**LINUX**

If you want to get started using Linux, you will LOVE this course! This 7+ hour Ubuntu Linux tutorial will take you from beginner to power user. We will cover installation, command line, administrative privileges, app development, server hosting, GitHub, and much more!

Time-stamps

[00:00](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=0s) Introduction to Linux

[08:44](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=524s) Linux distributions explained

[15:56](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=956s) Installing VirtualBox and setting up our virtual machine

[23:47](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=1427s) Ubuntu Linux installation on a virtual machine

[36:26](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=2186s) Disabling the ISO and first boot up

[38:40](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=2320s) VirtualBox guest additions for a better user experience

[46:14](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=2774s) Customizing our Ubuntu desktop

[54:41](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=3281s) Unity Tweak Tool for Ubuntu

[1:06:48](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=4008s) Installing Ubuntu alongside Windows (dual boot)

[1:23:09](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=4989s) Linux command line essentials

[1:36:17](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=5777s) Administrative privileges in terminal

[1:42:14](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=6134s) Using the package manager (apt-get) to install new applications

[1:46:17](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=6377s) Searching through the repositories to find new apps

[1:48:23](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=6503s) Installing packages that are not in the repository

[1:53:09](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=6789s) Keeping programs updated in Linux

[1:57:48](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=7068s) File permissions and ownership explained

[2:10:26](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=7826s) How to create files using the command line interface (CLI)

[2:15:24](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=8124s) Creating new directories and moving files

[2:19:59](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=8399s) Copying, renaming, and removing files

[2:24:43](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=8683s) The FIND command and it’s practical uses

[2:36:10](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=9370s) GREP command explained

[2:39:10](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=9550s) Using GREP in conjunction with FIND

[2:42:26](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=9746s) Redirecting the output of a command

[2:45:42](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=9942s) The TOP command and its uses

[2:47:01](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=10021s) How to view the entire list of processes and closing applications

[2:52:36](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=10356s) Services explained

[2:54:44](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=10484s) Configuring services using the command line

[2:59:20](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=10760s) Using CRONTABS to schedule tasks

[3:04:56](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=11096s) Choosing an integrated development environment (IDE)

[3:08:29](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=11309s) Eclipse installation and setup

[3:12:26](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=11546s) PyCharm installation and setup

[3:18:51](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=11931s) Introduction to GitHub, installation, and repository setup

[3:23:06](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=12186s) How to push/pull information from a repository

[3:29:13](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=12553s) How to remove/ignore directories in our repository

[3:34:25](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=12865s) Resolving merge conflicts through terminal

[3:41:42](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=13302s) How to setup and manage branches

[3:49:37](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=13777s) Meteor installation & setup

[3:55:32](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=14132s) Meteor project setup

[4:01:06](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=14466s) Router setup with React components

[4:13:31](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=15211s) Getting into the programming

[4:26:46](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=16006s) Rendering our blog posts

[4:42:06](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=16926s) Apache 2, PHP 5, and MySQL setup

[4:45:36](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=17136s) Server configuration

[4:51:14](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=17474s) Linux hosts file explained

[4:54:40](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=17680s) Deploying our Meteor app to an Apache 2 server

[5:00:03](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=18003s) MongoDB NoSQL database

[5:05:21](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=18321s) Virtual host setup

[5:16:46](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=19006s) phpMyAdmin setup

[5:24:50](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=19490s) Creating a basic virtual host

[5:33:00](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=19980s) Wordpress installation on top of our Apache 2 environment

[5:40:25](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=20425s) Database setup

[5:46:48](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=20808s) Python installation and CLI

[5:57:35](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=21455s) Adding/removing users through GUI

[6:01:09](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=21669s) Adding/removing users through CLI

[6:06:55](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=22015s) Adding users to a group

[6:10:51](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=22251s) Introduction to networking

[6:17:41](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=22661s) Local area network (LAN) explained

[6:25:08](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=23108s) Networking commands

[6:35:40](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=23740s) NETSTAT command

[6:40:39](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=24039s) Linux host file

[6:49:59](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=24599s) TRACEROUTE commands

[6:53:57](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=24837s) Network mapping explained

[7:08:29](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=25709s) Using SSH to access the command line of a remote host

[7:11:06](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=25866s) Using SFTP to transfer files between machines

[7:14:43](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=26083s) Setting up SSH on our local machine

[7:20:10](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=26410s) MAN command explained

**DESARROLLO:**

[**00:00**](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=0s) **Introduction to Linux.-**

Linus en los 80, Richard Stallman, crea un OS, pero con modificaciones difíciles de hacer porque tenías que modificar el código, es así que Linus Tolvalds, crea una especie de automatización , y es que Richard Stallman crea el grupo de soporte para el 1er Linux, llamado DEBIAN.

El principal problema del Linux es q hay muchos distribuidores, wallpapers desktops e icono q se refieren a lo mismo.



A principio de l 2000, empezó Ubunto, con el fin de distribuirlo económicamente gratis, la base fundamental de Ubunto la hereda de Debian, cuando se creo Ubuntu Unity para entorno desktop, se distribuyó en equipos móviles también, 3 de junio de 2010​, Ubuntu empezó a tener mayor estabilidad, y empezó a distribuirse en desktop y demás equipos



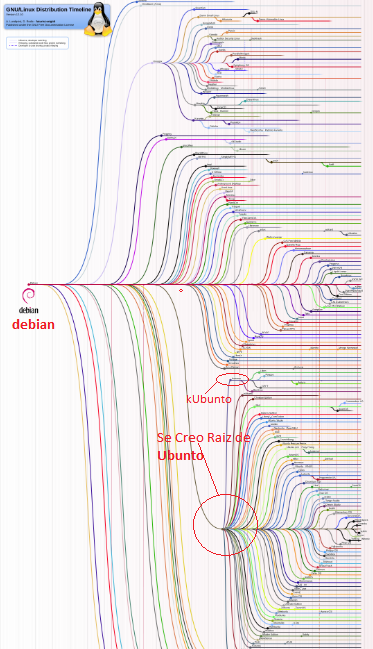
,

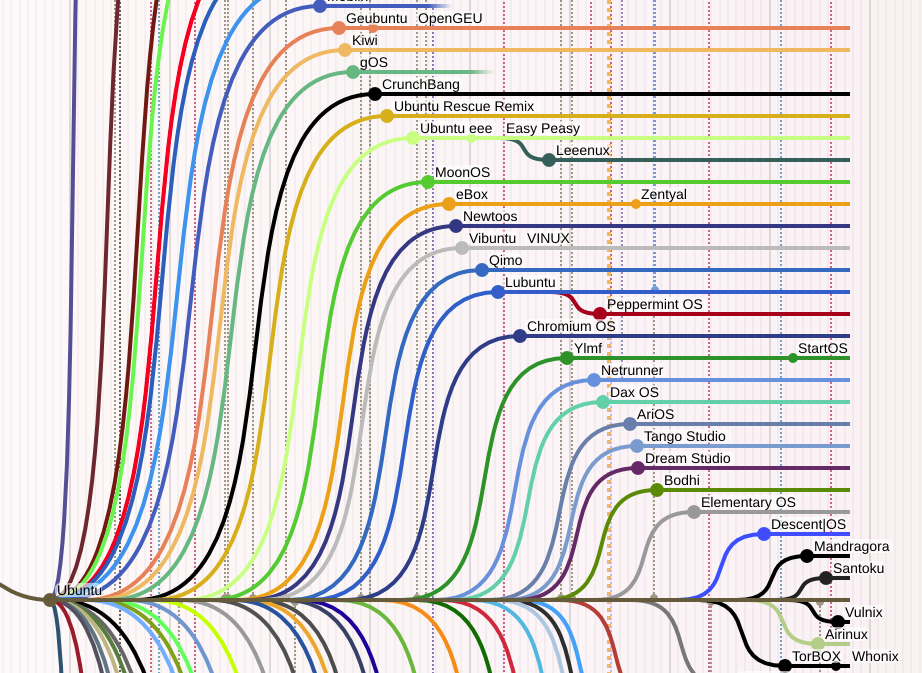
En October 2015 Ubuntu 15.10; con renews 2 vece al año uno en Abril y Octubre;

<https://ubuntu.com/tutorials/tutorial-install-ubuntu-desktop#1-overview>

[**08:44**](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=524s) **Linux distributions explained.-**

De la creación de Ubunto se generaron muchísimos otros derivados como kubunto, Ubunto Studi, etc,



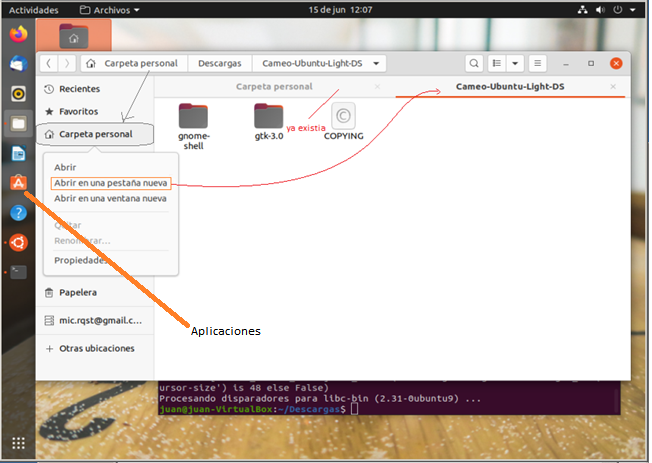


[**15:56**](https://www.youtube.com/watch?v=wBp0Rb-ZJak&t=956s) **Installing VirtualBox and setting up our virtual machine .-**

**$ ./ = directorio actual.**

**$ sudo ./VBoxLinuxAddition.run // sudo = corre como administrador.**

**Como agregar una nueva pestaña:**



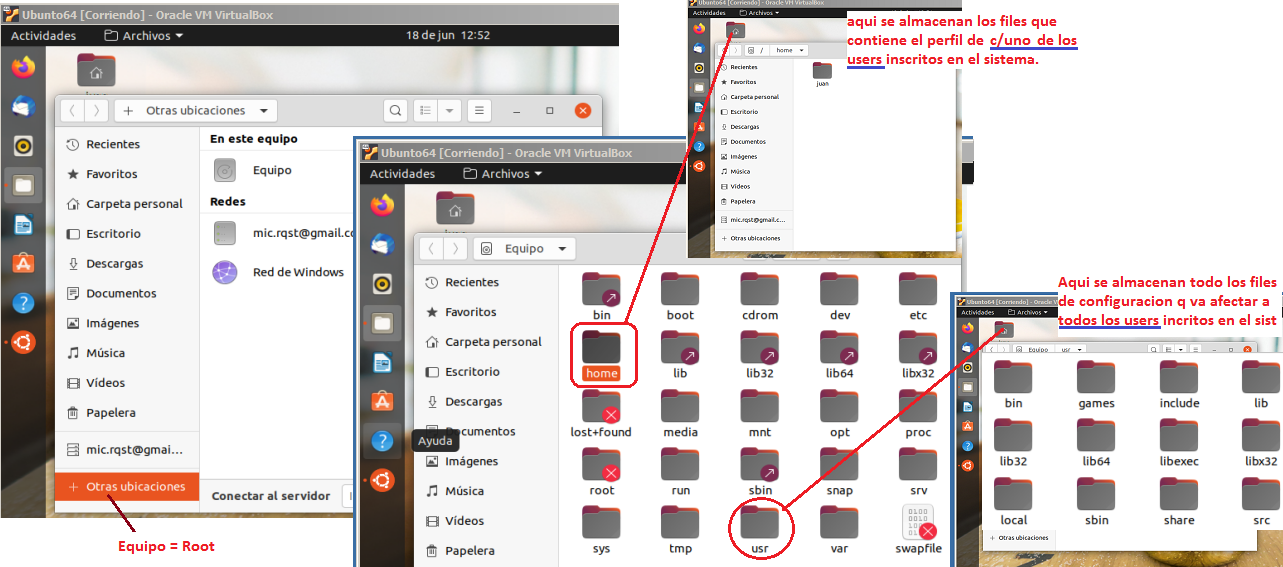
Con **Ubuntu Tweak** es posible personalizar tu escritorio con funciones como las siguientes:

* Visualizar información básica del sistema (distribución, kernel, CPU, memoria)
* GNOME Session Control
* Inicio automático de aplicaciones
* Personalizar la pantalla de bienvenida
* Ajustas los efectos de Compiz
* Establecer las preferencias de Nautilus
* Gestionar la energía del sistema
* Mostrar y ocultar elementos del escritorio: iconos, volúmenes, papelera, icono de red
* Establecer la seguridad del sistema
* Instalar aplicaciones de terceros
* Modificar las preferencias del GNOME Panel
* Hacer limpieza del sistema: paquetes no necesarios y caché
* Establecer los atajos de teclado

Instalo una Herramienta para una mejor Administracion de Ubunto:

**$sudo apt-get install unity-tweak-tool // sudo apt-get install <q quiero instalar = ‘unity-tweak-tool’>**

**Descripcion de Files en Linux Ubunto .- (Linux-image-5.4.0-42-generic)**



**root ‘/’**  // rais maxima del sistema, pide clave del superusurio.

**‘/home’** // Es donde se almacenan los perfiles de c/users en el sistema, ej : ‘juan’.

**‘/usr’** // Es usado por el sistema para todo las cuentas de users del sistema.

**‘/usr/bin’** // contiene binary files that you an run (ejecutables).

**‘/usr/share’** // es basicamente p las configuraciones de los binary applications.

**‘usr/share/applications’** // contiene todo las aplicaciones instaladas en el sistema, de donde, como etc.

**Kernel:**

Es el nucleo o Kernel, la parte central de un S.O. y se encarga de realizar toda la comunicación segura entre el software y el hardware del CPU.

* Kernel de Windows : MSDOS basado en basic.
* Kernel de MacOS : Unix basado en C.
* GNU/Linux : Linux basado en C. ( GNU – licencia gratuita, es por la Apps y Linux es el nucleo).
* Android : basado en java/C++

**Capas Graficas encima del Kenel para ser mas Amigable hay:**

* Windows : windows wpf (windows presentation fundation).
* MacOS : Aqua.
* Android : Capa de personalizacion (emui).
* Linux : KDE, GNOME, PLASMA, UNITY.

**Entornos Graficos de Linux:**

* Entorno KDE Plasma 5 .- quinta actual generacion de entorno de escritorio creado por KDE p sistemas Linux, y mejores convergencias entre dispositvos.
* OPENSUSE leap o tumblewee.
* Ubunto con GENOME, bien usado. ( si tiene la version LTS = indica soporte extendido).
* Fedora con GNOME (basado en RedHat).
* Debian con GENOME, bien estable.

**Sistemas de Archivos:**

* Windows : FAT, FAT32, NTFS.
* MacOS : HFS+.
* Android : EXT4.
* Linux : EXT2, EXT3 y EXT4 (usar este tipo p q sea mas compatible con otros SO).

**Extensiones que se Ejecutables:**

* Windows : EXE.
* MacOS : PKG y DMG.
* Android : APK.
* Linux : TAR, RPM, DEB. ( los ejecutables de debian/ubunto = .deb)

**Que progrmas ya bienen con Linux:**

* Ofimatica : Libre Office, open office, wps.
* Lectores de PDF : Okular.
* Compresor de Archvos.
* Codec de Multimedia.

**ACTUALIZAR SISTEMA:**

Siempre que empiece a usar el sistema reviamos si todos los packqetes estan actualizados p cualquier inconveniente.

**$sudo apt-get update // antes de cualqier instalacion.**

**/bin .- esta carpeta alamcena los files que se pueden correr, file you can run.**

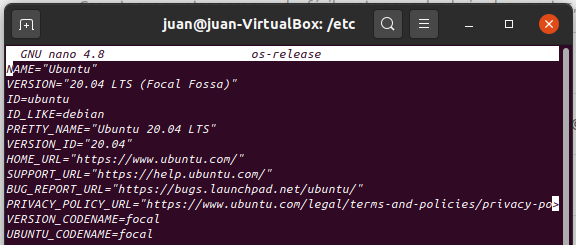
**/usr/share .- basicamente save the files for binary configuration,es la carpera donde se alamacenan las configuraciones de los bin file.**

**$ uname –r // podemos ver la versión de Kernel del sistema operativo.**

**/etc.- desde la raiz, ver la version demi S.O. debe estar en un file q termina con ‘release’ :. Buscamos.**

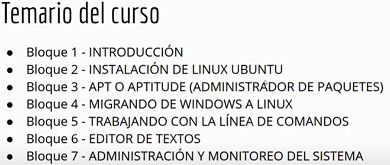
**$ ls \*-release // buscamos este file.**

**$ nano os-release // editamos el file q nos indica la version y kernel.**



**Del Video :** Curso de linux desde cero -2020

<https://www.youtube.com/watch?v=HuirxQ3Nxvc&t=57s>



* Install



* UpDate
* UpGrade
* Search
* Remove
* Puge
* Audacity: Edicion de Audio.
* Clementine: Repro de Musica.
* Azureus: Cliente BitTorrent
* GIMP: Edicion de Imágenes.
* Chromium: navegador web.
* Opera. Nav Web y App p Web.
* Inkscape: Edicion de Imag Vectoriales.
* LibreOfice: Ofimatica (~Excel, Word, Presnteacion)
* ThunderBird: Correo Cliente.

**Servidor SSH entre Windows y Linux.-**

Es para administrar tu servidor Linux desde un Windows Desktop.

Aprender de Memoria los comando:

* Navegar entre directorios y Rutas Absoluta y Relativas.
* Manejo de Archivos: mkdir, touch, cat, more, cp, mv, rm, pwd.
* Permisos: chmod, chown
* Grep: Busquedas.
* Kill de procesos: matar procesos y pausar procesos.
* Agregar Usuarios: useradd.
* Doucmentacion: man.

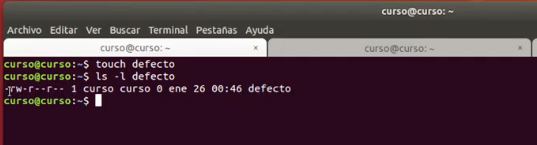
Administrar y Monitorear el Sistema Linux:

* App de ‘htop’ : Memoria RAM, Procesador, Procesos.
* App ‘nethogs’ : Revision de Trafico de Red.
* App ‘Gparted’ : Comportamineto DiscoDuro, Tipo de Particiones, Sist Archivos, Crear Eliminar particiones.

PERMISOS x DEFECTO:

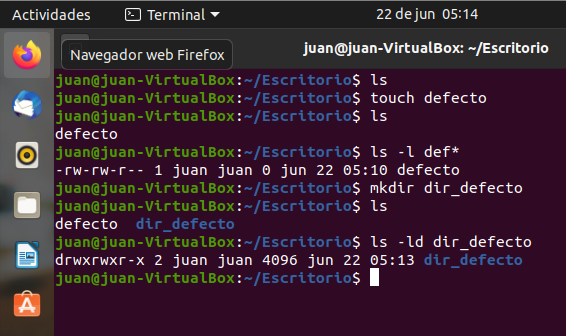
Los Directorios y Archivos se les crean unos permisos por defecto, ej el fichero ‘defecto’.

**$**touch defecto //crea un fichero llamado defecto.



Puedo hacer asi tambien,

**$** ls –l defe\* // si quiero buscar el file ‘defecto’ y saldra los mismo.



Para quitar los permisos, se debe hacer lo sgnt.

**$**umask

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Instalar postgresql12**

Link de Instalacion:

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04-es>

$ sudo apt-get update // actualizamos antes de hacer cualqier instalacion.

$ sudo apt-get install postgresql postgresql-contrib // instalamos solo la BD postgressql

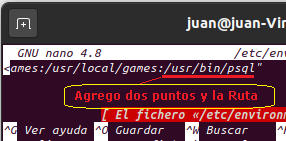
$ sudo su - postgres // el postgresql por defecto crea el user ‘postgress’ y debe accederse // desde el root.

Postgres=# // ya estas en la linea de cmd para crear/leer BDs en lenguaje ‘pqsl’.

**NOTA Postgresql**.-

Para que reconosca el lenguaje ‘**pqsl’**, debemos colocarlo en el path del sistema, antes de cualqier cosa adicional.

Por defecto, Postgres utiliza un concepto llamado “roles” para gestionar la autenticación,  parecidos a las cuentas de Unix, pero con la diferecnia q no distingue entre los usuarios y los grupos, y en su lugar prefiere el término más flexible de “rol”.´



Colocando ‘psql’ en el path del sistema.

$ sudo cp /etc/environment /etc/environment.copia // 1ro creamos un backup

$ sudo nano /etc/environment // edit el file de los paths.

Agregamos -🡪 *:/usr/bin/psql****”***

**Crear un nuevo rol.-** desde la cuenta del usuario postgres, **postgres@server:~$**

**$** createuser--interactive

$ Enter name of role to add: miclab

Shall the new role be a superuser?(y/n): y

**APT o APTITUDE (Adm de Paquetes):**

Podemos gestionar sobre repositorios o paquetes ya instalados o que se van a instalar;

Install, UpDate, UpGrade, Buscar, Remove, Purge.

Repositorios, dentro del sistema, si hago un UpDate, pues actualza algun software, codigo que el sistema ya lo tiene

APTITUDE es la version mejorada de Ubuntu.

**1ro** Instalemos la herramienta **aptitude**:

$sudo apt-get install aptitude

Instalamos ej **tilda** es un tipo de consola.

$sudo apt-get install tilda

**2do** busquemos x ejemplo si hay java, van a salir muchos paquetes, q estan en el sistema.

$aptitude search java

Buscando que paquetes de python hay:

$aptitude search py

**3ro** como se cuales son los sources o los repositorios que tengo aqi:

Usando el editor nano;

$nano /etc/apt/sources.list sources // va abrir el editor de ‘nano’ los repositorios o sources q ay en el sist.

Ej: Quiero instalar Node.JS y TypeScrip;

$sudo apt-get install npm // instala el node package manager, extension ‘.js’

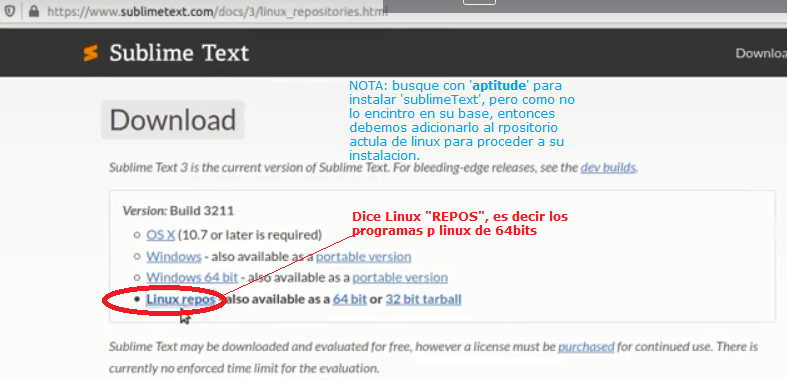
$sudo apt-get install node-typescript // instala el typescript, extension ‘.ts’

Ej: Quiero instalar sublimeText usando la tool ‘aptitude’;

**$**sudo apt-get install sublimetext

E: no se ah podido localizar el paquete sublimetext // no lo puede instalar x q no localiza el pqt modulo en su repositorio.

// Entonces: lo bajamos de la web:



Como no esta en el repositorio de Linux, entonces lo tenemos que agregar, actualizar y luego instarlo.

Desde la página que nos indica, hacer esto.

Si queremos remover algun paquete o app: solo quita los archivos de configuracion, Ej sublimetet o iprange

**$**sudo aptitude remove sublimetext $ sudo apt remove iprange

Si queremos purgar/remover; quita los paquetes de software, q no son necesarios.

**$**sudo aptitude purge sublimetext

Ej: Vamos actualizar la herramienta aptitude.-

**$**sudo aptitude update

Pero si quieres actualizar las apps y librerias del SO es decir toda las capas GNU de Linux.-

**$**sudo aptitude upgrade

NOTA para ver la lista de paquetes:

**$**nano /etc/apt/sources.list // Nos dan la lista de paquetsesinstalados.

**$**sudo nano /etc/sudoers // da la lista d usuarios n el sistema, podemos agregar debajo ‘root’ con privilegios.

**NOTA si sale un ERROR:**

Cuando queremos usar ‘**$**sudo aptitude update’ o queremos instalar un nuevo paquete o programa por

Ej: ‘**$**sudo apt-get install sublimetext y nos sale:

‘[ ERROR] Leyendo lista de paquetes’, bla bla

‘E: Leyendo lista de paquetes’, bla bla

‘E: Problem with MergeList /var/lib/apt/lists/repo…..

Es por que al instalar un pqt o soft o actualizando algo, el sistema se interrumpio, y debemos ractualizarlo;

(Cuando actualizamos desde el servidor ububto a nuestro linux este los descarga en ‘/var/lib/apt/lists’ )

**$**sudo rm /var/lib/apt/lists/\* -vf

**$**sudo apt-get update

(pd: http://jhomartinez.blogspot.com/2015/08/leyendo-lista-de-paquetes-error-gnulinux.html)

**NOTA**: Hay varias herramientas de office, “**OpenOffice**”, “**WPS** **Office**”, “**LibreOffice**”, “**FreeOffice**”;

* Lo q es ‘Word’ ‘.docx’ en Linux se gaurda como extension “.odt”.
* Lo q es ‘PowerPoint’ ‘.pptx’ en Linux se gaurda como extension “.ppsx”.
* Lo q es ‘Excel’ ‘.xls’ en Linux se llama ‘Calc’.

**NOTA PYTHON:** por que usar ‘py’ vs ‘python’

**$**python –V

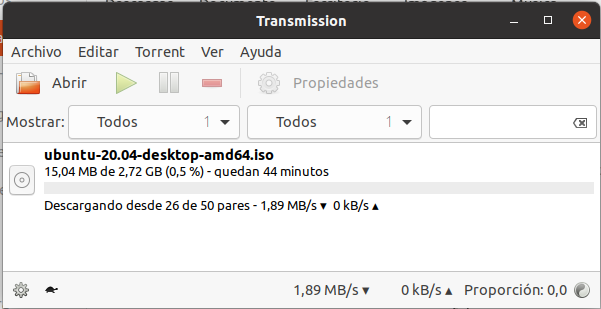
Python 3.6.5 Anaconda

**Cliente Torrent:**

Gestion la descarga de un archivo de diferentes maquinas con la idea q si una fuente se cae, busca de otra para confiar la descarga del archivo. Buscar en el google: ‘***links de programas en torrent*’**.

**App para BitToorent.-** se llama **“Transmission”** is a fast, easy, and free multi-platform BitTorrent client.

Al usar la herramienta saldra algo parecido a la sgnt figura.-



**NOTA ERROR /dev/sda5:**

**Para arreglar algun problema con el filesystem del sistema, correr.**

**$**fsck /dev/sda5

**INSTALANDO Google Chrome.-**

**1ro** Buscamos en el navegador “Descargar Google Chrome”

Bajas el file = “google-chrome-stable\_current\_amd64-deb”

**2do** comando de instalacion

**$**sudo apt-get install ~/Descargas/ google-chrome-stable\_current\_amd64-deb

**3ro** lo buscamos p ver con q nombre se instaldo

**$**aptitude search chrome

google-chrome-stable

**4to** lo corremos.-

**$** google-chrome-stable

**Linea de Comandos (parte 01):**

ls, pwd, cd

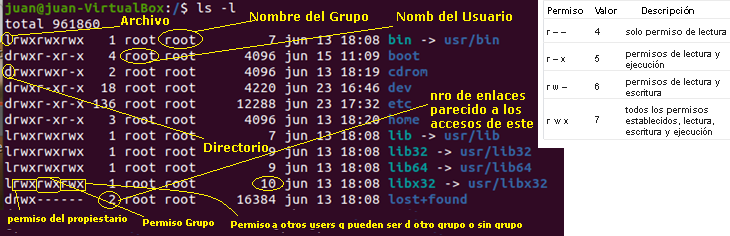
**$**cd ~ // te lleva a tu home del usuario. /home/juan

**$**cd / // te lleva al root.

**$**cd /cd\* // si estoy en ‘/home/juan’ mi home, y quiero ir a un directorio q esta en root ‘/cdrom’

**$**cd ../../ // etorcede 2 niveles.

**$**ls –l // lista pero con todos los directorios y archivos con sus respectivos permisos.



**$**chmod 775 mio // cambias los permisos a propietario(7=111=rwx), grupo(7=111=rwx) y // otros(5=101=r-x)

// pero si se quiere cambiar ‘own’ es necesario hacerlo desde el mismo espacio de // trabajo del nuevo propietario.

**$**mkdir // creamos una carpeta+

**$**rmdir // borrar una carpeta vacia.

**$**touch mitexto.txt // sirve para crear un archivo.

**$**rm mifile.txt // borra archivos.

**$**rm –r micarpeta // borra archivos y carpeta.

**$**nano mifile.txt // edito mifile.txt en el editor nano.

**$**cat mifile.txt // listo el contenido de mifile.txt ~ **$**more

**$**more mifile.txt // listo el contenido de mifile.txt ~ **$** cat

**$**mv file.txt mio // muevo file.txt a la carpeta ‘mio’, según si estas un nivel arriba de la carpeta mio.

**$**man cat // te da un help del comando ‘cat’

**$**cp hola.txt ../ // copia hola.txt un nivel ma alto.

**$**cp –R mio ../ // copia un directorio con fles incluidos a un nivel superior.(mio = directorio)

**$**sudo su // cambias del actual usuario a usuario root.

**$**aptitude search py | grep python // busca el contenido ‘py’ en todo el disco y luego sustrae de esta busqueda sustraemos // la palabra ‘python’

**$**grep “palabra” file.txt // hace la busqueda de esta palabra en el file.txt.

**$**pgrep firefox // nos devuelve el ID de proceso de un programa q se este ejecutando.

**$**kill <idproceso> // matas el Proceso. Kill 13389

**$**killall chrome // si tiene varios procesos.

**$**kill -stop 14659 // detiene un proceso q este corriendo en memoria, a veces demora 30seg en pararlo.

**$**kill -cont 14659 // continua el proceso q estaba detenido.

root# chown root juan <file> // cambiamos de own un file del user:’juan’ al ‘root’, p esto debemos cambiar a root 1ro.

// una vez propietario recien podemos cambiar los permisos con ‘chmod’.

**$**whereis firefox // nos entrega la ubicación y ruta del file, si el file=programa=app, entonces nos dara la.

*<resultado:>* // ubicación de su componentes, librerias y el ejecutable y su help.

usr/bin/firefox // aquí se almacenan los ejecutables.

usr/lib/firefox // aquí se almacenan las componentes y librerias.

ect/firefox // aquí se almacenan las configuraciones de todas las apps.

usr/share/man/main1/firefox.1.gz // el manual de ayuda “man”, podemos hacer man firefox y nos dara su ayuda.

**$**cat /etc/passwd // lista el archivo donde estan todos los users del sistema.

**$**cat /etc/passwd | grep juan // me lista unicamente el usario “juan”

**$**sudo adduser jacqueline // creo un nuevo usuario , y me va pedir sus clave nueva, sus datos personales.

**$**sudo passwd jacqueline // permite crear o cambiar su contraseña.

**$**sudo deluser jacqueline // elimina el user = jacqueline

**$**sudo deluser --remove-home jacqueline // elimina el user = jacqueline y su contenido.

TaskManage : Buscar en el entorno grafico la App “MONITOR” y va salir un grafico como el taskmanager de windows para ver todo los procesos de consumo de memoria y cpu, tambien nos da el trafico de red y sistemas de archivos.

**$**top // desde la lnea de comando nos da un taskmanager de los proc q se estan ejecuntando.

**$**sudo apt-get install htop // instalamos un taskmanager igual q ‘top’ pero grafico y con mouse desde la consola.

**$**htop // levanta la app htop = taskmanager.

Para ver recurso de Red desde la consola: nethogs

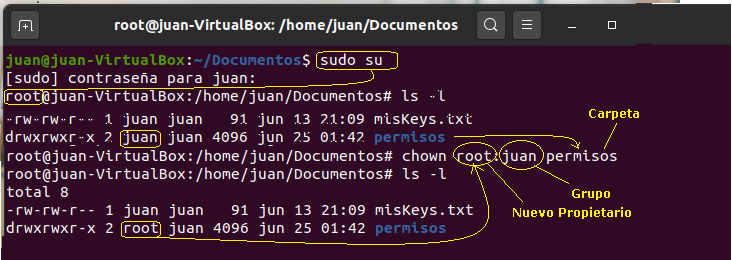
**$**sudo apt-get install nethogs // si no existe lo instalamos, podemos verificar con ‘**$**apttude search nethogs’.

**$**sudo nethogs // visualiza desde le terminal el trafico de red.

**NOTA**: Linus/unix a diferencia de Windows, la organización de sus programas estan distribuidos por genero, es decir,

**Cambiar el propietario y permisos de un documento:**

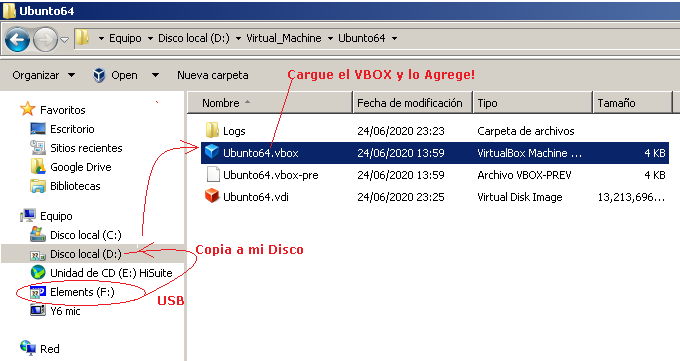
Si creamos un file, somo los ‘propietarios’ para pasarle el own al ‘root’, debemos hacerlo desde la misma cuenta del ‘root’, para eso desde nuestra cuenta “**$**sudo su”, pasamos al root y hacemos “chown”.



**NOTA : Ubuntu 18.04 requiere Python 3.6 para funcionar.**

Como Ubuntu también depende de Python para un gran funcionamiento interno, tenga en cuenta y sea extremadamente cuidadoso condesinstalar y cambiar versiones d python.

**NOTA se puede transportar tu SO Virtual en una USB a cualquier lugar:**



**File System Disco:** **POSIX (Portable Operating System Interface)**

El sistema de archivos nativo de Linux es el EXT2 otros EXT3 o el ultimo EXT4 , tamben hay ReiserFS y XFS.

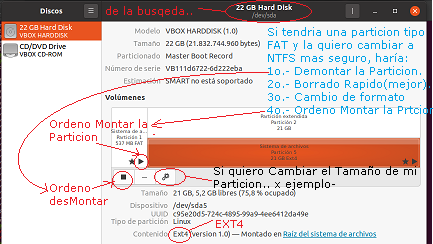
Sistema de Archivos = Linux = Unix.

Los discos en windows los identificamos como **‘C:\’**, en Linux es **‘sda’,**a veces **‘sdb’,**

* **C:\ = sda** // disco fisicos pero si hay particiones seria /dev/sda1, /dev/sda2 …
* **D:\ = sdb** // disco fisico per si hay particiones seria /dev/sdb1, /dev/sdb2 …
* **CD o DVD = scd0 o sr0** // linux los denomina como scd0 o sr0.

etc el HD, en buscar escribes ‘disco’ o ‘disk’ y saldra un icono con la forma de un disco duro con el que se puede hacer todo las opciones de particiones.

Para editar, crear particiones, ampliar particiones, todo respecto al DiscoDuro, desde el entorno grafico, “buscar-🡪 “disco” o “disk” nos traera un utilitario como se ve en la figura.



**NOTA:**

**Este Utilitario solo permite algunos formatos mas conocidos, pero si qeremos algo mas especializado, cambiar en caliente particiones, tamaños bien gr**



Leer las Particones desde la consola va listar discos **reales** y otros q son virtuales como “loop”.

**$**df –h | grep “/dev/sd” // grep filtra de la lista general los q empiezan con ‘sd”, discos **reales**.



**NOTA**:

La lista completa de los devices en linux en ‘/proc/mounts’.(nota: todo los cmds de linux en ‘/etc/mtab’.)

**$**sudo ln –s /media/cdrom /home/juan/mi-cdrom // permite crear un acc directo de CDROM al directorio del user = juan.

// ‘ln’ make links between files. **$**man ln

**$**findmnt // lista todo los devices montados en nuestro sistema.

**$**findmnt –t ext4 // lista las unidades con el tipo de fileSystem de Linux mas comun.



**$**df -hx tmpfs --output=source,target // filtrado y p ver opciones man df

**$**sudo lsblk –f // list all available and specific information devices.

**$**cat /proc/mounts // list a detail information for every device monted.

**Instalar y configurar SSH:**

Parecido al antiguo ‘Telnet pero fue reemplazo por SSH (SecureSHell), sirve para conectarse a otros servidores de forma remota,

En la actualidad los servidores de AWS, AZURE, GoogleCloud, tienen esto servicio habilitado para mejorar el manejo de los webSevices. (tambien lo puedes usar en una RaspBerry, Maq Virtual, Dockers, etc).

**NOTA**: tambien hay otros servicios de UNIX antoguos como SFTP (Secure FileTranferProcotol)

**$**sudo apt-get install openssh-server // Instalamos el SSH en Ubunto 20.4(LTS)

**$**sudo systemctl enable ssh // Nos aseguramos q el servicio SSH este corriendo.

**$**sudo nano /etc/ssh/ssh\_config // Editamos la *config* para cambiar el puerto.

**$**sudo systemctl restart ssh // si hicimos cambios restauramos el servidor ssh.

**$**service ssh status // me da el estado del servicio ssh, si esta activo, ***quienes se conectaron*** su ip y user.

// desde **Windows podemos usar “PuTYY”** -🡪 Session **SSH**.

**NET Tools Herraminetas de RED:**

**$**sudo apt update // 1ro siempre actualizamos librerias del SO.

**$**sudo apt-get install net-tools // instalamos todas la herramientas mas usadas de red.

**$nmcli device show** // visualizamos los parametros de red.

(interfaz de línea de comandos de Network Manager) utiliza para crear, editar, mostrar, eliminar, activar y desactivar conexiones.



$whoami // que usuario esta logeado en este terminal.

$hostname – I // me da la Ip del **HOSTNAME.**

**10.0.2.15**

**$**hostname // me el nombre del **HOSTNAME.**

**Juan-VirtualBox**

mostrar o configurar el nombre de host y el nombre de dominio de una computadora.

$netstat –atnp // me da los puertos q estan escuchando y de q Interface.

**$**ip addr

**$**ifconfig // da los detalles de la interface de red nombre, Ip, mac address.

**$**ifconfig –a

**$**ifconfig wlan0

**$**sudo ifconfig enp0s3 down // desactivamos la interface de red.

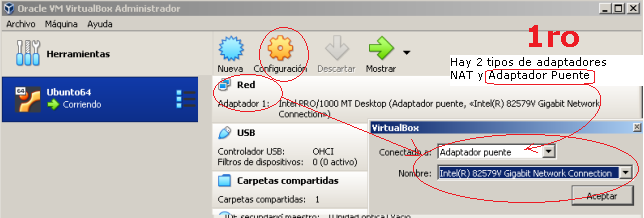
**$**sudo ifconfig enp0s3 up // activamos la interface de red.

**Change my Ip, Route Configuration NetWork from Console: Como solucione los Errores q me salieron..**

Las pruebas q hice es configurar la VM con DHCP autoconeccion, Rsltdos:

Se puede navegar x intrenet hay un GW y un IP que no son del nivel de mi red local, por lo q el servicio de SSH no funciona

Solucion.- en el Vbox en Red coloque Adaptador Puente, y en el VM de Ubuntu quite la coneccion automatica y quite el DHCP, como muestra la imagen:





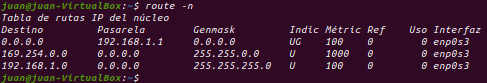
**Los Cambio que hice desde la consola no hicieron efecto, una vez q reinicie la VM todo regresaba a la config anterior.**

**Como manejar la config desde la COSOSLA:**

**$**sudo ifconfig enp0s3 192.168.0.15 netmask 255.255.255.0

**$**sudo route add default gw 192.168.1.1 enp0s3

**$**route –n



**NOTA Comando systemctl :**

El comando **systemctl** es una herramienta que sirve para poder controlar el sistema y sus servicios(demonios). ... La gran mayoría de los sistemas operativos **Linux** modernos ya usan este comando, que realmente es una herramienta que maneja un conjunto de demonios.

**NOTA Demonios:**

El primer servicio que inicia el Kernel de Linux es ‘**Init’** o ‘**Systemd’**. Seguidamente, init o systemd son los encargados de cargar el resto de servicios del sistema operativo. Por lo tanto Init o Systemd son los padres de todos los demonios o servicios que se inicializan en nuestro sistema operativo.

En el caso que systemd o init no se inicien, nuestro ordenador nunca podrá llegar a arrancar. Por lo tanto son demonios extremadamente importantes.

**NOTA:**

En windows 7 podemos usar la app “***PuTTY***” para crear un cliente o servidor SSH, o la aplicación “freeSSH”, en el lado del servidor hay q configurar los permisiso de acceso del usuario. Y en lado del remoto(cliente) puede ser Windows10, win7, dos, o desde Android con el app “Termux” es un terminal(lo probe con mi celular).

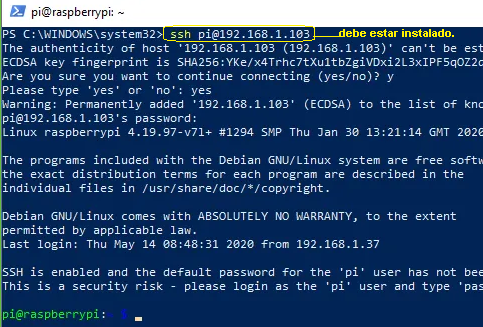
(win 10 ya tiene el utilitario ssh (cliente y servidor) , OpenSSH

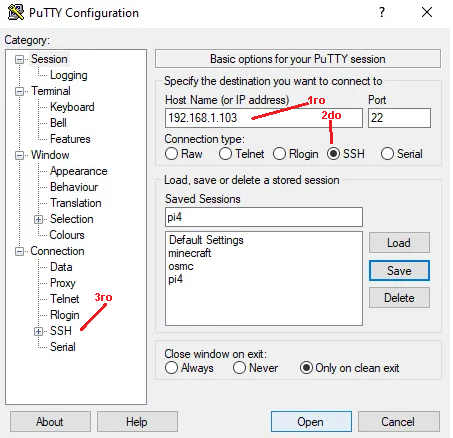
Para conectarse al Servidor Windows si ya esta configurado el SSH Server puerto por defecto es 22 desde Linux(Raspi).

$ssh [miclab@192.168.1.233](mailto:miclab@192.168.1.233) // <user ssh>@serverSHH te pedira la clave de este ususario, en este caso mi Corei7 Win7.

Hay Varios Utilitarios para conecciones de SSH: entre los mas populares es el PUTTY:

Como ver si ssh en windows: Usando el power shell de windows 10:





**Desde Win10 usando Putty-ssh:**

Usando Putty: solo colocamos el IP o HostName del Servidor ssh de linux y el purto 22, al conectarsenos pide user:pi contraseña:miclab.

Ya dentro del terminal remoto, para terminar $pi@raspi: $ logout

Para Descargar el Batch de GitHub:

**https://git-scm.com/downloads**

**Get your CURRENT VERSION FIRMAWARE:**

pi@raspberrypi:~ $ uname -a

Linux raspberrypi 5.10.25-v7l+ #1408 SMP Mon Mar 22 12:49:24 GMT 2021 armv7l GNU/Linux

**Listar todo los pckgs/prog instalados.-**

$sudo dpkg –list

Si Quiero Listar un Paquete/Package Programa/Aplicacion en particular: Ej: OpenCV

$sudo dpkg –list |grep libopencv

En RaspberryPi seria asi:

$ dpkg -l | grep libopencv

Y si quiero de la lista desinstalar **apache2** :.

$sudo apt-get –-purge remove apache2

Si quiero saber un pckg especifico si esta instaldo: Ej ‘apache’

pi@raspberrypi:~ $ sudo dpkg --get-selections | grep apache

apache2 install

apache2-bin install

apache2-data install

apache2-utils install

libapache-poi-java install

libapache-pom-java install

libapache2-mod-php7.3 install

pi@raspberrypi:~ $

sino sale ningun pgk de lo q buscas:.

pi@raspberrypi:~ $ apt list | grep opencv

**Acceso Directo a un File fuera de mi Directorio:**

En ocasiones los prog q hacemos necesitan una lib q esta en otro directorio, y nuestro prog no corre por que no esta localmente y tampoco esta referenciado, entonces una forma de referenciarlo p importarlo, es creando un accesodirecto o referencia directa ya q esta en otro directorio.

Para crear un acceso directo a un archivo/packg/programa q esta en otro directorio muy lejos, podemos crear una especie de links de acceso directo, como existe en windows, sin necesidad de mover el pckg/programa:

1º nos ubicamos en el dir q queremos tener acceso al file remoto.

2º $ ln -s <dirRemoto/fileRemoto> <nomAccesoDirecto\_local>

Ej:

// explicacion = ln -s < /usr/local/lib/python3.7/site-packages/cv2/python-3. 7/cv2.so > <cv2.so>

**$** **ln -s** /usr/local/lib/python3.7/site-packages/cv2/python-3. 7/cv2.so cv2.so

Verificar q versiones tenemos instalados;

pi@raspberrypi:~ $ pip freeze

arandr==0.1.9

.

.

.

unicornhathd==0.0.4

urllib3==1.24.1

Werkzeug==0.14.1

**Usando Script en files Bash.- (**[**http://www.gnu.org/software/bash**](http://www.gnu.org/software/bash)**)**

.***bashrces*** un [script de shell, el](http://en.wikipedia.org/wiki/Shell_script) Bash ejecuta siempre que se inicia de forma interactiva. Inicializa una sesión de shell interactiva.

Nota: Puede usar cualquier comando en ese file, q pueda ejecutar desde el prompt.

***.bashrc***runs on every interactive shell launch. If you say: $ bash ; bash ; bash; a diferencia de ***.bash\_profile*** y ***.profile*** que solo se ejecutan al comienzo de un nuevo shell de inicio de sesión.

Tambien hay otro ***.bash\_profile***; Ud elige si el comando va en ***.bashrc*** o ***.bash\_profile***, depende si quieres q el comando se ejecute una vez o por c/inicio shell.

Por ejemplo, un path es mejor colocarlo en ***.bashrc\_profil***, export PATH="$PATH:/some/addition", si esto lo coloco en ***.bashrc***, cada vez que lanzas un sub-shell interactivo: /some /added se agregaría nuevamente al final del PATH, creando trabajo adicional para el shell cuando escribes mal un comando.

El propósito de un archivo .bashrc es proporcionar un lugar donde pueda configurar variables, funciones y alias, definir su prompt(PS1) y definir otras configuraciones que desee usar cada vez que abra una nueva ventana de terminal. Funciona al ejecutarse cada vez que abre una nueva terminal, ventana o panel. Eje:

export CLICOLOR=1

export LANG="en\_US.UTF-8"

alias cp="cp -i"

alias ls="ls --color=auto"

export PS1="\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ "

export EDITOR="vim"

**NOTA:**

There are many different shell programs that run in Linux, but the one that runs on most distributions, including Raspian, is Bourne-Again shell (bash). bash was written by Brian Fox for the GNU project to add features and functionality to the original Bourne shell (sh) written by Steven Bourne

**Usando Script en file Shell(sh).-**

Creamos un shell simple q va repetir un saludo, puedes escribirlo en cualqier editor en este caso usaremos ‘nano’.

pi@raspi:~ $ nano hola.sh

escribimos el sgnt script:

#!/bin/bash

for i in 1 2 3 4 5

do

echo "Bienvenido $i times"

done

lo Corremos:

pi@raspi:~ $ sh hola.sh

Bienvenido 1 times

Bienvenido 2 times

Bienvenido 3 times

Bienvenido 4 times

Bienvenido 5 times

Otro Script:

pi@raspi:~ $ nano forloop.sh

#!/bin/bash

for i in `seq 1 10`;

do

echo Hello World!

sleep 1

done

lo Corremos:

pi@raspi:~ $ sh forloop.sh

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

**Howto see my space in SD:**

pi@raspberrypi:~ $ df -h

Filesystem Size Used Avail Use% Mounted on

/dev/root 57G 12G 44G 21% /

devtmpfs 1.8G 0 1.8G 0% /dev

tmpfs 1.9G 16K 1.9G 1% /dev/shm

tmpfs 1.9G 8.6M 1.9G 1% /run

tmpfs 5.0M 4.0K 5.0M 1% /run/lock

tmpfs 1.9G 0 1.9G 0% /sys/fs/cgroup

/dev/mmcblk0p1 253M 48M 205M 19% /boot

tmpfs 378M 0 378M 0% /run/user/33

tmpfs 378M 4.0K 378M 1% /run/user/1000

Si quieres quitar espacio en tu SD puedes liberar asi;

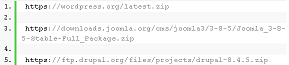
$ sudo apt-get purge wolfram-engine

$ sudo apt-get purge libreoffice\*

$ sudo apt-get clean

$ sudo apt-get autoremove

***Comando ‘wget’*** q es para bajar programas, podemos baja 1 o varios a la vez, si son varios debemos crear un file ej: example.txt y dentro debemos colocar los links de c/prog q qeremos bajar dejando una linea;



El comando solo p bajar pckgs de un link: **$wget –i example.txt**

Si qiero bajar un link con cambio de nomb al instalarlo: **$wget –O <nom> <link>**

Bajamos uno x uno los insaladores;

**pi@raspberrypi:~ $ wget -O opencv.zip** [**https://github.com/opencv/opencv/archive/4.1.1.zip**](https://github.com/opencv/opencv/archive/4.1.1.zip)

**pi@raspberrypi:~ $ wget -O opencv\_contrib.zip** [**https://github.com/opencv/opencv\_contrib/archive/4.1.1.zip**](https://github.com/opencv/opencv_contrib/archive/4.1.1.zip)

***Descomprimir*** en linux raspbian:

pi@raspberrypi:~ $ unzip opencv.zip

pi@raspberrypi:~ $ unzip opencv\_contrib.zip

pi@raspberrypi:~ $ mv opencv-4.1.1 opencv

pi@raspberrypi:~ $ mv opencv\_contrib-4.1.1 opencv\_contrib

***Como Buscar un File en Linux:***

**pi@raspi:~ $ find /home/pi -name image\*.jpg**

/home/pi/facial\_recognition/dataset/Z/image\_0.jpg

/home/pi/opencv/doc/py\_tutorials/py\_gui/images/image\_display.jpg

/home/pi/opencv/doc/py\_tutorials/py\_core/images/image\_arithmetic.jpg

/home/pi/mu\_code/image\_2.jpg

/home/pi/mu\_code/image\_0.jpg

/home/pi/mu\_code/image\_5.jpg

/home/pi/mu\_code/image\_1.jpg

/home/pi/mu\_code/image\_3.jpg

/home/pi/mu\_code/image\_4.jpg

***Instalar OpenCV en Windows es muy Facil:***

***Fuentes :*** [***https://pypi.org/project/opencv-contrib-python/***](https://pypi.org/project/opencv-contrib-python/) ***ó https://github.com/opencv/opencv***

Antes q anda actualizamos :

Go where is located your Python application path, which is the folder where you originally [installed Python](https://datatofish.com/install-python/).

C:\ python -m pip install --upgrade pip

Instalamos:

C:\ pip install opencv-contrib-python // o tambien como sigue;

C:\ pip install opencv-contrib-python –upgrade // full packages

ó

C:\ pip install opencv-python // bsic packages

**Test Library:**

C:\> python

>>> import cv2

>>> print(cv2.\_\_version\_\_)

'3.4.0' # your version may be a newer one

**Un Progr p Testear la Camara:**

import numpy as np

import cv2

cap = cv2.VideoCapture(0)

while(True):

# Capture frame-by-frame

ret, frame = cap.read()

# Our operations on the frame come here

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

# Display the resulting frame

cv2.imshow('frame',frame)

cv2.imshow('gray',gray)

if cv2.waitKey(20) & 0xFF == ord('q'):

break

# When everything done, release the capture

cap.release()

cv2.destroyAllWindows()

**Apache WebServer.-**

For deafault, Apache is available in **the default Raspbian** repositories and **the installation** is pretty straightforward. That's it,at this point you have Apache **web server installed** on **your Raspberry Pi**.

pi@raspberrypi:~ $ sudo apt update

pi@raspberrypi:~ $ sudo apt-get upgrade

pi@raspberrypi:~ $ sudo apt install apache2 // install apache

**Installing PHP**

If you want to serve dynamic PHP files on your Raspberry Pi you’ll need to install PHP and PHP Apache module:

pi@raspberrypi:~ $ sudo apt install php libapache2-mod-phpCopy

Once the packages are installed to enable the PHP module, restart the Apache service:

pi@raspberrypi:~ $ sudo systemctl restart apache2

**HowTo STOP RESTART START mi Apache2 web Server:**

pi@raspberrypi:~ $ sudo /etc/init.d/apache2 stop // para mi servidor

[ ok ] Stopping apache2 (via systemctl): apache2.service.

pi@raspberrypi:~ $ sudo /etc/init.d/apache2 start // start mi server

[ ok ] Starting apache2 (via systemctl): apache2.service.

pi@raspberrypi:~ $ sudo /etc/init.d/apache2 reload // recargar la configuracion

[ ok ] Reloading apache2 configuration (via systemctl): apache2.service.

pi@raspberrypi:~ $ sudo /etc/init.d/apache2 restart // restar server.

[ ok ] Restarting apache2 (via systemctl): apache2.service.

NOTA: You can use ***apt-get*** remove ... or ***apt-get purg***e ...  
purge removes all of the configuration files as well as the software package. remove only gets rid of the software package.

***GitHub***

**GitHub for Linux :** [**http://git-scm.com/**](http://git-scm.com/)[**http://git-scm.com/downloads**](http://git-scm.com/downloads)[**http://git-scm.com/download/linux**](http://git-scm.com/download/linux)

**‘GIT’** sirve para compartir y desarrollar paginas web, nos va permitir crear una copia del repositorio original, y generaremos versiones de forma automatica, para tener un mejor orden y rapides de ejecutar projectos.desde GitHub no se puede crear un repositorio desde la conosola, hay un tip para esto de lo contrario hay q crearlo manualmente desde la misma pag: http://www.github.com

**0ro.- Debemos Crear nuestro Repositorio en nuestra cuenta de GitHub en la Web:<desde linux>**

**jctesla/Raspi**

**1ro debes suscribir tu cuenta git en el S.O. Linux:**

**$ git config --global user.name "jctesla"**

**$ git config --global user.email "juancarlos.dergan@gmail.com"**

**2do verificas q en el config de github contenga las credenciales:**

**$ git config -l**

**credential.helper=wincred**

**user.email=juancarlos.dergan@gmail.com**

**user.name=jctesla**

**3ro El Serv de Git pide autenticacion con llave 'KeyPublic', con la conección SSH :.**

**a.-** **verifico si ya existe llave publica, listando en este directorio: $ ls -al ~/.ssh**

**b.- sino hay llave publica :. vamos al dir de ssh:**

**$ cd ~/.ssh**

**c.- generamos la llave publica: < te va pedir un nombre p la llave y un contenido:en mi caso nombre=mykey, clave=miclab.**

**$ ssh-keygen**

**nota: podemos usar asi tambien:**

**$ cd ~/.ssh && ssh-keygen**

**4to Verificamos q la llave publica se haya creado:<ej mi llave se llama mykey>**

**$ cd /home/pi/miRepositorio //regreso al dir q quiero copiar al repositorio de github.**

**$ ls -al ~/.ssh // verifico sy ya existe la llave publica; extension “.pub”.**

**total 20**

**drwx------ 2 pi pi 4096 Jun 27 03:54 .**

**drwxr-xr-x 32 pi pi 4096 Jun 27 03:56 ..**

**-rw-r--r-- 1 pi pi 3316 Jun 27 03:42 known\_hosts**

**-rw------- 1 pi pi 1856 Jun 27 03:54 mykey**

**-rw-r--r-- 1 pi pi 390 Jun 27 03:54 mykey.pub**

**5to Verificamos q el SSH este corriendo:**

**$ eval "$(ssh-agent -s)"**

**Agent pid 2207**

**6to Agregamos al agente del SSH la llave: la clave de mi llave es miclab = \*\*\*\*\*\***

**$ ssh-add ~/.ssh/mykey**

**Enter passphrase for /home/pi/.ssh/mykey: \*\*\*\*\***

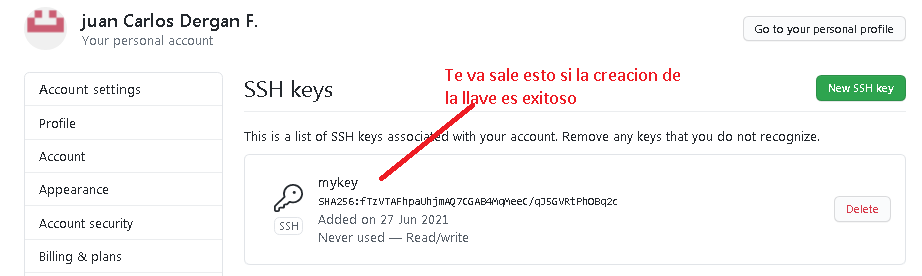
**Identity added: /home/pi/.ssh/mykey (pi@raspi)**

**7to Copiamos la llave creada de linux a nuestra cuenta de Git en la Web:**

**pi@raspi:~/.ssh $ cat mykey.pub | xclip // Instalamos previamente esta herramienta ‘xclip’ de copy page clipboard.**

**ó**

**$ cat ~/.ssh/mykey.pub | xclip**



**8o Pegamos la LLave a nuestra cuenta de GitHub en la web:**

**cuenta: jctesla http://www.github.com**

**entramos a:**

**Settings(click) --> SSH and GPG keys(click) -->**

**(opcion)SSH Key --> new SSH key(click)**

**en Titulo: mykey Body: <pegamos lo q copio el clipboard>**

**9no Una vez creado nuestro repositorio en la Pagina de github, ya podemos crear nuestro dir local en linux para ir copiando a nuestro nuevo repositorio remoto:**

**$ mkdir project**

**$ cd miRepositorio**

**$ git init // crea una dir virtuale ‘.git’ oculto, donde se va almacenar todo los commit y logs q ejecutemos.**

**$ git add . // Agrego todo los files del directorio actual.**

**$ git commit -m 'MI Copia a mi Repositorio'**

**$ git branch -M main //Creamos un branch con el nombre por defecto main, pero si qieres con otro <nombre> debes cambiar en // la linea mas abajo del comando git push origin <nombre>, esto sirve si son varios las personas q // cambian un mismo file.**

**$ git remote add origin git@github.com:jctesla/Raspi.get**

**$ git push origin main // ejecutamos todo los commit creado**

**NOTA**: comandos GIT

**$git log** **//nos permite ver que commit(comandos de git) hasta este momento hemos ejecutado.**

**$git show** **// lo mismo q git log pero ademas nos permite ver su contenido.**

Herramienta ***XCLIP***, p hacer un ***Copy to clipboard***:

**$ sudo apt install xclip**

**$ xclip -selection clipboard < mikeypublica //copia el contenido de este file al clipboard (es como hacer Copy-Page)**

Para ver un **pid** programa: Ej ***ssh-agent***

**$eval $(ssh-agent -s)**

**Agent pid 2207**

Para Clonar a mi Maq Local:

**$git clone git@gitlab.com:gitlab-tests/sample-project.git**

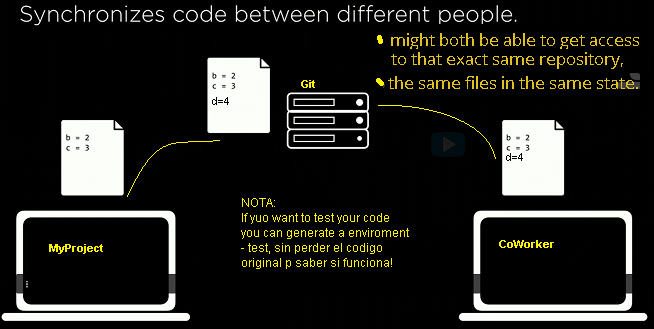
Vemos los files bajados:

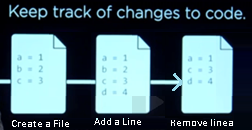
**$cd sample-project**

**$git clone https://github.com/midudev/curso-gratuito-svelte**

**Forma de Operar GitHub:**

Lo que va hacer “Git”, es keep tracking changes in your code to save the actual version an so on;





De esta herramienta existe en varias marcas, pero el más popular es ***“GitHub”***; va permitir crear escenarios de cooperación de trabajo, para documentos, programas, etc y de alguna forma generar mayor rapidez. En la sgnt figura indica un caso de trabajo q a veces necesitamos probar un codigo antes de integrarlo al codigo original, y para esto creamos un escenario donde probamos el nuevo codigo y si funciona lo integramos o sino solo mantenemos o regresamos al codigo original sin lso cambios.



El codigo el file siempre se Va copiar de alguna forma si

Queremos regresar al inicio es totalmente factible

NOTA:

**Usando SSH** desde Windows: me pide el publickey

D:\linuxSpace>git clone [git@github.com:jcdtesla/Rapsi.git](mailto:git@github.com:jcdtesla/Rapsi.git)

**Usando HTTPS** desde Windows: no ide nada

D:\linuxSpace>git clone https://github.com/jctesla/Raspi

Ej: Desde Win7

Cree un file “desdewindows.txt” y modifique el que baje del repo jctesla/Raspi “pipLista.txt”, al correr el git status, me indica el modificado y el creado y si aun no lo he enviado al Repo.

D:\linuxSpace\Raspi>***gi t status***

On branch main

Your branch is up to date with 'origin/main'.

**Changes not staged for commit:**

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

**modified: pipLista.txt**

**Untracked files:**

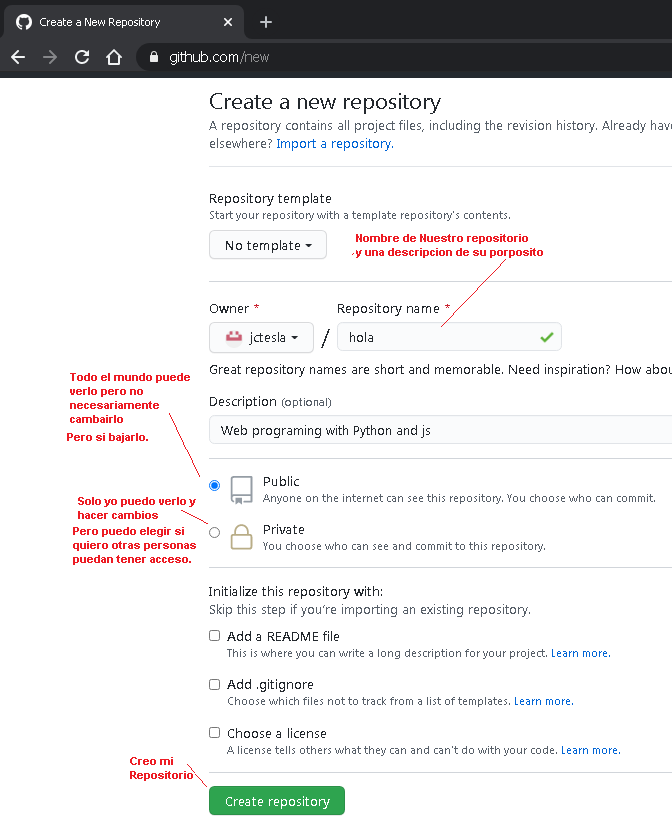
(use "git add <file>..." to include in what will be committed)

**desdewindows.txt**

no changes added to commit (use "git add" and/or "git commit -a")

Luego de subirlo con : git push, y luego correr git status, saldrá =***… nothing to commit…***

**CREAR NUESTRO REPOSITORIO en WINDOWS:**



**Nota:** <http://git-scm.com/> **y** http://git-scm.com/downloads

Git is a [free and open source](http://git-scm.com/about/free-and-open-source) distributed version control system designed to

handle everything from small to very large projects with speed and efficiency.

**1o** Como Crear nuestro Repositorio:

Navegamos a ***https://github.com/new***

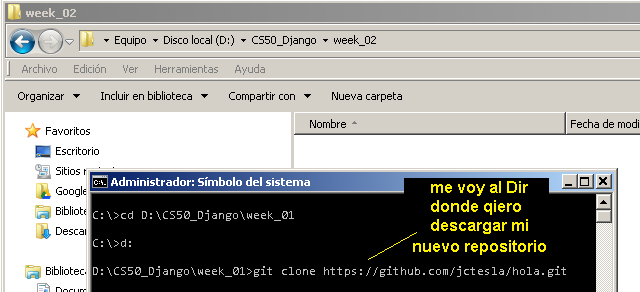
Creamos el repositorio, publico para luego bajarlo a nuestra pc, de esta forma se

demuestra el tracking del projecto, apezar que esta vacio.**(nota:** ya debes contar con un

user y Clave y debes logearte previamente**)**

**2o** elijo donde quiero bajar mi nuevo repositorio:

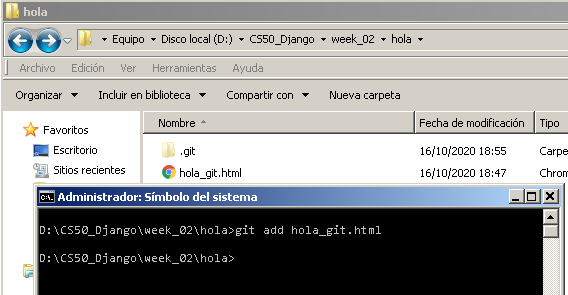
**D:\ CS50:Django\week\_02\git clone https://github.com/jctesla/hola.git**



**(nota : se creara un sub-directorio con el nombre \hola\<copia>**

debe haber instalado previamente el git batch: [***https://git-scm.com/downloads***](https://git-scm.com/downloads)**)**





**3o** Trabajamos los files q necesito cambiar, por ej. mi file “hola\_git.html”,

Una vez terminado, puedo subirlo a mi repositorio;

**D:\ CS50:Django\week\_02\hola\ git add hola\_git.html**

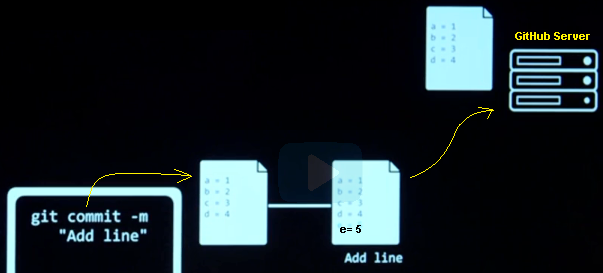
Le dejimos al Git que vamos a copiar un unico file, no toda la Carpeta, es decir

podemos copiar todo pero mejor es cambiar Solo el file q hemos modificado.

Hasta este momento no hemos ejecutado la sentencia solo le Hemos señalado

al Git lo que queremso copiar el snapShoot es Acontiuacion.

**4o** Ejecutamos copia en mi repositorio para luego subirlo al repositorio del servidor de GitHub



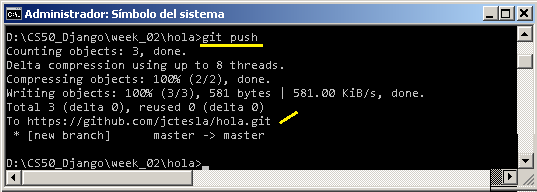
**D:\ CS50:Django\week\_02\hola\ git commit –m “Agrege una Lineas”**

**5o** debemos saber el estado de la copia en mi repositorio para enviarlo Al servidor GitHub.

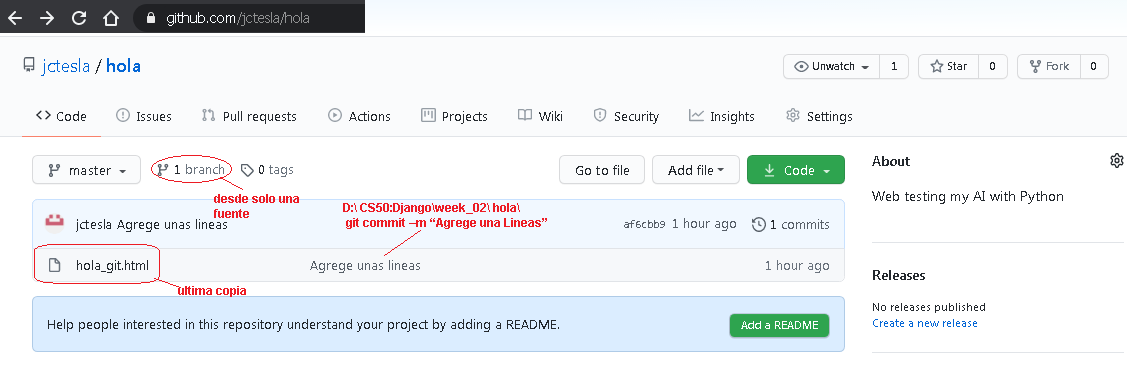
**D:\ CS50:Django\week\_02\hola\ git status**

**6o** Fuerzo a q todo los cambios hechos en mi repositorio local, se reflejen en la web de GitHUb.

**D:\ CS50:Django\week\_02\hola\ git push**

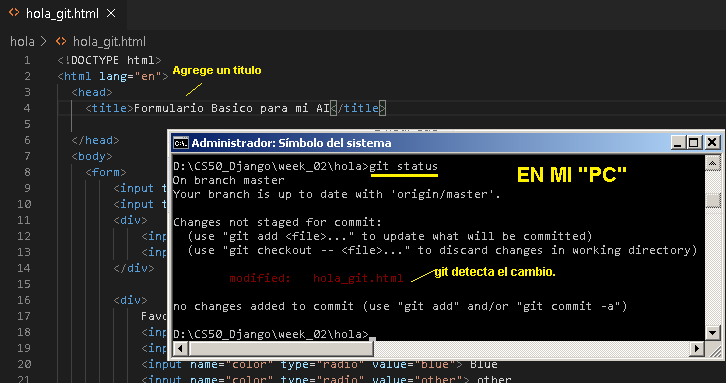


**7o** Resultado en la pagina del repositorio:



**8o** Si quieres hacer algunas modificaciones.-

Vamos a nuestro file hacemos algunas modificaciones y con el comando ***git status***, nos va decir que cambios se detectaron( todo esto en nuestro repositorio local) :



En la fig vemos q alodificar un file del proyecto, ej:

“hola\_git.html” al momento de hacer un git status

El git local detecta los cambios de todos los files q

Hemos modifcado,es como un reporte de todo

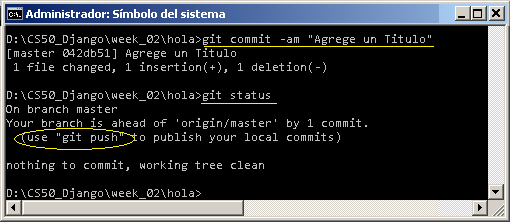
Con esta seguridad pordemo revisar y procesde a

subirlo

.

**D:\ CS50\_Django\week\_02\hola\git status**

**9o** antes de subir los files modificados, creo un commit pero considerarando todo los Cambios y solo de un file,



**D:\ CS50:Django\week\_02\hola\git commit –am “Agrege un Tiulo”**

**D:\ CS50\_Django\week\_02\hola\git status**

Para Actualizar mi Repositorio de GitHub;

**D:\ CS50:Django\week\_02\ hola\git push**

**10o** Ahora pensemos que el file q esta en mi repositorio de GitHub

sufre cambios, y es más actual que el que tengo en mi “pc”;

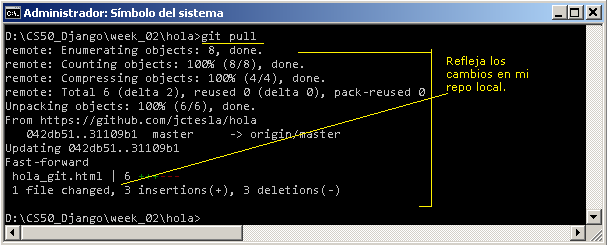
Como se ve e la figura, hice un cambio en la misma pagina de

GitHub.



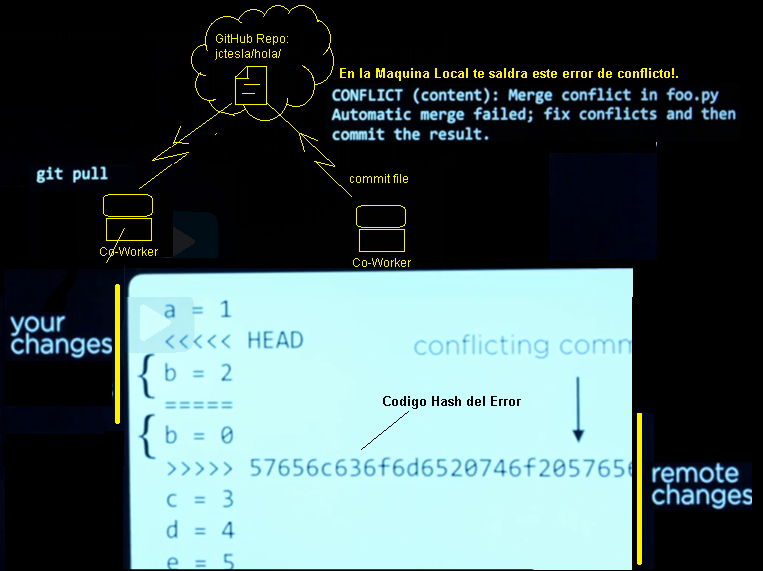
Ahora de mi *Repositorio de GitHub a mi Repositorio Local*:

**D:\ CS50:Django\week\_02\ hola\git pull**



Lo que ahora es ver si hay una actualización en el repositorio de GitHub y copiarlo a mi repositorio para actualizarlo con la versión más actualziada.

**11o** Que sucede si ambos coworker hacen



un cambio en el mismo file?:

a esto Git lo llama “Merge Conflict”,

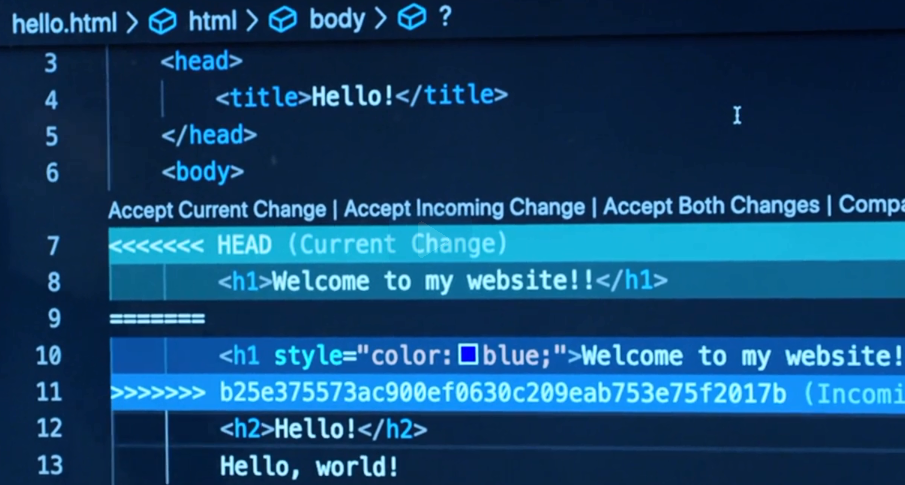
Cuando sucede esto podemos hacer

Cambios que nos convenga de alguna

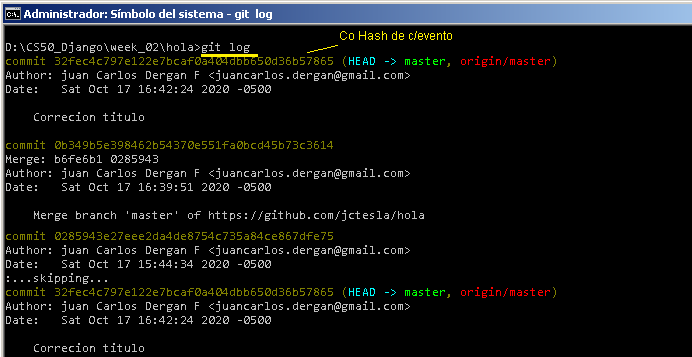
Forma, quitar las lineas del cambio

Remoto o dejarlas según sea el caso.

Ejemplo de Conflicto!:



**12o** Comando para ver los Log todo los



comandos y procesos que sena ordenado;

**D:\ CS50:Django\week\_02\ hola\ git push\git log**

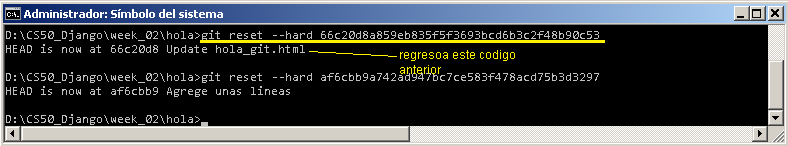
nota: pàra salir del “Log”, presiona ‘q’ de quit

**13vo** Comando para rollback : lo que va hacer es tomar el repositorio actual y lo va regresar al ultimo anterior.

Puedo regresar a cualquier estado anterior indicandole el codigo de has obtenido en el ***“git log”*** para regresar a ese codigo.

**D:\ CS50:Django\week\_02\ hola\ git push\ git reset –hard <commit hash>**

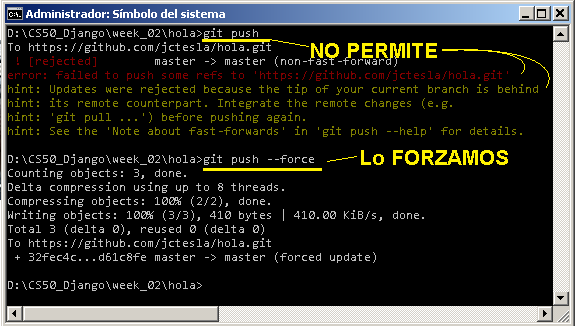
Ej:



**NOTA**:

Si en caso se hace un actualizacion en la nube, y en mi repolocal y c/vez q quiero hace un Git push me sale el msg:

**“hint: Updates were rejected because the tip of your current branch is behind”**

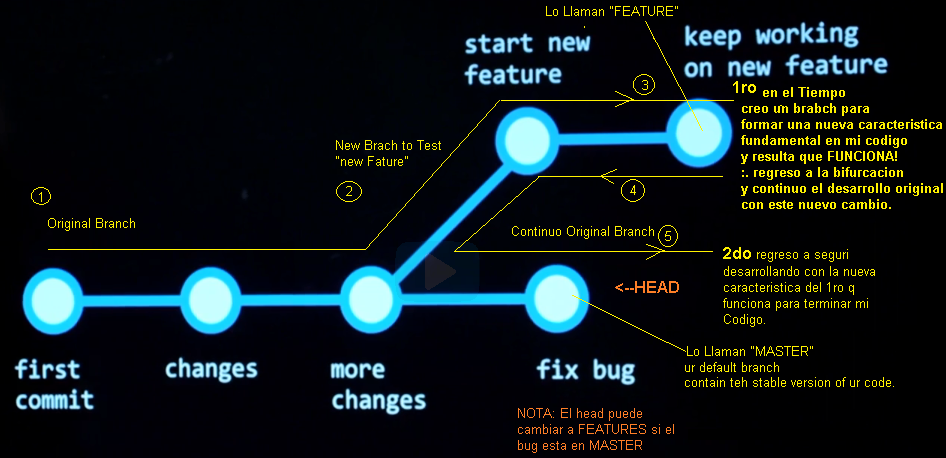


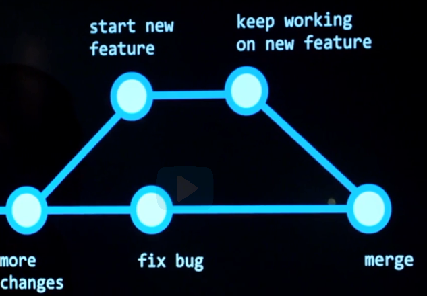
**No me deja por esta cola :. Podemos forzarlo:**

**git push --force-with-lease**

Ahora vemaos como podemos operar con Git,

Branches are Git’s way of working on different parts at the repository at the same time.





Finalmente Terminar asi todo mergen al mismo punto MASTER

para terminar el desarrollo.

**CRENDO BRANCHES.-**

**Vamos a crear ramificaciones para diferetes cambiosde un proyecto, supungamos q creamos 3 ramificaciones**

**1ro verificamos cuantas hay?**

**git branch**

**out:**

**\* master**

**2do creamos 2 mas;**

**git checkout –b style**

**out:**

**Switched to a new branch 'style'**

**3ro creamos un branch mas:**

**git checkout -b title**

**out:**

**Switched to a new branch 'title'**

**Copiar y Crear Directorio de mi PC a mi Repositorio (nota luego de los pasos 1,2,3 podemos subir al Repositorio para que haga una copia:**

**C:\nValiGVCO\git add .**

**C:\nValiGVCO\git commit -m "Se Agregarons Nuevos Folders"**

**C:\nValiGVCO\git push --force**

**Actualizo un File= default.asp de mi PC a mi Repositorio GitHub:**

**C:\nValiGVCO\git add default.asp**

**C:\nValiGVCO\git commit -m "Se Agregarons Nuevos Folders"**

**C:\nValiGVCO\git push --force**

**DOCKERS:**

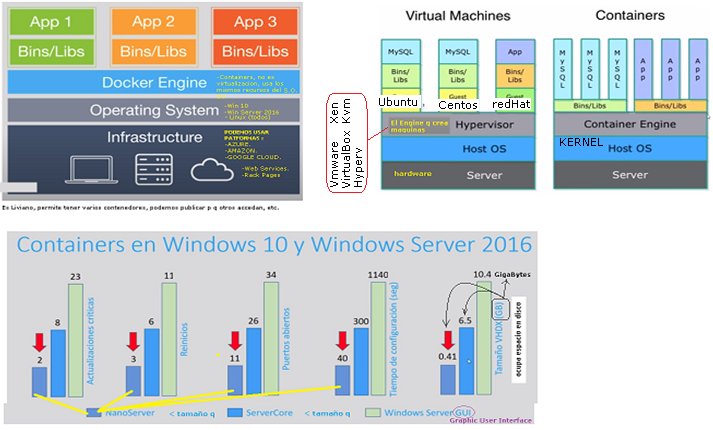
**VM.-** es parecido a los Dockers, akgunos proveedores de esta tecnologia;

* Vmware
* VirtualBox
* Xen
* Hyperv
* Kvm

En el Docker a diferencia de VM, usa el mismo Host OS, sus recursos los comparte con los otros contenedores de forma idependiente, lo que mejora el rendimiento de la Maquina Fisica.

Es decir el Contenedor contiene solo tu aplicación con las librerias(binarios) q necesita del SO en el que se creo para poder funcionar, en el VM en realidad tienes todo el SO de c/app por lo que es más pesado.

Por ejemplo en Contenedores puedes correr alrededor de 100 contenedores, mientras a una VM no se podrian > 10 x el hrdwr.



Los Dockers son mas Ligeros que el VM, ya que el VM carga todo un SO y despues carga los recursos (lib , componentes) y al final la Aplicación que he creado, en el caso del los Docker(Contendor), solo corres una Imagen!.

La **Imagen** es una copia de nuestra aplicación junto con los archvios de la distribucion de linux, es decir la Imagen contendría.

* SO = puede ser cualquier distribucuin de linux ya q usa el mismo kernel del SO HOST.
* Sowftwr = todo el softwr q mi aplicación necesita x Ej. Si mi App es PhP, necesito el Apache Serv, los binarios de PhP,etc.
* Aplicación = es mi aplicación propiamente, es decir el codigo que quiero correr embase a la imagen del SO y Sowftwr.

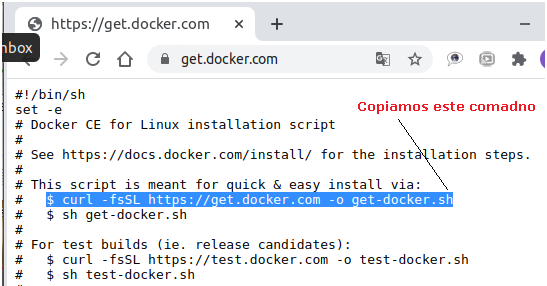
El **DockerFile**:

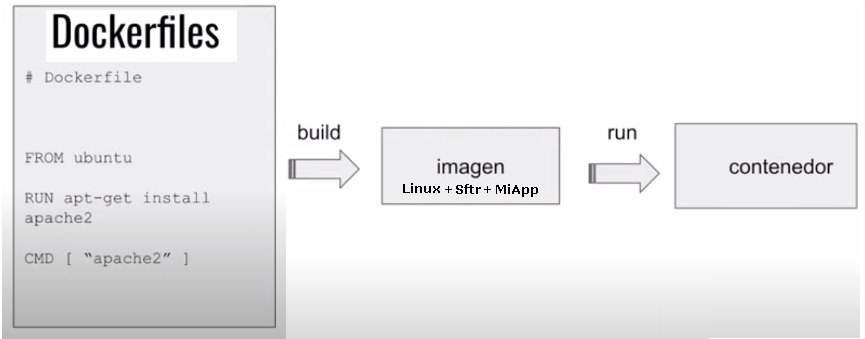
Es un file que indico la reutilización de una imagen que ya exista (o la puedes crear) y el nombre de tu aplicación, para luego recrear una imagen que es la que va ir en el Contenedor

Para poder unir en una sola imagen mi aplicación podemos usar una imagen q ya exista, lo bajamos y atravez de un archivo lo unimos y lo volvemos a compilar en una segunda imagen para luego decirle al engine que lo corra.

**NOTA:**

pueden crear varios contenedores con aplicaciones q son de diferentes Linux (Ubuntu , Centos, RedHat, etc) en una mísma maquina que tenga el Engine de Docker.





Hay un Servidor publico donde se almacenan todas las IMAGEs q podemos bajarlas para usarla, puedes subir tu imagen a ese mismo repositorio y asi otra persona q necesite una configuracion de imagen parecido puede usarlo.

$docker run hellow-world // va correr la imagen “hello-world”, sino lo encuentra localmente lo baja del repositorio.

$docker pull alpine // permite ver que imágenes hay en el repositorio.

-

-

-

$docker pull apline:3.7 // de la lista q me dio bajo la version 3.7

$docker run alpine:3.7 ls –l // una vez bajado la imagen puedo pedir un ls al contenedor, y va vizualizar los // directorios del mismo.

$docker run –it alpine:3.7 sh // it = iterativo, va permitir entrar al shell de esde contenedor con su prompt y todo en // tiempo real.

NOTA:

Puedo abrir otro terminal, y verificar como se esta usando el contendor alpine.

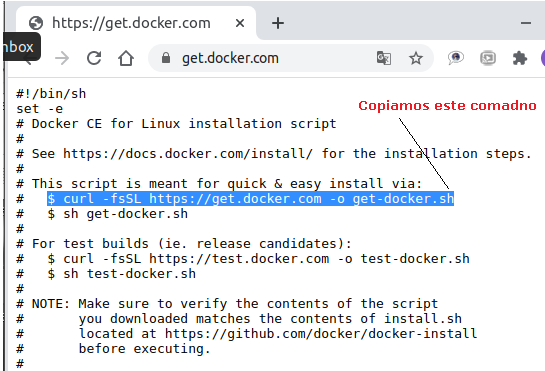
$docker ps // permite ver q contenedor se esta usando.

$docekr exec –it <ID> sh // para q sea terminal y sea iterativo.(tiempo real), abra un shell y puedo hacer un ‘ls’.



INSTALACION:

Hay varias formas de instalarlo una de las mas practicas seria ir [**http://get.docker.com**](http://get.docker.com)y nos va a dar un script.



**Compilar un app C++.-**

Es n lenguaje compilador a dif de python, necesita un compilador para luego poderlo ejecutar, es decir a dif de kotlin q es un interprete, el necesita tener instalado el sdk de python, de lo contraio no se podria correr, C++ en este caso necesita un compialdor q agrupa a todas las libr y convierte todo este codigo inlckuio el .c a un resultado binario.

Desde Linux:

pi@raspberrypi:~ $ cat hola.c // uso para editar un prog simple en c++.

#include <stdio.h>

void main (void)

{

/\* A print statement \*/

printf ("Hello world!\n");

}

pi@raspberrypi:~ $ gcc –o miprog hola.c // va compilar el prog hola.c en miprog binario

pi@raspberrypi:~ $ ls

Bookshelf Documents hola.c mu\_code Pictures Templates

Desktop Downloads miprog Music Public Videos

pi@raspberrypi:~ $ ./ miprog // corremos el binario el rslt del prog hola.c

Hello world! // la extension del file *miprog* puede ser cualqiera

pi@raspberrypi:~ $

otro Ejemplo:

#include <stdio.h>

int main (void){

int a = 0;

printf("Do Loop a < 100 o a =20, Inicialmente a = %d:\n",a);

do{

printf ("a = %d\n", a);

a++;

if (a == 20){

printf("Si a = 20, break:\n");

break;

}

} while (a < 100);

printf ("a = %d y genera BREAK \n", a);

}

***MinGW en* WINDOWS: *doc =*** <https://www.ics.uci.edu/~pattis/common/handouts/mingweclipse/mingw.html> **^ *installer =*** http://sourceforge.net/projects/mingw/files/

***GNU Compiler Collection* .-** es un conjunto de [compiladores](https://es.wikipedia.org/wiki/Compilador) creados por el proyecto [GNU](https://es.wikipedia.org/wiki/GNU).

***MinGW***.- anteriormente conocido como MinGW32,es un compilador de la familia GCC (*GNU Compiler Collection*) diseñado para ejecutarse de forma nativa en Windows, es decir,

***Lenguajes***.-

En su versión 4.6 incluye ***front ends*** para:

*- C (gcc)*

*- C++ (g++)*

*- Java (gcj)*

*- Ada (GNAT)*

*- Objective-C (gobjc)*

*- Objective-C++ (gobjc++)*

*- Fortran (gfortran).*

También está disponible, aunque no de forma estándar, soporte para

*- Go (gccgo)*

*- Modula-2*

*- Modula-3*

*- Pascal (gpc)*

*- PL/I, D (gdc)*

*- Mercury*

*-VHDL (ghdl)*