

RU-NB CS211 iLAB MACHINE GUIDE

Made by Aidan Andrucyk

[ILAB MACHINE CONNECTION](#)

[LINUX COMMANDS](#)

[TEXT EDITORS](#)

[MAKEFILE](#)

[TRANSFERRING FILES](#)

[TARs](#)

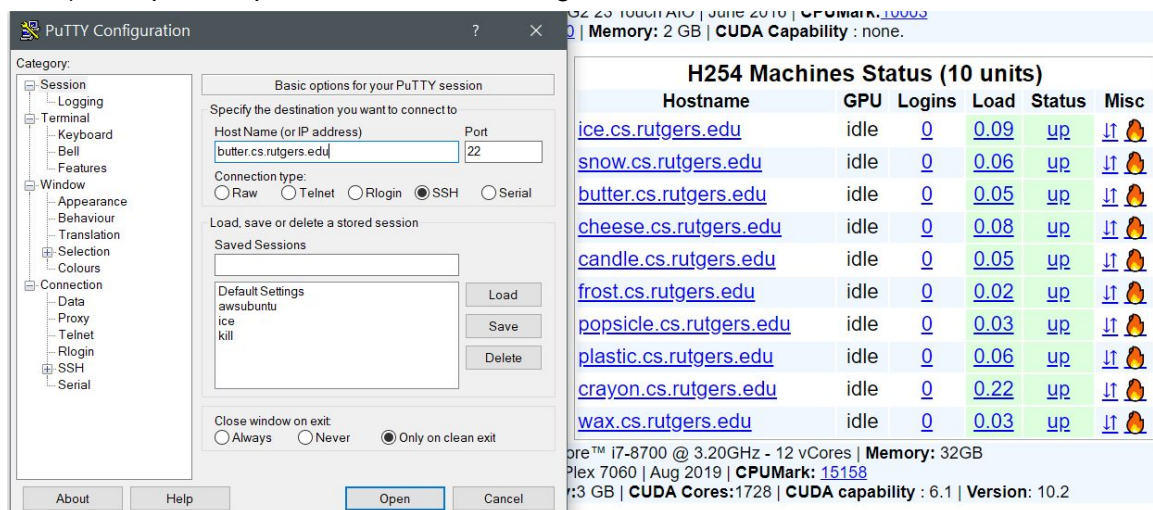
[MEMORY LEAKS](#)

[BOMB LAB](#)

ILAB MACHINE CONNECTION

PuTTY is used to remotely connect to iLab machines from a **windows device**, giving us a way to use a Linux environment as well as a standardized way of running our programs so that the grader and yourself work with the same settings.

1. Activate account + create a password @ [iLab Machines Accounts Page](#)
2. Look for a host [iLab Machine](#) with few logins and copy its hostname
3. Download and open [PuTTY](#)
4. Paste hostname into corresponding textbox, make sure connection type is SSH (Secure Shell), and press Open button on bottom right



The screenshot shows the PuTTY Configuration window on the left and a table of machine statuses on the right.

PuTTY Configuration Window:

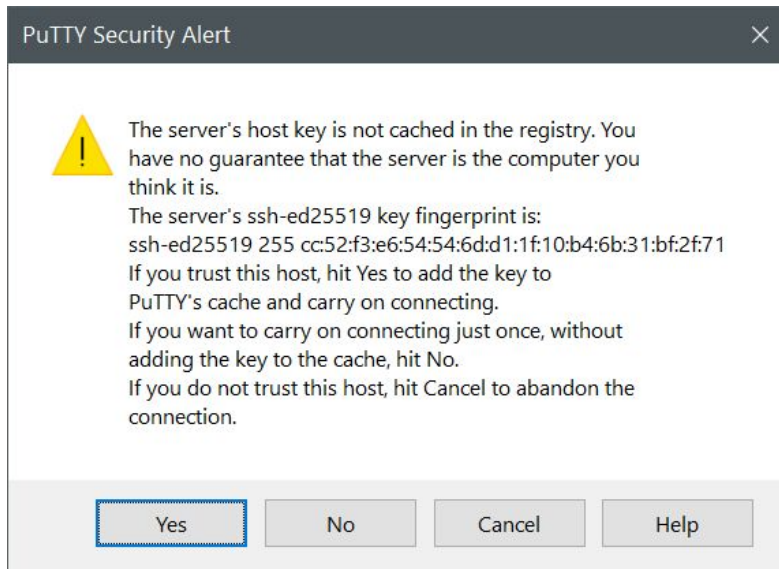
- Category: Session
- Basic options for your PuTTY session
- Specify the destination you want to connect to:
 - Host Name (or IP address): butter.cs.rutgers.edu
 - Port: 22
- Connection type: ☐ Raw ☐ Telnet ☐ Rlogin ☒ SSH ☐ Serial
- Load, save or delete a stored session:
 - Saved Sessions: (empty list)
 - Default Settings: awsubuntu, ice, kill
 - Buttons: Load, Save, Delete
- Close window on exit: ☐ Always ☐ Never ☒ Only on clean exit
- Buttons: About, Help, Open, Cancel

H254 Machines Status (10 units)

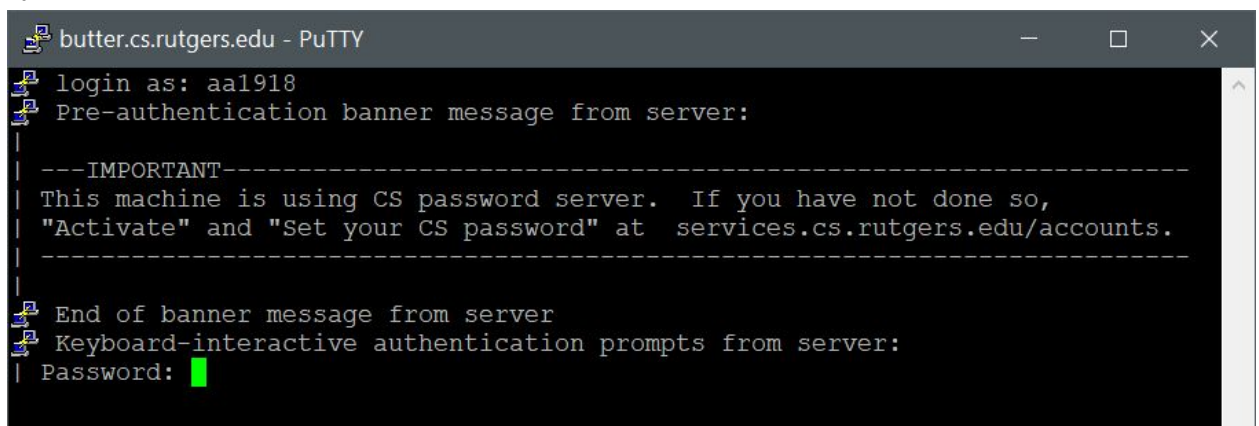
Hostname	GPU	Logins	Load	Status	Misc
ice.cs.rutgers.edu	idle	0	0.09	up	⬆️ 🔥
snow.cs.rutgers.edu	idle	0	0.06	up	⬆️ 🔥
butter.cs.rutgers.edu	idle	0	0.05	up	⬆️ 🔥
cheese.cs.rutgers.edu	idle	0	0.08	up	⬆️ 🔥
candle.cs.rutgers.edu	idle	0	0.05	up	⬆️ 🔥
frost.cs.rutgers.edu	idle	0	0.02	up	⬆️ 🔥
popsicle.cs.rutgers.edu	idle	0	0.03	up	⬆️ 🔥
plastic.cs.rutgers.edu	idle	0	0.06	up	⬆️ 🔥
crayon.cs.rutgers.edu	idle	0	0.22	up	⬆️ 🔥
wax.cs.rutgers.edu	idle	0	0.03	up	⬆️ 🔥

System Info: Core™ i7-8700 @ 3.20GHz - 12 vCores | Memory: 32GB
Plex 7060 | Aug 2019 | CPUMark: 15158
3 GB | CUDA Cores: 1728 | CUDA capability : 6.1 | Version: 10.2

5. Select yes for pop up



6. Type in NetID and password



7. Congratulations you're in!

os.verizon.net on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Mon Aug 10 04:26:17 2020 from pool-100-1-148-88.nwrknj.fios.verizon.net

CentOS Linux release 7.6.1810 (Core) 3.10.0-957.21.2.el7.x86_64

Machine Name:	kill.cs	IP No:	128.6.13.175
Mon Aug 10 05:28:12 EDT 2020		Uptime:	45 days 09:54

Processes:	363	Local/SSH/X2Go/Xrdp:	0/1/0/0 (1)
Connections:	10	System Load:	0
Free Memory:	4.3G of 31G	Free Swap:	39G of 39G

CPU Info:	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz - 8 cores		
System CPU:	1.14%	User CPU:	2.76%
CPU Idle:	96.03%	IO Wait:	0.06%

Login as:	aa1918	No. of Sessions:	1
Avail.UserDisk:		Avail.Freespace:	4.83 GB
CUDA Driver:	10.0	CUDA Cores:	384

[aa1918@kill ~]\$ clear

LINUX COMMANDS

- `pwd` : show name of current working directory
- `cd` : change directory
- `cd ..` : change to parent directory
- `ls` : list files
- `mkdir` : make a directory
- `rmdir` : remove a directory
- `touch` : create a file
- `cat` : print file contents
- `mv` : rename/move a file
- `rm` : remove a file
- `grep` : search texts in files

TEXT EDITORS

Nano:

- Easy to use and master.
- Nano has most of the shortcuts listed at the bottom of the window, making it extremely simple to use.
- Search function
- Search and replace
- "Goto line" command
- Automatic indentation

Vim:

- Tough to get started with and master. The editing and command modes will confuse beginners.
- Session recovery
- Split screen
- Tab expansion
- Completion commands
- Syntax coloring

MAKEFILE

Makefiles are largely used for creating shortcuts. There are a number of unique elements about Makefiles. For one, the name of the file must always be Makefile and is case sensitive. It is also important to note that Makefiles are whitespace sensitive and tabs are used to indicate that a given command belongs to a given shortcut. Let us take this sample Makefile:

```
all: first.c
    gcc -Wall -Werror -Wvla -fsanitize=address -o first first.c
clean:
    rm -f first
```

Typing “make” into the command line will run “all:” looking for first.c. If first.c exists, it will run the next line which compiles the program. Typing “clean” into the command line will run “clean:” looking for nothing and will remove all files named “first”.

TRANSFERRING FILES

This section describes how to move files between your remote desktop and the iLab Machine

ftp clients:

- (1) **Cyberduck** (<https://cyberduck.io/>)
- (2) **Filezilla** (<https://filezilla-project.org/>)

These clients will allow you to sftp into the ilab machine and you can use the UI to drag and drop files between the ilab machines and your local machine

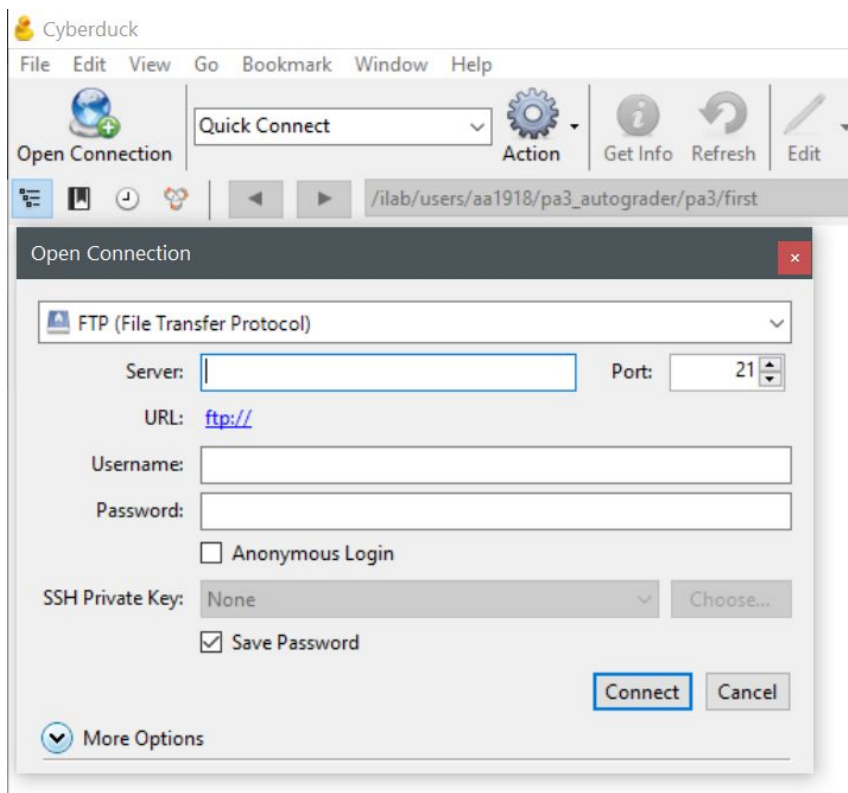
terminal:

1. **scp** if you have linux or mac os
(<https://linuxize.com/post/how-to-use-scp-command-to-securely-transfer-files/>)
2. **pscp** if you have windows (<https://www.ssh.com/ssh/putty/putty-manuals/0.68/Chapter5.html>)

Both use the scp protocol to transfer files back and forth

Let us use Cyberduck as an example:

1. Download and open [Cyberduck](#)
2. Click Open Connection on the top left of the screen



3. Change connection type to SFTP

Open Connection

SFTP (SSH File Transfer Protocol)

Server: Port:

URL: <sftp://kill.cs.rutgers.edu>

Username:

Password:

☐ Anonymous Login

SSH Private Key:

☒ Save Password

4. Enter login information and press Connect on the bottom right

Open Connection

SFTP (SSH File Transfer Protocol)

Server: Port:

URL: <sftp://aa1918@kill.cs.rutgers.edu>

Username:

Password:

☐ Anonymous Login

SSH Private Key:

☒ Save Password

5. Set Always to true and press Allow

Unknown fingerprint



Unknown fingerprint

The fingerprint for the ED25519 key sent by the server is
39:14:dd:cf:8a:93:2c:96:4c:c4:3f:b6:9b:c4:d0:90.

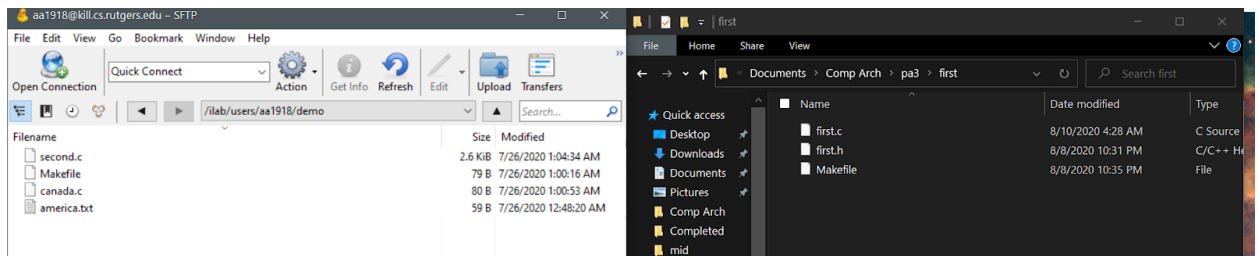
→ Allow

→ Deny

☒ Always

[Help](#)

6. Open file explorer
7. Congratulations! Just click and drag to transfer your files!



TARs

UNTAR:

```
tar xvf auto_grader.tar
```

TAR:

To create a tar file, put everything that you are submitting into a directory (folder) named pa1. Then, cd into the directory containing pa1 (that is, pa1's parent directory) and run the following command:

```
tar cvf pa1.tar pa1
```

To check that you have correctly created the tar file, you should copy it (pa1.tar) into an empty directory and run the following command:


```
tar xvf pa1.tar
```

This should create a directory named pa1 in the (previous) empty directory.

RUN AUTOGRADER FOR paX.tar:

1. Copy pa1.tar to the auto grader directory
2. Run the auto grader with pa1.tar as the argument:
`python auto_grader.py pa1.tar`

MEMORY LEAKS

On the off chance that you have a memory leak, use [Valgrind](#)

To use this on our example program, [test.c](#):

```
gcc -o test -g -fsanitize=address test.c
```

This creates an executable named test. To check for memory leaks during the execution of test:

```
valgrind --tool=memcheck --leak-check=yes --show-reachable=yes --num-callers=20  
--track-fds=yes ./test
```

To interpret the meaning of the output, refer to the page's [list of errors](#) as well as the more [extensive pdf](#)

BOMB LAB

1. Untar bomb__.tar
 2. Change directory (cd) to bomb__
 3. Make a new file named solution.txt (vim solution.txt)
 - a. Type in your bomb lab solutions line by line here as you learn to diffuse the bomb (each new line represents the solution corresponding to a given phase)
 4. (Optional)
 - a. Enter below to get the assembly instructions as a txt file
- ```
-bash-4.2$ objdump -d ./bomb > dump.txt
```
- b. Enter "strings bomb" to get a list of all strings that the program uses (can be useful hint hint)
5. Enter "gdb bomb" in the command line
  6. Enter "break explode\_bomb" (important!)
    - a. Enter "break phase\_[insert phase number]" if you would like to stop at the beginning of a given phase (recommended)
  7. Enter "run solution.txt"

8. Enter "disass" to see assembly
9. Congratulations you see the assembly code now!
10. Here are a couple things you may find yourself using:
  - a. i r (gets the register information)
  - b. ni (next line)
  - c. si (steps into a function if it is being called)
11. Enter "quit" to exit gdb