Version Control 101

> Exported from http://cepsltb4.curent.utk.edu/wiki/efficiency/vcs, please visit the link for the latest version and the best typesetting.

Version Control 101 is created in the hope to minimize the regret from lost files or untracked changes.

There are two things I regret. I should have learned Python instead of MATLAB, and I should have learned version control earlier. Version control is like a time machine. It allows you to go back in time and find out history files.

You might have heard of GitHub and Git and probably how steep the learning curve is. Version control is not just Git. Dropbox can do version control as well, for a limited time. This tutorial will get you started with some version control concepts from Dropbox to Git for your needs. More importantly, some general rules are suggested to minimize the chance of file losses.

Contents

Version Control 101 1
- General Rules 2
- Version Control for Files 2
  - DropBox or Google Drive 2
  - Version Control on Confluence 5
- Version Control for Source Code 6
  - LaTex Source on Overleaf 6
  - Source Code and General Version Control using Git 7
- Conclusion 12

General Rules

- Always start writing in a version-controlled folder or tool
- Develop a habit of pressing Ctrl + s (⌘Command + s for macOS)
- Don't put all the eggs in one basket. Nor should it for your files.
- Commit the changes frequently if you are using Git

Version Control for Files

This section introduces version control for files on the disk. The contents of this section are general for daily use.

DropBox or Google Drive

Cloud-based sync tools can sync your files on different devices to the cloud and store their historical versions. There are abundant choices such as DropBox, Google Drive, OneDrive, and Box. If you are already using one and find it handy, it is wise to stick to it.

The general rule is to use the popular ones that have a large user base and offer paid options. Their services are more stable, secure and sustainable. The recommended ones are Dropbox and Google Drive. My evaluations for Dropbox are the following:

1. Cross-platform support. Besides Windows and macOS, Dropbox supports major Linux distributions.
2. A slightly more responsive sync mechanism.
3. Small storage space for free and more expensive paid plans. You can only get 2GB for free and must choose 1TB for the basic paid plan.

For Google Drive, you could expect:

1. 15 GB shared storage with your personal Gmail and Google Apps.
2. Can backup files anywhere, including your desktop and home directory
3. Unlimited storage space for Google Docs.
4. Inexpensive, shareable paid plan. You could get 200 GB for $20/year and share it with five family members.

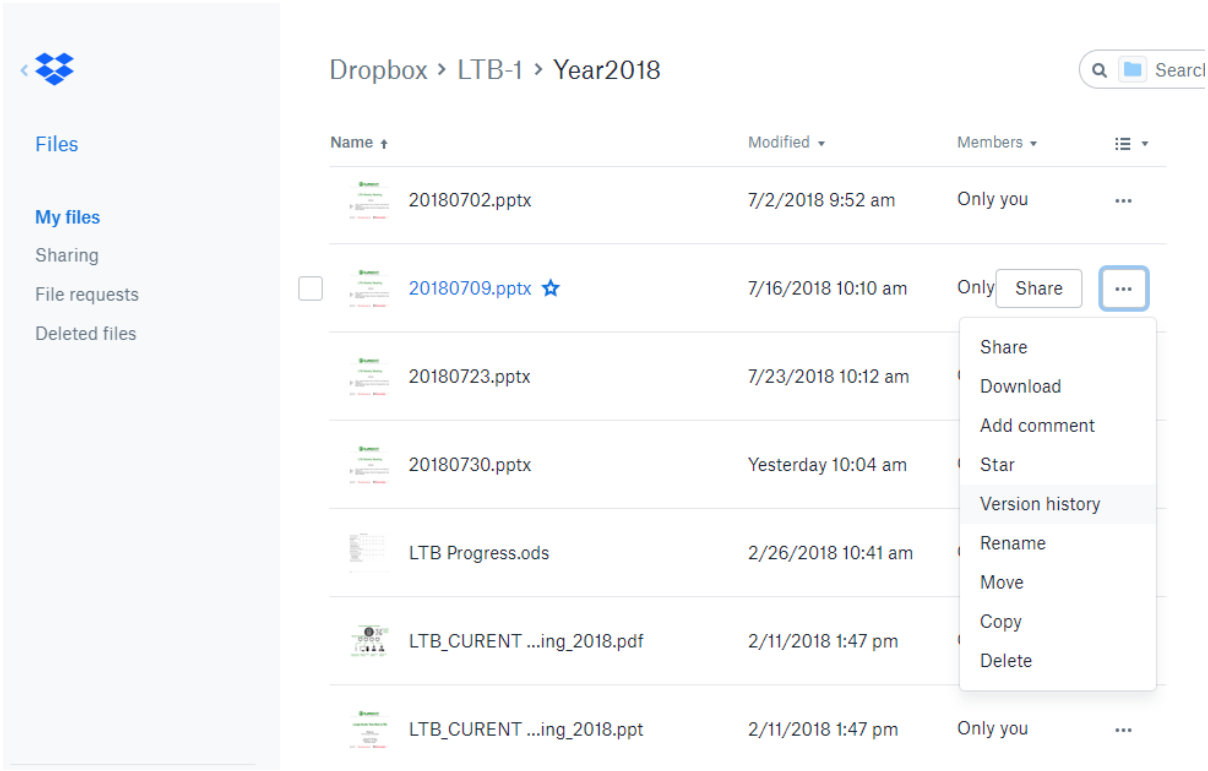
 It is recommended to avoid using the UT Volmail services (Google Drive or Microsoft Onedrive) to store personal files.

I have Dropbox (free plan) and Google Drive (paid, basic) because a) I use Linux, and b) Dropbox's offer is way beyond my needs.

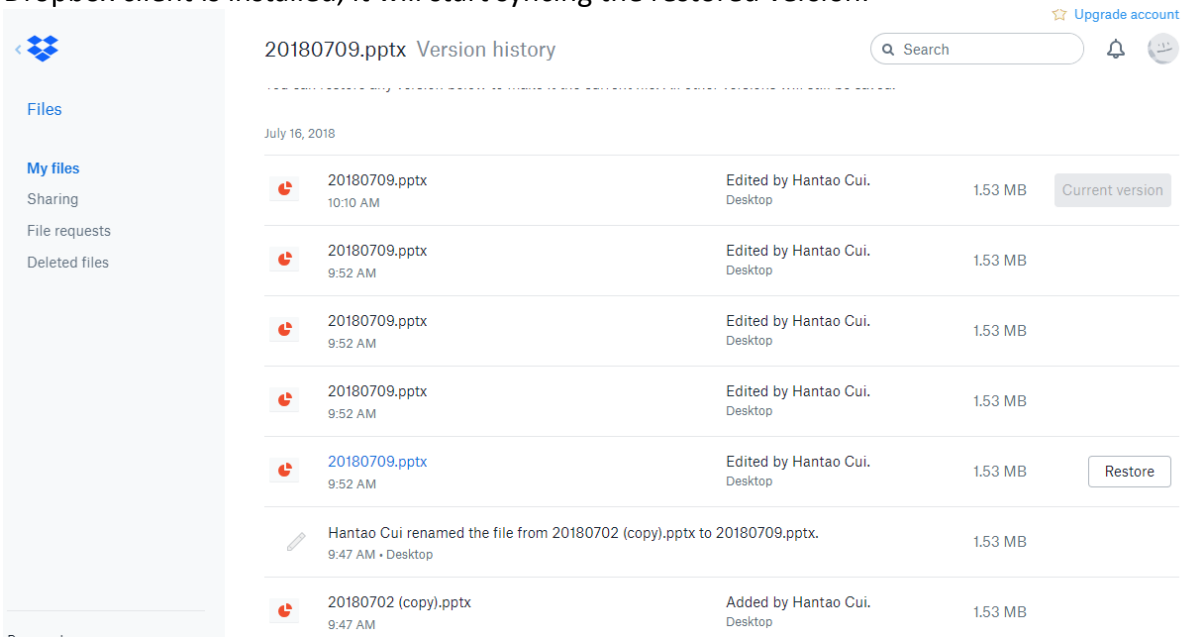
Restoring files in Dropbox

Dropbox stores the history version of your files for up to 30 days (120 days for paid plans). Restoring data in Dropbox can be done from the website.

- Go to the folder containing the file, and find “Version history” in the menu (“...”).



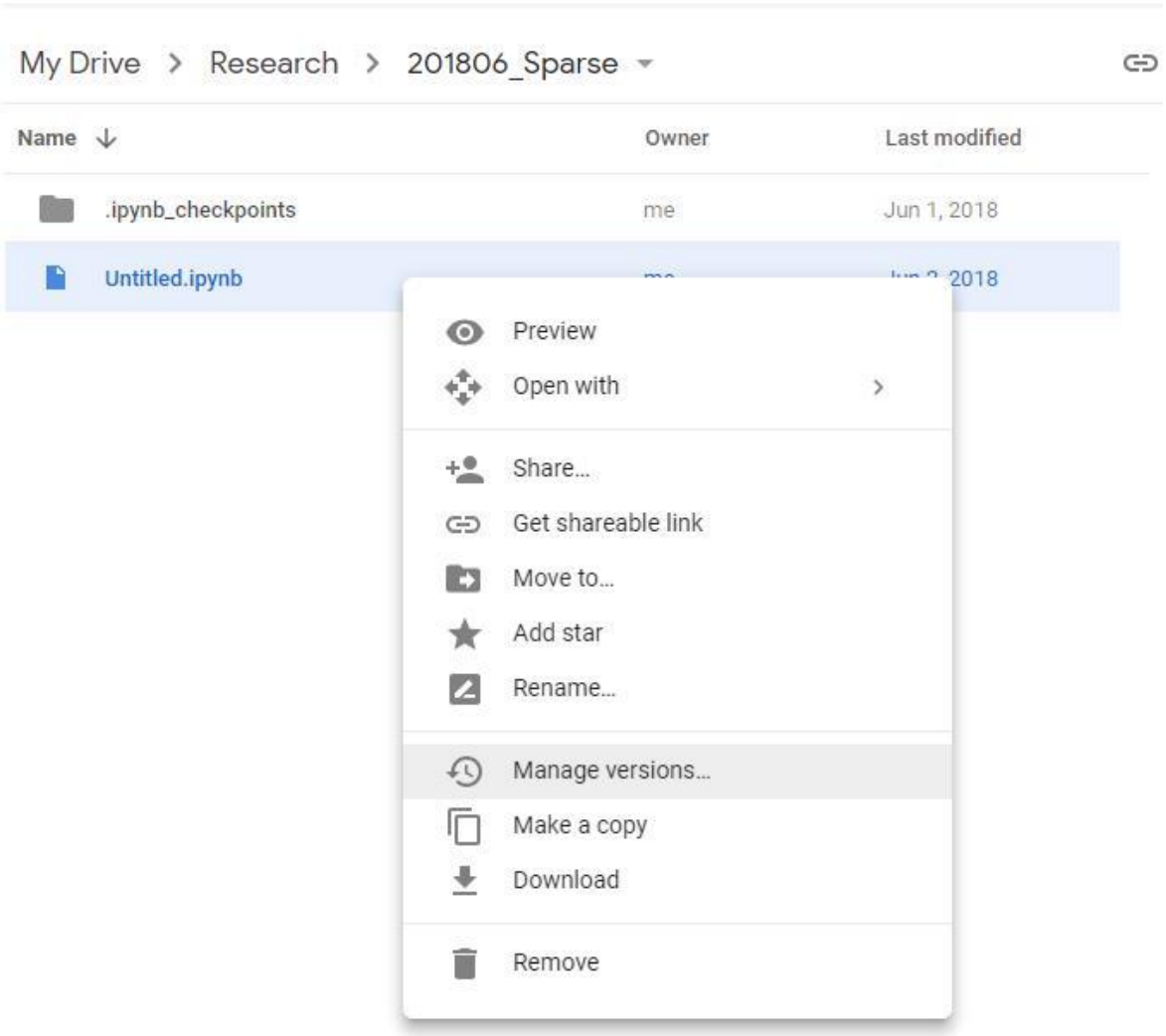
- Hover on the version you wish to restore and click on “Restore” button on the right. If your Dropbox client is installed, it will start syncing the restored version.



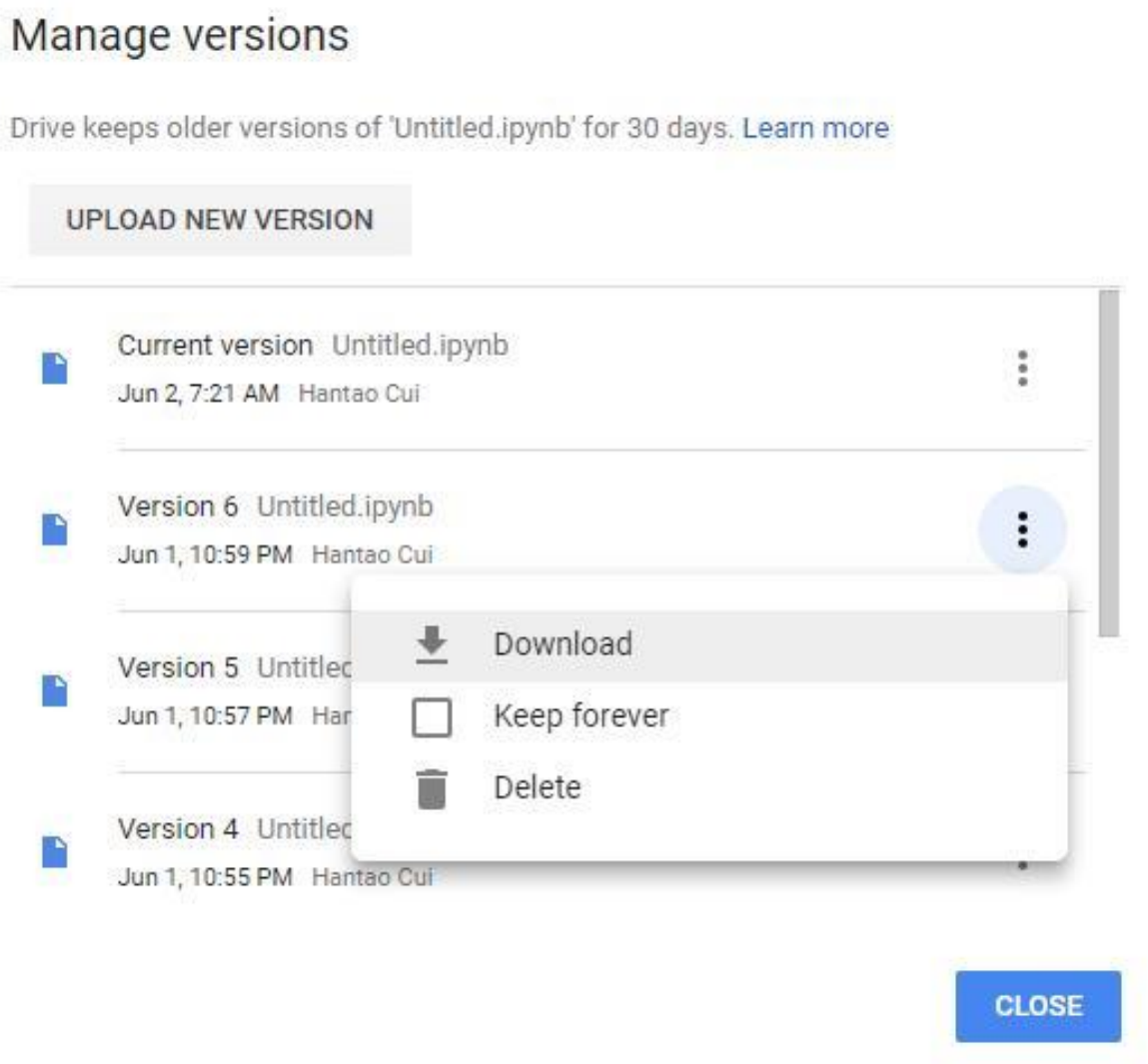
Restoring files in Google Drive

Google Drive stores history versions of your files for up to 30 days. But you have the option, within 30 days, to mark a version permanent.

- On https://drive.google.com, locate the file you wish to restore.



1. Right click on the file, in “Manage versions ...”, you can find the history versions of the file



1. You can download a copy of the history version, or mark it as permanent to keep.

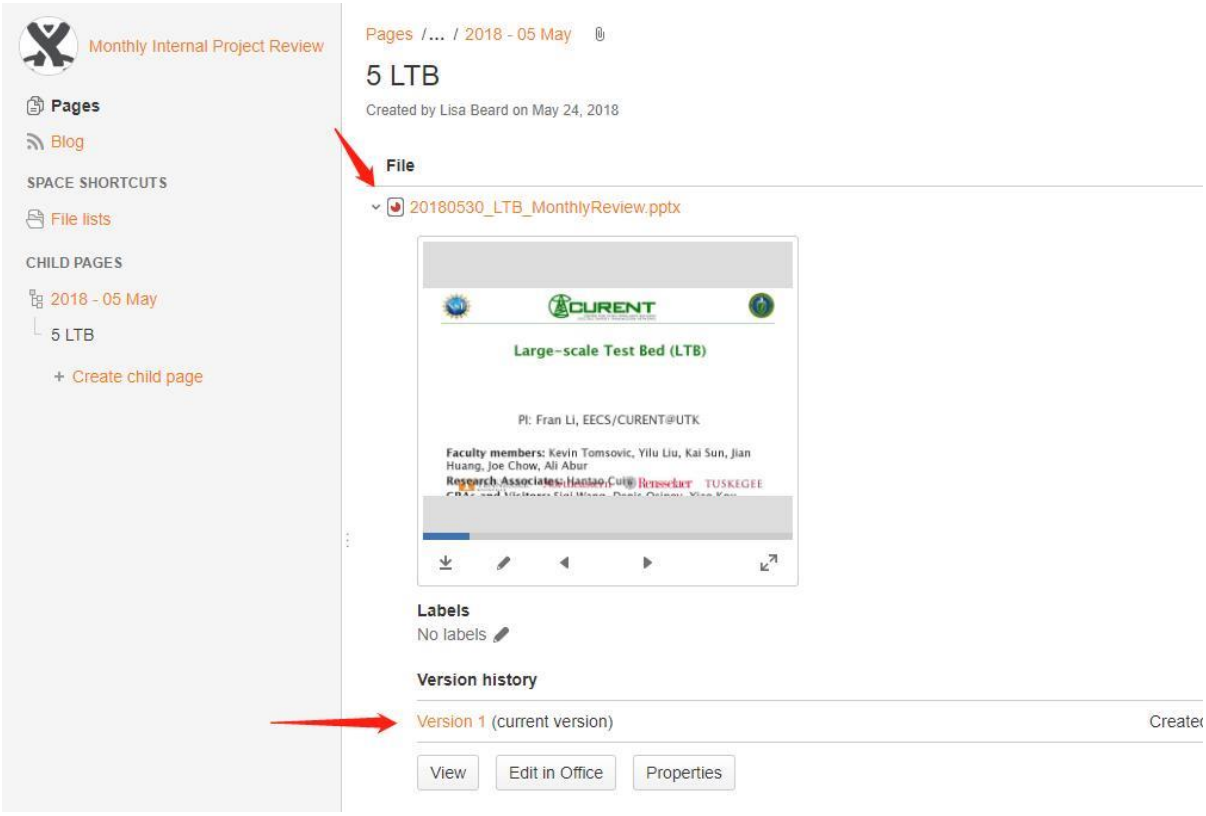
Version Control on Confluence

CURENT has an online collaboration tool called Confluence, available at http://confluence.curent.utk.edu/. We use confluence to share CURENT-project related reports, presentation slides, and research data.

Confluence uses a different credential system from the UT Net ID. If you need to gain or recover access, contact the CURENT IT Helpdesk at help@curent.org

Confluence has a built-in version control system which automatically tracks the history versions of uploaded files. To use this function, **use the same file name** when uploading the file. Confluence automatically tags the old version and overwrites it with the latest upload. In other words, refrain from adding any suffix like _v2 in the file name when uploading so that Confluence could recognize the file to overwrite.

To find out an older version, expand the left arrow and find at the bottom.



By default, you don't have delete access on Confluence. Double check and make sure the files are legit to upload, and the location is correct. Contact CURENT IT if you need to delete anything.

Version Control for Source Code

In addition to files on the drives, developers have invented tools to track the changes in source code so that a developer can revert the inappropriate changes. For a team, version control tools can simplify the collaborations between members who might happen to change the same segment of code.

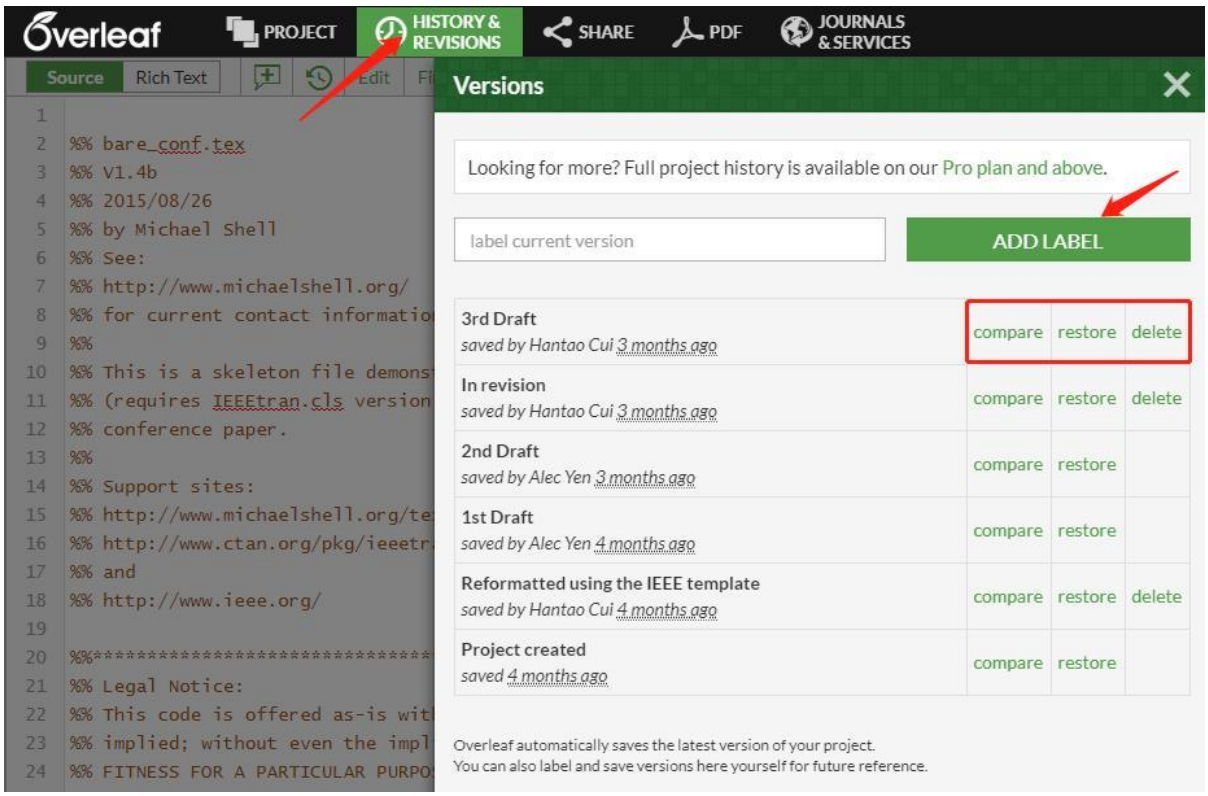
LaTex Source on Overleaf

If you use LaTeX for typesetting, you might have heard of Overleaf. Overleaf is a collaborative LaTeX editing website for writing reports and technical papers. Think of it as a “Google Doc” for LaTex. You can sign up using my referral link here if you haven't.

I do not recommend preparing thesis or dissertations on Overleaf because it could not handle nested folders, which I frequently use to organize figures.

Overleaf has a version control function slightly different than Google Docs. The Overleaf free plan lets you **label the revision you wish to save manually**, while Google Docs saves every change automatically. To label the current version, open “History and Revisions” on the top of the menu. You can also compare the differences between the LaTex source files and restore history versions.

Before you restore any history version, it is wise to save the current version in a new label.



Source Code and General Version Control using Git

General ideas

This section is a summary of the previous tutorials I gave for the LTB project. None of the above version controls are handy for a broad base of source code, especially when there are multiple collaborators. Git is a software developed by Linux Torvalds, the father of Linux, in his spare time for managing Linux source files contributed by people around the world.

Git is not an acronym. Torvalds said: “I'm an egotistical ***, and I name all my projects after myself. First 'Linux', now 'git'”

GitHub is not Git. GitHub is a popular host platform for Git-versioned projects.

There are alternatives such as bitbucket.org, which allows hosting private projects for free (for up to 5 collaborators).

The idea of Git is simple and elegant, but the commands have a modest learning curve. You will need to download git from https://git-scm.com/book/en/v2/Getting-Started-Installing-Git and install it. This tutorial aims to get you started by creating your repository on Bitbucket.

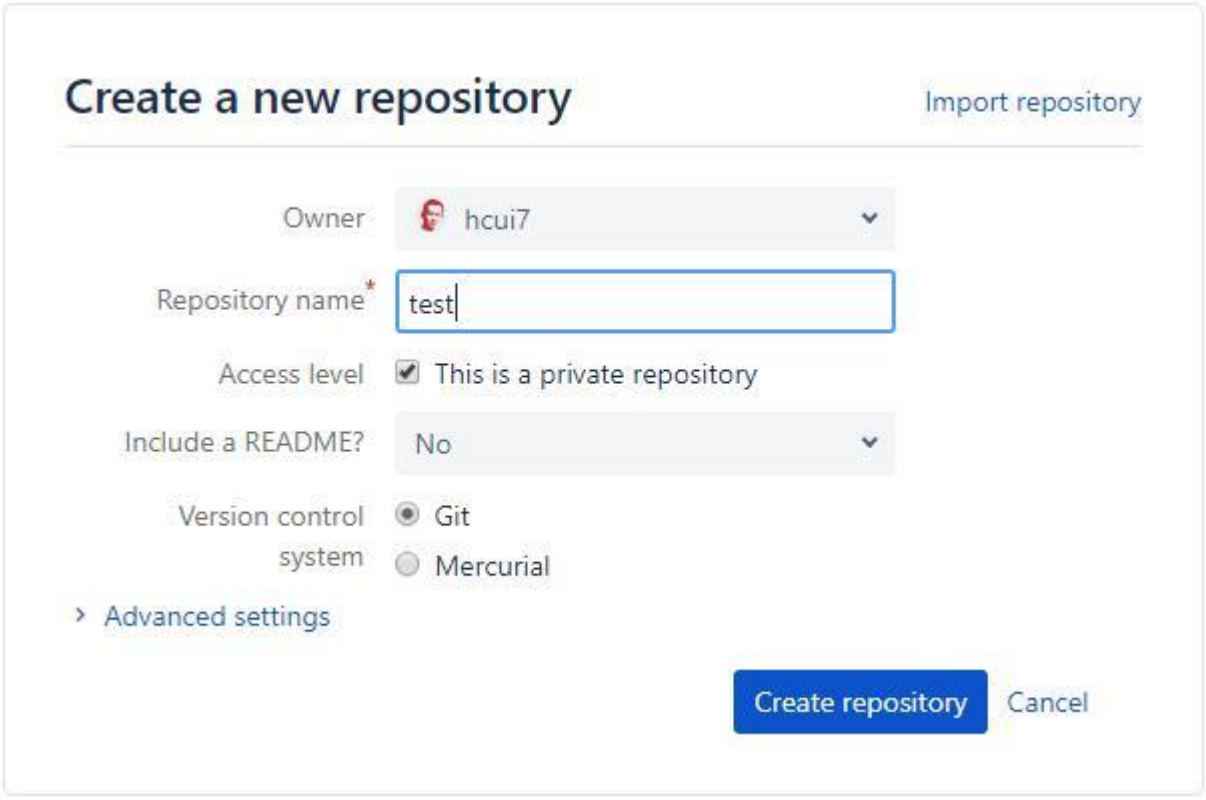
Before getting started, let's clarify some terms:

- repository (or repo): a place where a git project is stored. The URL of a GitHub project, for example, https://github.com/cuihantao/Andes, is the address to a repository called Andes owned by cuihantao.
- clone: make a copy of the repository into a folder on your local machine.
- commit: a tagged version, or the act of tagging a version

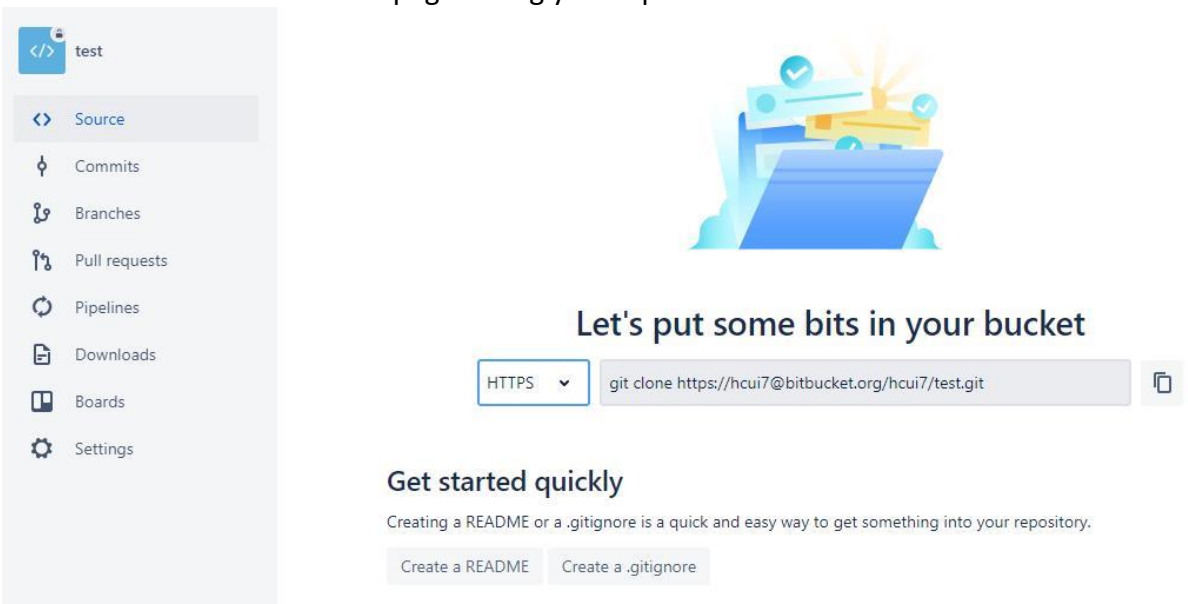
Creating your first repository

Now let's create a repository on bitbucket.

1. You need to register and login first.
2. On the left of your dashboard, click on the “+” button and add a repository. Choose a name for your new repo, for example, test, and create the repository.



1. You will land on a new page asking you to put in some stuff.



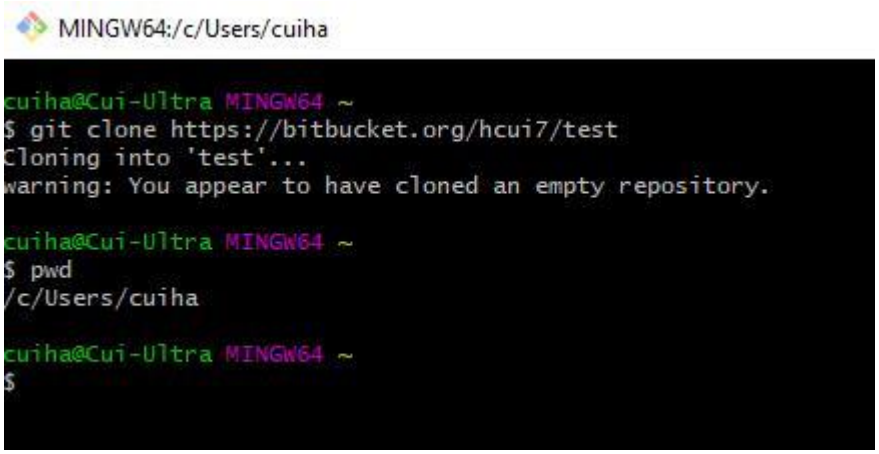
Say you have some code you wish to add to the repository. You cannot directly upload it like in Dropbox. The repository is on the remote server. You will need to clone it as a local copy, make the necessary changes, commit the changes, and push to the remote server.

Assume you have Git for Windows installed. Right-click on any blank space on the desktop and open “Git Bash Here” to open a command line terminal called git bash. Now we will learn a few useful git commands.

All git commands start with git and followed by action and other arguments. To clone the repository to a local folder, use

git clone https://bitbucket.org/hcui7/test/src

The above link is the address to my repository for the tutorial. Replace it with yours. Find out the location of the repository with the command pwd



You will get a warning of cloning an empty repository. Ignore it just because you did. Go to the directory and open the folder using the file explorer with the following commands cd test explorer .

Go ahead and copy your files to this folder, and move back to git bash. Next, we want to ask git to track the newly added files. View the status of the current repository with

git status

cuiha@Cui-Ultra MINGW64 ~/test (master)
$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      my-dissertation.tex
nothing added to commit but untracked files present (use "git add" to track)

It says that I have an untracked file. To include the file in the next commit, use

git add my-dissertation.tex

git add also works on a folder. To add everything, try git add --all.

Use the Tab key for auto-completion in git bash.

Now, run git status again, you can see the new file ready for commit

On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
      new file: my-dissertation.

You only need to add files to track for once. Make modifications to the files as needed. Whenever you want to add a checkpoint, namely, a commit, use

git commit -m "some commit message describing the changes"

[master (root-commit) d515498] my initial
 1 file changed, 164 insertions(+)
 create mode 100644 my-dissertation.tex

This creates a commit with the tag d515498. Files in this commit are as of the time when the commit is performed. You can keep modifying the files. Whenever you are ready, commit the new changes with a new commit message.

-m is a command to pass the commit message. If you run git commit, you will be prompted an editor to write in the commit message.

The commit message should be relevant and concise.

The changes are now committed in the local repository. We want to "upload" it to the remote on bitbucket by running

git push

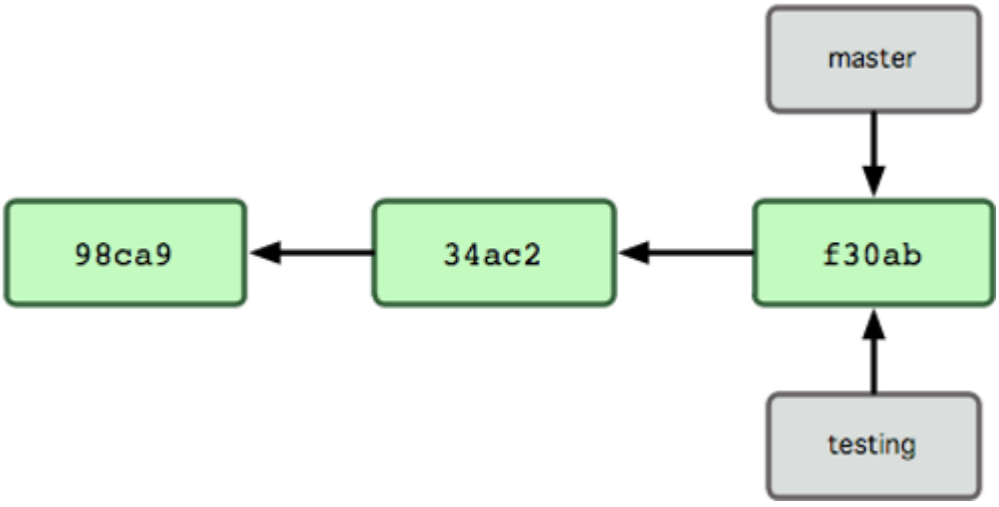
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 3.02 KiB | 3.02 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/hcui7/test
 * [new branch]   master -> master

The push command creates a branch called master for the first time you push. master is the default branch to clone on bitbucket. Refresh the page on bitbucket, and you can see the file pushed to the remote.

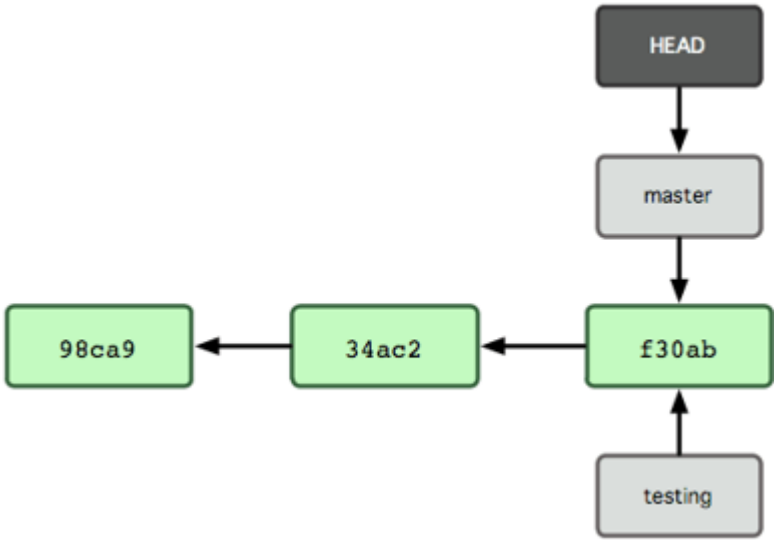
Understanding git

This section introduces the basics of how git works. In the previous example, you see the commit d515498 being pushed to the master branch on the remote. Git stores the commits in a linked fashion as new commits are linked to the previous commits.

There is an important concept called branch. A branch is a movable pointer pointing to a commit. In the figure shown below, there are three commits, from old to new are 98ca9, 34ac2 and f30ab. There are two branches called master and testing, both pointing to f30ab.

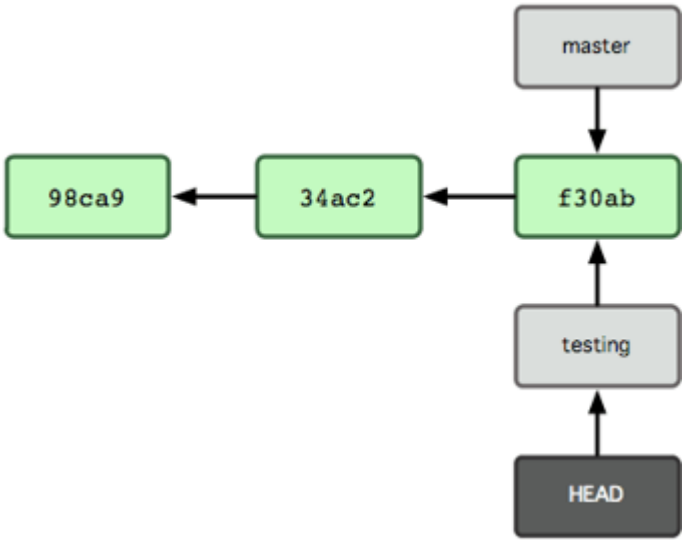


There is a special pointer called HEAD which points to a branch to indicate the current working branch as shown in git status. The next figure shows that HEAD is now at master. Committing new change from f30ab will take a snapshot of the files, assign a new commit, and move the master pointer. The testing pointer will stay unaffected as it is not the HEAD.



To switch to testing, use the checkout command. The HEAD pointer will then move to testing.

git checkout testing



To create a new branch from the HEAD, use the command branch

git branch new_branch

Switch to the new branch with the checkout command.

Note that the branch new_branch is created on your local repository. It has not yet been pushed to any branch on the remote. Nor does git know which remote branch this new branch is supposed to track. When you have committed your changes and ready to push to the remote, add the switch -u to indicate the upstream branch to track:

git push -u origin new_branch

Here origin is a nickname of a remote, and new_branch is the branch name you wish to create and track on the remote. To view the list of remotes, use git remote -vv. To view the list of branches, use git branch -vv.

If you have more than one working computers, it is recommended to push the commits to the remote before leaving. On other computers which do not have the latest commits, run git pull to pull the latest changes. If you have new branches, run git fetch --all.

You can have multiple branches working in progress at the same time. As the code base gets mature, you can merge some branches into one to consolidate the changes. Say you want to merge the dev branch to master, first checkout master and perform the merge, using

```
git checkout master
git merge dev
```

```
Updating 8d98ef0..9121b0a
Fast-forward
 README.md | 169 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 169 insertions(+)
 create mode 100644 README.md
```

If there is no conflict of changes, git will fast forward the files. Otherwise, you will need to fix the conflicts as indicated in git status manually.

Conclusion

This tutorial provides some basics for version controls for your files and source code. It is recommended for everyone to use at least one of these tools and develop good habits.

If you encounter any questions, please let me know at hcui7@utk.edu.