

# INFORME DE DESARROLLO DE PRACTICAS

Nombre del Estudiante: **Gabriel Andres Alzate - Johan Nicolas Cuellar**

## 1 Introducción

El presente documento es la realización de los talleres propuestos en la asignatura Aplicaciones Sobre Internet/La Nube, impartida por el docente Ph.D. Nelson Perez y corresponden a la sección denominada IaaS (Infraestructura como servicio), donde se desarrollan proyectos de orquestación sobre OpenStack, donde se levantan servidores, routers y redes completas dentro del servidor. Se utilizan virtualizadores clase 3 como virtualbox para ejecutar software de lanzamiento como Vagrant o Docker sobre Ubuntu y Ubuntu Server.

## 2 Taller 1 - Orquestación en OpenStack (I)

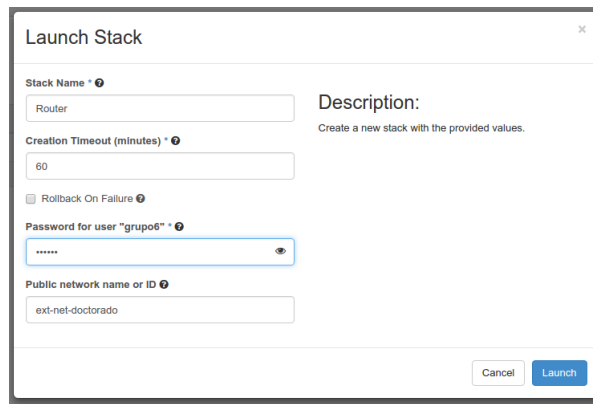
El primer paso para familiarizarse con el uso de OpenStack y el lenguaje por el cual accedemos al mismo para realizar la orquestación de servicios (RESTful) es crear un router, así para el desarrollo del proyecto se crea un archivo denominado “router.yaml” con el siguiente contenido:

```
1 heat\_template\_version: 2013-05-23
3
5 description: This template deploys a router with a port in the public interface
7
9 parameters:
11     public\_network:
12         type: string
13         label: Public network name or ID
14         description: Public network with floating IP addresses.
15         default: ext-net-doctorado
17
18 resources:
19     router:
20         type: OS::Neutron::Router
21         properties:
22             external\_gateway\_info:
23                 network: \{ get\_param: public\_network \}
```

Los que hace este código es crear un router y su configuración interna con una ip flotante, las imágenes que verifican su creación se aprecian en la figura 1, 2, 3.

Ahora se pretende visualizar la creación de una red completa se crea un archivo denominado “network.yaml” con el siguiente contenido:

```
2 heat\_template\_version: 2013-05-23
4
6 description: This template deploys a router with a port in the public interface
8
10 parameters:
12     private\_network\_cidr:
13         type: string
14         label: Private network CIDR
15         description: Private Network CIDR
```



**Launch Stack**

Stack Name \*

Description:  
Create a new stack with the provided values.

Creation Timeout (minutes) \*

☐ Rollback On Failure

Password for user "grupo6" \*

Public network name or ID \*

Figure 1: Router creado desde la orquestación

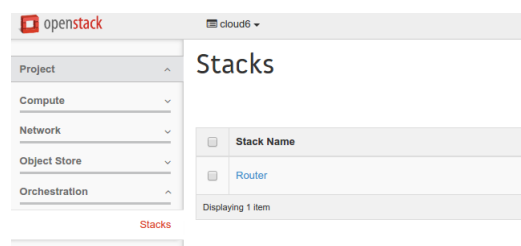


Figure 2: Router creado desde la orquestación en stack

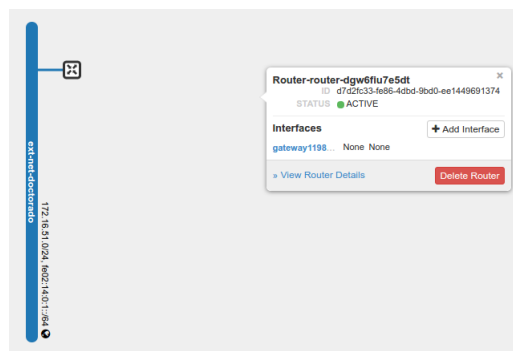


Figure 3: Router en la topología de red

```

12     default: 192.168.200.0/24
14 resources:
16     private_network:
17         type: OS::Neutron::Net
18
19     private_subnet:
20         type: OS::Neutron::Subnet
21         properties:
22             network_id: { get_resource: private_network }
23             cidr: {get_param: private_network_cidr}
24             dns_nameservers:
25                 - 8.8.8.8

```

Lo que hace esto es generar una red que consta de una interfaz de red y un router, los cuales se comunican

Figure 4: Red creada desde la orquestación

Stack Name
network

Figure 5: Red creada desde la orquestación en stack



Figure 6: Red vista en la topología de red

por la ip flotante configurada previamente. El siguiente paso de este esquema es enlazar el router con la red que se ha creado previamente. Como el fin de la orquestación es generar por medio de código configuraciones de red, y se pretende hacer una red completa creamos un archivo denominado “complete-network.yaml” con el siguiente contenido presentado a continuación. 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.

```
1 heat_template_version: 2013-05-23
3
```

```

description: This template deploys a router with a port in the public interface
5
parameters:
7
  public_network:
9    type: string
    label: Public network name or ID
    description: Public network with floating IP addresses.
11    default: ext-net-doctorado
13
  private_network_cidr:
15    type: string
    label: Private network CIDR
    description: Private Network CIDR
17    default: 192.168.200.0/24
19
resources:
21
  router:
23    type: OS::Neutron::Router
    properties:
25      external_gateway_info:
        network: { get_param: public_network }
27
  private_network:
29    type: OS::Neutron::Net
31
  private_subnet:
    type: OS::Neutron::Subnet
33    properties:
      network_id: { get_resource: private_network }
35      cidr: {get_param: private_network_cidr}
      dns_nameservers:
37        - 8.8.8.8
39
  router-interface:
    type: OS::Neutron::RouterInterface
41    properties:
      router_id: { get_resource: router }
43    subnet: { get_resource: private_subnet }

```

El desarrollo de esta sección de la actividad puede visualizarse en la figura 7, 8, 9.

Una vez esta desplegada 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 una IP flotante. Para esto lo que realizamos es:

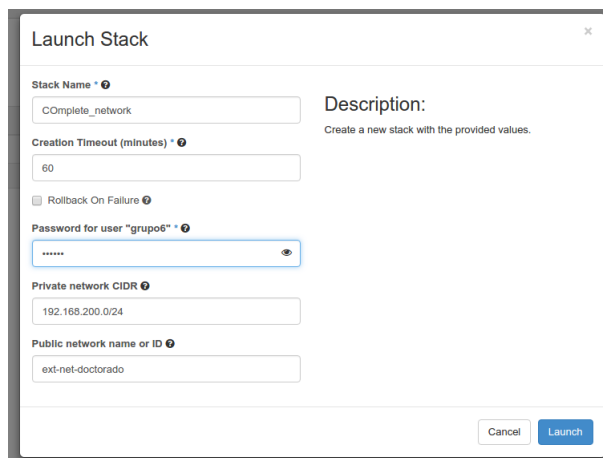
- Eliminar las pilas previamente creadas desde el control principal de pilas o stack.
- Crear un archivo denominar “network-server.yaml” con el contenido presentado a continuación.

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
2
description: This template deploys a router, a private network and a single basic
  server with a security group.
4
parameters:
6

```



**Launch Stack**

Stack Name \*

Creation Timeout (minutes) \*

☐ Rollback On Failure

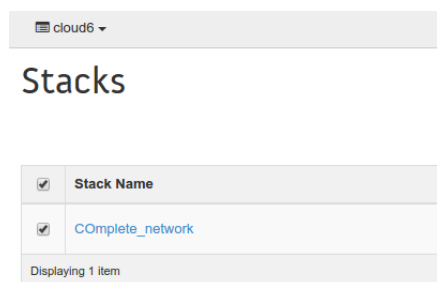
Password for user "grupo6" \*

Private network CIDR

Public network name or ID

Description:  
Create a new stack with the provided values.

Figure 7: Red completa creada desde la orquestación



cloud6	
Stacks	
<input checked="" type="checkbox"/>	Stack Name
<input checked="" type="checkbox"/>	CComplete_network
Displaying 1 item	

Figure 8: Red completa creada desde la orquestación en stack



Figure 9: Red completa desde el visualizador de topología

```

8 public_network:
   type: string
   label: Public network name or ID
10 description: Public network with floating IP addresses.

```

```
    default: ext-net-doctorado
12
private_network_cidr:
14
    type: string
16    label: Private network CIDR
    description: Private Network CIDR
18    default: 192.168.200.0/24

20 image:

22     type: string
    label: Image name or ID
24     description: Image to be used for compute instance
    default: Ubuntu-Server-14.04-CECAD-r20141201
26
flavor:
28     type: string
    label: Flavor
30     description: Type of instance (flavor) to be used
    default: m1.small
32
resources:
34
router:
36     type: OS::Neutron::Router
    properties:
38         external_gateway_info:
            network: { get_param: public_network }
40
private_network:
42     type: OS::Neutron::Net

44 private_subnet:
    type: OS::Neutron::Subnet
46     properties:
        network_id: { get_resource: private_network }
48         cidr: {get_param: private_network_cidr}
        dns_nameservers:
50             - 8.8.8.8

52 router-interface:
    type: OS::Neutron::RouterInterface
54     properties:
        router_id: { get_resource: router }
56         subnet: { get_resource: private_subnet }

58 web_server_security_group:
    type: OS::Neutron::SecurityGroup
60     properties:
        name: web_server_security_group
62         rules:
            - protocol: tcp
64              port_range_min: 80
              port_range_max: 80
66            - protocol: tcp
              port_range_min: 443
68              port_range_max: 443
            - protocol: icmp
70            - protocol: tcp
```

```

    port_range_min: 22
    port_range_max: 22

72
my_keypair:
  type: OS::Nova::KeyPair
74
  properties:
    name: cloudapps
76
    save_private_key: True
78

my_instance:
  type: OS::Nova::Server
80
  properties:
    image: { get_param: image }
82
    flavor: { get_param: flavor }
    key_name: { get_resource: my_keypair }
84
    networks:
      - network: { get_resource: private_network }
86
    security_groups:
      - { get_resource: web_server_security_group }
88
    user_data: |
      #!/bin/sh
90
      sudo apt-get -y update  sudo apt-get -y install apache2  sudo service
92
      apache2 restart
    user_data_format: RAW
94

outputs:
  my_instance_name:
96
    description: Name of the instance
    value: { get_attr: [my_instance, name] }
98
  my_instance_ip:
    description: IP address of the instance
100
    value: { get_attr: [my_instance, first_address] }
```

Launch Stack

Stack Name \*

network-server

Description:

Create a new stack with the provided values.

Creation Timeout (minutes) \*

60

☐ Rollback On Failure

Password for user "grupo6" \*

\*\*\*\*\*

Flavor

m1.small

Image name or ID

Ubuntu-Server-14.04-CEC4D-r20141201

Private network CIDR

192.168.200.0/24

Public network name or ID

ext-net-doctorado

Cancel Launch

Figure 10: Red con servidor creada desde la orquestación

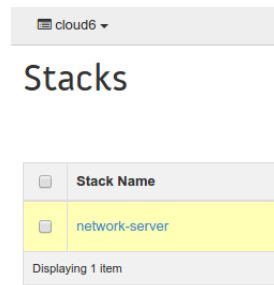


Figure 11: Red con servidor creada desde la orquestación en stack

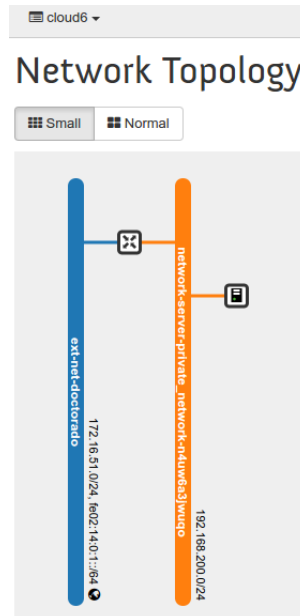


Figure 12: Red con servidor vista como topología

### 3 Taller 3

#### 3.1 Objetivo

#### 3.2 Actividades

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

- Abrir una consola de comandos.
- Validar la correcta instalación del software Vagrant ejecutando el comando `vagrant -h`, la validación se observa en la figura 13.
- Ahora procedemos a agregar la imagen del sistema operativo Ubuntu Trusty de 64 bits. Para ello, ejecutamos el comando `vagrant box add ubuntu/trusty64`. La descarga se demora segun la conexión a internet, es probable que dure un par de horas.
- Procedemos a crear un directorio de trabajo clouapps (o cualquier otro nombre). Para luego ingresar a ese directorio en la consola y ejecutar el comando `vagrant init ubuntu/trusty64`. Después de ejecutar el comando, verificamos que se haya creado un archivo denominado `Vagrantfile`. Esto se verifica en la figura 14.



```

pepo@pepo-pc: ~
File Edit View Search Terminal Help
pepo@pepo-pc:~$ sudo vagrant -h
[sudo] password for pepo:
Usage: vagrant [options] <command> [<args>]

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

Common commands:
    box                    manages boxes: installation, removal, etc.
    destroy               stops and deletes all traces of the vagrant machine
    global-status          outputs status Vagrant environments for this user
    halt                  stops the vagrant machine
    help                  shows the help for a subcommand
    init                  initializes a new Vagrant environment by creating a V
agrantfile
    login                 log in to HashiCorp's Atlas
    package               packages a running vagrant environment into a box
    plugin                manages plugins: install, uninstall, update, etc.
    port                  displays information about guest port mappings
    powershell            connects to machine via powershell remoting
    provision              provisions the vagrant machine
    push                  deploys code in this environment to a configured dest
ination
    rdp                   connects to machine via RDP
    reload                restarts vagrant machine, loads new Vagrantfile confi
guration
    resume                resume a suspended vagrant machine
    snapshot              manages snapshots: saving, restoring, etc.
    ssh                   connects to machine via SSH
    ssh-config            outputs OpenSSH valid configuration to connect to the
machine
    status                outputs status of the vagrant machine
    suspend              suspends the machine
    up                    starts and provisions the vagrant environment
    version               prints current and latest Vagrant version

For help on any individual command run `vagrant COMMAND -h`

Additional subcommands are available, but are either more advanced

```

Figure 13: Resultado de uso de comando

```

pepo@pepo-pc:~$ mkdir cloudapps
pepo@pepo-pc:~$ cd cloudapps
pepo@pepo-pc:~/cloudapps$ sudo vagrant init ubuntu/trusty64
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
pepo@pepo-pc:~/cloudapps$ ls
Vagrantfile

```

Figure 14: Resultado de uso de comando

- Procedemos a editar el archivo Vagrantfile de forma que contenga la información siguiente, este procedimiento y su salida se consigna en la figura 15.

```

1      # -*- mode: ruby -*-
2      # vi: set ft=ruby :
3      # All Vagrant configuration is done below. The "2" in Vagrant.
   configure
4      # configures the configuration version (we support older styles for
5      # backwards compatibility). Please dont change it unless you know what
6      # youre doing.
7      Vagrant.configure(2) do |config|
8      config.vm.box = "ubuntu/trusty64"
9      config.vm.network :forwarded_port, guest: 4000, host: 8100, host_ip: "

```

```

11 |         127.0.0.1"
    |     config.vm.provision "shell", path: "script.sh"
    | end

```

```
pepo@pepo-pc: ~/cloudapps
File Edit View Search Terminal Help
GNU nano 2.5.3 File: Vagrantfile Modified
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.

Vagrant.configure(2) do |config|

  config.vm.box = "ubuntu/trusty64"

  config.vm.network :forwarded_port, guest: 4000, host: 8100, host_ip: "127.0.0.1"

  config.vm.provision "shell", path: "script.sh"

end
```

<sup>^</sup>G Get Help    <sup>^</sup>O Write Out    <sup>^</sup>W Where Is    <sup>^</sup>K Cut Text    <sup>^</sup>J Justify  
<sup>^</sup>X Exit    <sup>^</sup>R Read File    <sup>^</sup>\ Replace    <sup>^</sup>U Uncut Text    <sup>^</sup>T To Spell

Figure 15: Resultado de uso de comando

- Creamos un archivo “script.sh” en el mismo directorio del archivo Vagrantfile e incluimos en el script la siguiente información:

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

Este paso se aprecia en la figura 16.



```
pepo@pepo-pc: ~/cloudapps
File Edit View Search Terminal Help
GNU nano 2.5.3 File: script.sh Modified
#!/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
```

Figure 16: Resultado de uso de comando

- Ejecutamos el comando `vagrant up --provision`, para verificar la correcta instalación y desempeño de la aplicación, esto se presenta en la figura 17.
- Verificar el correcto funcionamiento del despliegue accediendo en un navegador (browser) a la dirección `http://127.0.0.1:8100`, visto en la figura 18.

## 4 Taller 4

### 4.1 Objetivo

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

### 4.2 Actividades

- Abrir una consola de comandos.
- Validar la correcta instalación del software Vagrant ejecutando el comando `vagrant -h`
- Adicionar la imagen del sistema operativo Ubuntu Trusty de 64 bits. Para ello, ejecutar el comando `vagrant box add ubuntu/trusty64`.

```

==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
default: The guest additions on this VM do not match the installed ver
sion of
default: VirtualBox! In most cases this is fine, but in rare cases it
can
default: prevent things such as shared folders from working properly.
If you see
default: shared folder errors, please make sure the guest additions wi
thin the
default: virtual machine match the version of VirtualBox you have inst
alled on
default: your host and reload your VM.
default:
default: Guest Additions Version: 4.3.36
default: VirtualBox Version: 5.0
==> default: Mounting shared folders...
default: /vagrant => /home/pepo/cloudapps
==> default: Running provisioner: shell...
default: Running: /tmp/vagrant-shell20160429-3779-lqrrub.sh
==> default: stdin: is not a tty
==> default: Installing: nodejs, lynx, ruby and jekyll...
==> default: Configuration file: none
==> default: Source: /vagrant
==> default: Destination: /vagrant/_site
==> default: Incremental build: disabled. Enable with --incremental
==> default: Generating...
==> default: done in 0.014 seconds.
==> default: Auto-regeneration: enabled for '/vagrant'
==> default: Configuration file: none
==> default: Server address: http://0.0.0.0:4000/
==> default: Server running... press ctrl-c to stop.

```

Figure 17: Resultado de uso de comando

Name	Last modified	Size
Parent Directory	2016/04/29 19:49	-
Vagrantfile	2016/04/29 19:42	473
script.sh	2016/04/29 19:44	549

WEBrick/1.3.1 (Ruby/2.2.4/2015-12-16)  
at 127.0.0.1:8100

Figure 18: Resultado de uso de comando

- 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`.
- Editar el archivo `Vagrantfile` de forma que contenga la siguiente información (visualizado el resultado en la figura 19):

```
1 # -*- mode: ruby -*-
```

```

pepo@pepo-pc:~$ mkdir cloudapps2
pepo@pepo-pc:~$ cd cloudapps2
pepo@pepo-pc:~/cloudapps2$ vagrant init ubuntu/trusty 64
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
pepo@pepo-pc:~/cloudapps2$ ls
Vagrantfile
pepo@pepo-pc:~/cloudapps2$ sudo gedit Vagrantfile

```

Figure 19: Resultado de uso de comando

```

3      # vi: set ft=ruby :
      # All Vagrant configuration is done below. The "2" in Vagrant.
      configure
      # configures the configuration version (we support older styles for
5      # backwards compatibility). Please don't change it unless you know what
      # you're doing.
7      Vagrant.configure(2) do |config|
      config.vm.box = "ubuntu/trusty64"
9      config.vm.network :forwarded_port, guest: 4000, host: 8100, host_ip: "
        127.0.0.1"
      config.vm.provision "puppet"
11     config.vm.hostname = "www.cecad-example.com"
      end

```

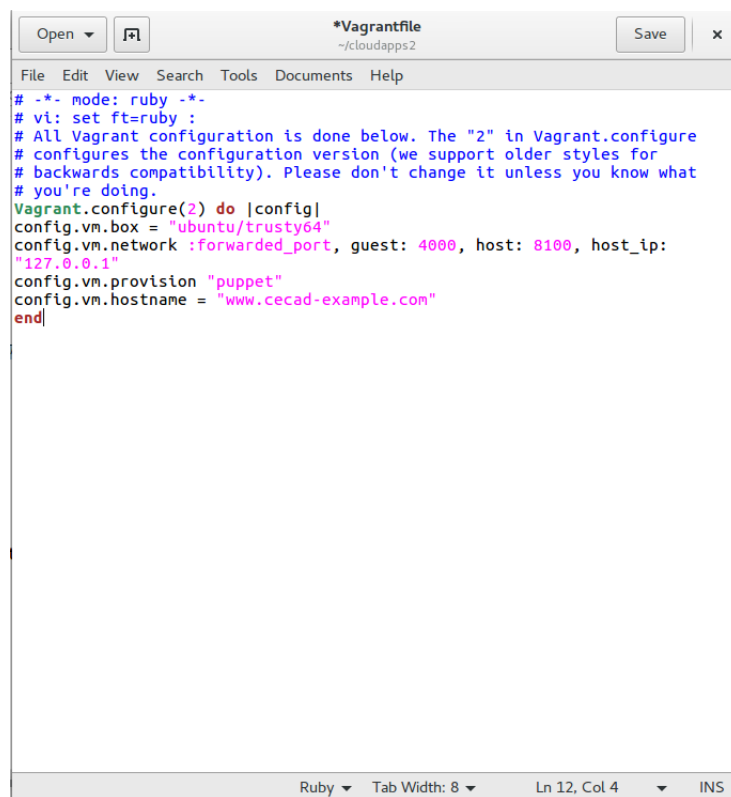


Figure 20: Resultado de uso de comando

- Crear un archivo “default.pp” en un directorio manifests que se encuentra en el mismo lugar del archivo Vagrantfile.
- Escribir el siguiente contenido en el archivo “default.pp”.

```
2      exec { apt-get update:
3        command => /usr/bin/apt-get update -y
4      }
5
6      package { nodejs:
7        require => Exec[apt-get update]
8      }
9
10     package { lynx-cur:
11       require => Exec[apt-get update]
12     }
13
14     package { ruby1.9.1-dev:
15       require => Exec[apt-get update]
16     }
```

- Ejecutar el comando `vagrant up --provision`. El resultado es verificado en la figura 20

## 5 Taller 6 - Fundamentos de Docker

### 5.1 Obetivo

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

### 5.2 Actividades

- Verificar la correcta instalación del servicio Docker ejecutando el comando(Verificado en la imagen 26):

```
sudo service docker status
```

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

```
sudo docker pull solr
```

- Listar las imágenes de Docker disponibles. Debe aparecer la imagen de solr descargada.

```
sudo docker images (Verificado en la imagen 22)
```

- 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:

1. `docker run` es el comando para ejecutar un nuevo contenedor Docker. Este comando recibe varios parámetros.
2. `-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.
3. `-d` especifica que el contenedor se va a ejecutar en background.
4. `--name` especifica un nombre para el contenedor. En el comando anterior, el contenedor se llama `mysolr`.
5. Cuando se lanza el contenedor, debe especificarse su imagen base; en este caso, `solr` es el nombre de la imagen.

```

pepo@pepo-pc: ~
File Edit View Search Terminal Help
pepo@pepo-pc:~$ sudo service docker status
[sudo] password for pepo:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor pre
   set: e
   Active: active (running) since vie 2016-04-29 15:10:12 COT; 56min ago
     Docs: https://docs.docker.com
    Main PID: 935 (docker)
      Tasks: 21 (limit: 512)
     Memory: 43.7M
        CPU: 1.558s
    CGroup: /system.slice/docker.service
            └─935 /usr/bin/docker daemon -H fd://
              967 docker-containerd -l /var/run/docker/libcontainerd/docker
-conta

abr 29 15:10:11 pepo-pc docker[935]: time="2016-04-29T15:10:11.994730922-0
5:00"
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.085610801-0
5:00"
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.168757250-0
5:00"
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.171585478-0
5:00"
abr 29 15:10:12 pepo-pc docker[935]: .
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.175309294-0
5:00"
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.175781730-0
5:00"
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.176060510-0
5:00"
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.203714214-0
5:00"
abr 29 15:10:12 pepo-pc systemd[1]: Started Docker Application Container E
ngine.

```

Figure 21: Resultado de uso de comando

```

pepo@pepo-pc:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
hello-world         latest             94df4f0ce8a4       2 days ago
solr                 latest             c6d71a5f9e4f       3 weeks ago
568 MB

```

Figure 22: Resultado de uso de comando

```

pepo@pepo-pc:~$ sudo docker run -p 8983:8983 -d --name mysolr solr
0de0941bcb146a5a0eb8a72393981e40d78f5e0a4e801311fa638dd000d5f728
pepo@pepo-pc:~$ sudo docker ps
CONTAINER ID        STATUS             IMAGE              PORTS              COMMAND              NAMES              CREATED
0de0941bcb14        Up 20 seconds     solr               0.0.0.0:8983->8983/tcp  "/opt/solr/bin/solr -" mysolr                21 second
ago

```

Figure 23: Resultado de uso de comando

- Verificar que el contenedor se esté ejecutando. Para ello, ejecutar el comando

sudo docker ps (Verificado en la imagen 23)

- 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> (Verificado en la imagen 24)

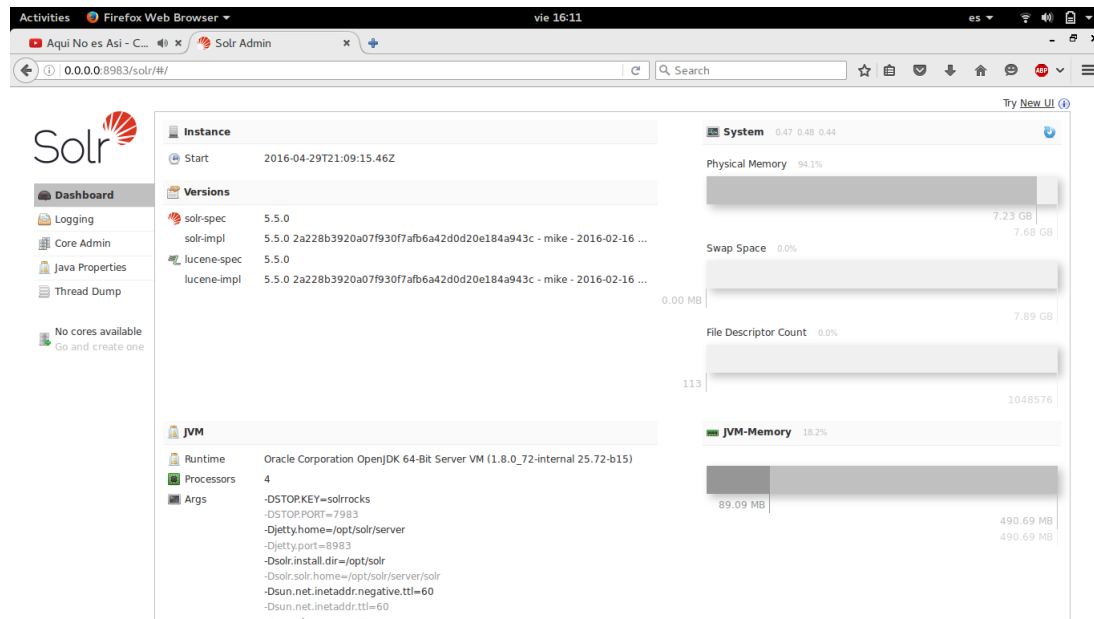


Figure 24: Resultado de uso de comando

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

- Ejecutar el comando

```
bin/solr create_core -c gettingstarted (Verificado en la imagen 25)
```

- 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:

```

1  <add>
2  <doc>
3    <field name="id">SOLR1000</field>
4    <field name="name">Solr, the Enterprise Search Server</field>
5    <field name="manu">Apache Software Foundation</field>
6    <field name="cat">software</field>
7    <field name="cat">search</field>
8    <field name="features">Advanced Full-Text Search Capabilities using
9      Lucene</field>
10   <field name="features">Optimized for High Volume Web Traffic</field>
11   <field name="features">Standards Based Open Interfaces - XML and HTTP
12   </field>
13   <field name="features">Comprehensive HTML Administration Interfaces</
14   field>
15   <field name="features">Scalability - Efficient Replication to other
16   Solr Search Servers</field>

```



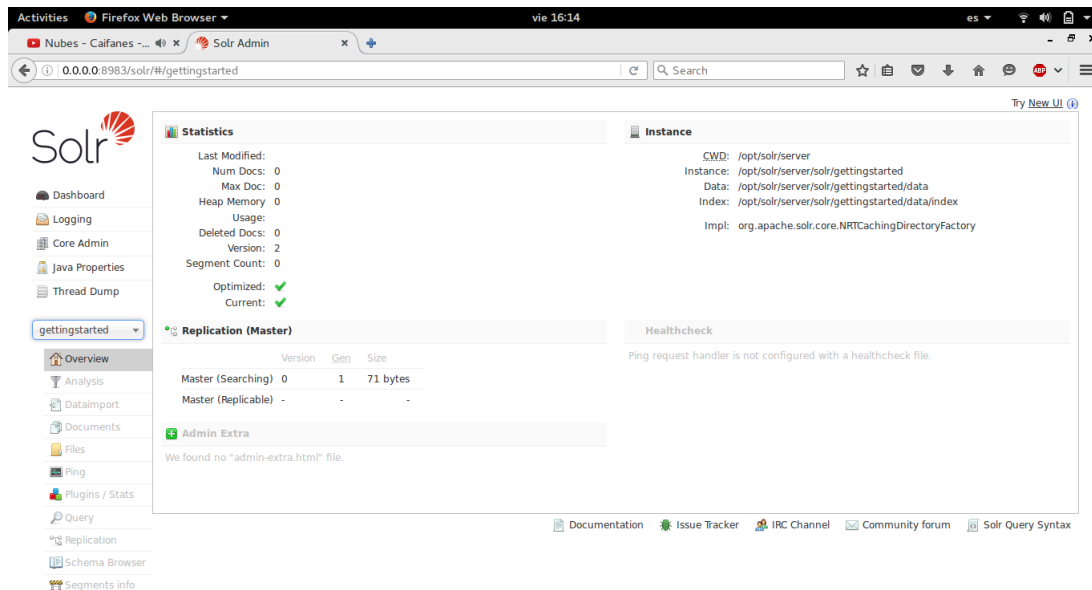


Figure 25: Resultado de uso de comando

```

14      <field name="features">Flexible and Adaptable with XML configuration
15          and Schema</field>
16      <field name="features">Good unicode support: h#xE9;llo (hello with an
17          accent over the e)</field>
18      <field name="price">0</field>
19      <field name="popularity">10</field>
20      <field name="inStock">true</field>
      <field name="incubationdate_dt">2006-01-17T00:00:00.000Z</field>
      </doc>
      </add>

```

- 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
```

- Ingresar nuevamente a la consola. Verificar la existencia del archivo `solr.xml` y ejecutar el comando

```
bin/post -c gettingstarted ./solr.xml
```

- 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.
- Detener el contenedor.

```
sudo docker stop mysolr
```

- Verificar que el contenedor aparezca como "Exited" en su estado después de ejecutar el comando

```
sudo docker ps -a
```

- Eliminar el contenedor.

```
sudo docker rm mysolr
```

- En caso de requerirse, es posible eliminar la imagen utilizando el comando

```
sudo docker rmi solr
```

## 6 Taller 7 - Fundamentos de Docker II

### 6.1 Objetivo

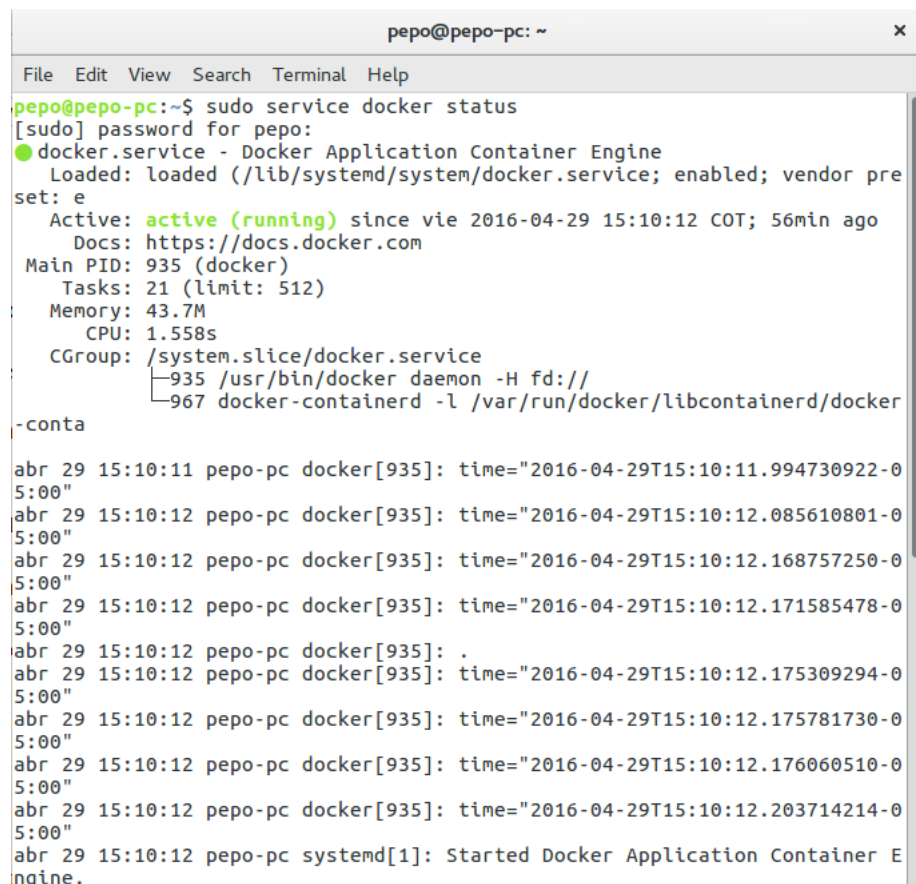
Realizar el despliegue de aplicaciones sencillas utilizando la tecnología Docker sobre el sistema operativo Linux Ubuntu Server.

### 6.2 Actividades

- Verificar la correcta instalación del servicio Docker ejecutando el comando.

```
sudo service docker status
```

Como se ve en la figura 26.



```
pepo@pepo-pc: ~  
File Edit View Search Terminal Help  
pepo@pepo-pc:~$ sudo service docker status  
[sudo] password for pepo:  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor pre  
   Set: e  
   Active: active (running) since vie 2016-04-29 15:10:12 COT; 56min ago  
     Docs: https://docs.docker.com  
    Main PID: 935 (docker)  
      Tasks: 21 (limit: 512)  
     Memory: 43.7M  
        CPU: 1.558s  
    CGroup: /system.slice/docker.service  
            └─935 /usr/bin/docker daemon -H fd://  
              └─967 docker-containerd -l /var/run/docker/libcontainerd/docker  
-conta  
abr 29 15:10:11 pepo-pc docker[935]: time="2016-04-29T15:10:11.994730922-0  
5:00"  
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.085610801-0  
5:00"  
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.168757250-0  
5:00"  
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.171585478-0  
5:00"  
abr 29 15:10:12 pepo-pc docker[935]: .  
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.175309294-0  
5:00"  
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.175781730-0  
5:00"  
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.176060510-0  
5:00"  
abr 29 15:10:12 pepo-pc docker[935]: time="2016-04-29T15:10:12.203714214-0  
5:00"  
abr 29 15:10:12 pepo-pc systemd[1]: Started Docker Application Container E  
ngine.
```

Figure 26: Resultado de uso de comando

- 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".

- Ingresar el siguiente contenido en el archivo:

```
2 FROM Ubuntu:trusty
  RUN sudo apt-get update sudo apt-get install cowsay fortune
```

- Construir una nueva imagen a partir del Dockerfile.

```
sudo docker build -t test/dockerfile-example
```

- Ejecutar un nuevo contenedor a partir de la imagen creada. (Verificado en la imagen 27)

```
sudo docker run test \dockerfileexample \usr/games/cowsay Hola
a todos!
```

```
pepo@pepo-pc:~/TalleresCloud/Taller7/Docker$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
test/dockerfile-example latest             9980e7ff41f8       41 minutos ago
hello-world         latest             94df4f0ce8a4       2 days ago
go                   967 B             8fa7f61732d6       4 days ago
ubuntu              188 MB            c6d71a5f9e4f       3 weeks ago
solr                 568 MB
pepo@pepo-pc:~/TalleresCloud/Taller7/Docker$ sudo docker run --name Docker
Tes 9980e7ff41f8 /usr/games/cowsay "Hola a todos"
< Hola a todos >
  -----
  \      ^__^
   \     (oo)\_______
      (__)\       )\/\
         ||----w |
         ||     ||
```

Figure 27: Resultado final que muestra la vaca hablando

## 7 Bibliografía

- Notas de clase y presentaciones realizadas