Ubuntu Server 16.04 LTS

Installation & Configuration

Last updated: 19.08.2018

0. Installing Ubuntu Server

0.1 Step-by-step Instructions:

- As part of your preparation, you should download the Ubuntu Server Installation ISO or Install CD from the official Ubuntu website. I recommend the first, so that you can create your own bootable LiveUSB flash memory stick, using the ISO you've downloaded
    - I created my bootable LiveUSB flash drive using "Rufus", on Windows.

- Remove all USB connected flash or hard disks, except for the LiveUSB or install CD you're about to install Ubuntu from.
- Boot from the LiveUSB or install CD
    - press F9 or whatever button required, as your machine starts, and direct the BIOS to boot from your CD-ROM or LiveUSB flash drive

- When GRUB starts up, select your preferred Language (e.g. English)
- In the menu, select the first option, "Install Ubuntu Server"
- Select your Language (again), location

- Configure the Keyboard
    - Use the TAB key to switch between onscreen options
    - If you know what keyboard standard layout you have > select NO
    - Else select YES to detect the keyboard layout > follow prompts

- Configure the Network
    - You should have your network login details ready for entry upon request
    - The installer will attempt to detect your network hardware
    - Primary Network Interface – select either Ethernet port, or wireless adapter
    - Hostname – pick a suitable hostname for your server, e.g. dycom-server

- Set up users and passwords
    - Enter the Full Name of the first (new) user when prompted (not root, or admin, etc) – capital letters is fine here, e.g. Andre Basson

    - Enter a username for your account – a short version of the Full Name – use lowercase only, e.g. andre
    - Enter a password (and prompted to confirm a second time)
    - Encrypt your home directory – select NO (or whatever you choose)

- Configure the clock – select the detected timezone if correct, else select NO and pick from a list
- Detecting harddisks, and partitions
    - Assuming you want to have a fresh install, and is not concerned of any data on the current drive (if any at all) – we'll now erase everything on the drive, and create new partitions
    - Unmount partitions that are in use? – select YES (can only erase and create new partitions if the partitions are unmounted)
    - Partition Disk(s)
        - Partition Method – select "Guided – use entire disk and LVM"
            - LVM (logical volume manager)
                - Most notably LVM allows us to extend (with LVM tools) the OS by adding more storage space later, without needing to reinstall the OS.
        - Select disk to partition – the one you want to install Ubuntu on
        - A summary of what's to happen is shown.
            - Remove existing logical volume data? Select YES.
        - Write the change to disks and configure LVM? Select YES.
        - Amount of volume group to use for guided partitioning:
            - Chose a smaller partition size for flexibility – can always increase it later – e.g. 35GB
        - Finish partitioning and write changes to disk – select YES
        - Write changes to disk - select YES
- Installer now partitions and formats your hard drive, followed by installing the system
- Configure the Package Manager
    - HTTP proxy information – leave this blank
- Configure tasksel
    - How to manage upgrades – you can select NO automatic updates (shown later how to enable this)
- Software Selection
    - Chose to install (use spacebar to select):
        - Standard system utilities
        - OpenSSH server – allows us to use **S**ecure **Sh**ell protocol to open a terminal interface to a remote computer, and control it as if we were sitting write behind it.
- Install the GRUB bootloader to the master boot record?
    - If Ubuntu will be the only OS installed on this machine, select YES.

    - Device for boot loader installation – choose /dev/sda – this will be the drive you boot from.
- Installation will now complete – you may remove the LiveUSB or install CD, and reboot.

- On reboot you'll be welcomed by the GRUB boot loader - you may choose to start Ubuntu or wait a few seconds for the timeout to select it by default. Ubuntu will now be booted, taking you to the terminal prompt, requesting a login and password you entered earlier during the installation.

1. Configure the Network for Your Server

At this point, you should have your Ubuntu Server installed, and you should have a terminal ready for entering some commands.

- Server needs a predictable IP address:
    - Dynamic IP can change
    - Static IP stays the same
- Configure a static IP address:
    - Set manually in /etc/network/interfaces
    - Set a reservation by MAC address on the DHCP server

Considering that you'd like your server to have a **static** IP address, but also do not want your DHCP service assigning the same number to another node/computer on the network, we pick the LAST option on the list above: to 'Set a reservation by MAC address on the DHCP server' or router. On a router the configuration is usually under your *network* > *LAN* > *DHCP reservation* dialogue or menu.

First find your network/wireless adapter's MAC address in your linux terminal with either of these commands:

```
ip addr show
ip a
ifconfig -a
```

You will likely have more than one Ethernet or Wireless network interface adapter, so the output will show all the adapters' connection details. You have to pick the one you intend to have connected to the network. The MAC address of the adapter is the 6 hexadecimal numbers separated by a colon:

e.g. 00:1c:df:47:92:85

Back in the router DHCP reservation dialogue, you'll add the MAC address, along with the IP address you'd like to reserve. Make sure the IP address is on the same network typology as is assigned to the rest of the network, usually this implies you only have to set the last number in the IP address, e.g.:

192.168.1.<u>100</u>

Once you have added the reservation, you'll usually either have to reboot your router, or restart the DHCP service (if the option is available). A restart of your Ubuntu server is required before you'll see the IP address being assigned:

```
sudo shutdown –r now
```

Once restarted and logged in; confirm the IP address with the command shown earlier.

1.2 Troubleshooting IP address from Router/DHCP Server

Sometimes your server fails to get an IP address from your DHCP server or router, and sometimes you would just like to release and renew your IP address.

Steps:

- First you should confirm whether your server's network adapter has an IP address assigned, by using either the following terminal commands:

```
        ip a
OR      ip addr show
```

- If no IP address is assigned for the network adapter connected to the network, renew the IP address with these commands:

```
sudo dhclient –r            #Release current address
sudo dhclient eth0          #Request new address (eth0 being your network
                            #connected adapter/interface, your's might be differ)
```

…if the above doesn't work, try these:

```
sudo dhclient –r            #Release current address
rm /var/lib/dhcp/dhclient*  #Client might not release IP if it thinks it's valid (file
                            #loction may differ on your machine).
sudo dhclient eth0          #Ask for new address (eth0 being your network
                            #connected adapter/interface).
```

2. SSH Server: Accessing from Another Computer

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users. If you didn't install SSH (aka *SSH Server*) during the Ubuntu installation process, do so now with:

```
sudo apt install openssh-server
```

The SSH demon (or SSHD, a service) listens for incoming connections (at port 22 by default), and allows for remote terminal access to the server. Allowing *root* to login remotely (or even locally) to the server, is a security risk. Instead we'll make sure that the user we created during the installation process does have access instead.

2.1 To disable *root* from login, we edit the /etc/ssh/sshd_config file:

```
    sudo vi /etc/ssh/sshd_config
OR  sudo nano /etc/ssh/sshd_config        #this one is easier to work with!!
```

Change the following line as follows:

From	*PermitRootLogin*	*prohibit-password*
to	*PermitRootLogin*	*no*

Now save the file:

- for vi: press ESC, type :wq, followed by ENTER
- for nano: press CTRL+x (to exit), then 'y' (for yes) to save, followed by ENTER

Confirm the change: **cat /etc/ssh/sshd_config | less** (press 'q' to quite the *less* application)

Restart the SSH server:	**sudo systemctl restart ssh**	(or **sshd**)
Check SSH server status:	**systemctl status ssh**	(or **sshd**)

2.2 Remote Access

Having started the SSH server, you can login remotely from any of the following platforms (note, the SSH terminal window will be tiny on tablets and phones!):

Platform	**Software (client)**
Windows	PuTTY (putty.org), Git BASH
macOS / Linux	Terminal application
iOS	Prompt, etc.
Android	JuiceSSH, ConnectBot, etc.

For access from Windows based machines; I highly recommend Git BASH, as it simulates the Linux terminal quite well.

Note that SSH works with port 22 by default, and we have no reason to change this at the moment. Do remember to configure all clients to use this port to connect with your server.

Example: SSH CONNECTION WITH GIT BASH

- Open Git BASH terminal window
- **sudo ssh username@serverIPaddress**
- e.g. **sudo ssh andre@192.168.178.40**
- you should now see the server SSH login prompt

3. Helpful Tips and Tricks

3.1 Customize BASH prompt:

Connecting via SSH software (client) to your Ubuntu SSH server (see earlier chapter) from a machine with a GUI will make customizing the BASH prompt a lot easier. So let's go ahead and login remotely.

Changing the prompt is purely for aesthetics, but it can make reading the prompt a lot easier:

- e.g. change colouring, spacing, and visible information
- /home/username/.bashrc (e.g /home/andre/.bashrc)
- www.bashrcgenerator.com
  - creates prompt configurations for you
  - easier way than manual editing .bashrc file
  - drag & drop what you want in your prompt
  - copy-paste output to bottom of .bashrc file
    - use terminal editor 'nano',
      - e.g. nano ~/.bashrc or $HOME/.bashrc

.bashrc runs when BASH starts – so type *exit* to stop (disconnect) your session, then login again. You should see the changed prompt.

3.2 Terminal Multiplexers

tmux is a software application that can be used to multiplex several virtual consoles, allowing a user to access multiple separate terminal sessions inside a single terminal window **or** remote terminal session. It is useful for dealing with multiple programs from a command-line interface, and for separating programs from the linux shell that started the program.

I.e.:

- it sets up a virtual terminal, within your terminal session
- SSH Terminal session stays active as long as tmux is running
- lets you connect and reconnect to the same terminal environment, even if your session dropped unexpectedly
- Two terminal multiplexer apps:
    - Screen – older
    - Tmux – newer
        - Install with: ***sudo apt install tmux***
        - Open with: ***tmux***

Tmux

The command *tmux* opens a virtual terminal window **on the server**, inside your original (SSH) terminal session. The window is tabbed (and numbered) in the taskbar at the bottom (e.g. *0 : bash*). We can have many windows (and even further sub-windows aka panes) in the terminal session, between which we can switch as we see fit.

Tmux is text based, so knowing short cut keys will help.

- [Ctrl + b]
    - known as the tmux *prefix*
    - it preludes many tmux commands you execute once you're in tmux
    - the prefix is pressed and released prior to the commands you wished execute

- [Ctrl + b] + c
    - i.e. press Ctrl + 'b', **release all buttons**, then press 'c'
    - creates a window (or even sub window, aka pane) inside your session
    - you should see another tab appear in the task bar for this window

- [Ctrl + b] + 0; [Ctrl + b] + 1; etc.....
    - To switch to numbered window in taskbar
    - An asterisk indicates which window is currently in view

- [Ctrl + b] + n; [Ctrl + p]
    - To switch to next or previous window in taskbar

- [Ctrl + b] + " - splits current window or window-pane into two halves horizontally
- [Ctrl + b] + % - splits current window or window-pane into two halves vertically
- [Ctrl + b] + arrow key - switches between window panes
- [Ctrl + b + arrow key] - resizes pane
    - resizing multiple panes with one of the five tmux presets:

- [Ctrl+b] [M-1] # vertical split, all panes same width
- [Ctrl+b] [M-2] # horizontal split, all panes same height
- [Ctrl+b] [M-3] # horizontal split, main pane on top,
  Other panes on bottom, vertically split, all same width
- [Ctrl+b] [M-4] # vertical split, main pane left, other panes right,
  horizontally split, all same height
- [Ctrl+b] [M-5] # tile, new panes on bottom, same height before
  same width

Where *M* denotes the *meta key*, usually bound to *ALT*.

- [Ctrl + b] + z - toggle window/pane fullscreen, or back to previous size
- [Ctrl + b] + x
  - To close the current window or pane (whichever is in focus)

- `[Ctrl + b] :kill-session`
  - To close the session at once, incl. all its windows and/or panes

- [Ctrl + b] + w to get an interactive index to choose from (0-9a-z)
- Add bindings to cycle through quickly in *tmux.conf*
```
bind -r C-h select-window -t :-
bind -r C-l select-window -t :+
```

The -r in the last one lets you repeat the key without having to repeat C-b. Typically the second one is the least number of keystrokes.

You can run more than one tmux session. To list all tmux sessions running, type:

- `tmux ls` or `tmux list-sessions`.

If you don't give a specific name to a tmux session you started, the session is given an incremental zero-indexed number (to differentiate from other sessions). Now we could just rely on the session numbers, but it would make our life much easier if we give our sessions names based on their intended use. To start a new session with a specific name we can just do the below:

- `tmux new –s "name of session"`

(or `tmux new-session –s "name of session"`)

where `–s` option refers to the source (i.e. supplied session name)

If you were to lose your connection (or `detach`), you can continue to where you were (as long as the tmux session is running on the server) by typing:

- `tmux attach` (or `tmux a #`) - to attach to the last created session
- `tmux attach-session –t 3` - to attach to session to session #3

- `tmux attach-session -t "session_name"` - to attach to session by name

If you want to detach from your tmux session on the server, you can simply type:

- `tmux detach`   or   [Ctrl + b] + d

Note: `detach`ing from a tmux session from a remote location will only close the remote window session. The tmux session session will remain on the server – hence why you can `attach` again.

Of course, should your server be restarted (eg. after power outage or reboot), tmux will not be running and your session windows will be gone.

3.3 Upload and Download Files from the Command Line

Download files

- **wget**
- i.e. wget someURL

Transfer files using SSH

SSH supports two different tools (client apps) for sending and receiving files:

- **sftp**
- **scp**
  - Secure Copy

If you're going to move files around, I suggest using SCP. If you're going to use terminal commands (e.g. from a local linux box or Windows running BASH shell software), I suggest you use SCP. SCP works like 'cp', and we can use it to copy a file (or files) to your remote server via SSH:

Assume you want to copy file *movie.mp4* from a local Downloads directory ie a user's home directory, to your remote server using SSH and SCP:

i.e.:
```
localUser@localMachineName: ~path/to/file $
        scp filename remoteUser@serverIPaddress:[pathRelativeToHomeFolder]
```

e.g.
```
andre@Asus-PC:~/temp $
        scp movie.mp4 andre@192.168.178.40:
```

The above example, with no input after the ':' will copy the file from local directory to andre's home directory. If you'd like to copy a file from your remote server to a local machine, you'd first supply the remote file location, followed by the local path you'd like to save it in:

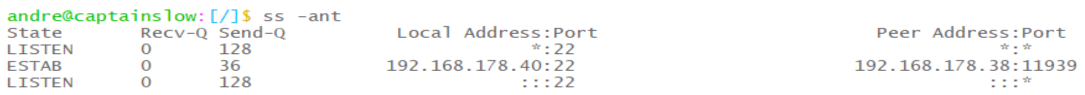
i.e.
```
localUser@localMachineName: ~path/to/file $
        scp remoteUser@serverIPaddress: [pathRelativeToHomeFolder] localPath
```
e.g.
```
andre@Asus-PC:~/temp$
        scp andre@192.168.178.40: movie.mp4   ~/temp
```

Sometimes (and preferably) we have the luxury of a GUI tool on our local machine. On windows, I suggest installing and *WinSCP* (*https://windscp.net*): a *Total Commander* like application and interface. Recall that both SFTP and SCP make use of the SSH protocol, so do remember to configure it to use port no. 22.

Also keep in mind, that whatever the user wrights are for the login/password entered, the client will be able to access remotely.

3.4 List Ports Listening and Connected

- Important to be able to view what ports are listening and is connected
- **ss** (socket statistics)
  - shows what application(s) has established connections (sockets) and is listening for data
  - for command help: *man ss* .
  - e.g.:
    - **ss –t** (where, t = tcp sockets)
    - **ss –ant** (where, a = all connections, n = port numbers)



```
andre@captainslow:[/]$ ss -ant
State       Recv-Q Send-Q          Local Address:Port                       Peer Address:Port
LISTEN      0      128                       *:22                                    *:*
ESTAB       0      36            192.168.178.40:22                    192.168.178.38:11939
LISTEN      0      128                      :::22                                   :::*
```

- socket (connection) established between remote peer at IP 192.168.178.38 (port 11939) to server port 22 (i.e. SSH connection)
- from this we can see that port 22 is listening, as is expected when we're running SSH.

3.5 Keyboard Reconfiguration

Scenario: Interchanging between `numpad` and `directional keys`.
Problem: `Shift+NumPad7` does not action a Home event, but prints 7 instead.

The solution lies in the form of configuring the Ubuntu keyboard driver to follow an effect as found in *Windows* operating systems. You need to add the `numpad:microsoft` option to the `XkbOptions`. On older Ubuntus, do that in your `xorg.conf`. On newer ones open the file `/etc/default/keyboard` and change this line:

```
XKBOPTIONS=""
```

to

```
XKBOPTIONS="numpad:microsoft"
```

Save and reboot (restarting `X` doesn't seem to work, at least not with `RAlt+PrintScreen+K`).

You may need to run the keyboard configuration package under the Debian package handler (`dpkg`) with terminal command

```
    sudo dpkg-reconfigure keyboard-configuration
```

for changes to take effect. Then, either restart of run the following *system control* command for the new keyboard configuration to take effect:

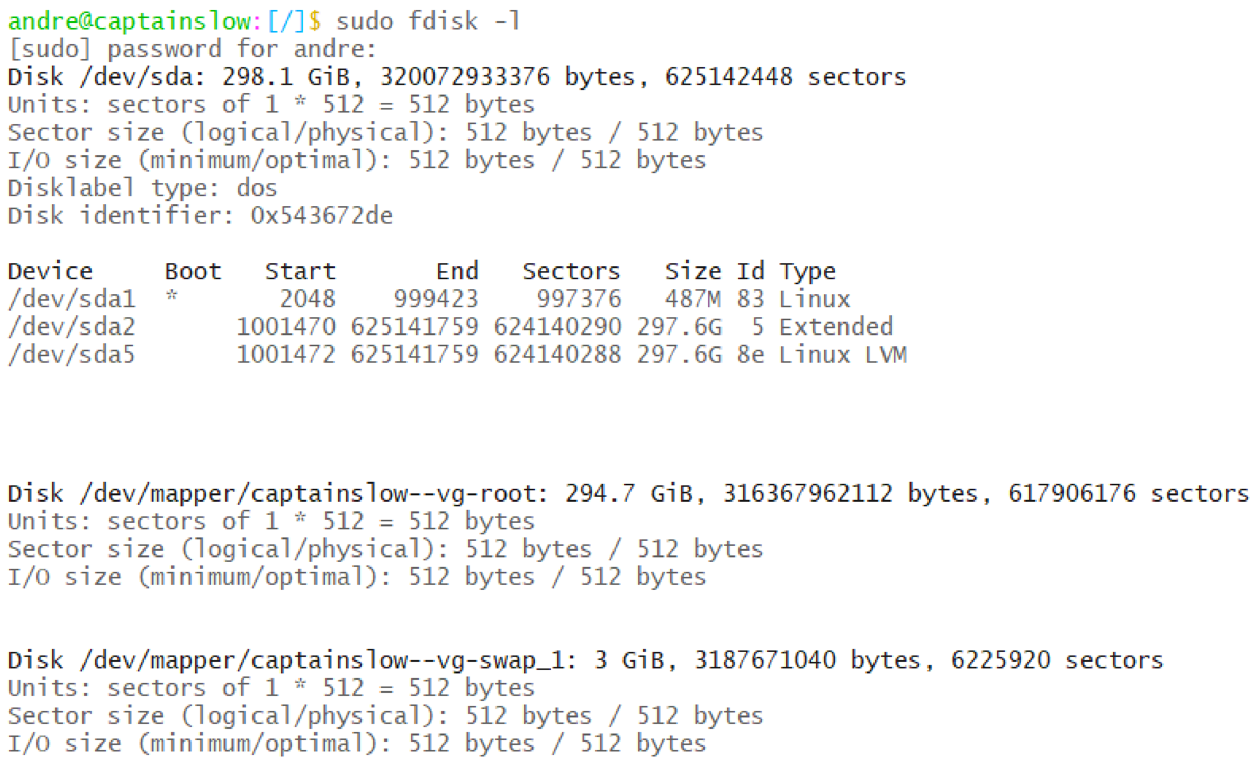
```
    sudo systemctl restart keyboard-setup
```

4. Adding Storage Devices

- always separate files from OS
- Add external or internal storage

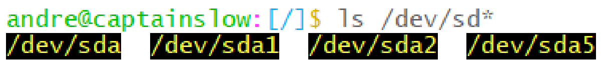
4.1 List Current Storage Devices

- **sudo fdisk –l**



```
andre@captainslow:[/]$ sudo fdisk -l
[sudo] password for andre:
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x543672de

Device     Boot   Start       End   Sectors   Size Id Type
/dev/sda1  *       2048    999423    997376   487M 83 Linux
/dev/sda2       1001470 625141759 624140290 297.6G  5 Extended
/dev/sda5       1001472 625141759 624140288 297.6G 8e Linux LVM



Disk /dev/mapper/captainslow--vg-root: 294.7 GiB, 316367962112 bytes, 617906176 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/captainslow--vg-swap_1: 3 GiB, 3187671040 bytes, 6225920 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

- All physically attached storage devices (called serial devices) are represented in the /dev folder
    - You can list them (and there partitions) with: **ls /dev/sd***



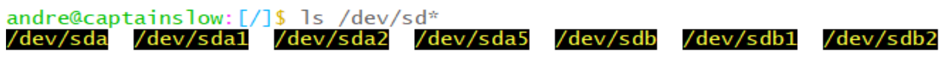
```
andre@captainslow:[/]$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda5
```

- Serial device 'a' (i.e. /dev/sda) usually represents your primary (OS installed) hard drive
- Partitions:
    - */dev/sda1* indicates partition 1 (Linux boot partition)
    - */dev/sda1* indicates partition 2 (Linux extended partition)
    - */dev/sda5* indicates partition 3 (Linux Logical Volume Partition)
- LVMs

- *Disk /dev/mapper/captainslow—vg-root* and
- *Disk /dev/mapper/captainslow—vg-swap_1* represents the two LVMs defined under the */dev/sda5* partition

4.2 Connect Storage Device (e.g. external USB hard disk)

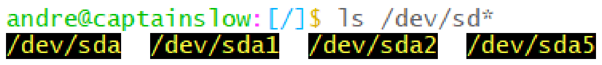
- Connect the USB hard drive
- List the serial devices: **ls /dev/sd***
  - Ex(1): disk with existing partitions (and file system(s))

    ```
    andre@captainslow:[/]$ ls /dev/sd*
    /dev/sda   /dev/sda1   /dev/sda2   /dev/sda5   /dev/sdb   /dev/sdb1   /dev/sdb2
    ```

    This output indicates that serial device 'b' has been connected, and that it has two partitions already. Run fdisk –l to see more information about the file systems (existing in each paritition) on sdb.

  - Ex(2): disk with no partitions (thus no file systems)

    ```
    andre@captainslow:[/]$ ls /dev/sd*
    /dev/sda   /dev/sda1   /dev/sda2   /dev/sda5
    ```

    The output indicates that serial device 'b' has been connected, but that it has no partitions.

4.3 Add a Partition and Filesystem

Having connected and listed the new serial device connected to the machine, we have yet to gain access to it. At this point the computer is electronically aware of the hard disk attached, but it cannot read the content on it. For this to happen, we have to **mount** the file system hosted within the partition (or volume) we want to access. And for this to happen, a partition (with a file system) has to exist. If not, the disk is considered 'clean' and we can go ahead and create any number of partitions.

But first an important difference about partition types! We'll consider NTFS (from Microsoft Windows) and EXT (from Linux). The primary difference is that NTFS and EXT are **not** compatible. Linux systems has built-in tools to read NTFS paritions, but Windows conversely cannot read EXT partitions.

As this distinction does not fall within our scope at this time, we'll continue assuming we're working with an *unpartitioned* hard disk as shown in example 2 in the previous section, or at the very least, a hard disk of which we do not wish to preserve the partitions (and data) on it.

Steps:

1) Create a new Linux partition table

- **sudo fdisk /dev/sdb**
    - note that fdisk doesn't write any changes we make until we give specifically chose the command to write the changes (a last chance to change your mind!!)

- press 'm' to see the menu of command options
- press 'p' to see the partition table
    - if the disk has existing paritions, you'll see it listed under /dev/sdb1, 2 etc.

- press 'd' if you wish to delete these partitions
- press 'g' to create a new empty GPT partition table
    - this allows us to create more than four partitions per disk
    - handy if we want to add more paritions later, or resize the partitions

- press 'n' to create a new partition inside the partition table
    - the size of a partition is determined by the number of disk sectors we chose
    - for now we only want to create one partition that'll take up all the size available in the partition table (we can always shrink it later), so we accept the defaults by pressing ENTER at the prompts
        - if we would've like to set the size of the new partition, we would change the LAST SECTOR entry to reflect the number of sectors we wish to ascribe to this partition
    - you should see an output indicating that a new Linux Filesystem partition has been created

- press 'p' to confirm the partition table (and partition) created
- press 'w' to write the partition table to the disk

2) Create a Linux filesystem on the partition

- At this point, we have yet to create a specific Linux filesystem on the partition we created above (confirm with fdisk –l command)
- To create one, we'll call the *make filesystem* command to create the filesystem on the *sdb1* partition:
    - **sudo mkfs.ext4 /dev/sdb1**

4.4 Mount the File System

In order for the OS to access the new filesystem (so we can read/write to and from it), we have to mount it to a file location we can access:

- usually mounted file systems are stored under /mnt
- let's make a new directory (aka mount point) under /mnt, to which we can

mount the filesystem. The mount point is the path at which the disk (partition) is made available to us for access:

  - **sudo mkdir /mnt/storage**

- now mount the filesystem on partition sdb1 to the mount point
  - **sudo mount /dev/sdb1 /mnt/storage**

- conversely, should you wish to unmount the partition:
  - **sudo umount /mnt/storage**

- confirm the mount with:
  - **df –h**

4.5 Change File/Directory Permissions

In order for users to be able to read, write and execute (or traverse) the filesystem, we must change the `rwx` permissions (aka `Base ACL entries`) of the files or directories on the filesystem. File permissions are assigned according to the `owner` (abbreviated to "u", for user), `group-owners` (or "g" for group), and all other `users` (or "o", for other).

The Linux command used to assign or modify permissions is `chmod`. `Chmod` (or Change Mode) will suffice in many (90%+) scenarios and should be the defacto method to assign permissions.

For all other scenarios where more fine tuning is required (e.g. assigning permissions to a subdirectory but not its parent, or assigning permissions to more than just one owner or group); `Access Control Lists` (aka ACLs or Extended ACLs) should be used instead. Do note that ACLs should be used at minimum, as assigning fine-grained permissions could easily turn into a giant mess. For that reason, when using ACLs (or chmod for that matter), it is advised to assign permissions on a *group* basis rather than individual *user*, as far as possible. This will make future amendments easier to maintain.

Also keep in mind that ACLs assigns *extra* (extended) permissions above and beyond (base) `rwx` permission set user, group and others. Any deletion of ACL entries has no bearing on base permissions of the file or directory in question.

4.5.1 chmod

We can confirm the current permissions to the mounted file system under the `/mnt` directory with command:

- **ls –l /mnt**
- output:

```
total 4
drwxr-xr-x 3 root root 4096 Jan  4 22:41 storage
```

- From the output, we deduce the following:
  - root root — - the 1st refers to the user owning the directory
    - the 2nd refers to the group owners of the directory
  - drwxr-xr-x — - 'd' indicates that 'storage' is a directory
    - rwx indicates permissions reserved for the OWNER (i.e. read/write/execute)

- r-x indicates permissions reserved for the GROUP OWNERS
  (i.e. read & execute only)
- r-x indicates permissions reserved for all OTHER users
  (i.e. read & execute only)

- We grant (or retract) permissions to a drive or directory using the 'change mode' command:
  - with the following command everyone will gain access to the mounted filesystem
    - **sudo chmod ugo+rw /mnt/storage**
      - where, *+rw* grants read & write access to the *user* (owner), *group* (group owners) and other (all other users)

  - conversely, should we wish to retract permissions, say…write permissions for other users, we can give the command:
    - **sudo chmod o-w /mnt/storage**
      - where *–w* retracts write permissions for *o* (other users)

  - note that we don't add execute wrights (ie. 'x', eg. '+rwx') because we're intending to use this drive for storage only
  - at a later stage we'll amend the permissions for subfolders as we see fit

- 'x' for *execution* wrights, but also for directory traversal
  - if you want to, say, grant access to a subdirectory by all OTHER users, but give no wrights to that subdirectory's parent (or ancestral) directory or its content, then you would have to at least grant *traversal* access so that other users can reach that directory:
    - **e.g. a hypothetical directory hierarchy**

      `/dir1/dir2/dir3`: read, write and execute to OTHER users
      `/dir1/dir2`: no permissions to OTHER users
      `/dir1`: no permissions to OTHER users

      In the hypothetical directory hierarchy above, in order for `OTHER` users to have [rwx] permissions to `dir3`, but none of the directories higher up in the hierarchy; each of parent/ancestral `dir1` and `dir2` will have to have at minimum 'x' permission added to their OTHER permission slots, in order to allow traversal of higher up directories, i.e.

      sudo chmod o+x /dir1
      sudo chmod o+x /dir1/dir2

      Now OTHER users can *traverse* (change directory, `cd`) all the way to `dir3`, while not having any read or write permissions to any of the content in the lower directories. **Do note** however, that if the exact file path to a file in a higher up directory is known, OTHER users will still be able to execute (access) those files if they know their exact names.

4.5.2 ACLs

`Access Control Lists` (ACLs) is a way to assign file and directory permissions in Linux, but in a fine-grained way that the `chmod` command cannot.

When the `chmod` command is used, only **one** *owner* and **one** *group* can be assigned permissions on a file or directory. If multiple users need access to a resource we need to place them in a group, and then give that group the necessary [rwx] permissions. But with ACLs in Linux, we can assign fine grained permissions to each user and group on a file (or directory) and even deny access to a particular user even if the file has world (other) permissions.

Following are the ACL basics that will suffice in most cases – more can be found online.
(e.g. https://help.ubuntu.com/community/FilePermissionsACLs)

4.5.2.1 ENABLING ACL IN THE FILESYSTEM

As of Ubuntu 14.04 and for file format ext4; ACLs are enabled by default. If the ACL package has not been installed, you will get a `command not found` error whenever the two main ACL commands (`getfacl` and `setfacl`) is executed.

You can however confirm that ACL is enabled and ready to use on storage device, by executing the following command on, say, the device at path `/dev/sdb1`, e.g.:

```
    sudo tune2fs -l /dev/sdb1 | grep acl
```

The command output is piped (`|`) to `grep` where it is matched for pattern `acl`. Should the final output from grep resemble an output similar to the following, then we know ACL is installed and functional:

```
        Default mount options:    user_xattr acl
```

Another method to verify ACL is installed and enabled can be done using mount (ref. Next section).

Older Ubuntu Releases

If ACLs are not installed, proceed with the following.

Before beginning to work with ACLs, the file system must be mounted with ACLs turned on. This can be done in `/etc/fstab` for the changes to be permanent.

- It may be necessary to install `acl` utilities from the repositories. In the Server Edition, this must be done, but in the desktop editions `acl` is installed by default.
  - `sudo apt-get install acl`

- Add the option `acl` to the partition(s) on which you want to enable ACL in `/etc/fstab`.
  - eg1:
    ```
    UUID=78ac9ba3-...... /mnt/dycom_rsync_bkp1 ext4 defaults,acl 0 2
    ```
  - eg2:
    ```
    /dev/sda3  /path/to/mount/point  ext4  defaults,acl 0 0
    ```

- If necessary, remount partition(s) on which ACLs were enabled for them to take effect, e.g.:
    - `sudo mount -o remount /path/to/mount/point`

- Now verify that ACLs are enabled on the partition(s) as follows; you should see a list of mounts indicating if acl is installed (highlighted as a mount flag) or not:
    - `Mount  |  grep acl`

4.5.2.2 ACL ENTRIES

ACL entries – also sometimes referred to as `Extended ACL Entries` - consist of a user (u), group (g), other (o) and an effective rights mask (m). An effective rights mask defines the most restrictive level of permissions.

`setfacl` **sets** the permissions for a given file or directory. `getfacl` **shows** the permissions for a given file or directory.

Defaults for a given object can be defined.

ACLs can be applied to users or groups but it is easier to manage groups. Groups scale better than continuously adding or subtracting users. Apply ACLs to users only when the intention is to fine-tune permission on individuals alone.

getfacl

To get the ACL for a file/directory use the `getfacl` command:

- i.e.: `getfacl /path/to/file/or/directory`

- e.g.: `getfacl /mnt/4TB_bkp1`

  output:
  ```
  getfacl: Removing leading '/' from absolute path names
  # file: mnt/4TB_bkp/
  # owner: root
  # group: andre
  user::rwx
  group::rwx
  group:macrium-users:--x
  mask::rwx
  other::---
  ```

  where:
  ```
     # owner: root                - indicates basic ACL entry
     owner
     # group: andre               - indicates basic ACL entry
     group
     user::rwx                    - basic ACL owner permissions
     group::rwx                   - basic ACL group permissions
     group:macrium-users:--x      - extended ACL group permissions
                                    (group 'macrium-users' has
  execute/
                                     traverse permissions only)
     mask::rwx
     other::---                   - basic ACL other permissions
  ```

setfacl

To set the ACL for a file/directory use the `setfacl` command:

- e.g.: `setfacl -m u:username:rw  /path/to/file/or/directory`

  In this command, `–m` is for *modify*, `u` is for *user*, followed by the username, `rw` for read and write permission, and finally the file or directory to apply the permission(s) to.

For giving all the permissions use `rwx`, for denying all permissions use minus (-). For example, to deny all rights to *user1* on `/path/to/file` even if the file has full "rwx" permissions to all of user, group and others (777), use the following command:

- e.g.: `setfacl –m u:user1:- /path/to/file/or/directory`

To grant all permissions to a *group* instead, follow the following syntax:

- e.g.: `setfacl -m g:groupname:rwx /path/to/file/or/directory`

To recursively set ACLs to all files inside a directory use the -R (recursive) option

- e.g.: `setfacl -R -m u:username:rwx /path/to/directory`

To delete an entry from the access list for a specific user or group

- e.g.: `setfacl -x u:username /path/to/file`
- e.g.2: `setfacl -x g:groupname /path/to/file`
- e.g.3: recursive deletion of ACL entries to directory incl. subdirectories
  `setfacl -R -x g:groupname /path/to/directory`

To delete all ACL entries (file/directory), we use the `-b` modifier

- e.g.: to remove all ACL entries for a file or directory – but retain the base (rwx) ACL entries of the owner, group and others:
  - `setfacl -b /path/to/file/or/directory`

4.5.2.3 *EXAMPLE SCENARIO:*

If we were to revisit the hypothetical scenario from section 4.5.1.; an example where we grant permission to all *other* users of a subdirectory (but not its parent directory); we can now instead grant fine-tuned permissions to *specific* users or groups that we add using (extended) ACLs.

For this scenario, we continue on the assumption that the following exist:

1. a group *macrium-users*
2. a user *macrium-backup_asus-x550j* belonging to group *macrium-users*

Let's assume the directory hierarchy has the following structure and permissions

`/dir1/dir2/dir3`:	drwxrwx--- root root (i.e. rwx permissions to root *user* & *group* only, none to *others*)
`/dir1/dir2`:	drwxrwx--- root root
`/dir1`:	drwxrwx--- root root

Let's assume the directory hierarchy should have the following extended permissions

`/dir1/dir2/dir3`:	read, write and execute to *macrium-backup_asus-x550j* user
`/dir1/dir2`:	execute/traverse to *macrium-users* group
`/dir1`:	execute/traverse to *macrium-users* group

ACL Entries (precede with `sudo` to gain root privilege):

- Assign traverse (execute) permission to **group** *macrium-users* to all higher directories, so as to allow traversal (change directory) permission to *dir3*, for all users in this group
  - `setfacl -m g:macrium-users:--x /dir1`
  - `setfacl -m g:macrium-users:--x /dir1/dir2`

- Now assign full `rwx` permissions for user *macrium-backup_asus-x550j*, to *dir3*
  - `setfacl -m u:macrium-backup_asus-x550j:rwx /dir1/dir2/dir3`

5. Mount External Disks at Boot

We don’t want to manually mount external disks every time the server is restarted. Moreover, the disk might not become connected at the location we expect it (i.e. in a different location other than /dev/sdb, e.g. /dev/sdc or /dev/sdd etc.).

Instead, we edit the filesystem table stored in the *fstab* file, at */etc/fstab*, to execute the mount (and any other mount we’d like) automatically at boot time. To bypass the problem of variant device location, fstab lets us make use of the more robust UUID number (Universally Unique Identifier) of the device, to mount the disk partition no matter the location in our filesystem.

5.1 Filesystem Table /etc/fstab

Lets take a look at an example filesystem table:

```
# fs - device or filesystem (or device's UUID)
# mount - where to make the disk available
# type - what filesystem to use
# options - any special parameters (ro, rw, etc.)
#  (defaults: rw, suid, dev, exec, auto, nouser, async)
# dump - whether the disk ought to be backed up
# pass - priority for filesystem checking
#
# fs        mount          type    options   dump  pass
#
/dev/sda1  /              xfs     defaults  0     0
UUID=nnn   /mnt/storage   ext4    defaults  0     0
```

All lines preceded with ‘#’ are comments, and are not considered by the operating system when reading the this file. You’ll notice the 2nd last commented line to show a list of tabulated entries:

• fs	- the filesystem to mount
• mount	- the mount point
• type	- filesystem type
• options	- e.g. readonly (for now we accepts defaults)
• dump / pass	- we don’t consider this for now.

5.2 Adding Mount to Filesystem Table

Continuing with the example above…

The first line lists the mounted boot partition (/dev/sda1), to it's mount point at root ('/'). This line was added during the Linux installation process, and we do not touch this whatsoever!!

The second line represents the configuration required to mount our external disk partition. Instead of mounting our external disk using it's expected location, we're going to use it's UUID number instead to mount the partition to our designated mount point (e.g. /mnt/storage).

Steps:

1. For this to work, the mount point must be created, if not already created, before we restart the OS, or reload all mounts (*sudo mount –a*).
   - E.g.: **sudo mkdir /mnt/storage**

2. We also need to get the disk's UUID - we use the block ID (blkid) command:
   - **sudo blkid**
   - example output:

```
/dev/sda1: UUID="658cb7cc-46f9-4e2f-86b3-786f6ee283a4" TYPE="ext2"
PARTUUID="543672de-01"
/dev/sda5: UUID="dmtINm-wV2s-tuAY-yW1h-3xvJ-UUir-SJVTTg"
TYPE="LVM2_member" PARTUUID="543672de-05"
/dev/sdb1: UUID="6d3032e5-a7e4-45e2-86cb-40d72938346b" TYPE="ext4"
PARTUUID="d455e5b3-74cd-47ba-8a8d-d4b59309c85b"
```

3. Copy-paste the UUID to our new entry in the filesystem table:
   - **sudo nano /etc/fstab** #open the file with nano editor

   - scroll down to the bottom and add the following line, following the same (space delimited) convention used in the filesystem table:

```
UUID=6d3032e5-a7e4-45e2-86cb-40d72938346b /mnt/storage ext4 defaults 0 0
```

     - should you be interested to use ACLs (Access Control Lists) on the partition:

```
UUID=6d3032e5-a7e4-45e2-86cb-40d72938346b /mnt/storage ext4 defaults 0 2
```

   - save the changes (ctrl+w, then ENTER)
   - exit nano (ctrl+x)
   - confirm the changes with: *cat /etc/fstab*

4. Automount all the filesystems listed in the fstab file, and confirm:

- **sudo mount –a** #automount
- **df –h** #show all mounted filesystems

Now this disk's first partition will always be automounted, at boot time, whenever the system is restarted.

6. Connecting to Network Storage

6.1 Overview & Preparation

- NAS (network-attached storage)
- Offer file sharing over SMB protocol
  - compatible with Windows, Mac, and Linux

- Enable sharing on the network device (which hosts the storage device) and create a user that'll have access to the storage devise:
  - SMB (server message block) or CIFS needs to be enabled (if not by default)
  - We'll need the device's IP address, and login/password
    - device should preferably have a static IP address on the network for predictable access (just like your server)

- Mount manually or automatically
  - Similar to mounting a local disk
    - First we'll do manual mounting
    - Then automatic mounting (fstab file)

- My share name (i.e. name given to share in router with NAS):
  - "HomeNetwork.NAS" at IP address: 192.168.178.1
    - e.g. //192.168.178.1/HomeNetwork.NAS

  - If your router supports local DNS, you can use domain names instead
    - e.g. //FRITZ-NAS/HomeNetwork.NAS

6.2 Configure Server & Mount NAS

After completing the preparations above, complete the following:

- Install CIFS
  - Common Internet File System (CIFS)
    - Network protocol mainly used to share resources over a network

        - Support for SMB protocol – but not built into Linux by default
        - needed to connect to SMB servers
    - **sudo apt install cifs-utils**

- Create mount point on server where NAS will attach
    - E.g.: **sudo mkdir /mnt/nas**

- Manually mount the NAS, e.g.:

sudo mount –t cifs //192.168.178.1/HomeNetwork.NAS/Seagate-ExpansionDesk-03/BACKUPs/HP.dx2810 /mnt/nas –o [vers=1.0,][uid=<uid>,][gid=<gid>,]user=userWithAccessToNAS,password=userPasswordHere

(above is all on one line)

...where

-t	option to set filesystem type
cifs	the network protocol driver that connects to the NAS drive
//192.168.178.1/ HomeNetwork.NAS/...	network location & share name
/mnt/nas	mount point
-o	option that enables us to list mount options
vers=x.x	**optional**: should you get "host unknown" errors, try different versions as directed by the `systemd` journal logger utility **journalctl –fb**
uid=<uid>	**optional**: to afford read/write permissions of the user with this uid (e.g. uid=1000; alternatively: uid=<username>), to the mount. Often required, along with gid, to preserve file attributes such as time-stamp, ownership, etc. You can find the uid and gid of a user with command: **id <user_name>**
gid=<gid>	**optional**: to afford read/write permissions of the user with this gid (e.g. gid=1000; alternatively: gid=<groupname>), to the mount.
user=....	mount option (following -o) to indicate which user has writes to access the NAS
password=...	the password of the user to be authenticated by the network device hosting the NAS

You can confirm the mount with: **df -h**

- Automount the NAS (edit /etc/fstab):
    - We now use the IP address of storage devise, rather than UUID
    - Open /etc/fstab for edit:
        - **sudo nano /etc/fstab**

    - scroll down to the bottom and add the following line, following the same (space delimited) convention used in the filesystem table for other mounts:

```
//192.168.178.1/HOMENETWORK.NAS/Seagate-ExpansionDesk-03/BACKUPs/HP.dx2810  /mnt/nas  cifs
uid=1000,gid=1000,user= userWithAccessToNAS,password=userPassword  0  0
```

    - save the changes (ctrl+w, then ENTER)
    - exit nano (ctrl+x)
    - confirm the changes with: *cat /etc/fstab*

- Note that the password is stored in clear text (no other way around this)
    - It is therefore a good idea to use a different password than what you normally would pick, just in case your Linux box is compromised and you could risk more than just access to your shared NAS.

7. Security and Maintenance

- Remote Access
- Software Updates
- Security Updates
- Firewall settings

7.1 Remote Access Recommendations

- Disable root login via SSH
- Use a key rather than a password
    - Passwords are fine on your personal network over which you have full control
    - But for more security on the internet or corporate network, consider disabling password access…and only allow a login with a cryptographic key
        - We'll look at this later

7.2 Software Updates

- Software updates provide fixes, patches, and features
    - In Ubuntu, we use the apt command (advanced package tool) to do all updates
- 2-step process:
    - Update installed package lists
        - Updates the local package lists with the latest package lists available at remote depositories.
        - **sudo apt update**

    - Update packages
        - Compare installed software packages, with what's available in the package list, then update if newer versions are found
        - **sudo apt upgrade**
            - you will be displayed the pending upgrades, and prompted y/n to continue

        - **sudo apt –y upgrade**
            - same as before, but no prompts, the upgrades will commence without input from user

7.3 Security Updates

- https://help.ubuntu.com/community/AutomaticSecurityUpdates
- Install the "unattended upgrades package"
    - **sudo apt install unattended-upgrades**

- Use the debian package manager (dpkg) to configure unattended upgrades
    - **sudo dpkg-reconfigure unattended-upgrades**
    - This will open up an interactive interface to configure the software:
        - "Automatically download and install stable upgrades?" - tab to select YES
        - Set up the packages that need to match in order to be installed – select OK
            - The defaults are the most essential

- Your system will now automatically check for new stable security updates, and install them as required. You should however still login from time to time to see if new security updates are available. The server will prompt you at login about available upgrades and updates.

7.4 Firewall Settings

It is best practice to run a firewall, even on the server controlled solely by yourself, since a rogue devise within your network might intentionally or unintentionally compromise your information.

UFW (Uncomplicated Firewall) is a firewall configuration tool for *iptables* (another firewall package) that is included with Ubuntu by default. This cheat sheet-style guide provides a quick reference to UFW commands that will create iptables firewall rules that are useful in common, everyday scenarios.

For Ubuntu 16.04LTS, the rules are stored in directory in `/etc/ufw/user.rules` and other `*.rules` files at this path. When you turn UFW on, it uses a default set of rules (profile) that should be fine for the average home user. That's at least the goal of the Ubuntu developers. In short, all 'incoming' is being denied, with some exceptions to make things easier for home users.

Steps:

- The default ruleset should already be implemented, but to make sure, run these:
    - **sudo ufw default allow outgoing**
    - **sudo ufw default deny incoming**

- Create a firewall rule to allow access on ports you need to use
    - e.g. for SSH: allow TCP traffic on port 22
        - **sudo ufw allow 22/tcp**
- Enable the firewall
    - **sudo ufw enable** / **ufw disable**
        - ufw (Uncomplicated Firewall)
        - less complicated than *IP Tables* (common firewall package)
- To see firewall status
    - **sudo ufw status**

    or

    - **sudo ufw status numbered**

    or

    - **sudo ufw status verbose**

Other helpful and more advanced commands:

- To enable / disable logging use:
    - **sudo ufw logging on** subsitute 'on' with 'off' to switch off
    - logs stored by default in `/var/log/ufw.log`
        - to see in real time: **sudo tail –f /var/log/ufw.log**

- Allow/Deny Connections by port number range and protocol

    - **sudo ufw allow 1500:2000/tcp** allows all tcp packets to these ports
    - **sudo ufw deny 1500:2000/tcp** denies all tcp packets to these ports

- To block the port ssh (default port 22) is using:
    - **sudo ufw deny 22/tcp** this denies tcp packets on port 22

    or
    - **sudo ufw deny ssh**

- To delete the rule if you don't need it any more:
    - **sudo ufw delete deny 22/tcp** #22/tcp being the port and protocol

    or
    - **sudo ufw delete deny ssh**

- Block an IP address and Subnet
    - **sudo ufw deny from 15.15.15.51** #blocks traffic from source IP address
    - **sudo ufw deny from 15.15.15.0/24** #blocks all IPs on the subnet 15.15.15.xxxx

    - **NOTE:**
        - **The order of rules are important.**
            - See section on this following.

        - **How to find out which IP ranges to allow**, for example, from a list of IP addresses your computer might be assigned dynamically by a service provider?
            - Do a *whois* lookup on the public IP of the router, the one the host you want to give access to, is connected to. You can also do a whois on the domain name of your service provider. The output from whois usually gives you an indication of the IP range
                - First intall whois: **sudo apt install whois -y**
                - Do the lookup: **sudo whois miit.co.nz**

- Block Connections to a Network Interface
    - **sudo ufw deny in on eth0 from 15.15.15.51** #blocks traffic from IP address to eth0 interface

- Allow Incoming SSH Connection from Specific IP or Subnet
    - **sudo ufw allow from 15.15.15.0/24 to any port 22** #allow IPs on subnet to port 22

- Allow Incoming Rsync from Specific IP or Subnet
    - **sudo ufw allow from 15.15.15.0/24 to any port 873** #default rsync port 873

- Allow All Incoming HTTP or HTTPS Traffic
  (e.g. Apache and Nginx listens for traffic on ports 80 & 443 for HTTP and HTTPS connections, respectively)
  - **sudo ufw allow http** or **sudo ufw allow 80** #default http port 80
  - **sudo ufw allow https** or **sudo ufw allow 443** #default http port 80

- Allow All Incoming HTTP and HTTPS Traffic
  - **sudo ufw allow proto tcp from any to any port 80,443**
    - proto tcp required when specifying multiple ports

- Allow an IP range on the same Subnet (great calculator at http://jodies.de/ipcalc)
  - You need to consider a network/bit mask (2,4,8,16, or 32), to mask in the network IP range required to allow the number of hosts on the network you would like to give access to (good explanation here https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing#IPv4_CIDR_blocks):

  - e.g. to allow TCP packets to port 22 for max 32 hosts from 192.168.1.0 to 192.168.1.31:
    sudo ufw allow proto tcp from 192.168.1.0/27 to 192.168.1.31 to any port 22

Details

```
 Address:   192.168.1.0                 11000000.10101000.00000001.0000 0000
 NetMask:   255.255.255.240  (27bits)   11111111.11111111.11111111.1110 0000
 Wildcard:  0.0.0.31                    00000000.00000000.00000000.0001 1111

 HostMin:   192.168.1.0                 11000000.10101000.00000001.0000 0000
 HostMax:   192.168.1.31                11000000.10101000.00000001.0001 1111

  Total Hosts: 30  (192.168.1.0 and 192.168.1.31 is used the network address,
broadcast address respectively - thus not usable by hosts)
```

- more examples available at:
  - https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands

7.4.1 Firewall Rule Order

The order of rules are important.

- When you add a rule it gets appended to the rule set. So if you had a rule that allows everyone to connect to, for example, your server on port 80 – and this rule precedes another where you allowed or more IPs to connect to your machine – then the first first rule prevails leaving the second not being implemented.

How can we change the rules or rule-order already added?

- You can alter the rules and the order of the rules directly in the `/etc/ufw/user.rules` and `/etc/ufw/user6.rules` files for IPv4 and IPv6 (respectively); by manually editing the rule starting with ### tuple ###. This way you can edit an existing rule, or cut-paste the rule to a different line up or down depending on the order you want.
  - **WARNING**: it is the quickest way, but is not recommended, as the rules are stored in the more complicated *iptables* command syntax. Proceed when you no more.

- Another way of reordering rules would be via terminal commands - something like this, for example:

```
  $ sudo ufw status numbered

     To                          Action      From
     --                          ------      ----
[ 1] 22                          ALLOW IN    Anywhere
[ 2] 80                          ALLOW IN    Anywhere
[ 3] 443                         ALLOW IN    Anywhere
[ 4] 22 (v6)                     ALLOW IN    Anywhere (v6)
[ 5] 80 (v6)                     ALLOW IN    Anywhere (v6)
[ 6] 443 (v6)                    ALLOW IN    Anywhere (v6)
[ 7] Anywhere                    DENY IN     [ip-to-block]
```

Say you accidentally added a rule to the end, but you wanted it up top. First you will have remove it from the bottom (7) and add it back.

```
  $ sudo ufw delete 7           where 7 refers to the last rule above
```

Note, be careful of removing multiple rules one after another, their position can change!

Add back your rule to the very top (1):

```
  $ sudo ufw insert 1 deny from [ip-to-block] to any
```

8. Local Network Services (i.e. behind your firewall)

Overview

- File Sharing (SMB protocol, Samba Server)
- Media Streaming (Plex)
- Name Service
- Ad Blocking
- Server Status (via web interface)
- Desktop Environment
- File Backup

8.1 File Sharing with SMB & Samba

a. Sharing a directory with full permissions to every local user

PREPERATION

- **sudo mkdir /mnt/storage/shared** #create the shared folder/directory
- **sudo chmod 777 /mnt/storage/shared** #give full permissions to all users (user, group and others)
  - **equivalent:** sudo chmod ugo+rw /mnt/storage/shared
  - **NOTE:** if subdirectories exists (or will in the future) use the –R option (recursive):
    - **sudo chmod –R 777 /mnt/strorage/shared**

Configure Samba

- **sudo apt install samba** #install samba server to share directory
- **man smb.conf** #full help file
- **sudo nano /etc/samba/smb.conf** #edit samba configuration file (to add shares)
  - a lot in this file, we'll focus on the shares we're going to add
  - for now, scroll to the end of the file, and enter the following parameters (comments included after each line for explanation only - <u>do not include any comments on any line containing parameters (code)</u>)

```
[fileshare]                       #share name (as appears to other users)
 comment = Shared Files           #description; can be anything
 path = /mnt/storage/shared       #path to shared directory
 read only = no                   #default = read only; make it writable

 #Apply masks (permission templates) to new files and directories created in the shared folder,
 #  to prevent sole ownership by the users who created it.
 #  Permissions are set by 4no. octal values, where:  7 = 4+2+1 = r+w+x permissions  (e.g. 5 = 4+1 = r+x )
 #  The    0th octal:  to set special modes (default = 0, for no modes)
 #         1st octal:  sets permissions for user (owner)
 #         1st octal:  sets permissions for group
 #           1st octal:  sets permissions for other
 # The following masks will give everyone read-write-execute permissions to any file or folder created
 # in the shared directory
 #
 create mask = 0777
 force create mode = 0777          #sometimes required when create mask not masking permissions
 directory mask = 0777
 force directory mode = 0777       #sometimes required when directory mask not masking

 #force group:  **** handy if you're still having global permission issues ****
 # This specifies a UNIX group name that will be assigned as the default primary group for
 # all users connecting to this service. This is useful for sharing files by ensuring that
 # all access to files on service will use the named group for their permissions checking.
 #   Thus, by assigning permissions for this group to the files and directories within this
 #   service the Samba administrator can restrict or allow sharing of these files.
 force group = a_group
```

- save (ctrl+w, ENTER) and exit (ctrl+x) the file

- **testparm** #test syntax and parameters in smb.conf
- **sudo systemctl restart smbd** #restart samba server
    - sometimes accompanied by **sudo systemctl restart nmbd**

Add Samba user passwords

Only the users created for the Linux server (i.e. those with a Linux login) can have access to Samba shares. Samba does however require you to add a password for that user specifically for Samba itself. It can be the same password the user entered for Linux if you like. Just keep in mind that if you change your linux login password for a user, that it won't change that user's Samba password.

To view the current Samba users:

- **sudo pdbedit –L –v** #where –L lists all users, -v is verbose

To add a new Samba user:

- **sudo smbpasswd –a userName** #you'll be prompted to add a Samba password for this user.
- **sudo smbpasswd –e userName** #enables the user in Samba

- if you want, you can add more users to Linux, then add a new password for samba, e.g.:
    - sudo useradd user2 #conversely to remove: sudo userdel user2
    - sudo passwd user2 #add linux login password for user2
    - sudo smbpasswd –a user2

- sudo smbpasswd –e user2

Other Samba commands:

sudo smbpasswd userName	# modify password of samba user
sudo smbpasswd –x userName	# deletes samba user
sudo smbstatus	# show shares and current connections
sudo smbstatus --shares	# show shares and current connections

Allow Access for File Sharing Through Firewall

Lets allow all TCP traffic through ports 139 and 445 to the Samba server (natively required for Samba to work)

- **sudo ufw allow 139/tcp**
- **sudo ufw allow 445/tcp**

b. Sharing a directory with limited permissions

Essentially, you repeat all the steps and commands from (a), amending only the commands related to permissions you want different. That is, you only change the octal values in the *chmod* command, and octal values in the *masks* (in the *smb.conf* file).

E.g.1: Lets suppose you only want the *owner* and *group owners* of the shared directory to have full read-write-execute permissions, but other users to NO permission at all. To implement this, you'll repeat the steps from (a); changing only the following items.

- **sudo chmod 770 /mnt/storage/shared**
  - or **sudo chmod –R 770 /mnt/storage/shared**
- in smb.conf
  - **create mask = 0770**
  - **force create mode = 0770** #include if create mask doesn't mask correctly
  - **directory mask = 0770**
  - **force directory mask = 0770** #include if create directory doesn't mask correctly
  - **force user = a_user** #include if masking is not succesful
  - **force group = a_group** #include if masking is not successful

These highlighted lines to include for troubleshooting: only when masking does NOT assign permissions, or you find you cannot access shared resources.

c. Connecting to the Shared Folder

- macOS/Linux via Desktop Environment (GUI)
  - "Connect to Server" in the file browser
    - smb://192.168.178.40/fileshare (protocol://server_ip/share_name)

- Windows
    - File Explorer:
        - \\192.168.178.40\fileshare (NOTE the backslashes)
    -
    - "run" dialog
        - \\192.168.178.40\fileshare

- You should now be greeted with a login dialog box, where you gain access using the username and password you used at the *smbpasswd* command earlier.

d. CIFS and Windows Incompatible Special Characters

As described in this Samba link – the *vfs_catia* (or VFS) module translates characters that are illegal in CIFS/SMB filenames, and is part of Samba suite 7 and later. Translation involves mapping characters to legal characters, so that they can be shared with CIFS clients, incl. Windows based machines.

Without this module the share might be visible in the LAN, but the filename might be truncated or altogether changed to the point of being unrecognisable or even unreadable in some instances.

To enable this module, simply add the following lines to any samba share in the */etc/samba/smb.conf* file:

```
##catia vfs module:
# To map Windows-incompatible special characters to
# compatible alternatives (e.g. ':' to '÷')
      vfs objects = catia
      catia:mappings =
      0x22:0xa8,0x2a:0xa4,0x2f:0xf8,0x3a:0xf7,0x3c:0xab,0x3e:0xbb,0x3f:
      0xbf,0x5c:0xff,0x7c:0xa6
```

8.2 Exposing Samba Shares to the Internet

***** Never share anything to the internet that you don't have to *****

Having said that…

With Samba, you would have to expose ports 139/tcp and 445/tcp to the outside –

usually this involves configuring "port forwarding" in your router. Additionally, you must make sure that your external IP address is pingable* from the outside.

After this, you'll be able to access the shares by entering \\youraddress in Explorer's address bar or in Start - Run. (Here youraddress is either your external IP address or your DNS name*, if you have one.)

Note, however, that the CIFS and SMBv2 protocol used by Windows file sharing does not provide data encryption (so anybody with a packet sniffer can monitor your file transfers), and its authentication is not especially strong either. Only SMBv3 gained encryption support.

Also, don't forget that the Windows' SMB service has in the past been a very frequent infection target. While most Windows exploits do not affect Samba in any way, this is still worth remembering (and often means that the SMB ports get blocked at ISP level).

Also note that Windows machines by default remember the login credentials for the entire local session. Unless you're connecting to Samba as "Guest", you must take special care on public machines: always use net use \\address before opening in Explorer / net use \\address /del to disconnect. (This is not needed on personal computers.)

For some extra security, add the following to general section in smb.conf:
(note to self: I have not included this for dycom-server, as I do not intend to open ports 139 / 445 on the router to the internet)

```
LANMAN auth = no
NTLM auth = no
invalid users = root
```

** see chapter "Accessing Resources Remotely"*

9. Serve Video on the Network with Plex

- Your home server can act as a video server
- Video file sharing works on your normal file server…
  - But not for tablets or set top boxes
- Plex Media Server
  - Multiplatform media library management and streaming

This section TO BE CONTINUED (or see *Lynda Building an Ubuntu Server*, video 03_03)

10. Name to IP Resolution with *DNSmasq*

10.1 Introduction: Talking to a Server by Name

- IP addresses can be tricky to remember
  - Not useful to non-technical users

- Every computer has a hostname
  - We gave our server a hostname during installation
    - See elsewhere on how to change the hostname post installation

  - We see the hostname in our terminal prompt when we're logged in, but other computers don't know this name
  - You can use a name to talk to a computer

10.2 Ways To Configure The Network To Allow Users To Talk To The Server By Name

The method we'll use is option no. 3 (dnsmasq), but we'll briefly cover the other two first.

1. Edit the "Hosts" file

- **Edit Hosts file on every computer that'll connect to the server**
  - file associates hostnames with IP addresses
  - most useful on a private network
  - but, not practical for many users

- **Linux / macOS**
  - /etc/hosts
- **Windows**
  - C:\windows\system32\drivers\etc\hosts
- **Format**
  - Typically a line with an IP address, then white space, follow by hostname,e.g.:

```
192.168.1.109       dycom-server
```

        - Must restart system, or network service, to take affect
            - **sudo systemctl restart networking**

2. Configure DNS (Domain Name System)

- **Central list (directory) of names and IP address**
    - Client computers use this to resolve IP addresses to hostnames

- **Methods:**
    - Router
        - If router supports it, can configure names there, and then setup clients to use your router as DNS server
    - Configure your server as a DNS server
        - Very tricky to get right, outside of the bounds of this manual
    - Use dnsmasq (see next)
        - A happy alternative to DNS server, that works like DNS

3. DNSmasq

- A lightweight DHCP and caching DNS server.
- Works like DNS – a names directory - without the complicated configuration
- Not only that, but it performs as dedicated DNS server, with which you can perform internet lookups (see further down)
- You can also enter other entries in your /etc/hosts file as well, e.g:
    - Shorter names for your file server
    - Names for other devices on the network

Steps:

- Install dnsmasq package:
    - **sudo apt install dnsmasq**

- Add name(s) and IP address(es) to server's /etc/hosts file:
    - **sudo nano /etc/hosts**
    - for your server - change the line with the loopback address (127.0.1.1):
        - from: *127.0.1.1* *server_hostname*
        - to: *serverIP* *server_hostname*
            - E.g. *192.168.178.40* *dycom-server*

- Restart the network service
    - **sudo systemctl restart networking**

- Open the port for DNS on the firewall
  - **sudo ufw allow 53/tcp**

- Configure the network router to list the server's IP address as a DNS service to it's DHCP clients:
  - i.e. open router's DHCP configuration page in the web-interface
  - update the Primary DNS (provided to clients) with your server's address, e.g:
    - **Primary DNS: 192.168.178.40**

- Restart client's network connection:
  - Typically, in your client OS, disable first then re-enable your network interface (e.g. wireless, or ethernet adapter connection)
  - Confirm in your client OS (typically in the adapter's interface) that the new DNS service has been populated

- The server's IP address and hostname is now interchangeable, because the name will resolve to an IP address, e.g.:

- Not only that, but your server now functions as a DNS server, and can perform lookups over the internet:
  - **nslookup domain_name_to_lookup your_server_name**
    - e.g.: nslookup linkedin.com dycom-server
    - output:

```
Server:         dycom-server
Address:        192.168.178.40

Non-authorative answer:
Name:           linkedin.com
Address:        108.174.10.10
```

11. Server Status Monitoring with *Cockpit*

- You can monitor your server status via the terminal, but it can be very cumbersome.
- Web-apps offer popular alternative
- "Cockpit"
  - Server Manager to easily monitor your Linux server status via a web browser
  - Can also perform basic administration tasks
- http://cockpit-project.org

- More information: click “try it out” > Ubuntu
- Uses port 9090 by default
- point your web browser to: **https://**ip-address-of-machine**:9090**

11.1 Adding Software with a PPA

- Cockpit is available as a PPA (Personal Package Archive)
  - A common way to add software to your system, that isn’t available from the standard repositories
- A PPA adds repository information from Launchpad.net

Steps:

- Add the PPA repository information to your server’s list of repositories
  - **sudo add-apt –repository ppa:cockpit-project/cockpit**
    - *cockpit-project* is the username
    - *cockpit* is the repository name
  - this retrieves information about keys, and software versions available

- Update the package lists of the server
  - **sudo apt update**

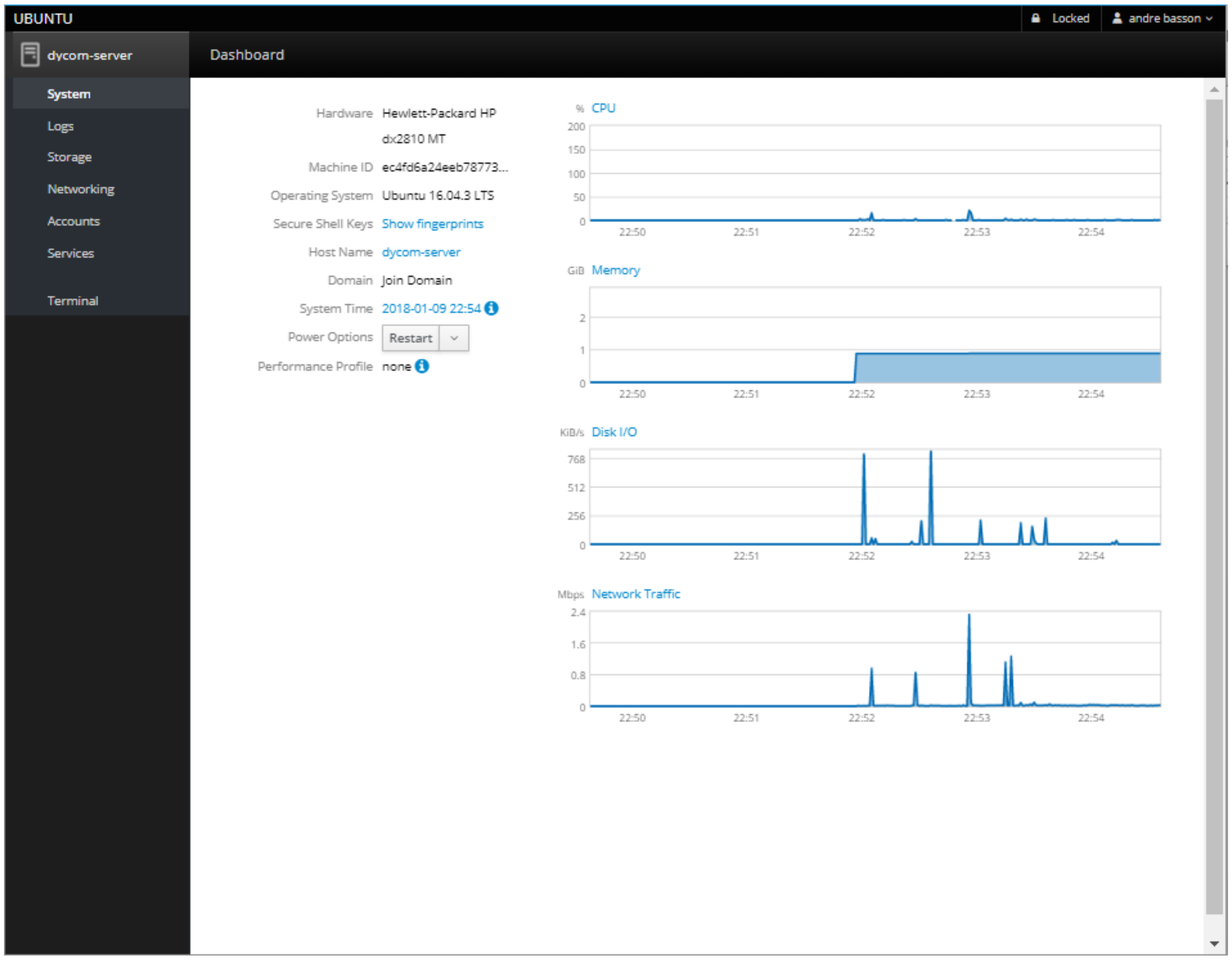
- The server’s package manager (e.g. apt) can now install packages from the repository
  - **sudo apt install cockpit**
  - **sudo apt install cockpit –y** (same, but without y/n prompts)

- Enable software to start up at boot automatically
  - **sudo systemctl enable cockpit**

- Open port 9090 on the firewall (Cockpit uses port 9090 by default)
  - **sudo ufo allow 9090/tcp**

- Point your web browser to: **https://**ip-address-of-machine**:9090**
  - Login with your server credentials

11.2 Cockpit Web Interface Controls



Hardware Hewlett-Packard HP dx2810 MT
Machine ID ec4fd6a24eeb78773...
Operating System Ubuntu 16.04.3 LTS
Secure Shell Keys Show fingerprints
Host Name dycom-server
Domain Join Domain
System Time 2018-01-09 22:54
Power Options Restart
Performance Profile none

- **Main Category Menu (LHS/black)** – i.e. the main categories
- **Dashboard** – where you can add more servers to monitor
- **System**
  - General Categories – i.e. general categories of selected main category
  - Visuals / Summary (right) - detail, summary or charts relating to general category.
- **Services** – to monitor running services (daemons/back ground processes)
- **Containers** – e.g. Docker (plugin to work with Services)
  - enabled in the Accounts category (see below)
- **Logs** – shows system logs
- **Storage** – detail/activity about storage devices
- **Networking** – detail about network transmissions transmitted and received
- **Accounts** – administer user accounts, create new accounts, types etc.
  - Enable Docker System for a user
- **Terminal** – limited command line terminal

12. Selective Blocking of Domains with *dnsmasq*

12.1 Introduction

Where /etc/hosts redirects (resolves) <u>single</u> hostname to IP; *dnsmasq* gives us the ability to redirect any number of hosts, in a domain, to an IP address:

- /etc/dnsmasq.conf – *address directive*: to redirect whole domain names
    - i.e. instead of redirecting (resolving) a single hostname, we can redirect an entire domain to a single IP address using the *address directive*

12.2 To redirect is to block

Sometimes we might want to block certain domains altogether. Blocking is essentially the same as redirection – but a redirection back to your own server. Ad-servers are often seen as a nuisance, and blocking all attempts at redirection away from your server is a must (unless you're intention is to allow it).

Steps:

- open the dnsmasq.conf file:
    - **sudo nano /etc/dnsmasq.conf**
- locate the *address directive* of a domain you would like to redirect back to your server (i.e. block)
    - e.g.: #address=/double-click.net/127.0.0.1
        - *double-click.net* is a common host for banner ads.
        - *127.0.0.1* is the local host address – i.e. to send traffic to local computers

- Uncomment, and change only the IP to your server's IP address, e.g.: 192.168.178.40
- This way you can add more address directives to block domains and redirect them back to your server
- Save and exit dnsmasq.conf
- Restart dnsmasq:
    - **sudo systemctl restart dnsmasq**
- Confirm the redirection-to-IP works with:
    - **ping double-click.net**

Depending on you BASH script fluency; more information on how to block ad servers with dnsmasq is available at https://goo.gl/SDkhzv.

12.3 Things to Be Aware Of

- Blocking large domains can have unintended effects
    - E.g. if you want to block www.google.com so that your users cannot access it, many websites might not work correctly as they often access a google API to perform certain web-services.

- DNS block only makes it hard to reach domains – it doesn't block content or IPs.
    - If you want to block IPs, you'll have to block it (if possible) on your router; and there are quite frankly too many public IPs web-services can have.

13. Connecting to a VPN server

(*https://www.debuntu.org/how-to-connect-to-a-cisco-vpn-with-vpnc/*)

For the purposes of this exercise, we're assuming the use of the *IPSEC Cisco VPN* client called *vpnc*, to connect to a remote IPSEC VPN server (or router).

vpnc is a VPN client compatible with **cisco3000 VPN Concentrator** (and any derivative, e.g. fritz!box), which runs in userspace and uses the **tun** kernel module. People who don't want to be bothered may rather use **network-manager-vpnc** or **kvpnc**. Otherwise, if you intend to connect to a Cisco VPN using the command line or a script, follow up.

13.1. Package requirement

There is only one package to install in order to connect to a cisco VPN: **vpnc**.
Let's install it by typing: **sudo apt-get install vpnc**

This will take care of installing all dependencies.

13.2. Configuration and connection

vpnc can either be used interactively or configuration files can be used.

13.2.1. Using a configuration file – this is what we use for the dycom-server (hp-dx2810)

When you try to connect to a cisco VPN by typing:

```
sudo vpnc
```

vpnc will look for the files */etc/vpnc.conf* or */etc/vpnc/default.conf*. If it does not find such files, vpnc will default to the *interactive mode* (see 13.2.2).

However, vpnc can support different configuration files and be called with the name of the file as an argument. For instance, if you create the configuration file */etc/vpnc/myconf.conf*, you will be able to call vpnc like this:

sudo vpnc myconf

or

sudo vpnc myconf.conf

CAUTION: The configuration file has to be in /etc/vpnc/ and it need to have the extension .conf

The syntax of the configuration file need to be as follow:

```
IPSec gateway gateway.to.use
IPSec ID groupname
IPSec secret passwordforgroup
Xauth username myusername
Xauth password mypassword
```

Where equivalents in a .pcf file are:

```
    IPSec gateway -> Host
    IPSec ID -> GroupName
    IPSec secret -> enc_GroupPwd decrypted using http://www.unix-
ag.uni-kl.de/~massar/bin/cisco-decode
    Xauth username and Xauth password are your username and password
```

13.2.2. Using intearactive mode

vpnc enters interactive mode if you call it without any arguments and there is no /etc/vpnc/default.conf or /etc/vpnc.conf.

It will also prompted the user for any argument which was not supplied in the configuration file. Here is the output when vpnc is called that way:

vpnc.conf

```
Enter IPSec gateway address: example.com
Enter IPSec ID for example.com: examplegroup
Enter IPSec secret for examplegroup@example.com:
Enter username for example.com: foobar
Enter password for foobar@example.com:
```

Arguments can be set or overridden by passing them though the command line. Use vpnc -h for more details.

3. Disconnecting from a vpn

Once connected, the client can be disconnected using:

```
sudo vpnc-disconnect
```

4. Checking if any VPN-connection is active

- **nmcli con | grep -i vpn**

OR

- **nmcli con status id your-vpn-connection-name**

5. More on decrypting the Group password

http://www.unix-ag.uni-kl.de/~massar/bin/cisco-decode provides the source code use to decrypt the group password.

Let's take a break....

As a note to myself, this is the point at which I would opt to make a full hard drive image backup. I chose *Redo & Backup* to make a full image. The software can be installed on a bootable LiveUSB, making it easy to backup or restore – to and from - an external drive.

The reason I choose to do a full image backup now is because the next section – the introduction of Desktop Interfaces – can be a tricky beast should you wish to install a different flavour of Desktop Environment than suggested. Or in fact revert the changes already made (a simple command often rewrites configuration files unbeknownst to the executioner).

It is also my experience that unless you know exactly what you're doing, Linux configurations can easily introduce errors that are extremely hard and/or time consuming to fix or undo.

Hence, back up now. You have been warned.

14. Using a Desktop Interface

14.1 Installing a Desktop Environment on Your Server

Propably the two most important **reasons not to install a Desktop Environment**, is not only because it's resource intensive, but also because it comes with a bloated list of packages – and the more the packages, the higher the chance these can compromise your server's security.

Your favourite Desktop Environment (e.g. Ubuntu Unity) can be quite resource heavy - resources which can be put to use by the server instead. Still for many users, the option of only working in the terminal can be quite daunting.

Often it is also much easier to resolve troublesome issues – like hardware driver installations and configurations - in a Desktop Environment. Just recently I had a case of not being able to find a driver for a wireless network adapter – not even from the manufacturer's online support. Ubunty Unity had no issues and installed a generic driver that works to this day!

Installing Ubuntu Unity

You can install the ubuntu-desktop metapackage that will install all the bells and whistles of GUI for you (as it's dependencies and the dependencies of dependencies and so on):

- **`sudo apt install ubuntu-desktop`**

`ubuntu-desktop` will install the following:

- `Window-manager: Compiz`
- `Display-manager: LightDM`
- `Desktop Environment: Unity`

In addition, `ubuntu-desktop` will also configure your server to start the Unity Desktop Environment at boot time.

Booting to Console / Terminal

Because we know Desktop Environments to be a potential security risk, it is a good idea to boot to terminal instead, leaving the option to startup the Desktop Environment when we chose.

To skip the login GUI without using `Ctrl+Alt+F1`, simply do the following:

- sudo nano /etc/default/grub
- Change the line that reads GRUB_CMDLINE_LINUX_DEFAULT="quiet splash" or GRUB_CMDLINE_LINUX_DEFAULT="" to GRUB_CMDLINE_LINUX_DEFAULT="text"
- Uncomment the line which reads #GRUB_TERMINAL=console by removing the leading #
- Press CTRL-x to exit, followed by Y (for 'yes') to save, and ENTER.
- Update /boot/grub/grub.cfg to have your change apply by running sudo update-grub

  If your computer uses *systemd*, you must tell *systemd* (using *systemctl* command) to skip the default login GUI thus:

  ```
  sudo systemctl enable multi-user.target --force
  sudo systemctl set-default multi-user.target
  ```
- Reboot your computer: sudo reboot

Now, the login GUI will never show up.

Start / Stop The Unity Desktop Environment

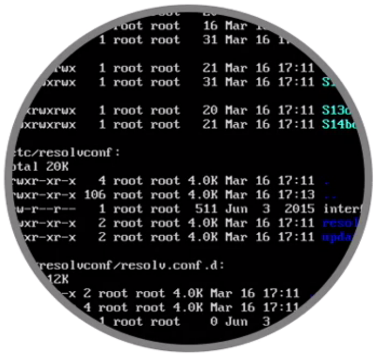
Once you're in the terminal, run `sudo systemctl start lightdm` to start the default desktop. If the Desktop Environment doesn't start, try running `startx`, and the default Desktop Environment should start up.

The Ubuntu Unity Desktop Environment occupies tty7 (terminal no.7), and should you wish to kill the process, do the following:

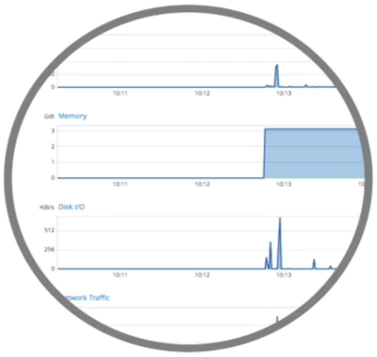
- press Ctrl+Alt+F1 – you'll be taken to terminal 1 (tty1)
- to terminate the desktop, run:

  sudo service lightdm stop
- if you would like to return to the Desktop instead, simply press Ctrl+Alt+F7

14.2 Ways to Interface to the Server Remotely



```
     root root   16 Mar 1
   1 root root   31 Mar 16 1

rwx   1 root root   21 Mar 16 17:11
xrwx   1 root root   31 Mar 16 17:11 S1

xrwxrwx   1 root root   20 Mar 16 17:11 S13c
xrwxrwx   1 root root   21 Mar 16 17:11 S14bc

etc/resolvconf:
otal 20K
rwxr-xr-x   4 root root 4.0K Mar 16 17:11 .
rwxr-xr-x 106 root root 4.0K Mar 16 17:13 ..
w-r--r--   1 root root  511 Jun  3  2015 inter
wxr-xr-x   2 root root 4.0K Mar 16 17:11 resol
xr-xr-x   2 root root 4.0K Mar 16 17:11 upda

resolvconf/resolv.conf.d:
12K
-x 2 root root 4.0K Mar 16 17:11
 4 root root 4.0K Mar 16 17:11
1 root root    0 Jun  3
```



Terminal Session — Web Browser from Client — Desktop Environment

In the previous section we've covered how to install a Desktop Environment to run locally on the server. Now we take a look at running a lighter desktop environment remotely, via VNC.

14.2.1 Many Desktop Environments

- Unity
- GNOME
- KDE
- LXDE
- Xfce
- And more

14.2.2 Xfce

- Small
- Lightweight
- VNC server (Virtual Network Computing)
  - great for connecting to any server remotely, using a graphical interface
- See "Linux: Desktops and Remote Access"

14.2.3 Installing Xfce Desktop on the Server

- **sudo apt install xfce4 xfce4-goodies**
    - where xfce4 is the desktop package, and xfce4-goodies useful package for xfce

- at this point you can start up the desktop (aka x-session) through the terminal if you want via:
    - **startx** or **sudo startx**

- **but** we'll rather start up a desktop session on the server, which we share via a VNC Server application called *TightVNC*
    - first we install the VNC server, then share the session
    - instructions next

14.2.4 Installing a VNC Server and Share a Desktop Session

- Virtual Network Computing (VNC)
- Connects to a graphical session of your installed Desktop Environment on the server, shares it, allowing a client to connect remotely
- Aka "screen sharing"
- Persistent session
    - Can have more than one session active
    - Each session runs on a different virtual display
        - Each virtual display is made available on a different port
- Clients available for all platforms and some browsers
    - macOS and Linux has built-in support for VNC
    - Windows you'll have to install a client VNC viewer

Steps:

- Install VNC server called *tightvncserver* on your server:
    - **sudo apt install tightvncserver**

- Run vncserver to go through initial setup: **vncserver**
    - You'll be prompted to enter a password to access the desktop sessions on the server
        - Password is separate from your user account; max 8 characters
    - You'll be prompted for View-Only password, for if a user only wants to view without interaction – NOT VERY HELPFUL - select 'no'

- The VNC server will now start up – a default startup script is created in the *xstartup* file in your directory under */home/your_username/.vnc/* - but the server doesn't know anything about the desktop environment yet. *xstartup* stores the commands to perform when vncserver starts up.

- We configure the *xstartup* file as follows:
  - **sudo nano ~/.vnc/xstartup** #open xstartup file in your home folder
  - cat ~/.vnc/xstartup

```
#!/bin/sh

xrdb $HOME/.Xresources
xsetroot -solid grey

#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP
Desktop" &
#x-window-manager &
# Fix to make GNOME work
export XKL_XMODMAP_DISABLE=1
/etc/X11/Xsession
```

  - The first command in the file, *xrdb $HOME/.Xresources*, tells VNC's GUI framework to read the server user's .Xresources file. .Xresources is where a user can make changes to certain settings of the graphical desktop, like terminal colors, cursor themes, and font rendering.
  - After the line starting with "xsetroot", include the following line:
    - **startxfce4 &** #the command to start up the desktop environment, followed by an ampersand

      i.e.:

```
...
xsetroot -solid grey
startxfce4 &
...
```

    - save and exit

- When we connect to a session, the window will be very small, but we can specify the resolution (or in terms of the session, the "session geometry") by creating a *.vncrc* file in your home folder
  - **sudo nano ~/.vncrc**
  - add the following line:
    - **geometry = "1360x768"**
- Stop the VNC server currently running (it started without any settings), and then start it back up:
  - **vncserver –kill :1**
    - When VNC is first set up, it launches a default server instance on port 5901. This port is called a virtual display port, and is referred to by VNC as *:1*.
  - **vncserver**
    - This starts up a session, delineated by a virtual display port '1' (or *:1*

to be precise) , with you as the user,

- i.e.: output is: *New 'X' desktop is dycom-server:1*

  - You can now connect to this virtual display '1 ', from your terminal or VNC client.
  - If you started a 2nd session, it would use virtual display '2', etc.
  - The session at virtual display 1 will be available at port 5901, the session at virtual display 2 will be available at port 5902, etc.

- To allow the session through the firewall so that clients can connect:
  - **sudo ufw allow 5901/tcp**

14.2.5 VNC Clients to Connect to the Shared Session

Connecting to VNC sessions on the server from a client:

- **macOS Finder / Ubunty Unity**
  - Connect to Server...
  - vnc://server-address:5901
- **Windows**
  - Applications RealVNC, TightVNC, and TigerVNC
    - Add address and port number in application

- **Smartphone and Tablet clients available as well**
- **Google Chrome Extension**
  - Add the "VNC Viewer for Google Chrome" extension available from chrome webstore, http://goo.gl/2U4aJf, or at www.realVNC.com
  - Open the link saved to the Google Startup screen
    - Enter IP address > click connect > enter password > click connect
    - For a private network, we can ignore the *unencrypted* message warning

  - The viewport should now connect to the desktop session > click *full screen* icon
  - Select *Default configuration*, ignore the error.
  - Unpin the toolbar at the bottom, so that you can see the dock at the bottom of the screen. The toolbar will slide down and become hidden below the dock.
  - From here you can chose to either use the GUI tools provided
  - or click on the terminal icon to run any commands as you've done up till now - even install software! Installed software will become available in the *Applications* menu at the top.

- When you want exit the session, slide your mouse to the bottom of the screen until the toolbar re-appears – click disconnect.

15. Accessing Resources Remotely

- Be aware that opening access to your server involves risks
- Keep a security mindset, be thoughtful, and the risks are minimized

15.1 Overview

Remote Network Services

- External Access with Dynamic DNS
  - giving the server a name on the internet
- Remote SSH access with a key (instead of password)
- Share and Sync files with Nextcloud

Opening Ports on a Router

- **CAUTION**:
  - Opened ports on a router/server are the central means by which a hacker attempts to circumvent a system's security measures, or gain access to a system.
  - Hence, do not open ports on your router to the internet UNLESS you intend to do so!
- Forwarding a request from the outside to your internal network (e.g. server, or other host) requires opening a port on your router
  - The router then forwards the request to a specific IP in the internal network, instead of blocking or trying to respond to the request itself
- Know that not all services are ideal for remote access
  - Samba ports (139/445) can be quite dangerous to expose to the internet
    - **i.e. Do not enable port forwarding of ports 139 & 445 on your router, to the internet**

  - Both SMB & CIFS protocols do NOT implement data encryption technologies – so anyone using a network packet sniffer will be able to monitor file transfers

- Be aware of your ISPs policies
  - They may not allow public facing services on home internet connections
  - Usually they will have policies on preventing webservers on hosting websites on

ports 80 and 443, or running a mail service

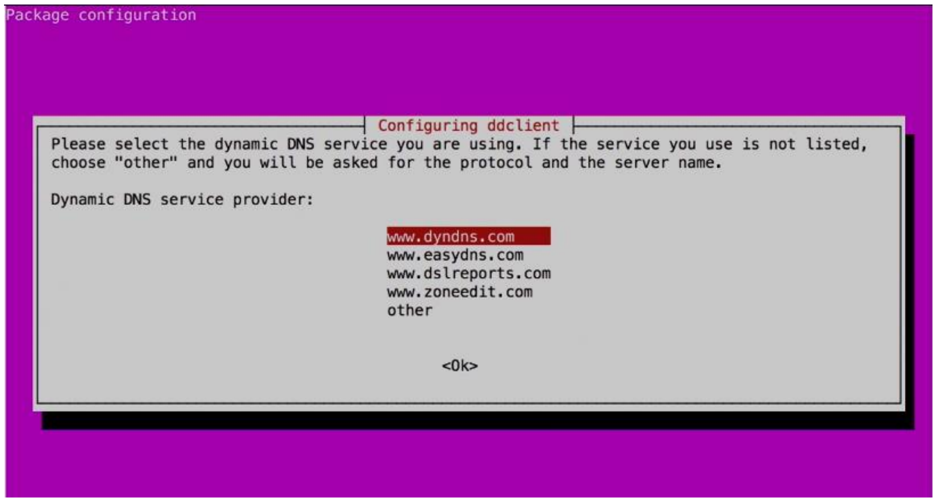
15.2 Remote Network Services

Dynamic DNS (DynDNS or DDNS) for Easy Internet Access to Your Server

- Many home connections have a dynamic IP address
- (Q) If the address changes, how do you connect to your services remotely?
- (A) A Dynamic DNS Service Provider provides the possibility of accessing the services on your server via a fixed domain name, instead of an IP address.
    - It does this by **resolving the domain name** to the actual IP address of the router your host is connected to.
- (Q) How does the DDNS provider know what the current IP address of the router is?
    - the server relays its current (public) IP address to the DDNS provider, with the use of a **DDNS Update Client**

- There are many dynamic DNS providers: No-IP (https://www.noip.com), DynDNS, (neither are free!), etc.
- Free accounts are available, but often require periodic logins (to keep the account active)
    - See *Lynda Building an Ubuntu Server*, video 04_02 (1m17s - 5m16s), for setting up a
      No-IP dynamic DNS service, and setting up a *ddclient* package on your server to connect to your dynamic DNS provider, i.e.:
        - 1st setup a no-ip account at www.no-ip.com
        - Setup a dynamic update client on your router or server:
            - The client checks your public IP and see if it has changed; then updates your DNS record for your sub-domain (i.e. domain to IP translation)
            - Ways to update the DDNS data:
                - Routers – many have a function that can update the DDNS data
                - www.no-ip offers their own client too
                - but to keep things general, we'll install the *ddclient* package on our server instead:
                    - **sudo apt install ddclient**

        - A configuration menu appears directly after installation:



```
Package configuration

                    ┤ Configuring ddclient ├
 Please select the dynamic DNS service you are using. If the service you use is not listed,
 choose "other" and you will be asked for the protocol and the server name.

 Dynamic DNS service provider:

                              www.dyndns.com
                              www.easydns.com
                              www.dslreports.com
                              www.zoneedit.com
                              other

                                   <Ok>
```

- *ddclient* can be configured to connect to many ddns providers:
  - Click *Other*
  - Then, for *Dynamic DNS server*: **dynupdate.no-ip.com**
  - DDNS protocol: **dyndns2**
  - Enter username & passwd of DDNS account: *****
  - Network Interface used for DDNS: **leave blank (conf'd later)**
  - DynDNS fully qualified domain name: dycomserver.ddns.net

- Manualy configure the Network Interface used for DDNS (the one we didn't in the package configuration application above):
  - **sudo nano /etc/ddclient.conf**
  - add the line in the space provided as seen below:

```
# Configuration file for ddclient generated by
debconf
#
# /etc/ddclient.conf

protocol=dyndns2
use=web
server=dynupdate.no-ip.com
login=skygode@gmail.com
password='***your-passwd-here***'
dycomserver.ddns.net
```

  - use=web will instruct the software to use a webservice to resolve the external IP address

- Configure the DDNS client to run as a daemon (i.e. in background on its own):
  - **sudo nano /etc/default/ddclient**
  - make the following updates:
    - ***run_daemon="true"***
    - *run_ipup="false"*

                - *daemon_interval="300"*
                    - i.e. to update the DDNS provider every 300 seconds
        - Restart the ddclient
            - **sudo systemctl restart ddclient**

        - Confirm that the ddclient daemon is running, with either:
            - **sudo systemctl status ddclient**
            - **sudo /etc/init.d/ddclient status**

        - You can view the connection (and or update detail) in your www.no-ip.com *Hostnames* webpage.

        - NOTE: the DDNS update client does **not** transmit your details securely; you can do so by using SSH. See more at https://help.ubuntu.com/community/DynamicDNS

    - The bonus with these Dynamic DNS providers, is that you get to pick a domain name of your choice.

- **NOTE TO SELF:**
    - Some router manufacturers supply their own dynamic DNS services. Currently you already have such a dynamic domain name configured for your home router, having used the manufacturer's online service (see the router's MyFRITZ!Account page in the router's web interface, to find the domain name and port number). Your router is thus already accessible from the internet, and you can continue with the following configurations.

    - If you also have a VPN service running on your home router - with which a remote host can connect via a secure tunnel DIRECTLY to your home network shares - port forwarding **should not be required** as the VPN connection has effectively bridged the remote network with the local network. **HOWEVER**: should the VPN connection be dropped or time-out; you will have no means to reconnect (e.g. SSH) to the remote host if you have no dynamic-DNS-assigned address setup for that host. Thus, it is good idea to atleast setup one (e.g. SSH) port with access to the remote host; even if just to re-open the VPN connection.

Troubleshooting

- Should you not be able to connect remotely via your registered domain name; it might be that your DDNS Update Client has not updated the DDNS service provider with the latest IP address of your host (server). Try the following:

- Do a lookup for your domain name
  - `nslookup your.domain.name`
    - if you see a reply, and with an IP address, it means that the DDNS Provider is resolving the domain name, to an IP address – this does not mean it's the actual/correct IP address – it's just an indication that the provider is resolving the domain name.

- try forcing the DDNS Update Client to update the dynamic DNS provider with your latest (public) IP address
  - `sudo ddclient –force`

- Still not working? Uninstall the DDNS Update Client, then reinstall as before
  - `sudo apt-get remove --purge ddclient`
  - `sudo apt-get install ddclient`
    - now following configurations as before
  - restart the update client with either:
    - `sudo systemctl restart ddclient`
    - `sudo /etc/init.d/ddclient restart`

- Still not working?
  - Is the firewall active, and does it block the port used?

- See more at https://help.ubuntu.com/community/DynamicDNS

Opening (Forwarding) Ports on the Router Firewall

So far you have dynamic domain name configured for your **network** connected device (i.e. router), with the help from a DynDNS (DDNS) provider, as explained in the previous section. What follows pertains to allowing access through the router's firewall, and to the host on the private side of the firewall:

- Any requests received by the router will have to be forwarded to your internal network (e.g. a server on the private side of the router).

- We enable this forwarding by opening up ports on the router's firewall
  - aka *Virtual Server* or *Port Forwarding* service, depending on your router manufacturer

- Some routers have a *port triggering* option
  - Not the same a port forwarding
  - Allows hosts from internal network, to start a connection to the outside

    - If you intend to connect TO your server from the outside, port triggering will not be applicable.

- Configure port forwarding
    - Open the Virtual Server / Port Forward / Port Sharing configuration page of your router.
    - Set the IP address of the host (e.g. server) you want accessible from the internet.
    - Choose the allowed protocol (e.g. TCP) to signify the type of allowed traffic through the port.
    - Set the external port and the internal ports (e.g. traffic from (SSH) port 22 on the external host, to port 22 on the internal host) – i.e. complete the port forward.
    - Supply a descriptive name – e.g. *port 22 to home-server*
    - Click *OK* or *ADD*

- You'll return to the router's port forwarding/sharing config page to open ports for other applications and types of traffic as necessary.

- You should also open the same port on the host you want to connect to, e.g.:
    - **sudo ufw allow 22/tcp**

16 Secure Remote Access with SSH (Secure SHell)

16.0 What we'll do

- Understand how SSH works
- Generate a cryptographic key pair to be used in SSH connections
- Disable password authentication
- More secure than passwords
    - Passwords, with some effort, is much easier to guess and therefore less secure

16.1 Introduction

SSH is a secure protocol used as the primary means of connecting to Linux servers remotely. It provides a text-based (terminal) interface by spawning a remote shell. After connecting, all commands you type in your local terminal are sent to the remote server and executed there.

16.2 SSH Overview

The most common way of connecting to a remote Linux server is through SSH. SSH stands for Secure Shell and provides a safe and secure way of executing commands, making changes, and configuring services remotely. When you connect through SSH, you log in using an account that exists on the remote server.

How SSH Works

When you connect through SSH, you will be dropped into a shell session, which is a text-based interface where you can interact with your server. For the duration of your SSH session, any commands that you type into your local terminal are sent through an encrypted SSH tunnel and executed on your server.

The SSH connection is implemented using a client-server model. This means that for an SSH connection to be established, the remote machine must be running a piece of software called an SSH daemon. This software listens for connections on a specific network port (default 22 for SSH), authenticates connection requests, and spawns the appropriate environment if the user provides the correct credentials.

The user's computer must have an SSH client. This is a piece of software that knows how to communicate using the SSH protocol and can be given information about the remote host to connect to, the username to use, and the credentials that should be passed to authenticate. The client can also specify certain details about the connection type they would like to establish.

How SSH Authenticates Users

Clients generally authenticate either using passwords (less secure and not recommended) or SSH keys, which are very secure.

Password VS Key-Based Authentication

Password logins are encrypted and are easy to understand for new users. However, automated bots and malicious users will often repeatedly try to authenticate to accounts that allow password-based logins, which can lead to security compromises. For this reason, we recommend always setting up SSH key-based authentication for most configurations.

SSH keys are a matching set of cryptographic keys which can be used for authentication.

Each set contains a public and a private key. The public key can be shared freely without concern, while the private key must be vigilantly guarded and never exposed to anyone.

Key Placement

To authenticate using SSH keys, a user (client) machine must have an SSH key pair on their local computer. On the remote server, the public key must be copied to a file within the user's home directory at `~/.ssh/authorized_keys`. This file contains a list of public keys, one-per-line, that are authorized to log into this account.

Authentication (short version)

The `known_hosts` file lets the client authenticate the server, to check that it isn't connecting to an impersonator. This file will be located on the client (`~/.ssh/known_hosts`), and it will contain a *host public key* from the server (e.g. `/etc/ssh_host_rsa_key.pub`) from which the server can be authenticated with. Note that known_hosts can contain many host keys, and in fact should contain one for every host (server) it connects to.

The `authorized_keys` file lets the server authenticate the user – therefore this file will be located on the server, and will contain a public key from the SSH keypair created earlier. This public key can now be used by the server to authenticate the user. As with `known_hosts`, `authorized_keys` can contain many public keys, and in fact should contain one for every client connecting to it.

Authentication (midlength version)

When a client connects to the host, wishing to use SSH key authentication, it will inform the server of this intent and will tell the server which public key to use. The server sends its public key to the client , and proves (thanks to public-key cryptography) to the client that it knows the associated private key. This authenticates the server: if this part of the protocol is successful, the client knows that the server is who it claims it is.

The server then checks its `authorized_keys` file for the public key, generate a random string and encrypts it using the public key. This encrypted message can only be decrypted with the associated private key (located on the client). The server will send this encrypted message to the client to test whether they actually have the associated private key.

Upon receipt of this message, the client will decrypt it using the private key and combine the random string that is revealed with a previously negotiated session ID. It then generates an MD5 hash of this value and transmits it back to the server. The server already had the original message and the session ID, so it can compare an MD5 hash generated by those values and determine that the client must have the private key.

Now that you know how SSH works, we can begin to discuss some examples to demonstrate different ways of working with SSH

Authentication (LONG version)

Server authentication

One of the first things that happens when the SSH connection is being established is that the server sends its public key to the client, and proves (thanks to public-key cryptography) to the client that it knows the associated private key. This authenticates the server: if this part of the protocol is successful, the client knows that the server is who it claims it is.

The client may check that the server is a known one, and not some rogue server trying to pass off as the right one. SSH provides only a simple mechanism to verify the server's legitimacy: it remembers servers you've already connected to, in the `~/.ssh/known_hosts` file on the client machine (there's also a system-wide file `/etc/ssh/known_hosts`). The first time you connect to a server, you need to check by some other means that the public key presented by the server is really the public key of the server you wanted to connect to. If you have the public key of the server you're about to connect to, you can add it to `~/.ssh/known_hosts` on the client manually.
By the way, `known_hosts` can contain any type of public key supported by the SSH implementation, not just DSA (also RSA and ECDSA).
Authenticating the server has to be done before you send any confidential data to it. In particular, if the user authentication involves a password, the password must not be sent to an unauthenticated server.

User authentication

The server only lets a remote user log in if that user can prove that they have the right to access that account. Depending on the server's configuration and the user's choice, the user may present one of several forms of credentials (the list below is not exhaustive).

- The user may present the password for the account that he is trying to log into; the server then verifies that the password is correct.
- The user may present a public key and prove that he possesses the private key associated with that public key. This is exactly the same method that is used to authenticate the server, but now the user is trying to prove its identity and the server is verifying it. The login attempt is accepted if the user proves that he knows the private

key and the public key is in the account's authorization list (`~/.ssh/authorized_keys` on the server).

- Another type of method involves delegating part of the work of authenticating the user to the client machine. This happens in controlled environments such as enterprises, when many machines share the same accounts. The server authenticates the client machine by the same mechanism that is used the other way round, then relies on the client to authenticate the user.

16.2 Generating and Working with SSH Keys

This section will cover how to generate SSH keys on a client machine and distribute the public key to servers where they should be used. This is a good section to start with if you have not previously generated keys due to the increased security that it allows for future connections.

Generating an SSH Key Pair

Generating a new SSH public and private key pair on your local computer is the first step towards authenticating with a remote server without a password. Unless there is a good reason not to, you should always authenticate using SSH keys.

A number of cryptographic algorithms can be used to generate SSH keys, including RSA, DSA, and ECDSA. RSA keys are generally preferred and are the default key type.
To generate an RSA key pair on your local computer, type:

```
       ssh-keygen
or     ssh-keygen –t rsa
```

```
Output:
       Generating public/private rsa key pair.
       Enter file in which to save the key (/home/demo/.ssh/id_rsa):
```

This prompt allows you to choose the location to store your RSA private key. Press ENTER to leave this as the default, which will store them in the .ssh hidden directory in your user's home directory. Leaving the default location selected will allow your SSH client to find the keys automatically.

```
       Enter passphrase (empty for no passphrase):
       Enter same passphrase again:
```

The next prompt allows you to enter a passphrase of an arbitrary length to secure your private key. By default, you will have to enter any passphrase you set here every time you use the private key, as an additional security measure. Feel free to press ENTER to leave this blank if you do not want a passphrase. Keep in mind though that this will allow anyone who gains control of your private key to login to your servers.
If you choose to enter a passphrase, nothing will be displayed as you type. This is a security precaution.

```
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
8c:e9:7c:fa:bf:c4:e5:9c:c9:b8:60:1f:fe:1c:d3:8a root@here
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|       +         |
|      o S   .    |
|     o   . * +   |
|      o + = O .  |
|       + = = +   |
|      ....Eo+    |
+----[SHA256]-----+
```

This procedure has generated an RSA SSH key pair, located in the `.ssh` hidden directory within your user's home directory, e.g. `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`:

- `~/.ssh/id_rsa`: The private key. DO NOT SHARE THIS FILE!
- `~/.ssh/id_rsa.pub`: The associated public key. This can be shared freely without consequence.
- `SHA256`: The hashing algorithm used to create a hash (shortened string value) of a key or key pair.

Displaying the SSH Key Fingerprint

Each SSH key pair share a single cryptographic "fingerprint" which can be used to uniquely identify the keys. A Key fingerprint is a unique hash of a key, and is generally used to identify/confirm a longer public key of a host you intend to connect to.

To find out the fingerprint of an SSH key, type:

```
      ssh-keygen -l
```

output:

```
      Enter file in which the key is (/root/.ssh/id_rsa):
```

You can press ENTER if that is the correct location of the key, else enter the revised location. You will be given a string which contains the bit-length of the key, the fingerprint, and account and host it was created for, and the algorithm used:

```
      4096 8e:c4:82:47:87:c2:26:4b:68:ff:96:1a:39:62:9e:4e  demo@test
(RSA)
```

You can also include the (file) name of the key, by including the –f option:

- **ssh-keygen -lf /path/to/ssh/key**
  - where -l means "list" instead of create a new key, -f means "filename"

Concrete example (if you use an RSA public key):

- ssh-keygen -lf ~/.ssh/id_rsa.pub

```
output:
2048 SHA256: ……. (RSA)
```

Bonus info:

- ssh-keygen -lf also works on known_hosts and authorized_keys files.

Copying your Public SSH Key to a Server with SSH-Copy-ID

To copy your public key to a server, allowing you to authenticate without a password, a number of approaches can be taken.

If you currently have password-based SSH access configured to your server, and you have the `ssh-copy-id` utility installed, this is a simple process. The `ssh-copy-id` tool is included in many Linux distributions' OpenSSH packages, so it very likely may be installed by default.

If you have this option, you can easily transfer your public key by typing:

```
ssh-copy-id username@remote_host
```

This will prompt you for the user account's password on the remote system:

```
    The authenticity of host '111.111.11.111 (111.111.11.111)' can't be
    established.
    ECDSA key fingerprint is
    fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
    Are you sure you want to continue connecting (yes/no)? yes
    /usr/bin/ssh-copy-id: INFO: attempting to log in with the new
    key(s), to filter out any that are already installed
    /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if
    you are prompted now it is to install the new keys
    demo@111.111.11.111's password:
```

After typing in the password, the contents of your `~/.ssh/id_rsa.pub` key will be appended to the end of the user account's `~/.ssh/authorized_keys` file:

```
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'demo@111.111.11.111'"
and check to make sure that only the key(s) you wanted were added.
```

You can now log into that account without a password:

```
ssh username@remote_host
```

Copying your Public SSH Key to a Server Without SSH-Copy-ID

If you do not have the `ssh-copy-id` utility available, but still have password-based SSH access to the remote server, you can copy the contents of your public key in a different way.

You can output the contents of the key and pipe it into the `ssh` command. On the remote side, you can ensure that the `~/.ssh` directory exists, and then append the piped contents into the `~/.ssh/authorized_keys` file:

```
cat ~/.ssh/id_rsa.pub | ssh username@remote_host "mkdir -p ~/.ssh && cat
>> ~/.ssh/authorized_keys"
```

You will be asked to supply the password for the remote account:

```
    The authenticity of host '111.111.11.111 (111.111.11.111)' can't be
    established.
    ECDSA key fingerprint is
```

```
fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
 Are you sure you want to continue connecting (yes/no)? yes
 demo@111.111.11.111's password:
```

After entering the password, your key will be copied, allowing you to log in without a password:

```
 ssh username@remote_IP_host
```

Copying your Public SSH Key to a Server Manually

If you do not have password-based SSH access available, you will have to add your public key to the remote server manually.

On your local machine, you can find the contents of your public key file by typing:

```
cat ~/.ssh/id_rsa.pub
```

output:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCqql6MzstZYh1TmWWv11q5O3pISj2ZFl9HgH1J
LknLLx44+tXfJ7mIrKNxOOwxIxvcBF8PXSYvobFYEZjGIVCEAjrUzLiIxbyCoxVyle7Q
+bqgZ8SeeM8wzytsY+dVGcBxF6N4JS+zVk5eMcV385gG3Y6ON3EG112n6d+SMXY0OEBI
cO6x+PnUSGHrSgpBgX7Ks1r7xqFa7heJLLt2wWwkARptX7udSq05paBhcpB0pHtA1Rfz
3K2B+ZVIpSDfki9UVKzT8JUmwW6NNzSgxUfQHGwnW7kj4jp4AT0VZk3ADw497M2G/12N
0PPB5CnhHf7ovgy6nL1ikrygTKRFmNZISvAcywB9GVqNAVE+ZHDSCuURNsAInVzgYo9x
gJDW8wUw2o8U77+xiFxgI5QSZX3Iq7YLMgeksaO4rBJEa54k8m5wEiEE1nUhLuJ0X/vh
2xPff6SQ1BL/zkOhvJCACK6Vb15mDOeCSq54Cr7kvS46itMosi/uS66+PujOO+xt/2FW
Yepz6ZlN70bRly57Q06J+ZJoc9FfBCbCyYH7U/ASsmY095ywPsBo1XQ9PqhnN1/YOorJ
068foQDNVpm146mUpILVxmq41Cj55YKHEazXGsdBIbXWhcrRf4G2fJLRcGUr9q8/lERo
9oxRm5JFX6TCmj6kmiFqv+Ow9gI0x8GvaQ== demo@test
```

You can copy this value, and manually paste it into the appropriate location on the remote server. You will have to log into the remote server through other means.

On the remote server, create the ~/.ssh directory if it does not already exist:

```
mkdir -p ~/.ssh
```

Afterwards, you can create or append the `~/.ssh/authorized_keys` file by typing:

```
    echo public_key_string >> ~/.ssh/authorized_keys
OR  cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

You should now be able to log into the remote server without a password.

16.3 Basic Connection Instructions

The following section will cover some of the basics about how to connect to a server with SSH.

Connecting to a Remote Server

To connect to a remote server and open a shell session there, you can use the `ssh` command.

The simplest form assumes that your username on your local machine is the same as that on the remote server. If this is true, you can connect using:

```
ssh remote_host -i /path/to/private_key
```

…where: ssh is the ssh client application
-i refers to the identity file (private key) – the last entry in the command

If your username is different on the remoter server, you need to pass the remote user's name like this:

```
ssh username@remote_host -i /path/to/private_key
```

Your first time connecting to a new host, you will see a message that looks like this:

```
The authenticity of host '111.111.11.111 (111.111.11.111)' can't be
established.
  ECDSA key fingerprint is
  fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
  Are you sure you want to continue connecting (yes/no)? yes
```

Type "yes" to accept the authenticity of the remote host.

If you are using password authentication, you will be prompted for the password for the remote account here. If you are using SSH keys, you will be prompted for your private key's passphrase if one is set, otherwise you will be logged in automatically.

16.4 Disable password authentication (/etc/ssh/sshd_config)

- **sudo nano /etc/ssh/sshd_config**
    - locate, and update the following line:
        - from: #PasswordAuthentication yes
        - to: **PasswordAuthentication no**
        - save & exit
    - we should now not be able to log in with a password anymore

- restart the SSH service:
    - **sudo systemctl restart sshd**

- connect (log in) from client machine to remote server:
    - **ssh andre@dycomserver.ddns.net –i id_rsa [–p <custom port.no>]**
        - where:
            - *andre* is the username
            - *dycomserver.ddns.net* is your domain name
            - The **–i** option precedes the path to the private key
                - In this case the folder the ssh command is executed from is the current local folder, file id_rsa.
            - **-p <custom port.no>** is an optional entry, to be used when SSH is configured to use a port other than port 22
                - E.g. `-p 2200`
    - No password will be required – we're now using the private key to log in.

16.5 Helpful SSH Troubleshooting and Commands

Verbose output to console if you have troubles with SSH connection:

- **ssh –v username@server-ip-or-domain –i /path/to/RSAprivateKey**

If you want to log to console the connection details from the server:

- Restart SSHD (ssh daemon) with verbose logging (if SSH configured as SSHD daemon)
    - **sudo systemctl stop sshd** #stop ssh daemon first

- **/usr/sbin/sshd –d** #manually start sshd with –d option

Permissions of ~/.ssh/ folder and (key) file

Typically you want:

- the .ssh directory permissions to be 700 (drwx------)
    - **sudo chmod 700 ~/.ssh**
- the public key (e.g. id_rsa.pub) to be 644 (-rw-r--r--)
    - **sudo chmod 644 ~/.ssh/id_rsa.pub**
- your private key (e.g. id_rsa) should be 600 (-rw-------)
- authorized_keys file needs 644 permissions (-rw-r--r--)
- lastly, your home directory should not be writeable by the group or others (at most 755 (drwxr-xr-x)).

16.6 SSH to host over VPN

If you had connected a remote host to your local network via VPN, you should consider that your IP routing tables on your local host might now have changed. Routing tables is a topic for another discussion. But hopefully no manual amendments to the routing tables will be required.
Another thing to consider is that the VPN server (and local network) might have a different subnet configured from the remote host. All in all, this can be a tricky scenario to figure out.

But in general, you should be able to SSH into the remote host using its **private IP**, as assigned by the VPN server (or router). You will likely **not** be able to use the host's DDNS provided domain name (if one is created), even if a ping reveals connectivity to that domain name; recall, that the ping would just relay packets via the DDNS service provider.

To connect via SSH over a VPN connection, the use of the private key still applies, i.e.:

- ssh USER@VPN-ASSIGNED-IP –i /path/to/file-with-private-key

e.g.:

- **ssh andre@192.168.178.201 –i id_rsa**
    - where: *andre* is the linux user
      192.168.178.201 is the IP assigned by the VPN server/router
      -i is the option to list the private key

id_rsa is the path to the private key (file) – here in the current location

17 Sceduling with CronTab **(Cron Table)**

17.1 Introduction

Cron is a system daemon used to execute desired tasks (in the background) at designated times.

A crontab file is a simple text file containing a list of commands meant to be run at specified times. It is edited using the crontab command. The commands in the crontab file (and their run times) are checked by the cron daemon, which executes them in the system background.

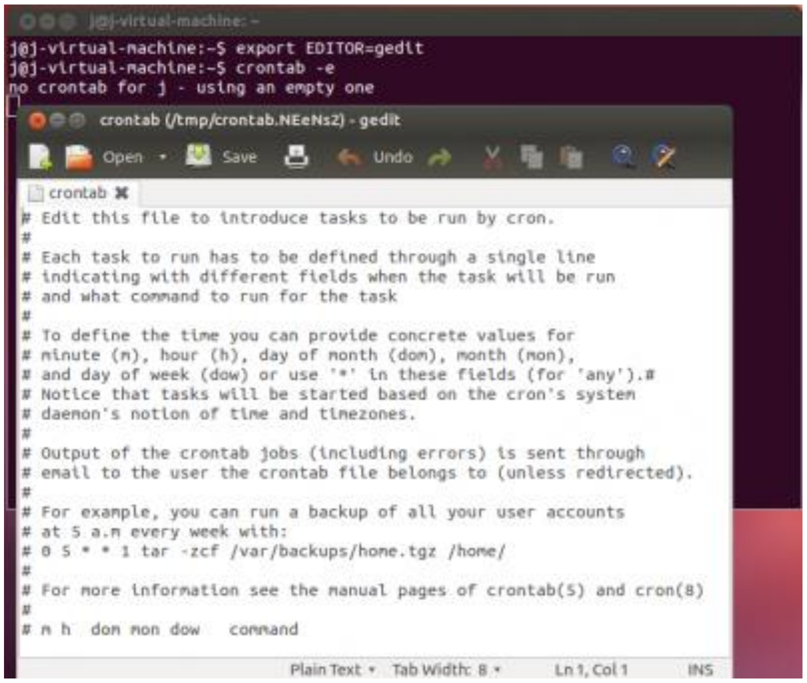
Each user (including root) has a crontab file. The cron daemon checks a user's crontab file regardless of whether the user is actually logged into the system or not.

To display the on-line help for crontab enter: **`man crontab`**

For a full description see https://help.ubuntu.com/community/CronHowto
For a startup tutorial see http://www.ubuntututorials.com/use-crontab-ubuntu/

For more detail information, you can run:

```
man crontab
```



```
j@j-virtual-machine:~$ export EDITOR=gedit
j@j-virtual-machine:~$ crontab -e
no crontab for j - using an empty one
```

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

17.2 How to Use crontab

To use cron for tasks meant to run only for your *user* profile, add entries to your own user's crontab file. To edit the crontab file enter:

- `crontab -e`

Edit the crontab using the format described in the next sections. Save your changes. (Exiting without saving will leave your crontab unchanged.) To display the on-line help describing the format of the crontab file enter:

- `man 5 crontab`

Commands that normally run with administrative privileges (i.e. they are generally run using sudo) should be added to the *root* crontab. To edit the root crontab enter:

- `sudo crontab -e`

Open crontab with your favorite editor. (By default it's vi or will let you choose one from list)

- `export EDITOR=gedit`
- `crontab -e`

You can append "export EDITOR=gedit" to your `~/.bashrc` file if necessary.

To list crontab content:

- `crontab -l`

To remove all your cron jobs:

- `crontab -r`

To view current running cron jobs in a useful and readable format:
(from http://manpages.ubuntu.com/manpages/precise/man1/ps.1.html)

- `ps -afx -o pid,sid,cmd | grep "( |/)cron( -f)?$"`

  where:
  - -fax indicates to print the process tree according to:
    - **f** (to include full format listing, e.g. additional columns, command arguments, etc.)
    - **a** (optional: print all processes)
    - **x** (optional: for BSD-like linux flavours)
  - -o indicates the output format by which to print to std output, as given by the comma-seperated-list (or space-seperated list) of format specifiers:
    - **pid** (process ID;)
    - **sid** (session ID; alias: sess, session)
    - **cmd** (command; alias: args, command)

They will appear in the first lines, something like this:

```
1108  1108 cron
4288  1108 \_ CRON
4289  4289    \_ /bin/sh -c /path/to/my/crontab/script1.sh
4290  4289        \_ /bin/bash /path/to/my/crontab/script1.sh
4295  4289            \_ /usr/bin/wget LINK
```

First column is PID, second is Session ID and third is the command started by cron. You can kill all the processes related to a specific cron task using the Session ID, so in the example above you should kill Session ID 4289:

- `pkill -s 4289`

17.3 Crontab syntax

```
*    *    *    *    *  command to be executed
┬    ┬    ┬    ┬    ┬
│    │    │    │    │
│    │    │    │    │
│    │    │    │    └───── day of week (0 - 6) (0 is Sunday, or use names)
│    │    │    └────────── month (1 - 12)
│    │    └─────────────── day of month (1 - 31)
│    └──────────────────── hour (0 - 23)
└───────────────────────── min (0 - 59)
```

Notes:

- Comma-separated values can be used to run more than one instance of a particular command within a time period. Dash-separated values can be used to run a command continuously
- You can use repeat pattern like */5 for every 5 minutes

17.4 Crontab examples

1. Run mycommand at 5:09am on January 1st plus every Monday in January

   ```
   09 05 1 1 1 /path/to/mycommand
   ```

2. Run mycommand at 05 and 35 past the hours of 2:00am and 8:00am on the 1st through the 28th of every January and July.

   ```
   05,35 02,08 1-28 1,7 *  /path/to/mycommand
   ```

3. Run mycommand every 5 minutes

   ```
   */5 * * * *  /path/to/mycommand
   ```

17.4.1 A Better Crontab example

You should be aiming to place all related commands in a separate executable script (.sh) file, rather than entering them one by one in the crontab configuration file.

Suppose you want to perform the following commands, scheduled daily at midnight:

1. VPN to a remote network (e.g. to a VPN server/router)
2. mount a network attached storage (NAS) device connected on the remote network

3. backup (with rsync) some files to the remote network
4. unmount all at a regular scheduled time

Typically, you could collect all these commands in a single (executable) script file; then reference it in the crontab configuration file, e.g.:

- **`touch ~/my_cronjobs.sh`** `#create the script file`
- **`sudo chmod +x ~/my_cronjobs.sh`** `#make it executable`

- **`sudo nano /home/andre/my_cronjobs.sh`** `#script file, edited with:`

```
#!/bin/bash

# connect to VPN
sudo vpnc vpn-ipsec-fritz-box.conf

# mount NAS on remote network via VPN
sudo mount -t cifs //192.168.178.1/HomeNetwork.NAS/Seagate-
ExpansionDesk-
03/BACKUPs/HP.i5.right /mnt/nas/ -o vers=1.0,uid=1000,sid=1000,
user=nas_user,password=***passwd here***

# rsync source from local host over VPN to NAS on remote network, and
# store rsync screen output to a log file
sudo rsync -avh --delete /home/andre/ /mnt/nas/profile_backups/andre/
>> /mnt/nas/profile_backup_log.txt

# unmount NAS-mount
sudo umount /mnt/nas

# disconnect VPN
sudo vpnc-disconnect
```

- **`sudo crontab -e`** `#edit the root crontab config file, and add:`

```
0 0 * * * /home/andre/my_cronjobs.sh
```

17.5 Predefined scheduling definitions

Cron also offers some special strings, which can be used in place of the five time-and-date fields:

Entry	Description	Equivalent To
`@yearly (or @annually)`	Run once a year, midnight, Jan. 1st	`0 0 1 1 *`
`@monthly`	Run once a month, midnight, first of month	`0 0 1 * *`
`@weekly`	Run once a week, midnight on Sunday	`0 0 * * 0`
`@daily`	Run once a day, midnight	`0 0 * * *`
`@hourly`	Run once an hour, beginning of hour	`0 * * * *`
`@reboot`	Run once at startup	

17.6 Run GUI apps via Cron

To do this just add a "env DISPLAY=:0" before the command you want to schedule,eg:

```
        00 06 * * * env DISPLAY=:0 gui_appname
```

You can verify your DISPLAY use below command in command line

```
        echo $DISPLAY
```

17.7 Crontab Options

- The -l option causes the current crontab to be displayed on standard output.
- The -r option causes the current crontab to be removed.
- The -e option is used to edit the current crontab using the editor specified by the EDITOR environment variable (in ~/.bashrc).

After you exit from the editor, the modified crontab is checked for errors and, if there are no errors, it is installed automatically. The file is stored in /var/spool/cron/crontabs but should only be edited using the crontab command.

17.8 Cron security: allowing/denying User-level Cron

By default,every user can create their own cron jobs in Ubuntu.

Files

- /etc/cron.allow
- /etc/cron.deny
- /var/spool/cron/cronjobs

There is one file for each user's crontab under the /var/spool/cron/crontabs directory, stored by their login names. Users are not allowed to edit the files under that directory to ensure that only users allowed by the system to run periodic tasks can add them, and only syntactically correct crontabs will be written here. This is enforced by having the directory writable only by the *crontab* group

If the /etc/cron.allow file exists, then users must be listed in it in order to be allowed to run the crontab command. If the /etc/cron.allow file does not exist but the /etc/cron.deny file does, then users must not be listed in the /etc/cron.deny file in order to run crontab.

In the case where neither file exists, the default on current Ubuntu (and Debian, but not some other Linux and UNIX systems) is to allow all users to run jobs with crontab.

17.9 Common Problems

Edits to a user's crontab and the cron jobs run are all logged by default to `/var/log/syslog` and that's the first place to check if things are not running as you expect.

- `cat /var/log/syslog | grep cron | less`

OR

- `tail -f /var/log/syslog`

If a user was not allowed to execute jobs when their crontab was last edited, just adding them to the allow list won't do anything. The user needs to re-edit their crontab after being added to `cron.allow` before their jobs will run.

Note that user-specific crontabs (including the root crontab) do not specify the user name after the date/time fields. If you accidentally include the user name in a user-specific crontab, the system will try to run the user name as a command.

Cron jobs may not run with the environment, in particular the PATH, that you expect. Try using full paths to files and programs if they're not being located as you expect.

The "%" character is used as newline delimiter in cron commands. If you need to pass that character into a script, you need to escape it as "\%".

17.9.1. Concurrent Cronjobs

Cron is only used to start programs, so try not to overlap running instances of a program or command where concurrency may be an issue – i.e. where concurrent processes access the same files.

If we consider, say, concurrent `rsync` (remote sync) processes with the same (source and/or destination) arguments; there would be no problem for files already synced between source and destination locations. However, there might be problems for files still in transit, or files that were in the process of being synced, and the process itself was terminated prematurely.

At startup, rsync first evaluates for differences (diffs), if any, between the destination and source directories (by size and time by default, or checksum if stipulated), and then transmits the differences to the destination file (or copy the whole file if no file at destination). If an rsync process (2) was to diff a file that is currently partially copied by a concurrent rsync (1), that diff might likely not take into consideration the blocks of data transmitted by rsync (1) after the diff was calculated. This could then result in the rsync completing last corrupting the destination file.

Ofcourse, should a **subsequent rsync** complete the source to destination sync without interruption or concurrent rsync diff issues (which will happen eventually after enough attempts); **the destination file will likely end up <u>un</u>corrupted**.

To circumvent possible concurrency issues, or just to prevent concurrency dragging down throughput (e.g. multiple instances over a network), one would have to consider one or more of the following options:

- **don't allow more than one instance of the same process/command at a time:**
    - a script to first check if a command is currently running, then either:
        - deny starting a new instance

        OR
        - (abrubtly) kill the current instance, and start the new one
    - probably the easier of the first two options here
- **ensuring a job with similar arguments (e.g. rsync) does not commence while a previous is still running**
    - for the rsync example, this could be evaluated with a script which programmatically check the command strings between the currently running command, and the pending command:
        - evaluate the output string(s) from, say:
```
ps –axf –o pid,sid,cmd | grep rsync >
~/Documents/temp.txt
```
        - if any of the lines in the output above matches a source or destination string in the pending command, then either:
            - refuse the pending command from starting up

            OR
            - kill the current instance, and start the new one
- **consider options within the command:**
    - e.g. rsync options:
        - --ignore-times
        - --size-only
        - --checksum

- Can any of the above options, or otherwise, be used to evaluate source and destination versions of a file; then take the safest action to limit the risk of corruption or lowered throughput?

- This option is probably the least likely to reduce the risk of concurrency issues, but nonetheless pertinent to avoid corruption issues and affect transfer throughput.

18. Backups with RSYNC

This chapter is part of the BackupYourSystem series. More introductory information can be found there. The text following is an excertp from https://help.ubuntu.com/community/rsync.

18.1 Introduction

From the man page:

- Rsync is a fast and extraordinarily versatile file copying tool. It can copy locally, to/from another host over any remote shell, or to/from a remote rsync daemon.
- It offers a large number of options that control every aspect of its behavior and permit very flexible specification of the set of files to be copied.
- It is famous for its delta-transfer algorithm, which reduces the amount of data sent over the network by sending only the differences (**diff**) between the source files and the existing files in the destination.
- Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.

In other words, rsync is a tool for efficiently copying and backing up data from one location (the *source*) to another (the *destination*). It is efficient because it only transfers files which are different between the source and destination directories.

18.1.1 Rsync Diffs

What is the difference between, and indeed significance of, the rsync command arguments `--size-only` and `--ignore-times`?

The short answer is that `--ignore-times` does more than its name implies. It ignores both the time and size. In contrast, `--size-only` does exactly what it says.

The long answer is that rsync has three ways to decide if a file is outdated:

1. Compare the size of source and destination.
2. Compare the timestamp of source and destination.
3. Compare the *static* checksum of source and destination.

These checks are performed before transferring data. Notably, this means the *static* checksum is distinct from the *stream* checksum - the latter is computed while transferring data.

By default, rsync uses 1 and 2. Both 1 and 2 can be acquired together by a single stat, whereas 3 requires reading the entire file (this is independent from reading the file for transfer).

- Running rsync on default settings (i.e. no modifiers) will give you the best overall transfer performance outside running the rsync daemon version (section 18.5).

Assuming only one modifier is specified, that means the following:

- By using `--size-only`, only 1 is performed - timestamps and checksum are ignored. A file is copied from source to destination, unless its size is identical on both ends.
    - Ensure 2nd highest throughput for either non-daemon rsync or standard rsync transfers
    - minor risk of missing file changes not reflected in file size.

- By using `--ignore-times`, neither of 1, 2 or 3 is performed. A file is always copied.
    - this option would grant you the poorest throughput. Not recommended, not feasible.

- By using `--checksum`, 3 is used in addition to 1 but 2 is not performed. A file is copied unless size and checksum match. The checksum is only computed if size matches.
    - this option would grant you a low throughput on non-daemon rsync (because remote checksum calculations are slow), albeit being the

18.2 Rsync

Rsync is a command line utility. Users attempting to use it should be familiar with the command line (see Using the Terminal). If you prefer a graphical interface, see the Grsync section of this chapter.

Basic syntax of rsync command

- `sudo rsync options source destination`

Installation

Rsync is installed in Ubuntu by default. But if you want to install it again (for whatever reason):

- `sudo apt-get install rsync`

Perform a Simple Backup

The simplest method for **backing up over a network** is to use rsync via SSH (using the -e ssh option). Alternatively, you can use the *rsync daemon* (see below) which requires much more configuration.

Local backup only requires rsync and read/write access to the folders being synchronized.
The same is true for **syncing over either VPN** and **folder shares on the local network** (say by Samba) requiring only that you have the remote source or target location to be mounted to the filesystem.

Below you will find examples of commands that can be used to backup in either case, where *folder1* represents a directory on the source machine, and *folder2* a directory on the target.

Local BackuP **(incl. VPN & LOCAL network SHARES)**

- `sudo rsync --dry-run -azvv /home/path/folder1/ /home/path/folder2`

Backup Over Network (WITH SSH)

Rsync with standard SSH port (22)

- `sudo rsync --dry-run --delete -azvv -e ssh /home/username/path/folder1/ remoteuser@remotehost-or-IP:/home/username/path/folder2`

Rsync with non-standard SSH port (e.g. 2200)

- `sudo rsync --dry-run --delete -azvv -e 'ssh -p <port-number>' /home/username/path/folder1/ remoteuser@remotehost-or-IP:/home/path/folder2`

Rsync with non-standard SSH port and different private key

- `sudo rsync --dry-run --delete -azvv -e 'ssh -i <path-to-private-key> -p <port-number>' /home/username/path/folder1/ remoteuser@remotehost-or-IP:/home/username/path/folder2`

- e.g:
  - sudo rsync -avh --progess [--dry-run] [--size-only] [--delete] -e "ssh **-i /home/andre/.ssh/id_rsa_samsungi3-300e -p 22** ~~-l samsungi3~~ **[-o StrictHostKeyChecking=no]**" /mnt/storage/backups_heidi/hdd.partitions samsungi3@samsungi3-300e.ddns.net:/mnt/nas/backups_heidi/hdd.partitions

TO CONFIRM IF THIS EXAMPLE WORKS

An explanation of above options to commands:

- `--dry-run` This tells rsync to not actually do anything. It will just write a log of what it would do to the screen. Once you've made sure everything will work as you expect, you have to remove this option, and run the command again to perform the actual backup.
- `--delete` deletes files that don't exist on the system being backed up to (Optional)
- `-a` preserves the date and times, and permissions of the files `(same as -rlptgoD)`.
  - With this option rsync will:
    - Descend recursively into all directories (`-r`),
    - copy symlinks as symlinks (`-l`),
    - preserve file permissions (`-p`),
    - preserve modification times (`-t`),
    - preserve groups (`-g`),
    - preserve file ownership (`-o`), and
    - preserve devices as devices (`-D`).
- `-z` compresses the data
- `-vv` increases the verbosity of the reporting process
- `-e` specifies remote shell to use
- `--progress` display in realtime the bytes synced and time remaining
- `-h` specifies that bytes transferred should be in human readable format (e.g. 1.2GB)
- `/folder1 and folder2` In the examples above, folder1 and 2 are placeholders for the directories to be synchronized. Folder1 is the original folder, and 2 is the new folder, or existing one to be brought in sync with the first. Replace them with the folders you'd like. A / was added after folder1 so that only the contents, rather than whole folder, would be moved into the second.

A complete synopsis of all the options with the rsync command can be found in the man pages under "Options Summary". The man page for rsync can also be found on linux.die.net
You can also use this command to learn more about rsync

- **rsync –help | less**

Backup Over Network (WITH SSH, AND SSHPASS to authenticate user and password)

Introduction

`sshpass` is a utility designed for running `ssh` using the mode referred to as "keyboard-interactive" password authentication, but in non-interactive mode.

ssh uses direct TTY access to make sure that the password is indeed issued by an interactive keyboard user. `Sshpass` runs ssh in a dedicated tty, fooling it into thinking it is getting the password from an interactive user.

The command to run is specified after sshpass' own options. Typically it will be "ssh" with arguments, but it can just as well be any other command. The password prompt used by ssh is, however, currently hardcoded into sshpass.

Full help available with `man sshpass` or `sshpass --help`.

Security Considerations

First and foremost, users of sshpass should realize that ssh's insistance on only getting the password interactively is not without reason. It is close to impossible to securely store the password, and users of sshpass should consider whether ssh's **public key authentication** provides the same end-user experience, while involving less hassle and being more secure. That is: you should rather authenticate as per previous section above.

Installation

You'll have to install sshpass on both client and server

- sudo apt install sshpass

Rsync with non-standard SSH port, login and password

- `sudo rsync -avv --rsh="sshpass -p ***password_here*** ssh -l samsungi3 -o`

```
StrictHostKeyChecking=no" /mnt/nas/SOFTWARE.\&.APPS/ samsungi3@samsungi3-
300e.ddns.net:/home/samsungi3/Downloads/delete_anytime/
```

Rsync Issues

Concurrency

With reference to section 17.9.1.

18.4 Grsync

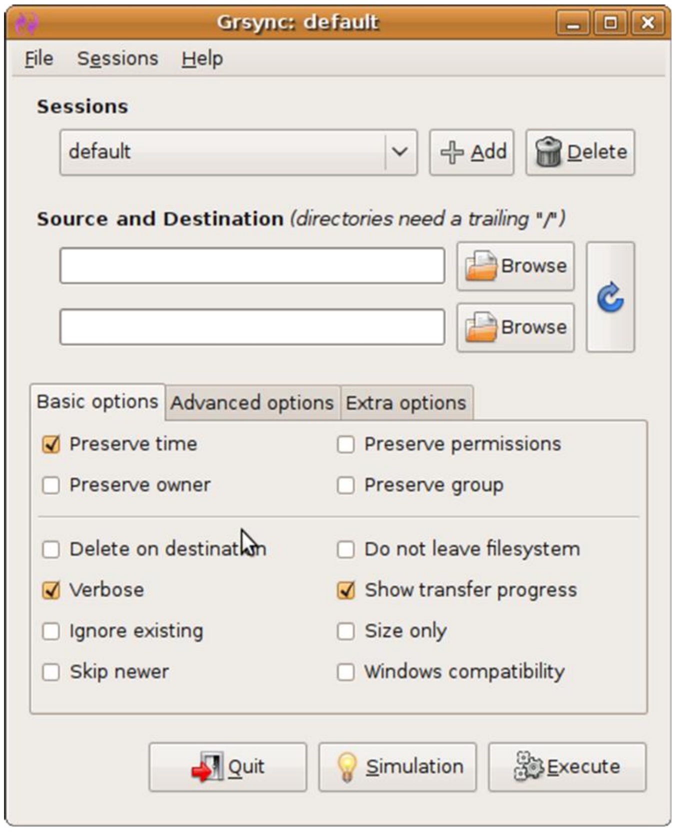
Grsync is a GUI frontend for the rsync utility, running in a Desktop (GUI) Environment. The simple interface of the GUI exposes many of the basic options available with rsync. It is useful for those who prefer not to use the command line.

Installation

The program grsync does not come installed by default on Ubuntu or any other distrubtion but it is easily available from the main Repositories. To get grsync ensure **Universe** section of the Ubuntu repositories is enabled in your Software Sources. Then to install this software in Ubuntu, install the following package: ***grsync***.

Configuration

To start up grsync go through the following menus: Applications --> System Tools --> grsync. Upon start up you'll be presented with the main window, where all the configuration takes place.



Sessions

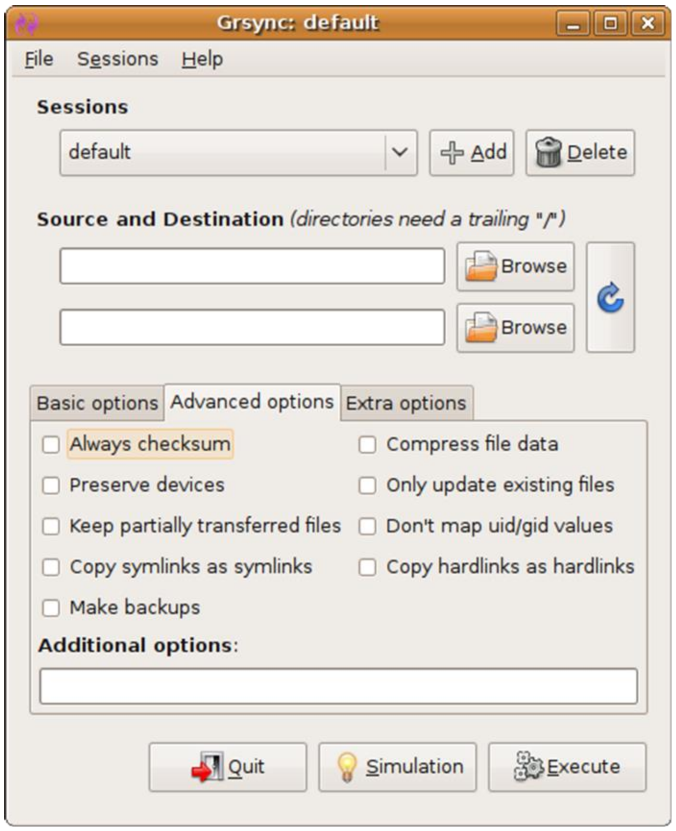
default

Source and Destination *(directories need a trailing "/")*

- [✓] Preserve time
- [ ] Preserve permissions
- [ ] Preserve owner
- [ ] Preserve group
- [ ] Delete on destination
- [ ] Do not leave filesystem
- [✓] Verbose
- [✓] Show transfer progress
- [ ] Ignore existing
- [ ] Size only
- [ ] Skip newer
- [ ] Windows compatibility

On this window are all of the options most users will ever need. To explain, the options will be listed and their effects mentioned.

- **Sessions** - This function is the same as profiles in others. Each session will store a different set of source and destination directories, as well as the configuration options associated with the folder pair. This allows for the synchronization of different sets of folders according to different options.
    - Management of sessions is simple, simply push the **Add** button to add a new one. To delete, select the session you no longer want from the drop down and push **Delete**.
- **Source and Destination** - These two boxes list the two folders (technically referred to as directories) that will be synchronized. The top one is the **Source** and the bottom the **Destination**. So when you **Execute** the synchronization, the files from Source will be copied over to the Destination according to the options a user selects.
    - To specify the directories either **Browse** for them from the GUI or type them in according to the standard path conventions.
- **Switch** - The universal reload sign located to the right of the **Browse** buttons is a handy button. It will instantly switch the **Source** with the **Destination**.
- **Import** and **Export** - After having configured sessions, a user may want to back them up for storage. To do so, simply go to the Sessions Menu at the top and select either **Import** or **Export**. The former will restore a session from a backup previously made, the latter will make a backup of the current session.
    - Note: This backup function works on a per session basis. This means, each session you want to back up must be selected from the drop down and then backed up. If you have 3 different sessions, select each in turn and **Export** them. Same when importing sessions.
- **Basic Options** - Most users will find most of the options they will ever need here. The first four will preserve the properties of the files transferred. The others will modify how the files are copied. For more information on what each does specifically, hover your stationary cursor over the option and it will display a small explanation. The options checked are of course the ones that will be applied during the session.



- **Advanced Options** - This tab holds more options, many are useful and self-explanitory. For those not understood, tooltips will be displayed when the mouse remains over an option long enough.
- **Additional Options** - This entry box allows the input of additional options not presented in the GUI but known to the user. Use is suggested only for experienced users, inputting malformed options may have unexpected consequences.

Simulation and Execution

The last two buttons on the window are **Simulation** and **Execute**. The button for simulation is very useful when uncertain what will happen based on the options selected. The normal transfer dialog screen will pop up and in the main pane, a list of files that would have been copied over is listed. The user can then verify if this is as desired or make changes. Once the session is initiated with the **Execute** button, the dialog will appear again but this time it will actually process the folders accordingly. Ensure before pushing **Execute** that you are happy with the simulation.

Remote Backup

Backup over a network is possible, preferably the user should mount the network share to be backed up to prior to launching the program. The share would then be listed in the Browse GUI and could easily be added. There is no separate section for network, if more advanced features are required the user is encouraged to look at alternatives, of which there are many.

Alternatives

There are many alternatives, in various stages of development. For an incomplete list, see here.

18.5 Rsync Daemon (another alternative; not secure though!!)

The rsync daemon is an alternative to rsync-over-SSH for remote backups. Although more difficult to configure, it does provide some benefits. For example, using SSH to make a remote backup of an entire system requires that the SSH daemon allow root login, which is considered a security risk. Using the rsync daemon allows for root login via SSH to be disabled.

Secure Rsync'ing: SSH & VPN

IMPORTANT: rsync transfers data in clear text over the internet – so there is a clear security risk in that man-in-the-middle attacks can eavesdrop on your data. As you'll learn below, you can configure a login name and password for authentication. Your login name is sent in clear text, while your password is not (obviously). This still **does not encrypt** any of your data.

To be absolutely secure, you should NEVER rsync data to an rsync daemon over an unsecure network such as the internet. The only secure way to rsync data to rsync daemon, is to do so **over VPN**. The only other secure option to take would be **rsync over SSH** as we've explained in a previous section.

Configuration of the rsync Daemon

1. Edit the file `/etc/default/rsync` to start rsync as daemon using `xinetd`.

```
SIDE NOTE:
xinetd performs  the  same  function as inetd: it starts programs that
provide Internet services.  Instead of having such servers started at
system initialization time, and be dormant until a connection request
```

arrives, **xinetd** is the only daemon process started and it listens on all service ports for the services listed in its configuration file. When a request comes in, **xinetd** starts the appropriate server. Because of the way it operates, **xinetd** (as well as **inetd**) is also referred to as a super-server.

The entry listed below, should be changed from false to inetd.

```
RSYNC_ENABLE=inetd
```

2. Install xinetd because it's not installed by default.

```
$ sudo apt-get -y install xinetd
```

3. Create the file /etc/xinetd.d/rsync to launch rsync via xinetd. It should contain the following lines of text.

```
service rsync
{
    disable = no
    socket_type = stream
    wait = no
    user = root
    server = /usr/bin/rsync
    server_args = --daemon
    log_on_failure += USERID
    flags = IPv6
}
```

4. Create the file /etc/rsyncd.conf configuration for rsync in daemon mode. The file should contain the following. In the file, **user** should be replaced with the name of user on the remote machine being logged into.

```
max connections = 2
log file = /var/log/rsync.log
timeout = 300

[share]
comment = Public Share
path = /home/share
read only = no
list = yes
uid = nobody
gid = nogroup
auth users = user
secrets file = /etc/rsyncd.secrets
```

5. Create */etc/rsyncd.secrets* for user's password. User should be the same as above, with password the one used to log into the remote machine as the indicated user.

```
$ sudo nano /etc/rsyncd.secrets
user:password
```

6. This step sets the file permissions for rsyncd.secrets.

```
$ sudo chmod 600 /etc/rsyncd.secrets
```

7. Start/Restart xinetd

```
$ sudo /etc/init.d/xinetd restart
```

Testing

Run the following command to check if everything is ok. The output listed is just a sample, should be what is on your shared remote machine. Hostname can be replaced by the IP address of the machine.

```
$ sudo rsync user@hostname::share
  Password:
  drwxr-xr-x        4096 2006/12/13 09:41:59 .
  drwxr-xr-x        4096 2006/11/23 18:00:03 folders
```

Transferring Files

TBC