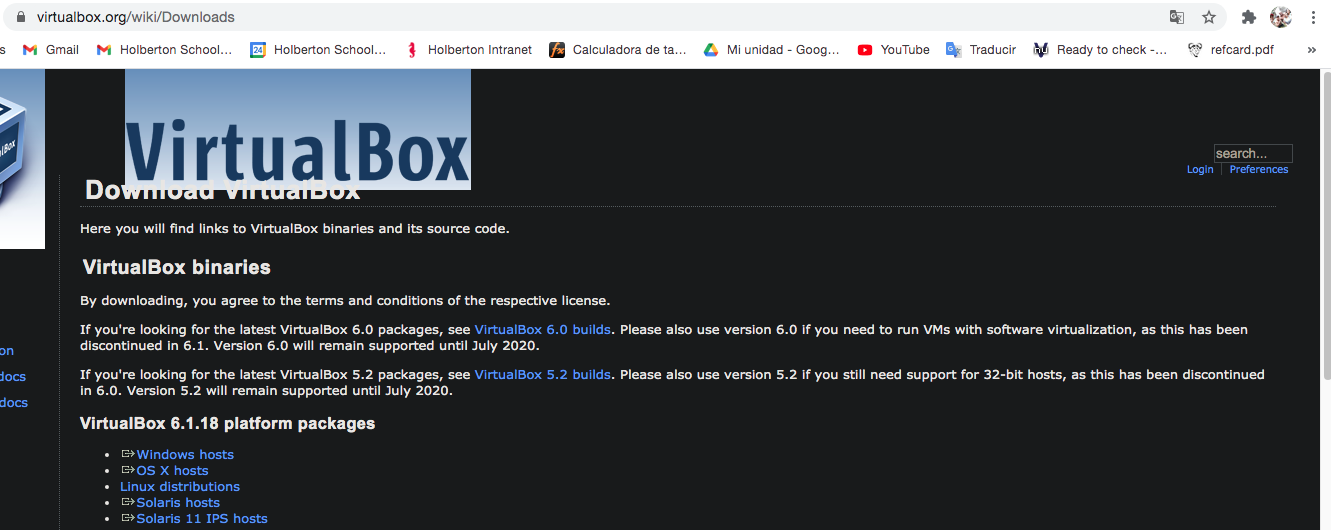
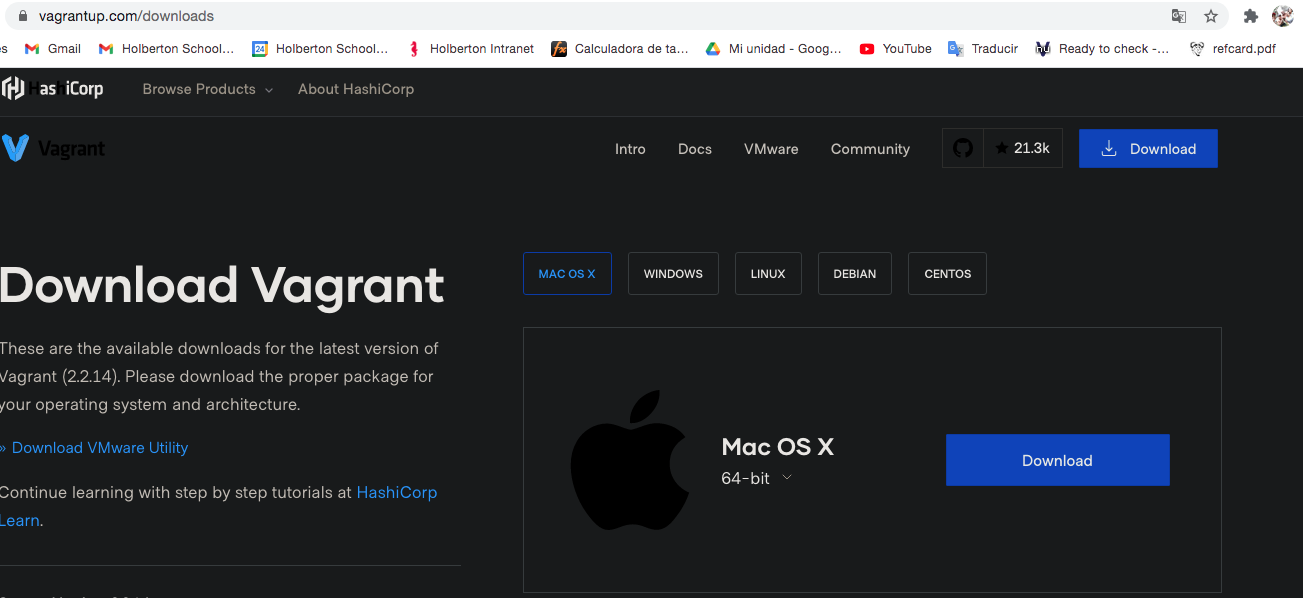
***Proceso de instalación de Virtual Box y Vagrant***

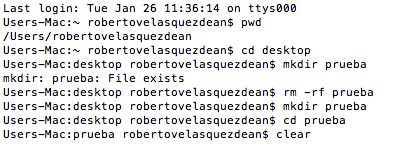
1. Primero vaya a sus páginas oficiales y busque el programa correspondiente a su sistema operativo, tal que así (Yo soy Mac, por lo que necesito programas compatibles a Mac).



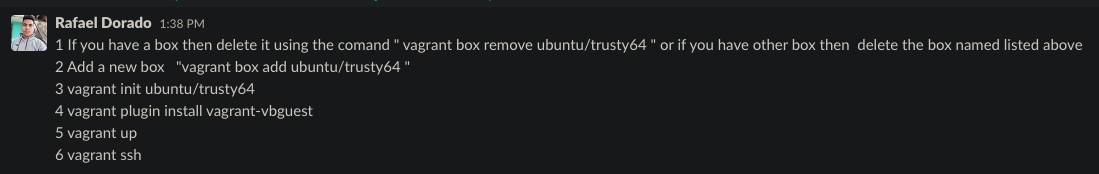


***Los descarga y luego crea el directorio en el cual desea trabajar (lo cual hará de la mano del VirtualBox y del Vagrant) Caso aparte: toda carpeta creada, en un proyecto, debe tener un archivo README.md incorporado dentro de ella y dentro del archivo README.md, sí o sí, debe escribir algo dentro.***

***pd: adicionalmente debe hacer un salto de linea siempre que termine con su contenido textual dentro de los archivos que crea, sean o no README.md. Continuemos. Este es el conjunto de lineas de codigo que debe ejecutar por cada directorio creado, del cual le interese manipular:***



Luego ejecute los siguientes comandos para poder incorporsarse dentro del virtual box y del vagrant. (inicie desde el punto 2. si ya hizo el anterior proceso -imagen 1-)



Y listo! ya podrá trabajar con un directorio que está fuera de su sistema operativo con toda comodidad. (si requiere hacer uso de GIT y de Emacs, o de cualquier tipo de archivo que requiera importar desde la terminal, debe instalarla. Así será siempre si el directorio es virgen y recien a inicializado en él VirtualBox y Vagrant)

Ahora bien, para que no se nos arruine VirtualBox ni Vagrant, a la hora de terminar de “codear”; usted tiene que apagar la maquina virtual, ¿cómo se hace eso? Pues, tal que así:

1. Cierra su terminal con Exit, verifique el comando propiamente para Linux si usa Mac (“Exit” se ejecuta dos veces hasta que la terminal se encuentre por fuera de la maquina virtual y ya esté propiamente dentro de su sistema operativo, el propio de su pc o laptop).

2. Ejecute, ya estando la terminal sincronizada nuevamente al sistema operativo propio a su dispositivo, el siguiente comando: “vagrant halt”. ***Ojo, todo esto debe hacerse dentro del directorio manipulado.***

Ahora bien, supongamos que desea, en una nueva ocasión volver a manipular este mismo directorio vinculado a una maquina virtual; toca primeramente asegurarse de estar ubicado en dicho directorio dentro de la terminal (ejecute el comando “pwd” para cerciorarse, y si no esta, vaya a ese directorio con el comando “cd”) luego vuelva a encender la maquina virtual, tal que así (ejecutando los siguientes comandos):

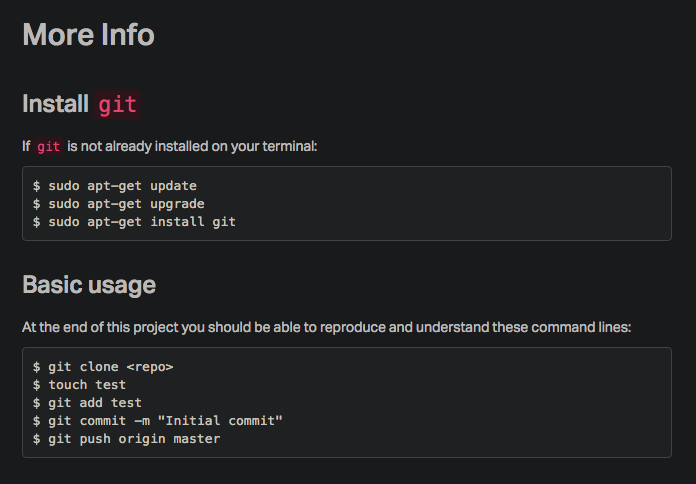
1. vagrant “up”  
2. vagrant “ssh”

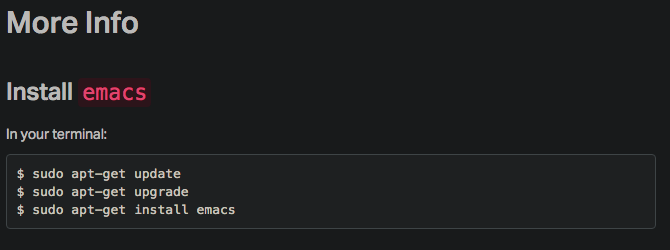
Y listo! De nuevo el directorio de su interés está trabajando dentro de la maquina virtual anteriormente vinculada.

***ACLARACIÓN ADICIONAL: si ya tenía con anterioridad instalado el GIT y, además, inicializado en el directorio que ya había sido emulado previamente; entonces, y atención, no puede inicializar más este directorio; es decir, no haga más esto: git init. NO.***

Ahora, si por ejemplo desea trabajar con otro directorio (ya sea uno nuevo o uno ya existente), del que nunca antes había vinculado una maquina virtual, pues le toca repetir los pasos de Rafael Dorado en su totalidad. Del resto, igual.

Programas de interés que desee instalar por cada nuevo directorio que desee manipular con una maquina virtual (y sus comandos de instalación dentro de la terminal)





Ahora bien, las descargas internas de estos programas sobre la terminal exigen un permiso: el permiso del propietario original del dispositivo. Si desea decirle a la terminal, de forma anticipada, que dé todos los permisos necesarios para manipular dentro de ella servicios o programas externos a ella; ustes puede hacerlo ejecutando el siguiente comando, tal que así: *“sudo su”.*

*Veamos qué es lo que hay detrás de* ***“sudo”*** *o de* ***“sudo su”****.*

*En sistemas operativos del tipo Unix, el* ***superusuario o root*** *es el nombre convencional de la cuenta de usuario que posee todos los derechos, a nivel general, en todos los modos (no sólo para importar archivos externos a la terminal). Normalmente es la cuenta de administrador. Entonces, cuando decimos* ***“sudo su”*** *lo que estamos haciendo es ejecutar el comando que nos permite ser ese* ***super usuario*** *para* ***todo.*** *Sin embargo, si sólo quisieramos ser un “superusuario” para ejecutar una función o script (lo explicaremos más adelante) en especifico, no habilitamos la función de tener todos los derechos sobre todas las cosas (sudo su), no. Sino, sólo sobre el script en especifico; en ese caso, sería ejecutar el comando:* ***“sudo” y al lado el script en cuestión*** *que queremos ejecutar como superusuario.**Tal que así, por ejemplo:*

*“****sudo******apt-get install git****”, lo que estamos diciendo es que queremos tener las funciones de un superusuario solamente para instalar git dentro de nuestra terminal. Así funciona.*

***Salir de superusuario:*** *Después de ejecutar el comando* ***su,*** *tenemos una nueva sesión de shell como superusuario. Para salir de la sesión de superusuario, escriba* ***“exit”*** *y regresaremos a su sesión anterior.*

***Nota:*** *En la mayoría de las distribuciones modernas, se utiliza un método alternativo. En lugar de usar su, estos sistemas emplean el sudocomando. Con sudo, uno o más usuarios reciben privilegios de superusuario según sea necesario. Para ejecutar un comando como superusuario, el comando deseado simplemente está precedido por el sudocomando. Después de ingresar el comando, se le solicita al usuario su propia contraseña en el lugar de la del superusuario.*

***Sobre los scripts:*** *Básicamente lo que hace un script es darle función a una variable que está siendo definida por medio de la creación de un archivo. Idealmente este archivo, como está siendo definido como una variable, no debería llevar una extensión (.ext) y, adicionalmente a eso, la primera línea de código dentro del archivo debe llevar la siguiente declaración:* ***#!/bin/bash.***

*En la segunda linea sí debe situar la función propiamente, lo que desea que se ejecute al llamar nuestra nueva variable definida por medio de este archivo. En el documento “Shell bases” podemos revisar ciertos ejercicios con “scripts” y comandos de Shell donde se pone en práctica este tema. Por otro lado, como son “scripts” y ejecutan una función, deben ser convertidos entonces en archivos “ejecutables”; tal que así: chmod u+x nombre\_de\_script; y, para finalmente ser ejecutados, debe ejecutarlo con la siguiente estructura: ./“nombre\_de\_script” (este archivo no se compila, pues, ya es un ejecutable en sí mismo).*