Linux/Git Bootcamp

15-213/513 TAs

Outline

1. Getting connected to the shark machines
2. Transferring files between your computer and the shark machines
3. Exploring and organizing your files on the shark machines
4. Editing files and building code on the shark machines
5. Using git to save your work history

Getting Connected

Setup bundle/examples for today:

https://cs.cmu.edu/~213/activities/gitbootcamp.zip

(Link also available in course schedule, next to the slides for today)

1. Download it onto **your local computer** (using your browser, probably).
2. Extract the files to your **Desktop**.

(We'll get the appropriate files onto AFS in a bit.)

Getting Connected

What’s in the bundle?

- “KiTTY” - an SSH client for **Windows**
  - If you’re **not** using Windows, feel free to delete this now
- “gitbootcamp-example” - example project

Goals for today:

1. Connect to the shark machines
2. Transfer the “gitbootcamp-example” project onto AFS
3. Edit and build the code in the example
4. Create a git repository and push your changes to a repo on GitLab

Getting Connected

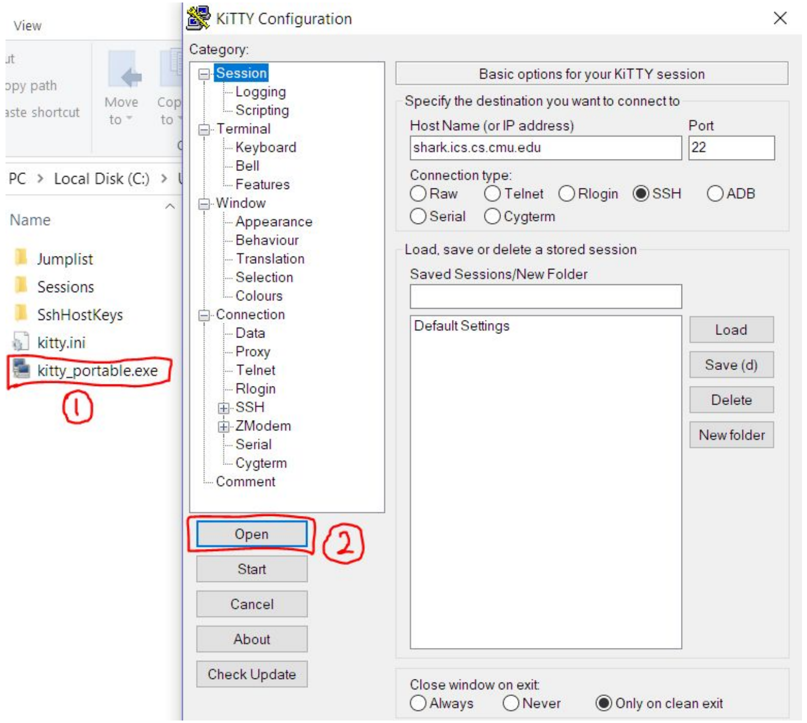
Windows: Inside the "KiTTY" folder:

1. Open "kitty_portable.exe"
2. Click "Open" to start the connection

Mac/Linux:

1. Open your terminal
2. Run the following command (replace "andrewid" with yours):

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



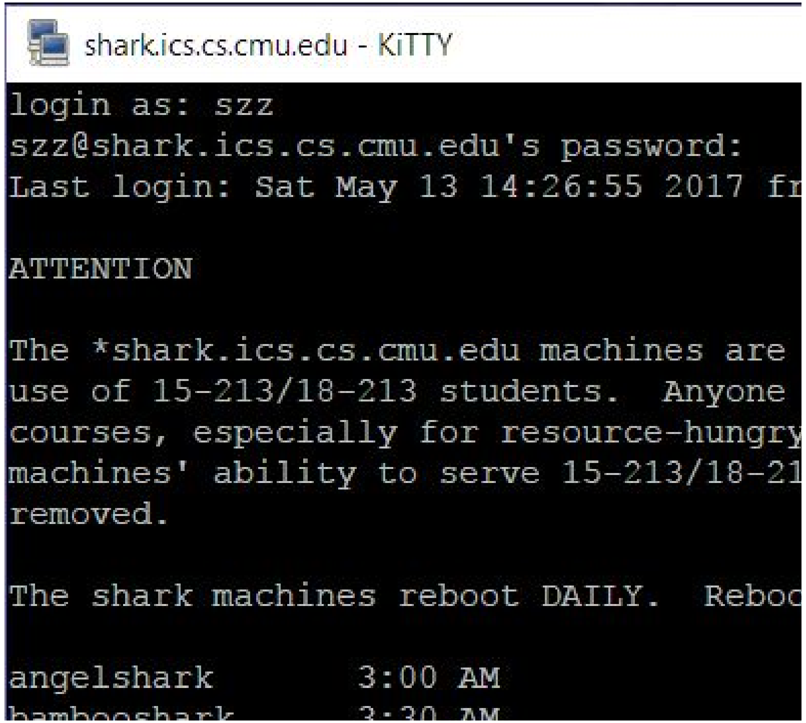
Specify the destination you want to connect to
Host Name (or IP address): shark.ics.cs.cmu.edu
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] ADB Serial Cygterm
Close window on exit: Always Never [✓ Only on clean exit]

Getting Connected

You should see something like the image on the right.

Enter your **username** (Windows) and **password** when prompted (it's normal if nothing shows up as you type).

If you see a warning about SSH host keys, just click or enter "yes".



```
login as: szz
szz@shark.ics.cs.cmu.edu's password:
Last login: Sat May 13 14:26:55 2017 fr

ATTENTION

The *shark.ics.cs.cmu.edu machines are
use of 15-213/18-213 students.  Anyone
courses, especially for resource-hungry
machines' ability to serve 15-213/18-21
removed.

The shark machines reboot DAILY.  Reboo

angelshark        3:00 AM
bambooshark       3:30 AM
```

Transferring Files

Going over using FileZilla, but a couple other ways to transfer files is

Windows Users: Kitty

```
$ sftp andrewid@shark.ics.cs.cmu.edu
```

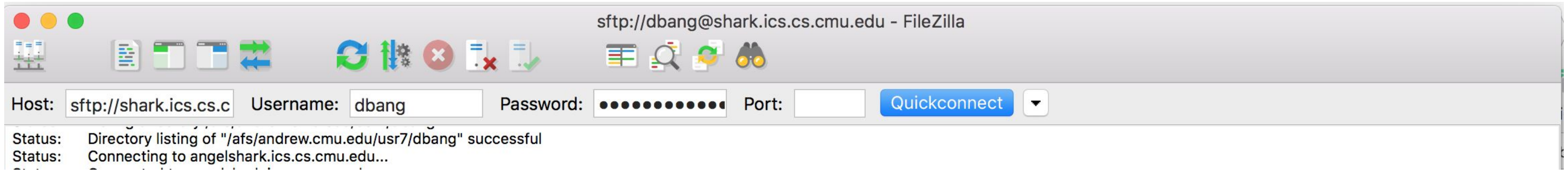
https://www.digitalocean.com/community/tutorials/how-to-use-sftp-to-securely-transfer-files-with-a-remote-server#transferring-files-with-sftp

```
$ scp <filepath> andrewid@shark.ics.cs.cmu.edu:<filepath>
```

Transferring Files

Download and install the FileZilla client: https://filezilla-project.org/download.php

You don't have to configure FileZilla, so you can start directly working with the program.


sftp://dbang@shark.ics.cs.cmu.edu - FileZilla
Host: sftp://shark.ics.cs.c
Username: dbang
Password: ●●●●●●●●●●●●
Port:
Quickconnect

```
Status:    Directory listing of "/afs/andrew.cmu.edu/usr7/dbang" successful
Status:    Connecting to angelshark.ics.cs.cmu.edu...
```

In the QuickConnect bar at the top, enter the hostname, username, and password

Hostname: shark.ics.cs.cmu.edu

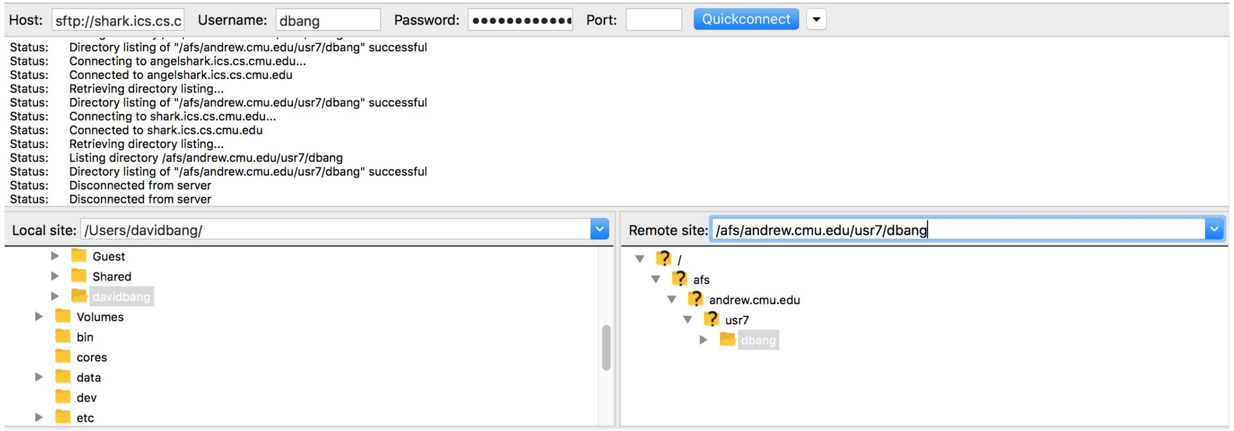
Username: andrewid (replace with your id)

Password: your password

Port: 22

Transferring Files

Navigating on the server



Host: sftp://shark.ics.cs.c Username: dbang Password: ●●●●●●●●●●●● Port: Quickconnect

```
Status:    Directory listing of "/afs/andrew.cmu.edu/usr7/dbang" successful
Status:    Connecting to angelshark.ics.cs.cmu.edu...
Status:    Connected to angelshark.ics.cs.cmu.edu
Status:    Retrieving directory listing...
Status:    Directory listing of "/afs/andrew.cmu.edu/usr7/dbang" successful
Status:    Connecting to shark.ics.cs.cmu.edu...
Status:    Connected to shark.ics.cs.cmu.edu
Status:    Retrieving directory listing...
Status:    Listing directory /afs/andrew.cmu.edu/usr7/dbang
Status:    Directory listing of "/afs/andrew.cmu.edu/usr7/dbang" successful
Status:    Disconnected from server
Status:    Disconnected from server
```

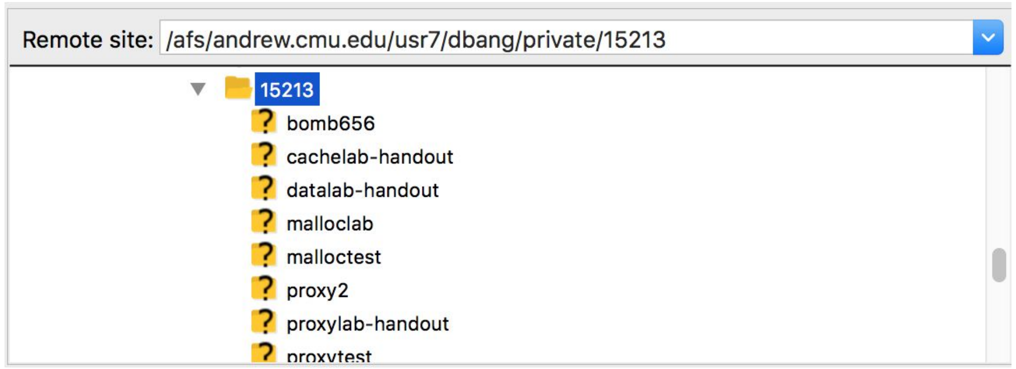
Local site: /Users/davidbang/

Remote site: /afs/andrew.cmu.edu/usr7/dbang

Left Side has your local directory tree

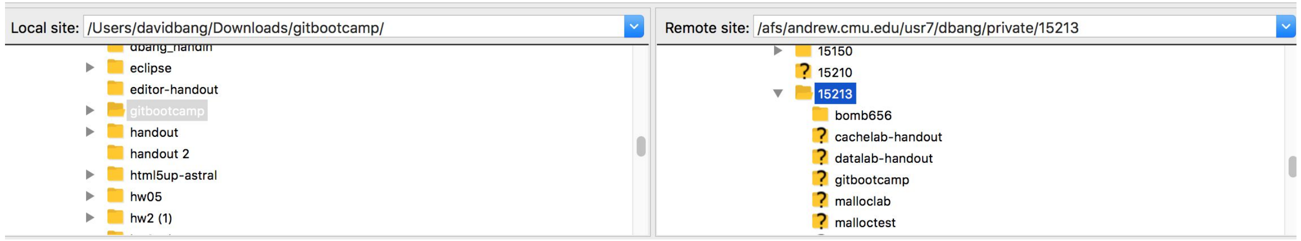
Right Side has has remote server directory tree

Transferring Files

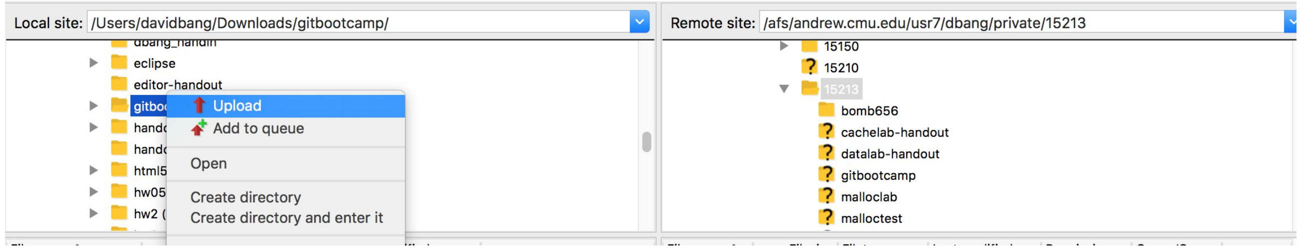

Remote site: /afs/andrew.cmu.edu/usr7/dbang/private/15213

Double Click Folder Tabs or type the directory name in edit field

Transferring Files


Local site: /Users/davidbang/Downloads/gitbootcamp/

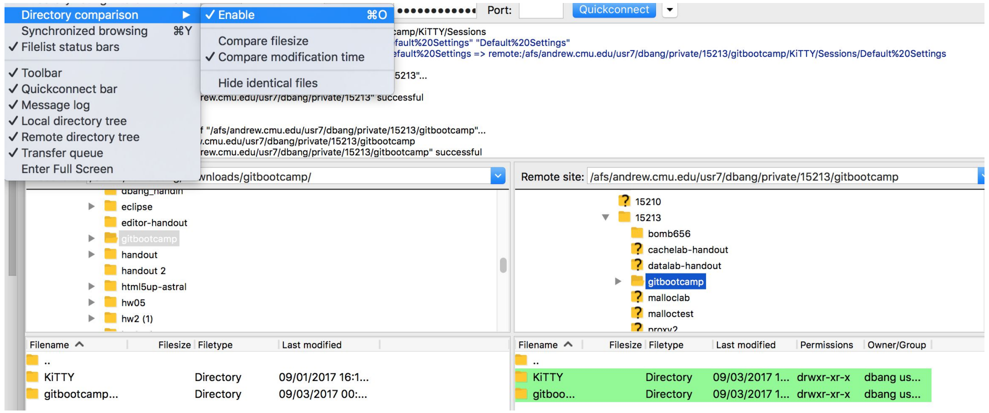
Remote site: /afs/andrew.cmu.edu/usr7/dbang/private/15213


Local site: /Users/davidbang/Downloads/gitbootcamp/

Remote site: /afs/andrew.cmu.edu/usr7/dbang/private/15213

To transfer files you can right click on the file and upload or you can drag and drop it into the appropriate directory in the remote server. Transferring files work both ways from the local machine to the remote server and vice versa.

Transferring Files



- Directory comparison ▶
- Synchronized browsing ⌘Y
- ✓ Filelist status bars
- ✓ Toolbar
- ✓ Quickconnect bar
- ✓ Message log
- ✓ Local directory tree
- ✓ Remote directory tree
- ✓ Transfer queue
- Enter Full Screen

- ✓ Enable ⌘O
- Compare filesize
- ✓ Compare modification time
- Hide identical files

```
camp/KiTTY/Sessions
fault%20Settings" "Default%20Settings"
efault%20Settings => remote:/afs/andrew.cmu.edu/usr7/dbang/private/15213/gitbootcamp/KiTTY/Sessions/Default%20Settings
5213"...
rew.cmu.edu/usr7/dbang/private/15213" successful
f "/afs/andrew.cmu.edu/usr7/dbang/private/15213/gitbootcamp"...
.cmu.edu/usr7/dbang/private/15213/gitbootcamp
rew.cmu.edu/usr7/dbang/private/15213/gitbootcamp" successful
```

wnloads/gitbootcamp/

Remote site: /afs/andrew.cmu.edu/usr7/dbang/private/15213/gitbootcamp

Filename	Filesize	Filetype	Last modified
..			
KiTTY		Directory	09/01/2017 16:1...
gitbootcamp...		Directory	09/03/2017 00:...

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
KiTTY		Directory	09/03/2017 1...	drwxr-xr-x	dbang us...
gitboo...		Directory	09/03/2017 1...	drwxr-xr-x	dbang us...

Under the View Tab: Directory Comparison

Identical directories and files in the local computer and remote server: NOT highlighted at all.

Directories and files with the same name in the local computer and remote server but with different time of the last change: GREEN highlighting shows the most recent version

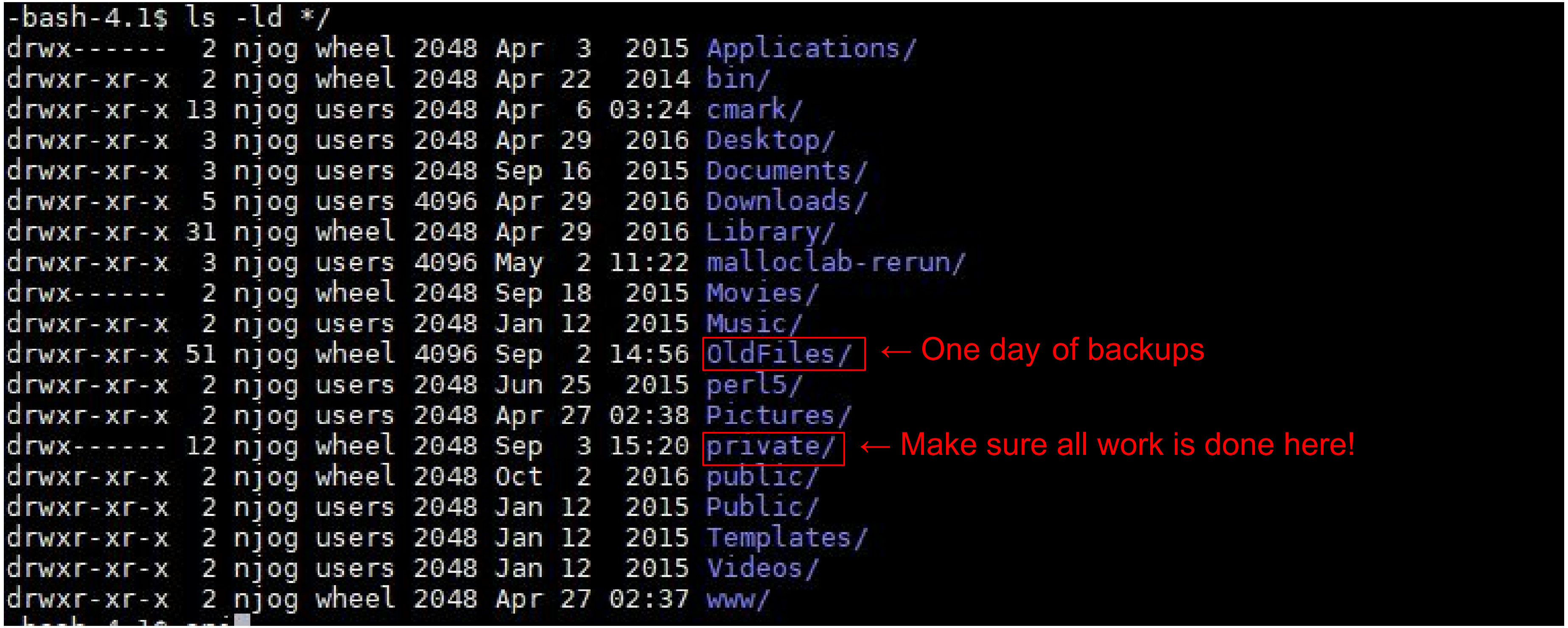
Exploring the File System

`ls <dir>`
`-l`
`-a`
	List all files in current directory
	- List permissions
	- Show hidden files (“dotfiles”)
		`tar <opt> <name>`
		`x`
		`v`
		`f`
			File Archive Utility
			- E**x**tract
			- **V**erbose
			- **F**ile

`cd <dir>`	Go to a directory - ~ → Home Directory - .. → Parent Directory - . → Current Directory	`rm <file>`	Delete a file - No Trash! - OldFiles
`mkdir <name>`	Make a new directory	`cat <file>`	Output file to terminal

`mv <src> <dest>`
`cp <src> <dest>`
	Move or Copy file from src to dest		

Exploring the File System



```
-bash-4.1$ ls -ld */
drwx------  2 njog wheel 2048 Apr  3  2015 Applications/
drwxr-xr-x  2 njog wheel 2048 Apr 22  2014 bin/
drwxr-xr-x 13 njog users 2048 Apr  6 03:24 cmark/
drwxr-xr-x  3 njog users 2048 Apr 29  2016 Desktop/
drwxr-xr-x  3 njog users 2048 Sep 16  2015 Documents/
drwxr-xr-x  5 njog users 4096 Apr 29  2016 Downloads/
drwxr-xr-x 31 njog wheel 2048 Apr 29  2016 Library/
drwxr-xr-x  3 njog users 4096 May  2 11:22 malloclab-rerun/
drwx------  2 njog wheel 2048 Sep 18  2015 Movies/
drwxr-xr-x  2 njog users 2048 Jan 12  2015 Music/
drwxr-xr-x 51 njog wheel 4096 Sep  2 14:56 OldFiles/   ← One day of backups
drwxr-xr-x  2 njog users 2048 Jun 25  2015 perl5/
drwxr-xr-x  2 njog users 2048 Apr 27 02:38 Pictures/
drwx------ 12 njog wheel 2048 Sep  3 15:20 private/   ← Make sure all work is done here!
drwxr-xr-x  2 njog wheel 2048 Oct  2  2016 public/
drwxr-xr-x  2 njog users 2048 Jan 12  2015 Public/
drwxr-xr-x  2 njog users 2048 Jan 12  2015 Templates/
drwxr-xr-x  2 njog users 2048 Jan 12  2015 Videos/
drwxr-xr-x  2 njog wheel 2048 Apr 27 02:37 www/
```

Editing Files and Building Code

- There are many editors out there
- We will show you vim, because it’s the ~~one true editor~~ has useful features
  - It’s installed on pretty much every machine
  - You can run it right inside your terminal
- After this short tutorial, you’ll hopefully be able to:
  - Edit your code
  - Configure some basic vim settings to make your life easier
  - Automatically indent your code (?!)
  - Build your code and see errors without ever quitting the editor (???!!!)

Editing Files and Building Code

1. We'll start by editing the vim configuration file
   - ".vimrc", a file in your home directory
2. From the Shark machine command line, run:
   - vim ~/.vimrc
3. Vim has a few "modes" that you will switch between. For now:
   - "normal" mode: run commands, save/quit, text manipulation
     - If you press <Escape>, you end up here
     - Type "**:w**" (colon, w, <Enter>) to save your file, and type "**:q**" (colon, q, <Enter>) to quit
     - Press "**i**" to enter "insert" mode
   - "insert" mode: works like a typical editor (type/delete text, move around with arrow keys)
     - "--INSERT--" will appear in the status bar at the bottom
4. Try adding the following line to the file:
   - colorscheme desert

Editing Files and Building Code

1. Let’s practice a bit more. Enter the code on the right into “~/.vimrc”.
    - If you can’t enter text, you’re probably in “normal” mode. Hit “**i**” to “insert” text.
2. Once you’re done, save and quit.
    - Get back to normal mode: <Escape>
    - Save the text: “**:w**” (colon, w, <Enter>)
    - Quit: “**:q**” (colon, q, <Enter>)

```
colorscheme desert
set mouse=a
set number
set cursorline
set colorcolumn=81
set tabstop=4
set shiftwidth=4
set softtabstop=4
set expandtab
set smartindent
```

If you’re curious about what any of this does, read the upcoming Piazza post about Vim tricks.

Next up: editing some C code!

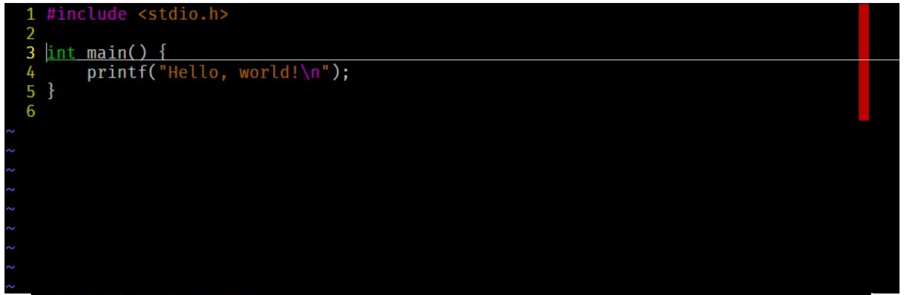
Editing Files and Building Code

1. cd to “gitbootcamp-example”
    - cd ~/private/15213/gitbootcamp/gitbootcamp-example
2. Build the code and run the program
    - make
    - ./hello
3. Output should look like this ->

```
$ make
cc -Wall -Wextra -pedantic -std=c99   -c -o
hello.o hello.c
cc   hello.o   -o hello
$ ./hello
Hello, world!
```

Editing Files and Building Code

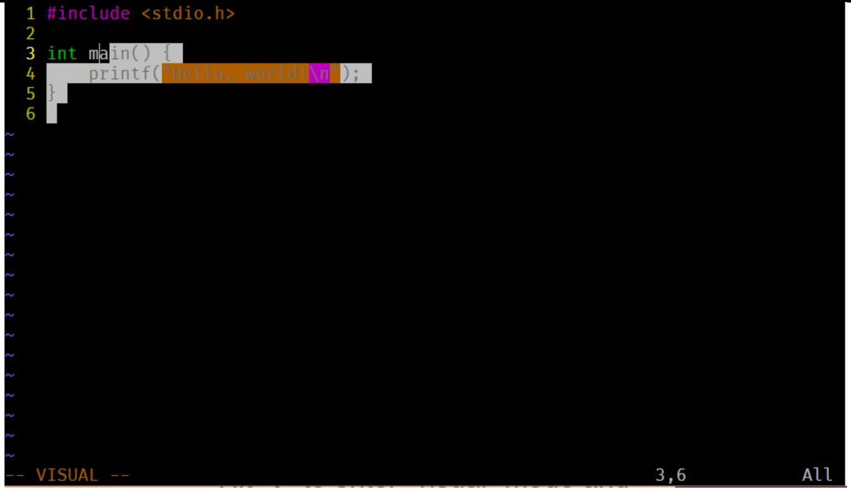
1. Let's edit "hello.c":
   - vim hello.c
2. Hopefully, it looks something like ->
   - Red bar on the right: colorcolumn=81
     - Easily see if a line is too long
3. Hit "v" to enter "visual" mode
   - Use the cursor keys to select text
   - "**y**": copy ("yank"), "**d**": cut ("delete")
   - "**p**": paste after cursor
   - "**=**": **auto-indent** selected text (!!!)
4. **Mouse support**: if your terminal supports it, click to move/select text



```
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
}

```



```
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
}

-- VISUAL --                                            3,6           All
```

Editing Files and Building Code

1. Edit your code to look like this ->
2. Save the file ("**:w**" in normal mode)
3. Now, for the magic: type "**:make**"
4. You should see some output in your terminal from make ->
5. Hit <Enter> to go back to your code
   - (might have to press it twice)

Congratulations, you just built your code from inside vim. But wait, there's more!

```c
#include <stdio.h>

int main() {
        foo();
        printf("Hello, world!\n", 12345);
}
```

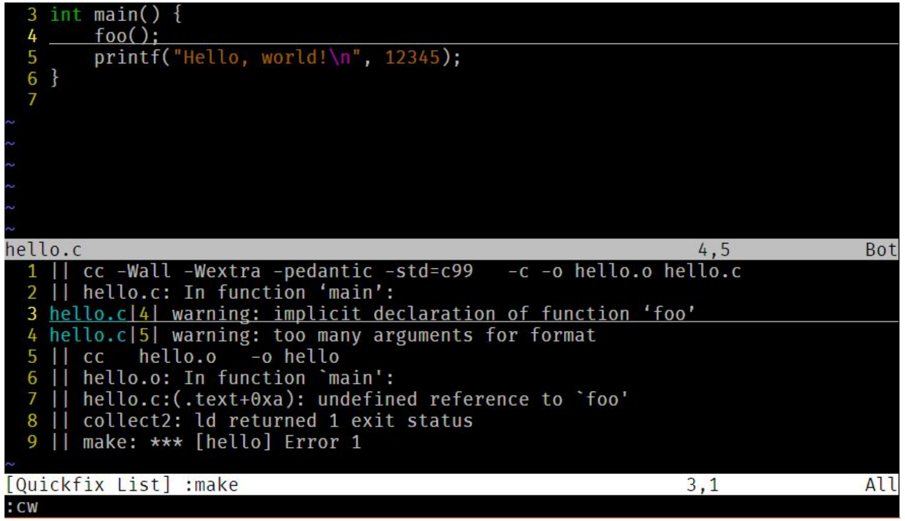
```
cc -Wall -Wextra -pedantic -std=c99   -c -o hello.o
hello.c
hello.c: In function ‘main’:
hello.c:4: warning: implicit declaration of function
‘foo’
hello.c:5: warning: too many arguments for format
cc   hello.o   -o hello
hello.o: In function `main':
hello.c:(.text+0xa): undefined reference to `foo'
collect2: ld returned 1 exit status
make: *** [hello] Error 1

Press ENTER or type command to continue
```

Editing Files and Building Code

We probably won’t have time for this, but on your own time you can:

1. Type “**:cw**” to bring up a window with all the build errors. (!!!)
2. Put your cursor on an error and press <Enter> to jump to it
   - You can also double-click on it
3. Switch windows: **<Ctrl-w> j** and **<Ctrl-w> k** (or with the mouse)
   - Use “**:q**” to close the active window



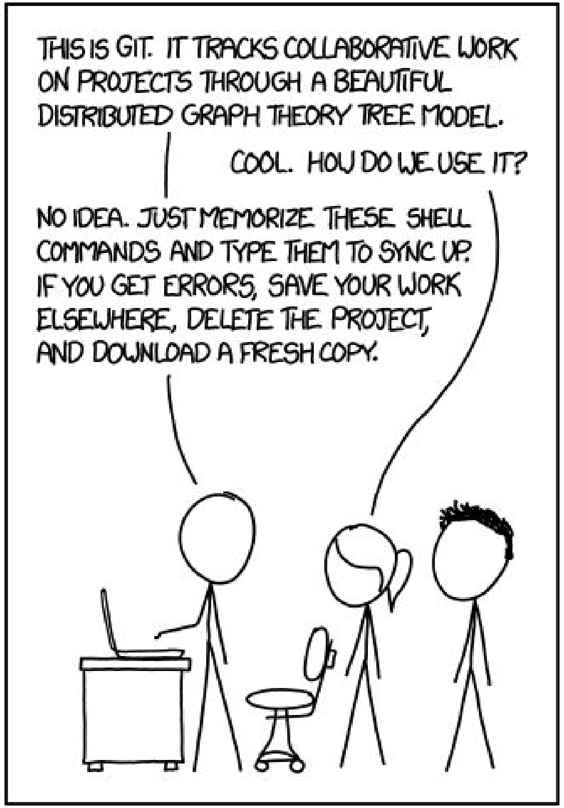
```
  3 int main() {
  4     foo();
  5     printf("Hello, world!\n", 12345);
  6 }
  7
~
~
~
~
~
~
hello.c                                                    4,5          Bot
  1 || cc -Wall -Wextra -pedantic -std=c99   -c -o hello.o hello.c
  2 || hello.c: In function ‘main’:
  3 hello.c|4| warning: implicit declaration of function ‘foo’
  4 hello.c|5| warning: too many arguments for format
  5 || cc    hello.o   -o hello
  6 || hello.o: In function `main':
  7 || hello.c:(.text+0xa): undefined reference to `foo'
  8 || collect2: ld returned 1 exit status
  9 || make: *** [hello] Error 1
~
[Quickfix List] :make                                      3,1          All
:cw
```

Editing Files and Building Code

- There's a lot more to vim. If you want to learn more:
  - Run the "**vimtutor**" program
  - Read "**:help**" inside Vim
  - Check out the upcoming Piazza post and other posts on the internet
- "Use a Single Editor Well"
  - "The editor should be an extension of your hand; make sure your editor is configurable, extensible, and programmable."
  - https://pragprog.com/the-pragmatic-programmer/extracts/tips
- If you don't like vim, feel free to use whatever works best for you! However:
  - In general, if you must edit your code on your local machine, you should transfer your files back to the Shark machines and build it there.
    - Various plugins to do this automatically for most popular editors (Sublime SFTP, etc.)
  - We do NOT recommend using a C IDE; build environment setup is painful/impossible.

Introduction to Git

- Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.
- It allows you to work on multiple versions of the same code and maintain it efficiently.
- We encourage you to use git for all your coding assignments in this course.


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Setting up Git on your Shark machine

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

Note: Make sure, the email you enter is the same one you use to login to GitLab

Setting up your first local git repository

```
#Switch to the ‘gitbootcamp-example’ directory and initialize
your empty git repo.
$ git init
$ git status


#Check the status of the repo. You’d notice that all the files
are untracked at the moment. Add ‘hello.c’ for tracking.
$ git add hello.c


#Alternatively, you could have added all the files in the
directory for tracking, using a single command.
$ git add -A
```

Your first commit

```
#Make your changes in the file(s). When you are ready, you
must 'commit' your changes

$ git commit hello.c
```

Opens a text editor (vi) for writing a commit message.
Time to use your vim skills you just acquired!

```
#Variations:
```

You may type in the commit message with the command itself

```
$ git commit hello.c -m "commit message"
```

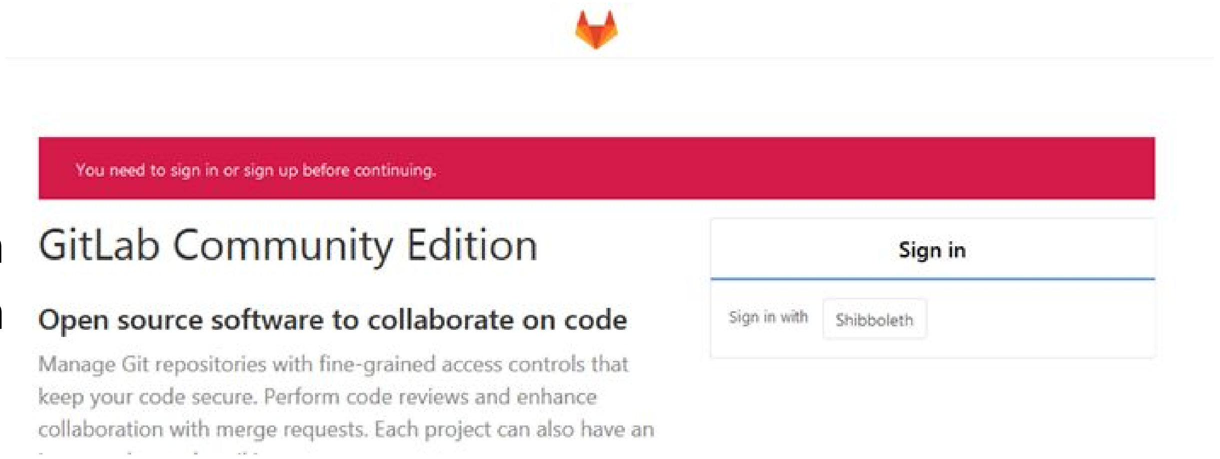
Commits all files with changes in the repo

```
$ git commit -a -m "commit message"
```

Note: All your changes are stored in the local repository. In the next slide we shall talk about linking this to your GitLab repository

Setting up GitLab

Sign into GitLab through Shibboleth


You need to sign in or sign up before continuing.

GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an

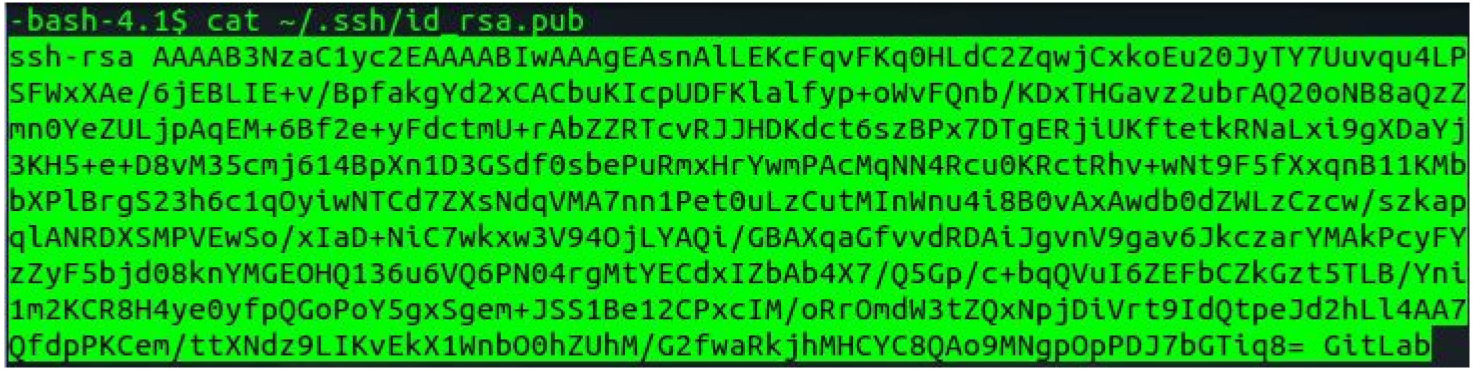
- Before you can link your local repository to your GitLab account, you must set up SSH Keys for authentication
- Generate SSH keys on the client machine (shark machines here) and add them to your GitLab account.

```
$ ssh-keygen -t rsa -C "GitLab" -b 4096
```

Use the default file path (press Enter). Optionally type in a password, else press Enter

Setting up GitLab

```
#Print out the public key on screen
$ cat ~/.ssh/id_rsa.pub
```



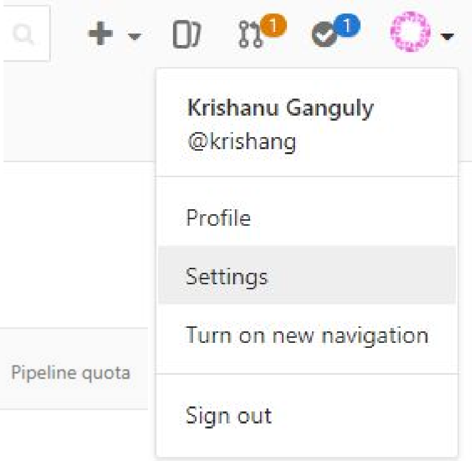
```
-bash-4.1$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAgEAsnAlLEKcFqvFKq0HLdC2ZqwjCxkoEu20JyTY7Uuvqu4LP
SFWxXAe/6jEBLIE+v/BpfakgYd2xCACbuKIcpUDFKlalfyp+oWvFQnb/KDxTHGavz2ubrAQ20oNB8aQzZ
mn0YeZULjpAqEM+6Bf2e+yFdctmU+rAbZZRTcvRJJHDKdct6szBPx7DTgERjiUKftetkRNaLxi9gXDaYj
3KH5+e+D8vM35cmj614BpXn1D3GSdf0sbePuRmxHrYwmPAcMqNN4Rcu0KRctRhv+wNt9F5fXxqnB11KMb
bXPlBrgS23h6c1qOyiwNTCd7ZXsNdqVMA7nn1Pet0uLzCutMInWnu4i8B0vAxAwdb0dZWLzCzcw/szkap
qlANRDXSMPVEwSo/xIaD+NiC7wkxw3V94OjLYAQi/GBAXqaGfvvdRDAiJgvnV9gav6JkczarYMAkPcyFY
zZyF5bjd08knYMGEOHQ136u6VQ6PN04rgMtYECdxIZbAb4X7/Q5Gp/c+bqQVuI6ZEFbCZkGzt5TLB/Yni
1m2KCR8H4ye0yfpQGoPoY5gxSgem+JSS1Be12CPxcIM/oRrOmdW3tZQxNpjDiVrt9IdQtpeJd2hLl4AA7
QfdpPKCem/ttXNdz9LIKvEkX1WnbO0hZUhM/G2fwaRkjhMHCYC8QAo9MNgpOpPDJ7bGTiq8= GitLab
```

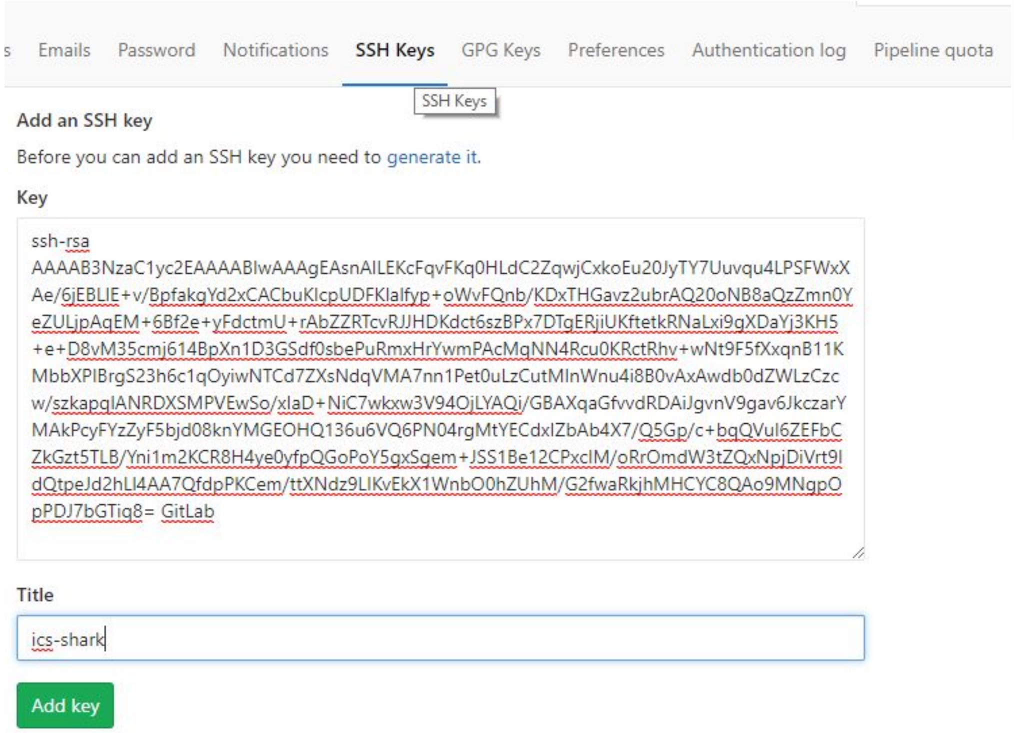
Copy the Public Key

- Select the entire public key with the mouse and
    - ■Cmd+c if you are on a mac
    - ■Ctrl+Insert if you are on windows using KiTTY or on Linux using SSH

Setting up GitLab

Go to your account settings-> SSH Keys and paste the key here.





Add an SSH key

Before you can add an SSH key you need to generate it.

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAgEAsnAILEKcFqvFKq0HLdC2ZqwjCxkoEu20JyTY7Uuvqu4LPSFWxX
Ae/6jEBLIE+v/BpfakgYd2xCACbuKIcpUDFKlalfyp+oWvFQnb/KDxTHGavz2ubrAQ20oNB8aQzZmn0Y
eZULjpAqEM+6Bf2e+yFdctmU+rAbZZRTcvRJJHDKdct6szBPx7DTgERjiUKftetkRNaLxi9gXDaYj3KH5
+e+D8vM35cmj614BpXn1D3GSdf0sbePuRmxHrYwmPAcMqNN4Rcu0KRctRhv+wNt9F5fXxqnB11K
MbbXPIBrgS23h6c1qOyiwNTCd7ZXsNdqVMA7nn1Pet0uLzCutMInWnu4i8B0vAxAwdb0dZWLzCzc
w/szkapqIANRDXSMPVEwSo/xIaD+NiC7wkxw3V94OjLYAQi/GBAXqaGfvvdRDAiJgvnV9gav6JkczarY
MAkPcyFYzZyF5bjd08knYMGEOHQ136u6VQ6PN04rgMtYECdxIZbAb4X7/Q5Gp/c+bqQVuI6ZEFbC
ZkGzt5TLB/Yni1m2KCR8H4ye0yfpQGoPoY5gxSgem+JSS1Be12CPxcIM/oRrOmdW3tZQxNpjDiVrt9I
dQtpeJd2hLI4AA7QfdpPKCem/ttXNdz9LIKvEkX1WnbO0hZUhM/G2fwaRkjhMHCYC8QAo9MNgpO
pPDJ7bGTiq8= GitLab
```

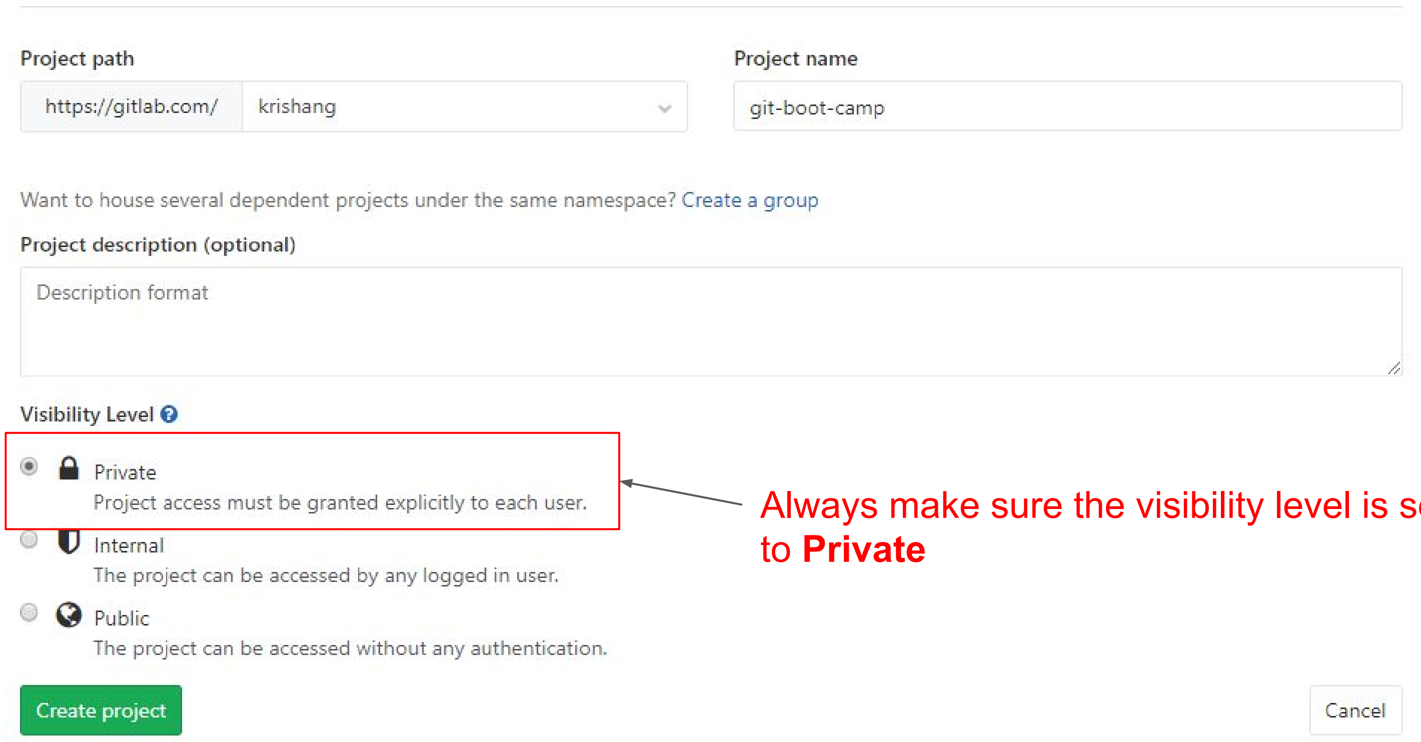
Title

ics-shark

Add key

Setting up GitLab

Create a new project with an appropriate name



Project path

https://gitlab.com/ krishang

Project name

git-boot-camp

Want to house several dependent projects under the same namespace? Create a group

Project description (optional)

Description format

Visibility Level

[✓ Private]
Project access must be granted explicitly to each user.

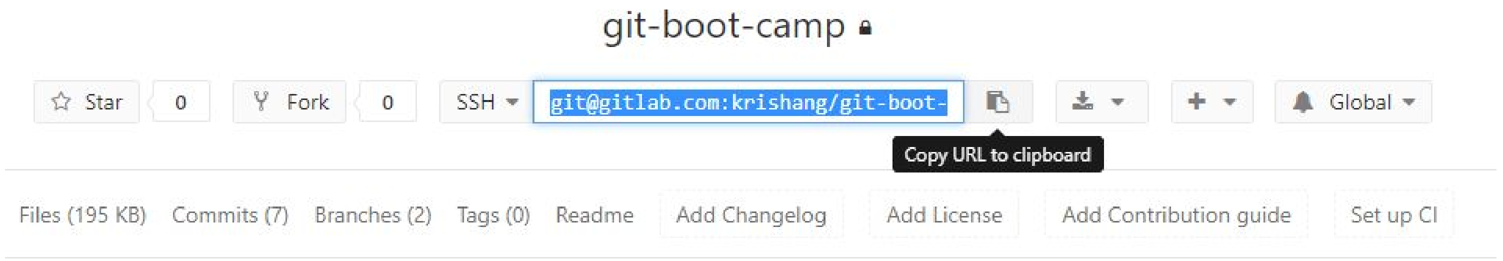
Internal
The project can be accessed by any logged in user.

Public
The project can be accessed without any authentication.

Always make sure the visibility level is set to **Private**

Setting up GitLab

Go to your project and copy the repo URL



git@gitlab.com:krishang/git-boot-

Add the path to your GitLab repo on your local git repo (on the shark machine). Your command should looks something like this:

```
$ git remote add origin
git@git.ece.cmu.edu:andrewid/git-boot-camp.git
```

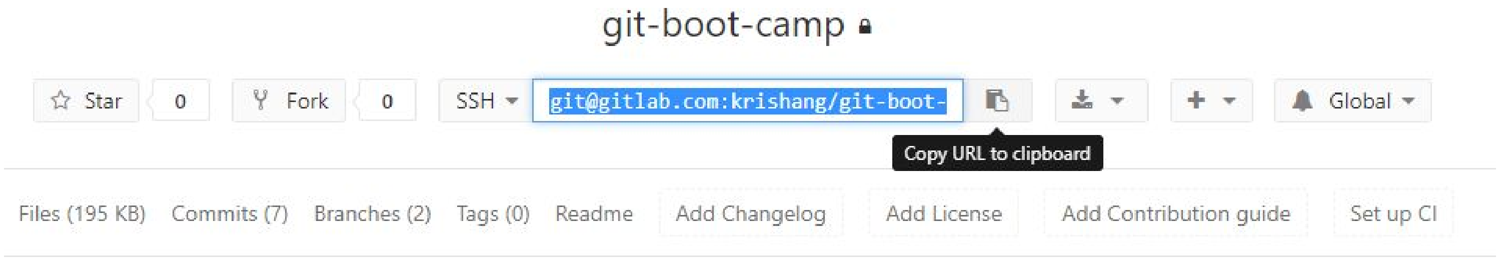
Pushing your commits

All your commits on the local git repo won’t get reflected on the GitLab repo until you push your commits.

```
$ git push origin master
```

This will push all the commits made on the ‘master’ branch of the repository pointed by ‘origin’.

Cloning an existing repository


git-boot-camp

git@gitlab.com:krishang/git-boot-

Copy URL to clipboard

Files (195 KB) Commits (7) Branches (2) Tags (0) Readme Add Changelog Add License Add Contribution guide Set up CI

Assuming you've already set up your SSH keys, copy the URL of the repo from GitLab and create a git repo on your local machine (shark machine in your case). Your command should looks something like this:

```
$ git clone
git@git.ece.cmu.edu:andrewid/git-boot-camp.git
```

You might want to create a separate folder for the new repo

Git Commands

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

More on Git

Getting help:

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

Some great git tutorials:

- https://try.github.io/
- https://www.atlassian.com/git/tutorials