# Lecture Notes: *DevOps Tools*

- pre-commit
- ssh — the Secure Shell
- tmux — Terminal Multiplexer
- dotfile management
- shells
- package managers

**Note:** if you're making changes to your system then you ought to document them for your own sake. Where? Creating a GitLab Project somewhere and raising an Issue would be appropriate!

## pre-commit

https://pre-commit.com/

Install with `pip install pre-commit` or with your favourite package manager.

- Quick Start https://pre-commit.com/#quick-start.
- Automatically enabling pre-commit on repositories https://pre-commit.com/#automatically-enabling-pre-commit-on-repositories
- Creating new hooks https://pre-commit.com/#creating-new-hooks

The demo repo https://github.com/pre-commit/demo-repo provides an extensive demonstration.

## ssh - The Secure Shell

### Create a Key Pair

```
ssh-keygen -t rsa -C "James.Quilty@ecs.vuw.ac.nz"
```

The `RSA` algorithm is at present secure.

### Fingerprints

Check fingerprints (via another channel) whenever practicable! Use:

- `ssh-keygen -l` to output the fingerprint,
- the `-E` option to specify a hash algorithm other than the default,

- the `-f` option to specify a target file.

```
ssh-keygen -E md5 -lf ~/.ssh/id_rsa.pub
```

## Authorised Keys

`~/.ssh/authorized_keys` are the public keys of key pairs which are allowed to authenticate for this user.

Can manually copy keys, but it's more convenient to use `ssh-copy-id`

> ssh-copy-id is a script that uses ssh to log into a remote machine... By default it adds the keys by appending them to the remote user's `~/.ssh/authorized_keys`

## Known Hosts

`~/.ssh/known_hosts` keeps track of host keys, public keys are added to this file when you accept a connection to a remote machine... after verifying its fingerprint!

Find a host's fingerprint by running `ssh-keygen -l`. The host keys are typically found in `/etc/ssh/`. Example:

```
for filename in /etc/ssh/ssh_host_*key.pub;
do
  ssh-keygen -E md5 -lf $filename;
done
```

## SSH Agent

Entering your password for each connection rapidly becomes tedious. Use `ssh-agent` to cache your ssh credentials for you (see: `man ssh-agent`). The usual command you'll use is `ssh-add`. Example:

```
ssh-add; ssh-add -E md5 -l
```

## Port Forwarding

Also called "tunnelling". Forwards connections to a local port to a port on remote through an ssh connection. Example:

`ssh quilty@pan-de-muerto.ecs.vuw.ac.nz -fNL 15548:127.0.0.1:548`

### Jump Hosts

Connect to a remote via an arbitrary number of intermediates, using the `-J` flag. Example:

```
ssh -J embassy.ecs.vuw.ac.nz pi@10.140.153.132
```

### SSH Configuration

There are lots of useful options, for example *Jump Hosts*. Use a `~/.ssh/config` file to define hosts (see: `man ssh_config`). Example:

```
Host HTLPi
  ProxyJump embassy.ecs.vuw.ac.nz
  Hostname 10.140.153.132
  User pi
```

Can now connect by `ssh HTLPi` and public key authentication functions transparently. Also works with `scp`, no more

```
scp -J embassy.ecs.vuw.ac.nz 10.140.153.132:"long/path/to/some/file/or/directory/whi
```

instead can write

```
scp HTLPi:"<source>" <destination>
```

# tmux - Terminal Multiplexer

https://github.com/tmux/tmux/wiki

Convenience & Stability: easy to create sessions and windows; network disconnections *don't* hang the session, can always reconnect.

- Linux | CLI | DevOps https://knowledge.rootknecht.net/tmux
- Tmux Cheat Sheet & Quick Reference https://tmuxcheatsheet.com/

### tmux Configuration

This is the per-user configuration file for `tmux`. Put useful settings in here. Example:

```
# Start windows and panes at 1, not 0
set -g base-index 1
setw -g pane-base-index 1
```

# dotfile management

You should keep your dotfiles under version control.

dotfiles https://dotfiles.github.io/ *Your unofficial guide to dotfiles on GitHub.*

1. Dotbot: A tool that bootstraps your dotfiles https://github.com/anishathalye/dotbot
2. Managing your Dotfiles https://www.anishathalye.com/2014/08/03/managing-your-dotfiles
3. Dotfiles https://wiki.archlinux.org/index.php/Dotfiles
4. dotfiles: Dotfile Management Made Easy https://github.com/jbernard/dotfiles
5. A curated list of dotfiles resources https://github.com/webpro/awesome-dotfiles
6. Managing Dotfiles With Ansible https://dzone.com/articles/managing-dotfiles-with-ansible
7. Manage your dotfiles with Ansible https://medium.com/espinola-designs/manage-your-dotfiles-with-ansible-6dbedd5532bb
8. Ansible-based dotfile setup for macOS https://github.com/frdmn/dotfiles

# Shells

`bash` : old, ubiquitous because it is the default Linux shell

- https://www.gnu.org/software/bash/
- https://tiswww.case.edu/php/chet/bash/bashtop.html

`zsh` : new, has a growing community and is now the default macOS shell

- https://www.zsh.org/
- https://ohmyz.sh/

Important points:

- Command completions are invaluable
  *install & activate whenever possible*
- Latest version of `bash` is 5.1.8
  *need to install this to enjoy the latest completions*
- Latest version of `zsh` is 5.8
  *significantly enhanced by use of Oh My Zsh*

Package managers will typically install command completions for the active/default shell at the same time the main package is installed.

# Package Managers

You'll need to use one sooner or later. Sooner is better.

- **macOS:** Homebrew https://brew.sh/
- **Linux:** whatever is the distribution's default. On Debian or Ubuntu use `apt`.
- **Windows 10:** install the Windows Subsystem for Linux https://docs.microsoft.com/en-us/windows/wsl/install-win10
  The default WSL Linux distribution is Ubuntu, so then use `apt` as the package manager.

---