INSTALACIÓN Y CONFIGURACIÓN DE NAGIOS CORE 4.0.4

Rafael Romero González
Junio de 2015

Permiso de uso

Índice de contenido

1. INTRODUCCIÓN

En este documento, explicaremos como instalar y configurar un servidor Nagios Core, con los plugins necesarios corriendo sobre un Debian 7.8 desde cero.

¿Qué voy a aprender?
En esta guía aprenderás:

- Qué es Nagios.
- Preparación e instalación de Nagios en un equipo Linux.
- Instalación de plugins.
- Configuración de la interfaz web.
- Configuración e instalación del software necesario en equipos a monitorizar.
- Configuración, modificación y uso de los ficheros del servidor Nagios y de los clientes.
- Configuración de las notificaciones vía email.
- Uso de la interfaz web.

¿Qué es Nagios? ¿Para qué sirve?
Nagios es un sistema de monitorización de redes de código abierto ampliamente utilizado, que vigila los equipos (hardware) y *servicios* que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. Entre sus características principales figuran la monitorización de servicios de red (SMTP, POP3, HTTP, SNMP, etc.), la monitorización de los recursos de sistemas hardware (carga del procesador, uso de los discos, memoria, estado de los puertos, etc.), independencia de sistemas operativos, posibilidad de monitorización remota mediante túneles SSL cifrados o SSH, y la posibilidad de programar plugins específicos para nuevos sistemas.

Aclarar aquí que el término **servicio** se usa muy libremente en Nagios y puede referirse a servicios que corren en un host (POP, SMTP, HTTP…) o a algún otro tipo de métrica asociada a un host (respuesta a un ping, número de usuarios logueados en el sistema, espacio libre en un disco…).

1.1. Antes de empezar

Durante esta guía se utilizará la siguiente estructura a modo de ejemplo:

- Un Nagios Core 4.0.4 corriendo sobre un Debian 7.8 con IP 192.168.1.10/24.
- Un equipo Windows con NSClient++ 0.4.3.131 e IP 192.168.1.50/24.
- Un Debian 7.8 con IP 192.168.1.51/24.

Es importante que las direcciones IP sean fijas.

1.1.1. Instalando paquetes requeridos

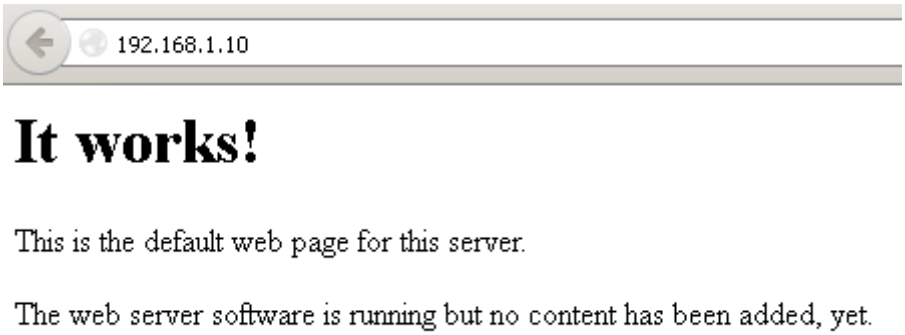
Los siguientes paquetes deben estar instalados antes de continuar:

- Apache 2
- PHP
- GCC compiler and development libraries
- GD development libraries

Instalación de apache2:

```
# apt-get install apache2
```

Podemos comprobar que la instalación fue correcta escribiendo en un navegador la dirección del equipo, en mi caso "http://192.168.1.10". Deberá apacer la siguiente página:



Instalación de PHP:

```
# apt-get install php5
```

Debemos asegurarnos que entre los paquetes instalados este presente "libapache2-mod-php5". De no ser así también deberemos instalarlo.

Instalación de las librerías necesarias:

```
# apt-get install make gcc g++ sudo libgd2-xpm libgd2-xpm-dev libpng12-dev
libgd-tools libpng3-dev
```

2. INSTALACIÓN DE NAGIOS

2.1. Creando un usuario y grupo

Como siempre en Linux, es mejor crear un usuario para cada servicio, de manera que si por algún motivo un atacante llegara a acceder al sistema a través de Nagios, solamente afectaría al usuario de Nagios y solo tendría los permisos de este usuario. Creamos el usuario:

```
# adduser nagios
```

A continuación nos pedirá unos datos, a nivel de información, se pueden dejar en blanco sin problemas (excepto la contraseña que la debemos de recordar).

Luego procedemos a crear el grupo, y le añadimos a los usuarios *nagios* y *www-data* (servidor web). Generalmente el grupo se crea al crear el usuario, y este pertenecerá al mismo. Pero aún así lo añadiremos, en el caso de que exista solo nos avisará.

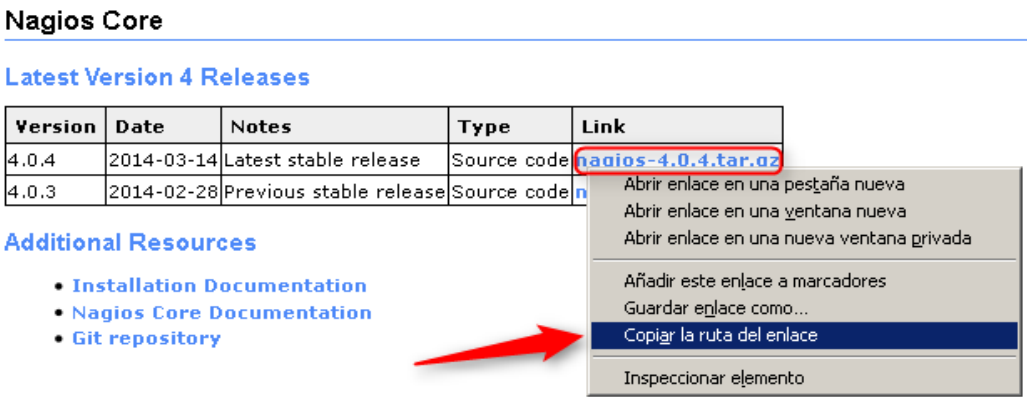
```
# groupadd nagios
# usermod -G nagios nagios
# usermod -G www-data,nagios www-data
```

2.2. Instalando Nagios Core

Primero entramos en la web de Nagios y buscamos la última versión de **Nagios Core**.

Así que entramos en aquí, donde encontraremos las versiones "Latest release candidate", "Latest stable release" y "Previous stable release".

Copiamos el enlace de la "**Latest stable release**".



Nos vamos al servidor y descargamos, descomprimimos e instalamos Nagios:

```
$ wget http://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.4/nagios-4.0.4.tar.gz
$ tar xvzf nagios-4.0.4.tar.gz
$ cd nagios-4.0.4/
$ ./configure --prefix=/usr/local/nagios --with-cgiurl=/nagios/cgi-bin --with-htmurl=/nagios/ --with-nagios-user=nagios --with-nagios-group=nagios -with-command-group=nagios
# make all
```

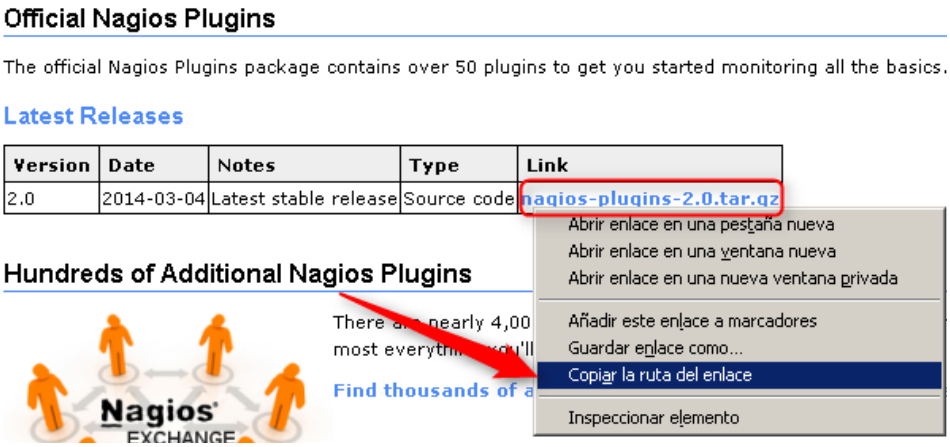
```
# make install
# make install-init
# make install-commandmode
# make install-config
$ cd ..
```

Una vez finalizado ya tenemos instalado Nagios Core.

2.3. Instalando Nagios Plugins

Nagios Plugins es un añadido que nos permite conectar con servicios más específicos en caso de querer conectar con un ordenador Windows y saber su carga de CPU, RAM, disco duro y otros servicios o dispositivos.

Volvemos a la web de Nagios, pero a su sección de plugins y copiamos el enlace de la última versión estable (Latest stable release).

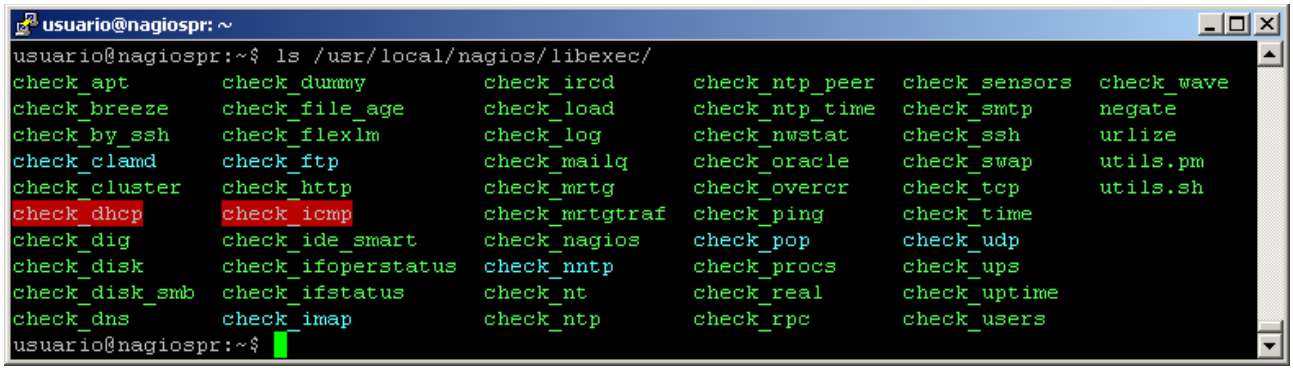


Luego, de vuelta a la consola los descargamos, descomprimimos e instalamos.

```
$ wget http://nagios-plugins.org/download/nagios-plugins-2.0.tar.gz
$ tar xvzf nagios-plugins-2.0.tar.gz
$ cd nagios-plugins-2.0/
$ ./configure
$ make && make install
```

Una vez terminado, comprobamos que se han instalado correctamente listando el directorio:

```
$ ls /usr/local/nagios/libexec/
```



2.4. Añadiendo Nagios al inicio del sistema

Para que Nagios se inicie con el sistema debemos activar el demonio usando las opciones y *runleves* por defecto del sistema. Así mismo tenemos también que crear un enlace simbólico. Para ello ejecutamos:

```
# update-rc.d -f nagios defaults 99
# ln -s /etc/init.d/nagios /etc/rcS.d/S99nagios
```

Si tenéis curiosidad sobre esto podéis buscar información sobre los **runlevel** de Linux.

A modo de explicación rápida sobre el ejemplo, comentar que el *rcS.d* corresponde al inicio del sistema bajo el modo de un solo usuario (single-user mode). Cuando el sistema se inicia, busca en este directorio scripts a ejecutar. Normalmente estos scripts se encuentran en `/etc/init.d` y simplemente se les crea un enlace a `/etc/rcS.d`, tal y como hemos hecho.

En cuanto a la terminología para el nombre:

- Los scripts que comienzan por "S" iniciarán el servicio, y los que empiecen por "K" lo pararán.
- El número que le sigue se utiliza para ejecutar los scripts por orden alfabético, así uno que empieza por "S08" se ejecutará antes que uno que empieza por "S99".
- Por último se usa un nombre identificativo para script, como "nagios".

3. CONFIGURACIÓN DE LA INTERFAZ WEB

Ya tenemos instalado Nagios, ahora nos queda configurar la interfaz web en la que poder observar la monitorización de los servicios. Para ello, crearemos un el sitio en *apache2*:

```
# nano /etc/apache2/sites-available/nagios
```

Dentro añadimos el siguiente texto (podéis copiar y pegar siempre y cuando no se modificasen las rutas durante la instalación. Si lo hacéis cuidado con las comillas.):

```
<VirtualHost *:80>
  DocumentRoot /usr/local/nagios/share
  ScriptAlias /nagios/cgi-bin /usr/local/nagios/sbin

  <Directory "/usr/local/nagios/sbin">
        Options ExecCGI
        AllowOverride None
        Order allow,deny
        Allow from all
        AuthName "Nagios Access"
        AuthType Basic
        AuthUserFile /usr/local/nagios/etc/htpasswd.users
        Require valid-user
  </Directory>

  Alias /nagios /usr/local/nagios/share

  <Directory "/usr/local/nagios/share">
        Options None
        AllowOverride None
        Order allow,deny
        Allow from all
        AuthName "Nagios Access"
        AuthType Basic
        AuthUserFile /usr/local/nagios/etc/htpasswd.users
        Require valid-user
  </Directory>
</VirtualHost>
```

Ahora activamos y recargamos el sitio con:

```
# a2ensite nagios
# service apache2 reload
```

También podemos eliminar los sitios por defecto (default y default-ssl), al menos es lo recomendable. Además debemos desactivar el sitio "000-default".

```
# a2dissite 000-default
# rm d* /etc/apache2/sites-available
```

A continuación creamos el fichero que será usado para almacenar las contraseñas de los usuarios con permisos para entrar a la interfaz web. Así mismo, crearemos a un usuario. Todo esto lo hacemos con:

```
# htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

Aclaraciones:

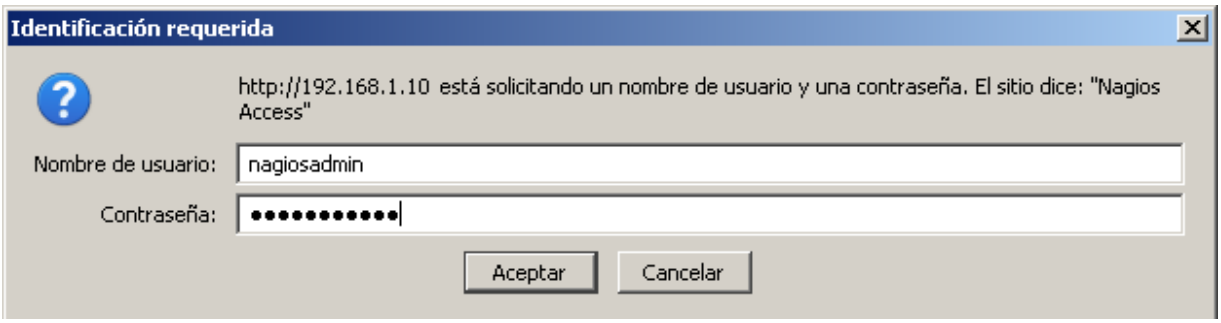
- `-c`: Este parámetro crea el fichero, y en caso de existir lo sobrescribe.
- `/usr/local/nagios/etc/htpasswd.users`: Es la ruta y el nombre del fichero que se usará para almacenar las contraseñas. Podemos utilizar otra ruta u otro nombre, pero entonces deberemos tenerlo en cuenta en el campo "AuthUserFile" del fichero anterior.
- `nagiosadmin`: Es el nombre que he elegido para el usuario, pero es posible elegir otro. Tras pulsar intro nos preguntará la contraseña dos veces.

Ahora tenemos que revisar que en el archivo `/usr/local/nagios/etc/cgi.cfg`, la opción "use_authentication=1" este activada, es decir sea igual a "1". Este es el valor por defecto y además es importante que esta autenticación este activada.

Una vez realizados todos estos cambios nos aseguramos de que todo esta bien reiniciando *apache2* y *nagios.*

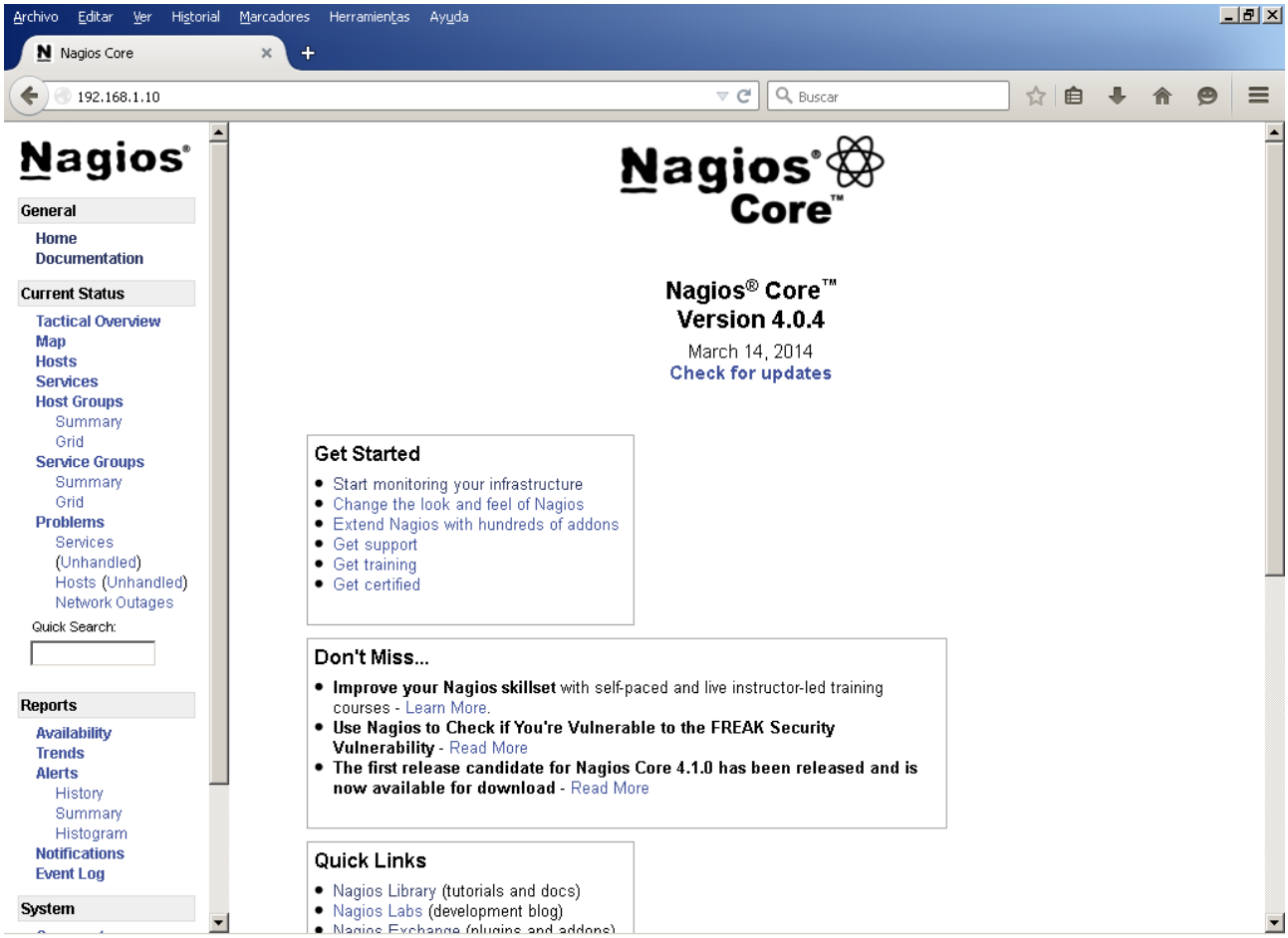
```
# service apache2 restart
# service nagios restart
```

Luego abrimos un navegador e introducimos la dirección IP del servidor Nagios.



Como podemos observar, nos pide que nos identifiquemos antes de continuar. Aquí es donde introduciremos las credenciales que creamos para el fichero "htpasswd.users".

Una vez dentro veremos la interfaz web.



4. CONFIGURACIÓN DE LOS EQUIPOS

En este punto vamos a ver cómo configurar equipos Windows y Linux para poder ser monitorizados.

Aprenderemos:

- Algunas de las distintas formas de conseguir monitorizar los equipos.
- La utilidad de algunos ficheros y cómo modificarlos.
- El uso de las plantillas para hosts y servicios.
- Instalaremos el software necesario en los equipos.

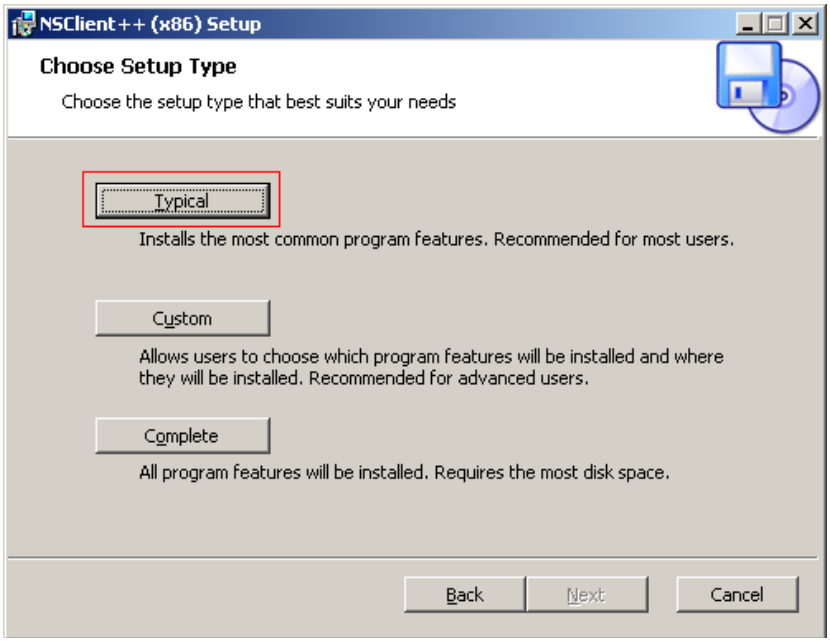
4.1. Configurando equipos Windows

4.1.1. Configurando el host

La configuración en los equipos Windows es muy sencilla. Lo primero es instalar un agente, el cual actúa como un proxy entre el plugin que hace la monitorización y la máquina. En nuestro caso instalaremos **NSClient++** que es el que recomiendan en la documentación de Nagios. Sin embargo hay más clientes que podrían servir, como NC_Net.

Así pues, simplemente hay que descargar la última versión estable de NSClient++ para nuestra plataforma. En mi caso fue la versión "0.4.3.131".

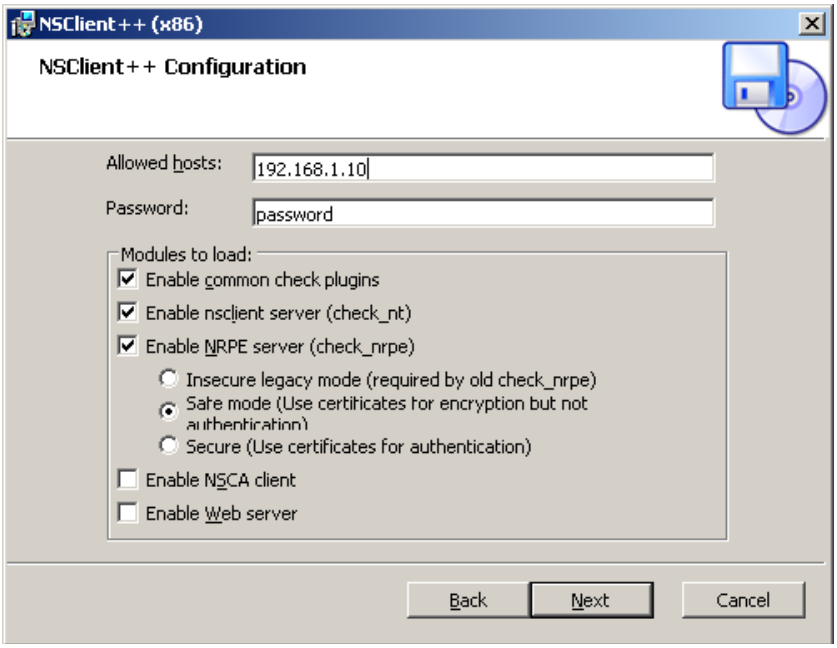
La instalación no es difícil, solo hay que seguir el asistente y aceptar la licencia. Cuando nos pregunte por el tipo de instalación elegiremos "Typical".



El único punto en el que debemos tener algo de cuidado es en la configuración del agente. Ahí debemos establecer en "Allowed hosts" la **IP del servidor Nagios**. Además, tenemos la opción de añadir una contraseña, aunque por defecto nos genera una aleatoria. Podemos utilizar esta, dejarlo en blanco o personalizarla a nuestro gusto. Es recomendable usar una contraseña ya que es una medida más de seguridad.

Además, habilitamos las tres primeras casillas tal y como se ve en la imagen siguiente. Con ello permitimos el uso de plugins comunes, del *check_nt* y del *check_nrpe*. Más adelante veremos como usarlos.

En cuanto al modo para el servidor NRPE (Nagios Remote Plugin Executor), podemos elegir el que más se adapte a nuestras necesidades. En esta guía usaremos la opción por defecto "Safe mode".



Tras esto solo quedará proceder con la instalación.

4.1.2. Configurando Nagios

Aquí haremos una parada para comentar qué es un **objeto** en Nagios. Esto no es más que el nombre que reciben las definiciones de hosts, contactos, servicios, comandos etc... Por lo tanto de aquí en adelante utilizaremos este término para referirnos a estas cosas.

La definición de los objetos en Nagios se realiza en ficheros con extensión ".cfg". Estos ficheros pueden ser creados en cualquier parte, pero deben ser incluidos en `/usr/local/nagios/etc/nagios.cfg`, y el usuario de Nagios (*nagios* en mi caso) debe tener permisos para leerlo. Podéis encontrar más información acerca de la definición de los objetos en la documentación.

Sin embargo, Nagios ya trae unos cuantos ficheros en la ruta `/usr/local/nagios/etc/objects` para la definición de los objetos. De todos ellos el primero que vamos a ver es "windows.cfg". Este es el fichero que trae Nagios por defecto para definir la monitorización de equipos Windows. Si le echamos un vistazo observaremos que ya vienen definidos como ejemplo un host, un hostgroup y varios servicios.

Una buena práctica antes de editar ningún fichero de configuración es hacerle una copia de seguridad. Esto no lo mencionaré más y dependerá de vosotros hacerlo o no. De cualquier modo en la documentación de Nagios podéis ver como monitorizar equipos Windows.

Además, a partir de ahora todos lo que hagamos, lo haremos con el usuario de Nagios (*nagios* en mi caso) a menos que se indique lo contrario.

```
# su nagios
# cd /usr/local/nagios/etc/objects/
# cp windows.cfg windows_org.cfg
```

Una vez realizada la copia del fichero original, modificamos windows.cfg y editamos los ejemplos que trae para amoldarlo a nuestro host Windows. Igualmente se puede definir desde cero eliminando o comentando los ejemplos. Es más, seguramente querremos monitorizar más de un equipo por lo que por cada nuevo equipo o servicio a monitorizar tendremos que añadirlo de cero.

4.1.2.1. Definiendo el host

Lo primero será definir el host:

```
define host{
  use                windows-server    ;
  host_name          Windows 50        ;
  alias              Equipo 50         ;
  address            192.168.1.50      ;
  hostgroups         equipos           ;
  }
```

Aclaraciones:

- `use`: Con esta directiva se le indica una plantilla de la que heredar la configuración. En caso de conflicto porque una misma directiva se utilice tanto en la plantilla como en la definición del host, siempre tendrá prioridad el valor que se establece en la definición host. De momento usaremos esta plantilla y más adelante las veremos más detalladamente.
- `host_name`: Nombre corto usado para identificar al host.
- `alias`: Nombre o descripción usada para identificar al host.
- `adress`: Dirección IP del equipo a monitorizar.
- `hostgroups`: Nombre del hostgroup al que pertenece.

Esta última directiva no viene en el ejemplo y además tampoco es obligatoria, pero como en el fichero viene definido un hostgroup, aprovecharemos para verlo. Simplemente añadimos la directiva y el nombre del hostgroup.

Aquí se pueden establecer muchas más directivas, y no todos los hosts tienen que estar definidos de la misma forma. Unos pueden tener unos valores y unas directivas, y otros otras. Incluso se puede evitar la utilización de plantillas, aunque siempre se deben incluir algunas directivas que son obligatorias, ya sean puestas explícitamente o heredadas de una plantilla. Para encontrar más información acerca de la definición de los host y sus directivas podéis recurrir a la documentación oficial.

4.1.2.2. Definiendo el hostgroup

Como mencionamos más arriba, los hostgroups no son obligatorios pero pueden ser muy útiles. Estos pueden ser usados para dos cosas:

- Agrupar equipos a la hora de visualizarlos en la interfaz web.
- Facilitar la gestión. Por ejemplo, se puede definir un servicio que será aplicado a todos los equipos de un hostgroup.

Un hostgruop solo necesita dos directivas, el nombre y el alias:

```
define hostgroup{
  hostgroup_name     equipos                 ;
  alias              Equipos NO servidores   ;
  }
```

Para saber más acerca de los hostgroups podéis leer la documentación de Nagios.

4.1.2.3. Definiendo los servicios

De momento reutilizaremos los que ya vienen como ejemplo, es decir:

- NSClient++ Version
- Uptime
- CPU Load
- Memory Usage
- C:\ Drive Space
- W3SVC
- Explorer

Vamos a ver tres ejemplos con fin explicativo, pero únicamente hay que modificar una directiva en todos los servicios. Empezamos con NSClient++ Version:

```
define service{
   use                             generic-service
   host_name                       Windows 50
   service_description             NSClient++ Version
   check_command                   check_nt!CLIENTVERSION
   }
```

Aclaraciones:

- `use`: Como en el caso de la definición del host, esta directiva se emplea para utilizar una plantilla, en este caso "generic-service".
- `host_name`: Es el nombre del host o hosts a los que se le aplicará la monitorización de este servicio. Si quisiéramos especificar un *hostgroup* podríamos hacerlo utilizando la directiva `hostgroup_name` en su lugar. Si lo hiciéramos ya no sería obligatorio el uso de la directiva `host_name`.
- `service_description`: Nombre descriptivo para el servicio.
- `check_command`: El comando que usará este servicio junto con su variable, en este caso "CLIENTVERSION".

Plugin check_nt

Aquí volvemos a hacer una pausa para explicar brevemente el plugin *check_nt*. Como recordaréis, durante la instalación de NSClient++ marcamos una casilla que habilitaba el servidor nsclient (check_nt). Gracias a eso podemos utilizar este plugin para monitorizar equipos Windows fácilmente. Este (check_nt) y otros comandos están definidos en el archivo `/usr/local/nagios/etc/objects/commands.cfg`. Más adelante aprenderemos más sobre este fichero.

Podéis ver más información sobre este plugin en su documentación.

El segundo de los servicios que vemos como ejemplo es Memory Usage:

```
define service{
  use                     generic-service
  host_name               Windows 50
  service_description     Memory Usage
  check_command           check_nt!MEMUSE!-w 80 -c 90
  }
```

Aclaraciones:

- `check_command`: Como vemos la definición de este servicio es igual que el anterior, pero en esta directiva el comando tiene dos nuevos parámetros. Estos hacen referencia a los *Warning* (-w 80) y los *Critical* (-c 90).

Estado del servicio

Volvemos a hacer otra parada para aclarar qué son los *Warning* y los *Critical*. Estos son dos de los estados que puede tener un servicio, pero hay dos más, *Unknown* y *Ok.* ¿Qué significan?

- Un servicio está en OK cuando funciona correctamente.
- Un estado UNKNOWN significa que hay algún problema desconocido. Por ejemplo, cuando hay errores en la configuración del servicio se devuelve este estado.
- Un estado WARNING se produce cuando hay un problema no grave o esta cerca de uno grave.
- Un estado CRITICAL se produce cuando hay un problema grave, como cuando el servicio no funciona.

En el servicio Memory Usage tanto los Warning como los Critical reciben un valor. Esto es así porque hay ciertos servicios en los que podemos definir cuando queremos que pasen a esos estados.

Así pues en Memory Usage no se controla si la memoria funciona o no, o tiene algún problema. Lo que se controla es el porcentaje de uso. Así pues, si el equipo que monitorizamos utiliza entre un 80 y 89 por ciento de la memoria, el servicio pasará a un estado Warning. Si el porcentaje supera el 90% pasará a Critical.

Por supuesto estos valores pueden cambiarse acorde a las necesidades que tengamos.

El tercer servicio que vamos a ver es CPU Load:

```
define service{
  use                     generic-service
  host_name               winserver
  service_description     CPU Load
  check_command           check_nt!CPULOAD!-l 5,80,90
  }
```

Aclaraciones:

- `check_command`: Este servicio es muy parecido al anterior, de nuevo monitoriza la carga de uso, pero esta vez le indicamos los valores de forma distinta. Con "-l" podemos pasar parámetros a check_nt. En este caso a la variable "CPULOAD" se le pasan tres parámetros separados por coma. Los dos últimos números hacen referencia a los porcentajes para pasar a estados Warning y Critical. El primer número es el tiempo que tiene que pasar la CPU (en minutos) con esa carga para cambiar de estado.

Con esto finalizamos la explicación de la definición de servicios a monitorizar con check_nt. Si necesitáis más información sobre el uso de este plugin y sus variables siempre podéis ver la documentación.

En este punto deberíais haber aprendido a definir servicios, qué son los estados de los servicios y cómo funciona el plugin check_nt.

Podéis encontrar más información sobre la definición de los servicios y sus variables en la documentación.

4.1.2.4. Añadiendo contraseña a check_nt

Ya tenemos casi todo listo para poder empezar a ver la monitorización de nuestro host Windows, sin embargo aún tenemos que editar un par de cosas.

Como mencioné antes el archivo "commands.cfg" contiene varios comandos predefinidos, entre ellos nuestro check_nt, y este es el que nos interesa. Por tanto abrimos el fichero y lo buscamos, su definición será la siguiente:

```
# 'check_nt' command definition
define command{
  command_name      check_nt
  command_line      $USER1$/check_nt -H $HOSTADDRESS$ -p 12489 -v $ARG1$
$ARG2$
  }
```

Aquí vemos como este comando usa varias variables. Las que nos interesan son los `$ARGX$`. Estas son usadas para recibir los parámetros. Por ejemplo `$ARG1$` siempre se usa para las variables de check_nt (CPULOAD, UPTIME, MEMUSE...), y `$ARG2$` en caso de que se le pasen parámetros con "-l".

Además, podemos ver como el comando recibe también el puerto al que conectarse (`-p 12489`). Este es el puerto por defecto por eso no lo especificamos al definir los servicios. En caso de que cambiásemos el puerto en la configuración del NSClient++, deberemos tenerlo en cuenta aquí.

Pero si recordáis cuando instalamos NSClient++, tuvimos que poner una contraseña, y ni en el comando ni en los servicios le hemos pasado ninguna. Es por ello que vamos a editar el "command_line" y añadírsela.

Esto lo hacemos con "-s":

```
# 'check_nt' command definition
define command{
  command_name      check_nt
  command_line      $USER1$/check_nt -H $HOSTADDRESS$ -p 12489 -s password
-v $ARG1$ $ARG2$
  }
```

Obviamente aquí cada uno deberá poner su contraseña o no poner nada si no la estamos utilizando. Sin embargo, ¿qué pasa si queremos utilizar diferentes contraseñas o puertos?

Pues además de aplicar las **reglas necesarias en los Firewalls** y en las configuraciones de NSClient++ (más adelante veremos como cambiar su configuración), deberemos pasar estos parámetros desde la definición del servicio. Así pues supongamos que la definición de nuestro comando check_nt es la de arriba. Si tuviésemos un host que escucha en el puerto 80 y su contraseña es "patata", la definición de sus servicios sería como la siguiente:

```
define service{
  use                             generic-service
  host_name                       Windows 50
  service_description             NSClient++ Version
  check_command                   check_nt!CLIENTVERSION! -p 80 -s patata
  }
```

La configuración que se establezca al definir el servicio sobrescribirá la que tengamos en la definición del comando.

4.1.2.5. Habilitando la utilización de windows.cfg

Ya solo nos queda un pequeño paso en el fichero configuración principal de Nagios (`/usr/local/nagios/etc/nagios.cfg`). Este fichero tiene bastantes opciones y es muy extenso por lo que no nos pararemos demasiado en él. Pero si queréis más información siempre puedes recurrir a la documentación oficial.

De momento lo único que haremos aquí será descomentar esta línea:

```
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

Con esto habilitaremos el uso del fichero windows.cfg para monitorizar nuestras máquinas Windows.

4.1.2.6. Reiniciando Nagios

Una vez finalizada toda la configuración, solo nos queda reiniciar y comprobar desde nuestra interfaz gráfica que la monitorización se realiza correctamente.

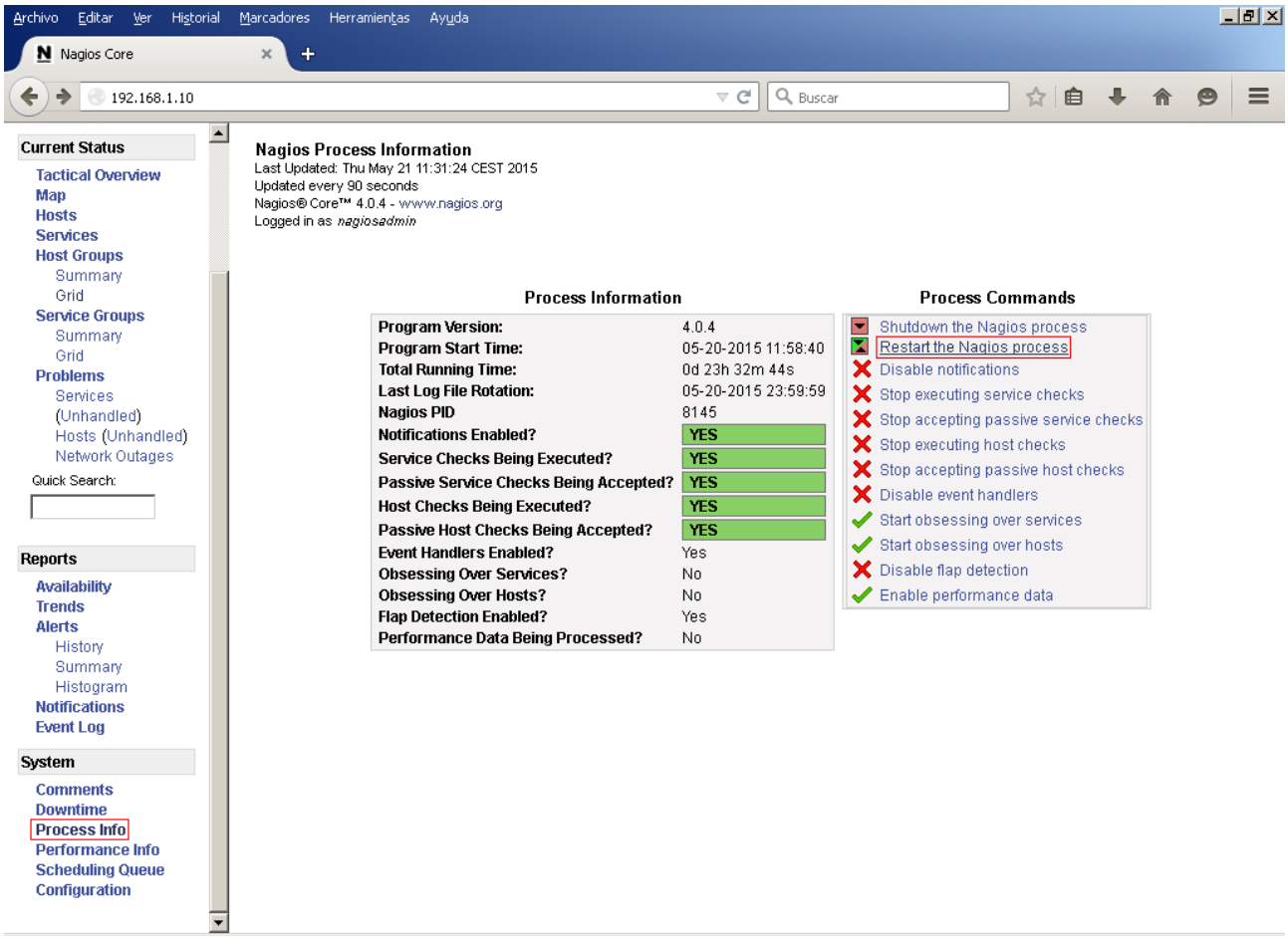
Si lo deseamos podemos verificar la configuración antes de reiniciar con:

```
$ /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

El reinicio podemos hacerlo de dos formas, desde la terminal o desde la interfaz gráfica. Para hacerlo desde la terminal necesitaremos permisos de superusuario (recordad que somos el usuario de Nagios):

```
# service nagios restart
```

Desde la interfaz gráfica tenemos que acudir al menú lateral y bajo la sección "System" pinchamos en "Process Info". Luego dentro de la sección "Process Commands" hacemos click en "Restart the Nagios process". Esto nos llevará a otra pantalla en la que debemos hacer click sobre "Commit".



Es probable que al hacerlo veamos el siguiente error:

Esto es debido a los permisos. Para solucionarlo simplemente añadimos como grupo propietario de la carpeta `/usr/local/nagios/var/rw/` a "www-data":

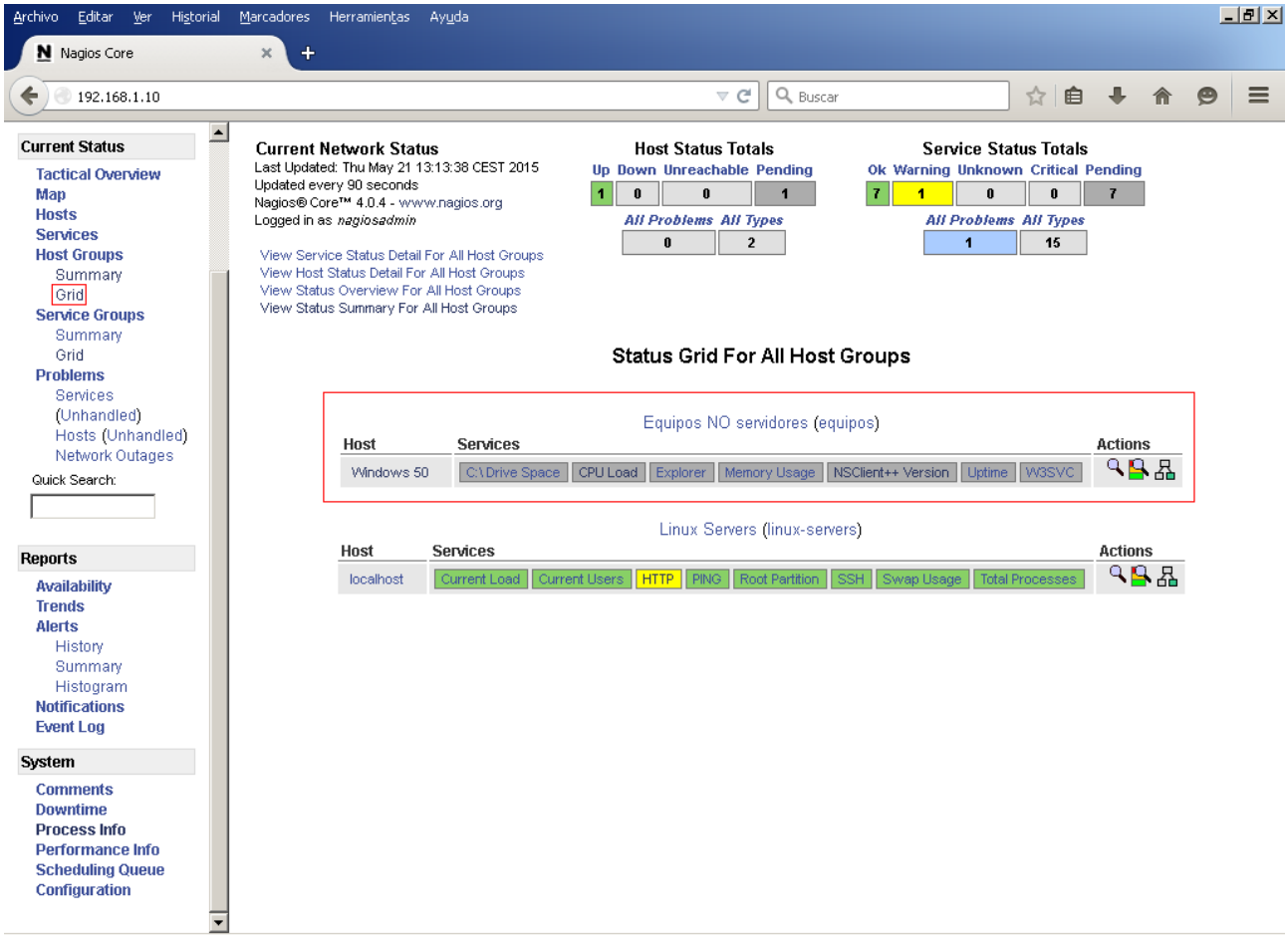
```
$ chown -R :www-data /usr/local/nagios/var/rw/
```

Ahora si volvemos a intentar reiniciar Nagios no deberíamos tener problemas.

4.1.3. Monitorizando equipos

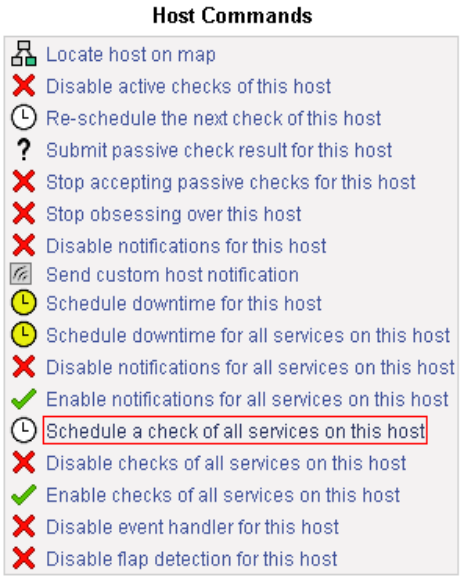
Tras reiniciar Nagios vamos a visualizar el conjunto de equipos que estamos monitorizando. Para ello, bajo la sección "Current Status" del menú lateral, pinchamos en la opción "Grid" de "Host Groups".

Entonces obtendremos una visión general de los servicios y host que monitorizamos (en principio solo deberíamos tener a Nagios como "localhost" y a nuestro recién añadido host Windows).

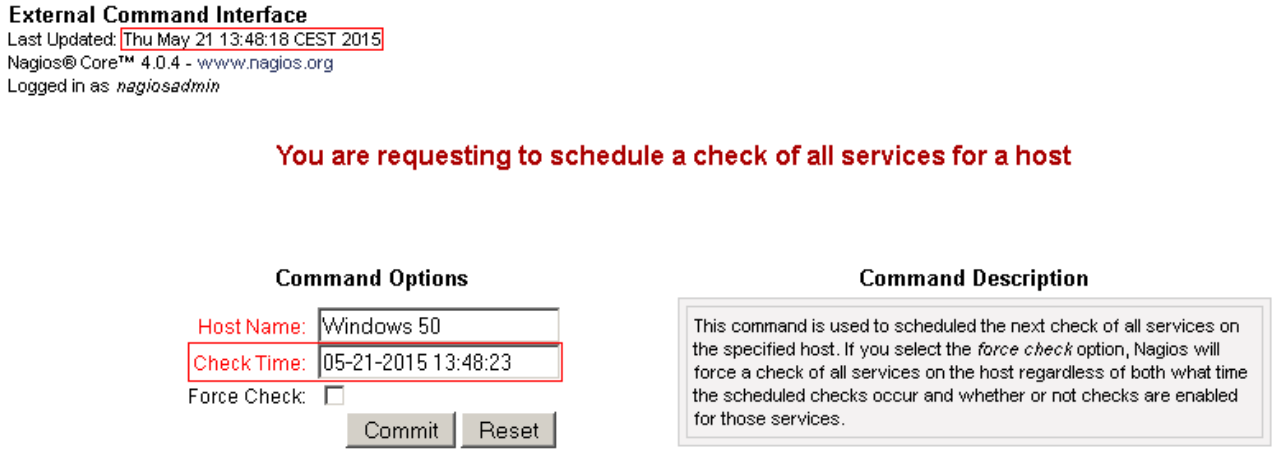


Al principio es normal que todos los servicios estén en gris (Pending) ya que aún no han sido comprobados, solo hay que tener un poco de paciencia. Pero si queréis forzarlo podéis hacerlo, así aprendemos también a forzar *checks*.

Para ello pinchamos en el nombre del host, en el ejemplo Windows 50, lo que nos redirigirá a su página de información. Aquí podemos ver que en la parte derecha tenemos un menú llamado "Host Commands" desde el que podemos hacer diversas acciones. La que nos interesa por el momento es "Schedule a check of all services on this host". Pinchamos en ella y nos dirige a otra página donde programar el siguiente check.



Simplemente tenemos que elegir una fecha futura que se produzca en poco segundos. Para ello tened en cuenta **la hora del servidor**.



Tras cambiar la fecha pulsamos en "Commit", volvemos al "Grid" y ya deberíamos poder ver todos los servicios con un estado correcto. De no ser así podemos repetir el proceso o esperar.

Es posible que el host lo veáis con dos errores:

El servicio W3SVC aparece con un estado Unknown debido a que en este host no hay ningún servidor web instalado. En cambio el problema del servicio Explorer es un simple error en su definición.

Por defecto el "check_command" de este servicio esta definido en windows.cfg:

```
check_command        check_nt!PROCSTATE!-d SHOWALL -l Explorer.exe
```

Simplemente tenemos que cambiar "Explorer.exe" por "explorer.exe" y su estado pasará a ser OK.

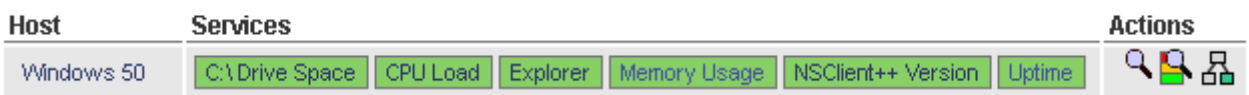
Así pues modificamos esto y eliminamos o comentamos el servicio W3SVC. Si en vuestro caso hay un servidor web instalado y su estado es OK, es probable que no te interese hacerlo y quieras seguir monitorizándolo.

De cualquier modo, una vez hechos los cambios debemos reiniciar Nagios. Esto es algo que tenemos que hacer **siempre** que realicemos modificaciones en los ficheros.

Del mismo modo que hicimos antes, es posible que tras reiniciar el estado de Explorer siga siendo Critical. Por tanto podemos volver a forzar el check de todos los servicios o solo de este.

Para forzar el *check* de este servicio, pinchamos en su nombre y en el mismo menú "Service Commands" buscamos la opción "Re-schedule the next check of this service". El resto del proceso es el mismo.

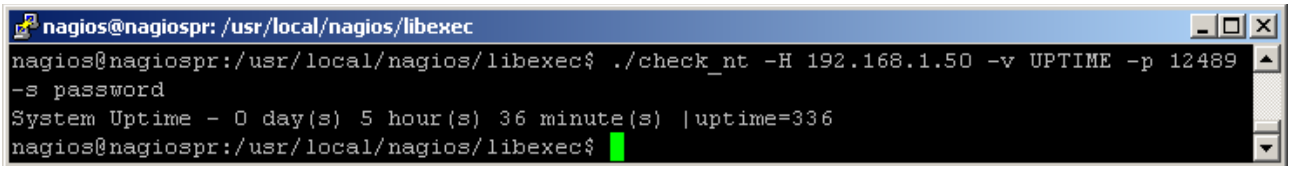
Ahora si, veremos todos los servicios en verde (OK).



4.1.3.1. Monitorizando equipos desde la terminal

Como alternativa podemos ejecutar los comandos de monitorización desde la terminal. Para ello simplemente tenemos que poner el comando seguido de los parámetros. Por ejemplo, para ver el tiempo que lleva un equipo encendido ejecutamos:

```
$ cd /usr/local/nagios/libexec/
$ ./check_nt -H 192.168.1.50 -v UPTIME -p 12489 -s password
```



Recuerda que para más información siempre puedes poder el comando seguido "-h":

```
$ ./check_nt -h
```

4.1.4. Monitorizando la memoria física

Una de las formas más sencillas de hacerlo es mediante el plugin check_nt, tal y como hemos visto. Sin embargo, el comando para monitorizar la memoria de esta forma (`check_nt!MEMUSE`) muestra la suma de la memoria física más la virtual. Si esto es lo que buscamos perfecto, pero si únicamente nos interesa la memoria física ya tenemos que recurrir a otros plugins.

Así pues, en este punto veremos uno de los métodos más comunes de monitorización, el uso del protocolo SNMP (Simple Network Management Protocol).

4.1.4.1. Instalando SNMP en Nagios

Es probable que SNMP ya venga instalado en nuestro sistema. Podemos comprobarlo con:

```
$ dpkg -l snmp
```

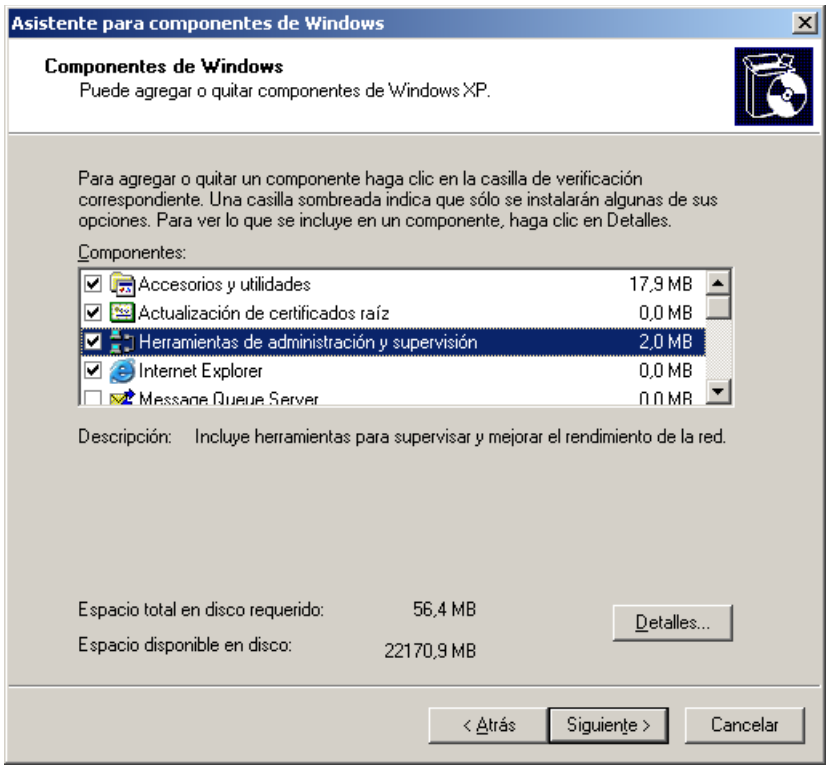
En caso de no estarlo podemos instalarlo con:

```
# apt-get install snmp
```

4.1.4.2. Instalando SNMP en Windows

En Windows es más común que no lo tengamos instalado. Para instalarlo en **Windows XP** necesitamos un CD de Windows XP. Luego podemos seguir estos pasos:

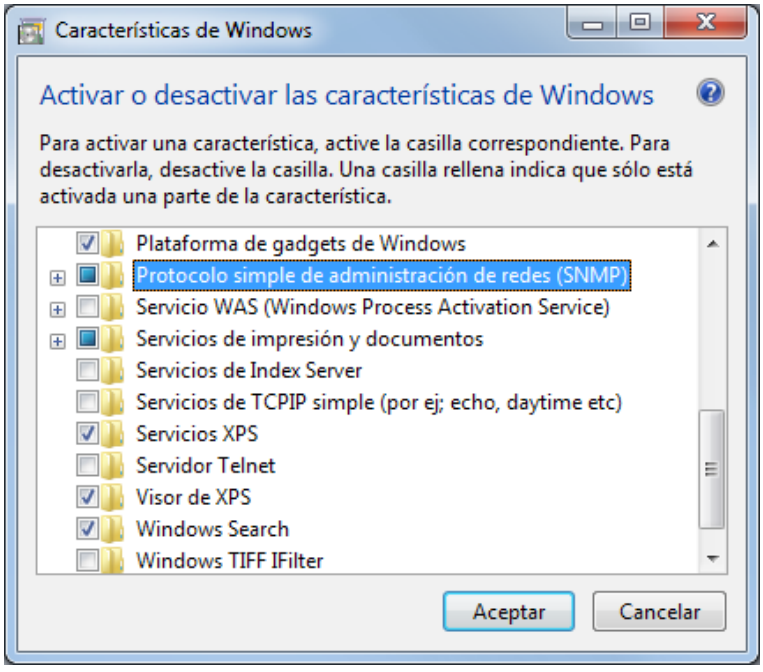
1. Vamos a Inicio > Panel de Control > Agregar o quitar programas.
2. Pinchamos en "Agregar o quitar componentes de Windows".
3. Nos fijamos en la casilla "**Herramientas de administración y supervisión**". En caso de venir marcada pinchamos en "Detalles" y nos aseguramos de que todas las opciones también lo están. Si es así ya tenemos SNMP instalado y podemos omitir esta parte. Si no estaba marcada la marcamos y pinchamos en "Siguiente".



4. Al pasar a la siguiente pantalla comenzará la instalación y pedirá el CD de Windows. Lo introducimos cuando lo haga si es que no lo hicimos ya.
5. Finalizamos la instalación.

En **Windows 7** seguimos estos pasos:

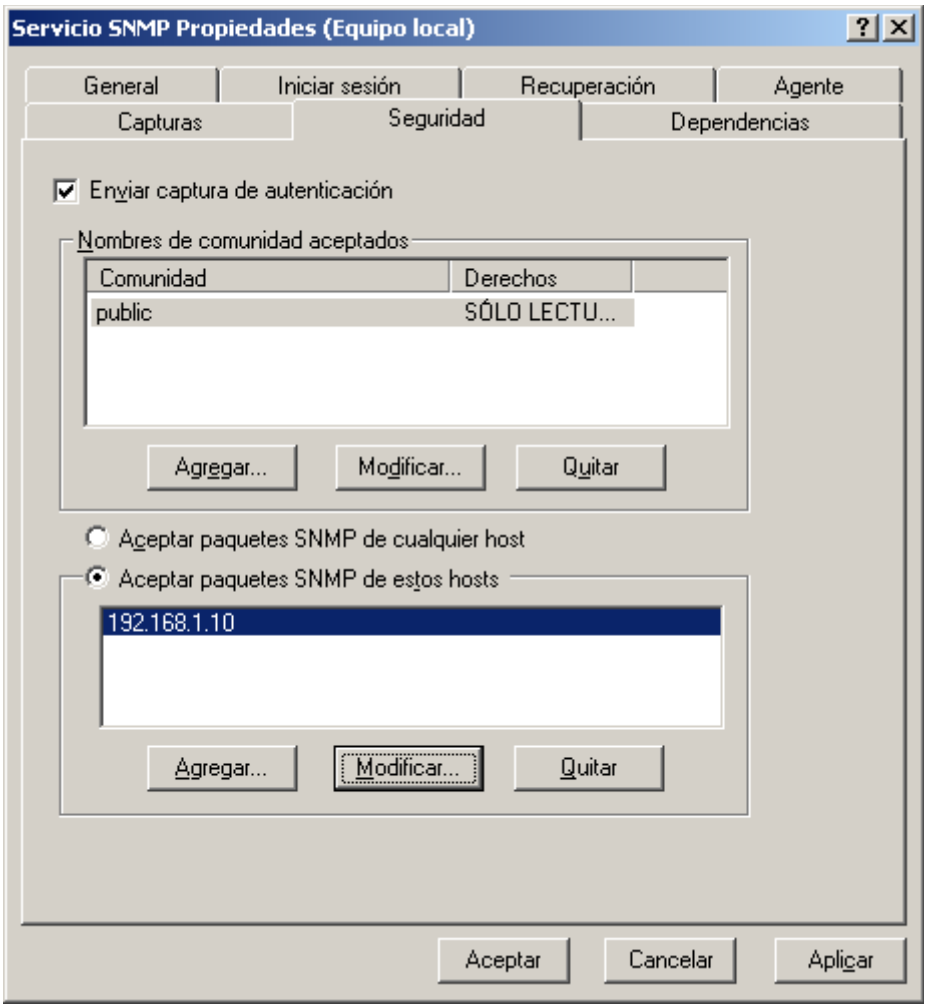
1. Vamos a Inicio > Panel de Control > Obtener programas > Activar o desactivar las características de Windows.
2. En la nueva ventana marcamos "**Protocolo simple de administración de redes (SNMP)**", luego aceptamos y se instalará. En caso de estar ya marcado no tenemos que hacer nada.



4.1.4.3. Configurando SNMP en Windows

Para configurar SNMP en **Windows XP** seguimos estos pasos:

1. Pinchamos en Inicio > Ejecutar... (o pulsamos Win + R).
2. En la nueva ventana escribimos "services.msc" y le damos a Aceptar.
3. En la nueva consola de "Servicios" que se nos abre buscamos "**Servicio SNMP**" y abrimos sus "Propiedades".
4. Nos movemos a la pestaña "Seguridad".
5. Aquí debemos tener la siguiente configuración:
    1. Nos aseguramos de tener marcada la casilla "Enviar captura de autenticación".
    2. En "Nombres de comunidad aceptados" debemos tener una comunidad creada. Por defecto esta presente "public" con permisos de sólo lectura. Esta comunidad nos vale pero podemos modificarla o crear una nueva si es lo que queremos.
    3. Seleccionamos "Aceptar paquetes SNMP de estos hosts" y agregamos la **IP de nuestro servidor Nagios**. También podríamos seleccionar "Aceptar paquetes SNMP de cualquier host" por lo que no tendríamos que agregar la IP del servidor. Sin embargo permitir solo a Nagios es un método más seguro.



6. Una vez configurado aceptamos y reiniciamos el servicio.

Para configurar SNMP en **Windows 7** seguimos estos pasos:

1. Pinchamos en Inicio y en el cuatro de búsqueda escribimos "services.msc". Esto nos deberá arrojar como resultado la consola de "Servicios de Windows". La ejecutamos como *administrador*.
2. En la nueva ventana buscamos "**Servicio SNMP**" y abrimos sus "Propiedades".
3. Nos movemos a la pestaña "Seguridad".
4. Aquí debemos tener la siguiente configuración:
    1. Nos aseguramos de tener marcada la casilla "Enviar captura de autenticación".
    2. En "Nombres de comunidad aceptados" debemos tener una comunidad creada. Por defecto en Windows XP esta presente "public" con permisos de sólo lectura. Con el fin de evitar diferencias en el resto de la guía agregaremos una nueva comunidad con ese mismo nombre y permisos. Sin embargo, si queremos, podemos ponerle otro nombre o permisos acorde a nuestras necesidades.
    3. Agregamos la **IP del servidor Nagios** en "Aceptar paquetes SNMP de estos hosts". También podríamos seleccionar "Aceptar paquetes SNMP de cualquier host" por lo que no tendríamos que agregar la IP del servidor. Sin embargo permitir solo a Nagios es un método más seguro.
    4. Al final debemos tener una configuración similar a la que se ve en la imagen superior para Windows XP.
5. Una vez configurado aceptamos y reiniciamos el servicio.

4.1.4.4. Descargando y probando check_snmp_storage.pl

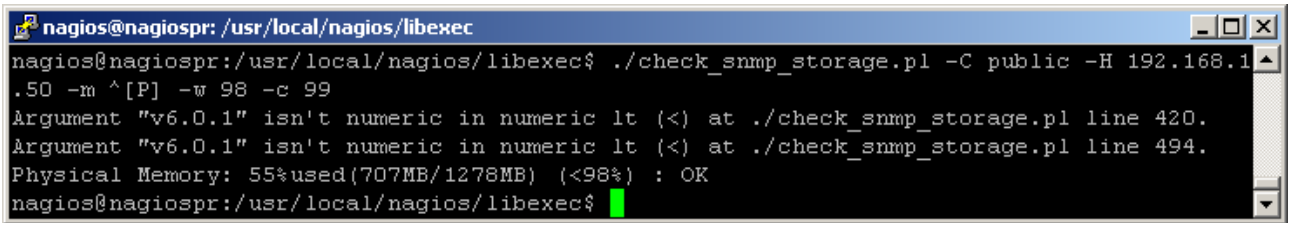
De vuelta en Nagios utilizaremos el plugin *check_snmp_storage.pl* para monitorizar la memoria.

Primero nos situamos en el directorio donde se encuentran nuestros plugins, luego descargamos check_snmp_storage.pl y le damos permisos de ejecución:

```
$ cd /usr/local/nagios/libexec
$ wget http://nagios.manubulon.com/check_snmp_storage.pl
$ chmod +x check_snmp_storage.pl
```

Tras esto probamos que efectivamente funciona y nos devuelve los datos de la memoria física:

```
$ ./check_snmp_storage.pl -C public -H 192.168.1.50 -m ^[P] -w 98 -c 99
```



Aclaraciones:

- `public`: Es el nombre de la comunidad.
- `^[P]`: Hace referencia a la memoría física.
- `Argument "v.6.0.1" isn't...`: Ignorad estos errores.

Ver error "Can't locate Net/SNMP.pm" en la siguiente página.

Error Can't locate Net/SNMP.pm

Si aparece este error durante la ejecución del comando que lanzamos para comprobar el funcionamiento, significa que no tenemos instalado "Net::SNMP". La forma más rápida y fácil de solucionar esto es ejecutando:

```
# apt-get install libnet-snmp-perl
```

Después de instalarlo deberíamos de poder ejecutar el comando anterior sin problemas. Información obtenida de Ubuntu Forums.

Para saber más sobre este plugin podemos ejecutar:

```
$ ./check_snmp_storage.pl -h
```

O acudir a su documentación web.

4.1.4.5. Configurando check_snmp_storage.pl como comando y servicio

Una vez comprobado que todo funciona bien es hora definir el plugin como comando para que pueda ser usando en los archivos de configuración de los objetos.

Lo primero será movernos a la carpeta que contiene la definición de los objetos, y luego editar el commands.cfg.

```
$ cd /usr/local/nagios/etc/objects
$ nano commands.cfg
```

Para mantener la estructura del fichero, definimos el comando a continuación del resto que permiten monitorizar tanto el host local como los remotos. Simplemente tenemos que agregar lo siguiente:

```
# 'check_phy_mem' command definition
  define command{
  command_name    check_phy_mem
  command_line    $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C public
-m ^[P] -w $ARG1$ -c $ARG2$
  }
```

Aclaraciones:

- `check_phy_mem`: Es el nombre que hemos elegido para el comando, pero se puede escoger otro en su lugar si es lo que se desea.
- `$ARGX$`: Al utilizar este comando se pasarán como argumentos los porcentajes para los Warning y los Critical. Así se podrá usar el mismo comando con diferentes valores. En caso de querer siempre los mismos valores para los Warning y los Critical, se puede usar la siguiente definición:

```
# 'check_phy_mem' command definition
define command{
  command_name    check_phy_mem
  command_line    $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C public
-m ^[P] -w 90 -c 95
  }
```

Tened también en cuenta que la comunidad (*public* en el ejemplo), también puede ser pasada como argumento en caso de tener configuradas comunidades distintas en diferentes equipos. De la misma forma, si leemos la documentación de check_snmp_storage.pl veremos como además de monitorizar la memoria física podemos monitorizar discos duros, CD, flopy, memoria virtual etc... Por tanto si queremos podemos también pasar esto como argumento.

Ahora debemos definir un servicio en el que usar nuestro comando. Para ello ya definimos el host Windows dentro del windows.cfg, por lo que ahora queda hacer lo propio con el servicio, así aprendemos también como definir más servicios de los que vienen.

```
$ nano windows.cfg
```

En su interior, junto al resto de servicios definidos para este host escribimos:

```
define service{
   use                     generic-service
   host_name               Windows 50
   service_description     Memory Usage
   check_command           check_phy_mem!90!95
   }
```

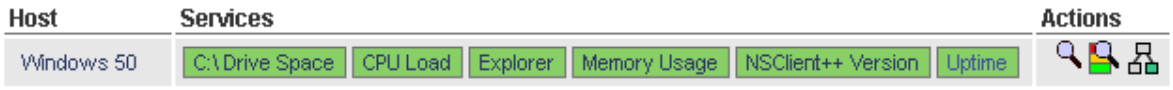
Aclaraciones:

- `Memory Usage`: Nombre descriptivo para el servicio, podéis personalizarlo a vuestro gusto. Como veis hemos utilizado el mismo que el otro servicio que monitoriza la memoria con check_nt. Esto es porque además aprovechamos para eliminar el otro servicio.
- `check_phy_mem!90!95`: Es el nombre que usamos para definir el comando seguido de los valores para los Warning y los Critical, que son pasados como parámetros. En caso de haber definido el comando sin parámetros simplemente habría que usar `check_phy_mem` en lugar de `check_phy_mem!90!95`.

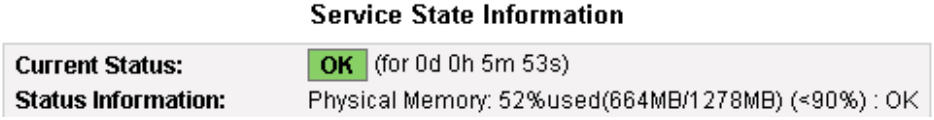
4.1.4.6. Comprobaciones

Una vez finalizados todos los cambios en la configuración, debemos reiniciar Nagios y veremos como ya monitorizamos la memoria física correctamente.

En Nagios comprobamos que todo funciona bien:



Si pinchamos en Memory Usage vemos:



Como podemos observar ahora si podemos monitorizar la memoria física correctamente. Recordad que los cambios pueden tardar un poco en hacerse visibles y puede que aquí sigáis viendo los datos del anterior Memory Usage. Tened paciencia o programad nuevos checks hasta que se actualice.

4.1.5. Monitorizando y configurando los estados de los discos duros por capacidad

Como pasa con la memoria física, check_nt tiene deficiencias en su uso. En este caso este plugin funciona correctamente pero solo permite configurar los estados por porcentajes, lo cual en muchos casos no es lo deseado ¿Por qué? Porque la capacidad de un disco a otro puede variar mucho.

Para el caso de la memoria u otros servicios como la carga de la CPU, el porcentaje es probablemente el dato que nos interese. Si un equipo tiene 2 GB de RAM y otro 1 GB, nos dará igual si esta usando 1 GB, lo que nos importará será que el equipo esta usando el 50% o el 100% de la memoria.

En cambio, en los discos duros si nos interesa la capacidad y no el porcentaje, pues pongamos que tenemos dos discos que se usan para la misma tarea, uno con 100 GB y otro con 1 TB. Si configuramos las alertas para que nos avise cuando quede un 10%, en el caso del primer disco lo hará cuando queden únicamente 10 GB, pero en el otro cuando queden 100 GB. Como se puede ver, en el segundo queda 10 veces más espacio que en el primero. Por tanto lo que en un disco es un Warning cuando solo queda el 10%, en otro no tiene por qué.

Esta problemática se puede resolver de tres formas:

- Haciendo el cálculo de a que porcentaje queremos que nos avise. Por ejemplo, si queremos un aviso cuando queden 10 GB, para un disco de 1TB deberá ser al 1% y para uno de 100 GB deberá avisarnos al 10%.
- Como "`USEDDISKSPACE`" de check_nt devuelve la capacidad y el uso del disco, podemos utilizarlo junto a un script al cual le pasamos como parámetro la capacidad en la que queremos que cambie de estado. Podemos ver un ejemplo en este post de Server Fault.
- Utilizando NRPE.

Aquí utilizaremos esta última alternativa. ¿Por qué? Porque NRPE es el método que suele usarse para monitorizar equipos Linux, pero también es posible usarlo para equipos Windows. Por tanto aprenderemos a instalar y configurar NRPE en Nagios, y utilizarlo en Windows.

4.1.5.1. Configurar NRPE en Windows

Con NSClient++, además de poder usar check_nt, podemos usar NRPE, pero hay que configurar un par de cosas.

El archivo de configuración de NSClient++ se llama "nsclient.ini" en la versión que esta ejecutando el cliente Windows, y se encuentra en "C:\Archivos de programa\NSClient++".

En este fichero ya deberíamos tener la dirección IP del servidor Nagios en "`allowed hosts`", bajo la sección "`[/settings/default]`". De no ser así este sería un buen momento para añadirlo.

Luego bajo la sección "`[/settings/NRPE/server]`" podemos encontrar la siguiente configuración por defecto (se omiten los comentarios):

```
[/settings/NRPE/server]
insecure = false
verify mode = peer-cert
ssl options = no-sslv2,no-sslv3
```

Esto puede cambiar en función de las opciones de NSClient++ que escogimos en la instalación, o dependiendo de nuestra versión del programa.

Aquí deberíamos estudiar las diferentes opciones y ajustarlas a nuestras necesidades. Sin embargo si nos vale con las opciones por defecto simplemente debemos comentar "`verify mode = peer-cert`" y añadir algunos campos más en cualquier parte dentro de esta sección.

Por tanto, una configuración válida bajo la sección "`[/settings/NRPE/server]`" sería:

```
use ssl = true
allowed ciphers = ADH
port = 5666
allow arguments = true
allow nasty characters = true
insecure = false
ssl options = no-sslv2,no-sslv3
```

También podría ser necesario poner a *true* algunos modulos como los siguientes. Sin embargo, por defecto ya deberían venir activados.

```
[/modules]
NSClientServer = 1
CheckSystem = 1
CheckDisk = 1
NRPEServer = 1
CheckExternalScripts = 1
```

Una vez realizados los cambios, guardamos y reiniciamos el servicio de NSClient++. Comentar aquí, que obviamente también es posible configurar NRPE sin SSL, pero sería menos seguro por lo que lo recomendable es hacerlo como se ha visto aquí.

4.1.5.2. Descargando, compilando y preparando NRPE en Nagios

Lo primero que debemos hacer en Nagios es descargar y descomprimir NRPE.

```
$ cd
$ wget http://downloads.sourceforge.net/project/nagios/nrpe-2.x/nrpe-
2.15/nrpe-2.15.tar.gz
$ tar xvzf nrpe-2.15.tar.gz
```

Luego nos introducimos en la carpeta descomprimida y ejecutamos "`./configure`".

```
$ cd nrpe-2.15
$ ./configure
```

Esto puede devolver algún error, por ejemplo durante la realización de la guía apareció el siguiente:

```
checking for SSL headers... configure: error: Cannot find ssl headers
checking for SSL libraries... configure: error: Cannot find ssl libraries
```

Para corregirlo habrá que instalar las librerías SSL faltantes.

```
# apt-get update && apt-get install libssl-dev
```

A continuación volvemos a ejecutar "`./configure`". Si volvemos a tener el mismo error, lo ejecutamos con los siguientes parámetros:

```
$ ./configure --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib/i386-
linux-gnu
```

Aclaraciones:

- `--with-ssl-lib=/usr/lib/i386-linux-gnu`: Aquí se indica la ruta donde se encuentra, en el ejemplo `/usr/lib/i386-linux-gnu`. Esta puede cambiar, por lo que en caso de error se debería poner la correcta. Para conocerla, podemos ejecutar "`dpkg -L libssl-dev | grep /usr/lib`". Ahí aparecerán varias rutas, de las cuales solo nos interesa la que empieza por `/usr/lib` y sigue con la arquitectura de nuestra máquina.

Una vez ejecutado "`./configure`" sin fallos, procederíamos a realizar lo siguiente:

```
$ make
$ make install
$ cp sample-config/nrpe.cfg /usr/local/nagios/etc/
```

Ahora tenemos que instalar *xinetd.* También es posible instalar NRPE como demonio aparte y no utilizar xinetd, pero en esta guía lo haremos así.

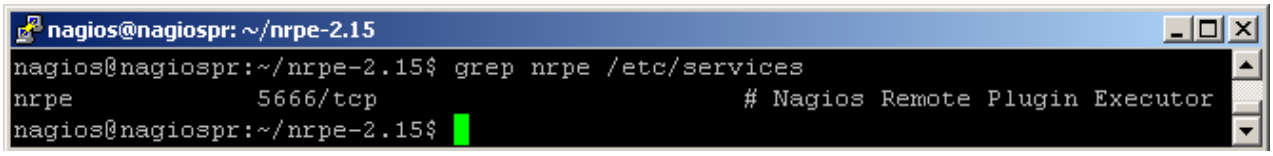
```
# apt-get install xinetd
```

Y después:

```
# make install-xinetd
```

Tras esto solo queda comprobar si en el fichero `/etc/services` aparece una línea descomentada con "`nrpe 5666/tcp`". En caso de estar comentada o no aparecer la añadiríamos. Para comprobar si esta presente podemos ejecutar:

```
$ grep nrpe /etc/services
```



A continuación iniciamos el servicio xinetd.

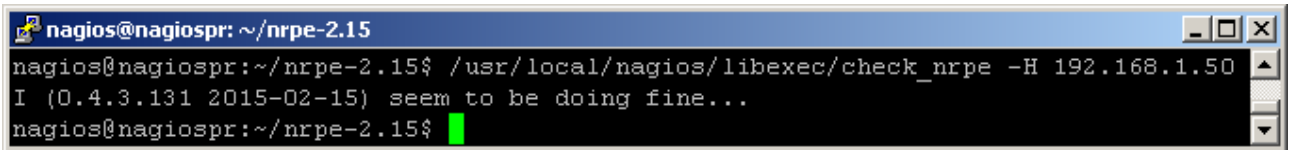
```
# service xinetd start
```

En caso de problemas es posible que necesitemos recargarlo con:

```
# service xinetd reload
```

Con esto ya deberíamos de tener preparado NRPE en el servidor. Para comprobarlo ejecutamos:

```
$ /usr/local/nagios/libexec/check_nrpe -H 192.168.1.50
```



4.1.5.3. Configurando check_disk como comando y servicio

Una vez instalado y habiendo realizado los ajustes necesarios para comprobar el correcto funcionamiento de *check_nrpe*, sería el turno de configurarlo como comando y servicio para una monitorización automática.

Lo primero será añadir el comando al commands.cfg.

```
$ cd /usr/local/nagios/etc/objects
$ nano commands.cfg
```

Para mantener la estructura del fichero, lo definiremos a continuación de "check_phy_mem" por ejemplo.

```
# 'check_disk' command definition
  define command{
  command_name    check_disk
  command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c CheckDriveSize -a
ShowAll=long MinWarnFree=$ARG1$ MinCritFree=$ARG2$ Drive=$ARG3$
  }
```

Aclaraciones:

- `check_disk`: Como ya se comentó es el nombre que decidimos ponerle.
- `MinWarnFree` y `MinCritFree`: Valor mínimo antes de cambiar a estados de tipo Warning y Critical. Estos valores son pasados como argumentos, aunque como vimos en la memoria física se pueden establecer unos valores fijos que aplicarán a todos los que usen el comando. Obviamente, a menos que todos los discos tengan la misma capacidad o se usen para lo mismo esto no nos interesará, y por tanto pasaremos los valores como argumentos.
- `Drive`: La letra del disco a monitorizar que también se pasará como parámetro.

Ahora abrimos el windows.cfg donde tenemos definido el host y el resto de servicios del equipo *Windows 50*. De nuevo, para mantener la estructura definimos el servicio a continuación del resto para ese host, por ejemplo después del Memory Usage.

```
define service{
  use                     generic-service
  host_name               Windows 50
  service_description     C:\ Drive Space
  check_command           check_disk!8G!5G!c
  }
```

Aclaraciones:

- `check_disk!8G!5G!c`: Utilizamos el nombre que escogimos para el comando y le pasamos tres parámetros. Los dos primeros para los estados, en este caso, cambiará a Warning cuando queden 8GB y a Critical cuando queden 5GB. El último parámetro, la "`c`", es la letra de la unidad a monitorizar.

4.1.5.4. Comprobaciones

Una vez terminados todos los cambios solo nos queda reiniciar Nagios y comprobar el funcionamiento.

Service State Information

Current Status:	OK (for 0d 0h 57m 42s)
Status Information:	OK c: Total: 37.245GB - Used: 15.61GB (42%) - Free: 21.635GB (58%)
Performance Data:	'c free'=21.63526GB;8;5;0;37.24477 'c free %'=58%;21;13;0;100

Aquí a simple vista no se observará ninguna diferencia con respecto a la monitorización que se realiza con "`check_nt!USEDDISKSPACE`", pero ahora los *Current Status* cambiarán en función de la capacidad que indicamos y no por porcentajes.

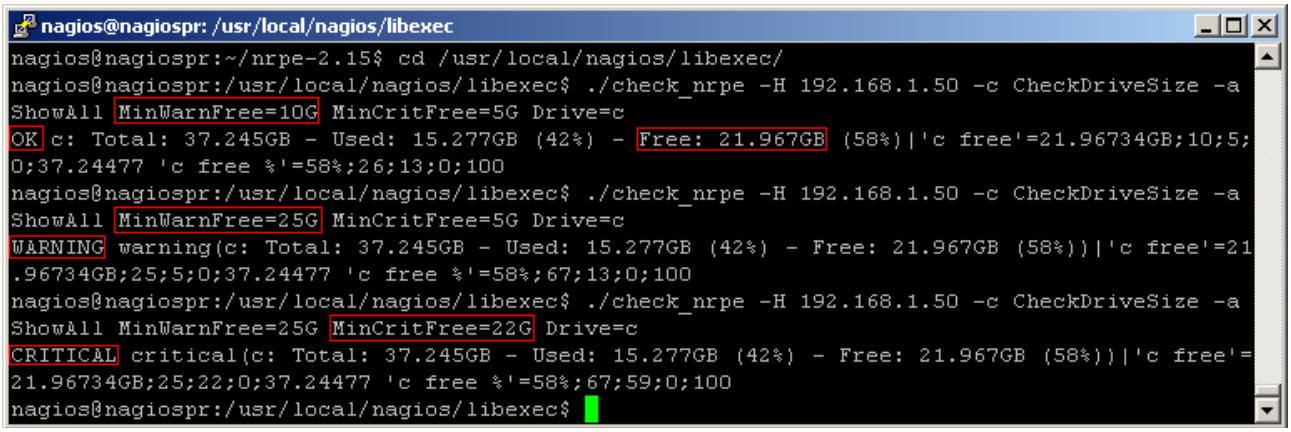
De todos modos, si nos fijamos bien en *Perfomance Data*, después del espacio libre (*21,63526GB* en el ejemplo), aparecen dos números separados por ";". Estos dos números son el 8 y el 5, que son los valores que fijamos al definir el servicio.

Aún así, si queremos podemos asegurarnos del correcto funcionamiento cambiando los valores que usamos al definir el servicio, y ajustándolos en función del espacio actualmente usado.

De la misma manera, si no deseamos andar editando el servicio y reiniciando Nagios, podemos probar a usar lo siguiente desde la consola:

```
$ cd /usr/local/nagios/libexec/
$ check_nrpe -H 192.168.1.50 -c CheckDriveSize -a ShowAll MinWarnFree=10G
MinCritFree=5G Drive=c
```

Donde iríamos cambiando los valores de *MinWarnFree* y *MinCritFree* y viendo si la salida es OK, Warning o Critical.



Una vez llegados a este punto, hemos aprendido a monitorizar equipos Windows, instalando el software necesario y configurándolo para un correcto funcionamiento. Además, aprendimos a utilizar el plugin check_nt, NRPE y el protocolo SNMP para realizar esta tarea.

También somos capaces de utilizar plugins extras que no vienen incluidos en el paquete Nagios Plugins, configurando los comandos con o sin argumentos, y utilizándolos posteriormente desde la terminal o definiéndolos como servicios.

4.2. Configurando y modificando las plantillas

Ya hemos visto como modificar el windows.cfg para añadir hosts y servicios, pero en estos utilizábamos plantillas y no configurábamos todas las directivas. En este punto vamos a editar el fichero "templates.cfg" que contiene plantillas ya definidas para contactos, hosts y servicios.

4.2.1. Modificando las plantillas para los hosts

Si vamos al fichero templates.cfg veremos cinco plantillas predefinidas para los hosts:

- generic-host: Plantilla genérica que usan el resto de plantillas.
- linux-server: Plantilla para servidores Linux. Es la que se usa para Nagios (localhost).
- windows-server: Plantilla para servidores Windows. Es la que venimos usando.
- generic-printer: Plantilla para monitorizar impresoras.
- generic-switch: Plantilla para monitorizar switches.

Nosotros nos quedaremos de momento solo con *windows-server*, que es la que venimos usando. Además si os fijáis bien todas las plantillas son iguales con algunas pequeñas modificaciones. Por lo tanto nos centraremos en la plantilla para los hosts Windows y a partir de esta iremos viendo las directivas.

4.2.1.1. Plantilla generic-host

Sin embargo primero le echaremos un vistazo a *generic-host*, que para algo es la que utilizan todas.

```
# cd /usr/local/nagios/etc/objects/
# nano templates.cfg
```

Aquí, bajo la sección "`HOST TEMPLATES`" encontramos a generic-host (se omiten los comentarios):

```
define host{
   name                            generic-host    ;
   notifications_enabled           1               ;
   event_handler_enabled           1               ;
   flap_detection_enabled          1               ;
   process_perf_data               1               ;
   retain_status_information       1               ;
   retain_nonstatus_information    1               ;
   notification_period             24x7            ;
   register                        0               ;
   }
```

Si leemos los comentarios nos podemos hacer una idea del significado de cada directiva. Además, siempre podemos ir a la documentación oficial para obtener más información.

De todos modos intentaremos explicar brevemente algunas de las directivas más interesantes. Pero antes dejaremos claro que muchas de estas directivas simplemente tienen como valor un "1" o un "0". Si esta a "1" es que esta activada, si esta a "0" desactivada.

Las directivas que no se usan estarán desactivadas, pero algunas otras tienen valores por defecto, para conocer mejor esto lo mejor es mirar la documentación de Nagios. Así mismo, como ya comentamos, una plantilla puede tener un valor para una directiva, pero cuando otra plantilla o host usa esa plantilla puede sobrescribir la directiva con un valor distinto.

Una vez aclarado esto veamos algunas de las distintas directivas:

- `name`: Nombre utilizado para la directiva.
- `notifications_enabled`: Si esta activada enviará notificaciones al o los contactos definidos. Aprenderemos más sobre contactos y notificaciones más adelante.
- `flap_detection_enabled`: Habilita o no la detección de "flapping". Ahora veremos brevemente que es.
- `notification_period`: Especifica el nombre del "time period" en el que se permite el envío de notificaciones. Veremos esto más adelante.
- `Register`: Se usa únicamente con las plantillas y su valor debe ser "0".

Aunque hay ciertas cosas que veremos más adelante, es oportuno realizar una explicación rápida de algunas cosas.

> Las **notificaciones** son mensajes de alertas enviados a un contacto cuando sucede algo. Por ejemplo, si un host se apaga o falla un servicio, es posible configurar que se envíe una notificación a uno o más contactos vía email, sms, mensajería instantánea etc...
>
> El **flapping** es un estado especial que tienen los host y los servicios. Básicamente podemos decir que esto ocurre cuando su estado cambia muchas veces en poco tiempo. Por ejemplo, cuando lo reiniciamos varias veces en poco tiempo. Una de las ventajas de esto es que cuando sucede no se envían notificaciones. Imagina lo molesto que es recibir muchas notificaciones en poco tiempo por un equipo que estamos reiniciando.
>
> Para saber más sobre el flapping podéis leer la documentación .
>
> Los **time periods** se definen en el fichero "timeperiods.cfg" y pueden ser usados para definir cuando queremos que se monitorice un equipo, o cuando queremos que se envíen notificaciones.

4.2.1.2. Plantilla windows-server

Ahora veamos la plantilla windows-server:

```
define host{
   name                    windows-server  ;
   use                     generic-host    ;
   check_period            24x7            ;
   check_interval          5               ;
   retry_interval          1               ;
   max_check_attempts      10              ;
   check_command           check-host-alive;
   notification_period     24x7            ;
   notification_interval   30              ;
```

```
    notification_options    d,r             ;
    contact_groups          admins          ;
    hostgroups              windows-servers ;
    register                0               ;
    }
```

Como vemos algunas directivas se repiten y sobrescriben, pero veamos algunas de las nuevas:

- `check_period`: Al igual que se puede indicar el time period en el que se pueden enviar notificaciones, también se puede establecer uno para indicar cuando se monitorizará.
- `check_interval`: Indica la cantidad de tiempo que pasará desde un check a otro en minutos. Por si aún no ha quedado claro, un check es una comprobación de un servicio o host mediante el comando correspondiente.
- `retry_interval`: Si un host cambia de estado hará los checks con más frecuencia hasta que cambie a un estado de tipo *HARD* (ahora lo vemos). Aquí se indica el tiempo en minutos.
- `max_check_attempts`: Número de checks que hará cuando un host cambie a un estado que no sea OK. Una vez realizado todos checks pasará a un estado de tipo HARD y los checks se harán en el tiempo indicado en `check_interval`.
- `check_command`: Comando utilizado para comprobar si el host esta encendido.
- `notification_interval`: Indica el tiempo en minutos que esperará hasta enviar una nueva notificación si el host aún sigue en un estado que no sea OK.
- `notification_options`: Especifica que tipo de notificaciones se enviarán. Se pueden enviar notificaciones cuando el host pase a estos estados:
  - w: Warning.
  - u: Unknown.
  - c: Critical.
  - f: Flapping.
  - s: Scheduled downtime.
  - r: Recovery.
  - n: None.
- `contact_groups`: Grupo o grupos a los que se le enviarán las notificaciones.
- `hostgroups`: Grupo al que pertenecen los host que utilicen esta plantilla.

Aquí vuelven a parecer un par de cosas nuevas en las notificaciones, y a parte los tipos de estado que se explicarán a continuación.

Las notificaciones de tipo "Recovery" se producen cuando el host o servicio se ha recuperado de un estado anterior, es decir pasa a un estado OK. Las de tipo "Scheduled downtime" se realizan cuando se ha programado un tiempo de apagado. Por ejemplo, apagamos un host para realizarle un mantenimiento y queremos o no, que se nos notifique cuando el tiempo de apagado empiece y termine. Podéis obtener más información sobre los Scheduled downtime en la documentación.

Por último si las notificaciones se ponen en "None", no se recibirá ninguna.

Tipos de estado

En Nagios como ya vimos hay varios estados (Ok, Warning, Critical...), pero solo dos tipos de estado, que son *SOFT* y *HARD*. Esto es así para evitar falsas alarmas. Así pues, cuando se detecta un cambio en un host o servicio, se utiliza la directiva "`max_check_attempts`" para estar seguro antes de confirmar el cambio de estado.

Imaginad un problema de red de un par de minutos. Antes de establecer el servicio como Critical y enviar una notificación, Nagios comenzará a realizar más comprobaciones en un nuevo intervalo de tiempo (`retry_interval`). Si tras alcanzar el "`max_check_attempts`" el problema continua, el estado del servicio cambiará a Critical y se notificará. En cambio, si se soluciona, el estado volverá a estar OK y no se notificará a nadie.

Como resumen podemos decir que un host o servicio alcanza un estado de tipo *SOFT* cuando hay un problema o se produce la recuperación de uno pero aún no se a confirmado. En cambio el tipo *HARD* se establece cuando el problema o la recuperación es real.

Puedes encontrar más información sobre esto en la documentación de Nagios.

4.2.1.3. Otras directivas

Otras directivas que no hemos visto ni aquí ni en la definición del host en windows.cfg y que pueden ser interesantes sin complicar mucho las cosas son:

- `display_name`: Nombre alternativo para que sea mostrado en la interfaz web.
- `event_handler`: Indica un comando a ejecutar cuando se produce un cambio de estado. En esta guía no lo veremos pero puede ser muy útil. Para saber más sobre el manejador de eventos puedes consultar la documentación de Nagios.
- `contacts`: Nombre del contacto o contactos a notificar.
- `first_notification_delay`: Tiempo en minutos que Nagios esperará antes de enviar la primera notificación de un problema.
- `notes`: Información perteneciente al host que podrá ser visualizada en la interfaz gráfica.
- `notes_url`: Permite definir una URL donde encontrar más información sobre el host.
- `icon_image`: Imagen asociada al host que será mostrada en varias partes de la interfaz gráfica. Se asume que estará almacenada en "`/usr/local/nagios/share/images/logos`".
- `icon_image_alt`: Descripción alternativa para la imagen.

Como siempre, puedes encontrar más variables y más información en la documentación de Nagios.

4.2.2. Modificando las plantillas para los servicios

En templates.cfg también encontramos dos plantillas predefinidas para los servicios:

- generic-service: Es la que hemos estado usando.
- local-service: Hereda de generic-service y es usada para los servicios de Nagios (localhost).

4.2.2.1. Plantilla generic-service

Empezaremos viendo esta plantilla, la cual tiene bastantes directivas:

```
define service{
        name                            generic-service         ;
        active_checks_enabled           1                       ;
        passive_checks_enabled          1                       ;
```

```
        parallelize_check               1                       ;
        obsess_over_service             1                       ;
        check_freshness                 0                       ;
        notifications_enabled           1                       ;
        event_handler_enabled           1                       ;
        flap_detection_enabled          1                       ;
        process_perf_data               1                       ;
        retain_status_information       1                       ;
        retain_nonstatus_information    1                       ;
        is_volatile                     0                       ;
        check_period                    24x7                    ;
        max_check_attempts              3                       ;
        normal_check_interval           10                      ;
        retry_check_interval            2                       ;
        contact_groups                  admins                  ;
        notification_options            w,u,c,r                 ;
        notification_interval           60                      ;
        notification_period             24x7                    ;
        register                        0                       ;
        }
```

Veamos algunas directivas:

- `active_checks_enabled`: En Nagios se pueden hacer checks tanto activos como pasivos. Los que nosotros estamos viendo son los activos y es en los que nos centraremos. Pero si queréis conocer más acerca de los checks pasivos podéis ver la documentación.
- `normal_check_interval`: Es el equivalente al "`check_interval`" para los servicios.
- `retry_check_interval`: Es el equivalente al "`retry_interval`" para los servicios.

El resto de directivas funcionan igual que en el host o exceden el ámbito de esta guía. Sin embargo aquí se muestra otra que puede usarse y no se ha visto:

- `servicegroups`: Es el equivalente al hostgroups, por lo que podemos agrupar varios servicios para facilitar la administración y utilización de los mismos.
- `hostgroup_name`: Como comentamos esta directiva puede sustituir a `host_name` a la hora de especificar a que equipos se le aplicará. Por supuesto se debe especificar uno o más hostgroups.

Como siempre, podéis encontrar más directivas e información sobre su uso en la documentación.

4.2.3. Time periods

En la definición de algunas directivas de los objetos y servicios es posible establecer un "periodo de tiempo" para indicar cuando será monitorizado o cuando se enviarán notificaciones.

Estos time periods están definidos en el fichero "timeperiods.cfg". Aquí podemos encontrar varios como 24x7, workhours, none... Su uso es muy sencillo ya que solo tiene cinco directivas, además de la global "use". Para verlas nos centraremos en *us-holidays* y *24x7_sans_holidays*.

Empecemos por la segunda donde vemos lo siguiente:

```
define timeperiod{
  timeperiod_name 24x7_sans_holidays
  alias           24x7 Sans Holidays

  use             us-holidays

  sunday          00:00-24:00
  monday          00:00-24:00
  tuesday         00:00-24:00
  wednesday       00:00-24:00
  thursday        00:00-24:00
  friday          00:00-24:00
  saturday        00:00-24:00
  }
```

Las directivas `timeperiod_name`, `alias` y `use` no tienen ningún misterio, ya vimos unas similares al definir objetos de tipo host y service. Por lo tanto las que nos interesan son `weekday`, `exception` y `exclude`.

Este primer timeperiod usa los días de la semana en inglés de sábado a domingo en horario de 24 horas. Este es el formato que se usa para los *weekday*. Si quisierais excluir un día entero bastaría con no ponerlo. Como veis su uso es muy sencillo, ahora pasemos a us-holidays:

```
define timeperiod{
  name                    us-holidays
  timeperiod_name         us-holidays
  alias                   U.S. Holidays

  january 1               00:00-00:00     ;
  monday -1 may           00:00-00:00     ;
  july 4                  00:00-00:00     ;
  monday 1 september      00:00-00:00     ;
  thursday 4 november     00:00-00:00     ;
  december 25             00:00-00:00     ;
  }
```

Aquí se emplean los *exception*, que excluyen ciertas fechas de nuestro periodo de tiempo. Estos en vez de usar los días de la semana utilizan otro tipo de fecha como:

- `january 1`: El 1 de Enero de cada año.
- `monday -1 may`: El último lunes de Mayo de cada año.
- `2016-01-28`: El 28 de Enero de 2016.
- `day 2`: El día 2 de cada mes.
- `april 10 - may 15`: Del 10 de Abril al 15 de Mayo de cada año.
- `day 1 - 15 / 5`: Del 1 al 15 de cada mes cada 5 días todos los años.

Por último, la directiva *exclude* puede usarse de la misma forma que *use*, pero serviría para excluir periodos de tiempo.

Como hemos visto el uso de los timeperiods es muy sencillo. Si quisierais aprender más sobre esto siempre podréis recurrir a la documentación.

4.3. Configurando equipos Linux

4.3.1. Configurando el host

Ya aprendimos a monitorizar equipos Windows, pero es probable que también queramos monitorizar equipos Linux, por lo que aquí vamos a aprender a hacerlo.

La definición para los servicios y los host es exactamente la misma, pero de todos modos veremos a modo de ejemplo la definición de un equipo Linux con algunos servicios.

Lo primero será instalar la lista informativa de los paquetes *build-essential*, así como *xinetd* en caso de que no venga instalado en el sistema:

```
# apt-get install build-essential xinetd
```

A continuación, como *root* seguimos prácticamente el mismo proceso que hicimos con Nagios para instalar NRPE y los plugins. Por tanto descargamos los Nagios Plugins.

```
# cd
# wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
# tar xvfz nagios-plugins-2.0.3.tar.gz
# cd nagios-plugins-2.0.3
# ./configure
# make
# make install
```

Ahora creamos un usuario para Nagios y le damos la propiedad de las siguientes carpetas:

```
# adduser nagios
# chown nagios.nagios /usr/local/nagios/
# chown -R nagios.nagios /usr/local/nagios/libexec/
```

A continuación descargamos, compilamos e instalamos NRPE. Sin embargo, antes conviene explicar una cosa. Desde el servidor Nagios, cuando definamos los servicios a monitorizar no podremos ajustar los argumentos según queramos. Es decir, los límites para los Warnings y los Criticals, así como otros parámetros como la ruta a monitorizar para *check_disk* etc... Estos ajustes deberán ser indicados en el fichero "nrpe.cfg" de cada equipo, al contrario de como lo hacíamos con los hosts Windows.

Aún así, si se deseamos pasar argumentos desde Nagios también podemos hacerlo, aunque esto supone un riesgo de seguridad. Para ello hay que pasar un parámetro al comando "`./configure`". Así pues, si deseamos instalar NRPE para que reciba argumentos desde Nagios ejecutaremos:

```
# cd
# wget http://downloads.sourceforge.net/project/nagios/nrpe-2.x/nrpe-2.15/nrpe-2.15.tar.gz
# tar xvzf nrpe-2.15.tar.gz
# cd nrpe-2.15
# ./configure --enable-command-args
# make all
# make install-plugin
# make install-daemon
# make install-daemon-config
# make install-xinetd
```

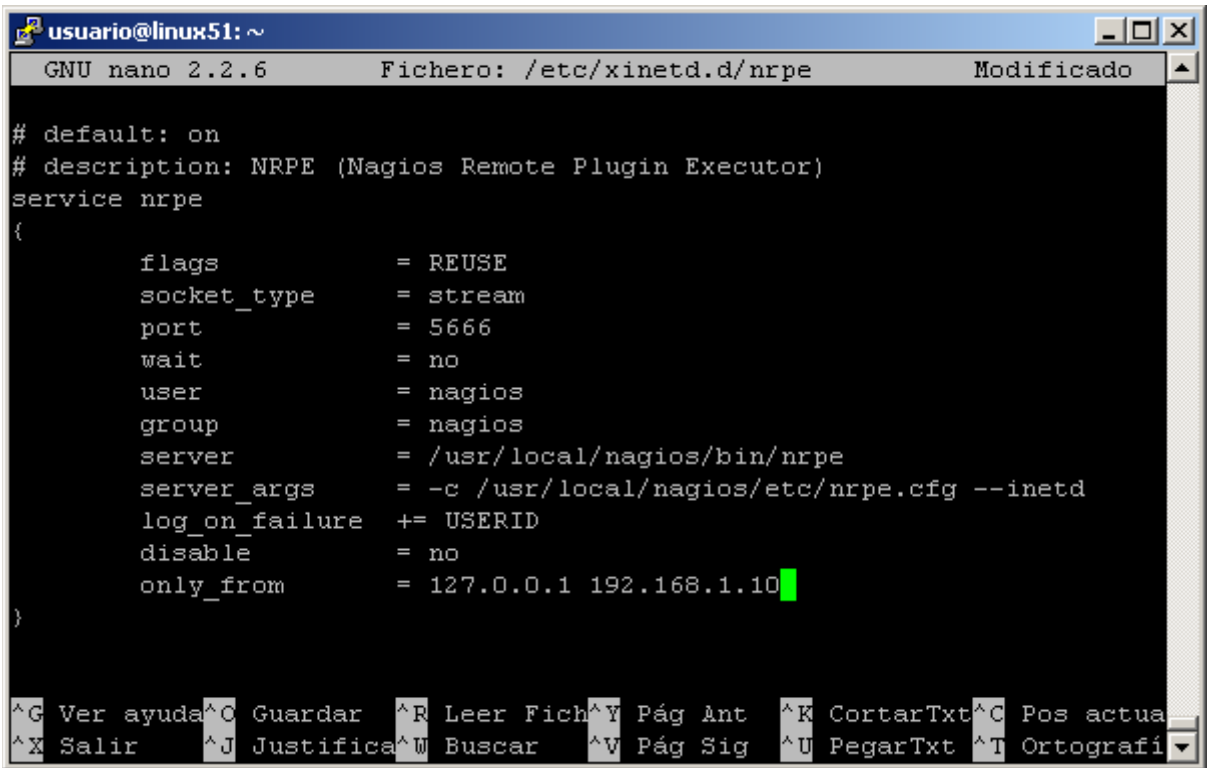
Recordad que si obtenéis un error con las librerías o cabeceras SSL al ejecutar "`./configure`" podéis arreglarlo tal y como vimos al instalar NRPE en Nagios.

En cambio, si no deseamos pasar argumentos desde Nagios, ejecutamos:

```
# cd
# wget http://downloads.sourceforge.net/project/nagios/nrpe-2.x/nrpe-2.15/nrpe-2.15.tar.gz
# tar xvzf nrpe-2.15.tar.gz
# cd nrpe-2.15
# ./configure
# make all
# make install-plugin
# make install-daemon
# make install-daemon-config
# make install-xinetd
```

En ambos casos añadimos la **IP del servidor** al fichero "nrpe" de "xinetd.d" donde pone "`only_from`".

```
# nano /etc/xinetd.d/nrpe
```



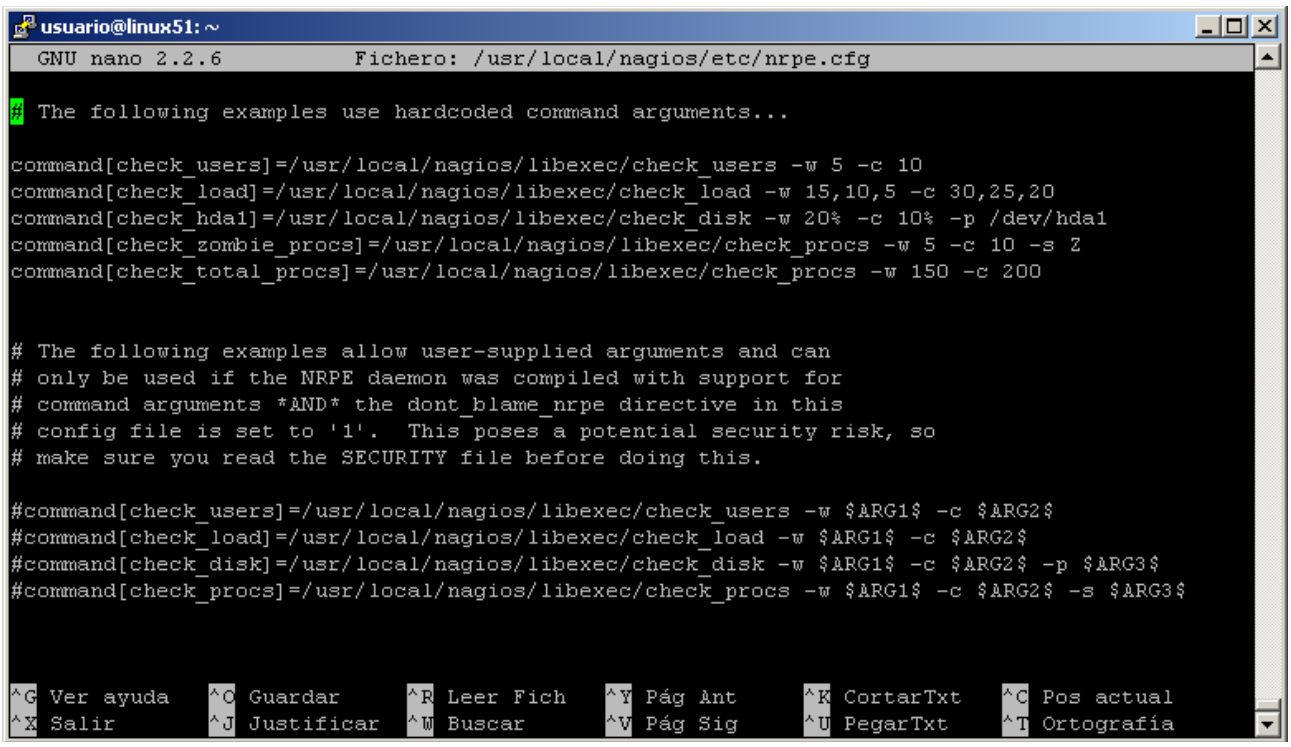
Así mismo, comprobamos que en el fichero `/etc/services` aparece una línea descomentada con "`nrpe 5666/tcp`". En caso de estar comentada o no aparecer la añadiríamos. Para comprobar si esta presente podemos ejecutar:

```
# grep nrpe /etc/services
```

4.3.1.1. Personalizando los comandos NRPE

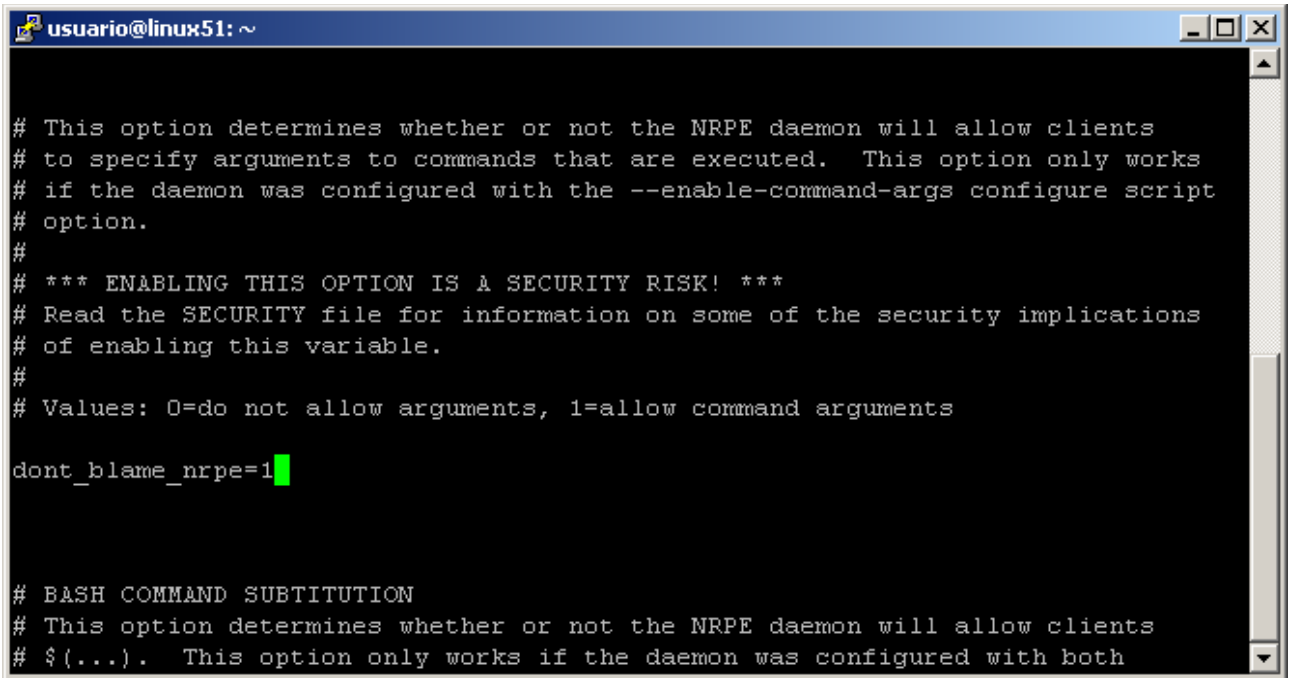
Ahora, si nos vamos al fichero nrpe.cfg vemos como ya tenemos unos cuantos comandos definidos, que podemos editar según queramos. Además podemos añadir nuevos comandos.

```
$ nano /usr/local/nagios/etc/nrpe.cfg
```



Aquí vemos que hay dos grupos de comandos. El primer grupo, el que viene descomentado es el que usa NRPE por defecto, y es el usaremos en caso de haberlo instalado para no recibir argumentos. Por tanto, estos comandos son los que tendemos que editar en los nrpe.cfg de cada Linux para adaptarlo a nuestras necesidades.

Por otra parte, si instalamos NRPE para recibir argumentos utilizaremos el segundo grupo de comandos. Pero para ello habrá que descomentarlos (no hace falta comentar el primer grupo), y además habrá que buscar la variable "`dont_blame_nrpe`" y activarla.



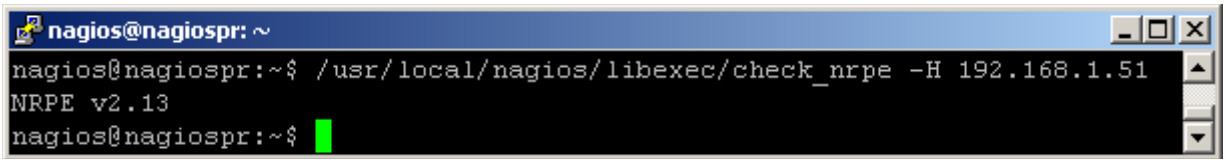
Una vez realizado los cambios oportunos solo queda reiniciar xinetd:

```
# service xinetd restart
```

4.3.1.2. Comprobaciones desde Nagios

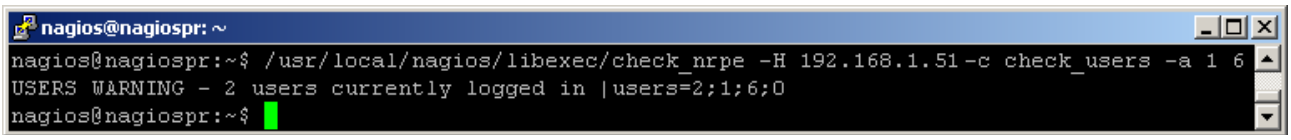
Esto es todo lo que tenemos que hacer en el host Linux, ya solo tenemos que comprobar desde Nagios que funciona.

```
$ /usr/local/nagios/libexec/check_nrpe -H 192.168.1.51
```



Si queremos comprobar el funcionamiento pasando argumentos podemos ejecutar:

```
$ /usr/local/nagios/libexec/check_nrpe -H 192.168.1.51 -c check_users -a 1 6
```



Aclaraciones:

- `-c check_users`: Con el parámetro "`-c`" indicamos el comando a ejecutar.
- `-a 1 6`: Con "`-a`" pasamos los argumentos, en este caso "1" para los Warning y "6" para los Critical.

Obviamente esto funciona porque en nuestro servidor Nagios ya instalamos NRPE para monitorizar los discos duros. Si no tuviésemos instalado NRPE tendríamos que hacerlo.

Como extra, es recomendable reiniciar el host Linux para comprobar que efectivamente NRPE se inicia con el sistema.

4.3.2. Configurando Nagios

Sin embargo, nosotros al igual que con el host Windows queremos que la monitorización sea automática por lo que tenemos que configurar Nagios para ello.

4.3.2.1. Definiendo el host

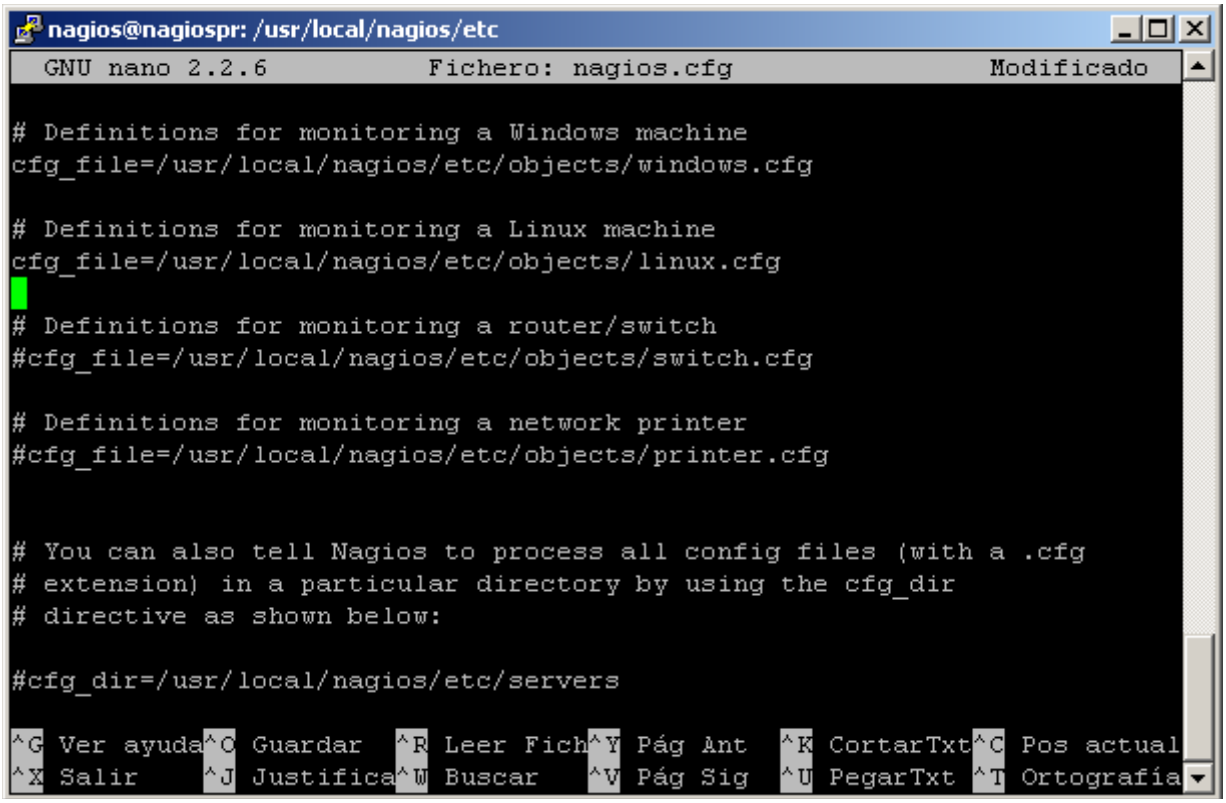
Para definir equipos Windows, Nagios incluía el fichero windows.cfg, pero para equipos Linux no trae ninguno. Como ya comentamos en su momento podemos definir los host donde queramos, siempre que lo indiquemos en el fichero de configuración de Nagios. Por ejemplo, podríamos definir el host Linux en windows.cfg, y funcionaría, pero con el fin de tener un mayor orden crearemos un fichero para monitorizar equipos Linux.

Primero indicamos esto en la configuración:

```
$ cd /usr/local/nagios/etc
$ nano nagios.cfg
```

Debajo de la sección "`OBJECT CONFIGURATION FILE(S)`" añadimos la referencia al que será nuestro fichero. Podemos añadir lo siguiente:

```
# Definitions for monitoring a Linux machine
cfg_file=/usr/local/nagios/etc/objects/linux.cfg
```



Ahora, en la misma ruta que hemos indicado creamos un fichero con el nombre elegido, "linux.cfg" en el ejemplo. En su interior definimos el host Linux de la misma forma que hicimos con el Windows. Para este ejemplo definiremos seis servicios, los cuatro que están en el nrpe.cfg y un par más.

```
$ cd objects
$ nano linux.cfg
```

Estos dos servicios extras son "PING" y "SSH", que no tienen ninguna complicación, simplemente los copiamos de "localhost.cfg". Los usamos para mostrar que no por ser un host Linux hay que utilizar NRPE obligatoriamente para monitorizar algo.

4.3.2.2. Definiendo servicios check_nrpe sin argumentos

Si al instalar NRPE en el host Linux no lo configuramos para recibir parámetros, la definición de los servicios no tiene ninguna dificultad. Por ejemplo *check_disk* sería:

```
define service{
  use                             generic-service
  host_name                       Linux 51
  service_description             Root Partition
  check_command                   check_nrpe!check_disk
  }
```

Como podéis observar utilizamos la misma plantilla que para los servicios de host Windows, pero podríamos usar otra o no usar ninguna (habría que definir todas las directivas obligatorias).

Además, en el `check_command` usamos el comando *check_nrpe* y le pasamos como argumento el comando a ejecutar. Este último es el que esta definido en el nrpe.cfg del host Linux.

Una vez definidos todos los servicios podéis pasar al siguiente punto.

4.3.2.3. Definiendo servicios check_nrpe con argumentos

Sin embargo lo interesante viene al definir los comandos con argumentos. En realidad los cuatro son prácticamente iguales por lo que solo veremos a modo de ejemplo uno de ellos.

```
define service{
  use                             generic-service
  host_name                       Linux 51
  service_description             Root Partition
  check_command                   check_nrpe!check_disk!-a 3072 1024 /
  }
```

Como podéis observar utilizamos la misma plantilla que para los servicios de host Windows, pero podríamos usar otra o no usar ninguna (habría que definir todas las directivas obligatorias). No obstante, donde más difiere este servicio de otros que hemos visto, es en el `check_command`.

Aquí vemos que utilizamos el comando *check_nrpe* y le pasamos dos argumentos. El primero será el comando a ejecutar, y último los parámetros que va a recibir este.

En el ejemplo con "`-a`" indicamos que se van a pasar parámetros al comando, es decir a "`check_disk`". Estos parámetros corresponden a los valores para los Warning, Critical y el *path* (ruta a monitorizar).

Esto lo sabemos gracias a los comandos del nrpe.cfg del host Linux. Allí podemos ver lo siguiente:

```
command[check_disk]=/usr/local/nagios/libexec/check_disk -w $ARG1$ -c $ARG2$
-p $ARG3$
```

Por ello no tenemos que indicar los "`-w`", "`-c`" y "`-p`" al definir el servicio. Así mismo todos los argumentos que espera el comando son obligatorios y no se pueden añadir más de los que espera.

Por supuesto este es el ejemplo para check_disk, otros comandos serán diferentes, como *check_users* que solo espera dos valores, el Warning y el Critical. De todas meneras, como decíamos, siempre podremos personalizar los comandos a nuestras necesidades. Por ejemplo, check_disk por defecto monitoriza en MB, pero si queremos hacerlo en GB también podemos hacerlo.

Para ello la definición del servicio sería así:

```
define service{
  use                             generic-service
  host_name                       Linux 51
  service_description             Root Partition
  check_command                   check_nrpe!check_disk!-a 10 5 / GB
  }
```

A su vez habría que editar el comando en el **nrpe.cfg del host Linux**, quedando así:

```
command[check_disk]=/usr/local/nagios/libexec/check_disk -w $ARG1$ -c $ARG2$
-p $ARG3$ -u $ARG4$
```

4.3.2.4. Definiendo el comando

Por último, antes de reiniciar y ver que todo funciona bien tenemos que definir el comando check_nrpe en el fichero commands.cfg. Si recordáis ya vimos esto y utilizamos check_nrpe para monitorizar el espacio de los discos en Windows. Sin embargo, definimos un comando específico para ello "check_disk". Esto podemos seguir haciéndolo, pero también podemos definir simplemente el comando check_nrpe y pasarle como argumento el comando final que se desea ejecutar, tal y como hemos visto al definir el servicio de arriba.

```
$ nano commands.cfg
```

En su interior, junto al resto de comandos añadimos la definición del siguiente:

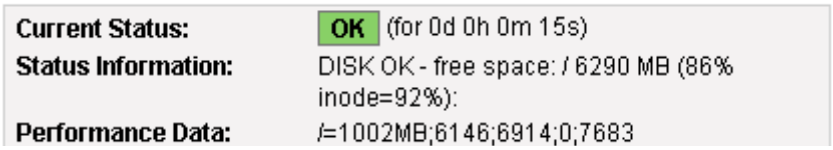
```
# 'check_nrpe' command definition
  define command{
  command_name    check_nrpe
  command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$ $ARG2$
  }
```

En caso de estar usando NRPE sin pasar argumentos desde Nagios, no necesitamos añadir `$ARG2$`.

Una vez terminado esto ya podemos reiniciar Nagios y ver que la monitorización se realiza correctamente. En caso de haber realizado cambios en el fichero nrpe.cfg del host, no olvides reiniciar xinetd también.

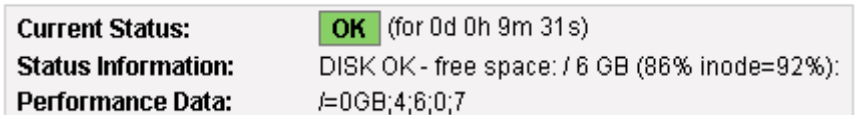
4.3.3. Comprobaciones

Si utilizamos NRPE sin argumentos debemos ver algo así (en el ejemplo se editó el comando del nrpe.cfg para monitorizar el raíz en lugar de hda1):



Si utilizamos NRPE con los argumentos que espera por defecto, el resultado será similar, dependiendo como siempre de lo valores para los Warning y Critical.

En cambio si utilizamos NRPE con argumentos y le añadimos el parámetro "`-u`" para que ponga las cantidades en GB, veremos algo como lo siguiente:



El resultado de 0GB para "Performance Data" no es que sea incorrecto, sino que como podemos observar arriba, el espacio usado (1002MB) no llega a los 1024MB que suponen un GB.

Por tanto, aquí ya es cuestión de las preferencias de cada uno a la hora de definir servicios y comandos. Además, como vimos para los hosts Windows, siempre somos libres de buscar plugins como alternativa a los que ya hemos instalado.

Una vez llegados aquí ya debemos saber monitorizar hosts y servicios en equipos Windows y Linux.

5. CONFIGURANDO LAS ALERTAS VÍA CORREO

Una de las características de Nagios es la de poder enviar notificaciones a ciertas personas cuando ocurre algo. Así, si un equipo esta apagado, tiene un problema de algún tipo, un servicio no funciona etc... Además de plasmarlo en la interfaz de Nagios, podrá enviar una notificación al personal oportuno.

Como ya comentamos también es posible configurar un manejador de eventos, para que se ejecute algo cuando sucede cierta cosa. Sin embargo aquí nos centraremos únicamente en las notificaciones.

Estas notificaciones o alertas pueden realizarse de muchos métodos, pero aquí solo veremos como configurarlas para el correo.

5.1. Instalando y configurando el correo

Las alertas por correo pueden configurarse de muchas formas, aquí lo veremos instalando los paquetes *sSMTP* y *mailutils* que permitirán a nuestra máquina Nagios enviar correos al exterior. Todo el proceso lo haremos **como superusuario**.

```
# apt-get install mailutils ssmtp
```

Ahora tenemos que editar dos ficheros que están en la carpeta "`/etc/ssmtp`":

```
# cd /etc/ssmtp
# nano ssmtp.conf
```

En este fichero solo necesitamos unas pocas opciones. El resultado final podría ser como el siguiente:

```
mailhub=smtp.openmailbox.org:587
UseSTARTTLS=Yes
AuthUser=nagios@openmailbox.org
AuthPass=password
```

Como veis, de las opciones que incluye el fichero solo usamos `mailhub`, con la que indicamos el servidor de correos y el puerto. Adicionalmente añadimos los campos `AuthUser` y `AuthPass` para las credenciales, y la opción `UseSTARTTLS`.

Si necesitáis saber más sobre este MTA y sus opciones siempre podéis buscar más información.

Ahora editamos el fichero "revaliases" y añadimos una línea como la siguiente:

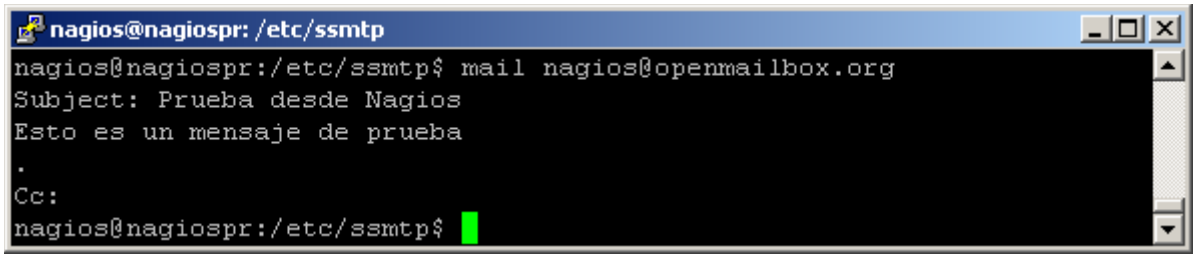
```
nagios:nagios@openmailbox.org:smtp.openmailbox.org:587
```

Dentro del propio documento explica el formato, pero es muy sencillo, en el primer campo se indica una cuenta local, que será del usuario de Nagios. En el segundo la cuenta saliente y por último el servidor de correo junto al puerto.

5.1.1. Comprobaciones

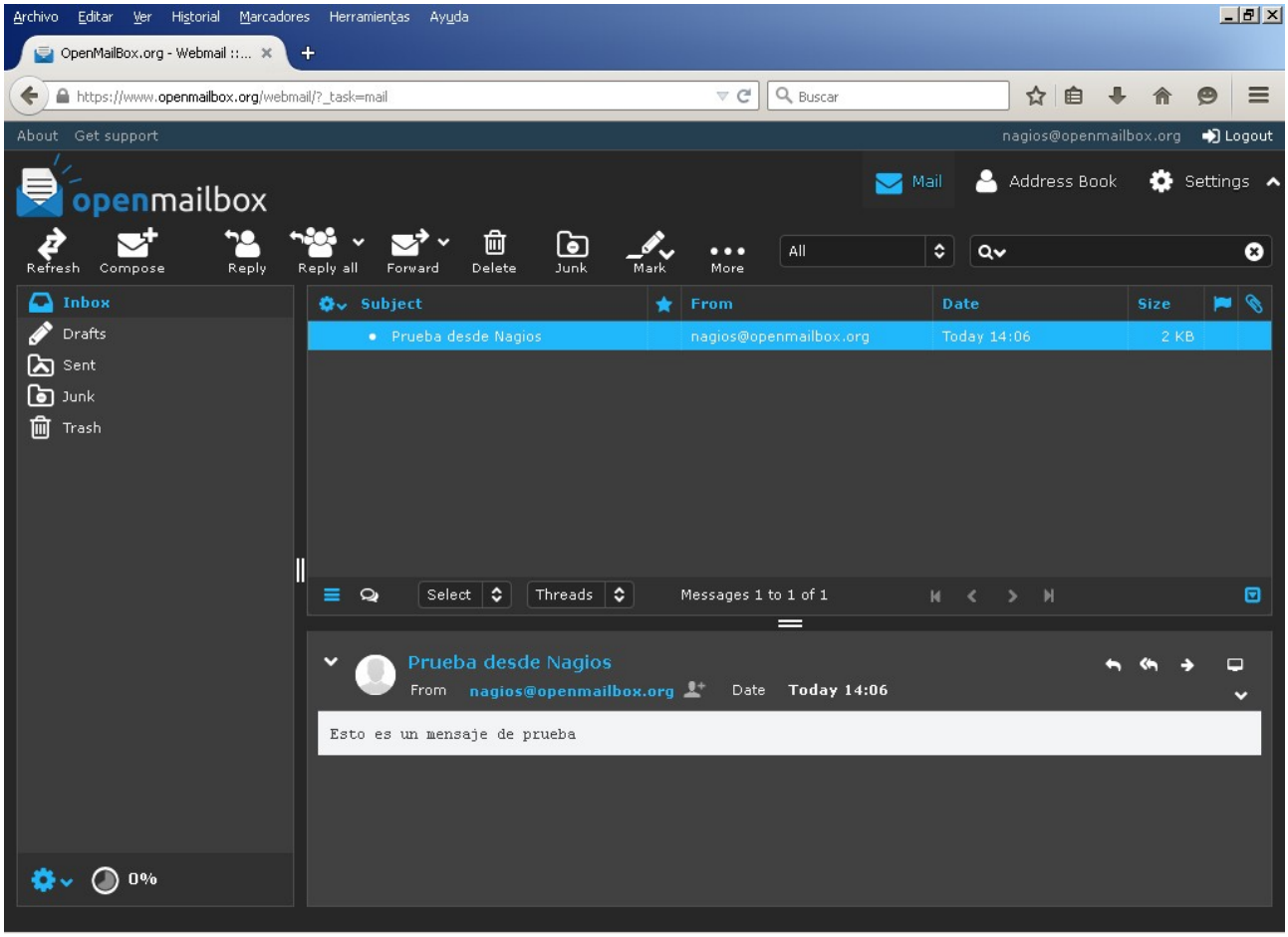
Para comprobar que lo hemos configurado correctamente podemos usar el comando *mail*. Aquí ya volvemos a ser el usuario **nagios** que es el que acabamos de configurar en revaliases.

Para enviar el mensaje basta con poner el comando *mail* seguido del correo. Luego escribiremos el asunto y el mensaje. Este debe terminar con un *intro*, un *punto* y otro *intro*. Al final podremos añadir más destinatarios.



Si queremos podemos escribir todo en una sola línea gracias a los parámetros de *mail*, para saber como podemos leer su ayuda.

Si no tenemos ningún error todo debe de haber ido bien, y si entramos en nuestro correo veremos el email.



6. CONTACTOS

Ya tenemos configurado Nagios y el correo, ya solo quedan los contactos. Si recordáis, en templates.cfg también había una plantilla para los contactos. Ahora vamos a ver qué son y como usarlos para recibir las alertas vía email.

Lo primero que haremos será analizar la plantilla:

```
define contact{
  name                            generic-contact         ;
  service_notification_period     24x7                    ;
  host_notification_period        24x7                    ;
  service_notification_options    w,u,c,r,f,s             ;
  host_notification_options       d,u,r,f,s               ;
  service_notification_commands   notify-service-by-email ;
  host_notification_commands      notify-host-by-email    ;
  register                        0                       ;
  }
```

Aclaraciones:

- `service_notification_period`: Periodo de tiempo en el cual el contacto puede ser notificado acerca de problemas con los servicios.
- `host_notification_period`: Igual que el anterior pero para las notificaciones de los hosts.
- `service_notification_options`: Tipo de notificaciones que serán enviadas para los servicios.
- `host_notification_options`: Igual que el anterior pero con los estados de los hosts.
- `service_notification_commands`: Comando que se ejecuta al enviar una notificación sobre un servicio.
- `host_notification_commands`: Igual que el anterior pero para las notificaciones de los hosts.

Los contactos al igual que otros objetos también tienen su fichero particular. En este caso es "contacts.cfg" y en su interior ya viene definido un contacto llamado *nagiosadmin*. Nosotros nos quedaremos con este contacto tal como esta y únicamente editaremos el *email*.

```
define contact{
  contact_name                    nagiosadmin             ;
  use                             generic-contact         ;
  alias                           Nagios Admin            ;
  email                           nagios@openmailbox.org  ;
  }
```

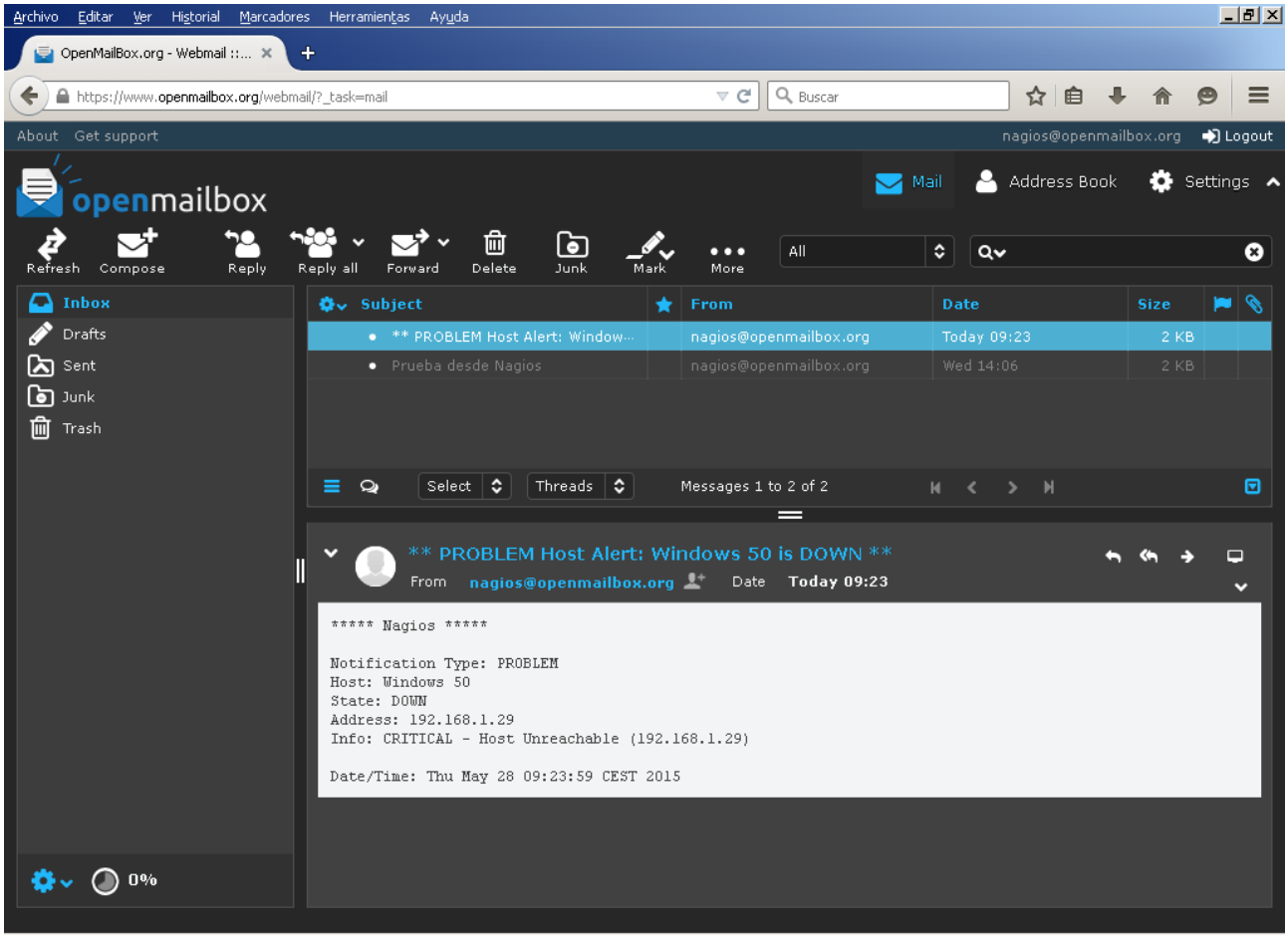
Las directivas con las que viene definido ya nos deben sonar de otros objetos. Sin embargo existen algunas directivas más que no hemos visto aquí. Para obtener más información sobre estas y la definición de los contactos podéis leer la documentación.

Si os fijáis, en el mismo documento hay una sección para los *contactgroup*. Por defecto ya hay un grupo de contactos definido con el contacto nagiosadmin como miembro. La finalidad de esos contactgroup es la misma que la ya vista para los hostgroup y los servicegroup. Si deseáis más información sobre estos podéis leer su documentación.

6.1. Comprobaciones

Una vez hemos configurado el correo y el contacto, si nuestros hosts o servicios tienen un problema recibiremos una notificación en nuestro correo. Para probar esto basta con causar algún problema, en el ejemplo desconectamos a Windows 50 de la red y esperamos a que Nagios cambiará su estado a **DOWN de tipo HARD**.

Una vez llegado a ese punto Nagios envía una notificación que veréis en vuestro correo:



Esto es todo sobre la configuración de Nagios, ya solo es cuestión de afinarla según vuestras necesidades.

Bibliografía

- Pedro Alcaraz Díaz. *Instalación y configuración de Nagios*. Jerez de la Frontera, 2014.
- *Nagios* [Consulta: 22 abril 2015]. Disponible en: https://www.nagios.org/
- *Nagios Core Documentation* [Consulta: 22 abril 2015]. Disponible en: http://nagios.sourceforge.net/docs/nagioscore/4/en/
- *NSClient++* [Consulta: 22 abril 2015]. Disponible en: http://www.nsclient.org/
- *Nagios Support Forum* [Consulta: 22 abril 2015]. Disponible en: http://support.nagios.com/forum/index.php
- *Monitoring-plugins* [Consulta: 24 abril 2015]. Disponible en: https://www.monitoring-plugins.org/
- *Server Fault* [Consulta: 6 mayo 2015]. Disponible en: http://serverfault.com/
- *Proy* [Consulta: 12 mayo 2015]. Disponible en: http://nagios.proy.org/
- *Digital Ocean* [Consulta: 12 mayo 2015]. Disponible en: https://www.digitalocean.com/
- *Wikipedia* [Consulta: 12 mayo 2015]. Disponible en: https://es.wikipedia.org/
- *Ubuntu Forums* [Consulta: 12 mayo 2015]. Disponible en: http://ubuntuforums.org/
- *Debian Wiki* [Consulta: 4 junio 2015]. Disponible en: https://wiki.debian.org/

Rafael Romero González
Junio 2015