MK Livestatus. Acceso a datos de Nagios con “unixcat” y lenguaje LQL.

Introducción

Después de tratar en artículos con MK Livestatus y con las distintas formas de instalarlo / obtenerlo ya sea integrado en check_mk , en OMD o bien mediante una instalación individual, vamos a iniciar con este articulo una serie para aprender a acceder a los datos de Nagios con distintas variantes.
Para familiarizarse un poco más con MK Livestatus recomiendo leer al menos este artículo de instalación o directamente la documentación oficial.
Lo más importante inicialmente es familiarizarse con los datos que podemos obtener y la forma de estos para lo que usaremos una utilidad propia de MK Livestatus para el acceso a estos, **unixcat**.
Posteriormente en otro artículo veremos como acceder a los datos a través de una de las APIs que nos incluye MKL en Python.

Acceso local mediante socket.

LQL - The Livestatus Query Language

Podemos consultar a MK Livestatus por datos de Nagios con un lenguaje propio, LQL, creado a tal efecto. Dicho lenguaje tiene una sintaxis similar en cierto punto a un SQL pero adaptado para consultar tablas de objetos de Nagios, Tenemos numerosos ejemplos en la documentación oficial.

Usaremos dicho lenguaje LQL tanto para acceder a los datos mediante la utilidad unixcat como posteriormente cuando veamos como acceder mediante la API de Python.

Acceso a datos mediante la utilidad “unixcat”

Como toma de contacto para conocer la presentación de los datos vamos a usar la utilidad “unixcat” proporcionada por la instalación de livestatus. Mediante dicha utilidad podemos, de forma muy sencilla, realizar consultas a CMK Livestatus adjuntándoles el fichero “lql” con la sintaxis de lo que buscamos. Unixcat está instalado normalmente con los binarios de check_mk pero es posible que no esté en los paths habituales.º

Un ejemplo sencillo. Creamos primero un fichero LQL “test.lql”.

```
GET services
Columns: host_name description state
Filter: state = 2
Filter: in_notification_period = 1
```

Ejecutamos unixcat pasándole el contenido del archivo al socket de livestatus y obtenemos las columnas y los registros que coinciden con el filtro.

```
$ unixcat < test.lql /opt/omd/sites/jrm2/tmp/run/live
localhost;Interface 2;2
```

```
srv-debian;Interface 2;2
srv-ubuntu;Interface 2;2
```

Si es una consulta sencilla podemos pasarla directamente en línea de comandos. P.e. obtener todos los hosts:

```
$ echo 'GET hosts' | unixcat /usr/local/nagios/var/rw/live
```

Como vemos la sintaxis es muy sencilla pero... ¿como sabemos los campos disponibles y valores de campos estándar para realizar filtrados?

Tablas de datos.

Para saber los campos que contiene una tabla (que no aparecen en la documentación) debemos preguntar a una tabla especial "columns". Así que la consulta lql para conocer los campos de todas las tablas sería obtener todas las columnas

```
$ echo 'GET columns' | unixcat /opt/omd/sites/jrm2/tmp/run/live
The name of the column within the table;name;columns;string
The name of the table;table;columns;string
….
```

Donde aparecerían todas los campos de todas las tablas siendo la penúltima columna la tabla a la que pertenece cada uno. Para poder ser más concretos podemos obtener la información de una tabla con la "query" siguiente en un fichero p.e. "hosts.lql"

```
GET columns
Filter: table = hosts
```

La ejecutamos y obtenemos todos los campos de la tabla hosts:

descripción (con posibles valores cuando proceda); nombre_campo; tabla; tipo_valor

```
$ unixcat < hosts.lql /opt/omd/sites/jrm2/tmp/run/live
description;name;table;type
Whether passive host checks are accepted (0/1);accept_passive_checks;hosts;int
Whether the current host problem has been acknowledged (0/1);acknowledged;hosts;int
Type of acknowledgement (0: none, 1: normal, 2: stick);acknowledgement_type;hosts;int
```

La salida por defecto es en formato CSV. Podríamos cambiar los caracteres e incluso obtener la salida en formato JSON o salida de Python. Solo habría que añadir una línea en la sentencia lql:

```
OutputFormat: json
OutputFormat: python
```

Un buen punto de partida será listar y guardar en un formato adecuado la información de cada tabla, bien creado los ficheros lql o bien en línea de comandos especificando la tabla. P.e.

```
GET columns
Columns: description name type
Filter: table = hosts
```

```
unixcat < hosts.lql /opt/omd/sites/jrm2/tmp/run/live
```

O bien mandando a fichero directamente la orden desde línea de comandos,

```
printf 'GET columns \nFilter: table = hosts \n' | unixcat /opt/omd/sites/jrm2/tmp/run/live >>
hosts.csv
```

En definitiva, tener estas tablas a mano ya que es posible las usemos muy a menudo.

Unixcat es muy útil para familiarizarse con las posibilidades y para con scripting con bash o cualquier otro lenguaje incluso desde otros equipos con ssh. Para programas / sripts más elaborados será mejor usar la API pensada para acceder a estos datos que veremos en un artículo posterior en su variante Python.

Podremos acceder al fichero socket directamente o, con muchas más posibilidades, a través de red habilitando el acceso a dicho fichero con xinetd.

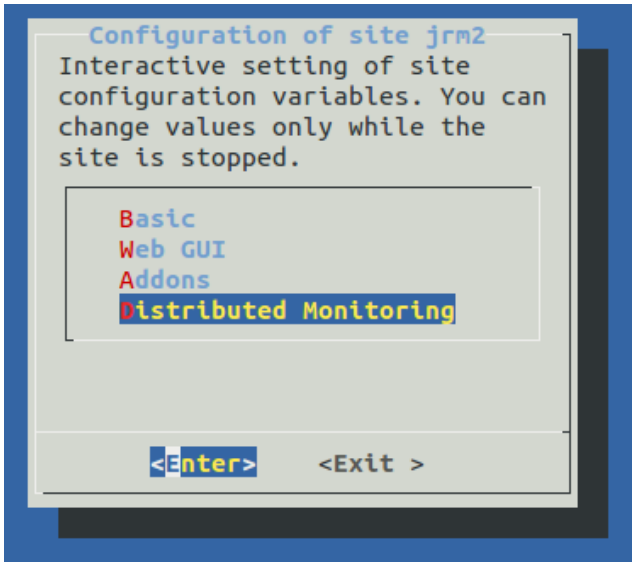
Configuración de acceso remoto al socket livestatus.

Una vez que tenemos check_mk funcionando la configuración para acceder a través de un socket TCP a Livestatus es sencilla y está bien documentada en la página de doc. de livestatus. Vermos ambas posibilidades: configurarlo en una instalación de OMD o bien en una instalación sencilla ya sea de MK Livestatus solamente o integrado con Check_mk.

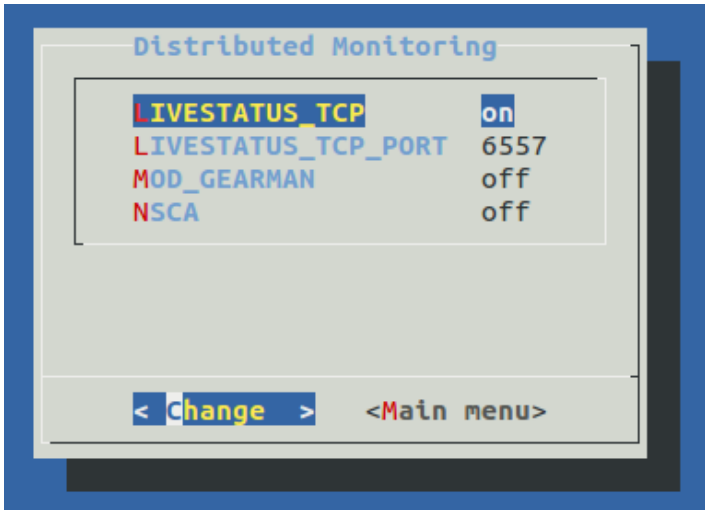
Configuración en una instalación de OMD (para un site concreto)

Para poder acceder a Livestatus por red en una instalación de OMD ejecutaremos la configuración para la "instancia / usuario" deseado. P.e. dada una instancia previamente creada "jrm2":

```
omd config jrm2
```



Seleccionaremos "Distributed Monitoring" y posteriormente activaremos "LIVESTUS_TCP"



Si la instancia está en ejecución es posible que sea necesario reiniciarla y nos pregunte a tal efecto. Básicamente lo que hace es añadir un fichero de configuración a xinetd para poder acceder a través de este al fichero socket usando la utilidad unixcat.

Una vez configurado creará el fichero de configuración en el etc/xinetd del usuario del site de OMD con las variables de configuración. Inicialmente nos interesa sobre todo "only from" si queremos acceder desde otro equipo al socket.

```
{
        type                = UNLISTED
        socket_type         = stream
        protocol = tcp
        wait                = no
        cps             = 100 3
        instances       = 500
        per_source      = 250
        flags           = NODELAY
        only_from       = 127.0.0.1 10.0.20.1 10.0.20.2
        disable             = no
        port                = 6557
        user                = jrm2
        server              = /omd/sites/jrm2/bin/unixcat
        server_args     = /omd/sites/jrm2/tmp/run/live
}
```

Será necesario recargar el site en el caso de OMD: omd reload jrm2

Instalación independiente de MK Livestatus (o integrada en check_mk)

En el caso de una instalación independiente de check_mk entero o solo de Livestatus deberemos realizar nosotros esta parte. Nos aseguraremos que esté instalado xinetd y crearemos el fichero para poder acceder desde otro sistemas a este. Tenemos un ejemplo para un fichero /etc/xinetd.d/livestatus en la documentación de cmk livestatus … Básicamente sería similar claro al de OMD.

```
service livestatus
{
        type                = UNLISTED
        port                = 6557
        socket_type         = stream
        protocol   = tcp
        wait                = no
# limit to 100 connections per second. Disable 3 secs if above.
        cps             = 100 3
# set the number of maximum allowed parallel instances of unixcat.
# Please make sure that this values is at least as high as
# the number of threads defined with num_client_threads in
# etc/mk-livestatus/nagios.cfg
      instances       = 500
# limit the maximum number of simultaneous connections from
# one source IP address
```

```
        per_source      = 250
# Disable TCP delay, makes connection more responsive
                flags           = NODELAY
                user                    = nagios
                server                  = /usr/bin/unixcat
                server_args     = /var/lib/nagios/rw/live
# configure the IP address(es) of your Nagios server here:
                only_from       = 127.0.0.1 10.0.20.1 10.0.20.2
                disable                 = no
}
```

Tendremos que tener cuidado de habilitar tanto las direcciones IP de los equipos permitidos (parámetro only_from) como, si procede, habilitar el acceso en el Firewall local al puerto 6557 desde el exterior y filtrarlo como se desee.

Probando acceso remoto a livestatus.

Podemos usar por ejemplo la utilidad Netcat para acceder de forma remota a livestatus de otro equipo y obtener datos:.

```
echo 'GET hosts' | nc host_remoto 6557
echo 'GET status' | nc host_remoto 6557
```

En próximos artículos echaremos un ojo a la API para Python que nos proporciona ML Livestatus para poder hacer ya desarrollos “serios”.