



---

# MONITORIZACIÓN DE SISTEMAS CON NAGIOS, GRAFANA E INFLUXDB

---





## Contenido

Introducción .....	4
Antes de empezar .....	5
Instalación de Nagios Core 4.0.4 .....	6
Instalando paquetes previos. ....	6
Creando un usuario y grupo .....	6
Instalando Nagios Core .....	7
Instalando Nagios Plugin .....	7
Configuración de la interfaz Web .....	8
Instalación y configuración de clientes Linux .....	10
Configurando el host .....	10
Personalizando los comandos NRPE .....	11
Comprobaciones desde Nagios .....	11
Configurando Nagios .....	12
Definiendo el host .....	12
Definiendo el hostgroup .....	14
Definiendo servicios en Linux con check_nrpe .....	14
Definiendo el comando .....	15
Configurando y modificando las plantillas .....	15
Modificando plantillas para los hosts .....	15
Configurando las alertas vía correo .....	17
Instalando y configurando el correo .....	17
Definiendo contactos en Nagios .....	19
Definiendo el comando para enviar notificaciones .....	20
Comprobaciones .....	20
Creando plugins de monitorización .....	21
Creando Event-Handlers .....	23
Instalación de InfluxDB y configuración con Nagios (nagflux). ....	25
Requisitos. ....	25
Instalando InfluxDB .....	25
Configuración de InfluxDB .....	25
Instalación y configuración de Nagflux. ....	26
Requisitos. ....	26
Instalando Nagflux .....	26

Configurando Nagflux.....	27
Comprobaciones .....	29
Configuración y comandos de Nagios para enviar PerfData a InfluxDB. ....	30
Comprobaciones finales.....	31
Instalación de Grafana .....	32
Requisitos .....	32
Instalación .....	32
Acceso y configuración.....	32
Acceso mediante túnel.....	32
Configuración .....	33
Creando un dashboard.....	36
Añadir paneles al dashboard.....	37
Hacer queries para los paneles .....	40
Instalar y configurar Histou.....	41
Instalacion de Histou .....	41
Configuración de Histou .....	41
Comprobaciones .....	42
Integración de Histou con Nagios .....	43
Comprobaciones .....	44
Bibliografía .....	45

## 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?

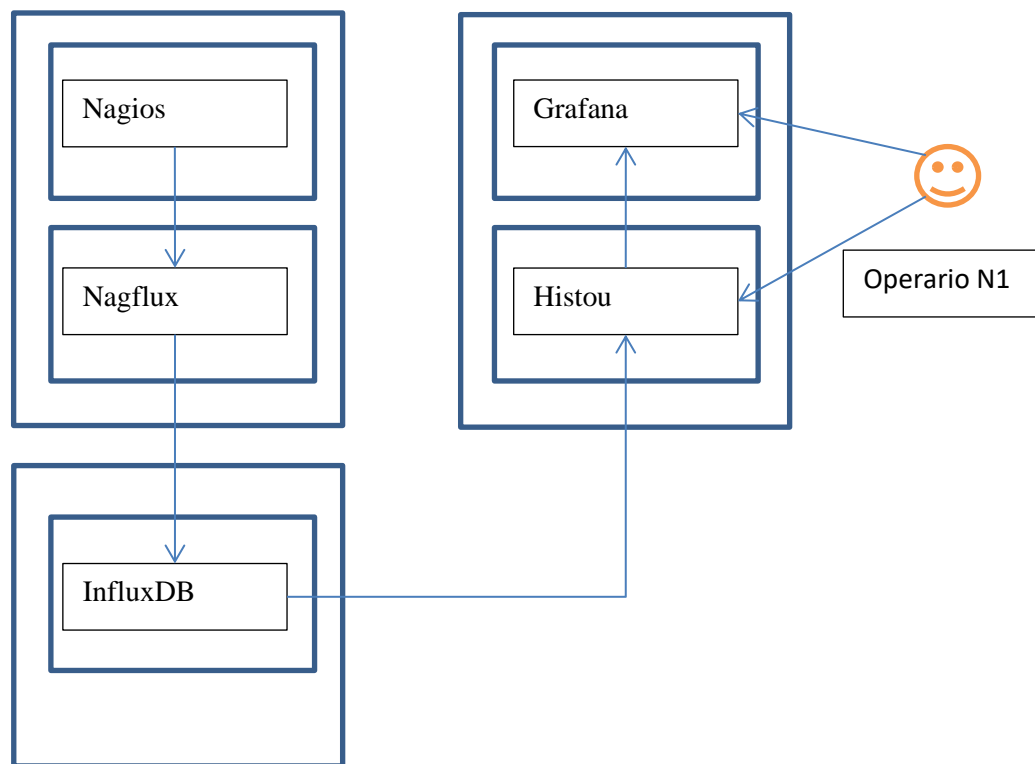
- **Nagios** es un sistema de monitorización de redes de código abierto ampliamente utilizado, que vigila los equipos (hardware) y *servicios* que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. Entre sus características principales figuran la monitorización de servicios 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.\*\*\*\*\*.com
- Un Grafana 6.1 corriendo sobre un portátil con Debian 9 stretch con DN davide.\*\*\*\*\*.com
- Una InfluxDB corriendo sobre una Raspberry PI 3 con sistema Raspbian (Debian) 9 stretch con DN mores\*\*\*\*\*.\*\*\*\*\*.org

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



## 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, descomprimos 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
```

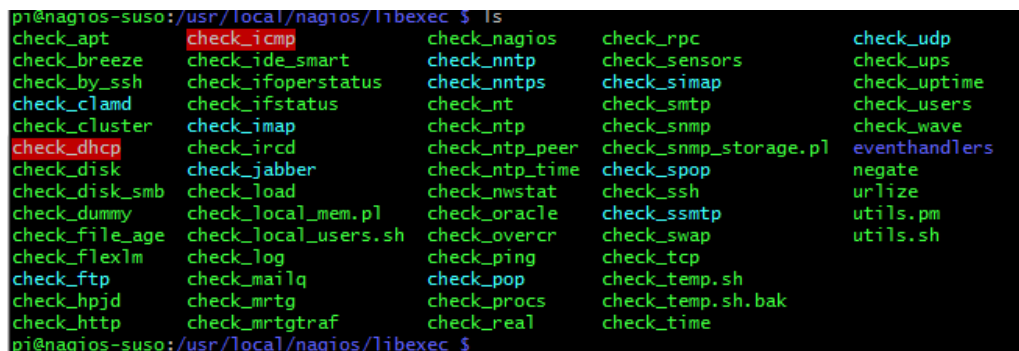
## Instalando Nagios Plugin

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`



```
pi@nagios-suso:/usr/local/nagios/libexec $ ls
check_apt      check_icmp      check_nagios    check_rpc       check_udp
check_breeze   check_ide_smart check_nntp       check_sensors   check_ups
check_by_ssh    check_ifoperstatus check_nntp      check_simap     check_uptime
check_clamd     check_ifstatus  check_nt        check_smtp      check_users
check_cluster   check_imap      check_ntp       check_snmp      check_wave
check_dhcp      check_ircd      check_ntp_peer  check_snmp_storage.pl eventhandlers
check_disk      check_jabber    check_ntp_time  check_spop      negate
check_disk_smb check_load      check_nwstat    check_ssh       urlize
check_dummy     check_local_mem.pl check_oracle     check_ssmtp     utils.pm
check_file_age  check_local_users.sh check_overcr     check_swap      utils.sh
check_flexlm    check_log       check_ping      check_tcp
check_ftp       check_mailq     check_pop       check_temp.sh
check_hpjd      check_mrtg      check_procs     check_temp.sh.bak
check_http      check_mrtgtraf  check_real      check_time
```



## 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
```

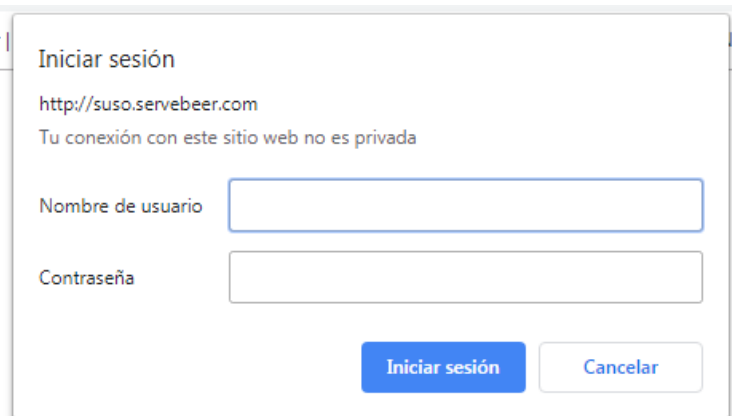
Es importante que si vamos a añadir más usuarios al fichero no usemos el parámetro `-c`, este se ha de usar solo la primera vez para "crear" el fichero. Si volvemos a ejecutar el comando con el parámetro lo que haremos será sobrescribir el fichero, perdiendo los usuarios que hubiera dentro.

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.



A screenshot of a web browser's login dialog box. The title is "Iniciar sesión". Below the title, the URL "http://sus0.servebeer.com" is displayed, followed by a warning: "Tu conexión con este sitio web no es privada". There are two input fields: "Nombre de usuario" and "Contraseña". At the bottom right, there are two buttons: "Iniciar sesión" (highlighted in blue) and "Cancelar".

## 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 más actual del agente, para lo cual ejecutaremos el sig. comando:

```
| # apt-get install nagios-nrpe-server nagios-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 continuar 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. 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.*****.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.

```
# The following examples use hardcoded command arguments...

command[check_users]=/usr/lib/nagios/plugins/check_users.new
command[check_load]=/usr/lib/nagios/plugins/check_load -w 15,10,5 -c 30,25,20
command[check_sda1]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -p /dev/sda1
command[check_zombie_procs]=/usr/lib/nagios/plugins/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/lib/nagios/plugins/check_procs -w 150 -c 200
command[check_http]=/usr/lib/nagios/plugins/check_apache.sh
command[check_ssh]=/usr/lib/nagios/plugins/check_ssh localhost
command[check_mem_free]=/usr/lib/nagios/plugins/check_mem.pl -f -w 20 -c 10
command[check_mem]=/usr/lib/nagios/plugins/check_mem.pl -u -w 80 -c 90
command[check_temp]=/usr/lib/nagios/plugins/check_temp.sh
command[check_disk_uses]=/usr/lib/nagios/plugins/check_disk_uses.sh /dev/sda1
command[check_swap]=/usr/lib/nagios/plugins/check_swap -w 20 -c 10
command[check_grfn]=/usr/lib/nagios/plugins/check_grafana.sh

command[check_cpu_user]=/usr/lib/nagios/plugins/check_cpu_user.sh
command[check_cpu_system]=/usr/lib/nagios/plugins/check_cpu_system.sh
command[check_cpu_free]=/usr/lib/nagios/plugins/check_cpu_free.sh
```

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:

```
pi@nagios-suso:/usr/lib/nagios/plugins $ ./check_nrpe -H mosestrife.zapto.org
NRPE v3.0.1
pi@nagios-suso:/usr/lib/nagios/plugins $
```

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...
```

```
pi@nagios-suso:~ $ /usr/lib/nagios/plugins/check_nrpe -H davide.servebeer.com -c check_sda1
DISK OK - free space: / 21075 MB (95% inode=97%);| / =1054MB;18670;21004;0;23338
pi@nagios-suso:~ $
```

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 comando.

## 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 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 hosts 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.

```
# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
#cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers
```

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`

```
define host {
    use                linux-server
    host_name          Alba
    alias              Servidor Apache
    address             davide.servebeer.com
    max_check_attempts 4
    check_period        24x7
    notification_interval 5
    notification_period 24x7
    icon_image          grafana.png
}
```

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 más adelante las veremos más 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`.

```
#####
#
# HOST GROUP DEFINITION
#
#####
# Define an optional hostgroup for Linux machines
define hostgroup {
    hostgroup_name    linux-servers      ; The name of the hostgroup
    alias             Linux Servers      ; Long name of the group
    members           Veronica,Alba,Sandra ; Comma separated list of hosts that belong to this group
}

define hostgroup {
    hostgroup_name    Bastiones          ; The name of the hostgroup
    alias             Bastiones          ; Long name of the group
    members           Bastion            ; Comma separated list of hosts that belong to this group
}
```

## Definiendo servicios en 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:

```
define service {
    use                generic-service
    host_name          Alba
    service_description Procs - Zombie
    check_command       check_nrpe!check_zombie_procs
    icon_image         zombie.png
}
```

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.
- `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`.

```
# Este comando ejecuta los scripts en los host remotos
define command{
  command_name    check_nrpe
  command_line     /usr/lib/nagios/plugins/check_nrpe -H $HOSTADDRESS$ -c $ARG1$ $ARG2$ $ARG3$ $ARG4$
}
```

En caso de estar 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 monitorización se realiza correctamente.

## Configurando y modificando las plantillas

Ya hemos visto como definir hosts, servicios y comandos, 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.

### Modificando 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
- `windows-server`: Plantilla para servidores Windows.
- `generic-printer`: Plantilla para monitorizar impresoras.
- `generic-switch`: Plantilla para monitorizar switches.

#### Plantilla *generic-host*

Al ser la plantilla genérica, usada por el resto de plantillas, hay en ella ciertas directivas que conviene tener en cuenta para personalizar nuestro sistema de monitorización. La mayoría de sus directivas tiene valor `"0"` (desactivado) o `"1"` (activado).

Aunque definamos en esta plantilla algunas directivas con un valor, luego en la plantilla del host donde la apliquemos podemos llamar a cualquier directiva de esta misma plantilla para asignarle un valor diferente si fuese necesario.

```
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            workhours
  register                      0
  use                           host-grafana
}
```



Algunas de las directivas que podemos encontrar en esta plantilla son:

- name: Nombre utilizado para la directiva
- notifications\_enabled: Si está activada enviará notificaciones al/los contacto/s definidos.
- event\_handler\_enabled: Habilita o no el control de eventos, útil para lanzar un evento en caso de alerta, p.ej: levantar un servicio tras un numero de comprobaciones con resultado critical.
- Flap\_detection\_enabled: Habilita o no la detección de “flapping”.
- Notification\_period: Especifica el nombre del “time period” en el que se permite el envío de notificaciones.

#### *Plantilla linux-server*

```
define host {  
  
    name                linux-server  
    use                 generic-host  
    check_period        24x7  
    check_interval      2  
    retry_interval      1  
    max_check_attempts  4  
    check_command       check-host-alive  
    notification_period workhours  
  
    notification_interval 5  
    notification_options  d,u,r  
    contact_groups        admins  
    register              0  
    use                   host-grafana  
}
```

Aquí podemos observar que algunas directivas se repiten y sobrescriben:

- 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 cada cuanto tiempo se comprobará el objeto. Expresado en minutos.
- retry\_interval: Si un host cambia de estado hará los checks con más frecuencia hasta que cambie a un estado de tipo HARD. Expresado en minutos.
- max\_check\_attempts: Numero de checks que hará cuando el objeto cambie a un estado que no sea OK.
- check\_command: Comando utilizado para, en este caso, comprobar si el host está 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

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.

### 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 está apagado, tiene problemas de algún tipo, un servicio no funciona, etc, además de mostrarlo en la interfaz web, podrá enviar una notificación al personal oportuno.

### Instalando y configurando el correo

Las alertas por correo pueden configurarse de muchas formas, en nuestro caso lo hemos hecho instalando los paquetes mailutils y postfix, que permitirán a nuestro servidor Nagios enviar correos al exterior. En nuestro caso, y para evitar que los correos acaben en la bandeja de spam, hemos creado una cuenta de correo en Gmail, la cual configuraremos como smtp en el servicio de postfix para enviar desde ahí los correos. Comenzamos por instalar los paquetes necesarios:

```
| # apt-get install postfix mailutils
```

Durante la instalación de postfix nos saldrá una ventana en la que podremos configurar el tipo y nombre de nuestro servidor de correo. En nuestro caso elegimos Internet Site y de nombre podemos usar el que más nos guste (nagios.alerts en nuestro caso). Al finalizar la instalación debemos de configurar postfix, para ello editamos el fichero “/etc/postfix/main.cf”. Al final del fichero añadiremos las siguientes líneas, sirven para indicar el certificado de seguridad (usaremos uno local) y las credenciales de la cuenta. Las líneas que ya se encuentren en el fichero se recomienda comentarlas.

```
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/postfix/cacert.pem
smtp_use_tls = yes
```

En el mismo fichero deberemos editar/incluir las siguientes líneas.

```
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = nagios-suso
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = $myhostname, nagios.alerts, nagios-suso, localhost.localdomain, localhost
relayhost = [smtp.gmail.com]:587
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = localhost
inet_protocols = all
```

Una vez tengamos el fichero editado deberemos generar el fichero de credenciales para Gmail, podemos usar nuestro editor de preferencia para crearlo. El nombre del fichero lo hemos definido en “*main.cf*”, por lo que deberá estar en la ruta y tener el nombre que le indicásemos anteriormente. En nuestro caso es “*/etc/postfix/sasl\_passwd*”. *Tan solo tendremos que añadir la siguiente línea:*

```
| [smtp.gmail.com]:587 cuentadegmail@gmail.com:Contraseña
```

*Deberemos de asignarle permisos al fichero, cargar el fichero en el mapeo de archivos de postfix y establecer permisos al fichero que se nos creará como base de datos de ficheros. Para ello ejecutaremos los siguientes comandos:*

```
| # chmod 600 /etc/postfix/sasl_passwd
| # postmap /etc/postfix/sasl_passwd
| # chmod 600 /etc/postfix/sasl_passwd.db
```

*Tras configurar todo esto debemos reiniciar el servicio y probar que podemos enviar correos de forma manual. En el siguiente punto detallamos el proceso de configuración en Nagios.*

```
| # systemctl restart postfix
| # echo “Cuerpo del mensaje de prueba” | mailx -s Prueba destinatario@correo.com
```

*Si todo ha ido bien deberemos de recibir el correo. Si por el contrario, no recibimos nada, podemos revisar el log del servidor para ver donde está el error, dicho log se almacena en “*/var/log/mail.info*”.*

## Definiendo contactos en Nagios

Ya tenemos configurado Nagios y el correo, ya solo quedan los contactos. En el fichero `"/usr/local/nagios/etc/objects/templates.cfg"` hay una plantilla para definir los contactos.

```
define contact {
    name                generic-contact        ; The name of this contact
    service_notification_period 24x7           ; service notifications
    host_notification_period  24x7           ; host notifications can
    service_notification_options w,u,c,r,f,s   ; send notifications for
    host_notification_options  d,u,r,f,s       ; send notifications for
    service_notification_commands notify-service-by-email ; send service notificat
    host_notification_commands  notify-host-by-email   ; send host notification
    register                   0                ; DON'T REGISTER THIS D
```

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"`.

```
define contact {
    contact_name      Jesus                ; S
    use               generic-contact      ; I
    alias             Jesus Nagios Admin   ; F
    email             je:                  )@gmail.com ; <
```

En el mismo fichero viene una definición para grupo de contactos.

```
define contactgroup {
    contactgroup_name  admins
    alias             Nagios Administrators
    members            Jesus,Jose,David
```

## Definiendo el comando para enviar notificaciones

Una vez tengamos configurado el servidor de correo y los contactos, es hora de definir los comandos que se ejecutarán en caso de alerta. Por suerte estos comandos vienen predefinidos tanto para notificar alertas de host como para alertas de servicios. En nuestro caso solo hemos de modificar la parte señalada en negrita en cada comando.

# 'notify-host-by-email' command definition

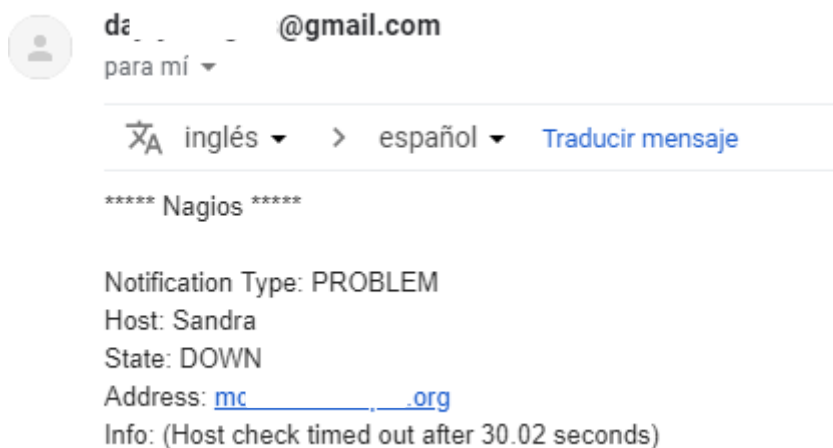
```
define command{
  command_name notify-host-by-email
  command_line /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type:
$NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress:
$HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" |
/usr/bin/mailx -s "*** $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ ***"
$CONTACTEMAIL$
}
```

# 'notify-service-by-email' command definition

```
define command{
  command_name notify-service-by-email
  command_line /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type:
$NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress:
$HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$\n" | /usr/bin/mailx -s "*** $NOTIFICATIONTYPE$ Service Alert:
$HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ ***" $CONTACTEMAIL$
}
```

## Comprobaciones

Una vez configurado el correo y los contactos, si nuestros hosts o servicios tienen un problema recibiremos una notificación en nuestro correo. Para probar esto basta con causar algún problema, por ejemplo, desconectamos el cable del servidor de base de datos (Sandra) y esperamos a que Nagios cambie su estado a **Down** de tipo **Hard**.



## Creando plugins de monitorización

Para crear un servicio propio debemos definir el comando y dirigirlo a un script. Nagios trae por defecto bastantes scripts propios en ficheros en código binario, además nos podemos descargar más de la página oficial de Nagios. Pero también podemos crear los nuestros.

Estos scripts van a estar escritos en Shell Script. Los vamos a crear en la misma carpeta en la que vienen por defecto, pero podemos crearlos en cualquier lugar en que el usuario nagios tenga permisos para llegar, además deberán tener permisos de ejecución (+x) para poder ejecutarlos.

Para empezar con el script debemos saber cómo sacar la métrica que queremos monitorizar. Como ejemplo vamos a comprobar la temperatura de la CPU. Esto se averigua con el siguiente comando:

```
| # cat /sys/class/thermal/thermal_zone*/temp
```

Esto nos devuelve la temperatura que buscamos multiplicada por 1000. De manera que guardaremos dicho valor en una variable y lo dividiremos por 1000:

```
GET_TEMPERATURE=`cat /sys/class/thermal/thermal_zone*/temp`  
CHECK_TEMPERATURE=`expr $GET_TEMPERATURE / 1000`
```

Ahora vamos a definir la salida del comando, lo cual debe devolver el mensaje que escribirá Nagios. Creamos una variable que almacenará el texto, el valor y el performance data que se usará en todas las salidas (OK, WARNING, CRITICAL, UNKNOWN):

```
output="CPU is ${CHECK_TEMPERATURE}°C. | cpu_temp=${CHECK_TEMPERATURE}°C;70;80"
```

Aclaraciones:

La variable contiene un mensaje que nos devolverá el texto "CPU is <temperatura>°C", al otro lado del pipe "|" se establecen los valores del performance data:

- cpu\_temp: El nombre que recibe el dato.
- \${CHECK\_TEMPERATURE}: Es el valor que recibe el dato.
- °C: Las unidades del dato, en este caso Grados Celsius.
- 70: Valor de Warning
- 80: Valor de Critical.

Nagios utiliza 4 tipos de salida, por lo que deberemos definir cuatro mensajes diferentes para cada uno de los tipos de salida.

```
if [ -z ${CHECK_TEMPERATURE} ]; then
    echo "UNKNOWN - ${output}"
    exit 3
elif [ ${CHECK_TEMPERATURE} -le 70 ]; then
    echo "OK - ${output}"
    exit 0
elif [ ${CHECK_TEMPERATURE} -le 80 ]; then
    echo "WARNING - ${output}"
    exit 1
else
    echo "CRITICAL - ${output}"
    exit 2
fi
```

Como podemos observar, los estados de nagios vienen definidos por el código de salida:

- exit 0: Si el valor entra en el umbral definido como ok nagios obtendrá el resultado como OK.
- exit 1: Si el valor obtenido entra en el umbral definido como warning, nagios obtendrá el resultado como WARNING.
- exit 2: Si el valor obtenido entra en el umbral definido como critical nagios obtendrá el resultado como CRITICAL
- exit 3: Si el valor obtenido es incorrecto, nulo o desconocido nagios obtiene el resultado como UNKNOWN.

La salida del código de error es importante, puesto que Nagios no comprueba el mensaje que le pasemos, sino que comprobará el código de error que devuelve. Por ello si es OK debe devolver 0, de no ser así podríamos encontrarnos con algo como esto:

WARNING	05-29-2019 14:41:09	0d 0h 1m 41s	1/4	OK - CPU is 63°C.
---------	---------------------	--------------	-----	-------------------

## Creando Event-Handlers

Con Nagios podemos aplicar ciertos procedimientos para intentar recuperar automáticamente un servicio que está fallando. Los event-handler combinados con nrpe pueden ayudarnos a ello.

Mediante el event-handler podemos ejecutar un comando cada vez que un servicio cambia de estado. Para saber dónde estamos tenemos tres variables.

- `$SERVICESTATE$`: El estado del servicio puede ser OK, WARNING, UNKNOWN O CRITICAL.
- `$SERVICESTATETYPE$`: Puede ser HARD (Ha llegado al número máximo de intentos) o SOFT (aún no ha llegado al máximo de intentos, por lo que aún no se ha mandado la notificación)
- `$SERVICEATTEMPT$`: El número de veces que se ha intentado el check en el estado actual.

Por lo que en un check que tuviera configurados 3 intentos el orden sería:

- CRITICAL SOFT 1: Primera vez que falla.
- CRITICAL SOFT 2: Segunda vez consecutiva que falla.
- CRITICAL HARD 3: Tercera vez consecutiva que falla y momento en el que se mandan las notificaciones.

Para poder utilizar los event handler necesitamos activar la opción de manera global, esto se hace en el fichero de configuración de nagios `"/usr/local/nagios/nagios.cfg"`.

```
| enable_event_handlers=1
```

Ahora ya podemos activar en los servicios que deseemos los event handler e indicar cual será el evento que se ejecutará, normalmente un comando o un script. Como ejemplo vamos a activarlo en el servicio de apache, de manera que si este servicio se llegase a caer, al 4 intento de check ejecutaría el evento handler. Para ello editamos el servicio definido en el fichero `"/usr/local/nagios/etc/servers/apache.cfg"`.

```
define service {
    use                generic-service
    host_name          Alba
    service_description Server - Apache
    check_command       check_nrpe!check_http
    max_check_attempts 4
    event_handler       reinicia_apache
    event_handler_enabled 1
    icon_image         apache.png
}
```



En el ejemplo el event-handler que se va a ejecutar se llama reinicia\_apache, como ya explicamos anteriormente, tenemos que definir los comandos, para este ejemplo lo hemos realizado en el fichero “/usr/local/nagios/etc/objects/commands.cfg”.

```
| define command{
|   command name      reinicia_apache
|   command_line      $USER2$/restart_apache.sh $SERVICESTATE$ $SERVICESTATETYPE$
|                     $SERVICEATTEMPT$ $HOSTADDRESS$
| }
```

En nuestro caso, el comando llama a un script que hemos creado en la ruta “/usr/local/nagios/libexec/eventhandlers”, el cual va a recibir las variables anteriormente citadas y le añadimos otra más, que contiene la dirección (IP o DNS) del host que tiene el servicio caído. El script contiene lo siguiente:

```
case $1 in
  OK)
    ;;

  WARNING)
    ;;

  CRITICAL)
    case $2 in
      case $3 in
        case $4 in
          3)
            echo "Restarting service apache..." >> /usr/local/nagios/var/milog.log
            /usr/lib/nagios/plugins/check_nrpe -H $4 -p 5666 -c wakeup_apache2
            exit 0
          ;;
        esac
      ;;
    esac

    echo "Waking up service apache..." >> /usr/local/nagios/var/milog.log
    /usr/lib/nagios/plugins/check_nrpe -H $4 -p 5666 -c wakeup_apache2
    exit 0
    ;;
  esac
;;

UNKNOWN)
  ;;
esac
```

Como se puede observar, es una estructura condicional, en la que solo nos interesa cuando el servicio está en CRITICAL, en cuyo caso comprobará las variables \$SERVICESTATE\$ y \$SERVICESTATETYPE\$ para realizar las acciones. Como podemos ver, \$4 hace referencia a \$HOSTADDRESS\$, necesaria para que el comando npre sepa a que host atacar.

Para terminar, solo hemos de definir los comandos a los que llamamos en cada estado del case, estos se definen en el host.

```
command[re wakeup_apache2]=/bin/systemctl restart apache2
command[wakeup_apache2]=/bin/systemctl start apache2
```

## Instalación de InfluxDB y configuración con Nagios (nagflux).

### Requisitos.

La instalación de InfluxDB requiere de privilegios administrativos, por lo que el procedimiento se realizará con el usuario root.

Por defecto, el servicio de InfluxDB necesita los siguientes puertos de red:

- El puerto TCP 8086 se utiliza para la comunicación cliente-servidor a través de la API HTTP de InfluxDB.
- El puerto TCP 8088 se utiliza para el servicio RPC para copia de seguridad y restauración. En nuestro caso no lo vamos a usar, pero sería aconsejable para no perder las métricas.

InfluxDB utiliza la hora local del host en UTC para asignar marcas de tiempo a los datos y para fines de coordinación. Deberemos de utilizar el protocolo de tiempo de red (NTP) para sincronizar el tiempo entre hosts. Si los relojes de los hosts no están sincronizados con NTP, las marcas de tiempo en los datos escritos en InfluxDB pueden ser inexactas.

La instalación de InfluxDB la realizaremos haciendo uso de los repositorios oficiales de InfluxData, empresa desarrolladora de InfluxDB. Para ello incluimos la URL del repositorio a nuestro fichero de repositorios.

```
| # curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add – source  
|/etc/osrelease  
| # test $VERSION_ID = "9" && echo "deb https://repos.influxdata.com/debian stretch  
|stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

### Instalando InfluxDB

El proceso de instalación de InfluxDB es bastante sencillo. Tan solo instalar el paquete e iniciar el servicio.

```
| # apt-get update && apt-get install influxdb  
| # systemctl start influxdb
```

### Configuración de InfluxDB

El Sistema tiene valores predeterminados internos para cada configuración de archivo de configuración. Se pueden ver estos ajustes predeterminados con el comando.

```
| $ influxd config
```

El fichero de configuración local *“/etc/influxdb/influxdb.conf”* tiene la mayoría de las configuraciones comentadas. Si deseamos modificar o habilitar alguna configuración, será ahí donde lo tengamos que hacer.

Para habilitar el fichero de configuración, tanto si hemos modificado el predeterminado, como si hemos creado uno nuevo, el comando será:

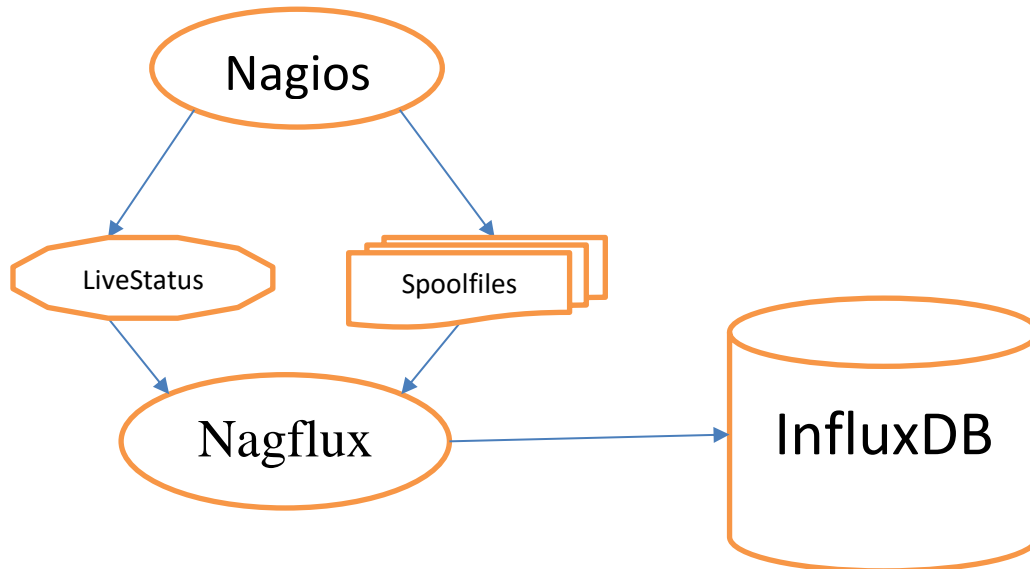
```
| # influxd -config /etc/influxdb/influxdb.conf (o la ruta al fichero creado en su defecto)
```

Para los ficheros de configuración personalizados hemos de asignar los siguientes permisos.

```
| # chown influxdb:influxdb /ruta/al/fichero (tanto al fichero como a la ruta)
```

## Instalación y configuración de Nagflux.

Nagflux es un conector que transforma los datos de rendimiento de Nagios a InfluxDB. Recopila datos de los *Spoolfiles de Nagios* y agrega información de Livestatus. Estos datos se envían a InfluxDB , y serán consultados por Grafana cuando empecemos a crear las gráficas.



### Requisitos.

Al igual que InfluxDB, Nagflux está programado en lenguaje Google, por lo que necesitamos instalar los paquetes y las dependencias necesarias previas a la instalación.

```
| # apt-get install -y golang golang-github-influxdb-usage-client-dev git
```

### Instalando Nagflux.

Una vez cumplido el requisito anterior procedemos a la instalación del conector.

Para instalarlo primero hemos descargar y compilar Nagflux en lenguaje Google.

```
| # export GOPATH=$HOME/gorepo
| # mkdir $GOPATH
| # go get -v -u github.com/griesbacher/nagflux
| # go build github.com/griesbacher/nagflux
| # mkdir -p /opt/nagflux
| # cp $GOPATH/bin/nagflux /opt/nagflux
| # mkdir -p /usr/local/nagios/var/spool/nagfluxperfddata
| # chown nagios:nagios /usr/local/nagios/var/spool/nagfluxperfddata
```

El siguiente paso será crear el servicio y habilitarlo para que se cargue al inicio del sistema.

```
| # cp $GOPATH/src/github.com/griesbacher/nagflux/nagflux.service /lib/systemd/system
| # chmod +x /lib/systemd/system/nagflux.service
| # systemctl daemon-reload
| # systemctl enable nagflux.service
```

## Configurando Nagflux

Una vez instalado el conector que se encargará de transmitir el performance data de Nagios hacia la base de datos InfluxDB, hemos de configurar como va a recolectar ese performance data, como lo va a transformar para introducirlo en la base de datos y como se va a conectar con la base de datos. Lo primero que hemos de hacer es crear el fichero de configuración, este ha de contener la siguiente información.

```
[main]
    NagiosSpoolfileFolder = "/usr/local/nagios/var/spool/nagfluxperfddata"
    NagiosSpoolfileWorker = 1
    InfluxWorker = 2
    MaxInfluxWorker = 5
    DumpFile = "nagflux.dump"
    NagfluxSpoolfileFolder = "/usr/local/nagios/var/nagflux"
    FieldSeparator = "&"
    BufferSize = 10000
    FileBufferSize = 65536
    DefaultTarget = "InfluxDB"

[Log]
    LogFile = ""
    MinSeverity = "INFO"

[Livestatus]
    Type = "tcp"
    Version = "Nagios"

[InfluxDBGlobal]
    CreateDatabaseIfNotExists = true
    NastyString = ""
    NastyStringToReplace = ""
    HostcheckAlias = "hostcheck"

[InfluxDB "nagflux"]
    Enabled = true
    Version = 1.0
    Address = "http://URL InfluxDB:8086"
    Arguments = "precision=ms&u=Usuario&p=contraseña&db=nagflux"
    StopPullingDataIfDown = true

[InfluxDB "fast"]
    Enabled = false
    Version = 1.0
    Address = "http://URL InfluxDB:8086"
    Arguments = "precision=ms&u=Usuario&p=contraseña&db=fast"
    StopPullingDataIfDown = false
```

Para hacerlo mas simple, podemos hacer uso del siguiente *“script”*, donde solo hemos de modificar la información que necesitemos según nuestra infraestructura.

```
cd /opt/nagflux
printf '[main]\n' > config.gcfg
printf '\tNagiosSpoolfileFolder = "/usr/local/nagios/var/spool/nagfluxperpdata"\n' >>
config.gcfg
printf '\tNagiosSpoolfileWorker = 1\n' >> config.gcfg
printf '\tInfluxWorker = 2\n' >> config.gcfg
printf '\tMaxInfluxWorker = 5\n' >> config.gcfg
printf '\tDumpFile = "nagflux.dump"\n' >> config.gcfg
printf '\tNagfluxSpoolfileFolder = "/usr/local/nagios/var/nagflux"\n' >> config.gcfg
printf '\tFieldSeparator = "&"\n' >> config.gcfg
printf '\tBufferSize = 10000\n' >> config.gcfg
printf '\tFileBufferSize = 65536\n' >> config.gcfg
printf '\tDefaultTarget = "all"\n' >> config.gcfg
printf '\n' >> config.gcfg
printf '[Log]\n' >> config.gcfg
printf '\tLogFile = ""\n' >> config.gcfg
printf '\tMinSeverity = "INFO"\n' >> config.gcfg
printf '\n' >> config.gcfg
printf '[InfluxDBGlobal]\n' >> config.gcfg
printf '\tCreateDatabaseIfNotExists = true\n' >> config.gcfg
printf '\tNastyString = ""\n' >> config.gcfg
printf '\tNastyStringToReplace = ""\n' >> config.gcfg
printf '\tHostcheckAlias = "hostcheck"\n' >> config.gcfg
printf '\n' >> config.gcfg
printf '[InfluxDB "nagflux"]\n' >> config.gcfg
printf '\tEnabled = true\n' >> config.gcfg
printf '\tVersion = 1.0\n' >> config.gcfg
printf '\tAddress = "http://127.0.0.1:8086"\n' >> config.gcfg
printf '\tArguments = "precision=ms&u=root&p=root&db=nagflux"\n' >> config.gcfg
printf '\tStopPullingDataIfDown = true\n' >> config.gcfg
printf '\n' >> config.gcfg
printf '[InfluxDB "fast"]\n' >> config.gcfg
printf '\tEnabled = false\n' >> config.gcfg
printf '\tVersion = 1.0\n' >> config.gcfg
printf '\tAddress = "http://127.0.0.1:8086"\n' >> config.gcfg
printf '\tArguments = "precision=ms&u=root&p=root&db=fast"\n' >> config.gcfg
printf '\tStopPullingDataIfDown = false\n' >> config.gcfg
```

Una vez creado el fichero tan solo nos queda inciar el servicio para que cargue la configuracion.

```
| # systemctl start nagflux.service
```

## Comprobaciones

Cuando Nagflux se inicia por primera vez creará una base de datos en InfluxDB, en nuestro caso hemos hecho que esa base de datos se llame nagflux (al crear el fichero de configuración). Si todo está correcto, podemos comprobar que se ha creado ejecutando el siguiente comando:

```
| # curl -G "http://localhost:8086/query?pretty=true" --data-urlencode "q=show databases"
```

Nos ha de devolver una salida como la siguiente, donde apreciaremos el nombre de la base de datos *"nagflux"*.

```
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "databases",
          "columns": [
            "name"
          ],
          "values": [
            [
              "_internal"
            ],
            [
              "nagflux"
            ]
          ]
        }
      ]
    }
  ]
}
```

Si el nombre de la base datos aparece en la lista de bases de datos quiere decir que la configuración es correcta e InfluxDB está preparado para recibir el performance data desde Nagios.

## Configuración y comandos de Nagios para enviar PerfData a InfluxDB.

### Configurando Nagios

El siguiente paso es configurar Nagios para que envíe el performance data a InfluxDB mediante el conector Nagflux.

Esta configuración se ha de realizar en el fichero “/usr/local/nagios/etc/nagios.cfg”, hemos de modificar las siguientes líneas.

```
process_performance_data=1

host_perfdata_file=/usr/local/nagios/var/host-perfdata
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::$HOSTCHECKCOMMAND$
host_perfdata_file_mode=a
host_perfdata_file_processing_interval=15
host_perfdata_file_processing_command=process-host-perfdata-file-nagflux

service_perfdata_file=/usr/local/nagios/var/service-perfdata
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::$SERVICEPERFDATA$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=15
service_perfdata_file_processing_command=process-service-perfdata-file-nagflux
```

Dado que el fichero tiene una cantidad ingente de líneas (aprox. 1500 líneas), para no andar buscandolas una a una, nos ayudamos del siguiente “script”.

```
sed -i 's/^process_performance_data=0/process_performance_data=1/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#host_perfdata_file=/host_perfdata_file=/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#host_perfdata_file_template=.* /host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::$HOSTCHECKCOMMAND$/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#host_perfdata_file_mode=/host_perfdata_file_mode=/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#host_perfdata_file_processing_interval=.* /host_perfdata_file_processing_interval=15/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#host_perfdata_file_processing_command=.* /host_perfdata_file_processing_command=process-host-perfdata-file-nagflux/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#service_perfdata_file=/service_perfdata_file=/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#service_perfdata_file_template=.* /service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::$SERVICEPERFDATA$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#service_perfdata_file_mode=/service_perfdata_file_mode=/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#service_perfdata_file_processing_interval=.* /service_perfdata_file_processing_interval=15/g' /usr/local/nagios/etc/nagios.cfg
sed -i 's/^#service_perfdata_file_processing_command=.* /service_perfdata_file_processing_command=process-service-perfdata-file-nagflux/g' /usr/local/nagios/etc/nagios.cfg
```

### Creando los comandos

Hemos de definir dos comandos en nagios para procesar el performance data por nagflux.

Estos comandos los vamos a definir en el fichero

`"/usr/local/nagios/etc/objects/commands.cfg"`

```
define command {
    command_name    process-host-perfdata-file-nagflux
    command_line    /bin/mv /usr/local/nagios/var/host-perfdata
/usr/local/nagios/var/spool/nagfluxperfdata/$TIMET$.perfdata.host
}

define command {
    command_name    process-service-perfdata-file-nagflux
    command_line    /bin/mv /usr/local/nagios/var/service-perfdata
/usr/local/nagios/var/spool/nagfluxperfdata/$TIMET$.perfdata.service
}
```

Como podemos ver, tenemos definidos dos comandos, uno para el performance data de los hosts y otro para los servicios.

Ya solo nos queda comprobar que la configuración de Nagios es correcta, para ello es conveniente ejecutar el sig. comando.

```
| # /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

Si todo ha ido bien, reiniciamos el servicio para que todo coja efecto y se empiecen a enviar el performance data a la base de datos.

```
| # systemctl restart nagios.service
```

### Comprobaciones finales

Ahora es momento de comprobar que Nagflux realmente está funcionando, la mejor manera es realizar un curl al servidor de InfluxDB en el que le lancemos una query a la base de datos donde veamos las series almacenadas. Para ello, el comando sería:

```
| # curl -G "http://localhost:8086/query?db=nagflux&pretty=true" --data-urlencode "q=show series"
```

```
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "columns": [
            "key"
          ],
          "values": [
            "metrics,command=80,host=Alba,performanceLabel=cpu_temp,service=CPU\\ -\\ Temperature"
          ],
          "metrics,command=check-host-alive,crit-fill=none,host=Alba,performanceLabel=pl,service=hostcheck,unit=%,warn-fill=none"
        },
        {
          "metrics,command=check-host-alive,crit-fill=none,host=Alba,performanceLabel=rta,service=hostcheck,unit=ms,warn-fill=none"
        },
        {
          "metrics,command=check-host-alive,crit-fill=none,host=Bastion,performanceLabel=pl,service=hostcheck,unit=%,warn-fill=none"
        },
        {
          "metrics,command=check-host-alive,crit-fill=none,host=Bastion,performanceLabel=rta,service=hostcheck,unit=ms,warn-fill=none"
        },
        {
          "metrics,command=check-host-alive,crit-fill=none,host=Sandra,performanceLabel=pl,service=hostcheck,unit=%,warn-fill=none"
        },
        {
          "metrics,command=check-host-alive,crit-fill=none,host=Sandra,performanceLabel=rta,service=hostcheck,unit=ms,warn-fill=none"
        },
        {
          "metrics,command=check-host-alive,crit-fill=none,host=Veronica,performanceLabel=pl,service=hostcheck,unit=%,warn-fill=none"
        },
        {
          "metrics,command=check-host-alive,crit-fill=none,host=Veronica,performanceLabel=rta,service=hostcheck,unit=ms,warn-fill=none"
        }
      ]
    }
  ]
}
```



## Instalación de Grafana

### Requisitos

Al igual que en el resto de las instalaciones, necesitamos añadir los repositorios, instalar ciertas dependencias y descargar el fichero .deb. Para ello ejecutamos los siguientes comandos:

```
| # wget https://dl.grafana.com/oss/release/grafana\_6.1.4\_amd64.deb
```

Instalamos una librería y el paquete adduser (por si no se encuentra en nuestro sistema)

```
| # apt-get install -y adduser libfontconfig
```

Y una vez terminado, instalamos el paquete que nos descargamos anteriormente

```
| # dpkg -i grafana_6.1.4_amd64.deb
```

Como último requisito añadimos el repositorio de lanzamientos estables y la firma para garantizar la instalación de paquetes firmados.

```
| # add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
```

```
| # curl https://packages.grafana.com/gpg.key | sudo apt-key add -
```

### Instalación

La instalación es bastante sencilla, tan solo hemos de actualizar los repositorios e instalar grafana desde los mismos.

```
| # apt-get update
```

```
| # apt-get install Grafana
```

Una vez instalado, iniciamos el servicio y lo habilitamos para que se cargue al inicio del sistema

```
| # systemctl daemon-reload
```

```
| # systemctl start grafana-server
```

```
| # systemctl enable grafana-server.service
```

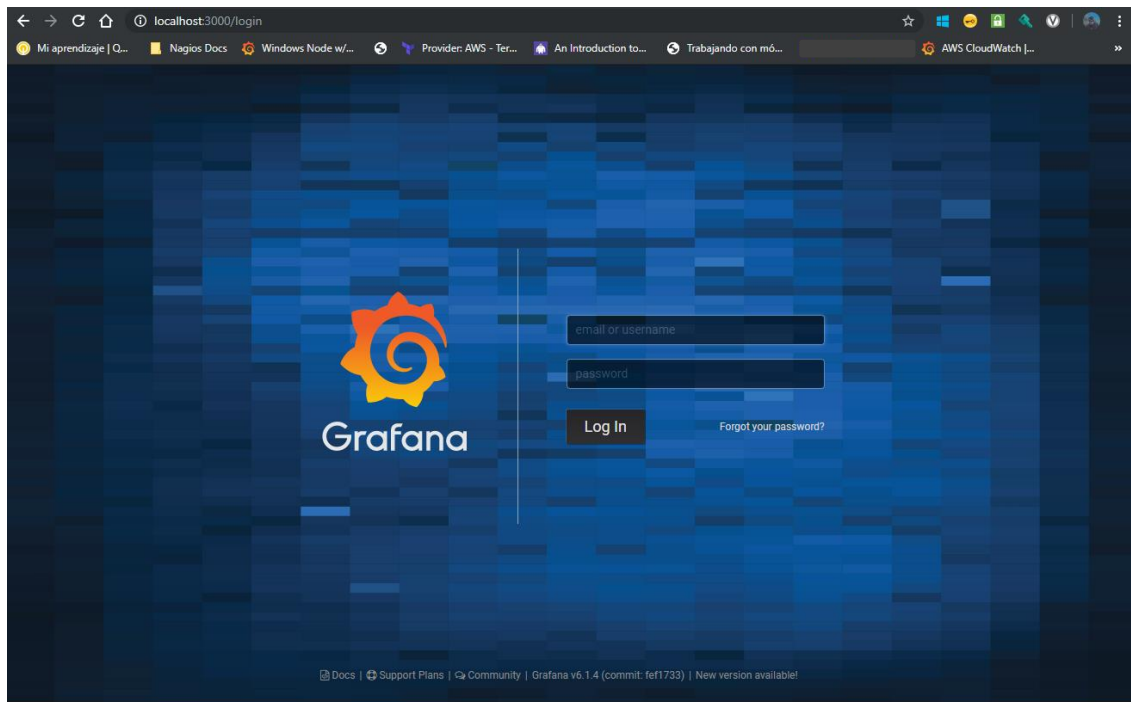
### Acceso y configuración

#### Acceso mediante túnel.

El acceso al interfaz web de Grafana se realizará a través del puerto 3000 del servidor y para mayor seguridad se establecerá un túnel ssh en las máquinas que van a tener conexión con el servidor de Grafana, principalmente serán, el servidor de Nagios, para utilizar Histou, y posteriormente, cada vez que vayamos a realizar una conexión desde cualquier equipo al servidor de grafana, tanto para monitorizar (modo viewer), como para crear dashboards (modo editor). Dicho túnel se realizará bien mediante los túneles de ssh de putty o mediante consola de comandos, en nuestro caso usamos consola de comandos, la sintaxis sería:

```
| # ssh -L 3000:localhost:3000 usuario@ipservergrafana.com
```

Una vez realizado el tunel el procedimiento para conectar es mediante un browser, le pasamos la URL: localhost:3000 y nos saldrá la siguiente ventana:

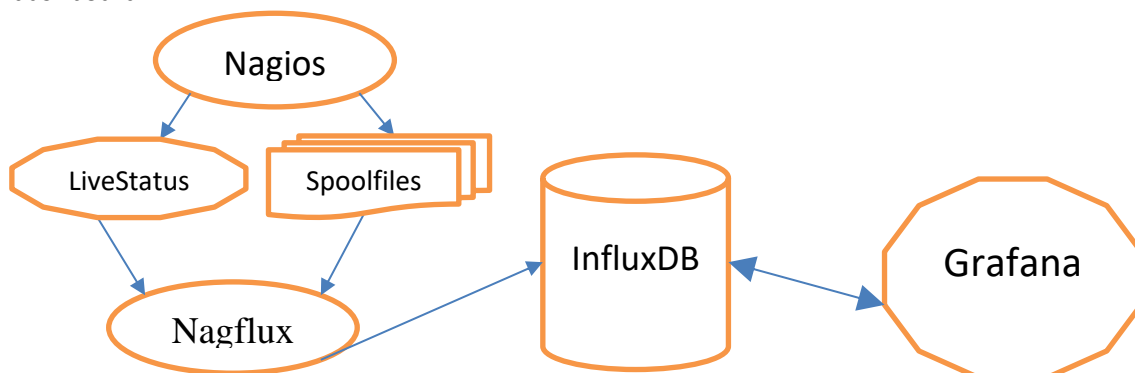


Por defecto, el usuario y la contraseña es admin/admin. Como consejo conviene cambiar dicho usuario y password, así como crear diferentes perfiles de usuario. Entre los perfiles podemos encontrar:

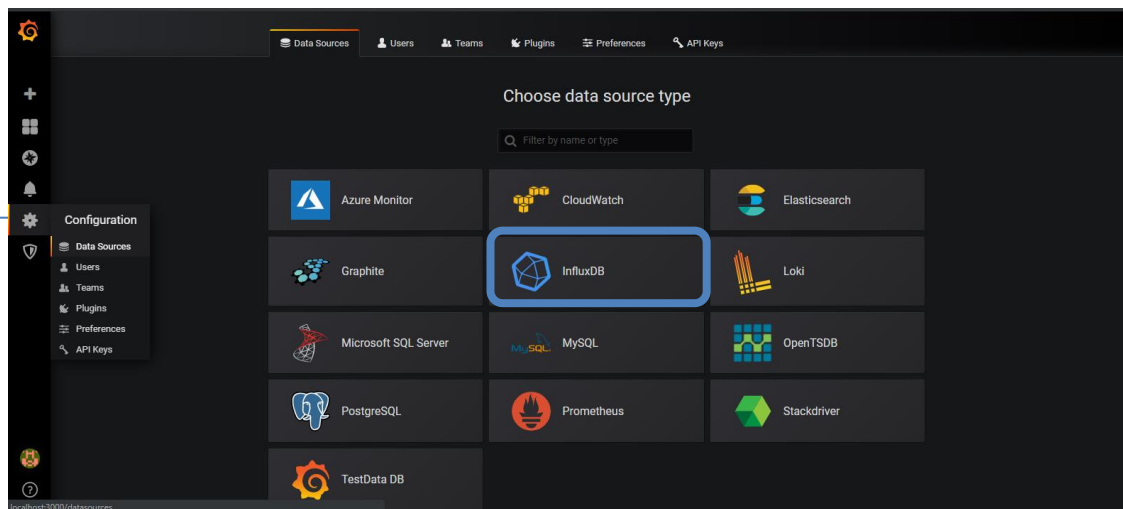
- Admin: Puede realizar operaciones de configuración de los datasources, instalar o desinstalar plugins y crear/editar/borrar dashboards y usuarios
- Editor: Puede crear, editar o borrar dashboards.
- Viewer: Tan solo puede ver dashboards y entrar en el modo kiosk, ideal para departamentos de nivel 1.

### Configuración

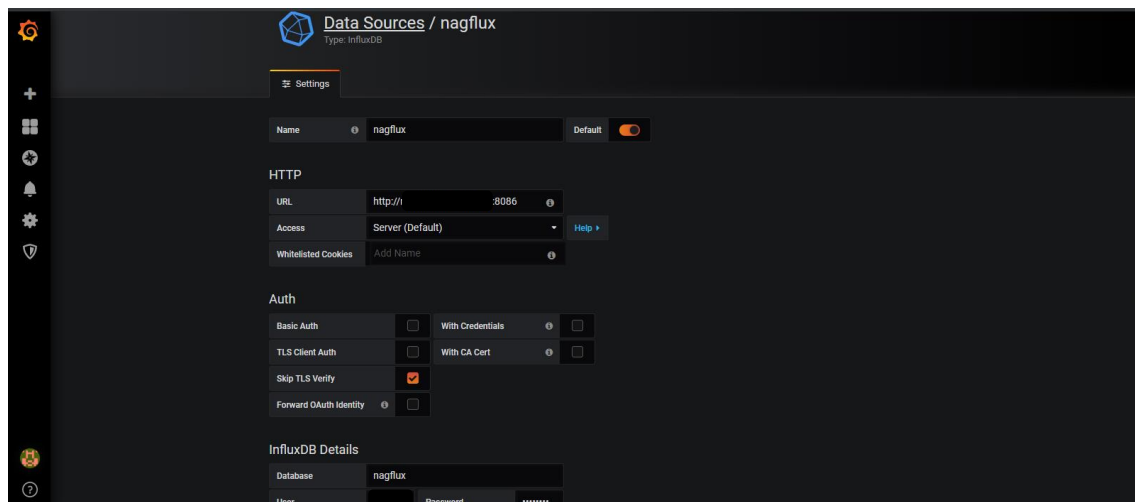
Como hemos comentado anteriormente, Grafana se conectará a la base de datos de InfluxDB para realizar las diferentes queries que serán representadas en los paneles que compondrán el dashboard.



Para ello necesita de un plugin, en el cual definiremos el data source o fuente de datos a la que conectarnos. En nuestro caso el plugin se llama InfluxDB. Para instalar el plugin y configurarlo los pasos son:



- Seleccionamos la opción de configuración/settings, y dentro elegimos la opción data sources.
- Nos debería salir una lista de plugins y data sources, si no es así, deberemos tener un botón de add data source. En la lista elegimos el tipo InfluxDB.



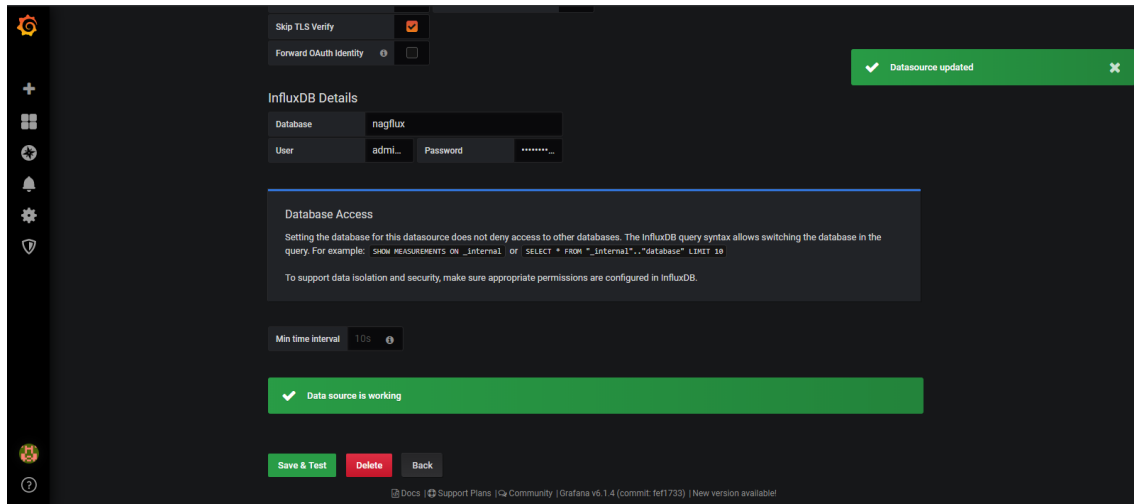
Para configurar el data source hemos de rellenar las opciones de la siguiente manera:

- name: El nombre con el que identificaremos nuestro data source, podemos poner uno libremente.
- URL: la dirección de nuestra base de datos con el puerto.  
[http://urlbasedatos:8086\(puerto](http://urlbasedatos:8086(puerto) por defecto)
- Access: Influxdb tiene dos formas de acceso, mediante server o interfaz web, en nuestro caso usamos el terminal para acceder a la base de datos, por lo que seleccionamos server(default)
- En la parte de Auth hemos de marcar la opción skip tls verify, ya que no hemos configurado ese tipo de autenticación y podría dar error y no conectar.

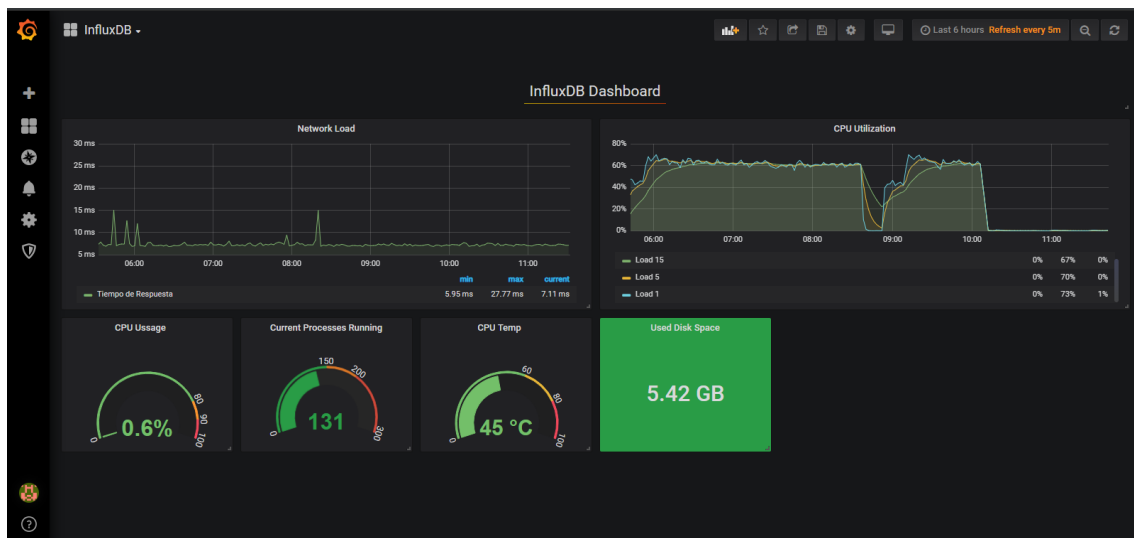
## Instalación y configuración de Nagios Core 4, Grafana 6.1, InfluxDB 1.7 y Histou

- Database: Aquí hemos de poner el nombre de la base de datos sobre la que vamos a realizar las consultas.
- User y Password: El usuario y la Contraseña para conectarnos a la base de datos.

Al final de la ventana tenemos el botón de Save & Test, si hemos introducido los datos correctamente deberíamos de ver los dos checks verdes como en la siguiente imagen.

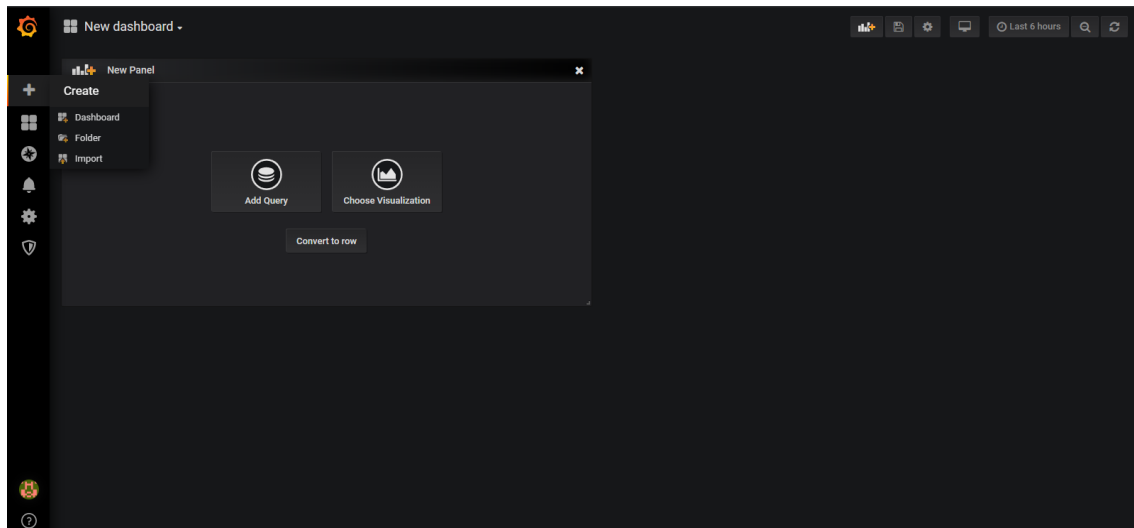


Hasta aquí ya tenemos todo configurado, en este punto ya podemos crear dashboards y añadir paneles de monitorización. Un ejemplo de un dashboard de monitorización sería:

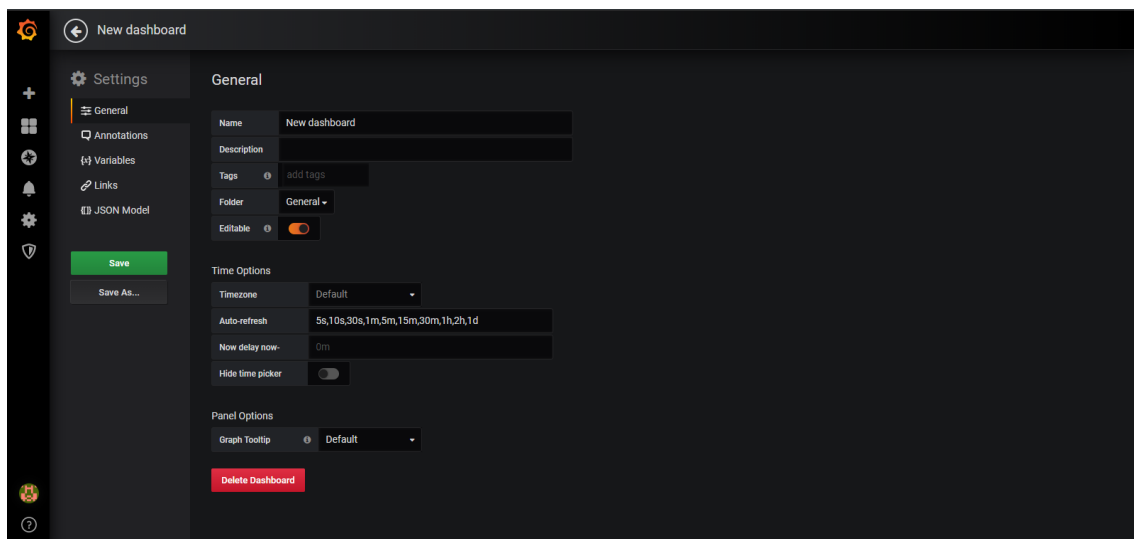


## Creando un dashboard

Para crear un dashboard hemos de seleccionar la opción con forma de cruz en el panel de la izquierda de la interfaz web, se nos creará automáticamente un nuevo dashboard. Podemos empezar a añadir directamente paneles seleccionando las opciones que nos salen en la nueva pantalla, pero vamos a detallar primero los parámetros que definen la configuración del dashboard.



En la ventana anterior vemos en la parte superior izquierda un pequeño menú de iconos, seleccionamos la “rueda dentada” (Dashboard Settings) y nos saldrá una ventana como la siguiente.



Entre todas las opciones, detallamos algunas de ellas por ser las menos obvias.

- Tags: Nos permite añadir etiquetas a nuestro dashboard, para futuras referencias o clasificación.
- Folder: Es el directorio donde se guardará el dashboard, si no existe deberemos crearlo en las opciones de folder.
- Editable: Nos permite habilitar la opción editar o no el dashboard y los paneles.

## Añadir paneles al dashboard.

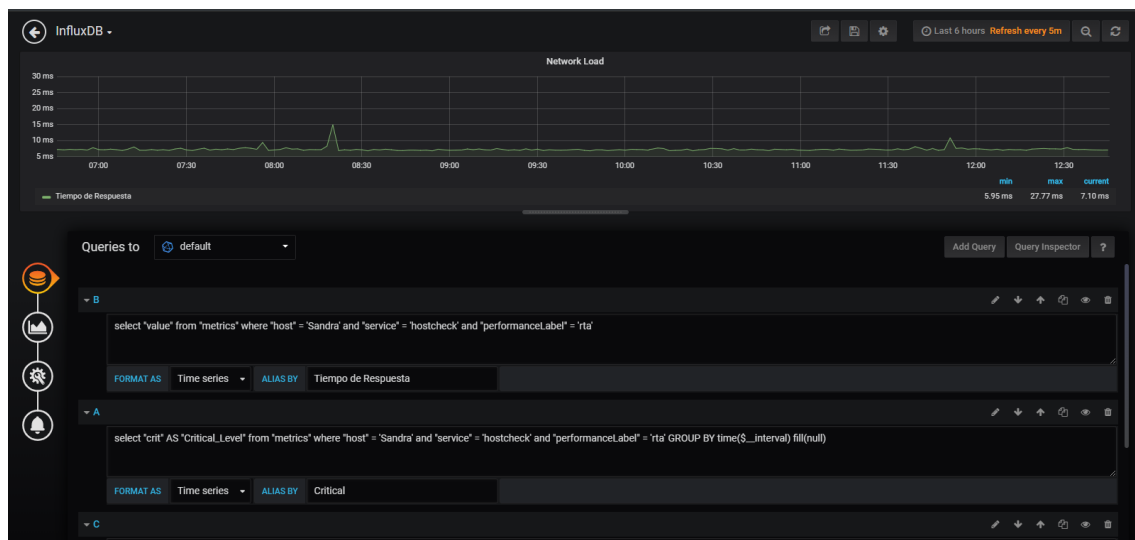
Hay varios tipos de paneles a escoger, entre todos ellos utilizaremos los siguientes:

### Graph:

Nos permite hacer un “historial” de la métrica que utilicemos. Por ejemplo, el uso de la CPU para ver si ha tenido algún pico alto. Tiene varias opciones de personalización como elegir si será un gráfico de barras, líneas o puntos, escoger distinta personalización visual para cada uno, como el grosor de las líneas. También tiene la opción de stack para poder almacenar las líneas unas encima de otras, esto crea una “suma” de todas las métricas.

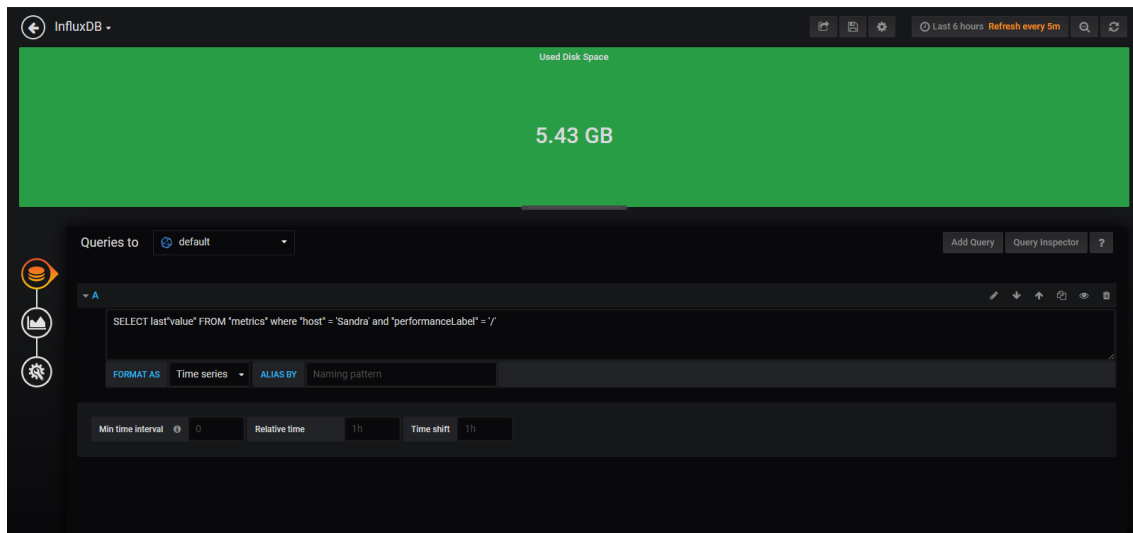
En el apartado de axes podremos elegir las unidades del valor que estamos monitorizando (el porcentaje de CPU), esta opción es muy útil pues hace cambios de unidades automáticas, por ejemplo, si le estamos pasando bytes, al llegar a 1024 nos pondrá automáticamente kb, y así respectivamente con MB, Gb, Tb, etc. Podemos ponerle un mínimo y un máximo, los cuales si no activamos va aumentando o disminuyendo automáticamente, por lo cual si la diferencia entre el máximo y el mínimo es muy pequeña hará “zoom” para que se vean mejor los cambios, en cambio si le ponemos máximo y mínimo nos haremos una mejor idea de cómo está la métrica con respecto al total. También podremos elegir la cantidad de decimales que mostrarnos.

En el apartado de legend podremos ponerle una leyenda a la métrica, para que nos muestre a qué corresponde cada línea, además de poder ponerle las opciones de: min, max, avg, current y total. Pudiendo organizar esta información en formato de tabla y ponerlo a la derecha, para que se vea mejor en caso de ser varias métricas. Finalmente podremos elegir que hará el programa en caso de que haya una métrica con valor “null”, pudiendo igualarlo a 0 o poder conectar ambos extremos.



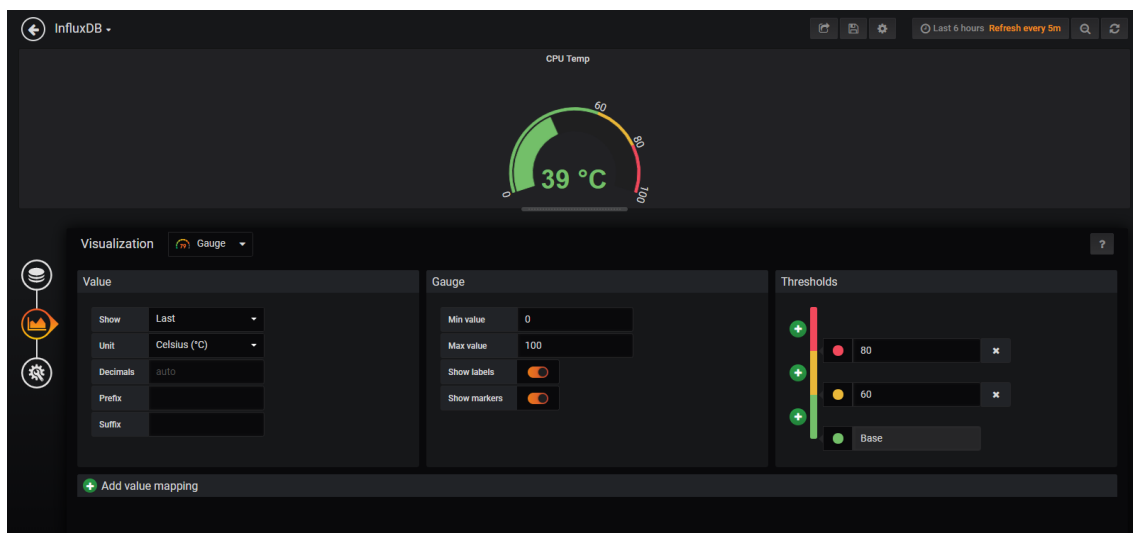
## Singlestat.

Nos mostrará simplemente un valor, ya sea numérico o una cadena de texto. Por ejemplo, lo podemos utilizar para que nos devuelva UP si un servicio está levantado y DOWN si está caído. Entre las opciones de personalización podemos ponerle un Prefix y un Postfix al valor, los cuales se quedarán fijos, aunque cambie dicho valor. En el apartado de coloring podremos escoger los colores que queremos usar en función de las alertas definidas en Nagios, pudiendo personalizarlo con los códigos de color RGB, pudiendo elegir si queremos que pinte el valor solo o el fondo entero del panel.



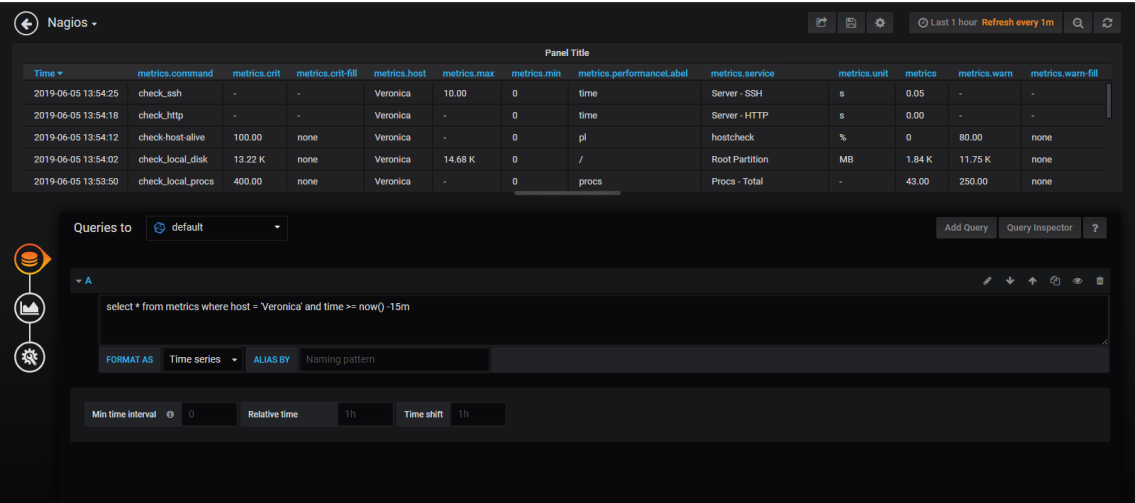
## Gauge:

Este panel es muy parecido al anterior, pero es más útil a la hora de monitorizar una métrica con valor numérico, pues podemos poner un calibre, lo cual nos da una mejor idea de cómo está la métrica respecto a un total fijo. En la personalización encontramos opciones muy parecidas a otros paneles, en el apartado de Gauge podremos ponerle un mínimo y un máximo, en el apartado de Thresholds podremos elegir los valores entre el máximo y el mínimo escogidos para que cambie de color, como el WARNING y el CRITICAL del panel anterior.



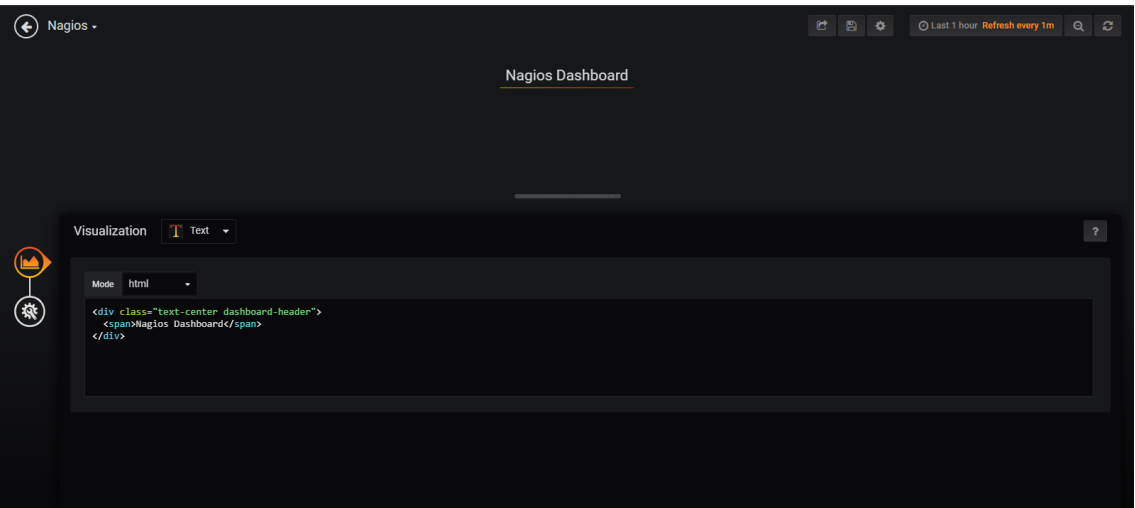
**Table:**

Nos muestra la tabla según como le pedimos, por si queremos comprobar varios valores juntos. En la personalización podremos escoger si queremos algún estilo concreto para cada columna, como definir una cabecera, o incluso que pinte un valor, una celda, o una fila entera en función del valor escogido, además de poder volver a elegir las unidades, decimales y rango de valores para que pinte de diferentes colores.



**Text:**

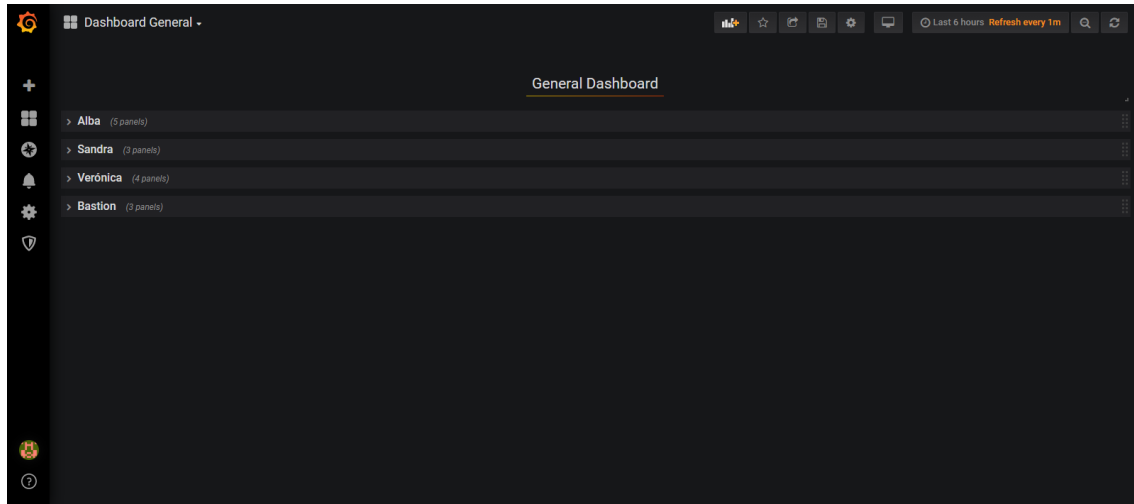
Este panel nos incluirá un texto que le incluyamos. Aceptando formato HTML. Ideal para poner títulos en los dashboards.





## Row:

Para terminar este apartado, al ir a agregar un panel, tenemos la opción de Convert to Row. Este botón nos agregará una fila desplegable, a la cual se añadirán automáticamente todos los paneles que incluyamos debajo suya, a no ser que incluyamos un Row distinto, no son acumulables:



## Hacer queries para los paneles

Las queries son prácticamente iguales a las que se usan en cualquier consulta SQL. En nuestro caso solo vamos a utilizar una tabla, la tabla “metrics”, la cual recoge toda la información que recibe de las métricas del servicio nagflux de nagios. La tabla tiene varias columnas de las cuales, las que más nos interesan son las siguientes:

- time: Marca de tiempo a la que se añade la fila.
- command: comando que se utiliza para comprobar (siempre es “check\_nrpe”).
- crit: Valor al que se considera situación crítica.
- host: Host a comprobar.
- performanceLabel: Etiqueta personalizada que se pone a la métrica recoger.
- service: Servicio que se quiere comprobar (Un service puede llevar tantos performanceLabel como queramos, se pueden guardar varios valores al recoger una información, como por ejemplo utilizada y libre, para luego poder usar la que mejor nos convenga). Utiliza los mismos nombres que Nagios.
- unit: Unidades en las que nos devuelve la métrica.
- value: Valor de la métrica.
- warn: Valor al que se considera situación de Alarma.

Un ejemplo útil es el siguiente:

```
SELECT “value” FROM “metrics” WHERE “host”=‘Alba’ AND “service”=‘CPU - Temperature’
```

Nos devuelve los valores de la temperatura de la CPU de Alba, si queremos que simplemente nos muestre el último, pondríamos: *last(“value”)*.

## Instalar y configurar Histou

Histou está diseñado para mostrar plantillas en Grafana desde el perfdato de Nagios. Por un lado, Nagflux envía las métricas obtenidas por Nagios a InfluxDB, por otro lado, Grafana se usa para mostrar en paneles ese performance data. Histou agrega un panel personalizado a Grafana, que mostrará paneles dinámicos, dependiendo del host y el nombre del servicio que se pasará mediante una URL.

Histou obtiene las primeras informaciones adicionales de InfluxDB y luego selecciona una plantilla. Las plantillas pueden ser archivos JSON o PHP que contienen un panel de Grafana, también tienen una regla, que define cuando se usa esta plantilla.

## Instalación de Histou

Para instalar Histou debemos ejecutar los siguientes comandos en la terminal del servidor donde se encuentra Grafana instalado.

```
| # cd /tmp
| # wget -O histou.tar.gz https://github.com/Griesbacher/histou/archive/v0.4.3.tar.gz
| # mkdir -p /var/www/html/histou
| # cd /var/www/html/histou
| # tar xzf /tmp/histou.tar.gz --strip-components 1
| # cp histou.ini.example histou.ini
| # cp histou.js /usr/share/grafana/public/dashboards/
```

## Configuración de Histou

Lo primero que debemos hacer es incluir el dns o ip del origen de datos (InfluxDB) en el fichero `"/var/www/html/histou/histou.ini"`.

```
| # vi /var/www/html/histou/histou.ini
```

```
[influxdb]
url = http://IPoDNSdeInfluxDB:8086/query?db=nagflux
hostcheckAlias = "hostcheck"
```

Después hemos de configurar el fichero que se va a encargar de crear las plantillas, debemos indicar en la variable de la URL la ruta IP o DNS del servidor de grafana, ha de ser mediante ip o dns, no siendo posible usar localhost, ya que este fichero va a ser llamado desde fuera del servidor local de grafana.

Para ello hemos de editar el fichero `"/usr/share/grafana/public/dashboards/histou.js"`, para hacerlo más fácil podemos usar el siguiente comando, cambiando los datos contenidos entre `<>` por los de nuestro servidor Grafana.

```
|# sed -i 's/localhost/<datosservergrafana>/g' /usr/share/grafana/public/dashboards/histou.js
```

## Comprobaciones

Histou devuelve datos a Grafana para que puedan ser visualizados. El siguiente ejemplo utilizará el servicio http de Nagios para el host Verónica (son parte de las configuraciones predeterminadas de Nagios). Estos valores se expresan como host = localhost & service = check\_http en los siguientes comandos.

```
| # curl -G http://localhost:3000/histou/?host=Veronica&command=check\_http
```

Si todo está bien configurado nos ha de devolver una cantidad enorme de datos. También podemos comprobar que histou está funcionando en Grafana, para ello hemos de introducir la siguiente URL en el navegador, recordar que solo funcionará si previamente hemos creado el túnel ssh en el equipo desde el que hagamos la petición.

<http://localhost:3000/dashboard/script/histou.js?orgId=1&host=Veronica&service=CPU%20Temperature%20Control>



## Integración de Histou con Nagios

Grafana/Histou también se puede integrar en la interfaz web de Nagios Core, lo cual es bastante útil, sin embargo, debemos de realizar algunos cambios en las definiciones de objetos de Nagios.

Nagios Core utiliza la directiva `action_url` en las definiciones de objetos para proporcionar un ícono / enlace cuando se visualizan objetos de host o de servicio en la interfaz web.

Esto significa que cada objeto en Nagios Core requiere que se defina la directiva `action_url`. Esto se puede lograr fácilmente usando una plantilla y usando esa plantilla en sus definiciones de objetos.

En nuestro caso hemos incluido dichas definiciones en el fichero `"/usr/local/nagios/etc/objects/templates.cfg:"`.

```
define host {
    name      host-grafana
    action_url http://localhost:3000/dashboard/script/histou.js?host=$HOSTNAME$
    register  0
}

define service {
    name      service-grafana
    action_url http://localhost:3000/dashboard/script/histou.js?host=$HOSTNAME$&service=$SERVICEDESC$
    register  0
}
```

Para poder usar las plantillas que acabamos de crear hemos de incluirlas en las directivas de servicios y host. En nuestro caso las hemos incluido en las plantillas genéricas de ambos, de manera que automáticamente se verán en todos nuestros servicios y hosts. Las definiciones de estas directivas han de quedar de la siguiente manera.

```
define host{
    name      generic-host    ; The name of this host template
    use       host-grafana
}

define service{
    name      generic-service  ; The 'name' of this service template
    use       service-grafana
}
```

En nuestro caso se pueden ver dichas directivas definidas en la página 14 de esta guía.

## Comprobaciones

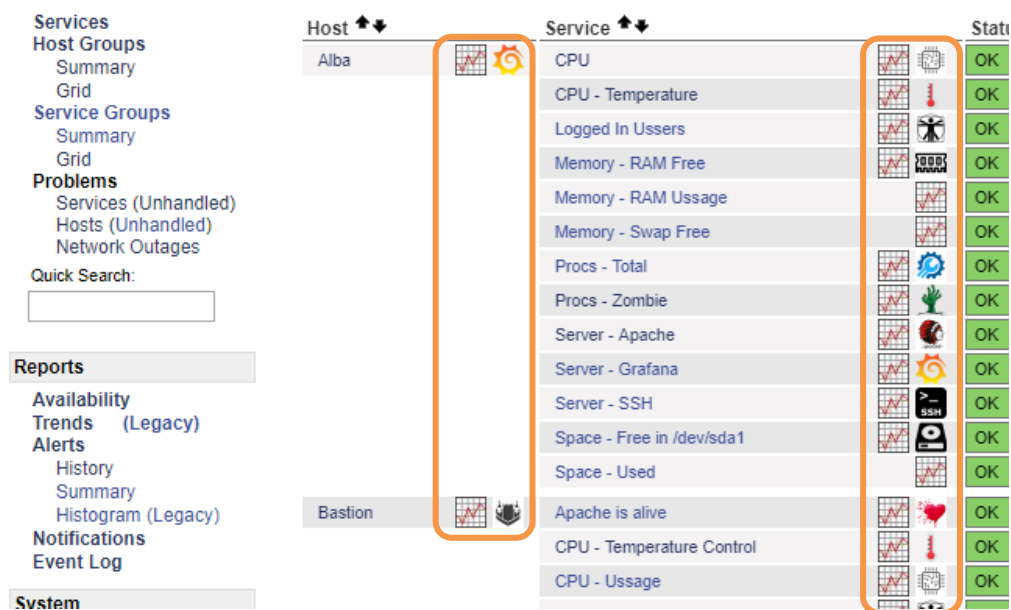
Para comprobar que las nuevas configuraciones que hemos añadido están correctas ejecutamos el comando de comprobación de nagios.

```
| # /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

Si todo ha ido bien reiniciamos el servicio para que coja efecto la nueva configuración.

```
| # systemctl restart nagios.service
```

Finalmente, si todo ha ido correctamente, deberíamos de ver en la interfaz web, en la pestaña de servicios, un icono con forma de grafica para todos los servicios y hosts.



Si hacemos click en cualquiera de ellos deberíamos de ver que se abre una pestaña nueva con un panel de grafana, en el que se nos mostrará la métrica seleccionada, usando las plantillas de histou.



## Bibliografía

- Rafael Romero González, Instalación y configuración de Nagios Core 4.0.4, Junio 2015  
<https://exchange.nagios.org/directory/Translations/Spanish/Instalaci%C3%B3n-y-configuraci%C3%B3n-de-Nagios-Core-4-2E0-2E4/details>
- Nagios Support Knowledgebase, artículo 802 <https://support.nagios.com>
- Philip Griesbacher, Adds templates to Grafana in combination with nagflux,  
<https://github.com/Griesbacher/histou>
- Assets Nagios Core, Event Handlers <https://assets.nagios.com/>



**JUNIO 2019**

DAJEJO SYSTEMS

J. Frontelo, D. Pérez, J. F. Rodríguez