Using GitHub© for a Shared LaTeX Writing Project

Linda Briesemeister
linda.briesemeister@sri.com
SRI International

May 22, 2014

Abstract

This is a short tutorial on how to use GitHub for shared LaTeX writing projects as it provides free hosting of git repositories, which may be sufficient when authors collaborate without having a common IT infrastructure. Note that such repositories are by default public so your text is effectively public as well. Private repositories in GitHub are a paid-for service.

Disclaimer: This tutorial may be outdated if GitHub changes its user interface. We tried to explain the steps needed based on how this service operates at the time of writing. No warranty is provided.

Contents

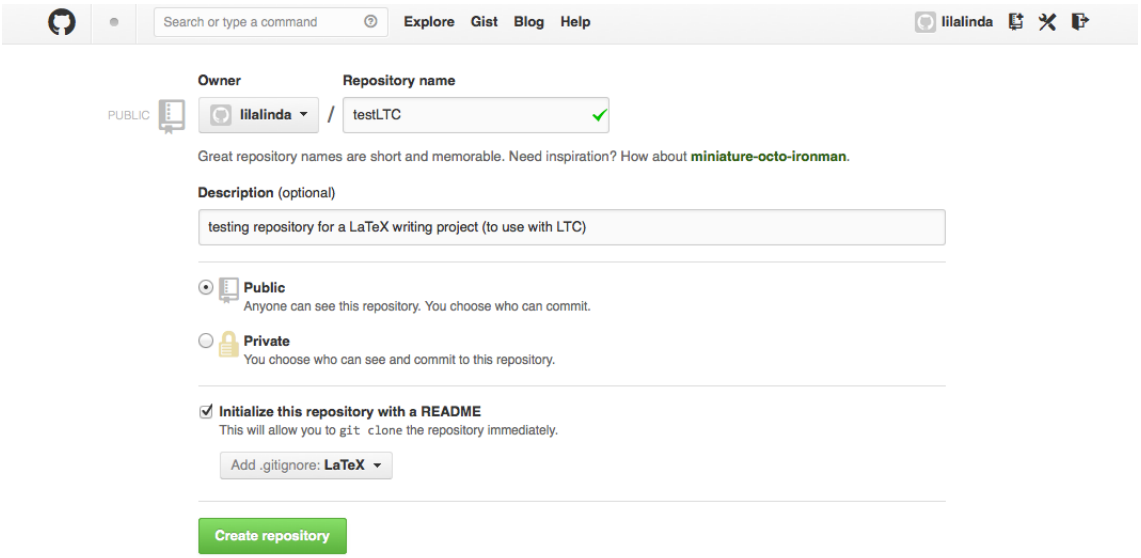

Figure 1: Create repository on GitHub

1 Creating Repository

In this section, we will first create a repository on GitHub and then clone it to your local machine. We will also add collaborators to the repository to be able to have others push their changes to it. If a coauthor has already set up a repository on GitHub, see Section 1.4 for more details on cloning an existing repository.

1.1 On GitHub

Log into your GitHub account (or create one) and then click the icon on the top right that shows the tool tip "Create a new repo" or use https://github.com/new. This should open a page similar to the one shown in Figure 1, with your GitHub user name instead of "lilalinda." Now enter a repository name such as "testLTC" and optionally a description such as "testing repository for a LaTeX writing project (to use with LTC)" in our example. We selected this to be a public repository as we do not have the paid-for service of private repositories. Finally, we choose "LaTeX" for a predefined `.gitignore` file—this will by default ignore some build products under LaTeX, which is quite useful. Finally, click the button "Create repository" and proceed.

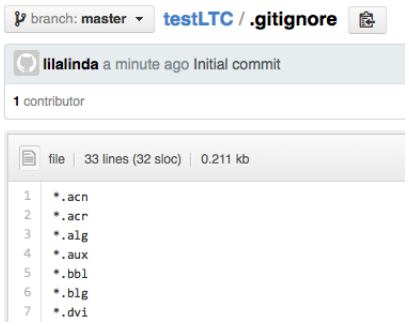

Figure 2: Generated `.gitignore` file

When the repository is created, it shows the page at https:

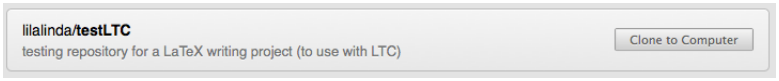

Figure 4: Ready to clone repository with GitHub for Mac

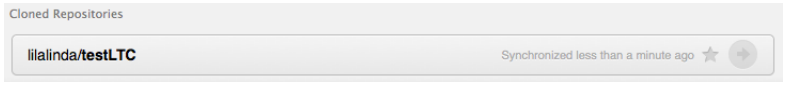

Figure 5: After cloning repository with GitHub for Mac

//github.com/lilalinda/testLTC, where your user and repository names are replaced. Now click on the link of the file `.gitignore` to bring up the contents of this generated file as seen partly in Figure 2. This file lists all common helper file types to ignore in a LaTeX writing project under git version control.

1.2 On Your Computer

Now it is time to clone the repository on GitHub to your local machine. There are two recommended options for working with GitHub repositories.

1. Using the GitHub for Mac application
2. Using the command line

We cover both in the sections below.

1.2.1 Cloning with GitHub for Mac

If you are using the GitHub for Mac application, click on the button "Clone in Desktop" at the bottom right corner of the main repository page, in our example at https://github.com/lilalinda/testLTC (again, replace your user and repository name accordingly if using the URL). If you have GitHub for Mac already installed and are using Firefox as your browser, you may encounter a window asking about the application to use for such links. You should select "GitHub" and may also choose to check the box to remember this setting for the future as seen in Figure 3. Otherwise, it will redirect you to download and install GitHub for Mac.

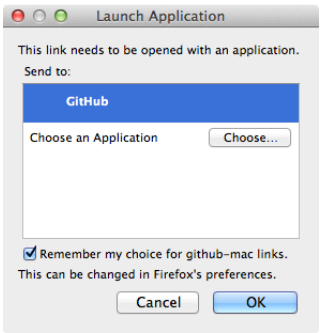

Figure 3: Firefox dialog when opening `github-mac` links

Once the GitHub for Mac application is open, select your username under the heading "GITHUB.COM" in the left panel and see a list of repositories on the server that you have created. The panel on the right should show the test repository looking similar to Figure 4. Now click the button "Clone to Computer" in order to bring up a dialog to select the location of your local repository. Once this has finished,

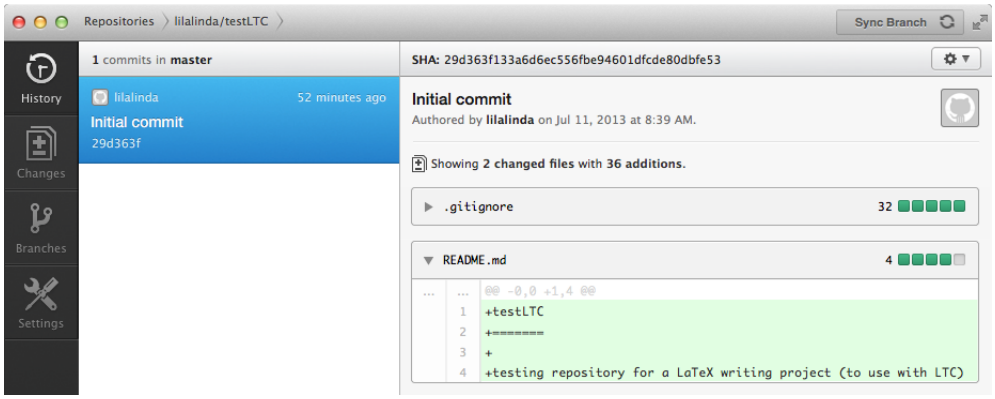

Figure 6: Initial history of repository with GitHub for Mac

the panel in GitHub for Mac will show the repository now under Cloned Repositories as seen in Figure 5

Next click the arrow pointing right in the repository panel to open the history of the repository. With only the initial commit, the history looks similar to Figure 6 when you collapse the contents of the `.gitignore` file using the small arrow to the left of it.

1.2.2 Cloning with the Command Line

Figure 7: Clone URL in GitHub

If you choose to work with git from the command line instead of the GitHub for Mac application, open a terminal window and change into the directory where you want the local repository to reside. Then, you can copy the URL for cloning from the text field labeled "SSH clone URL" (click on the link "SSH" first if the label is different) at the bottom right of the repository page as seen in Figure 7. You may want to choose a different URL, but SSH works well if you have uploaded your public key to GitHub using https://github.com/settings/ssh. There, you will also find more detailed instructions on how to generate an SSL key. Now back at the repository page, copy the SSH clone URL and add it to the `git clone` command as follows.

```
$ git clone git@github.com:lilalinda/testLTC.git
Cloning into 'testLTC'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
Receiving objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
```


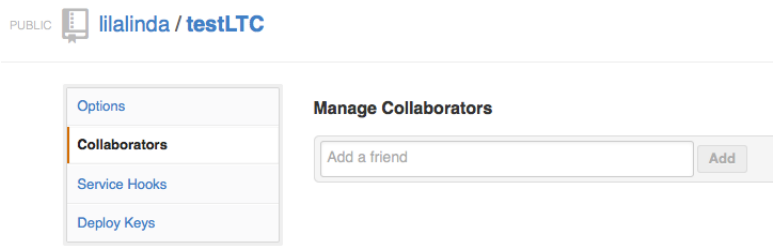

Figure 8: Managing collaborators ih GitHub

1.3 Adding Collaborators

In GitHub, we can add other users as collaborators, which is useful for our coauthors (who need to have their own GitHub accounts) so that they can contribute text. From the repository page, we select the link "Settings" in the right column currently pointing at `https://github.com/<username>/<reponame>/settings` (replace user and repository names). On the repository settings page, find the link "Collaborators" in the top-left menu; currently pointing at `https://github.com/<username>/<reponame>/settings/collaboration` (again, replace user and repository names). You may have to authenticate again in order to see the page with the heading "Manage Collaborators" as seen in Figure 8.

Here you can add the GitHub account names of coauthors who will be then able to upload their changes to the repository. Once you are collaborating, it is prudent to communicate with your coauthors by other means such as email or phone to decide who is editing which file in the repository to avoid merge conflicts. Git can handle merge conflicts to a certain extent but when the same file contains too many changes from different authors it may need human guidance to resolve the problem. See the tutorial sections in the LTC manual for examples of git merge conflicts and how to resolve them.

1.4 Using an Existing Repository

Now let us look at the situation when one author has already created a git repository on GitHub containing the files for a LaTeX project. Your coauthor will give you a link to the repository that should be of the form `https://github.com/<username>/<reponame>`. Once you are on this page, you will see again the button "Clone in Desktop" to be used with the GitHub for Mac application or you can copy the SSH clone URL from the text field (potentially clicking the link SSH first). See Section 1.2 above for more details on cloning. In the latter case of copying the URL, you would use the copied text with a the clone command from the command line after changing into the local directory where you want your working copy to reside.

```
$ cd <dir>  # where you want your repository to be created
$ git clone git@github.com:<username>/<reponame>.git
```

Whether you are allowed to push your changes back to the coauthor's repository is defined under the collaborator settings of the repository. If you get an error message when you try to sync your changes back, ask your coauthor to add you as a collaborator under GitHub.

2 Adding Content

On your local machine, create a LaTeX file with the following minimal content in the directory where you cloned the repository. We assume for the remainder of this tutorial that the file name is `mydocument.tex` but feel free to change the name and adjust actions as necessary.

```
\documentclass{article}

\begin{document}
\end{document}
```

Now before we commit the new file, we may also want to edit the already committed `.gitignore` file to include the build product from running LaTeX on our main file. Therefore, add a line with the PDF file name of the main LaTeX document to the file `.gitignore` so that it looks like the following.

```
...
*.tdo
mydocument.pdf
```

To do so, you can either use a text editor or this command if you are using bash (use the first command to verify this):

```
$ echo $SHELL
/bin/bash
$ cat >> .gitignore <<EOF
> mydocument.pdf
> EOF
```

The sections below show how to add and commit the new file and any edits using the GitHub for Mac application or with the command line.

2.1 Committing with GitHub for Mac

If you are using GitHub for Mac, the "Changes" tab when inside the repository becomes illuminated when we save the new file and edit the existing one. Clicking shows a view similar to the one in Figure 9.

To commit, provide a meaningful message like "main document and ignoring matching build product" as seen in Figure 10 and

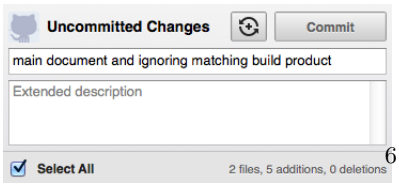

Figure 10: Commit message in GitHub for Mac

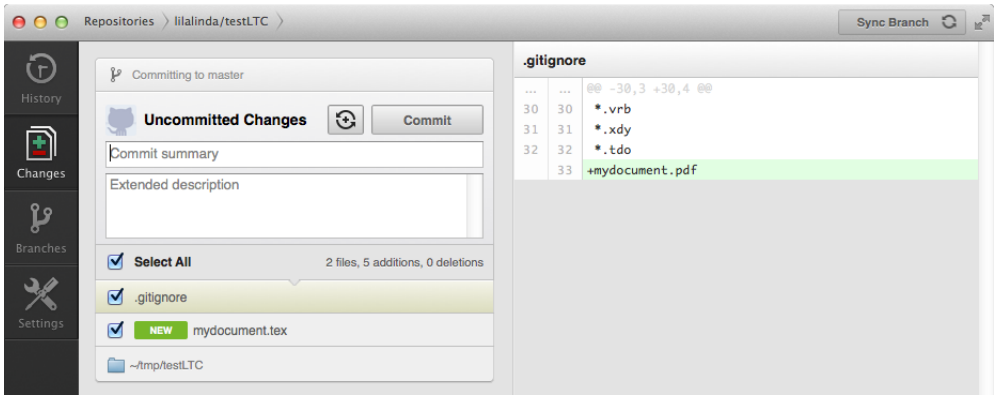

Figure 9: New and edited files with GitHub for Mac

then click the "Commit" button. After the commit, the application shows Unsynced Commits in the bottom panel, which you can ignore until it is time to sync with GitHub pushing your changes there for your coauthors to enjoy.

2.2 Committing with the Command Line

If you are working with the command line, check the status of the repository.

```
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   .gitignore
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       mydocument.tex
no changes added to commit (use "git add" and/or "git commit -a")
```

Now add the new file and then commit both using the `-a` switch as seen below.

```
$ git add mydocument.tex
$ git commit -am "main document and ignoring matching build product"
[master 0893956] main document and ignoring matching build product
 2 files changed, 5 insertions(+)
```

```
 create mode 100644 mydocument.tex
```

If you check the status after the commit, it will tell you that you are no longer in sync with the server (origin of repository), which you can ignore until it is time to push your changes back to GitHub for your coauthors to enjoy.

```
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
nothing to commit, working directory clean
```

3 Sharing Your Work

To share your work with your coauthors, you will want to push your local repository at times to the server. Also, if others are pushing their changes to the server you will want to occasionally pull their changes into your local repository. The sections below show how to share your work using the GitHub for Mac application or with the command line.

3.1 Synchronizing with the Server with GitHub for Mac

In the "Changes" panel of the application, click the "Sync" button. This may take a moment as the application is exchanging data with the remote GitHub server. After a successful synchronization, the icon in the left bar becomes gray scale signaling that there are no lingering edits or commits and your local repository is in sync with the remote repository.

Let us edit the main document to show ongoing work, for example by adding the following line to the LaTeX preamble:

```
\usepackage{url} % for typesetting URL's
```

Once saved, the application shows the uncommitted change in the "Changes" panel. Let us commit with a message such as "things for the LaTeX preamble" and then do further edits to the file, for example another package import statement, and save it to disk.

```
\usepackage{color}
```

Even though the "Sync" button appears in the "Unsynced Commits" panel, clicking it results in a failure message as we have uncommitted changes to the file. Only when all edits for tracked files are committed will the synchronization work.

3.2 Synchronizing with the Server with the Command Line

To upload your latest commits to the GitHub server, you use the `push` command, for example:

```
$ git push
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 442 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
To git@github.com:lilalinda/testLTC.git
   387b684..672c28b  master -> master
```

If you are working with git branches, you may have to specify the branch with this command but that is for more complicated scenarios (often not needed for writing projects).

The opposite command to obtain changes from the server, as a coauthor had pushed some edits to the text, use the `pull` command, for example:

```
$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2)
Unpacking objects: 100% (3/3), done.
From github.com:lilalinda/testLTC
   498aac2..70beabe  master     -> origin/master
Updating 498aac2..70beabe
Fast-forward
 mydocument.tex | 1 +
 1 file changed, 1 insertion(+)
```

As long as there are no conflicts that git cannot resolve, this command merges any remote changes with your working copy. You should make sure to have no lingering changes waiting to be committed when you pull from the remote repository. The man page for the `git-pull` command says:

> If any of the remote changes overlap with local uncommitted changes, the merge will be automatically cancelled and the work tree untouched. It is generally best to get any local changes in working order before pulling or stash them away with `git-stash`.

4 Typical Work Cycle

In Figure 11, we show a diagram of how a typical work cycle with a shared writing project looks like. We added the details for using Github; either with the application for Mac or from the command line. However, this also applies to git repositories hosted elsewhere and other version control systems (although centralized systems such as svn typically require connectivity when committing while distributed systems do not need to be online then.)

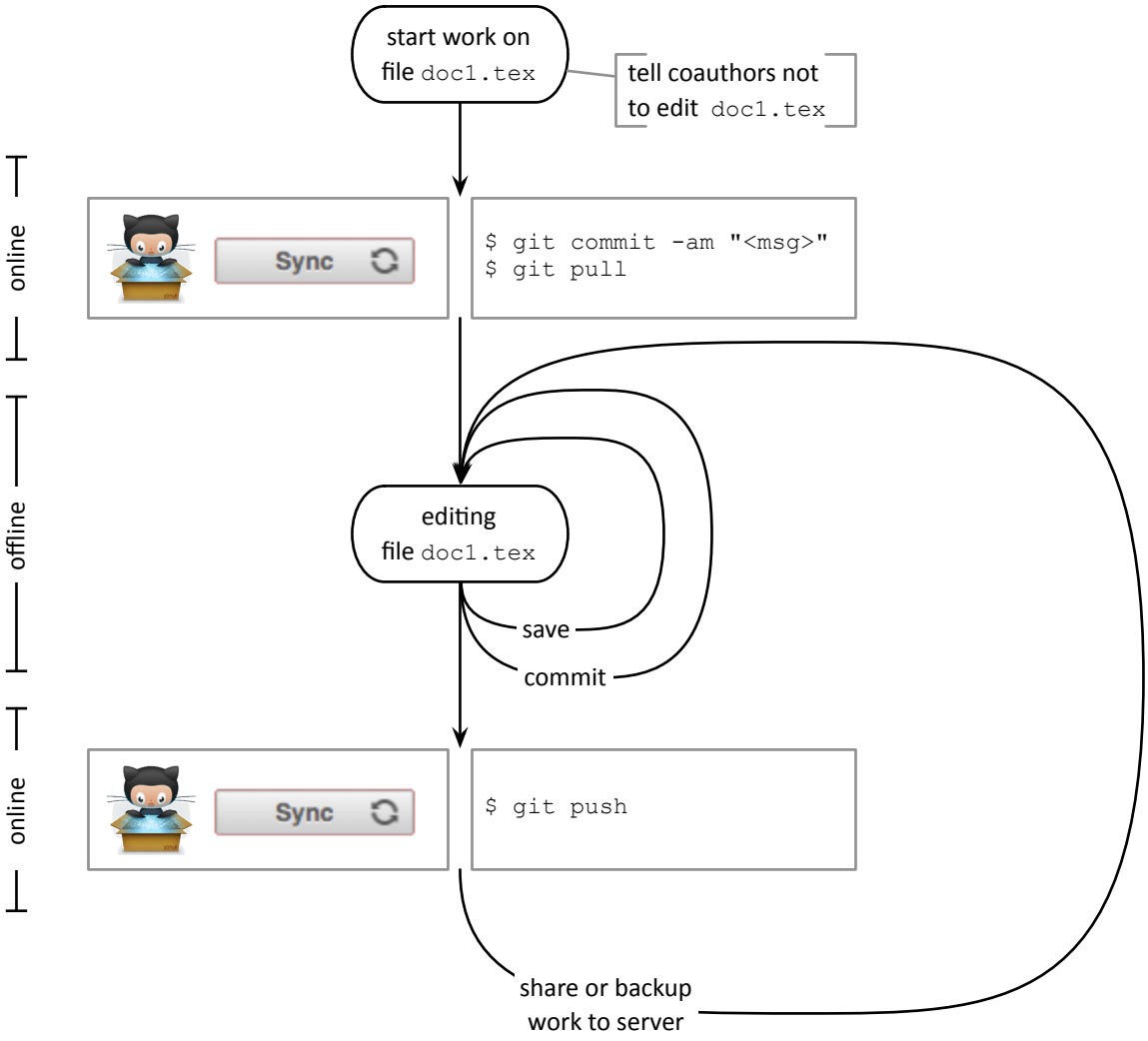

Figure 11: Typical work cycle for shared writing project

