Contenido

[Introducción 2](#_Toc9686125)

[Antes de empezar 3](#_Toc9686126)

[Instalación de Nagios Core 4.0.4 4](#_Toc9686127)

[Instalando paquetes previos. 4](#_Toc9686128)

[Creando un usuario y grupo 4](#_Toc9686129)

[Instalando Nagios Core 5](#_Toc9686130)

[Instalando Nagios Plugins 5](#_Toc9686131)

[Configuración de la interfaz Web 6](#_Toc9686132)

# Introducción

En este documento explicaremos como instalar y configurar un servidor Nagios Core 4, un servidor Grafana 6.1 y un servidor InfluxDB 1.7, todos ellos con los plugin necesarios corriendo sobre los sistemas Debian 9 stretch y Raspbian 9 stretch, desde cero.

**¿Qué son y para qué sirven cada uno de los servidores?**

* **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 de 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 plugin específicos para nuevos sistemas.  
  Aclarar aquí que el termino *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…)
* **Grafana** es una herramienta de código abierto para el análisis y visualización de métricas. Se utiliza frecuentemente para visualizar de una forma elegante series de datos en el análisis de infraestructuras y aplicaciones. Suele ir acompañada de otras aplicaciones/plugin que la complementan, en nuestro caso InfluxDB, para proporcionarle las fuentes de datos que alimentan las gráficas y cuadros de mando creados.
* **InfluxDB** es una base de datos basada en series de tiempo (time-series database), ideal para logs o datos de graficas que se generen en vivo (dashboards). Programado en go (Lenguaje Google) permite la interacción vía API HTTP(S)(JSON) e interfaz web y los datos se gestionan con un lenguaje similar a SQL. Este tipo de bases de datos ha experimentado un crecimiento exponencial en los últimos años, con la popularización de IoT, Big Data y otras tecnologías que recogen mucha información en el tiempo. Se necesita un sistema que maneje de forma eficiente esas series de datos, con miles de datos por segundo, y de los que necesitamos hacer cálculos y agregar información de manera eficiente, tales como medias, máximos, búsquedas en el tiempo, etc., y todo ello en tiempo real. Dentro de esta categoría, InfluxDB es una base de datos que supera los esquemas SQL y NoSQL, permitiendo hacer análisis de miríadas de datos en tiempo real.

## Antes de empezar

Para este proyecto se utilizará la siguiente estructura.

* Un Nagios Core 4.0.4 corriendo sobre una Raspberry PI 3 con sistema Raspbian (Debian) 9 stretch con DN suso.servebeer.com
* Un Grafana 6.1 corriendo sobre un portátil con Debian 9 stretch con DN david.servebeer.com
* Una InfluxDB corriendo sobre una Raspberry PI 3 con sistema Raspbian (Debian) 9 stretch con DN mosestrife.zapto.org

El esquema de cómo se comunican entre ellos es el siguiente:

Operario N1

Nagios

Nagflux

InfluxDB

Grafana

Histou

# Instalación de Nagios Core 4.0.4

## Instalando paquetes previos.

Para asegurar el correcto funcionamiento de toda la infraestructura hemos de instalar los siguientes paquetes en el servidor de Nagios.

* Apache 2
* PHP (En nuestro caso la versión 7)
* GCC Compiler and Development libraries
* GD Development libraries.

El proceso de instalación de estos paquetes es sencillo.

Instalación de Apache 2:  
 | # apt-get install apache2

Instalación de PHP:  
 | # apt-get install php  
Deberemos asegurarnos que entre los paquetes instalados de php esté incluido “libapache2-mod-php”. De no ser así deberemos instalarlo manualmente.

Instalación de las librerías necesarias:  
 | # apt-get install make gcc g++ sudo libgd2-xpm libgd2-xpm-dev libpng12-dev libgd-tools libpng3-dev

## 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

Tras crear el usuario procederemos a crear el grupo (aunque el grupo se crea de manera automática al crear el usuario), y le añadimos a los usuarios nagios y www-data (servidor web).

| # groupadd nagios  
 | # usermod –G nagios nagios  
 | # usermod –G www-data,nagios www-data

## Instalando Nagios Core

Para instalar Nagios Core hemos de obtener la URL de la versión que se desee instalar, en nuestro caso la “*Latest stable release*”. Una vez obtenido descargamos, descomprimimos e instalamos Nagios con los siguientes comandos:  
  
 | # 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-htmlurl=/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

Es importante que ejecutemos el siguiente comando, ya que será el encargado de establecer Nagios en el inicio del sistema.  
  
 | # make install-daemoninit

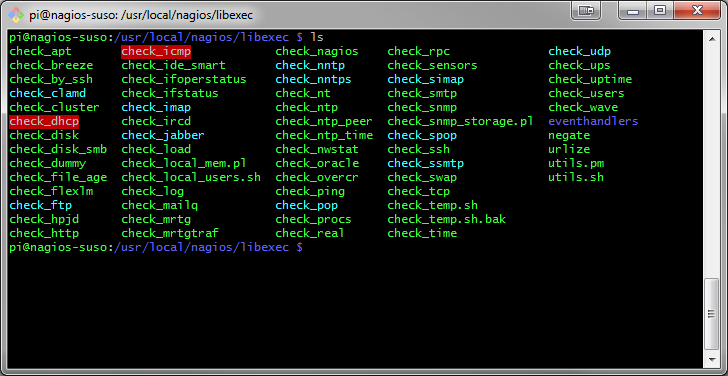
## Instalando Nagios Plugin

Nagios plugin es un añadido que nos permite conectar con servicios más específicos en caso de querer conectar con un host y saber su carga de CPU, RAM HDD y otros servicios o dispositivos.

Para la instalación hemos de obtener el URL del plugin que queremos instalar, para ello hemos de volver a la web de Nagios, pero a la sección de plugin. Ya con la URL ejecutamos los siguientes comandos en la terminal:

| # 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

Si todo ha ido correctamente deberemos ver los plugin en la carpeta */usr/local/nagios/libexec*



## 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 sitio en apache2:  
 | # nano /etc/apache2/sites-available/nagios.conf

Dentro añadimos el siguiente texto:  
   
<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>

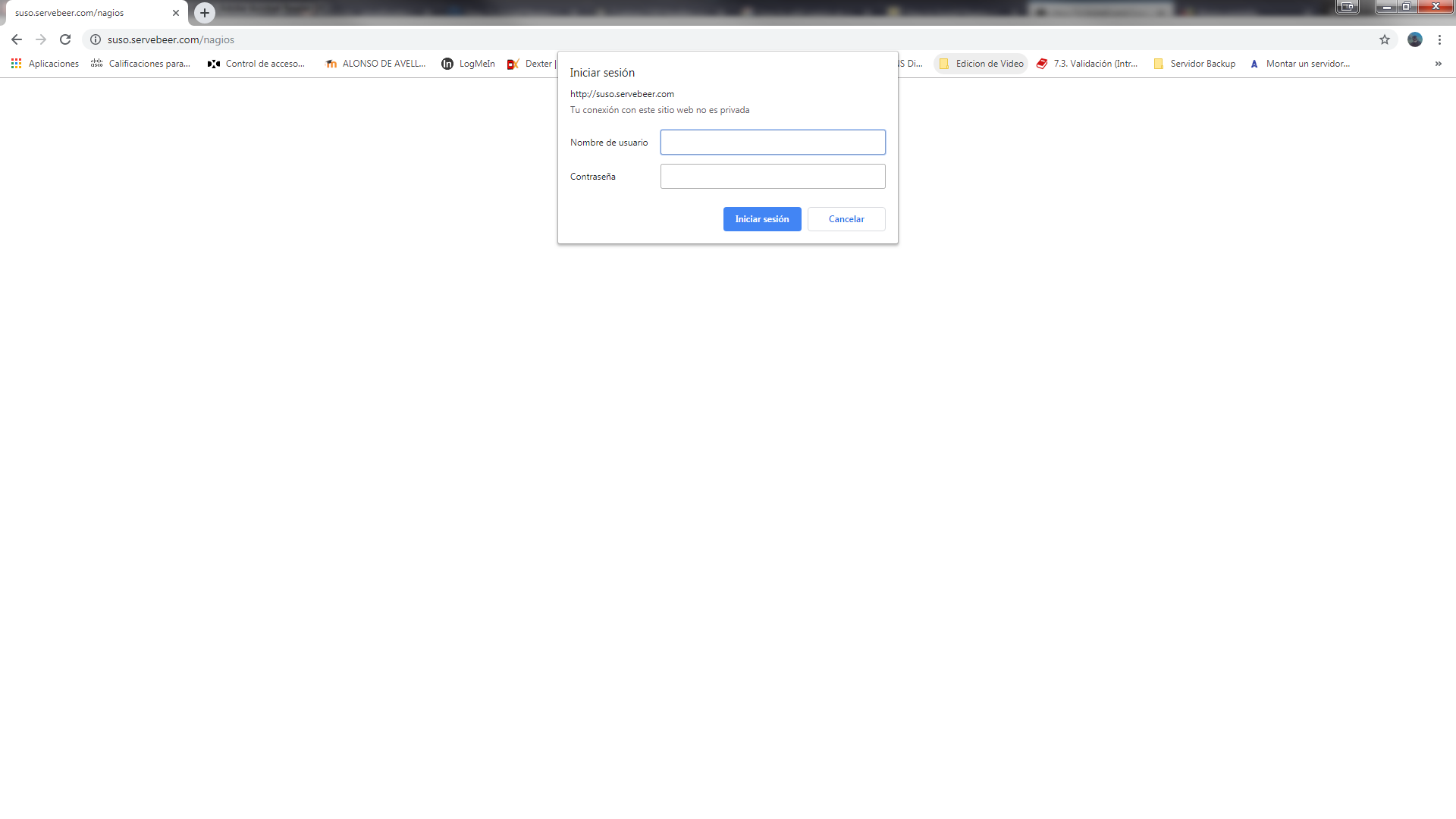
Como cada vez que se define un sitio nuevo en apache, hemos de activar el sitio y recargar el servicio de apache2.  
 | # a2ensite nagios.conf  
 | # systemctl reload apache2.service

A continuación creamos el fichero que será usado para almacenar las contraseñas de los usuarios con permisos para entrar a la interfaz web.   
 | # htpasswd –c /usr/local/nagios/etc/htpasswd.users nagiosadmin

Es importante que si vamos a añadir mas usuarios al fichero no usemos el parámetro –c, este se ha de usar solo la primera vez para “c”rear el fichero. Si volvemos a ejeuctar el comando con el parámetro lo que haremos será sobreescribir el fichero, perdiendo los usuarios que hubiera dentro.

Por último, en el fichero */usr/local/nagios/etc/cgi.cfg* hemos de cambiar la opción “use\_authentication=0” ha de estar en “use\_authentication=1”, por defecto viene a 1, pero es conveniente revisarlo.

Una vez efectuados los pasos anteriores hemos de reiniciar Apache2 y Nagios, para que todo coja efecto.  
 | # systemctl restart apache2  
 | # systemctl restart nagios

Si todo ha ido correctamente se nos solicitará autenticación en el momento de acceder a la ip del servidor de nagios desde un navegador.  


## Instalación y configuración de clientes Linux

En este punto vamos a ver como configurar equipos Linux para ser monitorizados.

Se detallará:

* Algunas de las distintas formas de conseguir monitorizar los equipos
* La utilidad de algunos ficheros y como modificarlos
* El uso de las plantillas para hosts y servicios
* Instalación del software necesario en los equipos.

### Configurando el host

Para realizar la monitorización del cliente debemos instalar en cada uno de los clientes que deseemos monitorizar el agente nrpe de nagios, el cual se encargará de establecer la comunicación entre el servidor de monitorización y el cliente mediante un puerto, que podremos especificar. Mediante este agente podremos llamar, desde el servidor, a los diferentes plugins de monitorización del cliente, siendo esta, una herramienta muy potente, ya que nos permitirá crear nuestros propios plugins de manera sencilla.

Lo primero será actualizar la caché de los repositorios de nuestro sistema.  
 | # apt-cache search nrpe

Tras realizar esta operación podremos descargar e instalar la versión mas actual del agente, para lo cual ejecutaremos el sig. comado:  
 | # apt-get install nagios-nrpe-server nagos-plugins-basic nagios-plugins

Con esto ya tendríamos el agente instalado en los equipos, solo nos quedaría activa el servicio e iniciarlo:   
 | # systemctl enable nagios-nrpe-server  
 | # systemctl start nagios-nrpe-server

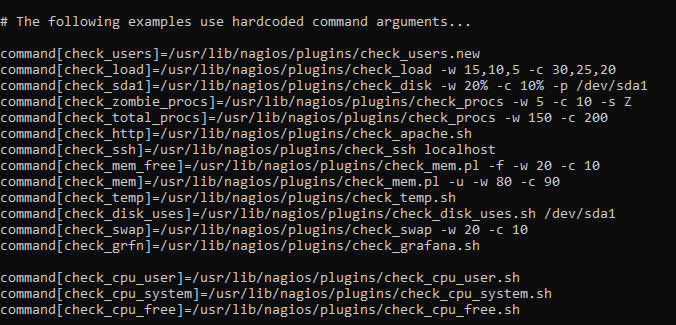
Antes de continaur conviene explicar una cosa. Desde el servidor Nagios, cuando definamos los servicios a monitorizar no podremos ajustar los argumentos según queramos, es decir, los limites para los warnings y los criticals, así como otros parámetros como la ruta a monitorizar para check\_disk etc. Estos ajustes deberán ser indicados en el fichero “nrpe.cfg” de cada equipo. Este fichero se encuentra en */etc/nagios/nrpe.cfg* , deberemos modificar las siguientes líneas para poder monitorizar correctamente el equipo.

La primera línea indica que host tiene permiso de conectar con el equipo y llamar a los plugins.  
 | allowed\_hosts=127.0.0.1, suso.servebeer.com

La siguiente línea nos permitirá incluir argumentos mediante variables que serán pasadas desde el servidor nagios a la hora de llamar al comando/script que nos devolverá las métricas.  
 | dont\_blame\_nrpe=1

### Personalizando los comandos NRPE

Casi al final del fichero “*nrpe.cfg”* encontramos los comandos que vienen definidos por defecto, podemos editarlos, eliminarlos o incluso crear nuevos comandos. En nuestro caso tenemos ya unos cuantos personalizados y creados.



Como podemos ver, el proceso para definir el comando es bastante sencillo.   
command[nombrecomando]=/path/al/script/perlshpythonentreotros.extension argumentos (si los necesita)  
Es importante indicar que el [nombrecomando] es el nombre que usará después el servidor de Nagios para llamar a ese comando, podemos poner el nombre que nos apetezca, pero ha de ser el mismo en el servidor de Nagios.

### Comprobaciones desde Nagios

Para comprobar que el host está correctamente configurado y que el servidor Nagios se comunica correctamente con el podemos ejecutar el siguiente comando en el servidor:  
 | $ /usr/lib/nagios/plugins/check\_nrpe -H IPDELHOSTODNS

En caso de estar todo correcto nos devolverá algo del tipo:   


Si queremos probar el funcionamiento pasando argumentos la sintaxis será, p.ej:  
| $ /usr/lib/nagios/plugins/check\_nrpe -H IPDELHOSTODNS -c comandodefinidoenelhost -a |$arg1 $arg2…



Con el parámetro -c le indicamos el comando que ha de ejecutar en el host  
Con el parámetro -a le indicamos cada uno de los argumentos que necesita que le pasemos al comado.

## Configurando Nagios

Aquí haremos una parada para comentar qué es un objeto en Nagios. Esto no es mmás 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 ellos.

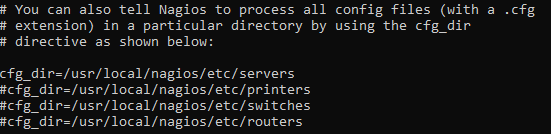
La definición de los objetos se realiza en ficheros con extensión “.cfg”. Estos ficheros pueden ser creados en cualquier parte, pero deben ser incluidos en */usr/local/nagios/etc/nagios.cfg*, y el usuario de Nagios debe tener permisos para leerlo.

Sin embargo, Nagios ya trae unos cuantos ficheros en la ruta */usr/local/nagios/etc/objects* para la definición de los objetos. Para los equipos Linux, en nuestro caso, el único fichero que hay es *“localhost.cfg”*. Si le echamos un vistazo observaremos que ya vienen definidos como ejemplo el host, un hostgroup y varios servicios (locales).

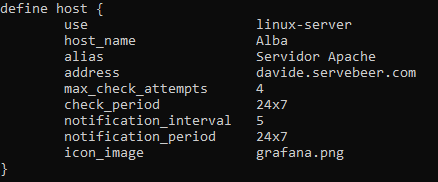
Una buena práctica antes de editar ningún fichero de configuración es hacerle una copia de seguridad. A la hora de guardar los ficheros que editemos y antes de reiniciar el servicio, deberemos comprobar que el usuario nagios tiene permisos de lectura (como mínimo) en esos ficheros.

### Definiendo el host

Como se ha comentado antes, podemos definir los host donde queramos ,siempre que lo indiquemos en el fichero de configuración de Nagios. Con el fin de tener un mayor orden, crearemos un fichero para monitorizar equipos Linux y lo almacenaremos en */usr/local/nagios/etc/servers/*

En el fichero de configuración de Nagios le indicamos la ruta en la que almacenaremos los ficheros de cada host.  
 | # nano /usr/local/nagios/etc/nagios.cfg   
  
Encontraremos una sección del fichero en la que se nos permite incluir rutas en las que almacenar los ficheros .cfg, le indicamos la nuestra.   


Ahora, en la misma ruta que hemos indicado creamos un fichero, en el cual definiremos uno de los host y los servicios del host que se van a monitorizar. Para este ejemplo crearemos el fichero apache.cfg



Aclaraciones:

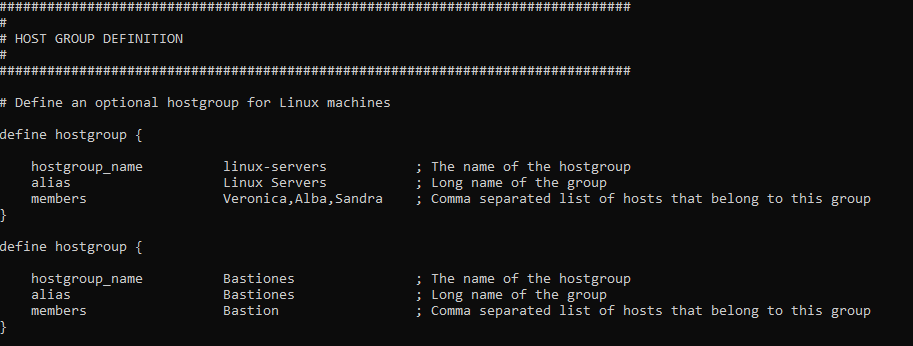
* Use: Con esta directiva se le indica una plantilla de la que heredar la configuración. En caso de conflicto porque una misma directiva se utilice tanto en la plantilla como en la definición del host, siempre tendrá prioridad el valor que se establece en la definición del host. De momento usaremos esta plantilla y mas adelante las veremos mas detalladamente.
* host\_name: Nombre corto usado para identificar el host.
* alias: Nombre o descripción usada para identificar al host
* address: Dirección IP del equipo a monitorizar (se pueden usar nombres DNS)
* max\_check\_attempts: Cantidad de veces que reintentará Nagios comprobar el servicio en caso de error
* check\_period: Horario en el que se comprobará el servicio, hay un fichero con las plantillas de los horarios en */usr/local/nagios/etc/objects/timeperiods.cfg*
* notification\_interval: Intervalo de tiempo entre notificaciones de caída de servicio, expresado en minutos.
* notification\_period: La pesadilla de todo IT, serás notificado 24x7 si el servicio se cae. También hay una definición en la plantilla timeperiods.cfg para workhours (8x5)
* icon\_image: Añade un icono junto al nombre del objeto. Este ha de ser 40x40 en formato png (preferentemente) o jpg.

Aquí se pueden establecer muchas mas 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.

### Definiendo el hostgroup

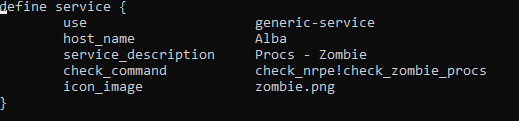
Los hostgroup no son obligatorios pero pueden ser muy útiles. Estos pueden ser usados para dos cosas:

* 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.

En nuestro caso hemos definido los hostgroups y los equipos que lo componen en el fichero */usr/local/nagios/etc/objects/localhost.cfg.*

### Definiendo servicion Linux con check\_nrpe

La definición del servicio la vamos a realizar en el fichero de cada host, aunque se podrían definir los servicios en un fichero independiente. Por ejemplo, el servicio de check\_zombie\_procs (procesos zombi) sería:

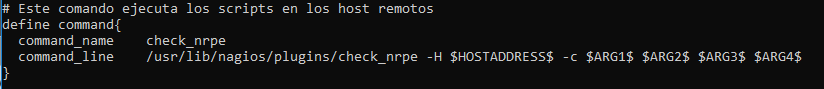


Aclaraciones:

* use: Como en el caso de la definición del host, esta directiva se emplea para utilizar na 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.
* service\_description: Nombre descriptivo para el servicio.
* check\_command: El comando que usará este servicio junto con su variable, en este caso *check\_zombie\_procs*

### Definiendo el comando

Por último, antes de reiniciar y ver que todo funciona bien tenemos que definir el comando check\_nrpe en el fichero */usr/local/nagios/etc/objects/commands.cfg*.



En caso de estas usando NRPE sin pasar argumentos desde Nagios no necesitaríamos añadir $ARG2$ en adelante.  
Una vez terminado esto ya podemos reiniciar Nagios y ver que la monitorizacion se realiza correctamente.