



ugr | Universidad
de Granada

Grado en Ingeniería Informática.

Cuestiones optionales.

Nombre de la asignatura:
Ingeniería de Servidores.

Realizado por:
Néstor Rodríguez Vico



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS
INFORMÁTICA Y DE TELECOMUNICACIÓN.

Granada, 29 de enero de 2017.

Índice general

1. Práctica 1.	6
1.1. Cuestión opcional 1: Muestre (con capturas de pantalla) cómo ha comprobado que el RAID1 funciona.	6
1.2. Cuestión opcional 2: ¿Qué relación hay entre los atajos de teclado de emacs y los de la consola bash? ¿y entre los de vi y las páginas del manual?	8
2. Práctica 2.	10
2.1. Cuestión opcional 1: Instale y pruebe terminator y/o tmux. Con screen, pruebe su funcionamiento dejando sesiones ssh abiertas en el servidor y recuperándolas posteriormente.	10
2.2. Cuestión opcional 2: Instale el servicio y pruebe su funcionamiento.	13
2.3. Cuestión opcional 3: Instale el servicio y pruebe su funcionamiento.	15
2.4. Cuestión opcional 4: Realice la instalación de uno de estos dos “web containers” y pruebe su ejecución.	16
2.5. Cuestión opcional 5: Realice la instalación de MongoDB en alguna de sus máquinas virtuales. Cree una colección de documentos y haga una consulta sobre ellos.	19
3. Práctica 3.	22
3.1. Cuestión opcional 1: Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.	22
3.2. Cuestión opcional 2: Instale Nagios en su sistema (el que prefiera) documentando el proceso y muestre el resultado de la monitorización de su sistema comentando qué aparece.	24

3.3. Cuestión opcional 4: Pruebe a instalar este monitor en alguno de sus tres sistemas. Realice capturas de pantalla del proceso de instalación y comente capturas de pantalla del programa en ejecución.	29
3.4. Cuestión opcional 5: Pruebe a instalar este monitor en alguno de sus tres sistemas. Realice capturas de pantalla del proceso de instalación y comente capturas de pantalla del programa en ejecución.	37
3.5. Cuestión opcional 6: Instale el monitor. Muestre y comente algunas capturas de pantalla.	44
3.6. Cuestión opcional 10: Escriba un script en PowerShell y analice su comportamiento usando el profiler presentado.	48
3.7. Cuestión opcional 11: Al igual que ha realizado el “profiling” con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).	50
4. Práctica 4.	54
4.1. Cuestión opcional 1: ¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto.	54

Índice de figuras

1.1.1.Simulación fallo software.	6
1.1.2.Comprobación del fallo software.	7
1.1.3.Retirada en caliente del disco.	7
1.1.4.Añadir el disco a <i>/dev/sdb1</i>	7
1.1.5.Comprobación final.	8
2.1.1.Creación de las sesiones de <i>screen</i>	11
2.1.2.Restauración de las sesiones de <i>screen</i>	12
2.1.3.Sesiones de <i>screen</i> restauradas.	12
2.1.4.Sesiones de <i>screen</i> cerradas.	13
2.2.1.Copia del archivo <i>/etc/fail2ban/jail.conf</i>	13
2.2.2.Acceso incorrecto (máquinas conectadas en modo <i>host-only</i>). .	14
2.2.3.Dirección IP bloqueada (máquinas conectadas en modo bridge). .	14
2.3.1.Creación de la base de datos para <i>rkhunter</i>	15
2.3.2.Resultado del análisis del sistema con <i>rkhunter</i>	16
2.4.1.Java no se encuentra instalado de Ubuntu Server.	17
2.4.2.Dirección IP de mi servidor.	17
2.4.3.Servicio <i>tomcat</i> funcionando correctamente (máquinas conectadas en modo <i>host-only</i>).	18
2.4.4.Sesión iniciada en <i>tomcat</i> (máquinas conectadas en modo <i>host-only</i>).	18
2.5.1.Instalación de <i>MongoDB</i>	19
2.5.2.Cambio de <i>SELINUX</i>	20
2.5.3.Creación de la colección, inserción de documentos y consulta. .	21
3.1.1.Comprobamos que tenemos dos RAID.	23
3.1.2.Provocamos un fallo software en el RAID.	23
3.1.3.Quitamos en caliente y añadimos de nuevo el RAID.	23
3.1.4.Observamos el proceso de recuperación con <i>watch</i>	23
3.1.5.RAID completamente restaurado.	24
3.2.1.Processo de descarga de Nagios y preparación para la instalación.	25
3.2.2.Processo de instalación de Nagios.	25

3.2.3.Conección con Nagios (máquina conectadas en modo <i>host-only</i>).	26
3.2.4.Paso 1 de la configuración de la interfaz web (máquina conectadas en modo <i>host-only</i>).	26
3.2.5.Paso 2 de la configuración de la interfaz web (máquina conectadas en modo <i>host-only</i>).	27
3.2.6.Conección con Nagios (máquina conectadas en modo <i>host-only</i>).	27
3.2.7.Monitorización del rendimiento con Nagios (máquina conectadas en modo <i>host-only</i>).	28
3.2.8.Hypermap de mi servidor (máquina conectadas en modo <i>host-only</i>).	28
3.3.1.Archivo <i>/etc/apt/sources.list</i>	29
3.3.2.Instalación de <i>Zabbix</i>	30
3.3.3.Configuración del servidor de <i>Zabbix</i>	30
3.3.4.Extracción de ficheros SQL.	31
3.3.5.Configuración de MySQL.	31
3.3.6.Importación de archivos necesarios para <i>Zabbix</i>	32
3.3.7.Parámetros del archivo <i>/etc/php5/apache2/php.ini</i> modificados.	32
3.3.8.Parámetros del archivo <i>/etc/zabbix/zabbix.conf.php</i> modificados.	33
3.3.9.Configuración archivo apache de <i>Zabbix</i>	33
3.3.10Inicio de <i>Zabbix</i>	34
3.3.11Instalación del Agente de <i>Zabbix</i>	34
3.3.12Dirección IP del servidor con <i>Zabbix</i> instalado (máquinas conectadas en modo <i>host-only</i>).	34
3.3.13Parámetro <i>hostname</i>	35
3.3.14 <i>Zabbix</i> en funcionamiento.	35
3.3.15Host disponibles.	35
3.3.16Cambio del estado del servidor.	36
3.3.17Parámetros monitorizados.	36
3.3.18Valores procesados por el servidor <i>Zabbix</i> por segundo.	37
3.4.1.Instalación de <i>Cacti</i>	38
3.4.2.Iniciamos los servicios.	38
3.4.3.Creación base de datos para <i>Cacti</i>	39
3.4.4.Importación de las tablas para <i>Cacti</i>	39
3.4.5.Parámetros modificados.	40
3.4.6.Permisos modificados.	40
3.4.7.Archivo de configuración de <i>Apache</i> y archivo <i>crontab</i> modificados.	41
3.4.8. <i>Cacti</i> en funcionamiento.	41
3.4.9.Instalación nueva.	42
3.4.10Finalización de la instalación.	42
3.4.11Instalación de <i>Cacti</i> completada.	42

3.4.12Cambio forzado de contraseña.	43
3.4.13Añadimos el gráfico al árbol.	43
3.4.14Datos recogidos.	44
3.5.1.Configuración de <i>AWStats</i>	45
3.5.2.Configuración de <i>Apache</i>	46
3.5.3.Fichero de <i>crontab</i> modificado.	47
3.5.4.Información proporcionada por <i>AWStats</i>	47
3.6.1.Resultado del script 3.6.	48
3.6.2.Error al ejecutar el profiler.	49
3.6.3.Ejemplo de ejecución de PoshProfiler [1].	50
3.7.1.Creación de la colección y de los documentos.	51
3.7.2.Inserción y consulta de los documentos.	52
3.7.3.Entrada más reciente.	53
3.7.4.8 entradas más recientes.	53
4.1.1.Instalación de <i>Scala</i>	55
4.1.2.Ejecución de <i>Gatling</i> y posibles escenarios.	55
4.1.3.Resultado de la ejecución de <i>Gatling</i>	55
4.1.4.Copia de los resultados y dirección IP del servidor.	56
4.1.5.Resultado de <i>Gatling</i> desde el navegador.	56

Capítulo 1

Práctica 1.

1.1. Cuestión opcional 1: Muestre (con capturas de pantalla) cómo ha comprobado que el RAID1 funciona.

Para probar que funciona correctamente vamos a hacer una simulación de fallo y a arreglar el daño causado para ver si el disco se sincroniza correctamente. El proceso es el que sigue: [2]

Primero vamos a simular un fallo software. Para ello usamos el comando *sudo mdadm --manage --set-faulty /dev/md0 /dev/sdb1* como se muestra en la figura 1.1.1.

```
nrv/2016-10-15:~$ sudo mdadm --manage --set-faulty /dev/md0 /dev/sdb1
[ 203.171400] md/raid1:md0: Disk failure on sdb1, disabling device.
[ 203.171400] md/raid1:md0: Operation continuing on 1 devices.
mdadm: set /dev/sdb1 faultu in /dev/md0
```

Figura 1.1.1: Simulación fallo software.

Para ver que correctamente se ha producido el fallo veremos el estado del disco. Para ello usamos el comando *sudo mdadm --detail /dev/md0* como se muestra en la figura 1.1.2.

```

nrv/2016-10-15:~$ sudo mdadm --detail /dev/md0
/dev/md0:
      Version : 1.2
      Creation Time : Thu Oct 13 09:46:46 2016
      Raid Level : raid1
      Array Size : 20952960 (19.98 GiB 21.46 GB)
      Used Dev Size : 20952960 (19.98 GiB 21.46 GB)
      Raid Devices : 2
      Total Devices : 2
      Persistence : Superblock is persistent

      Update Time : Sat Oct 15 17:12:11 2016
                  State : clean, degraded
      Active Devices : 1
      Working Devices : 1
      Failed Devices : 1
      Spare Devices : 0

            Name : nestor:0 (local to host nestor)
            UUID : fcab7918:44e8419b:eca85b7b:d04aea6b
            Events : 132

      Number  Major  Minor  RaidDevice State
          0      8        1         0     active sync   /dev/sda1
          1      0        0         1     removed
          2      8        17        -     faulty spare   /dev/sdb1

```

Figura 1.1.2: Comprobación del fallo software.

Para arreglar el fallo producido vamos a retirar en caliente el disco. Para ello usamos el comando `sudo mdadm /dev/md0 -r /dev/sdb1` como se muestra en la figura 1.1.3.

```

nrv/2016-10-15:~$ sudo mdadm /dev/md0 -r /dev/sdb1
mdadm: hot removed /dev/sdb1 from /dev/md0

```

Figura 1.1.3: Retirada en caliente del disco.

Para finalizar, volvemos a añadir el disco. Para ello usamos el comando `sudo mdadm /dev/md0 -a /dev/sdb1` como se muestra en la figura 1.1.4.

```

nrv/2016-10-15:~$ sudo mdadm /dev/md0 -a /dev/sdb1
mdadm: added /dev/sdb1

```

Figura 1.1.4: Añadir el disco a `/dev/sdb1`.

Para ver que todo ha ido correctamente sólo tenemos que comprobar el estado del disco. Para ello usamos el comando *sudo mdadm --detail /dev/md0* como se muestra en la figura 1.1.5.

```
nrv/2016-10-15:~$ sudo mdadm --detail /dev/md0
/dev/md0:
      Version : 1.2
      Creation Time : Thu Oct 13 09:46:46 2016
      Raid Level : raid1
      Array Size : 20952960 (19.98 GiB 21.46 GB)
      Used Dev Size : 20952960 (19.98 GiB 21.46 GB)
      Raid Devices : 2
      Total Devices : 2
      Persistence : Superblock is persistent

      Update Time : Sat Oct 15 17:14:03 2016
                     State : clean, degraded, recovering
      Active Devices : 1
      Working Devices : 2
      Failed Devices : 0
      Spare Devices : 1

      Rebuild Status : 5% complete

              Name : nestor:0  (local to host nestor)
              UUID : fcab7918:44e8419b:eca85b7b:d04aea6b
              Events : 151

      Number  Major  Minor  RaidDevice State
          0      8        1         0     active sync   /dev/sda1
          2      8       17         1     spare  rebuilding /dev/sdb1
```

Figura 1.1.5: Comprobación final.

1.2. Cuestión opcional 2: ¿Qué relación hay entre los atajos de teclado de emacs y los de la consola bash? ¿y entre los de vi y las páginas del manual?

La relación se debe a que el *bash* se puede poner en varios modos, y uno de ellos es el modo *emacs*. [3]

La relación es que en ambos se usan los mismos comando. Para ver los comandos que se usan en *man* podemos ejecutar *man man* y a continuación

pulsar h para ver los diferentes comandos. Los comandos de vi podemos verlos en la página de IBM [4]. Por ejemplo, para ir a la última línea de la página en ambos casos debemos pulsar la tecla G .

Capítulo 2

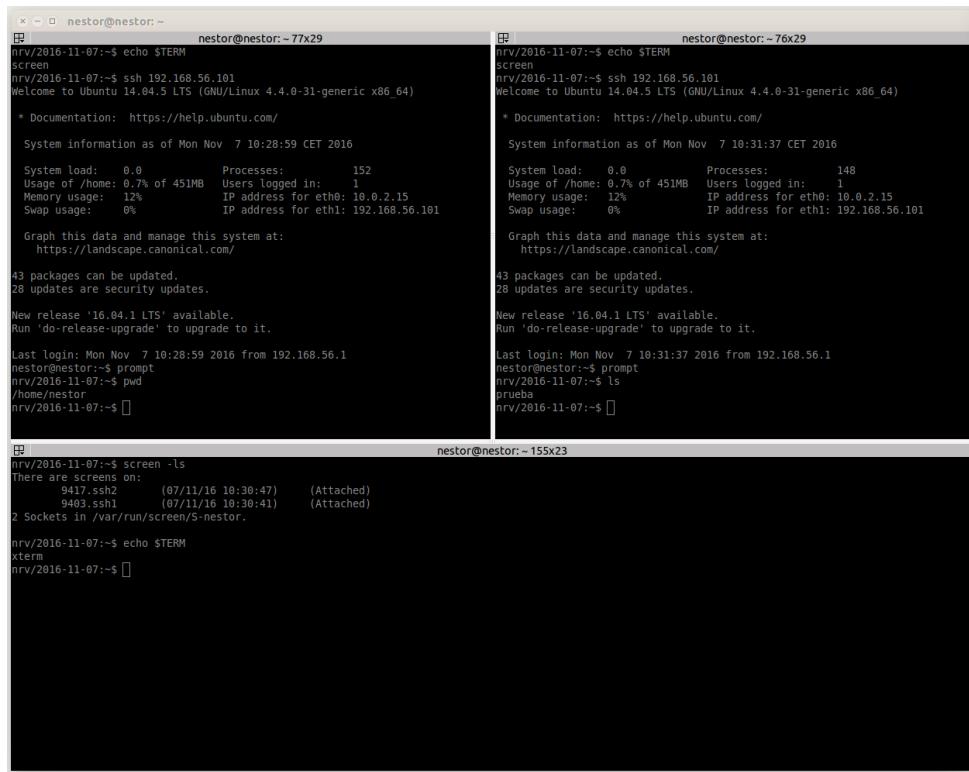
Práctica 2.

2.1. Cuestión opcional 1: Instale y pruebe terminator y/o tmux. Con screen, pruebe su funcionamiento dejando sesiones ssh abiertas en el servidor y recuperándolas posteriormente.

Para instalar *terminator* debemos ejecutar `sudo apt install terminator`. Para instalar *screen* debemos ejecutar `sudo apt install screen`. Aprender a usar *terminator*, a un nivel suficiente para la realización de este ejercicio, es fácil. Pero para aprender a usar *screen* he consultado las páginas del manual [5].

1. Lo primero que he hecho ha sido crear tres terminales en *terminator*, dos para usar *screen* y realizar dos conexiones ssh a mi servidor y una tercera para comprobar el estado de las dos anteriores. En las dos terminales superiores que se ven en la figura 2.1.1 he abierto dos sesiones de *screen*. Para ello he ejecutado `screen -S ssh1` en la terminal de la izquierda y `screen -S ssh2` en la terminal de la derecha. Para corroborar que estamos en *screen*, he ejecutado en ambas `echo $TERM` y efectivamente podemos ver que estamos en *screen* mientras que en la terminal inferior no. También podemos ver en la terminal inferior como hay dos sesiones de *screen* abiertas. En las terminales superiores he realizado dos conexiones ssh, y he ejecutado dos comandos para ver que, efectivamente, funcionan. Todo este proceso se puede ver en la figura 2.1.1.

2. A continuación, cerramos *terminator*. Lo abrimos de nuevo y creamos las mismas tres terminales. Podemos comprobar que no estamos en una sesión de *screen* ejecutando *echo \$TERM* y vemos que no estamos en una sesión de *screen*. Para restaurar las sesiones, ejecutamos *screen -r 9417.ssh2* y *screen -r 9403.ssh1*, como podemos ver en la figura 2.1.2.
3. Tras ejecutar los dos comandos anteriores, podemos ver que se restauran las sesiones de *screen* y las conexiones ssh. Una vez más ejecutamos *echo \$TERM* para ver que, correctamente, estamos en las sesiones de *screen*. Todo este proceso lo podemos ver en la figura 2.1.3.
4. Para cerrar la sesión de *screen* ejecutamos *exit* dos veces, la primera de ellas para cerrar la conexión ssh y la segunda para cerrar la conexión de *screen*. Finalmente, ejecutamos de nuevo *echo \$TERM* para ver que no estamos en una sesión de *screen*. En la terminal inferior ejecutamos *screen -ls* para ver que no queda ninguna sesión de *screen* abierta. Todo lo comentado podemos verlo en la figura 2.1.4.



The figure displays three terminal windows side-by-side, showing the process of creating screen sessions and checking their status.

- Terminal 1:** Shows the initial state where the user is in a regular terminal session. They run `echo $TERM` which returns "screen". Then they run `ssh 192.168.56.101` to connect to a remote host via SSH. The terminal then shows the Ubuntu 14.04.5 LTS welcome message and system information for Mon Nov 7 10:28:59 CET 2016.
- Terminal 2:** Shows the user running `echo $TERM` again, which now returns "xterm". They then run `screen -r 9417.ssh2` to reattach to the previous screen session (9417.ssh2). The terminal then shows the same system information as Terminal 1.
- Terminal 3:** Shows the user running `echo $TERM` again, which now returns "xterm". They then run `screen -r 9403.ssh1` to reattach to another screen session (9403.ssh1). The terminal then shows the same system information as Terminal 1.

Figura 2.1.1: Creación de las sesiones de *screen*.

The screenshot shows three terminal windows side-by-side. The left window has a title bar 'nestor@nestor:~' and a status bar 'nrv/2016-11-07:~\$ echo \$TERM xterm'. It contains the command 'screen -r 9417.ssh2' and its output, which is a blank black screen. The middle window has a title bar 'nestor@nestor:~ 69x21' and a status bar 'nrv/2016-11-07:~\$ echo \$TERM xterm'. It contains the command 'screen -r 9403.ssh1' and its output, which is a blank black screen. The right window has a title bar 'nestor@nestor:~ 141x20' and a status bar 'nrv/2016-11-07:~\$ echo \$TERM xterm'. It contains the command 'screen -ls' and its output, which shows two detached sessions: '9417.ssh2 (07/11/16 10:30:47) (Detached)' and '9403.ssh1 (07/11/16 10:30:41) (Detached)'. Below this, it says '2 Sockets in /var/run/screen/S-nestor.' and ends with 'nrv/2016-11-07:~\$'.

Figura 2.1.2: Restauración de las sesiones de *screen*.

The screenshot shows three terminal windows side-by-side. The left window has a title bar 'nestor@nestor:~' and a status bar 'nrv/2016-11-07:~\$ echo \$TERM xterm'. It contains the command 'ssh 192.168.56.101' and its output, which includes system information for Mon Nov 7 10:28:59 CET 2016. The middle window has a title bar 'nestor@nestor:~ 69x30' and a status bar 'nrv/2016-11-07:~\$ echo \$TERM xterm'. It contains the command 'ssh 192.168.56.101' and its output, which includes system information for Mon Nov 7 10:31:37 CET 2016. The right window has a title bar 'nestor@nestor:~ 141x12' and a status bar 'nrv/2016-11-07:~\$ echo \$TERM xterm'. It contains the command 'screen -ls' and its output, which shows two detached sessions: '9417.ssh2 (07/11/16 10:30:47) (Detached)' and '9403.ssh1 (07/11/16 10:30:41) (Detached)'. Below this, it says '2 Sockets in /var/run/screen/S-nestor.' and ends with 'nrv/2016-11-07:~\$'.

Figura 2.1.3: Sesiones de *screen* restauradas.

The figure consists of three vertically stacked terminal windows, each showing a different stage of a screen session being closed and cleaned up.

- Top Terminal:** Shows two screen sessions closing. The first session, 9403.ssh1, is terminating. The second session, 9417.ssh2, is also terminating. Both sessions have detached from the terminal.
- Middle Terminal:** Shows the user closing both remaining sessions. The command `screen -ls` is run, showing two detached sessions. Then, `screen -r` is run on both, which causes them to close again.
- Bottom Terminal:** Shows the final state where all sessions have been closed. The command `screen -ls` is run, and the output shows "No Sockets found in /var/run/screen/S-nestor." indicating that no sessions are left.

Figura 2.1.4: Sesiones de *screen* cerradas.

2.2. Cuestión opcional 2: Instale el servicio y pruebe su funcionamiento.

Para ver el funcionamiento de *fail2ban* podemos visitar la página oficial de dicho servicio [6]. Como página complementaria voy a usar Digital Ocean [7]. Para instalar *fail2ban* debemos ejecutar el comando *sudo apt install fail2ban*. Una vez instalado, lo recomendable es copiar el archivo */etc/fail2ban/jail.conf* para evitar problemas en futuras actualizaciones. Para ello, ejecutamos el comando *sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local*, como podemos ver en la figura 2.2.1.

```
nrv/2016-11-05:~$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
[sudo] password for nestor:
nrv/2016-11-05:~$ ls -l /etc/fail2ban/ | grep jail
-rw-r--r-- 1 root root 11885 nov 18 2013 jail.conf
drwxr-xr-x 2 root root 4096 nov 18 2013 jail.d
-rw-r--r-- 1 root root 11885 nov 5 20:20 jail.local
nrv/2016-11-05:~$
```

Figura 2.2.1: Copia del archivo */etc/fail2ban/jail.conf*.

Para ver su funcionamiento, voy a intentar conectarme a mi servidor *ssh* pero fallaré la contraseña para ver que *fail2ban* funciona correctamente. Como en mi máquina anfitriona tengo configurado *ssh* para permitir el acceso sin contraseña, voy a realizar el intento de acceso desde CentOS, como podemos ver en la figura 2.2.2.

The screenshot shows two terminal windows side-by-side. The left window is titled 'ISE - US14 [Corriendo] - Oracle VM VirtualBox' and the right window is titled 'ISE - CentOS [Corriendo] - Oracle VM VirtualBox'. Both windows show command-line output.

ISE - US14 [Corriendo] - Oracle VM VirtualBox:

```
nrv/2016-11-05:~$ ifconfig
eth0      Link encap:Ethernet direcciónHW 08:00:27:be:e9:b6
          Direc. inet:10.0.2.15 Difus.:10.0.2.255 Masc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:febe:e9b6/64 alcance:Elink
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Metrica:1
          Paquetes RX:4680 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:2102 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatTX:1000
          Bytes RX:5908139 (5.9 MB) TX bytes:148792 (148.7 KB)

eth1      Link encap:Ethernet direcciónHW 08:00:27:fb:d2:b0
          Direc. inet:192.168.56.255 Masc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:feb:d2b6/64 alcance:Elink
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Metrica:1
          Paquetes RX:334 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:102 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatTX:1000
          Bytes RX:70305 (70.3 KB) TX bytes:25141 (25.1 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1 Masc:255.0.0.0
          Dirección inet6: ::1/128 alcance:localhost
          BUCLE MTU:1500 Metrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatTX:1
          Bytes RX:0 (0.0 B) TX bytes:0 (0.0 B)

nrv/2016-11-05:~$
```

ISE - CentOS [Corriendo] - Oracle VM VirtualBox:

```
nrv/2016-11-05:~$ ssh 192.168.56.101
nrv@192.168.56.101's password:
Permission denied, please try again.
nrv@192.168.56.101's password:
Permission denied, please try again.
nrv@192.168.56.101's password:
Permission denied (publickey,password).
nrv@192.168.56.101's password:
nrv@192.168.56.101's password:
Permission denied, please try again.
nrv@192.168.56.101's password:
```

Figura 2.2.2: Acceso incorrecto (máquinas conectadas en modo *host-only*).

Desde nuestro servidor, ejecutando el comando *sudo iptables -S* vemos que se ha bloqueado la conexión *ssh* para la IP que es la IP de mi máquina virtual de CentOS, como podemos ver en la figura 2.2.3.

The screenshot shows two terminal windows side-by-side. The left window is titled 'ISE - US14 [Corriendo] - Oracle VM VirtualBox' and the right window is titled 'ISE - CentOS [Corriendo] - Oracle VM VirtualBox'. Both windows show command-line output.

ISE - US14 [Corriendo] - Oracle VM VirtualBox:

```
nrv/2016-11-05:~$ sudo iptables -S | grep fail2
-N fail2ban-ssh
-A INPUT -p tcp -m multiport --dports 22 -j fail2ban-ssh
-A fail2ban-ssh -s 192.168.56.102/32 -j REJECT --reject-with icmp-port-unreachable
-A fail2ban-ssh -j RETURN
nrv/2016-11-05:~$ _
```

ISE - CentOS [Corriendo] - Oracle VM VirtualBox:

```
nrv/2016-11-05:~$ ifconfig enp0s8
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
          inet 192.168.56.102 netmask 255.255.255.0 broadcast 192.168.56.255
              inet6 fe80::a00:27ff:fe85:afde prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:85:af:de txqueuelen 1000 (Ethernet)
          RX packets 149 bytes 29298 (28.6 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 126 bytes 28156 (27.4 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

nrv/2016-11-05:~$ _
```

Figura 2.2.3: Dirección IP bloqueada (máquinas conectadas en modo *bridge*).

2.3. Cuestión opcional 3: Instale el servicio y pruebe su funcionamiento.

Para ver el funcionamiento de *rkhunter* podemos visitar la página oficial de dicho servicio [8]. Para instalar el servicio ejecutamos *sudo apt install rkhunter*. Lo primero que hacemos es crear una base de datos de como se encuentra nuestro sistema en el momento actual, para luego usarla como referencia. Para ello ejecutamos el comando *sudo rkhunter --propupd*, como podemos ver en la figura 2.3.1.

```
nrv/2016-11-06:~$ sudo rkhunter --propupd
[sudo] password for nestor:
[ Rootkit Hunter version 1.4.0 ]
File updated: searched for 167 files, found 134
nrv/2016-11-06:~$
```

Figura 2.3.1: Creación de la base de datos para *rkhunter*.

Una vez hemos creado la base de datos, podemos analizar nuestro sistema ejecutando *sudo rkhunter -c --enable all*. Una vez acabado el análisis, podemos ver un resumen del análisis, como podemos ver en la figura 2.3.2.

```
System checks summary
=====

File properties checks...
    Files checked: 134
    Suspect files: 1

Rootkit checks...
    Rootkits checked : 307
    Possible rootkits: 0

Applications checks...
    All checks skipped

The system checks took: 55 seconds

All results have been written to the log file (/var/log/rkhunter.log)

One or more warnings have been found while checking the system.
Please check the log file (/var/log/rkhunter.log)

nrv/2016-11-06:~$ _
```

Figura 2.3.2: Resultado del análisis del sistema con *rkhunter*.

Como podemos ver en la figura 2.3.2, si queremos ver el resultado completo debemos ver el contenido del archivo */var/log/rkhunter.log*. Finalmente, si queremos actualizar la base de datos que creamos anteriormente, debemos ejecutar el comando *sudo rkhunter --update*

2.4. Cuestión opcional 4: Realice la instalación de uno de estos dos “web containers” y pruebe su ejecución.

Voy a probar Apache Tomcat [9] en Ubuntu Server. Para instalar Apache Tomat, primero debemos ver si tenemos instalado Java instalado. Para ello, ejecutamos el comando *java -version*. Como podemos ver en la figura 2.4.1 Java no se encuentra instalado, para instalarlo ejecutamos *sudo apt-get install default-jdk*

```
nrv/2016-11-06:~$ java -version
El programa «java» puede encontrarse en los siguientes paquetes:
 * default-jre
 * gcj-4.8-jre-headless
 * openjdk-7-jre-headless
 * gcj-4.6-jre-headless
 * openjdk-6-jre-headless
Intenta: sudo apt-get install <paquete seleccionado>
nrv/2016-11-06:~$ _
```

Figura 2.4.1: Java no se encuentra instalado de Ubuntu Server.

Una vez hemos instalado java, ejecutamos *sudo apt install tomcat7* para instalar Apache Tomcat. Para ver su funcionamiento, desde la máquina anfitriona vamos a un navegador y escribimos la dirección IP de nuestro servicio web seguido de *:8080*. En mi caso, como podemos ver en la figura 2.4.2, la dirección IP de mi servidor es *192.168.56.101*, por lo tanto en el navegador debemos ir a la dirección *192.168.56.101:8080*. Como podemos ver en la figura 2.4.3, el servicio funciona correctamente.

```
nrv/2016-11-06:~$ ifconfig eth1
eth1      Link encap:Ethernet  direcciónHW 08:00:27:fb:d2:b0
          Direc. inet:192.168.56.101  Difus.:192.168.56.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:feb:d2b0/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:316 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:82 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatX:1000
          Bytes RX:70162 (70.1 KB)  TX bytes:27043 (27.0 KB)

nrv/2016-11-06:~$
```

Figura 2.4.2: Dirección IP de mi servidor.

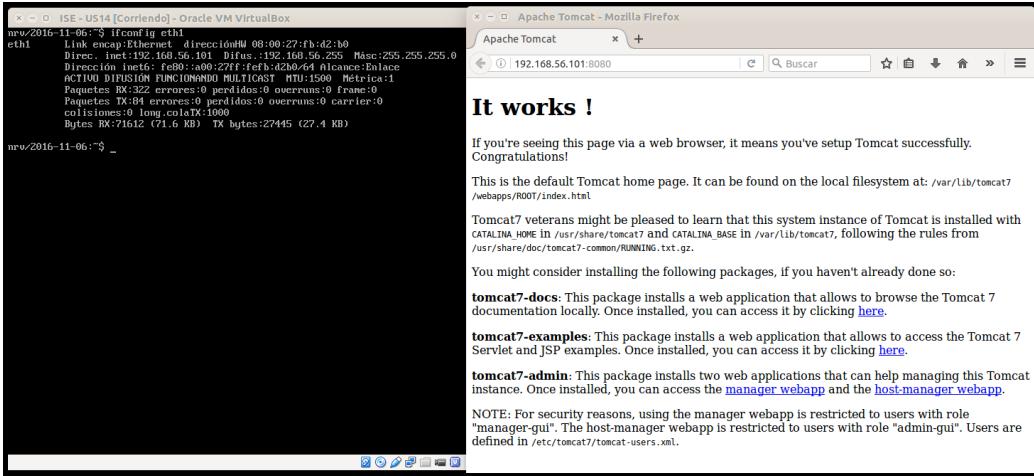


Figura 2.4.3: Servicio *tomcat* funcionando correctamente (máquinas conectadas en modo *host-only*).

Para acceder al gestor de aplicaciones, como podemos ver en la página de Apache Tomcat [10] debemos acceder a la dirección *192.168.56.101:8080/manager/html*. Una vez en esa dirección, nos pide un usuario y una contraseña. Para acceder he creado un usuario con los roles que se puede ver en la figura 2.4.4 y he accedido con el. Dentro de tomcat podemos ver que se puede gestionar diferentes parámetros, como puede ser el tiempo que tardará en expirar una sesión inactiva, como se puede ver en la figura 2.4.4.

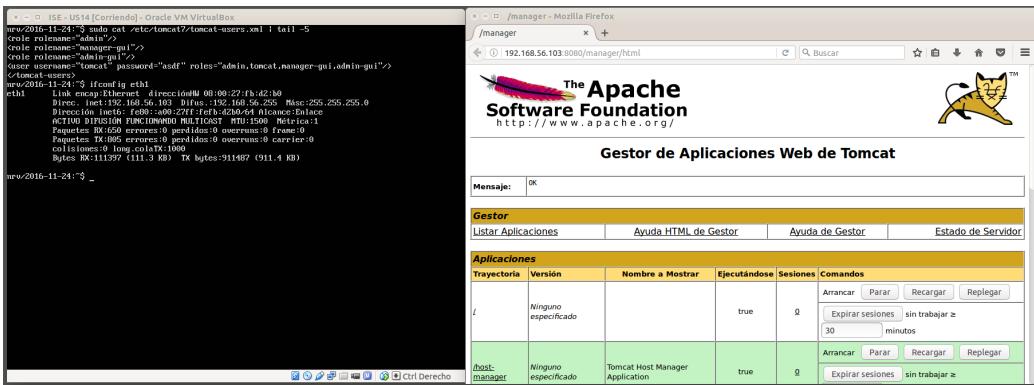


Figura 2.4.4: Sesión iniciada en *tomcat* (máquinas conectadas en modo *host-only*).

2.5. Cuestión opcional 5: Realice la instalación de MongoDB en alguna de sus máquinas virtuales. Cree una colección de documentos y haga una consulta sobre ellos.

Voy a instalar MongoDB en CentOS. Para ello voy a seguir los pasos de la documentación oficial [11].

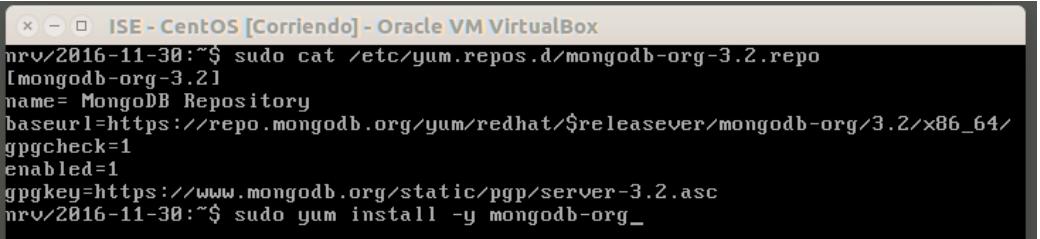
1. Creamos el archivo `/etc/yum.repos.d/mongodb-org-3.2.repo` y añadimos las siguientes líneas:

```
[mongodb-org-3.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/
    mongodb-org/3.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.2.asc
```

El resultado de este paso lo podemos ver en la figura 2.5.1.

2. Instalamos MongoDB ejecutando `sudo yum install -y mongodb-org`, como podemos ver en la figura 2.5.1.

Una vez instalado, debemos cambiar el valor de *SELINUX*. Para ello editamos el fichero `/etc/selinux/config` y cambiamos el valor de *SELINUX* a *permissive*, como podemos ver en la figura 2.5.2. Una vez cambiado, debemos reiniciar CentOS.¹



```
ISE - CentOS [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-30:~$ sudo cat /etc/yum.repos.d/mongodb-org-3.2.repo
[mongodb-org-3.2]
name= MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.2.asc
nrv/2016-11-30:~$ sudo yum install -y mongodb-org_
```

Figura 2.5.1: Instalación de *MongoDB*.

¹Se podría usar *setenforce* pero el cambio no sería permanente.

```

nrv/2016-11-30:~$ sudo cat /etc/selinux/config
[sudo] password for nrv:
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
# SETLOCALDEFS= Check local definition changes
SETLOCALDEFS=0
nrv/2016-11-30:~$ _

```

Figura 2.5.2: Cambio de *SELINUX*.

Para entrar en MongoDB ejecutamos *mongo*. Una vez dentro de MongoDB seguimos los siguientes pasos:

1. Creamos la colección que vamos a usar, *Opcional5_ISE*, para ello ejecutamos
db.createCollection('Opcional5_ISE').
2. Creamos dos documentos para luego insertarlos en la colección que hemos creado en el paso anterior. Para ello ejecutamos:
 - a) *doc = { name: "Nestor", age: 20 }*
 - b) *doc2 = { name: "NestorJunior", age: 10 }*
3. Insertamos los documentos en la colección *Opcional5_ISE*. Para ello ejecutamos:
 - a) *db.Opcional5_ISE.insert(doc)*
 - b) *db.Opcional5_ISE.insert(doc2)*
4. Finalmente realizamos la consulta en la colección ejecutando *db.Opcional5_ISE.find()*

El resultado de este proceso lo podemos ver en la figura 2.5.3.

```
> db.createCollection('Opcional15_ISE')
{ "ok" : 1 }
> doc = {name: "Nestor", age: 20}
{ "name" : "Nestor", "age" : 20 }
> doc2 = {name: "NestorJunior", age: 10}
{ "name" : "NestorJunior", "age" : 10 }
> db.Opcional15_ISE.insert(doc)
WriteResult({ "nInserted" : 1 })
> db.Opcional15_ISE.insert(doc2)
WriteResult({ "nInserted" : 1 })
> db.Opcional15_ISE.find()
{ "_id" : ObjectId("583e966e012a3a714cb873e9"), "name" : "Nestor", "age" : 20 }
{ "_id" : ObjectId("583e9674012a3a714cb873ea"), "name" : "NestorJunior", "age" :
10 }
```

Figura 2.5.3: Creación de la colección, inserción de documentos y consulta.

Capítulo 3

Práctica 3.

3.1. Cuestión opcional 1: Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.

Lo que he hecho ha sido seguir la misma idea que en la cuestión opcional 1 de la práctica 1 de esta asignatura. Al igual que en la práctica 1, me he basado en la información que podemos obtener de la Wiki de Linux [12]. Los pasos que he seguido son los siguientes:

1. Primero comprobamos que efectivamente tenemos dos RAID. Para ello ejecutamos el comando `cat /proc/mdsatat` como se puede ver en la figura 3.1.1. También se podría haber hecho con `watch -n2 cat /proc/mdstat` pero como no se producen cambios en el fichero, no le sacamos partido a la utilidad `watch`.
2. A continuación, producimos un fallo por software en el RAID ejecutando el comando `sudo mdadm --manage --set-faulty /dev/md0 /dev/sdb1` como podemos ver en la figura 3.1.2.
3. Retiramos el RAID en caliente ejecutando `sudo mdadm /dev/md0 -r /dev/sdb1` y lo añadimos ejecutando `sudo mdadm /dev/md0 -a /dev/sdb1`, como podemos ver en la figura 3.1.3.
4. Ahora si podemos sacarle partido a `watch`. Para comprobar el proceso de recuperación del RAID ejecutamos `watch -n2 cat /proc/mdstat`. De

este modo, cada dos segundos se ejecutará `cat /proc/mdstat`, proporcionándonos así una idea de como va el progreso, como se puede ver en la figura 3.1.4.

5. Tras un tiempo de recuperación, el RAID se ha recuperado correctamente y vuelve a estar en funcionamiento, como podemos ver en la figura 3.1.5.

```
nrv/2016-11-23:~$ cat /proc/mdstat
Personalities : [raid1] [linear] [multipath] [raid0] [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 sda1[0] sdb1[2]
      20952960 blocks super 1.2 [2/2] [UU]

unused devices: <none>
nrv/2016-11-23:~$
```

Figura 3.1.1: Comprobamos que tenemos dos RAID.

```
nrv/2016-11-23:~$ sudo mdadm --manage --set-faulty /dev/sdb1
[sudo] password for nestor:
[ 1093.933066] md/raid1:md0: Disk failure on sdb1, disabling device.
[ 1093.933066] md/raid1:md0: Operation continuing on 1 devices.
mdadm: set /dev/sdb1 faulty in /dev/	md0
nrv/2016-11-23:~$ cat /proc/mdstat
Personalities : [raid1] [linear] [multipath] [raid0] [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 sda1[0] sdb1[2](F)
      20952960 blocks super 1.2 [2/1] [U_]

unused devices: <none>
nrv/2016-11-23:~$ _
```

Figura 3.1.2: Provocamos un fallo software en el RAID.

```
nrv/2016-11-23:~$ 
nrv/2016-11-23:~$ sudo mdadm /dev/md0 -r /dev/sdb1
mdadm: hot removed /dev/sdb1 from /dev/md0
nrv/2016-11-23:~$ sudo mdadm /dev/md0 -a /dev/sdb1
mdadm: added /dev/sdb1
nrv/2016-11-23:~$
```

Figura 3.1.3: Quitamos en caliente y añadimos de nuevo el RAID.

```
Cada 2,0s: cat /proc/mdstat                                         Wed Nov 23 07:56:31 2016
Personalities : [raid1] [linear] [multipath] [raid0] [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 sdb1[2] sda1[0]
      20952960 blocks super 1.2 [2/1] [U_]
      [>.....] recovery = 2.5% (536576/20952960) finish=6.3min speed=53657K/sec

unused devices: <none>
```

Figura 3.1.4: Observamos el proceso de recuperación con `watch`.

```

x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
Cada 2,0s: cat /proc/mdstat
Wed Nov 23 08:00:19 2016

Personalities : [raid1] [linear] [multipath] [raid0] [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 sdb1[2] sda1[0]
      20952960 blocks super 1.2 [2/2] [UU]

unused devices: <none>

```

Figura 3.1.5: RAID completamente restaurado.

3.2. Cuestión opcional 2: Instale Nagios en su sistema (el que prefiera) documentando el proceso y muestre el resultado de la monitorización de su sistema comentando qué aparece.

El proceso de instalación lo podemos ver en la página oficial de Nagios [13]. Yo lo voy a hacer en CentOS. Los pasos a seguir son:

1. El proceso de instalacion se hace desde el directorio */tmp*, así que ejecutamos *cd /tmp* para irnos a dicha carpeta, como podemos ver en la figura 3.2.1.
2. Descargamos la última versión de *Nagios*. Para ello ejecutamos el comando *wget http://assets.nagios.com/downloads/nagiosxi/xi-latest.tar.gz*, como podemos ver en la figura 3.2.1.
3. Descomprimimos Nagios ejecutando *tar xzf xi-latest.tar.gz*, como podemos ver en la figura 3.2.1.
4. Nos vamos a la carpeta que se ha creado ejecutando *cd /tmp/nagiosxi*, como podemos ver en la figura 3.2.2.
5. Ejecutamos el script de instalación con el comando *sudo ./fullinstalation*¹, como podemos ver en la figura 3.2.2. Nos preguntara si queremos continuar, le decimos que sí pulsando la tecla *y*.

¹Cuidado: este script cambia la contraseña del usuario root de MySQL a *nagiosxi*

```

ISE - CentOS [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-23:/tmp$ cd /tmp/ && wget http://assets.nagios.com/downloads/nagiosxi/xi-latest.tar.gz
--2016-11-23 08:50:12--  http://assets.nagios.com/downloads/nagiosxi/xi-latest.tar.gz
Resolviendo assets.nagios.com (assets.nagios.com)... 72.14.181.71, 2600:3c00::f0
3c:91ff:fedf:b821
Conectando con assets.nagios.com (assets.nagios.com)[72.14.181.71]:80... conectado.
Petición HTTP enviada, esperando respuesta... 301 Moved Permanently
Localización: https://assets.nagios.com/downloads/nagiosxi/xi-latest.tar.gz [siguiendo]
--2016-11-23 08:50:13--  https://assets.nagios.com/downloads/nagiosxi/xi-latest.tar.gz
Conectando con assets.nagios.com (assets.nagios.com)[72.14.181.71]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 52190204 (50M) [application/x-gzip]
Grabando a: "xi-latest.tar.gz"

100%[=====] 52.190.204 1,80MB/s en 51s
2016-11-23 08:51:05 (990 KB/s) - "xi-latest.tar.gz" guardado [52190204/52190204]

nrv/2016-11-23:/tmp$ tar xzf xi-latest.tar.gz
nrv/2016-11-23:/tmp$ _

```

Figura 3.2.1: Proceso de descarga de Nagios y preparación para la instalación.

```

ISE - CentOS [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-23:/tmp$ cd nagiosxi/
nrv/2016-11-23:/tmp/nagiosxi$ sudo ./fullinstall
[sudo] password for nrv:
=====
Nagios XI Full Installer
=====

This script will do a complete install of Nagios XI by executing all necessary
sub-scripts.

IMPORTANT: This script should only be used on a 'clean' install of CentOS or
RedHat. Do NOT use this on a system that has been tasked with other purposes
or has an existing install of Nagios Core. To create such a clean install
you should have selected ONLY the 'Base' package in the OS installer.
Do you want to continue? [Y/n] _

```

Figura 3.2.2: Proceso de instalación de Nagios.

Una vez hemos instalado Nagios, en el manual de instalación [13] podemos ver que para acceder a Nagios desde mi máquina anfitriona debemos poner en el buscador *http://IP/nagios*, donde *IP* es la dirección del servidor al que nos queremos conectar. En mi caso, la dirección IP es *192.168.56.101*, como podemos ver en la figura 3.2.3. Tras introducir la dirección en el navegador, podemos ver que nos conectamos correctamente a Nagios, como se puede ver en la figura 3.2.3.

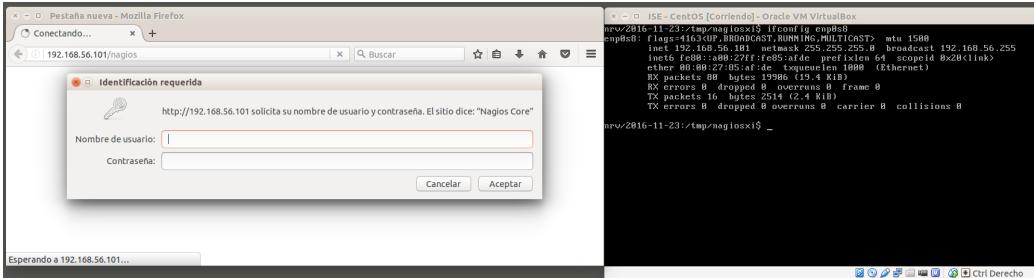


Figura 3.2.3: Conexión con Nagios (máquina conectadas en modo *host-only*).

Vemos que nos pide una contraseña. Para poder acceder debemos seguir los pasos que nos indica Nagios en su guía para configurar la interfaz web [14]. Debemos acceder a la dirección de nuestro servidor, como podemos ver en la figura 3.2.4. Una vez en dicha página, le damos a acceder y ahí configuraremos nuestros datos, como podemos ver en la figura 3.2.5. A continuación, pulsamos *install*.

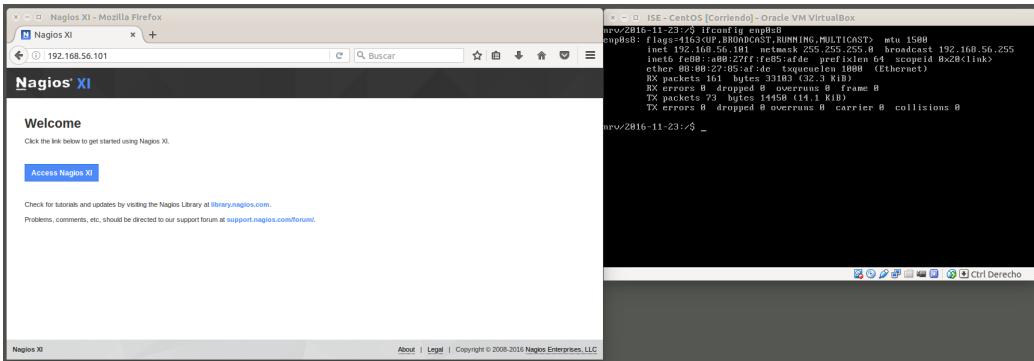


Figura 3.2.4: Paso 1 de la configuración de la interfaz web (máquina conectadas en modo *host-only*).

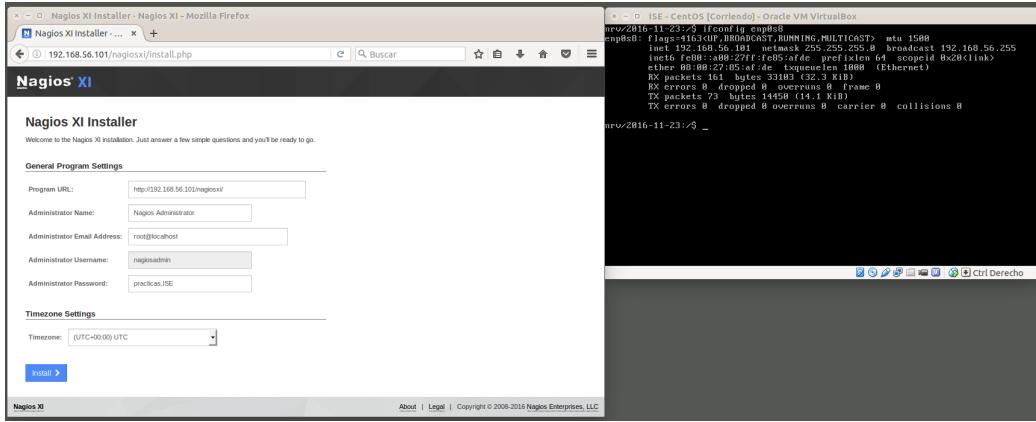


Figura 3.2.5: Paso 2 de la configuración de la interfaz web (máquina conectadas en modo *host-only*).

Ahora si podemos acceder usando los datos que hemos introducido en el paso anterior (ver figura 3.2.5), como podemos ver en la figura 3.2.6.

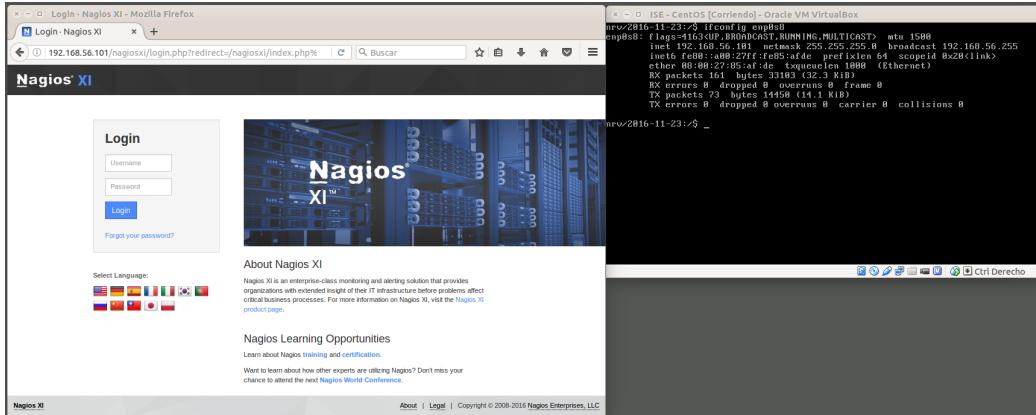


Figura 3.2.6: Conexión con Nagios (máquina conectadas en modo *host-only*).

Una vez hemos accedido a Nagios podemos ver que se pueden monitorizar diferentes parámetros. Como la mayoría de monitores, nos permite monitorizar el rendimiento de nuestro servidor, como podemos ver en la figura 3.2.7. Una de las características que más me ha llamado la atención es la llamada *Hypermap*. Esta opción nos muestra un mapa con el estado actual de los dispositivos de red (hosts), como podemos ver en la figura 3.2.8.

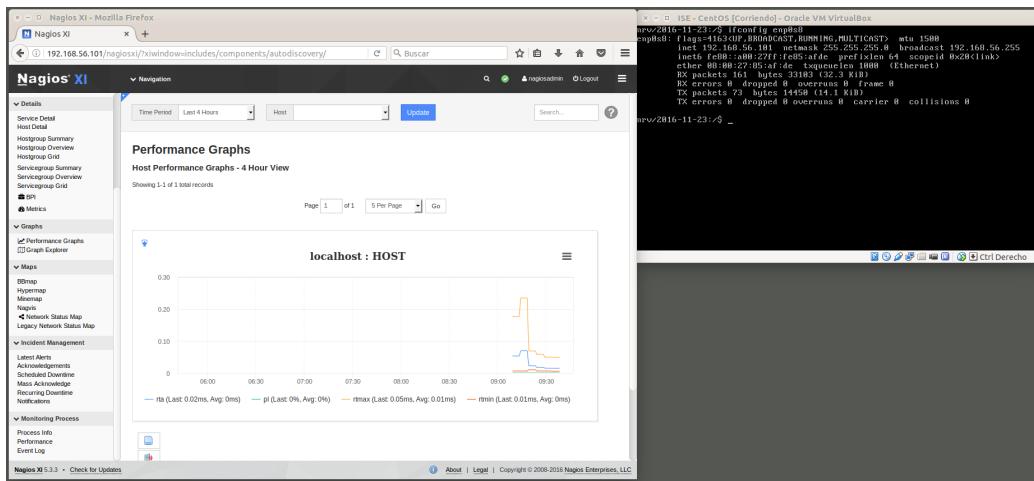


Figura 3.2.7: Monitorización del rendimiento con Nagios (máquina conectadas en modo *host-only*).

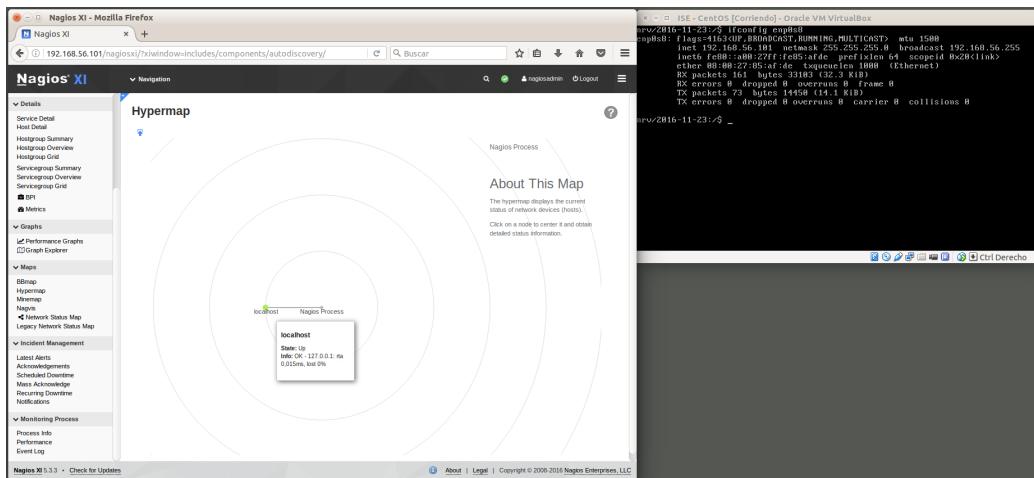


Figura 3.2.8: Hypermap de mi servidor (máquina conectadas en modo *host-only*).

3.3. Cuestión opcional 4: Pruebe a instalar este monitor en alguno de sus tres sistemas. Realice capturas de pantalla del proceso de instalación y comente capturas de pantalla del programa en ejecución.

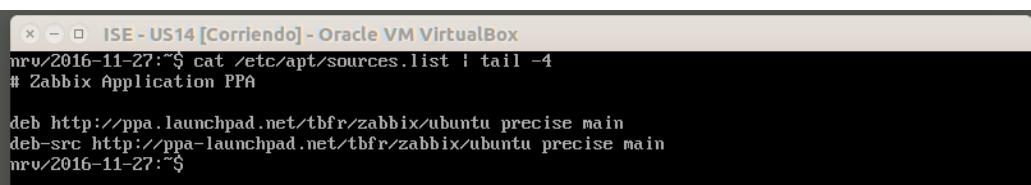
Voy a instalar *Zabbix* en Ubuntu Server siguiendo los pasos que nos indica Digital Ocean [15]. Los pasos a seguir para la instalación son:

1. *Zabbix* se encuentra en los repositorios de Ubuntu, pero está desactualizado, por ello vamos a instalarlo desde los repositorios del programa. Para ello, editamos el fichero */etc/apt/sources.list* y añadimos las líneas:

```
# Zabbix Application PPA
deb http://ppa.launchpad.net/tbfr/zabbix/ubuntu precise main
deb-src http://ppa.launchpad.net/tbfr/zabbix/ubuntu precise main
```

El archivo quedaría como podemos ver en la figura 3.3.1.

2. A continuación añadimos la clave del PPA para que *apt* confíe en la fuente. Para ello ejecutamos: *sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys C407E17D5F76A32B*, como podemos ver en la figura 3.3.2.
3. Una vez hecho esto, actualizamos los repositorios con *sudo apt-get update* e instalamos *Zabbix* ejecutando *sudo apt-get install zabbix-server-mysql php5-mysql zabbix-frontend-php*, como podemos ver en la figura 3.3.2.



```
ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:~$ cat /etc/apt/sources.list | tail -4
# Zabbix Application PPA

deb http://ppa.launchpad.net/tbfr/zabbix/ubuntu precise main
deb-src http://ppa.launchpad.net/tbfr/zabbix/ubuntu precise main
nrv/2016-11-27:~$
```

Figura 3.3.1: Archivo */etc/apt/sources.list*.

```

ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:~$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys C407E17D5F76A32B
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir /tmp/tmp.1hEAj3gDR
R --no-auto-check-trustdb --trust-model always --keyring /etc/apt/trusted.gpg --primary-keyring /etc
/apt/trusted.gpg --keyserver keyserver.ubuntu.com --recv-keys C407E17D5F76A32B
gpg: solicitando clave 5F76A32B de hkp servidor keyserver.ubuntu.com
gpg: clave 5F76A32B: clave pública "Launchpad PPA for Tobias Frederick" importada
gpg: Cantidad total procesada: 1
gpg:           importadas: 1 (RSA: 1)
nrv/2016-11-27:~$ sudo apt-get update && sudo apt-get install zabbix-server-mysql php5-mysql zabbix-
frontend-php

```

Figura 3.3.2: Instalación de *Zabbix*.

Los siguiente que debemos hacer es configurar el server de *Zabbix*, para ello editamos el archivo */etc/zabbix/zabbix_server.conf*, y cambiamos el valor de *DBName*, *DBUser* y *DBPassword*. Dichos parámetros han quedado como podemos ver en la figura 3.3.3.

```

ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:~$ sudo cat /etc/zabbix/zabbix_server.conf | grep DBName=zabbix
DBName=zabbix
nrv/2016-11-27:~$ sudo cat /etc/zabbix/zabbix_server.conf | grep DBUser=zabbix
DBUser=zabbix
nrv/2016-11-27:~$ sudo cat /etc/zabbix/zabbix_server.conf | grep DBPassword=pr
DBPassword=practicas,ISE
nrv/2016-11-27:~$ _

```

Figura 3.3.3: Configuración del servidor *Zabbix*.

A continuación configuraremos MySQL. Para ello seguimos los siguientes pasos:

1. Extramos los ficheros SQL del directorio *cd /usr/share/zabbix-server-mysql/* ejecutando *sudo gunzip *.gz*, como podemos ver en la figura 3.3.4.
2. Entramos a MySQL como usuarios root ejecutando *mysql -u root -p*, como podemos ver en la figura 3.3.5.
3. Creamos un usuario para *Zabbix* que coincida con los datos que escribimos en el fichero */etc/zabbix/zabbix_server.conf* ejecutando *create user 'zabbix'@'localhost' identified by 'practicas,ISE';*, como podemos ver en la figura 3.3.5.
4. Creamos una base de datos para *Zabbix* ejecutando *create database zabbix;*, como podemos ver en la figura 3.3.5.
5. Le damos permisos sobre dicha base de datos al usuario que hemos creado con anterioridad ejecutando *grant all privileges on zabbix.* to 'zabbix'@'localhost';*, como podemos ver en la figura 3.3.5.

6. Actualizamos los permisos ejecutando *flush privileges;*, como podemos ver en la figura 3.3.5.
7. Salimos de MySQL ejecutando *exit;*, como podemos ver en la figura 3.3.5.
8. Importamos los archivos que *Zabbix* necesita para funcionar, para ello y como podemos ver en la figura 3.3.6, ejecutamos:
 - *mysql -u zabbix -p zabbix < schema.sql*
 - *mysql -u zabbix -p zabbix < images.sql*
 - *mysql -u zabbix -p zabbix < data.sql*

```
nrv/2016-11-27:/usr/share/zabbix-server-mysql$ sudo gunzip *.gz
[sudo] password for nestor:
nrv/2016-11-27:/usr/share/zabbix-server-mysql$
```

Figura 3.3.4: Extracción de ficheros SQL.

```
nrv/2016-11-27:/usr/share/zabbix-server-mysql$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 44
Server version: 5.5.53-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create user 'zabbix'@'localhost' identified by 'practicas,ISE';
Query OK, 0 rows affected (0.00 sec)

mysql> create database zabbix;
Query OK, 1 row affected (0.00 sec)

mysql> grant all privileges on zabbix.* to 'zabbix'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> exit;
Bye
nrv/2016-11-27:/usr/share/zabbix-server-mysql$ _
```

Figura 3.3.5: Configuración de MySQL.

```

x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:/usr/share/zabbix-server-mysql$ mysql -u zabbix -p zabbix < schema.sql
Enter password:
nrv/2016-11-27:/usr/share/zabbix-server-mysql$ mysql -u zabbix -p zabbix < images.sql
Enter password:
nrv/2016-11-27:/usr/share/zabbix-server-mysql$ mysql -u zabbix -p zabbix < data.sql
Enter password:
nrv/2016-11-27:/usr/share/zabbix-server-mysql$ _

```

Figura 3.3.6: Importación de archivos necesarios para *Zabbix*.

Una vez hemos configurado MySQL, pasamos a la configuración de PHP. Para ello editamos el fichero */etc/php5/apache2/php.ini*. Buscamos y modificamos los siguientes parámetros para que queden como a continuación:

- `post_max_size = 16M`
- `max_execution_time = 300`
- `max_input_time = 300`
- `date.timezone = UTC`

El resultado lo podemos ver en la figura 3.3.7.

```

x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:/$ sudo cat /etc/php5/apache2/php.ini | grep post_max_size
post_max_size = 16M
nrv/2016-11-27:/$ sudo cat /etc/php5/apache2/php.ini | grep max_execution_time
max_execution_time = 300
nrv/2016-11-27:/$ sudo cat /etc/php5/apache2/php.ini | grep 'max_input_time = 300'
max_input_time = 300
nrv/2016-11-27:/$ sudo cat /etc/php5/apache2/php.ini | grep 'date.timezone = UTC'
date.timezone = UTC
nrv/2016-11-27:/$

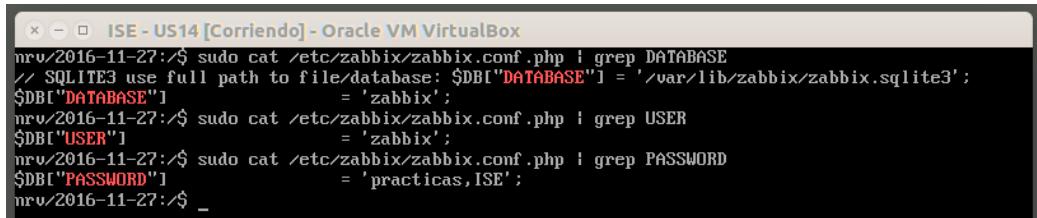
```

Figura 3.3.7: Parámetros del archivo */etc/php5/apache2/php.ini* modificados.

Copiamos el archivo de configuración de php específico de *Zabbix* ejecutando `sudo cp /usr/share/doc/zabbix-frontend-php/examples/zabbix.conf.php.example /etc/zabbix/zabbix.conf.php` para poder ejecutarlo y cambiar los parámetros que podemos ver a continuación para que queden de la siguiente manera:

- `$DB[“DATABASE”] = ‘zabbix’;`
- `$DB[“USER”] = ‘zabbix’;`
- `$DB[“PASSWORD”] = ‘practicas,ISE’`

El resultado lo podemos ver en la figura 3.3.8.



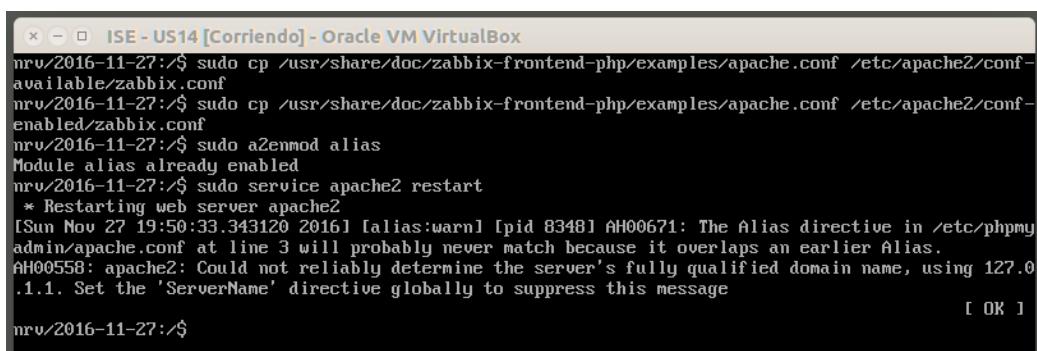
```
nrv/2016-11-27:~$ sudo cat /etc/zabbix/zabbix.conf.php | grep DATABASE
// SQLITE3 use full path to file/database: $DBI["DATABASE"] = '/var/lib/zabbix/zabbix.sqlite3';
$DBI["DATABASE"] = 'zabbix';
nrv/2016-11-27:~$ sudo cat /etc/zabbix/zabbix.conf.php | grep USER
$DBI["USER"] = 'zabbix';
nrv/2016-11-27:~$ sudo cat /etc/zabbix/zabbix.conf.php | grep PASSWORD
$DBI["PASSWORD"] = 'practicas,ISE';
nrv/2016-11-27:~$ _
```

Figura 3.3.8: Parámetros del archivo */etc/zabbix/zabbix.conf.php* modificados.

A continuación copiamos el archivo apache de *Zabbix* en los archivos de configuración de apache ejecutando. Para ello ejecutamos:

- *sudo cp /usr/share/doc/zabbix-frontend-php/examples/apache.conf /etc/apache2/conf-available/zabbix.conf*
- *sudo cp /usr/share/doc/zabbix-frontend-php/examples/apache.conf /etc/apache2/conf-enable/zabbix.conf*

Nos aseguramos de que los alias están habilitados dentro de Apache ejecutando *sudo a2enmod alias* y reiniciamos *Apache* ejecutando *sudo service apache2 restart*. Todo este proceso lo podemos ver en la figura 3.3.9. Lo siguiente que tenemos que hacer es editar el fichero */etc/default/zabbix-server* y cambiar el valor de *START* a “yes” e iniciamos *Zabbix* ejecutando *sudo service zabbix-server start* como podemos ver en la figura 3.3.10.



```
nrv/2016-11-27:~$ sudo cp /usr/share/doc/zabbix-frontend-php/examples/apache.conf /etc/apache2/conf-available/zabbix.conf
nrv/2016-11-27:~$ sudo cp /usr/share/doc/zabbix-frontend-php/examples/apache.conf /etc/apache2/conf-enabled/zabbix.conf
nrv/2016-11-27:~$ sudo a2enmod alias
Module alias already enabled
nrv/2016-11-27:~$ sudo service apache2 restart
* Restarting web server apache2
[Sun Nov 27 19:50:33.343120 2016] [alias:warn] [pid 8348] AH00671: The Alias directive in /etc/phpmyadmin/apache.conf at line 3 will probably never match because it overlaps an earlier Alias.
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive globally to suppress this message
[ OK ]
```

Figura 3.3.9: Configuración archivo apache de *Zabbix*.

```

x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:~$ sudo cat /etc/default/zabbix-server | grep START
START=yes
nrv/2016-11-27:~$ sudo service zabbix-server start
zabbix-server start/running, process 7444
nrv/2016-11-27:~$
```

Figura 3.3.10: Inicio de *Zabbix*.

Ya está todo listo en el servidor, ahora debemos configurar el cliente. El cliente en mi caso va a ser mi máquina anfitriona. Debemos instalar el *Agente de Zabbix*, para ello seguimos los 2 primeros pasos que realizamos en el servidor 3.3, como podemos ver en la figura 3.3.11. A continuación instalamos el *Agente de Zabbix* ejecutando *sudo apt-get install zabbix-agent*.

```

x - nestor@nestor ~
nrv/2016-11-27:~$ sudo cat /etc/apt/sources.list | tail -4
# Zabbix Application PPA
deb http://ppa.launchpad.net/tbfr/zabbix/ubuntu precise main
deb http://ppa.launchpad.net/tbfr/zabbix/ubuntu precise/main
nrv/2016-11-27:~$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys C407E17D5F76A32B
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir '/tmp/tmp.t1lkqkf1Fu' --no-auto-check-trustdb --trust-model always --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyring /etc/apt/trusted.gpg.d/webupd8team-java.gpg --keyring /etc/apt/trusted.gpg.d/webupd8team-javafx.gpg --keyserver keyserver.ubuntu.com --recv-keys C407E17D5F76A32B
gpg:clave C407E17D5F76A32B: clave pública "Launchpad PPA for Tobias Frederick" importada
gpg: Cantidad total procesada: 1
gpg:           importadas: 1 (RSA: 1)
```

Figura 3.3.11: Instalación del *Agente de Zabbix*.

Una vez instalado, pasamos a modificar los archivos de configuración. El primero de ellos es */etc/zabbix/zabbix_agentd.conf*. Como podemos ver en la figura 3.3.12 la dirección IP de mi servidor es *192.168.56.103* así que en dicho archivo, en el parámetro *Server* debemos poner dicha dirección, como se ve en la figura 3.3.12 y reiniciamos el servicio ejecutando *sudo service zabbix-agent restart*.

```

[x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:~$ sudo cat /etc/zabbix/zabbix_agentd.conf | grep 192.168.56.103
Server=192.168.56.103
nrv/2016-11-27:~$ sudo service zabbix-agent restart
zabbix-agent stop/waiting
zabbix-agent start/running, process 16874
nrv/2016-11-27:~$
```



```

[x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-11-27:~$ ifconfig
eth0 Link encap:Ethernet Brdrg:00:0c:29:be:c6:16
      BROADCAST Multicast MAC:255.255.255.255
      Dirección inet: 192.168.56.103 Brdrg:255.255.255.255
      Dirección inet6: fe80::a0c:29ff:febe:c616 Brdrg:ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
      ACTIUO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Metrica:1
      Páginas TX:3172 errores:0 perdidos:0 overrun:0 frame:0
      Paquetes RX:3172 errores:0 perdidos:0 overrun:0 carrier:0
      colisiones:0 long_colat:1000
      Bytes RX:19655101 (1.9 MB) TX bytes:249886 (249.8 KB)
      Bytes RX:19655101 (1.9 MB) TX bytes:249886 (249.8 KB)

eth1 Link encap:Ethernet Brdrg:00:0c:29:7f:21:14
      BROADCAST Multicast MAC:255.255.255.255
      Dirección inet: 192.168.56.104 Brdrg:255.255.255.255
      Dirección inet6: fe80::a0c:29ff:fe7f:2114 Brdrg:ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
      ACTIUO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Metrica:1
      Páquetes TX:93 errores:0 perdidos:0 overrun:0 frame:0
      Paquetes RX:93 errores:0 perdidos:0 overrun:0 carrier:0
      colisiones:0 long_colat:1000
      Bytes RX:176551 (176.5 KB) TX bytes:172 (172.0 B)
      Bytes RX:176551 (176.5 KB) TX bytes:172 (172.0 B)

nrv/2016-11-27:~$
```

Figura 3.3.12: Dirección IP del servidor con *Zabbix* instalado (máquinas conectadas en modo *host-only*).

Dentro del mismo fichero debemos editar el valor de *Hostname* y poner el *hostname* de la máquina cliente, *nestor* en mi caso, y reiniciamos el servicio, como podemos ver en la figura 3.3.13

```

x - nester@nestor: ~
nrv/2016-11-27:-$ sudo cat /etc/zabbix/zabbix_agentd.conf | grep nestor
Hostname=nestor
nrv/2016-11-27:-$ sudo service zabbix-agent restart
zabbix-agent stop/waiting
zabbix-agent start/running, process 17524
nrv/2016-11-27:-$ 

```

Figura 3.3.13: Parámetro *hostname*.

Finalmente, para acceder al servicio debemos escribir en la barra del navegador la dirección IP de nuestro servidor seguido de */zabbix*, como podemos ver en la figura 3.3.14. Por defecto, el usuario y contraseña para entrar son *admin* y *zabbix* respectivamente.

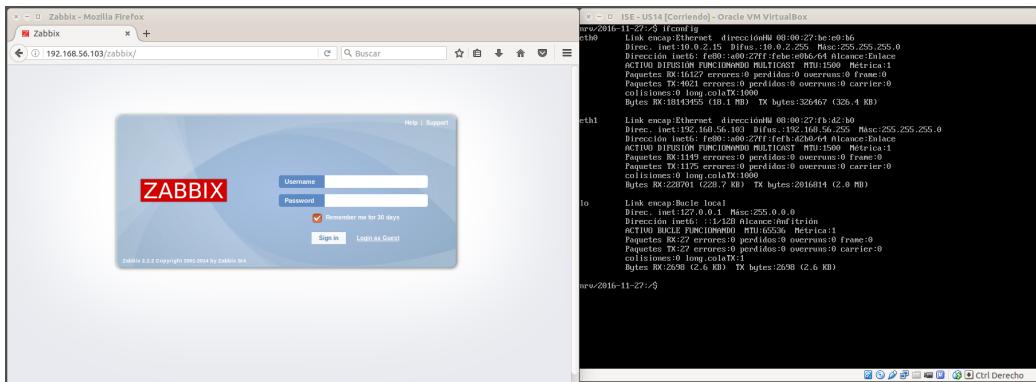


Figura 3.3.14: Zabbix en funcionamiento

Una vez hemos entrado, nos vamos a la pestaña *Configuration* y dentro de dicha pestaña elegimos la de *Hosts*, como podemos ver la figura 3.3.15. Una vez aquí, pinchamos sobre *Zabbix Server*.

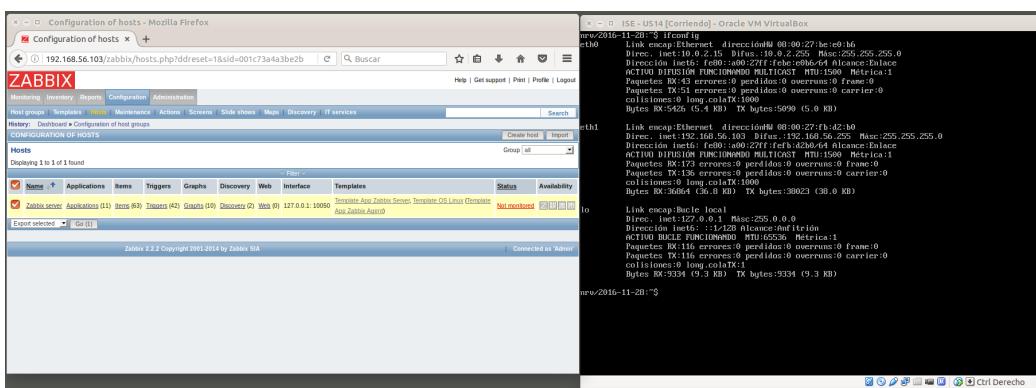


Figura 3.3.15: Host disponibles.

Dentro del servidor de *Zabbix*, al final de todo, en la sección de *Status* elegimos *Monitored*, como podemos ver en la figura 3.3.16.

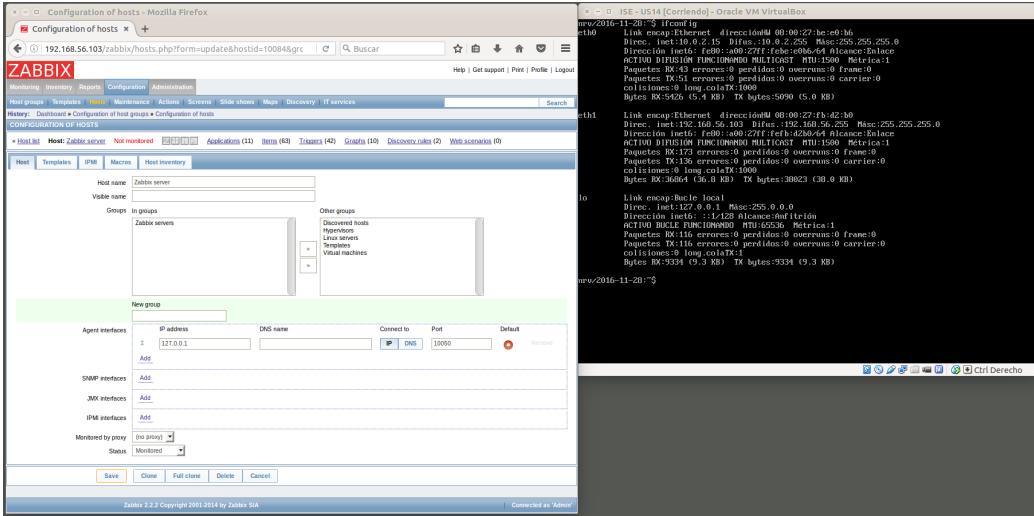


Figura 3.3.16: Cambio del estado del servidor.

Tras unos instantes, podemos ver que en la pestaña *Last data* dentro de *Monitorig* los distintos hosts que se pueden modificar. Dentro de nuestro servidor *Zabbix*, podemos ver una gran cantidad de parámetros monitorizados. Parte de ellos los podemos ver en la figura 3.3.17.

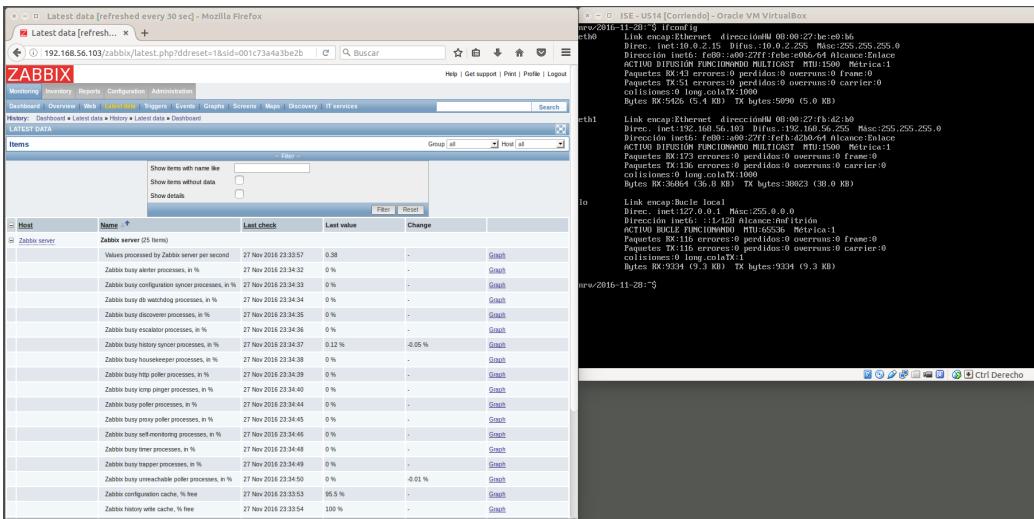


Figura 3.3.17: Parámetros monitorizados.

Uno de los parámetros que podemos ver es el número de valores procesados por el servidor *Zabbix* por segundo. Podemos ver una gráfica sobre la información recogida de dicho parámetro en la figura 3.3.18. En dicha gráfica podemos ver que no hay mucha actividad en el servidor. El servidor se ha puesto en marcha a las 23:14 y la gráfica llega hasta las 23:36. En este periodo el valor se mantiene “constante” en un valor algo superior a 0.38. Podemos ver que hay dos picos a las 23:18 y 23:28 con un valor cercano a 0.42 valores procesados por el servidor *Zabbix* por segundo.

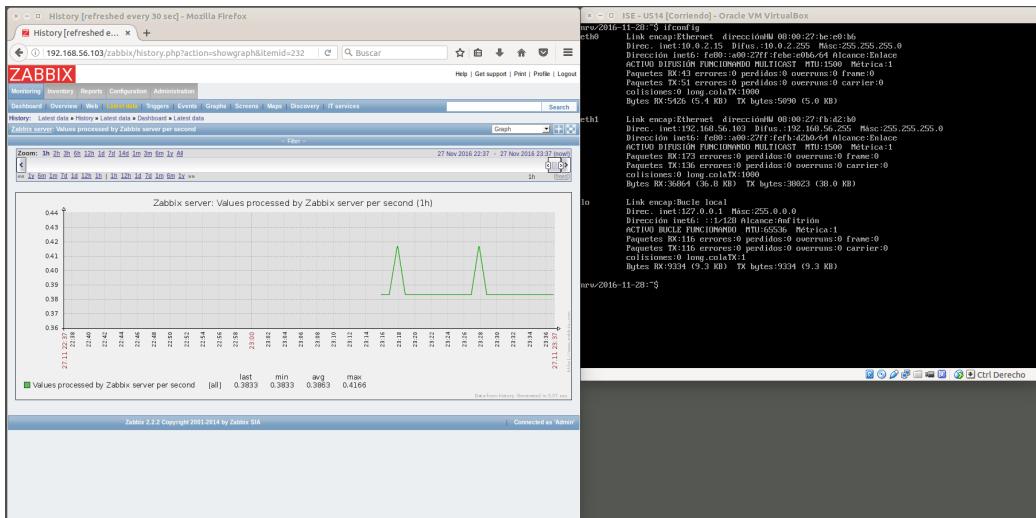
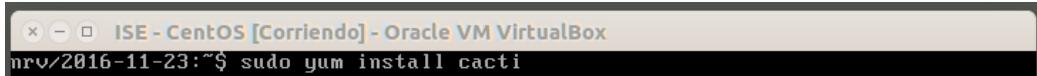


Figura 3.3.18: Valores procesados por el servidor *Zabbix* por segundo.

3.4. Cuestión opcional 5: Pruebe a instalar este monitor en alguno de sus tres sistemas. Realice capturas de pantalla del proceso de instalación y comente capturas de pantalla del programa en ejecución.

La instalación voy a hacerla en CentOS. Siguiendo la documentación de la página oficial de Cacti [16], debemos ejecutar el comando *sudo yum install cacti*, como podemos ver en la figura 3.4.1.



```
nrv/2016-11-23:~$ sudo yum install cacti
```

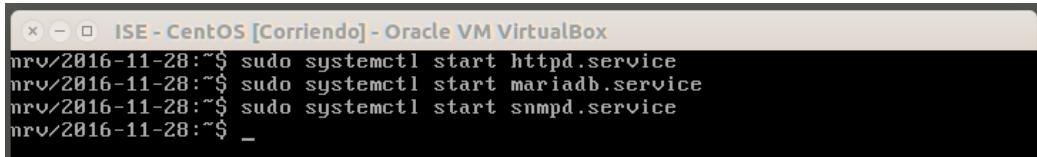
Figura 3.4.1: Instalación de Cacti.

Cómo podemos ver en la documentación de Cacti [17], para su correcto funcionamiento necesitamos que los siguientes paquetes esté instalados:

- httpd
- php
- php-mysql
- php-snmp
- mysql
- mysql-server ²
- net-snmp

La mayoría de estos paquetes los hemos ido instalando a lo largo de las prácticas y los que no, ya vienen instalados por defecto en CentOS. Una vez está todo instalado, iniciamos los servicios. Como podemos ver en la figura 3.4.2, ejecutamos:

- systemctl start httpd.service
- systemctl start mariadb.service
- systemctl start snmpd.service



```
nrv/2016-11-28:~$ sudo systemctl start httpd.service
nrv/2016-11-28:~$ sudo systemctl start mariadb.service
nrv/2016-11-28:~$ sudo systemctl start snmpd.service
nrv/2016-11-28:~$ _
```

Figura 3.4.2: Iniciamos los servicios.

A continuación, debemos configurar las bases de datos de MySQL (MariaDB en mi caso) tal y como nos indican en la documentación de Cacti [18].

²Dado que lo estoy haciendo en CentOS, yo uso *mariadb*, como ya explique en prácticas anteriores.

Lo primero que hacemos es entrar a MariaDB ejecutando `mysql -u root -p`. Una vez dentro, como podemos ver en la figura 3.4.3, realizamos los siguientes pasos:

1. Creamos la base de datos para *Cacti* ejecutando `create database cacti;`
2. Concedemos permisos sobre esa base de datos y creamos un usuario ejecutando `grant all on cacti.* to 'cacti'@'localhost' identified by 'practicas,ISE';`
3. Actualizamos los permisos ejecutando `flush privileges;`
4. Salimos de MariaDB ejecutando `exit;`

```

nrv/2016-11-28:~$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 713
Server version: 5.5.50-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database cacti;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> grant all on cacti.* to 'cacti'@'localhost' identified by 'practicas,ISE';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> exit;
Bye
nrv/2016-11-28:~$ 

```

Figura 3.4.3: Creación base de datos para *Cacti*.

Importamos las tablas de *Cacti* ejecutando `mysql -u cacti -p cacti < /usr/share/doc/cacti-0.8.8b/cacti.sql`, como podemos ver en la figura 3.4.4.

```

nrv/2016-11-28:~$ mysql -u cacti -p cacti < /usr/share/doc/cacti-0.8.8b/cacti.sql
Enter password:
nrv/2016-11-28:~$ 

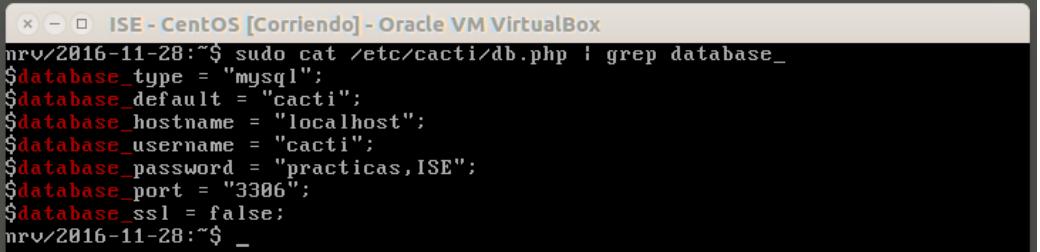
```

Figura 3.4.4: Importación de las tablas para *Cacti*.

A continuación, debemos editar el archivo `/etc/cacti/db.php` y cambiar los parámetros que podemos ver a continuación para que queden de la siguiente manera:

- `$database_type = "mysql";`
- `$database_default = "cacti";`
- `$database_hostname = "localhost";`
- `$database_username = "cacti";`
- `$database_password = "practicas,ISE";`

El resultado de modificar dicho archivo lo podemos ver en la figura 3.4.5.

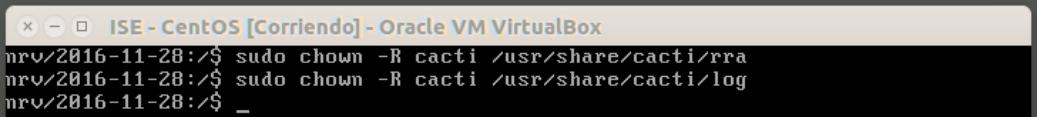


```
nrv/2016-11-28:~$ sudo cat /etc/cacti/db.php | grep database_
$database_type = "mysql";
$database_default = "cacti";
$database_hostname = "localhost";
$database_username = "cacti";
$database_password = "practicas,ISE";
$database_port = "3306";
$database_ssl = false;
nrv/2016-11-28:~$ _
```

Figura 3.4.5: Parámetros modificados.

Asignamos los permisos a `/usr/share/cacti/rra` y a `/usr/share/cacti/log` correctamente. Para ello, como podemos ver en la figura 3.4.6 ejecutamos:

- `chown -R cacti /usr/share/cacti/rra/`
- `chown -R cacti /usr/share/cacti/log/`



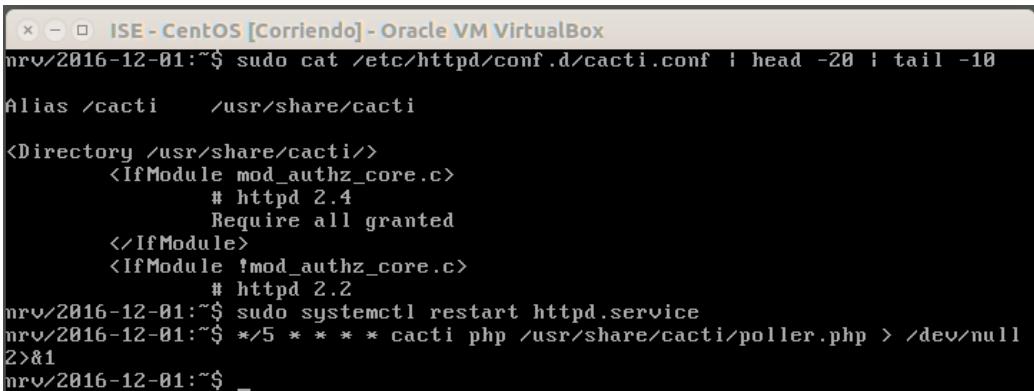
```
nrv/2016-11-28:~$ sudo chown -R cacti /usr/share/cacti/rra
nrv/2016-11-28:~$ sudo chown -R cacti /usr/share/cacti/log
nrv/2016-11-28:~$ _
```

Figura 3.4.6: Permisos modificados.

A continuación debemos editar el fichero de configuración de *Apache* en función de la versión que tenemos. En CentOS, tengo la versión 2.4.6³, por lo

³Para ver la versión que tenemos instalada debemos ejecutar `httpd -v`

tanto debemos modificar la parte correspondiente a versiones 2.4 de *Apache*. Debemos permitir conexiones desde el exterior de nuestro servidor, por eso cambiamos la línea que hay debajo de la línea `# httpd 2.4` por `Require all granted`. Una vez hecho el cambio, reiniciamos *Apache* ejecutando `sudo systemctl restart httpd.service`. Por último editamos nuestro fichero *crontab* ejecutando `* /5 * * * * cacti php /usr/share/cacti/poller.php > /dev/null 2>&1`. Este proceso lo podemos ver en la figura 3.4.7.



```

ISE - CentOS [Corriendo] - Oracle VM VirtualBox
nrv/2016-12-01:~$ sudo cat /etc/httpd/conf.d/cacti.conf | head -20 | tail -10
Alias /cacti      /usr/share/cacti

<Directory /usr/share/cacti/>
    <IfModule mod_authz_core.c>
        # httpd 2.4
        Require all granted
    </IfModule>
    <IfModule !mod_authz_core.c>
        # httpd 2.2
nrv/2016-12-01:~$ sudo systemctl restart httpd.service
nrv/2016-12-01:~$ */5 * * * * cacti php /usr/share/cacti/poller.php > /dev/null
2>&1
nrv/2016-12-01:~$ _

```

Figura 3.4.7: Archivo de configuración de *Apache* y archivo *crontab* modificados.

Finalmente, para acceder a *cacti* desde nuestra máquina anfitriona, debemos introducir la dirección IP de nuestro servidor seguido de `/cacti`. En mi caso, la dirección IP de mi servidor es `192.168.56.101` (máquinas conectadas en modo *host-only*), así que debemos escribir `192.168.56.101/cacti`, como podemos ver en la figura 3.4.8.

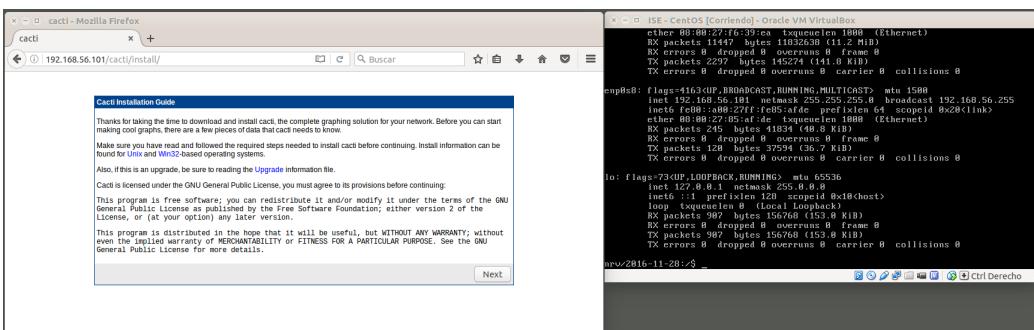


Figura 3.4.8: *Cacti* en funcionamiento.

Desde el navegador, debemos instalar cacti. Para ello, pulsamos el botón de *Next*. En la pantalla que podemos ver en la figura 3.4.9, elegimos *New*

Install y le damos a *Next*. En la siguiente ventana, que podemos ver en la figura 3.4.10, le damos a *Finish*. Finalmente, podemos ver la pantalla de inicio de sesión, como se ve en la figura 3.4.11.

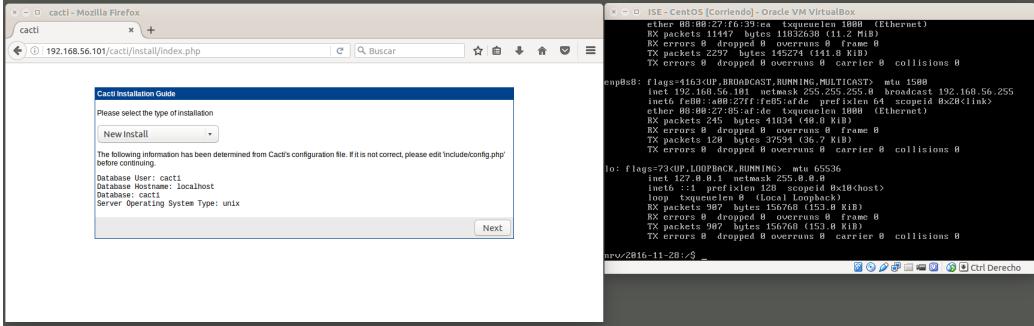


Figura 3.4.9: Instalación nueva.

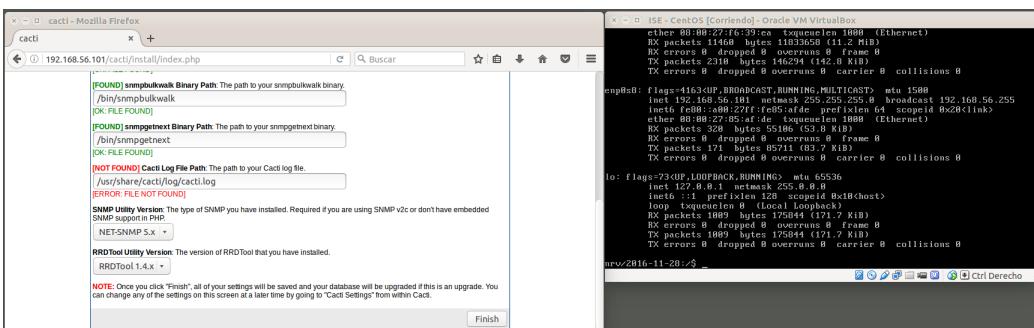


Figura 3.4.10: Finalización de la instalación.

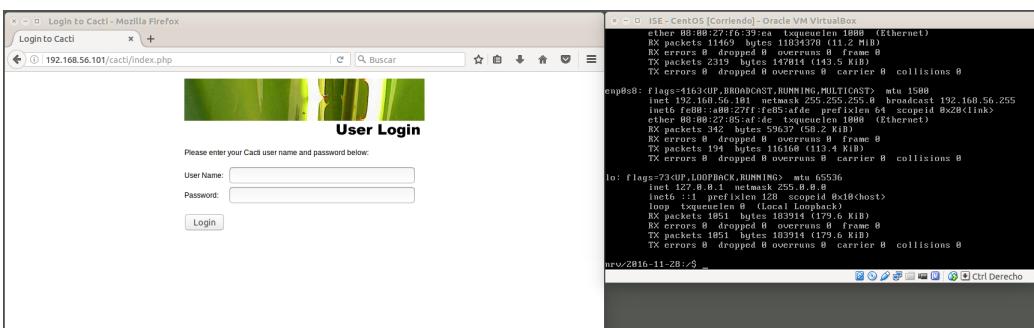


Figura 3.4.11: Instalación de *Cacti* completada.

En la página que vemos en la figura 3.4.11 accedemos poniendo como usuario y contraseña *admin* y *admin* correspondientemente. Una vez intro-

ducida los datos, se nos obliga a cambiar la contraseña, como podemos ver en la figura 3.4.12. En mi caso, he introducido la de siempre: *practicas,ISE*.

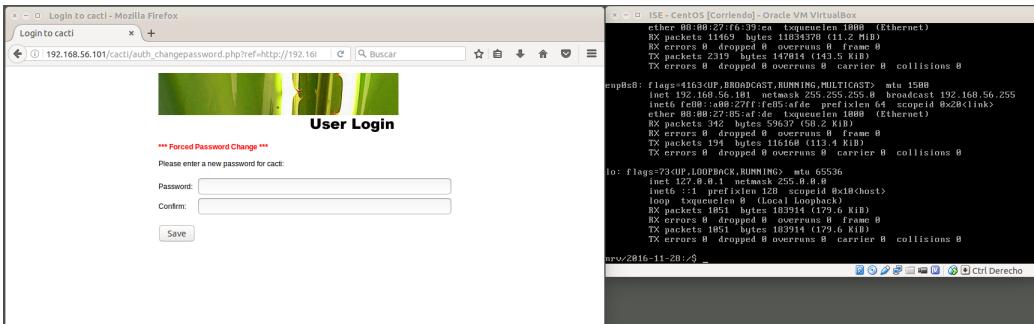


Figura 3.4.12: Cambio forzado de contraseña.

Una vez instalado hemos instalado *Cacti*, seleccionamos el gráfico que queremos añadir al árbol de gráfico y lo añadimos, tal y como podemos ver en la figura 3.4.13. Para ver el gráfico, pinchamos sobre la opción de *graphs* que podemos ver en la parte superior de *cacti*. Podemos ver los datos en la figura 3.4.14.

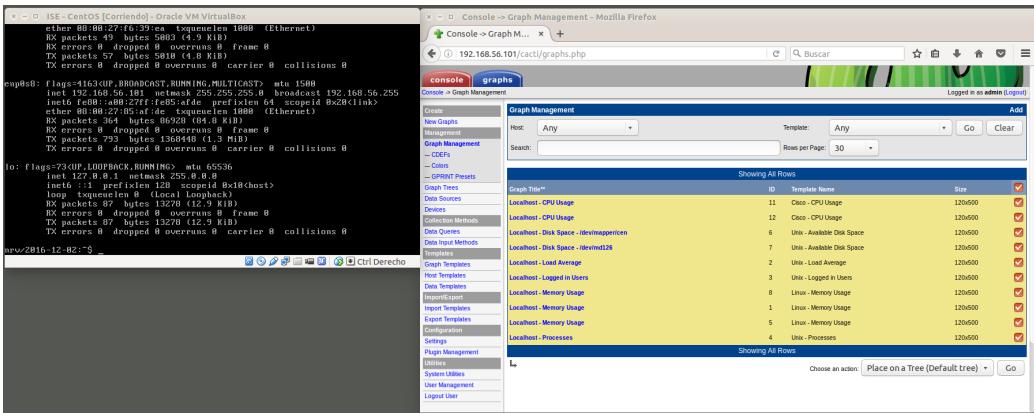


Figura 3.4.13: Añadimos el gráfico al árbol.

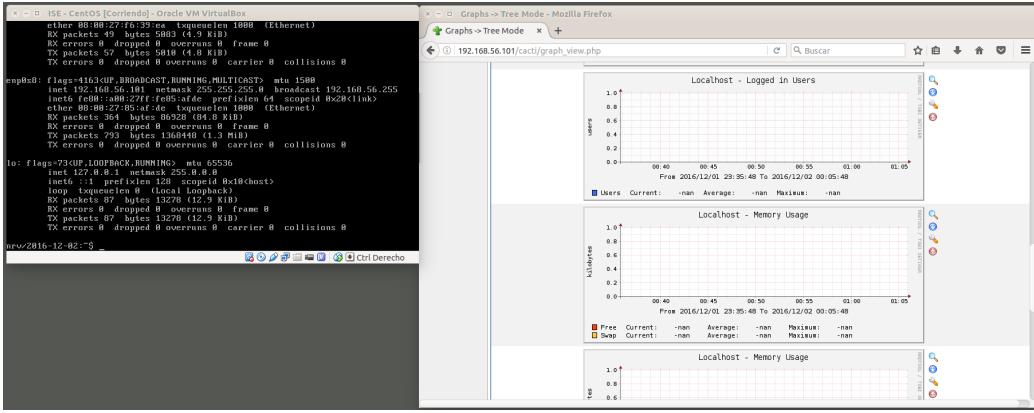


Figura 3.4.14: Datos recogidos.

Los gráficos que podemos ver en la figura 3.4.14 representan la carga media del servidor y número de usuarios identificados en el sistema. En ambos casos parece que no hay datos ya que no se ha producido nada en el servidor en el intervalo de tiempo seleccionado.

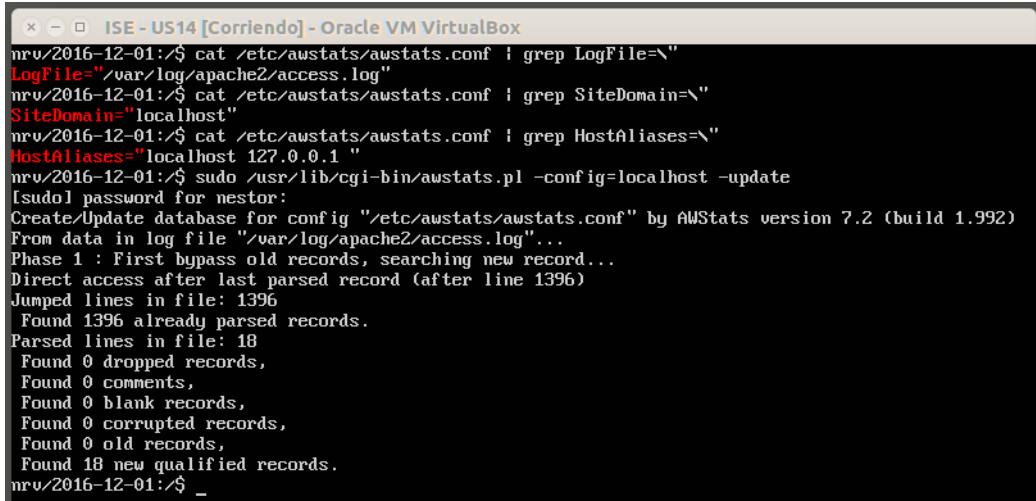
3.5. Cuestión opcional 6: Instale el monitor. Muestre y comente algunas capturas de pantalla.

Para instalar *AWStats* ejecutamos *sudo apt-get install awstats*. Una vez lo hemos instalado, como podemos ver en la *Community Help Wiki* de Ubuntu [19], si quisiésemos monitorizar varios dominios, debemos crear un archivo de configuración para cada uno. En mi caso sólo voy a monitorizar el localhost, así que con modificar el archivo que tenemos es suficiente. Debemos editar el fichero cambiando los parámetros que vemos a continuación para que reflejen la información correcta. En mi caso, el valor de los parámetros es el siguiente:

- LogFile=“/var/log/apache2/access.log”
- SiteDomain=“localhost”
- HostAliases=“localhost 127.0.0.1”

El resultado de la modificación de dicho archivo lo podemos ver en la figura 3.5.1. Una vez editado el fichero, creamos las estadísticas iniciales para

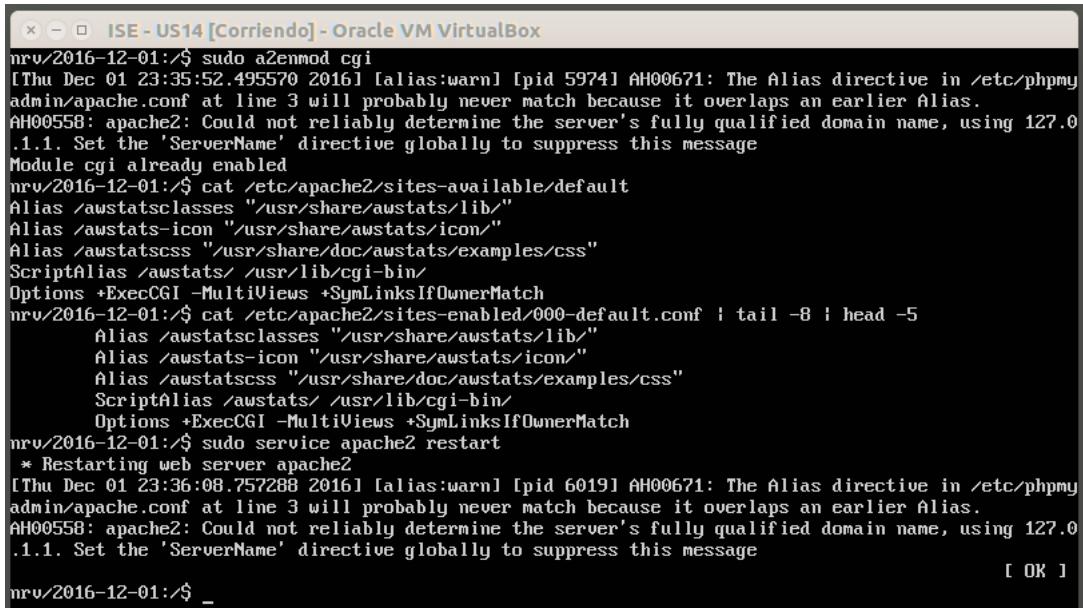
AWStats. Para ello ejecutamos `sudo /usr/lib/cgi-bin/awstats.pl -config=localhost -update`, tal y como se puede ver en la figura 3.5.1.



```
nrv/2016-12-01:~$ cat /etc/awstats/awstats.conf | grep LogFile="LogFile=\"/var/log/apache2/access.log\""
nrv/2016-12-01:~$ cat /etc/awstats/awstats.conf | grep SiteDomain="SiteDomain=\"localhost\""
nrv/2016-12-01:~$ cat /etc/awstats/awstats.conf | grep HostAliases="HostAliases=\"localhost 127.0.0.1\""
nrv/2016-12-01:~$ sudo /usr/lib/cgi-bin/awstats.pl -config=localhost -update
[sudo] password for nestor:
Create/Update database for config "/etc/awstats/awstats.conf" by AWStats version 7.2 (build 1.992)
From data in log file "/var/log/apache2/access.log"...
Phase 1 : First bypass old records, searching new record...
Direct access after last parsed record (after line 1396)
Jumped lines in file: 1396
Found 1396 already parsed records.
Parsed lines in file: 18
Found 0 dropped records,
Found 0 comments,
Found 0 blank records,
Found 0 corrupted records,
Found 0 old records,
Found 18 new qualified records.
nrv/2016-12-01:~$ _
```

Figura 3.5.1: Configuración de *AWStats*.

Una vez hemos configurado *AWStats*, pasamos a configurar *Apache*. Lo primero que debemos hacer indicarle a *Apache* que use *mod_cgi* ejecutando `sudo a2enmod cgi`. Dado que no tenemos ningún *Host Virtual*, creamos el archivo `/etc/apache2/sites-available/default` y añadimos las líneas que podemos ver en la figura 3.5.2. También debemos añadir estas líneas al fichero `/etc/apache2/sites-enabled/000-default.conf` y reiniciamos *Apache* ejecutando `sudo service apache2 restart`. Este proceso lo podemos ver en la figura 3.5.2



```

ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-12-01:/$ sudo a2enmod cgi
[Thu Dec 01 23:35:52.495570 2016] [alias:warn] [pid 5974] AH00671: The Alias directive in /etc/phpmyadmin/apache.conf at line 3 will probably never match because it overlaps an earlier Alias.
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Module cgi already enabled
nrv/2016-12-01:/$ cat /etc/apache2/sites-available/default
Alias /awstatsclasses "/usr/share/awstats/lib/"
Alias /awstats-icon "/usr/share/awstats/icon/"
Alias /awstatscss "/usr/share/doc/awstats/examples/css"
ScriptAlias /awstats/ /usr/lib/cgi-bin/
Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
nrv/2016-12-01:/$ cat /etc/apache2/sites-enabled/000-default.conf | tail -8 | head -5
    Alias /awstatsclasses "/usr/share/awstats/lib/"
    Alias /awstats-icon "/usr/share/awstats/icon/"
    Alias /awstatscss "/usr/share/doc/awstats/examples/css"
    ScriptAlias /awstats/ /usr/lib/cgi-bin/
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
nrv/2016-12-01:/$ sudo service apache2 restart
 * Restarting web server apache2
[Thu Dec 01 23:36:08.757288 2016] [alias:warn] [pid 6019] AH00671: The Alias directive in /etc/phpmyadmin/apache.conf at line 3 will probably never match because it overlaps an earlier Alias.
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
[ OK ]
nrv/2016-12-01:/$ _

```

Figura 3.5.2: Configuración de *Apache*

Por último, modificamos nuestro archivo crontab para actualizar *AWStats* cada minuto. Para ello ejecutamos `* * * * * /usr/lib/cgi-bin/awstats.pl -config=localhost -update`, como podemos ver en la figura 3.5.3.

Para acceder a *AWStats*, debemos introducir en el navegador de nuestra máquina anfitriona `http://<IP del servidor>/awstats/awstats.pl?`. En mi caso, como podemos ver en la figura 3.5.4, la dirección IP de mi servidor es `19.168.56.101` (máquinas conectadas en modo *host-only*) así que debemos introducir `http://192.168.56.101/awstats/awstats.pl?`. *AWStats* nos muestra información como puede ser el número de visitas o el ancho de banda. Toda esta información se puede ver representada en gráficas, existiendo una gráfica para cada mes. En mi caso, podemos ver que la actividad empieza en Octubre, que fue cuando se instaló el servidor web para las prácticas de la asignatura.

```
x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
GNU nano 2.2.6                               Archivo: /tmp/crontab.QXFz79/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
# 00 11 * * * ~/script.sh

* * * * * /usr/lib/cgi-bin/awstats.pl -config=localhost -update
```

Figura 3.5.3: Fichero de *crontab* modificado.

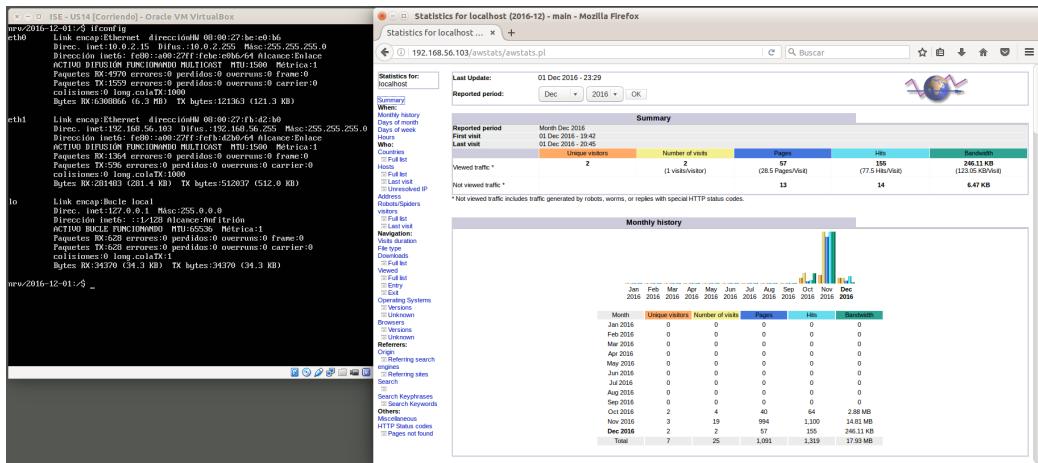


Figura 3.5.4: Información proporcionada por *AWStats*

3.6. Cuestión opcional 10: Escriba un script en PowerShell y analice su comportamiento usando el profiler presentado.

El script que voy a hacer en PowerShell es el script que hice para la cuestión 8 de la práctica 3 en Python. El resultado lo podemos ver en el script ⁴ 3.6.

```
$suma=0

for ($i=0; $i -le 100; $i++){
    If ($i % 2 -ne 0) {
        $i
        $suma += $i
    }
}

‘‘La suma es: ’’ + $suma
```

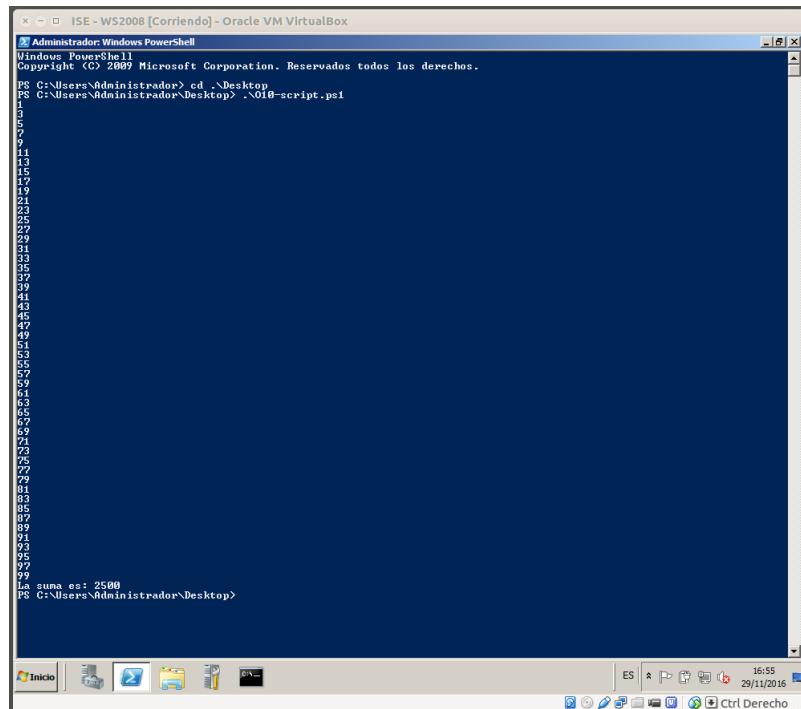


Figura 3.6.1: Resultado del script 3.6.

⁴El script “O10-script.ps1” se encuentra dentro de la carpeta *Archivos auxiliares*.

Para usar el profiler, debemos descargarlo de la página de Microsoft [20]. Una vez descargado, para ejecutarlo he tenido que instalar .NET Framework [21]. Tras descargarlo, lo ejecutamos. Pero, tal y como hable con usted en tutoría el día 1 de Diciembre, da error. El error lo podemos ver en la figura 3.6.2.

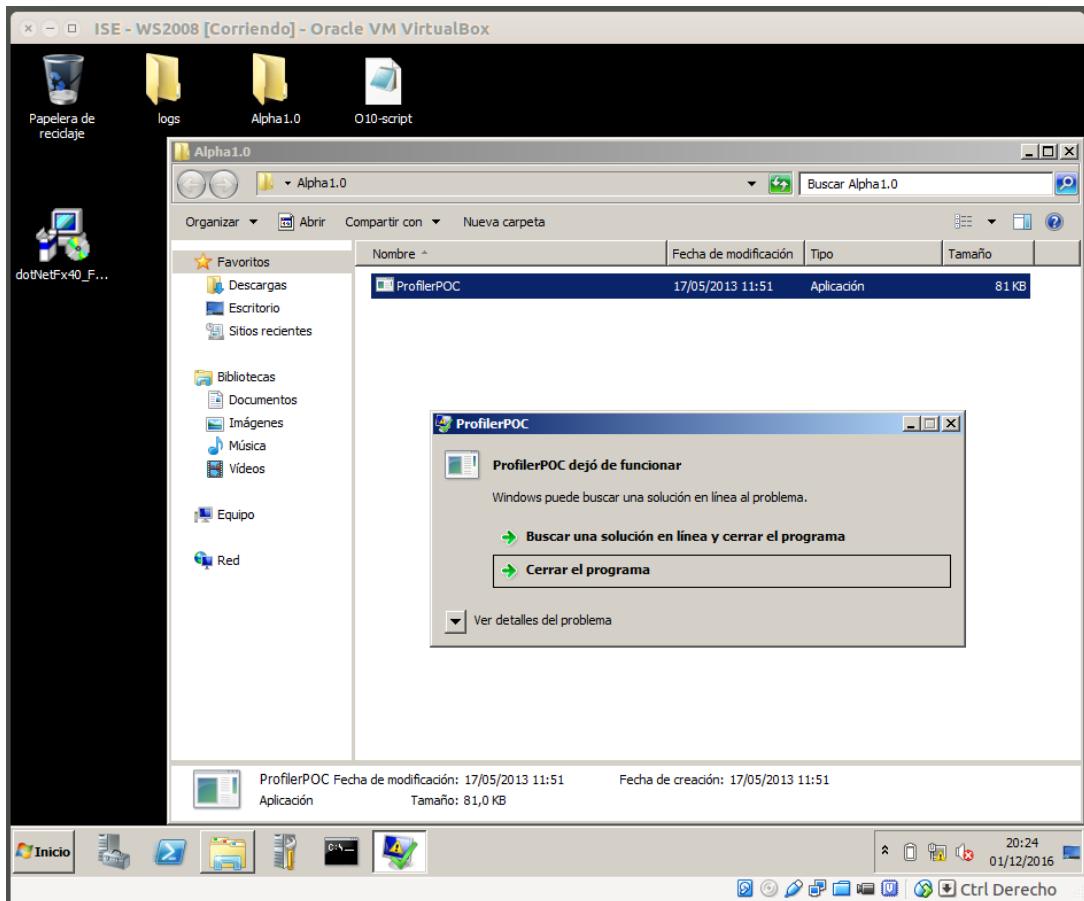


Figura 3.6.2: Error al ejecutar el profiler.

El resultado que veríamos es el siguiente:

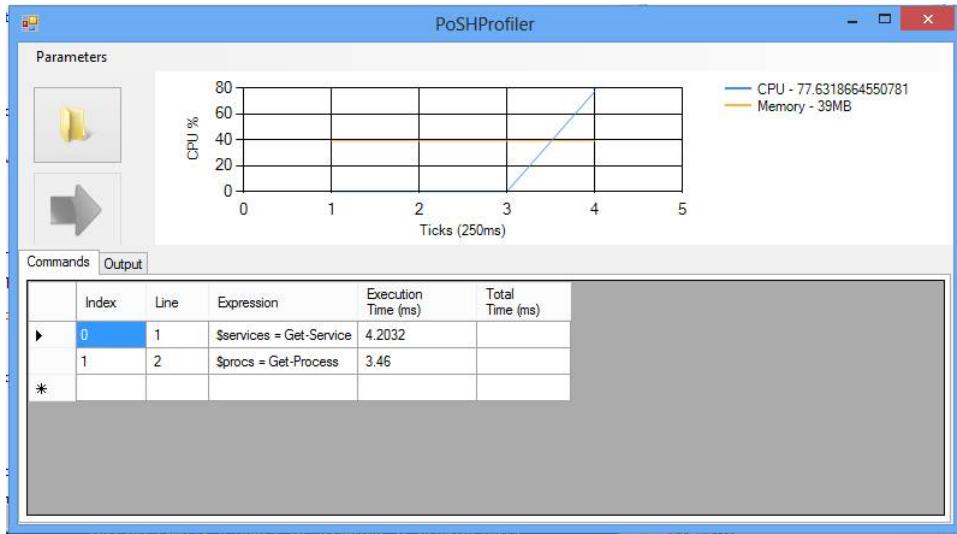


Figura 3.6.3: Ejemplo de ejecución de PoshProfiler [1].

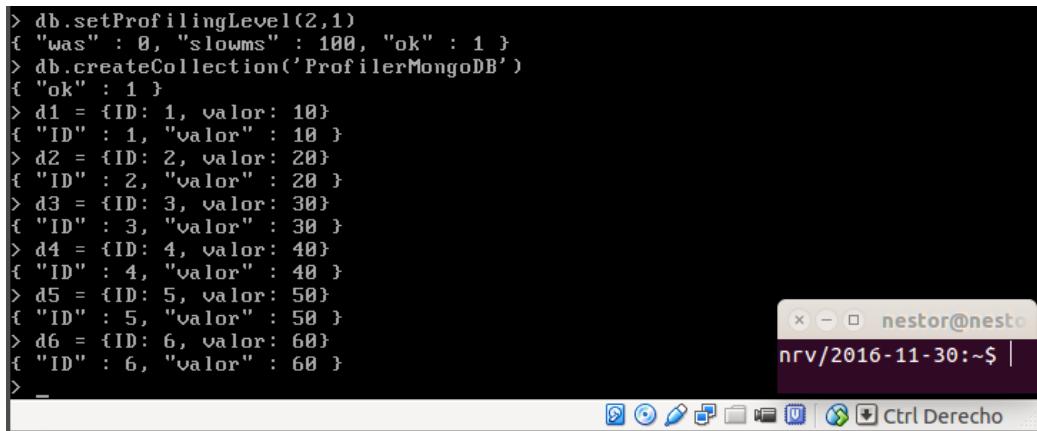
3.7. Cuestión opcional 11: Al igual que ha realizado el “profiling” con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).

Para ver como usar el profiler de *MongoDB* he usado la página oficial de *MongoDB* [22]. Dentro de *MongoDB* ejecutamos `db.setProfilingLevel(2, 1)`, como podemos ver en la figura 3.7.1, para activar el profiler en el segundo nivel (para “profilear” todas las operaciones) y poner como umbral para determinar cuando una consulta se considera lenta en 1ms. Lo primero que voy a hacer es crear la colección con los documentos que voy a usar, para ello ejecutamos las siguientes instrucciones:

1. Creamos la colección que vamos a usar, *ProfilerMongoDB*, para ello ejecutamos
`db.createCollection('ProfilerMongoDB')`, como podemos ver en la figura 3.7.1.
2. Creamos seis documentos para luego insertarlos en la colección que

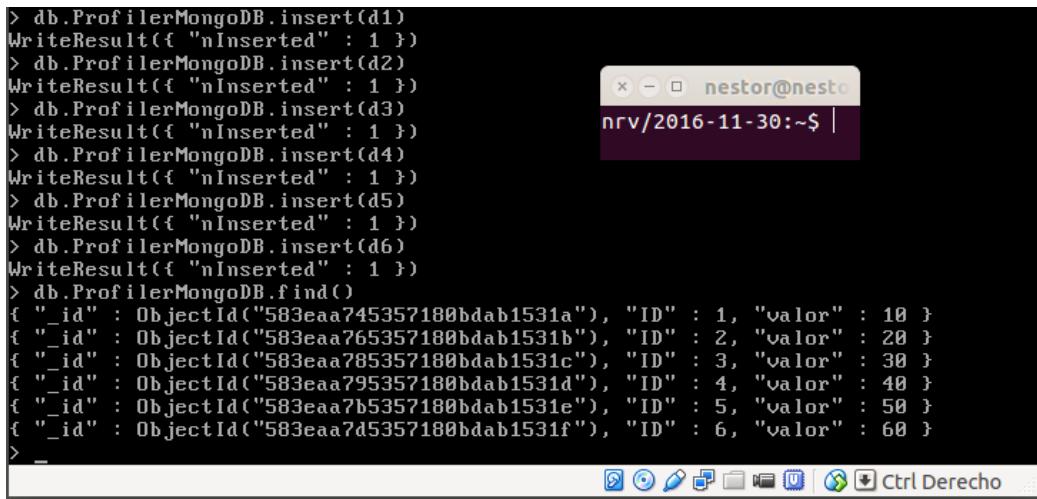
hemos creado en el paso anterior. Como podemos ver en la figura 3.7.1 ejecutamos:

- a) $d1 = \{ ID: 1, valor: 10 \}$
 - b) $d2 = \{ ID: 2, valor: 20 \}$
 - c) $d3 = \{ ID: 3, valor: 30 \}$
 - d) $d4 = \{ ID: 4, valor: 40 \}$
 - e) $d5 = \{ ID: 5, valor: 50 \}$
 - f) $d6 = \{ ID: 6, valor: 60 \}$
3. Insertamos los documentos en la colección *ProfilerMongoDB*. Como podemos ver en la figura 3.7.2 ejecutamos:
- a) $db.ProfilerMongoDB.insert(d1)$
 - b) $db.ProfilerMongoDB.insert(d2)$
 - c) $db.ProfilerMongoDB.insert(d3)$
 - d) $db.ProfilerMongoDB.insert(d4)$
 - e) $db.ProfilerMongoDB.insert(d5)$
 - f) $db.ProfilerMongoDB.insert(d6)$
4. Realizamos la consulta ejecutando $db.ProfilerMongoDB.find()$, como podemos ver en la figura 3.7.2.



```
> db.setProfilingLevel(2,1)
{ "was" : 0, "slowms" : 100, "ok" : 1 }
> db.createCollection('ProfilerMongoDB')
{ "ok" : 1 }
> d1 = {ID: 1, valor: 10}
{ "ID" : 1, "valor" : 10 }
> d2 = {ID: 2, valor: 20}
{ "ID" : 2, "valor" : 20 }
> d3 = {ID: 3, valor: 30}
{ "ID" : 3, "valor" : 30 }
> d4 = {ID: 4, valor: 40}
{ "ID" : 4, "valor" : 40 }
> d5 = {ID: 5, valor: 50}
{ "ID" : 5, "valor" : 50 }
> d6 = {ID: 6, valor: 60}
{ "ID" : 6, "valor" : 60 }
> _
```

Figura 3.7.1: Creación de la colección y de los documentos.



```

> db.ProfilerMongoDB.insert(d1)
WriteResult({ "nInserted" : 1 })
> db.ProfilerMongoDB.insert(d2)
WriteResult({ "nInserted" : 1 })
> db.ProfilerMongoDB.insert(d3)
WriteResult({ "nInserted" : 1 })
> db.ProfilerMongoDB.insert(d4)
WriteResult({ "nInserted" : 1 })
> db.ProfilerMongoDB.insert(d5)
WriteResult({ "nInserted" : 1 })
> db.ProfilerMongoDB.insert(d6)
WriteResult({ "nInserted" : 1 })
> db.ProfilerMongoDB.find()
{ "_id" : ObjectId("583eaa745357180bdab1531a"), "ID" : 1, "valor" : 10 }
{ "_id" : ObjectId("583eaa765357180bdab1531b"), "ID" : 2, "valor" : 20 }
{ "_id" : ObjectId("583eaa785357180bdab1531c"), "ID" : 3, "valor" : 30 }
{ "_id" : ObjectId("583eaa795357180bdab1531d"), "ID" : 4, "valor" : 40 }
{ "_id" : ObjectId("583eaa7b5357180bdab1531e"), "ID" : 5, "valor" : 50 }
{ "_id" : ObjectId("583eaa7d5357180bdab1531f"), "ID" : 6, "valor" : 60 }
> -

```

Figura 3.7.2: Inserción y consulta de los documentos.

Una vez hecha la consulta, vamos a ver los datos recogidos por el profiler, para ello ejecutamos `db.system.profile.find().limit(1)`. La consulta anterior nos devolvería la entrada más reciente realizada sobre la base de datos. Como podemos ver en la figura 3.7.3, la opción `command` nos indica que dicha entrada corresponde a la creación de la colección. También podemos ver más información. Para ello ejecutamos `db.system.profile.find().limit(8)`. La consulta anterior nos devolvería las 8 entradas más recientes realizadas sobre la base de datos. ¿Por qué 8? En el proceso visto anteriormente 3.7 se ejecutaban 8 comandos sobre la base de datos ya que los comandos para la creación de los documentos no cuentan porque no son sobre la base de datos; uno para crear la colección, seis para insertar los documentos y uno para la consulta. Parte de los datos los podemos ver en la figura 3.7.4. En este caso podemos ver información sobre la inserción del último documento y sobre la recuperación de los documentos de la colección. En la información acerca del `insert` podemos ver el contenido del documento que se está insertando y en qué base se está insertando. En la información acerca del `find` podemos ver que el atributo `nReturned` es 6. Este valor indica el número de elementos de la colección que han sido devueltos.

```

> db.system.profile.find().limit(1)
{
  "op" : "command",
  "ns" : "test.ProfilerMongoDB",
  "command" : {
    "create" : "ProfilerMongoDB"
  },
  "keyUpdates" : 0,
  "writeConflicts" : 0,
  "numYield" : 0,
  "locks" : {
    "Global" : {
      "acquireCount" : {
        "r" : NumberLong(1),
        "w" : NumberLong(1)
      }
    },
    "Database" : {
      "acquireCount" : {
        "W" : NumberLong(1)
      }
    }
  },
  "responseLength" : 22,
  "protocol" : "op_command",
  "millis" : 61,
  "execStats" : {},
  "ts" : ISODate("2016-11-30T10:29:36.369Z"),
  "client" : "127.0.0.1",
  "allUsers" : [],
  "user" : ""
}
> _

```

nestor@nest: ~ | nrv/2016-11-30:~\$ |

Figura 3.7.3: Entrada más reciente.

```

> db.system.profile.find().limit(8)
{
  "op" : "command",
  "ns" : "test.ProfilerMongoDB",
  "command" : {
    "create" : "ProfilerMongoDB"
  },
  "keyUpdates" : 0,
  "writeConflicts" : 0,
  "numYield" : 0,
  "locks" : {
    "Global" : {
      "acquireCount" : {
        "r" : NumberLong(1),
        "w" : NumberLong(1)
      }
    },
    "Database" : {
      "acquireCount" : {
        "W" : NumberLong(1)
      }
    }
  },
  "responseLength" : 25,
  "protocol" : "op_command",
  "millis" : 0,
  "execStats" : {},
  "ts" : ISODate("2016-11-30T10:31:23.667Z"),
  "client" : "127.0.0.1",
  "allUsers" : [],
  "user" : ""
},
{
  "op" : "insert",
  "ns" : "test.ProfilerMongoDB",
  "query" : {
    "insert" : "ProfilerMongoDB"
  },
  "documents" : [
    {
      "_id" : ObjectId("583eaa7d5357180bdab1531f"),
      "ID" : 6,
      "valor" : 60
    }
  ],
  "ordered" : true,
  "nInserted" : 1,
  "keyUpdates" : 0,
  "writeConflicts" : 0,
  "numYield" : 0,
  "locks" : {
    "Global" : {
      "acquireCount" : {
        "r" : NumberLong(1),
        "w" : NumberLong(1)
      }
    },
    "Database" : {
      "acquireCount" : {
        "W" : NumberLong(1)
      }
    }
  },
  "Collection" : {
    "acquireCount" : {
      "w" : NumberLong(1)
    }
  },
  "responseLength" : 25,
  "protocol" : "op_command",
  "millis" : 0,
  "execStats" : {},
  "ts" : ISODate("2016-11-30T10:31:25.410Z"),
  "client" : "127.0.0.1",
  "allUsers" : [],
  "user" : ""
},
{
  "op" : "query",
  "ns" : "test.ProfilerMongoDB",
  "query" : {
    "find" : "ProfilerMongoDB"
  },
  "filter" : {},
  "keysExamined" : 0,
  "docsExamined" : 6,
  "cursorExhausted" : true,
  "keyUpdates" : 0,
  "writeConflicts" : 0,
  "numYield" : 0,
  "locks" : {
    "Global" : {
      "acquireCount" : {
        "r" : NumberLong(2)
      }
    },
    "Database" : {
      "acquireCount" : {
        "r" : NumberLong(1)
      }
    }
  },
  "Collection" : {
    "acquireCount" : {
      "r" : NumberLong(1)
    }
  },
  "nReturned" : 6,
  "responseLength" : 423,
  "protocol" : "op_command",
  "millis" : 0,
  "execStats" : {
    "stage" : "COLLSCAN",
    "filter" : {
      "$and" : [
        {}
      ]
    },
    "nReturned" : 6,
    "executionTimeMillisEstimate" : 0,
    "works" : 8,
    "advanced" : 6,
    "needTime" : 1,
    "needYield" : 0,
    "saveState" : 0,
    "restoreState" : 0,
    "isEOF" : 1,
    "invalidates" : 0,
    "direction" : "forward",
    "docsExamined" : 6
  },
  "ts" : ISODate("2016-11-30T10:31:37.153Z"),
  "client" : "127.0.0.1",
  "allUsers" : [],
  "user" : ""
}
> db.system.profile.find().limit(8)

```

nestor@nest: ~ | nrv/2016-11-30:~\$ |

Figura 3.7.4: 8 entradas más recientes.

Capítulo 4

Práctica 4.

4.1. Cuestión opcional 1: ¿Qué es Scala? Instale Gatling y pruebe los escenarios por defecto.

Para saber que es *Scala* no hay nada mejor que verlo en la página oficial [23]. *Scala* es un lenguaje de programación. *Scala* es el acrónimo de “*Scalable Language*”. Podemos usar *Scala* para programación orientada a objetos o para una programación más funcional. Se ejecuta sobre la máquina virtual de Java. Las clases de Java y *Scala* pueden combinarse sin ningún problema.

Como podemos ver en la página de *Gatling* [24], *Gatling* se basa en *Scala*, así que primero tenemos que instalar *Scala*, en mi caso en Ubuntu Server. Para ello ejecutamos `sudo apt install scala`, como podemos ver en la figura 4.1.1. Una vez instalado, pasamos a instalar *Gatling* siguiendo las instrucciones que podemos ver en la página oficial [25]. Debemos descargar un fichero zip y luego extraerlo. Una vez extraído, dentro de la carpeta de *Gatling* nos encontramos la carpeta *bin* y dentro de ella el fichero *gatling.sh*. Lo ejecutamos mediante el comando `./gatling.sh` como podemos ver en la figura 4.1.2. A continuación nos pedirá que escenario queremos usar como podemos ver en la figura 4.1.2, yo he elegido el 0. A continuación comenzará la prueba. El resultado lo podemos ver en la figura 4.1.3.

```

x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-12-14:~$ sudo apt install scala
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  libhawtjni-runtime-java libjansi-java libjansi-native-java scala-library
Paquetes sugeridos:
  scala-doc
Se instalarán los siguientes paquetes NUEVOS:
  libhawtjni-runtime-java libjansi-java libjansi-native-java scala
  scala-library
0 actualizados, 5 se instalarán, 0 para eliminar y 77 no actualizados.
Necesito descargar 21,6 MB de archivos.
Se utilizarán 26,0 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] _

```

Figura 4.1.1: Instalación de *Scala*.

```

x - ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrv/2016-12-14:~/gatling-charts-highcharts-bundle-2.2.3/bin$ ./gatling.sh
GATLING_HOME is set to /home/nestor/gatling-charts-highcharts-bundle-2.2.3
Choose a simulation number:
[0] computerdatabase.BasicSimulation
[1] computerdatabase.advanced.AdvancedSimulationStep01
[2] computerdatabase.advanced.AdvancedSimulationStep02
[3] computerdatabase.advanced.AdvancedSimulationStep03
[4] computerdatabase.advanced.AdvancedSimulationStep04
[5] computerdatabase.advanced.AdvancedSimulationStep05

```

Figura 4.1.2: Ejecución de *Gatling* y posibles escenarios.

```

Simulation computerdatabase.BasicSimulation completed in 23 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...
=====
----- Global Information -----
> request count                      13 (OK=13    KO=0    )
> min response time                  58 (OK=58    KO=--   )
> max response time                  224 (OK=224   KO=--   )
> mean response time                 93 (OK=93    KO=--   )
> std deviation                     46 (OK=46    KO=--   )
> response time 50th percentile      76 (OK=76    KO=--   )
> response time 75th percentile      80 (OK=80    KO=--   )
> response time 95th percentile      181 (OK=181   KO=--   )
> response time 99th percentile      215 (OK=215   KO=--   )
> mean requests/sec                 0.542 (OK=0.542 KO=--   )
----- Response Time Distribution -----
> t < 800 ms                         13 (100%)
> 800 ms < t < 1200 ms                0 ( 0%)
> t > 1200 ms                        0 ( 0%)
> failed                             0 ( 0%)
=====

Reports generated in 1s.
Please open the following file: /home/nestor/gatling-charts-highcharts-bundle-2.2.3/results/basicimulation-1481711704035/index.html
nrv/2016-12-14:~/gatling-charts-highcharts-bundle-2.2.3/bin$ _

```

Figura 4.1.3: Resultado de la ejecución de *Gatling*.

Para poder ver los resultados de una mejor forma, copiamos el contenido de la carpeta `/results/basicsimulation-1481711704035` a `/var/www/html`, para poder acceder desde el navegador de mi máquina anfitriona. Para ello, desde la carpeta `/results/basicsimulation-1481711704035` ejecutamos `cp -r */var/www/html`, como podemos ver en la figura 4.1.4. A continuación, accedemos desde la máquina anfitriona. Para ello introducimos la dirección IP de nuestro servidor seguido de `/index.html`. Como podemos ver en la figura 4.1.4 la dirección IP de mi servidor es `192.168.56.103` (máquinas conectas en modo *host-only*), así que debemos introducir `192.168.56.103/index.html`. Podemos ver los resultados en la figura 4.1.5. Como podemos ver, se han respondido todas las solicitudes de manera correcta y todas en menos de 800ms.

```

ISE - US14 [Corriendo] - Oracle VM VirtualBox
nrw/2016-12-14:~/gatling-charts-highcharts-bundle-2.2.3/results/basicsimulation-1481711704035$ sudo
cp -r * /var/www/html/
nrw/2016-12-14:~/gatling-charts-highcharts-bundle-2.2.3/results/basicsimulation-1481711704035$ ifconfig eth1
eth1      Link encap:Ethernet  direcciónHW 08:00:27:fb:d2:b0
          Direc. inet:192.168.56.103  Difus.:192.168.56.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:febf:d2b0/64  Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500  Métrica:1
          Paquetes RX:638 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:444 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatX:1000
          Bytes RX:120434 (120.4 KB)  TX bytes:347208 (347.2 KB)

nrw/2016-12-14:~/gatling-charts-highcharts-bundle-2.2.3/results/basicsimulation-1481711704035$ _

```

Figura 4.1.4: Copia de los resultados y dirección IP del servidor.

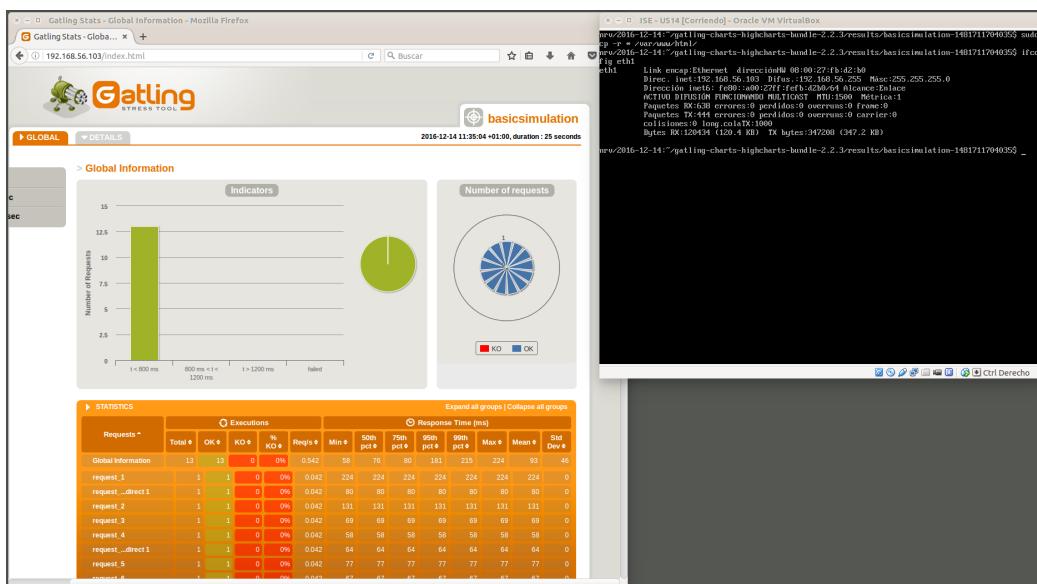


Figura 4.1.5: Resultado de *Gatling* desde el navegador.

Bibliografía

- [1] <https://poshprofiler.codeplex.com/>. Fecha de acceso: 01/12/2016.
- [2] https://raid.wiki.kernel.org/index.php/Detecting,_querying_and_testing#Simulating_a_drive_failure. Fecha: 2011 - Fecha de acceso: 15/10/2016.
- [3] https://www.gnu.org/software/bash/manual/html_node/The-Set-Builtin.html. Fecha de acceso: 12/10/2016.
- [4] <http://www.ibm.com/developerworks/ssa/linux/library/1-lpic1-v3-103-8/>. Fecha: 2010 - Fecha de acceso: 18/10/2016.
- [5] <http://ss64.com/bash/screen.html>. Fecha de acceso: 07/11/2016.
- [6] http://www.fail2ban.org/wiki/index.php/Main_Page. Fecha de acceso: 05/11/2016.
- [7] <https://www.digitalocean.com/community/tutorials/how-to-protect-ssh-with-fail2ban-on-ubuntu-14-04>. Año de publicación/última edición: 2014 - Fecha de acceso: 05/11/2016.
- [8] <http://rkhunter.sourceforge.net/>. Fecha de acceso: 06/11/2016.
- [9] <http://tomcat.apache.org/>. Fecha de acceso: 06/11/2016.
- [10] <https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html>. Año de publicación/última edición: 2016 - Fecha de acceso: 06/11/2016.
- [11] <https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-red-hat/>. Fecha de acceso: 30/11/2016.
- [12] https://raid.wiki.kernel.org/index.php/Detecting,_querying_and_testing. Fecha de acceso: 23/11/2016.

- [13] https://assets.nagios.com/downloads/nagiosxi/docs/XI_Manual_Installation_Instructions.pdf#_ga=1.242062663.413415278.1479885662. Fecha de acceso: 23/11/2016.
- [14] <https://library.nagios.com/library/products/nagios-xi/tutorials/nagios-xi-web-interface-setup-guide/>. Fecha de acceso: 23/11/2016.
- [15] <https://www.digitalocean.com/community/tutorials/how-to-install-zabbix-on-ubuntu-configure-it-to-monitor-multiple-vps-servers>. Fecha de acceso: 23/11/2016.
- [16] http://www.cacti.net/download_cacti.php. Fecha de acceso: 23/11/2016.
- [17] http://www.cacti.net/downloads/docs/html/install_unix.html. Fecha de acceso: 28/11/2016.
- [18] http://www.cacti.net/downloads/docs/html/unix_configure_cacti.html. Fecha de acceso: 28/11/2016.
- [19] <https://help.ubuntu.com/community/AWStats>. Año de publicación/última edición: 2015 - Fecha de acceso: 29/11/2016.
- [20] <https://gallery.technet.microsoft.com/Powershell-script-profiler-4382ffad>. Fecha de acceso: 29/11/2016.
- [21] <https://www.microsoft.com/es-es/download/details.aspx?id=17718>. Fecha de acceso: 29/11/2016.
- [22] <http://docs.mongodb.org/manual/tutorial/manage-the-database-profiler/>. Fecha de acceso: 30/11/2016.
- [23] <https://www.scala-lang.org/what-is-scala.html>. Fecha de acceso: 14/12/2016.
- [24] <http://gatling.io/#/>. Fecha de acceso: 14/12/2016.
- [25] <http://gatling.io/#/resources/download>. Fecha de acceso: 14/12/2016.