

DESARROLLO DE TALLERES

Nombre del Estudiante: **Maria Fernanda Díaz Hernandez, Pedro J. Vargas Barrios**

1. Taller 1

1.1. Objetivo

Realizar despliegues de infraestructura sencillos utilizando el lenguaje de orquestación de OpenStack.

1.2. Actividades

1. Crear un archivo denominado **“router.yaml”** con el siguiente contenido:

```
heat template version 2013-05-23
description: This template deploys a router with a port in the public interface
parameters:
  public network: type: string label: Public network name or ID description: Public network with floating IP
                  addresses. default: ext-net-doctorado
resources:
  router: type: OS::Neutron::Router properties: external gateway info: network: { get param: public network }
```

Figura 1: formulario para seleccionar plantilla

Lanzar pila

Nombre de la pila *

router-g2

Tiempo de espera de creación (minutos) *

60

☒ Restauración tras fallo

Contraseña para usuario "grupo2" *

.....

Public network name or ID

ext-net-doctorado

Descripción:

Crear una nueva pila con los valores proporcionados.

Cancelar

Lanzar

Figura 2: formulario para lanzar plantilla

<input type="checkbox"/>	Nombre de la pila	Creada	Actualizada	Estado	Actions
<input type="checkbox"/>	router-g2	0 minutos	Nunca	Creación Finalizada	Check Stack ▾

Displaying 1 item

Figura 3: Verificamos que la plantilla fue lanzada correctamente

2. Verificar la correcta creación del router:

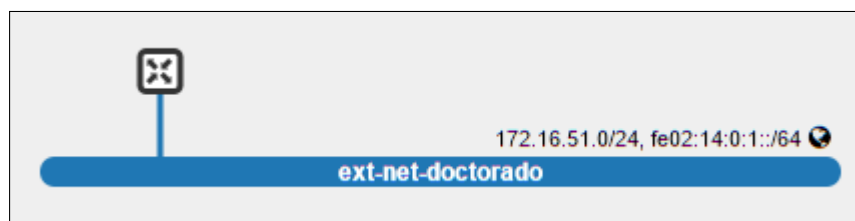


Figura 4: Se verifica creación del Router en la topología de red

3. Crear un archivo denominado “**network.yaml**” con el siguiente contenido:

```
heat template version: 2013-05-23

description: This template deploys a router with a port in the public interface

parameters:

private network cidr: type: string
label: Private network CIDR
description: Private Network CIDR
default: 192.168.200.0/24

resources:

private network:
type: OS::Neutron::Net

private subnet:
type: OS::Neutron::Subnet
properties:
networkid: { get resource: private network }
cidr: {get param: private network cidr}
dns nameservers:
- 8.8.8.8
```

Seleccionar plantilla

Origen de la plantilla *

Fichero

Fichero de plantilla ?

Seleccionar archivo network.yaml

Fuente de Entorno

Fichero

Archivo de Entorno ?

Seleccionar archivo No se eligió archivo

Descripción:

Utilice una las opciones de recurso de plantilla disponible para especificar la plantilla que se utilizará al crear esta pila.

Cancelar Siguiente

Figura 5: formulario para seleccionar plantilla

Lanzar pila ✕

Nombre de la pila * ⓘ

Tiempo de espera de creación (minutos) * ⓘ

☒ Restauración tras fallo ⓘ

Contraseña para usuario "grupo2" * ⓘ
 ⓘ

Public network name or ID ⓘ

Descripción:
Crear una nueva pila con los valores proporcionados.

Figura 6: formulario para lanzar plantilla

<input type="checkbox"/>	Nombre de la pila	Creada	Actualizada	Estado
<input type="checkbox"/>	red-g2	0 minutos	Nunca	Creación Finalizada
<input type="checkbox"/>	router-g2	7 minutos	Nunca	Creación Finalizada
Displaying 2 items				

Figura 7: Verificamos que la plantilla fue lanzada correctamente

4. Verificar la correcta creación de la red:

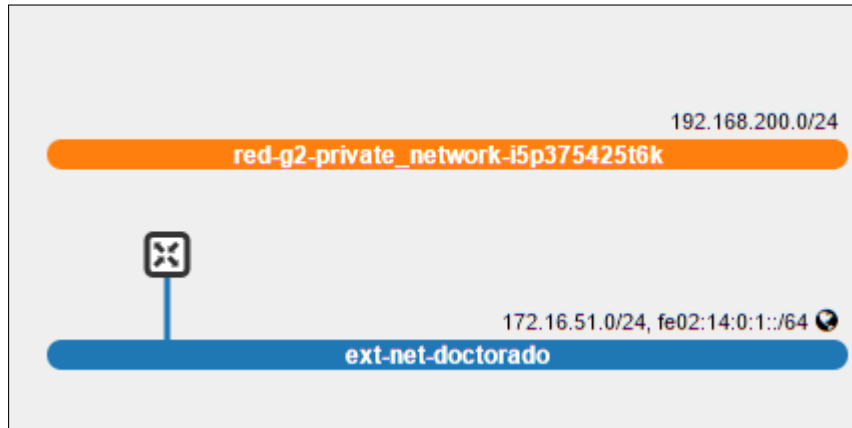


Figura 8: Se verifica creación del Router en la topología de red

5. Eliminar las pilas previamente creadas:
6. Crear un archivo denominar “complete-network.yaml” con el siguiente contenido y ejecutar la plantilla. En este paso, se va a crear un router, una red privada, y se le va a asignar un puerto al router dentro de esa red.

```
heat_template_version: 2013-05-23

description: This template deploys a router with a port in the public interface

parameters:

  public_network:
    type: string
    label: Public network name or ID
    description: Public network with floating IP addresses.
    default: ext-net-doctorado

  private_network_cidr:
    type: string
    label: Private network CIDR
    description: Private Network CIDR
    default: 192.168.200.0/24

  resources:

    router:
      type: OS::Neutron::Router
      properties:
        external_gateway_info:
          network: get_param: public_network

    private_network:
      type: OS::Neutron::Net

    private_subnet:
      type: OS::Neutron::Subnet
      properties:
```

```

network_id: get_resource: private_network
cidr: get_param: private_network_cidr
dns_nameservers:
- 8.8.8.8

router-interface:
type: OS::Neutron::RouterInterface
properties:
router_id: get_resource: router
subnet: get_resource: private_subnet

```

Lanzar pila

Nombre de la pila *

complete-network-g2

Descripción:
Crear una nueva pila con los valores proporcionados.

Tiempo de espera de creación (minutos) *

60

☐ Restauración tras fallo

Contraseña para usuario "grupo2" *

.....

Private network CIDR

192.168.200.0/24

Public network name or ID

ext-net-doctorado

Figura 9: formulario para lanzar plantilla

<input type="checkbox"/>	Nombre de la pila	Creada	Actualizada	Estado
<input type="checkbox"/>	complete-network-g2	0 minutos	Nunca	Creación Finalizada
Displaying 1 item				

Figura 10: Verificamos que la plantilla fue lanzada correctamente

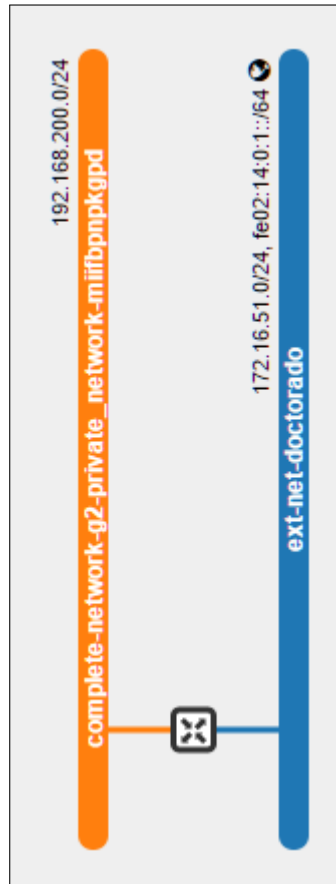


Figura 11: Se verifica creación de la topología de red

Una vez la infraestructura de red, el siguiente paso es crear servidores. Inicialmente, se desplegará únicamente el servidor con su respectivo grupo de seguridad. Posteriormente se le configurará en la plantilla el software a instalar y se le asignará un IP flotante.

7. Eliminar las pilas previamente creadas.
8. Crear un archivo denominar **“network-server.yaml”** con el siguiente contenido y ejecutar la plantilla. En este paso, se va a crear un router, una red privada, se le va a asignar un puerto al router dentro de esa red, y se va a lanzar una instancia con un grupo de seguridad denominado “web_server_security_group” y una llave “cloudapps” .

```
heat_template_version: 2013-05-23
```

```
description: This template deploys a router, a private network and a single basic server with a security group.
```

```
parameters:
```

```
public_network:
```

```
type: string
```

```
label: Public network name or ID
```

```
description: Public network with floating IP addresses.
```

```
default: ext-net-doctorado
```

```
private_network_cidr:
```

```
type: string
```

```
label: Private network CIDR
description: Private Network CIDR
default: 192.168.200.0/24

image:
type: string
label: Image name or ID
description: Image to be used for compute instance
default: Ubuntu-Server-14.04-CECAD-r20141201

flavor:
type: string
label: Flavor
description: Type of instance (flavor) to be used
default: m1.small

resources:

router:
type: OS::Neutron::Router
properties:
external_gateway_info:
network: get_param: public_network

private_network:
type: OS::Neutron::Net

private_subnet:
type: OS::Neutron::Subnet
properties:
network_id: get_resource: private_network
cidr: get_param: private_network_cidr
dns_nameservers:
- 8.8.8.8

router-interface:
type: OS::Neutron::RouterInterface
properties:
router_id: get_resource: router
subnet: get_resource: private_subnet

web_server_security_group:
type: OS::Neutron::SecurityGroup
properties:
name: web_server_security_group
rules:
- protocol: tcp
port_range_min: 80
port_range_max: 80
- protocol: tcp
port_range_min: 443
port_range_max: 443
- protocol: icmp
- protocol: tcp
port_range_min: 22
port_range_max: 22
```



```

my_keypair:
  type: OS::Nova::KeyPair
  properties:
    name: cloudapps
    save_private_key: True

my_instance:
  type: OS::Nova::Server
  properties:
    image: get_param: image
    flavor: get_param: flavor
    key_name: get_resource: my_keypair
    networks:
      - network: get_resource: private_network
    security_groups:
      - get_resource: web_server_security_group
    user_data: —
    #!/bin/sh
    sudo apt-get -y update && sudo apt-get -y install apache2 && sudo service apache2 restart
    user_data_format: RAW

outputs:
  my_instance_name:
    description: Name of the instance
    value: get_attr: [my_instance, name]
  my_instance_ip:
    description: IP address of the instance
    value: get_attr: [my_instance, first_address]

```

	network-server-g2	27 minutos	Nunca	Creación Finalizada
---	-------------------	------------	-------	---------------------

Figura 12: Verificamos que la plantilla fue lanzada correctamente



Figura 13: Verificamos que la instancia fue lanzada correctamente

- Finalmente, se le va a asignar una IP flotante a la instancia. Modificar el archivo “network-server.yaml” en la sección “resources” para que luzca de la siguiente forma:

heat_template_version: 2013-05-23

description: This template deploys a router, a private network and a single basic server with a security group.

parameters:

public_network:

type: string

label: Public network name or ID

description: Public network with floating IP addresses.

default: ext-net-doctorado

private_network_cidr:

type: string

label: Private network CIDR

description: Private Network CIDR

default: 192.168.200.0/24

image:

type: string

label: Image name or ID

description: Image to be used for compute instance

default: Ubuntu-Server-14.04-CECAD-r20141201

flavor:

type: string

label: Flavor

description: Type of instance (flavor) to be used

default: m1.small

resources:

router:

type: OS::Neutron::Router

properties:

external_gateway_info:

network: get_param: public_network

private_network:

type: OS::Neutron::Net

private_subnet:

type: OS::Neutron::Subnet

properties:

network_id: get_resource: private_network

cidr: get_param: private_network_cidr

dns_nameservers:

- 8.8.8.8

router-interface:

type: OS::Neutron::RouterInterface

properties:

router_id: get_resource: router

subnet: get_resource: private_subnet

web_server_security_group:

type: OS::Neutron::SecurityGroup

```

properties:
name: web_server_security_group
rules:
- protocol: tcp
port_range_min: 80
port_range_max: 80
- protocol: tcp
port_range_min: 443
port_range_max: 443
- protocol: icmp
- protocol: tcp
port_range_min: 22
port_range_max: 22

my_keypair:
type: OS::Nova::KeyPair
properties:
name: cloudapps
save_private_key: True

floating_ip:
type: OS::Neutron::FloatingIP
properties:
floating_network: get_param: public_network

server_port:
type: OS::Neutron::Port
properties:
network: get_resource: private_network
security_groups:
- get_resource: web_server_security_group

my_instance:
type: OS::Nova::Server
properties:
image: get_param: image
flavor: get_param: flavor
key_name: get_resource: my_keypair
availability_zone: hpc-test
networks:
- port: get_resource: server_port
user_data: —
#!/bin/sh
sudo apt-get -y update && sudo apt-get -y install apache2 && sudo service apache2 restart
user_data_format: RAW

floating_ip_assoc:
type: OS::Neutron::FloatingIPAssociation
properties:
floatingip_id: get_resource: floating_ip
port_id: get_resource: server_port

outputs:
my_instance_name:
description: Name of the instance
value: get_attr: [my_instance, name]
my_instance_ip:
description: IP address of the instance
value: get_attr: [my_instance, first_address]

```

```
my_instance.floating_ip:  
description: The IP address of the deployed instance  
value: get_attr: [floating_ip, floating_ip_address]
```

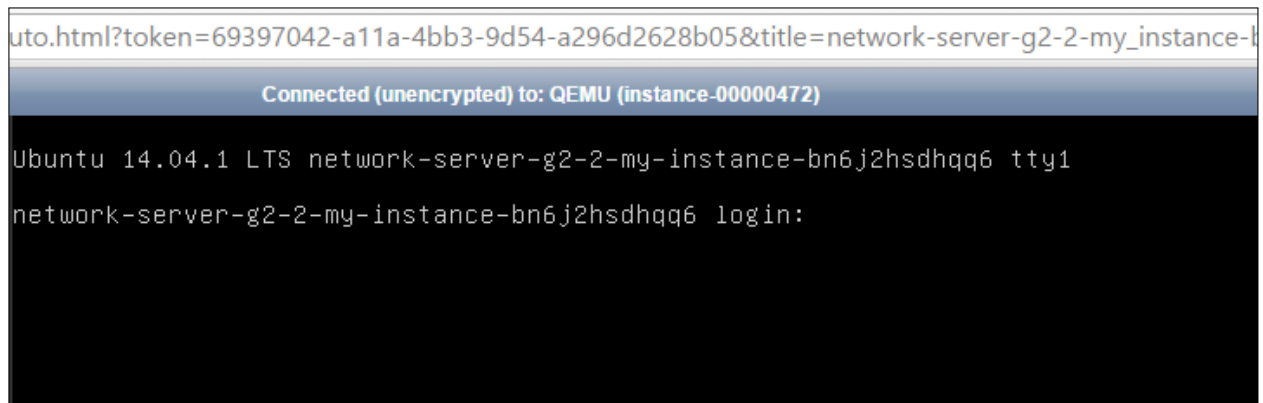


Figura 14: Verificamos que la instancia fue lanzada correctamente

10. Esperar un breve tiempo y acceder a la página pre-establecida de Apache direccionando cualquier explorador a la dirección `http://ip-flotante-instancia/`.

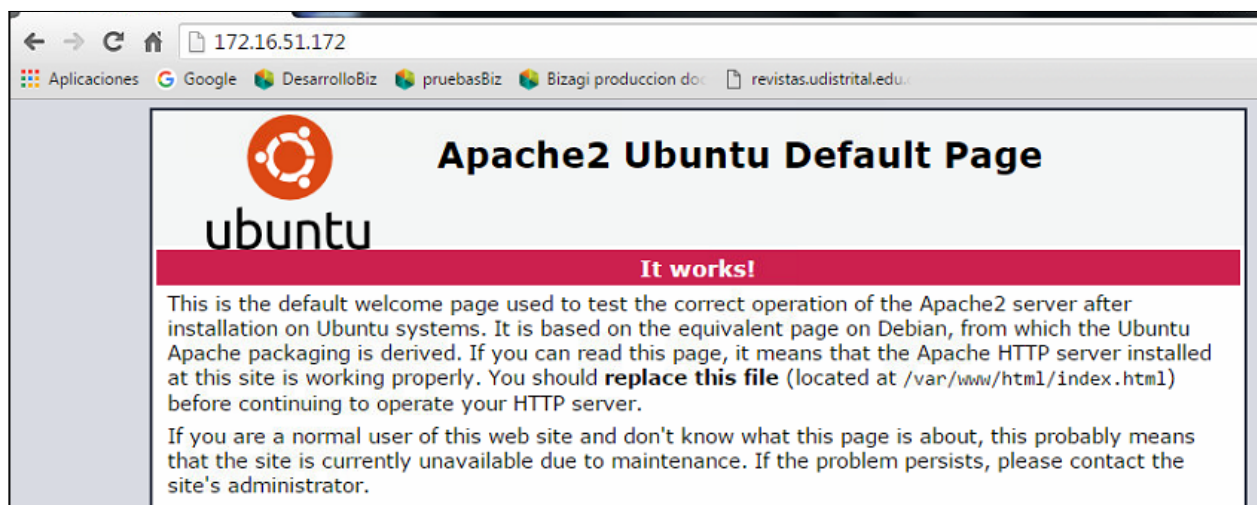


Figura 15: Verificamos que podemos acceder al servidor apache con la ip flotante

2. Taller 3

2.1. Objetivo

Realizar el despliegue de entornos de desarrollo sencillos mediante la tecnología Vagrant utilizando como proveedor de infraestructura la tecnología VirtualBox.

2.2. Actividades

Nota: Debe estar instalado virtualbox 4.3, 4.2 o 4.1 en el equipo para el desarrollo de la actividad.

Agregar repositorio para linux Mint 17 con el comando: `sudo sh -c 'echo "deb http://download.virtualbox.org/virtualbox/debian trusty contrib" > /etc/apt/sources.list'`

Agregar llaves con el comando: `wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add --`

```
pedrojvar@pedrojvar-VirtualBox ~ $ wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add --
```

Figura 16: Agregar llave

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo apt-get install virtualbox-4.3
```

Figura 17: Instalar Virtualbox 4.3

1. Abrir una consola de comandos.
2. Validar la correcta instalación del software Vagrant ejecutando el comando `vagrant -h`
 - Verificar instalación

```
pedrojvar@pedrojvar-VirtualBox ~ $ vagrant -h
El programa «vagrant» no está instalado. Puede instalarlo escribiendo:
sudo apt-get install vagrant
```

Figura 18: Programa Vagrant NO instalado

- En caso de no estar instalado, ejecutar el siguiente comando para su instalación

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo apt-get install vagrant
[sudo] password for pedrojvar:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  bsdtar curl ruby-childprocess ruby-erubis ruby-ffi ruby-i18n ruby-log4r
  ruby-net-scp ruby-net-ssh
Paquetes sugeridos:
  bsdcpio virtualbox
Se instalarán los siguientes paquetes NUEVOS:
  bsdtar curl ruby-childprocess ruby-erubis ruby-ffi ruby-i18n ruby-log4r
  ruby-net-scp ruby-net-ssh vagrant
0 actualizados, 10 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 840 kB de archivos.
Se utilizarán 5.342 kB de espacio de disco adicional después de esta operación.
```

Figura 19: Comando de instalación de Vagrant

- En caso de no estar instalado, ejecutar el siguiente comando para su instalación

```

pedrojvar@pedrojvar-VirtualBox ~ $ vagrant -h
Usage: vagrant [-v] [-h] command [<args>]

-v, --version      Print the version and exit.
-h, --help         Print this help.

```

Figura 20: Se verifica instalación de Vagrant

3. Adicionar la imagen del sistema operativo Ubuntu Trusty de 64 bits. Para ello, ejecutar el comando `vagrant box add ubuntu/trusty64`. La descarga debe tomar alrededor de 5 minutos.

- En caso de no poder descargar la imagen con ese comando, podemos buscar imagenes de Vagrant en el siguiente link: <https://cloud-images.ubuntu.com/vagrant/trusty/20160423/>
- La imagen se descargaría con el siguiente comando: `vagrant box add trusty-server-i386 https://cloud-images.ubuntu.com/vagrant/trusty/20160422/trusty-server-cloudimg-i386-vagrant-disk1.box`

```

pedrojvar@pedrojvar-VirtualBox ~/vagrant001 $ vagrant box add trusty-server-i386
https://cloud-images.ubuntu.com/vagrant/trusty/20160422/trusty-server-cloudimg-
i386-vagrant-disk1.box
Downloading box from URL: https://cloud-images.ubuntu.com/vagrant/trusty/2016042
2/trusty-server-cloudimg-i386-vagrant-disk1.box
Extracting box...te: 1546k/s, Estimated time remaining: 0:00:01)
Successfully added box 'trusty-server-i386' with provider 'virtualbox'!

```

Figura 21: Comando de instalación de Vagrant, la imagen queda con nombre trusty-server-i386

4. Crear un directorio de trabajo clouapps (o cualquier otro nombre). Ingresar a ese directorio en la consola y ejecutar el comando `vagrant init ubuntu/trusty64`. Después de ejecutar el comando, verificar que se haya creado un archivo denominado Vagrantfile.

- para este caso se crea la carpeta `vagrant001`
- Dentro de la carpeta creada se ejecuta el comando: `vagrant init trusty-server-i386`, con esto se crea un archivo llamado Vagrantfile.

5. Editar el archivo Vagrantfile de forma que contenga la siguiente información

```

# -*- mode: ruby -*- # vi: set ft=ruby :
# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do --config--
  # All Vagrant configuration is done here. The most common configuration
  # options are documented and commented below. For a complete reference,
  # please see the online documentation at vagrantup.com.

  # Every Vagrant virtual environment requires a box to build off of.
  config.vm.box = "trusty-server-i386"
  config.vm.network :forwarded_port, guest: 4000, host: 8100, host_ip: "127.0.0.1"
  config.vm.provision "shell", path: "script.sh"

end

```

```
config.vm.box = "trusty-server-i386"
config.vm.network :forwarded_port, guest: 4000, host: 8100, host_ip: "127.0.0.1"
config.vm.provision "shell", path: "script.sh"
```

Figura 22: Configuración Vagrantfile

6. Crear un archivo “script.sh” en el mismo directorio del archivo Vagrantfile.
7. Escribir el siguiente contenido en el archivo “script.sh”.

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

echo "Installing: nodejs, lynx, ruby and jekyll..."

apt-add-repository ppa:brightbox/ruby-ng && /tmp/provision-script.log 2&&1
apt-get -y update && /tmp/provision-script.log 2&&1
apt-get install -y nodejs && /tmp/provision-script.log 2&&1
apt-get install -y lynx-cur && /tmp/provision-script.log 2&&1
apt-get install -y ruby2.2 && /tmp/provision-script.log 2&&1
apt-get install -y ruby2.2-dev && /tmp/provision-script.log 2&&1
gem install jekyll && /tmp/provision-script.log 2&&1
cd /vagrant
jekyll serve -H 0.0.0.0 -detach
```

```
#!/usr/bin/env bash
echo "Installing: nodejs, lynx, ruby and jekyll..."
apt-add-repository ppa:brightbox/ruby-ng && /tmp/provision-script.log 2>&1
apt-get -y update && /tmp/provision-script.log 2>&1
apt-get install -y nodejs && /tmp/provision-script.log 2>&1
apt-get install -y lynx-cur && /tmp/provision-script.log 2>&1
apt-get install -y ruby2.2 && /tmp/provision-script.log 2>&1
apt-get install -y ruby2.2-dev && /tmp/provision-script.log 2>&1
gem install jekyll && /tmp/provision-script.log 2>&1
cd /vagrant
jekyll serve -H 0.0.0.0 -detach
```

Figura 23: Contenido del Script

8. Ejecutar el comando `vagrant up --provision`.

```
pedrojvar@pedrojvar-VirtualBox ~/vagrant001 $ vagrant up --provision
Bringing machine 'default' up with 'virtualbox' provider...
[default] Importing base box 'trusty-server-i386'...
[default] Matching MAC address for NAT networking...
[default] Clearing any previously set forwarded ports...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] -- 4000 => 8100 (adapter 1)
[default] Booting VM...
[default] Waiting for machine to boot. This may take a few minutes...
[default] Machine booted and ready!
[default] Mounting shared folders...
[default] -- /vagrant
[default] Running provisioner: shell...
[default] Running: /tmp/vagrant-shell20160426-14082-nriyk5
Installing: nodejs, lynx, ruby and jekyll...
```

Figura 24: Contenido del Script

9. Verificar el correcto funcionamiento del despliegue accediendo en un navegador (browser) a la dirección `http://127.0.0.1:8100`.

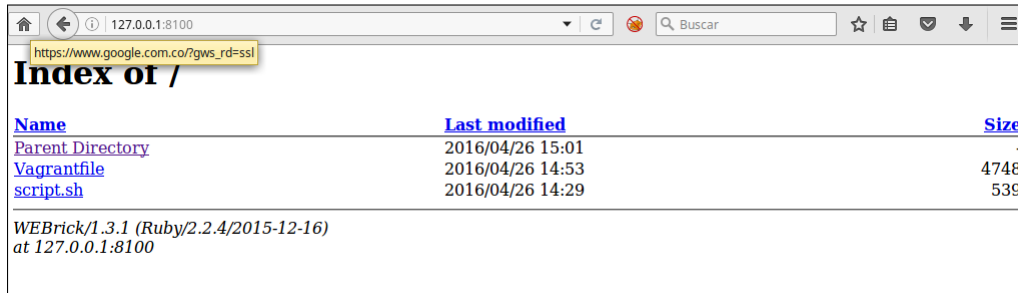


Figura 25: Verificar despliegue del servicio Vagrant en el navegador

3. Taller 4

3.1. Objetivo

Realizar el despliegue de entornos de desarrollo sencillos mediante la tecnología Vagrant utilizando como proveedor de infraestructura la tecnología VirtualBox.

3.2. Actividades

1. Abrir una consola de comandos.
2. Validar la correcta instalación del software Vagrant ejecutando el comando `vagrant -h`

```
pedrojvar@pedrojvar-VirtualBox ~ $ vagrant -h
Usage: vagrant [-v] [-h] command [<args>]

-v, --version      Print the version and exit.
-h, --help         Print this help.
```

Figura 26: Verificación de instalación de Vagrant

3. Adicionar la imagen del sistema operativo Ubuntu Trusty de 64 bits. Para ello, ejecutar el comando `vagrant box add ubuntu/trusty64`. La descarga debe tomar alrededor de 5 minutos.

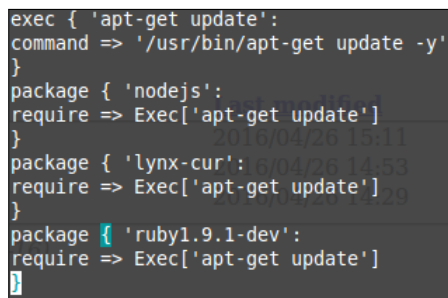
```
pedrojvar@pedrojvar-VirtualBox ~/vagrant001 $ vagrant box add trusty-server-i386
https://cloud-images.ubuntu.com/vagrant/trusty/20160422/trusty-server-cloudimg-
i386-vagrant-disk1.box
Downloading box from URL: https://cloud-images.ubuntu.com/vagrant/trusty/2016042
2/trusty-server-cloudimg-i386-vagrant-disk1.box
Extracting box...te: 1546k/s, Estimated time remaining: 0:00:01)
Successfully added box 'trusty-server-i386' with provider 'virtualbox'!
```

Figura 27: Bajar imagen de Vagrant

4. Crear un directorio de trabajo clouapps (o cualquier otro nombre). Ingresar a ese directorio en la consola y ejecutar el comando `vagrant init ubuntu/trusty64`. Después de ejecutar el comando, verificar que se haya creado un archivo denominado `Vagrantfile`.
 - se crea la carpeta `Vagrant002`
 - Dentro de la carpeta creada se ejecuta el comando: `vagrant init trusty-server-i386`, con esto se crea un archivo llamado `Vagrantfile`.
5. Editar el archivo `Vagrantfile` de forma que contenga la siguiente información


```
# -*- mode: ruby -*- # vi: set ft=ruby :  
  
# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!  
VAGRANTFILE_API_VERSION = "2"  
  
Vagrant.configure(VAGRANTFILE_API_VERSION) do ---config---  
  # All Vagrant configuration is done here. The most common configuration  
  # options are documented and commented below. For a complete reference,  
  # please see the online documentation at vagrantup.com.  
  
  # Every Vagrant virtual environment requires a box to build off of.  
  config.vm.box = "trusty-server-i386"  
  config.vm.network :forwarded_port, guest: 4000, host: 8100, host_ip: "127.0.0.1"  
  config.vm.provision "puppet"  
  config.vm.hostname = "www.cecad-example.com"  
end
```

6. Crear un archivo "default.pp" en un directorio manifests que se encuentra en el mismo lugar del archivo Vagrantfile.
7. Escribir el siguiente contenido en el archivo "default.pp".



```
exec { 'apt-get update':  
  command => '/usr/bin/apt-get update -y'  
}  
package { 'nodejs':  
  require => Exec['apt-get update']  
}  
package { 'lynx-cur':  
  require => Exec['apt-get update']  
}  
package { 'ruby1.9.1-dev':  
  require => Exec['apt-get update']  
}
```

Figura 28: Contenido del archivo default.pp

8. Ejecutar el comando `vagrant up --provision`.

4. Taller 6

4.1. Objetivo

Realizar el despliegue de aplicaciones sencillas mediante la tecnología Docker sobre el sistema operativo Linux Ubuntu Server.

4.2. Actividades

1. Verificar la correcta instalación del servicio Docker ejecutando el comando:

sudo service docker status

Instalación de Docker en Linux Mint

```
# Agregar llave- Add the new gpg key
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D
```

```
# Agregar repositorio - Add /etc/apt/sources.list.d/docker.list
sudo vim /etc/apt/sources.list.d/docker.list
```

```
# Eliminar versiones inestables sudo apt-get update
sudo apt-get purge lxc-docker
```

```
# instalación - sudo apt-get install linux-image-extra-$(uname -r)
sudo apt-get install docker-engine
```

```
# Bloquear acceso como root - Give non-root access
sudo groupadd docker
sudo gpasswd -a $USER docker
sudo service docker restart
```

First run:

```
sudo service docker start
sudo docker run hello-world
```

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo service docker status
docker start/running, process 7926
```

Figura 29: Verificar que el servicio está en ejecución

2. Descargar la imagen oficial de Docker para el software Apache Solr (Motor de búsqueda de código abierto)

sudo docker pull solr

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker pull solr
Using default tag: latest
latest: Pulling from library/solr

efd26ecc9548: Downloading 6.291 MB/51.34 MB
a3ed95caeb02: Download complete
d1784d73276e: Downloading 8.437 MB/18.53 MB
52a884c93bb2: Downloading 32.27 kB/566.6 kB
070ee56a6f7e: Waiting
```

Figura 30: Descargando imagen de Solr

3. Listar las imágenes de Docker disponibles. Debe aparecer la imagen de solr descargada
`sudo docker images`

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
hello-world         latest             94df4f0ce8a4       29 hours ago
967 B
solr                 latest             c6d71a5f9e4f       3 weeks ago
568 MB
```

Figura 31: Listado de imagenes en Docker

4. Iniciar el servidor de Apache Solr ejecutando un contenedor de Docker.

```
sudo docker run -p 8983:8983 -d --name mysolr solr
```

Este comando merece una explicación:

- `docker run` es el comando para ejecutar un nuevo contenedor Docker. Este comando recibe varios parámetros.
- `-p` especifica un mapeo de puertos, `puerto-host:puerto-contenedor` en donde se le dice que un puerto determinado en el huésped redirecciona al puerto del contenedor, usualmente el puerto de un servicio determinado. En este caso, 8983 es el puerto del servicio Solr.
- `-d` especifica que el contenedor se va a ejecutar en background.
- `--name` especifica un nombre para el contenedor. En el comando anterior, el contenedor se llama `mysolr`
- Cuando se lanza el contenedor, debe especificarse su imagen base; en este caso, `solr` es el nombre de la imagen.

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker run -p 8983:8983 -d --name mysolr solr
e670866eca19de4b9414a68cd3df315006fc123dc53c9868dc7147482f9e151f
```

Figura 32: Iniciar Servidor Apache Solr

5. Verificar que el contenedor se esté ejecutando. Para ello, ejecutar el comando
`sudo docker ps`

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
e670866eca19        solr                "/opt/solr/bin/solr -"   3 minutes ago       Up 3 minutes       0.0.0.0:8983->8983/tcp   mysolr
```

Figura 33: Verificar que el contenedor se ejecuta

6. En el anterior comando, se listan varias características del contenedor, incluido su identificador. Con este identificador, es posible acceder a los logs del contenedor, si es necesario verificar en detalle las acciones sobre el mismo.

```
sudo docker logs -f id-contenedor
```

```

pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker logs -f e670866eca19
Starting Solr on port 8983 from /opt/solr/server
0 INFO (main) [ ] o.e.j.u.log Logging initialized @940ms
611 INFO (main) [ ] o.e.j.s.Server jetty-9.2.13.v20150730
641 WARN (main) [ ] o.e.j.s.h.RequestLogHandler !RequestLog
644 INFO (main) [ ] o.e.j.d.p.ScanningAppProvider Deployment monitor [file:/opt/solr/server/contexts/] at interval 0
1534 INFO (main) [ ] o.e.j.w.StandardDescriptorProcessor NO JSP Support for /solr, did not find org.apache.jasper.serv
1554 WARN (main) [ ] o.e.j.s.SecurityHandler ServletContext@0.e.j.w.WebAppContext@57ffcd7{/solr,file:/opt/solr/server
ver/solr-webapp/webapp} has uncovered http methods for path: /
1570 INFO (main) [ ] o.a.s.s.SolrDispatchFilter SolrDispatchFilter.init(): WebAppClassLoader=1926764753@72d818d1

```

Figura 34: Verificando logs de contenedor con el ID

- Acceder a la consola de administración del servidor Apache Solr. Para ello, desde un navegador ingresar a la URL <http://direccion-huesped-docker:89983>

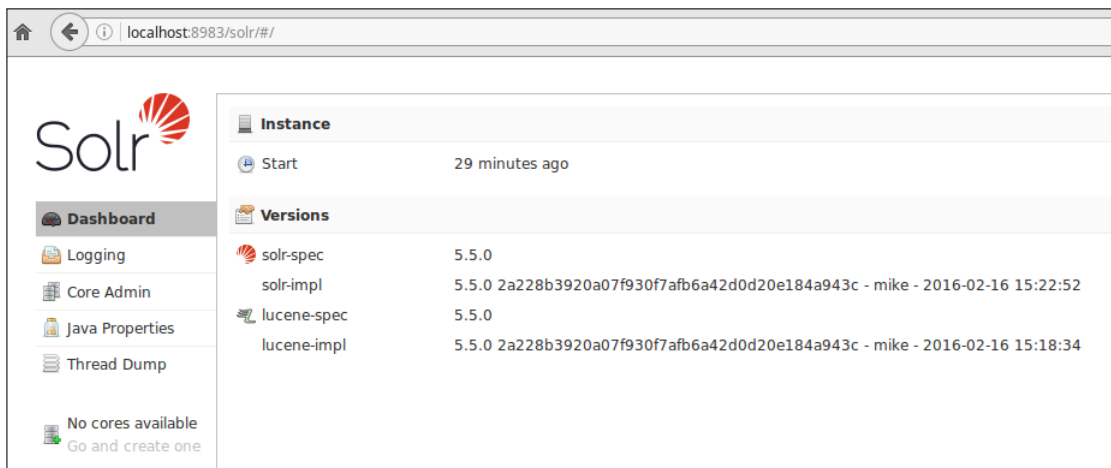


Figura 35: Verificando logs de contenedor con el ID

- En este momento, la aplicación ya está desplegada en el contenedor, disponible para utilización. Por ejemplo, se desea utilizar el servidor Solr recién desplegado para crear un índice núcleo. Lo primero es acceder a la consola del contenedor, ya que este se encuentra ejecutándose como un proceso en background.

```
sudo docker exec -it --user=solr mysolr bash
```

```

pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker exec -it --user=solr mysolr bash
solr@e670866eca19:/opt/solr$ _

```

Figura 36: Verificando logs de contenedor con el ID

- Ejecutar el comando
`bin/solr create_core -c gettingstarted`

```

solr@e670866eca19:/opt/solr$ bin/solr create_core -c gettingstarted
Copying configuration to new core instance directory:
/opt/solr/server/solr/gettingstarted
Creating new core 'gettingstarted' using command:
http://localhost:8983/solr/admin/cores?action=CREATE&name=gettingstarted&instanceDir=gettingstarted
{
  "responseHeader":{
    "status":0,
    "QTime":4658},
  "core":"gettingstarted"}

```

Figura 37: Comando desde la consola de Solr

10. Una de las funcionalidades más interesantes de Docker es poder copiar directamente un archivo creado en la máquina huésped a cualquier directorio dentro del contenedor. Para ello, crear en la máquina huésped (no en el contenedor) un archivo `solr.xml` con el siguiente contenido a manera de ejemplo:

```
<add>
<doc>
<field name="id">SOLR1000</field>
<field name="name">Solr, the Enterprise Search Server</field>
<field name="name">Apache Software Foundation</field>
<field name="cat">software</field>
<field name="cat">search</field>
<field name="features">Advanced Full-Text Search Capabilities using Lucene</field>
<field name="features">Optimized for High Volume Web Traffic</field>
<field name="features">Standards Based Open Interfaces - XML and HTTP</field>
<field name="features">Comprehensive HTML Administration Interfaces</field>
<field name="features">Scalability - Efficient Replication to other Solr Search Servers</field>
<field name="features">Flexible and Adaptable with XML configuration and Schema</field>
<field name="features">Good unicode support: h#xE9 llo (hello with an accent over the e)</field>
<field name="price">0</field>
<field name="popularity">10</field>
<field name="inStock">true</field>
<field name="incubationdate_dt">2006-01-17T00:00:00.000Z</field>
</doc>
</add>
```

Figura 38: Comando desde la consola de Solr

Acto seguido, copiar el archivo al directorio `/opt/solr` en el contenedor. Reemplazar `id-contenedor` con el respectivo valor.

```
sudo docker cp solr.xml id-contenedor:/opt/solr/solr.xml
```

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker cp solr.xml e670866eca19:/opt/solr/solr.xml
pedrojvar@pedrojvar-VirtualBox ~ $
```

Figura 39: Copiar archivo al contenedor

11. Ingresar nuevamente a la consola. Verificar la existencia del archivo `solr.xml` y ejecutar el comando `bin/post -c gettingstarted ./solr.xml`

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker exec -it --user=solr mysolr bash
solr@e670866eca19:/opt/solr$
```

Figura 40: Ingresar a la consola de Solr

```
solr@e670866eca19:/opt/solr$ bin/post -c gettingstarted ./solr.xml
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -classpath /opt/solr/dist/solr-core-5.5.0.jar -Dauto=yes -Dc=gettingstarted
stTool ./solr.xml
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/gettingstarted/update...
Entering auto mode. File endings considered are xml,json,jsonl,txt,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,r
POSTing file solr.xml (application/xml) to [base]
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/solr/gettingstarted/update...
```

Figura 41: Ejecutar el comando para trabajar con el archivo

12. El anterior comando debe permitir indexar archivos en el servidor Solr. (El tutorial no trata directamente de Solr sino de Docker, luego no es necesario dominar completamente la consola de administración de Solr). Acceder a la consola de administración del Solr y verificar la indexación.
13. Detener el contenedor.
- ```
sudo docker stop mysolr
```

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker stop mysolr
mysolr
```

Figura 42: Detener MySolr

- Verificar que el contenedor aparezca como “Exited” en su estado después de ejecutar el comando `sudo docker ps -a`

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND                | CREATED        | STATUS                     |
|--------------|-------|------------------------|----------------|----------------------------|
| e67866eca19  | solr  | "/opt/solr/bin/solr -" | 46 minutes ago | Exited (143) 2 minutes ago |

Figura 43: Detener MySolr

14. Eliminar el contenedor.  
`sudo docker rm mysolr`

```
pedrojvar@pedrojvar-VirtualBox ~ $ sudo docker rm mysolr
mysolr
```

Figura 44: Eliminar contenedor

15. En caso de requerirse, es posible eliminar la imagen utilizando el comando `sudo docker rmi solr`

## 5. Taller 7

### 5.1. Objetivo

Realizar el despliegue de aplicaciones sencillas utilizando la tecnología Docker sobre el sistema operativo Linux Ubuntu Server.

### 5.2. Actividades

1. Verificar la correcta instalación del servicio Docker ejecutando el comando `sudo service docker status`
2. En este taller se va a utilizar un archivo denominado Dockerfile (similar al Vagrantfile) que establece el conjunto de pasos para desplegar una imagen Docker. Crear un directorio, entrar a ese directorio y crear un archivo llamado “Dockerfile”.  
`mkdir Dockerfile`
3. Ingresar el siguiente contenido en el archivo:

```
FROM ubuntu:trusty RUN sudo apt-get update && sudo apt-get -y install cowsay fortune
```

```
FROM ubuntu:trusty
RUN sudo apt-get update && sudo apt-get -y install cowsay fortune
```

Figura 45: Crear e ingresar contenido en Dockerfile

4. Construir una nueva imagen a partir del Dockerfile.

*sudo docker build -t test/dockerfile-example .*

```
pedrojvar@pedrojvar-VirtualBox ~/dockerTest $ sudo docker build -t test/dockerfile-example .
Sending build context to Docker daemon 2.048 kB
Step 1 : FROM ubuntu:trusty
trusty: Pulling from library/ubuntu
943c334059c7: Pull complete
a1acf99303d2: Pull complete
27616aacb7b3: Pull complete
35d12cd1c9fc: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:34e732efa056124a9480c5abce11f9a02fc5e411c6617d2cdb6509958e0cd55f
Status: Downloaded newer image for ubuntu:trusty
--> 8fa7f61732d6
Step 2 : RUN sudo apt-get update && sudo apt-get -y install cowsay fortune
```

Figura 46: Crear e ingresar contenido en Dockerfile

5. Ejecutar un nuevo contenedor a partir de la imagen creada.

*sudo docker run test/dockerfile-example /usr/games/cowsay "Hola a todos!"*

```
pedrojvar@pedrojvar-VirtualBox ~/dockerTest $ sudo docker run test/dockerfile-example /usr/games/cowsay "Hola a todos!"
< Hola a todos! >
 ^__^
 (oo)_______
 (__)\)\/\
 ||----w |
 || ||
```

Figura 47: Ejecutar nuevo contenedor

## 6. Bibliografía

### Accesos WEB

- Orquestación Multi-instancia
- Instalación Vagrant Linux Mint
- Imágenes Vagrant
- Instalación Docker Linux Mint