

# Linux Administrator

Alejandro Campos

October, 2023

# Contents

<b>1</b>	<b>Linux Directory Structure Explained</b>	<b>6</b>
1.1	/ – The Root Directory . . . . .	6
1.2	/bin – Essential User Binaries . . . . .	6
1.3	/sbin – System Administration Binaries . . . . .	6
1.4	/boot – Static Boot Files . . . . .	6
1.5	/dev – Device Files . . . . .	6
1.6	/etc – Configuration Files . . . . .	6
1.7	/home – Home Folders . . . . .	6
1.8	/root – Root Home Directory . . . . .	7
1.9	/lib – Essential Shared Libraries . . . . .	7
1.10	/lost+found – Recovered Files . . . . .	7
1.11	/media – Removable Media . . . . .	7
1.12	/mnt – Temporary Mount Points . . . . .	7
1.13	/opt – Optional Packages . . . . .	7
1.14	/tmp – Temporary Files . . . . .	7
1.15	/usr – User Binaries & Read-Only Data . . . . .	7
1.16	/var – Variable Data Files . . . . .	8
<b>2</b>	<b>Managing Linux Users and Groups</b>	<b>9</b>
2.1	Introduction . . . . .	9
2.2	Managing Linux Users . . . . .	9
2.2.1	Type of Users in Linux . . . . .	9
2.2.2	Understanding the /etc/passwd file . . . . .	9
2.2.3	Understand the /etc/shadow file . . . . .	10
2.2.4	Add User to Linux System . . . . .	11
2.2.5	Switching User in Linux - su command . . . . .	13
2.2.6	Delete User in Linux System . . . . .	13
2.2.7	Manage User from Linux System . . . . .	14
2.2.8	SSH Key-Based Authentication . . . . .	14
2.3	Managing Linux Groups . . . . .	15
2.3.1	Understanding the /etc/group file . . . . .	15
2.3.2	Add a Group in Linux . . . . .	15
2.3.3	Change the group ID . . . . .	16
2.3.4	Rename a group . . . . .	16
2.3.5	How to Assign Users to Groups in Linux . . . . .	16
2.3.6	How to Delete Users from Groups in Linux . . . . .	16

2.3.7	How to delete a group . . . . .	16
2.4	Sudo Command . . . . .	16
2.4.1	Give sudo permissions with password . . . . .	16
2.4.2	Give only some actions sudo permission . . . . .	18
2.4.3	How to enable sudo without entering a password . . . . .	19
2.4.4	How to gain su permissions with sudo . . . . .	19
2.5	Su - root user . . . . .	20
2.5.1	First Approach - Enabling root account temporary . . . . .	20
2.5.2	Enable root user . . . . .	20
2.5.3	Disable root user . . . . .	20
<b>3</b>	<b>Basic Linux Commands</b>	<b>22</b>
3.1	Linux System Information . . . . .	22
3.2	Linux System Consumption . . . . .	22
3.3	Linux Services . . . . .	23
3.3.1	ps . . . . .	23
3.3.2	systemctl . . . . .	23
3.4	Service . . . . .	24
3.5	Moving between directories . . . . .	24
3.5.1	Introduction . . . . .	24
3.5.2	ls -la Analyzing Results . . . . .	24
3.6	chmod . . . . .	25
3.7	chown . . . . .	26
3.8	Ctrl+R . . . . .	27
3.9	Mkdir (create files) . . . . .	27
3.10	Mv / Cp . . . . .	27
3.11	Simbolik Links . . . . .	28
3.12	Aliases . . . . .	28
3.12.1	Ephimeral Aliases . . . . .	28
3.12.2	Permanent Aliases . . . . .	28
3.13	Source . . . . .	29
3.14	Shutdown machines or terminals . . . . .	30
3.15	Others . . . . .	30
<b>4</b>	<b>.bashrc, .bash_profile, /etc/bashrc &amp; path</b>	<b>31</b>
4.1	.bashrc vs .bash_profile . . . . .	31
4.2	PATH . . . . .	31
4.3	Set your PATH . . . . .	31
4.3.1	For current terminal . . . . .	31

4.3.2	Permanently for interactive sessions . . . . .	31
<b>5</b>	<b>Useful Tools</b>	<b>32</b>
5.1	Cat . . . . .	32
5.2	Head & Tail . . . . .	32
5.3	Diff (File comparator) . . . . .	33
5.4	Grep (Global search for Regular Expressions and Print out) . . . . .	34
5.4.1	Two flavors of grep usage . . . . .	34
5.4.2	Useful Flags . . . . .	34
5.5	Find . . . . .	35
5.6	Sed (Stream Editor) . . . . .	36
5.7	AWK . . . . .	37
5.8	Tar Command . . . . .	38
5.8.1	Compress . . . . .	38
5.8.2	Extract . . . . .	38
5.8.3	More flags . . . . .	39
5.9	Base64 Encoding . . . . .	39
5.10	JQ . . . . .	39
<b>6</b>	<b>Linux Default Text Editors</b>	<b>40</b>
6.1	Vi . . . . .	40
6.2	Vim . . . . .	40
6.2.1	Vim Config . . . . .	40
6.2.2	Hacer y deshacer . . . . .	42
6.2.3	Borrar lineas enteras . . . . .	42
6.2.4	Buscar Ocurrencias . . . . .	42
6.3	Nano . . . . .	43
<b>7</b>	<b>Network</b>	<b>44</b>
7.1	FQDN . . . . .	44
7.2	SSH . . . . .	45
7.2.1	What is SSH? . . . . .	45
7.2.2	Authenticate SSH connections using user-password . . . . .	46
7.2.3	Authenticate SSH connections using Asimetric Key authentication method . . . . .	46
7.2.4	SSH Extra Arguments . . . . .	49
7.2.5	SFTP . . . . .	49
7.3	Localhost Networking . . . . .	50
7.3.1	Ports Exposed . . . . .	50
7.3.2	Localhost Adresses . . . . .	51
7.4	Certificates . . . . .	52

<b>8</b>	<b>Network Tools</b>	<b>53</b>
8.1	Ping . . . . .	53
8.2	Curl . . . . .	53
8.3	Nmap . . . . .	54
8.4	Netcat . . . . .	54
8.5	Telnet . . . . .	54
8.6	Proxy variables definition for binaries . . . . .	54
<b>9</b>	<b>Tmux</b>	<b>56</b>
9.1	Arguments . . . . .	56
9.2	Commands Inside tmux . . . . .	57
<b>10</b>	<b>Others</b>	<b>58</b>
10.1	Docker Desktop . . . . .	58
10.2	Helm . . . . .	59
10.3	JSONPath . . . . .	59
10.3.1	Basic Usage . . . . .	59
10.3.2	Filters . . . . .	60
10.3.3	Example . . . . .	61
10.3.4	JSONPath K8s . . . . .	65
10.4	Ubuntu vs RH7 . . . . .	66
10.5	Vagrant . . . . .	66
10.6	Red Hat Package Installer . . . . .	67
10.6.1	RPM . . . . .	67
10.6.2	YUM . . . . .	67
10.7	Timeshift to Backup and Restore Your Linux System . . . . .	67
10.7.1	Introduction . . . . .	67
10.7.2	Timeshift Installation . . . . .	67
10.7.3	Create a Snapshot . . . . .	67

# 1 Linux Directory Structure Explained

## 1.1 / – The Root Directory

Everything on your Linux system is located under the / directory, known as the root directory. You can think of the / directory as being similar to the C:\ directory on Windows. But this isn't strictly true, as Linux doesn't have drive letters. While another partition would be located at D:\ on Windows, this other partition would appear in another folder under / on Linux.

## 1.2 /bin – Essential User Binaries

The /bin directory contains the essential user binaries (programs) that must be present when the system is mounted in single-user mode. Users applications such as Firefox are stored in /usr/bin, while important system programs and utilities such as the bash shell, python, ansible, docker are located in /bin. Placing these files in the /bin directory ensures the system will have these important utilities even if no other file systems are mounted.

## 1.3 /sbin – System Administration Binaries

The /sbin directory is similar to the /bin directory. It contains essential system administration binaries, which are generally intended to be run by the root user for system administration.

## 1.4 /boot – Static Boot Files

The /boot directory contains the files needed to boot the system. For example, the GRUB boot loader's files and your Linux kernels are stored here. The boot loader's configuration files aren't located here, though they're in /etc with the other configuration files.

## 1.5 /dev – Device Files

Linux exposes devices as files, and the /dev directory contains a number of special files that represent devices. This directory also contains pseudo-devices, which are virtual devices that don't actually correspond to hardware. For example, /dev/random produces random numbers. /dev/null is a special device that produces no output and automatically discards all input. When you pipe the output of a command to /dev, you discard it

## 1.6 /etc – Configuration Files

The /etc directory contains configuration files, which can generally be edited by hand in a text editor. Note that the /etc directory contains system-wide configuration files.

### NOTE

user-specific configuration files are located in each user's home directory.

## 1.7 /home – Home Folders

The /home directory contains a home folder **for each user**. For example, if your user name is bob, you have a home folder located at /home/bob. This home folder contains the user's data files and user-specific configuration files. **Each user only has write access to their own home folder** and must obtain elevated permissions (become the root user) to modify other files on the system.

## 1.8 /root – Root Home Directory

The `/root` directory is the home directory of the root user. Instead of being located at `/home/root`, as the rest of the users, it's located at `/root`. This is distinct from `/`, which is the system root directory.

## 1.9 /lib – Essential Shared Libraries

The `/lib` directory contains libraries needed by the essential binaries in the `/bin` and `/sbin` folder.

### NOTE

Libraries needed by the binaries in the `/usr/bin` folder are located in `/usr/lib`.

## 1.10 /lost+found – Recovered Files

Each Linux file system has a `lost+found` directory. If the file system crashes, a file system check will be performed at next boot. Any corrupted files found will be placed in the `lost+found` directory, so you can attempt to recover as much data as possible.

## 1.11 /media – Removable Media

The `/media` directory contains subdirectories where removable media devices inserted into the computer are mounted. For example, when you insert a CD into your Linux system, a directory will automatically be created inside the `/media` directory. You can access the contents of the CD inside this directory.

## 1.12 /mnt – Temporary Mount Points

Historically speaking, the `/mnt` directory is where system administrators mounted temporary file systems while using them. For example, if you're mounting a Windows partition to perform some file recovery operations, you might mount it at `/mnt/windows`. However, you can mount other file systems anywhere on the system.

## 1.13 /opt – Optional Packages

The `/opt` directory contains subdirectories for optional software packages. It's commonly used by proprietary software that doesn't obey the standard file system hierarchy. For example, a proprietary program might dump its files in `/opt/application` when you install it.

## 1.14 /tmp – Temporary Files

Applications store temporary files in the `/tmp` directory. These files are generally deleted whenever your system is restarted and may be deleted at any time by utilities such as `tmpwatch`.

## 1.15 /usr – User Binaries & Read-Only Data

The `/usr` directory contains applications and files used by users, as opposed to applications and files used by the system. For example, non-essential applications are located inside the `/usr/bin` directory instead of the `/bin` directory and non-essential system administration binaries are located in the `/usr/bin` directory instead of the `/sbin` directory.

## 1.16 /var – Variable Data Files

The `/var` directory is the writable counterpart to the `/usr` directory, which must be read-only in normal operation. Log files and everything else that would normally be written to `/usr` during normal operation are written to the `/var` directory. For example, you'll find log files in `/var/log`.



## 2 Managing Linux Users and Groups

### 2.1 Introduction

Linux is a multi-user system, which means that more than one person can interact with the same system at the same time. As a system administrator, you have the responsibility to manage the system's users and groups by creating and removing users and assign them to different groups .

### 2.2 Managing Linux Users

In a Linux system, users refer to individuals or entities that interact with the operating system by logging in and performing various tasks. User management plays a crucial role in ensuring secure access control, resource allocation, and system administration.

A user in Linux is associated with a user account, which consists of several properties defining their identity and privileges within the system. These properties are a username, UID (User ID), GID (Group ID), home directory, default shell, and password.

#### 2.2.1 Type of Users in Linux

Linux supports two types of users: system users and regular users.

- **System users:** are created by the system during installation and are used to run system services and applications.
- **Regular users:** are created by the administrator and can access the system and its resources based on their permissions.

#### 2.2.2 Understanding the `/etc/passwd` file

User account information is stored in the `/etc/passwd` file. This information includes the account name, home directory location, and default shell, among other values. Each field is separated by a ":" character, and not all fields must be populated, but you must delineate them.

```
username:password:UID:GID:comment:home:shell
```

- **UID:** User Identifier
  - **0:** reserved for root user
  - **1-999:** reserved by system for administrative and system users.
- **GID:** Group Identifier
  - **0:** reserved for root group
  - **1-99:** reserved by system for administrative and system groups.

#### NOTE

UID and GUI assignation policies are defined in the file `/etc/login.defs`

```

acamos@BCNLT5CG3284PRF:~$ cat /etc/login.defs | grep -i uid
# for private user groups, i. e. the uid is the same as gid, and username is
# Min/max values for automatic uid selection in useradd
UID_MIN                1000
UID_MAX                60000
#SYS_UID_MIN           100
#SYS_UID_MAX           999
# (examples: 022 -> 002, 077 -> 007) for non-root users, if the uid is
acamos@BCNLT5CG3284PRF:~$ cat /etc/login.defs | grep -i gid
# If you have a "write" program which is "setgid" to a special group
# In Debian /usr/bin/bsd-write or similar programs are setgid tty
# for private user groups, i. e. the uid is the same as gid, and username is
# Min/max values for automatic gid selection in groupadd
GID_MIN                1000
GID_MAX                60000
#SYS_GID_MIN           100
#SYS_GID_MAX           999
# the same as gid, and username is the same as the primary group name.

```

So you can trim the output using AWK or cut:

```
$ awk -F ":" '{ print $1 }' /etc/passwd
```

```
$ cut -d ":" -f1 /etc/passwd
```

## NOTE

We will discuss passwords in the next sect, but expect to see an "x" in the password field of this file.

### 2.2.3 Understand the /etc/shadow file

Long ago, password hashes were stored in the /etc/passwd file. This file was world-readable, allowing inquisitive users to pull password hashes for other accounts from the file and run them through password-cracking utilities. Eventually, the password hashes were moved to a file readable only by root: /etc/shadow. Today, the password field in the /etc/passwd file is marked with an x.

## NOTE

To know more about the passwords, the value we can see in the /etc/passwd is a hashed value, produced by the crypt function in Linux when we set the passphrase for the user using the passwd command. The hashed passphrase follows a specific format:

```
$id$salt$hashedpassword
```

- **id:** is the hashing method used when hashing the passphrase. For example, if the hash value is produced by yescrypt, the ID will be y, and 6 if the sha512crypt method is used. For a complete list of hashing methods and their ID, we can refer to the [Official Documentation](#).
- **salt:** adding random data to the input of a hash function to guarantee a unique output, the hash, even when the inputs are the same.
- **hashedpassword:** the password encrypted

## 2.2.4 Add User to Linux System

### 2.2.4.1 Basic Addition

```
$ sudo useradd [OPTIONS] user_name
```

When invoked, `useradd` creates a new user account according to the options specified on the command line and the default values set in the `/etc/default/useradd` file.

#### NOTE

When executed without any option, `useradd` creates a new user account using the default settings specified in the `/etc/default/useradd` file.

#### WARNING

Only root or users with sudo privileges can use the `useradd` command to create new user accounts.

To check the user has been correctly created

```
$ id username
```

To be able to log in as the newly created user, you need to set the user password

```
$ sudo passwd username
```

You will be prompted to enter and confirm the password. Make sure you use a strong password.

### 2.2.4.2 Addition with Home Directory Creation

On most Linux distributions, when creating a new user account with `useradd`, **by default the user's home directory is not created**.

Use the `-m` (`--create-home`) option to create the user home directory as `/home/username`

```
$ sudo useradd -m username
```

The command above creates the new user's home directory and copies files from `/etc/skel` directory to the user's home directory. If you list the files in the `/home/username` directory, you will see the initialization files:

- `.bash_logout`
- `.bashrc`
- `.profile`

```
acampos@BCNLT5CG3284PRF:~$ ls -la /etc/skel/
total 20
drwxr-xr-x  2 root root 4096 May  1 23:35 .
drwxr-xr-x 73 root root 4096 Oct  5 14:51 ..
-rw-r--r--  1 root root  220 Jan  6  2022 .bash_logout
-rw-r--r--  1 root root 3771 Jan  6  2022 .bashrc
-rw-r--r--  1 root root  807 Jan  6  2022 .profile
```

## NOTE

Within the home directory, the user can write, edit and delete all files and directories.

### To create another different home

```
$ sudo useradd -d /custom/home username
```

### 2.2.4.3 Custom Shell

```
$ sudo useradd -s /custom/shell username
```

### 2.2.4.4 Addition Specifying the UID

By default, when a new user is created, the system assigns the next available UID from the range of user IDs specified in the `/etc/login.defs` file.

```
acampos@BCNLT5CG3284PRF:~$ cat /etc/login.defs | grep -i uid
# for private user groups, i. e. the uid is the same as gid, and username is
# Min/max values for automatic uid selection in useradd
UID_MIN                1000
UID_MAX                60000
#SYS_UID_MIN           100
#SYS_UID_MAX           999
# (examples: 022 -> 002, 077 -> 007) for non-root users, if the uid is
```

Invoke `useradd` with the `-u` (`--uid`) option to create a user with a specific UID. For example to create a new user named `username` with UID of 1500 you would type:

```
$ sudo useradd -u 1500 username
```

## WARNING

If the specified UID is already allocated to another user, you're alerted that the UID is unavailable and the operation aborts. Rerun it with a different UID number.

## NOTE

An easy way to look your user UID:

```
$ id -u username
```

### 2.2.4.5 Addition Specifying the GID

When creating a new user, the default behavior of the `useradd` command is to create a group with the same name as the username, and same GID as UID.

The `-g` (`--gid`) option allows you to create a user with a specific initial login group. You can specify either the group name or the GID number. The group name or GID must already exist.

```
$ sudo useradd -g group username
```

#### WARNING

If the specified group ID is already allocated to another group, you're alerted that the GID is unavailable and the operation aborts. Rerun it with a different group ID number.

#### NOTE

An easy way to look your user primary group:

```
$ id -gn username
```

### 2.2.4.6 Addition Assigning Multiple Groups

There are two types of groups in Linux operating systems Primary group and Secondary (or supplementary) group. Each user can belong to **exactly one primary** group and **zero or more secondary groups**.

You to specify a list of supplementary groups which the user will be a member of with the `-G (--groups)` option.

```
$ sudo useradd -g primary_group -G sec_group1, sec_group2, sec_group3 username
```

#### NOTE

An easy way to look all your user configuration:

```
$ id username
```

### 2.2.4.7 Addition with more Configurations

To create users with more different configurations, you can check the [documentation](#).

### 2.2.5 Switching User in Linux - su command

Su allows you to change the existing user to some other user. Use the `-l username` method to define a user account if you need to execute a command as someone other than root.

### 2.2.6 Delete User in Linux System

Deleting a user requires deleting both the user account as well as the files that are connected with the user account. With this command you do both:

```
$ userdel -r username
```

### WARNING

The above command deletes the user whose username is provided. Make sure that the user is not part of a group. If the user is part of a group then it will not be deleted directly, hence we will have to first remove him from the group and then we can delete him.

## 2.2.7 Manage User from Linux System

### Change the home directory

```
$ usermod -d new_home_directory_path username
```

### Change login name

```
$ sudo usermod -l new_login_name old_login_name
```

### Change the user password

```
$ passwd
```

### WARNING

It will require authentication with the old password, so you should know the old password to change it this way.

If you do not know the user password, you can use the sudo user to do it: **Change the user password**

```
$ sudo passwd username
```

### NOTE

Even further, you can force Linux user to change password at their next login, check the [documentation](#)

### Modify the UID of a user

```
$ usermod -u new_uid username
```

### Modify the group GID of a user

```
$ usermod -g new_gid username
```

## 2.2.8 SSH Key-Based Authentication

Secure Shell (SSH) key-based authentication provides a more secure alternative to password-based authentication. Users generate a public-private key pair, where the public key is stored on the server and the private key is kept securely on the user's device.

With SSH key-based authentication, users like Lisa, a system administrator at CTechCo, can authenticate without entering a password. Instead, the server verifies the user's identity based on the possession of

the private key.

To configure SSH key-based authentication for Lisa, the following steps can be taken:

- Generate an SSH key pair on Lisa's machine using the `ssh-keygen` command.
- Copy the public key to the server's `/home/lisasmith/.ssh/authorized_keys` file.
- Configure the server to allow SSH key-based authentication.

## 2.3 Managing Linux Groups

In Linux, groups are collections of users. Creating and managing groups is one of the simplest ways to deal with multiple users simultaneously, especially when dealing with permissions. It's more efficient to group user accounts with similar access requirements than to manage permissions on a user-by-user basis.

### 2.3.1 Understanding the `/etc/group` file

Similar to the `/etc/passwd` file above, the `/etc/group` file contains group account information. This information can be essential for troubleshooting, security audits, and ensuring users can access the resources they need.

The fields in the `/etc/group` file are:

```
groupname:password:GID:group members
```

### 2.3.2 Add a Group in Linux

To create a new group:

```
$ sudo groupadd group_name
```

To view the group you just added, run the command below:

```
$ cat /etc/group | grep -i group_name
```

#### NOTE

When a group is created, a unique group ID gets assigned to that group. You can verify that the group appears (and see its group ID) by looking in the `/etc/group` file.

If you want to create a group with a specific group ID (GID), use the `--gid` or `-g` option:

```
$ sudo groupadd -g 1009 group_name
```

#### WARNING

If the specified group ID is already allocated to another group, you're alerted that the GID is unavailable and the operation aborts. Rerun it with a different group ID number.

### 2.3.3 Change the group ID

You can change the group ID of any group with the `groupmod` command and the `--gid` or `-g` option:

```
$ sudo groupmod -g 1011 demo1
```

### 2.3.4 Rename a group

You can rename a group using `groupmod` with the `--new-name` or `-n` option:

```
$ sudo groupmod -n test demo1
```

### 2.3.5 How to Assign Users to Groups in Linux

Once a group is created, users can be added to it in the following way:

```
$ sudo usermod -aG group_name username
```

To check the user has been successfully added:

```
$ id username
```

### 2.3.6 How to Delete Users from Groups in Linux

To remove a specific user from a group, you can use the `gpasswd` command to modify group information:

```
$ sudo gpasswd --delete username group_name
```

### 2.3.7 How to delete a group

When a group is no longer needed, you delete it by using the `groupdel` command:

```
$ sudo groupdel demo
```

## 2.4 Sudo Command

The `sudo` command allows you to run programs as the root user. Using `sudo` instead of login in as root is more secure because you can grant limited administrative privileges to individual users without them knowing the root password. **That's why the root user account in Ubuntu is disabled by default for security reasons, and users are encouraged to perform system administrative tasks using `sudo`.** The initial user created by the Ubuntu installer is already a member of the `sudo` group, so if you are running Ubuntu, chances are that the user you are logged in as is already granted with `sudo` privileges.

### 2.4.1 Give sudo permissions with password

By default, on most Linux distributions granting `sudo` access is as simple as adding the user to the `sudo` group defined in the `sudoers` file. Members of this group will be able to run any command as root. The



name of the **group** may differ from distribution to distribution.

- **sudo**: Debian, Ubuntu and derivatives groups for sudo permissions.

```
$ usermod -aG sudo username
```

- **wheel**: RedHat, CentOS and Fedora group for sudo permissions

```
$ usermod -aG wheel username
```

We can see the permission groups allowed in the `/etc/sudoers` file:

```
$ sudo cat /etc/sudoers
```

```
acamp@BCNLT5CG3284PRF:/home$ sudo tail -n 13 /etc/sudoers

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
```

So in my Linux System, any user added to the groups **sudo** or **admin** will have all sudo permissions enabled.

What would happen with a user that is not in these groups?

```
$ whoami
jimony
$ id jimony
uid=1001(jimony) gid=1002(jimony) groups=1002(jimony)
$ sudo whoami
[sudo] password for jimony:
jimony is not in the sudoers file. This incident will be reported.
$ |
```

But if we add him to the admin group... Voilà!

```
acampos@BCNLT5CG3284PRF:/home$ sudo usermod -aG admin jiminy
acampos@BCNLT5CG3284PRF:/home$ su jiminy
Password:
$ whoami
jiminy
$ id jiminy
uid=1001(jiminy) gid=1002(jiminy) groups=1002(jiminy),115(admin)
$ sudo whoami
[sudo] password for jiminy:
root
$ |
```

#### NOTE

The password you have to introduce to access to sudo permissions is the user password, in this case, the jiminy password.

#### 2.4.2 Give only some actions sudo permission

To allow a specific user to run only certain programs as sudo, instead of adding the user to the sudo group, add the users to the `/etc/sudoers` file. For example, to allow the user linuxize to run only the `mkdir` command:

#### WARNING

Modifying `/etc/sudoers` file can be very dangerous, so do it carefully. **Se puede liar muy parda!**

Advices:

- Use **sudo visudo**, because visudo will warn you that there's a syntax error and asks to undo the changes.
- It is hardly recommended to make a copy of the file before

```
$ sudo cp /etc/sudoers sudoerscopy
```

Just if you know what you're doing, keep going:

```
$ sudo visudo
```

```
/etc/sudoers
```

```
linuxize ALL=/bin/mkdir
```

#### NOTE

The file you are opening with `sudo visudo` is the `/etc/sudoers` file with the Nano editor! If you are not used to Nano, you can open it with vim. But vim does not notify you if there are some syntax error in the file, nano does.

```
$ sudo vim /etc/sudoers
```

### 2.4.3 How to enable sudo without entering a password

Modifying the `/etc/sudoers` adding the user with the following:

#### WARNING

Modifying `/etc/sudoers` file can be very dangerous, so do it carefully. **Se puede liar muy parda!**

Advices:

- Use **sudo visudo**, because visudo will warn you that there's a syntax error and asks to undo the changes.
- It is hardly recommended to make a copy of the file before

```
$ sudo cp /etc/sudoers sudoerscopy
```

```
$ sudo visudo
```

```
/etc/sudoers
```

```
jiminy ALL=(ALL:ALL) NOPASSWD: ALL
```

```
$ whoami
jiminy
$ sudo tail -2 /etc/sudoers
@includedir /etc/sudoers.d
jiminy ALL=(ALL:ALL) NOPASSWD: ALL
$ sudo ls
acampos jiminy
$ |
```

Also you can modify it in the group configuration:

```
/etc/sudoers
```

```
# Members of the admin group may gain root privileges without pwd
%admin ALL=(ALL) NOPASSWD: ALL
```

### 2.4.4 How to gain su permissions with sudo

If you try to redirect the output of a command to a file that your user has no write permissions, you will get a "Permission denied" error.

```
$ sudo echo "test" > /root/file.txt
# Output: bash: /root/file.txt: Permission denied
```

This happens because the redirection ">" of the output is performed under the user you are logged in, not the user specified with sudo. The redirection happens before the sudo command is invoked.

One solution is to invoke a new shell as root by using `sudo sh -c`:

```
$ sudo sh -c 'echo "test" > /root/file.txt'
```

Another option is to pipe the output as a regular user to the tee command , as shown below:

```
$ sudo echo "test" | sudo tee /root/file.txt
```

## 2.5 Su - root user

As we comment in the previous section, Using sudo instead of login in as root is more secure because you can grant limited administrative privileges to individual users without them knowing the root password. **That's why the root user account in Ubuntu is disabled by default for security reasons, and users are encouraged to perform system administrative tasks using sudo.**

### 2.5.1 First Approach - Enabling root account temporary

If you only need the root account for a particular task or job, run the following command and supply the super user password to authenticate the action with sudo.

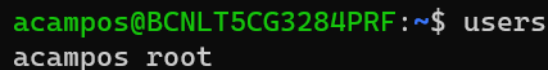
```
$ sudo -i
```

What behind the scenes this command makes is to enable the root user only in the current shell (associated Linux process), and when this shell is shot down (for example typing exit) and you come back to you user, the root will be disabled again.

### 2.5.2 Enable root user

```
$ sudo -i passwd root
```

You will have to enter the new password for the root user and go ahead.

A terminal window showing the prompt 'acamp0s@BCNLT5CG3284PRF:~\$' followed by the command 'users'. The output of the command is 'acamp0s root', indicating that the user is now root.

```
acamp0s@BCNLT5CG3284PRF:~$ users
acamp0s root
```

### 2.5.3 Disable root user

```
$ sudo passwd -dl root
```

```
acampes@BCNLT5CG3284PRF:/home$ sudo cat /etc/shadow
root:!:19635:0:99999:7:::
daemon:!:19478:0:99999:7:::
bin:!:19478:0:99999:7:::
root:!:19478:0:99999:7:::
```

## 3 Basic Linux Commands

### 3.1 Linux System Information

**Prints information about your machine's kernel, name, and hardware**

**The order is:** kernel name, network node hostname, kernel release, kernel version, machine hardware name, processor type, hardware platform and os.

```
$ uname -a
```

**To check Linux Distro**

```
$ uname -or
```

**To know more about Linux Distro**

```
$ cat /etc/os-release
```

```
$ cat /etc/lsb-release
```

```
$ lsb_release -a
```

```
$ hostnamectl
```

### 3.2 Linux System Consumption

**Displays running processes and the system's resource usage**

```
$ top
```

```
$ htop
```

**Displays the system's overall disk space usage**

```
$ df -h
```

**Displays a folder/file disk space usage**

```
$ df -h folder_name
```

**Checks a file or directory's storage consumption**

```
$ du -h directory_name
```

## NOTE

-h flag shows an humanize output, it specifies K, M, G, etc. If you want to have the output in Kibibyte do not use the flag.

## 3.3 Linux Services

### 3.3.1 ps

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use top instead. **Display a snapshot of the currently running processes**

```
$ ps waux
```

- **w**: display the full command line of each process
- **a**: list processes from all users
- **u**: detailed information about each process
- **x**: list processes without controlling terminals

### 3.3.2 systemctl

**Systemctl** manages systemd services and other units such as sockets, devices, mount points, and timers in Linux systems. It is a powerful command-line tool used to query or send control commands to the system manager.

**Systemd** is a system and service manager for Linux operating systems

#### Start a Service

```
$ systemctl start <service_name>
```

#### Stop a Service

```
$ systemctl stop <service_name>
```

#### Check the status of a Service

```
$ systemctl status <service_name>
```

#### Restart a Service

```
$ systemctl restart <service_name>
```

#### Reload configuration of a Service

```
$ systemctl reload <service_name>
```

#### Enable a service to start on Boot

```
$ systemctl enable <service_name>
```

#### Disable a service to start on Boot

```
$ systemctl disable <service_name>
```

### 3.4 Service

The **service** command is commonly used in older versions of Linux distributions or those that haven't fully adopted systemd yet. It provides a simpler interface compared to **systemctl**, but it offers similar functionality for starting, stopping, restarting, and checking the status of services.

It is very similar to **systemctl**, with almost same functionalities, just reversing arguments

#### Start a Service

```
$ service <service_name> start
```

#### Check the status of a Service

```
$ service <service_name> status
```

### 3.5 Moving between directories

#### 3.5.1 Introduction

- **pwd**: which directory we are
- **cd == cd ~** : go to users \$HOME
- **cd /** : go to /
- **cd .** : does not do anything
- **cd ..** : go one directory back
- **ls -l** : list all directory content (not hidden)
- **ls -la** : list all directory content (hidden or not)

#### 3.5.2 ls -la Analyzing Results

The first character on the left represents the type, possible values for this position are as follows:

- **-**: file
- **d**: directory
- **l**: symbolic link
- **b**: binary (executable file)

The following 9 represent the file permissions and should be viewed in groups of 3.

- Los **first 3** owner permissions



- Los **3 in the middle** group permissions
- Los **last 3** rest of the world permissions
- **r**: read
- **w**: write
- **x**: execute

#### Note

The `/etc/group` is a text file which defines the groups to which users belong under Linux and UNIX operating system. Under Unix / Linux multiple users can be categorized into groups. Unix file system permissions are organized into three classes, user, group, and others. The use of groups allows additional abilities to be delegated in an organized fashion, such as access to disks, printers, and other peripherals

### 3.6 chmod

You may need to know how to change permissions in numeric code in Linux, so to do this you use numbers instead of "r", "w", or "x". Cause Basically, you add up the numbers depending on the level of permission you want to give.

- **0**: No Permission
- **1**: Execute
- **2**: Write
- **4**: Read

#### Examples:

- To give read, write, and execute permissions for everyone:

```
$ chmod 777 folder/file_name
```

- To give read, write, and execute permissions for the owner user only.

```
$ chmod 700 folder/file_name
```

- To give write and execute (3) permission for the owner user, w (2) for the group, and read, write, and execute for the rest of users.

```
$ chmod 321 foldername
```

Permiso	Valor	Descripción
rw- -- - --	600	El propietario tiene permisos de lectura y escritura
rw- --x --x	711	El propietario lectura, escritura y ejecución, el grupo y otros solo ejecución
rw- r-x r- --x	755	El propietario lectura, escritura y ejecución, el grupo y otros pueden leer y ejecutar el archivo
rw- rw- -- rw-	777	El archivo puede ser leído, escrito y ejecutado por quien sea
r- -- -- --	400	Solo el propietario puede leer el archivo, pero ni el mismo puede modificarlo o ejecutarlo y por supuesto ni el grupo ni otros pueden hacer nada en el
rw- r- -- --	640	El usuario propietario puede leer y escribir, el grupo puede leer el archivo y otros no pueden hacer nada

Figure 1: Permissions

### 3.7 chown

The **chown** command allows you to change the user and/or group ownership of a given file, directory, or symbolic link.

In Linux, all files are associated with an owner and a group and assigned with permission access rights for the file owner, the group members, and others.

```
$ chown [OPTIONS] USER[:GROUP] FILE(s)
```

**USER** is the user name or the user ID (UID) of the new owner. **GROUP** is the name of the new group or the group ID (GID). **FILE(s)** is the name of one or more files, directories or links.

#### NOTE

Numeric IDs should be prefixed with the + symbol.

For example, the following command will change the ownership of a file named file1 to a new owner named linuxize:

```
$ chown linuxize file1
```

**To change recursively**

```
$ chown -R linuxize /var/www/
```

#### NOTE

If the directory contains symbolic links you should add the flag -h

```
$ chown -hR linuxize /var/www/
```

**Using a Reference File** The **--reference=ref\_file** option allows you to change the user and group ownership of given files to be same as those of the specified reference file (**ref\_file**).

```
$ chown --reference=REF_FILE FILE
```

### WARNING

If the reference file is a symbolic link chown will use the user and group of the target file.

## 3.8 Ctrl+R

With this command we can search in the history the past commands we have run on the machine.

Be carefull using Ctrl+R, because for example if you click on tab, you will crash it:

- **Intro:** to execute the command it is showing you
- **Ctrl+R:** to go to the command behind that matches the pattern

## 3.9 Mkdir (create files)

- `mkdir -p`: create a directory recursively, with different subfolders (`mkdir -p first/second/third...`)
- `mkdir -m a=rwx`: create a directory which files are going to have the permissions specified
- `rm -rf <file>`: remove recursively the file
- `rm *.txt`: remove all the files which contains the string "txt" on the current directory

## 3.10 Mv / Cp

**To move / copy all the content from one folder to another:**

```
$ cp -R path/to/source/* /path/dest/folder/
```

```
$ mv -R path/to/source/* /path/dest/folder/
```

**To move / copy one folder to another:**

```
$ cp -R path/to/source/ /path/dest/folder/
```

```
$ mv -R path/to/source/ /path/dest/folder/
```

**To copy / move multiple directories on Linux**

```
$ cp -R path/to/source/ /path/dest/folder/
```

```
$ mv -R path/to/source1/ /path/dest/folder2/ ... path/to/sourcen/ /path/dest/foldern/
```

### NOTE

Both directories should exist

### 3.11 Simbolik Links

To create it we should be on the folder where we want to create it and make reference to the folder where we want to point, the reference can start from this folder ( `./` ) or from `/`.

```
$ ln -s /Users/titocampis/Desktop/TFG/Proyecto TFG
```

### 3.12 Aliases

To know all the aliases we have in our system

```
$ alias
```

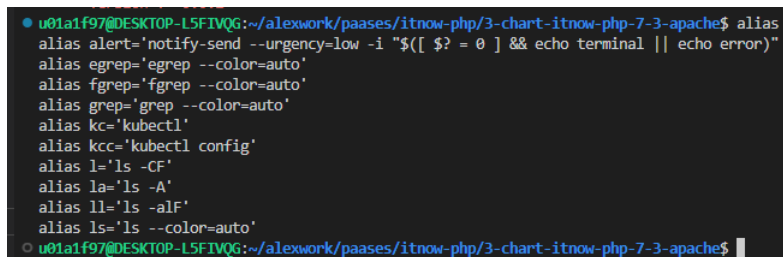
A terminal window screenshot showing the output of the 'alias' command. The prompt is 'u01a1f97@DESKTOP-L5FTVQG:~/alexwork/paases/itnow-php/3-chart-itnow-php-7-3-apache\$'. The output lists several aliases: 'alias alert=\'notify-send --urgency=low -i "[ \$? = 0 ] && echo terminal || echo error\'"', 'alias egrep=\'egrep --color=auto\'', 'alias fgrep=\'fgrep --color=auto\'', 'alias grep=\'grep --color=auto\'', 'alias kc=\'kubect1\'', 'alias kcc=\'kubect1 config\'', 'alias l=\'ls -CF\'', 'alias la=\'ls -A\'', 'alias ll=\'ls -aF\'', and 'alias ls=\'ls --color=auto\''. The prompt at the bottom is 'u01a1f97@DESKTOP-L5FTVQG:~/alexwork/paases/itnow-php/3-chart-itnow-php-7-3-apache\$'.

Figure 2: Default aliases

#### 3.12.1 Ephemeral Aliases

To generate an ephemeral alias:

```
$ alias <alias_name>="command arg1 arg2 ..."
```

##### Examples

```
$ alias docker=podman
```

```
$ alias kcc="kubect1 config"
```

#### 3.12.2 Permanent Aliases

To maintain alias in our system and do not lose them when we change of terminal we need to configure them on the shell config file we use:

- Bash → `~/.bashrc`
- ZSH → `~/.zshrc`
- Fish → `~/.config/fish/config.fish`

```
#Mis alias personalizados
alias imágenes="cd /home/sapoclay/Imágenes/"
alias actualizaristema="sudo apt update && sudo apt upgrade"
alias pingxbmc="ping 192.168.1.100"
```

Figure 3: Alias in permanent files

### 3.13 Source

```
$ source filename [arguments]
```

When you execute a command or a script from a shell, a subprocess (child process) of the shell is created to execute the command or script (parent process).

If the script that executes the child process creates or modifies any environment variable, those changes or variables disappear when the command or script finishes.

If we want those changes to remain, we can use the Bash command `source`. This command causes the process or command to execute without creating any child process, so that changes made to environment variables and others are maintained when the file finishes.

More information: [Source in Bash](#)

**Example:**

```
script.sh

#!/bin/bash

export RHT_OCP4_DEV_USER="nwmwsv"
export RHT_OCP4_DEV_PASSWORD="1ecc25feca97466e81c5"
export RHT_OCP4_MASTER_API="https://api.eu410.prod.nextcle.com:6443"
export Console_Web_Application=\
"https://console-openshift-console.apps.eu410.prod.nextcle.com"
export Cluster_Id="5650752a-edc7-4546-a1ff-8900d7e8e35b"
```

1. If we execute it with `./`

- First, we need to give it permissions.

```
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ ll d280lab.sh
-rw-r--r-- 1 u01a1f97 u01a1f97 319 Dec  5 08:20 d280lab.sh
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ ./d280lab.sh
bash: ./d280lab.sh: Permission denied
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ chmod 744 d280lab.sh
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ ./d280lab.sh
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$
```

- Secondly, as we anticipated, the changes are not saved in the terminal, since a subprocess is executed in the background which never merges system-level changes into the parent process.

```
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ ./d280lab.sh
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ echo $RHT_OCP4_DEV_USER
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$
```

2. If we execute it with `bash/sh`

- In this case, permissions are not necessary, as it is passed as an argument to the bash executable.
- Secondly, as we anticipated, the same thing happens as with './', the changes are not saved in the terminal (parent process).

```

• u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ ll d280lab.sh
-rw-r--r-- 1 u01a1f97 u01a1f97 319 Dec  5 08:20 d280lab.sh
• u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ bash d280lab.sh
• u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ echo $RHT_OCP4_DEV_USER

○ u01a1f97@DESKTOP-L5FIVQG:~/alexwork$

```

### 3. If we execute it with source

- In this case, permissions are not necessary, as it is passed as an argument to the source executable.
- Secondly, as we anticipated, the changes are transmitted to the parent process! And therefore they are saved in our terminal.

```

• u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ ll d280lab.sh
-rw-r--r-- 1 u01a1f97 u01a1f97 319 Dec  5 08:20 d280lab.sh
• u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ source d280lab.sh
• u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ echo $RHT_OCP4_DEV_USER
rwmwsv
• u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ echo $RHT_OCP4_DEV_PASSWORD
1ecc25feca97466e81c5
○ u01a1f97@DESKTOP-L5FIVQG:~/alexwork$

```

## 3.14 Shutdown machines or terminals

- Shutdown the linux machine now

```
$ sudo shutdown -r now
```

- Just shutdown the terminal

```
$ exec "$SHELL"
```

## 3.15 Others

To create a new empty file

```
$ touch main.py
```

Checks a file's type

```
$ file main.py
```

Prints command outputs in Terminal and a file

```
$ echo "Hello" | tee output.txt
```

[More basic interesting linux commands](#)

## 4 .bashrc, .bash\_profile, /etc/bashrc & path

### 4.1 .bashrc vs .bash\_profile

- **.bashrc [non interactive login]:** for changes to take effect, we need to open another interactive terminal (changes don't apply to the current one because this file is executed upon starting a new terminal session).
- **.bash\_profile [interactive login]:** it only executes when there's a login process, upon restarting the machine, when using ssh, sudo... But not when opening a new terminal.

### 4.2 PATH

**PATH** is an environmental variable in Linux that **tells the shell which directories to search for executable files** (i.e., ready-to-run programs) in response to commands issued by a user.

When you type a command into the command prompt in Linux, all you're doing is telling it to run a program. Even simple commands, like `ls`, `mkdir`, `rm`, and others are just small programs that usually live inside a directory on your computer called `/usr/bin`. Other places to search for executables: `/usr/local/bin`, `/usr/local/sbin`, and `/usr/sbin`.

When you type a command into your Linux shell, it doesn't look in every directory to see if there's a program by that name. It only looks to the ones you specified in the `$PATH` environment var.

Sometimes, you may wish to install programs into other locations on your computer, but be able to execute them easily without specifying their exact location. You can do this easily by adding a directory to your `$PATH`.

View your PATH

```
$ echo $PATH
```

### 4.3 Set your PATH

#### 4.3.1 For current terminal

```
$ export PATH=$PATH:/place/to/the/binary/file
```

#### 4.3.2 Permanently for interactive sessions

But what happens if you restart your computer or create a new terminal instance? Your addition to the path is gone! This is by design. The variable `$PATH` is set by your shell every time it launches. The exact way to do this depends on which shell you're running.

Add the following line to `~/.bash_profile`, `~/.bashrc`, or `/.profile`

```
$ export PATH=$PATH:/place/with/the/file
```

#### Nota

Be very careful editing these files, so one error in the configuration of these files can make some binaries crash (example: `mkdir`, `ls`, etc.)

## 5 Useful Tools

### 5.1 Cat

Cat will allow us to view the contents of files without opening them, create files if they do not exist or redirect terminal outputs.

- **Basic usage:** show the content of a file

```
$ cat test
```

- Show the content of both files test and test2

```
$ cat test1 test2
```

- Adds the content of test2 to the file test1, if there was content it overwrites, if there was nothing, it creates the new file

```
$ cat test1 > test2
```

- Appends the content of test2 into the file test1, if there is no test2, it will create it

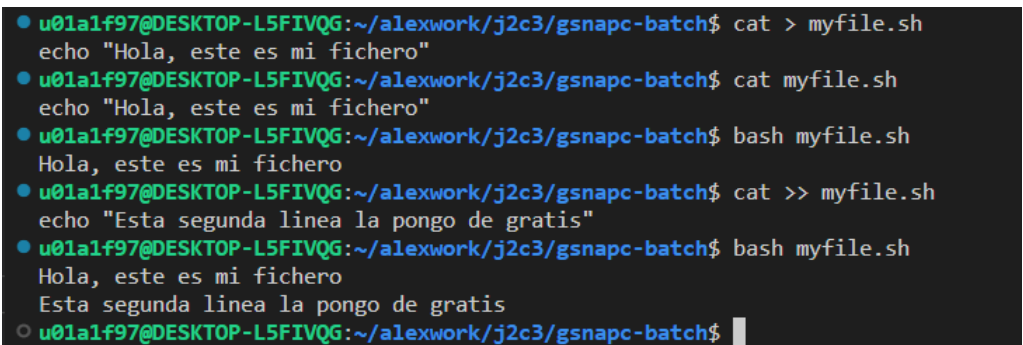
```
$ cat test1 >> test2
```

- Show line number

```
$ cat -n file
```

- Write in a file the content we want, as we were with text editor (Exit: Ctrl+D)

```
$ cat > filename
```



```
u01a1f97@DESKTOP-L5FIVQG:~/alexwork/j2c3/gsnapc-batch$ cat > myfile.sh
echo "Hola, este es mi fichero"
u01a1f97@DESKTOP-L5FIVQG:~/alexwork/j2c3/gsnapc-batch$ cat myfile.sh
echo "Hola, este es mi fichero"
u01a1f97@DESKTOP-L5FIVQG:~/alexwork/j2c3/gsnapc-batch$ bash myfile.sh
Hola, este es mi fichero
u01a1f97@DESKTOP-L5FIVQG:~/alexwork/j2c3/gsnapc-batch$ cat >> myfile.sh
echo "Esta segunda linea la pongo de gratis"
u01a1f97@DESKTOP-L5FIVQG:~/alexwork/j2c3/gsnapc-batch$ bash myfile.sh
Hola, este es mi fichero
Esta segunda linea la pongo de gratis
u01a1f97@DESKTOP-L5FIVQG:~/alexwork/j2c3/gsnapc-batch$
```

Figure 4: amazing cat

### 5.2 Head & Tail

**Head:** print the first 10 lines of each FILE to standard output



- Print the first NUM lines instead of the first 10

```
$ head -n <number_of_lines> <filename>
```

**Tail:** print the last 10 lines of each FILE to standard output.

- Print the last NUM lines instead of the last 10

```
$ head -n <number_of_lines> <filename>
```

- Attach the terminal following the evolution of the file

```
$ head -f <filename>
```

### 5.3 Diff (File comparator)

Command to compare FILES line by line.

**Basic usage**

```
$ diff main.py main_old.py
```

**Report only when files differ**

```
$ diff -q ...
```

**Report when two files are the same**

```
$ diff -s ...
```

**Ignore case**

```
$ diff -i ...
```

**Ignore changes in the amount of white space**

```
$ diff -b ...
```

**Ignore all white space**

```
$ diff -w ...
```

**Ignore changes where lines are all blank**

```
$ diff -B ...
```

**Output NUM (default 3) lines of copied context**

```
$ diff -c ...
```

Output side by side (2 columns)

```
$ diff -y ...
```

## 5.4 Grep (Global search for Regular Expressions and Print out)

### Nota

grep, egrep, fgrep, rgrep - print lines that match patterns. In addition, the variant programs egrep, fgrep and rgrep are the same as grep -E, grep -F, and grep -r, respectively. These variants are deprecated, but are provided for backward compatibility.

### 5.4.1 Two flavors of grep usage

1. Use grep to search text in a file / files

```
$ grep "texto-buscado" <archivo/archivos>
```

2. Use grep to search text in the Terminal STDOUT

```
$ COMMAND [args...] | grep "texto-buscado"
```

The result of this is the occurrences of the pattern (by the line it is on) in the / stdout files. If there is no match, no output will be printed to the terminal.

### Note

The "" are necessary to convert to String. If what we introduce is already a String, they would not be necessary. Example:

- grep alias file.txt: We dont need ""
- grep "alias pepito" file.txt: We need ""

### 5.4.2 Useful Flags

line Number

```
$ grep -n ...
```

Count the number of coincident lines

```
$ grep -c ...
```

IgnoreCase

```
$ grep -i ...
```

Execute in silent mode, suppress all normal output

```
$ grep -q ...
```

**Accept extended regular expressions**

```
$ grep -E ...
```

**Use PATTERNS for matching, you can specify multiple patterns to search for**

```
$ grep -e 'pattern1' -e 'pattern2' file.txt
```

**Match only full words**

```
$ grep -w
```

**Match only full lines**

```
$ grep -x
```

**Print non-matching lines**

```
$ grep -v
```

**Before Context or Aftercontext**

```
$ grep "text" file -A NUMBER_A -B NUMBER_B
```

**Recursive Search** By default, grep cannot search for directories. If you try to do so, you will get an error ("It is a directory"). With the -R option, searching for files across directories and subdirectories becomes possible.

```
$ grep -R
```

**Print only names of FILES with selected lines**

```
$ grep -l
```

**Print only names of FILES with no selected lines**

```
$ grep -L
```

## 5.5 Find

Command to search for files in a directory hierarchy.

**Find all files and directories from "/" with the name "linux"**

```
$ find / -name "linux"
```

**Find all files and directories from "/" with the name "linux" and list them**

```
$ find / -name "linux" -ls
```

Find just files"

```
$ find / -type f -name "linux"
```

Find just directories"

```
$ find / -type d -name "linux"
```

Find from our current directory all the files which name contains the substring "Thr"

```
$ find . -name "*Thr*"
```

```
[default@iks03-iksphpitnowr-pre-59f9b86968-hlj2k ~]$ find /var/template/ -name "*envvar*" -ls
 8  4 -rw-r--r--  1 100      1000      423 Jan  4 14:46 /var/template/batch-db3-envvar
 7  4 -rw-r--r--  1 100      1000      423 Jan  4 14:46 /var/template/mysql-db3-envvar
 6  4 -rw-r--r--  1 100      1000      348 Jan  4 14:46 /var/template/postgresql-envvar
 5  4 -rw-r--r--  1 100      1000      600 Jan  4 14:46 /var/template/oracle-envvar
 4  4 -rw-r--r--  1 100      1000      179 Jan  4 14:46 /var/template/mysql-envvar
 3  4 -rw-r--r--  1 100      1000      179 Jan  4 14:46 /var/template/db2-envvar
[default@iks03-iksphpitnowr-pre-59f9b86968-hlj2k ~]$
```

Figure 5: use of find

## NOTE

The "" are optional, because they are just to define strings (process spaces and weird characters as part of the string)

## 5.6 Sed (Stream Editor)

It is a very essential tool for text processing, specifically it is common used for text replacement. Con Sed podemos editar archivos, incluso sin abrirlos, de manera individual o masiva. Dicho sea, que esta forma, es mucho más rápida para encontrar y reemplazar algo en un archivo de manera manual.

Print the content of "fichero.txt" replacing the first occurrence of the string "Microsoft Windows" by "GNU Linux" in the file "fichero.txt"

```
$ sed "s/Microsoft Windows/GNU Linux/" fichero.txt
```

Save the changes in the file:

```
$ sed -i "s/Microsoft Windows/GNU Linux/" fichero.txt
```

Replace all the occurrences of the string "Microsoft Windows" by "GNU Linux" in the file "fichero.txt" and save.

```
$ sed -i "s/Microsoft Windows/GNU Linux/g" fichero.txt
```

Replace just the 3rd occurrence of "Microsoft Windows" by "GNU Linux" in the text and save

```
$ sed "s/Microsoft Windows/GNU Linux/3g" fichero.txt
```

Replace the string "Microsoft Windows" just in line 1:

```
$ sed "1 s/Microsoft Windows/GNU Linux/" fichero.txt
```

#### NOTE

By default sed is **case SENSITIVE** but to change it by case insensitive, you can add **I** at the end of the REGEX:

```
$ sed -i "s/Microsoft Windows/GNU Linux/gI" fichero.txt
```

#### WARNING

Sed will replace all the string matches, without differentiating if it is completed or partial, for example:

```
acampos@BCNLT5CG3284PRF:~$ echo "Alex and Alexandra go to Pedraforca" | sed 's/Alex/Helena/g'  
Helena and Helenaandra go to Pedraforca
```

To replace the exact match you should use spaces, in **REGEX** `space=\s`:

```
$ sed "s/Alex\s/Claudia /g"
```

For more information: [Uso del Comando Sed](#)

## 5.7 AWK

Powerful text processing tool in Linux and Unix environments. It's primarily used for pattern scanning and processing. awk reads input line by line and divides each line into fields (by default, using whitespace as the delimiter). You can then specify patterns and actions to perform on those patterns.

**Basic:** Perform an action in the lines that match the pattern in the file

```
$ awk '/pattern/ { action }' <filename>
```

Print the first field of each line in the file (separated by default by space)

```
$ awk '{ print $1 }' <filename>
```

Print the 3rd field of each line in the file (separated by ":")

```
$ awk -F ":" '{ print $3 }' <filename>
```

Print specific fields of each line in the file (separated by ":")

```
$ awk -F ":" '{ print $2, $3 }' <filename>
```

## 5.8 Tar Command

The tar command on Linux is often used to create .tar.gz or .tgz archive files, or extract them into local directories.

### 5.8.1 Compress

#### Basics

Use the following command to compress an entire directory or a single file on Linux. It'll also compress every other directory inside a directory you specify—in other words, it works recursively.

```
$ tar -czvf name-of-archive.tar.gz /path/to/directory-or-file
```

#### Compress multiple files/directories in a unique archive

```
$ tar -czf FILENAME.tar.gz path/to/dir1 path/to/file1 path/to/dir2 ...
```

#### Exclude files or directories

```
$ tar -czf FILENAME.tar.gz path/to/dir1 \  
--exclude=/path/to/dir1/excludeddir \  
path/to/dir2 --exclude=/path/to/dir1/*.mp4
```

### 5.8.2 Extract

#### Basics

Once you have an archive, you can extract it with the tar command. The following command will extract the contents of archive.tar.gz to the current directory.

```
$ tar -xzvf archive.tar.gz
```

#### Choose the extract directory

You may want to extract the contents of the archive to a specific directory. You can do so by appending the -C switch to the end of the command:

```
$ tar -xzvf archive.tar.gz -C /path/to/directory/to/extract
```

More information in: [How-To Geek tar](#)

Here's what those switches actually mean:

- **-c:** create an archive
- **-z:** compress into gzip
- **-v:** verbose [removable]
- **-f:** allows you to specify the FILENAME **filename** of archive

### Nota

-f switch must be the last one, because is the flag which specifies the name of the file indicated after.

#### 5.8.3 More flags

- **-x:** Extract the archive
- **-t:** displays or lists files in archived file
- **-u:** archives and adds to an existing archive file
- **-A:** concatenates the archive files
- **-j:** filter archive tar file using tbzip
- **-W:** verify a archive file
- **-r:** update or add file or directory in already existed .tar file

## 5.9 Base64 Encoding

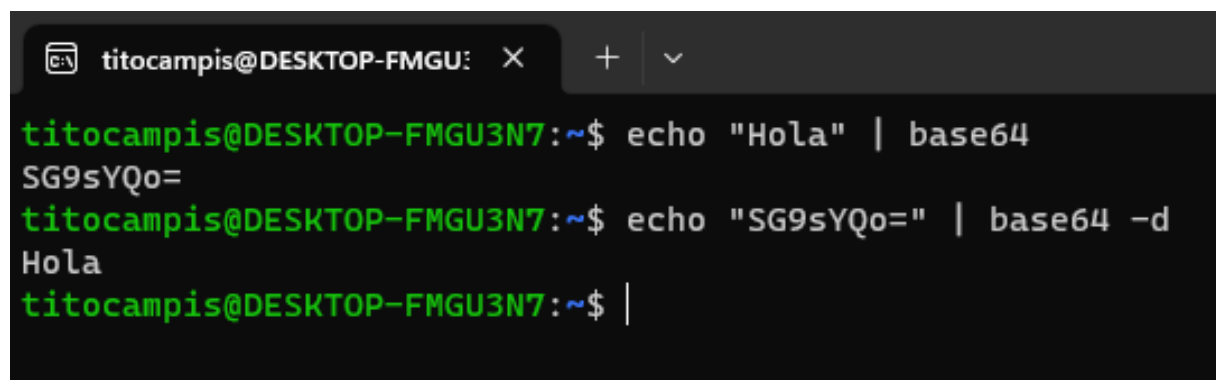
Base64 encode or decode FILE, or standard input, to standard output.

### Code

```
$ echo -n $PASSWORD_CLEAR | base64
```

### Decode

```
$ echo -n $PASSWORD_CODED | base64 -d
```



```
titocampis@DESKTOP-FMGU: X + v
titocampis@DESKTOP-FMGU3N7:~$ echo "Hola" | base64
SG9sYQo=
titocampis@DESKTOP-FMGU3N7:~$ echo "SG9sYQo=" | base64 -d
Hola
titocampis@DESKTOP-FMGU3N7:~$ |
```

## 5.10 JQ

To parse into a beautiful json format:

```
$ cat file.json | jq "."
```

## 6 Linux Default Text Editors

### 6.1 Vi

### 6.2 Vim

#### 6.2.1 Vim Config

```
$ sudo vim ~/.vimrc
```



```

" Disable compatibility with vi which can cause unexpected issues.
set nocompatible

" Enable type file detection. Vim will be able to try to detect the type of file in use.
filetype on

" Enable plugins and load plugin for the detected file type.
filetype plugin on

" Load an indent file for the detected file type.
filetype indent on

" Set authomatic indentation
" set autoindent

" Turn syntax highlightin
syntax on

" Add numbers to each line on the left-hand side.
set number

" Highlight cursor line underneath the cursor horizontally.
set cursorline

" Highlight cursor line underneath the cursor vertically.
set cursorcolumn

" Set shift width to 4 spaces.
set shiftwidth=4

" Set tab width to 4 columns.
set tabstop=4

" Use space characters instead of tabs.
set expandtab

" Colorful (),[],{}
set showmatch

" While searching though a file incrementally highlight matching characters as you type.
set incsearch

" Ignore capital letters during search.
set ignorecase

" Use highlighting when doing a search.
set hlsearch

" Enable auto completion menu after pressing TAB.
set wildmenu

" Make wildmenu behave like similar to Bash completion.
set wildmode=list:longest

" Enable black theme
" set background=dark

" Tab helper
" set smarttab

```

If you want to persist this configuration when using `sudo vim` you will need to do something a little bit tricky:

- Create an alias in your `~/.bashrc` file:

```
alias svim="sudo -E vim"
```

- Use always

```
$ svim <file>
```

### 6.2.2 Hacer y deshacer

**undo:** para deshacer acciones

```
vim editor - Command Mode  
:u
```

**redo:** para volver a hacer acciones

```
vim editor - Normal Mode  
Ctrl + r
```

```
vim editor - Normal Mode  
:r
```

### 6.2.3 Borrar líneas enteras

```
{vim editor - Normal Mode}  
dd + intro
```

### 6.2.4 Buscar Ocurrencias

**Para buscar**

```
{vim editor - Normal Mode}  
/<palabra>
```

**Para ir a cualquier ocurrencia:** Intro

**Para pasar entre ocurrencias:** n

**Si queremos hacerlo insensible a minúsculas y mayúsculas:** antes de buscar, ejecutar lo siguiente.

```
{vim editor - Normal Mode}  
:set ignorecase
```

### 6.3 Nano

## 7 Network

### 7.1 FQDN

El término Fully Qualified Domain Name (FQDN) se refiere a la dirección completa y única necesaria para tener presencia en Internet. Está compuesta por el nombre de host y el de dominio y se utiliza para localizar hosts específicos en Internet y acceder a ellos mediante la resolución de nombres. Este nombre de dominio único que contiene toda la información necesaria para poder acceder a una máquina a través de una red pública, como puede ser internet. A través de un FQDN un equipo es capaz de conectarse con cualquier otro.

La estructura del FQDN viene determinada por el sistema de nombres de dominio (DNS) y está conformada por etiquetas. Cada etiqueta se corresponde con el nombre de un nivel en el espacio de nombres de dominio y está separada de la siguiente por un punto. Ha de constar de entre 1 y 63 caracteres, que pueden ser números, letras y guiones (aunque realmente los guiones no se pueden usar al comenzar una etiqueta) y el total de caracteres del FQDN no debe exceder los 255.

El Fully Qualified Domain Name consta como mínimo de tres etiquetas: el dominio de nivel superior, el nombre de dominio y el nombre de host. Se lee al revés.

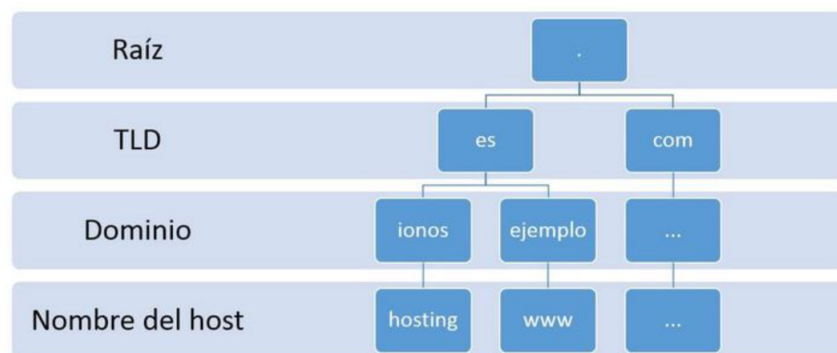


Figure 6: Representación esquemática de la estructura del Fully Qualified Domain Name

#### Ejemplo de un FQDN

[Nombre de host].[Dominio].[TLD].[Raíz]

hosting.ionos.es

1. La etiqueta del dominio raíz tras el punto permanece vacía
2. El **dominio de primer nivel**: en nuestro ejemplo es ".es", dominio de nivel superior geográfico, también conocido por las siglas ccTLD (country code-top level domain). Frente a ellos se encuentran los TLD genéricos como .com o .org, también designados gTLD (generic top-level domain).
3. El **dominio de segundo nivel**, también conocido como nombre de dominio. En el ejemplo se corresponde con "ionos".
4. El **dominio de tercer nivel**, en el extremo izquierdo, tenemos el nombre del host, en nuestro ejemplo "hosting".

Los FQDN se suelen utilizar en cualquier interacción en Internet, ya que son más fáciles de recordar que las direcciones IP. Otro ejemplo `www.wordpress.com`

Para más información consultad: [FQDN](#) / [FQDN 2](#)

## 7.2 SSH

### 7.2.1 What is SSH?

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network. Secure Shell provides strong password authentication and public key authentication, as well as encrypted data communications between two computers connecting over an open network, such as the internet.

In addition to providing strong encryption, SSH is widely used by network administrators to manage systems and applications remotely, enabling them to log in to another computer over a network, execute commands and move files from one computer to another.

The most basic use of SSH is to connect to a remote host for a terminal session. The form of that command is the following:

- IP:

```
$ ssh user_name@host_IP
```

- Hostname:

```
$ ssh user_name@SSHserve.example.com
```

The SSH key command instructs your system that you want to open an encrypted Secure Shell Connection. `user_name` represents the account you want to access. For example, you may want to access the root user or `acampus` user.

#### NOTE

To connect via ssh using a user, it is needed to have this user created in the host server.

- **host server:** refers to the remote server you are trying to access.
- **client server:** refers to the server you are using to access the host.

For the first time negotiating a connection between the client server and the host server, the user will be prompted with the remote host's public key fingerprint and prompted to connect.

```
The authenticity of host 'sample.ssh.com' cannot be established.  
DSA key fingerprint is 01:23:45:67:89:ab:cd:ef:ff:fe:dc:ba:98:76:54:32:10.  
Are you sure you want to continue connecting (yes/no)?
```

Answering yes to the prompt will cause the session to continue, and the host key is stored in the client server local system's `known_hosts` file. This is a hidden file, stored by default in a hidden directory, called `/.ssh/known_hosts`, in the user's home directory. Once the host key has been stored in the `known_hosts` file, the client system can connect directly to that server again without need for any approvals; the host key authenticates the connection.

```
acampus@CNL15CG3284PRF:~/work/opticlimb$ cat ~/.ssh/known_hosts  
[1]/9rmhuubZ4qqATyc+oET4qS1qBY=|WcLLPZ50ha/xbtLEYwd3oX5e94U= ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEFFvzDJXiUu2iuu8j4cEVQb0H9cSXS4YwvJ4IaPE0I  
[1]Bzdxkx0QRMEmFu4wv/rvtd0qW4=|ygfyp0pkHAScl5bNv7KAHfx108U= ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOx2nnszRk5QD3SCVA+agB7Vx6HRRKZoLwRITweV3eUc  
[1]xQeGrSv4REETw69a7Y010z0X/NQ=|4sJm12fVfJYujzcUJkPLBbU0GA8= ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIT6CTK0N0qT0yGH16KLUL6em6TYEeLbEblOMxSvr1kmI  
[1]8KcWmKTTB0d4FTR+a0WXPxtK3A=|IDLy4NnsUJ4chEQUEmv/NZ86jd0= ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgODUaRNe39+gxSvGiqeN5otQYteMswkaexBYqHYDRkm1xT488c8Se9pWVD  
J/zokG9vUQIz597RAIMRIKHRdgn0EUpigJR3+4627Fa1NY+V0hjS76S3XkuTI3VD08drZ0Kv2KoFLh27TdPHVBw73G67eVsC3+Zhqo9VMGgfk8bpUCswYct5NI7o0VC3UPYLL6NmQ2PcdhLDAi6Esvqjay6L  
Gr92PD5JfV0+zUd2t0n5Uk4oLc07Yt5h04b6ytOKNydmh2VaRYJCqg/IdCxSg1j5B2HmZJImJFdemuiGLym6PJm1Wx9JnPKtLAnaWUQ3/2+/zid8W5wiXyC+BfKfs/G69t7AJVDJ0b7PeKw6E7KETabbwGz  
T1fLXiePPs8UkGSyqzS3urStqyzQ88qDVtxPqKwxKsM2pJ483fbVn/e5dfx/0mNuHxJtAzNnsQxPJ+Z053jz0ji8B1g27rm7cvGjVa7SruRpULt69p9jgnjKjZGBXWNJf2i/50KrrbZIIjsAc=  
[1]B2iLUO/qmD5TLV9HgE55a4Bwo=|CQixy4AXDHMFouFMBST5MgBBges= ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAYNTYAAAAIbmlzdHAYNTYAAABBBP/mJkG+VEIPYxRfZA/1WW  
noiMVvAeyawseNXyhr3++kpNmMGEbyEu0KcJBDGTmQj00rMbHhFvPipjbeDppFW4=  
[1]qEPdg8lgCj14tW2Ut0BoT8HRXZ0=|wEecMAGcXcUILLH9Qszjcv7B/cmg= ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAYNTYAAAAIbmlzdHAYNTYAAABBBBLJXQm24i4gw0lfMNHJjtYA  
iym3dsN0cqfRymYcJr2LE2HoThierLJF0unoCvVbL0qU7eukvHMD/euVjZ1q/A0=  
[1]Qn5XbJFL0m+B+BB68K3FYLfNqF0=|9Xtv+hhmS6xNuSG0spzdV3fLzz= ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPcsdudljzZEFQMTcN/tlug/ePhi+PvEnDNf6mZcStde  
[1]LxrxehRUjYopfqzNPv/4ZfBFqQ=|2dgVAaRBD0LLy+AgGGR+1yFouhA= ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKGDU4gq4CXHYzBjn5R3RjPvpwph9ykv27j03wf/QQJsm  
[1]soAYHGrrkoPjFTHth/jmUMAtLpc=|2A+I7E0UtKpotZMqfMAv4IcTpII= ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIIsLePk3hZW8YPLHkn7H8HVaXkd+Own6S6mD0V2nZF  
[1]S9ACTDvCiQCRw7jJ/+2/cWfQe=|ks/xJmQ7UVi9JuIQmzSBZsUL580= ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAYNTYAAAAIbmlzdHAYNTYAAABBBOK+WvgdSx6WIM/vpolFGTh  
d4KVsuxXaQ8k9KraYfXp8StkUEzPGFmpT8YgclLqEkcnjxwb+et7eXJ2VnQe/d3k=
```

Present in all data centers, SSH ships by default with every Unix, Linux and Mac server. SSH connections have been used to secure many different types of communications between a local machine and a remote host, including secure remote access to resources, remote execution of commands, delivery of software patches, and updates and other administrative or management tasks, manage routers, server hardware, virtualization platforms, operating systems (OSes), and inside systems management and file transfer applications.

Secure Shell is used to connect to servers, make changes, perform uploads and exit, either using tools or directly through the terminal. SSH keys can be employed to automate access to servers and often are used in scripts, backup systems and configuration management tools.

#### NOTE

SSH operates on **TCP port 22** by default (though SSH port can be changed if needed). The host server listens on port 22 (or any other SSH assigned port) for incoming connections. It organizes the secure connection by authenticating the client and opening the correct shell environment if the verification is successful.

It is hardly recommended to change the port in which the host server listens, because to let port 22 is more easily to enter the machine without authorization.

### 7.2.2 Authenticate SSH connections using user-password

By default SSH protocol doesn't use public key cryptography for authenticating hosts and users. It still uses the **Linux user-password authentication method**, so you only have to pass the **username** of the Linux user via the ssh command and introduce the **password** of this user:

```
$ ssh -p custom_port linux_username@hostname_or_ip
```

#### NOTE

If you run the command without specifying user, ssh will take the client server Linux current **username**.

### 7.2.3 Authenticate SSH connections using Asimetric Key authentication method

Generally, the SSH service is configured to use Asimetric Key cryptography for authenticating hosts and users. SSH introduced Asimetric Key authentication as a more secure alternative to the older **.rhosts** authentication. It improved security by avoiding the need to have password stored in files, and eliminated the possibility of a compromised server stealing the user's password.

However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to user names and passwords. They should have a proper termination process so that keys are removed when no longer needed.

#### 7.2.3.1 Generate public and private keys on the client server side

For that reason, we should generate SSH public and private key on our client and authorize them into the host.

Depending on the technology of the host you want to connect, you need to create the keys:

- **rsa**: an old algorithm based on the difficulty of factoring large numbers. A key size of at least 2048 bits is recommended for RSA; 4096 bits is better. All SSH clients support this algorithm.

```
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- **ed25519**: this is a new algorithm added in OpenSSH. Support for it in clients is not yet universal.

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

## WARNING

For SSH to recognize the SSH keys they should be stored in:

```
~/.ssh/
```

```
acampos@BCNLT5CG3284PRF:~/work/opticlimb$ ls -la ~/.ssh/
total 24
drwxr-xr-x  2 acampos acampos 4096 Oct  4 14:02 .
drwxr-x--- 11 acampos acampos 4096 Oct  4 11:49 ..
-rw-----  1 acampos acampos 3369 Oct  4 10:51 id_rsa
-rw-r--r--  1 acampos acampos  733 Oct  4 10:51 id_rsa.pub
-rw-----  1 acampos acampos 2798 Oct  4 13:49 known_hosts
-rw-----  1 acampos acampos 2576 Oct  4 13:49 known_hosts.old
```

Also, **rsa keys** should have the **permissions** you see in the image above, if not, it will not be valid to authenticate in any server. If you have generated the keys, they are generated with the correct permissions, but if you have copied, it can be wrong.

For more information, check the documentation of [how to do if for github servers](#).

### 7.2.3.2 Upload the Public Key into the `authorized_keys` of the host server

To use public key authentication, the public key must be **installed in the** `authorized_keys` file of the server. This can be conveniently done using the `ssh-copy-id` tool. Like this:

```
$ ssh-copy-id -i ~/.ssh/my_ssh_key.pub user_name@hostname
```

## WARNING

With the command above, we will append the host **pubkey** in the `authorized_keys` file of the server.

The authorized keys to enable an specific user accessing the server through ssh are stored in:

```
~/.ssh/authorized_keys
```

So you can copy your public key directly to this directory inside the server.

## NOTE

For git you should do it different, just adding your public ssh key in the GUI, accessing to your profile settings. You can follow the [Official Documentation](#)

### 7.2.3.3 Configure the ssh service into the host server

To enable the SSH connection, the host server we want to connect should have the ssh service running and configured.

We can check if the sshd (ssh daemon) service is running. SSH Daemon listens for incoming SSH connections and handles authentication, encryption, and communications:

```
$ systemctl status sshd
```

#### WARNING

So not confuse it with the ssh service, which is the one used on the client-side to establish a secure connection to an SSH server.

The configuration file for the SSH Daemon is:

```
/etc/ssh/sshd_config
```

As we said before, by default SSH protocol doesn't use public key cryptography for authenticating hosts and users. So to enable it, we will need to modify this file in order to decomment the line:

```
/etc/ssh/sshd_config  
PubkeyAuthentication yes
```

And then restart the sshd service:

```
$ systemctl restart sshd
```

#### NOTE

As well you can disable the password authentication:

```
/etc/ssh/sshd_config  
PasswordAuthentication no
```

To see all the configurable parameters in the `/etc/ssh/sshd_config` you can check

- [All configurable parameters in sshd file](#)
- Your template in ansible

### 7.2.3.4 Use the Asimetric Key authentication method

By default, when you try to stablish ssh connection, if the host server enables PubkeyAuthentication, your ssh service will try to authenticate with all this pub keys:

- `~/.ssh/id_ecdsa.pub`
- `~/.ssh/id_ecdsa_sk.pub`
- `~/.ssh/id_ed25519.pub`



- ~/.ssh/id\_ed25519\_sk.pub
- ~/.ssh/id\_xmss.pub
- ~/.ssh/id\_xmss\_sk.pub
- ~/.ssh/id\_dsa.pub
- ~/.ssh/id\_rsa.pub

If you want to enforce the client server ssh service use just one, or a custom one different from those shown, you should:

```
$ ssh -p <Port_Number> -i /path/to/the/priv_key
```

#### NOTE

You should have both keys on the same folder, public and private, if not, it will fail.

### 7.2.4 SSH Extra Arguments

Custom Port (by default is 22)

```
$ ssh -p <Port_Number>
```

Use specific pub\_key to authenticate

```
$ ssh -i /path/to/the/priv_key ...
```

Disable Check on the hosts file for this host

```
$ ssh -o StrictHostKeyChecking=no ...
```

Verbose Mode

```
$ ssh -v ...
```

Set timeout

```
$ ssh -o ConnectTimeout=30 ...
```

Using Proxy to establish the connection

```
$ ssh -o ProxyCommand="ssh -p <Proxy_Port_Number> -W %h:%p \
-q <username>@<proxy_hostname_or_ip>"
```

### 7.2.5 SFTP

FTP, or "File Transfer Protocol" was a popular unencrypted method of transferring files between two remote systems.

SFTP, which stands for SSH File Transfer Protocol, or Secure File Transfer Protocol, is a separate protocol packaged with SSH that works in a similar way but over a secure connection. The advantage is the ability to leverage a secure connection to transfer files and traverse the filesystem on both the local and remote system.

In almost all cases, SFTP is preferable to FTP because of its underlying security features and ability to piggy-back on an SSH connection. FTP is an insecure protocol that should only be used in limited cases or on networks you trust.

### Basic Usage

```
$ sftp -P <Port_Number> <username>@<hostname_or_ip>
```

### Set timeout

```
$ sftp -o ConnectTimeout=30 ...
```

You will be asked to insert the password if the sshd service is configured with password, so there is a way to insert it without being asked.

### Send the password without interaction

```
$ sshpass -p "your_password" sftp -P ...
```

## 7.3 Localhost Networking

### 7.3.1 Ports Exposed

Para saber que puertos tiene escuchando nuestro localhost:

#### 1. Using lsof

```
$ sudo lsof -i -P -n
```

```
$ sudo lsof -i -P -n | grep -i listen
```

#### 2. Using netstat

```
$ netstat -putona | grep numero-de-puerto
```

- **p:** muestra las conexiones para el protocolo especificado que puede ser TCP o UDP
- **u:** lista todos los puertos UDP
- **t:** lista todos los puertos TCP
- **o:** muestra los timers
- **n:** muestra el numero de puerto
- **a:** visualiza todas las conexiones activas del sistema

### NOTE

`netstat` command shows the system's network information, like routing and sockets.

### 7.3.2 Localhost Addresses

#### De puertas para adentro

##### 1. Ipv4:

- localhost
- 127.0.0.1

##### 2. Ipv6

- 0:0:0:0:0:0:1
- ::1

#### De puertas para fuera

- `ifconfig`: displays the system's network interfaces and their configurations. In this case our localhost direction to outside is 172.21.220.125/20 (172.21.220.125 with mask 255.255.240.0)

```
$ ifconfig
```

```
titocampis@DESKTOP-FMGU3N7:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.21.220.125 netmask 255.255.240.0 broadcast 172.21.223.255
    inet6 fe80::215:5dff:feab:2d48 prefixlen 64 scopeid 0x20<link>
    ether 00:15:5d:ab:2d:48 txqueuelen 1000 (Ethernet)
    RX packets 63 bytes 8021 (8.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7 bytes 586 (586.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 7: `ifconfig`

- `ip -br addr list eth0`

```
titocampis@DESKTOP-FMGU3N7:~$ ip -br addr
lo          UNKNOWN    127.0.0.1/8 ::1/128
bond0       DOWN
dummy0      DOWN
tunl0@NONE  DOWN
sit0@NONE   DOWN
eth0        UP             172.21.220.125/20 fe80::215:5dff:feab:2d48/64
titocampis@DESKTOP-FMGU3N7:~$
```

Figure 8: `ip -br`

## Nota

### *Netmask*

24 (number of 1's):

255.255.255.0

11111111 11111111 11111111 00000000

20 (number of 1's):

255.255.240.0

11111111 11111111 11110000

## 7.4 Certificates

## 8 Network Tools

### 8.1 Ping

### 8.2 Curl

curl is a tool for transferring data from or to a server. It supports these protocols: DICT, FILE, FTP, FTPS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, ... TELNET or TFTP. The command is designed to work without user interaction.

#### Without using proxy

```
$ curl --noproxy "*"
```

#### Using proxy

```
$ curl -x proxy_url conn_url
```

#### With specific timeout in seconds

```
$ curl --connect-timeout 5
```

#### Not show content, only info

```
$ curl -I
```

#### Force the protocol

```
$ -0,      --http1.0      Use HTTP 1.0
           --http1.1      Use HTTP 1.1
           --http2        Use HTTP 2
           --http2-prior-knowledge Use HTTP 2 without HTTP/1.1 Upgrade
           --http3        Use HTTP v3
```

#### Target different port than 443 or 8080

```
$ curl domain:port
```

#### Verbose

```
$ curl -v
```

#### HTTPS Protocol

```
$ curl https://hostname
```

#### Allow insecure connections, skip certificate validation

```
$ curl -k hostname
```

## Authentication

```
$ curl -u user_name:pwd hostname
```

## Pass Headers

```
$ curl -H "Content-Type:application/json" hostname
```

## GET HTTP Requests

```
$ curl -XGET -H hostname
```

## PUT HTTP Requests

```
$ curl -XPUT hostname
```

## POST HTTP Requests

```
$ curl -XPOST hostname
```

## DELETE HTTP Requests

```
$ curl -XDELETE hostname
```

## 8.3 Nmap

## 8.4 Netcat

## 8.5 Telnet

### Netcat

```
$ nc -zv hostname port
```

### Telnet

```
$ telnet hostname port
```

## 8.6 Proxy variables definition for binaries

Configuring the proxy in the system environment variables with a specific naming convention will cause many system binaries to use them to establish their connections. For example: docker, curl, nslookup...

Include the following variables in the file `/etc/bash.bashrc`:

```
# Configure Default Proxy
export http_proxy=http://yourproxyserver.com
export https_proxy=http://yourproxyserver.com
export HTTP_PROXY=http://yourproxyserver.com
export HTTPS_PROXY=http://yourproxyserver.com
export NO_PROXY=localhost
export no_proxy=localhost
```

## 9 Tmux

Tmux, short for "Terminal Multiplexer," is a powerful command-line tool that enhances the capabilities of your terminal sessions. It allows you to organize multiple terminal windows or sessions within a single terminal window, effectively turning your terminal into a multi-pane workspace.

With Tmux, you can create multiple independent terminal sessions within a single window. This means you can work on different tasks or projects simultaneously without cluttering your desktop with multiple terminal windows.

As well tmux enable separate the execution from terminal, so if you lose the connection, what you launched continues executing.

### 9.1 Arguments

#### Configuration File

The configuration file is passed through arguments:

```
$ tmux -f /path/to/file
```

What we can do is generate a file in `/tmp/custom-tmux.conf` as follows: The configuration file is passed through arguments:

```
/tmp/custom-tmux.conf

set -g history-limit 10000
setw -g mode-keys vi

# MAJOR ops
set-option -g prefix C-a
unbind C-b
bind-key C-a last-window

# Set status bar
set -g status-left-length 30
set -g status-bg black
set -g status-fg white
set -g status-left "[fg=green]#H"
set -g status-right "[fg=yellow]#(uptime | awk -F\ : '{print $5}')"
setw -g monitor-activity on
set -g visual-activity on
```

In tmux, when we want to go to **command mode**, by default we need to use **Ctrl+b**. But we are going to configure it as **screen**, with **Ctrl+a**, it is configured in the config file.

**-2**

tells Tmux to start in 256-color mode. Tmux supports both 256-color and 16-color modes for terminal color support. The **-2** option enables the 256-color mode.

```
$ tmux -f /path/to/file -2
```

**-L [socket\_name]**

Sets the name of the socket used by the tmux server. The socket is a communication endpoint that allows



different processes to communicate with each other, in this case, allowing the tmux client (the terminal you're using) to communicate with the tmux server (the process managing your terminal sessions).

```
$ tmux -f /path/to/file -2 -L test-tmux
```

### attach

Attach to tmux session X, it is used when you want to join an already created tmux session:

```
$ tmux -f /path/to/file -2 -L attach
```

## 9.2 Commands Inside tmux

### General

Command	What it does?
CTRL-a c	New tmux "tab"/session
CTRL-a n	Next "tab"
CTRL-a p	Prev "tab"
CTRL-a #N	Next "tab"
exit	exit current tab
CTRL-d	exit current tab
CTRL-a ,	change current tab name
CTRL-a d	de-attach tmux
CTRL-a &	Destroy current tab

### Sppliting

Command	What it does?
CTRL-a "	New tmux "tab"/session
CTRL-a %	Next "tab"
CTRL-a <arrow-keys>	Prev "tab"
CTRL-a CTRL-<arrow-keys>	Next "tab"
exit	exit current tab
CTRL-a x	exit current tab
CTRL-a CTRL-o	change current tab name
CTRL-a q	de-attach tmux
CTRL-a META-o	de-attach tmux
CTRL-a space &	Destroy current tab

## 10 Others

### 10.1 Docker Desktop

- Docker desktop corre en nuestro Ubuntu desde Windows. Por tanto, si queremos que el Docker de nuestro Ubuntu tenga los certificados necesarios para conectarse a los recursos de nuestra empresa, debemos tener los certificados descargados, actualizados e instalados en nuestro sistema Windows nativo.
- Desde un terminal rellenamos el contenido del fichero de configuración de docker `~/.docker/config.json`:

```
~/.docker/config.json
{
  "auths": {
    "docker-registry.cloud.yourcompany.com": {}
  },
  "credsStore": "desktop.exe",
  "proxies": {
    "default": {
      "httpProxy": "http://urltoproxy",
      "httpsProxy": "http://urltoproxy/"
    }
  }
}
```

- En Docker Engine modificar el fichero que ahí introducimos por el siguiente:

```
{
  "builder": {
    "gc": {
      "defaultKeepStorage": "20GB",
      "enabled": true
    }
  },
  "experimental": false,
  "features": {
    "buildkit": false
  },
  "insecure-registries": [
    "docker-registry.cloud.yourcompany.com"
  ]
}
```

- Ejecutar el comando para logearnos contra el registry que hayamos escogido:

```
$ docker login -u USER -p PASSWORD docker-registry.cloud.yourcompany.com
```

- Probar el acceso al registry:

```
$ docker pull docker-registry.cloud.yourcompany.com/catalog/paas/j
```

## 10.2 Helm

### Compilar

Estando en el directorio que contiene el fichero Chart.yaml:

```
$ helm lint .
```

### Prueba de despliegue Resources en Cluster with file

```
$ helm install -f ./values_ssp.yaml . --debug \
--name-template standalone-11 --dry-run > output.yaml
```

### Prueba de despliegue Resources en Cluster with file

```
$ helm install -f ./values_ssp.yaml . --debug \
--name-template standalone-11
```

## 10.3 JSONPath

JSONPath is a query language for JSON, similar to XPath for XML. AlertSite API endpoint monitors let you use JSONPath in assertions to specify the JSON fields that need to be verified.

### 10.3.1 Basic Usage

A JSONPath expression specifies a path to an element (or a set of elements) in a JSON structure.

#### Dot Notation

```
$.store.book[0].title
```

#### Bracket Notation

```
$["store"]["book"][0]["title"]
```

#### Mix

```
["store"].book[0].title
```

#### Nota

Note that dots are only used before property names not in brackets.

#### Nota 2

The leading "\$" represents the root object or array and can be omitted. For example, *\$.foo.bar* and *foo.bar* are the same, and so are *\$\$[0].status* and *[0].status*.

### Nota 3

Using Bracket Notation be care to put **single quotes**, because double quotes are not accepted:

```
["name"]
```

### Nota 4

JSONPath expressions, including property names and values, are case-sensitive.

## Syntax Elements

- `$`: The root object or array.
- `.property = ["property"]`: Selects the specified property in a parent object.
- `[n]`: Select the n-th element from an array. Indexes are 0-based.
- `[index1,index2,...]`: Selects array elements with the specified indexes. Returns a list.
- `..property`: Recursive descent: Searches for the specified property name recursively and returns an array of all values with this property name. Always returns a list, even if just one property is found.
- `*`: Wildcard selects all elements in an object or an array, regardless of their names or indexes. For example, `address. "*"`  means all properties of the address object, and `book[*]`  means all items of the book array.
- `[:n]`: Selects the first n elements of the array. Returns a list.
- `[-n]`: Selects the last n elements of the array. Returns a list.

### 10.3.2 Filters

We can use some expression to filter the output of the JSON in function of parameters introduced in the syntax.

## Syntax Elements

- `[?(expression)]`: Filter expression. Selects all elements in an object or array that match the specified filter. Returns a list.
- `@`: Used in filter expressions to refer to the current node being processed.

### 10.3.3 Example

#### *JSON Example*

```
{
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
      {
        "category": "fiction",
        "author": "J.R.R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95
    }
  },
  "expensive": 10
}
```

#### Expressions

##### Nota

In all these examples, the leading \$. is optional and can be omitted.

```
$.store.* = ["store"][*]
```

- **Meaning:** All direct properties of store (not recursive).
- **Result:**

```
[
  "book": [
    {
      "category": "reference",
      "author": "Nigel Rees",
      "title": "Sayings of the Century",
      ...
    }
  ]
]
```

```

        },
        { ...
      }
    ],
    "bicycle": {
      "color": "red",
      ...
    }
  }
]

```

```
$.store.bicycle.color = ["store"]["bicycle"]["color"]
```

- **Meaning:** The color of the bicycle in the store.
- **Result:** red

```
$.store..price = \$.price = ..["price"]
```

- **Meaning:** The prices of all items in the store.
- **Result:** [8.95, 8.99, 22.99, 19.95]

```
$.store.book[*] = \$.book[*] = ..["book"][*]
```

- **Meaning:** All books in the store.
- **Result:**

```

[
  {
    "category": "reference",
    "author": "Nigel Rees",
    "title": "Sayings of the Century",
    "price": 8.95
  },
  {
    "category": "fiction",
    "author": "J.R.R. Tolkien"
    ...
  }
]

```

```
$.book[*].title = ..["book"][*]["title"]
```

- **Meaning:** The title of all books in the store.
- **Result:**

```

[
  Sayings of the Century,
  Moby Dick,
  The Lord of the Rings
]

```

```
$.book[0]
```

- **Meaning:** The first book.
- **Result:**

```
[
  {
    "category": "reference",
    "author": "Nigel Rees",
    "title": "Sayings of the Century",
    "price": 8.95
  }
]
```

```
$.book[0].title
```

- **Meaning:** The title of the first book.
- **Result:** Sayings of the Century

```
$.book[0,1].title
```

- **Meaning:** The titles of the first two books.
- **Result:** [Sayings of the Century, Moby Dick]

```
$.book[-1:].title
```

- **Meaning:** The title of the last book.
- **Result:** The Lord of the Rings

```
$.book[?(@.author=="J.R.R. Tolkien")].title
```

- **Meaning:** The titles of all books by J.R.R. Tolkien (exact match, case-sensitive).
- **Result:** It is a list because -n always returns a list. [The Lord of the rings]

```
$.book[?(@.isbn)]
```

- **Meaning:** All books that have the isbn property.
- **Result:** [Moby Dick, The Lord of the Rings]

```
$.book[?!@.isbn]
```

- **Meaning:** All books without the isbn property.
- **Result:** [Sayings of the Century]

```
$..book[?(@.price < 10)].title
```

- **Meaning:** All books cheaper than 10 titles.
- **Result:** [Sayings of the Century, Moby Dick]

```
$..book[?(@.price > \$.expensive)]
```

- **Meaning:** All expensive books.
- **Result:**

```
$ [
  {
    "category": "fiction",
    "author": "J.R.R. Tolkien",
    "title": "The Lord of the Rings",
    "isbn": "0-395-19395-8",
    "price": 22.99
  }
]
```

```
$..book[?(@.category == "fiction" ||
@.category == "reference")].title
```

- **Meaning:** All fiction and reference books titles.
- **Result:** [Sayings of the Century, Moby Dick, The Lord of The Rings]

```
$..book[?(@.category != "fiction")]
```

- **Meaning:** All not fiction books.
- **Result:**

```
[
  {
    "category": "reference",
    "author": "Nigel Rees",
    "title": "Sayings of the Century",
    "price": 8.95
  }
]
```

For more information about JSONPath, check the following [page](#)



### 10.3.4 JSONPath K8s

JSONPath template is composed of JSONPath expressions enclosed by curly braces . Kubectl uses JSONPath expressions to filter on specific fields in the JSON object and format the output. In addition to the original JSONPath template syntax, the following functions and syntax are valid:

- `range, end`: iterate list. `{range .items[*]}{.metadata.name}`
- `{"\n"}, {"\t"}`: to write salto de linea or tab. `{..metadata.name}{"\t"}{..matadata.image}`

#### Example 1

Get all **containers** / **initContainers** running and their images used inside a **Pod** or declared in pod.spec of **Deployment/Statefulset** . It is exactly the same changing little things. At least, it is showed in column view separated by tab.

```
$ kc get RESOURCE_TYPE RESOURCE_NAME \
-o jsonpath="{range ..containers[*]}{.name}{\"\t\"}{.image}{\"\n\"}" \
|sort|column -t
```

1. To choose between **Pod** or **Deploy/Sateful**: modify `RESOURCE_TYPE` & `RESOURCE_NAME`
2. To choose between **containers** or **initContainers**: modify the range.
  - containers: `{range ..containers[*]}`
  - initContainers: `{range ..initContainers[*]}`

```
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ kc get po slb03a-s3javastandaloner-tst-88cd46994-kg8gq \
> -o jsonpath='{range ..containers[*]}{.name}{\"\t\"}{.image}{\"\n\"}' \
> |sort|column -t
cloudapphealth      docker-registry.cloud.caixabank.com/catalog/paas/java-runtime-health-standalone-1-8:1.0.0
java-standalone-1-8  docker-registry.cloud.caixabank.com/containers/slb03a/s3javastandaloner:latest
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$
```

Figure 9: Get all containers parsing a Pod JSON

```
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ kc get deploy slb03a-s3javastandaloner-tst \
> -o jsonpath='{range ..initContainers[*]}{.name}{\"\t\"}{.image}{\"\n\"}' |sort|column -t
init-cacerts  docker-registry.cloud.caixabank.com/catalog/docker-init-setup:2.3.0
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$
```

Figure 10: Get all initContainers parsing a Deployment JSON

#### Example 2

Get all ports exposed by a Pod, Deployment or Statefulset.

```
$ kc get RESOURCE_TYPE RESOURCE_NAME \
-o jsonpath="{range ..containers[*]}{.name}{\"\t\"}{.ports}{\"\n\"}" \
|sort|column -t
```

```
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ kc get po slb03a-s3javastandaloner-tst-88cd46994-kg8gq \
> -o jsonpath='{range ..containers[*]}{.name}{\"\t\"}{.ports}{\"\n\"}' \
> |sort|column -t
cloudapphealth      [{"containerPort":8180,"name":"health","protocol":"TCP"}]
java-standalone-1-8 [{"containerPort":8080,"name":"http","protocol":"TCP"}, {"containerPort":8090,"name":"metrics","protocol":"TCP"}]
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$
```

Figure 11: Get all ports exposed in a Pod parsing a Pod JSON

```

u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ kc get deploy slb03a-s3javastandaloner-tst \
> -o jsonpath='{range ..containers[*]}.{.name}{"\t"}{.ports}{"\n"}}' \
> |sort|column -t
cloudapphealth [{"containerPort":8180,"name":"health","protocol":"TCP"}]
java-standalone-1-8 [{"containerPort":8080,"name":"http","protocol":"TCP"},{"containerPort":8090,"name":"metrics","protocol":"TCP"}]
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$

```

Figure 12: Get all ports exposed in a Pod parsing a Deployment JSON

### Example 3

Get all Volume Mounts of container called CONTAINER\_NAME inside a Pod, Deployment or Statefulset.

```

$ kc get RESOURCE_NAME -o jsonpath=\
"{range ..containers[?(@.name == "CONTAINER_NAME")]\
.volumeMounts[*]}.{.name}{"\t"}{.mountPath}{"\n"}}" \
|sort|column -t

```

```

u01a1f97@DESKTOP-L5FIVQG:~/alexwork$ kc get deploy slb03a-s3javastandaloner-tst \
> -o jsonpath='{range ..containers[?(@.name == "java-standalone-1-8")].volumeMounts[*]}.{.name}{"\t"}{.mountPath}{"\n"}}' \
> |sort|column -t
.
init-cacerts /opt/ca-certs
pvc-mounted-heapdump-cxb-slb03a-s3javastandaloner-tst /storage
volume-from-configmap-testproperty /tmp/properties/testproperty
volume-from-configmap-tsttest /tmp/properties/tsttest
volume-from-secret-certbueno /tmp/secretos/certbueno
volume-from-secret-certmalo /tmp/secretos/certmalo
u01a1f97@DESKTOP-L5FIVQG:~/alexwork$

```

Figure 13: Get all volumeMounts in a container parsing a Deployment JSON

## 10.4 Ubuntu vs RH7

Ubuntu es una distribución de Linux que se basa en un kernel Debian, lo que significa que no funciona exactamente igual que otras distribuciones con otros kernel. RH7 al igual que Centos (Open Source) trabajan con un kernel diferente y por tanto los comandos son diferentes. Por ejemplo el gestor de paquetes de Debian es apt-get y el de RH7 es yum.

## 10.5 Vagrant

Vagrant is a tool for building and managing virtual machine environments in a single workflow.

- **vagrant global-status:** nos muestra el estado de todas las maquinas vagrant que tenemos corriendo. Ojo! **This data is cached and may not be completely up-to-date (use "vagrant global-status -prune" to prune invalid entries).**
- **vagrant global-status -prune:** same as before, but up-to-date.
- **vagrant status vagrant\_id or vagrant\_name:** shows the status of the vagrant with "vagrant\_id" or "vagrant\_name" (vagrant\_name == cloudlabxxx)
- **vagrant up vagrant\_name:** initialize a new vagrant with the id "vagrant\_id", it can take a lot of time (15 minutes).
- **vagrant halt vagrant\_id or vagrant\_name:** stops the vagrant
- **vagrant destroy -f vagrant\_id or vagrant\_name:** destroy the vagrant with the id "vagrant\_id", you lose all your configurations, but not your data into /vagrant.
- **ps -ef | grep cloudlabxxx:** show the processes running in vagrant.
- **kill -9 process\_id:** kill the process with "process\_id"

## 10.6 Red Hat Package Installer

### 10.6.1 RPM

**RPM (Red Hat Package Manager)** is a free open and **powerful package management system**. The name refers to file format **.rpm** and the package manager program itself. An RPM package can contain an arbitrary set of files. Most RPM files are “binary RPMs” (or BRPMs) containing the compiled version of some software. There are also “source RPMs” (or SRPMs) containing the source code used to build a binary package.

It is capable of:

- **Building computer software from source** into easily distributable packages.
- **Installing, updating and uninstalling** packaged software.
- **Querying detailed information** about the packaged software, whether installed or not.
- **Verifying integrity** of packaged software and resulting software installation.

### 10.6.2 YUM

## 10.7 Timeshift to Backup and Restore Your Linux System

### 10.7.1 Introduction

Linux is susceptible to system failures caused by incorrect commands or system operations. So if you use Linux on your main computer, you may frequently encounter problems. Fortunately, there are system restoration tools that create snapshots of your files and settings, which you can restore on your system to put it back to its previous functioning point in case any of your operations renders it unusable.

Timeshift works by creating a snapshot of your system using either rsync or btrfs mode, depending on your Linux distro. To do this, what Timeshift essentially does is create a restore point for your system at a time when everything's running smoothly. This backup includes all the system files and settings—and no user files or documents. That way, when you accidentally mess up something on your system while configuring or customizing it, you can restore it back to this restore point and revert all your changes.

### 10.7.2 Timeshift Installation

The first step is install timeshift, depending on your system you will use different commands. For Ubuntu:

```
$ sudo apt install timeshift
```

### 10.7.3 Create a Snapshot

```
$ sudo timeshift --create --comments "A new backup" --tags D
```