



Alumno: Francisco José Basurte Garrido

Enginyeria Tècnica en Informàtica de Sistemes

Consultor: Miquel Angel Senar Rosell

(Thanks to Alexei Vladishev and all the developers of Zabbix)

Enero 2011



NetEye - Monitoring by Francisco Basurte Garrido is licensed under a [Creative Commons Reconocimiento 3.0 Unported License](https://creativecommons.org/licenses/by/3.0/).

ÍNDICE DE CONTENIDOS

ÍNDICE DE IMÁGENES	4
RESUMEN	6
CAPÍTULO 1 INTRODUCCIÓN AL TFC	7
1.1 Justificación del TFC, contexto de despliegue.	7
1.2 Plataforma existente	8
1.3 Plataforma seleccionada	9
1.4 Software de monitorización	10
1.4.1 Propietario	10
1.4.2 GNU / GPL	12
1.5 Objetivos	15
1.6 Beneficios	15
1.7 Planificación del TFC	16
1.7.1 Desviaciones	18
1.7.2 Correcciones	18
CAPÍTULO 2 IMPLEMENTACIÓN DE LA PLATAFORMA	21
2.1 Instalar plataforma base	21
2.2 Configurar plataforma	23
2.3 Parametrización de Zabbix	25
2.4 Instalación Clientes:	26
2.4.1 Instalación básica agente Windows.	26
2.4.2 Monitorización simple de una Web. Página de inicio.	26
2.4.3 Monitorización de espacio en unidad local	28
2.4.4 Monitorización SNMP	28
CAPÍTULO 3 ANÁLISIS DEL APLICATIVO - JERARQUÍA DE OBJETOS	29
3.1 Introducción	29
3.2 Modelo	30
3.2.1 <i>Hosts</i>	30
3.2.2 <i>Items</i>	30
3.2.2.1 Disponibilidad del Host	31
3.2.2.2 File Systems	32
3.2.2.3 Estado de servicio	33
3.2.2.4 Consultas SQL	35
3.2.2.5 Comprobación de red	35
3.2.2.6 Recursos del sistema	37
3.2.2.7 Zabbix Trapper	38
3.2.2.8 Otros	38
3.2.3 Host Groups	39
3.2.4 Templates	40
3.2.5 <i>Triggers</i>	41
3.2.6 <i>Actions</i>	42
3.2.7 Discovery	42
3.2.8 Maps	43

3.2.8.1	Geográficos	43
3.2.8.2	Servicios	44
3.2.9	Graphs	45
3.2.10	Notificaciones	46
CAPÍTULO 4 ESTUDIO FASE2		47
4.1	Alcance	47
4.1.1	Checklist diario	47
4.1.2	Formación HelpDesk	47
4.1.3	Backup del aplicativo	48
4.1.4	Monitorización completa de servidores	49
4.1.5	Monitorización completa de la red	49
4.1.6	Avisos por SMS	50
4.1.7	Despliegue Worldwide, servidores Zabbix Proxy	50
4.1.8	Solución de upgrade de aplicativo, servidor y cliente	51
4.2	Recursos e inversión	51
CAPÍTULO 5 PRUEBAS REALIZADAS		53
CAPÍTULO 6 CONCLUSIONES		56
6.1	Mejoras	56
6.2	Conclusiones	56
GLOSARIO		60
BIBLIOGRAFÍA		61

Índice de Imágenes

<i>Figure 1 - Checklist actual</i>	8
<i>Figure 2 - Histórico - Listado de informes antiguos</i>	9
<i>Figure 3 - Nagios vs. Software licenciado</i>	12
<i>Figure 4 - Zabbix vs. Software licenciado</i>	12
<i>Figure 5 - GPL Licenses</i>	14
<i>Figure 6 - Comparativa software libre</i>	14
<i>Figure 7 - Media type para el envío de correo</i>	24
<i>Figure 8 - Template Status - Comprobaciones básicas comunes</i>	25
<i>Figure 9 - Creación de un host</i>	26
<i>Figure 10 - Vista de host (nombre e ip's borrados)</i>	27
<i>Figure 11 - Creación de escenario</i>	27
<i>Figure 12 - Disponibilidad y velocidad de respuesta de la Web http://loquesea.com</i>	27
<i>Figure 13 - Visualización base del estado de un Server + disco E:\</i>	28
<i>Figure 14 - Value mappings- Permiten traducir la información recogida</i>	31
<i>Figure 15 - Versión de agente Zabbix instalada. Útil de cara a upgradear la versión</i>	31
<i>Figure 16 - Comprobación de ocupación de File Systems</i>	32
<i>Figure 17 - Trigger comprobación % de espacio libre en C:\</i>	33
<i>Figure 18 - Ejemplo de Item deshabilitado</i>	33
<i>Figure 19 - Ejemplo comprobación 'Listening' en puerto 25 TCP (usualmente SMTP). Alternativa</i>	33
<i>Figure 20 - Datos básicos recopilados de Servicio MS SQL Server</i>	34
<i>Figure 21 - Línea OK (línea verde)</i>	36

<i>Figure 22 – Línea con Error (red dashed line)</i>	36
<i>Figure 23 – Creación del nuevo item tipo ‘Zabbix Agent’ personalizado "DominoSMTP"</i>	37
<i>Figure 24 – Definición de item personalizado en el cliente</i>	37
<i>Figure 25 – Ejemplo de item tipo Zabbix Trapper</i>	38
<i>Figure 26 – Monitoring – Overview. Selección de visualización de SMTP Group</i>	40
<i>Figure 27 – Ejemplo de asignación de permisos a grupos de usuarios</i>	40
<i>Figure 28 - Items básicos definidos para Windows Servers</i>	41
<i>Figure 29 – Ejemplo trigger “espacio por debajo del 10%”</i>	41
<i>Figure 30 - Action de tipo Discovery (ejemplo)</i>	43
<i>Figure 31 - Zoom al mapa de Europa</i>	44
<i>Figure 32 - Zoom al mapa de Iberia, tras clickar en Iberia</i>	44
<i>Figure 33 - Zoom Centro de Coornella, tras clickar en Cornellà HQ</i>	44
<i>Figure 34 – Ejemplo Servicio OK</i>	45
<i>Figure 35 - Ejemplo Servicio erróneo</i>	45
<i>Figure 36 - Mapa de servicios. Cada icono es de tipo mapa.</i>	45
<i>Figure 37 - Ejemplo de gráfico consumo de disco</i>	45
<i>Figure 38 - Proxy server schema con Zabbix- Image from Zabbix on-line doc v1.8</i>	50

Resumen

El trabajo pretende implementar la base (Fase 1, ‘need to’) de un sistema de monitorización de la infraestructura de servidores/red de una organización real. La idea es que sirva como plataforma de trabajo para un proyecto más amplio, de carácter global, de monitorización de todos los sistemas de la compañía (Fase 2, ‘want to’). Se utilizará para ello software libre (para el servidor de monitorización y los clientes). Dado el gran número de servidores y dispositivos objeto del proyecto, sería poco realista pretender, con un único administrador a tiempo parcial, completar la tarea de despliegue en todos los dispositivos. Ello ha motivado la división del proyecto en dos fases. En la primera, se construirá una plataforma estable de Suse / Zabbix en un entorno productivo real, que permitirá monitorizar diferentes sistemas y servicios, diferentes grados de alertas y generar en la medida de lo posible, diferentes templates de monitorización para los equipos de la organización. Esto lo motiva el hecho que no es la misma información la que necesita un director de Ventas, o un técnico de IT, por poner un ejemplo. En el primer caso, puede ser suficiente el hecho de saber de un vistazo si los procesos *batch* de entrada de pedidos se han realizado de forma correcta, en el segundo, puede ser interesante conocer en tiempo real, los errores que se producen en el monitor de eventos de un servidor crítico, o el estado de una línea de comunicación. La foto final, tras la fase 2 finalizada, pretende la instalación de monitores en diferentes áreas del HQ que muestren el estado de los sistemas/procesos.

Una vez la plataforma base quede completada (Fase 1), se hará un estudio, lo más amplio posible de las tareas pendientes y la inversión a realizar en man/days y equipamiento para su conclusión. Se recopilará información de los posibles clientes a implementar, coste del despliegue para extender la plataforma base a todos los sistemas productivos, etc.

La idea es poder presentar la Fase2 a la dirección de la organización como proyecto, de cara a obtener el presupuesto necesario para llevarlo a cabo.

Capítulo 1 Introducción al TFC

El trabajo se ha de realizar durante un trimestre lectivo. Está valorado en 7.5 créditos, lo cual requiere un mínimo de dedicación de 6 horas semanales. Durante el mismo se implementará una plataforma Linux/Suse con el software Zabbix con el fin de monitorizar la red corporativa. El alcance del proyecto, dado su envergadura, se dividirá en dos fases. El TFC abarcará la primera fase en la que se realizará un estudio del contexto actual de monitorización con el fin de justificar la implementación. Se instalará la plataforma descrita anteriormente y se configurarán clientes modelo de:

- Monitorización de Sistema operativo Linux
- Monitorización de Sistema operativo Windows de Microsoft
- Monitorización de Microsoft SQL Servers
- Alertas SNMP (Switches Nortel, Sonicwall's FW)
- Alertas pasivas desde procesos *batch* en Servidores Wintel
- Estado de la red (Mapa LAN y WAN que incluyan servidores y equipos principales de comunicaciones)

Esta base ha de permitir presentar el proyecto a la dirección con el fin de que se apruebe la fase siguiente desplegando la plataforma de forma *Worldwide*. Por ello se aportará un estudio de lo más detallado posible de los recursos necesarios y su coste.

1.1 Justificación del TFC, contexto de despliegue.

En la empresa a la que estoy dando servicio actualmente como consultor técnico, se realiza un chequeo diario muy básico de los sistema. Esto nos permite ser proactivos a muy pequeña escala, ya que en la mayoría de ocasiones el fallo de los sistemas se detecta por un aviso del usuario final, cuando su solución es urgente, en el mejor de los casos. Los motivos que me llevan a proponer la implementación de esta herramienta son:

- El check list actual no es en tiempo real, se realiza una vez al día, por las mañanas, de lunes a viernes.

- Se comprueban únicamente ciertas webs corporativas, el resultado del backup diario y la disponibilidad de los entornos productivos del ERP SAP. Poco alcance pues para un entorno corporativo con más de 600 usuarios directos (200 de ellos en el Headquarters).
- La comprobación se realiza manualmente, un técnico ha de invertir 15 minutos diariamente para su realización.
- No se dispone, ni hay previsión de integrar en IT un grupo de operaciones que realice un seguimiento online de los sistemas y procesos batch.
- Falta de presupuesto para implementación de solución propietaria (HP Openview, Tivoli, BMC, Unicenter, etc.)

Por todo ello se pretende utilizar un servidor GNU/Linux con el aplicativo Zabbix que de forma centralizada reciba el estado de los sistemas ya sea por SNMP, alertas pasivas o por información obtenida directamente de clientes instalados en cada uno de los dispositivos. Más adelante veremos en qué consiste cada uno de estos conceptos.

1.2 Plataforma existente

Actualmente se dispone de una BDD de Lotus Notes (IBM) desarrollada por el equipo de la empresa donde

se refleja el estado de varios sistemas. De lunes a viernes y a primera hora de la mañana se revisan los siguientes sistemas y el resultado se traslada a dicha hoja (Véase

Figure 1 - Checklist actual):

- CHECKLIST -			
Autor: User 1		Fecha: 29/10/2008 08:29	
SERVICIO Y COMPROBACION	RESULTADO	OBSERVACIONES	RESPONSABLE
1 AS-LotusNotes-Correo Internet			
Helpdesk - Correo Salida a Internet	Ok		User x
Helpdesk - Servidor Aragón	Ok		User y
Helpdesk - Servidor Pla	Ok		User y
Helpdesk - Servidor Irlanda	Ok		User y
Helpdesk - Servidor Italia	Ok		User y
Helpdesk - Servidor USA	Ok		User y
Helpdesk - Servidor Luis	Ok		User y
2 Navegación Internet			
http://...	Ok		User x
http://...	Ok		User x
http://...	Ok		User x
http://...	Ok		User x
http://...	Ok	Ver si la fecha es del día	User x
http://...	Ok		User x
http://...	Ok		User x

Figure 1 - Checklist actual

Al grabar el informe, si alguno de los registros tiene el status de *error*, se crea una incidencia de forma automática en la herramienta de ticketing de HelpDesk, desarrollada también bajo Lotus Notes.

Estado	Autor	Fecha Creado
Abierto		29/10/2008 08:11
Abierto		11/07/2008 08:11
Abierto		03/12/2007 09:11
Abierto		17/07/2008 07:11
Abierto		14/07/2008 08:11
Abierto		17/08/2006 10:11
Abierto		18/01/2006 09:11
Abierto		15/07/2008 07:11
Abierto		22/10/2007 09:11
Abierto		05/12/2005 09:11
Cerrado		14/03/2006 10:11
Cerrado		11/03/2005 09:11
Cerrado		03/01/2006 09:11
Cerrado		11/11/2008 08:11
Cerrado		29/03/2007 10:11
Cerrado		20/05/2005 08:11
Cerrado		03/04/2009 09:11
Cerrado		31/05/2006 08:11
Cerrado		20/09/2006 08:11
Cerrado		22/03/2008 09:11
Cerrado		15/04/2008 12:11
Cerrado		07/07/2005 08:11
Cerrado		28/06/2007 08:11
Cerrado		17/10/2007 09:11
Cerrado		09/12/2008 08:11
Cerrado		02/04/2007 08:11
Cerrado		16/12/2004 09:11
Cerrado		26/07/2005 08:11
Cerrado		17/08/2007 10:11
Cerrado		21/12/2004 09:11
Cerrado		14/02/2007 09:11
Cerrado		03/03/2008 12:11
Cerrado		29/06/2009 12:11
Cerrado		23/03/2007 10:11
Cerrado		14/07/2005 08:11
Cerrado		27/02/2007 08:11
Cerrado		11/05/2007 08:11
Cerrado		19/12/2008 08:11
Cerrado		15/11/2007 09:11
Cerrado		17/06/2005 08:11
Cerrado		10/05/2005 08:11
Cerrado		23/06/2009 08:11
Cerrado		30/10/2008 08:11
Cerrado		12/01/2005 09:11
Cerrado		06/04/2009 10:11
Cerrado		05/09/2006 09:11
Cerrado		24/02/2005 09:11

Los informes antiguos se pueden consultar accediendo a través de tres vistas, *todos*, por *fecha* y por *estado*. Se archivan de forma automática, y periódicamente se purgan los más antiguos.

Figure 2 - Histórico - Listado de informes antiguos

Desde mi punto de vista esta solución, pese a haber dado servicio de forma eficiente durante años, debería actualizarse. Algunos de los motivos que en mi opinión refuerzan esta idea son los siguientes:

- El chequeo se realiza una única vez al día. Esto permite ser consciente de un problema que ocurra durante la noche, a primera hora de la mañana. El resto del día sin embargo son los usuarios, habitualmente, quienes dan la voz de alarma del fallo de los sistemas.
- El número de monitorizaciones que se realizan es muy básico y además no están basadas en la criticidad del entorno. En algunos casos no se realiza por su complejidad, y en la mayoría, por el tiempo que ha de invertir el técnico en revisar un gran número de servidores y dispositivos.
- El número de dispositivos y servicios que se precisan monitorear ha crecido y no es viable realizar las comprobaciones manualmente.
- Se dispone de un histórico, pero no de estadísticas o gráficos de rendimiento.

1.3 Plataforma seleccionada

Como Hardware se ha elegido la plataforma Vmware ESX que nos permitirá ajustar las necesidades de disco, CPU y memoria a los requerimientos del aplicativo. El *datastore* es una cabina Hitachi 2100 con RAID5+HotSpare.

El uso de una VM, nos permitirá una gestión remota del equipo, básicamente con el fin de poder realizar reinicios desde fuera de la oficina.

En cuanto al Software se ha seleccionado Zabbix. Se procederá a descargar e integrar una *virtual appliance* de Zabbix en el entorno ESX/Hitachi comentado anteriormente.

Principalmente se ha seleccionado Zabbix por dos motivos. Por un lado dispone de clientes nativos para multitud de plataformas. Dichos clientes incluyen monitorizaciones con un espectro muy amplio de servicios y resulta muy fácil la inclusión de scripts propios para la recogida de información específica en cada una de las plataformas.

Por otro lado el software creado por la empresa de **Alexei Vladishev** dispone de una interfaz gráfica realmente buena, desarrollada en **php**. Desde este punto común resulta relativamente sencillo el manejo de la aplicación, tanto para monitorizar como para configurar el entorno. No se trata únicamente de la facilidad de cara a la implementación inicial (que no es desdeñable), sino que por el contrario, se espera poder formar fácilmente a primer nivel para su gestión y monitorización de forma autónoma, y para esto sí que considero que esta interfaz es ideal.

1.4 Software de monitorización

Este tipo de programas aportan la posibilidad de monitorizar de forma centralizada y desde una única interfaz, un gran número de dispositivos y servicios, en plataformas heterogéneas.

Existen en el mercado numerosas soluciones tanto libre como propietarias con esta finalidad. En estas líneas no se pretende dar cabida a todas ellas, pero sí enumerar las más significativas actualmente, y un breve comentario de cada una de ellas.

1.4.1 Propietario

- HP Openview

Este producto de HP se vende como una solución global de software con diferentes módulos, licenciados individualmente. Abarca desde la monitorización de la red hasta herramientas de ticketing para HelpDesk, pasando por la monitorización de VoIP o la integración de datos.

Se trata de uno de los software líderes en monitorización. Los tres módulos principales son:

- i. Network Node Manager: Es el *core* de la aplicación. Se encarga de escanear la red, descubriendo su topología y gestiona las alarmas que se puedan producir. También sirve de base para el resto de herramientas de gestión
- ii. Operations Center: Se trata de la interfaz de usuario
- iii. Admin Center: Se utiliza para gestionar los servidores, WS, etc.

- Unicenter TNG

Se trata de un sistema de monitorización con fuerte presencia en el mercado, al menos en la década de los 90's. Se trata de una plataforma integrada para la gestión del SW y HW de la organización. Se escaparía un poco del objetivo del proyecto que '*simplemente*' abarca la monitorización. Este SW aparte de la monitorización, es capaz de realizar predicciones en función de las especificaciones del fabricante y de administrar los dispositivos HW, que no tienen por qué ser de IT, puede manejarse con todo tipo de appliances.

- Tivoli de IBM

IBM ofrece con este producto una solución con diferentes productos integrados bajo el nombre de IBM Tivoli Monitoring Solutions. Engloba tanto clientes como aplicativos para monitorizar equipos, comunicaciones y servicios

Esta solución engloba numerosos módulos que no veremos por problemas de espacio y porque, francamente, el objetivo es dar unas pinceladas de los software más significativos de cara a ubicarnos en el entorno adecuado, en ningún momento un estudio de ellos,

En la *Figure 3 - Nagios vs. Software licenciado* se puede apreciar una comparativa de los hits de Nagios y Zabbix vs. software licenciado que se producen en Internet (según Google Trends). En ellas podemos observar en la parte superior una gráfica de las búsquedas en Internet relacionadas con cada uno de los términos, y justo debajo, información en Internet sobre el término en cuestión. En ella destacaría claramente que la presencia de Nagios es significativa en la red, por encima incluso de software licenciado como Unicenter y Openview, en el caso de Zabbix (*Véase Figure 4 - Zabbix vs. Software licenciado*), la presencia es superior a la de Openview de igual forma. Son datos que no hacen referencia a penetración en el mercado ni son estadística alguna del uso empresarial de las diferentes soluciones. No obstante me ha parecido significativo el hecho de que ambos software tengan más presencia en la red que plataformas consolidadas, y que algunos como el caso de Openview, son casi un referente en este campo.

<http://www.google.com/trends?q=nagios%2C+openview%2C+unicenter%2C+tivoli%2C+bmc&ctab=0&geo=all&date=all&sort=1>

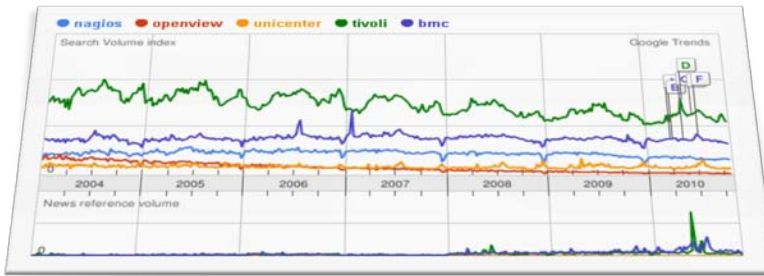


Figure 3 - Nagios vs. Software licenciado



Figure 4 - Zabbix vs. Software licenciado

1.4.2 GNU / GPL

- Opsview

Se trata de un producto que ofrece una distribución Enterprise en la que se pueden adquirir módulos adicionales para la generación de informes, integración con la herramienta de ticketing corporativa (de manera que se pueden abrir incidencias de forma automática), el envío de SMS como parte del sistema de alertas y un módulo especializado en la monitorización de las comunicaciones con soporte a la mayoría del equipamiento vigente.

Se encuentra dentro de la licencia GNU/GPL. Dispone al igual que la mayoría de SW de este tipo, de virtual appliances para una puesta en marcha rápida y sencilla de la plataforma.

<http://www.opsview.com/>

- Nagios / Nagios XI

Quizás una de las distribuciones primeras y más seguidas por la comunidad hasta hace unos años. Inició su andadura en 1999 y actualmente está patrocinada por Nagios Enterprise (Nagios XI), la cual ofrece, previo pago, una plataforma orientada a empresa. Según la página oficial de Nagios es

utilizada por más de 250.000 usuarios en todo el mundo y se implementa en multinacionales de renombre como IBM, HP, Google, etc.

Es sin duda la de un mayor uso dentro de las distribuciones *libres*, aunque no he optado por ella por la dificultad añadida (pareja a la potencia en la mayoría de ocasiones) que supone la implementación de esta distribución y por tener un entorno más parco para la gestión y generación de informes y gráficos. Al menos con la instalación base del producto. <http://www.nagios.org/>

- Icinga

Se trata de un software totalmente compatible con Nagios. En este caso también ofrecen soporte y mantenimiento. Aparte de la monitorización también venden una solución para la generación de informes y gráficos, y una solución de alertas basada en SMS, teléfono, email, etc.

Respecto a Nagios, Icinga ofrece API's para la integración con otros entornos, tiene detrás una BDD, mapas dinámicos y un *core* distribuido. Todo ello según la página oficial de Icinga. Ver fuentes electrónicas

- Zabbix

Dentro del software libre existe la licencia GNU/GPL. Este tipo de licencias implican una colaboración acordada con el proyecto GNU por ambas partes, en cuanto a que es GNU y por otro lado el GPL hace referencia a la libre distribución no sólo del software original sino de las modificaciones que puedan realizarse por cualquier otra organización. Dentro de este ámbito de distribución se encuentra el software seleccionado en esta implementación, Zabbix.

Pese a ser gratuito solicitan en su Web que, de usarse en entornos empresariales, se contrate algún tipo de soporte o mantenimiento que apoye el desarrollo del producto. Cito textualmente desde la Web del producto:

"ZABBIX software is released under the GNU General Public License (GPL) version 2. The formal terms of the GPL can be found at <http://www.fsf.org/licenses/>.

For additional details, including answers to common questions about the GPL, see the generic FAQ from the Free Software Foundation at <http://www.fsf.org/licenses/gpl-faq.html>.

If you use ZABBIX in a commercial context such that you profit by its use, we ask that you further the development of ZABBIX by purchasing some level of support."

En *Figure 5 - GPL Licenses* se puede ver el tipo de licencia de cada una de ellas (fuente Opsview oficial Web page).

	Nagios®	Nagios XI	Icinga	Zabbix	Opsview Community
Free (and always will be)	✓	✗	✓	✓	✓
Full GPL license	✓	✗	✓	✓	✓

Figure 5 - GPL Licenses

Presencia de las plataformas libres en la red, según Google Trends. Se puede apreciar claramente la mayor presencia de la comunidad Nagios, y en segundo lugar y con una ligera tendencia al alza, Zabbix. En todo caso son las dos plataformas con mayor presencia en la red, al menos en el buscador de Google.

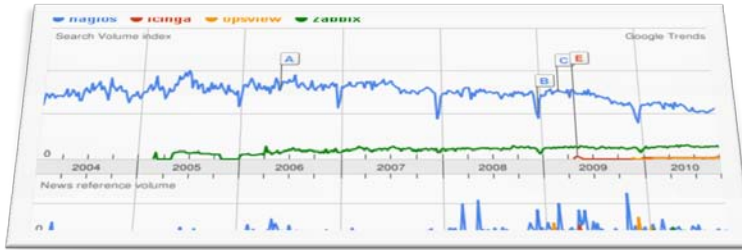


Figure 6 - Comparativa software libre

Después de ver los gráficos, nos podemos preguntar por qué no se ha seleccionado Nagios en vez de Zabbix. Pues básicamente porque su despliegue me parece más complejo, teniendo en cuenta que no dispongo de grandes conocimientos de Linux. El objetivo es implementar la monitorización, o al menos la Fase I, comentada al inicio de este texto, con los recursos existentes y en un plazo corto, ya que la dedicación no es *full time*. Por otro lado no pierdo de vista que la comunidad, donde más esfuerzos se aportan, es en proyectos de Nagios. Por ello no descarto la posibilidad de, una vez explotado Zabbix a fondo, si se detectaran carencias que Nagios pudiera suplir, migrar a él o directamente realizar la Fase II en Nagios.

Por el momento se ha instalado Zabbix a partir de una virtual appliance. Obviamente esto facilita mucho la labor inicial de crear una base estable para el software. De otra manera habría que instalar una *distro* estable, instalar las aplicaciones necesarias, subir los niveles de las librerías a los que el software necesita, configurar el servidor con los parámetros de la red que lo acoge.

Además, de forma nativa soporta multitud de clientes, especialmente interesante me parece el de HP UX, pues los servicios de ERP (SAP) están corriendo bajo este sistema operativo.

En este link hay una muy buena comparativa de las diferentes herramientas de monitorización , libres. <http://workaround.org/node/304>

1.5 Objetivos

Inicialmente se ha realizado un estudio de las posibilidades disponibles en el mercado de cara a la monitorización libre. Tras algunas pruebas con Opsview y Zabbix, al final se ha optado por Zabbix.

Durante el transcurso de la asignatura se recogerá información corporativa (pues esta información se utilizará para la propuesta de la Fase 2, que se desarrollará como colofón del trabajo) con el fin de conocer los servidores/servicios, procesos batch y comunicaciones existentes, y así estimar el alcance del proyecto.

Para el desarrollo de la Fase 1, necesitamos conocer qué servicios hay actualmente en productivo, si bien no es necesario conocer el número de ellos por el momento. Obviamente habrá que asegurarse de que es capaz de monitorizar estos servicios y la manera de llevarlo a cabo.

Se construirá la plataforma de Monitoring con un cliente mínimo de cada tipo, con el fin de estar seguros que la plataforma cumple con las necesidades de la organización. Se documentará tanto la configuración del aplicativo en la parte servidor y cliente, como los resultados obtenidos.

Una vez todo funcione de manera correcta con algunos agentes, se han de definir los *templates* del entorno que vamos a monitorizar. Cada uno con sus *hosts* determinados. Se ha de pensar si es mejor distribuir los grupos por servicios, tipos de Server, ubicación, etc.

Se definirán los *triggers* que harán saltar las alarmas y las acciones a llevar a cabo en cada caso. Envío de correo, envío de SMS a teléfonos móviles, generación de incidencias en la herramienta de *ticketing*, etc.

En la última fase del trabajo se pretende realizar un estudio de las tareas y recursos necesarios para el despliegue de la plataforma a toda la organización.

1.6 Beneficios

Con esta implantación se espera poder ser más proactivos en el nivel de servicio que se está dando desde infraestructura. Debido al crecimiento en el número de servidores y servicios, actualmente no conocemos con suficiente antelación los posibles problemas, tanto los que se han producido, como los que están próximos a ocurrir.

La plataforma antivirus centralizada Worldwide y la de actualizaciones de productos Microsoft (la mayoría de sistemas existentes son de este tipo) ya está desplegada con estupendos resultados. Pero aún carecemos

de un sistema que avise de los errores que se han producido en un proceso *batch*, el estado de la ocupación de discos, memoria y CPU (especialmente me parece interesante para el *cluster* ESX de Vmware que alberga más de 40 servidores). Los errores en los servicios Web de la organización se están detectando en la mayoría de casos por llamadas de usuarios con problemas. A partir de ahora recibiremos una alerta, tan pronto uno de estos servicios esté inoperativo. La comprobación básica es una comprobación del estatus que devuelve la petición GET HTTP, pero el aplicativo va más allá y es capaz de simular los clicks de ratón y la introducción de datos, con el fin de realizar comprobaciones más avanzadas.

Otro aspecto importante es el de las comunicaciones, sobre todo en los casos en que se dispone de líneas de backup o discriminación de servicios por línea. En estos casos, pese a no haber caído el servicio, es crítico conocer la línea afectada lo antes posible para poner medidas y restaurar el servicio lo antes posible.

1.7 Planificación del TFC

El proyecto se ha de realizar durante un trimestre lectivo. Está valorado en 7.5 créditos, lo cual requiere un mínimo de dedicación de 6 horas semanales. Antes de empezar, y con la perspectiva del curso por delante, creo que la dedicación será bastante mayor por dos motivos: no soy un gran conocedor de Linux, con lo cual habré de emplear bastante tiempo para solucionar pequeños contratiempos que puedan surgir, y por otro lado me gustaría aportar alguna parte de desarrollo en la parte cliente del aplicativo (aún por determinar, SQL, Oracle, Vmware, etc.)

La planificación del proyecto se ha realizado en una hoja de Project para utilizar el diagrama de Gantt como seguimiento. Permite ver la progresión y las dependencias de cada uno de los hitos y tareas. Se ha insertado bajo estas líneas. Se ha dividido en dos partes para una mejor visibilidad, ya que se veía muy pequeño si se incluía toda la información en una sola hoja.

En cuanto a las imágenes, el enunciado del TFC especifica muy claramente que no se abuse de ellas. Tras la PAC3 el consultor ha vuelto a hacer hincapié en el tema y en resultas he eliminado casi la mitad de las imágenes, la verdad es que muchas de ellas no aportaban información relevante al trabajo. No obstante al tratarse de una aplicación de monitoring con un fuerte componente de parametrización y mensajes gráficos, he visto conveniente dejar algunas. Creo que aún se podrían eliminar algunas más, pero en mi opinión perdería la visión que ofrece parte del documento, enfocado a explicar y detallar la implementación y parametrización del aplicativo. No he querido abandonar un estilo de redacción didáctico en la elaboración de este informe, y creo que sobre todo en lo que al capítulo 1 y 2 se refiere, las fotos aportan una visión clara y directa del funcionamiento de Zabbix. Se me ocurre un ejemplo muy claro: si hubiera realizado el

trabajo sobre Nagios y comentara que se 'crea un host', en ambas plataformas lo escribiría igual. La diferencia estriba en que mientras que en el caso de Nagios, debería mostrar un fichero plano de texto con la definición del host (algo críptico a priori), en el caso de Zabbix se muestra una ventana emergente con apenas dos campos trascendentes que necesitan cumplimentarse. Esta es la visión que quiero mostrar del aplicativo, pues uno de sus puntos fuertes (como ya he comentado) es la *interface* gráfica de configuración desarrollada en PHP. Por otro lado, parte de las mejoras que se han implementado por mi parte están relacionadas con la forma en la que se presentan los datos e iconos personalizados. En el resto de capítulos (del 4 en adelante) no se ha incluido apenas documentación gráfica porque no lo he visto necesario. Conforme he ido profundizando en Zabbix, Internet me ha supuesto una gran ayuda, tanto en la parametrización del aplicativo, como en la definición y comprensión de los conceptos básicos. No obstante, casi toda la información que he encontrado es en inglés, y me gustaría colaborar en el proyecto en español con este documento que aún mucha de la información encontrada con la experiencia propia de una implementación real.

Inicialmente se dejaba un *gap* de 2 semanas aprox. en la recta final (en la planificación original). En la medida de lo posible se intentará acabar el proyecto antes de la fecha indicada del 12 de Enero, ya que hay que completar otras asignaturas también, y siendo realista, durante el periodo de fiestas navideñas, la dedicación será sensiblemente menor. Al final no ha sido posible y el trabajo se concluyó en la fecha próxima a la entrega.

Dada la dedicación a tiempo parcial en el proyecto, he optado por asignar amplios márgenes a las tareas. Deberían de ser concluidas en menos tiempo del indicado en el diagrama de Gantt. Los tiempos indicados son estimaciones de los plazos para realizar cada una de las tareas. En ningún caso indican el tiempo de dedicación a cada una de las tareas, primero porque no lo he realizado nunca, y segundo porque seguramente se realizarán por partes, no de forma continuada, alternando trabajo, familia y TFC...

En otras ocasiones conviene especificar la duración de cada una de las tareas, sobre todo si hay varios recursos implicados, pero no es el caso que nos ocupa, y he optado por hacerlo como comentario. En resumen la planificación es más una estructura temporal jerárquica de las actividades que se llevarán a cabo, que un detalle minucioso de los tiempos a emplear.




































1.7.1 Desviaciones

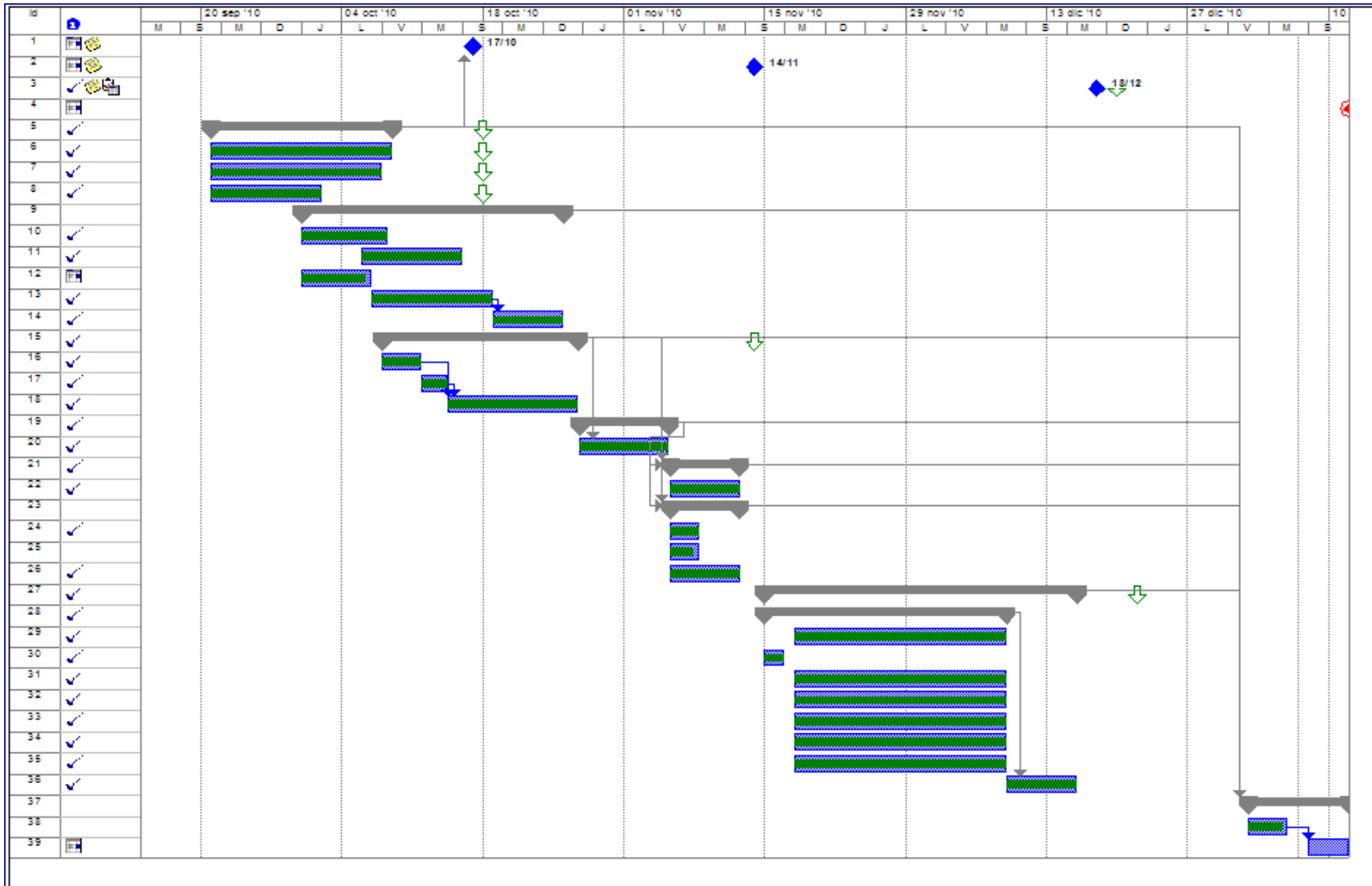
La desviación más importante es que esperaba haber terminado el documento final antes de acabar el año (ver párrafo anterior), pero debido a las fiestas familiares y a las entregas voluminosas que he tenido que realizar de otra asignatura me ha sido imposible terminar antes. Por lo demás el trabajo ha transcurrido con las previsiones que se hicieron en el diagrama de Gantt. Sino bien en cuanto al tiempo sí en la temporalidad y el orden cronológico. Algunas de las tareas que originalmente se especificaron no han sido llevadas a cabo, como la de generar reportes, que queda obsoleta en cuanto a que es algo deseable en todo caso para la Fase II del proyecto. Por otro lado comentar, que si bien en un principio el alcance de la Fase I iba a ser un entorno de pruebas y un informe final de la viabilidad, lo cierto es que se han adelantado acontecimientos, el proyecto fue aprobado, y por tanto se ha realizado una labor paralela de implementación en algunos aspectos. A fecha de hoy son ya bastantes los servidores incluidos en la monitorización básica, y la sustitución del checklist diario se producirá casi con seguridad la primera semana de Enero.

1.7.2 Correcciones

No se han requerido acciones remarcables para corregir problemas en la planificación aparte de las referentes a las imágenes. Sí que es cierto que en el transcurso de la PAC2 se hizo un esfuerzo extra en la estructura y diseño de este documento final, recuperando el tiempo empleado en la siguiente PAC. Así se comentó en la entrega de la PAC2.

Se han mostrado de forma separada el diagrama y la lista de tareas para facilitar la lectura, ya que la fuentes quedaba muy pequeña y prácticamente ilegible, si representaba todo junto.

1		Entrega PAC1
2		Entrega PAC2
3	 	Entrega PAC3
4		Entrega final
5		Fase previa
6		Presentación del proyecto
7		Justificación del proyecto
8		Planificación del proyecto
9		Recogida de información.
10		Un análisis de la infraestructura existente en la organización, se obtienen por motivos de seguridad las IPs, y toda aquella información susceptible de fec
11		Recogida de información de los servicios críticos para el negocio y los componentes de Hardware y software asociados.
12		Enlaces de datos entre sites remotos y LAN de HQ
13		Posibilidades del aplicativo de forma nativa.
14		Busqueda de clientes específicos (DHCP, DNS, PROXY, SQL, http, SMTP, VMWARE, procesos batch, etc.)
15		Construcción de plataforma central de Nagios
16		Seleccionar plataforma para instalar Nagios
17		Instalación de cero ó appliance pre-build VMware
18		Instalación de la plataforma-Zabbix
19		Instalación de clientes
20		Instalar clientes dentro del alcance del proyecto. [...]a determinar tras fase de recogida de información]
21		Construcción de informes
22		Grupos de informes
23		Generar alarmas ante eventos críticos.
24		Alarmas SMTP de envío de correo
25		Alarmas SMS a móviles para eventos críticos de servicio.
26		Alarmas visuales en monitores
27		Estudio de despliegue corporativo de la herramienta (Fase 2)
28		Estudiar viabilidad y dar estimación de recursos necesarios para:
29		Despliegue de los clientes de la Fase 1 a todos los dispositivos corporativos
30		Estudio de Integración de OCMIS del ERP SAP con Nagios (SNMP)
31		Recogida de información de los procesos batch e interfaces entre sistemas heterogéneos.
32		Cliente AS 400
33		Posibilidad de recoger información de Shor (IPS).
34		Estado de backups (Legato)
35		Recoger alertas de Nessus tras escanear.
36		Estudiar disposición final de pantallas en el edificio de HQ con el estado de los servicios según necesidades departamentales.
37		Maquetación documento final y presentación
38		Realizar el documento final
39		Realizar presentación en video



Capítulo 2 Implementación de la plataforma

En este capítulo veremos el proceso de instalación de la plataforma que servirá de base para la monitorización. Se han realizado pruebas con Opsview y Zabbix, y finalmente me he decantado por Zabbix, por motivos detallados más abajo.

Tras la instalación inicial es necesario parametrizar el sistema operativo y el aplicativo para encajar en la estructura de red de la organización. Así mismo se configurarán las alarmas de envío de correo para poder notificar las alertas obtenidas.

2.1 Instalar plataforma base

A día de hoy se ha implementado la solución Opsview instalándola en un entorno virtualizado con Vmware ESX. Se ha instalado el agente que distribuye Opsview en un *laptop* Windows 7 con el fin de realizar pruebas.

El proceso de instalación ha consistido en:

- Descargar la virtual appliance de la página oficial de Opsview.
- Migración del tipo de disco de VMPlayer a entorno ESX.
- Configuración de red del servidor Linux del aplicativo para trabajar con la red actual.
- Instalación del agente en un laptop Windows 7 y configuración del mismo
- Definir el *host* del laptop para su monitorización. Configurar el Firewall del laptop para permitir la comunicación con el servidor.

Las pruebas iniciales han sido satisfactorias, pese a que alguno de los ítems que se monitorizan no parecen recoger información de forma correcta. El resto funcionan OK.

De todas maneras, estoy mucho más contento con los resultados realizados con **Zabbix**. Los pasos que se han seguido han sido los mismos que con Opsview, aunque he podido realizar comprobaciones adicionales, como la disponibilidad de una página Web corporativa. Como ya se ha comentado Zabbix nos permite incluso simular los clics enviados al servidor Web y profundizar en la navegación para conocer la disponibilidad de apartados concretos, basado en el tiempo de respuesta.

El manual de que dispone Zabbix (más de 300 páginas) me está resultando de gran utilidad, y estoy encontrando mucha información en foros y documentos de la red.

- Descargar la **virtual appliance** de la página oficial de Zabbix.
- **Migración** del tipo de **disco** de VMPlayer a entorno ESX. Los discos utilizados en VMPlayer difieren de los de ESX. Se pueden convertir manualmente o utilizar el software de VMWare Converter para ello. He optado por la segunda opción y se ha instalado sin problemas.
- **Configuración** del servidor **Linux** del aplicativo para trabajar con la red actual. La configuración de red. Se ha modificado el teclado por defecto para trabajar cómodamente desde la consola

```
Neteye: /usr/share/kbd/keymaps/i386/qwerty # loadkeys es.map.gz
```

Se ha activado el uso del ratón por comodidad, para trabajar desde la consola.

```
Neteye: /etc/init.d # gpm start
```

Se ha modificado el fichero */etc/ntp.conf*, añadiendo las siguientes líneas.

```
server 192.168.32.xxx
server Name.mydomain.com
```

Mediante el comando */etc/init.d/ntp restart* se reinicia el servicio de sincronización horaria en Linux.

Se fija el cambio para que se mantenga en reboots posteriores con *chkconfig --level 345 ntp on*, y comprobamos con *ntpq -p* la diferencia horaria con el time Server.

```
Neteye: /etc/init.d # chkconfig --level 345 ntp on
Neteye: /etc/init.d # ntpq -p
  remote      refid      st t when poll reach  delay  offset jitter
=====
*LOCAL(0)    .LOCL.      101 5 64 377 0.000 0.000 0.001
srvName.mydoma 85.125.223.113 3 u 676 1024 377 0.891 -642.29 31.257
```

2.2 Configurar plataforma

Con el fin de que funcionen bien los sufijos **DNS** (actualmente no estaba funcionando bien, habiéndose modificado tan solo el archivo **/etc/resolv.conf**) se han realizado las siguientes configuraciones:

```
En /etc/sysconfig/network/config file:
# NETCONFIG_DNS_STATIC_SEARCHLIST="loquesea.domain.local"
# NETCONFIG_DNS_STATIC_SERVERS="192.168.x.x"

Y ejecutar :
linux-nnmr:/etc # netconfig update -f
linux-nnmr:/etc/INIT.d# ./network restart
```

En el fichero **/etc/php5/apache2/php.ini** se ha de modificar la siguiente línea para que las páginas ofrecidas por Zabbix vía Web contengan la hora correcta de la **zona local**, en este caso **Europa/Madrid**.

```
File : /etc/php5/apache2/php.ini
Original: date.timezone = Europe/Riga

New lines:      ;date.timezone = Europe/Riga → Línea comentada con ';'
               date.timezone = Europe/Madrid
```

Pese a que el *frontend* del aplicativo se puede modificar para que aparezca en castellano, he preferido dejarlo en inglés por un motivo: la búsqueda de información en Internet es mayor en este idioma, y tener mensajes y nombres de campos, ventanas, etc, en este idioma, facilitará la tarea de búsqueda.

Con el fin de tener un control más detallado de los errores se puede revisar el Log del aplicativo en **/var/log/zabbix/zabbix-server.log**. Gracias a este Log se pueden detectar fácilmente 'items' que no están soportados o están dando error por algún motivo.

Se ha cambiado el **logo** de Zabbix en la parte superior derecha para que muestre el logo corporativo

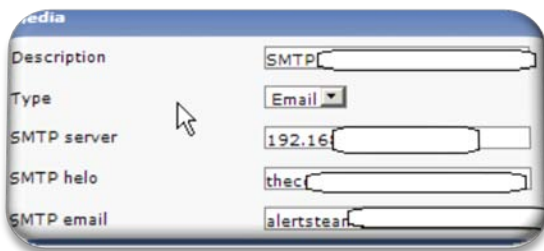
Se trata del archivo `zabbix.png` (118x31 pixels). Que se encuentra en la carpeta `/usr/shared/Zabbix/images/general`.

Para el envío de emails se utilizará un servidor SMTP corporativo. Con el fin de que los usuarios del aplicativo puedan utilizarlo se define como *'media type'* dentro de Zabbix. Así de sencillo !!!

En el campo SMTP Server indicamos la IP del servidor SMTP corporativo.

Ex: myEmaildomain.com

El SMTP helo será el del dominio corporativo. *Ex: myEmaildomain.com*



El SMTP email indica el *'sender'* de las notificaciones. En este caso debemos utilizar la dirección real del dominio de correo, ya que sino las reglas antispam no permitirían salir al mensaje. *Ex: alertsteam@myEmaildomain.com*

Figure 7 – Media type para el envío de correo.

Otro cambio pequeño pero muy vistoso de cara a la foto final, es la modificación de los logos básicos entregado por el aplicativo en la instalación inicial.

He encontrado en la red otros administradores de Zabbix que han puesto a disposición del resto logos modificados, o completamente nuevos para las diferentes opciones de Zabbix, estado de los hosts (Off, up, maintenance, etc.), diferentes OS (Linux, Windows, etc.) Por mi parte voy a contribuir con algunos iconos modificados, como los siguientes:

Se han modificado los iconos de **OS**.



Se han modificado los iconos de **estado**




Se han descargado iconos para los equipos de red, servicios específicos, SMTP, FTP, http, etc.

En algunos casos se han hecho los iconos con el fondo transparentes, modificándolos y guardándolos como formato *png*. De esta manera, en el mapa, no tenemos un cuadrado tapando el fondo, sino sólo se cubre la parte del dibujo.



Como se aprecia a la izquierda, el dibujo del router tapa el fondo, pero no así el

cuadro blanco de alrededor. Da una visión de estar más integrado en el mapa.

En algún caso, se han mezclado dos iconos para indicar varias cosas a la vez. Como por ejemplo un servidor Windows con problemas. De la misma manera, se ha cambiado el  fondo a transparente.

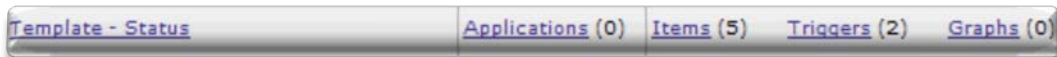
2.3 Parametrización de Zabbix

Antes de empezar a crear objetos en el aplicativo, es preferible decidir antes la estructura de objetos que vamos a utilizar.

Las plantillas en Zabbix son jerárquicas, por lo que la pertenencia puede ser anidada.

Es aconsejable también, el uso de grupos de Host de cara a la monitorización y a la administración de los elementos. Las gestiones se realizan sobre todo el grupo.

En primer lugar se crea un Template de objetos simples que será común a toda la plataforma Wintel.



En ella se monitorizan y se disparan alertas básicas comunes a todos los equipos. Tales como el estatus, el tiempo que lleva la máquina arriba y el espacio libre en los *drives* básicos, el c:\ del sistema operativo y la D:\ o E:\ para datos o aplicativos.

Log	Description	Triggers	Key
<input type="checkbox"/>	Free disk space on e:		vfs.fs.size[e:,free]
<input type="checkbox"/>	Host status		status
<input type="checkbox"/>	Host information		system.uname
<input type="checkbox"/>	Host uptime (in sec)	Triggers (1)	system.uptime
<input type="checkbox"/>	Host status	Triggers (1)	icmpping[]

Figure 8 – Template Status – Comprobaciones básicas comunes

En el capítulo 3 se verá con detalle la creación de los diferentes objetos con la intención de facilitar el despliegue de la monitorización y facilitar en la medida de lo posible la administración. Por otro lado la pretensión es crear usuarios restringidos y formar a primer nivel de HelpDesk para que de forma autónoma puedan gestionar la herramienta.

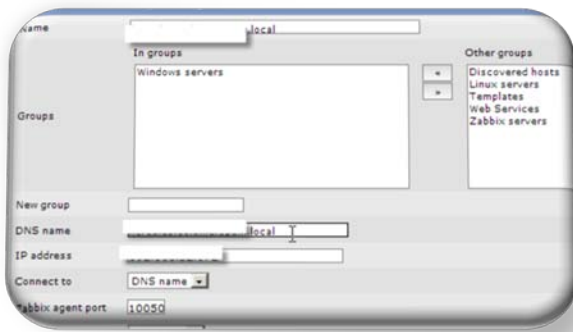
2.4 Instalación Clientes:

2.4.1 Instalación básica agente Windows.

Instalación del agente en un servidor Windows Server 2003 y configuración del mismo. La instalación ha resultado muy sencilla. El agente se ha desarrollado en C y está disponible para su descarga, pre compilado para multitud de plataformas. Tras la instalación se ha modificado a mano el fichero de configuración del cliente en “C:\Archivos de programa\Zabbix Agent\Zabbix_Agentd.conf” con las siguientes modificaciones.

```
DebugLevel=1 → para poder ver si hay errores en el cliente
LogFile=C:\Archivos de programa\Zabbix Agent\Zabbix_agentd.log
EnableRemoteCommands=1 → para probar la ejecución de comandos remotos
Server=1.1.1.1 → Dirección IP del servidor Zabbix (ip no real)
Hostname=clientName.domain.local → necesario para chequeos activos
ListenPort=10050 → El puerto de escucha se mantiene por defecto
```

2.4.2 Monitorización simple de una Web. Página de inicio.



Simplemente se ha de definir el servidor que alberga la Web como host. Se puede hacer referencia a él (dentro del aplicativo), por IP o por nombre DNS de dominio. (véase *Figure 9 - Creación de un host*)

Figure 9 - Creación de un host

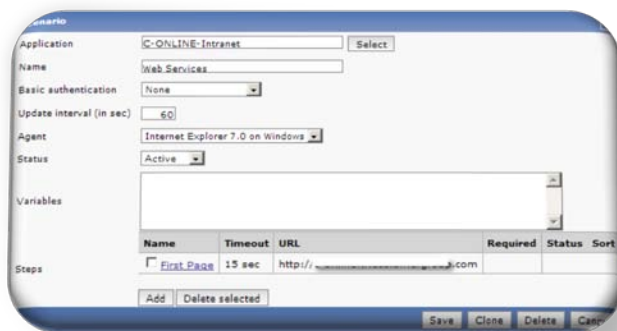
Ya se encuentran disponibles *templates* para monitorizar sistemas Windows, por lo que no es necesario crear ítems para comprobaciones básicas relativas al espacio libre, memoria, etc. Sí que se ha creado un *item* nuevo para monitorizar la unidad de disco e:\, pues el estándar contemplaba sólo

c:\. Este nuevo *item* se puede reutilizar para otro host, no hay que crearlo en cada ocasión. Estamos adelantando al idea del uso de los *Templates*.



Figure 10 - Vista de host (nombre e ip's borrados)

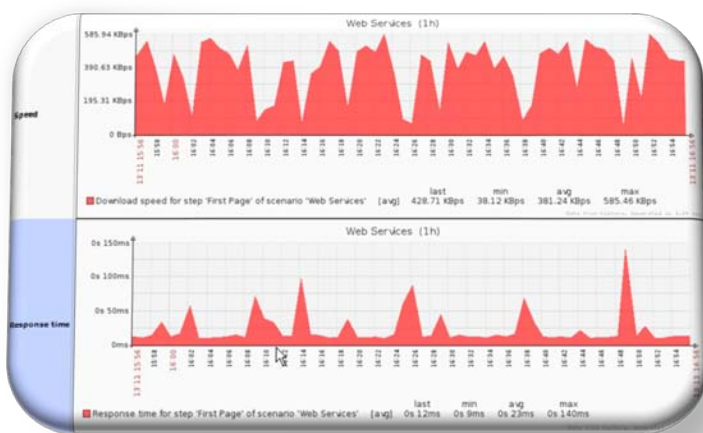
Se ha creado una comprobación de la intranet corporativa en Configuración-Webs. Para ello se crea un *scenario* para la comprobación de la Intranet



Sobre este escenario se configura un *STEP* para monitorizar que la página principal de la intranet esté disponible. Al igual que en imágenes anteriores se han ocultado las referencias a nombres o IP's de la organización.

Figure 11 - Creación de escenario

En el caso de la Web <http://loquesea.com> podemos obtener incluso gráficos de su disponibilidad y



el tiempo de respuesta en milisegundos. Actualmente no tenemos negocio basado en la red, pero de ser así este tipo de herramientas pueden resultar muy útiles.

Figure 12 - Disponibilidad y velocidad de respuesta de la Web <http://loquesea.com>

2.4.3 Monitorización de espacio en unidad local

Aquí se aprecia la manera en que la información se muestra por pantalla. Se trata de un *item* que mide la ocupación de disco. Se aprecia que ahora aparece una comprobación para el disco E:\ del



servidor. En la <

Figure 13 - Visualización base del estado de un Server + disco E:\ se aprecia esta información. Se puede llegar a ella desde el

menú Monitoring-Overview y seleccionando 'Windows servers'.

Figure 13 - Visualización base del estado de un Server + disco E:\

2.4.4 Monitorización SNMP

Para aquellas máquinas de las que no se dispone de agente nativo se puede utilizar SNMP (por ejemplo AS400). SNMP es un protocolo IP de gestión de red. Con él podemos obtener información de los dispositivos mediante consultas SNMP. Zabbix ya dispone de forma nativa de objetos predefinidos para realizar este tipo de consultas. Durante la fase inicial de identificar los objetos SNMP que queremos monitorizar, puede ser de gran utilidad el uso de *snmwalk*, un script de Linux que nos permite realizar consultas SNMP a hosts con el fin de saber si esos MIBS están disponibles, y ver la información que devuelve. El ejemplo de abajo realizaría una consulta al host indicado con la versión de protocolo SNMP '1' y 'public' como nombre de comunidad de lectura.

```
Neteye#> snmpwalk -v 1 -c public -t 10 192.168.##.## .1
```

Capítulo 3 Análisis del aplicativo - Jerarquía de objetos

A lo largo del capítulo veremos una descripción detallada y un breve resumen del uso de cada uno de los principales objetos del aplicativo. Veremos ejemplos de uso de algunos de ellos y algún 'atajo' que puede facilitar la ardua tarea inicial de configurar la aplicación, como el uso de la herramienta '*discovering*' junto con una '*action*', que asigne grupos o template a los *hosts* encontrados en la red. En este capítulo entramos a fondo en los conceptos básicos de Zabbix y su estructura. El capítulo se ha desarrollado con el fin de acercar al lector tanto una idea global del funcionamiento del producto, como del detalle de los conceptos, con claros ejemplos que nos ayuden a interrelacionarlos. Obviamente no puede ser de otra manera, pues los objetos están fuertemente vinculados en el diseño de la aplicación, y es esta visión la que se pretende transmitir en estas líneas. Obtendremos una visión global y un breve resumen durante la introducción, y posteriormente, se entrará a tratar cada objeto en detalle.

3.1 Introducción

Como ya se ha comentado con anterioridad, la etapa inicial de esquematizar la jerarquía de objetos, será crucial conforme avance el proceso de monitorización. Para ser sincero y dado que es la primera implementación, lo más seguro es que el esquema inicial sufra ligeras modificaciones conforme se avance en el proyecto.

A modo de visión global del funcionamiento del aplicativo, decir que está basado en *hosts*. Estos hosts son objetos (servidores, switches, routers, etc.) de los que queremos conocer cierta información (estado del sistema, logs, consumo de recursos, etc.). Esta información son los llamados '*items*' en el aplicativo. Con el fin de facilitar la monitorización y asignación de permisos, es interesante agruparlos en '*Host groups*', para una gestión más sencilla.

Una vez recogida la información del Host, es deseable que notifique al administrador u operador cuando alguno de los umbrales está próximo a generar un problema. Los objetos que cumplen este cometido son los '*triggers*'. Como en el resto de casos, será preferible que la gestión sea todo lo común posible, aunque en algunos casos serán triggers específicos de un Host. Un claro ejemplo puede ser la comprobación de espacio en un recurso local. En la mayoría de casos una alerta al 80% de ocupación, por ejemplo, sería suficiente. No obstante hay servidores antiguos, donde el espacio está muy ajustado, hay poco movimiento, y la cuota de ocupación ya se encuentra en un 90%. En estos casos el disparador o *trigger*, tendrá que ser diferente.

Una vez tenemos disparado un *trigger*, este se reflejará en el GUI de Zabbix con un pop up (o ventana emergente), pero si además queremos que se inicien acciones de notificación o actuaciones en los sistemas, deberemos crear '*actions*'. Estas acciones nos permitirán desde enviar un SMS o un email al responsable del servicio/Host afectado, hasta el lanzamiento de scripts locales o remotos que inicien una acción para solucionar el problema o mitigarlo. Por ejemplo, el estado '*down*' de un servicio en Windows, puede disparar un trigger, que a su vez ejecute una *action*, y ésta reinicie el servicio en el Host afectado. Estas acciones se definen como envío de mensajes o ejecución de script remoto. En el caso del script remoto, debemos tener activado el EnableRemoteCommands (=1) en el zabbix_agentd.conf file (fichero de configuración del agente Zabbix).

El ejemplo descrito más abajo nos permitiría reiniciar un Host Windows ante una condición detectada que lo requiriese. Se trata pues de un **monitoring activo**.

PARAMETER	Description
Action type	'Remote command'
Remote command	host:c:\windows\system32\shutdown.exe -r -f Replace 'host' with Zabbix hostname of Windows server.

3.2 Modelo

3.2.1 Hosts

Se trata del objeto básico de monitorización, puede ser cualquier dispositivo de una red IP con el agente Zabbix instalado o SNMP activo para su monitorización. En la monitorización activa, Zabbix irá a estos hosts de forma periódica a buscar la información requerida. En la comunicación pasiva, será el host quien conectará con el nodo Zabbix .

3.2.2 Items

Los *items* engloban aquella información específica que se desea conocer sobre un *host*. Son bastante flexibles y heterogéneos, pues tenemos desde consultas SNMP, a agentes predefinidos que recogen información local, scripts personalizados que corren localmente en el Host y que permiten recoger información, *items* que trabajan sobre información ya recibida, solicitudes de ejecución de scripts por Telnet o ssh, etc.

Habitualmente se recibirán valores como un ‘0’ o un ‘1’. Para facilitar la lectura de la información en pantalla, y la manipulación, pues siendo numérica permite operadores aritméticos, disponemos de los llamados ‘*value mappings*’ que permiten trasladar los valores recogidos por unos más ‘*human readable*’. Ex: ‘0’ → error ; ‘1’ → OK.

Abajo podemos ver el mapeo y el valor entre paréntesis. Valor=1 → Muestra **OK (1)**

Muestra OK del *value mapping* y el valor original entre paréntesis. (Véase *Figure 14*)



Figure 14– Value mappings- Permiten traducir la información recogida

El uso de macros también es muy potente. Desde la versión 1.6.1 se introdujeron en los *items*. Nos permiten por ejemplo, en un *template*, hacer referencia a propiedades del Host. De esta manera un *item* creado bajo el grupo ‘SMTP Servers’ lo podemos compartir con el resto de Hosts que tengan *SMTP services* (que tengan este *template* asociado, para ser *exactos*). La macro HOST.CONN tendrá el valor de la IP o el Domain name del host en cuestión. Depende de cómo lo hayamos especificado en la creación del *host*.

Para más datos acerca de las macros predefinidas, consultar Doc. On-line de Zabbix o el manual 1.6 en PDF. Véase <http://www.zabbix.com/documentation.php> Doc. on-line.

En la fecha de entrega de este documento, ya se ha liberado la minor release 1.8.4 que extiende el uso de macros a los triggers adicionalmente, entre otras cosas.

3.2.2.1 Disponibilidad del Host

En esta categoría y como comprobación básica se chequea que el host responda a ping o que el agente Zabbix_agentd conteste. En ambos casos son paquetes echo ICMP lo que se envía.

También se ha creado un *item* que recoge información relativa a la versión del cliente Zabbix instalado en el cliente con la *key* que se observa en *<Figure 15 – Versión de agente Zabbix instalada. Útil,* por ejemplo, cuando queramos subir de versión el cliente.

Description	Version of zabbix_agentd() running
Type	Zabbix agent
Key	agent.version
Type of information	Character

Figure 15 – Versión de agente Zabbix instalada. Útil de cara a upgradear la versión

3.2.2.2 File Systems

La comprobación de los FS resulta un poco engorrosa de unificar. El problema es que los ratios específicos son variables en función del Host. Para un FS de 200Gb, sería deseable que avisara cuando tenga menos de 10Gb (por poner un ejemplo), en cambio para una unidad C:\ de 10Gb, el aviso sería deseable que se produjera al tener menos de un 1Gb. Una de las posibilidades es establecer cuotas relativas al tamaño global.

```
below 10% | ({{Template - Status:vfs.fs.size[c:,free].last(0)}}/({{Template - Status:vfs.fs.size[c:,total].last(0)}})*100<10
```

En la imagen de arriba vemos una alerta establecida para el 10% de espacio libre sobre la unidad de disco c:\.

Otra posibilidad, y seguramente la que deberé implementar en varios Host, sobre todo los más antiguos, pasa por establecer *triggers* específicos para estas máquinas. Creo que la generalización facilita la tarea administrativa, pero no se ha de perder el objetivo final del proyecto que es la monitorización de una plataforma real.

En el *template* Template-Status, el template creado para las monitorizaciones básicas, se han incluido los siguientes *items* (Figure 26):

Log	Description	Triggers	Key
1	Disk space on e: Free		vfs.fs.size[e:,free]
2	Disk space on c: Free		vfs.fs.size[c:,free]
3	Disk space on e: Free in %	Triggers (1)	vfs.fs.size[e:,pfree]
4	Disk space on d: Free in %	Triggers (1)	vfs.fs.size[d:,pfree]
5	Disk space on c: Free in %	Triggers (1)	vfs.fs.size[c:,pfree]
6	Disk space on c: Total		vfs.fs.size[c:,total]

Figure 16 – Comprobación de ocupación de File Systems.

El **1** y **2** nos muestran el espacio libre en las unidades e:\ y c:\ respectivamente

El **3**, **4** y **5** nos muestran el % de espacio libre en las unidades E:\, D:\ y C:\ respectivamente.

Nótese que, como se ha comentado anteriormente, se han añadido *triggers* que disparan en caso de que la cuota de espacio libre en % sea inferior al 10%.

El **6** nos muestra el espacio total de la unidad C:\

Un ejemplo del *trigger* sería:

Severity	Name	Expression	Status
Average	Disk Space below 10% on 10	{Template - Status:vfs.fs.size[c:,pfree].last(0)}<10	Enabled

Figure 17 – Trigger comprobación % de espacio libre en C:\

En la *Figure 17* se observa que se comprueba que el último dato devuelto por pfree (% de espacio libre en la unidad) sobre el objeto c:\ es inferior a 10, lo cual equivale a comprobar el hecho de que en la unidad C:\ tengamos menos del 10% de espacio libre.

Dado que se aplican por *Template* los *items* y los *triggers*, nos encontraremos que hay servidores que no tienen unidad D:\, o que tienen unidades F:\ o G:\. He decidido que dado que la mayoría incluyen la C:\ y adicionalmente D:\ o E:\, estos *items* se incluyan en el *template*. Las unidades adicionales se añadirán a mano en cada uno de los *hosts*. Por otro lado, es conveniente no dejar *items* en el *status* 'not supported' por lo que se deberemos desactivar la recogida de información de aquellos *items* que lean ocupación de FS inexistentes. En el ejemplo de abajo se deshabilita la comprobación sobre D:\ por no existir dicha unidad en uno de los *hosts* (véase *Figure 18*).

Disk space on d: Free in %	Triggers (1)	vfs.fs.size[d:,pfree]	300	7	365	Zabbix agent	Disabled
----------------------------	--------------	-----------------------	-----	---	-----	--------------	----------

Figure 18 – Ejemplo de Item deshabilitado

3.2.2.3 Estado de servicio

Existen varias formas de comprobar el estado de un servicio aplicación. Desde la más sencilla como comprobar si un servicio está activo, a las más sofisticadas (siempre se debería evaluar la calidad de la comprobación y la cantidad de recursos necesarios para llevarla a cabo). En el caso de **SMTP** una comprobación sencilla pasa por utilizar SMTP de 'simple checks' (nativo en el *appliance* de Zabbix).

Otra forma similar de hacerlo sería con la comprobación del servicio de red TCP por el puerto 25 (en este caso el puerto SMTP estándar). Véase *Figure 30*

Description	SMTP - net.tcp.service
Type	Zabbix agent
Key	net.tcp.service[smtp,{HOST.CONN},25]
Type of information	Numeric (unsigned)
Data type	Decimal

Figure 19 – Ejemplo comprobación 'Listening' en puerto 25 TCP (usualmente SMTP). Alternativa

Nota: En este último caso se ha utilizado una macro que se introdujo en la versión 1.6.1 {HOSTNAME},{HOST.CONN},{IPADDRESS},{HOST.DNS}.

En el caso de los servicios de Windows existe la posibilidad de hacer comprobaciones directamente, tan sólo con el agente Zabbix instalado. Por ejemplo, si se quiere conocer el estado de un servicio SQL Server de Microsoft, basta utilizar la `key="service_state[SQLServerAgent]"`, de tipo numérico unsigned.

Esta comprobación devuelve un número con el siguiente ‘*value mapping*’

Windows service state	Value
0	Running
1	Paused
3	Pause pending
4	Continue pending
5	Stop pending
6	Stopped
7	Unknown
255	No such service

Así un servicio parado mostraría el mensaje ‘Stopped(6)’

Comprobación de servicios MS SQL Server (Véase *Figure 20 - Datos básicos recopilados de Servicio MS SQL Server*) En este ejemplo, se observa un posible *trigger* que sería conveniente. Vemos que el *logfile* es bastante mayor que el *datafile*, y las transacciones no son muy grandes. Seguramente nos está indicando que el vaciado del log tras la copia de seguridad del *datafile* es erróneo o no se ha podido purgar y comprimir el *logfile*. Por tanto sería conveniente un *trigger* que avisara, por ejemplo, cuando el *logfile* se acerque al *datafile* en espacio.

SQL: Data File Size	1.78 Gb
SQL: Database Pages	63866
SQL: Log File Size	6.54 Gb
SQL: Number Failed Jobs	22
SQL: Service State - SQL Agent	Running (0)
SQL: Service State - SQL Browser	Stopped (6)
SQL: Service State - SQL Server	Running (0)
SQL: Total Server Memory	866.53 Mb
SQL: Transactions per second	0

Figure 20 - Datos básicos recopilados de Servicio MS SQL Server

3.2.2.4 Consultas SQL

De momento se está comprobando que los servicios SQL estén activos.

Se pretende poder realizar en la Fase II un **UserParameter** en los agentes que realice consultas SQL y recoja información de los SGDB MS SQL Server, Oracle y MAX DB. A priori me gustaría empezar por tener log de las tareas de mantenimiento diarias y semanales y de avisos de problemas críticos de los motores de BDD.

3.2.2.5 Comprobación de red

Se realiza la comprobación de rutas WAN entre *sites*. Dado que disponemos de líneas redundadas entre algunas *sites* (dan cabida a servicios diferenciados) se hace necesaria la comprobación de cada una de ellas. Un simple ping a los diferentes servicios habría sido suficiente para comprobar las líneas (podrían ser dos Host por tema de tolerancia a fallo en los OS, y no a los comunicaciones), sin embargo dado que los paquetes ICMP se mueven en una capa OSI intermedia entre la 3 y la 4, es decir se encapsula en datagramas IP, pero no utiliza puertos concretos ni campos de direccionamiento, en ocasiones los *pings* alcanzan el destino, pero la ida y la vuelta no se realiza por el mismo canal físico. Esto enmascara el hecho de que un problema de *routing*, por ejemplo deje servicios que utilizan capas superiores (orientados a conexión) sin funcionar. En la práctica he podido comprobar este problema en diferentes ocasiones. Para salvar este escollo se ha desarrollado un script en *bash* (shell script) que recibe 3 parámetros, **\$1** con la dirección de destino del comando *traceroute*, **\$2** con la cadena que se espera encontrar en el *log* del comando y **\$3** con la cadena que aparecerá en el resultado del script. En caso de no llegar a destino (problema de routing, DNS, línea, etc.) devuelve 'error', en caso de llegar a destino por la línea correcta devuelve OK-Route-\$3, en caso de llegar por una ruta alternativa devuelve OK-Route-Other. Se puede utilizar con un único parámetro, devolviendo entonces 'OK' o 'error' en función del resultado.

```
#!/bin/bash
LOGFILE=/tmp/z_trace.log;
rm -f ${LOGFILE} 2>/dev/null;
salida=error
if test -z "$1"
then
echo "No command-line arguments."
exit 1;
fi
# Trace to host passed as parameter $1
traceroute -m 5 ${1} >${LOGFILE} 2>&1
if [ $? -eq 0 ]
then salida=OK
fi
wait
# chown zabbix:zabbix ${LOGFILE}
chmod 666 ${LOGFILE}
check=`grep -i "H!" ${LOGFILE}|wc -l`
check2=`tail -1 ${LOGFILE} |grep -i "\* \*"|wc -l`
ch=`expr $check + $check2`
if [ ${ch} -gt 0 ]; then
echo error
exit
fi
if [ $# -eq 1 ]; then
echo $salida
exit
fi
if [ $salida != "OK" ]
then echo error
exit
fi
wl=`grep -i ${2} ${LOGFILE}|wc -l`
if [ ${wl} -gt 0 ]; then
ruta=OK
else ruta=Other
fi
if [ $# -eq 2 ]; then echo OK-Route-$ruta;exit;fi
if [ "$ruta" = "OK" ]; then
echo OK-Route-$3;exit
else echo OK-Route-Other ;exit
fi
```

Otras de las comprobaciones que se han implementado es el ping a los diferentes host para comprobar que tienen visibilidad IP.

A partir de la información devuelta por el `z_trace` comentado anteriormente se ha añadido un *trigger* en los enlaces entre centros de estos mapas (que indican el enlace de comunicaciones WAN). La idea es que cuando el *trigger* se dispare porque falla la línea, el color y la forma de la línea varíen para informarlo, en nuestro caso un resultado positivo del trigger cambiaría la línea de verde a rojo.



Figure 21 – Línea OK (línea verde)



Figure 22 – Línea con Error (red dashed line)

Una mejora significativa podría ser añadir un *item* que monitorice el ancho de banda consumido, y en caso de llegar al CIR de la línea, cambiar el color a naranja, indicando saturación.

3.2.2.6 Recursos del sistema

De forma nativa, los agentes Windows permiten recoger información detallada de varios recursos. Memoria, CPU, y performance counters (bandwidth NIC, paginación, etc), entre otros. Es posible también definir nuevas consultas añadiendo *UserParameter* personalizados en el fichero de configuración de Zabbix-agent, y recoger esta nueva información de manera activa desde el servidor Zabbix.

Se ha realizado la prueba de añadir una key nueva llamada 'DominoSMTP' no incluida en la configuración nativa del cliente (véase *Figure 23 – Creación del nuevo item tipo 'Zabbix Agent' personalizado 'DominoSMTP'*).



Figure 23 – Creación del nuevo item tipo 'Zabbix Agent' personalizado "DominoSMTP"

Por tanto en Host cliente se ha de añadir un nuevo *item UserParameter* y reiniciar el servicio Windows de Zabbix.

Incluiremos esta línea y reiniciamos el servicio (*Figure 24 – Definición de item personalizado en el cliente*).

```
PerfCounter= DominoSMTP, "\Process(nsmtp)\Handle Count", 60
```

Figure 24 – Definición de item personalizado en el cliente

3.2.2.7 Zabbix Trapper

Este modo de alertas pasivas permite enviar el contenido de una *key* de un *host* a un *item* del servidor Zabbix. Se puede realizar desde cualquier Host con el agente instalado, para ello se utiliza **Zabbix_sender.exe**. A modo de ejemplo se ha creado el *item* que se aprecia a la derecha (Figure 25 – Ejemplo de *item* tipo Zabbix Trapper). En el envío se utilizará el mismo nombre que aparece en el campo 'key', en este caso 'test.error'.

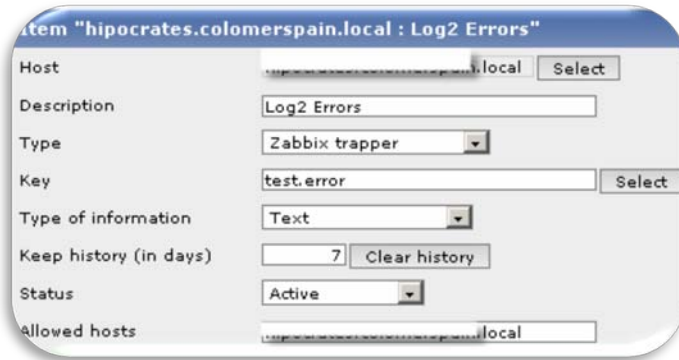


Figure 25 – Ejemplo de *item* tipo Zabbix Trapper

Este nuevo *item* de la figura de arriba recoge la información enviada con el comando `zabbix_sender.exe`, véase la imagen inferior de un envío de prueba:

```
C:\Program Files\Zabbix Agent>zabbix_sender.exe -z192.168.1.100 -s hipocrates.colomerspain.local -k test.error -o "UOC Test de envío a Zabbix"
Info from server: "Processed 1 Failed 0 Total 1 Seconds spent 0.000707"
sent: 1; skipped: 0; total: 1
```

Hemos creado un *screen* que muestra textos planos a partir de los datos recibidos de los *items*. Y esta es la información del ejemplo anterior tal y como aparece en el *screen*. Nótese que el parámetro '-o' especifica el texto a enviar. En este caso el acento lo entiende como un salto de campo. A tener en cuenta a la hora de generar envíos !!!



3.2.2.8 Otros

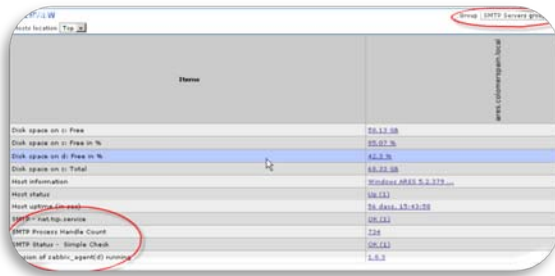
- En los *Hosts* donde se ha desplegado el agente Zabbix, será de utilidad conocer la **versión** del mismo que tenemos instalada. Esta monitorización me parece también necesaria con el fin de identificar que los futuros *upgrades* se despliegan de forma correcta en toda la plataforma.
- Otra información útil es el *up.time* o tiempo que lleva el sistema encendido. Principalmente nos permite alertar cuando un host ha sido reiniciado. Este reinicio de no haber sido planificado, puede deberse a la existencia de un error en el sistema.

En caso de reinicio de un equipo se recibe una notificación vía email del suceso, indicando fecha y hora del evento y el nombre del host afectado

- **Typeperf** nos permite desde los servidores Windows, conocer estadísticas del sistema, que se pueden consultar mediante el agente `zabbix_agentd`.
Ex: **`typeperf -qx | findstr smtp`** (listaría estadísticas relacionadas con SMTP, por ejemplo).
- Con el fin de detectar fallos en procesos batch, he encontrado a priori dos formas de comprobarlo. Una de ellas consiste en leer el log remotamente con un tipo de item específico para estos casos. Incluso se dispone de forma nativa de un tipo de lectura de ficheros de log para logs rotativos (por ejemplo, los que incluyen la fecha o la hora en el nombre de archivo). Se puede leer el log completo, y establecer un trigger que se dispare en caso de encontrar cadenas específicas, o directamente se puede crear un item que tan sólo recoja la información significativa. La otra forma pasa por el envío con `zabbix_sender.exe` de la información relativa al error a un item tipo `zabbix trapper`. Del primer caso se ha implementado a modo de ejemplo la recogida de información de un fichero de log remoto en un servidor. Paralelamente se ha configurado el aplicativo de copias de seguridad (Legato Networker de EMC) para que genere un log en caso de producirse problemas en las copias diarias. De esta forma podemos comprobar si han habido problemas en el proceso diario de backup, simplemente revisando este archivo una vez al día. La mayoría de días no incluirá información nueva, con o que sabemos que el proceso ha sido correcto.

3.2.3 Host Groups

Para una mejor gestión y monitorización del entorno resulta imprescindible una clasificación de los Hosts en grupos. Un hosts puede pertenecer a varios grupos, por lo que una primera aproximación interesante, puede ser agrupar por servicios (un Host puede estar en el grupo ‘Windows servers’ y en el de ‘SMTP servers’, etc.). Es un trabajo adicional pero veremos que resulta interesante hacerlo así por varios motivos:



queremos visualizar y en qué conjunto de hosts.

Figure 26 – Monitoring – Overview. Selección de visualización de SMTP Group

Monitorización: Ante una gran cantidad de datos, puede resultar difícil ver toda la información sin hacer scroll en la pantalla. La visualización de monitor-overview, por ejemplo, se puede realizar a nivel de grupo. De esta manera podemos seleccionar únicamente el servicio que

Permisos: Los permisos a los usuarios, veremos que se asignan en función de los grupos definidos.



De nuevo nos resultará útil una agrupación, ya que los permisos no se asignan a usuarios individuales. Por otro lado, el uso de grupos, al igual que en un Directorio Activo o cualquier otra aplicación resulta en una gestión más sencilla.

Figure 27 – Ejemplo de asignación de permisos a grupos de usuarios

3.2.4 Templates

Los templates nos van a permitir definir item, trigger, y *graph* de forma conjunta para todos los hosts que estén vinculados (linked) a ella. Es una forma fácil de desplegar objetos para hosts nuevos. De hecho me parece mucho más potente que todo eso, pues los *templates* se pueden exportar/importar de forma sencilla en XML, de manera que la comunidad puede compartir *templates* ya creadas para un sistema operativo concreto (de hecho he importado uno de AS400 basado en ítems SNMP) o un equipo determinado de un fabricante (switch, Router, etc.). en el caso de los equipos de comunicaciones puede ser muy útil. (Pensemos en la definición de n ítems para 48 interfaces de un switch, por ejemplo). Podemos exportar el *template* con una boca definida, duplicar la entrada en el fichero con el resto de interfaces e importar el *template* de nuevo con la información añadida.

En nuestro caso se ha creado una plantilla básica de *monitoring* para el entorno Windows que se despliega de forma automática para los *Hosts* descubiertos (más adelante veremos cómo, en el apartado de *Discovery*).

Description	Triggers	Key
Disk space on e: Free		vfs.fs.size[e:,free]
Disk space on c: Free		vfs.fs.size[c:,free]
Disk space on e: Free in %	Triggers (1)	vfs.fs.size[e:,:pfree]
Disk space on d: Free in %	Triggers (1)	vfs.fs.size[d:,:pfree]
Disk space on c: Free in %	Triggers (1)	vfs.fs.size[c:,:pfree]
Disk space on c: Total		vfs.fs.size[c:,total]
Host information		system.uname
Host status		status
Host status	Triggers (1)	icmping[]
Host uptime (in sec)	Triggers (1)	system.uptime
Version of zabbix_agent(d) running		agent.version

Contiene los siguientes ítems (Véase *Figure 28 - Items básicos definidos para Windows Servers*).

Figure 28 - Items básicos definidos para Windows Servers

Aquí surge un problema. Se están asignando ítems sobre la unidad E:\, por ejemplo, y es posible que este servidor no tenga esta unidad definida. En este caso, con el fin de no cargar el sistema con intentos vanos de monitorizar este objeto, deberíamos ir al Host y desactivar este *item*. Se presenta la disyuntiva si crear *templates* específicos para un único objeto, para no tener que hacer modificaciones a posteriori en el Host (creo que pierde un poco el objetivo de un *template*) o como se ha mostrado anteriormente, con una acción a posteriori de desactivar el *item*. Es importante adaptarse a la estructura de la aplicación al dibujar nuestro escenario con el fin de sacar el mayor partido y crear una estructura fácil de administrar y eficaz.

No obstante creo que en la siguiente *minor release* 1.8.4 o en la versión 1.9 estos descubrimientos de ítems en los equipos, se podrán realizar de forma automática por el proceso de *Discovering*. (Actualmente está instalada en nuestro entorno la versión 1.8.3).

3.2.5 Triggers

Los *triggers* son condiciones que disparan una alerta en el sistema ante el estado de un cierto *item*. En este ejemplo se controla que el espacio libre no baje por debajo del 10%.

Template - Status:Disk Space below 10% on {HOST} | {apbpc2.colomerspain.local:vfs.fs.size[c:,:pfree].last(0)}<10

Figure 29 – Ejemplo trigger “espacio por debajo del 10%”

Como se aprecia en la imagen de la izquierda (línea superior de la ventana), se ha definido en un *template* y no en un *Host*, de manera que el *deployment* es más rápido y se reutilizan objetos ya creados en nuevos *Hosts*. Este *trigger* nos servirá, al igual que pasó con los *items*, para todos los *hosts* vinculados a este *template*.

3.2.6 Actions

De momento ha sido suficiente la configuración del correo electrónico para el envío de alertas. Estas acciones se activan con diferentes disparadores. En algunos casos por el resultado de un *discovering*, y en otros porque haya saltado un *trigger*. Hay una tercera opción referente al auto-registro, pero no la he estudiado aún. Tanto en uno como en otro, la definición ha de incluir la condición de inicio de la *action*, esto es con qué *trigger* está vinculada la *action*, o cualquier otra condición, de un amplio abanico de posibilidades, como severidad del *trigger*, parámetros de tiempo, pertenencia a un grupo, etc. Por otro lado se especifican las acciones que se llevarán a cabo si el resultado de las operaciones lógicas de condición resulta favorable.

Por otro lado se va a crear un script que permitirá el envío de notificaciones SMS a dispositivos móviles de telefonía. La parte complicada de envío de SMS ya se encuentra implementada en un Servidor AS400. Este equipo responde a peticiones por FTP con el *quote remote command* en el que se le pasan como parámetros el destinatario y el texto a enviar en el SMS. El script aún no se encuentra implementado, pero básicamente se reescribirá el código en shell script de unas bat's de Windows ya implementadas en productivo.

En cuanto al *discovering* se han creado *actions* que mueven el host descubierto al grupo adecuado y al mismo tiempo le asignan el/los *templates/s* adecuados. (*Véase Discovery*).

3.2.7 Discovery

Como se ha comentado en el apartado dedicado a los *templates*, estos se pueden aplicar automáticamente si se detecta un equipo durante el proceso de descubrimiento que tenga el agente instalado.

En este caso se escanea un segmento de red, y se *vinculan* los *Hosts* encontrados que tengan el agente instalado, con el *Template – Status* (*template* básico comentado anteriormente).

Posteriormente se crea una *action* de tipo ‘*discovery*’ con las acciones a llevar a cabo al descubrir un nuevo Host con el agente instalado.

Name	Conditions	Operations
Add windows Servers	Received value like "Windows" Uptime/Downtime >= "3600" Discovery status = "Up" Service type = "Zabbix agent"	Link to template "Template - Status" Add to group "Windows servers"

Figure 30 - Action de tipo Discovery (ejemplo)

En este caso añadimos el Host al grupo ‘Windows Servers’ y *linkamos* (vinculamos) el objeto con el *template* ‘Template - Status’.

Nota: Existe una tercera opción en las actions aparte de trigger y discovery), está implementada a partir de la versión 1.8.x. Al parecer comienza a monitorizar un host de forma automática. Sin embargo no estoy seguro si requiere el agente, y de momento no deseo que las WS de usuario se agreguen a la BDD. Investigaré esta opción un poco más a fondo si hay tiempo.

3.2.8 Maps

Vamos a partir de dos estructuras de mapas. Por un lado mapas **geográficos** que abarcarán desde el estado Worldwide de las sedes y las líneas WAN entre los diferentes centros, y por otro un mapa lógico de **servicios** de cara a la monitorización de HelpDesk o del negocio.

3.2.8.1 Geográficos

Se ha creado un mapa mundial en el que están reflejados los principales centros o sitres, tanto productivos como de oficinas. En ellos se pretende reflejar el estado de los centros (un problema en el centro hace que el icono aparezca en rojo y remarcado), y el estado de las líneas de comunicación. Aún estoy pensando como reflejar los enlaces múltiples a nivel gráfico para que no se solapen las líneas (la solución inmediata es generar dos iconos por *site*, pero no me convence). Véase Figure 21 – Línea OK (línea verde), seguramente tiraremos por acá...

Los objetos ‘site’ son de tipo mapa, por tanto al clickar en ellos, ampliamos la zona. Así haciendo click en ‘Europe’ visualizamos el mapa de Europa. En ‘Spain’ de nuevo, mostrará el mapa de Iberia



Figure 31 - Zoom al mapa de Europa



Figure 32 - Zoom al mapa de Iberia, tras clicar en Iberia



Figure 33 - Zoom Centro de Coornella, tras clicar en Cornellà HQ

Por último el icono del centro mostraría Cornellà HQ, en este caso. Se ha utilizado una foto retocada del centro en cuestión para colocar sobre ella los servidores y equipos de comunicaciones. En la implementación final tengo dudas si representar los Host imitando de alguna forma la topología de red o simplemente enumerarlos con su estado correspondiente. La ventaja del dibujo actual, es poder mostrar en el mismo mapa el estado de las comunicaciones de esos Host con el resto de la LAN.

Se crearán en principio mapas diferentes para mostrar en pantalla grande en la pared, y los que dispondrá HD y los administradores de sistemas. Estos últimos mapas serán más técnicos y de más bajo nivel.

3.2.8.2 Servicios

Este mapa, en principio mostraría el nombre de los servicios principales y su estado. No es excepcional que un *Host* contenga más de un servicio, por lo que si tenemos DNS y DHCP Server en un mismo Host, este mapa contendría dos entradas diferenciadas para ellas. Se pretende que de un vistazo, HelpDesk disponga de suficiente información para comunicar al negocio de forma rápida y adecuada si se produce una incidencia masiva de corte en un servicio.

Se está creando un icono transparente en el que se indica el **SERVICIO**, **SERVIDOR** y **STATUS** de cada servicio. Se ha creado un icono diferente en función del estado del servicio (Véase Figure 34 – Ejemplo Servicio OK).



Figure 34 – Ejemplo Servicio OK



Figure 35 - Ejemplo Servicio erróneo

Estos iconos aparecerán en un mapa con el fondo de la sede. En el caso de arriba, el icono está asociado un *trigger* que cambiará entre los dos iconos según el estado del servicio(OK o ERROR).



Figure 36 - Mapa de servicios. Cada icono es de tipo mapa.

El concepto de monitorización de servicios será el siguiente. El hecho de que el icono 'correo electrónico' este OK, indica que no hay problemas con ningún servicio o servidor de correo. Si lo hay el icono cambiará por el de servicio erróneo. Al tratarse de objeto mapa,

pulsando encima del icono 'correo electrónico' accederemos a un mapa de objetos 'host' con cada uno de los servidores de correo Domino , los de BlackBerry, correo web, etc. Por tanto la monitorización es en dos niveles, uno que indica el estado del servicio general (Correo electrónico), y uno más profundo que indica individualmente, el estado de cada uno de ellos.

3.2.9 Graphs

Del tema de gráficos, tan sólo he realizado algunas pruebas, puedo decir que son funcionales y muestran la información de forma correcta.

Se ha creado uno de prueba y seguramente no ahondaré más de momento en el tema.

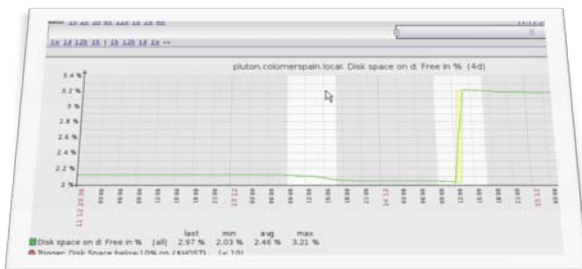


Figure 37 - Ejemplo de gráfico consumo de disco

Desde *Monitoring – Overview* mostraría la información como se aprecia en la imagen de la derecha. De hecho ya me ha sido útil en dos ocasiones para identificar procesos que se estaban comiendo el espacio disco en la máquina. El *scroll* de la parte superior permite de

manera fácil e intuitiva, la modificación del rango de visualización en el tiempo, se actualiza bien de

forma automática. De esta manera resulta fácil hacer zoom, ante pendientes pronunciadas en el gráfico y ver la hora exacta en la que se produjo el evento. A partir de aquí realizar el trabajo forense pertinente para identificar el proceso o usuario que lo ha producido.

3.2.10 Notificaciones

Se han configurado los *media types* de email de dos maneras diferentes. Posteriormente se decidirá la que se deja en productivo. En una de ellas se utiliza el motor SMTP propio del SUSE, y en la otra

Description	envío de correo
Type	Email (combo)
SMTP Server	localhost
SMTP HELO	ourEmailDomain.com

se dirige la solicitud de envío a un servidor SMTP corporativo. El envío local requiere la configuración de la izquierda. El SMTP email contiene la información de la cuenta '*sender*'.

Quedará pendiente pues la notificación por SMS, que dado que ya la tenemos implementada mediante el protocolo FTP y el comando 'remote execute command', no debería crear dificultades.

(Véase Alcance).

Capítulo 4 Estudio Fase2

En estas líneas se hace un estudio de las modificaciones e implementaciones a realizar una vez concluida la fase 1 del proyecto. En principio la Fase 1 se encargaba de realizar las pruebas de viabilidad del proyecto y se centraba en trabajar con unos pocos servidores, con el fin de asegurar que el software encajaba en las necesidades de la organización. En esta fase 2 se analizaba y se valoraba en recursos el despliegue al resto de servidores y la implementación de mejoras. Pues bien, el despliegue del cliente en los servidores locales de HQ se ha completado casi del todo durante la Fase 1. Se había barajado la creación de una instalación del cliente Zabbix por política de AD de Microsoft. Sin embargo dada la sencillez de la instalación y la poca configuración necesaria, (*Véase Instalación básica agente Windows.*), casi todos los servers tienen ya el cliente instalado y operativo con la configuración básica del Template-Status. No obstante se trata de una mejora a implementar para futuros *upgrades*. La migración del cliente de cada uno de los servidores y equipos sería bueno tenerla automatizada.

4.1 Alcance

4.1.1 Checklist diario

La primera acción a llevar a cabo será la sustitución del **checklist** diario llevado a cabo por el equipo de HelpDesk. Tras la fase 1 se han implementado monitorizaciones que sustituyen las comprobaciones que se realizan hasta la fecha con el checklist manual. La parte que tenía menos clara de todas ha sido la monitorización de las copias de seguridad diarias. La heterogeneidad de los resultados hacían difícil una solución estándar al problema. Finalmente se ha optado por informar desde el aplicativo de copias de seguridad a un archivo plano en caso de producirse algún fallo, y desde Zabbix leer este archivo una vez al día. La lógica que se ha seguido es la siguiente, si existe algún problema el operador acude a la consola de gestión para solucionarlo. En dicha consola se puede consultar toda la información relativa a las copias, por lo que no tenía sentido montar un sistema paralelo de *monitoring* y *reporting* de esta aplicación, pues ya incluye uno, y además está parametrizado correctamente. (*Véase Otros*)

4.1.2 Formación HelpDesk

Con el fin de que puedan llevar a cabo su labor, se formará a primer nivel en el uso de la aplicación. Para ello se creará una cuenta con permisos de lectura sobre todos los objetos definidos. Esta cuenta se utilizará durante la formación. Hay por tanto un trabajo previo de preparar la cuenta de usuario y

crear las pantallas (*screens*) y dispositivas (*slides*) necesarios. Asimismo se creará una pequeña presentación, al margen de la entrega para el proyecto con los conceptos básicos y cómo navegar por el aplicativo. Se instalará un acceso directo en el escritorio de los PC's de HelpDesk que abra la aplicación con el fin de que se monitorice de forma continua la plataforma. Las alarmas acústicas se dejarán encendidas. Durante la Fase I, he tenido la alarma acústica activa en mi laptop, y he podido comprobar su utilidad. El sonido no es estridente, ni molesto, y el aviso es inmediato ante cualquier problema. Tomando estas acciones como punto de partida, queda un trabajo continuo con los responsables de HelpDesk para definir las acciones a tomar en caso de cada tipo de alarma, los grupos a los que hay que dirigir las notificaciones en función del tipo y gravedad del asunto. Por otro lado existirá una formación continua en paralelo, con el fin de que las incidencias que se escalen a nivel 2 contengan toda la información posible. De ser necesario se incluirán objetos en el aplicativo conforme vayan surgiendo necesidades.

4.1.3 Backup del aplicativo

Hasta ahora se ha instalado la plataforma de Zabbix y se han ido definiendo objetos en la misma. Asimismo se ha ido recogiendo información con la que la aplicación es capaz de ofrecer estadísticas e informes. Por todo ello se hace imprescindible, y en cierta manera debería preceder a los dos puntos anteriores, la definición e implementación de un procedimiento de copia de seguridad del entorno Zabbix. En la parte cliente, en la mayoría de casos no serán necesarias labores de copia. No obstante se debería guardar y documentar los casos específicos de servidores con capturas de datos (o envíos con `zabbix_sender.exe`) fuera del alcance del cliente nativo. La idea es poder recuperar la monitorización de esos servidores en caso de fallo. El resto como se ha visto anteriormente, sólo precisan indicar el host de Zabbix y el nombre del server sujeto a monitorización.

En la parte server de Zabbix el trabajo es diferente. Por un lado sería deseable conservar copia del sistema operativo tal y como se encuentra actualmente, ya que se ha configurado la red, servicios y clientes para adaptarlos a la LAN de la organización. Esta copia será periódica pero puede estar espaciada en el tiempo. Sin embargo la copia de BDD de **MySQL** que da soporte a Zabbix, sí que se debería de copiar de forma diaria para asegurarse de no perder los cambios realizados. Pensemos que en la BDD no sólo se encuentra la definición de los objetos (hosts, mapas, triggers, *actions*, etc.), en caso de fallo o corrupción de la BDD, nos arriesgamos a perder las imágenes cargadas y la

información estadística de los objetos obtenida hasta la fecha. Esto último no evitaría el funcionamiento del aplicativo, pero es información que puede resultar útil a medio plazo, con el fin de saber el nivel de servicio que se está ofreciendo a la organización. Por otro lado puede servir para identificar los puntos más conflictivos, y en cierta medida puede servir de soporte para el establecimiento de los SLA's. Para no extenderme más con este apartado, pues puede alargarse, cito dos URL con información detallada de los procesos de copia, aparte de la incluida en el MAnual de Zabbix.

*<http://www.zabbix.com/forum/showthread.php?t=19495&highlight=backup> y **Zmanda backup for mysql dbb** en <http://www.zmanda.com/backup-mysql.html>*

Una vez se disponga del punto anterior se creará el documento de recuperación de **contingencias** y DR de Zabbix y se incluirá en el documento corporativo de **DR**.

4.1.4 Monitorización completa de servidores

Ya se ha comentado que durante la Fase I se ha cubierto HQ casi completamente. Pues bien, ahora toca terminar de desplegar el aplicativo en el resto de servidores de HQ y las sites remotas. En primera instancia, sólo para Iberia. Más adelante se estudiará de dar cobertura al resto de países. Esto cubre el despliegue del cliente básico de servidor en Iberia. Ahora bien, sería deseable disponer de items específicos para cada uno de los servicios, SMTP, SQL, FTP, Backup, etc. Asimismo se crearán los objetos necesarios a todos los niveles (hosts, alertas, mapas, *scenes*, etc.) que permitan el control de los servicios desde HQ y por otro lado facilitar información de primera mano a los **key users** de las sites remotas (*véase Despliegue Worldwide, servidores Zabbix Proxy*), con el fin de asignarles cierta autonomía a este respecto. Obviamente se establecerán personas a las que avisar en caso de que se produzca un corte en alguno de los servicios, al igual que se realizó con HD Management en apartados anteriores. *Véase [Formación HelpDesk](#).*

4.1.5 Monitorización completa de la red

Al igual que ocurría con los servidores en el capítulo anterior, la comprobación de las comunicaciones se extenderá al resto de Iberia. En este caso sí que se incluirán las redes de otros países, no las LAN, pero si las conexiones WAN entre ellas. La idea es detectar problemas en las comunicaciones lo antes posible, y a ser posible, ser nosotros quienes avisemos a los usuarios remotos del problema existente. Se trata de transmitir una imagen de serenidad y control de los

sistemas y las comunicaciones. La organización ha crecido según necesidades del mercado, algunas de las veces de forma un poco caótica en cuanto a la disposición y gestión de los equipos y servicios, por lo que una imagen de seguridad y control a este respecto no vendría mal. Esta parte es la más delicada en cuanto que se buscará encontrarse con el mínimo número de falsos positivos posibles. Dado que existen varios enlaces redundados hasta tres veces, el trabajo deberá ser meticuloso para no dejarse opciones en el camino.

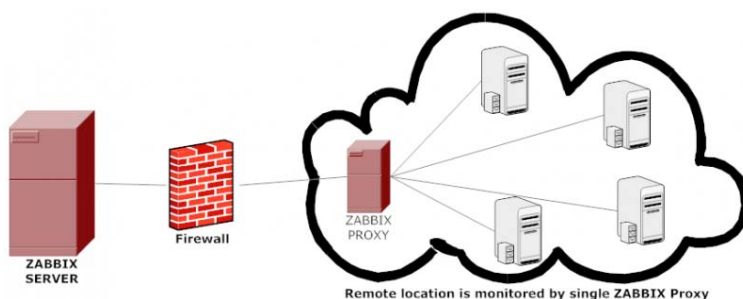
4.1.6 Avisos por SMS

Este apartado aunque aún no implementado, está suficientemente maduro para no dar demasiados quebraderos de cabeza. Se utilizará un dispositivo de envío de SMS conectado a un server **AS-400**. Actualmente ya se encuentra dando servicio, de la siguiente manera: Se realiza una conexión **FTP** al servidor en cuestión desde cualquier cliente. Se procede a la ejecución remota mediante **quote rcmd** del envío de SMS. Los parámetros de este comando incluyen el texto y el destinatario móvil del mensaje. No obstante y en previsión del desmantelamiento de este server se va a estudiar conectar al SUSE Server de Zabbix un dispositivo nuevo de estas características por el puerto serie, con el fin de que autónomo en este respecto. De hacerse, los envíos se podrían hacer directamente desde Zabbix, dado que dispone de la definición de este tipo de Media de forma nativa.

Existen otros servicios de Internet que ofrecen servicios similares (Jabber), se estudiarán también, aunque es de suponer que el sistema principal de envío se encuentre en nuestras instalaciones.

4.1.7 Despliegue Worldwide, servidores Zabbix Proxy

De cara a la cobertura mundial de la plataforma; Zabbix dispone de la posibilidad de usar lo que



llaman servidores proxy. Con una configuración de este tipo, existe un servidor central, en este caso se trataría del server actual, y servidores satélite distribuidos por las diferentes sites, los cuales se encargarían de recoger la

información local referente al rendimiento y disponibilidad de servicios y remitirla de forma periódica al server central.

Figure 38 - Proxy server schema con Zabbix- Image from Zabbix on-line doc v1.8

El uso de los nodos aporta varias ventajas. Dispone de una instalación de BDD automatizada y de fácil mantenimiento. La gestión se hace de forma centralizada. Por contra, os nodos no disponen de consola gráfica, aunque como la gestión es centralizada tampoco se hace necesario. Tampoco dispone de notificaciones, por lo que en todo caso hay que esperar que la información llegue al nodo central, y este envíe las notificaciones especificadas.

A la hora de definir un host, podemos seleccionar si su monitorización se va a realizar a través de un proxy server Zabbix o desde el propio nodo principal. En caso de ser a través de un proxy, la información se almacena localmente y según la planificación se envía al nodo central para su análisis.

4.1.8 Solución de upgrade de aplicativo, servidor y cliente

Con el fin de facilitar la renovación tanto de la parte servidor como de la parte cliente de Zabbix se diseñará y documentará un procedimiento de upgrade de ambos entornos. En cuanto al **cliente** la idea es disponer de un *paquete de instalación* de MS en el AD corporativo. Dicho software hará una copia del fichero de configuración del cliente Zabbix, desinstalará el antiguo e instalará el cliente nuevo. Una vez completado el proceso sustituirá el fichero de configuración con la copia realizada y reiniciará el servicio del agente en el cliente. El lanzamiento de esta renovación se hará por **política de grupo** de AD, deseablemente sin necesidad de una actuación manual por parte del administrador. En la parte **servidor** se deberá crear el procedimiento en paralelo con el de DR (*véase Backup del aplicativo*). Siempre incluirá un backup previo del sistema y de la BDD con el fin de tener **rollback** en caso de problemas. Por otro lado se recomienda el uso de *snapshot* de VMWare dado que el servidor está virtualizado bajo esta plataforma. Supone un punto adicional de vuelta atrás.

4.2 Recursos e inversión

Se ha conseguido de momento, presupuesto para la compra de una pantalla de 50 pulgadas con el fin de tener en el departamento de IT una visual directa con el estado de los servicios y las comunicaciones. Se conectará un equipo antiguo, reciclado a esta pantalla. No deberíamos tener problemas puesto que los requerimientos son un simple navegador (Internet Explorer, Firefox, etc.), sin embargo si que será necesario una tarjeta gráfica que permite resoluciones altas del orden de 1900x1080, con el fin de plasmar correctamente los mapas voluminosos.

En cuanto a material humano, desgraciadamente, los recursos a emplear no van a ser externos casi con seguridad. De todas maneras el análisis de recursos e inversión se ha realizado igualmente, teniendo en cuenta las figuras de **technical consultant** y **technical support**.

Inversión			
Technical support	25€/hora		
Technical consultant	60€/hora		
	Horas	Recursos	Estimación
Completar checklist diario	8	Technical consultant	480,00 €
Preparación formación(screens,doc, etc.)	5	Technical consultant	300,00 €
Formación HelpDesk	4	Technical consultant Technical support x5	740,00 €
Formación continuada	indeterminadas	Technical consultant Technical support	
Backup aplicativo	20	Technical consultant	1.200,00 €
Monitorización Servidores	20	Technical consultant	1.200,00 €
Monitorización red	40	Technical consultant	2.400,00 €
Avisos SMS (solucion actual)	5	Technical consultant	300,00 €
Avisos por SMS (solución final)	30	Technical consultant	1.800,00 €
Despliegue Worldwide	indeterminadas	Technical consultant	
Procedimiento de upgrade	15	Technical consultant	900,00 €
		TOTAL	9320

Se comprará e instalará un equipo de envío de SMS por el puerto serie, conectado al servidor VMWare ESX. Se ha de pasar el dispositivo directamente a la máquina virtual para que se pueda utilizar y posteriormente configurar el Linux para su uso, añadiendo los drivers necesarios. Por tanto la inversión es escasa, ya que sólo incluirá la compra del dispositivo. Esto hace que se pueda realizar en cualquier momento contra el centro de gastos del departamento. En el cuadro anterior no se han incluido los gastos de compra, tan sólo los de horas, aunque como ya he comentado la segunda fase se realizará con recursos internos de la organización.

Capítulo 5 Pruebas realizadas

Mientras se ha ido configurando el aplicativo se han ido realizando numerosas pruebas para comprobar el funcionamiento del mismo. Muchas de ellas se encuentran documentadas en capítulos anteriores, de paso que se explicaba su funcionamiento. En general y a modo de resumen, se enumeran a continuación las pruebas realizadas:

Mapas

- Creación de mapas personalizados. Se han construido jerárquicamente, desde el mapa global Worldwide hasta llegar a una *site* concreta y ver los servicios disponibles en ella. Por en medio nos quedan los mapas de Europa e Iberia.
- Los mapas de servicios también son jerárquicos, teniendo por ejemplo, un mapa global con el estado de SMTP, y accesible mediante clic a este icono, una nueva pantalla (mapa) con el detalle de los diferentes servidores SMTP y su estado. Una alerta en el primero informaría de que al menos un servicio SMTP está con problemas. Entrando en el segundo mapa, podríamos ver cuáles son estos servicios.

Web Monitoring

- Se han creado monitorizaciones específicas de publicaciones webs internas. Se ha comprobado que parando el servicio IIS en los servidores monitorizados, o la publicación web específica, el sistema reporta error, aunque el servidor esté encendido y no reporte fallo de la comprobación ping.

Items

- Ejecución de comprobaciones propias con UserParameter ('*addon*' sobre el agente estándar aportado por Zabbix). Se han creado *items* personalizados y se ha comprobado que el contenido llega al nodo principal.
- Comprobación de rutas WAN entre sites. Se ha comprobado que durante los lapsos en los que hemos tenido problemas en la línea (no se ha provocado manualmente), el sistema reporta la línea discontinua en rojo en el mapa, entre Iberia y USA y el *trigger/action* alerta del fallo. Tras recuperarse el servicio la línea vuelve a su color verde inicial y línea continua en cuanto se soluciona el problema.

- Comprobación de servicio SQL. Se ha añadido la monitorización estándar de SQL, no hay mucho que comprobar, cuando he parado el servicio de MS SQL, la alarma ha saltado.
- Comprobación de espacio libre en disco. Esta comprobación nos está resultando útil, pues no sólo nos permite conocer el estado de ocupación en real, sino que se guarda un histórico también, esto nos ha permitido identificar (hasta el momento), procesos que estaban consumiendo espacio desmesuradamente. Está resultando realmente útil.
- Test de Zabbix traper. Se ha probado el envío de información al nodo Zabbix desde servidores con el agente instalado. Se ha realizado con el comando **zabbix_sender.exe**. Se trata de monitorización pasiva, desde el punto de vista de Zabbix.
- Leer logs en hosts en busca de cadenas de texto. Este tipo de monitorización ha permitido, por ejemplo, la comprobación diaria de los backups que estábamos realizando. *Véase Otros*
- Estado de servicios SMTP. Se han añadido comprobaciones de SMTP mediante chequeo del puerto 25 TCP. Se puede hacer mediante una *key* estándar, pero dado que disponemos de servicios de correo en puertos TCP no estándar (25), se ha optado por este método.

Notificaciones

- El envío de correo se ha comprobado en numerosas ocasiones, de hecho estoy recibiendo en una cuenta externa y en paralelo en la cuenta interna de la organización, los avisos y alertas que se producen en el sistema.
- Asimismo se ha montado el envío de un email a una cuenta específica de Lotus Notes ante algunas notificaciones. Este envío provoca que se cree de forma automática un ticket en la herramienta de gestión de incidencias.

Frontend / Visualización

- Con la opción `FullScreen=1` en la URL que se le pasa al navegador como parámetro, se obtiene una presentación del aplicativo sin los menús superiores, si sumamos F11 en el caso de Firefox, nos permite ver el entorno de monitorización a pantalla completa totalmente. Realmente útil esta prueba, pues uno de los objetivos marcados para la Fase II es la instalación de una pantalla que mostrará mapas, para ello será fundamental disponer de la pantalla completa. (se verá en el video explicativo).

- Se ha personalizado el logo corporativo y se ha recopilado información de cómo customizar el producto con *themes* personalizados. Sin embargo no se ha hecho nada al respecto, por el momento el entorno actual es aceptable, y el tiempo se invertirá en otras necesidades.
- Es posible que en un futuro se habilite un acceso web a los clientes (usuarios finales de IT) en modo lectura y con una visión a muy alto nivel del estado de los sistemas, con el fin de reducir llamadas a HD. Si el servicio ya está marcado como 'con problemas', se espera se reduzca en número de llamadas. Ante un problema masivo se han de recibir y atender numerosas llamadas desde HD en relación al mismo problema. De esta manera el usuario final podrá comprobar el estado del servicio siempre que quiera, incluso se pueden poner a su disposición estadísticas de servicio.

Capítulo 6 Conclusiones

6.1 Mejoras

En este apartado no se incluyen las nueva implementaciones y mejoras comentadas en el alcance de la Fase II. *Véase Alcance..* Soy consciente de algunas mejoras que se pueden ir aplicando sobre lo aquí presentado, pero dado el tiempo disponible se implementarán más adelante. El script `z_trace` debería estar basado en 'case' y no en condicionales 'if-fi' y la verdad es que se podría mejorar (he buscado la funcionalidad inmediata). En el caso de los mapas, de momento la única línea que cambia de color en caso de avería (`z_trace` devuelve 'error') es la que une Spain y USA del mapa Worldwide. Se trata en todo caso de comprobar las posibilidades del aplicativo, por lo que en principio no se incluirán más de momento en la Fase I. De la misma manera el número de servidores actuales monitorizados tampoco es el final, el *scope* total supera mis posibilidades en tan corto plazo. *Véase Alcance.*

El script documentado para el `z_trace` (*Véase Comprobación de red*) pese a que reconozco que no es complejo, me ha costado más tiempo del previsto, no he desarrollado mucho en shell scripting y lo que he hecho lo tenía bastante oxidado. De todas maneras es funcional y cubre de momento las necesidades que han aparecido. No obstante, y sobre todo si se va a desplegar a nivel Worldwide, me gustaría reescribirlo con las mejoras comentadas y tenerlo mejor comentado.

Como se ha puntualizado anteriormente, una mejora inmediata podría ser medir periódicamente el ancho de banda consumido y avisar si se mantiene un consumo equivalente el CIR por más de 10 minutos.

Por otro lado, creo que este tipo de implementaciones tienen que estar vivas, con el fin de adaptarse a los nuevos requerimientos y necesidades de la organización, de manera que se genere un ciclo continuo de detectar necesidades, implementar soluciones a esas necesidades y formar a Help Desk de las modificaciones. A su vez Help Desk, más cercano al usuario final, y con una visión global de los problemas que se producen, puede detectar nuevas necesidades y comunicarlas al administrador con lo que el ciclo se inicia de nuevo. (*Véase Presentación del TFC en pps, actualización constante en Estudio Fase II*).



6.2 Conclusiones

Francamente, en el inicio del proyecto, veía el trabajo como una caja negra y no veía por dónde empezarlo. En parte mis dudas venían porque conocía muy someramente los diferentes aplicativos (algún pinito con

Nagios), y sabía que una mala elección podía suponer una pérdida de tiempo valiosísima pues el semestre (cuatrimestre) es corto y se dispone de muy poco tiempo. Inicialmente opté por Opsview por ser un Nagios *alike*, y ser éste el de más presencia en la Red. Sin embargo un primer acercamiento me mostró que la configuración pese a no ser complicada, si que la encontraba tediosa y problemática para un despliegue masivo. Soy consciente que para un administrador de entornos Linux curtido, la consecución de scripts para cargas masivas de hosts o duplicar *triggers* puede resultar una tarea sencilla. Sin embargo para mi poca experiencia en el tema y el poco tiempo disponible, lo consideré una carga extra de stress que podía tirar abajo el proyecto.

Así que finalmente decidí emplear una semana en probar el aplicativo Zabbix y tomar una determinación. Afortunadamente elegí Zabbix, y como ya he comentado en varias ocasiones a lo largo del documento, estoy muy contento de haberlo hecho pues ha cumplido ampliamente mis expectativas.

He tendido a dispersarme, pues la verdad es que he encontrado temas interesantísimos desde un punto de vista técnico y de desarrollo durante el transcurso de buscar información en el Red. Por ello he tenido que ser auto-disciplinado y estructurar el trabajo desde un inicio (mi forma de trabajo es bastante más caótica de por sí). Me ha sido de gran ayuda ponerme la limitación de las necesidades y mejoras a aplicar en mi entorno de trabajo, pues me ha mantenido con los pies en el suelo y no he divagado ni perdido *demasiado* tiempo, leyendo implementaciones interesantes en Internet pero de poca utilidad real o no aplicable en mi entorno. He procurado en todo momento, limitarme a buscar la información que necesitaba para cubrir una necesidad.

Durante la PAC3 del proyecto me he metido a fondo en el aplicativo. La verdad es que está superando con creces mis expectativas. Como ya he comentado hace unos años estuve ‘tonteando’ con Nagios, pero a diferencia de éste, Zabbix me ha parecido muy intuitivo, la jerarquía de objetos más manejable y por supuesto mucho más ‘friendly’ el interfaz gráfico desarrollado en php, que aporta de forma nativa el producto. Aparte de la sencillez (aquí no puedo comparar, pues no llegué tan a fondo con Nagios), Zabbix me está aportando soluciones en prácticamente todos los objetivos marcados. En otros, la generación de scripts auxiliares aporta soluciones a las (yo no diría carencias del producto) particularidades de la infraestructura de la organización.

En la recta final de la entrega (al leer el documento completo), me ha sorprendido desagradablemente que el documento final carecía un poco de hilo argumental en algunos apartados. Seguramente por haberse

realizado sin una dedicación plena y por incluir modificaciones varias por mi parte, sin una lectura completa del capítulo. Por ello se han modificado algunos de los apartados antes de la entrega final, eliminando frases excesivamente largas y buscando por otro lado una homogeneidad en el documento. Espero que, aunque a última hora, haya logrado darle una unidad al escrito.

Conforme he profundizado en la aplicación han ido surgiendo ideas continuamente. Al inicio del proyecto estaba un poco abrumado por no saber con certeza qué iba a incluir en el proyecto, pero tengo que decir que las ideas han ido saliendo de forma fluida. Me ha ayudado el hecho de tener un campo de trabajo con problemas reales y bien conocidos, en la medida que he ido creando objetos que cubrieran las necesidades de la organización.

La idea era presentar a la dirección el proyecto una vez concluido y la valoración de recursos realizada. Pese a ello, por temas coyunturales he tenido que adelantar la presentación del proyecto y tengo que decir que ha sido aprobado (aunque con escasos recursos, no está el horno para bollos). Para ser sincero creo que el producto se vende por sí solo, pero también pienso que ha ayudado el hecho de poder mostrar **soluciones concretas** a problemas conocidos dentro del departamento, y sobre todo el comentar la foto final esperada del proyecto. La experiencia me dicta ser cauto con *software mágicos*, pues estos no existen, y habitualmente tras una presentación abrumadora de posibilidades y 'buenas intenciones' por parte del proveedor, la cruda realidad es que cuando profundizas, las cosas son bastante más complicadas de lo que parecían, suelen surgir imposibilidades y las implantaciones suelen dilatarse en tiempo y recursos, tanto humanos como económicos. Sorprendentemente, con este producto me ha pasado al contrario, ha habido una etapa previa (obvia) de entender los conceptos base y el funcionamiento a *grosso modo* del aplicativo (tengo que decir que no podría haberlo llevado a cabo en este corto plazo sin la ayuda de los foros de la comunidad en Internet), pero tras esta, apenas han surgido complicaciones, y los resultados me han compensado con creces el esfuerzo invertido.

En general, y dejando de banda el stress que producen las fechas de entrega y los momentos de estancamiento, estoy muy contento con el trabajo realizado. Independientemente de la valoración, creo que en el TFC he adquirido un manejo más fluido del procesador de textos, así como depurar la técnica de presentación. He sido capaz de estructurar, planificar y llevar a la consecución el TFC en los tiempos estipulados, entregando todos los parciales. Por otro lado, y aun no siendo relevante académicamente, el proyecto se presentó y se aprobó en la organización donde estoy prestando servicio, pudiendo así sentir una

utilidad añadida al esfuerzo, incluso durante el trascurso del trabajo. Normalmente es un sentimiento que se tiene en la entrega final del trabajo, pero en mi caso me ha acompañado durante todo el semestre. Obviamente también supone una carga adicional de presión, el hecho de tener que avanzar el proyecto en dos líneas diferentes, una más orientada a resultados inmediatos (que den solución a problemas concretos) y la otra a una buena planificación, estructura del trabajo y una cuidada presentación en los informes. Insisto de nuevo en el apartado de las imágenes, pues creo sinceramente que las imágenes que he dejado hacen un énfasis positivo en la faceta gráfica de la herramienta de gestión (hecho diferencial con otras aplicaciones de este tipo) y por otro lado creo que aportan vistosidad al trabajo, no convirtiéndolo en una retahíla de conceptos específicos de la aplicación, que puede resultar mareante al cabo de un rato, sin las concreciones que supone ver la foto del 'objeto' (su representación).

Glosario

Action: Acciones a llevar a cabo por el nodo principal Zabbix en respuesta a un trigger o una acción de descubrimiento de host nuevo.

Appliance: Nombre común para dispositivos electrónicos con una función concreta y una configuración limitada.

Graph: Gráfico obtenido a partir de valores obtenidos por el aplicativo

Host: Objeto básico del que queremos obtener información, en el contexto del documento se refiere a servidores, WS, *switches*, *routers* o cualquier otro dispositivo que se encuentre en red y sea susceptible de instalar el agente o disponga de servidor SNMP

Host Group: Agrupación de hosts, creados para facilitar su gestión.

Item: Particularidad que se desea conocer de un host concreto.

Log: Archivo habitualmente en texto plano donde se dejan las trazas de un sistema operativo o de una aplicación concreta.

Map: Se trata de un tipo de objeto parametrizable, tanto el fondo como de los objetos que incluye que permite mostrar el estado de los *hosts*, *triggers*, o anidar otros mapas, principalmente.

Media type: Nombra los tipos de notificaciones que pueden llevarse a cabo desde la aplicación Zabbix, en concreto se dispone de avisos por email, SMS o aplicaciones de Internet de mensajería.

Monitorización activa: En este tipo de monitorización el nodo Zabbix realiza periódicamente consultas a los hosts observados en busca de la información configurada a tal efecto.

Monitorización pasiva: En este tipo de monitorización es el host quien hace un *push* de la información cuando dispone de ella o así se ha establecido en el fichero de configuración del agente. También se pueden hacer llamadas fuera del agente, desde el *prompt* de comandos.

SNMP: Acrónimo de *Simple Network Transfer Protocol*. Protocolo de Internet que permite la gestión de dispositivos en redes IP.

Template: Plantilla que incluye principalmente *items*, *triggers* y *graphs* y permite asignar ágilmente estos objetos a hosts o grupos de éstos.

Traps: Recogida de mensajes de *zabbix_sender* que se recogen desde el nodo Zabbix.

Trigger: Alerta que se dispara en condiciones determinadas de un *item/host*.

Bibliografía

Zabbix 1.8 Networking Monitoring ISBN : 184719768X, ISBN 13 : 978-1-847197-68-9

Author(s) : Rihards Olups

Zabbix Manual v1.6.pdf (electronic support) Version 017

Copyright 2008 ZABBIX SIA, REGISTERED IN LATVIA NO: LV40003738045

Fuentes electrónicas – Web

<http://www.gnu.org/licenses/gpl-faq.html#WhyUseGPL>

<http://www.gnu.org/licenses/license-list.html>

<http://www.zabbix.com/es/licence.php>

<http://www.opsview.com/learn/compare-opsview>

<http://www.nagios.org/>

<http://www.google.com/trends?q=nagios%2C+openview%2C+unicenter%2C+tivoli%2C+bmc&ctab=0&geo=all&date=all&sort=1>

<http://www.icinga.org/wp-content/uploads/2010/08/Icinga-vs-Nagios-matrix1.pdf>

http://en.wikipedia.org/wiki/HP_OpenView

http://www.google.es/imgres?imgurl=http://www.ramonmillan.com/imagenes/fotostutoriales/openview.gif&imgrefurl=http://www.ramonmillan.com/tutoriales/gestionred.php&usq=__tBXssrX95TnUCfSFFjS7cSyH0zQ=&h=759&w=922&sz=81&hl=es&start=0&zoom=1&tbnid=ImWjvUrgdh68JM:&tbnh=145&tbnw=176&prev=/images%3Fq%3Dhp%2Bopenview%26hl%3Des%26biw%3D1024%26bih%3D605%26gbv%3D2%26tbs%3Disch:1&itbs=1&iact=hc&vpx=737&vpy=78&dur=18&hovh=204&hovw=247&tx=130&ty=87&ei=tEneTJqTFsHqOdm-kMMO&oei=tEneTJqTFsHqOdm-kMMO&esq=1&page=1&ndsp=12&ved=1t:429,r:3,s:0

<http://www-01.ibm.com/software/tivoli/products/monitor/compare.html>

ftp://public.dhe.ibm.com/software/tivoli/products/Set_6_Final_TivoliMonitoring-ITCAM-ProductsComparison_-_Divyesh_input.pdf

<http://workaround.org/node/304>

<http://zabbix-es.blogspot.com/>

<http://www.zabbix.com/forum/showthread.php?t=8167&page=2>