git command shell
Purpose	Cloning of repositories presented in class	Cloning of repositories presented in class; Creating own repositories; Checking in changes, committing and pushing to cloud hosting; Creating branches and checking out branches;
Tuition Offered	4 hours over 2 weeks	8 hours over 2 weeks
Adoption	>80% used Git to clone exercises	>90% used Git for all above purposes

Table 2 - Involvement of students

Git[4] (version 1.8) was deployed to each Windows PC in the laboratory. The Visual Studio Tools for Windows Git Extension was installed to Visual Studio 2012 - this extension is now integrated in Visual Studio 2013. Neither installation requires administrator access and can be performed by students.

Though several GUI Git applications were examined and one tested with students, it was found that, for basic operations, the VS2012 Git Extension presented the clearest model of operation. A number of online Git teaching resources were referenced[3] for students in addition to the allocated in-class teaching. I recorded a video[5] to illustrate how to raise issues regarding a repository so that problems (and their resolution) could be communicated more comprehensively throughout the class.

Initial teaching encouraged students to clone exercises developed in-class. This required the creation of a Github account and so 100% self sign-up was achieved promptly. Later, these Github user account addresses were solicited for the purpose of assignment submission.

The basic workflow for employing Git for continuous assessment was as follows[5]:

- Assessment issued via class web site (or email) with (where necessary) an accompanying Github repository link to house starter code (partial solution)
- If assessment has starter code, then student clones this to local repository (this can be anywhere)
- Student works on code, committing regularly to his/her local repository
- When time to submit, student pushes completed solution to individually assigned private repository (typically named after Student ID for clarify/ uniqueness)

Initial Process

For Github, the lecturer acquires in the first instance a standard user account. From here an organisation account can be created which will act as the store for class-based repositories. Github can assign this organisation account educational status which offers an initial 125 free private repositories[7].

At the same time, students register a new user account with their own credentials, ideally their student ID. The lecturer creates a number of Teams into which are added the IDs of those students belonging to that team. In this way, different groups of students can be assigned access to the same repository independently. Any combination of pull and push

[1] http://git-scm.com/
[2] www.bitbucket.org
[3] http://www.visualstudio.com/products/visual-studio-online-overview-vs

[5] http://youtu.be/xGYj1bAqg0A

(read and write) privileges can be assigned at a group level. Indeed, individual students can belong to more than one group and have different privileges on each. It is trivial to grant or revoke access to a Team. However, specific branches cannot be assigned different access privileges.

For in-class exercises, work can be pushed to a repository created in a particular organisation. A team of students is then assigned access. Publicising the availability of a repository is convenient as Github will show the repository as public to the team member.

Organising assignments

1. If starter code is applicable to the assignment, the lecturer posts this to the organisation account and grants access to the appropriate team. The student clones this starter project:

   ```
   git clone https://github.com/itsligo/
   ca_1.git
   ```

   This creates a local copy of the cloned project. If, however, there is no starter code and the student begins with a self-created project, then all that is needed is to add the new solution to Git source control, whilst in the root folder of the solution:

   ```
   git init
   ```

2. Before beginning work, the student creates a distinct branch of the current project named CA1, CA2 or whatever reflects the number of the assessment. This new branch (e.g. CA1) will be the basis for the work of the assessment. It also protects the original provided code should the student wish to revert to it.

   ```
   git checkout -bf ca1
   ```

   Note -f (optionally) discards any changes made to the code (since the last commit) and -b (optionally) switches to the new branch (ca1).

3. The student maintains a single repository for her assessment work. Ideally, this private repository is created by the lecturer and the student added with full privileges as a collaborator to the repository. This ensures that other students cannot be granted access to, and that lecturer access cannot be revoked from, the repository. Also, under the Github educational licence the number of private repositories is initially generous (over 125) and can be extended.

   ```
   git remote add ca1 https://
   s01234567@github.com/s01234567.git
   ```

   This adds a remote branch to the student's Github private repository so that she can regularly push the local repository to the cloud hosting.

4. The student works on the assignment, modifying as necessary to complete the work. She regularly commits changes made so that milestones are captured by the Git repository. It is important that she shows regular commits to underwrite the fact that the assessment is being regularly worked on. A single large commit will draw attention and prompt an interview to establish provenance.

   ```
   git add .

   git commit -m 'Added key feature as per
   part 2 of assessment'

   git push -u ca1 ca1
   ```

   The first line adds all changes to local repository. The next line commits those changes to the local repository. The final line pushes this commit to the remote repository previously associated with the local repository. Future pushes need only say 'git push' since the -u switch specifies the s01234567 private remote repository as the default destination for the push. Also, the mention of 'ca1 ca1' is redundant so long as the student remains with the ca1 branch.

5. To complete the submission, the student sends an email with link to the repository to the lecturer. As the same repository can be used to house (in separate branches) all continuous assessment, the repository address can be kept for use with future submissions.

Problems Encountered

The GUI tools held much promise but were not well received. From feedback from students, it appeared that though visually attractive, the range of features served to their confound their nascent understanding[8].

The Visual Studio extension proved useful to access basic functionality - creating and cloning a repository, adding and committing changes, pushing and pulling to a remote repository. However, it proved buggy and task flow, particularly around branching, led to confusion. Visual Studio also, requires that two explicit settings be made to switch from Microsoft's Team Foundation Service VCS to Git and to enable the extension post-installation.

Surprisingly, the Bash Git command shell gained in popularity as it allowed students to engage with Git to the level of their ability at a particular stage. It also reassured students that, as their understanding grew, they could retain the same tool though-out.

Though Github is designed to avoid issues with firewalls (using http/https to do so), there were complications initially successfully reaching remote repositories. This required identifying the local proxy server to Git. A more complete initialisation of Git required setting of user name, user email and Github username and password caching (to avoid manual entry on pushing to cloud). However, some of these settings were possible within Visual Studio.

Github worked well for both student and lecturer. However, the mechanism of granting private repositories for education purposes - using Github Organisations - was at times restrictive. One unfortunate limitations was that individual user accounts could not be granted access to the repositories

contained in an organisation. Were this possible, I would have created a single private repository for each student. As owner of that repository, I would invite each student as a read-write collaborator. This would ensure that only the student and lecturer had access to the repository. Though this would require some initial setup on behalf of the lecturer, it is a one-off operation as each repository could hold many branches (for separate submissions).

By allowing the student create the repository, it is possible for other individuals to be added by the student as collaborators who could contribute to the code base. However, this third-party contribution would be revealed in the commit history. The student could also choose to deny the lecturer access to the repository at any time, though forks made of that repository would continue to persist.

One solution was to create a Team for each student, adding only a single student to each Team. In this case, the lecturer again creates a private repository in the Github organisation for each student. Adding the individual 'student' Team as collaborator to that repository cements control of the contributions and associated code with the lecturer.

A more practical solution was to use Bitbucket to hold the individual private student repositories. Ideally, these would be created by the lecturer so that access control rests exclusively with the lecturer.

V. Derived benefits

Overall the system was well received and execution beyond the initial technical issues was uneventful. The time allocated was limited and so exposure to the finer aspects of branching and commits was restricted. However, the tuition did adequately support the process necessary to manage continuous assessment submission.

Secure submission

Using free, off-site, established services, it was possible to establish a secure submission channel between student and lecturer that required no input from local IT services. Git also proved secure from accidental code deletion given its inherent backup characteristics.

Underwriting provenance and derivation of submitted work

When regularly pushed to the hosting service, the ongoing work by the student was clearly visualised. Indeed, the sudden appearance of an established solution as a single commit served to draw attention to the submission, effectively acting as a plagiarism alert.

Better source file organisation

A regular problem with submissions was the proliferation of redundant and/or temporary files as well as duplicate versions of submissions buried in sub-folders.

In the first instance, Git provides a *gitignore* file which can be customised to ignore (i.e. not track) certain files. This (together with the general efficiency of the Git protocol) greatly reduced the volume of submissions, easing the archiving burden.

In the second instance, the ability to create branches of code permitted students to experiment with alternate scenarios for their code development. These could later be merged into the assignment branch for pushing to the server - or - they could be discarded or left as is.

Overall, the greater attention to the complexity that evolving code branches can bring, encouraged students to be more diligent in their management of the code - particularly, now that a tool existed to support their efforts.[4]

Integrated Issue Tracking

Both Github and Bitbucket provide integrated issue tracking. This proved a useful adjunct to the existing learning management system as it placed questions precisely at the point in the code where the issue arose. By employing Git regularly (e.g. distributing class exercises), familiarity with the online service grew. With the set up described above, all team members could view the same class repositories and so comments and responses were automatically shared with all.

Exposure to standard industry practice

The notion that they were using industry standard tooling appealed to students. Though the module did not credit use of Git directly, the uptake was >90% by the first submission. Several groups of students undertook to use Git for other project work on their programme of study.

Conclusion

CS faculty and students are to be encouraged to adopt Git as a standard mechanism to manage both class materials and continuous assessment submissions. Particularly in light of budget constraints and the significance of VCS to industry, the value proposition for off-site, free, hosting services is sound. Students responded well to the use of such well-established industry tooling. The Git system was forgiving to novices though the GUI tools would likely be of greater value when knowledge of the system matures for students. The online services proved robust and reliable and the need for minimal on-site IT support resulted in rapid deployment of the described process. Therefore, I would recommend the adoption of Git in the classroom.

References

1. Eclipse Foundation. Eclipse Community Survey Report 2013. Retrieved November 21, 2013 from http://slidesha.re/1ekmbH1
2. Zappe, Sarah, et al. "Flipping" the classroom to explore active learning in a large undergraduate course." American Society for Engineering Education. American Society for Engineering Education, 2009.
3. Chacon, Scott. Pro git. Apress, 2009.
4. Griffin, Terry, and Shawn Seals. "GitHub in the classroom: not just for group projects." Journal of Computing Sciences in Colleges 28.4 (2013): 74-74.
5. Lawrance, Joseph, Seikyung Jung, and Charles Wiseman. "Git on the cloud in the classroom." Proceeding of the 44th ACM technical symposium on Computer science education. ACM, 2013.
6. Novak, M., et al. "Student's progress tracking on programming assignments." Emerging eLearning Technologies & Applications (ICETA), 2012 IEEE 10th International Conference on. IEEE, 2012.
7. Halstead-Nussloch, Rich, Jonathan Lartigue, and Orlando Karam. "Free hosting options for student projects." Journal of Computing Sciences in Colleges 29.2 (2013): 203-204.
8. Hart, Delbert. "A survey of source code management tools for programming courses." Journal of Computing Sciences in Colleges 24.6 (2009): 113-114.