Nagios. Introducción a los objetos. (I)

Para empezar a entender y configurar correctamente Nagios es básico que conozcamos los tipos de objetos que se pueden definir y tengamos un conocimiento mínimo de cómo se relacionan entre ellos.
Los objetos tienen relaciones de dependencia unos de otros lo que hace que si no conocemos bien dichas relaciones sea difícil inicialmente configurar Nagios y nos topemos constantemente con errores.
Los objetos disponibles en Nagios son los siguientes:

- **host**: Define el dispositivo a monitorizar, llevará asociado un nombre, alias, dirección ip,…
- **service**: Define el servicio (del host) a monitorizar, también con un nombre y otras propiedades asociadas a él.
- **hostgroup**: Grupo de Hosts. Sirve para agrupar objetos de tipo host bien con la finalidad de visualizarlos juntos en el interface gráfico o bien con la finalidad de facilitar la configuración.
- **servicegroup**: Grupo de Servicios. Sirve para agrupar objetos de tipo service bien con la finalidad de visualizarlos juntos en el interface gráfico o bien con la finalidad de facilitar la configuración.
- **command**: Define un comando a ejecutar. Se usa entre otros para definirle el comando concreto que ejecutará para chequear un host o servicio (plugin). También se usa para otras cosas que veremos posteriormente.
- **contact**: Se usa para definir un contacto que debería ser alertado como consecuencia de problemas detectados en los chequeos.
- **contactgroup**: Grupos de contactos. Como alternativa podremos avisar a todo un grupo.
- **timeperiod**: Periodo de tiempo. Se usa para definir periodos que luego usaremos para delimitar en que periodos se realizan chequeos, avisos de alerta,…
- **servicedependency**: Mediante la definición de dependencias de servicios podemos relacionar servicios de un Host con servicios del mismo u otro hosts para que se produzcan chequeos u avisos condicionalmente relacionados al estado en el que se encuentran servicios
- **serviceescalation**: Nagios dispone de una lógica interesante relativa al escalado de alertas. Con este objeto podemos definir como realizará este ante problemas en un servicio.
- **hostdependency**: Similar a servicedependency.
- **hostescalation**: Similar a serviceescalation pero para Hosts.

Hablamos en el artículo de archivos de configuración donde podemos localizar los objetos en los ficheros tanto en una instalación estándar de Nagios por paquetes como en una instalación con OMD Distro. Mostraremos ejemplos de dichas ubicaciones en este artículo.

Recordad que para cualquier cambio en la configuración de Nagios debemos recargar la configuración. Para OMD se explicaba en el artículo mencionado. Para Nagios en instalación estándar debemos verificar primero la sintaxis y luego recargar la configuración:

```
nagios3 -v /etc/nagios3/nagios.cfg
service nagios3 reload
```

host

Si vemos una definición tipo de Nagios (ejemplo de "serie" para localhost):

```
define host{
use             generic-host ; Name of host template to use
host_name       localhost
alias           localhost
address         127.0.0.1
}
```

Podemos ver que tiene las directivas host_name, alias, y address que no necesitan explicación. El alias normalmente se usará el mismo que hostname pero en determinadas ocasiones nos puede interesar usar distinto valor en ambas.
El objeto host tiene muchas más directivas que se pueden definir de las que tiene este ejemplo que vemos, algunas opcionales y otras muchas obligatorias. Para evitar tener que definir todas estas en cada objeto se usa la herencia de plantillas y eso es lo que hace precisamente con la propiedad "use", heredar el resto de directivas de dicha plantilla. Si buscamos la plantilla "generic host" en nuestros archivos de definición de objetos encontramos:

```
# Generic host definition template - This is NOT a real host, just a template!
define host{
name                            generic-host
generic-notifications_enabled   1
event_handler_enabled           1
flap_detection_enabled          1
failure_prediction_enabled      1
process_perf_data               1
retain_status_information        1
retain_nonstatus_information     1
check_command                   check-host-alive
max_check_attempts              10
notification_interval           0
notification_period             24x7
notification_options            d,u,r
contact_groups                  admins
register                        0
}
```

Vemos que tiene definidas muchas más directivas, probablemente al menos todas las obligatorias. No vamos a entrar de momento en verlas todas, resaltamos dos:

- **name**: nombre del host (plantilla)

- **register**: (0/1). Si te estabas preguntando porque esta definición de objeto que se define exactamente igual que un objeto host, es una plantilla (template)… aquí tienes la repuesta. Cualquier objeto host que tenga la directiva "register" a 0 será considerada como una plantilla y no como un objeto. Su valor por defecto si no la definimos en nuestro objeto host es "1". El resultado es que nuestro Host tiene con muy pocas líneas muchas directivas necesarias ya definidas.

Podemos ver todas las directivas posibles para el objeto host en la documentación de Nagios. En otros artículos veremos mucho más sobre estas.

hostgroup

Si echamos un vistazo a la definición de grupos que vienen en una instalación de Nagios vemos dos ejemplos:

```
# A simple wildcard hostgroup
define hostgroup {
  hostgroup_name    all
  alias             All Servers
  members           *
}

# A list of your Debian GNU/Linux servers
define hostgroup {
  hostgroup_name    debian-servers
  alias             Debian GNU/Linux Servers
  members           localhost
}
```

En el primero define un grupo "all" al que pertenecerán todos los host que definamos (members = *). Si. Podemos usar algunas expresiones regulares en las definiciones pero con cuidado. Dedicaremos otro artículo a dicha funcionalidad.
La segunda definición es más habitual, define un grupo "debían-servers" con un solo miembro "localhost". Debemos saber tres cosas importantes de momento:

- En la directiva "members" podemos añadir hosts separándolos por comas: host1,host2,…
- Podemos añadir un host a uno o más hostgroups también desde una directiva en el objeto host (hostgroups).
- Podemos dejar la directiva "members" vacía (pero tiene que existir) por tanto y añadir los hosts posteriormente.

service

Vemos ahora una definición tipo de servicio, de las que vienen por defecto para localhost en una instalación inicial de Nagios:

```
define service{
use                     generic-service ; Name of service template to use
host_name               localhost
service_description     Disk Space
check_command           check_all_disks!20%!10%
}
```

Cada servicio definido tiene que ter su nombre (service_descripcion) y estar asociada al menos a un host (host_name). Podremos asociarlo también a un grupo de Hosts.
Cada host definido debería tener al menos un servicio asociado a él. Realmente Nagios te deja que exista un host sin servicios pero te informará con un Warning al chequear la sintaxis de los ficheros de objetos.
Vemos que también usa herencia de plantillas mediante la directiva use. Si vemos la plantilla "generic-service" de la que está heredando vemos que tiene definidas muchas más directivas.

```
# generic service template definition
define service{
name                            generic-service ;
active_checks_enabled           1 ; Active service checks are enabled
passive_checks_enabled          1 ; Passive service checks are enabled/accepted
parallelize_check               1 ; Active service checks
obsess_over_service             1 ; We should obsess over this service (if necessary)
check_freshness                 0 ; Default is to NOT check service 'freshness'
notifications_enabled           1 ; Service notifications are enabled
event_handler_enabled           1 ; Service event handler is enabled
flap_detection_enabled          1 ; Flap detection is enabled
failure_prediction_enabled      1 ; Failure prediction is enabled
process_perf_data               1 ; Process performance data
retain_nonstatus_information    1 ; Retain non-status information across restart
notification_interval           0 ; Only send notifications on status change by
is_volatile                     0
check_period                    24x7
normal_check_interval           5
retry_check_interval            1
max_check_attempts              4
notification_period             24x7
notification_options            w,u,c,r
contact_groups                  admins
register                        0 ; TEMPLATE
}
```

Una directiva muy importante, **check_command**. Esta es la que llama a otro objeto crucial en Nagios, el objeto command. En dicho objeto que veremos más adelante se define el plugin (programa externo) que usará para realizar los chequeos para este servicio.
De momento no entraremos al detalle del resto, iremos viendo las más importantes en sucesivos

artículos. Podemos ir mirando también en la documentación oficial de Nagios relativa a las directivas de los servicios.

servicegroup.

El uso de grupos de servicios no parece tener tanta utilidad como el de grupos de hosts. Al menos yo no recuerdo haberlo usado muhco en muchos años configurando Nagios. Un ejemplo extraído de la documentación de Nagios:

```
define servicegroup{
servicegroup_name  dbservices
alias              Database Services
members            ms1,SQL Server,ms1,SQL Server Agent,ms1,SQL DTC
}
```

command

Los objetos command son los que definen las acciones que podemos hacer con Nagios ya sea proactivas (chequeos) o reactivas (envíos de correos, manejadores de eventos,…).
Vimos previamente en servicios un ejemplo de servicio que llamaba a un objeto command de la forma:

```
check_command  check_all_disks!20%!10%
```

Vemos que además de nombrar el objeto añade, separados por "!" dos valores adicionales. Dicha sintaxis nos permite pasarle los valores que deseemos separándolos de esta forma. Lógicamente el command tendrá que saber qué hacer con ellos (los descartará si no). Si vamos a ver la definición del objeto command "check_all_disks" que lo encontramos en nuestra instalación por paquetes de Nagios en: /etc/nagios-plugins/config/disk.cfg

```
# 'check_all_disks' command definition
 define command{
 command_name    check_all_disks
 command_line    /usr/lib/nagios/plugins/check_disk -w '$ARG1$' -c '$ARG2$' –e
}
```

Vemos que la definición es muy sencilla, un nombre y una línea de comandos a ejecutar. Dicha línea de comandos llama a un binario (plugin standard de Nagios) y le pasa a su vez los parámetros que ha recibido, en orden numérico, desde la directiva del servicio que le llamaba. En este caso un valor de Warning (-w) y un valor de Crítical (-c). Si queremos saber los parámetros que acepta nuestro plugin lo mejor es siempre mirar su ayuda. En este caso:
/usr/lib/nagios/plugins/check_disk –help

Los plugins de Nagios, al menos los estándar se rigen por unas reglas que definen los parámetros a aceptar y con qué letras se nombran, como tiene que ser la respuesta,… La mayoría de plugins de

terceros que nos encontremos se regirán más o menos por esas reglas pero no necesariamente. Debemos mira siempre la ayuda o documentación.

Básicamente de momento tenemos que saber que cuando llamamos a un binario desde un comando de Nagios este nos tiene que devolver al menos una cadena de texto descriptiva del resultado del chequeo y un valor de retorno del "éxito" de este (0=ok, 1=Warning, 2=Crítical, 3=Desconocido o NPI de que pasa).
Comentábamos que los objetos command sirven además de para llamar a plugins en los chequeos para otras tareas. Un ejemplo extraído de nuestra instalación estándar de Nagios es el usado para las notificaciones (enviar un email):

```
# '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/mail -s "** $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$
}
```

En este caso vemos que usa infinidad de macros de Nagios para definir ciertos valores. Es un tema que también veremos más a fondo pero puedes revisar la documentación de Nagios Core relativa a las macros para saber más. Continuamos examinando los objetos en el siguiente artículo de la serie. Es posteriores artículos continuaremos revisando el resto de objetos e iremos entrando más a fondo en temas de plugins, intervalos de Warning / Crítical,…