

Ingeniería de Servidores (2014-2015)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Memoria Práctica 4

Pablo Vílchez García

10 de diciembre de 2014

Índice

1. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles? 3
2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ?
¿y -n 1000? 4
3. Ejecute ab contra a las tres máquinas virtuales (desde el SO anfitrión a las máquinas virtuales de la red local) una a una (arrancadas por separado) y muestre las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina? 5
4. Instale y siga el tutorial en <http://jmeter.apache.org/usermanual/build-web-test-plan.html> realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando alguna de sus máquinas virtuales (Puede hacer una página sencilla, usar las páginas de phpmyadmin, instalar un CMS, etc.). 10
5. Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir: 1) Objetivo del benchmark 2) Métricas (unidades, variables, puntuaciones, etc.) 3) Instrucciones para su uso 4) Ejemplo de uso analizando los resultados Tenga en cuenta que puede comparar varios gestores de BD, lenguajes de programación web (tiempos de ejecución, gestión de memoria, ...), duración de la batería, etc. 14

1. Instale la aplicación. ¿Qué comando permite listar los benchmarks disponibles?

Vamos a la página web de phoronix y en descargas, obtenemos el paquete `phoronix-test-suite_5.2.1_all.deb`¹ y lo instalamos² como se muestra en la siguiente imagen 1

```
vilchez@vilchez-K52Dr:~$ sudo dpkg -i Escritorio/phoronix-test-suite_5.2.1_all.deb
[sudo] password for vilchez:
Seleccionando el paquete phoronix-test-suite previamente no seleccionado.
(Leyendo la base de datos ... 425525 ficheros o directorios instalados actualmente.)
Desempaquetando phoronix-test-suite (de .../phoronix-test-suite_5.2.1_all.deb) ...
Configurando phoronix-test-suite (5.2.1) ...
Procesando disparadores para man-db ...
Procesando disparadores para hicolor-icon-theme ...
Procesando disparadores para bamfdaemon ...
Rebuilding /usr/share/applications/bamf.index...
Procesando disparadores para desktop-file-utils ...
Procesando disparadores para gnome-menus ...
vilchez@vilchez-K52Dr:~$
```

Figura 1.1: Instalación de Phoronix

A continuación, ejecutamos el comando `phoronix-test-suite list-available-test` que nos muestra los benchmarks disponibles

```
vilchez@vilchez-K52Dr:~$ phoronix-test-suite list-available-test
PHP Deprecated: Comments starting with '#' are deprecated in /etc/php5/cli/conf.d/ming.ini on line 1 in
Unknown on line 0

Phoronix Test Suite v5.2.1 (Khanino)

The Phoronix Test Suite is the most comprehensive testing and benchmarking platform available for Linux,
Solaris, Mac OS X, and BSD operating systems. The Phoronix Test Suite allows for carrying out tests in a
fully automated manner from test installation to execution and reporting. All tests are meant to be easil
y reproducible, easy-to-use, and support fully automated execution. The Phoronix Test Suite is open-sourc
e under the GNU GPLv3 license and is developed by Phoronix Media in cooperation with partners.

View the included PDF / HTML documentation or visit http://www.phoronix-test-suite.com/ for full details.

TEST INSTALLATION

install [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
install-dependencies [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
make-download-cache
remove-installed-test [Test]

TESTING

auto-compare
benchmark [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
finish-run [Test Result]
run [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
run-random-tests
run-tests-in-suite

BATCH TESTING

batch-benchmark [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
batch-install [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
batch-run [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
batch-setup
default-benchmark [Test | Suite | OpenBenchmarking.org ID | Test Result] ...
```

Figura 1.2: Comando para listar benchmark disponibles

¹<http://phoronix-test-suite.com/?k=downloads>

²<https://wiki.ubuntu.com/PhoronixTestSuite>

2. De los parámetros que le podemos pasar al comando ¿Qué significa -c 30 ? ¿y -n 1000?

El parámetro -c 30 quiere decir que se van a hacer 30 peticiones concurrentes.

El parámetro -n 1000 quiere decir que se van a hacer 1000 peticiones.³ A continuación se muestra una ejecución de apache benchmark22

```
vilchez@vilchez-K52Dr:~$ ab -c30 -n1000 http://localhost/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:      Apache/2.2.22
Server Hostname:      localhost
Server Port:          80

Document Path:        /
Document Length:      177 bytes

Concurrency Level:    30
Time taken for tests:  0.306 seconds
Complete requests:    1000
Failed requests:       0
Write errors:          0
Total transferred:    454000 bytes
HTML transferred:     177000 bytes
Requests per second:  3268.53 [#/sec] (mean)
Time per request:     9.178 [ms] (mean)
Time per request:     0.306 [ms] (mean, across all concurrent requests)
Transfer rate:        1449.13 [Kbytes/sec] received
```

Figura 2.1: Ejecución de ab -c30 -n1000

³<http://blog.diacode.com/testeando-el-rendimiento-de-tu-aplicacion-con-apache-bench>

Connection Times (ms)				
	min	mean[+/-sd]	median	max
Connect:	0	0 0.6	0	4
Processing:	2	9 1.7	9	13
Waiting:	1	8 1.5	7	12
Total:	5	9 1.6	9	15

Percentage of the requests served within a certain time (ms)	
50%	9
66%	9
75%	11
80%	11
90%	11
95%	12
98%	12
99%	13
100%	15 (longest request)

Figura 2.2: Ejecución de ab -c30 -n1000

3. Ejecute ab contra a las tres máquinas virtuales (desde el SO anfitrión a las máquina virtuales de la red local) una a una (arrancadas por separado) y muestre las estadísticas. ¿Cuál es la que proporciona mejores resultados? Fíjese en el número de bytes transferidos, ¿es igual para cada máquina?

Vemos la ip de la máquina a la que vamos a hacer ab 3

```
eth2      Link encap:Ethernet  dirección
          Direc. inet:192.168.56.101  D
          Dirección inet6: fe80::a00:27
          ACTIVO DIFUSIÓN FUNCIONANDO M
          Paquetes RX:64627 errores:0 p
```

Figura 3.1: ifconfig en la primera VM

En las siguientes imágenes se muestra la ejecución de ab contra la primera VM33

```

vilchez@vilchez-K52Dr:~$ ab -c40 -n200 http://192.168.56.101/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.101 (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests


Server Software:        Apache/2.2.22
Server Hostname:        192.168.56.101
Server Port:            80

Document Path:          /
Document Length:        177 bytes

Concurrency Level:      40
Time taken for tests:    0.208 seconds
Complete requests:      200
Failed requests:        0
Write errors:           0
Total transferred:      90600 bytes
HTML transferred:       35400 bytes
Requests per second:    960.61 [#/sec] (mean)
Time per request:       41.640 [ms] (mean)
Time per request:       1.041 [ms] (mean, across all concurrent requests)
Transfer rate:          424.96 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      1   1.2      0      6
Processing:      1     38  43.2     26    198
Waiting:        1     28  27.8     23    190
Total:          1     39  43.3     28    201

```

Figura 3.2: Ejecución de `ab -c40 -n200 http://192.168.56.101`

```

Percentage of the requests served within a certain time (ms)
 50%      28
 66%      36
 75%      41
 80%      45
 90%      92
 95%     159
 98%     195
 99%     199
100%     201 (longest request)

```

Figura 3.3: Ejecución de `ab -c40 -n200 http://192.168.56.101`

Vemos la ip de la siguiente máquina a la que vamos a hacer ab 3

```
eth1      Link encap:Ethernet  direcciónHW
          Direc. inet:192.168.56.102  Difu
          Dirección inet6: fe80::a00:27ff:
          ACTIVO DIFUSIÓN FUNCIONANDO MULT
          Paquetes RX:2 errores:0 perdidos
```

Figura 3.4: ifconfig en la segunda VM

En las siguientes imágenes se muestra la ejecución de ab contra la segunda VM33

```
vilchez@vilchez-K52Dr:~$ ab -c40 -n200 http://192.168.56.102/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.102 (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests

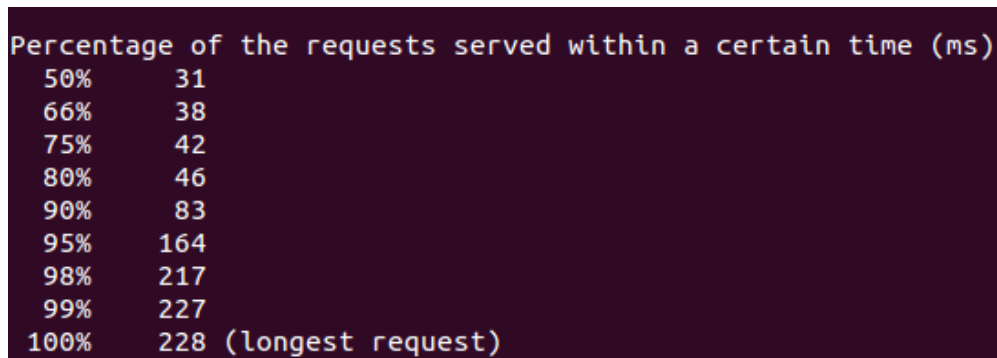
Server Software:      Apache/2.2.22
Server Hostname:      192.168.56.102
Server Port:          80

Document Path:        /
Document Length:      177 bytes

Concurrency Level:     40
Time taken for tests:  0.235 seconds
Complete requests:     200
Failed requests:        0
Write errors:           0
Total transferred:     90600 bytes
HTML transferred:     35400 bytes
Requests per second:   851.62 [#/sec] (mean)
Time per request:      46.969 [ms] (mean)
Time per request:      1.174 [ms] (mean, across all concurrent requests)
Transfer rate:         376.74 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      1   2.1      0      7
Processing:      5     43  44.9     30    223
Waiting:        4     29  18.4     27    163
Total:         10     44  45.3     31    228
```

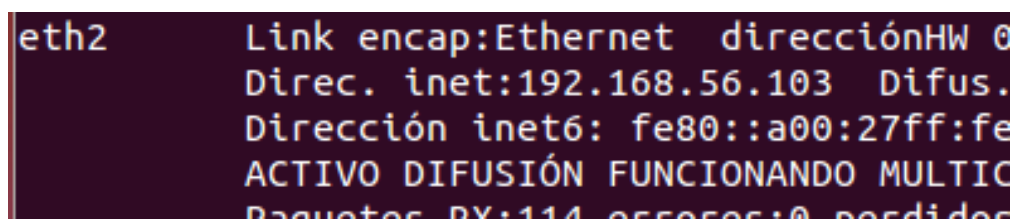
Figura 3.5: Ejecución de ab -c40 -n200 http://192.168.56.102



Percentage of the requests served within a certain time (ms)	
50%	31
66%	38
75%	42
80%	46
90%	83
95%	164
98%	217
99%	227
100%	228 (longest request)

Figura 3.6: Ejecución de `ab -c40 -n200 http://192.168.56.102`

Vemos la ip de la tercera máquina a la que vamos a hacer ab 3



```
eth2      Link encap:Ethernet  direcciónHW 0
          Direc. inet:192.168.56.103  Difus.
          Dirección inet6: fe80::a00:27ff:fe
          ACTIVO DIFUSIÓN FUNCIONANDO MULTIC
          Paquetes RX:114 0550505:0 perdidos
```

Figura 3.7: ifconfig en la tercera VM

En las siguientes imágenes se muestra la ejecución de ab contra la tercera VM33


```

vilchez@vilchez-K52Dr:~$ ab -c40 -n200 http://192.168.56.103/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.103 (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests


Server Software:      Apache/2.2.22
Server Hostname:      192.168.56.103
Server Port:          80

Document Path:        /
Document Length:      177 bytes

Concurrency Level:    40
Time taken for tests:  0.223 seconds
Complete requests:    200
Failed requests:       0
Write errors:          0
Total transferred:    90600 bytes
HTML transferred:     35400 bytes
Requests per second:  898.08 [#/sec] (mean)
Time per request:     44.540 [ms] (mean)
Time per request:     1.113 [ms] (mean, across all concurrent requests)
Transfer rate:        397.29 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      1   1.1      0      4
Processing:      1    40  33.6     32    212
Waiting:         1    24  15.0     23     75
Total:          1    41  33.7     32    215

```

Figura 3.8: Ejecución de `ab -c40 -n200 http://192.168.56.103`

```

Percentage of the requests served within a certain time (ms)
 50%      32
 66%      46
 75%      58
 80%      66
 90%      83
 95%      96
 98%     139
 99%     177
100%     215 (longest request)

```

Figura 3.9: Ejecución de `ab -c40 -n200 http://192.168.56.103`

Una vez recogidos los datos, hacemos una comparativa de transferencia de datos, veloci-

dad y bytes transferidos 3

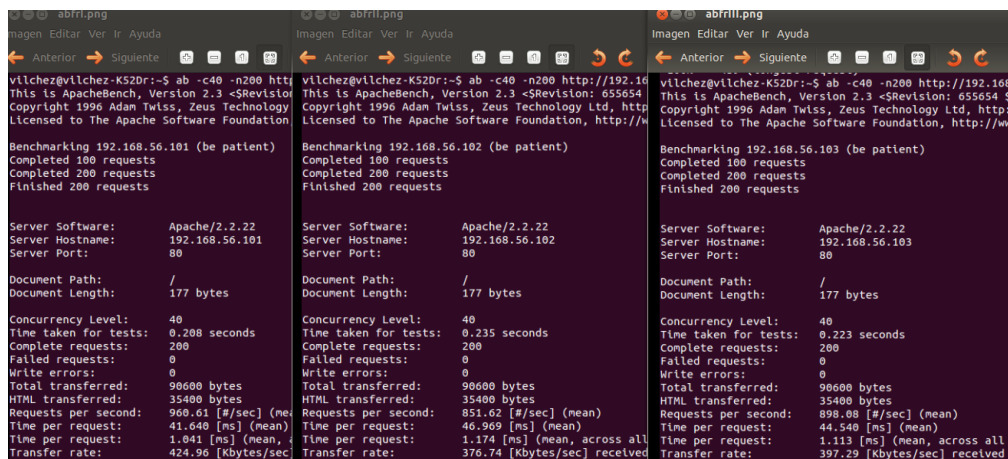


Figura 3.10: Comparamos los datos obtenidos

La comparativa muestra que los bytes transferidos son los mismos, ya que de todas las VMs se ha obtenido la misma web, por eso todas tienen el mismo tamaño. La máquina que ha obtenido mejor resultado en la transferencia ha sido la segunda, que ha obtenido una tasa de transferencia de 376.74KB/s, sin embargo, es la que más tiempo se ha tomado haciendo el test, 0.235 segundos.

4. Instale y siga el tutorial en <http://jmeter.apache.org/usermanual/build-web-test-plan.html> realizando capturas de pantalla y comentándolas. En vez de usar la web de jmeter, haga el experimento usando alguna de sus máquinas virtuales (Puede hacer una página sencilla, usar las páginas de phpmyadmin, instalar un CMS, etc.).

Descargamos Jmeter⁴ lo descomprimos, y en bin/ está el ejecutable .jar de Jmeter. Lo ejecutamos como se muestra a continuación⁴.

⁴http://jmeter.apache.org/download_jmeter.cgi

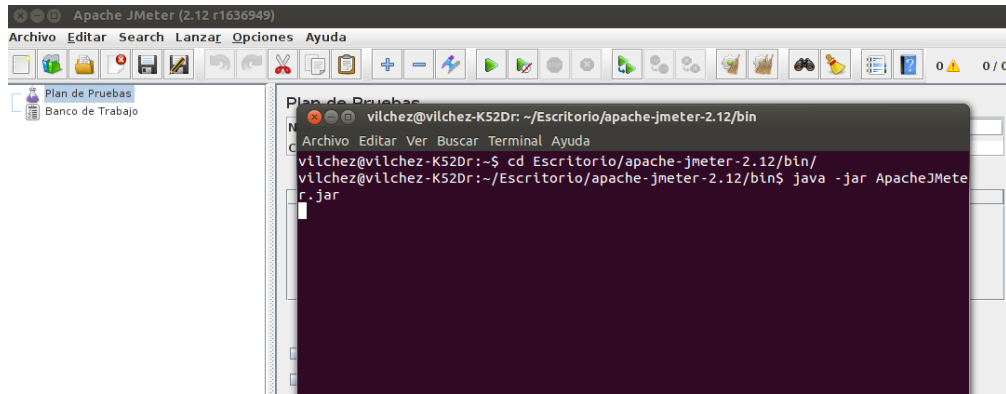


Figura 4.1: Ejecución de Jmeter después de descargarlo

Le damos a crear nuevo grupo de hilos. Ponemos 5 hilos, 1 segundo y 2 iteraciones. Se muestra en la siguiente figura4.

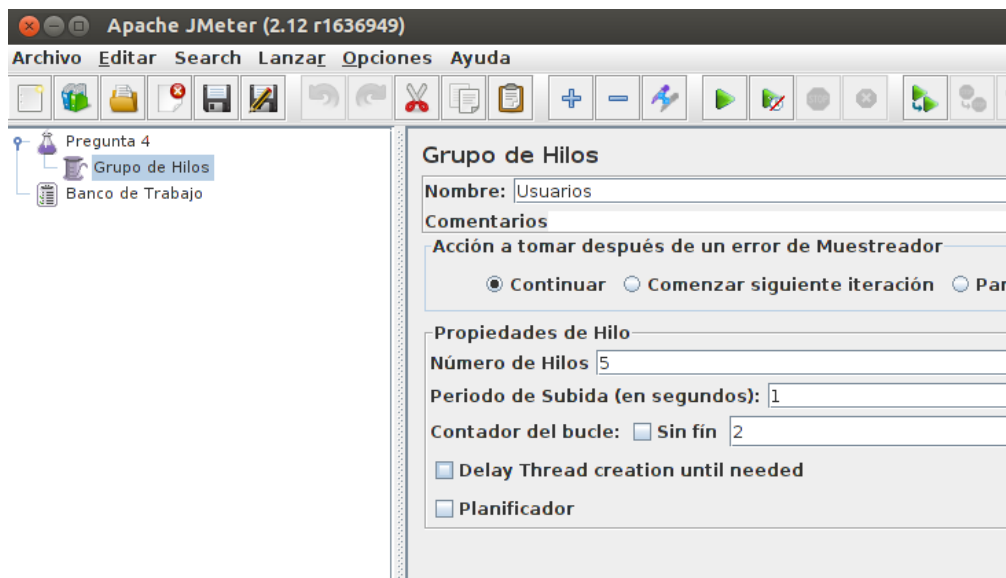


Figura 4.2: Añadimos hilos y los configuramos

Añadimos la configuración por defecto para las peticiones HTTP como se muestra en la siguiente imagen4, añadiendo la ip 192.168.56.101 de la VM.

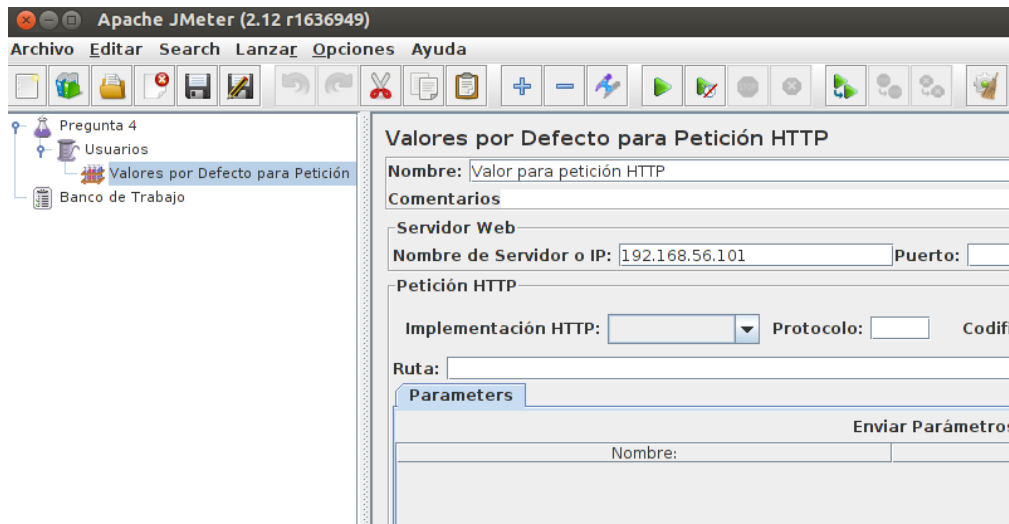


Figura 4.3: Configurar peticiones HTTP

Añadimos la configuración de la petición HTTP poniendo index.html como se muestra en la imagen4

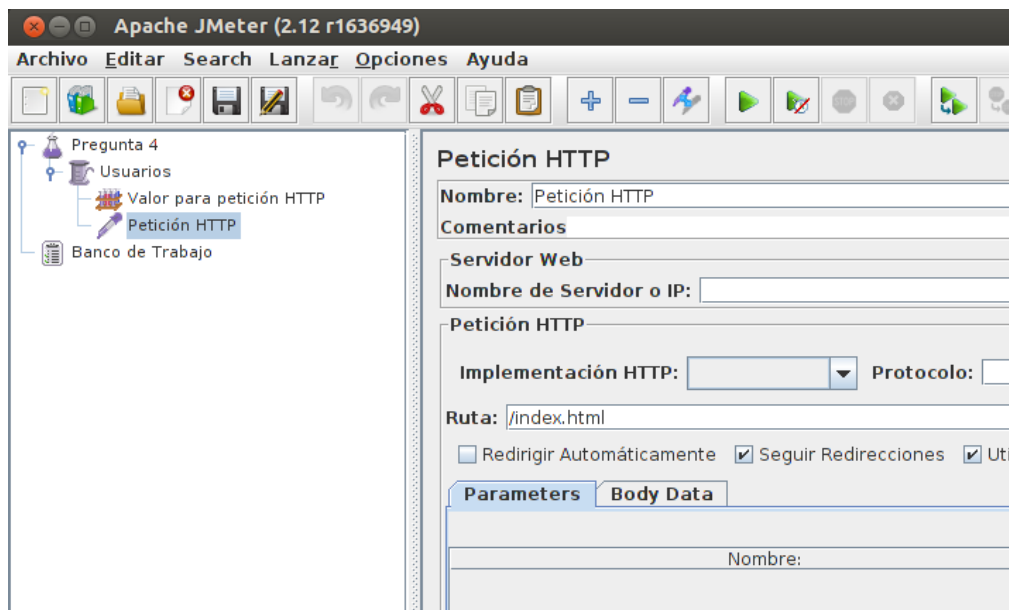


Figura 4.4: Configurar petición HTTP

Por último, para ejecutarlo y visualizar los datos, añadimos un receptor y dándole al que pone ver resultados en árbol. Después le damos al play verde del panel de opciones. En la siguiente imagen se muestra el resultado4.

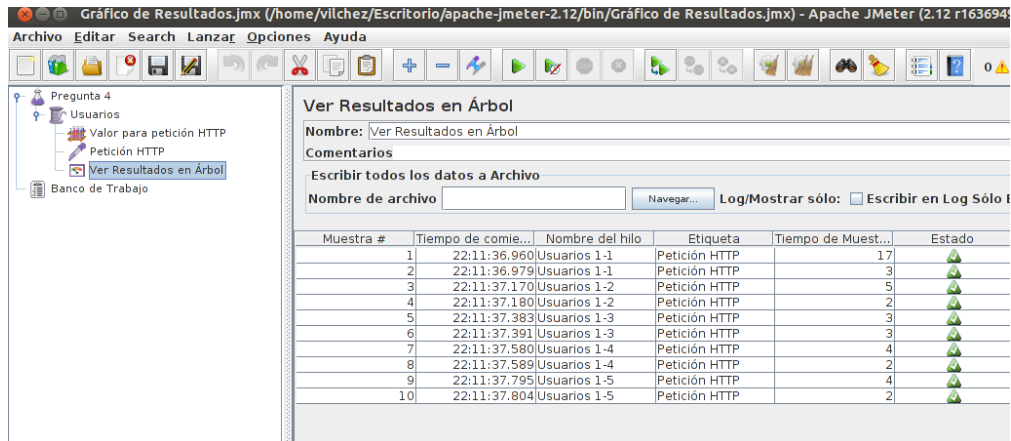


Figura 4.5: Ejecutamos test

Para ver que realmente estaba funcionando, he cerrado la máquina virtual, y he vuelto a ejecutar el test, donde nos damos cuenta que después de un tiempo de espera da resultados negativos el test4.

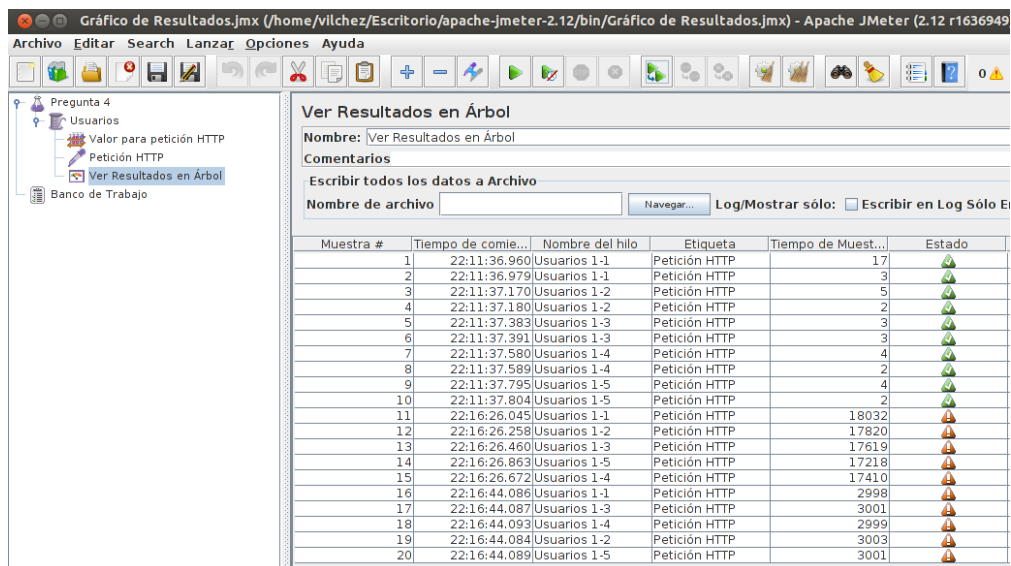


Figura 4.6: Ejecutamos test de comprobación

Como habíamos configurado, se hacen 5 hilos con dos intentos cada uno, dando un total de 10 intentos.

5. Programe un benchmark usando el lenguaje que desee. El benchmark debe incluir: 1) Objetivo del benchmark 2) Métricas (unidades, variables, puntuaciones, etc.) 3) Instrucciones para su uso 4) Ejemplo de uso analizando los resultados Tenga en cuenta que puede comparar varios gestores de BD, lenguajes de programación web (tiempos de ejecución, gestión de memoria, ...), duración de la batería, etc.

- El objetivo de este benchmark es comparar el tiempo de consulta en dos bases de datos exactamente iguales pero una en mongo y otra en mysql.
- La unidad de medida que vamos a utilizar es el tiempo.
- Para usarlo, ejecutar script.sh con el comando `sh script.sh` y tener los archivos `mongo.js` `mongoini.js` `mongofin.js` `mysql.sql` `mysqlini.sql` y `mysqlfin.sql` en el mismo directorio, además de tener instalado en el sistema mysql y mongo
- En el siguiente ejemplo se muestran los archivos y el script necesarios para realizar el benchmark, además de una prueba mostrando los resultados.

Antes de comenzar, es necesario tener MySQL y Mongo en el sistema. Para la instalación de MySQL ejecutar en la terminal⁵:

- `sudo apt-get install mysql-server`

. Para la instalación de Mongo ejecutar en la terminal⁶:

- `sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10`
- `echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list`
- `sudo apt-get update`
- `sudo apt-get install -y mongodb-org`

Hacemos un script shell para ejecutar los comandos de mysql⁷ y los comandos mongo⁸⁹ necesarios para crear la base de datos y hacer las consultas. A continuación se muestra el script con la medición de tiempos¹⁰¹¹ de las ejecuciones⁵.

⁵<https://help.ubuntu.com/12.04/serverguide/mysql.html>

⁶<http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/>

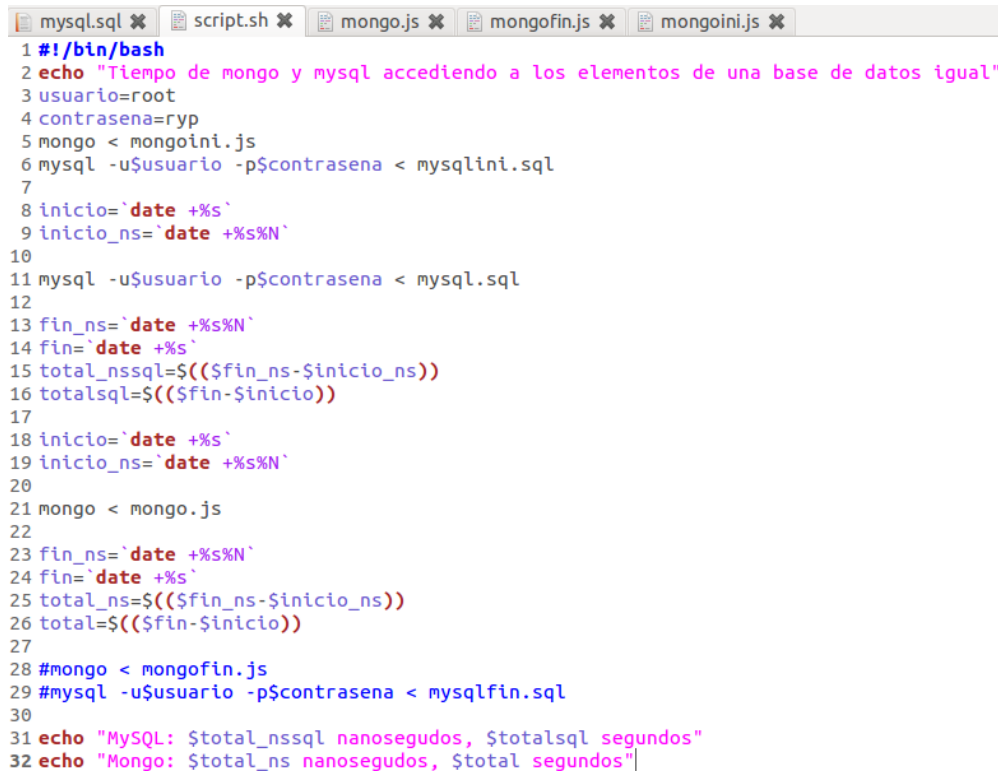
⁷<http://www.shellhacks.com/en/HowTo-Execute-a-MySQL-Command-from-a-Linux-BASH-Shell>

⁸<http://stackoverflow.com/questions/4837673/how-to-execute-mongo-commands-through-shell-scripts>

⁹<http://blog.manueltviera.es/empezando-con-mongodb/>

¹⁰<http://egomez.es/2008/01/31/calcular-tiempo-de-ejecucion-de-un-comando-en-bash-shell/>

¹¹<http://www.the-evangelist.info/2009/12/bash-operaciones-aritmeticas/>



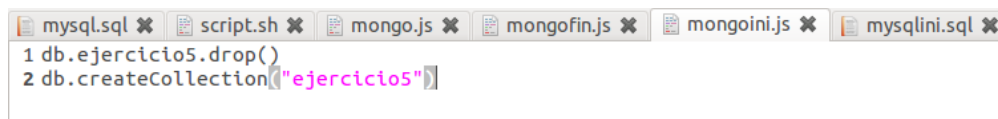
```

mysql.sql ✕ script.sh ✕ mongo.js ✕ mongofin.js ✕ mongoini.js ✕
1 #!/bin/bash
2 echo "Tiempo de mongo y mysql accediendo a los elementos de una base de datos igual"
3 usuario=root
4 contrasena=ryp
5 mongo < mongoini.js
6 mysql -u$usuario -p$contrasena < mysqlini.sql
7
8 inicio=`date +%s`
9 inicio_ns=`date +%s%N`
10
11 mysql -u$usuario -p$contrasena < mysql.sql
12
13 fin_ns=`date +%s%N`
14 fin=`date +%s`
15 total_nssql=$(( $fin_ns - $inicio_ns ))
16 totalsql=$(( $fin - $inicio ))
17
18 inicio=`date +%s`
19 inicio_ns=`date +%s%N`
20
21 mongo < mongo.js
22
23 fin_ns=`date +%s%N`
24 fin=`date +%s`
25 total_ns=$(( $fin_ns - $inicio_ns ))
26 total=$(( $fin - $inicio ))
27
28 #mongo < mongofin.js
29 #mysql -u$usuario -p$contrasena < mysqlfin.sql
30
31 echo "MySQL: $total_nssql nanosegundos, $totalsql segundos"
32 echo "Mongo: $total_ns nanosegundos, $total segundos"

```

Figura 5.1: Script para ejecución del Benchmark

El script ejecuta la inicialización de la base de datos de mongo y de mysql, con los archivos mongoini.js5 y mysqlini.sql5.

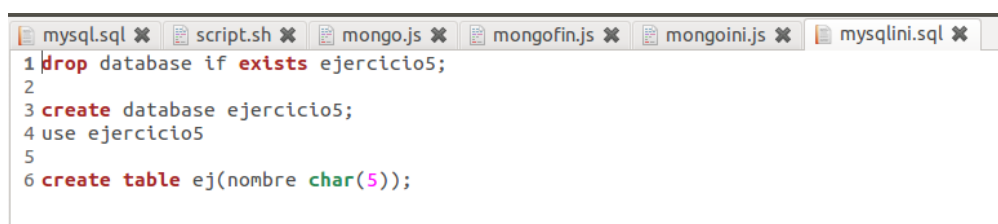


```

mysql.sql ✕ script.sh ✕ mongo.js ✕ mongofin.js ✕ mongoini.js ✕ mysqlini.sql ✕
1 db.ejercicio5.drop()
2 db.createCollection("ejercicio5")

```

Figura 5.2: Archivo de comandos de inicialización de la BD de mongo



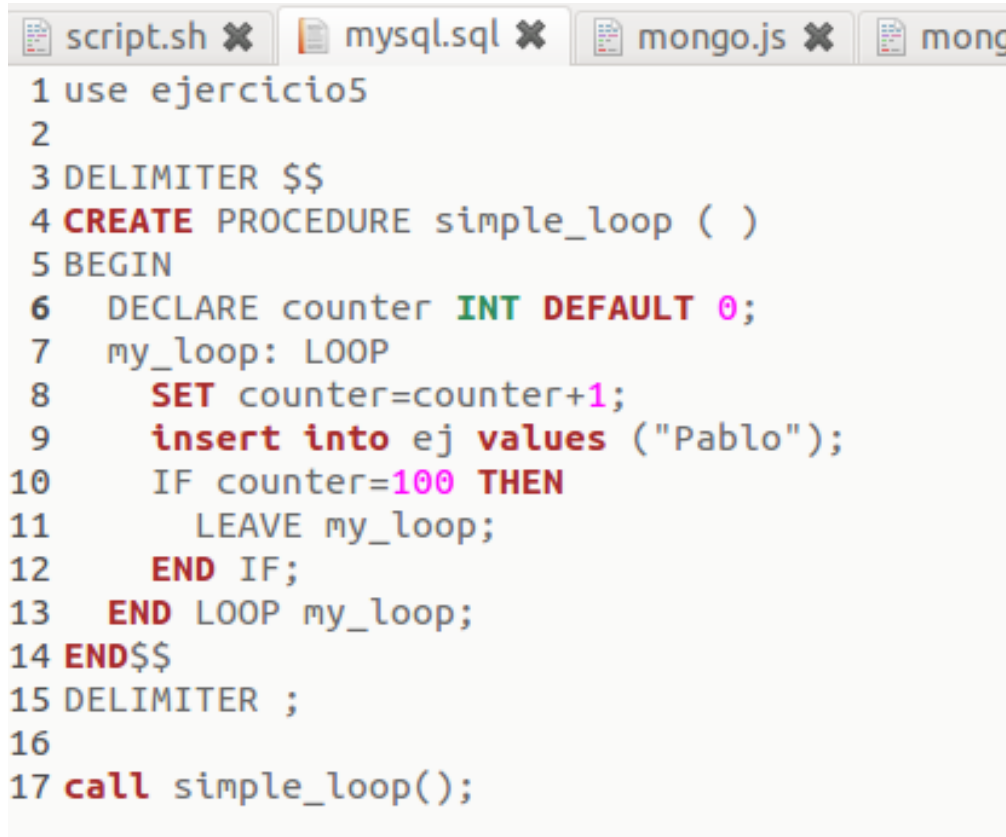
```

mysql.sql ✕ script.sh ✕ mongo.js ✕ mongofin.js ✕ mongoini.js ✕ mysqlini.sql ✕
1 drop database if exists ejercicio5;
2
3 create database ejercicio5;
4 use ejercicio5
5
6 create table ej(nombre char(5));

```

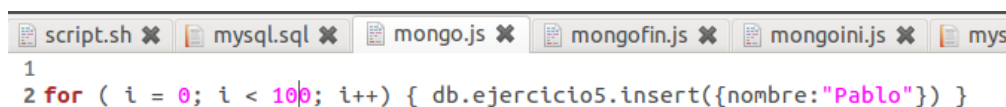
Figura 5.3: Archivo de comandos de inicialización de la BD de mysql

A continuación, se ejecuta la inserción de los valores en las bases de datos con su respectiva medición de tiempos. Se utilizan los archivos mongo.js5 y mysql.sql5.



```
script.sh ✕ mysql.sql ✕ mongo.js ✕ mong
1 use ejercicio5
2
3 DELIMITER $$
4 CREATE PROCEDURE simple_loop ( )
5 BEGIN
6   DECLARE counter INT DEFAULT 0;
7   my_loop: LOOP
8     SET counter=counter+1;
9     insert into ej values ("Pablo");
10    IF counter=100 THEN
11      LEAVE my_loop;
12    END IF;
13  END LOOP my_loop;
14 END$$
15 DELIMITER ;
16
17 call simple_loop();
```

Figura 5.4: Archivo de comandos de inserción de datos en mysql



```
script.sh ✕ mysql.sql ✕ mongo.js ✕ mongofin.js ✕ mongoini.js ✕ mys
1
2 for ( i = 0; i < 100; i++) { db.ejercicio5.insert({nombre:"Pablo"}) }
```

Figura 5.5: Archivo de comandos de inserción de datos en mongo

En las imágenes anteriores⁵⁵, se incluye el bucle de inserción de mysql¹² y el de inserción en mongo¹³

Finalmente, el script ejecuta el borrado de las bases de datos creadas con los archivos mongofin.js5 y mysqlfin.sql5.

¹²<http://totaki.com/poesiabinaria/2013/12/bucles-y-cursores-en-mysql-con-ejemplos/>

¹³<http://lineadecodigo.com/mongodb/recorrer-un-cursor-en-mongodb/>

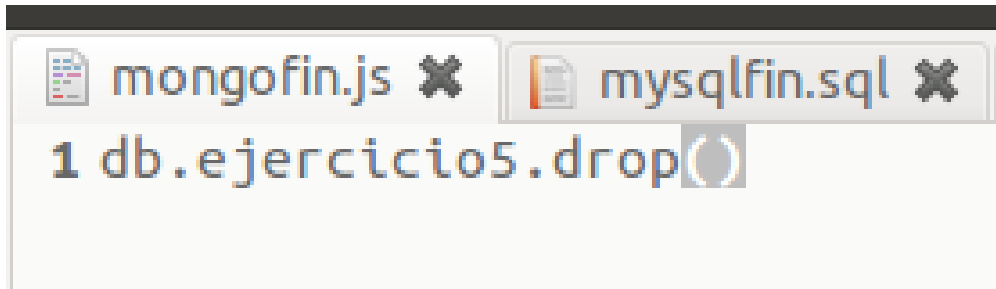


Figura 5.6: Archivo de comandos de borrado de datos en mongo

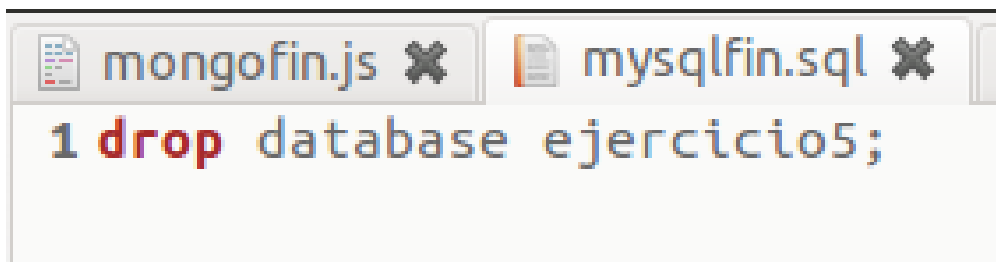


Figura 5.7: Archivo de comandos de borrado de datos en mysql

En la imagen del script5 salen las dos líneas comentadas, ya que a continuación veremos la comprobación de que se han creado correctamente.

Ejecutamos el script y obtenemos el siguiente resultado5.

```
vilchez@vilchez-K52Dr:~/Escritorio/Practica4$ sh script.sh
Tiempo de mongo y mysql accediendo a los elementos de la base de datos
MongoDB shell version: 2.0.4
connecting to: test
false
{ "ok" : 1 }
bye
MongoDB shell version: 2.0.4
connecting to: test
bye
MySQL: 4090123645 nanosegundos, 4 segundos
Mongo: 47755259 nanosegundos, 1 segundos
```

Figura 5.8: Resultado de ejecutar el script

Como podemos observar en la anterior imagen5, el script devuelve finalmente el valor del tiempo que tarda MySQL y Mongo. En este caso MySQL tarda en insertar 100 datos 4 segundos y 090123645 nanosegundos y Mongo tarda menos de un segundo y 47755259 nanosegundos. Podemos contrastar que Mongo lo hace mucho más rápido que MySQL. Si a continuación nos metemos en MySQL, como teníamos comentado el borrado de la base de datos, revisamos la BD ejercicio555 y vemos que tiene los 100 elementos insertados

```
vilchez@vilchez-K52Dr: ~/Escritorio/Practica4
Archivo Editar Ver Buscar Terminal Ayuda
mysql> use ejercicio5
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with 'mysql> set sql_mode=NO_AUTO_CREATE_TABLE'

Database changed
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| ejercicio5 |
| mysql |
| performance_schema |
| phpmyadmin |
+-----+
5 rows in set (0.00 sec)

mysql> select * from ej
-> ;
+-----+
| nombre |
+-----+
| Pablo |
| Pablo |
| Pablo |
| Pablo |
+-----+
```

Figura 5.9: Elementos insertados en MySQL

```
| Pablo |
| Pablo |
| Pablo |
| Pablo |
+-----+
100 rows in set (0.00 sec)

mysql> exit
Bye
```

Figura 5.10: Elementos insertados en MySQL

También comprobamos la base de datos en Mongo5.

```
vilchez@vilchez-K52Dr:~/Escritorio/Practica4$ mongo
MongoDB shell version: 2.0.4
connecting to: test
> show collections
ejercicio5
system.indexes
> db.ejercicio5.find()
{ "_id" : ObjectId("5487983fb6300c3d959b72fa"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b72fb"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b72fc"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b72fd"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b72fe"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b72ff"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7300"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7301"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7302"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7303"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7304"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7305"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7306"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7307"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7308"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b7309"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b730a"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b730b"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b730c"), "nombre" : "Pablo" }
{ "_id" : ObjectId("5487983fb6300c3d959b730d"), "nombre" : "Pablo" }
has more
> db.ejercicio5.count()
100
> exit
bye
```

Figura 5.11: Elementos insertados en Mongo