

SENG265 SPRING 2025

Lab 01 & A0:
Remote Lab Connection -
Software Compatibility in SENG265
Labs on week of January 13th

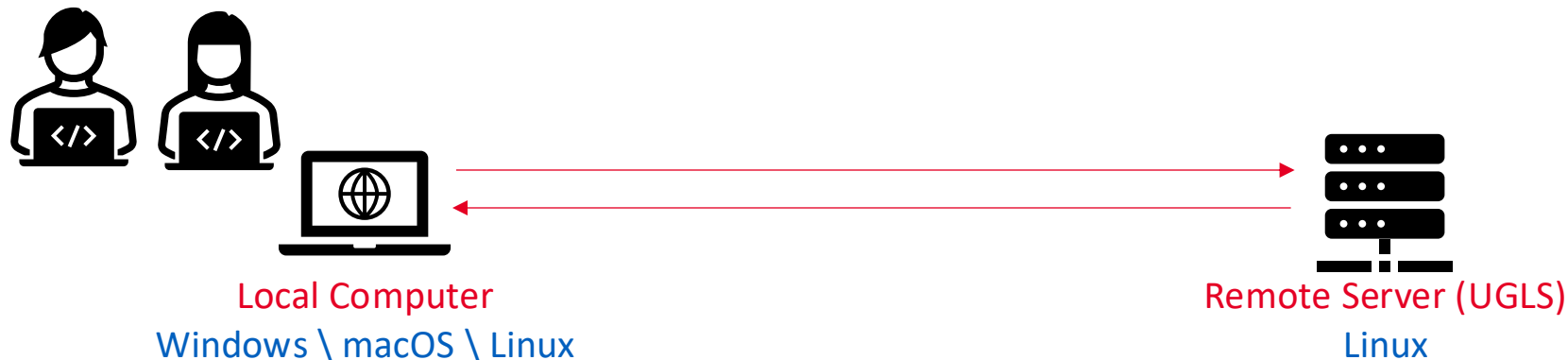
Software Compatibility

- As any other software-related product (e.g., operating systems, programs), programming languages come in different flavours and versions
- Effective testing of the quality of a program written in a specific programming language requires making sure that both the programming\development and testing environments are equivalent (e.g., in terms of libraries installed)
- This allows testers to focus on evaluating functional and non-functional aspects of software and their compliance with user requirements, ruling out software defects due to potential misconfigurations
- Therefore, SENG265 defines a shared server (i.e., UGLS, a remote connection) that can be used by both programmers (i.e., students) and testers (i.e., TAs) to assess whether a required code submission satisfies expected requirements

IMPORTANT: regardless of the IDE\text editor\programming environment used to develop code-related assignments for SENG265, you **MUST** make sure that your code works in the UGLS server introduced in Lab 01

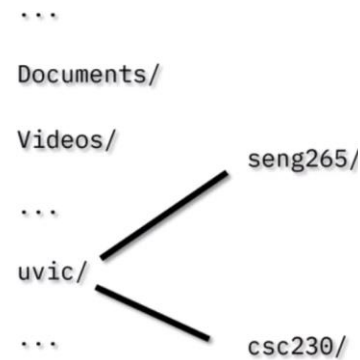
Compatibility in SENG265

- When coding\programming for assignments in SENG265, you **MUST** use either:
 - The physical computers in ELW B238 (where labs take place)
 - A remote connection to the Electrical and Computer Engineering **UGLS** Shell Server (SSH, ugls.ece.uvic.ca). This is a connection to one the physical computers in ELW B238.
- As getting remote access to the lab computers is very convenient for this course, Lab 01 concentrates on explaining the steps required to achieve it. The following figure depicts this process



STEP I: Local Computer Config.

- To facilitate future explanations and improve the organization of files to be used throughout the course, we suggest creating the following directory structure in your local computer:



A. Windows CLI

- Search for and open the program “Windows Terminal”\“Command Prompt”\“CMD”
- Type and execute the following command (use the “Enter”\“Return” key to execute it):

```
mkdir uvic\seng265
```

B. macOS/Linux CLI

- Search for and open the program “Terminal”\“Console”
- Type and execute the following command (use the “Enter”\“Return” key to execute it):

```
mkdir -p uvic/seng265
```

CLI = Command-Line
Interface

STEP II: Navigating to seng265

- Using your OS CLI, we'll navigate to the seng265 folder created in STEP I.

A. Windows CLI (from STEP I)

- i. Type and execute the following command (use the "Enter"\"Return" key to execute it):

```
cd uvic\seng265
```

B. macOS/Linux CLI (from STEP I)

- i. Type and execute the following command (use the "Enter"\"Return" key to execute it):

```
cd uvic/seng265
```

CLI = Command-Line
Interface

OS = Operating
System

STEP III: Using the seng265 Folder

- We'll create a test file in the seng265 folder created in STEP I.



A. Windows CLI (from STEP II)

- Type and execute the following command (use the “Enter”\”Return” key to execute it):

```
type nul > new-empty-file.txt
```

B. macOS/Linux CLI (from STEP II)

- Type and execute the following command (use the “Enter”\”Return” key to execute it):

```
touch new-empty-file.txt
```

NOTE:

This only works in CMD. if you're using PowerShell, you'll have to use the File Explorer, navigate to USER\uvic\seng265, and create the file with the menus displayed after right-clicking once inside the folder

CLI = Command-Line Interface

STEP IV: Accessing UGLS

- i. Open a new CLI of your operating system (i.e., “CMD” for Windows, “Terminal” for macOS/Linux)
- ii. Execute the following command (replace “NETLINKID” with your actual UVic NetlinkID)

```
ssh ughs.ece.uvic.ca -l NETLINKID
```

- iii. Provide your password (i.e., same credentials used to connect to Brightspace) →
- iv. Once you provide your correct password and using the “Enter”\”Return” key), your connection to UGLS should be established:

NOTE: At this point, as a security measure, typing won't show anything on screen, but your input is still being captured by the CLI. We recommend copying and pasting your password to avoid typos

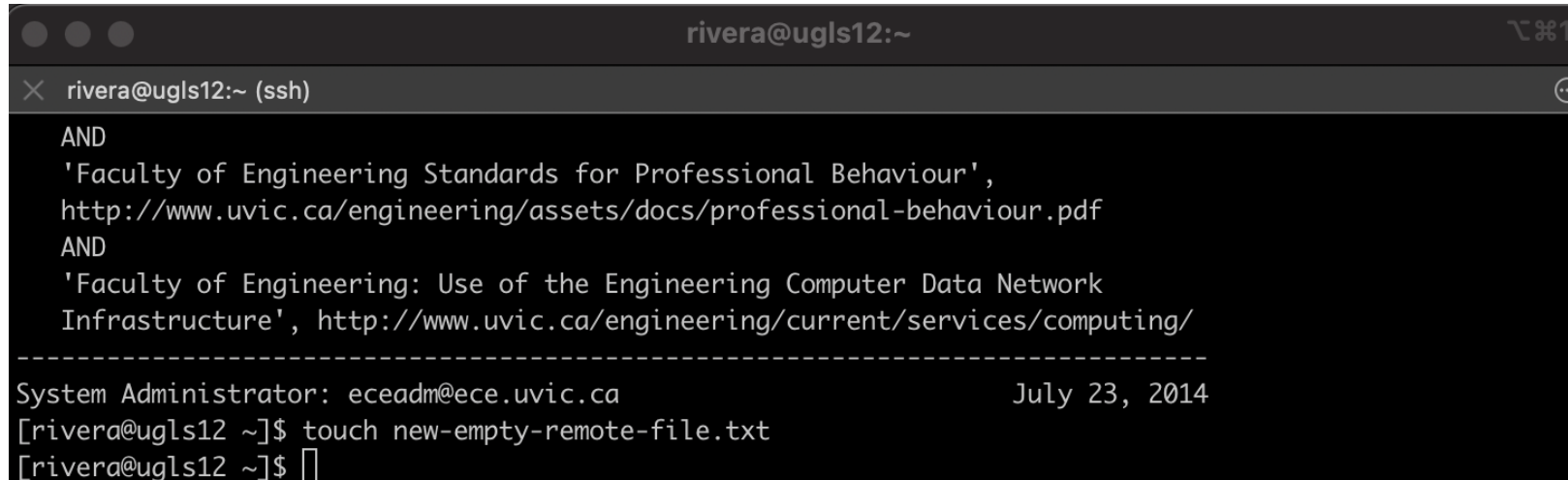
```
rivera@ugls13:~  
PS C:\Users\lfeli> ssh ughs.ece.uvic.ca -l rivera  
rivera@ughs.ece.uvic.ca's password:  
Last login: Wed May 10 00:47:50 2023 from s010600cb7a10c811.gv.shawcable.net  
-----  
- To access this machine remotely, use Secure Shell protocol 2 (SSH2),  
Secure FTP (SFTP), and / or Secure Copy Protocol (SCP) as detailed at:  
http://www.ece.uvic.ca/computing/remote-access.shtml  
-----  
- Use of this facility must adhere to:  
'Policy IM7200: Responsible Use of Information Technology Services',  
http://www.uvic.ca/universitysecretary/assets/docs/policies/IM7200_6030_.pdf  
AND  
'Faculty of Engineering Standards for Professional Behaviour',  
http://www.uvic.ca/engineering/assets/docs/professional-behaviour.pdf  
AND  
'Faculty of Engineering: Use of the Engineering Computer Data Network  
Infrastructure', http://www.uvic.ca/engineering/current/services/computing/  
-----  
System Administrator: eceadm@ece.uvic.ca  
[rivera@ugls13 ~]$
```

NOTE: To close the connection to UGLS you can type and execute the command:
exit

STEP V: Using UGLS

- i. Open the CLI used in Step IV.
- ii. Execute the following command:

```
touch new-empty-remote-file.txt
```



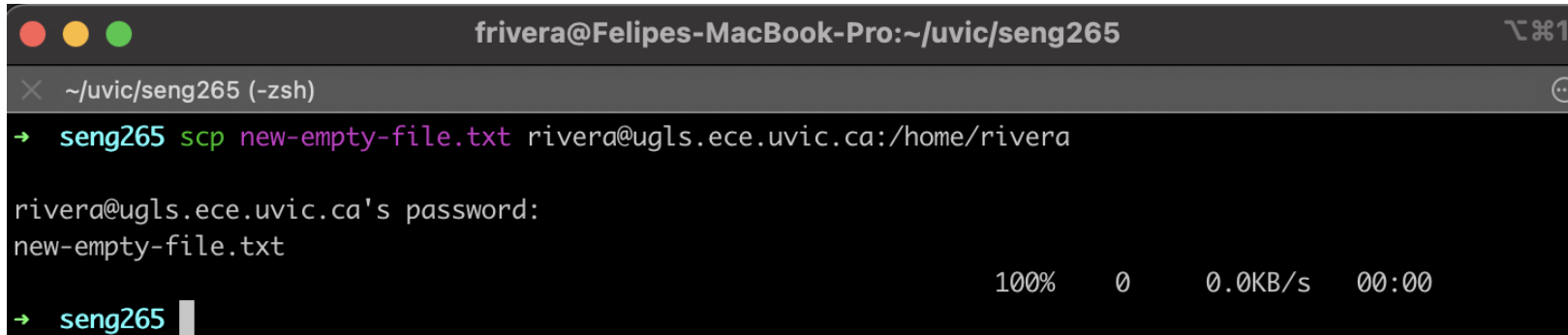
```
rivera@ugls12:~  
X rivera@ugls12:~ (ssh)  
AND  
'Faculty of Engineering Standards for Professional Behaviour',  
http://www.uvic.ca/engineering/assets/docs/professional-behaviour.pdf  
AND  
'Faculty of Engineering: Use of the Engineering Computer Data Network  
Infrastructure', http://www.uvic.ca/engineering/current/services/computing/  
-----  
System Administrator: eceadm@ece.uvic.ca July 23, 2014  
[rivera@ugls12 ~]$ touch new-empty-remote-file.txt  
[rivera@ugls12 ~]$
```


STEP VI: Sending files to UGLS

- i. Open the CLI used in Step III (or open a new one and execute the commands in Step II)
- ii. Execute the following command to transfer the file new-empty-file.txt to UGLS (replace “NETLINKID” with your actual UVic NetlinkID):

```
scp new-empty-file.txt NETLINKID@ugls.ece.uvic.ca:/home/NETLINKID/
```

- vi. Once you provide your correct password, the file will be transferred:



```
frivera@Felipes-MacBook-Pro:~/uvic/seng265
~/uvic/seng265 (-zsh)
→ seng265 scp new-empty-file.txt rivera@ugls.ece.uvic.ca:/home/rivera

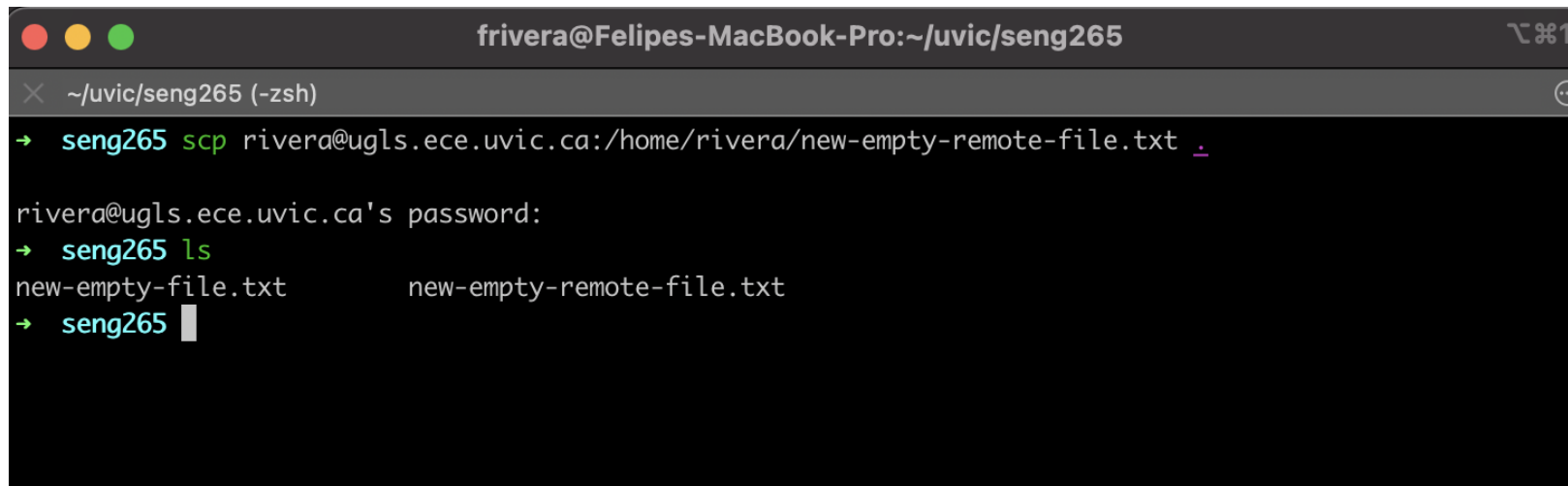
rivera@ugls.ece.uvic.ca's password:
new-empty-file.txt
100% 0 0.0KB/s 00:00
→ seng265
```

STEP VII: Retrieving files from UGLS

- i. Open the CLI used in Step III (or open a new one and execute the commands in Step II)
- ii. Execute the following command to transfer the file (i.e., new-empty-remote-file.txt) from UGLS to your local computer (replace “NETLINKID” with your actual UVic NetlinkID):

```
scp NETLINKID@ugls.ece.uvic.ca:/home/NETLINKID/new-empty-remote-file.txt .
```

- vi. Once you provide your correct password, the file will be transferred. You can then use the command **dir** (Windows) or **ls** (macOS/Linux) to see the content of the current folder and confirm that new-empty-remote-file.txt was transferred:



```
frivera@Felipes-MacBook-Pro:~/uvic/seng265
~/uvic/seng265 (-zsh)
→ seng265 scp rivera@ugls.ece.uvic.ca:/home/rivera/new-empty-remote-file.txt .
rivera@ugls.ece.uvic.ca's password:
→ seng265 ls
new-empty-file.txt      new-empty-remote-file.txt
→ seng265
```

NOTE: Make sure to include the period at the end of the scp command

STEP VIII: Assignment 0 (A0) – Part 1

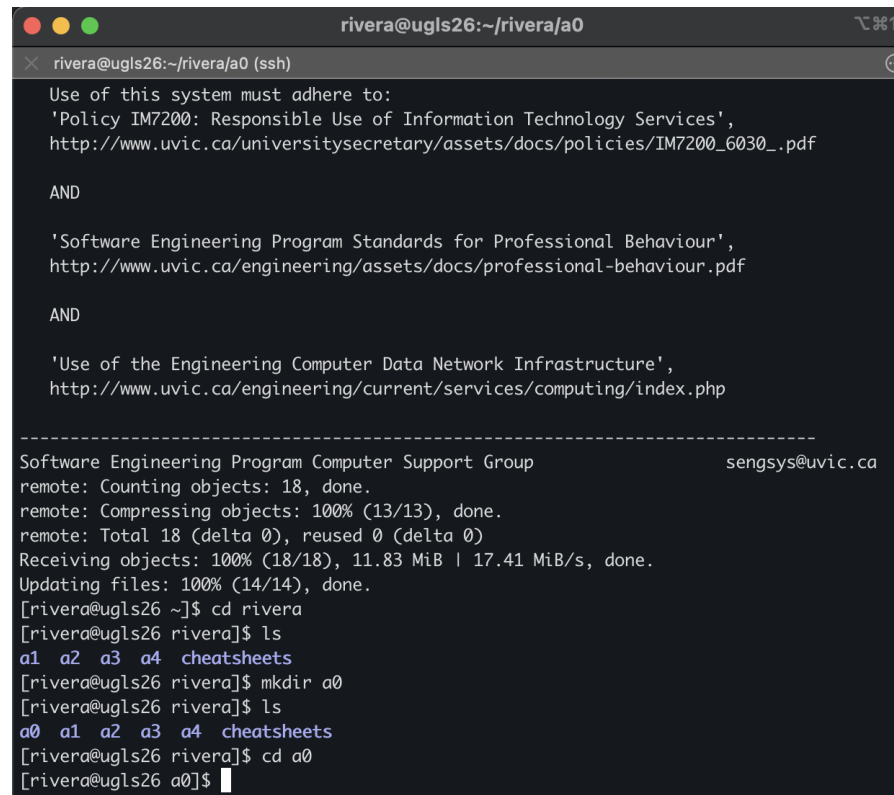
- i. Open the CLI used in STEP V (or open a new one and execute the commands in Step IV)
- ii. Execute the following two commands (replace “NETLINKID” with your actual UVic NetlinkID):

```
git clone ssh://NETLINKID@git.seng.uvic.ca/seng265/NETLINKID
```

```
cd NETLINKID
```

```
mkdir a0
```

```
cd a0
```



```
rivera@ugls26:~/rivera/a0
rivera@ugls26:~/rivera/a0 (ssh)
Use of this system must adhere to:
'Policy IM7200: Responsible Use of Information Technology Services',
http://www.uvic.ca/universitysecretary/assets/docs/policies/IM7200_6030_.pdf

AND

'Software Engineering Program Standards for Professional Behaviour',
http://www.uvic.ca/engineering/assets/docs/professional-behaviour.pdf

AND

'Use of the Engineering Computer Data Network Infrastructure',
http://www.uvic.ca/engineering/current/services/computing/index.php

-----
Software Engineering Program Computer Support Group      sengsys@uvic.ca
remote: Counting objects: 18, done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 18 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (18/18), 11.83 MiB | 17.41 MiB/s, done.
Updating files: 100% (14/14), done.
[rivera@ugls26 ~]$ cd rivera
[rivera@ugls26 rivera]$ ls
a1 a2 a3 a4 cheatsheets
[rivera@ugls26 rivera]$ mkdir a0
[rivera@ugls26 rivera]$ ls
a0 a1 a2 a3 a4 cheatsheets
[rivera@ugls26 rivera]$ cd a0
[rivera@ugls26 a0]$
```

NOTE: git commands will
be explained later in the
course

STEP IX: Assignment 0 (A0) – Part 2

- i. Open the CLI used in STEP VIII
- ii. Execute the following command to create a “Hello, World” C program:

```
echo -e "#include <stdio.h> \n int main(void){puts(\"Hello, World\");}" >> hello.c
```

- iii. Compile the program (i.e., hello.c) by executing the following command:

```
gcc hello.c -o hello
```

- iv. Run the program by executing the following command:

```
./hello
```

- v. Verify that the execution of the program produces:

NOTE: The echo command should only be used once. If you have a typo and need to run it again, you'll need to delete hello.c first with the command:

```
rm -f hello.c
```



```
rivera@ugls1:~/rivera/a0
X rivera@ugls1:~/rivera/a0 (ssh)
[rivera@ugls1 a0]$ echo -e "#include <stdio.h> \n int main(void){puts(\"Hello, World\");}" >> hello.c
[rivera@ugls1 a0]$ gcc hello.c -o hello
[rivera@ugls1 a0]$ ./hello
Hello, World
[rivera@ugls1 a0]$
```

STEP X: Assignment 0 (A0) – Part 3

- i. Open the CLI used in STEP IX
- ii. Execute the following command to create a “Hello, World” Python script:

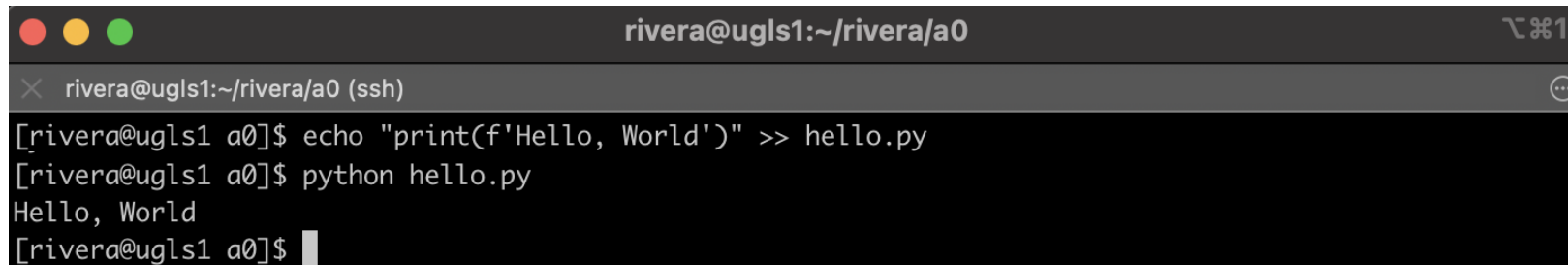
```
echo "print(f'Hello, World')" >> hello.py
```

- iii. Run the program by executing the following command:

```
python hello.py
```

- iv. Verify that the execution of the script produces:

NOTE: The echo command should only be used once. If you have a typo and need to run it again, you'll need to delete hello.c first with the command:
`rm -f hello.py`



```
rivera@ugls1:~/rivera/a0
X rivera@ugls1:~/rivera/a0 (ssh)
[rivera@ugls1 a0]$ echo "print(f'Hello, World')" >> hello.py
[rivera@ugls1 a0]$ python hello.py
Hello, World
[rivera@ugls1 a0]$
```

STEP XI: Assignment 0 (A0) – Part 4

- i. Open the CLI used in STEP X
- ii. Execute the following commands to submit your assignment:

```
cd ..
```

```
git add -A
```

```
git commit -m "a0 submission."
```

```
git push
```

Expected result from STEP XI

```
rivera@ugls1:~/rivera
```

```
X rivera@ugls1:~/rivera (ssh)
```

```
[rivera@ugls1 a0]$ cd ..  
[rivera@ugls1 rivera]$ git add -A  
[rivera@ugls1 rivera]$ git commit -m "a0 submission."  
[master 713ffe7] a0 submission.  
3 files changed, 3 insertions(+)  
create mode 100755 a0/hello  
create mode 100644 a0/hello.c  
create mode 100644 a0/hello.py  
[rivera@ugls1 rivera]$ git push  
warning: push.default is unset; its implicit value is changing in  
Git 2.0 from 'matching' to 'simple'. To squelch this message  
and maintain the current behavior after the default changes, use:  
  
    git config --global push.default matching  
  
To squelch this message and adopt the new behavior now, use:  
  
    git config --global push.default simple  
  
See 'git help config' and search for 'push.default' for further information.  
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode  
'current' instead of 'simple' if you sometimes use older versions of Git)
```

```
-----  
| ____ | _ _ _ _ _ ( ) _ _ _ _ _ ( ) _ _ _ _ _  
| _ | | ' \ / _ | | | ' \ / _ | | | ' \ / _ | | | | | | | | | | |
| | _ | | | | | ( | | | | | | _ | | | | | | ( |  
| _____ | | \ , | | | | | | \ _ | | | | | \ _ |  
      | _/_/                      | _/_/
```

```
-----
```

```
rivera@ugls1:~/rivera
```

```
x rivera@ugls1:~/rivera (ssh)
```

```
|_| |'| \ / _' |||| '| \ / _\| |_| || '| \ / _'  
|_|_| |'|| |( |)|| | | | | _/| _/| | | | | | (|  
|_|_|_|_| |'| \_, |||| |'| \_|| \_||_| | | | | | \_, |  
      |_/_/                      |_/_/
```

Use of this system must adhere to:
'Policy IM7200: Responsible Use of Information Technology Services',
http://www.uvic.ca/universitysecretary/assets/docs/policies/IM7200_6030_.pdf

AND


'Software Engineering Program Standards for Professional Behaviour',
<http://www.uvic.ca/engineering/assets/docs/professional-behaviour.pdf>

AND

'Use of the Engineering Computer Data Network Infrastructure',
<http://www.uvic.ca/engineering/current/services/computing/index.php>

Software Engineering Program Computer Support Group
Counting objects: 8, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 2.74 KiB | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To ssh://rivera@git.seng.uvic.ca/seng265/rivera
eb2c265..713ffe7 master -> master
[rivera@ugls1 rivera]\$

sengsys@uvic.ca



University
of Victoria



STEP XII: Validate A0 Submission

SENG265 DOES NOT use Brightspace for assignment submissions. Consequently, you must learn how to confirm that your assignment was submitted correctly.

- Open the CLI used in STEP XI (or a new one and connect to UGLS using the commands in STEP IV)
- Execute the following commands (use the proper NetlinkID):

```
cd ~
```

```
git clone ssh://NETLINKID@git.seng.uvic.ca/seng265/NETLINKID remote
```

```
ls remote/a0
```

Make sure you get a result like: 

A0 requires those 3 files displayed in the output of the last command

```
rivera@ugls1:~  
[rivera@ugls1 ~]$ cd ~  
[rivera@ugls1 ~]$ git clone ssh://rivera@git.seng.uvic.ca/seng265/rivera remote  
Cloning into 'remote'...  
  
-----  
Use of this system must adhere to:  
'Policy IM7200: Responsible Use of Information Technology Services',  
http://www.uvic.ca/universitysecretary/assets/docs/policies/IM7200\_6030\_.pdf  
  
AND  
  
'Software Engineering Program Standards for Professional Behaviour',  
http://www.uvic.ca/engineering/assets/docs/professional-behaviour.pdf  
  
AND  
  
'Use of the Engineering Computer Data Network Infrastructure',  
http://www.uvic.ca/engineering/current/services/computing/index.php  
  
-----  
Software Engineering Program Computer Support Group  
remote: Counting objects: 16, done.  
remote: Compressing objects: 100% (10/10), done.  
remote: Total 16 (delta 1), reused 0 (delta 0)  
Receiving objects: 100% (16/16), done.  
Resolving deltas: 100% (1/1), done.  
[rivera@ugls1 ~]$ ls remote/a0  
hello hello.c hello.py  
[rivera@ugls1 ~]$
```


QUESTIONS?