Check_mk. Chequeos. Primeros pasos (II).

Siguiendo con la introducción a la configuración de objetos de Nagios con check_mk vamos a avanzar un poco más en las posibilidades que nos ofrece continuando el articulo anterior, (http://eldespistado.com/check_mk-chequeos-primeros-pasos/).

Definición de equipos a monitorizar.

Vimos que la sintaxis más sencilla para definir Hosts era la siguiente:

```
all_hosts = [ 'localhost', 'srv_windows' ]
```

Recordad también que era necesario que el host tuviera el agente check_mk instalado previamente (o que tenga snmp, ya veremos este tema posteriormente) y que responda adecuadamente por nombre.

En el array denominado all_hosts pondremos todos los equipos a monitorizar con la sintaxis especificada: dentro del array, contenidos entre comillas simples y separados por comas.

Tags.

En la definición de los hosts también podemos incluir "tags" que nos servirán posteriormente para referenciar todos los objetos que contengan ese tag. Veámoslo como conjuntos pensados para referirse a todos los hosts que pertenezcan a dicho conjunto. Esto nos facilitará mucho la vida posteriormente en la configuración de otros parámetros. Podemos definir tantos tags como queramos a cada hosts pero solo podemos usar letras, números y subrayado bajo. Por ejemplo (marcados los tags en rojo):

```
all_hosts = [ 'localhost|linux|cpd1', 'srv_windows|windows|cpd1']
```

Vemos que la sintaxis es separar los tags con un "pipe" del host y entre ellos. Hemos agregado los tags "linux" y "windows" para poder distinguir el Sistema Operativo de los hosts. También hemos añadido un tag "cpd1" para reflejar que ambos equipos están en la misma ubicación. Posteriormente podremos modificar chequeos y propiedades de estos en función del Sistema Operativo y de la ubicación, en lugar de tener que referenciar a todos los hosts en una lista.

Para mejorar la legibilidad podemos también separar los elementos del array.

```
all_hosts = [ 'localhost|linux|cpd1', 'srv_windows|windows|cpd1' ]
```

Solo por el hecho de definir los hosts a chequear, check_mk ya creará en estos una serie de **servicios por defecto** a chequear como vimos previamente. Las directivas que veremos a continuación son para definir más chequeos que no realiza por defecto, para alterar estos, decirle que no los realice o para añadir funcionalidades y parámetros de configuración adicionales.

Chequeo de servicios en servidores windows.

Empezamos con las directivas que nos serán de utilidad a la hora de definir chequeos adicionales. Esta nos permitirá chequear el estado de servicios en servidores Windows que tenga el agente check_mk instalado claro.

En este caso la directiva implicada es "inventory_services" y define los servicios que queremos chequear. Ejemplo:

```
inventory_services = ['Sophos_Agent','AVP',]
```

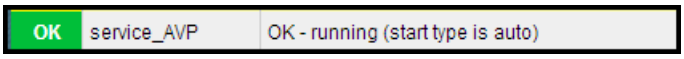
Este es un ejemplo sencillo, define unos servicios a inventariar (distintos servicios de antivirus) siempre que aparezcan en cualquier Host (Windows). Si detecta alguno de estos servicios en cualquier Host inventariado lo añadirá como servicio a chequear. OJO, debemos usar el nombre corto del servicio Windows (Nombre de servicio) no la descripción.

Actualizar el inventario.

Después de realizar una modificación siempre tendremos que actualizar el inventario y recargar la configuración. Con la opción –I inventariábamos un equipo, con –II lo "reinventaríamos" (para añadir o para eliminar chequeos que ya no existen). En este caso le decimos que vuelva a inventariarlos.

```
check_mk -II
check_mk -O
```

Si vamos al interface de visualización de check_mk y vemos los servicios de nuestro Host habrá detectado y añadido el de AVP (antivirus Kaspersky).



La forma expuesta aquí es la más sencilla. Nos permite realizar muchas más cosas como tener en cuenta si el servicio está en auto o no, usar expresiones regulares para definir los servicios, usar tags para incluir / excluir equipos,… En la documentación oficial explica concisamente todos los ejemplos. Puedes hacer prácticamente lo que se te ocurra para chequear servicio Windows.

Pertenencia a grupos.

Para definir la pertenecía a grupos de determinados Host usamos la directiva de configuración "host_groups". Ejemplo:

```
host_groups = [( "Servidores_Windows", ["windows"], all_hosts),]
```

Explicación:

```
"Servidores_Windows"    Grupo de hosts de Nagios (debe existir en la configuración de Nagios !!!)
["win"]    Incluir en este grupo todos los equipos con el tag "win"…
all_hosts  Del grupo (array) de all_host
```

Llegados a este punto por tanto necesitaremos tener creados los grupos en formato Nagios. Si estás

iniciándote con estos artículos no tendrás ni idea así que rápidamente sería:

Crear un nuevo fichero:

/opt/omd/sites/foo/etc/nagios/conf.d/grupos.cfg

Con el contenido:

```
define hostgroup{
hostgroup_name    Servidores_Windows
alias             Servidores Windows
}
```

Y ejecutar la recarga de los ficheros de config de Nagios con los comandos OMD (como el usuario de OMD recuerda).

```
OMD[foo]:~$ omd restart
```

Para aplicar los cambios, en este caso valdría con recargar la configuración de check_mk ya que no hemos modificado servicios a chequear si no definición de objetos de Nagios:

```
check_mk -R.
```

Aparecerá nuestro grupo en la vista “Hostgroups” con nuestro host.

Como siempre, podemos ampliar información en la documentación de check_mk relativa a variables de configuración (sección “host_groups”).

Chequeos de procesos (Windows / Linux).

Podemos decirle también que nos chequee procesos activos, tanto en servidores Windows como en Linux/NIX. Hay que decirle que procesos tiene que incluir y en qué casos. Para ello se usa la directiva checks. Ejemplo:

CHEQUEOS CONCRETOS DE PROCESOS LEVANTADOS

```
checks = [
# PARA Servidores Windows
( ["windows"], all_hosts, "ps", "Antivirus Kas", ( "avp.exe", 1, 1, 1, 1 ) ),
( ["windows"], all_hosts, "ps", "Explorer", ( "explorer.exe", 1, 1, 1, 1 ) ),
```

```
]
```

Cogiendo una de las líneas vemos la configuración:

- (**["windows"]**, Indica que tags deben tener los Host para incluir el chequeo de este proceso.
- **all_hosts** Indica sobre que grupo de Host vamos a buscar, en principio será siempre all_host.
- **“ps”** Indicador fijo que define que chequeamos un proceso en ejecución.
- **“Antivirus Kas”** El nombre con el que aparecerá el servicio chequeado en Nagios.
- **(“avp.exe”, 1, 1, 1, 1))** Tupla que define el proceso a chequear, el resto de valores numéricos indican cuantas veces puede aparece min/max y para que valores será Warning/Critical. En este caso son procesos Windows que solo aparecen un vez y que hay que alertar en el momento que no aparezcan.

CRIT	proc_Antivirus Kas	CRIT - 2 processes (ok from 1 to 1)
OK	proc_Explorer	OK - 1 processes

Vemos que ha encontrado dos procesos ejecutándose por lo que debemos ajustar nuestra definición.

Ignorar chequeos.

Como comenté, por defecto check_mk tiene definido una serie de chequeos que incorporará siempre que los detecté en un Host. Mediante estas entradas podemos hacer que ignore ciertos chequeos para todos los equipos, algunos, por tags,…

Es muy importante inicialmente que los LOGs de Windows ya que por defecto añadirá todos los logs del equipo a chequear (Sistema, aplicación, AD) y nos ocasionará muchas alarmas. Posteriormente podemos comentar la línea para que lo haga pero sabiendo lo que nos vamos a encontrar.

```
ignored_services = [
# IGNORA LOGS DE WINDOWS (chequeos “LOG” de todos los equipos con el tag ‘windows’
( [ "windows" ], all_hosts, [ "LOG" ] ),
# OTROS
# Ignora chequeo servicio DHCP Stats de todos los equipos con el tag ‘windows’
( [ "windows" ], all_hosts, [ "DHCP Stats" ] ),
# Ignora chequeo fs home de todos los equipos con el tag ‘linux’
( [ "linux" ], all_hosts, [ "fs_/home$" ] )
]
```

Como vemos la sintaxis es muy sencilla. Podemos ser mucho más específicos e ignorar solo logs concretos, por expresiones regulares,…

Como siempre después de hacer cambios:

```
check_mk -O
```

OJO: Es muy “pijo” con los caracteres usados en el fichero de configuración. Es habitual que nos pueda dar errores al ejecutar el “-”R” relativos a esto dándonos la línea. Normalmente será un acento (eliminarlos), unas comillas o unas comillas dobles (borrarlas y ponerlas de nuevo, el copy/

paste cambia caracteres a veces). Tampoco nos los dejará usar ni en los comentarios..
Encontraremos mucha más información en la documentación en línea.

Otros artículos de interés relacionados

- Monitorizar ficheros log con check_mk en Linux – Oracle alertlog y otros.
- Instalación de packages de check_mk. Oracle RMAN package.
- Monitorizar Oracle con Nagios y check_mk.
- Iconos en Nagios (y check_mk).