Sistemas Distribuidos
Informe y Desarrollo del Primer Parcial
GitHub: https://github.com/andres2508/VagrantConsultDB

Jaime Andres Aristizabal Lozada
11104055

Introducción:

En el presente informe se expone la arquitectura y los métodos necesarios para realizar la arquitectura propuesta para el primer parcial, la cual se define en la ilustración 1. El aprovisionamiento planteado en la solución está basado en Vagrant y Cookbooks de Chef. Este informe se dividirá en tres partes, la primera parte comprenderá el aprovisionamiento basado en Vagrant. Así mismo en la segunda parte se presentaran los Cookbooks necesarios para aprovisionar un micro framework, el cual será el puente para consultar la base de datos. Por último, la tercera parte responde a la maquina encargada de administrar la base de datos.

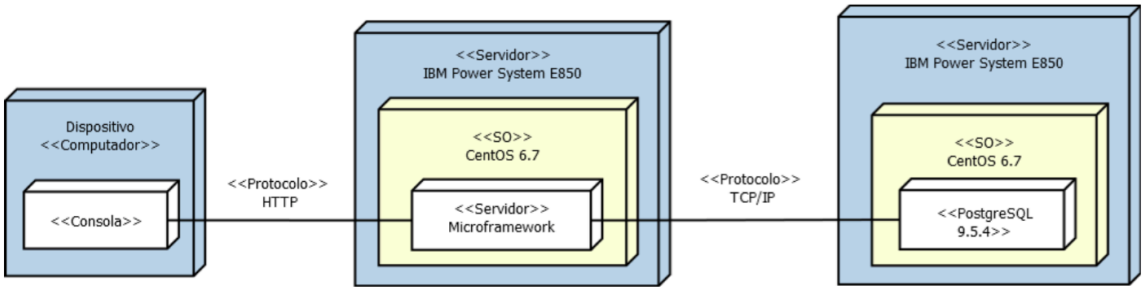


Ilustración 1, Arquitectura primer parcial

1. Aprovisionamiento en Vagrant:

Según la arquitectura que teníamos que aprovisionar, se decidió levantar dos máquinas que se encargarían de contener los dos servicios. La primera máquina que se encarga de contener el micro framework CherryPy y la segunda maquina contendrá la base de datos PostgresSQL. La configuración de la primera máquina es presentada en la ilustración 2.

```
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.define :centos_web_1 do |node|
    node.vm.box = "centos6.4"
    node.vm.network :private_network, ip: "192.168.56.52"
    node.vm.network "public_network", :bridge => "eth2", ip:"192.168.131.52", :auto_config => "false", :netmask => "255.255.255.0"
    node.vm.provider :virtualbox do |vb|
      vb.customize ["modifyvm", :id, "--memory", "1024","--cpus", "2", "--name", "centos_web_1" ]
    end
    config.vm.provision :chef_solo do |chef|
        chef.cookbooks_path = "cookbooks"
        chef.add_recipe "mirror"
        chef.add_recipe "cherrypy"
        chef.json = {"aptmirror" => {"server" => "192.168.131.254", "server_ip" => "192.168.131.52", "server_port" => "80",
                "database_ip" => "192.168.131.54"}}
    end

  end
```

Ilustración 2, Configuración primera máquina, CherryPy

Como se puede ver en la configuración se están utilizando nodos sobre centos6.4, así mismo se les está dando una dirección ip privada y pública. Dentro de esta configuración podemos ver que se están agregando dos recipes, uno que se encarga de levantar toda la configuración del mirror. Esta configuración es básica, lo único que hace es cambiar la dirección ip del mirror por defecto, al mirror del laboratorio. Del mismo modo se agrega un recipe que contiene toda la instalación y configuración del micro framework CherryPy. Este recipe contiene archivos que se alimentan de la dirección ip publica y el puerto del micro framework para exponer el servicio y que sea visible por el protocolo HTTP, así mismo también recibe la dirección ip de la base de datos, esta será la dirección donde se realizara la consulta al servicio PostgreSQL.

Por otro lado se encuentra la configuración de la máquina que contiene el servicio de la base de datos. La configuración de esta máquina se presenta en la ilustración 3.

```
  config.vm.define :centos_database do |db|
    db.vm.box = "centos6.4"
    db.vm.network :private_network, ip: "192.168.56.54"
    db.vm.network "public_network", :bridge => "eth2", ip:"192.168.131.54", :auto_config => "false", :netmask => "255.255.255.0"
    db.vm.provider :virtualbox do |vb|
      vb.customize ["modifyvm", :id, "--memory", "1024","--cpus", "1", "--name", "centos_database" ]
    end
    config.vm.provision :chef_solo do |chef|
      chef.cookbooks_path = "cookbooks"
      chef.add_recipe "mirror"
      chef.add_recipe "postgres"
      chef.json ={"aptmirror" => {"server" => "192.168.131.254"}}
    end
  end


end
```

Ilustración 3, Configuración de Segunda Maquina, PostgreSQL

El aprovisionamiento de esta máquina es esencialmente parecido a la primera máquina. La diferencia radica en que esta máquina se carga un recipe que contiene toda la configuración del servicio PostgreSQL y también tiene una tabla cargada con un ejemplo, con el cual se realizaran las pruebas. Más adelante en la presentación de los cookbooks se explicara detalladamente este servicio.

2. Cookbooks para Aprovisionar CherryPy:

Para el aprovisionar la máquina que contendrá el micro framework, se tuvo la estructura de archivos presentada en la ilustración 4. Esta estructura contempla levantar las dependencias necesarias para que CherryPy funcione sobre una maquina Centos y también contiene el script en Python que expone el servicio de consulta de base de datos por el protocolo http.

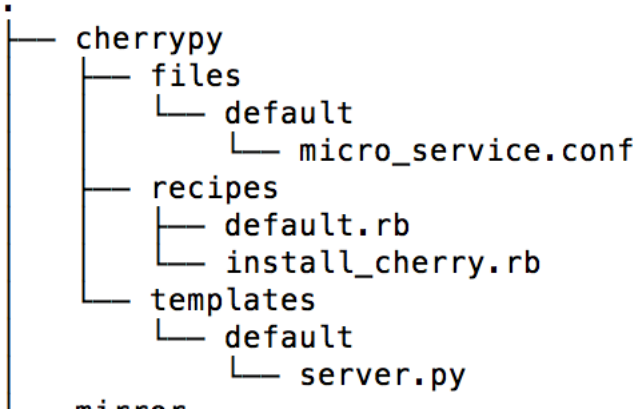


Ilustración 4, Estructura cookbook cherrypy

La primera parte que se explicara será como se realizó la instalación del micro framework. Dentro de la carpeta recipes se encuentra el archivo install_cherry.rb. Dentro de este Ruby, se instala el cliente de PostgreSQL, el micro framework CherryPy, el cliente de PostgreSQL para Python y se realizan operaciones para que sirva el servicio web. En la ilustración 5 se muestra la primera parte de la configuración y en la ilustración 6 se muestra la segunda parte.

```
yum_package 'php'do
    action :install
end
yum_package "php-pgsql" do
    action :install
end
execute 'pg' do
 command 'yum localinstall http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-2.noarch.rpm -y'
end

yum_package 'postgresql94' do
 action :install
end

bash "install cherrypy" do
        user "root"
        code <<-EOH
        curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
        sudo python get-pip.py
        sudo pip install CherryPy
        EOH
end

yum_package 'python-psycopg2.i686'do
    action :install
end

bash "open port" do
        user "root"
        code <<-EOH
        iptables -I INPUT 5 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
        service iptables save
        EOH
end

template "/home/vagrant/server.py" do
        source "server.py"
        mode 0644
        owner "root"
        group "wheel"
        variables({
                :ip_database => "#{node[:aptmirror][:databaseip]}",
                :ip_server => "#{node[:aptmirror][:serverip]}",
                :port_server => "#{node[:aptmirror][:serverport]}"
        })
end
```

Ilustración 5, Primera parte configuración install_cherry.rb

En esta parte podemos encontrar que se pasan los parámetros de la ip del servidor de base de datos como el del micro framework. Estos parámetros se pasan al archivo server.py el cual contiene lo necesario para arrancar el servicio web que consulta la base de datos. También se agrega una librería de Python que se llama python-psycopg2.i686, esta librería sirve para ejecutar consultas de PostgreSQL en Python. Otra cosa a tener en cuenta en la configuración, es que CherryPy se instala en por medio de PIP de Python. Esto se hace porque CherryPy no se encuentra disponible en los repositorios de Centos, por eso se realiza la instalación por comandos de BASH. Por último se abre el puerto 80 para que admita las conexiones en las iptables.

```
cookbook_file "/etc/init/micro_service.conf" do
        source "micro_service.conf"
        mode 0644
        owner "root"
        group "wheel"
end

bash "server execute" do
        user "root"
        code <<-EOH
        start micro_service
        EOH
end
```

Ilustración 6, Segunda parte configuración install_cherry.rb

Para esta parte se puede observar que se copia un archivo llamado micro_service.conf. Este archivo nos va a permitir ejecutar el server.py como un servicio de la maquina Centos. Esto se hace porque al momento de levantar el Vagrant se queda esperando una respuesta al ejecutar normalmente el server.py, pero este se queda ejecutando en BASH mostrando las conexiones entrantes. Esto es porque server.py no corre en un hilo de la máquina, por eso debe ejecutarse como un servicio. La última línea hace referencia a la ejecución del micro_service.

Por otro lado, en la carpeta "files" se encuentran dos archivos, el server.py y el micro_service.conf. La configuración del server.py se presenta en la ilustración 7. Este archivo comprende la consulta hacia la base de datos que se realiza por medio de psycopg2. Además expone el servicio web de la consulta de la base de datos por medio de un @cherrypy.expose. Por ultimo este servicio se expone sobre la ip publica de la máquina del micro framework y el puerto 80 que es pasado por parámetros del json del vagrant file.

```
import cherrypy
import psycopg2
import pprint

class DatabaseConsult(object):
    @cherrypy.expose
    def index(self):
        host = "host="+"<%= @ip_database %>"+" "
        port = "port="+str(5432)+" "
        db = "dbname="+'swn'+" "
        user = "user="+'pi'+" "
        passwd = "password="+'security++'+" "

        conn_string = host+port+db+user+passwd
        conn = psycopg2.connect(conn_string)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM devices")
        records = cursor.fetchall()
        #pprint.pprint(records)
        #print((records['device_name']))
        return "This are database values"+" id_divice: "+str(records[0][0]) + " name_device: "+str(records[0][1])

    cherrypy.config.update({'server.socket_host': '<%= @ip_server %>', 'server.socket_port': <%= @port_server %>})
if __name__ == '__main__':
    cherrypy.quickstart(DatabaseConsult())
```

Ilustración 7, Configuración server.py

La configuración del archivo micro_service.conf, se encuentra en la ilustración 8. La configuración es sencilla, lo que hace es poner el servicio en ejecución de algún bin, el

cual es python2.6 y se expone en un puerto, el cuál es el 2345. Este archivo tiene que ir sobre la carpeta /etc/init/.

```
start on started sshd
stop on runlevel [!2345]

exec /usr/bin/python2.6 /home/vagrant/server.py
respawn
```

Ilustración 8, Configuración micro_service.conf

3. Cookbooks para Aprovisionar PostgreSQL:

La configuración de la máquina que sostiene la base de datos, se basó en el cookbook pasado por el profesor. Sin embargo se realizaron algunas modificaciones las cuales se mencionan a continuación, se irán en orden por carpeta. Cabe aclarar que cada uno de los archivos de configuración se anexaron en la entrega de este documento, para posterior validación. La distribución de carpetas con archivos es la presentada en la ilustración 9:

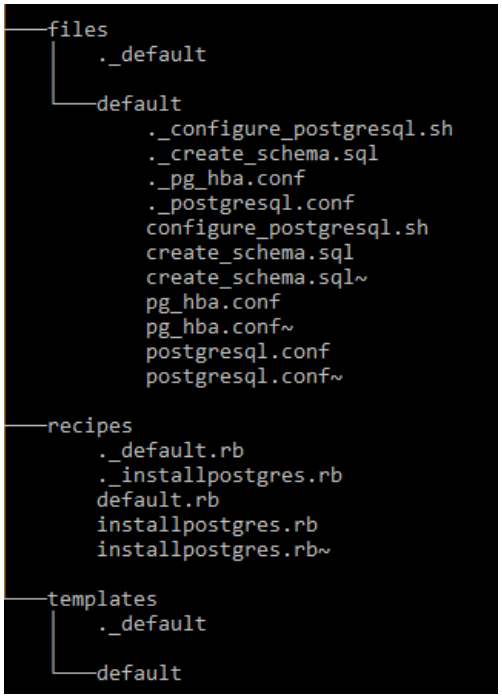


Ilustración 9, Tree de carpeta postgreSQL

Los archivos configure_postgresql.sh y create_schema.sql no tuvieron ningún cambio. Los cambio se dieron en pg_hba.conf, los cuales incluyeron la validación de las direcciones ip que podían acceder a la base de datos. La ilustración 10 representa el cambio realizado en las direcciones ip.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            ident
host    all             all             192.168.56.52/24         trust
host    all             all             192.168.56.53/24         trust
host    all             all             192.168.56.51/24         trust
host    all             all             192.168.131.53/24         trust
host    all             all             192.168.131.52/24         trust
host    all             all             192.168.131.51/24         trust
# IPv6 local connections:
host    all             all             ::1/128                 ident
```

Ilustración 10, Cambios en pg_hba.conf

En el archivo postgresql.conf, se realizaron cambios al principio que indican cuál va a ser la dirección ip de entrada, el puerto que está escuchando y el número de conexiones. La Ilustración 11 representan los cambios sobre el archivo.

```
# - Connection Settings -

listen_addresses = '*'      # what IP address(es) to listen on;
                    # comma-separated list of addresses;
                    # defaults to 'localhost'; use '*' for all
                    # (change requires restart)
port = 5432             # (change requires restart)
max_connections = 100           # (change requires restart)
```

Ilustración 11, Cambios en postgresql.conf

Dentro de la carpeta recipes se realizaron cambios en installpostgres.rb. Estos cambios corresponden al reinicio del servicio al final. Nos dimos cuenta que cuando terminaba de levantar las máquinas, el servicio de postgresql quedaba detenido por los cambios en los archivos de configuración, por lo tanto procedimos a reiniciar el servicio. Esa fue la línea que agregamos al final. La ilustración 12 representa los cambios realizados en el archivo installpostgres.rb.

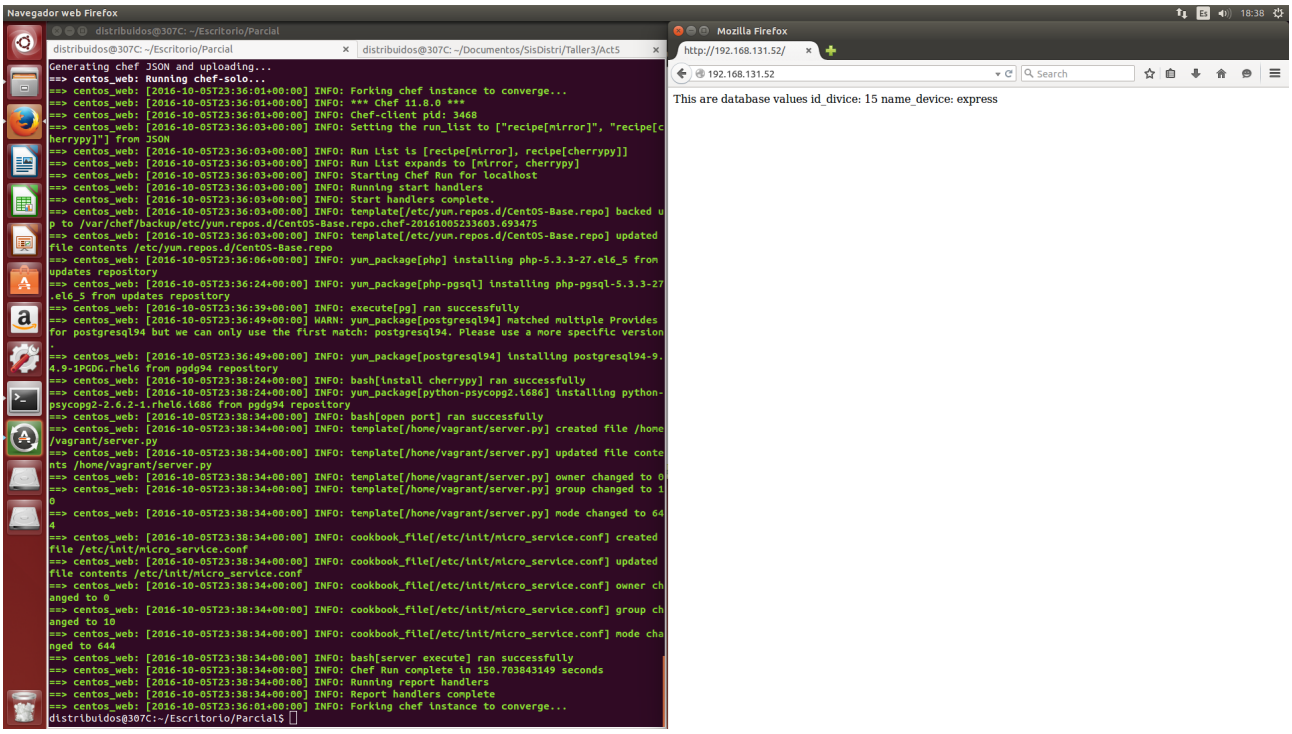
```
bash "postgres restart" do
    user "root"
    code <<-EOH
    service postgresql-9.4 restart
    EOH
end
```

Ilustración 12, Cambios en installpostgres.rb

Con esto se finaliza la configuración de la máquina que sostiene el servicio de la base de datos en el vagrantfile.

4. **Resultados:**

Se realizó la consulta por medio de la página web que exponía cherry py y el resultado es el presentado en la ilustración 13. El Vagrant se levanta con normalidad y se expone el servicio como un servicio del sistema operativo.


Navegador web Firefox
18:38
distribuidos@307C: ~/Escritorio/Parcial
distribuidos@307C: ~/Escritorio/Parcial
distribuidos@307C: ~/Documentos/SisDistri/Taller3/Act5
Generating chef JSON and uploading...
==> centos_web: Running chef-solo...
==> centos_web: [2016-10-05T23:36:01+00:00] INFO: Forking chef instance to converge...
==> centos_web: [2016-10-05T23:36:01+00:00] INFO: *** Chef 11.8.0 ***
==> centos_web: [2016-10-05T23:36:01+00:00] INFO: Chef-client pid: 3468
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: Setting the run_list to ["recipe[mirror]", "recipe[cherrypy]"] from JSON
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: Run List is [recipe[mirror], recipe[cherrypy]]
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: Run List expands to [mirror, cherrypy]
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: Starting Chef Run for localhost
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: Running start handlers
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: Start handlers complete.
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: template[/etc/yum.repos.d/CentOS-Base.repo] backed up to /var/chef/backup/etc/yun.repos.d/CentOS-Base.repo.chef-20161005233603.693475
==> centos_web: [2016-10-05T23:36:03+00:00] INFO: template[/etc/yum.repos.d/CentOS-Base.repo] updated file contents /etc/yun.repos.d/CentOS-Base.repo
==> centos_web: [2016-10-05T23:36:06+00:00] INFO: yum_package[php] installing php-5.3.3-27.el6_5 from updates repository
==> centos_web: [2016-10-05T23:36:24+00:00] INFO: yum_package[php-pgsql] installing php-pgsql-5.3.3-27.el6_5 from updates repository
==> centos_web: [2016-10-05T23:36:39+00:00] INFO: execute[pg] ran successfully
==> centos_web: [2016-10-05T23:36:49+00:00] WARN: yum_package[postgresql94] matched multiple Provides for postgresql94 but we can only use the first match: postgresql94. Please use a more specific version.
==> centos_web: [2016-10-05T23:36:49+00:00] INFO: yum_package[postgresql94] installing postgresql94-9.4.9-1PGDG.rhel6 from pgdg94 repository
==> centos_web: [2016-10-05T23:38:24+00:00] INFO: bash[install cherrypy] ran successfully
==> centos_web: [2016-10-05T23:38:24+00:00] INFO: yum_package[python-psycopg2.i686] installing python-psycopg2-2.6.2-1.rhel6.i686 from pgdg94 repository
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: bash[open port] ran successfully
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: template[/home/vagrant/server.py] created file /home/vagrant/server.py
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: template[/home/vagrant/server.py] updated file contents /home/vagrant/server.py
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: template[/home/vagrant/server.py] owner changed to 0
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: template[/home/vagrant/server.py] group changed to 10
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: template[/home/vagrant/server.py] mode changed to 644
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: cookbook_file[/etc/init/micro_service.conf] created file /etc/init/micro_service.conf
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: cookbook_file[/etc/init/micro_service.conf] updated file contents /etc/init/micro_service.conf
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: cookbook_file[/etc/init/micro_service.conf] owner changed to 0
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: cookbook_file[/etc/init/micro_service.conf] group changed to 10
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: cookbook_file[/etc/init/micro_service.conf] mode changed to 644
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: bash[server execute] ran successfully
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: Chef Run complete in 150.703843149 seconds
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: Running report handlers
==> centos_web: [2016-10-05T23:38:34+00:00] INFO: Report handlers complete
==> centos_web: [2016-10-05T23:36:01+00:00] INFO: Forking chef instance to converge...
distribuidos@307C:~/Escritorio/Parcials
Mozilla Firefox
http://192.168.131.52/
192.168.131.52
Search
This are database values id_divice: 15 name_device: express