



Linux and Git Boot Camp

Fernando, Ed

May 24, 2018

Connecting Clients

SSH

Windows users: MobaXterm, PuTTY, SSH Tectia

Mac & Linux users: Terminal (Just type `ssh`)

```
ssh andrewid@shark.ics.cs.cmu.edu
```

I Need You To Make A Directory

```
$ ls
```

```
$ cd private
```

```
$ mkdir 15-213
```

```
$ cd 15-213
```

- All work **MUST** be done in private directory or any subfolder within
- For more information on AFS directories and permission see <https://www.cs.cmu.edu/~help/afs/afshome.html>

File Transfers

- Useful for transferring handins to local machine for submission to Autolab.
- Use MobaXTerm's file transfer dialog if you're on Windows
- On Linux or Mac OS X:

```
$ sftp andrew@shark.ics.cs.cmu.edu:private/15-213
```

```
sftp> help
```

```
(read help for 'cd', 'lcd', 'pwd', 'lpwd', 'get', 'put', etc.)
```

```
$ scp andrew@shark.ics.cs.cmu.edu:private/file.txt /local/folder
```

```
$ scp file.txt andrew@shark.ics.cs.cmu.edu:private/folder
```

Also, you can use FileZilla! Here's a detailed guide:

http://cs.cmu.edu/~213/recitations/using_filezilla.pdf

Continue On...

```
$ ls
```

```
$ cd private
```

```
$ mkdir 15-213
```

```
$ cd 15-213
```

```
$ cd lab-handout (Once obtained from GitHub!)
```

Git



Git Setup (User Information)

```
$ git config --global user.name "<Your Name>"  
$ git config --global user.email <Your email>  
$ git config --global push.default simple
```

Sample Git Repo Creation

In a new folder `(mkdir)`

```
$ git init
```

```
$ echo "a sample file" > readme.txt
```

To save your progress:

```
$ git add readme.txt
```

```
$ git commit -m "my first commit"
```


Git Ignore

For those who want to use

```
git add -all or git add .
```

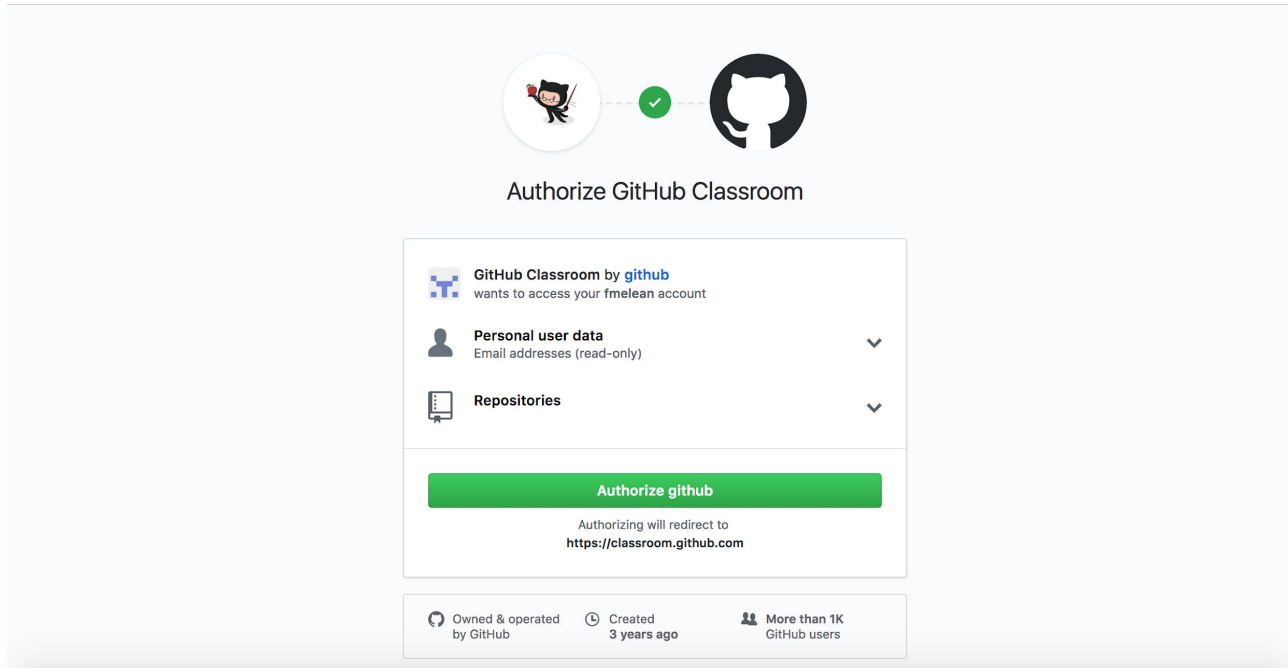
Do not track .o files or executable files!!!

Create a file `.gitignore` in your git repository and add files that you do not want to track

gitignore rules: <https://git-scm.com/docs/gitignore>

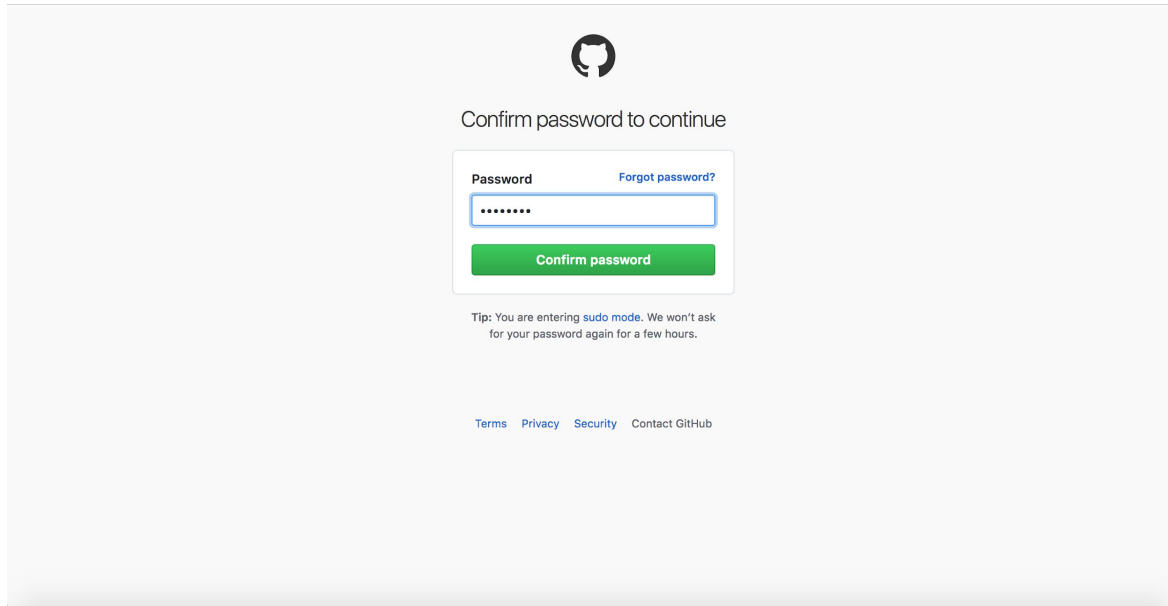
GitHub Setup

Go to link provided. Give access.



GitHub Setup

Enter GitHub Password.



The image shows a GitHub password confirmation screen. At the top center is the GitHub logo. Below it, the text "Confirm password to continue" is displayed. The main form area contains a "Password" label, a "Forgot password?" link, a password input field with seven dots, and a green "Confirm password" button. Below the form, a tip states: "Tip: You are entering sudo mode. We won't ask for your password again for a few hours." At the bottom, there are links for "Terms", "Privacy", "Security", and "Contact GitHub".

Confirm password to continue

Password [Forgot password?](#)

.....

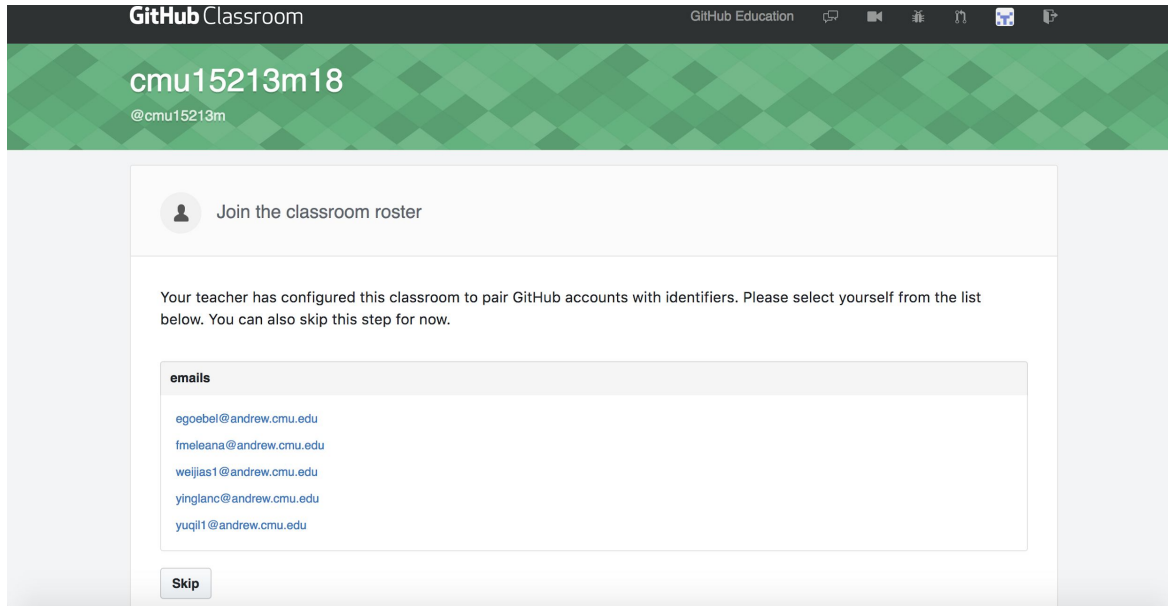
Confirm password

Tip: You are entering [sudo mode](#). We won't ask for your password again for a few hours.

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

GitHub Setup

Select *YOUR* andrew email.



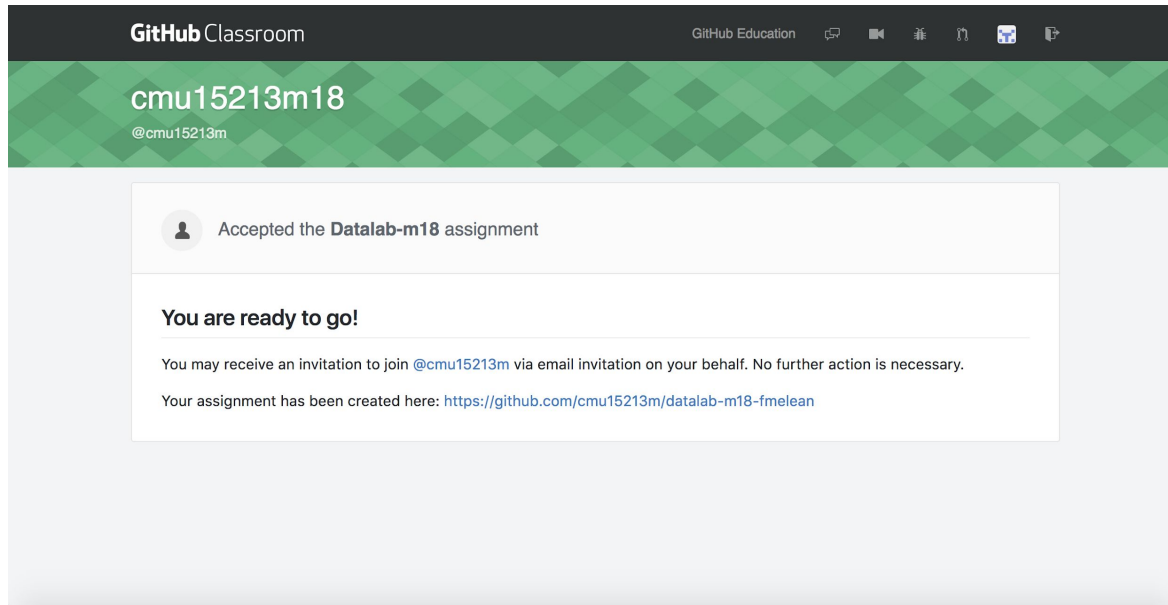
The screenshot shows the GitHub Classroom interface. At the top, the header reads "GitHub Classroom" and "GitHub Education". Below the header, the classroom name "cmu15213m18" and the user handle "@cmu15213m" are displayed. A button labeled "Join the classroom roster" is visible. Below this, a message states: "Your teacher has configured this classroom to pair GitHub accounts with identifiers. Please select yourself from the list below. You can also skip this step for now." A list of email addresses is provided under the heading "emails":

- egoebel@andrew.cmu.edu
- fmeleana@andrew.cmu.edu
- wejjas1@andrew.cmu.edu
- yinglanc@andrew.cmu.edu
- yuqil1@andrew.cmu.edu

A "Skip" button is located at the bottom left of the email list.

GitHub Setup

Reach this screen. If not, raise hand for help.



GitHub Setup

Now you should have access to a repo like this. Click lower link to get there from previous image.

The screenshot shows a GitHub repository page for 'cmu15213m / datalab-m18-fmelean'. The repository is private and has 1 watch, 0 stars, and 0 forks. It was created by GitHub Classroom. The repository has 3 commits, 1 branch, 0 releases, and 1 contributor. The current branch is 'master'. There are buttons for 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history is as follows:

Commit	Message	Time
bprail	Revision for support for Summer	Latest commit c689d8 an hour ago
bddcheck	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
Driverhdrs.pm	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
Driverlib.pm	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
Makefile	Revision for support for Summer	an hour ago
README	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
bits.c	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
bits.h	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
btest.c	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
btest.h	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago

Set up SSH Keys

First check if you already have an ssh key:

```
$ cat ~/.ssh/id_rsa.pub should print a string
```

If not:

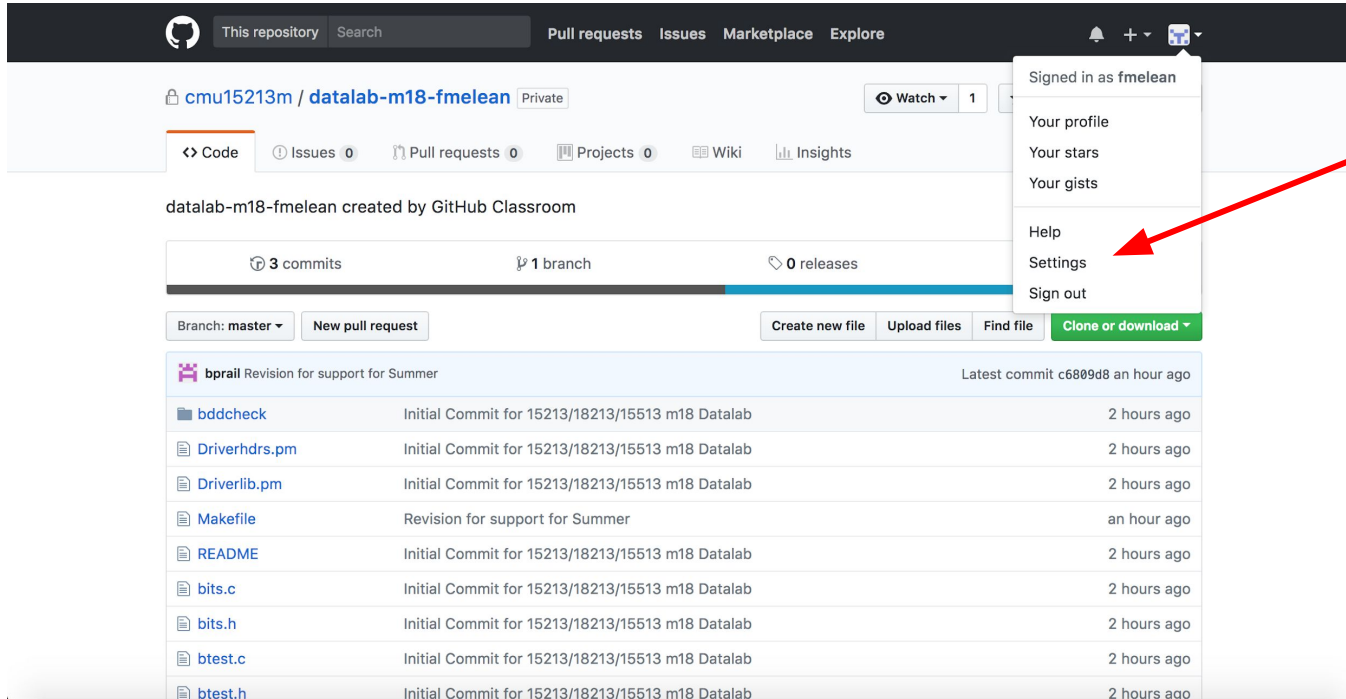
```
$ ssh-keygen -t rsa -C "213GitHub" -b 4096
```

Use the default file path (press Enter).

Optionally type in a password. (press Enter for no password)

Unlocking Your Github Repo

Enter Github Settings

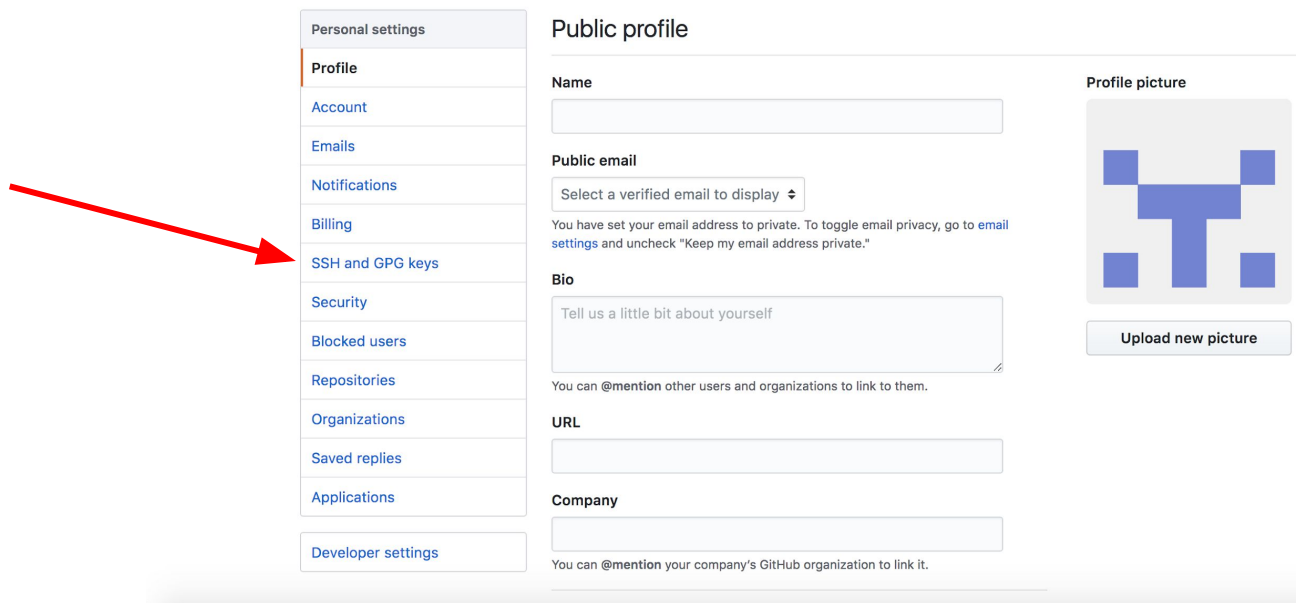


The screenshot shows the GitHub interface for a repository named 'cmu15213m / datalab-m18-fmelean'. The user is signed in as 'fmelean'. The user menu is open, showing options: 'Signed in as fmelean', 'Your profile', 'Your stars', 'Your gists', 'Help', 'Settings', and 'Sign out'. A red arrow points to the 'Settings' option. The repository page shows 3 commits, 1 branch, and 0 releases. The commit history table is as follows:

Commit	Message	Time
bprail	Revision for support for Summer	Latest commit c6809d8 an hour ago
bddcheck	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
Driverhdrs.pm	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
Driverlib.pm	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
Makefile	Revision for support for Summer	an hour ago
README	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
bits.c	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
bits.h	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
btest.c	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago
btest.h	Initial Commit for 15213/18213/15513 m18 Datalab	2 hours ago

Unlocking Your Github Repo

Select SSH and GPG keys from left-side panel.



The image shows a screenshot of the GitHub account settings page. On the left, there is a sidebar with a list of settings categories. A red arrow points to the 'SSH and GPG keys' option, which is highlighted in blue. The main content area is titled 'Public profile' and contains several sections: 'Name' with an input field, 'Public email' with a dropdown menu and a note about email privacy, 'Bio' with a text area, 'URL' with an input field, and 'Company' with an input field. On the right side, there is a 'Profile picture' section with a placeholder image and an 'Upload new picture' button.

Personal settings

- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys
- Security
- Blocked users
- Repositories
- Organizations
- Saved replies
- Applications

Developer settings

Public profile

Name

Public email

Select a verified email to display ▾

You have set your email address to private. To toggle email privacy, go to [email settings](#) and uncheck "Keep my email address private."

Bio

Tell us a little bit about yourself


You can @mention other users and organizations to link to them.

URL

Company

You can @mention your company's GitHub organization to link it.

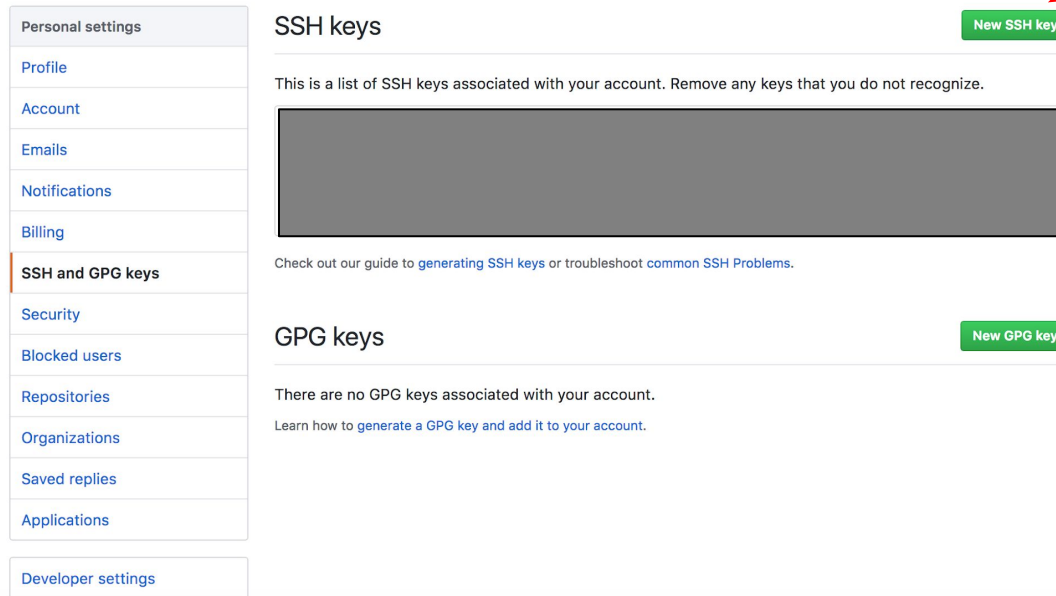
Profile picture



Upload new picture

Unlocking Your Github Repo

Click New SSH Key button.



The screenshot shows the GitHub account settings page. On the left is a sidebar menu with options: Personal settings, Profile, Account, Emails, Notifications, Billing, SSH and GPG keys (highlighted), Security, Blocked users, Repositories, Organizations, Saved replies, Applications, and Developer settings. The main content area is titled 'SSH keys' and includes a green 'New SSH key' button. Below the title is a message: 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' A large grey rectangular box is present where the list of keys would be. Below this box is a link to a guide on generating SSH keys and troubleshooting common problems. At the bottom of the main content area, there is a section for 'GPG keys' with a green 'New GPG key' button and a message stating 'There are no GPG keys associated with your account.' and a link to learn how to generate a GPG key.

Unlocking Your Github Repo

From your terminal, type:

```
$ cat ~/.ssh/id_rsa.pub
```

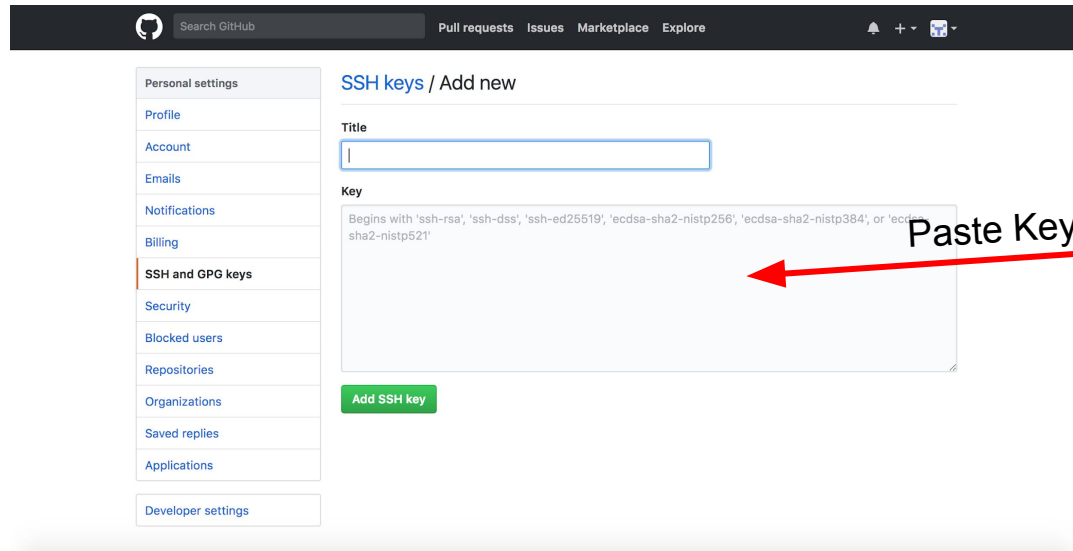
Your public key will be printed.

Highlight it with the mouse and copy

Unlocking Your Github Repo

Give a title and paste *entire* SSH key.

Should start with 'ssh-rsa' and end with '213Github'.



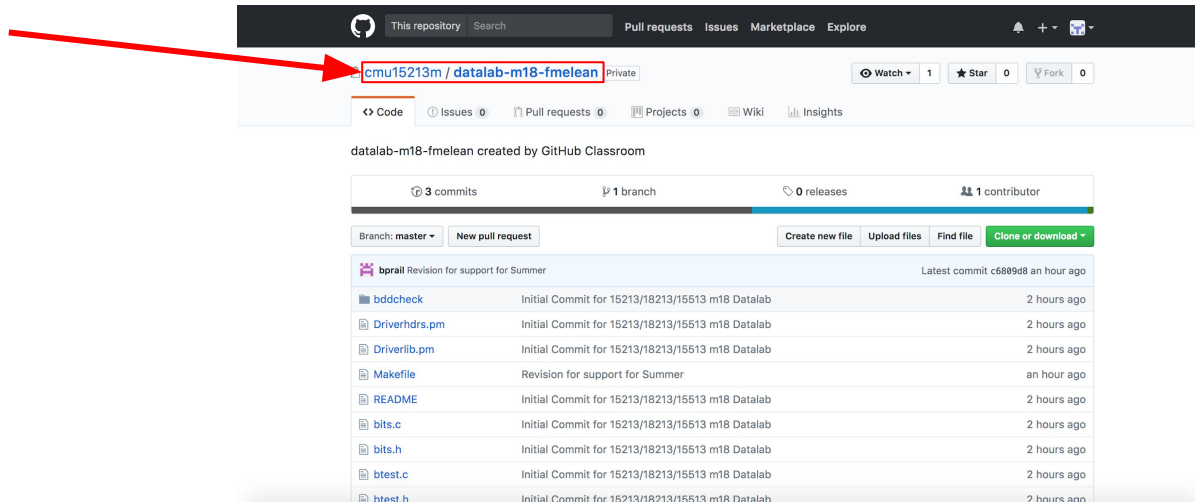
The screenshot shows the GitHub 'SSH keys / Add new' page. On the left is a sidebar with 'Personal settings' and various sub-sections, with 'SSH and GPG keys' highlighted. The main content area has a 'Title' input field and a 'Key' text area. A red arrow points from the text 'Paste Key in this box.' to the 'Key' input field. Below the key field is a green 'Add SSH key' button.

Accessing Your Github Repo

From your **private** 15213 directory, type the following:

```
$ git clone git@github.com:<directory path>.git
```

Image below shows where to find <directory path>:



Initial Commit

Enter cloned directory and do following:

```
$ echo "turtwig" > myteam.txt
```

```
$ git add myteam.txt
```

```
$ git commit -m "initial commit"
```



Push your git commit

```
$ git push -u origin master
```

You should now be able to see your updated repo in
GitHub

`$ git push` is all that is required for future pushes in
the same folder

Updating Repo (Essentials)

When you edit a file:

```
$ git add <file-name>
```

When you want to save a version locally

```
$ git commit -m "version-name"
```

When you want to backup your commits in the Git Repo:

```
$ git push
```


Git Commands

<code>add</code>	Stage new or changed files	<code>rebase</code>	Modify, combine, delete, ... previous commits
<code>commit</code>	Save current staged files	<code>merge</code>	Combine commits from specified branch into current branch
<code>push/pull</code>	Push/pull local index to/from the remote server	<code>checkout</code>	Examine a different commit/branch/file
<code>log</code>	Show history of git commits	<code>stash</code>	Temporarily save your current uncommitted changes
<code>status</code>	Shows working directory status (added/modified/deleted files)	<code>stash pop</code>	Restore previously stashed changes
<code>show</code>	Show a file from a different commit or branch	<code>diff</code>	Show changes between commits, files, unstaged changes, ...
<code>branch</code>	Create a new branch (use a new branch for experimenting safely)	<code>clone</code>	Clone a git repository (like a remote GitHub repo)

More Git

Getting help:

- `git help <command>`
- Piazza/Office hours

Git tutorials:

- <https://www.atlassian.com/git/tutorials> (focused tutorials)
- <https://try.github.io> (basic interactive introduction)

Terminal Shortcuts

The command line operates on one directory at a time (the “working directory”).

You can use these shortcuts whenever a directory or file path is expected.

	Meaning	Example
~	Home directory	<code>cp foo.txt ~</code>
.	Working (current) directory	<code>cp ~/foo.txt .</code>
..	Parent directory	<code>cp ~/foo.txt ..</code>
-	Previous directory	<code>cd -</code>
*	Match as many characters as possible	<code>cp ~/*.txt</code> <code>rm *.c</code>

- **Be very very very careful with `rm`!!!**
 - **There is no trash with `rm`. It is gone.**

More Terminal Shortcuts

- Pressing tab will autocomplete file/directory names.
- Use the up+down arrow keys to scroll through your previous commands.
- Control+R lets you search your command history.
- Control+A jumps to the beginning of the line.
- Control+E jumps to the end of the line.
- Control+U clears everything to the left of the cursor.
- Control+C kills your current program.
- Control+D (on a blank line) exits the terminal.
- Control+L clears your screen.

```
ls <dir>
```

- Lists files in the present working directory, or, if specified, `dir`.
 - `-l` lists ownership and permissions.
 - `-a` shows hidden files (“dotfiles”).
- `pwd` tells you your present working directory.

```
cd <directory>
```

- Try running `cd -` to return to the previous directory.
- Try running `cd ..` to return to the parent directory.
- Changes your present working directory.

```
mkdir <dirname>
```

- Makes a directory `dirname` in your present working directory.
- Directories and folders are the **same thing!**

```
mv <src> <dest>
```

- `cp` works in exactly the same way, but copies instead
 - for copying folders, use `cp -r`
- `dest` can be into an existing folder (preserves name), or a file/folder of a different name
- `src` can be either a file or a folder


```
tar <options> <filename>
```

- For full list of options, see `man tar`
- `tar` stands for **t**ape **a**rchive. Was used on tapes!
- `x` - extract, `v` - verbose, `f` - file input, `p` - keep perms

```
rm <file1> <file2> ... <filen>
```

- To remove an (empty) directory, use `rmdir`
- To remove a folder and its contents, use `rm -rf`
 - **Please be careful, don't delete your project.**
 - **There is no "Trash" here. It's gone.**
 - **Contact ugradlabs@cs.cmu.edu to restore.**
 - **Latest restore is up to a day old!**
- **Restore most recent version yourself if you use git!**

pipes and redirects

- A *pipe* redirects output from one program as input to another program.
 - Ex1: `cat filename | outputfile`
 - Ex2: `cat filename | grep 15213`
 -
- Can *redirect* output to a file.
 - Ex3: `echo hello > file.txt`
 - Ex4: `echo hello >> file.txt`

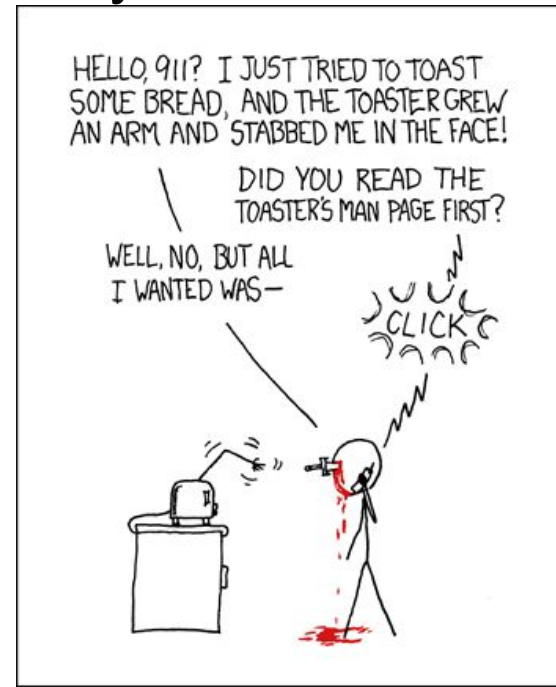
What's in a file? (using `cat`)

- `cat <file1> <file2> ... <filen>` lets you display the contents of a file in the terminal window.
 - Use `cat -n` to add line numbers!
- You can *combine* multiple files into one!
 - `cat <file1> ... <filen> >> file.txt`
- Good for seeing what's in small files.



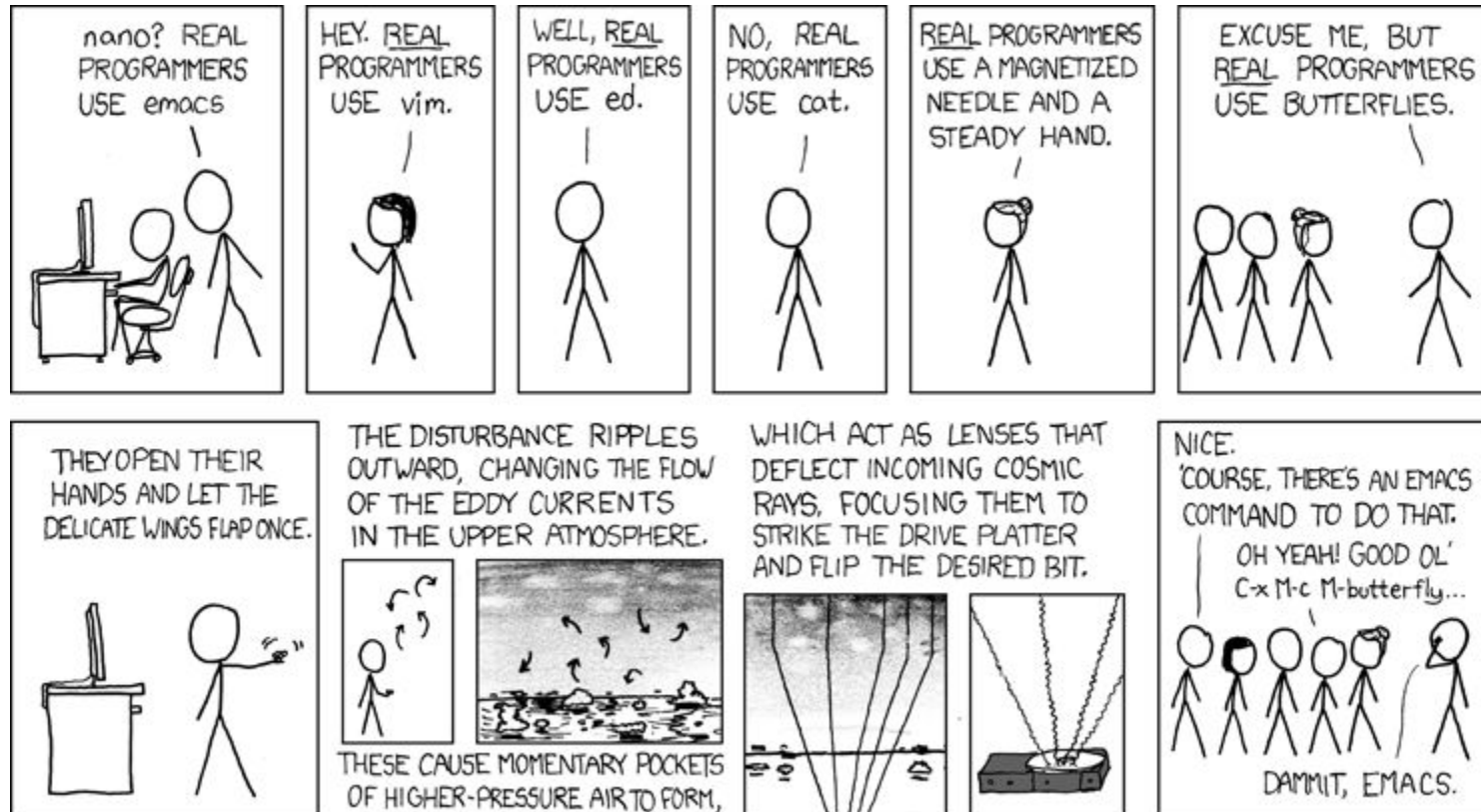
man <thing>

- What is that command? What is this C standard library function? What does this library do?
- Try it!
 - man grep
 - man tar
 - man strlen
 - man 3 printf
 - man stdio.h
 - man man



Appendix

Editors (a touchy subject)



Editors (a touchy subject)

- `vim` is nice, made for very powerful text editing
 - Try running `vimtutor` to get started learning
- `emacs` is nice, made to be more versatile
 - Emacs tutorial in emacs: “Ctrl-h t”
- `gedit` has a GUI
 - Requires X Forwarding: See Appendix
- I **strongly** recommend editing on the terminal.
- **Gist**: Use an editor with auto-indent and line numbers

Configuring bash

The file `~/ .bashrc` is run every time you log in.

Put the following code:

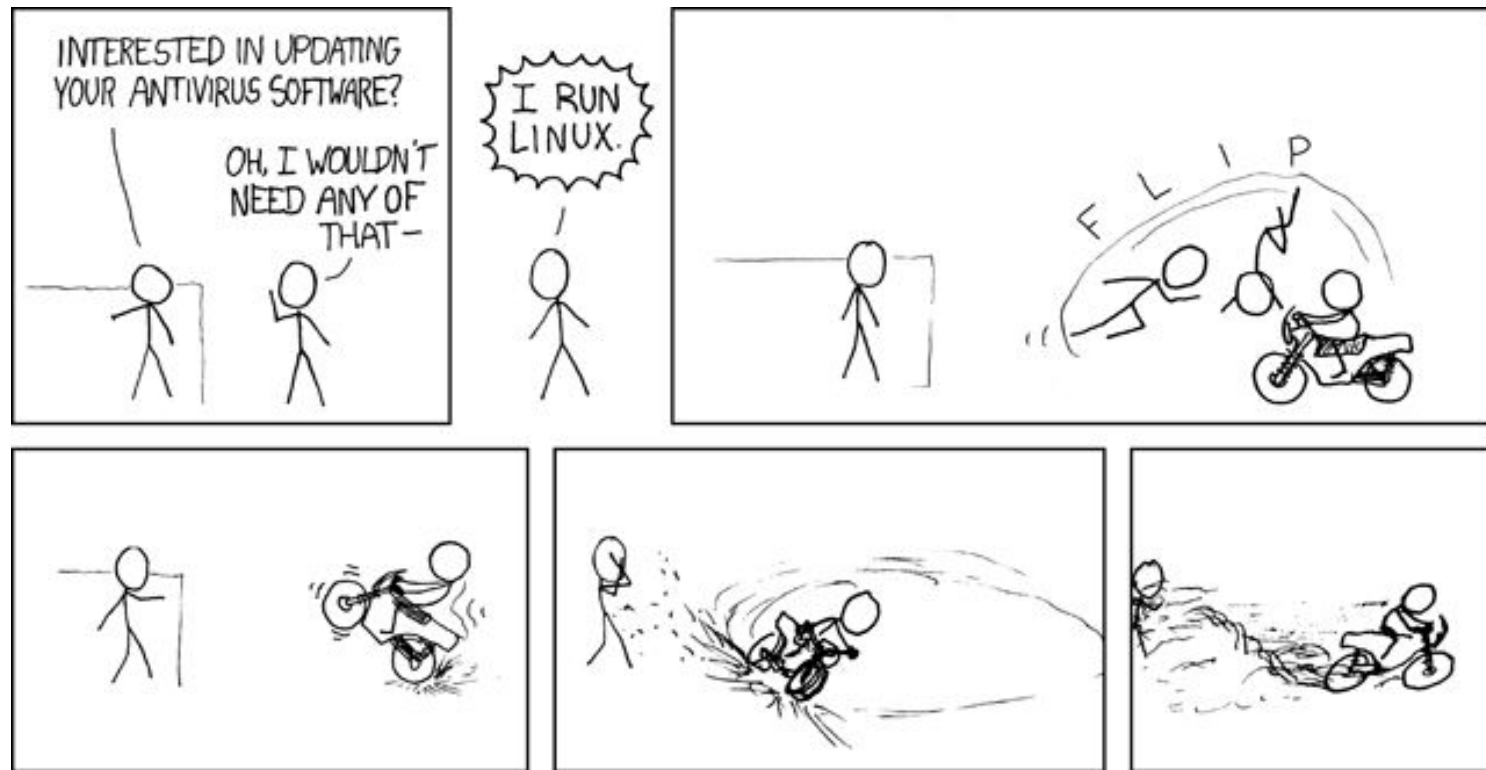
```
PS1=" [\u@\h:\w] \$ "  
alias ls='ls --color=auto'
```

to change your prompt to look like:

```
[szz@makoshark:~/private/15213] $ ls  
attacklab  bomblab  lab-answers
```

Commands related to 15-213

- `gdb`, the **GNU Debugger**, will be used for bomb lab.
- `objdump` displays the symbols in an executable.
- `gcc` is the **GNU C Compiler**.
- `make` is a configurable build system often used for compiling programs.
- We will provide other tools in the handouts as well



Vimtutor Walkthrough

- Chapters 1-3
- Cheatsheet: <http://bit.ly/2c101J0>

Resources

- Quick references: cs.cmu.edu/~213/resources.html
- CMU Computer Club
 - www.contrib.andrew.cmu.edu/~sbaugh/emacs.html
 - club.cc.cmu.edu/talks/fall15/power-vim.html
 - club.cc.cmu.edu/talks/fall15/power-git.html
- Great Practical Ideas
 - www.cs.cmu.edu/~15131/f15/topics/bash/
 - www.cs.cmu.edu/~15131/f15/topics/git/
- Official manuals
 - `info bash`
 - `info emacs`
 - `:help` in Vim

tmux

```
$ tmux
```

Ctrl+b, then c: create a new tab

Ctrl+b, then n: move to next tab

Ctrl+b, then p: move to previous tab

Ctrl+b, then x: kill the current tab

Ctrl+b, then ?: help

Ctrl+b, then |: split horizontal

Ctrl+b, then %: split vertical

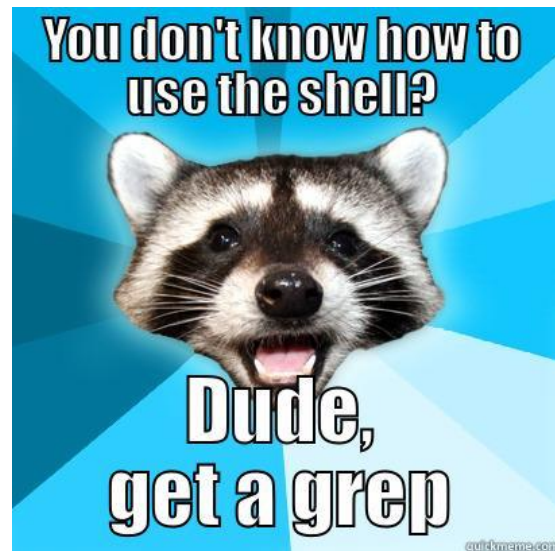
Ctrl+b, then arrow keys: move between panes

Fancy Terminal Shortcuts

- Bash automatically splits things up in brackets!
 - Ex: `cp foo{1,2}.txt = cp foo1.txt foo2.txt`
 - Ex: `cp foo.txt{,.bak} = cp foo.txt foo.txt.bak`
 - For when typing the same filename gets annoying
- Bash has `for` loops!
 - Ex: Append “15-213” to every file ending in `.c`
`for file in *.c; do echo “15-213” >> $file; done`
- Have fun, but don’t break things or lose track of time

What's in a file? (using `grep`)

- `grep <pattern> <file>` will output any lines of file that have `pattern` as a substring
 - `grep -v` will output lines *without* `pattern` as substring
 - `grep -n` prints line numbers
 - `grep -R` will search *recursively*
- Try it: `grep 'phase' bomb.c`
 - `grep -n 'printf' src.c`
 - `grep -R 'unsigned' .`



Looking for something? `grep -A -B`

```
~/test
✓ $ ls
bar.txt  foo.txt  foobar.txt
~/test
✓ $ ls | grep foo
foo.txt
foobar.txt
~/test
✓ $ ls | grep bar
bar.txt
foobar.txt
~/test
✓ $ ls | grep foo > file.txt
~/test
✓ $ cat file.txt
foo.txt
foobar.txt
```

- `grep -B <x>`: include x lines **Before** match.
- `grep -A <y>`: include y lines **After** match.
- Ex: `objdump -d | grep -A 25 explode_bomb`
- Ex: `grep -B 20 return *.c`