COMP30670
Software Engineering (Conversion)

Server Workflow
09/03/2016

Dr. Aonghus Lawlor

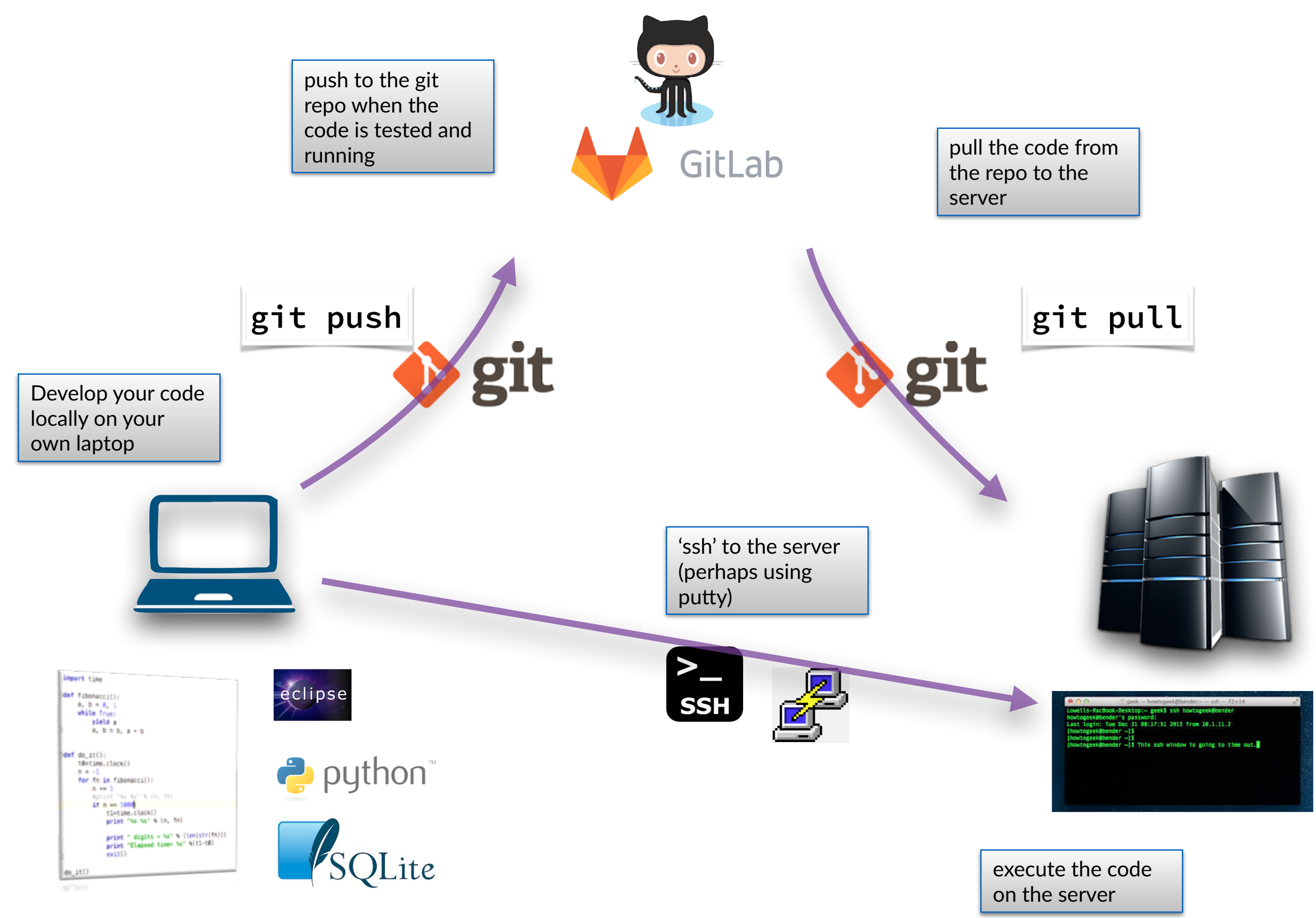
aonghus.lawlor@insight-centre.org

Workflow

Title Text

- There are many possible ways to work with the server
- Here is one possible workflow

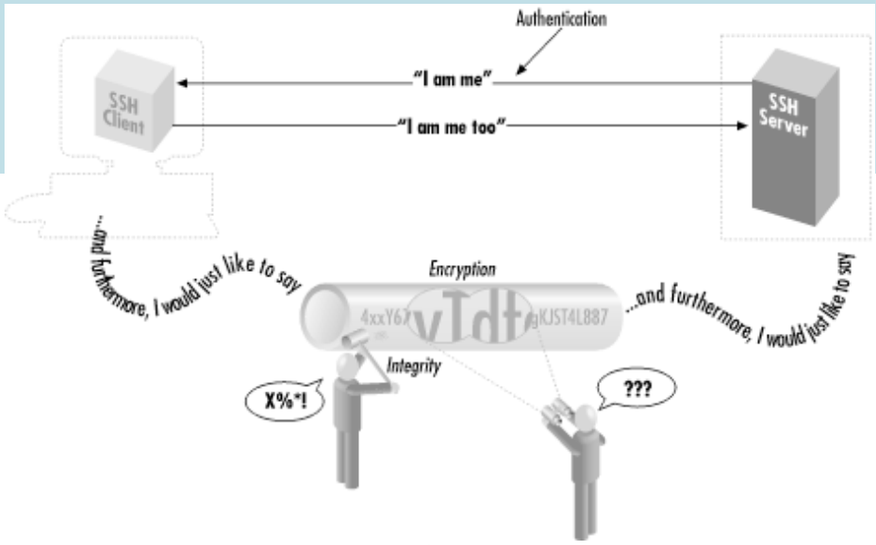
- Use your own machine to develop your code
- This includes writing the code and testing
- Push the code to your git repo (git.ucd.ie)
- SSH to server
- Run a ‘git clone’/‘git pull’ on the server to update the code
- Execute the code on the server


push to the git repo when the code is tested and running
GitLab
pull the code from the repo to the server
git push
git
git pull
git
Develop your code locally on your own laptop
'ssh' to the server (perhaps using putty)
>_
SSH
eclipse
python
SQLite
execute the code on the server

Server

- The server is required to collect the data continuously for a few days.
- But, you may be able to manage this data collection on your own machine, if you can leave it on continuously.
- The server is really only required to collect/store the data and will not be necessary after that (although you may still find it useful)
- There are a number of places to get access to a linux server:
  - Microsoft offer a free trial on Azure with hosting of a variety of virtual machines (linux and windows)
  - Amazon also offer free servers, checkout EC2- this should be more than enough capacity for this project
  - If you have not set these up before the overhead can seem quite high, so you can contact me for an account on one of my linux servers.

Server access



- SSH
- ssh is a protocol which specifies how to conduct secure communication over a network
- Authentication
  - Reliably determines someone's identity. If you try to log into an account on a remote computer, SSH asks for digital proof of your identity. If you pass the test, you may log in; otherwise SSH rejects the connection.
- Encryption
  - Scrambles data so it is unintelligible except to the intended recipients. This protects your data as it passes over the network.
- Integrity
- Guarantees the data traveling over the network arrives unaltered. If a third party captures and modifies your data in transit, SSH detects this fact.

SSH terms

Local computer (local host, local machine)	A computer on which you are logged in and, typically, running an SSH client.
Remote computer (remote host, remote machine)	A second computer you contact from your local computer. Typically, the remote computer is running an SSH server and is contacted via an SSH client. As a degenerate case, the local and remote computers can be the same machine.
Local user	A user logged into a local computer.
Remote user	A user logged into a remote computer.
Server	An SSH server program.
Server machine	A computer running an SSH server program. We will sometimes simply write "server" for the server machine when the context makes clear (or irrelevant) the distinction between the running SSH server program and its host machine.
Client	An SSH client program.
Client Machine	A computer running an SSH client. As with the server terminology, we will simply write "client" when the context makes the meaning clear.
~ or $HOME	A user's home directory on a Unix machine, particularly when used in a file path such as ~/filename. Most shells recognize ~ as a user's home directory, with the notable exception of Bourne shell. $HOME is recognized by all shells.

OSX Terminal

- OSX comes with a command line ssh client program
- open /Applications/Terminal.app
- the ssh command takes lots of options, but you will not need most of them- type

```
man ssh
```

for more information

aonghus — less ◂ man ssh — 98×17

```
NAME
     ssh -- OpenSSH SSH client (remote login program)

SYNOPSIS
     ssh [-1246AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
         [-D [bind_address:]port] [-E log_file] [-e escape_char] [-F configfile]
         [-I pkcs11] [-i identity_file] [-L address] [-l login_name] [-m mac_spec]
         [-O ctl_cmd] [-o option] [-p port] [-Q query_option] [-R address] [-S ctl_path]
         [-W host:port] [-w local_tun[:remote_tun]] [user@]hostname [command]

DESCRIPTION
     ssh (SSH client) is a program for logging into a remote machine and for executing
     commands on a remote machine.  It is intended to provide secure encrypted communi-
     cations between two untrusted hosts over an insecure network.  X11 connections,
     arbitrary TCP ports and UNIX-domain sockets can also be forwarded over the secure
:
```

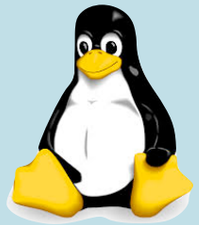
SSH Windows

- Windows does not come with an ssh client by default

- Putty is a good open source ssh client

- There are many other options too, but putty will do the job

SSH command line



The general format of the command is

```
ssh user@hostname -p 22
```

port: the port number of the ssh server- defaults to 22

hostname: the name of the server machine you are logging into (eg. buzzer.ucd.ie)

user: user name on the server machine- defaults to local user name

SSH

```
$ ssh alawlor@buzzer.ucd.ie
```

aonghus — ssh alawlor@buzzer.ucd.ie — 98×17

```
~
aonghus@aonghusmac$ ssh alawlor@buzzer.ucd.ie
alawlor@buzzer.ucd.ie's password:
```

password is not shown

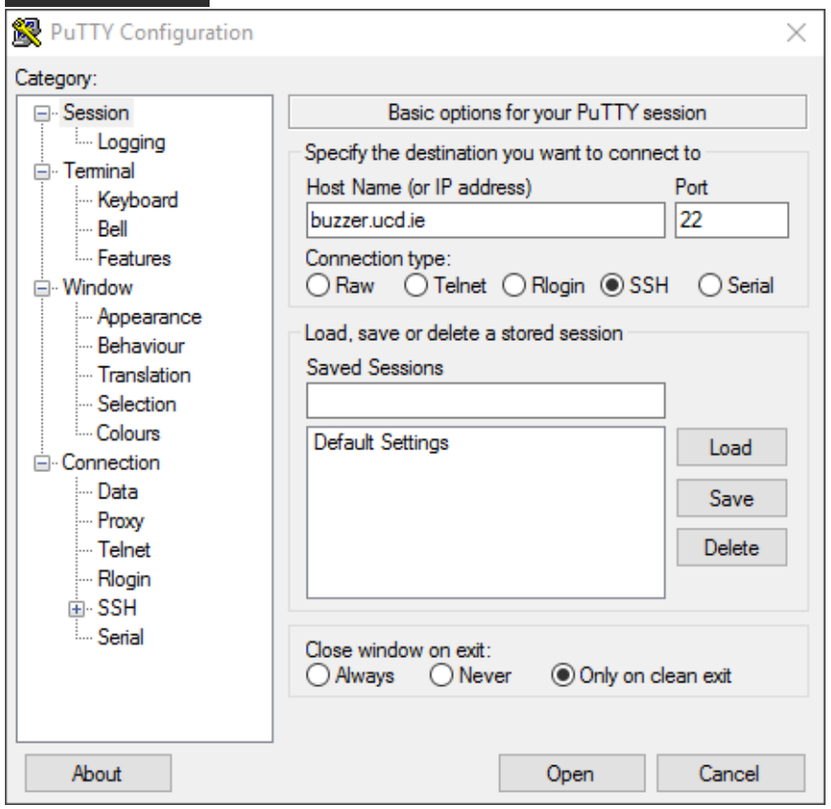
aonghus — ssh alawlor@buzzer.ucd.ie — 98×17

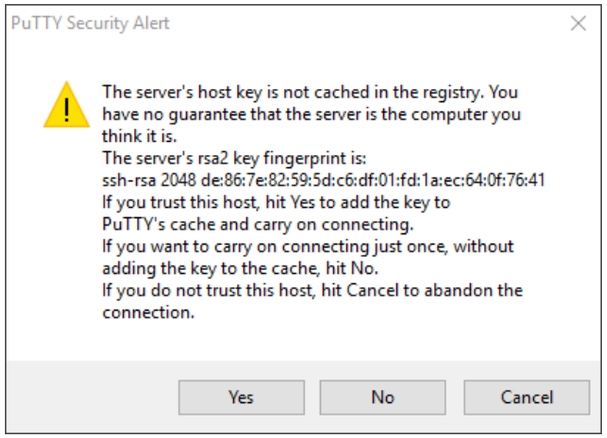
```
~
aonghus@aonghusmac$ ssh alawlor@buzzer.ucd.ie
alawlor@buzzer.ucd.ie's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-63-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

*** System restart required ***
Last login: Wed Mar  9 11:11:46 2016 from dhcp-892b81d5.ucd.ie
~
alawlor@buzzer$
```

the login screen shows basic information about the system (version etc) and last login

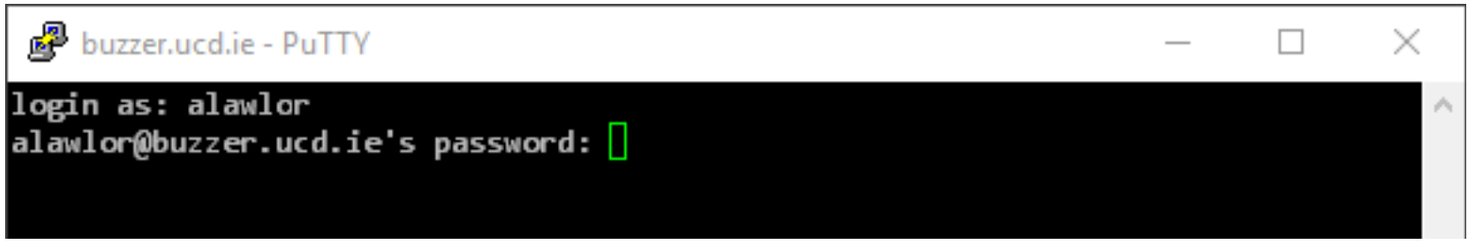
SSH Windows

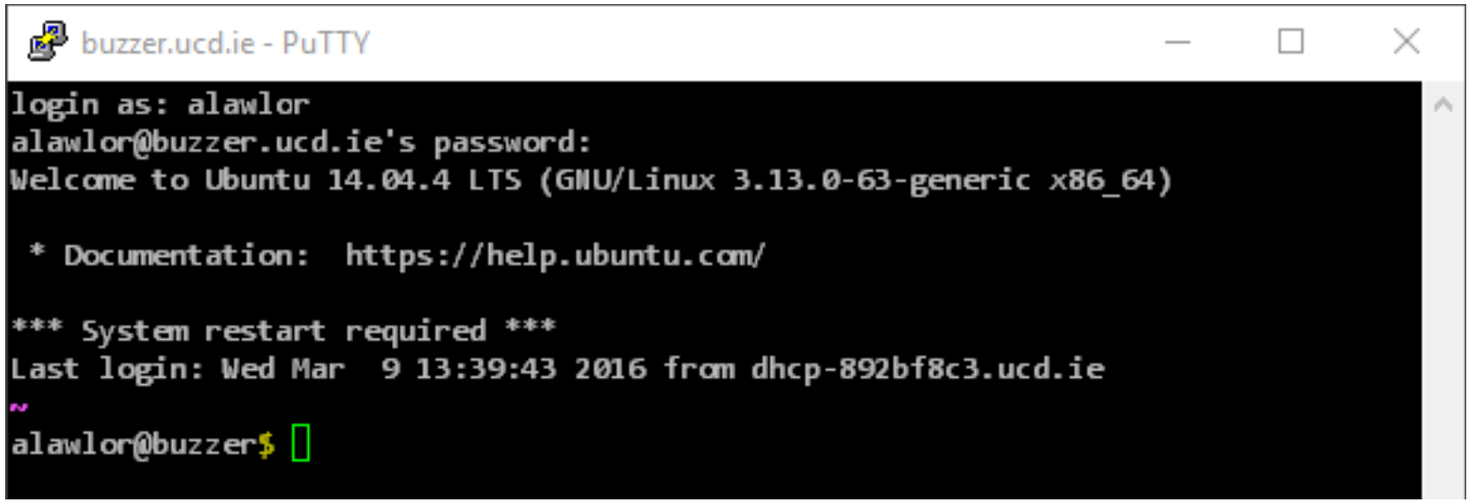

putty


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
buzzer.ucd.ie
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel


PuTTY Security Alert
The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 de:86:7e:82:59:5d:c6:df:01:fd:1a:ec:64:0f:76:41
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.
Yes
No
Cancel


buzzer.ucd.ie - PuTTY
login as: alawlor


buzzer.ucd.ie - PuTTY
login as: alawlor
alawlor@buzzer.ucd.ie's password:


buzzer.ucd.ie - PuTTY
login as: alawlor
alawlor@buzzer.ucd.ie's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-63-generic x86_64)
* Documentation: https://help.ubuntu.com/
*** System restart required ***
Last login: Wed Mar 9 13:39:43 2016 from dhcp-892bf8c3.ucd.ie
~
alawlor@buzzer$

Command line (linux)

- Check out LinuxCommand
- Basic Shell Commands
- Intro to Shell Commands

Navigation

Linux Command	DOS Command	Description
`pwd`	`cd`	“Print Working Directory”. Shows the current location in the directory tree.
`cd`	`cd, chdir`	“Change Directory”. When typed all by itself, it returns you to your home directory.
`cd directory`	`cd directory`	Change into the specified directory name. Example: `cd /tmp`
`cd ~`		“~” is an alias for your home directory. It can be used as a shortcut to your “home”, or other directories relative to your home.
`cd ..`	`cd ..`	Move up one directory. For example, if you are in `/home/vic` and you type “`cd ..`”, you will end up in `/home`.
`cd -`		Return to previous directory. An easy way to get back to your previous location!
`ls`	`dir /w`	List all files in the current directory, in column format.
`ls directory`	`dir directory`	List the files in the specified directory. Example: `ls /var/log`
`ls -l`	`dir`	List files in “long” format, one file per line. This also shows you additional info about the file, such as ownership, permissions, date, and size.
`ls ./data/*.txt`	`dir data/*.txt`	List all files whose names begin with the letter “d” in the `/usr/bin` directory.

Files & Directories

Linux Command	DOS Command	Description
`file`		Find out what kind of file it is. For example, "`file /bin/ls`" tells us that it is a Linux executable file.
`cat`	`type`	Display the contents of a text file on the screen. For example: `cat readme.txt` would display the file.
`head`		Display the first few lines of a text file. Example: `head /etc/services`
`tail`		Display the last few lines of a text file. Example: `tail /etc/services`
`tail -f`		Display the last few lines of a text file, and then output appended data as the file grows (very useful for following log files!). Example: tail -f /var/log/messages
`cp`	`copy`	Copies a file from one location to another. Example: `cp mp3files.txt /tmp` (copies the mp3files.txt file to the /tmp directory)
`mv`	`rename, res, move`	Moves a file to a new location, or renames it. For example: `mv mp3files.txt /tmp` (copy the file to /tmp, and delete it from the original location)
`rm`	`del`	Delete a file. Example: `rm /tmp/mp3files.txt`
`mkdir`	`md`	Make Directory. Example: `mkdir /tmp/myfiles/`
`rmdir`	`rd, rmdir`	Remove Directory. Example: `rmdir /tmp/myfiles/`

Screen

Sometimes you are connected to your server with SSH and in the middle of some long-running task, and suddenly your connection drops for some reason, and you lose your work. A small utility called *screen* allows you to reattach to a previous session so that you can finish your task. Tutorial here

```
$ screen -S work
```

Creates a new screen session called work. Looks exactly like your existing console.

```
$ screen -ls
```

get a list of open screen sessions

```
alawlor@buzzer$ screen -ls
There is a screen on:
        16015.work      (10/03/16 00:01:38)     (Attached)
1 Socket in /var/run/screen/S-alawlor.
```

```
$ screen -r
```

when you log in the next time, run 'screen -r' to resume the screen session. You should see the console just where you left it

detach a screen session

```
"Ctrl-a" "d"
```

```
alawlor@buzzer$
[detached from 16015.work]
```

re-attach a screen session

```
$ screen -r 16015.work
```

Using Multiple Screens

When you need more than 1 screen to do your job, is it possible? Yes it is. You can run multiple screen window at the same time. There are 2 (two) ways to do it. First, you can detach the first screen and the run another screen on the real terminal. Second, you do nested screen.

Switching between screens

When you use nested screens, you can switch between screens using command "Ctrl-A" and "n". This will move to the next screen. When you need to go to the previous screen, just press "Ctrl-A" and "p". To create a new screen window, just press "Ctrl-A" and "c".

Setting up

```
$ which python2 python3
```

```
aonghus — s
~
alawlor@buzzer$ which python2 python3
/usr/bin/python2
/usr/bin/python3
```

both python2 and python3 are installed, you can use either

```
$ virtualenv --python=python2 venv2
```

```
alawlor@buzzer$ virtualenv --python=python2 venv2
Running virtualenv with interpreter /usr/bin/python2
New python executable in venv2/bin/python2
Also creating executable in venv2/bin/python
Installing setuptools, pip...done.
```

```
uzzer$ virtualenv --python=python3 venv3
Running virtualenv with interpreter /usr/bin/pythor
/usr'
le in venv3/bin/python3
table in venv3/bin/python
ls, pip...done.
```

```
$ source venv2/bin/activate
```

```
alawlor@buzzer$ source venv2/bin/activate
(venv2)~
```

activate your virtualenv
do this every time you log in, or put something in your .bashrc to do this for you automatically

```
$ python -V
```

```
alawlor@buzzer$ python -V
Python 2.7.6
(venv2)~
```

```
$ pip install requests
```

```
alawlor@buzzer$ pip install requests
Downloading/unpacking requests
  Downloading requests-2.9.1-py2.py3-none-any.whl (501kB): 501kB downloaded
Installing collected packages: requests
Successfully installed requests
Cleaning up...
(venv2)~
```

use pip to install whatever packages you need

Using Conda

to install conda use this command (I already downloaded the installer)

```
$ bash ~alawlor/Anaconda2-2.5.0-Linux-x86_64.sh
```

```
alawlor@buzzer$ bash ~alawlor/Anaconda2-2.5.0-Linux-x86_64.sh

Welcome to Anaconda2 2.5.0 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

```
Do you approve the license terms? [yes|no]
>>> yes
```

```
Anaconda2 will now be installed into this location:
/home/alawlor/anaconda2

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/alawlor/anaconda2] >>>
```

```
installing: datashape-0.5.0-py27_0 ...
installing: decorator-4.0.6-py27_0 ...
installing: docutils-0.12-py27_0 ...
installing: dynd-python-0.7.1-py27_0 ...
```

```
Python 2.7.11 :: Continuum Analytics, Inc.
creating default environment...
installation finished.
Do you wish the installer to prepend the Anaconda2 install location
to PATH in your /home/alawlor/.bashrc ? [yes|no]
[no] >>> yes
```

Conda

```
$ conda --version
```

conda version information

```
$ conda update conda
```

if you need to update conda

```
$ conda info --envs
```

get a list of virtualenvs you have created

Now conda is installed you can:
Create and activate an environment
Use the conda create command, followed by any name you wish to call it:

```
$ conda create --name venv pip
```

```
$ source activate venv
```

after creating the virtualenv, activate it

```
#
# To activate this environment, use:
# $ source activate venv
#
# To deactivate this environment, use:
# $ source deactivate
#
~
alawlor@buzzer$ source activate venv
discarding /home/alawlor/anaconda2/bin from PATH
prepending /home/alawlor/anaconda2/envs/venv/bin to PATH
```

conda is using an up-to-date python, good!

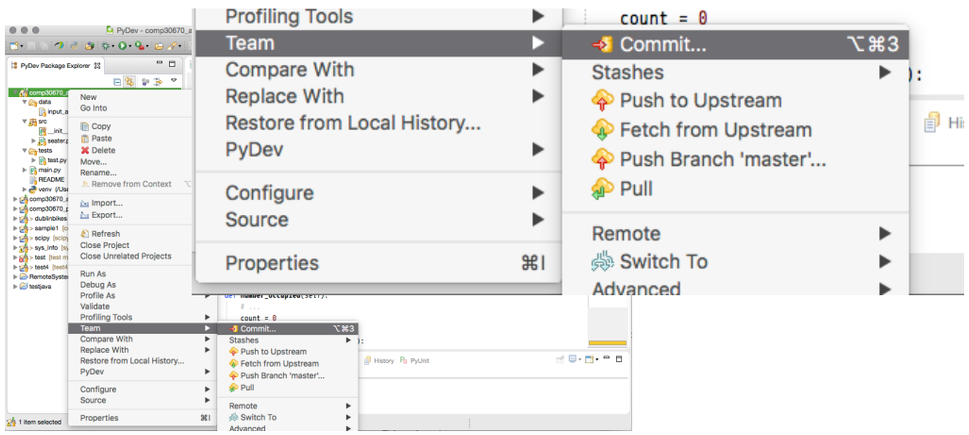
```
alawlor@buzzer$ python -V
Python 2.7.11 :: Continuum Analytics, Inc.
(venv)~
```

get a list of virtualenvs you have installed

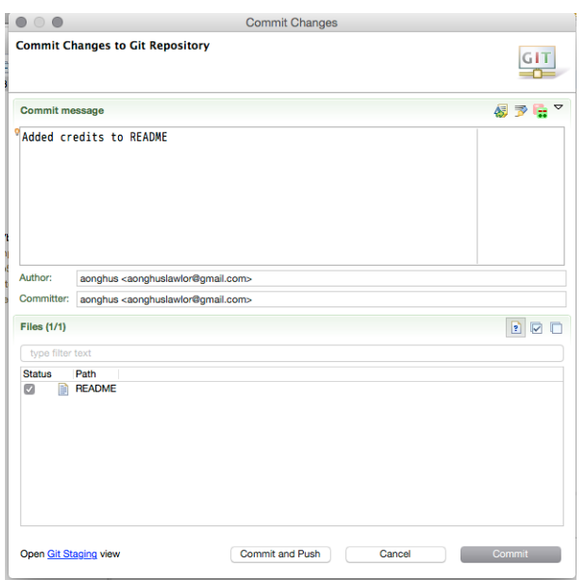
```
alawlor@buzzer$ conda info --envs
Using Anaconda Cloud api site https://api.anaconda.org
# conda environments:
#
venv                  *  /home/alawlor/anaconda2/envs/venv
root                     /home/alawlor/anaconda2
```

Git

on your local machine:

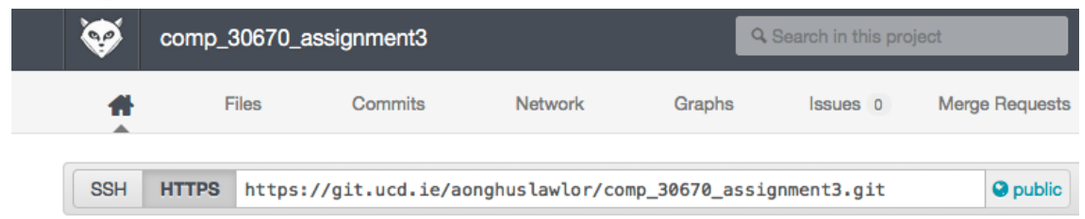


in Eclipse, do “commit and push”



on the server:

git.ucd.ie




```
git clone https://git.ucd.ie/aonghuslawlor/comp_30670_assignment3.git
```

```
aonghus — ssh alawlor@buzzer.ucd.ie — 98×17
(venv)~
alawlor@buzzer$ git clone https://git.ucd.ie/aonghuslawlor/comp_30670_assignment3.git
Cloning into 'comp_30670_assignment3'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 22 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (22/22), done.
Checking connectivity... done.
(venv)~
alawlor@buzzer$
```

Workflow

```
$ git clone https://git.ucd.ie/aonghuslawlor/comp_30670_assignment3.git
```

clone the repository

```
aonghus — ssh alawlor@buzzer.ucd.ie — 98×17
(venv)~
alawlor@buzzer$ git clone https://git.ucd.ie/aonghuslawlor/comp_30670_assignment3.git
Cloning into 'comp_30670_assignment3'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 22 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (22/22), done.
Checking connectivity... done.
(venv)~
alawlor@buzzer$
```

```
$ which python
```

check which virtualenv you are using (activate if necessary)

```
alawlor@buzzer$ which python
/home/alawlor/anaconda2/envs/venv/bin/python
```

change into the source directory

```
alawlor@buzzer$ cd comp_30670_assignment3/
```

```
alawlor@buzzer$ ls
data/  main.py  README  src/  tests/
```

run your program...

```
alawlor@buzzer$ python ./main.py
```

if you want the program to continue running after you log out, then put '**nohup**' before the command and '**&**' after.

```
alawlor@buzzer$ nohup python ./main.py &
```