

# COMP30670

## Software Engineering (Conversion)

15 - Project & DB  
27/03/2017

*Dr. Aonghus Lawlor*

[aonghus.lawlor@insight-centre.org](mailto:aonghus.lawlor@insight-centre.org)



# Timetables

Lecture	Monday	11.00	50mins	B003 (CSI)
Lecture	Friday	10.00	50mins	B003 (CSI)
Practical	Monday	13.00	110mins	B003
Practical	Friday	11.00	110mins	B003

# Project Overview

# Project Overview

## Product Backlog

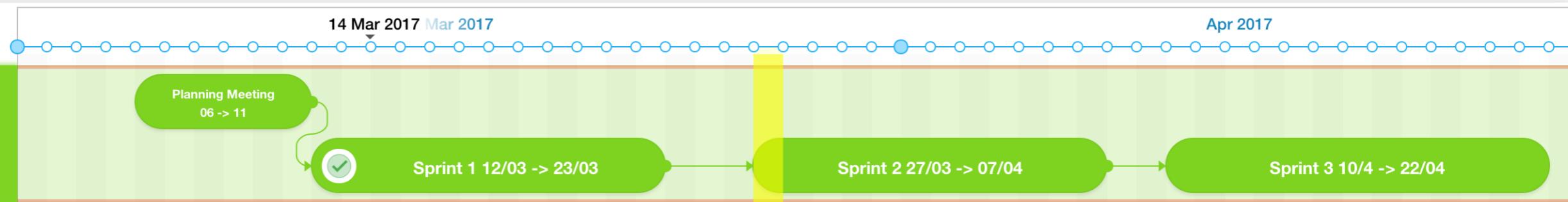
select station  
display weather  
heat map of occupancy/  
availability  
...



scraping from API  
setup

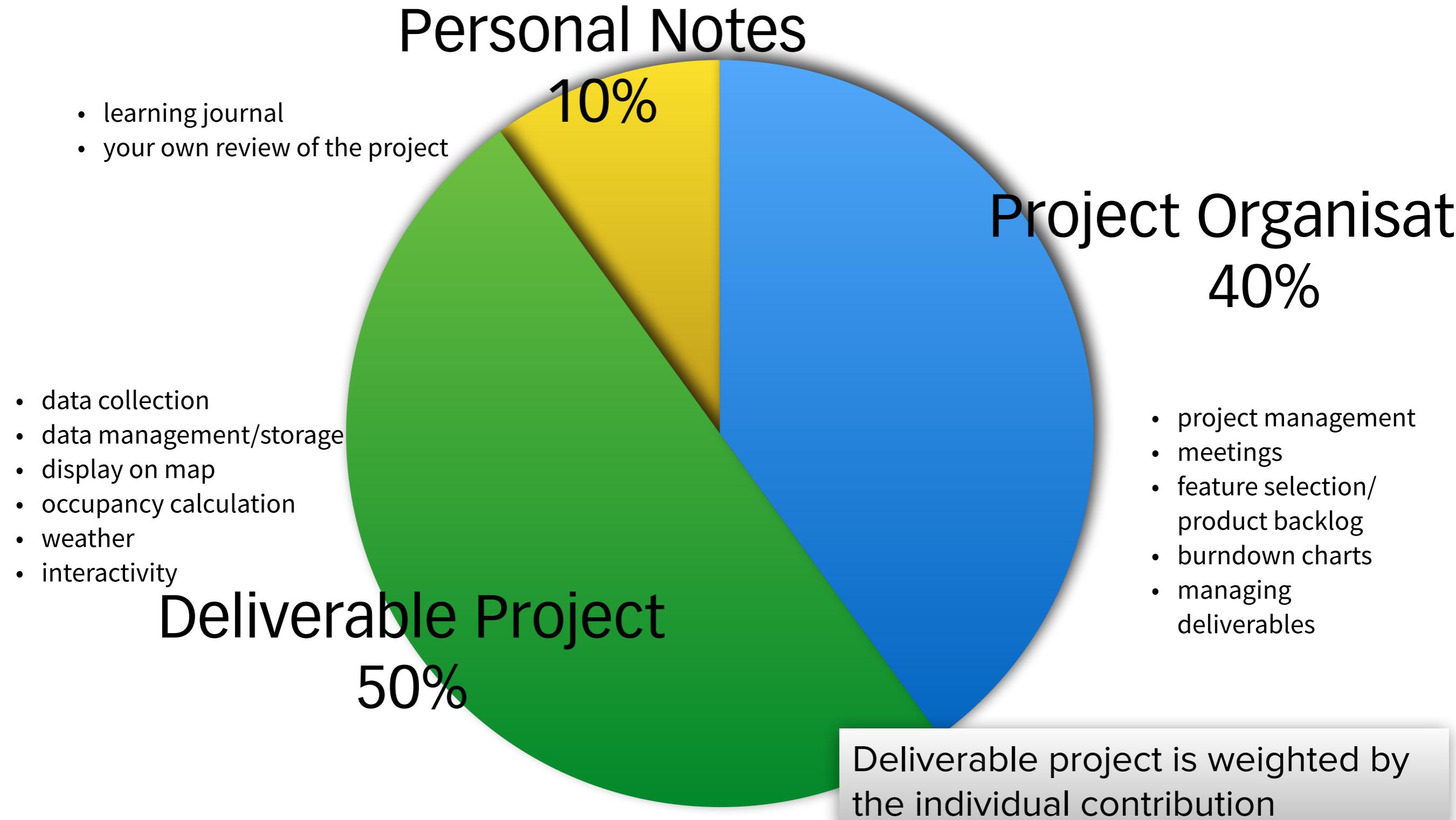
flask  
web app  
html/js

better analytics  
interaction with web app  
design...



# Group Assessment

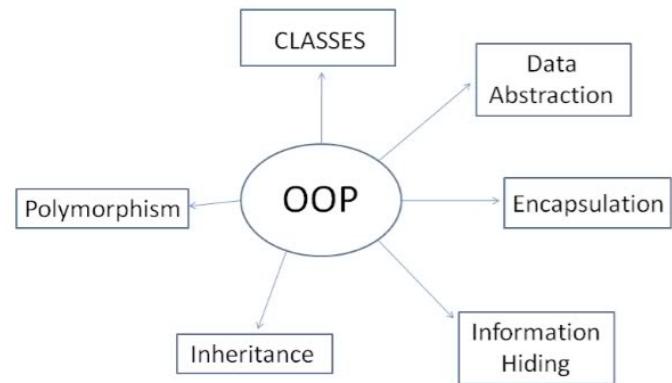
# Marking



# Outline

<b>27 Mar 2017</b>	<b>29</b>	Mon	11:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>29</b>	Mon	13:00	110mins	1	<b>24919</b>	<b>PRA</b>
	<b>29</b>	Fri	10:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>29</b>	Fri	11:00	110mins	1	<b>24919</b>	<b>PRA</b>
<b>03 Apr 2017</b>	<b>30</b>	Mon	11:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>30</b>	Mon	13:00	110mins	1	<b>24919</b>	<b>PRA</b>
	<b>30</b>	Fri	10:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>30</b>	Fri	11:00	110mins	1	<b>24919</b>	<b>PRA</b>
<b>10 Apr 2017</b>	<b>31</b>	Mon	11:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>31</b>	Mon	13:00	110mins	1	<b>24919</b>	<b>PRA</b>
	<b>31</b>	Fri	10:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>31</b>	Fri	11:00	110mins	1	<b>24919</b>	<b>PRA</b>
<b>17 Apr 2017</b>	<b>32</b>	Mon	11:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>32</b>	Mon	13:00	110mins	1	<b>24919</b>	<b>PRA</b>
	<b>32</b>	Fri	10:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>32</b>	Fri	11:00	110mins	1	<b>24919</b>	<b>PRA</b>
<b>24 Apr 2017</b>	<b>33</b>	Mon	11:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>33</b>	Mon	13:00	110mins	1	<b>24919</b>	<b>PRA</b>
	<b>33</b>	Fri	10:00	50mins	1	<b>24919</b>	<b>LEC</b>
	<b>33</b>	Fri	11:00	110mins	1	<b>24919</b>	<b>PRA</b>

# Future Areas to Cover

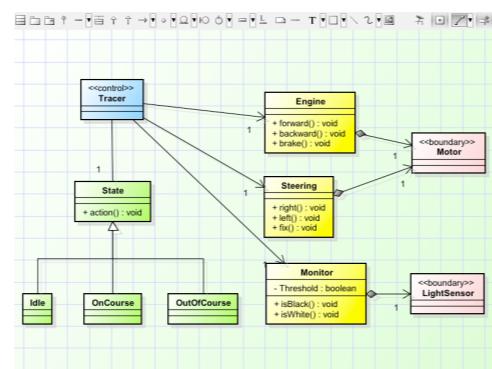


- Software Architecture
- Software Design
- Web Apps/Flask

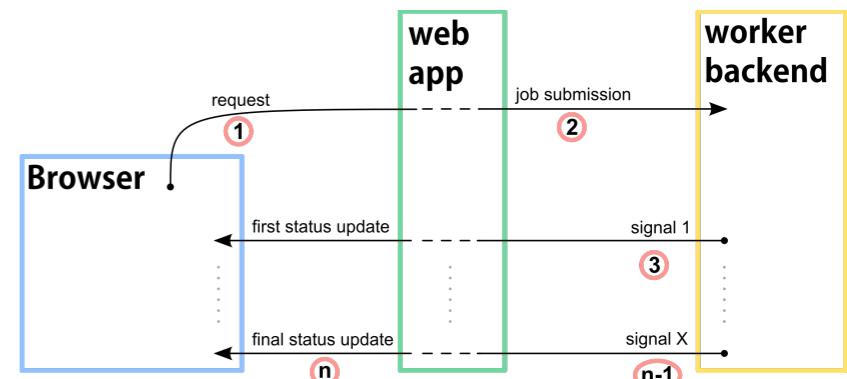
## Object Oriented Python



## Databases/Storage



## Object Oriented Python



## client/server backend

# Workflow

# Workflow

- 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 ([github.com](https://github.com))
  - 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

pull the code from  
the repo to the  
server

**git push**

Develop your code  
locally on your  
own laptop



**git**

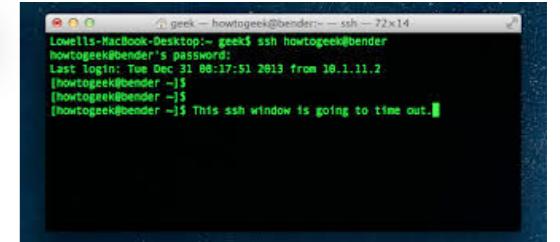
**git**

**git pull**



'ssh' to the server  
(perhaps using  
putty)

>  
—  
SSH

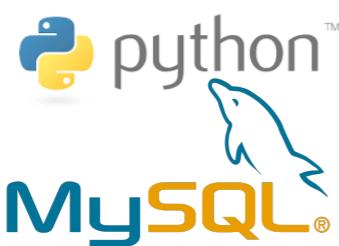


```
import time
def Fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

def do_it():
    t0 = time.clock()
    n = -1
    for f in Fibonacci():
        n += 1
        if n == 1000:
            t1 = time.clock()
            print "%d %s" % (n, f)
            print "Elapsed time %s" % (t1-t0)
            exit()

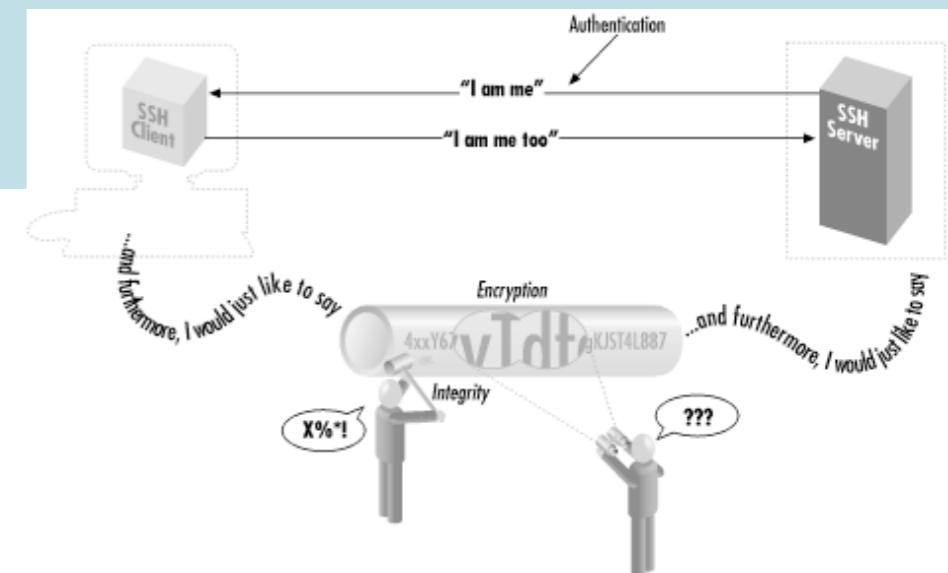
        print "%d %s" % (len(str(f)))
        print "Elapsed time %s" % (t1-t0)
        exit()

do_it()
```



execute the code  
on the server

# 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 — 98x17
```

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

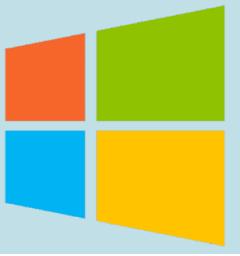
**SYNOPSIS**

```
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 communications between two untrusted hosts over an insecure network. X11 connections, arbitrary TCP ports and UNIX-domain sockets can also be forwarded over the secure

:

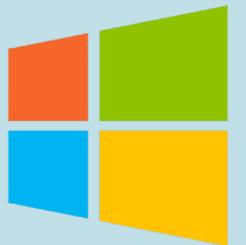


# SSH Windows

- 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



# 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, “ <code>file /bin/ls</code> ” tells us that it is a Linux executable file.
cat	type	Display the contents of a text file on the screen. For example: <code>cat readme.txt</code> would display the file.
head		Display the first few lines of a text file. Example: <code>head /etc/services</code>
tail		Display the last few lines of a text file. Example: <code>tail /etc/services</code>
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: <code>tail -f /var/log/messages</code>
cp	copy	Copies a file from one location to another. Example: <code>cp mp3files.txt /tmp</code> (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: <code>mv mp3files.txt /tmp</code> (copy the file to /tmp, and delete it from the original location)
rm	del	Delete a file. Example: <code>rm /tmp/mp3files.txt</code>
mkdir	md	Make Directory. Example: <code>mkdir /tmp/myfiles/</code>
rmdir	rd, rmdir	Remove Directory. Example: <code>rmdir /tmp/myfiles/</code>

# Linux Commands

## File Commands

**ls** - directory listing  
**ls -al** - formatted listing with hidden files  
**cd dir** - change directory to *dir*  
**cd** - change to home  
**pwd** - show current directory  
**mkdir dir** - create a directory *dir*  
**rm file** - delete *file*  
**rm -r dir** - delete directory *dir*  
**rm -f file** - force remove *file*  
**rm -rf dir** - force remove directory *dir*\*  
**cp file1 file2** - copy *file1* to *file2*  
**cp -r dir1 dir2** - copy *dir1* to *dir2*; create *dir2* if it doesn't exist  
**mv file1 file2** - rename or move *file1* to *file2*  
if *file2* is an existing directory, moves *file1* into directory *file2*  
**ln -s file link** - create symbolic link *link* to *file*  
**touch file** - create or update *file*  
**cat > file** - places standard input into *file*  
**more file** - output the contents of *file*  
**head file** - output the first 10 lines of *file*  
**tail file** - output the last 10 lines of *file*  
**tail -f file** - output the contents of *file* as it grows, starting with the last 10 lines

## System Info

**date** - show the current date and time  
**cal** - show this month's calendar  
**uptime** - show current uptime  
**w** - display who is online  
**whoami** - who you are logged in as  
**finger user** - display information about *user*  
**uname -a** - show kernel information  
**cat /proc/cpuinfo** - cpu information  
**cat /proc/meminfo** - memory information  
**man command** - show the manual for *command*  
**df** - show disk usage  
**du** - show directory space usage  
**free** - show memory and swap usage  
**whereis app** - show possible locations of *app*  
**which app** - show which *app* will be run by default

## Searching

**grep pattern files** - search for *pattern* in *files*  
**grep -r pattern dir** - search recursively for *pattern* in *dir*  
**command | grep pattern** - search for *pattern* in the output of *command*  
**locate file** - find all instances of *file*

## Compression

**tar cf file.tar files** - create a tar named *file.tar* containing *files*  
**tar xf file.tar** - extract the files from *file.tar*  
**tar czf file.tar.gz files** - create a tar with Gzip compression  
**tar xzf file.tar.gz** - extract a tar using Gzip  
**tar cjf file.tar.bz2** - create a tar with Bzip2 compression  
**tar xjf file.tar.bz2** - extract a tar using Bzip2  
**gzip file** - compresses *file* and renames it to *file.gz*  
**gzip -d file.gz** - decompresses *file.gz* back to *file*

## Process Management

**ps** - display your currently active processes  
**top** - display all running processes  
**kill pid** - kill process id *pid*  
**killall proc** - kill all processes named *proc*\*  
**bg** - lists stopped or background jobs; resume a stopped job in the background  
**fg** - brings the most recent job to foreground  
**fg n** - brings job *n* to the foreground

## INSTALL GIT

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

### GitHub for Windows

<https://windows.github.com>

### GitHub for Mac

<https://mac.github.com>

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

### Git for All Platforms

<http://git-scm.com>

## CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```

Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```

Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```

Enables helpful colorization of command line output

## CREATE REPOSITORIES

Start a new repository or obtain one from an existing URL

```
$ git init [project-name]
```

Creates a new local repository with the specified name

```
$ git clone [url]
```

Downloads a project and its entire version history

## MAKE CHANGES

Review edits and craft a commit transaction

```
$ git status
```

Lists all new or modified files to be committed

```
$ git diff
```

Shows file differences not yet staged

```
$ git add [file]
```

Snapshots the file in preparation for versioning

```
$ git diff --staged
```

Shows file differences between staging and the last file version

```
$ git reset [file]
```

Unstages the file, but preserve its contents

```
$ git commit -m "[descriptive message]"
```

Records file snapshots permanently in version history

## GROUP CHANGES

Name a series of commits and combine completed efforts

```
$ git branch
```

Lists all local branches in the current repository

```
$ git branch [branch-name]
```

Creates a new branch

```
$ git checkout [branch-name]
```

Switches to the specified branch and updates the working directory

```
$ git merge [branch]
```

Combines the specified branch's history into the current branch

```
$ git branch -d [branch-name]
```

Deletes the specified branch

## REFACTOR FILENAMES

Relocate and remove versioned files

**\$ git rm [file]**

Deletes the file from the working directory and stages the deletion

**\$ git rm --cached [file]**

Removes the file from version control but preserves the file locally

**\$ git mv [file-original] [file-renamed]**

Changes the file name and prepares it for commit

## SUPPRESS TRACKING

Exclude temporary files and paths

**\*.log  
build/  
temp-\***

A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

**\$ git ls-files --other --ignored --exclude-standard**

Lists all ignored files in this project

## SAVE FRAGMENTS

Shelve and restore incomplete changes

**\$ git stash**

Temporarily stores all modified tracked files

**\$ git stash pop**

Restores the most recently stashed files

**\$ git stash list**

Lists all stashed changesets

**\$ git stash drop**

Discards the most recently stashed changeset

## REVIEW HISTORY

Browse and inspect the evolution of project files

**\$ git log**

Lists version history for the current branch

**\$ git log --follow [file]**

Lists version history for a file, including renames

**\$ git diff [first-branch]...[second-branch]**

Shows content differences between two branches

**\$ git show [commit]**

Outputs metadata and content changes of the specified commit

## REDO COMMITS

Erase mistakes and craft replacement history

**\$ git reset [commit]**

Undoes all commits after `[commit]`, preserving changes locally

**\$ git reset --hard [commit]**

Discards all history and changes back to the specified commit



# Screen

# 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 run another screen on the real terminal. Second, you do nested screen.

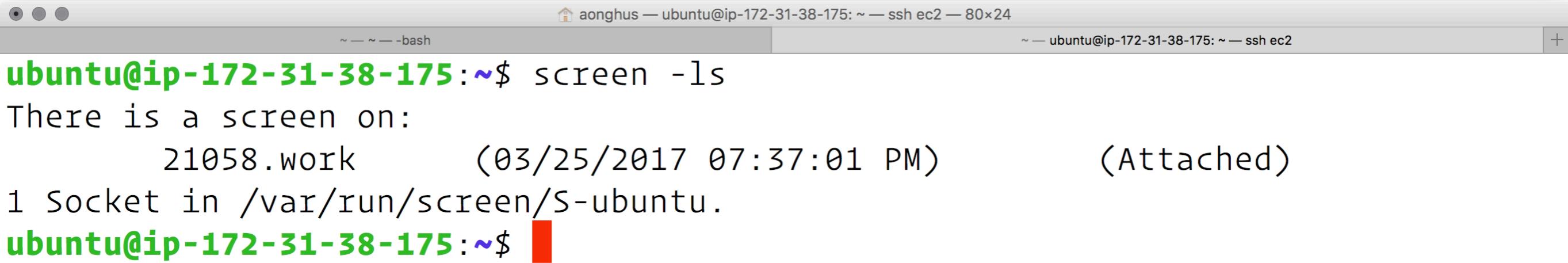
## Switching between screens

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”.



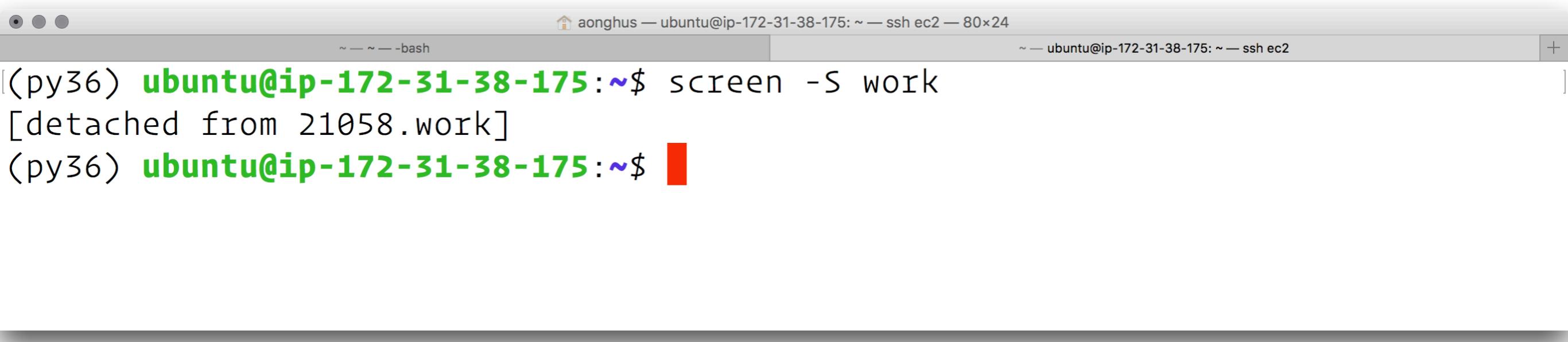
The image shows a screenshot of a macOS terminal window. The window has a light gray header bar with three circular icons on the left. The main area contains a terminal session. At the top of the session, there is a green status bar with the following information from left to right: a house icon, the user name "aonghus", the host name "ubuntu@ip-172-31-38-175", the directory "~", the command "ssh ec2", and the size "80x24". Below this, the terminal prompt "(py36) " is followed by the command "ubuntu@ip-172-31-38-175:~\$ screen -S work". A red rectangular box highlights the word "work" in the command line.

```
(py36) ubuntu@ip-172-31-38-175:~$ screen -S work
```



The screenshot shows a terminal window with two tabs. The active tab on the left is titled 'aonghus — ubuntu@ip-172-31-38-175: ~ — ssh ec2 — 80x24' and contains the command 'screen -ls'. The output of this command is displayed, showing one screen session named '21058.work' which was created on 03/25/2017 at 07:37:01 PM and is currently attached. A red box highlights the end of the command line in the first tab. The second tab on the right is titled 'ubuntu@ip-172-31-38-175: ~ — ssh ec2' and is currently inactive.

```
aonghus — ubuntu@ip-172-31-38-175: ~ — ssh ec2 — 80x24
~ — ~ — -bash
ubuntu@ip-172-31-38-175:~$ screen -ls
There is a screen on:
    21058.work          (03/25/2017 07:37:01 PM)          (Attached)
1 Socket in /var/run/screen/S-ubuntu.
ubuntu@ip-172-31-38-175:~$
```



aonghus — ubuntu@ip-172-31-38-175: ~ — ssh ec2 — 80x24

~ — ~ — -bash

~ — ubuntu@ip-172-31-38-175: ~ — ssh ec2

```
[py36] ubuntu@ip-172-31-38-175:~$ screen -S work
[detached from 21058.work]
(py36) ubuntu@ip-172-31-38-175:~$
```

top - 19:38:18 up 62 days, 3:20, 1 user, load average: 0.02, 0.11, 0.13											
Tasks: 125 total, 1 running, 123 sleeping, 1 stopped, 0 zombie											
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st											
KiB Mem : 1014452 total, 269616 free, 67312 used, 677524 buff/cache											
KiB Swap: 0 total, 0 free, 0 used. 896572 avail Mem											
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
21077	ubuntu	20	0	40500	3748	3168	R	0.5	0.4	0:00.03	top
1	root	20	0	37656	4848	3128	S	0.0	0.5	0:19.82	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:02.13	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0.0	0.0	0:04.23	rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
10	root	rt	0	0	0	0	S	0.0	0.0	0:24.73	watchdog/0
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
12	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
13	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	perf
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xenwatch
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xenbus
17	root	20	0	0	0	0	S	0.0	0.0	0:01.15	khungtaskd
18	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	writeback
19	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd

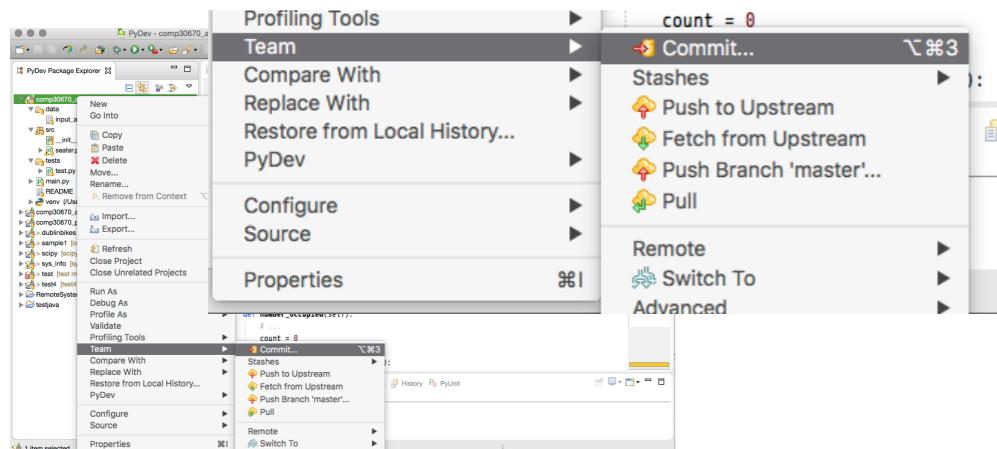
```
aonghus — ubuntu@ip-172-31-38-175: ~ — ssh ec2 — 80x24
~ — ~ — -bash
~ — ubuntu@ip-172-31-38-175: ~ — ssh ec2 +
```

(py36) **ubuntu@ip-172-31-38-175:**~\$ screen -S work  
[detached from 21058.work]  
(py36) **ubuntu@ip-172-31-38-175:**~\$ screen -ls  
There is a screen on:  
 21058.work (03/25/2017 07:37:01 PM) (Detached)  
1 Socket in /var/run/screen/S-ubuntu.  
(py36) **ubuntu@ip-172-31-38-175:**~\$ screen -r 21058.work  
[detached from 21058.work]  
(py36) **ubuntu@ip-172-31-38-175:**~\$ screen -r 21058.work  
[detached from 21058.work]  
(py36) **ubuntu@ip-172-31-38-175:**~\$ █

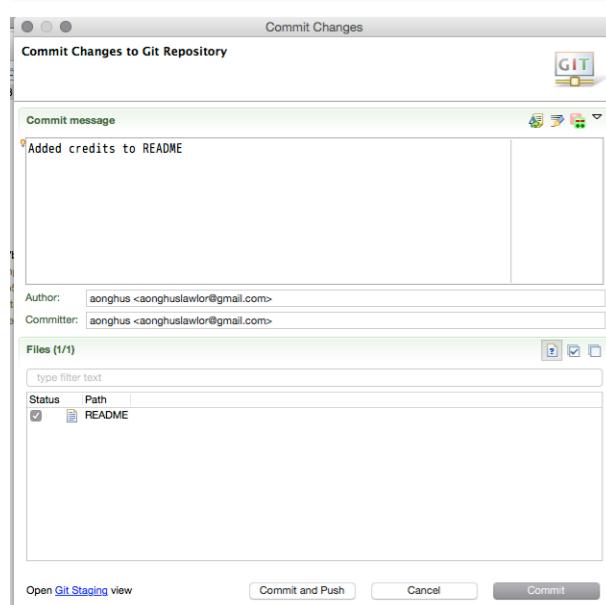
top - 19:38:38 up 62 days, 3:20, 1 user, load average: 0.02, 0.11, 0.12											
Tasks: 125 total, 1 running, 123 sleeping, 1 stopped, 0 zombie											
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st											
KiB Mem : 1014452 total, 269616 free, 67312 used, 677524 buff/cache											
KiB Swap: 0 total, 0 free, 0 used. 896572 avail Mem											
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1198	root	10	-10	5724	3516	2424	S	0.5	0.3	10:45.12	iscsid
21077	ubuntu	20	0	40500	3748	3168	R	0.5	0.4	0:00.05	top
1	root	20	0	37656	4848	3128	S	0.0	0.5	0:19.82	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:02.13	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0.0	0.0	0:04.23	rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
10	root	rt	0	0	0	0	S	0.0	0.0	0:24.73	watchdog/0
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
12	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
13	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	perf
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xenwatch
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xenbus
17	root	20	0	0	0	0	S	0.0	0.0	0:01.15	khungtaskd
18	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	writeback

# Git

on your local machine:

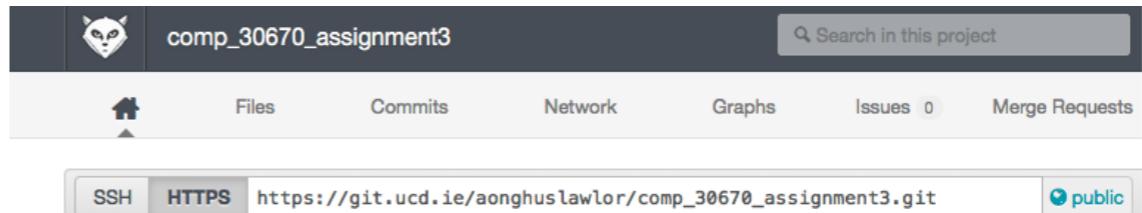


in Eclipse, do “commit and push”



on the server:

github.com



git clone [https://github.com/aonghuslawlor/comp\\_30670\\_assignment3.git](https://github.com/aonghuslawlor/comp_30670_assignment3.git)

```
aonghus — ssh alawlor@buzzer.ucd.ie — 98x17
(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$
```

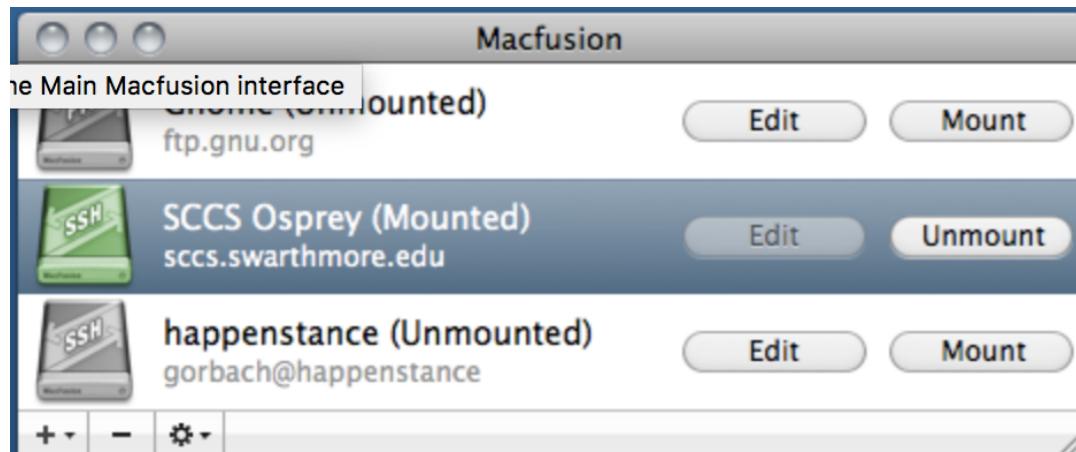
THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZIZE 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.



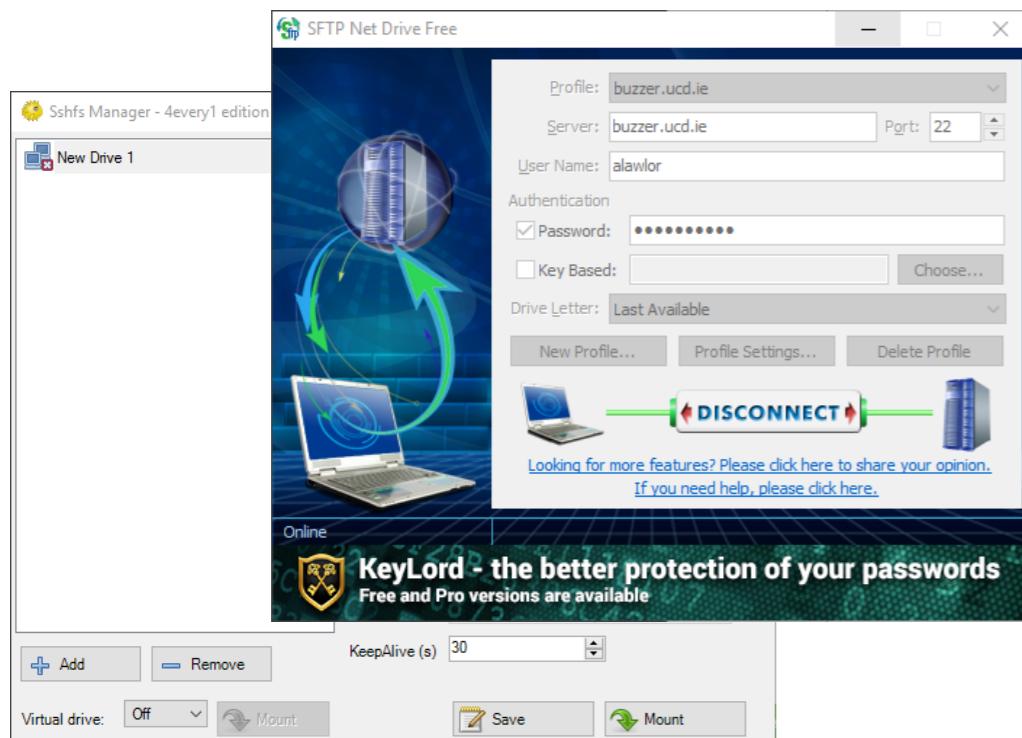
# SCP/SSHFS



You can use scp (file copy over ssh) to transfer files to the server

Alternatively, you can use ssh to create a virtual mounted drive

win-sshfs, sftp net-drive  
macfusion



	Name	Date modified	Type
Quick access	.git	3/9/2016 3:06 PM	File folder
	data	3/9/2016 3:06 PM	File folder
	src	3/9/2016 4:48 PM	File folder
	tests	3/9/2016 3:06 PM	File folder
	.gitignore	3/9/2016 3:06 PM	GITIGNORE File
	.project	3/9/2016 3:06 PM	PROJECT File
	.pydevproject	3/9/2016 3:06 PM	PYDEVPROJECT File
	main.py	3/9/2016 3:06 PM	PY File
	README	3/9/2016 3:06 PM	File

# Workflow

clone the repository

```
(py36) ubuntu@ip-172-31-38-175:~$ git clone https://github.com/aonghus/dublinbikesproject.git  
Cloning into 'dublinbikesproject'...  
remote: Counting objects: 3, done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (3/3), done.  
Checking connectivity... done.  
(py36) ubuntu@ip-172-31-38-175:~$
```

check which virtualenv you are using  
(activate if necessary)

```
ubuntu@ip-172-31-38-175:~$ source activate py36  
(py36) ubuntu@ip-172-31-38-175:~$
```

change into the source directory

```
(py36) ubuntu@ip-172-31-38-175:~/dublinbikesproject$ ls  
data main.py README.md setup.py src tests
```

run your program...

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

```
(py36) ubuntu@ip-172-31-38-175:~/dublinbikesproject$ nohup python ./main.py &
```

# Workflow

- There are numerous posts and tutorials about the perfect workflow (none are!)
- Do some experimentation and find what works for you

The Perfect Workflow, with Git, GitHub, and SSH

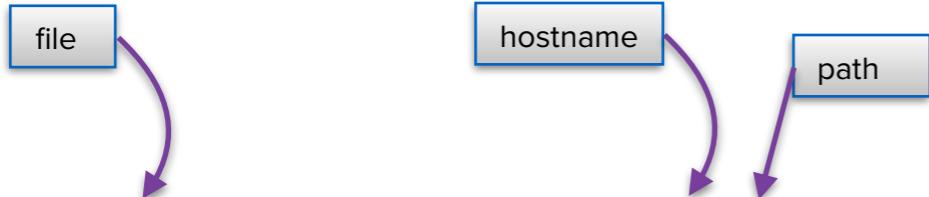
# **Cloning Python Virtual Environment**

# Using pip

On your local machine, use pip to get a list of the installed packages in your virtual environment:

local machine

```
$ conda env export > environment.yml
```



```
$ scp environment.yml ec2:
```

environment.yml  
"package name"==version

```
beautifulsoup4==4.4.1
blosc==1.2.8
bokeh==0.11.1
boto==2.39.0
brewer2mpl==1.4.1
bz2file==0.98
cffi==1.5.2
chardet==2.3.0
...
```

server

```
$ conda env update -f environment.yml
```

- You can now easily wipe the virtual environment on the server, re-create it and then install all the packages from scratch
- This is how you ensure you have the identical setup on all working machines

aonghus ~ — bash — 80x24

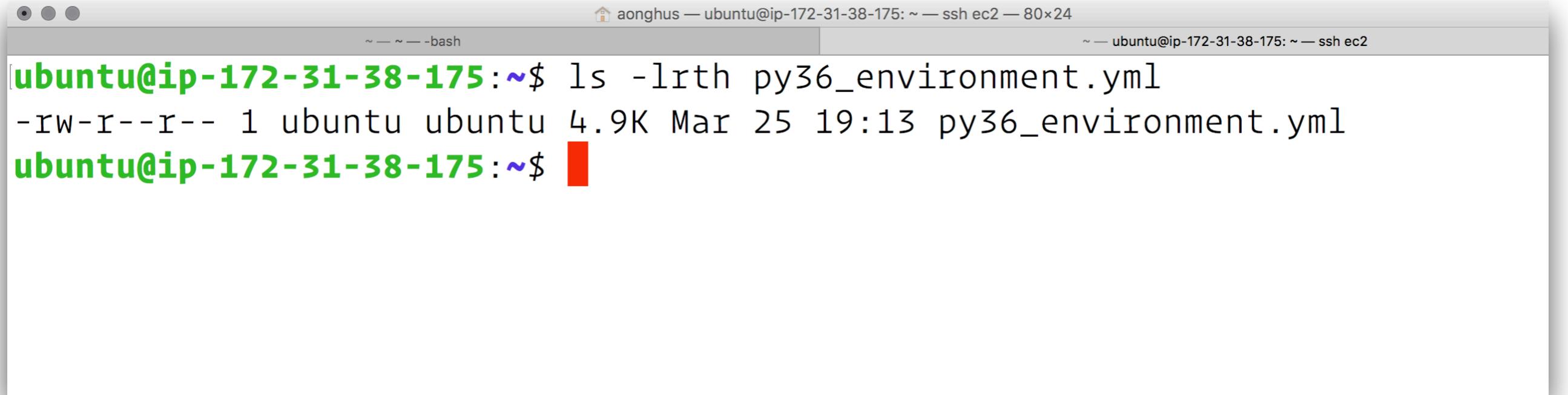
```
aonghus@seldon ~  
$ source activate py36  
(py36)  
aonghus@seldon ~  
$
```

```
● ● ● aonghus — ~ — -bash — 80x24
(py36)
aonghus@seldon ~
[$ conda env export > py36_environment.yml
DEPRECATION: The default format will switch to columns in the future. You can use --format=(legacy|columns) (or define a format=(legacy|columns) in your pip.conf under the [list] section) to disable this warning.
(py36)
aonghus@seldon ~
$ █
```

```
name: py36
channels: !!python/tuple
- conda-forge
- defaults
dependencies:
- boto=2.46.1=py36_0
- bz2file=0.98=py36_0
- conda-forge::appnope=0.1.0=py36_0
- conda-forge::blas=1.1=openblas
- conda-forge::bleach=1.5.0=py36_0
- conda-forge::ca-certificates=2017.1.23=0
- conda-forge::cairo=1.14.6=4
- conda-forge::cairocffi=0.7.2=py36_1
- conda-forge::certifi=2017.1.23=py36_0
- conda-forge::cffi=1.7.0=py36_0
- conda-forge::click=6.7=py36_0
- conda-forge::cycler=0.10.0=py36_0
- conda-forge::decorator=4.0.11=py36_0
- conda-forge::elasticsearch=5.2.0=py36_0
- conda-forge::entrypoints=0.2.2=py36_1
- conda-forge::flask=0.12=py36_0
- conda-forge::fontconfig=2.12.1=4
- conda-forge::freetype=2.7=1
```

py36\_environment.yml

```
● ● ● aonghus — ~ — -bash — 80x24
(py36)
aonghus@seldon ~
$ scp py36_environment.yml ec2:
py36_environment.yml          100% 4951      2.0MB/s   4.8KB/s   00:00
(py36)
aonghus@seldon ~
$ █
```



The image shows a terminal window with two tabs. The active tab displays the command `ls -lrth py36_environment.yml` and its output, which shows a single file named `py36_environment.yml` with permissions `-rw-r--r--`, owned by `ubuntu`, modified on `Mar 25 19:13`. The file size is `4.9K`. The terminal window has a light gray background and a dark gray header bar. The title bar includes the user name `aonghus`, session information `ubuntu@ip-172-31-38-175: ~ — ssh ec2`, and the terminal size `80x24`. The prompt is `[ubuntu@ip-172-31-38-175:~$`.

```
aonghus — ubuntu@ip-172-31-38-175: ~ — ssh ec2 — 80x24
~ — ~ — bash
~ — ubuntu@ip-172-31-38-175: ~ — ssh ec2
[ubuntu@ip-172-31-38-175:~$ ls -lrth py36_environment.yml
-rw-r--r-- 1 ubuntu ubuntu 4.9K Mar 25 19:13 py36_environment.yml
ubuntu@ip-172-31-38-175:~$
```

aonghus — ubuntu@ip-172-31-38-175: ~ — ssh ec2 — 80x24

~ — ~ — -bash

~ — ubuntu@ip-172-31-38-175: ~ — ssh ec2

[ubuntu@ip-172-31-38-175:~\$ conda create -n py36 python=3.6

Fetching package metadata .....

Solving package specifications: .

Package plan for installation in environment /home/ubuntu/anaconda3/envs/py36:

The following NEW packages will be INSTALLED:

openssl:	1.0.2k-1
pip:	9.0.1-py36_1
python:	3.6.1-0
readline:	6.2-2
setuptools:	27.2.0-py36_0
sqlite:	3.13.0-0
tk:	8.5.18-0
wheel:	0.29.0-py36_0
xz:	5.2.2-1
zlib:	1.2.8-3

Proceed ([y]/n)?

The image shows a terminal window with two tabs. The active tab on the left has a grey header bar with three small circular icons on the left. The main area of the tab displays the command: `(py36) ubuntu@ip-172-31-38-175:~$ conda env update -f=py36_environment.yml`. The text is in black font on a white background. To the right of the main area is a vertical scroll bar. The header bar of the active tab also contains the text "aonghus — ubuntu@ip-172-31-38-175: ~ — ssh ec2 — 80x24". The tab bar above the active tab shows another tab with the text "ubuntu@ip-172-31-38-175: ~ — ssh ec2". On the far right of the window frame, there is a small red square icon.

# Databases/MySQL

# Data Storage Zoo



Relational  
MySQL, Postgres, SQLite, ...



Document Stores  
SimpleDB, CouchDB, MongoDB, Terrastore



Key-Value Stores  
Voldemort, Riak, Redis, Scalaris, Tokyo Cabinet, Memcached/Membrain/Membase



Extensible Record Stores:  
HBase, HyperTable, Cassandra, PNUT, BigTable



# MySQL Workbench

## Connect to Database

Stored Connection:  Select from saved connection settings

Connection Method:  Method to use to connect to the RDBMS

**Parameters** **SSL** **Advanced**

Hostname:  Port:  Name or IP address of the IP port.

Username:  Name of the user to connect.

Password:  Store in Keychain ... Clear The user's password. Will not be set.

Default Schema:  The schema to use as default to select it later.

Cancel

OK

# Column Types

• INT	integer
• FLOAT	Small floating-point number
• DOUBLE	Double-precision floating-point number
• CHAR(N)	Text N characters long (N=1..255)
• VARCHAR(N)	Variable length text up to N characters long
• TEXT	Text up to 65535 characters long
• LONGTEXT	Text up to 4294967295 characters long

A table consists of several columns each of which has a specific data type (e.g. integer, text).

It is useful to define columns correctly because it has implications for sorting the data (numeric versus text), and for the size of the data allowed in a column

# Create Table

```
DROP TABLE IF EXISTS gene_map;  
CREATE TABLE gene_map (  
    gene VARCHAR(255) NOT NULL,  
    chromosome INT NOT NULL,  
    cM_position FLOAT NOT NULL,  
    id INT NOT NULL AUTO_INCREMENT,  
);
```

# Create Table

```
DROP TABLE IF EXISTS gene_map;  
CREATE TABLE gene_map (  
    gene VARCHAR(255) NOT NULL,  
    chromosome INT NOT NULL,  
    cM_position FLOAT NOT NULL,  
    id INT NOT NULL AUTO_INCREMENT,  
    PRIMARY KEY (id),  
    INDEX i_g (gene),  
    INDEX i_c (chromosome),  
    INDEX i_cp (cM_position)  
);
```

# Insert

```
CREATE TABLE gene_map (  
gene VARCHAR(255) NOT NULL,  
chromosome INT NOT NULL,  
cM_position FLOAT NOT NULL,  
id INT NOT NULL AUTO_INCREMENT,  
);
```

Note that if you specify a value for every column in order it is not necessary to declare the column names in the statement (e.g. (a)). If you only want to add data to some columns, or to add them in a different order, then you must declare the column names after the table name in the statement (e.g., (b)). In this case columns with no value specified will be “NULL” or will get the default value (“0” for numeric columns, “” for text columns, and the next integer for auto increment columns) if “NOT NULL” was specified during table definition.

- (a) **INSERT INTO gene\_map VALUES('agrc259', '1', '233.3', 'NULL');**
- (b) **INSERT INTO gene\_map (gene, chromosome) VALUES('adp0', '10');**

# Cheat Sheet

## A Quick Guide To MySQL Tables & Queries

This is a Quick reference Guide for MySQL 5.x. MySQL is a relational database management system (RDBMS) based on SQL (Structured Query Language). MySQL is available under GNU public license and through subscription to MySQL Network for business applications. It runs on Unix, iMac, and Windows and provides rich API for many programming languages including C, C++, Java, Perl, Python, Ruby and PHP.

### Database Queries:

List all databases

```
>SHOW databases;
```

Select the database

```
>USE <database name>
```

```
>USE university;
```

Create a database

```
>CREATE DATABASE <database name>
>CREATE DATABASE university;
```

Delete a database

```
>DROP DATABASE <database name>
>DROP DATABASE university;
```

Rename a database

```
>ALTER DATABASE <database name> RENAME <new
database name>
>ALTER DATABASE university RENAME faculty
```

### Table Queries:

Create a table

```
>CREATE TABLE <table name> (<field name> <field type>
<field size>, ... )
>CREATE TABLE teachers (name varchar(20),
age INT(10));
```

List all tables in the database

```
>SHOW tables;
```

Show table format with column names and data types

```
>DESCRIBE <table name>
>DESCRIBE teachers;
```

Modify the structure of table

```
>ALTER TABLE <table name> <alter
specifications>
>ALTER TABLE teachers DROP COLUMN salary;
>ALTER TABLE teachers ADD COLUMN salary
INT(5);
>ALTER TABLE teachers CHANGE firstName
name VARCHAR(20);
```

Delete the table

```
>DROP TABLE <table name>
>DROP TABLE teachers;
```

### Retrieving Data:

```
>SELECT <columns> FROM <tables> WHERE <conditions>
```

Retrieve from all columns

```
>SELECT * FROM <tables>
>SELECT * FROM teachers;
```

Retrieve from selected columns

```
>SELECT <column 1>, <column 2> FROM <tables>
>SELECT id, name FROM teachers;
```

Retrieve from selected tables

```
>SELECT <columns> FROM <table 1>, <table 2>
>SELECT teachers.name, students.name FROM
teachers, students;
```

Retrieve unique values

```
>SELECT DISTINCT <column name> FROM <table>
>SELECT DISTINCT name FROM teachers;
```

Retrieve data satisfying a given condition

```
>SELECT <columns> FROM <tables> WHERE <condition>
>SELECT name FROM teachers WHERE age > 35;
```

Retrieve data satisfying multiple conditions

```
>SELECT <columns> FROM <tables> WHERE <condition>
AND <condition>
>SELECT name FROM teachers WHERE age > 35
AND gender = 'female';
```

### Inserting Data:

```
>INSERT INTO <table> (<columns>) VALUES (<data>)
> INSERT INTO teachers (id, name, age) VALUES
(NULL, 'John', '12');
```

### Loading Data from Files:

```
>LOAD DATA LOCAL INFILE '<filename>' INTO TABLE
<table>
>LOAD DATA LOCAL INFILE 'file.sql' INTO
TABLE teachers;
```

This is very convenient to load data directly from files when you have thousands of entries.

### Modifying Data:

```
>UPDATE <table> SET <field1> = <value1> AND <field2>
= <value2> WHERE <conditions>
>UPDATE teachers SET status = 'enrolled'
WHERE fees = 'paid';
```

### Deleting Data:

```
>DELETE FROM <table> WHERE <condition>
>DELETE FROM teachers WHERE fees = 'paid';
```

### Pattern Matching Examples:

```
>SELECT * FROM teachers WHERE name LIKE
'j%';
Wildcard % selects joe, john, jones, etc.
```

```
> SELECT * FROM teachers WHERE name LIKE
'_';
Selects 3 character values.
```

```
> SELECT * FROM teachers WHERE name
REGEXP '^A';
Selects all entries beginning with A.
```

```
> SELECT * FROM teachers WHERE name
REGEXP 'p$';
Selects all entries ending with p.
```

[abc]	match a, b, or c
[^abc]	match all expect a, b, or c
[A-Z]	match uppercase
[a-z]	match lowercase
[0-9]	match any digit

# Create DB

```
In [120]: import sqlalchemy as sqa
          from sqlalchemy import create_engine
          import traceback
          import glob
          import os
          from pprint import pprint
          import simplejson as json
          import requests
          import time
          from IPython.display import display
```

```
In [17]: URI="dbikes.cilfi4edpsps.eu-west-1.rds.amazonaws.com"  
PORT="3306"  
DB = "dbikes"  
USER = "aonghus"
```

```
In [18]: engine = create_engine("mysql+mysqldb://{}:{}@{}:{}/{}".format(USER, PASSWORD,  
URI, PORT, DB), echo=True)
```



```
In [16]: sql = """
CREATE DATABASE IF NOT EXISTS dbikes;
"""

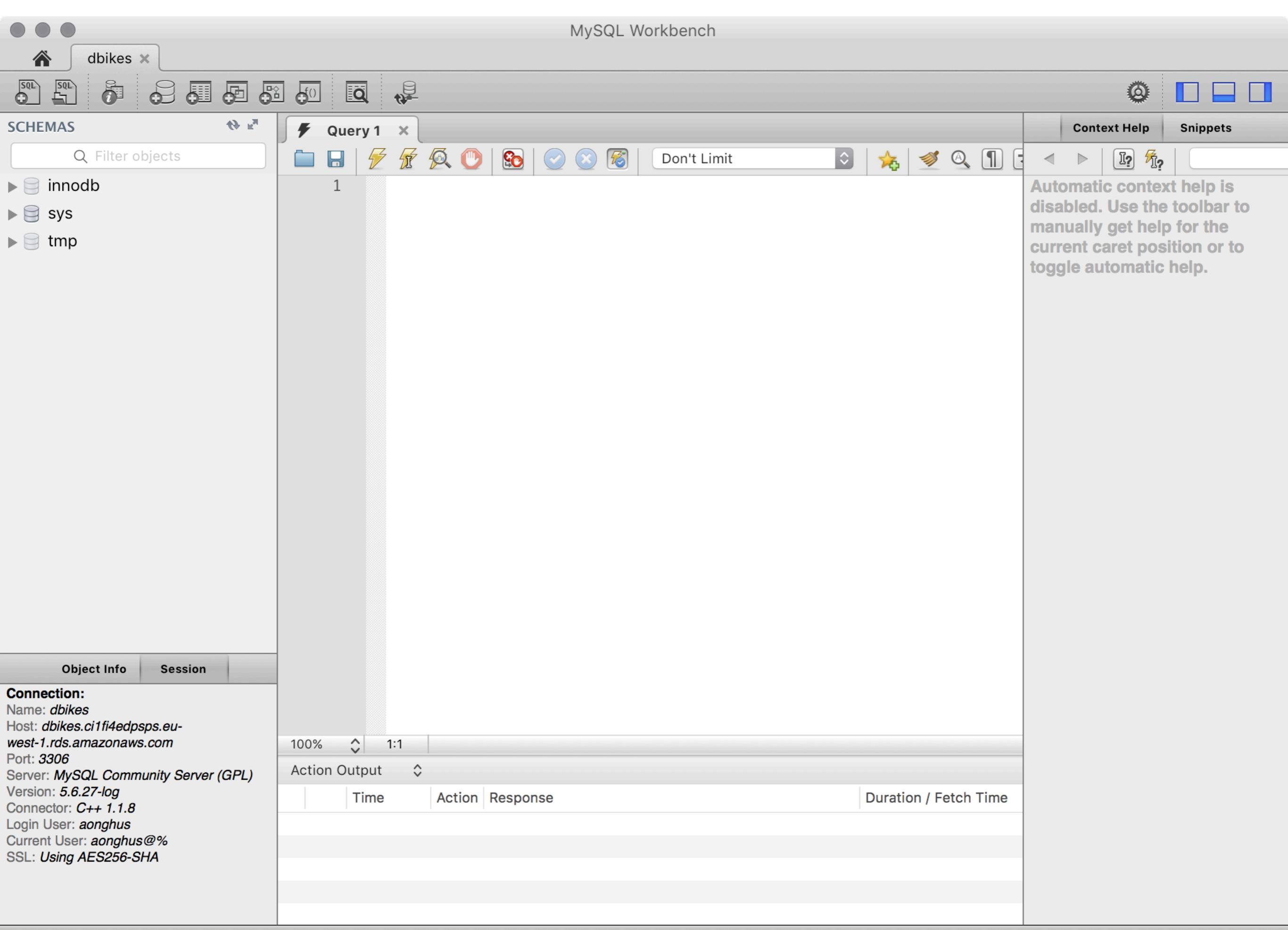
engine.execute(sql)
```

```
2017-03-26 17:05:24,152 INFO sqlalchemy.engine.base.Engine
CREATE DATABASE IF NOT EXISTS dbikes;

2017-03-26 17:05:24,154 INFO sqlalchemy.engine.base.Engine ()
2017-03-26 17:05:24,159 INFO sqlalchemy.engine.base.Engine COMMIT
```

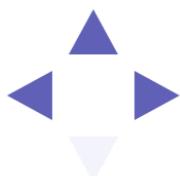
```
Out[16]: <sqlalchemy.engine.result.ResultProxy at 0x10787bda0>
```





```
In [11]: for res in engine.execute("SHOW VARIABLES;"):
    print(res)
```

```
('auto_increment_increment', '1')
('auto_increment_offset', '1')
('autocommit', 'OFF')
('automatic_sp_privileges', 'ON')
('avoid_temporal_upgrade', 'OFF')
('back_log', '63')
('basedir', '/rdsdbbin/mysql/')
('big_tables', 'OFF')
('bind_address', '*')
('binlog_cache_size', '32768')
('binlog_checksum', 'CRC32')
('binlog_direct_non_transactional_updates', 'OFF')
('binlog_error_action', 'IGNORE_ERROR')
('binlog_format', 'MIXED')
('binlog_gtid_simple_recovery', 'OFF')
('binlog_max_flush_queue_time', '0')
('binlog_order_commits', 'ON')
('binlog_row_image', 'FULL')
('binlog_rows_query_log_events', 'OFF')
('binlog_stmt_cache_size', '32768')
('binlogging_impossible_mode', 'IGNORE_ERROR')
('block_encryption_mode', 'aes-128-ecb')
('bulk_insert_buffer_size', '8388608')
('character_set_client', 'utf8')
('character_set_connection', 'utf8')
('character_set_database', 'latin1')
('character_set_filesystem', 'binary')
('character_set_results', 'utf8')
('character_set_server', 'latin1')
('character_set_system', 'utf8')
('character_sets_dir', '/rdsdbbin/mysql-5.6.27.R1/share/charsets/')
'collation_connection' 'utf8_general_ci'
```

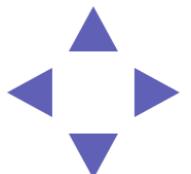


# Create Tables

```
In [100]: sql = """
CREATE TABLE IF NOT EXISTS station (
address VARCHAR(256),
banking INTEGER,
bike_stands INTEGER,
bonus INTEGER,
contract_name VARCHAR(256),
name VARCHAR(256),
number INTEGER,
position_lat REAL,
position_lng REAL,
status VARCHAR(256)
)
"""

try:
    res = engine.execute("DROP TABLE IF EXISTS station")
    res = engine.execute(sql)
    print(res.fetchall())
except Exception as e:
    print(e)
```

```
2017-03-26 18:29:43,944 INFO sqlalchemy.engine.base.Engine DROP TABLE IF EXISTS station
2017-03-26 18:29:43,947 INFO sqlalchemy.engine.base.Engine ()
2017-03-26 18:29:43,955 INFO sqlalchemy.engine.base.Engine COMMIT
2017-03-26 18:29:43,990 INFO sqlalchemy.engine.base.Engine
CREATE TABLE IF NOT EXISTS station (
address VARCHAR(256),
banking INTEGER,
bike_stands INTEGER,
bonus INTEGER,
contract_name VARCHAR(256),
name VARCHAR(256),
number INTEGER,
position_lat REAL
```



```
In [29]: sql = """
CREATE TABLE IF NOT EXISTS availability (
    number INTEGER,
    available_bikes INTEGER,
    available_bike_stands INTEGER,
    last_update INTEGER
)
"""

try:
    res = engine.execute(sql)
    print(res.fetchall())
except Exception as e:
    print(e#, traceback.format_exc())
```

```
2017-03-26 17:16:06,540 INFO sqlalchemy.engine.base.Engine
CREATE TABLE IF NOT EXISTS availability (
    number INTEGER,
    available_bikes INTEGER,
    available_bike_stands INTEGER,
    last_update INTEGER
)

2017-03-26 17:16:06,542 INFO sqlalchemy.engine.base.Engine ()
2017-03-26 17:16:06,562 INFO sqlalchemy.engine.base.Engine COMMIT
This result object does not return rows. It has been closed automatically.
```



dbikes

SQL SQL i + f() f( )

SCHEMAS Filter objects

dbikes

Tables

stations

Columns

- address
- banking
- bike\_stands
- bonus
- contract\_name
- name
- number
- position\_lat
- position\_lng
- status

Indexes

Foreign Keys

Triggers

Object Info Session

**Connection:**  
Name: dbikes  
Host: dbikes.ci1fi4edpsps.eu-west-1.rds.amazonaws.com  
Port: 3306  
Server: MySQL Community Server (GPL)  
Version: 5.6.27-log  
Connector: C++ 1.1.8  
Login User: aonghus  
Current User: aonghus@%  
SSL: Using AES256-SHA

Query 1 new\_schema - Schema

Don't Limit

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

1

100% 1:1

Action Output

	Time	Action	Response	Duration / Fetch Time
✖ 1	16:55:28	Appl...		

Connection reopened.

# Web/API Crawling



```
#!/usr/bin/env python
import requests
import traceback
import datetime
import time
APIKEY="908234203984"
NAME="Dublin"
STATIONS="https://api.jcdecaux.com/vls/v1/stations"

def write_to_file(text):
    with open("data/bikes_{}".format(now).replace(" ", "_"), "w") as f:
        f.write(r.text)

def write_to_db(text):

def main():
    while True:
        try:
            now = datetime.datetime.now()
            r = requests.get(STATIONS, params={"apiKey": APIKEY, "contract": NAME})
            print(r, now)
            write_to_file(r.text)
            write_to_db(r.text)
            time.sleep(5*60)
        except:
            print(traceback.format_exc())
            if engine is None:

return
```



# Insert row into table (SQL)



```
In [101]: def stations_to_db(text):
    stations = json.loads(text)
    print(type(stations), len(stations))
    for station in stations:
        print(station)
        vals = (station.get('address'), int(station.get('banking')),
    station.get('bike_stands'), int(station.get('bonus')),
                           station.get('contract_name'), station.get('name'),
    station.get('number'),
                           station.get('position').get('lat'),
    station.get('position').get('lng'), station.get('status')
)
        engine.execute("insert into station
values(%s,%s,%s,%s,%s,%s,%s,%s,%s)",
                           vals)
    break
return
stations_to_db(r.text)
```

```
<class 'list'> 101
{'number': 42, 'name': 'SMITHFIELD NORTH', 'address': 'Smithfield North', 'position': {'lat': 53.349562, 'lng': -6.278198}, 'banking': True, 'bonus': False, 'status': 'OPEN', 'contract_name': 'Dublin', 'bike_stands': 30, 'available_bike_stands': 22, 'available_bikes': 8, 'last_update': 1490545430000}
2017-03-26 18:30:25,565 INFO sqlalchemy.engine.base.Engine insert into station
values(%s,%s,%s,%s,%s,%s,%s,%s,%s)
2017-03-26 18:30:25,709 INFO sqlalchemy.engine.base.Engine ('Smithfield Nort
h', 1, 30, 0, 'Dublin', 'SMITHFIELD NORTH', 42, 53.349562, -6.278198, 'OPEN')
2017-03-26 18:30:25,712 INFO sqlalchemy.engine.base.Engine COMMIT
```



## MySQL Workbench

dbikes

SQL

SQL

Tables

availability

station

Columns

- address
- banking
- bike\_stands
- bonus
- contract\_name
- name
- number
- position\_lat
- position\_lng
- status

Indexes

Foreign Keys

Object Info

Session

Schema: dbikes

Query 1

dbikes.station

Don't Limit

1 • select \* from station;

Result Grid

Filter Rows:

Search

Export:

address	banking	bike_stands	bonus	contract_name	name	number	position_lat	position_lng	status
Smithfield North	1	30	0	Dublin	SMITHFIELD NORTH	42	53.349562	-6.278198	OPEN

station 3

Read Only

Context Help

Snippets

My Snippets

Result Grid

Form Editor

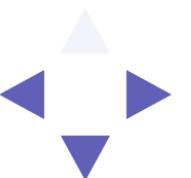
Field Types

Query Stats

Execution Plan

Query Completed

# Create Table (sqlalchemy)



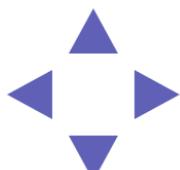
```
In [114]: metadata = sqla.MetaData()
station = sqla.Table("station", metadata,
    sqla.Column('address', sqla.String(256), nullable=False),
    sqla.Column('banking', sqla.Integer),
    sqla.Column('bike_stands', sqla.Integer),
    sqla.Column('bonus', sqla.Integer),
    sqla.Column('contract_name', sqla.String(256)),
    sqla.Column('name', sqla.String(256)),
    sqla.Column('number', sqla.Integer),
    sqla.Column('position_lat', sqla.REAL),
    sqla.Column('position_lng', sqla.REAL),
    sqla.Column('status', sqla.String(256))
)

availability = sqla.Table("availability", metadata,
    sqla.Column('available_bikes', sqla.Integer),
    sqla.Column('available_bike_stands', sqla.Integer),
    sqla.Column('number', sqla.Integer),
    sqla.Column('last_update', sqla.BigInteger)
)

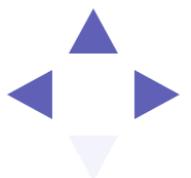
try:
    station.drop(engine)
    availability.drop(engine)
except:
    pass

metadata.create_all(engine)
```

```
2017-03-26 18:48:08,219 INFO sqlalchemy.engine.base.Engine
DROP TABLE station
2017-03-26 18:48:08,221 INFO sqlalchemy.engine.base.Engine ()
2017-03-26 18:48:08,237 INFO sqlalchemy.engine.base.Engine COMMIT
2017-03-26 18:48:08,245 INFO sqlalchemy.engine.base.Engine
DROP TABLE availability
```



```
In [ ]: # use this code to load the Tables if you have already created them  
# metadata = sqla.MetaData(bind=engine)  
# print(metadata)  
# station = sqla.Table('station', metadata, autoload=True)  
# print(station)
```



# Insert row (sqlalchemy)



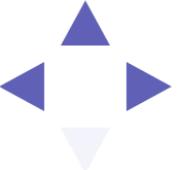
```
In [122]: # need to make sure the right keys are in the rows being inserted into the db
def stations_fix_keys(station):
    station['position_lat'] = station['position']['lat']
    station['position_lng'] = station['position']['lng']
    return station

stations = json.loads(open('stations.json', 'r').read())

engine.execute(station.insert(), *map(stations_fix_keys, stations))
```

```
2017-03-26 19:24:37,977 INFO sqlalchemy.engine.base.Engine INSERT INTO station
(address, banking, bike_stands, bonus, contract_name, name, number, position_l
at, position_lng, status) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
2017-03-26 19:24:37,991 INFO sqlalchemy.engine.base.Engine (('Smithfield Nort
h', True, 30, False, 'Dublin', 'SMITHFIELD NORTH', 42, 53.349562, -6.278198,
'OPEN'), ('Parnell Square North', True, 20, False, 'Dublin', 'PARNELL SQUARE
NORTH', 30, 53.353462, -6.265305, 'OPEN'), ('Pearse Street', True, 30, False,
'Dublin', 'PEARSE STREET', 32, 53.344304, -6.250427, 'OPEN'), ('Excise Walk',
False, 40, False, 'Dublin', 'EXCISE WALK', 48, 53.347777, -6.244239, 'OPEN'),
('Fitzwilliam Square West', False, 30, False, 'Dublin', 'FITZWILLIAM SQUARE WE
ST', 13, 53.336074, -6.252825, 'OPEN'), ('St. James Hospital (Central)', Fals
e, 40, False, 'Dublin', 'ST. JAMES HOSPITAL (CENTRAL)', 81, 53.339983, -6.2955
94, 'OPEN'), ('Hanover Quay', True, 40, False, 'Dublin', 'HANOVER QUAY', 68, 5
3.344115, -6.237153, 'OPEN'), ('Oliver Bond Street', False, 30, False, 'Dubli
n', 'OLIVER BOND STREET', 74, 53.343893, -6.280531, 'OPEN') ... displaying 10
of 101 total bound parameter sets ... ('Heuston Bridge (North)', False, 40, F
alse, 'Dublin', 'HEUSTON BRIDGE (NORTH)', 92, 53.347802, -6.292432, 'OPEN'),
('Blackhall Place', False, 30, False, 'Dublin', 'BLACKHALL PLACE', 88, 53.348
8, -6.281637, 'OPEN'))
2017-03-26 19:24:38,016 INFO sqlalchemy.engine.base.Engine COMMIT
```

```
Out[122]: <sqlalchemy.engine.result.ResultProxy at 0x1074f3e48>
```



## MySQL Workbench

dbikes

SQL

SCHEMAS

Filter objects

dbikes

- Tables
  - availability
  - station
- Views
- Stored Procedures
- Functions

innodb

sys

tmp

Query 1 dbikes.station x

Don't Limit

select \* from station;

Result Grid | Filter Rows: Search Export:

address	banking	bike_stands	bonus	contract_name	name	number	position_lat
Smithfield North	1	30	0	Dublin	SMITHFIELD NORTH	42	NULL
Parnell Square North	1	20	0	Dublin	PARNELL SQUARE NORTH	30	NULL
Pearse Street	1	30	0	Dublin	PEARSE STREET	32	NULL
Excise Walk	0	40	0	Dublin	EXCISE WALK	48	NULL
Fitzwilliam Square West	0	30	0	Dublin	FITZWILLIAM SQUARE WEST	13	NULL
St. James Hospital (Central)	0	40	0	Dublin	ST. JAMES HOSPITAL (CENTRAL)	81	NULL
Hanover Quay	1	40	0	Dublin	HANOVER QUAY	68	NULL
Oliver Bond Street	0	30	0	Dublin	OLIVER BOND STREET	74	NULL
Collins Barracks Museum	1	38	0	Dublin	COLLINS BARRACKS MUSEUM	87	NULL
Brookfield Road	0	30	0	Dublin	BROOKFIELD ROAD	84	NULL
Clonmel Street	0	33	0	Dublin	CLONMEL STREET	54	NULL
James Street East	0	30	0	Dublin	JAMES STREET EAST	20	NULL
Mount Street Lower	0	40	0	Dublin	MOUNT STREET LOWER	56	NULL
Christchurch Place	0	20	0	Dublin	CHRISTCHURCH PLACE	6	NULL
Grantham Street	1	30	0	Dublin	GRANTHAM STREET	18	NULL
York Street East	0	32	0	Dublin	YORK STREET EAST	52	NULL
Portobello Road	1	30	0	Dublin	PORTOBELLO ROAD	43	NULL
Parnell Street	0	20	0	Dublin	PARNELL STREET	31	NULL
Frederick Street South	1	30	0	Dublin	FREDERICK STREET SOUTH	98	NULL
Fownes Street Upper	0	30	0	Dublin	FOWNES STREET UPPER	14	NULL
Chatham Street	0	29	0	Dublin	CHATHAM STREET	1	NULL

station 4

Read Only

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Query Completed

MySQL Workbench

dbikes

SQL File 2\*

1 • select \* from availability;

Result Grid

	available_bikes	available_bike_stands	number	last_update
▶	22	8	42	1464735626000
▶	0	20	30	1464735674000
▶	0	30	32	1464735548000
▶	1	39	48	1464735651000
▶	8	22	13	1464735765000
▶	40	0	81	1464735581000
▶	7	33	68	1464735661000
▶	25	5	74	1464735628000
▶	31	7	87	1464735249000
▶	30	0	84	1464735194000
▶	4	29	54	1464735396000
▶	0	30	20	1464735613000
▶	7	33	56	1464735285000
▶	1	19	6	1464735632000
▶	16	14	18	1464735177000
▶	0	32	52	1464735390000
▶	29	1	43	1464735472000
▶	0	20	31	1464735458000
▶	0	29	98	1464735604000
▶	0	30	14	1464735544000

availability 2

Read Only

Object Info Session

Table: availability

Columns:

available_bikes	int(11)
available_bike_stands	int(11)
number	int(11)
last_update	bigint(20)

SCHEMAS

Filter objects

dbikes

Tables

availability

Columns

Indexes

Foreign Keys

Triggers

station

Views

Stored Procedures

Functions

innodb

sys

tmp

Context Help Snippets

My Snippets

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

Query Completed

# Reading DB with Pandas



```
In [124]: import pandas as pd
```

```
df = pd.read_sql_table("station", engine)
```

```
2017-03-26 19:24:47,929 INFO sqlalchemy.engine.base.Engine SHOW FULL TABLES FROM `dbikes`
2017-03-26 19:24:47,932 INFO sqlalchemy.engine.base.Engine ()
2017-03-26 19:24:47,936 INFO sqlalchemy.engine.base.Engine SHOW FULL TABLES FROM `dbikes`
2017-03-26 19:24:47,938 INFO sqlalchemy.engine.base.Engine ()
2017-03-26 19:24:47,943 INFO sqlalchemy.engine.base.Engine SHOW CREATE TABLE `station`
2017-03-26 19:24:47,945 INFO sqlalchemy.engine.base.Engine ()
2017-03-26 19:24:47,974 INFO sqlalchemy.engine.base.Engine SELECT station.address, station.banking, station.bike_stands, station.bonus, station.contract_name, station.name, station.number, station.position_lat, station.position_lng, station.status
FROM station
2017-03-26 19:24:47,976 INFO sqlalchemy.engine.base.Engine ()
```



```
In [126]: display(df.head())
```

	address	banking	bike_stands	bonus	contract_name	name	nbikes
0	Smithfield North	1	30	0	Dublin	SMITHFIELD NORTH	41
1	Parnell Square North	1	20	0	Dublin	PARNELL SQUARE NORTH	30
2	Pearse Street	1	30	0	Dublin	PEARSE STREET	31
3	Excise Walk	0	40	0	Dublin	EXCISE WALK	40
4	Fitzwilliam Square West	0	30	0	Dublin	FITZWILLIAM SQUARE WEST	11



# Simple Queries

```
In [128]: sql = "select count(*) from station;"  
#for row in engine.execute(sql):  
#    print(row)  
print(engine.execute(sql).fetchall())
```

```
2017-03-26 20:17:24,678 INFO sqlalchemy.engine.base.Engine select count(*) fro  
m station;  
2017-03-26 20:17:24,683 INFO sqlalchemy.engine.base.Engine ()  
[(101,)]
```

```
In [130]: sql = "select name from station limit 10;"  
for row in engine.execute(sql):  
    print(row)
```

```
2017-03-26 20:18:02,341 INFO sqlalchemy.engine.base.Engine select name from station limit 10;  
2017-03-26 20:18:02,343 INFO sqlalchemy.engine.base.Engine ()  
('SMITHFIELD NORTH',)  
('PARNELL SQUARE NORTH',)  
('PEARSE STREET',)  
('EXCISE WALK',)  
('FITZWILLIAM SQUARE WEST',)  
('ST. JAMES HOSPITAL (CENTRAL)',)  
('HANOVER QUAY',)  
('OLIVER BOND STREET',)  
('COLLINS BARRACKS MUSEUM',)  
('BROOKFIELD ROAD',)
```

```
In [150]: # JOIN
# (SELECT name, number from station)
# on AVAIL.number = station.number
sql = """
select * from
(select available_bike_stands, available_bikes, number from availability
    where available_bikes > 0
    # add in more conditions here (eg. date/time)
    GROUP BY number)
a1
JOIN
(select name,number from station) a2
ON a1.number = a2.number limit 10;
"""

for row in engine.execute(sql):
    print(row)
```

```
2017-03-26 20:40:05,828 INFO sqlalchemy.engine.base.Engine
select * from
(select available_bike_stands, available_bikes, number from availability
    where available_bikes > 0
    # add in more conditions here (eg. date/time)
    GROUP BY number)
a1
JOIN
(select name,number from station) a2
ON a1.number = a2.number limit 10;
```

```
2017-03-26 20:40:05,860 INFO sqlalchemy.engine.base.Engine ()
(8, 22, 42, 'SMITHFIELD NORTH', 42)
(19, 1, 30, 'PARNELL SQUARE NORTH', 30)
(29, 1, 32, 'PEARSE STREET', 32)
(39, 1, 48, 'EXCISE WALK', 48)
(22, 8, 13, 'FITZWILLIAM SQUARE WEST', 13)
(0, 40, 81, 'ST. JAMES HOSPITAL (CENTRAL)', 81)
```



# Scrum Review

- During the first sprint planning meeting, pick the user stories for your initial sprint.
- Focus mainly on stories that will get your infrastructure and processes in place, but plan to have working software for the demo at the end of the sprint.
- Keep the coding simple for this first sprint.
- Coordinating the use of tools and processes and establishing high-quality practices from the start is important
- For the next sprint, focus on getting the skeleton/prototype of a working app up and running.
- We will discuss Flask in the lecture on Friday