Using GitHub in 5 Pages

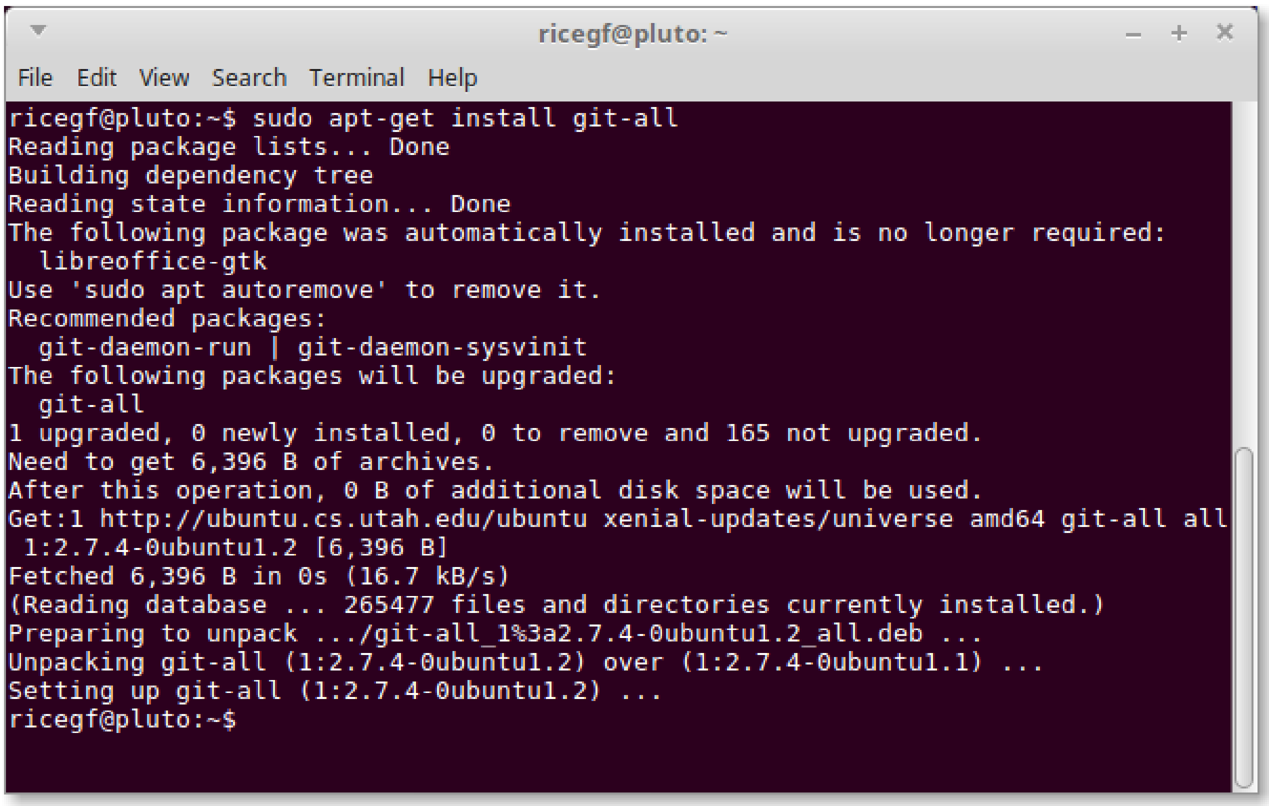
(cloud-hosted version control on the bash command line)

1 Getting Started

GitHub is much like Dropbox, Google Drive, Microsoft OneDrive, or Apple iCloud, but for storing code rather than just documents. You can use a GUI to interact, but we'll learn the command line here.

Installing git

1. If using the CSE-VM VirtualBox appliance, skip to step 2.
   On Linux bash, type **sudo apt install git-all** followed by your password (other distros see here).
   On Windows, download the latest version from https://git-scm.com/download/win.
   On Mac OS X, download the latest version from http://git-scm.com/download/mac,
   or just try to use it and follow the installation prompts.



```
ricegf@pluto:~$ sudo apt-get install git-all
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libreoffice-gtk
Use 'sudo apt autoremove' to remove it.
Recommended packages:
  git-daemon-run | git-daemon-sysvinit
The following packages will be upgraded:
  git-all
1 upgraded, 0 newly installed, 0 to remove and 165 not upgraded.
Need to get 6,396 B of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://ubuntu.cs.utah.edu/ubuntu xenial-updates/universe amd64 git-all all
 1:2.7.4-0ubuntu1.2 [6,396 B]
Fetched 6,396 B in 0s (16.7 kB/s)
(Reading database ... 265477 files and directories currently installed.)
Preparing to unpack .../git-all_1%3a2.7.4-0ubuntu1.2_all.deb ...
Unpacking git-all (1:2.7.4-0ubuntu1.2) over (1:2.7.4-0ubuntu1.1) ...
Setting up git-all (1:2.7.4-0ubuntu1.2) ...
ricegf@pluto:~$
```

2. Configure git with the following commands, using your own name and email address. These commands allow git to credit you with your version management activity in your repositories.

- **git config --global user.name "Prof Rice"**
- **git config --global user.email "george.rice@uta.edu"**

  If you aren't using the provided VirtualBox appliance, where these are already configured, you'll probably also want to request colorful output:

- **git config --global color.ui auto**

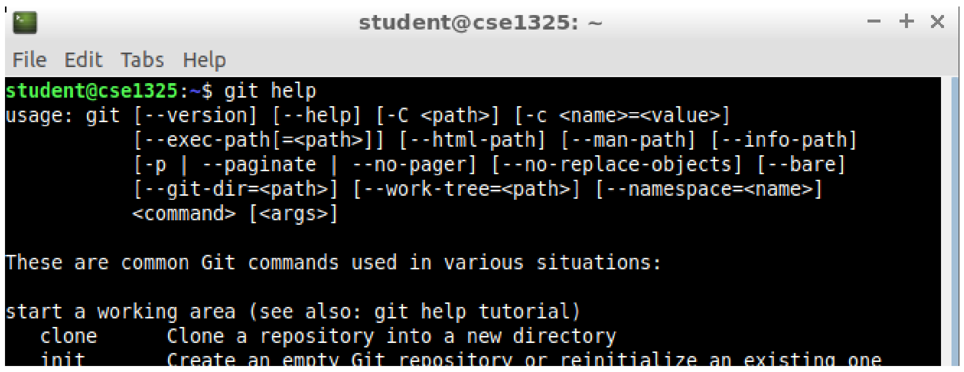
  and (*if not on a shared computer*) for the machine to remember your Personal Access Token until it expires:

- **git config --global credential.helper store**

  (Secure shell (ssh) is also available, but we can't support you with that. Check the last section.)

Getting help with git

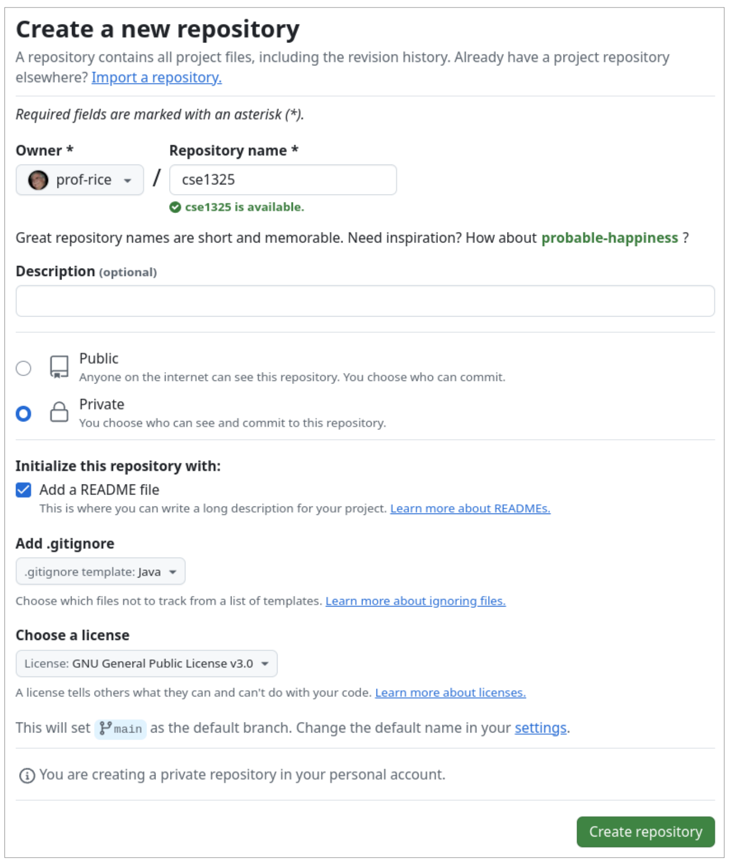
1. **A list of git commands hyperlinked to a wealth of additional documentation and tutorials is available on-line at https://git-scm.com/docs/git.**
2. **git help** lists all common commands.
   git help *command* provides additional help on a command.



```
student@cse1325: ~
File  Edit  Tabs  Help
student@cse1325:~$ git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one
```

Creating a GitHub repository

- Create an account at http://github.com.
- From the GitHub homepage, click **Repositories** > **New**. Name your repository "cse1325". **Select Private.** Enable "Initialize this repository with a README" and set .gitignore to "Java". Set other options as you prefer, and click "**Create repository**".
- Click **Settings** > **Collaborators** > **Add People** and add those listed in Canvas > Announcements.
- Click the green button labeled **Code** > **Local** > **HTTPS** > to copy your URL (see below).
- **In Canvas, add the URL** you just copied *as the entire solution to assignment #1*. We'll remember it in the future, and obtain your assignment solutions from it directly.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * prof-rice / **Repository name *** cse1325

cse1325 is available.

Great repository names are short and memorable. Need inspiration? How about **probable-happiness** ?

Description (optional)

○ Public — Anyone on the internet can see this repository. You choose who can commit.

● Private — You choose who can see and commit to this repository.

Initialize this repository with:

☑ Add a README file — This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: Java

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: GNU General Public License v3.0

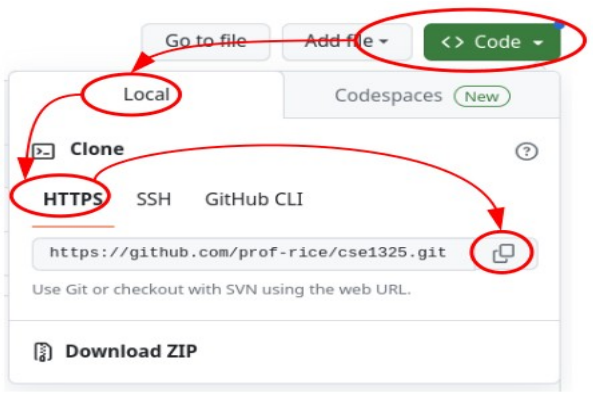
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set main as the default branch. Change the default name in your settings.

You are creating a private repository in your personal account.

Cloning your GitHub repository

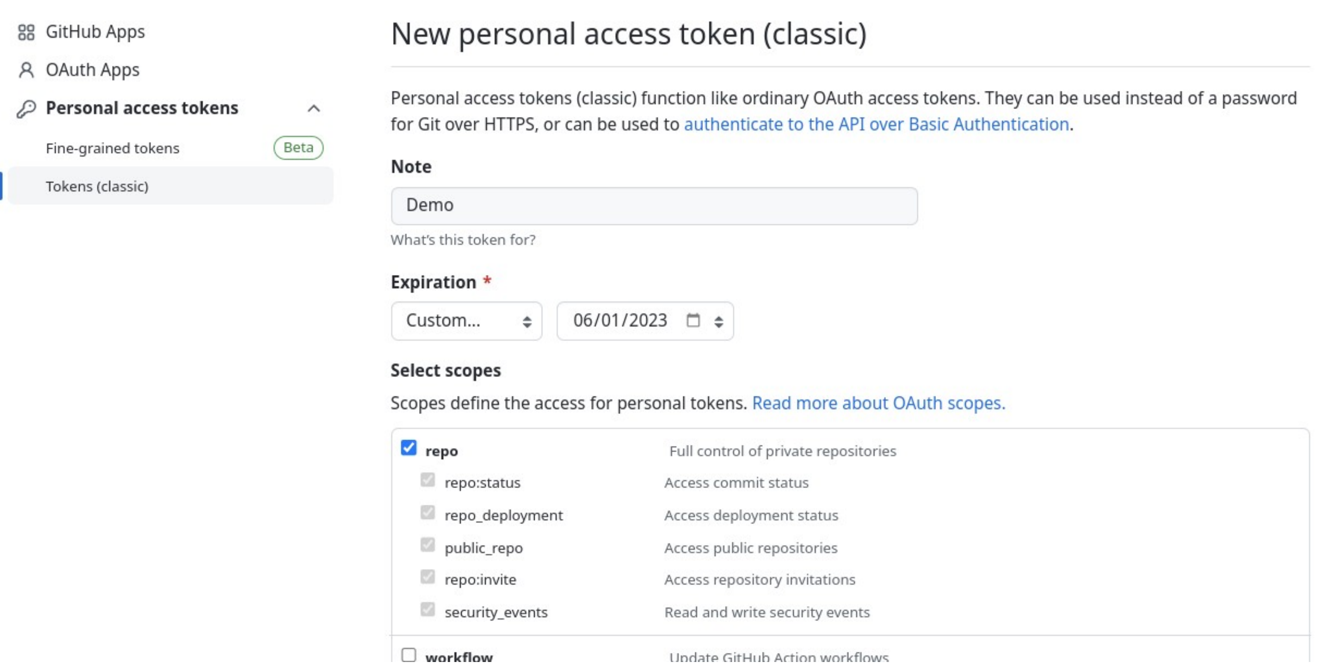
- In bash, in your home directory (**cd**) type **git clone *URL*** where *URL* is what you copied above (right-click > Paste, Ctrl-Shift-v, or Shift-Ins). Pause when prompted for your username – **your GitHub password will NOT work on the bash command line**, so you need a PAT!



```
https://github.com/prof-rice/cse1325.git
```

Use Git or checkout with SVN using the web URL.

- Create a **Personal Access Token (PAT)** on the GitHub website.
  - On the GitHub website, click your account icon in the upper right and select "**Settings**" from the drop-down menu. In the bottom of the left pane, select "**<> Developer Settings**", then "**Personal access tokens**", then "**Tokens (classic)**". Select "**Generate new token**" then "**Generate new token (classic)**" and enter your GitHub password if requested.



New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note: Demo

What's this token for?

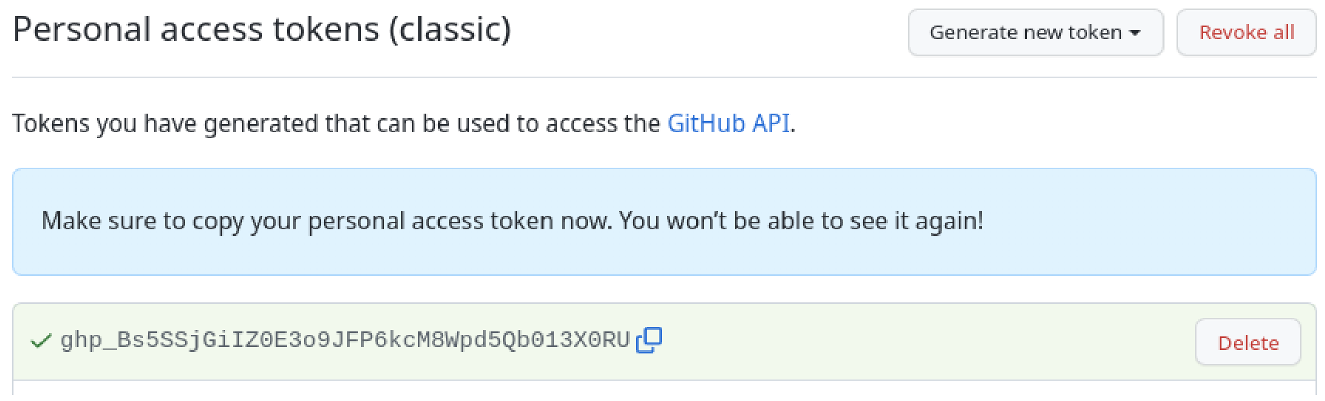
Expiration *: Custom... 06/01/2023

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events
workflow	Update GitHub Action workflows

- Under “**Expiration**” select “**Custom**” and pick a date AFTER the final exam. In Select Scopes click the “**repo**” box (all subscopes will also be selected). Click “**Generate Token**”. Copy the token using . (You can always click Delete for this PAT later and then generate a new one if needed.)



Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

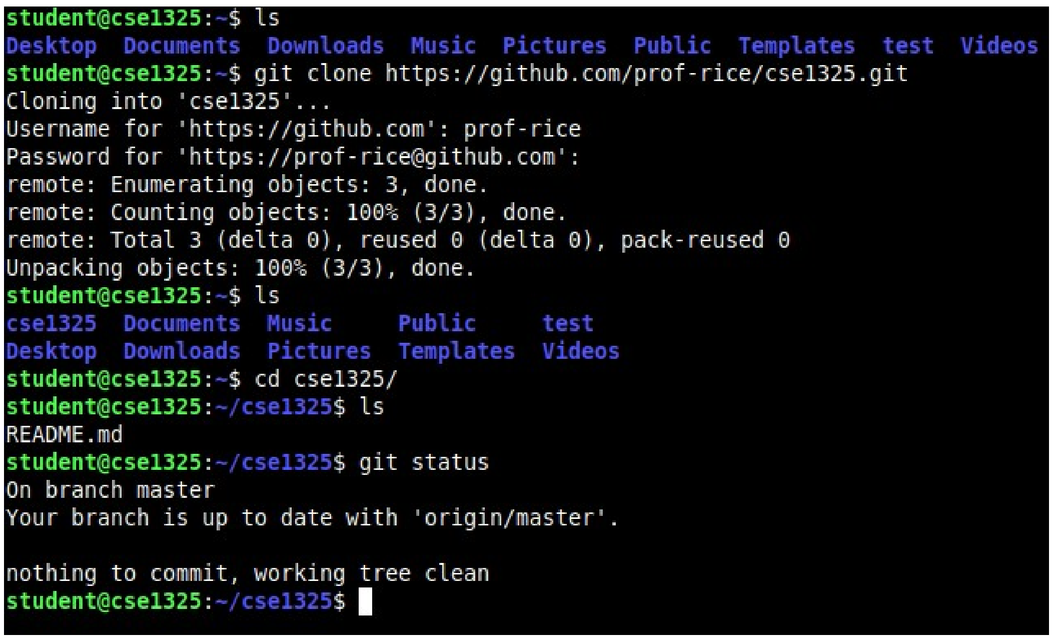
✓ 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

- Return to bash, which should still be waiting for your username. Enter your GitHub username and press Enter. Then paste your Personal Access Token (right-click > paste or Control-Shift-v or Shift-Ins). I**MPORTANT: You will NOT see your PAT on the terminal** – it will appear as if nothing pasted, but it did. **Paste your PAT EXACTLY ONCE** and then press Enter. Your repository should clone into the cse1325 subdirectory.

2 The Basics of Using Git

Creating and adding files with git

- Git stores your code in a hidden directory in the cse1325 directory that you manipulate with commands – the word “git” and a second word such as “status”, “add”, “commit”, or “push”.



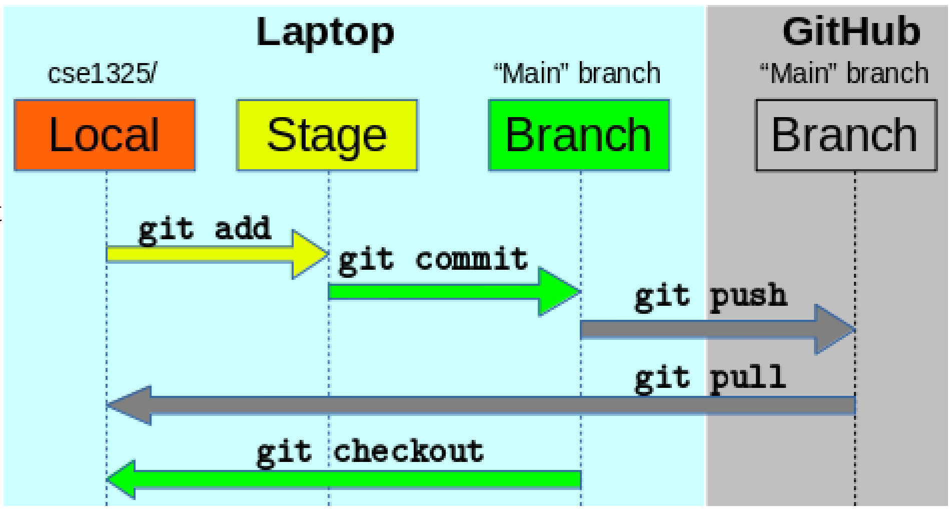
```
student@cse1325:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  test  Videos
student@cse1325:~$ git clone https://github.com/prof-rice/cse1325.git
Cloning into 'cse1325'...
Username for 'https://github.com': prof-rice
Password for 'https://prof-rice@github.com':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
student@cse1325:~$ ls
cse1325  Documents  Music     Public     test
Desktop  Downloads  Pictures  Templates  Videos
student@cse1325:~$ cd cse1325/
student@cse1325:~/cse1325$ ls
README.md
student@cse1325:~/cse1325$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
student@cse1325:~/cse1325$ 
```

- **git status** reports info about tracked and untracked files in the current directory.

  "On branch main" just means that you're working on your main "branch". We won't ask you to do anything with branches this semester.

  - "Your branch is up to date with origin/main" means you're in sync with GitHub.
  - "Changes to be committed" lists the files that are "staged" (if any), meaning git will store them in its repository the next time you issue a **git commit** command (below).
  - "Changes not staged to commit" are files that git is tracking, and have changed, but are not staged to store in the repository – a **git add** command is needed to stage them.
  - "Untracked files" are files that git is ignoring – a **git add** command is needed to track them.

- **git add *files*** tells git to "stage" (make a copy of) the listed files, and be ready to store those copies in the repository at the next commit. **git add -u** stages any updates to tracked files since the last commit, but continues to ignore any untracked files. **git add .** stages the entire directory.



- **git commit -m '*message*'** stores all added files to the *local* repository as a "commit". This is similar to a checkpoint in a game – you can always revert to an earlier commit and try again!
  - Think of a commit as a single version of ALL of your files in the repository. Each commit has a name (its "hash"), e.g., fecb2790d1c8a3a19fed445a017870aabcd577d2, of which thankfully only the first few characters are usually needed, e.g., fecb279.

Pushing to / pulling from GitHub

- **Your repository is on your local disk.** You need to **push** it to GitHub for safety and grading!
- **git push** uploads your commit(s) (your code) from your local repository to GitHub. You can verify the files were uploaded on your project homepage by checking your website at https://github.com/prof-rice/cse1325 (replacing prof-rice with your own GitHub user ID).
- **git pull** downloads and merges changes from GitHub, either changes you made and pushed on another machine, or changes made by the Professor or TA (e.g., new assignments or suggested solutions), into your local repository and directory. If a "merge conflict" is reported, see https://help.github.com/en/articles/resolving-a-merge-conflict-using-the-command-line.

Submitting your homework

- The assignment will be posted on Canvas Assignments, and if indicated, some parts may require cloning or pulling some code from my https://github.com/prof-rice/cse1325-prof repository.
  - Develop your full credit solution in the P01/full_credit subdirectory of your own cse1325 directory, using **git add *files***, **git commit -m "*comment*"** and **git push** for any version of your code worth not losing. That's probably no less often than every 15 minutes!
  - Also develop your bonus and extreme bonus solutions in the bonus and extreme_bonus subdirectories. Add, commit, and push. Once you're done... you're done! We'll grade the latest commit before the assignment deadline, and post your grade in Canvas.
  - Verify that your files are all safely on GitHub by using the https://github.com web interface!

3 More Advanced Git Commands

Exploring the local repository on your disk

- **git log** shows all changes to the local repository (sometimes called the "local branch").
    - **git log --oneline** shows just the name of the commit and your associated commit message. Or (in CSE-VM) use **git lg** to see more *colorful* logs.
- **git diff** lists all changes between your repository and your local directory. **git diff *filename*** compares only the specified file. (Check the documentation for difftool and mergetool to use fancier graphical tools for this.)
    - White text is unchanged (shown for context),
    - Red text preceded by a minus ("-") has been deleted,
    - Green text preceded by a plus ("+") have been added. Note that a *changed* line is represented by a deletion and an addition.

(Slightly) more powerful git commands

- **git add -u** adds all files previously added to the repository that have changed. It also notes removals (otherwise, you'd use **git rm *files*** to remove files). It does *not* add brand new files!
- **git reset *files*** is the 'undo' of add – it tells git to NOT store the listed file(s) that have been added into the repository at the next commit after all. (That is, it removes them from the stage.)
- **git commit --amend -m 'new message'** replaces your *most recent* (only!) commit message with new message. I use this to fix the occasional typo in my most recent commit message.
- **git checkout [*commit*] *file*** will discard the current contents of your local file, and replace it with the same file from the specified commit. This is an *irrevocable* "undo my edits" command for a single file. If [commit] is omitted (and it most often is), the most recent commit is used.

4 Fixing a Bad Repository

If your repository begins to complain, for example, "loose object stored in .git/objects/61/ is corrupt":

1. **Rename your cse1325 directory** to cse1325-bad.
2. **Clone your repository again**, which creates a new working cse1325 directory.
3. **Copy the files you want** to push from cse1325-bad into cse1325.
4. In cse1325, **add, commit, and push the files**. Life is good. Back to the homework!
5. When you're sure you have all the files you need, **delete cse1325-bad like a bad dream.**

5 Expanding your git skills

The git suite has become the most common version control tool among software developers. **You need to know git well.** Here are some resources to get you started.

- For readers, the (free!) official book is at https://git-scm.com/book or from major book sellers.
- If you like videos, try https://www.youtube.com/@GitHub.
- If you prefer interactive tutorials, try https://skills.github.com/.