Auxiliary Materials for OOPSLA ′16 Paper “Purposes, Concepts, Misfits, and a Redesign of Git”

August 22, 2016

1 Overview

This package contains auxiliary materials for the paper “Purposes, Concepts, Misfits, and a Redesign of Git”. A description of the contents of this package is included in Table 1.

The “Getting Started Guide” (§2) contains instructions on how to run and test Gitless. The “Step-by-Step Instructions” (§3) explains how to reproduce the user study.

2 Getting Started Guide

The instructions included in this section focus on key features that make Gitless different to Git as explained in §7 of the paper. There is more to Gitless than what appears in the paper and you are welcome to explore other features if you want. In addition to the documentation at `doc/`, all Gitless commands have a `-h/--help` flag that you can use to get information on the command.

1. *Install Gitless.* If are using the VM you can skip this step. To install Gitless in your machine open a Terminal and do `cp bin/gl_v0.8.3-darwin-x86_64/gl /usr/local/bin/gl` if you are running Mac OS, or `cp bin/gl_v0.8.3-linux-x86_64/gl /usr/local/bin/gl` if you

`bin/`	Gitless binaries corresponding to v0.8.3 and v0.8.2 for Mac OS and Linux. v0.8.3 is the most recent version as of June 15, 2016. v0.8.2 is the version that was used for the experiment. The binaries should work as long as you are running a recent version of Mac OS or Linux and have Git installed. If you are a Windows user, or if you prefer to avoid installing Gitless in your machine, use the included virtual machine
`doc/`	The Gitless's v0.8.3 documentation. This is also available online at `http//gitless.com`
`playground/`	The playground folder contains a Gitless repository, `playground/dogs`, that you can use to play with Gitless. It has files, commits, and branches. The other repository, `playground/remote-dogs`, serves as the remote for `playground/dogs`. This is so that you can execute commands that work with remotes without requiring an Internet connection
`src/`	The source code of Gitless v0.8.3. The code is also available on GitHub at `http://github.com/sdg-mit/gitless`
`stack-overflow/`	The Stack Overflow questions we analyzed: all questions with more than 30 upvotes tagged with the keyword "git" as of July 18, 2016. To view a particular question online visit `http://stackoverflow.com/questions/<Id>` where `<Id>` is the number corresponding to the "Id" column in `questions.csv`
`user-study/`	All material necessary to reproduce the user study
`vm/`	An Ubuntu virtual machine with Git v2.5 and Gitless v0.8.3 installed. Open a Terminal and if you type `gl` it should work. The `~/aux` folder contains the `bin/`, `doc/`, `playground/`, and `user-study/` contents of this package

Table 1: Contents of aux package

are running Linux. You could also just leave the binary where it is and update your `PATH`. For other installation options, like installing from source or via the Python Package Index, see `source/README.md`.

2. *Playground Repository.* Change directory to `playground/dogs`. The dogs repository contains a bunch of files with dog breeds. If you now do `gl status` you'll notice that one file, `sporting`, is a tracked file with changes and that there's a new file, `terrier`, that is currently untracked.

3. *Tracking and Untracking Files.* Make `terrier` a tracked file (`gl track terrier`) and `sporting` an untracked file (`gl untrack sporting`). In Gitless, files can move freely between these tracked/untracked classifications (files could also be ignored with a `.gitignore` file like in Git).

4. *Commit.* Tracked files with modifications are automatically considered for commit, untracked files aren't. Untrack both `terrier` and `sporting` and try doing a `gl commit`—Gitless will complain saying that there's nothing to commit. But explicitly tracking and untracking files is not the only way to specify what to commit. The `gl commit` command lets you easily customize the set of files to commit: you can use the `-o/--only` flag to list the set of files you want to commit only, `-e/--exclude` to exclude tracked modified files whose changes would otherwise be included, and `-i/--include` to include untracked files whose changes would otherwise be left out of the commit. Go ahead and create new commits, make some changes to files, create new ones, and play around with the commit flags and the track/untrack set. At any time you can publish your changes to the remote repository using `gl publish` if you want. Note that the `gl diff` command also accepts the same set of `only`, `exclude`, `include` flags.

5. *Branching.* If you do `gl branch` you'll see you are currently on the `master` branch. Let's create a new branch `develop` with `gl branch -c develop`. In Gitless, each branch has a head. Since we didn't specify a head for the new branch, the head of `develop` will be the head of the current branch `master`, but you can choose a different commit when you create a new branch with the `-dp/--divergent-point` flag. You can

also change the head of the current branch with the `-sh/--set-head` flag. Do `gl branch -v` to see the heads of all branches.

6. *Branch Independence.* In Gitless branches are independent from each other. Try this out by making a change to some file and, without committing the changes, switch to branch `develop` (with `gl switch develop`). You'll see that your changes didn't follow you over. Now switch back to `master` and you'll see the changes you were working on before you switched appear again in the working directory. In the case you want the uncommitted changes made in the current branch to be moved to the destination branch the `gl switch` command has a `-mo/--move-over` flag that will do just that.

   In Gitless you can switch branches even if you are in the middle of fixing conflicts. To test this you can do the following: in the `master` branch add a new breed, say "Welsh Terrier," to the beginning of the `terrier` file and commit; switch to the `develop` branch and also add some new (different) breed, say "Airedale Terrier," to the beginning of `terrier` and commit; now do a `gl merge master`. This will show a conflict in the first line of `terrier`. Note how in this conflict state you can still move between branches: for example, switch to `add-hounds`, make some changes and commits there, and then switch back to `develop` to finally resolve the conflict.

3 Step-by-Step Instructions

The instructions included here are for reproducing the user study as described in §8 of the paper. Study participants completed the tasks in a desktop machine running Mac OS v10.10, Git v2.5 and Gitless v0.8.2. It should also be possible to use Gitless v0.8.3 for the study since the changes introduced (a tagging feature and partial commits) are not necessary or useful to complete the tasks.

1. *Recruit Participants.* To recruit participants we sent an email to a lab mailing list. In the email there was a link to a Google Form that corresponds to `user-study/forms/user_study_application.pdf`

2. *Session Plan and Scheduling.* Each participant completes one session using Git and the other one using Gitless. To account for learning

effects, we randomly assigned participants to use Git or Gitless for their first session and schedule the sessions at least a day apart from each other.

3. *Bootstrap Session.* The instructions for bootstrapping the environment for a session are the following:
   (a) Open a Terminal
   (b) Source bootstrap script `user-study/scripts/set_up.sh`. This script will create a folder `~/.ut` that contains the fit-cli remote repository, necessary commands (`ut-run`, `ut-pr-kilos-send`, `ut-pr-kilos-update`, `ut-meters-send` and `ut-pr-meters-update`) and environment variables (`FIT_CLI_REMOTE` with the path to fit-cli remote repository, `SNIPPETS` with the path to the snippets file).
   (c) Change directory to `~/`
   (d) Open the slides that correspond to the current session: `user-study/slides/git_tasks.pdf` for Git or `user-study/slides/gl_tasks.pdf` for Gitless.
   (e) Start a screen recorder. (We used QuickTime Player v10.4.)
4. *End of Session Surveys.* At the end of the session the participant needs to answer the end of session questionnaire on their experience using the tool to complete the tasks. The corresponding forms are `user-study/forms/git_end_of_session_survey.pdf` for Git, and `user-study/forms/gl_end_of_session_survey.pdf` for Gitless. If the session were the participant's last session, they would also complete the end of study survey `user-study/forms/end_of_study_survey.pdf`.