Practica Balanceo de Carga
Por Prof. Oscar H. Mondragón

1. Objetivos

- Comprender el funcionamiento del Balanceador de Carga HAProxy
- Construir un balanceador de carga usando HAProxy y Linux Containers LXD/LXC

2. Herramientas a utilizar

- Vagrant
- VirtualBox
- HAProxy
- Linux Containers LXD/LXC

3. Introducción

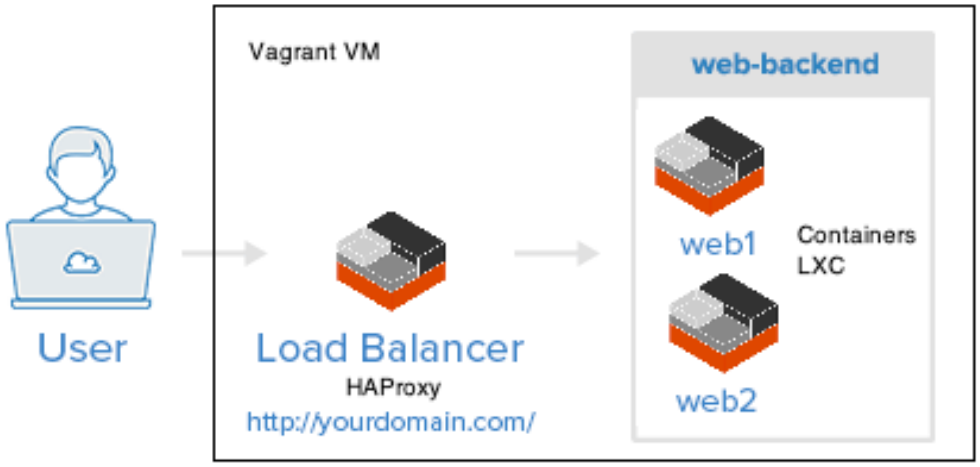
En esta práctica implementaremos balanceo de carga con la ayuda de HAProxy (www.haproxy.org/). Enviaremos una petición al balanceador de carga HAProxy y obtendremos respuestas desde dos servidores web.

Las peticiones no se realizan directamente a los servidores web, sino que el balanceador de carga decidirá que servidor será el encargado de procesar la petición.

Los dos servidores tendrán solo apache corriendo y en el balanceador de carga se ejecutará HAProxy.

La GUI del balanceador de carga será accesible desde la máquina anfitriona para visualizar el estado y estadísticas detalladas de los servidores web.

Se plantea implementar un balanceador de carga usando contenedores LXD/LXC y HAProxy como se muestra en la figura. El balanceador estará compuesto por un contenedor corriendo HAProxy y dos contenedores de backend corriendo apache. Los tres contenedores corren sobre una maquina virtual de Ubuntu en virtualbox y administrada por Vagrant.



4. Desarrollo de la práctica

4.1. Configuración de Vagrant

Esta práctica la desarrollaremos usando boxes de Ubuntu 20.04 en Vagrant. El Vagrantfile que usaremos es el siguiente (el mismo con el que venimos trabajando):

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

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

  if Vagrant.has_plugin? "vagrant-vbguest"
    config.vbguest.no_install  = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote   = true
  end

  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
```

4.2. Instalación de LXD

Si aun no ha instalado LXD en la maquina virtual ServidorUbuntu, realice los siguientes pasos:

Instalar LXD

La manera mas simple de probar LXD es usando su herramienta de línea de comandos.

Abra usa sesión SSH con servidorUbuntu

```
vagrant ssh servidorUbuntu
```

Para instalar LXD en Ubuntu ejecute (puede tardar varios minutos):

```
sudo apt-get install lxd -y
```

Luego ejecute el siguiente comando para loguearse en el nuevo grupo creado:

```
newgrp lxd
```

Inicializar lxd

```
lxd init --auto
```

4.3. Instalación de servidores de backend corriendo apache

Lance dos contenedores llamados web1 y web2

```
vagrant@servidorUbuntu:~$ lxc launch ubuntu:18.04 web1
vagrant@servidorUbuntu:~$ lxc launch ubuntu:18.04 web2
```

Siga los siguientes pasos para instalar un servidor apache en web1:

1. Ingresar al Shell de web1 y ejecutar

```
vagrant@servidorUbuntu:~$ lxc exec web1 /bin/bash
```

2. Ejecutar

```
root@web1:~# apt update && apt upgrade
root@web1:~# apt install apache2
root@web1:~# systemctl enable apache2
```

3. Incluir en el contenido del index.html algo como:

```
Hello from web1
```

4. Iniciar el servidor Apache

```
root@web1:~# vim /var/www/html/index.html
root@web1:~# systemctl start apache2
```

5. Probar el servidor apache

```
vagrant@servidorUbuntu:~$ curl <ip web1>
```

Para verificar la ip puede ejecutar `lxc list`

Repetir los pasos 1 a 5 para el contenedor web2, incluyendo en el contenido del index.html algo como:

```
Hello from web2
```

y actualizando la ip del servidor apropiadamente para la prueba.

```
vagrant@servidorUbuntu:~$ curl <ip web2>
```

4.4. Configurar contenedor HAProxy

Lance un contenedor llamado haproxy

```
vagrant@servidorUbuntu:~$ lxc launch ubuntu:18.04 haproxy
```

Siga los siguientes pasos para instalar un servidor apache en haproxy:

1. Ingresar al Shell de web1 y ejecutar

```
vagrant@servidorUbuntu:~$ lxc exec haproxy /bin/bash
```

2. Ejecutar

```
root@haproxy:~# apt update && apt upgrade
root@haproxy:~# apt install haproxy
root@haproxy:~# systemctl enable haproxy
```

3. Modifique el archivo haproxy.cfg

```
root@haproxy:~# vim /etc/haproxy/haproxy.cfg
```

Cree un frontend y backend. Agruegue (puede descargar el haproxy.cfg de classroom) las siguientes lineas al final del archivo haproxy.cfg.
Actualice las IPs apropiadamente.

```
backend web-backend
   balance roundrobin
   stats enable
   stats auth admin:admin
   stats uri /haproxy?stats

   server web1 240.101.0.172:80 check
   server web2 240.102.0.58:80 check

frontend http
  bind *:80
  default_backend web-backend
```

4. Iniciar el haproxy

```
root@haproxy:~# systemctl start haproxy
```

5. Probar

   Ejecute varias veces

```
root@haproxy:~# curl <ip haproxy>
```

   La respuesta cambiará cada vez que haga peticiones al balanceador de carga ya que este las distribuirá de manera equilibrada entre los servidores web.

   Verifíquelo.

4.5. Probar desde la red local de la maquina Vagrant

Para probar desde afuera, hacer forwarding de puertos:

```
vagrant@servidorUbuntu:~$ lxc config device add haproxy http
proxy listen=tcp:0.0.0.0:80 connect=tcp:127.0.0.1:80
```

1. Abra el navegador usando la ip de la maquina Vagrant y actualice varias veces

   Desde el navegador del anfitrión ejecute http://192.168.100.30/

   La respuesta cambiará cada vez que haga peticiones al balanceador de carga ya que este las distribuirá de manera equilibrada entre los servidores web.

   Verifíquelo.

2. Apague el contenedor de backend web2

   En este caso todas las peticiones se deben dirigir a webserver1 ya que es el único disponible.

3. Apague los dos contenedores de backend: web1 y web2

En este caso ambos servidores web estarán apagados por lo que el balanceador de carga mostrará un mensaje de error. Este error es configurable en HAProxy,

La respuesta será algo como:

```
503 Service Unavailable

No server is available to handle this request.
```

4.6. Visualización de Estadísticas

HAProxy ofrece un sitio web para visualizacion de estadisticas.

Para ver las estadísticas de HAProxy, abrir:

http://192.168.100.30/haproxy?stats

login: admin
password: admin

Ejemplos de salida

HAProxy version 1.4.24, released 2013/06/17

Statistics Report for pid 1999

> General process information

pid = 1999 (process #1, nbproc = 1)
uptime = 0d 0h00m06s
system limits: memmax = unlimited; ulimit-n = 4013
maxsock = 4013; **maxconn** = 2000; **maxpipes** = 0
current conns = 1; current pipes = 0/0
Running tasks: 1/3

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
Note: UP with load-balancing disabled is reported as "NOLB".

http-in

	Queue			Session rate			Sessions					Bytes		Denied		Errors		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Res
Frontend				0	1	-	1	1	2 000	1		409	415	0	0	0		

webservers

	Queue			Session rate			Sessions					Bytes		Denied		Errors			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Re
webserver1	0	0	-	0	1		0	1	-	1	1	409	415		0		0	0	
webserver2	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	
Backend	0	0		1	1		1	1	0	2	1	409	415	0	0		0	0	

HAProxy version 1.4.24, released 2013/06/17

Statistics Report for pid 1058

> General process information

pid = 1058 (process #1, nbproc = 1)
uptime = 0d 0h09m57s
system limits: memmax = unlimited; ulimit-n = 4013
maxsock = 4013; **maxconn** = 2000; **maxpipes** = 0
current conns = 1; current pipes = 0/0
Running tasks: 1/3

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
Note: UP with load-balancing disabled is reported as "NOLB".

http-in

	Queue			Session rate			Sessions					Bytes		Denied		Errors		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp
Frontend				1	1	-	1	1	2 000	2		998	830	0	0	0		

webservers

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warn
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr
webserver1	0	0	-	0	1		0	1	-	1	1	499	415		0		0	0	0
webserver2	0	0	-	0	1		0	1	-	1	1	499	415		0		0	0	0
Backend	0	0		1	1		1	1	0	3	2	998	830	0	0		0	0	0

HAProxy version 1.4.24, released 2013/06/17

Statistics Report for pid 1058

> General process information

pid = 1058 (process #1, nbproc = 1)
uptime = 0d 0h10m12s
system limits: memmax = unlimited; ulimit-n = 4013
maxsock = 4013; **maxconn** = 2000; **maxpipes** = 0
current conns = 1; current pipes = 0/0
Running tasks: 1/3

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
Note: UP with load-balancing disabled is reported as "NOLB".

http-in

	Queue			Session rate			Sessions					Bytes		Denied		Errors		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp
Frontend				1	1	-	1	1	2 000	3		1 420	10 403	0	0	0		

webservers

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Wa
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr
webserver1	0	0	-	0	1		0	1	-	1	1	499	415		0		0	0	0
webserver2	0	0	-	0	1		0	1	-	1	1	499	415		0		0	0	0
Backend	0	0		1	1		1	1	0	4	2	1 420	10 403	0	0		0	0	0

4.7. Pruebas de Carga

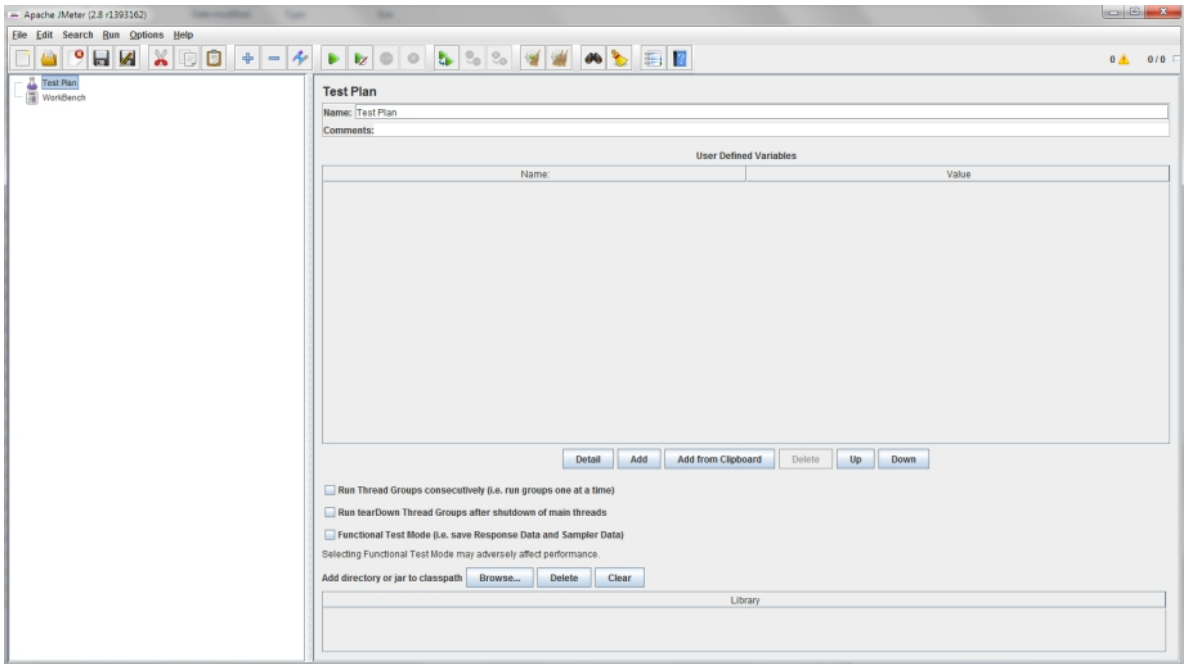
Para las pruebas de carga usaremos JMeter. Una aplicación de fuente abierta para pruebas de desempeño. Permite probar servidores web, así como otros servicios.

Con estas pruebas podemos verificar el numero de usuarios concurrentes que la aplicación soportará y que carga hará que la aplicación deje de funcionar.

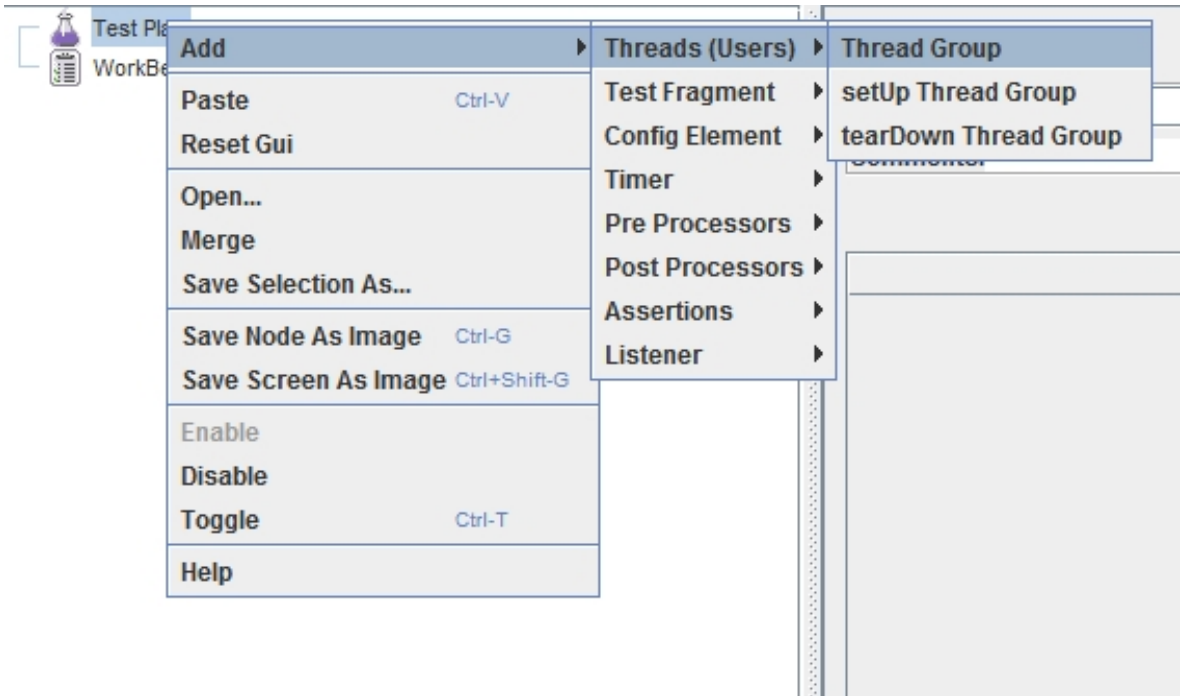
Instale JMeter en su maquina anfitrión desde

https://jmeter.apache.org/download_jmeter.cgi

Una vez instalado ejecútelo. Al ejecutar jmeter, visualizará la siguiente interfase:



Cree un grupo de usuarios



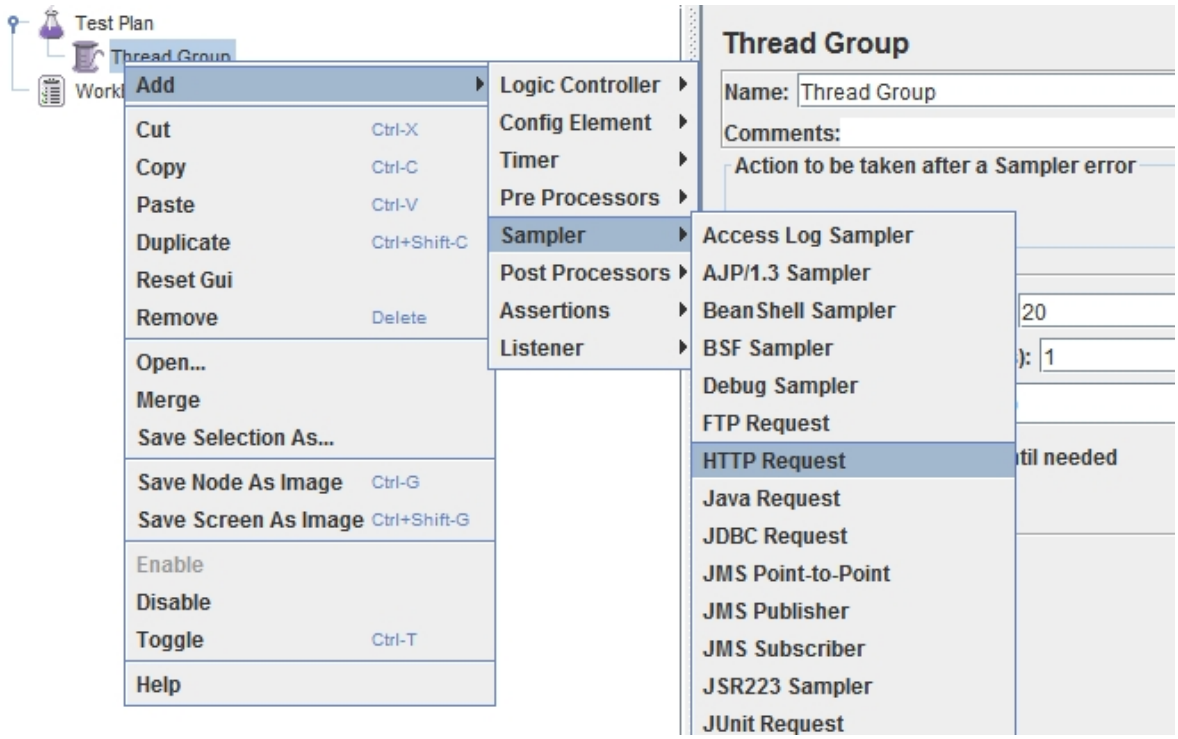
Configure los siguientes parámetros

Number of threads (usuarios): 2000
Rump-up period (tiempo total en el que ejecutara las peticiones): 0.5 segundos

Loop count (numero de veces en que se ejecutará el test): 50

Agregue un Sampler (Peticion HTTP)

En este caso se agregará una petición http.

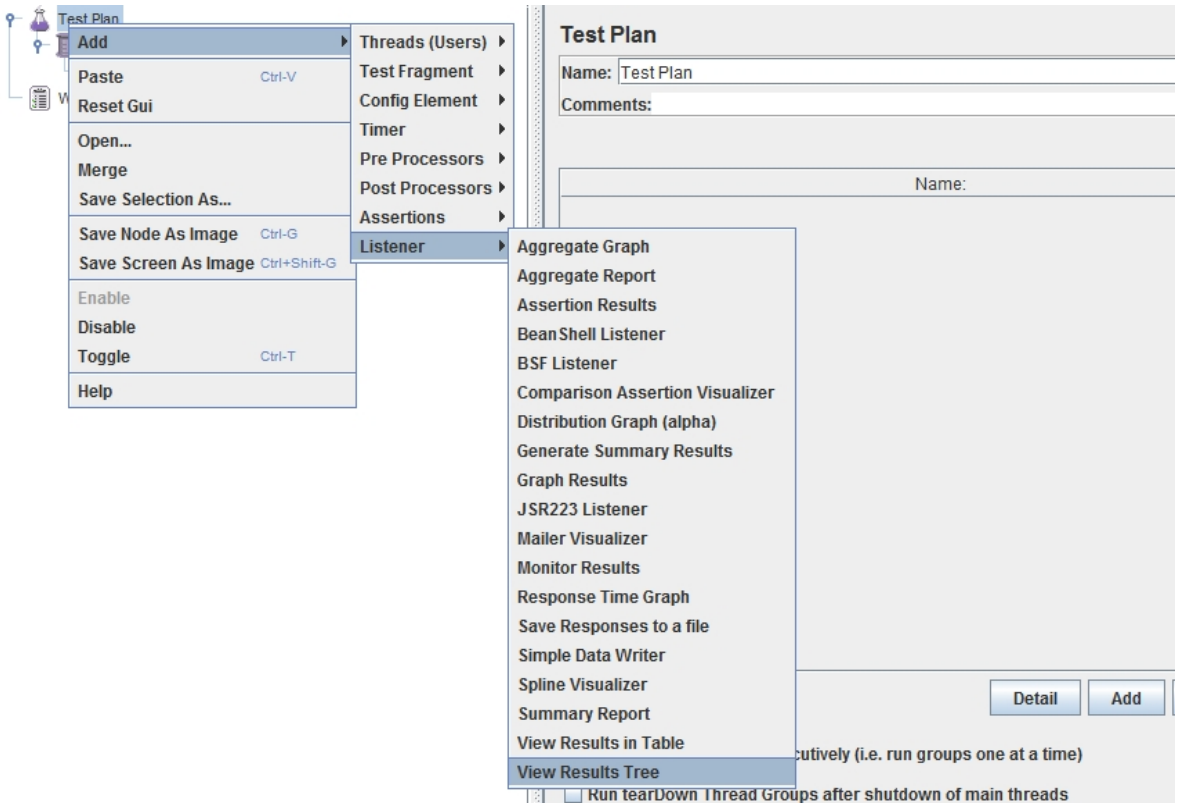


Una vez agregue la petición http configure únicamente la opción “Server Name or IP”.

En nuestro caso la IP es la del balanceador de carga (maquina Vagrant): 192.168.100.30.

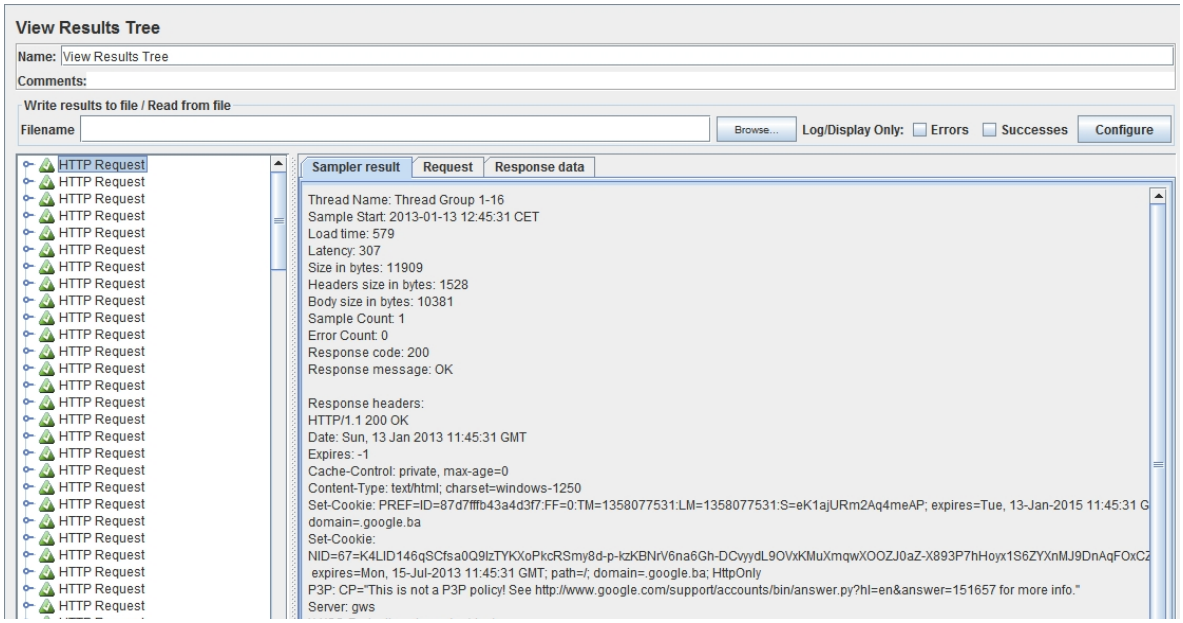
Agregar un Listener

En este punto nuestro test ya está listo, solo debe agregar un listener para ver los resultados:



Los resultados serán mostrados con todos los detalles de peticiones realizadas y respuestas obtenidas.

Puede mirar los resultados en tiempo real si va a la opción “View Results Tree”.



5. *Ejercicio*

1. Configure un balanceador de carga con un frontend corriendo HAProxy y dos Backends como se describe en el punto 3.

2. Realice pruebas de carga variando los parámetros de las pruebas y analice que sucede

3. Agregue diferentes números de servidores web (contenedores de backend) y verifique como responde el sistema

4. Explore diferentes opciones de Listeners

5. Asegurese de entender la información estadística arrojada por HAProxy: Queue, Sesion rate, Sessions, Bytes, etc

6. *Desafío [Hasta 0.5 Puntos en una nota del corte de tareas/quices/exposiciones]*

1. Realice una implementacion similar a la propuesta en la practica, pero esta vez corriendo HAProxy y los backends directamente en maquinas Vagrant. Sin usar contenedores.

7. *Entregables y Evaluación*

- Sustentación de la práctica

8. *Bibliografía*

- HAProxy, http://www.haproxy.org/
- How to Install JMeter on Windows. https://octoperf.com/blog/2018/04/16/how-to-install-jmeter-windows/
- HTTP Load Testing with JMeter. http://www.techbar.me/http-load-testing-with-jmeter/
- Using LXC/LXD Containers with HAProxy: https://autoize.com/lxc-lxd-containers-with-haproxy/
- An introducction to HAProxy and Load Balancing Concepts: https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts