	UNIVERSIDAD AUTONOMA DE OCCIDENTE			<i>Valoración</i>
	FACULTAD DE INGENIERIA			
	DEPARTAMENTO DE AUTOMATICA Y ELECTRONICA			
	ESTUDIANTES:		Fecha:	
CONFIGURACION ENTORNO VIRTUALIZADO CON VAGRANT				

OBJETIVOS

- Configurar el ambiente de desarrollo Vagrant + VirtualBox + Ubuntu que se usará en el curso.
- Aprender como se publican boxes en Vagrant Cloud.

1. Instalar VirtualBox 6.1.12

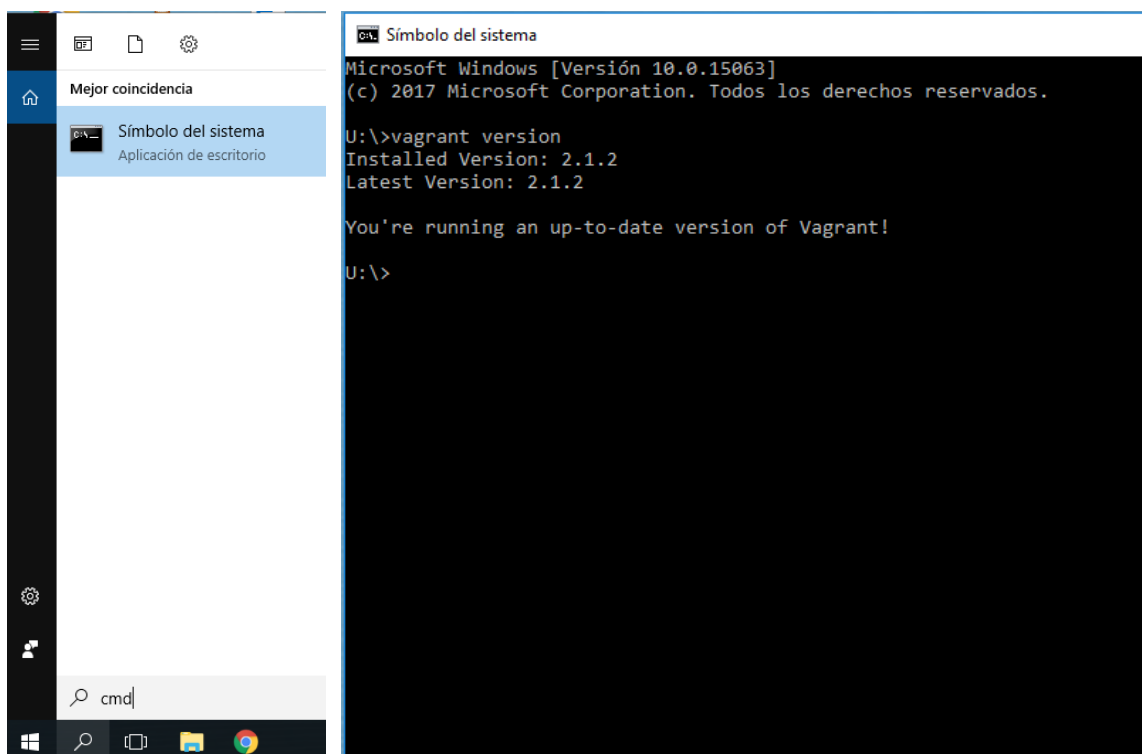
- Descargue e instale VirtualBox del sitio <https://www.virtualbox.org/>

2. Instalar Vagrant

- Descargue e instale Vagrant desde <https://releases.hashicorp.com/vagrant/2.2.9/>

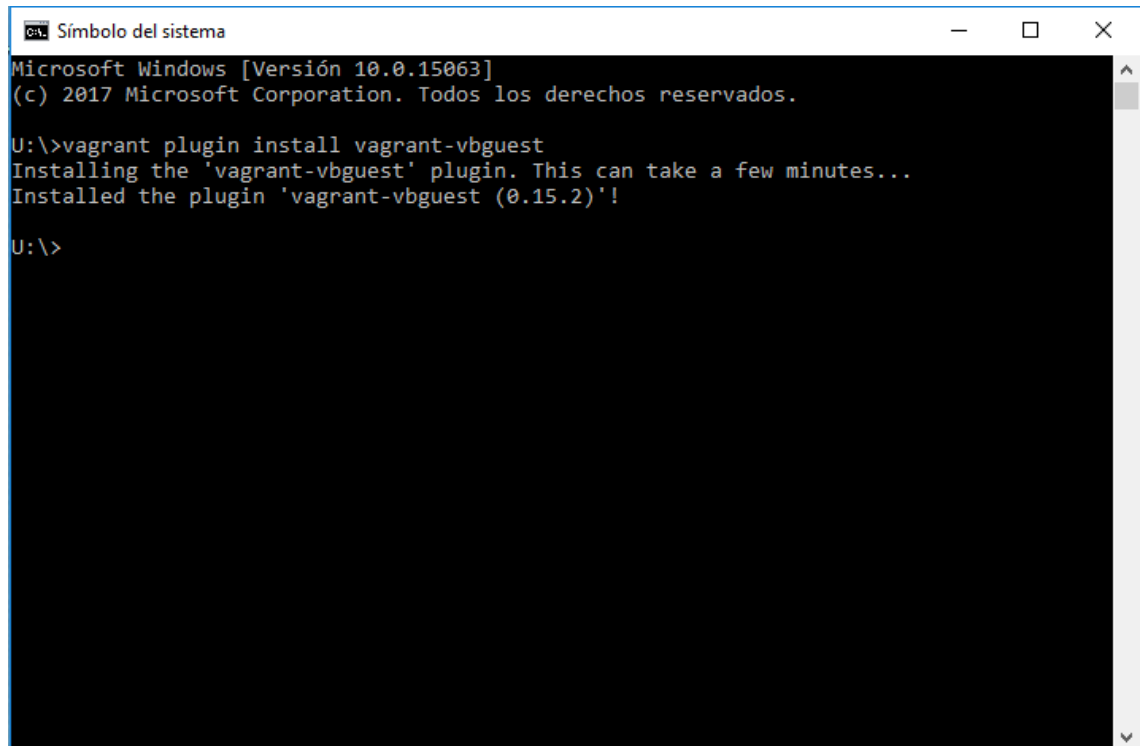
NOTA: Usaremos la versión 2.2.9 para 64 bits (vagrant_2.2.9_x86_64.msi) (para usuarios mac use el mismo archivo con extensión .dmg)

3. Abra una consola de Windows y verifique la versión de Vagrant



- En la misma consola de Windows ejecute el siguiente comando para instalar el plugin vbguest

```
vagrant plugin install vagrant-vbguest
```



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.15063]
(c) 2017 Microsoft Corporation. Todos los derechos reservados.

U:\>vagrant plugin install vagrant-vbguest
Installing the 'vagrant-vbguest' plugin. This can take a few minutes...
Installed the plugin 'vagrant-vbguest (0.15.2)'!

U:\>
```

Este es un plugin de Vagrant para mantener las adiciones de los guest de VirtualBox actualizadas.

4. **Opcional.** Powershell

Para algunos usuarios puede ser mas cómodo usar Powershell, el cual es similar a los Shells de Unix. Este ya viene preinstalado en Windows 10.

Para ejecutarlo, puede usar el buscador de aplicaciones de Windows o ejecutar “powershell” directamente en una consola de Windows, como se muestra a continuación.

```
Símbolo del sistema - powershell
U:\>powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos los derechos reservados.

PS U:\> pwd
Path
----
U:\

PS U:\> c:
PS C:\> pwd
Path
----
C:\

PS C:\> ls

    Directorio: C:\

Mode                LastWriteTime         Length Name
----                -
d-----         24/07/2018   5:46 p. m.         HashiCorp
d-----         22/06/2017   3:44 p. m.         inetpub
d-----          1/02/2018   5:37 p. m.         Intel
d-----         20/03/2018  10:21 a. m.         PerfLogs
d-r-----        24/07/2018   5:37 p. m.         Program Files
d-r-----        23/07/2018   5:49 p. m.         Program Files (x86)
d-r-----        23/01/2018  10:22 a. m.         Users
d-----         24/07/2018   5:48 p. m.         Windows
d-----        23/01/2018  12:18 p. m.         Windows10Upgrade
d-----          4/07/2018   2:20 p. m.         Yopal
-a-----        11/07/2017   9:47 a. m.         52 network.lic

PS C:\>
```

Se puede ver en este ejemplo que se ejecutaron comandos como pwd, ls al estilo Unix. El comando pwd permite ver el directorio actual de trabajo, mientras que el ls el contenido de dicho directorio.

5. Configuración el entorno virtualizado

Muchos servicios de red usan el modelo cliente-servidor. Usaremos dos maquinas virtuales, una que alojará los servicios configurados y otra que los consumirá.

Configurar el Vagrantfile de la siguiente manera (ver instrucciones abajo en "CONFIGURACION").

```
Vagrant.configure("2") do |config|
  config.vm.define :clienteUbuntu do |clienteUbuntu|
    clienteUbuntu.vm.box = "bento/ubuntu-20.04"
    clienteUbuntu.vm.network :private_network, ip: "192.168.100.2"
    clienteUbuntu.vm.hostname = "clienteUbuntu"
  end

  config.vm.define :servidorUbuntu do |servidorUbuntu|
    servidorUbuntu.vm.box = "bento/ubuntu-20.04"
    servidorUbuntu.vm.network :private_network, ip: "192.168.100.3"
    servidorUbuntu.vm.hostname = "servidorUbuntu"
  end
end
```

Este Vagrantfile define dos maquinas virtuales, una llamada servidor con dirección ip 192.168.100.3 y la otra cliente con dirección ip 192.168.100.2, ambas instanciadas

desde un box en el repositorio de bento llamado bento/Ubuntu-20.04.

CONFIGURACION

1. Cree un directorio llamado “prueba”
2. Dentro del directorio prueba cree un archivo llamado “Vagrantfile”.

Para crear un Vagrantfile de ejemplo puede usar el siguiente comando dentro de la carpeta prueba:

```
vagrant init
```

Modifique el Vagrantfile para que tenga UNICAMENTE el contenido mostrado en el ejemplo de arriba (al inicio de la sección)

3. Cree y configure las maquinas mediante el comando `vagrant up` ejecutado desde consola
4. Verifique el Puerto de reenvío para cada maquina virtual. En el ejemplo mostrado abajo, el puerto de reenvío es el 2222

```
$ vagrant up

Bringing machine 'default' up with 'virtualbox' provider...==> default:
Importing base box 'hashicorp/precise64'...==> default: Forwarding ports...

default: 22 (guest) => 2222 (host) (adapter 1)==> default: Waiting for
machine to boot...
```

5. Verifique el estado de las maquinas creadas con el commando

```
vagrant status
```

6. Establezca una sesión ssh con la maquina servidor

```
vagrant ssh servidorUbuntu
```

7. Autenticarse como super usuario

```
sudo -i
```

8. Instalar algunas herramientas para configuración de la red

```
apt-get install net-tools
```

9. Instalar el editor Vim

```
apt-get install vim
```

10. Repita los pasos 5 a 8 para la maquina cliente.
11. Confirme la ip de las maquinas virtuales usando `ifconfig` y pruebe conectividad con el comando `ping`.

6. Boxes

En lugar de construir una maquina virtual desde cero, lo cual puede ser un proceso tedioso, Vagrant usa una imagen base para rápidamente clonar una maquina virtual. Estas imágenes bases se conocen como **boxes** en Vagrant. Después de crear un Vagrantfile debe especificar el **box** a usar.

Instalar un box

El comando para instalar un box es:

```
$ vagrant box add <nombre del box>
```

Ejemplo:

```
$ vagrant box add bento/centos-7.8
```

Ruta de los boxes ya instalados

En las siguientes rutas puede verificar los boxes previamente instalados:

- Mac OS X and Linux: `~/.vagrant.d/boxes`
- Windows: `C:\Users\USERNAME\vagrant.d\boxes`

Verifique los boxes que tiene instalados en la ruta correspondiente.

Los boxes son almacenados globalmente para el usuario actual. Cada proyecto usa un box como una imagen inicial, la cual se clona. De esta manera nunca se modifica la imagen inicial.

Usar un box

Una vez el box haya sido agregado a Vagrant solo necesitamos configurar nuestro proyecto para usarlo como base.

En una nueva carpeta, crear un Vagrantfile con

```
vagrant init
```

Abrir el Vagrantfile y cambiar el contenido por lo siguiente:

```
Vagrant.configure("2") do |config|  
  
  config.vm.box = "hashicorp/precise64"
```

```
end
```

En este caso "hashicorp/precise64" debe corresponder a uno de los boxes previamente instalados

Encontrar mas boxes

Un buen sitio para descargar boxes es <https://app.vagrantup.com/boxes/search> el cual es un directorio publico de boxes disponibles libremente que corren diferentes plataformas.

7. Aprovisionamiento e instalación de servicios

Vamos a instalar un servidor http en la maquina guest.

Podríamos simplemente conectarnos a la maquina a través de SSH e instalar el servidor http, pero en este caso cada persona que usa Vagrant tendría que hacer la misma cosa. En lugar de esto Vagrant ofrece soporte para aprovisionamiento automático. Usando esta característica, Vagrant automáticamente instala software al ejecutar el comando **vagrant up**, de tal manera que la maquina guest pueda ser creada y puesta a disposición del usuario varias veces con la misma configuración.

Instalación de Apache

Vamos a configurar Apache usando un script.

Cree el siguiente script y guárdelo como [bootstrap.sh](#) en el mismo directorio de su Vagrantfile:

```
#!/usr/bin/env bash

apt-get update

apt-get install -y apache2
```

Ahora vamos a configurar Vagrant para correr este script cuando la Maquina virtual arranque. Para esto necesitaremos editar el Vagrantfile, para que quede de la siguiente forma:

```
Vagrant.configure("2") do |config|

  config.vm.box = "hashicorp/precise64"

  config.vm.provision :shell, path: "bootstrap.sh"
```

```
end
```

La línea de aprovisionamiento le dice a Vagrant que debe usar archivo `bootstrap.sh` para aprovisionar la maquina. El path del archivo es relativo a la localización del directorio root del proyecto.

8. Provisión

Después de que todo este configurado, solo se debe correr **vagrant up** para crear la maquina y Vagrant realizara el aprovisionamiento automático.

Si la maquina ya esta corriendo puede correr **vagrant reload --provision** para rápidamente reiniciar su maquina saltando el proceso de carga inicial.

La opción `--provision` en el comando de recarga (`vagrant reload --provision`) le indica a Vagrant que debe correr los provisioners dado que Vagrant solo hará esto la primera vez que se ejecute **vagrant up**.

Después de que Vagrant inicie, el servidor web estará activo y corriendo.

9. Networking

Port Forwarding

Port Forwarding permite especificar los puertos en la maquina guest para compartir a través de un puerto en la maquina host. Esto le permitirá acceder un puerto en su propia maquina, pero reenviando todo el trafico de red a un puerto especifico en la maquina guest.

Vamos a configurar un puerto de reenvío de tal manera que podamos acceder el servidor web apache en nuestro guest.

Edite el Vagrantfile para que contenga lo siguiente:

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/precise64"
  config.vm.provision :shell, path: "bootstrap.sh"
  config.vm.network :forwarded_port, guest: 80, host: 4567
end
```

Ejecute **vagrant reload** o **vagrant up** (dependiendo si la maquina esta corriendo) para que los cambios tengan efecto.

Una vez inicie la maquina de nuevo, cargue <http://127.0.0.1:4567> en su browser. Debería ver una pagina web servida desde la maquina virtual que fue automáticamente configurada por Vagrant.

Para realizar configuraciones adicionales de networking consulte la documentación en la siguiente pagina:

<https://www.vagrantup.com/docs/networking/>

10. Detener o suspender una maquina virtual

vagrant suspend guarda el estado actual de la maquina y la detiene. Para volver a la maquina desde el punto en que la suspendio puede ejecutar **vagrant up**.

Vagrant halt apaga la maquina virtual de manera segura conservando los contenidos del disco y permitiendo un inicio seguro de nuevo. Para levantar la maquina de nuevo puede usar **vagrant up**.

Vagrant destroy remueve la maquina guest del sistema compelamente. Para levantar la maquina de nuevo puede usar **vagrant up**.

11. Proveedores

En esta guia se uso VirtualBox para correr las maquinas guests. Sin embargo Vagrant funciona con diferentes proveedores tales como Vmware, AWS y otros. Vagrant permite especificar el provider a usar:

```
$ vagrant up --provider=vmware_fusion
```

Si desea montar su ambiente en la nube puede usar:

```
$ vagrant up --provider=aws
```

Las siguientes veces que ejecute **vagrant up** Vagrant usara el ultimo provider especificado.

2. Parte 2: Publicar Boxes en Vagrant Cloud

Una vez modificada la maquina virtual podemos crear un nuevo box y subirlo a Vagrant Cloud para usarlo posteriormente. Para eso seguiremos los siguientes pasos:

1. Re empaquetar la maquina virtual en un nuevo Vagrant Box

```
vagrant package servidorUbuntu --output mynew.box
```

2. Agregar el box creado a su instalación de Vagrant.

El comando anterior creara un archivo mynew.box. Con el siguiente comando agregaremos el box a nuestra instalación de Vagrant:

```
vagrant box add mynewbox mynew.box
```

Esto permitirá usar el box desde cualquier ubicación en su computador.

3. Publique el box en Vagrant Cloud
 - Diríjase a <https://app.vagrantup.com/boxes/new>
 - Ingrese un nombre y una descripción para su box
 - Cree la primera versión del box. Esta versión debe cumplir con el formato [0-9].[0-9].[0-9]. Por ejemplo 0.0.1.
 - Cree un provider para el box. Virtualbox es el provider mas común.
 - Cargue el archivo .box que corresponde al provider creado
 - Una vez cargado el box puede encontrarlo en la sección de boxes de <https://app.vagrantup.com/>
 - Antes de usar una versión del box, debe liberarlo “release”
 - Una vez creado y liberado un box, puede liberar nuevas versiones dando click en “create new version” en el menú de versiones de la pagina del box.

Referencias

Creating a New Vagrant Box. <https://atlas.hashicorp.com/help/vagrant/boxes/create>

Ejercicios:

PARTE A

1. Crear el ambiente de trabajo inicial con una maquina cliente y una maquina servidor, de acuerdo al Vagrantfile mostrado en la seccion 4.
2. Instalar net-tools y vim
3. Subir la imagen modificada a Vagrant Cloud
4. Realizar el taller Linux disponible en Classroom

5. Investigue en qué consisten los *directorios sincronizados de Vagrant*. Demuestre su funcionamiento.
6. Instale Jupiter notebooks

```
vagrant@servidorUbuntu:~$sudo apt install python3-pip
vagrant@servidorUbuntu:~$pip3 install jupyter
vagrant@servidorUbuntu:~$export PATH="$HOME/.local/bin:$PATH"
```

Ejecute el servicio

```
vagrant@servidorUbuntu:~$ jupyter notebook --ip=0.0.0.0
[I 20:46:31.981 NotebookApp] Serving notebooks from local directory: /home/vagrant
[I 20:46:31.982 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 20:46:31.982 NotebookApp] http://servidorUbuntu:8888/?token=e1f0e1b5c7f8653f31b80fde8e1d67304ddabb4b0b5b2737
[I 20:46:31.982 NotebookApp] or
http://127.0.0.1:8888/?token=e1f0e1b5c7f8653f31b80fde8e1d67304ddabb4b0b5b2737
[I 20:46:31.982 NotebookApp] Use Control-C to stop this server and shut down all kernels
(twice to skip confirmation).
[W 20:46:31.986 NotebookApp] No web browser found: could not locate runnable browser.
[C 20:46:31.987 NotebookApp]
```

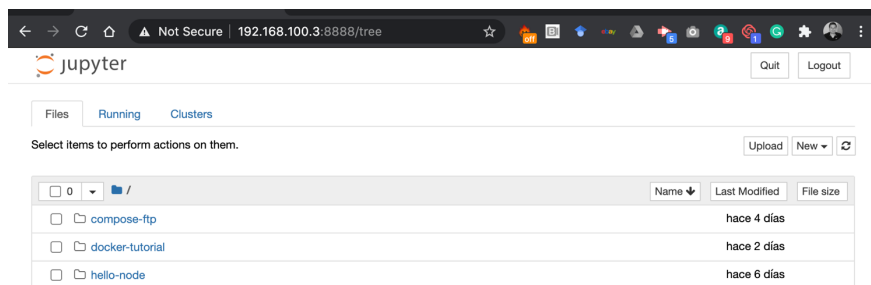
To access the notebook, open this file in a browser:

file:///home/vagrant/.local/share/jupyter/runtime/nbserver-3881-open.html

Or copy and paste one of these URLs:

http://servidorUbuntu:8888/?token=e1f0e1b5c7f8653f31b80fde8e1d67304ddabb4b0b5b2737
or http://127.0.0.1:8888/?token=e1f0e1b5c7f8653f31b80fde8e1d67304ddabb4b0b5b2737

Copie la ultima linea de la salida anterior y reemplace 127.0.0.1 por 192.168.100.3 y ejecute en el browser de su equipo anfitrión para acceder a Jupiter notebooks



PARTE B

1. Investigue cómo funcionan los repositorios git.
 - a. Revise el video de YouTube sobre los pasos básicos para empezar a usar GitHub:
<https://www.youtube.com/watch?v=SqbGliTKVoE&t=39s>

- b. Si no lo tiene, cree un repositorio git en github (<https://github.com/>) y utilícelo desde su máquina virtual "servidor" para subir y descargar archivos.
- c. En el repositorio, cree una estructura de directorios en la cual subirá los archivos relacionados con cada una de sus prácticas a lo largo del Semestre. Por Ejemplo:

```
.└─ compunube
   └─ Practical_Vagrant
      └─ Practica2_LXD
```

Nota: Recuerde primero instalar el cliente git en su máquina servidor.

PARTE C (RETOS – OPCIONAL: Valido por 0.5 puntos en una de las notas de tareas).

1. Busque una maquina virtual Vagrant con Tensorflow y compruebe su funcionamiento.

REFERENCIAS

Vagrant. <https://www.vagrantup.com/>

GitHub. <https://github.com/>

Tutorial GIT. <https://git-scm.com/docs/gittutorial>

Ejercicios Linux. <https://learnpythonthehardway.org/book/appendix-a-cli/ex1.html>