Universidad Zaragoza

Trabajo Fin de Grado

Diseño y despliegue de una arquitectura y políticas de monitorización de sistemas

Design and deployment of a system monitoring architecture and policies

Autor

Beatriz Pérez Cancer

Director

David Roman Esteban

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Beatriz Pérez Cancer,

con nº de DNI 25206663C en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Diseño y despliegue de una arquitectura y políticas de monitorización de sistemas,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 30 de agosto de 2017

Fdo: Beatriz Pérez Cancer

Dedicado a mi padre que siempre ha confiado en mi,
a mi madre que me ha apoyado,
y a Alberto que me ha dado fuerzas en todo momento.

Diseño y despliegue de una arquitectura y políticas de monitorización de sistemas

RESUMEN

La empresa IA Soft, perteneciente al grupo OESIA, contaba con un sistema de monitorización que cubría aspectos de 10 de las sedes con las que cuenta el grupo. Debido a la división de la monitorización en dos herramientas, nace la necesidad de diseñar un nuevo sistema donde se pueda unir la monitorización de todo el sistema en una sola herramienta actual que facilite la escalabilidad y que cubra servidores y hosts, servicios que se alojan, servicios web, dispositivos de redes como switches, firewalls y routers y servicios ambientales y de corriente respectivos al centro de datos.

Este nuevo sistema va a aportar nuevas características que no presentaba el anterior, destacando la utilización de una única herramienta para facilitar el mantenimiento del sistema y la visualización de la información, la personalización de periodos de almacenamiento de información y la posibilidad de escalado del sistema conforme aumente la cantidad del elementos a monitorizar.

La realización de este proyecto se ha dividido en varias fases. La primera consiste en la realización de un estudio a cerca de los sistemas de monitorización y de las diversas herramientas que existen hoy en día para seleccionar la más adecuada en función de los requisitos.

La siguiente fase consiste en realizar el diseño de la arquitectura del sistema y las políticas, que se dividen en función de los tipos de elementos a monitorizar y las características que presentan.

Finalmente se ha realizado la implementación del sistema utilizando Zabbix y aplicando las políticas diseñadas, donde se instala el servidor y los agentes en los hosts necesarios o se monitorizan remotamente los dispositivos, realizando al mismo tiempo las pruebas que permiten comprobar que la información se recoge y se muestra de forma correcta para poder ser analizada y también se generan las alertas definidas cuando se producen fallos en los sistemas para facilitar una rápida actuación.

Design and deployment of a system monitoring architecture and policies

ABSTRACT

IA Soft company, which belongs to OESIA group, had a monitoring system that covered 10 of its offices. The necessity of designing a new system borns because of division of the actual system in two different tools and the fact that the actual system is outdated. The main objective is joining the monitoring system under one unique tool, making it easier to scale and cover all servers and hosts, hosted services, network devices and ambient services related with datacenter.

This system is going to add new features that the previous one did not have, standing out the use of a single tool to make system maintenance and the representation of information easier, customization of periods in which information is saved and the posibility of system scaling depending on quantity of items.

This project has been divided in several periods. First consists in the study of monitoring systems and various existing tools. After studying their advantages and disadvantages, Zabbix has been chosen.

Next phase consists in designing the monitoring system architecture and its policies, which are divided according to different kinds of items to monitorize in the system and their features.

Finally, deployment has been done using Zabbix and applying designed policies, where server and agents are installed in necessary hosts, or devices are remotely monitored. At the same time, tests have been done to verify if the information is properly gathered up and shown with the final purpose of analyzing it and also if defined alerts are triggered when a system failure is detected, allowing the fastest possible intervention.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

La monitorización de sistemas [1] es un aspecto muy importante en el ámbito tecnológico ya que permite vigilar y conocer el estado de los sistemas en cada momento, pudiendo detectar los problemas en tiempo real mediante las políticas de recogida de información y alertas definidas. Esto permite conocer la disponibilidad de los sistemas y si ocurre algún fallo, detectarlo antes que los usuarios y anticiparse para solucionarlos de forma manual o automática, gracias a los sistemas proactivos que ayudan a evitar la falta de disponibilidad de los sistemas.

El almacenamiento de información histórica de los sistemas también es una parte importante de la monitorización ya que permite visualizar y analizar cómo evolucionan los sistemas en el tiempo y los eventos que han ocurrido para poder detectar patrones de comportamiento en situaciones en las que se van a producir fallos permitiendo así la anticipación al fallo completo para poder solucionar los problemas con anterioridad, o modificar los sistemas añadiendo más recursos o eliminándolos en función de las necesidades.

1.1. Contexto y motivación

Este proyecto se ha llevado a cabo en la empresa IA Soft, perteneciente al grupo OESIA [2], que tiene presencia a nivel global y cuenta con 15 sedes corporativas repartidas entre España y América. El grupo OESIA trabaja en diferentes aspectos, destacando entre ellos administración pública, sanidad, banca y seguros, telecomunicaciones, seguridad y defensa, sector aeroespacial e industria. Además, entre estos sectores, la sede de IA Soft destaca en productos y servicios para administraciones públicas.

Mediante la implantación de la nueva arquitectura de monitorización se pretende reemplazar al sistema actual y se pretende cubrir 10 de las sedes pertenecientes al grupo donde se incluyen servidores y hosts, los servicios que estos alojan, servicios ambientales y de corriente respectivos al centro de datos y dispositivos de redes como routers, switches, firewalls y puntos de acceso WiFi. Este cambio se ve motivado principalmente por la dificultad de mantener dos sistemas de monitorización distintos que se encuentran separados, ya que esto supone tener que manejar dos herramientas diferentes y contar con personal que tenga conocimientos sobre ambas, por lo que supone un problema de costes tanto en tiempo como en dinero. Además, en estas herramientas se encuentra falta de personalización de los periodos durante los cuales se puede guardar la información y las tendencias, falta de personalización en la visualización de los gráficos, falta de elementos de monitorización para los que hay que crear scripts personalizados en cada host y se presentan problemas para el escalado de los sistemas en caso de que la cantidad de elementos a monitorizar continue creciendo.

1.2. Objetivos

El objetivo de este proyecto es diseñar un nuevo sistema de monitorización que sustituya al anterior presentando mejoras y solucionando los problemas presentados para ahorrar costes en tiempo y dinero. Por tanto, debe tratarse de un sistema único que monitorice todos los elementos que lo componen de forma estructurada controlando su disponibilidad de forma automática, generando las alertas y acciones definidas cuando se produzca algún fallo para facilitar una actuación rápida. También debe permitir personalizar los periodos de almacenamiento de información y tendencias y facilitar la escalabilidad para poder ir añadiendo todos los elementos según el crecimiento de los sistemas sin que se produzcan fallos en el sistema central de monitorización.

1.3. Estructura

La estructura de la memoria se basa en las fases en las que se ha dividido el desarrollo del proyecto, por tanto se ha dividido en las secciones de introducción y estado del arte donde se explica el contexto del problema a resolver, los principales objetivos, y la evolución y situación actual, la sección de análisis donde se explica en detalle el entorno a monitorizar, las herramientas existentes que se han analizado y la selección de las que se han considerado adecuadas para el contexto del problema, la sección de diseño donde se explica la arquitectura y políticas que se han diseñado para la monitorización, la sección de desarrollo donde se explican los pasos seguidos para la implementación y despliegue del sistema y finalmente la sección de pruebas donde se explican las pruebas que se han llevado a cabo sobre el sistema realizadas en un entorno de pruebas antes del paso al entorno de producción.

En los anexos se puede observar el tiempo invertido en este proyecto dividido en las diferentes fases, detalles del análisis de las herramientas, el estudio del cálculo del espacio que se va a necesitar para almacenar toda la información, detalles de configuración del sistema y de la implementación, algunos ejemplos de gráficos y pantallas donde se visualiza como muestra el sistema la información que recoge y los resultados de las pruebas realizadas.

Capítulo 2

Estado del arte

Siempre se ha tenido la necesidad de conocer el estado de los sistemas pero la forma en la que se realiza la monitorización de sistemas ha sufrido una evolución en función del paso del tiempo y dependiendo del tipo y tamaño de la infraestructura que se desea monitorizar. En un principio, la mayoría de las infraestructuras eran de un tamaño pequeño por lo que bastaba con realizar algunas comprobaciones de forma manual. Sin embargo llegó un momento en el que no era posible tener a alguien encargado de esto ya que el volumen era algo mayor, por lo que se creaban scripts ad hoc que recogían información relevante centralizándola en un único sistema para poder ver de forma rápida cual era el estado global.

A partir de esta visión de recogida centralizada de la información, surgen los sistemas de monitorización clásicos que son más elaborados y soportan más tipos de comprobaciones y personalización. La mayoría de estos sistemas cuentan con un servidor central en el cual se almacena toda la información en bases de datos de tipo RRDTool [3], orientadas a optimizar el almacenamiento de series de datos temporales realizando medias de los datos almacenados en intervalos de tiempos especificados. Estos permiten también configurar las comprobaciones de los agentes centralizando la configuración mediante ficheros de texto o interfaces web, o gestionar las comprobaciones de los agentes en cada uno en concreto. Dentro de este campo existen herramientas tanto open source como de software propietario. Una de las más destacadas hoy en día es Nagios [4], que es muy utilizada por ser una herramienta que ofrece una parte gratuita y otra parte de pago que mejora algunas características y permite ampliarla. Además, su éxito se debe a que es una herramienta muy trabajada que lleva muchos años en el mercado y sobre la cual existen muchos plugins realizados por terceros que funcionan correctamente, permitiendo monitorizar casi cualquier aspecto. Además, ofrece un alto grado de personalización, aunque esto requiere un amplio conocimiento de la herramienta y tener que invertir bastante tiempo ya que no es una tarea sencilla.

El tamaño de las infraestructuras de sistemas siguió creciendo, por lo que aparecieron sistemas de monitorización más escalables que admiten mayor número de hosts como por ejemplo Zabbix [5], que cuenta con un sistema de proxies que realizan la recogida de información de grupos de hosts para enviarla posteriormente al servidor central y almacenarla ahí, pero de esta manera los proxies son los que realizan todas las comprobaciones y el servidor central va recogiendo la información en grupos, por lo que no soporta tanta carga de trabajo y admite más hosts.

Actualmente, se han visto incrementados los sistemas que se están migrando a cloud por lo que las plataformas cloud han implementado sus propios sistemas de monitorización como es AWS Cloudwatch. Esta permite monitorizar todos los recursos y aplicaciones que se encuentran en AWS almacenando logs, definiendo métricas y alarmas personalizadas y visualizando gráficos y estadísticas.

A parte de esta división según el tamaño y tipo de infraestructura, también se pueden encontrar sistemas de monitorización centrados en un campo en concreto como por ejemplo en monitorización de aplicaciones web, monitorización de redes o análisis de logs siendo una de las más recientes ELK Stack [6], formado por Elasticsearch, Logstash y Kibana, que juntas permiten realizar la recolección de logs, su procesamiento y análisis, y mostrar gráficamente los resultados del análisis. A partir de este stack open source, se han desarrollado varias herramientas apoyadas en él pero de pago que ofrecen características adicionales, como Splunk y Logz.

Para esta infraestructura, se considera que los tipos de sistemas que más se adecuan para la monitorización son los que soportan diferentes elementos incluyendo monitorización básica, de red y web ya que se desea monitorizar un entorno completo. Para ello, además de las herramientas nombradas anteriormente de Nagios y Zabbix, se han extraido como apropiadas para el sistema Icinga2 [7] y Sensu [8] que están basadas en Nagios y sus plugins son compatibles, y Pandora FMS [9] y Shinken [10]. Todos estos sistemas son de tipo pull, a excepción de Sensu, y permiten configurar algunos elementos de monitorización para realizar comprobaciones tipo push. Sin embargo, Sensu permite configurar sus elementos como elementos de tipo pull pero también de tipo push, según lo que se crea más conveniente para cada uno. Para todas ellas se ha realizado un análisis más exhaustivo [11] buscando características concretas [12] basándose en los requisitos definidos para poder compararlas y elegir las más apropiadas.

2.1. Características y terminología

En monitorización existen dos tipos de servidores y comprobaciones de los elementos. Estos pueden ser de tipo pull, donde existe un servidor centralizado y es este quien realiza las peticiones a los diferentes clientes y espera la respuesta con la información obtenida, o de tipo push, donde es el cliente quien envía la información al servidor de forma asíncrona sin necesidad de que este realice ninguna petición.

Además de este tipo de consultas, destaca también la utilización de los protocolos ICMP [13] (Internet Control Message Protocol), para comprobar errores de red mediante el envío de peticiones a los hosts que envían una respuesta para determinar si estos están disponibles o no, y SNMP [14] (Simple Network Management Protocol), que se utiliza principalmente para consultar información sobre los dispositivos de red mediante la consulta de MIBs que organizan la información jerárquicamente utilizando una comunidad, que es una palabra clave que se utiliza para la autenticación.

Otra terminología importante que se utiliza en este contexto es que los elementos a monitorizar de cada host se denominan *items*, sobre los cuales se definen disparadores llamados *triggers*, que comprueban los resultados recogidos por estos items para ver si se ha producido algún error en la información esperada estableciendo una condición, expresión regular o umbral, llamado *threshold*, de tal forma que si lo cumple, se creará una alerta.

Habitualmente, también se habla de la funcionalidad de las tendencias o *trends*, que resulta importante ya que consiste en realizar una media de la información recogida en un periodo de tiempo para almacenar menos información y que esta ocupe menos espacio, aunque esto provoca una pérdida de precisión.

Un aspecto muy importante en estas arquitecturas es la metamonitorización [15], que consiste en la monitorización del propio servidor central que comprueba el estado del resto de sistemas. Es muy importante controlar lo que pasa en el propio sistema de monitorización ya que si este falla, se puede llegar a perder toda la información del resto de sistemas. Por ello, es importante recoger información básica de forma externa para conocer su estado y crear alertas preventivas que permiten anticiparse a los errores y evitar pérdidas importantes de información sobre todos los sistemas. Esta externalización resulta interesante ya que permite consultar la información y saber qué esta ocurriendo aunque se produzca algún error.

Capítulo 3

Análisis

Para realizar este proyecto, ha sido necesario estudiar el entorno en el que se mueve la empresa, los sistemas a monitorizar y los principales requisitos para poder comparar qué herramientas existen, las ventajas y desventajas presentadas por cada una y seleccionar la que se considere más adecuada. Desde la sede de Zaragoza y con este sistema de monitorización, se pretende cubrir la monitorización de 10 de las sedes pertenecientes al grupo donde se incluyen servidores y hosts, los servicios que estos alojan, servicios ambientales y de corriente respectivos al centro de datos y dispositivos de redes como routers, switches, firewalls y puntos de acceso WiFi.

3.1. Análisis del entorno

Inicialmente se ha realizado un estudio del entorno que se desea monitorizar. De este estudio se extraen los recursos que se van a tener que monitorizar para poder elegir posteriormente una herramienta que sea capaz de manejar todo tipo de sistemas y pueda manejar dicho volumen de monitorización.

Entre los recursos que se necesita monitorizar están incluidos servidores y hosts donde destaca la heterogeneidad de los sistemas ya que la empresa cuenta con sistemas Unix, Linux, Windows y ESXi de VMware. Esta monitorización debe ser tanto a nivel hardware como a nivel de red donde se incluyen los siguientes dispositivos:

- 138 Switches
- 18 Firewalls
- 3 Routers
- 65 Puntos de acceso WiFi
- 417 Máquinas virtuales
- 35 Hosts

Además es necesario monitorizar los servicios y webs que alojan los servidores con sus correspondientes tiempos de respuesta, diversas bases de datos y los servicios ambientales y de corriente respectivos al centro de datos. Algunos de los elementos más críticos de los sistemas son bases de datos que almacenan información crítica, controladores de dominio y los servicios que permiten a los usuarios autenticarse en los diferentes dispositivos, servicios SAP para gestión de clientes, empleados y nóminas, disponibilidad de sistemas de control de versiones, disponibilidad de webs corporativas y sus correspondientes certificados, estado de backups o elementos de red que permiten acceder a los recursos.

3.2. Requisitos del sistema

Tras estudiar el entorno mediante la información facilitada, se decidieron cuales eran los principales requisitos funcionales que debía cubrir el sistema de monitorización. En estos requisitos se les da especial prioridad a aquellos que no incluía el sistema anterior como la unificación de la monitorización, gestión y visualización en un mismo sistema, la personalización de los periodos de almacenamiento de información y tendencias y la gestión remota desde el servidor. Todos los requisitos se pueden observar en la siguiente Tabla 3.1.

3.3. Selección de la herramienta adecuada

Basándose en los requisitos definidos para el sistema, se ha realizado una comparación de las diferentes herramientas, como se puede observar en la siguiente Tabla 3.2 y en el Apéndice B se encuentra una explicación más detallada de los aspectos más relevantes de la comparación.

En dicha tabla, se puede observar como el color verde representa los requisitos que cumple cada herramienta, el color rojo aquellos que no, el amarillo los que tienen dicha característica pero de pago y el gris son aquellos en los que no se ha podido determinar si lo cumplen o no.

Identificador	Requisito
RF1	El sistema de monitorización debe permitir monitorizar clientes Linux y Windows desde Windows Server 2003.
RF2	El sistema de monitorización debe permitir monitorizar dispositivos de red, como switches, routers, firewalls y puntos de acceso WiFi.
RF3	El sistema de monitorización debe permitir monitorizar aspectos básicos locales de cada máquina como CPU, disco y memoria.
RF4	El sistema de monitorización debe permitir monitorizar los servicios y procesos de cada host.
RF5	El sistema de monitorización debe permitir monitorizar aspectos de forma remota mediante el protocolo SNMP.
RF6	El sistema de monitorización debe permitir analizar los logs de las máquinas que se quieren monitorizar.
RF7	El sistema de monitorización debe tener integrada la parte de gestión, alertas y gráficas temporales.
RF8	El sistema de monitorización debe permitir la personalización de los tiempos y la información que se desea almacenar.
RF9	El sistema de monitorización debe permitir la personalización creando plugins propios para ampliar la funcionalidad del sistema en caso de que sea necesario.
RF10	El sistema de monitorización debe permitir configurar los thresholds de los avisos de forma flexible e independiente.
RF11	El sistema de monitorización debe permitir el escalado de los avisos en función de la importancia de cada alerta y el tiempo transcurrido desde el inicio del aviso.
RF12	El sistema de monitorización debe permitir la creación de informes.
RF13	El sistema de monitorización debe permitir la gestión de eventos interna de diferentes usuarios para poder notificar el proceso y evolución que se sigue en las alertas que se generan.
RF14	El sistema de monitorización debe permitir la gestión remota de los clientes desde el servidor de monitorización para facilitar las tareas de mantenimiento y gestión.
RF15	El sistema de monitorización debe permitir visualizar los principales problemas que se han producido de forma sencilla e intuitiva.

Tabla 3.1: Requisitos funcionales del sistema de monitorización

Requisito	Icinga2	Nagios	PandoraFMS	Sensu	Shinken	Zabbix
RF1						
RF2						
RF3						
RF4						
RF5						
RF6						
RF7						
RF8						
RF9						
RF10						
RF11						
RF12						
RF13						
RF14						
RF15						

Tabla 3.2: Requisitos del sistema de monitorización

Finalmente, se decidió que las más adecuadas eran Zabbix y Pandora FMS por ofrecer todas las funcionalidades requeridas. Sin embargo, Pandora FMS se ha acabado descartando porque se ha comprobado que algunas de las funcionalidades se encuentran en la versión de pago y otras funciones principales se pueden realizar de forma manual pero no se permite la automatización o aplicación a varios hosts simultáneamente para facilitar la configuración y mantenimiento del entorno, haciendo que esta tarea sea más compleja tanto a corto como a largo plazo. Las principales tareas que son necesarias para el sistema y sin embargo, son mas costosas debido a la falta de automatización en la versión gratuita de Pandora son la configuración de los agentes centralizada, la creación de plantillas que permita crear una política y aplicarlas a varios hosts y la falta de algunos módulos básicos para la creación de elementos de monitorización.

Por este motivo, se ha decidido utilizar Zabbix para la monitorización de los sistemas de la empresa ya que esta cumple todos los requisitos planteados y todo está disponible de forma libre. Además se trata de una herramienta muy completa que ofrece gran capacidad de personalización, destacando la de las ventanas de tiempo durante las que se quieren almacenar los datos que ofrece comparada con el resto de herramientas. Esta capacidad de personalización también hace que la curva de aprendizaje sea más costosa en tiempo comparada con otras de pago, pero finalmente se podrá construir un sistema con las características deseadas.

Capítulo 4

Diseño del sistema

En la monitorización se pueden destacar dos partes sobre las que se va a trabajar:

La monitorización en tiempo real, que se refiere a la recogida de datos que se realiza en cada momento y a la creación de políticas donde se define el tipo de alertas que se van a producir en cada caso, y su escalado cuando sea necesario debido a la gravedad del problema. Su objetivo es la detección de los problemas en tiempo real y avisar al equipo encargado de los sistemas para que estos puedan solucionar el problema lo antes posible evitando así falta de disponibilidad para los usuarios y la sobrecarga de los equipos que pueda producir fallos mayores. En la política de alertas se debe definir qué alertas son generadas y almacenadas, cuántos usuarios reciben las alertas, cómo se reciben las alertas, cómo se comunica el equipo técnico de la gestión de las incidencias entre los diferentes miembros, cómo escalar las alertas si estas persisten según su gravedad, qué políticas aplicar a cada alerta en función de su gravedad, cuándo y cómo se debe silenciar un problema durante un periodo determinado.

Por otra parte, se puede diferenciar la monitorización histórica cuando se refiere al almacenamiento de la información que se ha ido recolectando en el tiempo y las políticas que se van a aplicar para su almacenamiento, visualización y tratado. Su objetivo es poder analizar mediante esta información histórica como han ido evolucionando los sistemas y su utilización para poder utilizar dicha información para planes de capacidad y poder reestructurar el sistema, añadiendo mas recursos donde sea necesario y eliminándolos donde no se utilicen.

4.1. Arquitectura del sistema

Existe un servidor central de Zabbix como se puede observar en la Figura 4.1, donde se encuentra la base de datos MySQL que recoge y almacena toda la información de los sistemas monitorizados. Además, desde este servidor central los usuarios, que en función del tipo de usuario tendrá permisos o no, pueden realizar todas las tareas de gestión mediante el interfaz web que ofrece el servidor web que contiene el servidor de Zabbix.

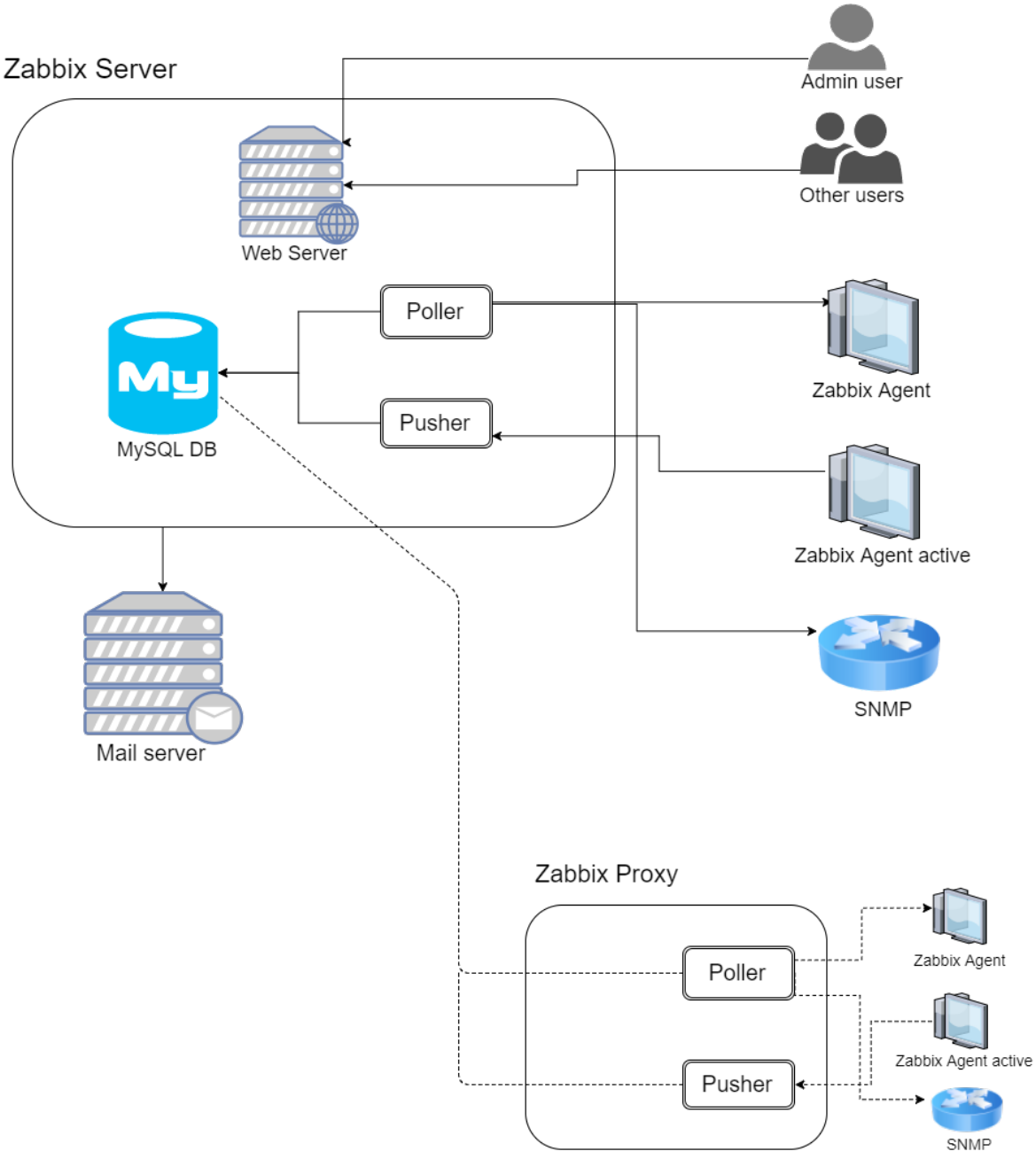

Figura 4.1: Arquitectura del sistema

El sistema sigue una arquitectura de tipo polling donde la mayoría de la información la solicita el servidor y son los agentes los que la envían sin encriptar, a través del puerto 10050 como se observa en la Figura 4.2. Para la consulta de elementos de red se ha utilizado el protocolo SNMP mediante la utilización de las mibs proporcionadas por los fabricantes, de tal forma que es el servidor quien las consulta directamente haciendo una petición SNMP como se observa en la Figura 4.3. También existe la posibilidad de configurar algunos chequeos como activos siguiendo un formato de tipo push de tal manera que el servidor envía la información sobre los elementos que se desean monitorizar y son los agentes los que se encargan de mandar la información a través del puerto 10051 cuando pasa el intervalo de tiempo configurado para dicho elemento como se observa en la Figura 4.4.

Además de estos tipos de consultas, existen otros que se pueden observar en el siguiente enlace [16] aunque no ha sido necesario utilizarlos en este caso.

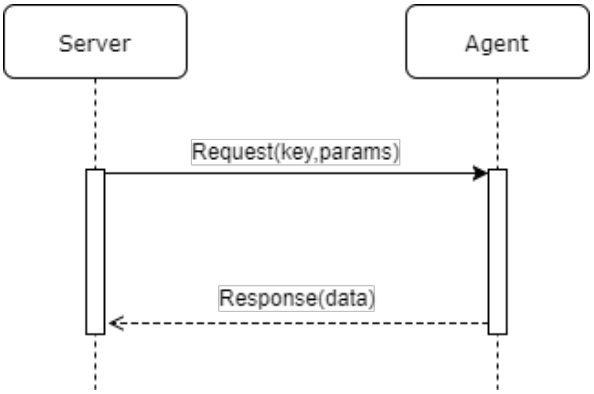

Figura 4.2: Comunicación entre servidor y cliente de forma pasiva

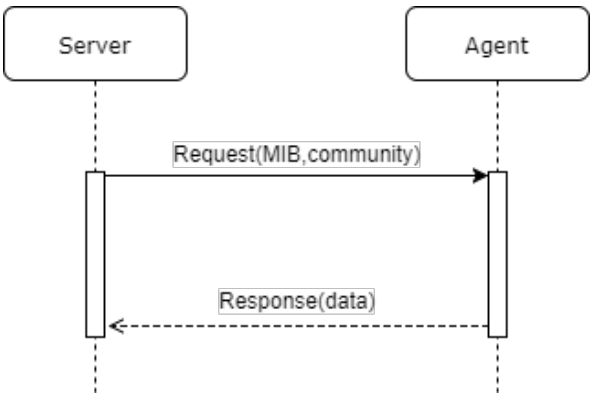

Figura 4.3: Comunicación entre servidor y cliente utilizando SNMP

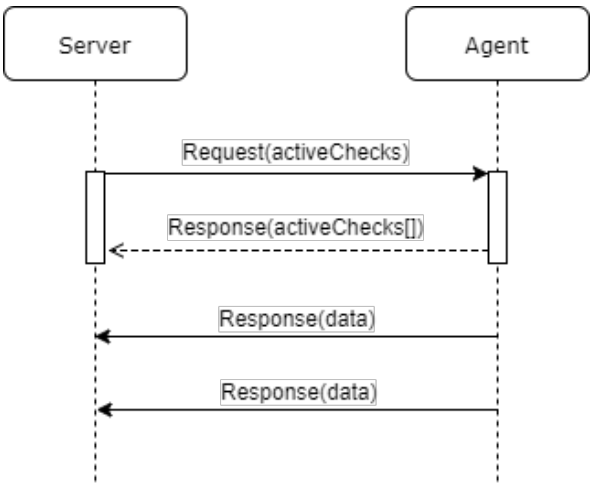

Figura 4.4: Comunicación entre servidor y cliente de forma activa

El sistema también presenta alta escalabilidad debido a la posibilidad de ampliar el sistema en un futuro mediante la instalación de proxies de Zabbix. En el diagrama de la Figura 4.1 se puede observar su funcionamiento que, al igual que el servidor, se encarga de recoger la información pero en lugar de almacenarla, la envía al servidor principal de Zabbix en bloque para que este pueda guardarla en la base de datos y disparar las alertas necesarias. De esta forma se evita si hay gran número de hosts la sobrecarga del servidor de chequeos y de envíos y recepciones de información.

4.2. Monitorización histórica

Zabbix, independientemente de la base de datos utilizada, permite definir los rangos temporales en los cuales se quiere almacenar información, tanto toda la información recogida como las tendencias. Además permite hacerlo aplicando una política general para todos los elementos monitorizados por el sistema, pero también permite definirlo de forma individual para cada elemento concreto que se va a monitorizar. En este caso se ha decidido almacenar la información en una base de datos MySQL con InnoDB [17] ya que, aunque requiere más CPU, incrementa el paralelismo mejorando el rendimiento para poder monitorizar mayor numero de host con más comprobaciones, proporciona mayor rapidez en lectura para ver y analizar los datos mediante el interfaz web sin problemas y garantiza las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).

Por una parte, se permite el almacenamiento de toda la información recogida durante el periodo de tiempo deseado. Esta información permite consultar con exactitud los datos recogidos para poder analizar la utilización de los sistemas.

Por otra parte, la funcionalidad de las tendencias permite condensar la información y almacenar la media por cada hora para los valores numéricos. El principal objetivo de esta función es reducir la cantidad de información almacenada reduciendo así el espacio necesario pero por contra, la gran desventaja que presenta es que no se permite consultar la información concreta para un instante determinado, lo que produce la pérdida de exactitud de la información si se desea analizar, por ejemplo, posibles picos en determinados momentos.

Para este caso concreto, se ha decidido definir de forma genérica el almacenamiento de las tendencias durante dos años para poder hacer un análisis anual que permita una vista general del sistema. Sin embargo, el almacenamiento de toda la información se ha definido de forma genérica en 90 días para que, si se desea realizar un análisis en profundad destacando picos máximos y mínimos, este pueda hacerse a corto plazo en un rango de 3 meses. De esta forma se reduce alrededor de un 81 % la información almacenada como se detalla en el Apéndice C, por tanto, el espacio necesario para ello.

Estos valores se pueden personalizar para cada elemento concreto, por tanto en la Sección 4.4 donde se detallan todos los elementos, se explica si algún elemento modifica su periodo de almacenamiento de información y el motivo.

Para la visualización de la información histórica recogida, Zabbix permite la creación de pantallas específicas para cada hosts donde se pueden representar las gráficas simples que muestran todos los valores recogidos para cada elemento por separado, y además permite definir gráficas personalizadas donde se pueden representar diferentes elementos para poder compararlos en una gráfica de forma más sencilla. En estas gráficas se puede ajustar el periodo de tiempo que se desea ver para poder visualizar con más detalle cualquier intervalo de tiempo.

4.3. Monitorización en tiempo real

La monitorización en tiempo real se divide a su vez en monitorización de disponibilidad y de capacidad. El primer tipo se basa en la detección de fallos en los hosts, servicios o webs monitorizadas mientras que el segundo tipo se refiere a la detección de la sobrecarga de algún elemento del sistema, como CPU o disco, que si se prolonga en el tiempo puede llegar a provocar la falta de disponibilidad de los sistemas.

Para detectar estos problemas, es necesario monitorizar los elementos necesarios del sistema y crear alertas basadas en thresholds que sean capaces de detectar los problemas a tiempo y avisar a los administradores de sistemas.

En Zabbix, la monitorización en tiempo real se basa en la creación de triggers que se asocian a cada elemento a monitorizar. Estos triggers realizan comprobaciones basadas en un threshold, de tal forma que si el valor recogido cumple la condición definida en el trigger, este se dispara y crea una notificación en el panel de problemas de Zabbix. Estos problemas se clasifican en los siguientes niveles:

- Información: Informa de un cambio que ha ocurrido pero que no puede producir problemas en los equipos.
- Aviso: Informa de un problema que ha ocurrido que puede provocar fallos a largo plazo.
- Crítico: Informa de un problema que ha ocurrido que puede provocar fallos a corto plazo.
- Desastre: Informa de una situación de error que se debe solucionar inmediatamente.

Estos problemas se solucionan automáticamente volviendo a un valor por debajo del indicado en el trigger y se marcan como resueltos en el panel de Zabbix. En otros casos puede configurarse una expresión complementaria que se define como *Recovery expression* y cuando esta expresión se cumple es cuando el problema se considera resuelto. Además del panel de problemas, se puede realizar la notificación de los problemas de diversas formas, como enviar una notificación por correo a diferentes usuarios, SMS, Jabber o Ez Texting.

Para este sistema se ha decidido utilizar la notificación de los problemas mediante el panel de notificaciones de Zabbix y también por correo a diferentes usuarios para los errores más graves calificados como nivel desastre y para otros casos concretos ya que es una forma rápida de notificación porque los usuarios a los que se les va a notificar pueden recibir y leer los correos en su Smartphone y enterarse al momento de los problemas indicados.

Uno de los problemas más habituales en los sistemas de monitorización es la creación de notificaciones de problemas en cascada. Para evitar este problema se han creado dependencias entre los triggers lo que permite que si un trigger depende de otro y el trigger del que depende falla, este no genera alerta para evitar alertas en cascada.

4.4. Políticas de monitorización

En primer lugar, para la creación de las políticas se ha decidido subdividir los sistemas en diferentes grupos que reúnen características similares para poder aplicar a cada uno unas políticas concretas.

Los grupos que se encuentran en el sistema son:

- Comprobación de red: Disponibilidad y tiempo de respuesta.
- Características básicas: CPU, memoria, interfaces red, sistemas de ficheros, e información del sistema.
    - Información individual: Logs.
    - Servicios y procesos concretos: Servicios de autenticación, control de versiones y servicios web.
    - Bases de datos: Disponibilidad, conexiones y operaciones.
- Dispositivos de red: Tráfico de entrada y salida, estado y errores de cada interfaz.

A continuación se desarrollan las políticas diseñadas que se van a aplicar a cada grupo. La política general de recogida de la información que se aplica es cada 5 minutos para los elementos más críticos. Se debe tener en cuenta que hay algunos servicios que se van a monitorizar para los cuales es difícil encontrar características conjuntas, por lo que dichas políticas permiten la personalización para elementos que tengan características diferentes.

Políticas de comprobación de red

En primer lugar se va a comprobar la disponibilidad de los diferentes hosts. En caso de que el host no esté disponible, se generará una alerta de nivel crítico.

También se va a comprobar el tiempo de respuesta teniendo en cuenta que en este entorno, el tiempo medio está por debajo de los 2 milisegundos, por tanto se va a establecer una alerta de advertencia cuando la media durante los últimos 10 minutos sobrepase este valor.

Además existe una dependencia entre las variables recogidas ya que si en algún momento el host deja de estar disponible, el tiempo de respuesta va a aumentar. Por este motivo y para evitar alertas en cascada, es necesario definir esta dependencia en el sistema para que si se dispara la alerta de que el host no está disponible, no se dispare también la que indica que el tiempo de respuesta es elevado.

Políticas de características básicas de los hosts

Se recoge la información del sistema y el tiempo que lleva el sistema funcionando. Esta información es menos relevante y no cambia continuamente o los cambios son predecibles por lo que basta con recogerla cada hora. Su almacenamiento tampoco es tan importante ya que no se va a analizar ningún resultado a partir de esta por lo que solo se guarda durante una semana y un día respectivamente. Ambos tienen asociados dos alertas que reportan información sobre si la información del sistema ha cambiado o el sistema ha sido reiniciado.

Se mide la carga de CPU, considerándose un nivel crítico cuando llega a 5 la media de los últimos 10 minutos.

Se mide la memoria total, disponible y utilizada, y el porcentaje de la memoria libre. Con los datos de memoria recogidos se pretende representarlas gráficamente para poder analizar a largo plazo los recursos del sistema disponibles y utilizados en cada momento, mientras que con el porcentaje se establece un nivel de advertencia cuando el sistema llega al 80 % y un nivel crítico al alcanzar el 90 %. Del mismo modo se mide la memoria swap, estableciendo los mismos niveles de alerta.

También se recoge el número de procesos que se están ejecutando en cada momento para poder analizarlos a largo plazo y en caso de superar los 300 procesos se emite una alerta de advertencia.

También se van a monitorizar las diferentes particiones de disco con su espacio total, disponible y utilizado, y el porcentaje libre para poder analizarlo gráficamente a largo plazo y además establecer alertas de advertencia cuando quede 10 % del espacio libre, y de nivel crítico cuando quede sólo 5 % libre. Para esta información solo se va a almacenar toda la información durante una semana ya que cada host puede tener varias particiones y esto puede requerir bastante espacio si se almacena durante 3 meses, y no resulta muy relevante. Además la información del espacio total se va a recoger cada hora ya que es un dato que se va a mantener constante.

Se va a recoger la información relativa a los interfaces de red más importantes, tanto de su tráfico de entrada como del tráfico de salida y, al igual que para las particiones de disco, solo se va a guardar su información completa durante 7 días ya que también puede resultar muy costoso en espacio. En este caso no se va a definir ninguna alerta a nivel general, ya que solo en algunos casos particulares será necesario establecer un umbral de alerta.

Políticas de información de sistemas individuales

Se quiere conocer información concreta de cada sistema. Para ello se recoge la información del sistema que indica que se ha producido algún fallo y se lanza una alerta cuando se detectan, pero también se podrían definir elementos concretos para monitorizar la información de aplicaciones concretas.

Políticas de servicios de autenticación

Se comprueba el estado del servicio que se encarga de obtener la información correspondiente sobre usuarios del controlador de dominio. En caso de que el servicio no esté funcionando correctamente, se avisa con un nivel de alerta de advertencia y se dispara una acción que reinicia el servicio. Si esta acción no es suficiente para solucionar el problema, este se escala enviando un correo al administrador para que sea consciente de que el problema no se puede resolver de forma automática y es necesaria la intervención.

Políticas de servidores de sistemas de control de versiones

Se comprueba el correcto funcionamiento del servicio de sistemas de control de versiones para que los usuarios puedan acceder al contenido y en caso de fallo se lanza una alerta de nivel crítico.

Políticas de servidores web

Se recoge la información necesaria para comprobar que el puerto que ofrece el servicio web está disponible y en caso de fallo se lanza una alerta de nivel crítico que avisa de este fallo.

También se comprueba que la comunicación entre cliente y servidor es segura y la información que se transmite se encuentra cifrada, para lo que es necesario comprobar el estado del certificado que garantiza esta condición. Por tanto se monitoriza cada día ya que esta información no cambia continuamente y se almacena solo durante una semana y un día respectivamente ya que se desea conocer esta información solo en el momento en que se consulta, pero no interesa conocer el histórico. Para avisar del tiempo de expiración con suficiente antelación, se han establecido diferentes niveles de alerta, comenzando cuando faltan 90 días con una alerta informativa, igual que cuando faltan 60 días. Después el nivel de alerta va subiendo a aviso cuando faltan 30, aviso medio cuando falta 15, aviso alto cuando faltan 7 y desastre cuando ha expirado el certificado.

Otro dato importante es conocer si es posible acceder correctamente al contenido de la web, para lo que es necesario acceder a dicho contenido y comprobar si es el adecuado y su tiempo de respuesta correspondiente. Por tanto se genera una alerta cuando el contenido no es el esperado o cuando su tiempo de respuesta sobrepasa un timeout.

Políticas de bases de datos

Para monitorizar las bases de datos se ha decidido recoger información sobre el estado actual de la base de datos, es decir, si está funcionando correctamente, el número de conexiones que hay con la base de datos, los diferentes tipos de operaciones (begin, insert, delete, commit, rollback, update, select) por segundo que se realizan para poder comprobar en que momentos hay mayor actividad de cada tipo de operación, las queries que se realizan para tener información de la actividad que hay en cada momento y poder analizarla a posteriori y la versión. Este último dato no cambia continuamente por lo que solo se recoge una vez al día y se almacena durante una semana para conocer qué versión hay actualmente, pero no es necesario conocer el histórico. Es necesario conocer cuando se produce un fallo en la base de datos, por lo que se crea una alerta de nivel crítico que avise cuando lo detecte.

Políticas de dispositivos de red

Se comprueba la información general de los dispositivos de red como el nombre del dispositivo, su descripción, la localización y los detalles de contacto una vez al día ya que se trata de una información que no cambia a menudo y simplemente interesa conocerla sin tener que recopilarla en tiempo real. Además, esta información tampoco hace falta almacenarla durante un periodo largo de tiempo porque solo interesa conocer la información asociada en el momento, por lo que el periodo de almacenamiento se ha fijado solo en una semana.

En los dispositivos de red, es importante conocer la información respectiva a los interfaces como el número de interfaces de red que tiene el dispositivo. Sobre estos interfaces se va a monitorizar su estado, la información de trafico de red entrante y saliente y los errores que se producen, tanto no poder transmitir paquetes como los paquetes de entrada que contienen errores, para cada uno. Esto permitirá en casos concretos establecer umbrales y crear alertas para poder avisar en caso de que haya demasiado trafico y también para analizarlo a largo plazo y ver en que momentos se producen picos y cuando casi no hay tráfico, y estudiar sus motivos. Además se va a definir un aviso crítico que avisa si se ha producido algún error en alguno de los interfaces.

4.5. Metamonitorización

La metamonitorización, como se ha explicado anteriormente en la Sección 2.1, es un aspecto muy importante. Para controlarlo, se ha decidido monitorizar los mismos aspectos básicos referentes a la base de datos MySQL utilizando las mismas características que se han explicado anteriormente Sección 4.4 y algunos aspectos relacionados con los procesos internos del servidor desde el propio servidor de Zabbix.

Los procesos que se quieren monitorizar son aquellos que se encargan de realizar las comprobaciones icmp, http, snmp, y del agente, tanto activas como pasivas. También se van a monitorizar los procesos que se encargan de tareas internas como eliminación de datos antiguos, gestión de alertas, escalado de acciones y sincronización con la base de datos. Para estos elementos, se van a establecer dos niveles de alerta, uno de aviso cuando estos procesos están ocupados al 75 %, y otro crítico para cuando están ocupados al 90 % ya que el sistema podría estar cercano a un fallo grave en el cual se puede producir pérdida de información.

También se va a realizar un seguimiento de colas de elementos de monitorización que están pendientes, creando una alerta de aviso cuando en los últimos 5 minutos haya 50 elementos pendientes, y de alerta crítica cuando en los últimos 10 minutos el número de elementos supere a 100, porque se está retrasando la recogida información de algunos elementos.

Se va a recoger la información sobre los valores de zabbix procesados por segundo para poder observar la evolución con el tiempo, y los valores de cache libre en lectura y escritura para los históricos, configuraciones, índices y tendencias. En estos casos, habrá un aviso cuando quede menos del 25 % libre y un aviso crítico cuando sea menos del 10 %.

Toda esta información se va a almacenar sólo durante una semana porque resulta importante conocer en el momento si el sistema esta sufriendo algún problema, pero no es necesario almacenar todos los detalles durante más tiempo, basta con las tendencias para conocer la información global y poder analizarla.

El servidor central de Zabbix es el elemento más importante del sistema ya que si falla, se pierde la información del resto, por ello, es necesario que si se produce algún problema en él y resulta imposible acceder a su información, se pueda conocer la información básica de su estado. Por este motivo, se ha decidido crear un script que realice una monitorización de los aspectos más básicos que almacene la información en una máquina externa al servidor para poder acceder a ella en caso de fallo grave del sistema, y que envíe notificaciones vía email siguiendo las Políticas de comprobación de red y las Políticas de características básicas de los hosts.

Capítulo 5

Implementación

El desarrollo del proyecto se divide en las siguientes fases que se desarrollan a continuación: Puesta en marcha del servidor central Zabbix y los agentes, implementación de las políticas, y finalmente la comprobación de la correcta recogida de datos.

Todas las figuras que se referencian en este apartado se encuentran en el Apéndice E.

5.1. Puesta en marcha del sistema de monitorización

5.1.1. Puesta en marcha del servidor de Zabbix

Inicialmente se ha creado una máquina virtual con sistema operativo CentOs 7 a la que se ha configurado su dirección IP 10.250.14.107 y se le ha asignado un espacio total de 100 GB teniendo en cuenta el espacio total que ocupan todos los elementos como se ha calculado en el Apéndice C.

En dicha máquina virtual se ha instalado inicialmente una base de datos MySQL versión 5.7.18 siguiendo los pasos explicados en [18] y se ha configurado para que el usuario Zabbix tenga todos los permisos necesarios para modificar la información que se va a recoger de la monitorización mediante la ejecución del siguiente comando:

```
grant all privileges on zabbix.* to zabbix@localhost identified by
'<password>';
```

A continuación se ha instalado el servidor de Zabbix siguiendo los pasos explicados en la documentación oficial [5]. Un aspecto importante ha sido desactivar selinux en el sistema mediante la modificación de *selinux* ya que si no el sistema no funciona correctamente y modificar el fichero de configuración de Zabbix *zabbix_server.conf* para introducir los parámetros del host y nombre, usuario y contraseña de la base de datos que va a utilizar.

5.1.2. Puesta en marcha de los agentes de Zabbix

La instalación de los agentes se ha realizado siguiendo los pasos que aparecen en la documentación oficial [5] para diferentes sistemas operativos, desde Windows Server 2003 hasta Windows Server 2016 y en diferentes Linux como CentOs, Debian, Ubuntu y sus diferentes versiones.

Para la correcta comunicación entre servidor y agentes ha sido necesario habilitar la comunicación entre los puertos 10050 y 10051 de los hosts que tienen los agentes y el servidor Zabbix en los firewalls corporativos ya que a través de ellos se envían la información necesaria.

También se ha modificado el fichero de configuración *zabbix_agent.conf* en cada cliente, donde se ha añadido el servidor y servidor activo, el hostname de la máquina y los metadatos que coincidirán con el grupo al que va a pertenecer el host para que este sea añadido automáticamente. Para ello se han modificado las siguientes líneas:

```
Server=10.250.14.107
ServerActive=10.250.14.107
Hostname=Hostname de cada host*
HostMetadata=Grupo/s a los que va a pertenecer*
```

Además, para aquellos host que se desea ejecutar comandos de forma remota desde el servidor es necesario habilitarlo modificando en el mismo fichero la siguiente línea:

```
EnableRemoteCommands=1
```

Y se necesita habilitar al usuario Zabbix para ejecutar los comandos remotamente sin necesidad de utilizar contraseña utilizando el comando sudo. Para esto hay que añadir a */etc/sudoers* la siguiente línea:

```
zabbix ALL=NOPASSWD: ALL
```

Una vez instalado el agente, es necesario configurar en el servidor el host, a qué grupos pertenece y qué es lo que se desea monitorizar en dicho host. Para añadir el host al servidor y asignarle un grupo y las plantillas se puede realizar de forma manual añadiendo su información pero para facilitar esta tarea se ha configurado el servidor para que al instalar un agente, este envíe una petición activa y se añada automáticamente, se le asigne un grupo y sus plantillas correspondientes, y además lea la información relacionada con el sistema operativo.

Para ello se han configurado varias acciones siguiendo las reglas definidas en la Tabla 5.1 donde se especifican los principales grupos predefinidos en el sistema, la información que se debe incluir en los metadatos y las plantillas que le corresponden a cada grupo. Estas reglas permiten que cualquier host que esté dentro de ellas sea detectado y se le apliquen los grupos y plantillas especificados. Para ello, se define una acción donde los metadatos del hosts cumplen una expresión regular, que coincidirá con el grupo al que va a pertenecer y se ha añadido previamente en el fichero de configuración, y a continuación se definen las operaciones asociadas a dicha acción. En este caso existe una regla que añade al servidor a todos los hosts que se detectan y otras que añaden el host al grupo que se desea y les asignan las plantillas predefinidas que se consideran necesarias para ese grupo. Por defecto, cualquier host añadido al sistema de esta forma pertenecerá al grupo *Discovered hosts* pero posteriormente se podrán modificar los elementos a monitorizar si fuera necesario.

En el Anexo Subsección D.11.1 se puede observar como se debe configurar en Zabbix las acciones que se han tenido que definir para cumplir las especificaciones.

Grupos	Metadatos	Plantillas
Linux servers	linux	Template OS Basic y Template Linux Log Messages
Windows servers	windows	Template OS Basic y Template Windows Event Log
SQL servers	sqls	Template Microsoft SQLServer
MySQL servers	mysql	Template App MySQL

Tabla 5.1: Reglas de asignación de hosts a grupos y plantillas

5.2. Implementación de políticas

Para implementar las políticas diseñadas, se ha decidido utilizar el mecanismo de plantillas, *templates* en Zabbix, que permite automatizar las políticas y crear los elementos que recogen la información necesaria y sobre ellos definir los triggers que se disparan cuando se superan los umbrales establecidos. En el siguiente mapa de relaciones Figura 5.1 se observan las plantillas definidas y las relaciones que se han establecido entre ellas.

Estas plantillas se pueden asociar a cada host y además se pueden modificar los elementos para cada host concreto si se desean cambiar los parámetros por defecto para algunos casos en concreto. También se permite añadir elementos y alertas independientes en casos concretos donde no es necesario crear una plantilla para aquellos que solo se dan en algún host en particular.

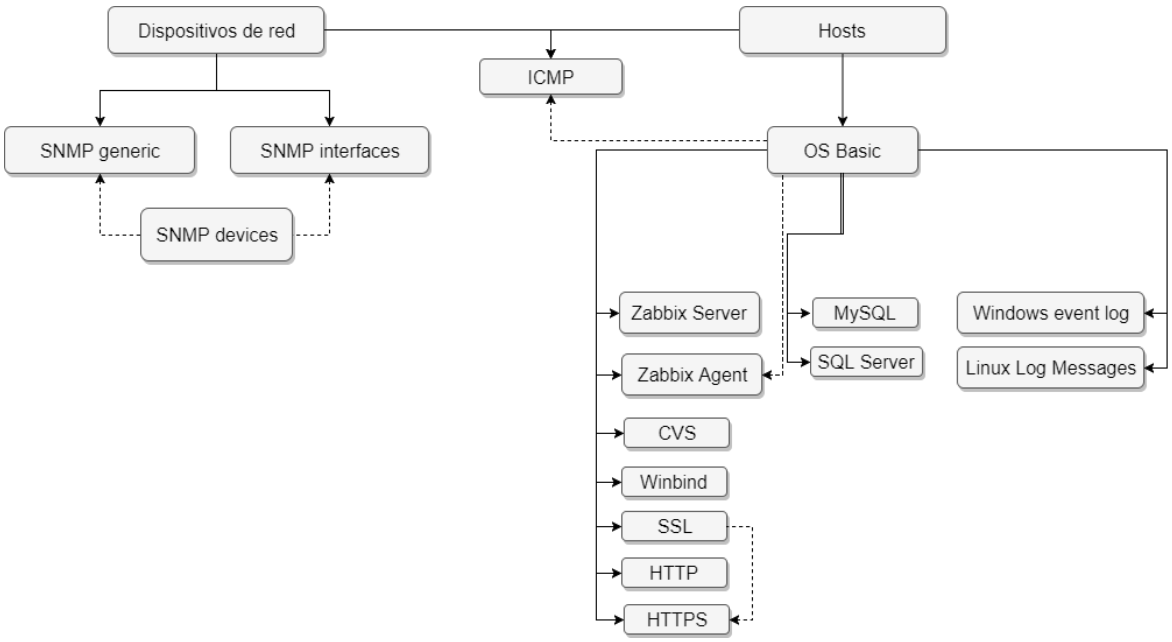

Figura 5.1: Diagrama de relaciones entre políticas

En el Apéndice D se explican los elementos que forman parte de las plantillas, los parámetros configurables del sistema y a continuación se describen los elementos que se han definido para las políticas definidas en la Sección 4.4, donde las relaciones entre políticas se establecen mediante relaciones en las plantillas utilizando *Linked templates*.

Template ICMP Ping

Para las Políticas de comprobación de red se va a utilizar el protocolo ICMP, que utiliza la herramienta ping para realizar las comprobaciones de disponibilidad y de tiempo de respuesta. Ambos elementos se agrupan en la aplicación ICMP.

Para evitar falsos positivos y asegurarse de que el host contesta a la petición, el ping espera 3 respuestas y si una de las 3 es correcta, la comprobación devuelve que el host está disponible.

Para la implementación de esta plantilla se han definido los siguientes items Figura E.1 con sus correspondientes keys y sus intervalos, y los triggers con las siguientes expresiones y dependencias Figura E.2, basándose en las reglas definidas en el diseño del sistema.

En este caso no se ha considerado relevante definir gráficos ni pantallas ya que pocos hosts van a monitorizarse utilizando esta plantilla de forma individual, por tanto sus gráficos se incluirán en las plantillas que están relacionadas con esta.

Template App Zabbix Agent

Esta plantilla recoge información relevante al agente de Zabbix, como la versión una vez al día ya que no es un elemento crítico y simplemente se almacena durante 7 días para detectar qué versión se tiene en cada host. En caso de actualizar el agente, se notifica con una alerta de información.

También el agente de Zabbix recoge un heartbeat continuo que se realiza cada 5 minutos, como el resto de comprobaciones, pero solo se almacena durante un día ya que no interesa analizarlo a largo plazo. Esta información solo se desea contrastar con la respuesta producida por el ping en caso de que se produzca algún fallo en un momento concreto para conocer si se está produciendo un fallo en el agente de Zabbix o en el host en general. En caso de no recibir información del agente en los últimos 5 minutos, se dispara una alerta con nivel crítico para informar de la situación.

Los elementos correspondientes a esta plantilla se agrupan en la aplicación Zabbix agent y se definen los siguientes elementos Figura E.3 y triggers Figura E.4.

Template OS Basic

Recoge las métricas de las Políticas de características básicas de los hosts y se incluyen las plantillas de ICMP Ping y Zabbix Agent para comprobar la disponibilidad en todo momento de los hosts.

Los elementos que se monitorizan con esta plantilla se ven agrupados en los siguientes grupos: General, CPU, Memory, Processes y Network interfaces, y los elementos y triggers definidos se pueden ver en la Figura E.5 y Figura E.6 respectivamente.

Además se han definido reglas de descubrimiento que permiten descubrir todos los elementos referentes a un aspecto concreto para monitorizar particiones de disco, como se puede ver en Figura E.7 y Figura E.8, e interfaces de red , como se puede ver en Figura E.9.

Esto facilita la tarea de monitorización ya que algunos hosts tienen muchos elementos y evita introducirlos manualmente porque cambian de un host a otro. La desventaja de utilizar este método es que se monitorizan todos los interfaces de red y particiones, y puede que algunas no resulten interesantes, por lo que posteriormente habrá que eliminar o deshabilitar los que se consideren irrelevantes para evitar la recolección innecesaria de información.

Además, en este caso se han definido 4 gráficos combinando diferentes elementos para facilitar la visualización y comprensión de:

- La memoria total, utilizada y libre.
- La swap total, utilizada y libre.
- Espacio total, utilizado y libre de cada partición.
- Tráfico de entrada y salida para cada interfaz.

También se han definido 3 pantallas, que incluyen los gráficos anteriores, para facilitar la visualización de los elementos en conjunto:

- Métricas generales del sistema: CPU, memoria, swap, ping, tiempo respuesta ping, pérdida ping y numero de procesos. Ejemplo Figura E.38.
- Métricas de los elementos de descubrimiento de tráfico de red por interfaz. Ejemplo Figura E.39.
- Métricas de los elementos de descubrimiento de utilización de particiones de disco. Ejemplo Figura E.40.

Template Windows Event Log

Esta plantilla es la implementación de las Políticas de información de sistemas individuales para sistemas Windows. Todos los items se han agrupado en la aplicación de Windows Event Log y se recoge la información de event log con niveles Warning y Error para tener conocimiento de aquellos fallos críticos que se producen en el sistema y en las aplicaciones, como se puede ver en los items Figura E.10. Además cuando se aplica esta plantilla, se recoge la información a partir de ese momento, pero no los eventos anteriores. También se han definido los triggers Figura E.11 que avisan cuando se detecta algún problema.

Template Linux Log Messages

Esta plantilla es la implementación de las Políticas de información de sistemas individuales para sistemas Linux. Se ha definido un item que pertenece a la aplicación de Logs y recoge la información almacenada en /var/log/messages que indica que se ha producido algún error utilizando el siguiente item Figura E.12. Además cuando se aplica esta plantilla, se recoge la información a partir de ese momento, pero no los logs anteriores. También se ha definido un trigger Figura E.13 que avisa cuando se detecta algún problema.

Template App Winbind

Para la implementación de las Políticas de servicios de autenticación se monitoriza el servicio winbind, que es el que se encarga en sistemas Unix de consultar al controlador de dominio la información relacionada con los usuarios. Para esto se ha definido la aplicación Winbind a la que se le ha asignado el siguiente item Figura E.14 que se trata de la ejecución de un comando remoto y la recogida su resultado. Para poder ejecutarlo hay que habilitarlo en el cliente tal y como se explica en Puesta en marcha de los agentes de Zabbix.

Para saber si el resultado ha sido correcto o no, es necesario analizar el resultado utilizando una expresión en el trigger Figura E.15 que comprueba si el resultado no contiene "succeed", entonces el trigger se dispara avisando del fallo.

Además, se ha definido una acción para poder reiniciar el servicio cuando se detecte el fallo de Winbind. Para ello se ha definido la condición Figura E.16 que detecta cuando el trigger se dispara porque ha fallado el servicio y se definen las operaciones Figura E.17 y Figura E.18 que se van a ejecutar al detectarlo, que permiten rearrancar el servicio ejecutando de forma remota un comando, y si este no soluciona el problema, envía un correo al administrador.

En caso de que la ejecución remota se realice correctamente, en la pantalla de problemas se observará como se ha realizado dicha acción y aparecerá el problema resuelto.

Template App CVS Service

El sistema utilizado para control de versiones es CVS (Concurrent Versions System), el cual utiliza el puerto 2401 para acceder a la información del repositorio. Por este motivo, para la monitorización de su disponibilidad se comprueba este puerto como se observa en el item definido Figura E.23. Este se ha incluido en la aplicación CVS service y también se ha definido el trigger Figura E.24 que avisa cuando se produzca un fallo.

Template App HTTP Service

Para ofrecer el contenido web se hace mediante HTTP (Hypertext Transfer Protocol), el cual utiliza el puerto 80. Por tanto para la implementación de la disponibilidad de las Políticas de servidores web se debe monitorizar dicho puerto. Se ha utilizado una plantilla donde solo es necesario definir la aplicación HTTP service y un item Figura E.19 que pertenece a ella donde se monitoriza este elemento, con su correspondiente trigger Figura E.20 que crea una alerta cuando se produce un fallo.

Template App HTTPS Service

Siguiendo el mismo esquema de monitorización que para HTTP, se debe monitorizar HTTPS (Hypertext Transfer Protocol Secure) ya que proporciona el contenido al servidor web de forma segura mediante el puerto 443. Por ello se ha definido la aplicación HTTPS service con su correspondiente item Figura E.21 y un trigger Figura E.22 para avisar de los fallos.

Se comprueba el tiempo de validez que le queda al certificado SSL y quién es el distribuidor del certificado cada día ya que esta información no cambia continuamente y se almacena solo durante una semana y un día respectivamente ya que se desea conocer esta información solo en el momento en que se consulta, pero no interesa conocer el histórico. También se comprueba que el servicio HTTPS está funcionando correctamente como en la Sección 4.4

Template SSL Cert

Para garantizar una comunicación segura entre cliente y servidor, se han utilizado certificados SSL (Secure Socket Layer). De este modo se debe comprobar quién es el proveedor y la validez de los certificados, para lo que ha sido necesario utilizar un script que comprueba esta información mediante la utilización de openssl. Este script se ha obtenido de [19] y recibe unos parámetros, los cuales se le pasan desde la comprobación configurada en Zabbix. Estos parámetros son *HOST.CONN* que es el nombre del host y lo lee de forma automática desde esta variable de Zabbix, *$SSL_PORT* que es una macro definida a nivel de la plantilla donde se define el puerto como el 443, y *$SNI* que es otra macro, pero esta tiene que definirse a nivel de host para cada host al que se aplique dicha plantilla.

Además, para que se pueda monitorizar utilizando este script como un script externo es necesario realizar los pasos explicados en la Sección D.7.

Para poder realizar esta monitorización se han definido los siguientes items Figura E.25 y los siguientes triggers Figura E.26 siguiendo las políticas establecidas y se han establecido dependencias entre los triggers para evitar alertas en cascada ya que, si faltan pocos días para que caduque, todas las alertas anteriores no deben aparecer. Ambos incluyen los definidos en Sección 4.4 ya que se ha incluido esta plantilla.

Web scenarios

Para comprobar el acceso a la información que ofrece el servidor web, comprobar que contenido se espera y el tiempo de respuesta que forman parte de las Políticas de servidores web, es necesario configurar un escenario web en particular para cada web que se quiera monitorizar. Esto se debe a que es necesario introducir la URL y esto varía en función de cada sitio web, por lo que no se puede crear una plantilla que englobe a todos. Además, para confirmar que el contenido se devuelve correctamente, se tiene que comprobar el contenido mediante el uso de una expresión regular que variará en cada caso. Además, en este caso no es necesario definir triggers concretos ya que si no se obtiene el contenido en un tiempo inferior al timeout especificado o el contenido no cumple con la expresión regular definida, se disparan las alertas automáticamente.

Template App MySQL

Para la monitorización de bases de datos MySQL se implementan las Políticas de bases de datos. Algunos elementos más relevantes de los permitidos se pueden encontrar en el siguiente enlace [20]. Para poder recoger la información del agente es necesario incluir en el directorio en */var/lib/zabbix* el fichero *.my.cnf* con la siguiente información:

```
[client]
user=Usuario de la base de datos
password=Contraseña de la base de datos
```

Para esta plantilla solo se ha definido el grupo de aplicaciones MySQL ya que todos los elementos a monitorizar pertenecen a este único grupo.

A continuación, en la Figura E.27 se pueden observar los items creados basados en las políticas de bases de datos y en la Figura E.28 el trigger.

Se han definido dos gráficos que se recogen en una pantalla, donde también se representarán el número de conexiones y las queries por segundo, se puede visualizar un ejemplo en la Figura E.41. Los gráficos compuestos agrupan:

- Métricas de ancho de banda: Se representan los bytes enviados y los recibidos por segundo.
- Métricas del número de operaciones por segundo de: Begins, commits, deletes, inserts, rollbacks, selects y updates.

Template Microsoft SQLServer

Para la monitorización de esta base de datos se implementan las Políticas de bases de datos y se ha utilizado como base la siguiente plantilla [21], modificándola para adecuarla a las políticas de bases de datos. A continuación, se pueden observar los elementos Figura E.29 y triggers Figura E.30 configurados y en el siguiente enlace [22] se observan algunos elementos adicionales que se pueden añadir a la monitorización del mismo modo.

Todos los elementos definidos pertenecen al grupo de SQL server y además, los servicios pertenecen al grupo Services.

Para facilitar la visualización de la información y poder compararlos, se ha definido un gráfico compuesto por los recorridos completos, los recorridos de intervalos, los recorridos de sondeo y las búsquedas en índices. Además este gráfico se ha incluido en una pantalla donde se muestra toda la información importante relacionada con la base de datos como se puede observar en la Figura E.42 y Figura E.43.

Template SNMP Generic

Esta plantilla implementa parte de las Políticas de dispositivos de red para aportar modularidad al sistema, recogiendo la información genérica de los dispositivos de red, donde todos ellos se agrupan en la aplicación de GeneralSNMP. Para ello se han definido los siguientes items Figura E.31.

Template SNMP Interfaces

Esta plantilla implementa la parte relacionada con los interfaces de las Políticas de dispositivos de red recogiendo la información relacionada con los interfaces de los dispositivos de red, donde todos ellos se agrupan en la aplicación de InterfacesSNMP. Para ello se ha definido una regla de descubrimiento que permite descubrir automáticamente los diferentes interfaces de red que están activos para evitar recoger información innecesaria, y a partir de ahí definir qué elementos se quieren monitorizar. Para ello es necesario definir la regla que permite descubrir los interfaces a partir de sus índices y su estado como se puede ver en la Figura E.32, y se ha tenido que añadir un filtro para especificar que el estado de los elementos que recoge sea solo los que están activos como se puede ver en la Figura E.33. También se pueden ver los items Figura E.34 y triggers Figura E.35 que se han definido sobre ellos, donde se puede ver las dependencias que se han creado entre triggers.

Para poder comparar visualmente el tráfico y los errores de entrada y de salida de cada interfaz, se han definido dos gráficos sobre los elementos a descubrir que representan en el mismo gráfico ambos datos, y dos pantallas que recogen estos gráficos de todos los interfaces de red como se puede observar en la Figura E.44 y en la Figura E.45.

Template SNMP Device

Esta plantilla es una unión de las dos anteriores para poder unir todos sus items para monitorizar los dispositivos de red como switches y routers pero permite mantener la modularidad del sistema ya que se pueden aplicar ambas también por separado.

Template App Zabbix Server

En esta plantilla se han incluido las comprobaciones internas del propio zabbix pertenecientes a las Políticas de Metamonitorización, las cuales se pueden ver en detalle el siguiente enlace [23].

En la Figura E.36 y la Figura E.37 se pueden observar unos ejemplos de como se han definido los elementos y triggers en esta plantilla y las dependencias que se han establecido entre ellos para que cuando se genera una alerta de nivel crítico, desaparezcan las alertas de avisos inferiores.

Para la visualización de la información recogida, se han creado los siguientes gráficos compuestos por diferentes elementos:

- Porcentajes de ocupación de procesos de recogida de datos.
- Porcentajes de ocupación de procesos de gestión interna.
- Porcentajes de espacio libre de cache en diferentes procesos.
- Número de hits y misses en cache.

Estos se representan en una pantalla, junto al número de elementos retrasados que se encuentran en cola y los valores que se procesan por segundo en el servidor de Zabbix. En la Figura E.46 se puede observar un ejemplo.

Script Stats

Se trata de un script que realiza una monitorización remota del servidor Zabbix para obtener información sobre la disponibilidad, el uso de CPU, la memoria y el uso de disco. Este script obtiene la información mediante la ejecución de comandos del sistema y utiliza las herramientas awk y tr para parsearla y almacenarla en los ficheros de log availability.log, CPU.log, memory.log y disk.log junto con la fecha en la que se ha recogido. Estos ficheros almacenan la información durante el último día para comprobar los fallos recientes y siguen las mismas políticas para alertas, pero las notificaciones son enviadas vía email utilizando un servidor SMTP de la empresa mediante la ejecución del siguiente comando:

```
echo $body | mail -S $smtpServer -r fromEmail -s ``subject'' -v
$toEmail
```

Estas comprobaciones se realizan cada 5 minutos como la mayoría de comprobaciones del sistema, para lo que se ha programado una tarea en cron.

El script permite además conocer el estado de qué alertas están activas y cuales no, y permite activar y desactivar las alertas. Si las alertas se encuentran desactivadas, no se envían los emails pero se sigue almacenando la información en los ficheros de log. Por tanto, las funcionalidades que permite el script son las que se muestran en su ayuda:

```
stats
     Save info about Zabbix server in log files and check items for
active alerts
stats status
     Show alert status
stats var status
     Change alert status
     Vars: availability, cpu, memory, disk
     Status: 1 (active), 0 (inactive)
```

Capítulo 6

Pruebas

Para realizar las pruebas en el sistema cuyos resultados se encuentran en Apéndice F, se ha seguido la siguiente metodología:

1. Crear una máquina virtual en el entorno real.
2. Instalar y configurar el agente de Zabbix.
3. Configurar los diferentes elementos a monitorizar en el entorno real sobre esta máquina para realizar pruebas.
4. Comprobar el correcto funcionamiento de dichos elementos.
5. Configurar los elementos en las máquinas reales a monitorizar.
6. Comprobar su funcionamiento.

Inicialmente se ha comprobado para los elementos que se han configurado que la información se recoge y almacena de forma correcta. Para ello, tras configurar cada elemento se accede al apartado *Monitoring - Latest data* y se selecciona el host o grupo para el cual se ha configurado el elemento para poder ver como la información se está recogiendo correctamente. En este caso se puede observar en la Figura F.1 como se está aplicando la plantilla de monitorización básica para un host y se está recogiendo toda su información adecuadamente. También se ha comprobado como se recoge la información correctamente mediante SNMP para los elementos de red que se monitorizan como se puede observar en la Figura F.2.

También se ha comprobado que la información solo se almacena durante el periodo establecido. Esto se puede observar en la Figura F.3 ya que se ha modificado la configuración del elemento que recoge información cada 24 horas para que almacene la información durante un periodo de un día, y aunque el sistema lleva más tiempo monitorizándose solo aparece la información recopilada durante el último día dado que el resto ha sido eliminada.

Otras pruebas importantes son la comprobación de la creación de notificaciones en el panel de problemas cuando se producen fallos y la realización de las acciones programadas cuando se detecta un fallo. En este caso se puede observar en la Figura F.4 como se ha creado un aviso al detectar el fallo y además como este trigger tiene una acción asociada, se ejecuta. En el primer caso se observa como la acción que es la ejecución de un comando remoto ha solucionado el problema, por lo que el problema ha pasado a estado Resuelto. En el segundo caso se puede ver en la Figura F.5 como la primera acción no ha conseguido solucionar el problema, por tanto se ejecuta un segundo paso para escalar el problema que envía un correo Figura F.6 al administrador para avisar de la persistencia del problema y que este lo pueda resolver.

También se ha comprobado como los hosts son autodescubiertos por el servidor de Zabbix inicialmente, por tanto estos pertenecen automáticamente al grupo de Discovered hosts y además se ha configurado para que se le aplique la plantilla básica como se ha explicado en la Subsección 5.1.2, por tanto también aparece desde el principio como se observa en la Figura F.7.

Para comprobar que la metamonitorización externa funciona, se ha comprobado que la información se almacena correctamente cada 5 minutos y que el resultado coincide con la información recogida por el propio servidor, como se puede observar en Sección F.

También se ha comprobado que los emails se envían correctamente tras provocar que el sistema no esté disponible y al superar diferentes umbrales, que se han modificado para forzar las comprobaciones. Estas pruebas se pueden observar en la Figura F.12.

Capítulo 7

Conclusiones

Este proyecto presenta una solución a la monitorización del entorno específico basada en la arquitectura de los sistemas de la empresa IA Soft, grupo OESIA. Para conseguir desarrollar este sistema, ha sido necesario estudiar el entorno en el que se va a desarrollar, los fundamentos en los que se basan los sistemas de monitorización y las herramientas que existen actualmente. Tras este estudio y junto con los requisitos del sistema, se ha considerado que la herramienta que mejor se ajustaba es Zabbix. Tras haber estudiado el entorno y sus elementos críticos, se ha realizado el diseño de la arquitectura y de las políticas para poder implementarlas a continuación en Zabbix.

Cabe destacar que el desarrollo del proyecto se ha podido utilizar la funcionalidad de los templates en Zabbix para implementar las políticas definidas, lo que ha facilitado la modularidad del sistema. Otra ventaja es la facilidad que ofrece para modificar elementos partiendo de templates ya que esta funcionalidad permite modificar fácilmente los umbrales en algunos casos concretos.

La funcionalidad de autodescubrimiento que ofrece Zabbix, tanto para los hosts en los que se instalan los agentes a los que se les pueden asignar plantillas según las acciones definidas, como para los elementos a monitorizar y las opciones para modificar sus reglas permitiendo añadir expresiones regulares para añadir o eliminar elementos que se descubren también, aportan un ahorro de tiempo a la hora de configurar y mantener el sistema ya que permite la automatización de tareas. Aunque también supone una complejidad extra ya que hay que tener especial cuidado con las reglas definidas para añadir templates y para añadir o eliminar elementos porque se aplican a todos los elementos del sistema.

También se considera una ventaja la posibilidad de ejecución de comandos remotos y de creación de scripts e integración de los mismos en el servidor central, ya que permite monitorizar elementos que no se encuentran predefinidos entre los elementos de monitorización de Zabbix.

Sin embargo, a pesar de ser la herramienta que mejor se ajustaba a los requisitos, cabe destacar que no se ha conseguido satisfacer completamente el requisito referente a la generación de informes ya que en este aspecto, Zabbix presenta dificultades en la presentación de los datos. Esto se debe a que presenta un estado general del sistema y la disponibilidad de cada host dividida por los elementos monitorizados pero no se puede exportar esta información. Para resolver este problema es necesario capturar cada salida que se desea presentar en los informes, tanto los porcentajes de disponibilidad como las gráficas de evolución para poder analizarlas.

7.1. Trabajo futuro

A partir de este proyecto, se debe continuar implantando el sistema para reemplazar por completo el sistema anterior monitorizando todos los hosts restantes, ya que el proyecto solo se trata de un prototipo. También permite debido a la modularidad ofrecida por las políticas definidas, añadir los nuevos elementos que se vayan incorporando al sistema e incluso se podría llegar a utilizar como sistema de monitorización para todas las sedes utilizando el servidor central de Zabbix para controlar la monitorización de los diferentes elementos pero utilizando diferentes proxies para las distintas sedes que se encargaran de recolectar la información y enviarla al servidor central.

También se podría crear un sistema de automatización para permitir automatizar el proceso de puesta en marcha de los elementos a monitorizar. Este tipo de sistemas permiten crear una arquitectura cliente-servidor para definir los estados en los que se quiere que se encuentren los diferentes clientes. De este modo, se podría definir un estado para automatizar tanto la instalación de los nuevos hosts que se quieran incorporar al sistema de monitorización independientemente de sus sistemas operativos, como la actualización a las nuevas versiones de Zabbix de todos los hosts pertenecientes al sistema para evitar que el sistema quede desactualizado y así poder incluir las nuevas funcionalidades que se desarrollan. Algunos de los sistemas de automatización que destacan hoy en día son Puppet [24] y Chef [25].

Capítulo 8

Bibliografía

[1] Wikipedia the free encyclopedia. System monitoring. https://en.wikipedia.org/wiki/System_monitoring.

[2] GRUPO OESÍA 2017. Grupo oesia. http://grupooesia.com/.

[3] Wikipedia the free encyclopedia. Rrdtool. https://en.wikipedia.org/wiki/RRDtool.

[4] Nagios Team. Nagios core documentation. https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/toc.html.

[5] Zabbix SIA. Zabbix documentation. https://www.zabbix.com/documentation/3.2/.

[6] Elasticsearch. The open source elastic stack. https://www.elastic.co/products.

[7] Icinga Team. Icinga2 documentation. https://docs.icinga.com/icinga2/latest/doc/module/icinga2/toc.

[8] Inc. Sensu. What is sensu? https://sensuapp.org/docs/latest/overview/what-is-sensu.html.

[9] Equipo de documentación de Pandora FMS. Pandora: Documentation. http://wiki.pandorafms.com/index.php?title=Pandora:Documentation.

[10] Shinken Team. Welcome to shinken's documentation! http://shinken.readthedocs.io/en/latest/.

[11] Wikipedia the free encyclopedia. Comparison of network monitoring systems. https://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems.

[12] Dave Josephsen. *Monitoring Taxonomy, Laying put the tools landscape.* O'Reilly, 2017.

[13] Wikipedia the free encyclopedia. Internet control message protocol. https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol.

[14] Wikipedia the free encyclopedia. Simple network management protocol. https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol.

[15] Thomas A. Limoncelli, Christina J. Hogan, and Strata R. Chalup. *The Practice of System and Network Administration.* Addison Wesley, 2007.

[16] Zabbix SIA. Item types. https://www.zabbix.com/documentation/3.2/manual/config/items/itemtypes.

[17] Oracle Corporation and/or its affiliates. Benefits of using innodb tables. https://dev.mysql.com/doc/refman/5.7/en/innodb-benefits.html.

[18] Melissa Anderson. How to install mysql on centos 7. https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-centos-7.

[19] Zabbix. Docs/howto/ssl certificate check. https://www.zabbix.org/wiki/Docs/howto/ssl_certificate_check.

[20] Oracle Corporation and/or its affiliates. Mysql 5.7 reference manual - server status variables. https://dev.mysql.com/doc/refman/5.7/en/server-status-variables.html.

[21] Steinar Andersen. Sql server 2005 - 2012 (multi instance). https://share.zabbix.com/databases/microsoft-sql-server/sql-server-2005-2012-multi-instance.

[22] Microsoft. Supervisar el uso de recursos (monitor de sistema). https://docs.microsoft.com/es-es/sql/relational-databases/performance-monitor/monitor-resource-usage-system-monitor.

[23] Zabbix SIA. Internal checks. https://www.zabbix.com/documentation/3.2/manual/config/items/itemtypes/internal?s[]=zabbix&s[]=items.

[24] Puppet. Puppet. https://puppet.com.

[25] Inc Chef Software. Achieve speed, scale, and consistency by automating your infrastructure with chef. https://www.chef.io/chef/.

[26] Sam Newman. *Lightweight Systems for Realtime Monitoring.* O'Reilly, 2014.

[27] Preetam Jinka and Baron Schwartz. *Anomaly Detection for Monitoring.* O'Reilly, 2015.

[28] Stawek Ligus. *Effective Monitoring & Alerting.* O'Reilly, 2013.

[29] James Turnbull. *The Art of Monitoring.* 2016.

[30] Rob Ewaschuk. *Monitoring Distributed Systems.* O'Reilly, 2016.

Anexos

Anexos A

Gestión del tiempo

El proyecto se ha desarrollado a lo largo de 6 meses en la empresa IA Soft invirtiendo en total alrededor de 380 horas. A continuación se puede observar el Diagrama de Gantt en Figura A.1 donde se puede observar el tiempo invertido dividido por semanas y desglosado en las tareas realizadas.

Las diferentes fases del proyecto se dividen en análisis, donde se ha tenido que estudiar como funcionan los sistemas de monitorización y las diferentes herramientas existentes a día de hoy, la fase de diseño donde se ha diseñado tanto la arquitectura del sistema como las políticas que se iban a aplicar, la fase de implementación, la fase de pruebas donde se ha comprobado que cada elemento configurado funciona correctamente y la fase de documentación.

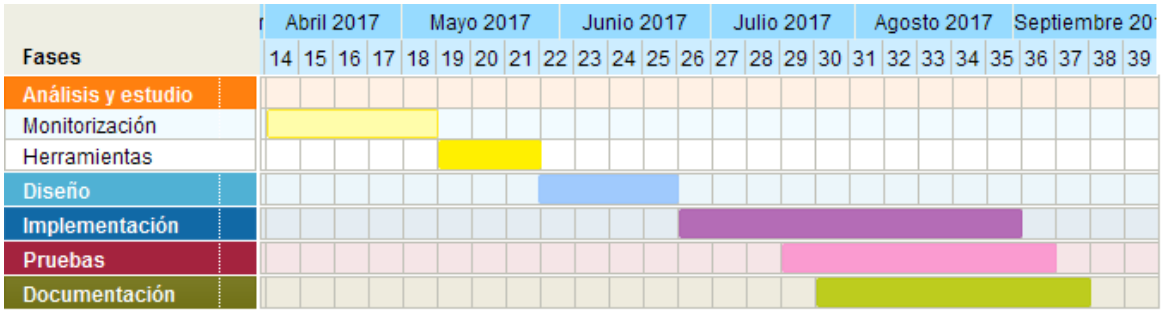

Figura A.1: Diagrama de Gantt con esfuerzo invertido por semana

Anexos B

Detalles del análisis de herramientas

Se puede observar como los requisitos 1, 2, 3, 4, 5, 9, 10 y 11 los cumplen todas las herramientas ya que se trata de elementos de monitorización básicos.

Para el requisito funcional 6 se ha observado como todas las herramientas seleccionadas permiten recoger los logs del sistema para analizarlos de diferentes formas, sin embargo, para Nagios esta funcionalidad solo esta disponible en la versión de pago.

Para el requisito funcional 7, todas las herramientas lo cumplen permitiendo la integración de la visualización gráfica junto con la parte de gestión, sin embargo, en algunas de ellas como Sensu y Shinken supone realizar la integración de la propia herramienta con otras como Graphite o Grafana. Esto supone un incremento en la dificultad a la hora de construir el sistema debido a la integración de diferentes herramientas.

Para el requisito funcional 8 es necesario tener en cuenta el tipo de base de datos que utiliza cada herramienta ya que la mayoría utilizan bases de datos del tipo Round-Robin Database (RRD) por estar orientadas al almacenamiento de información de series temporales que se utilizan para guardar un histórico de las métricas recogidas del sistema y poder visualizar y analizar a posteriori los eventos que han ocurrido en el sistema. Sin embargo, el principal inconveniente de este tipo de bases de datos es la dificultad para definir diferentes rangos para el almacenamiento de diferentes datos. En este aspecto, sobre Zabbix se ha encontrado una información mucho más clara ya que al utilizar MySQL o PostgreSQL, ambas permiten personalizar la información que se puede guardar, y esto es una de las grandes ventajas que presenta Zabbix ya que permite definir a nivel global y de cada elemento durante cuando tiempo se quiere guardar su información concreta y durante cuanto tiempo se quiere guardar las tendencias de sus datos.

Para el requisito funcional 12 se puede observar como todas lo permiten aunque Shinken requiere de un programa de pago para ello. A pesar de esto, los informes que aportan cada herramienta son muy distintos y es difícil realizar una comparación de ellos. También una buena opción para la realización de los informes consiste en analizar correctamente los gráficos que reporta la herramienta para poder estudiar a partir de ahí la disponibilidad y otros aspectos relevantes.

Para el requisito funcional 13 se puede observar como todas disponen de *acknowledgments* para permitir retroalimentacion por parte de los diferentes usuarios del sistema de monitorización a cada evento para informar de su situación si este está siendo reparado para que el resto de usuarios lo tengan en cuenta y no haya dos usuarios realizando la misma tarea al mismo tiempo.

Para el requisito funcional 14 se puede observar que todos lo permiten pero en Pandora FMS y Nagios algunos aspectos suponen la personalización manual en cada cliente, lo que dificulta el proceso de configuración de los clientes ya que no permiten aplicar a varios la misma configuración al mismo tiempo.

Finalmente, el requisito funcional 15 es subjetivo ya que la forma de visualización depende de cada usuario. En este caso concreto se ha decidido que aquellos que tenían un interfaz más intuitivo son Zabbix, Pandora e Icinga2.

Anexos C

Cálculo del espacio ocupado

Para calcular de forma aproximada el tamaño total que va a ocupar la información del sistema, es necesario establecer las siguientes restricciones:

- Cada comprobación realizada por Zabbix que almacena valores numéricos ocupa 90B.
- Como las comprobaciones numéricas superan a las que recogen información textual, de la cual además no se pueden realizar las tendencias ni se sabe cuanto van a ocupar ya que varía, estas no se van a tener en cuenta en estos cálculos y se va a aproximar suponiendo que todos los datos son numéricos.
- Las comprobaciones, a nivel general, se van a realizar cada 5 minutos = 300 segundos.
- Se considera que, de media, cada elemento a monitorizar consta de 30 comprobaciones en el periodo de 5 minutos.

Partiendo de estas premisas, a continuación se calculan los elementos totales aproximados que se van a monitorizar:

- 138 Switches
- 18 Firewalls
- 3 Routers
- 65 Puntos de acceso WiFi
- 417 Máquinas virtuales
- 35 Hosts

En total se quieren monitorizar alrededor de 676 elementos. Teniendo en cuenta que, aproximadamente cada elemento va a contar con unas 30 comprobaciones distintas:

$$676\ elementos\ x\ 30\ checks\ =\ 20280\ checks\ totales$$

$$20280\ checks\ /\ 300\ segundos\ =\ 67{,}6\ checks/segundo$$

Se calcula lo que ocuparía la información si se guardara de forma total durante 2 años completos:

$$(730\ dias\ x\ 24\ h\ x\ 3600\ s)\ x\ 67{,}6\ checks/s\ x\ 90\ B\ =\ 386\ x\ 10^9\ B\ /\ 1024^3\ =\ 383{,}7\ GB$$

Se calcula lo que ocuparía la información si se guardara de forma total durante 3 meses y las tendencias hasta llegar al periodo de dos años:

$$(90\ dias\ x\ 24\ h\ x\ 3600\ s)\ x\ 67{,}6\ checks/s\ x\ 90\ B + (640\ dias\ x\ 24\ h)\ x\ 20280\ checks\ x\ 90\ B\ =$$

$$(47{,}3\ +\ 28{,}03)\ x\ 10^9\ B\ =\ 75{,}34\ x\ 10^9\ B\ /\ 1024^3\ =\ 70{,}17\ GB$$

Finalmente, se calcula el porcentaje que se ahorra almacenando información durante el mismo periodo, guardando tendencias en lugar de la información completa:

$$100\ -\ (\frac{70{,}17}{383{,}7} x\ 100)\ =\ 81{,}7\,\%$$

Anexos D

Configuración en Zabbix

Para definir una plantilla en Zabbix es necesario ir a *Configuration - Templates - Create template*. Cada plantilla permite definir los elementos explicados a continuación:

- **Applications**: Son los diferentes grupos/tipos en los que se dividen los elementos que se van a monitorizar.
- **Items**: Elementos que recogen información para la monitorización del sistema.
- **Triggers**: Disparadores/alertas que se generan cuando se cumple la expresión que se especifica. Estos se asocian a un item y se pueden establecer dependencias entre ellos de tal forma que si se producen fallos que desencadenan otros, se eviten alertas en cascada.
- **Graphs**: Permite definir gráficos compuestos de varias métricas para después analizar la información recogida y poder mostrarlos en las pantallas asociadas a cada host.
- **Screens**: Permite definir diferentes pantallas con los gráficos combinados que se han generado, o con gráficos simples de los elementos establecidos.
- **Discovery rules**: Permite definir reglas para descubrir elementos de forma automática para el host. También se permite definir triggers y gráficos sobre estos.
- **Web scenarios**: Define un escenario web principal sobre el que se van a realizar diferentes comprobaciones mediante diferentes pasos, por ejemplo, que el acceso a la web se puede realizar correctamente, que se puede realizar el login o una acción y esto devuelve el resultado esperado.

D.1. Elementos

La configuración de los items que se encargan de recoger la información a monitorizar consta de los elementos que aparecen en la siguiente Figura D.1.

Cabe destacar que existen diferentes tipos en los elementos que se desean monitorizar, donde los más utilizados en ese sistema son:

- **Simple checks**: Realiza consultas directamente sobre los hosts sin necesidad de instalar el agente. En la documentación [5] se encuentran las comprobaciones que se pueden realizar con sus correspondientes claves y parámetros.
- **SNMP agent**: Este tipo permite monitorizar dispositivos de red realizando sobre ellos consultas SNMP consultando sus correspondientes MIBs sin necesidad de instalar el agente de Zabbix. Para este tipo de consulta es necesario definir los siguientes parámetros Figura D.2.
- **Zabbix agent**: La mayoría de la información se recoge de esta forma, donde es el servidor quien realiza la petición de información sobre el agente de Zabbix de cada host, y este le envía la información para que el servidor la procese y la almacene. En la documentación [5] se encuentran las diferentes claves que existen con los parámetros que estas necesitan.
- **Zabbix agent active**: Este tipo de recogida de la información es más costoso ya que es el agente Zabbix quien tiene que preguntar al servidor por la lista de consultas que tiene que realizar (se configura en fichero de configuración de cada host cada cuanto se realiza), y cuando recibe la información comienza a realizar los chequeos y enviar la información correspondiente cada intervalo de tiempo definido.

Además de estos elementos, también es posible crear elementos calculados a partir de otros, monitorizar ficheros de log, realizar chequeos mediante ssh, telnet, ipmi y otros que se encuentran en el siguient enlace [16].

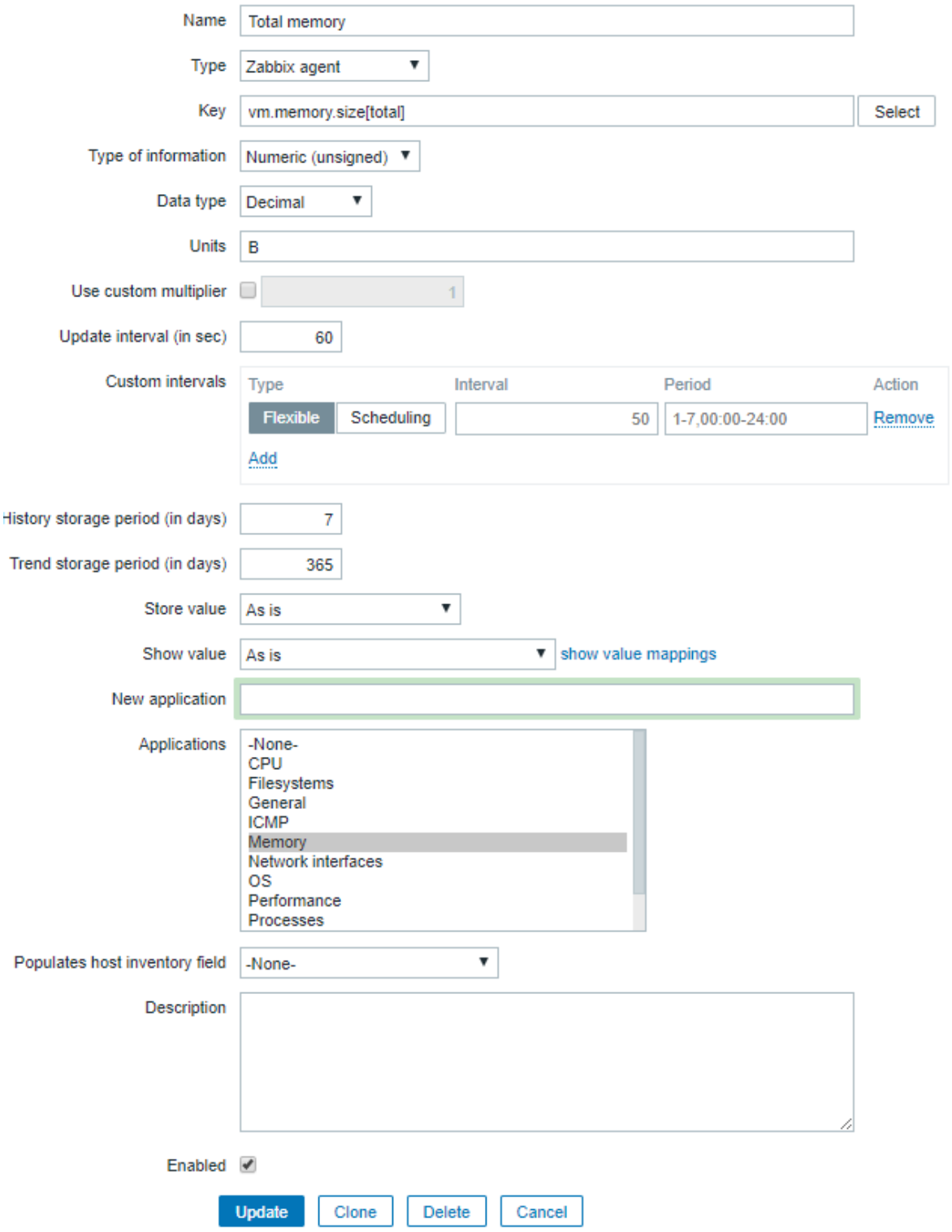

Figura D.1: Elementos de configuración de un item

Figura D.2: Parámetros de configuración de SNMP agent

Otro de los elementos relevantes es la personalización del periodo durante el cual se desea monitorizar. En este caso se ha decidido monitorizar todos los elementos 24/7 ya que se desea detectar posibles fallos en todo momento pero en otros casos podría resultar interesante reducir la cantidad de información recogida en momentos innecesarios.

La forma en la que se puede almacenar cada dato puede adaptarse al tipo de dato, siendo tal y como se recoge, como la diferencia entre el valor anterior y el actual, o como la diferencia entre ambos entre el tiempo que ha pasado para calcular la velocidad. Y para la forma en la que se muestra, existen los siguientes mapeos de valores predefinidos según diferentes aplicaciones Figura D.3 donde se relaciona el dato recogido con el estado que este representa, y se pueden añadir nuevos mapeos a esta lista desde *Administration - General - Value mapping*.

D.2. Triggers

Para los triggers se permite definir los elementos que se distinguen en la Figura D.4. Entre ellos destacan:

- **Nivel**: La importancia de la alerta generada, que se divide en 6 niveles.
- **Expresión**: Es el elemento más importante del trigger ya que cuando esta expresión se cumple, se genera la alerta. Se define sobre un elemento y para crear la expresión existen diferentes modos que se pueden observar en la documentación [5].
- **Recuperación**: La recuperación puede realizarse cuando se deje de cumplir dicha condición, cuando se cumpla otra expresión definida como *recovery expressión*, o de ninguna forma.
- **Modo de generación de eventos**: En general, se definirá como simple ya que si es múltiple generará un nuevo evento cada vez que se detecte el mismo fallo.
- **Evento de cierre**: Puede cerrar todos los problemas o solo si los tags coinciden.
- **Tags**: Permiten etiquetar elementos a nivel de trigger para aportar más flexibilidad como asociar acciones en función de los tags. En este caso no se han utilizado.

- **Permitir cierre manual**: Por defecto, se marcará para permitir que el administrador pueda cerrarlo cuando desee.

Maintenance status	0 ⇒ normal 1 ⇒ in maintenance 2 ⇒ no data collection
MySQL - Status	0 ⇒ No 1 ⇒ Yes
Service state	0 ⇒ Down 1 ⇒ Up
SNMP device status (hrDeviceStatus)	1 ⇒ unknown 2 ⇒ running 3 ⇒ warning 4 ⇒ testing 5 ⇒ down
SNMP interface status (ifAdminStatus)	1 ⇒ up 2 ⇒ down 3 ⇒ testing
SNMP interface status (ifOperStatus)	1 ⇒ up 2 ⇒ down 3 ⇒ testing 4 ⇒ unknown 5 ⇒ dormant 6 ⇒ notPresent 7 ⇒ lowerLayerDown
Value cache operating mode	0 ⇒ Normal 1 ⇒ Low memory
VMware status	0 ⇒ gray 1 ⇒ green 2 ⇒ yellow 3 ⇒ red
VMware VirtualMachinePowerState	0 ⇒ poweredOff 1 ⇒ poweredOn 2 ⇒ suspended
Windows service startup type	0 ⇒ Automatic 1 ⇒ Automatic delayed 2 ⇒ Manual 3 ⇒ Disabled 4 ⇒ Unknown
Windows service state	0 ⇒ Running 1 ⇒ Paused 2 ⇒ Start pending 3 ⇒ Pause pending 4 ⇒ Continue pending 5 ⇒ Stop pending 6 ⇒ Stopped 7 ⇒ Unknown 255 ⇒ No such service
Zabbix agent ping status	1 ⇒ Up

Figura D.3: Mapeo según aplicación valor-estado

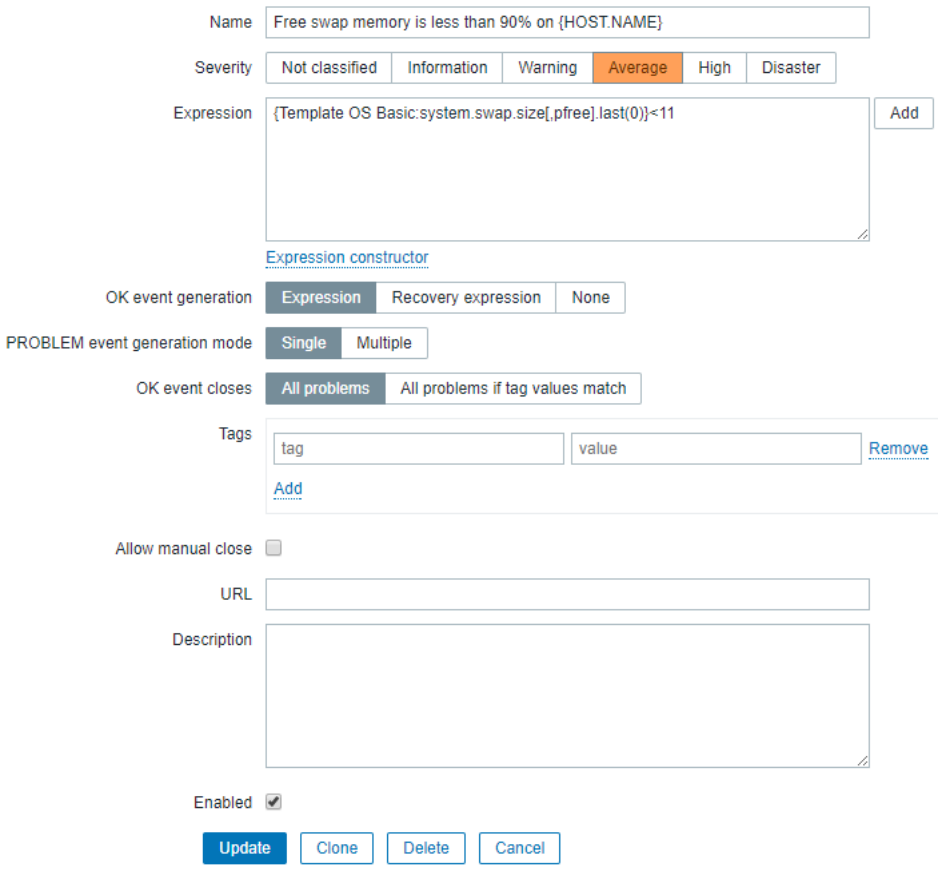

Figura D.4: Elementos de configuración de un trigger

D.3. Gráficos

En la siguiente Figura D.5 se pueden observar los elementos de configuración de los gráficos.

Cabe destacar que se permite definir 4 tipos de gráficos, de líneas que marcan la evolución de las métricas, de líneas con el área rellena, de porciones, y de porciones separadas. Para la mayoría de información los gráficos utilizados son de los dos primeros tipos ya que estos permiten ver el histórico mientras que los otros solo permiten ver la información en un momento concreto y en este caso se desea poder analizar la información histórica. También se permite fijar los valores máximos y mínimos a mostrar de forma fija, calculada o en función de elementos monitorizados.

D.4. Pantallas

Se permite definir varias pantallas donde se van a mostrar los elementos monitorizados en la disposición que se quiera, formando filas y columnas y ajustando el ancho tanto de las pantallas como de los elementos que se encuentran en ellas como se quiera, ajustando los parámetros mostrados en la Figura D.6 para cada elemento.

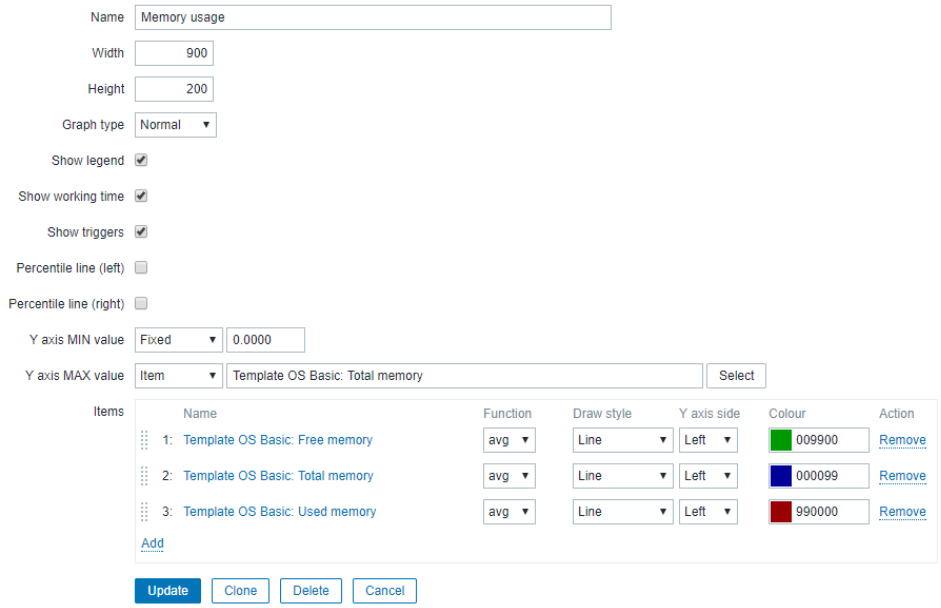

Figura D.5: Elementos de configuración de un gráfico

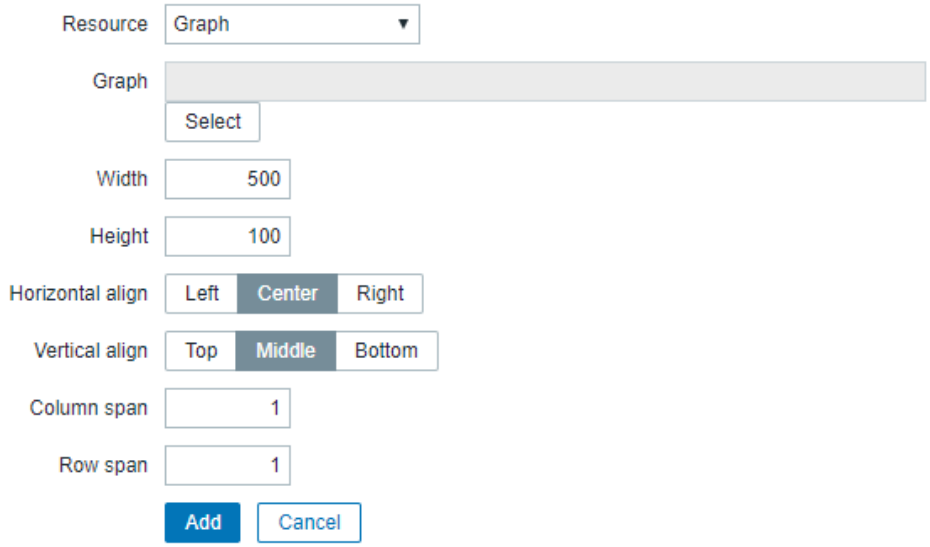

Figura D.6: Elementos de configuración de una pantalla

Además, los elementos gráficos que se pueden representar en las pantallas y se pueden mostrar son:

- **Gráficos simples**: Generados automáticamente a partir de la información recogida de cada item.
- **Gráficos**: Gráficos que se han definido como se ha indicado en el apartado anterior Sección D.3.
- **Prototipos de gráficos simples**: Generados automáticamente a partir de los elementos que se descubren a partir de las reglas definidas, como se explica en la Sección D.5.

- **Prototipos de gráficos**: Gráficos que se han definido para los elementos que se descubren, como se explica en la Sección D.5.
- **Texto**: Información recogida de los items definidos pero de forma textual en vez de en gráficos.
- **URL**: URL introducida para crear, por ejemplo, un enlace directo a otra pantalla de monitorización o a una web concreta que se está monitorizando.

D.5. Reglas descubrimiento

Para facilitar la tarea de monitorización, Zabbix proporciona un mecanismo que permite descubrir automáticamente algunos de los elementos de monitorización. Estos elementos son:

- Sistemas de ficheros.
- Interfaces de red.
- SNMP OIDs.
- CPUs y cores.
- ODBC SQL queries.
- Servicios de Windows.

En este sistema, se han utilizado los tres primeros descubrimientos en plantillas como se ha explicado anteriormente.

Para definir este tipo de monitorización, solo se necesita conocer la clave necesaria para cada tipo de descubrimiento, y después sobre él definir los distintos items, triggers, gráficos o hosts que constan de los mismos elementos que se han ido explicando, excepto que para la monitorización en lugar de poner en nombre concreto, hay que utilizar una macro que posteriormente se completará con el nombre del elemento descubierto. En la documetación se encuentra información más detallada y algunos ejemplos [5].

D.6. Escenarios web

Para la monitorización de webs, se permite definir un escenario con los elementos que se ven en la Figura D.7, donde se pueden incluir variables para utilizarlas posteriormente en los diferentes pasos.

Tras definir el escenario, se definen los diferentes pasos que se desean comprobar Figura D.8, por ejemplo el correcto funcionamiento de diferentes webs o el login Figura D.9 en ellas mediante la utilización de variables de usuario y contraseña.

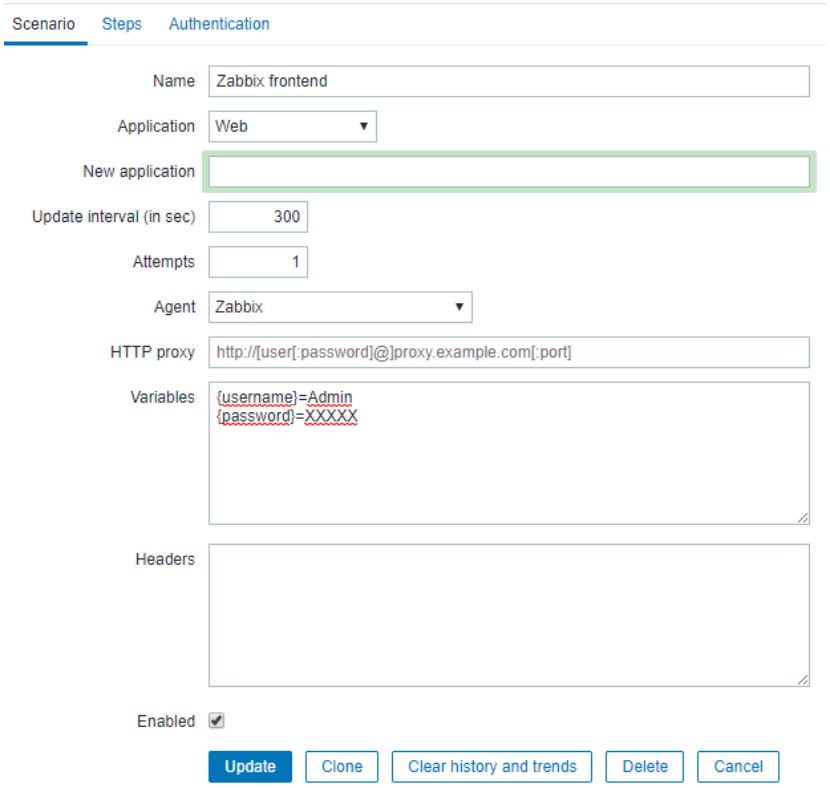

Figura D.7: Elementos de configuración de un escenario web

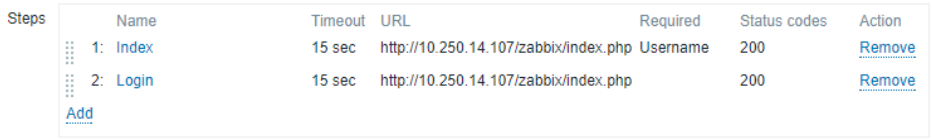

Figura D.8: Pasos definidos en monitorización web

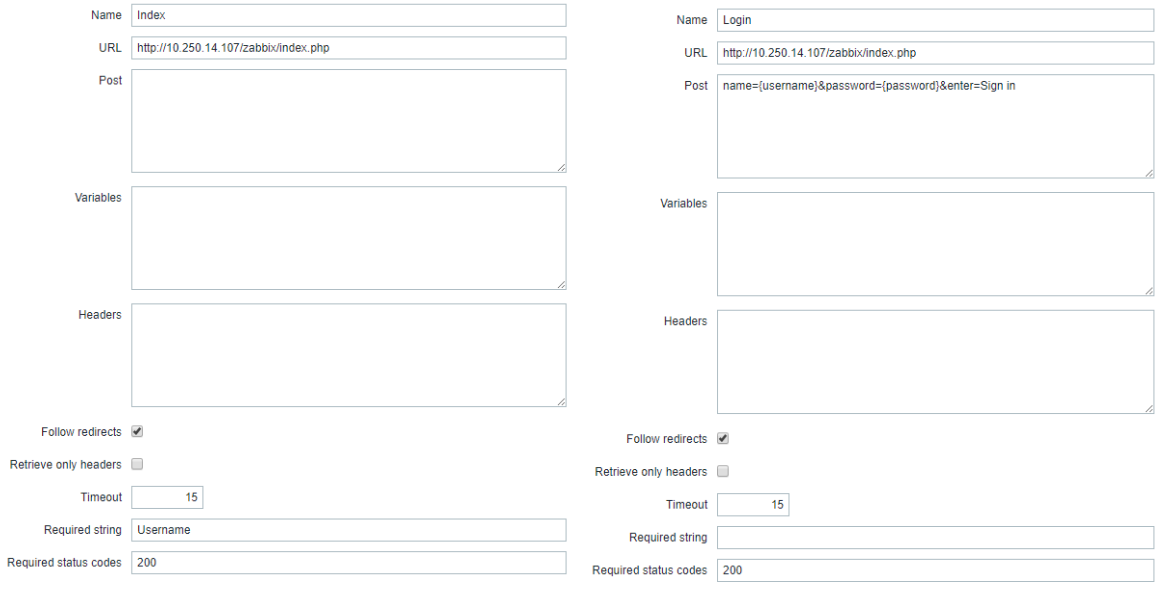

Figura D.9: Elementos de configuración de pasos en escenarios web

D.7. Scripts externos

Para ejecutar scripts externos a Zabbix que se encuentran en el servidor es necesario ubicar el script a ejecutar en el servidor de Zabbix en el directorio */usr/lib/zabbix/externalscripts/* y asignarle todos los permisos para que pueda ser ejecutado remotamente. Para monitorizar los elementos del script se deberá crear un item con el nombre del script y los parámetros que se desea pasar entre corchetes de la siguiente forma: *nombreScript[par1,par2..]*

D.8. Escalado de problemas

El escalado de problemas que se puede utilizar para avisar a un usuario o administrador del sistema cuando el problema es muy grave, cuando perdura en el tiempo sin solucionarse o cuando se ha intentado solucionar automáticamente pero no se ha resuelto, en Zabbix se puede implementar mediante la definición de acciones en *Configuration - Actions - Triggers*. También se ofrece el mecanismo para realizar acciones cuando se ha resuelto un problema utilizando Recovery operations. Los pasos que se deben seguir para su implementación se explican en la Subsección D.11.2.

D.9. Mantenimiento

Para ejecutar tareas de mantenimiento en los hosts, se pueden establecer periodos de mantenimiento en el servidor de monitorización para cada hosts desde *Configuration - Maintenance*. De esta forma se permite seleccionar si en este periodo va a seguir recogiendo la información del hosts Figura D.10 y por tanto, disparando los triggers si se encuentran fallos para poder analizarlos a posteriori sabiendo que en este periodo se han realizado cambios en el hosts pero se quiere ver como se ha comportado ante estos, o si por el contrario se va a anular la recogida de datos ya que se trata de periodos de mantenimiento donde se pueden producir fallos y dar falsas alertas, y se pretende evitar este comportamiento. Además permite definir subperiodos dentro del periodo principal Figura D.11 y los hosts o grupos a los que afecta Figura D.12.

Figura D.10: Definición de configuración principal del periodo de mantenimiento

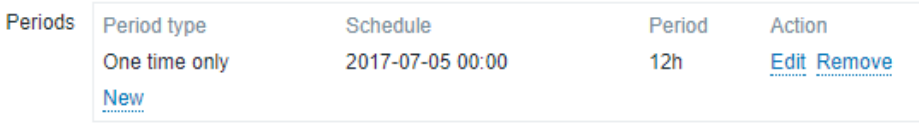

Figura D.11: Definición de periodos de mantenimiento

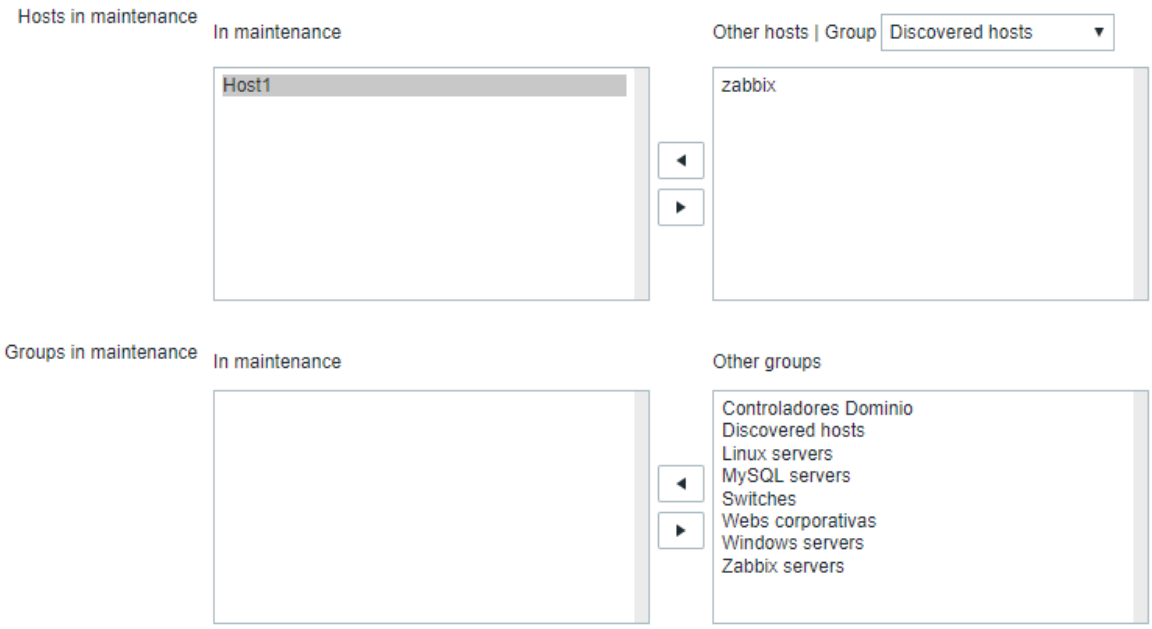

Figura D.12: Hosts o grupos a los que afecta el mantenimiento

D.10. Plantillas

En la Figura D.13 se observan los elementos básicos de configuración de un template, donde se le asigna un nombre, los grupos a los que pertenece, los hosts a los que se se asigna y una breve descripción. En la siguiente Figura D.14 se puede relacionar la plantilla con otras, de tal forma que al aplicar esta plantilla, se aplicarán también las que han sido relacionadas con esta. A continuación en Figura D.15 se observa donde definir las macros que se aplican a los elementos de dicha plantilla en concreto.

Después se pueden observar los diferentes elementos que forman parte de una plantilla y que se ha explicado anteriormente en detalle como configurarlos. Dichos elementos son las aplicaciones Figura D.16 que se han definido en la plantilla y cuantos elementos se le ha asignado a cada una, los elementos definidos Figura D.18 junto con su información principal como los triggers que tienen asociados, la clave que se usa para recoger la información, la frecuencia con la que se recoge la información, durante cuánto se guarda el histórico y las tendencias, el tipo de chequeo, la aplicación/es a la que pertenece y su estado, es decir, si está activada o no y los posibles filtros Figura D.17 que se pueden aplicar para la búsqueda de dichos elementos.

También se pueden ver los triggers definidos Figura D.19 junto a su estado, los gráficos Figura D.20 compuestos por los datos recogidos pertenecientes a distintos elementos, las pantallas Figura D.21 donde se representan los distintos gráficos para poder observar en la misma ventana todas las gráficas y datos relevantes de un host y las reglas de descubrimiento Figura D.22 junto a los elementos, triggers y gráficos asociados a los datos recogidos por estas.

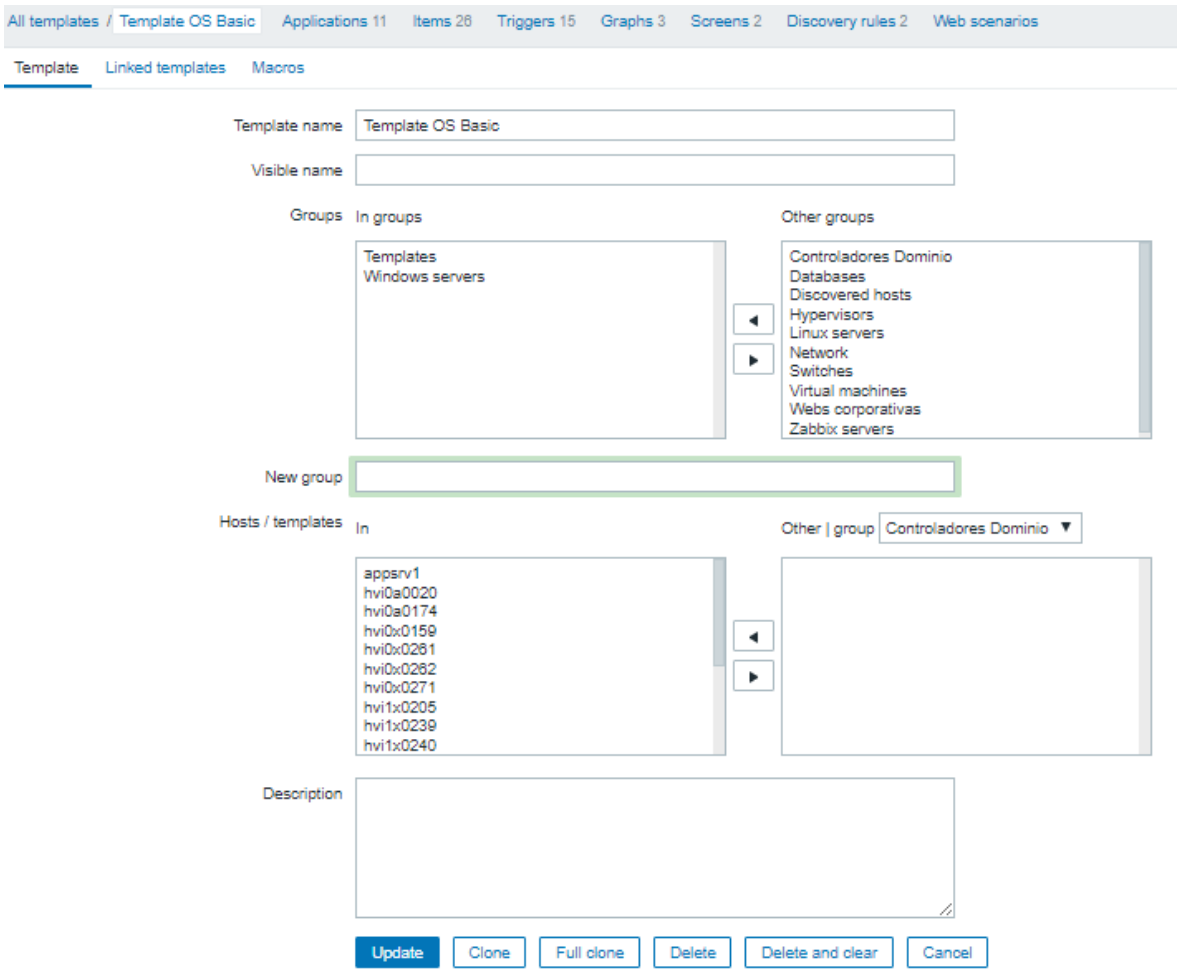

Figura D.13: Aspectos básicos de una plantilla

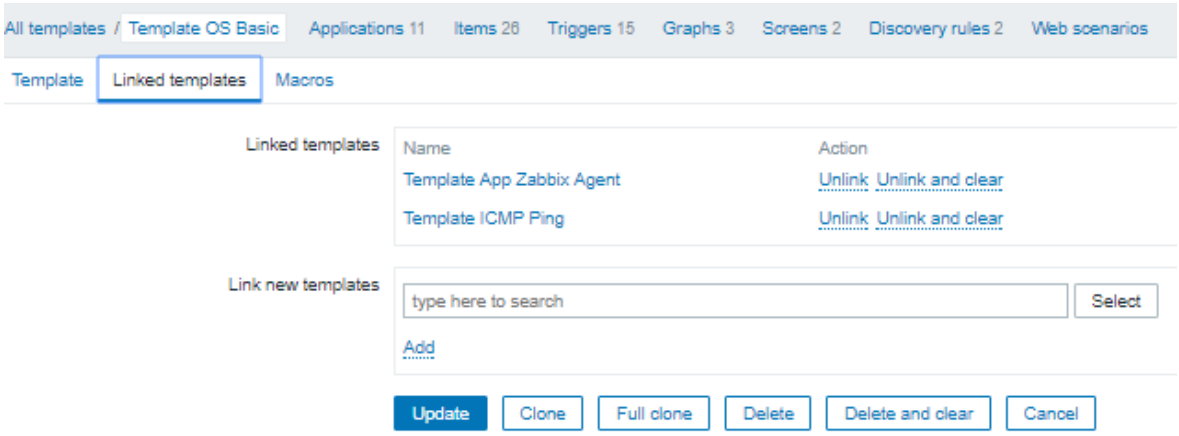

Figura D.14: Relación de plantillas

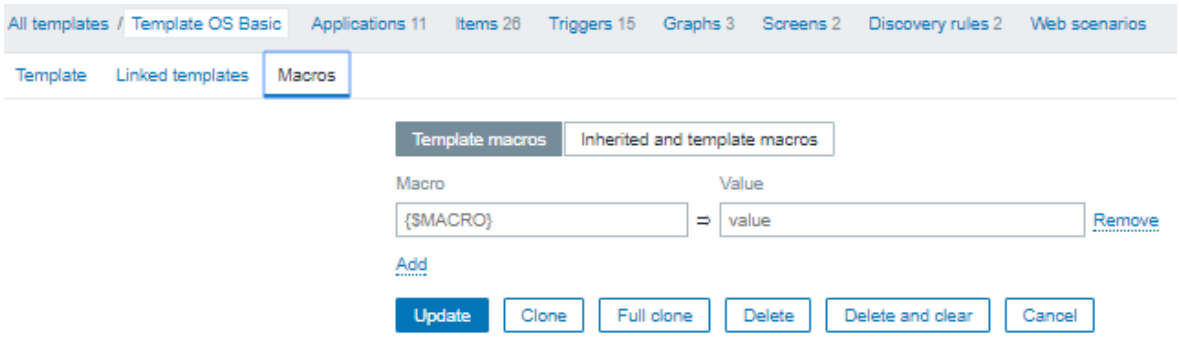

Figura D.15: Definición de macros

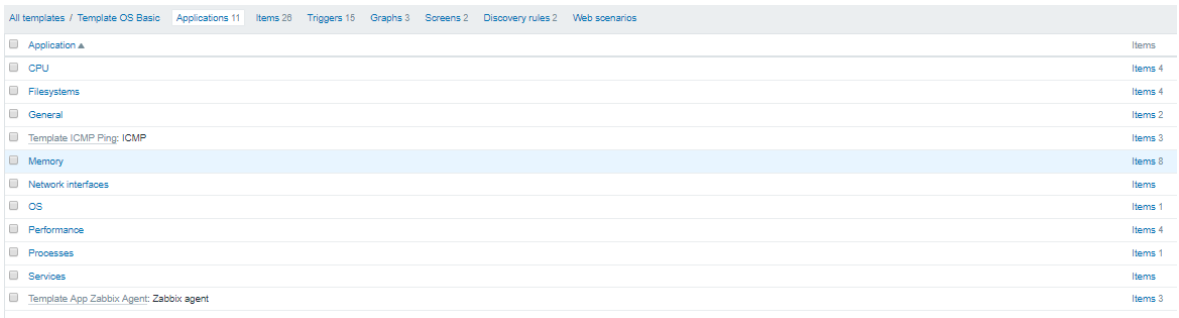

Figura D.16: Aplicaciones definidas

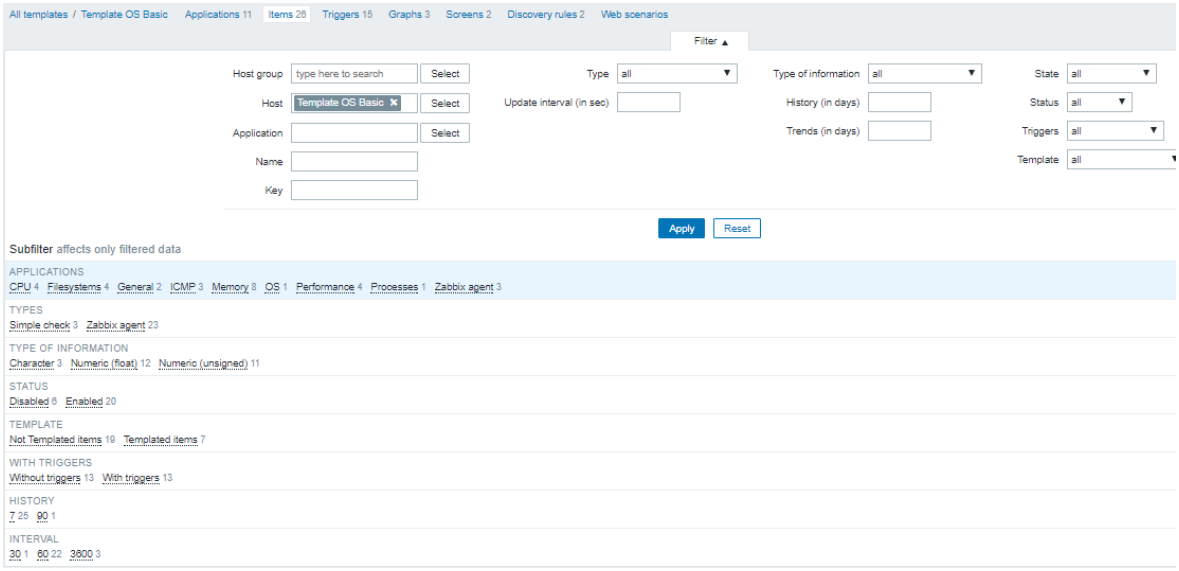

Figura D.17: Campos de filtrado y búsqueda de elementos

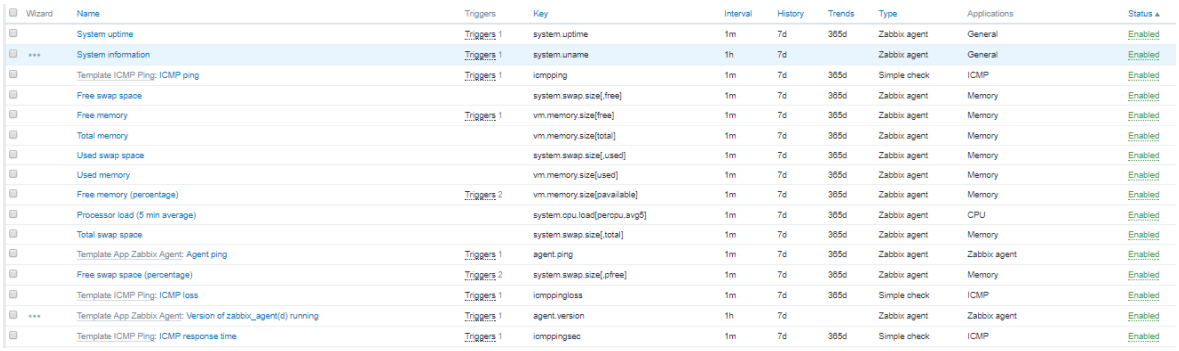

Figura D.18: Elementos y características principales de estos

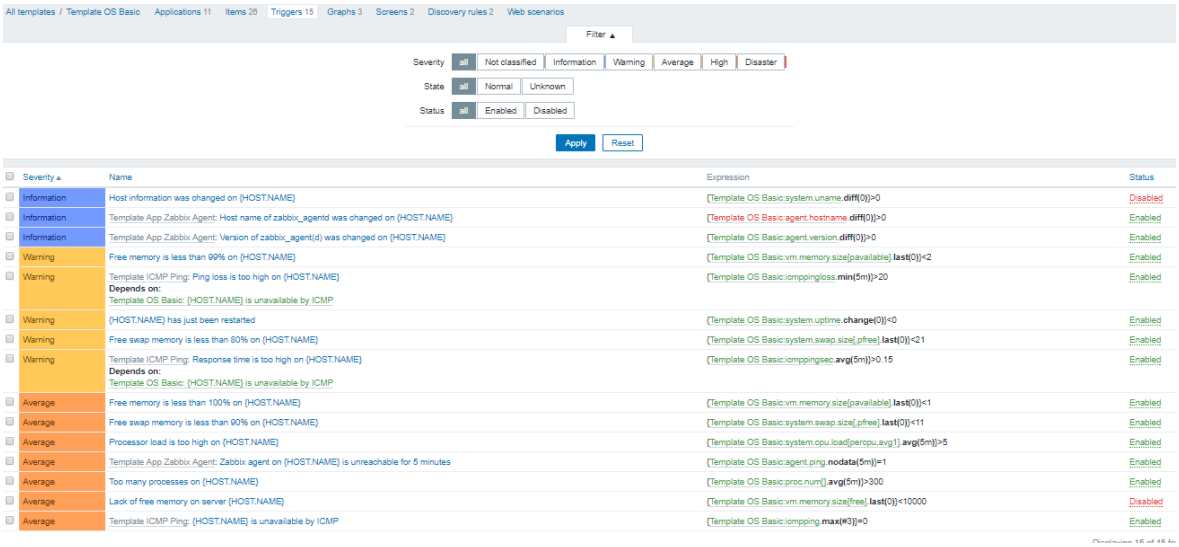

Figura D.19: Triggers definidos

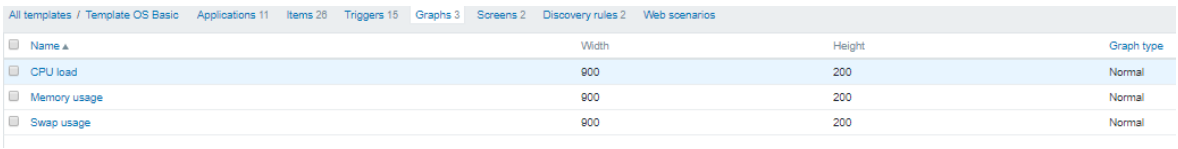

Figura D.20: Gráficos compuestos definidos

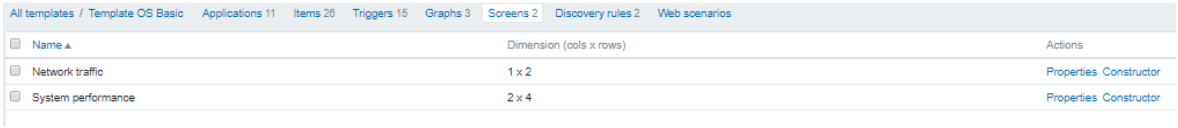

Figura D.21: Pantallas de gráficos

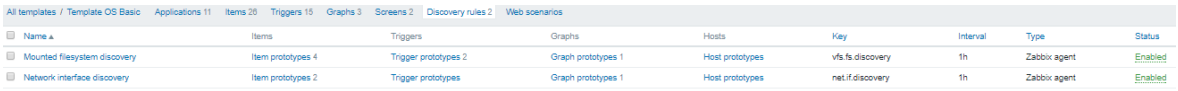

Figura D.22: Reglas de descubrimiento y sus elementos principales

D.11. Acciones

D.11.1. Auto registro

En la siguiente Figura D.23 se puede observar las operaciones que se realizan para cada acción y la condición que se debe cumplir para auto descubrir los hosts, añadirlos a los grupos y asignarles las plantillas.

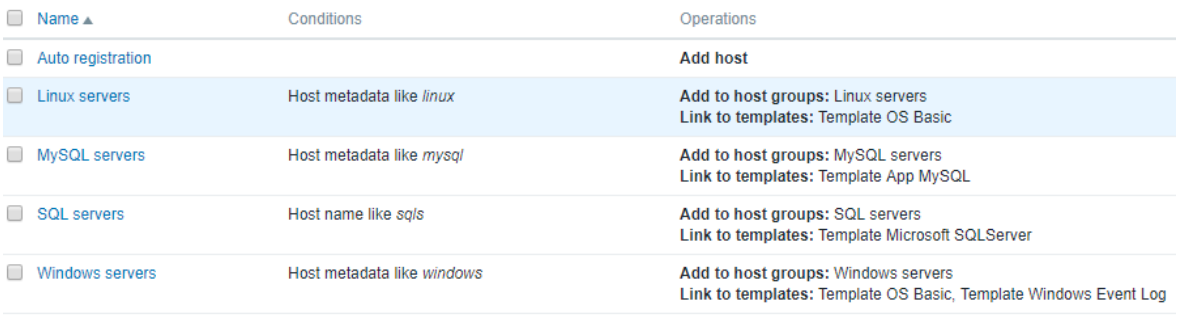

Figura D.23: Reglas de auto registro definidas en Zabbix

D.11.2. Triggers

Para realizar diferentes acciones cuando se detecta un problema y escalarlo en función del tiempo que transcurra y las acciones que se realicen, es necesario definir una acción donde mediante una condición Figura D.24, se detecta el problema que se quiere escalar. A continuación se definen los siguientes pasos Figura D.25, que se trata en este ejemplo de la ejecución de un comando remoto en primer lugar, y en segundo lugar se envía un email a todos los usuarios del grupo Zabbix administrators.

Estas acciones, se van ejecutando en orden y esperando el tiempo especificado entre una y otra, excepto si no se especifica ninguno que se ejecutan secuencialmente conforme se van ejecutando. Esto implica que si el problema se resuelve al realizar la primera operación, la segunda no se ejecutará. Además, del mismo modo que se han definido estas operaciones, se podrían definir operaciones de recuperación sobre la misma operación, en el apartado de *Recovery operations*.

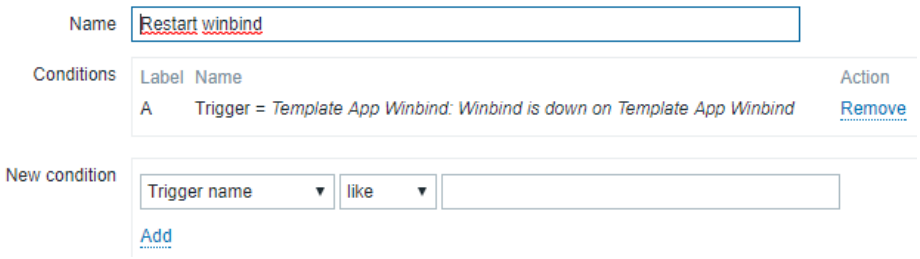

Figura D.24: Condiciones para detección de los problemas

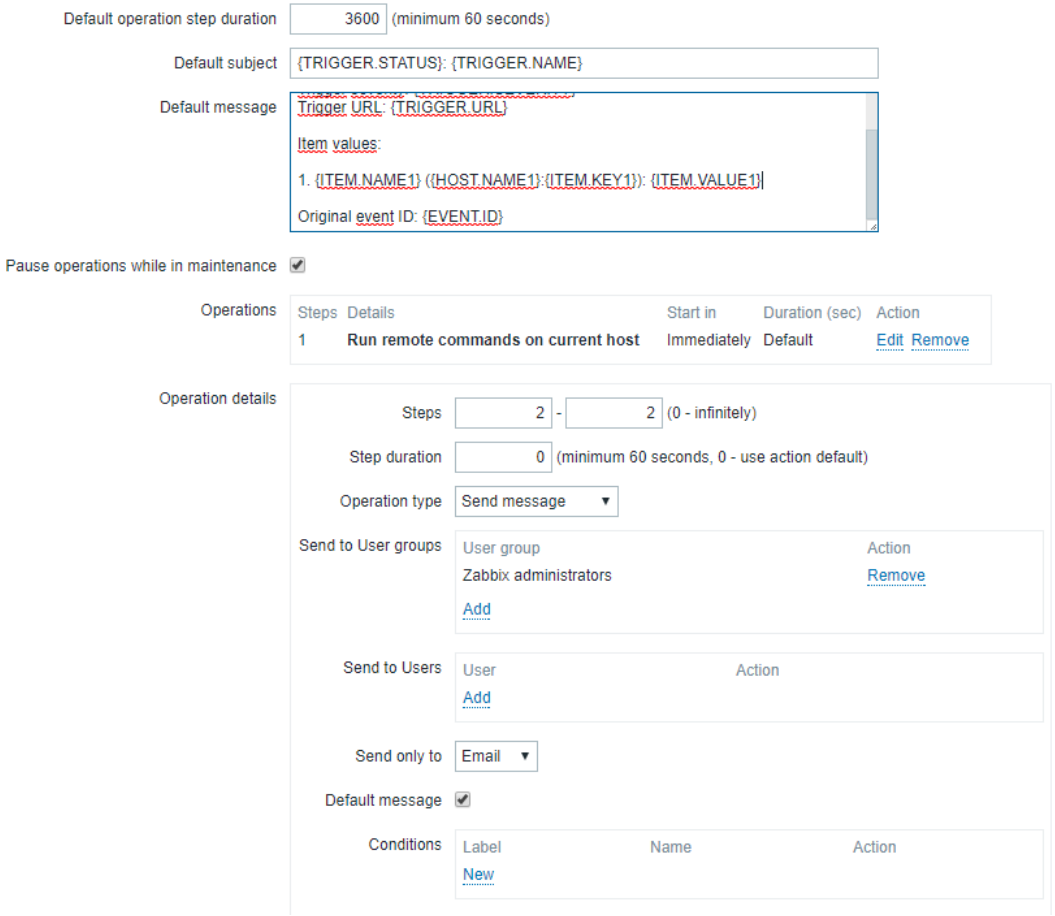

Figura D.25: Definición de operaciones de escalado o recuperación de problemas

Anexos E

Detalle de implementación de políticas

E.1. Template ICMP Ping

ICMP response time	Triggers 1	icmppingsec	5m	90d	730d	Simple check
ICMP ping	Triggers 1	icmpping	5m	90d	730d	Simple check

Figura E.1: Items y características de Template ICMP Ping

Warning	Response time is too high on {HOST.NAME} **Depends on:** Template ICMP Ping: {HOST.NAME} is unavailable by ICMP	{Template ICMP Ping:icmppingsec.**avg**(10m)}>2
Average	{HOST.NAME} is unavailable by ICMP	{Template ICMP Ping:icmpping.**last**(0)}=0

Figura E.2: Triggers, expresiones y dependencias de Template ICMP Ping

E.2. Template App Zabbix Agent

Agent ping	Triggers 1	agent.ping	5m	1d	0d	Zabbix agent
Version of zabbix_agent(d) running	Triggers 1	agent.version	1d	7d		Zabbix agent

Figura E.3: Items y características de Template App Zabbix Agent

Information	Version of zabbix_agent(d) was changed on {HOST.NAME}	{Template App Zabbix Agent:agent.version.**diff**(0)}>0
Average	Zabbix agent on {HOST.NAME} is unreachable for 5 minutes	{Template App Zabbix Agent:agent.ping.**nodata**(5m)}=1

Figura E.4: Triggers, expresiones y dependencias de Template App Zabbix Agent

E.3. Template OS Basic

Free memory	Triggers 1	vm.memory.size[free]	5m	90d	730d	Zabbix agent
CPU load	Triggers 1	system.cpu.load[,avg1]	5m	90d	730d	Zabbix agent
System information	Triggers 1	system.uname	1h	7d		Zabbix agent
Total memory		vm.memory.size[total]	5m	90d	730d	Zabbix agent
Free memory (percentage)	Triggers 2	vm.memory.size[pavailable]	5m	90d	730d	Zabbix agent
Used swap space		system.swap.size[,used]	5m	90d	730d	Zabbix agent
Used memory		vm.memory.size[used]	5m	90d	730d	Zabbix agent
Total swap space		system.swap.size[,total]	5m	90d	730d	Zabbix agent
System uptime	Triggers 1	system.uptime	1h	1d	0d	Zabbix agent
Template ICMP Ping: ICMP response time	Triggers 1	icmppingsec	5m	90d	730d	Simple check
Template ICMP Ping: ICMP loss	Triggers 1	icmppingloss	5m	90d	730d	Simple check
Free swap space		system.swap.size[,free]	5m	90d	730d	Zabbix agent
Template App Zabbix Agent: Agent ping	Triggers 1	agent.ping	5m	1d	0d	Zabbix agent
Template ICMP Ping: ICMP ping	Triggers 1	icmpping	5m	90d	730d	Simple check
Template App Zabbix Agent: Version of zabbix_agent(d) running	Triggers 1	agent.version	12h	7d		Zabbix agent
Free swap space (percentage)	Triggers 2	system.swap.size[,pfree]	5m	90d	730d	Zabbix agent
Number of processes	Triggers 1	proc.num[]	5m	90d	730d	Zabbix agent

Figura E.5: Items y características de Template OS Basic

Information	Host information was changed on {HOST.NAME}	{Template OS Basic:system.uname.**diff**(0)}>0
Information	Template App Zabbix Agent: Version of zabbix_agent(d) was changed on {HOST.NAME}	{Template OS Basic:agent.version.**diff**(0)}>0
Information	{HOST.NAME} has just been restarted	{Template OS Basic:system.uptime.**change**(0)}<0
Warning	Template ICMP Ping: Response time is too high on {HOST.NAME} **Depends on:** Template OS Basic: {HOST.NAME} is unavailable by ICMP	{Template OS Basic:icmppingsec.**avg**(10m)}>2
Warning	Free swap memory is less than 20% on {HOST.NAME}	{Template OS Basic:system.swap.size[,pfree].**last**(0)}<20
Warning	Free memory is less than 10% on {HOST.NAME}	{Template OS Basic:vm.memory.size[pavailable].**last**(0)}<10
Warning	Template ICMP Ping: Ping loss is too high on {HOST.NAME} **Depends on:** Template OS Basic: {HOST.NAME} is unavailable by ICMP	{Template OS Basic:icmppingloss.**min**(10m)}>20
Warning	Too many processes on {HOST.NAME}	{Template OS Basic:proc.num[].**avg**(5m)}>300
Average	Template App Zabbix Agent: Zabbix agent on {HOST.NAME} is unreachable for 5 minutes	{Template OS Basic:agent.ping.**nodata**(5m)}=1
Average	Lack of free memory on server {HOST.NAME}	{Template OS Basic:vm.memory.size[free].**last**(0)}<10000
Average	Free memory is less than 5% on {HOST.NAME}	{Template OS Basic:vm.memory.size[pavailable].**last**(0)}<5
Average	Processor load is too high on {HOST.NAME}	{Template OS Basic:system.cpu.load[,avg1].**avg**(10m)}>5
Average	Free swap memory is less than 10% on {HOST.NAME}	{Template OS Basic:system.swap.size[,pfree].**last**(0)}<10
Average	Template ICMP Ping: {HOST.NAME} is unavailable by ICMP	{Template OS Basic:icmpping.**last**(0)}=0

Figura E.6: Triggers, expresiones y dependencias de Template OS Basic

Free disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},free]	5m	7d	730d	Zabbix agent
Free disk space on {#FSNAME} (percentage)	vfs.fs.size[{#FSNAME},pfree]	5m	7d	730d	Zabbix agent
Total disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},total]	1h	7d	730d	Zabbix agent
Used disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},used]	5m	7d	730d	Zabbix agent

Figura E.7: Items y características de descubrimiento de particiones de disco

Average	Free disk space is less than 5% on volume {#FSNAME}	{Template OS Basic:vfs.fs.size[{#FSNAME},pfree].last(0)}<5
Warning	Free disk space is less than 10% on volume {#FSNAME}	{Template OS Basic:vfs.fs.size[{#FSNAME},pfree].last(0)}<10

Figura E.8: Triggers y expresiones de descubrimiento de particiones de disco

Incoming network traffic on {#IFNAME}	net.if.in[{#IFNAME}]	5m	7d	730d	Zabbix agent
Outgoing network traffic on {#IFNAME}	net.if.out[{#IFNAME}]	5m	7d	730d	Zabbix agent

Figura E.9: Items y características de descubrimiento de interfaces de red

E.4. Template Windows Event Log:

Name	Triggers	Key	Interval	History	Trends	Type
Event log Application		eventlog[Application,,"Warning|Error",,,,skip]	5m	7d		Zabbix agent (active)
Event log System		eventlog[System,,"Warning|Error",,,,skip]	5m	7d		Zabbix agent (active)
Event log general		eventlog[,,,,,,]	5m	1d		Zabbix agent (active)

Figura E.10: Items y características de Template Windows Event Log

Average	Application Error in Windows event log on {HOST.NAME}	({{Template Windows Event Log:eventlog[Application,,"Warning|Error",,,,skip].iregexp(.*Error.*)}}<>0)
Warning	Application Warning in Windows event log on {HOST.NAME}	({{Template Windows Event Log:eventlog[Application,,"Warning|Error",,,,skip].iregexp(.*Warning.*)}}<>0)
Average	System Error in Windows event log on {HOST.NAME}	({{Template Windows Event Log:eventlog[System,,"Warning|Error",,,,skip].iregexp(.*Error.*)}}<>0)
Warning	System Warning in Windows event log on {HOST.NAME}	({{Template Windows Event Log:eventlog[System,,"Warning|Error",,,,skip].iregexp(.*Warning.*)}}<>0)

Figura E.11: Triggers y expresiones de Template Windows Event Log

E.4.1. Template Linux Log Messages:

Var/log/messages	Triggers 1	log[/var/log/messages,,,,skip]	5m	730d	Zabbix agent (active)

Figura E.12: Item y características de Template Linux Log Messages

Average	Error in /var/log/messages on {HOST.NAME}	({{Template Linux Log Messages:log[/var/log/messages,error,,,skip].iregexp(.*)}}<>0)

Figura E.13: Trigger y expresión de Template Linux Log Messages

E.5. Template App Winbind:

Name	Triggers	Key	Interval	History	Trends	Type
Winbind is running	Triggers 1	system.run[wbinfo -p]	5m	90d		Zabbix agent

Figura E.14: Item y características de Template App Winbind

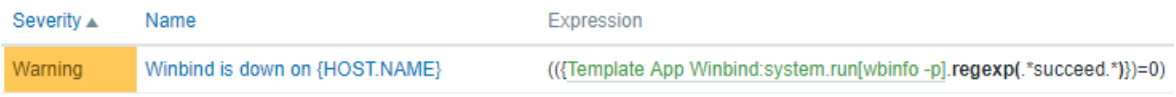

Figura E.15: Trigger y expresión de Template App Winbind

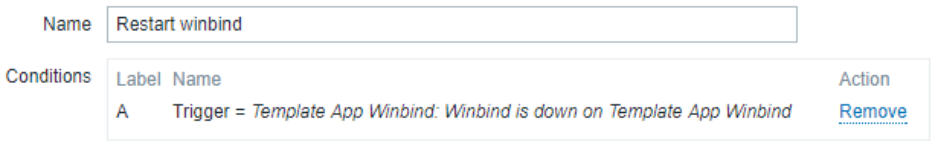

Figura E.16: Condición para realizar la acción

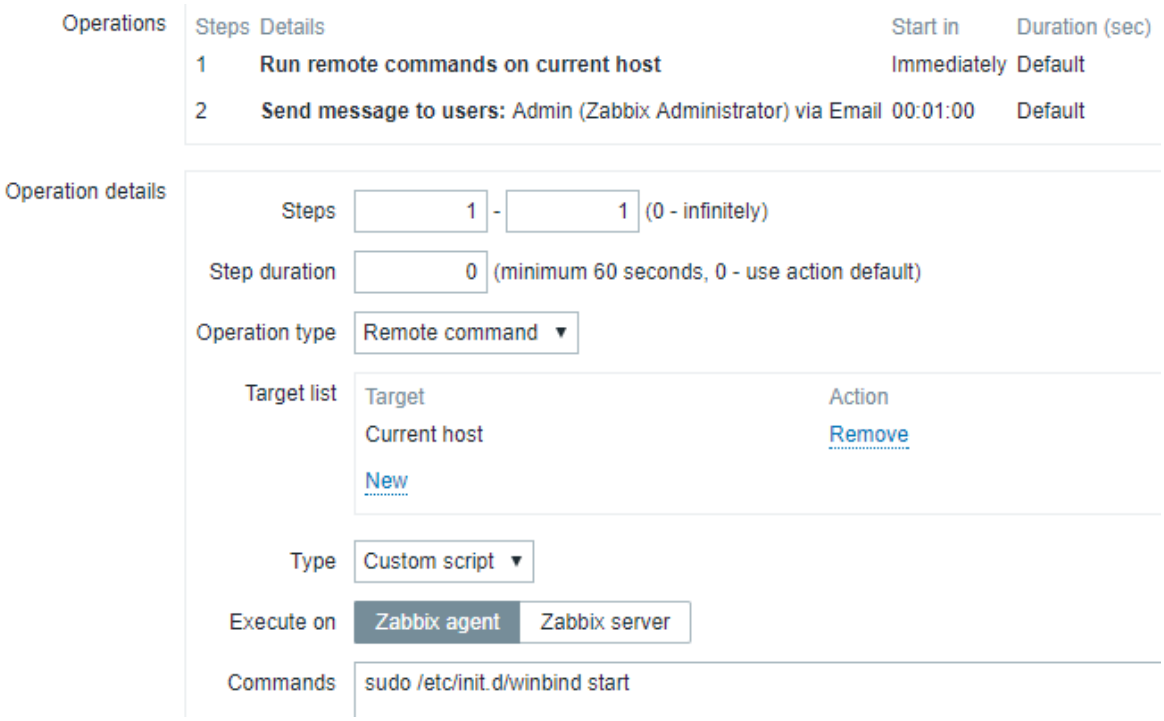

Figura E.17: Ejecución del comando remoto

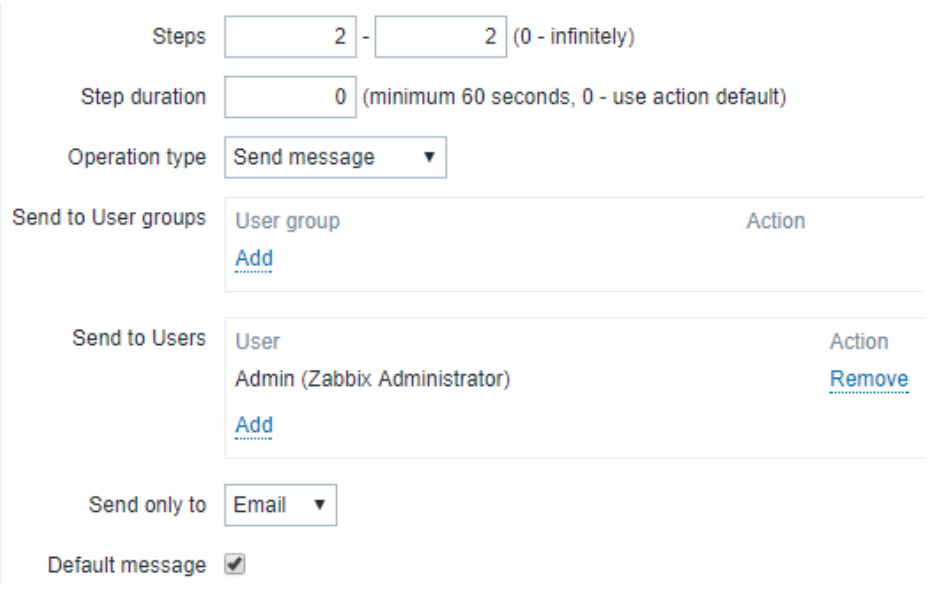

Figura E.18: Envío del correo al administrador

E.6. Template App HTTP Service

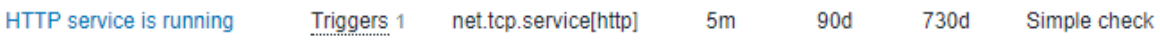

Figura E.19: Item y características de Template App HTTP Service

Figura E.20: Trigger y expresión de Template App HTTP Service

E.7. Template App HTTPS Service

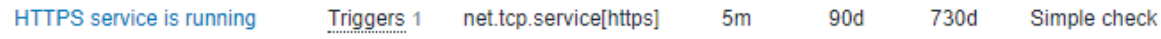

Figura E.21: Item y características de Template App HTTPS Service

Figura E.22: Trigger y expresión de Template App HTTPS Service

E.8. Template App CVS Service:

Figura E.23: Item y características de Template App CVS Service

Average	CVS service is down on {HOST.NAME}	{Template App CVS Service:net.tcp.service[tcp,,2401].last()}=0

Figura E.24: Trigger y expresión de Template App CVS Service

E.9. Template SSL Cert:

Template App HTTPS Service: HTTPS service is running	Triggers 1	net.tcp.service[https]	5m	90d	730d	Simple check
{$SNI} SSL certificate validity	Triggers 6	zext_ssl_cert.sh[-d,{HOST.CONN},{$SSL_PORT},{$SNI}]	1d	7d	0d	External check
SSL certificate issuer		zext_ssl_cert.sh[-i,{HOST.CONN},{$SSL_PORT},{$SNI}]	1d	1d		External check

Figura E.25: Items y características de Template SSL Cert

Not classified	SSL certificate on {HOSTNAME} expires in less than 90 days ({ITEM.VALUE} days remaining) **Depends on:** Template SSL Cert: SSL certificate on {HOSTNAME} expires in less than 60 days ({ITEM.VALUE} days remaining)	{Template SSL Cert:zext_ssl_cert.sh[-d,{HOST.CONN},{$SSL_PORT},{$SNI}].last(0)}<90
Information	SSL certificate on {HOSTNAME} expires in less than 60 days ({ITEM.VALUE} days remaining) **Depends on:** Template SSL Cert: SSL certificate on {HOSTNAME} expires in less than 30 days ({ITEM.VALUE} days remaining)	{Template SSL Cert:zext_ssl_cert.sh[-d,{HOST.CONN},{$SSL_PORT},{$SNI}].last(0)}<60
Warning	SSL certificate on {HOSTNAME} expires in less than 30 days ({ITEM.VALUE} days remaining) **Depends on:** Template SSL Cert: SSL certificate on {HOSTNAME} expires in less than 15 days ({ITEM.VALUE} days remaining)	{Template SSL Cert:zext_ssl_cert.sh[-d,{HOST.CONN},{$SSL_PORT},{$SNI}].last(0)}<30
Average	Template App HTTPS Service: HTTPS service is down on {HOST.NAME}	{Template SSL Cert:net.tcp.service[https].last()}=0
Average	SSL certificate on {HOSTNAME} expires in less than 15 days ({ITEM.VALUE} days remaining) **Depends on:** Template SSL Cert: SSL certificate on {HOSTNAME} expires in less than 7 days ({ITEM.VALUE} days remaining)	{Template SSL Cert:zext_ssl_cert.sh[-d,{HOST.CONN},{$SSL_PORT},{$SNI}].last(0)}<15
High	SSL certificate on {HOSTNAME} expires in less than 7 days ({ITEM.VALUE} days remaining) **Depends on:** Template SSL Cert: SSL certificate on {HOSTNAME} expired	{Template SSL Cert:zext_ssl_cert.sh[-d,{HOST.CONN},{$SSL_PORT},{$SNI}].last(0)}<7
Disaster	SSL certificate on {HOSTNAME} expired	{Template SSL Cert:zext_ssl_cert.sh[-d,{HOST.CONN},{$SSL_PORT},{$SNI}].last(0)}<0

Figura E.26: Triggers, expresiones y dependencias de Template SSL Cert

E.10. Template App MySQL:

MySQL delete operations per second		mysql.status[Com_delete]	5m	90d	730d	Zabbix agent
MySQL begin operations per second		mysql.status[Com_begin]	5m	90d	730d	Zabbix agent
MySQL bytes sent per second		mysql.status[Bytes_sent]	5m	90d	730d	Zabbix agent
MySQL bytes received per second		mysql.status[Bytes_received]	5m	90d	730d	Zabbix agent
MySQL slow queries		mysql.status[Slow_queries]	5m	90d	730d	Zabbix agent
MySQL queries per second		mysql.status[Questions]	5m	90d	730d	Zabbix agent
MySQL connections		mysql.status[Connections]	5m	90d	730d	Zabbix agent
MySQL commit operations per second		mysql.status[Com_commit]	5m	90d	730d	Zabbix agent
MySQL version		mysql.version	1d	7d		Zabbix agent
MySQL status	Triggers 1	mysql.ping	5m	90d	730d	Zabbix agent
MySQL rollback operations per second		mysql.status[Com_rollback]	5m	90d	730d	Zabbix agent
MySQL insert operations per second		mysql.status[Com_insert]	5m	90d	730d	Zabbix agent
MySQL select operations per second		mysql.status[Com_select]	5m	90d	730d	Zabbix agent
MySQL update operations per second		mysql.status[Com_update]	5m	90d	730d	Zabbix agent

Figura E.27: Items y características de Template App MySQL

Average	MySQL is down	{Template App MySQL:mysql.ping.last(0)}=0

Figura E.28: Trigger y expresión de Template App MySQL

E.11. Template Microsoft SQLServer:

SQL Default Instance: Service State - SQL Server	Triggers 1	service_state[MSSQLSERVER]	5m	90d	730d	Zabbix agent
SQL Default Instance: Service State - SQL Agent	Triggers 1	service_state[SQLServerAgent]	5m	90d	730d	Zabbix agent
SQL Default Instance: Page Life Expectancy	Triggers 1	perf_counter["\SQLServer:Buffer Manager\Page Life Expectancy"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Total Server Memory		perf_counter["\SQLServer:Memory Manager\Total Server Memory (KB)"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Transactions per second		perf_counter["\SQLServer:Databases(_Total)\Transactions/sec"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Number of Range Scans/sec		perf_counter["\SQLServer:Access Methods\Range Scans/sec"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Index Searches/sec		perf_counter["\SQLServer:Access Methods\Index Searches/sec"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Full Scans / sec		perf_counter["\SQLServer:Access Methods\Full Scans/sec"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Number Users Connected		perf_counter["\SQLServer:General Statistics\User Connections"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Number of Probe Scans/sec		perf_counter["\SQLServer:Access Methods\Probe Scans/sec"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Database Pages		perf_counter["\SQLServer:Buffer Manager\Database pages"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Blocked Processes	Triggers 2	perf_counter["\SQLServer:General Statistics\Processes blocked"]	5m	90d	730d	Zabbix agent
SQL Default Instance: % Processor Time		perf_counter["\Process(sqlservr)\% Processor Time"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Number of Deadlocks per second	Triggers 1	perf_counter["\SQLServer:Locks(_Total)\Number of Deadlocks/sec"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Data File Size		perf_counter["\SQLServer:Databases(_Total)\Data File(s) Size (KB)"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Lock Waits per second		perf_counter["\SQLServer:Locks(_Total)\Lock Waits/sec"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Longest Running Transaction	Triggers 2	perf_counter["\SQLServer:Transactions\Longest Transaction Running Time"]	5m	90d	730d	Zabbix agent
SQL Default Instance: Number Failed Jobs	Triggers 1	perf_counter["\SQLAgent:Jobs(_Total)\Failed jobs"]	1h	90d	730d	Zabbix agent
SQL Default Instance: Log File Size		perf_counter["\SQLServer:Databases(_Total)\Log File(s) Size (KB)"]	5m	90d	730d	Zabbix agent

Figura E.29: Items y características de Template Microsoft SQLServer

Warning	SQL Default Instance: Long running transaction: {ITEM.VALUE} seconds	{Template Microsoft SQLServer:perf_counter["\SQLServer:Transactions\Longest Transaction Running Time"].last(0)}>7200
Warning	SQL Default Instance: Many Blocked Processes: {ITEM.VALUE}	{Template Microsoft SQLServer:perf_counter["\SQLServer:General Statistics\Processes blocked"].last(600)}>10
Warning	SQL Default Instance: Long running transaction: {ITEM.VALUE} seconds	{Template Microsoft SQLServer:perf_counter["\SQLServer:Transactions\Longest Transaction Running Time"].last(0)}>3600
Average	SQL Default Instance: Many Blocked Processes: {ITEM.VALUE}	{Template Microsoft SQLServer:perf_counter["\SQLServer:General Statistics\Processes blocked"].last(600)}>20
Average	SQL Default Instance: Page Life Expectancy low: {ITEM.VALUE}	{Template Microsoft SQLServer:perf_counter["\SQLServer:Buffer Manager\Page Life Expectancy"].last(300)}<300
Average	SQL Default Instance: Number of Deadlocks per second	{Template Microsoft SQLServer:perf_counter["\SQLServer:Locks(_Total)\Number of Deadlocks/sec"].last(0)}>0
High	SQL Default Instance: Number Failed Jobs	{Template Microsoft SQLServer:perf_counter["\SQLAgent:Jobs(_Total)\Failed jobs"].change(0)}>0
High	SQL Default Instance: Service State - SQL Server	{Template Microsoft SQLServer:service_state[MSSQLSERVER].last(0)}>0 and {Template Microsoft SQLServer:service_state[MSSQLSERVER].last(0)}<255
High	SQL Default Instance: Service State - SQL Agent	{Template Microsoft SQLServer:service_state[SQLServerAgent].last(0)}>0 and {Template Microsoft SQLServer:service_state[SQLServerAgent].last(0)}<255

Figura E.30: Triggers y expresiones de Template Microsoft SQLServer

E.12. Template SNMP Generic:

Device contact details	sysContact	1d	7d		SNMPv2 agent
Device description	sysDescr	1d	7d		SNMPv2 agent
Device location	sysLocation	1d	7d		SNMPv2 agent
Device name	sysName	1d	7d		SNMPv2 agent
Device uptime	sysUpTime	1d	7d	0d	SNMPv2 agent

Figura E.31: Items y características de Template SNMP Generic

E.13. Template SNMP Interfaces:

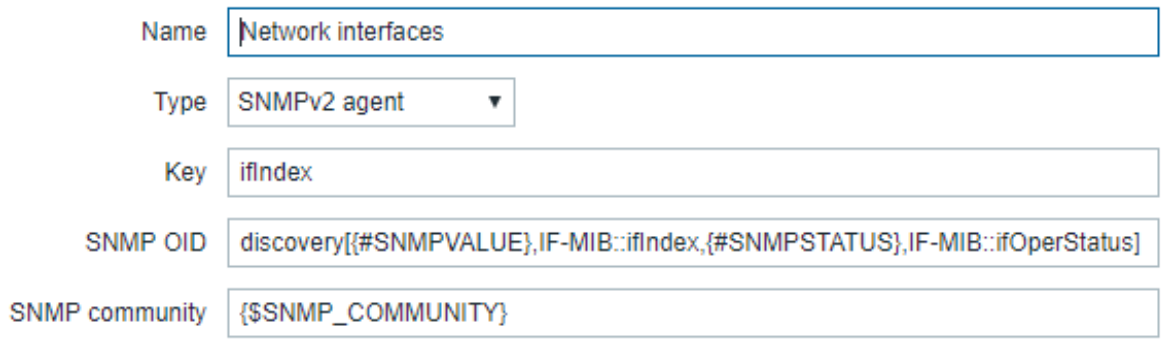

Figura E.32: Regla de auto descubrimiento de Template SNMP Interfaces

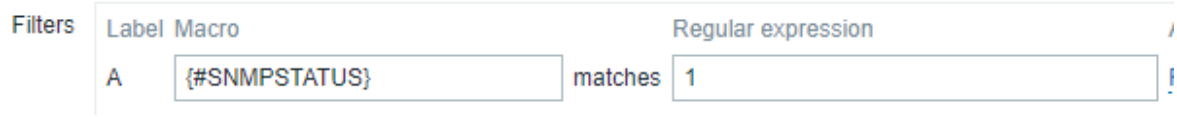

Figura E.33: Filtro de estado de auto descubrimiento de Template SNMP Interfaces

Inbound errors on interface {#SNMPVALUE}	ifInErrors[{#SNMPVALUE}]	5m	90d	730d	SNMPv2 agent
Incoming traffic on interface {#SNMPVALUE}	ifInOctets[{#SNMPVALUE}]	5m	90d	730d	SNMPv2 agent
Operational status of interface {#SNMPVALUE}	ifOperStatus[{#SNMPVALUE}]	5m	90d	730d	SNMPv2 agent
Outbound errors on interface {#SNMPVALUE}	ifOutErrors[{#SNMPVALUE}]	5m	90d	730d	SNMPv2 agent
Outgoing traffic on interface {#SNMPVALUE}	ifOutOctets[{#SNMPVALUE}]	5m	90d	730d	SNMPv2 agent

Figura E.34: Items y características de auto descubrimiento de Template SNMP Interfaces

Information	Operational status was changed to INACTIVE on {HOST.NAME} interface {#SNMPVALUE}	{Template SNMP Interfaces:ifOperStatus[{#SNMPVALUE}].diff(0)}=1 and {Template SNMP Interfaces:ifOperStatus[{#SNMPVALUE}].last()}=0
Information	Operational status was changed to ACTIVE on {HOST.NAME} interface {#SNMPVALUE}	{Template SNMP Interfaces:ifOperStatus[{#SNMPVALUE}].diff(0)}=1 and {Template SNMP Interfaces:ifOperStatus[{#SNMPVALUE}].last()}=1
Average	Outbound errors on {HOST.NAME} interface {#SNMPVALUE} **Depends on:** Template SNMP Interfaces: Operational status was changed to INACTIVE on {HOST.NAME} interface {#SNMPVALUE}	{Template SNMP Interfaces:ifOutErrors[{#SNMPVALUE}].last()}<>0
Average	Inbound errors on {HOST.NAME} interface {#SNMPVALUE} **Depends on:** Template SNMP Interfaces: Operational status was changed to INACTIVE on {HOST.NAME} interface {#SNMPVALUE}	{Template SNMP Interfaces:ifInErrors[{#SNMPVALUE}].last()}<>0

Figura E.35: Triggers y expresiones de auto descubrimiento de Template SNMP Interfaces

E.14. Template App Zabbix Server:

Zabbix busy alerter processes, in %	Triggers 2	zabbix[process,alerter,avg,busy]	5m	7d	730d	Zabbix internal
Zabbix busy configuration syncer processes, in %	Triggers 2	zabbix[process,configuration syncer,avg,busy]	5m	7d	730d	Zabbix internal
Zabbix busy db watchdog processes, in %	Triggers 2	zabbix[process,db watchdog,avg,busy]	5m	7d	730d	Zabbix internal
Zabbix busy discoverer processes, in %	Triggers 2	zabbix[process,discoverer,avg,busy]	5m	7d	730d	Zabbix internal
Zabbix busy escalator processes, in %	Triggers 2	zabbix[process,escalator,avg,busy]	5m	7d	730d	Zabbix internal

Figura E.36: Items y características de Template App Zabbix Server

Warning	More than 50 items having missing data for more than 5 minutes **Depends on:** Template App Zabbix Server: More than 100 items having missing data for more than 10 minutes	{Template App Zabbix Server:zabbix[queue].**min**(5m)}>50
Average	More than 100 items having missing data for more than 10 minutes	{Template App Zabbix Server:zabbix[queue,10m].**min**(10m)}>100
Warning	Zabbix alerter processes more than 75% busy **Depends on:** Template App Zabbix Server: Zabbix alerter processes more than 90% busy	{Template App Zabbix Server:zabbix[process,alerter,avg,busy].**avg**(10m)}>75
Average	Zabbix alerter processes more than 90% busy	{Template App Zabbix Server:zabbix[process,alerter,avg,busy].**avg**(10m)}>90

Figura E.37: Triggers y expresiones de Template App Zabbix Server

E.15. Ejemplos gráficos

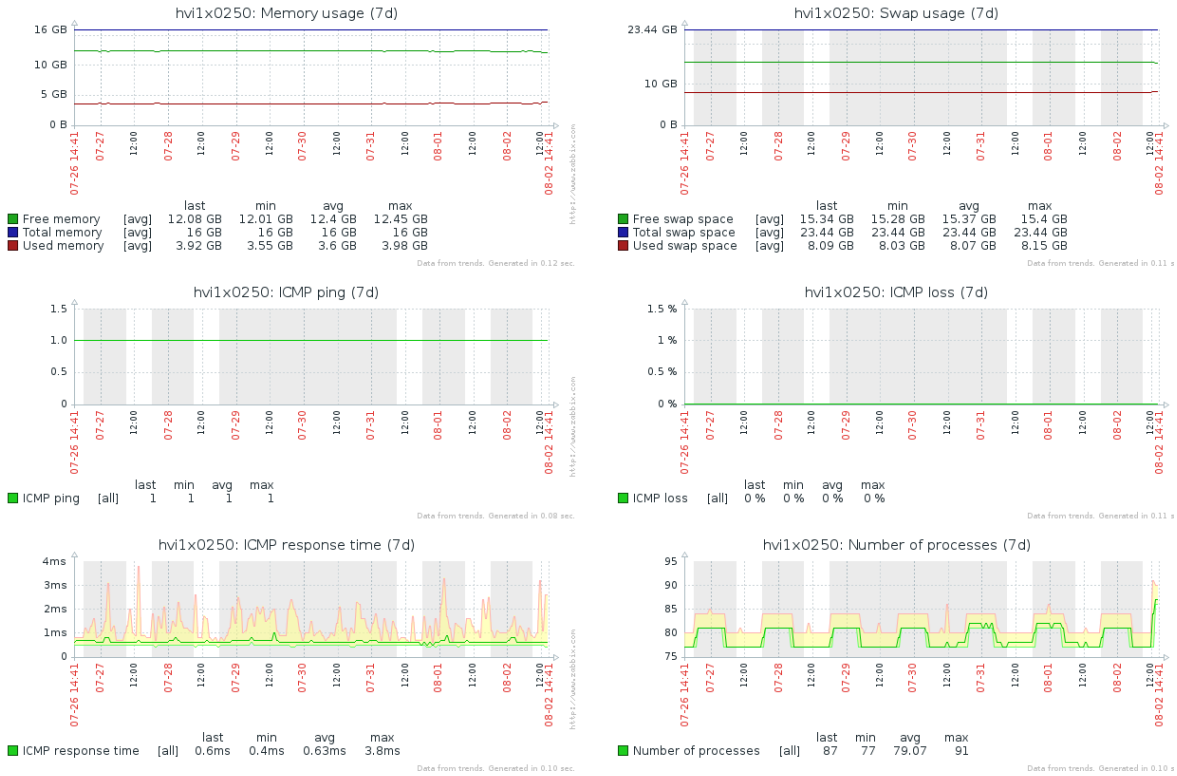

Figura E.38: Ejemplo gráfico pantalla básica Template OS Basic

Figura E.39: Ejemplo gráfico pantalla interfaces red Template OS Basic

Figura E.40: Ejemplo gráfico pantalla particiones disco Template OS Basic

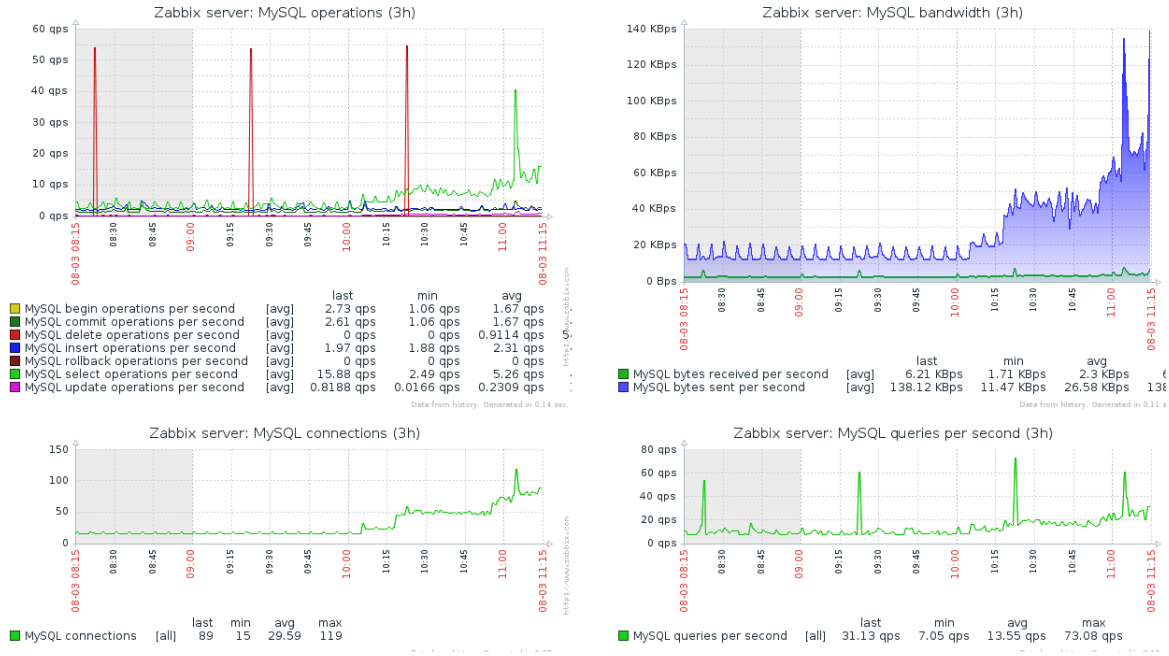

Figura E.41: Ejemplo gráfico pantalla Template App MySQL

Figura E.42: Ejemplo gráfico pantalla Template Microsoft SQLServer

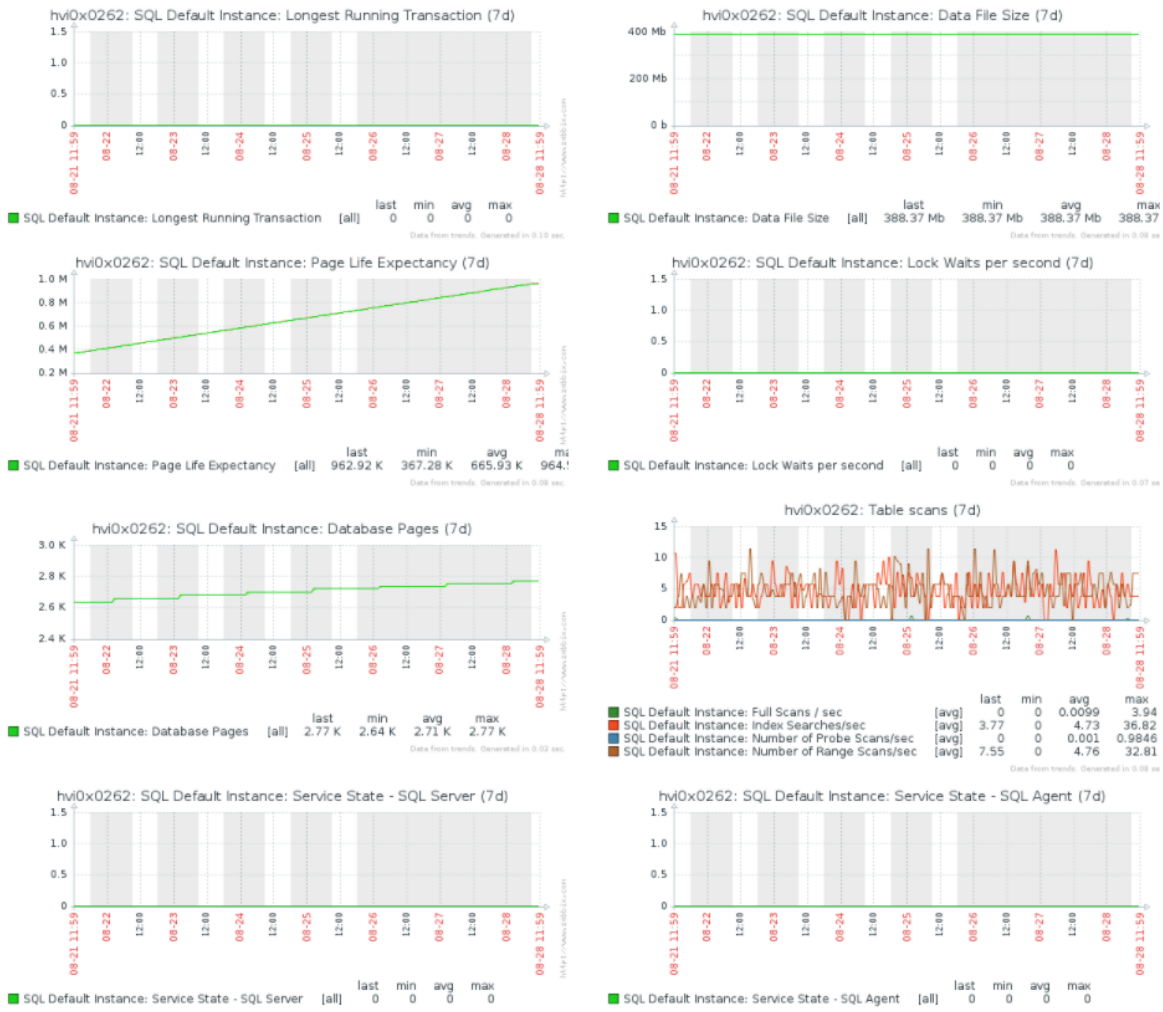

Figura E.43: Ejemplo gráfico pantalla Template Microsoft SQLServer

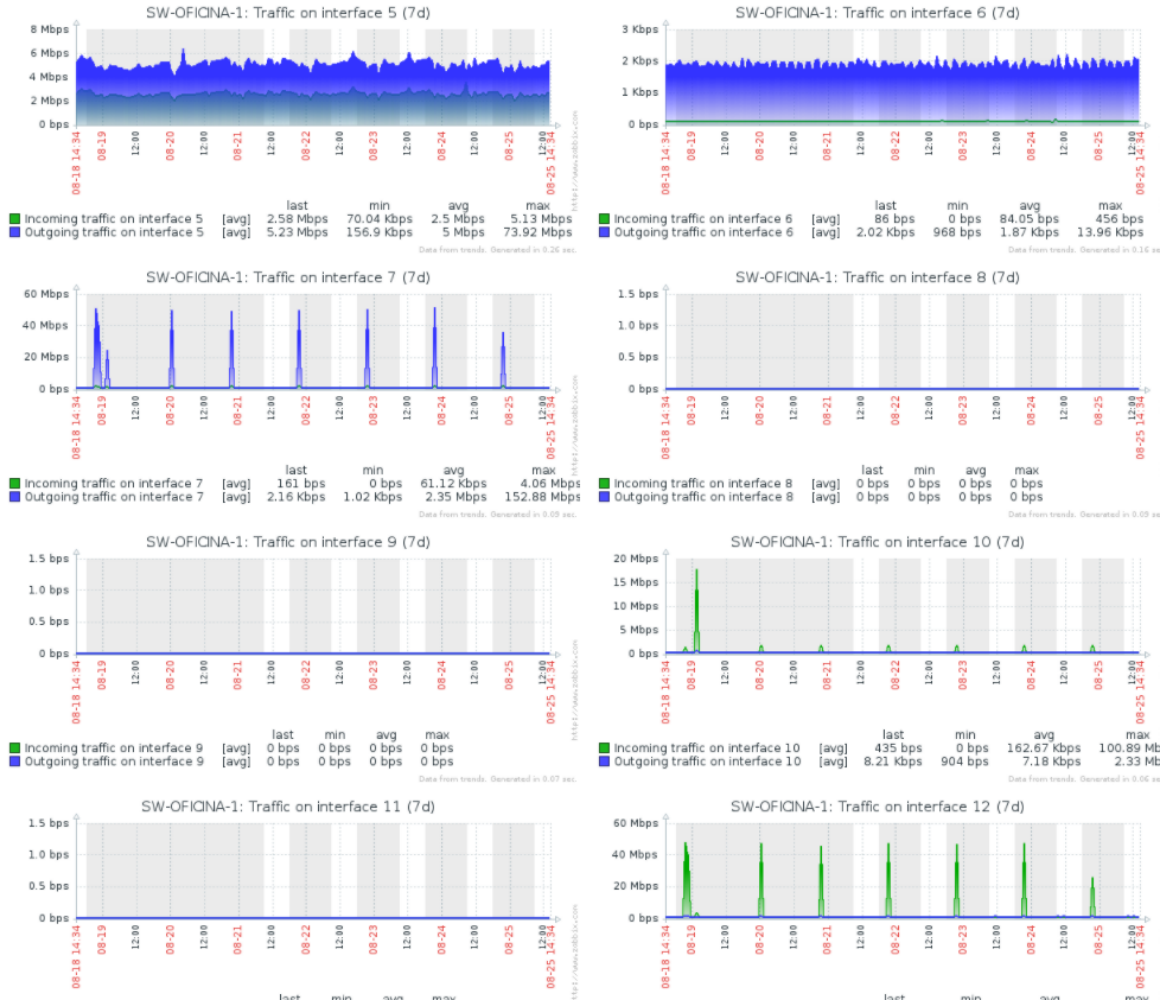

Figura E.44: Ejemplo gráfico pantalla Template SNMP Interfaces

Figura E.45: Ejemplo gráfico pantalla Template SNMP Interfaces

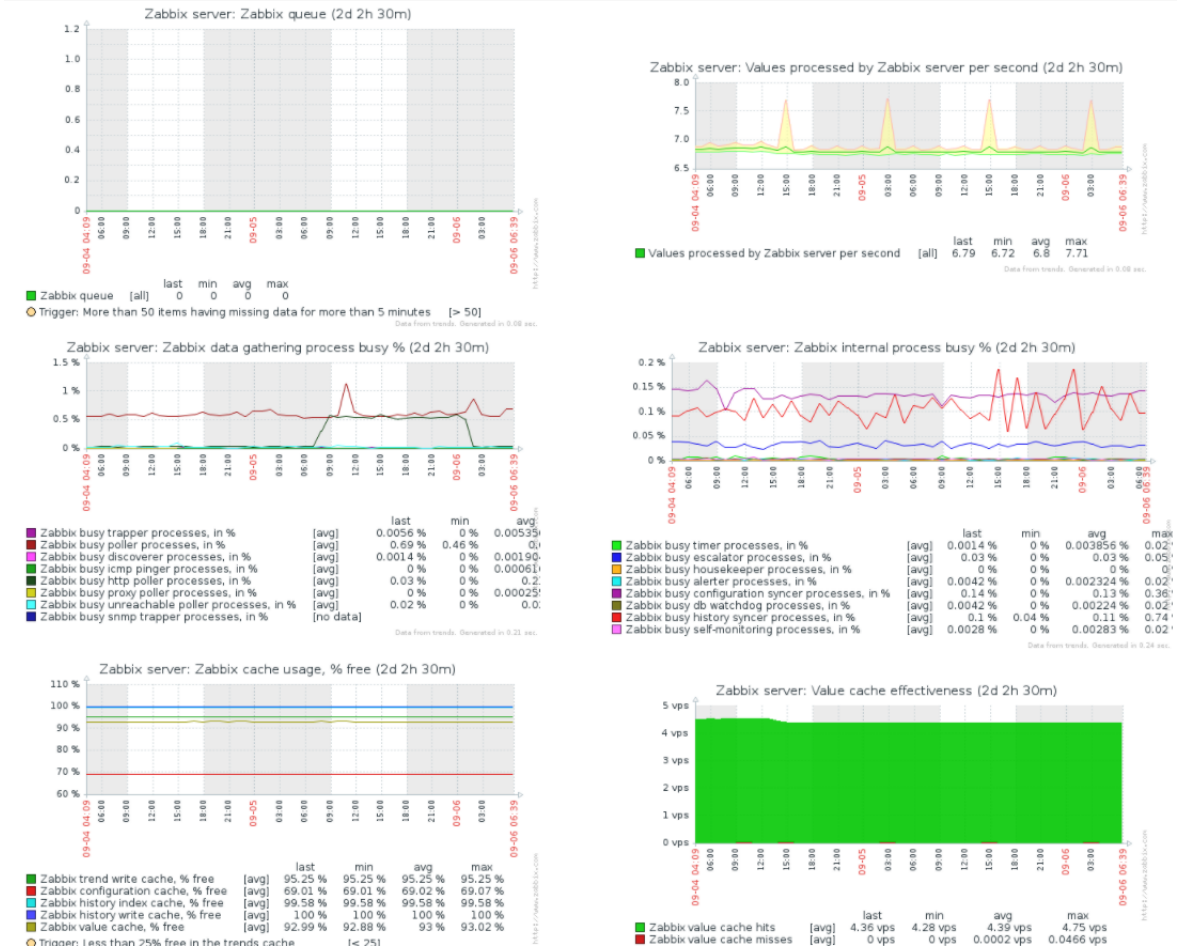

Figura E.46: Ejemplo gráfico pantalla Template App Zabbix Server

Anexos F

Resultados de las pruebas

▼	**Filesystems** (8 Items)				
	Free disk space on /	2017-08-17 12:05:49	10.42 GB	-24 KB	Graph
	Free disk space on / (percentage)	2017-08-17 12:05:51	77.82 %		Graph
	Free disk space on /boot	2017-08-17 12:05:50	785.47 MB		Graph
	Free disk space on /boot (percentage)	2017-08-17 12:05:52	77.46 %		Graph
	Total disk space on /	2017-08-17 11:27:53	13.38 GB		Graph
	Total disk space on /boot	2017-08-17 11:27:54	1014 MB		Graph
	Used disk space on /	2017-08-17 12:05:55	2.97 GB	+24 KB	Graph
	Used disk space on /boot	2017-08-17 12:05:56	228.53 MB		Graph
▼	**General** (2 Items)				
	System information	2017-08-17 11:27:19	Linux pandora 3.1...		Hist...
	System uptime	2017-08-17 12:05:21	72 days, 01:57:24	+00:01:01	Graph
▼	**ICMP** (3 Items)				
	ICMP loss	2017-08-17 12:05:39	0 %		Graph
	ICMP ping	2017-08-17 12:05:39	Up (1)		Graph
	ICMP response time	2017-08-17 12:05:39	0.2ms		Graph
▼	**Memory** (8 Items)				
	Free memory	2017-08-17 12:05:26	1.92 GB	-148 KB	Graph
	Free memory (percentage)	2017-08-17 12:05:27	79.84 %		Graph
	Free swap space	2017-08-17 12:05:21	1.6 GB		Graph
	Free swap space (percentage)	2017-08-17 12:05:21	100 %		Graph
	Total memory	2017-08-17 12:05:28	3.7 GB		Graph
	Total swap space	2017-08-17 12:05:21	1.6 GB		Graph
	Used memory	2017-08-17 12:05:29	1.78 GB	+180 KB	Graph
	Used swap space	2017-08-17 12:05:21	0 B		Graph
▼	**Network interfaces** (2 Items)				
	Incoming network traffic on ens160	2017-08-17 12:05:47	2.09 Kbps	-192 bps	Graph
	Outgoing network traffic on ens160	2017-08-17 12:05:48	1.34 Kbps	+72 bps	Graph
▼	**Processes** (1 Item)				
	Number of processes	2017-08-17 12:05:10	103		Graph
▼	**Zabbix agent** (2 Items)				
	Agent ping	2017-08-17 12:05:57	Up (1)		Graph

Figura F.1: Recolección de información para un host

Incoming traffic on interface 1	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 2	2017-09-11 13:45:22	2.12 Kbps
Incoming traffic on interface 3	2017-09-11 13:45:22	1.54 Kbps
Incoming traffic on interface 4	2017-09-11 13:45:22	182.94 Kbps
Incoming traffic on interface 5	2017-09-11 13:45:22	182.21 Kbps
Incoming traffic on interface 9	2017-09-11 13:45:22	1.38 Kbps
Incoming traffic on interface 12	2017-09-11 13:45:22	14.42 Kbps
Incoming traffic on interface 14	2017-09-11 13:45:22	3.04 Kbps
Incoming traffic on interface 18	2017-09-11 13:45:22	25.69 Kbps
Incoming traffic on interface 19	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 21	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 23	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 24	2017-09-11 13:45:22	23.93 Mbps
Incoming traffic on interface 25	2017-09-11 13:45:22	4.09 Kbps
Incoming traffic on interface 26	2017-09-11 13:45:22	1.31 Kbps
Incoming traffic on interface 27	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 36	2017-09-11 13:45:22	17.28 Kbps
Incoming traffic on interface 38	2017-09-11 13:45:22	9.99 Kbps
Incoming traffic on interface 40	2017-09-11 13:45:22	10.9 Kbps
Incoming traffic on interface 42	2017-09-11 13:45:22	3.02 Kbps
Incoming traffic on interface 43	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 44	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 47	2017-09-11 13:45:22	0 bps
Incoming traffic on interface 48	2017-09-11 13:45:22	0 bps

Figura F.2: Recolección de información para un elemento de red

Host1: Version of zabbix_agent(d) running
2017-08-17 00:26:58 1502918818 3.2.6

Figura F.3: Recolección de la información durante un periodo acotado

Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags
14:46:02	Average	14:46:32			Host1	Service unavailable	30s	No	Done 1	

Step	Time	User	Details	Status	Info
Problem					
1	2017-08-17 14:46:04		Remote command	Executed	

Figura F.4: Acciones aplicadas para resolver un problema

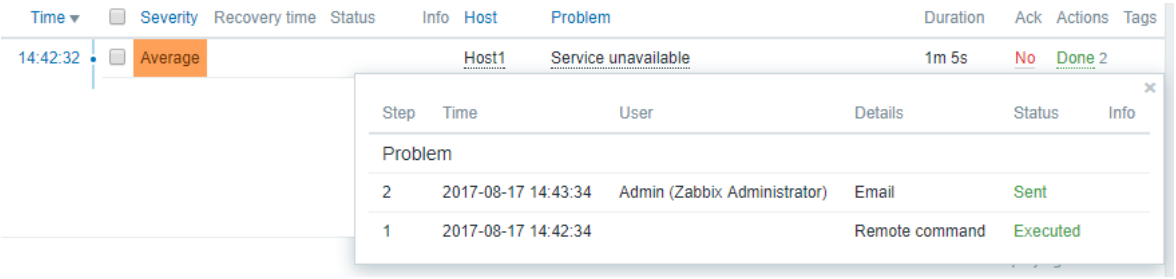

Figura F.5: Acciones aplicadas para resolver un problema

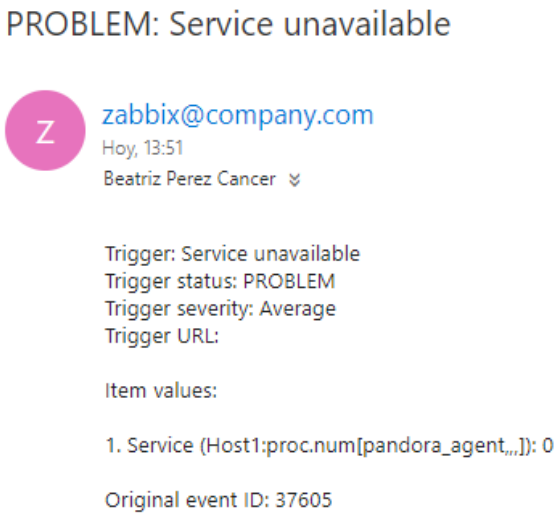

Figura F.6: Correo enviado al administrador

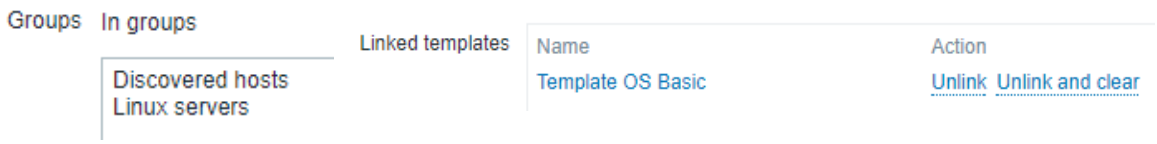

Figura F.7: Pertenencia a grupos y aplicación de la plantilla asociados en autodescubrimiento

Información recogida de logs de metamonitorización cada 5 minutos:

```
:::::::::::::: /var/log/monitoringLogs/availability.log ::::::::::::::
vie sep 15 10:45:01 CEST 2017 - Zabbix Server 10.250.14.107 is available
vie sep 15 10:50:01 CEST 2017 - Zabbix Server 10.250.14.107 is available
vie sep 15 10:55:01 CEST 2017 - Zabbix Server 10.250.14.107 is available
:::::::::::::: /var/log/monitoringLogs/cpu.log ::::::::::::::
vie sep 15 10:45:01 CEST 2017 - Load average in last 5 minutes is 0.01
vie sep 15 10:50:01 CEST 2017 - Load average in last 5 minutes is 0.01
vie sep 15 10:55:01 CEST 2017 - Load average in last 5 minutes is 0.04
:::::::::::::: /var/log/monitoringLogs/disk.log ::::::::::::::
vie sep 15 10:45:01 CEST 2017 - /: Total 14G Used 6,3G Free 7,2G Used 47%
```

```
vie sep 15 10:45:01 CEST 2017 - /boot: Total 1014M Used 229M Free 786M Used
23%
vie sep 15 10:50:01 CEST 2017 - /: Total 14G Used 6,3G Free 7,2G Used 47%
vie sep 15 10:50:01 CEST 2017 - /boot: Total 1014M Used 229M Free 786M Used
23%
vie sep 15 10:55:01 CEST 2017 - /: Total 14G Used 6,3G Free 7,2G Used 47%
vie sep 15 10:55:01 CEST 2017 - /boot: Total 1014M Used 229M Free 786M Used
23%
:::::::::::::: /var/log/monitoringLogs/memory.log ::::::::::::::
vie sep 15 10:45:01 CEST 2017 - Mem: Total 3791 Used 687 Free 2156 Free
56.87%
vie sep 15 10:45:01 CEST 2017 - Swap: Total 1639 Used 0 Free 1639 Free 100%
vie sep 15 10:50:01 CEST 2017 - Mem: Total 3791 Used 686 Free 2156 Free
56.87%
vie sep 15 10:50:01 CEST 2017 - Swap: Total 1639 Used 0 Free 1639 Free 100%
vie sep 15 10:55:01 CEST 2017 - Mem: Total 3791 Used 687 Free 2155 Free
56.84%
vie sep 15 10:55:01 CEST 2017 - Swap: Total 1639 Used 0 Free 1639 Free 100%
```

ICMP ping	2017-09-15 10:55:01	Up (1)

Figura F.8: Disponibilidad Zabbix Server

CPU load	2017-09-15 10:58:20	0.05

Figura F.9: CPU Zabbix Server

Free disk space on /	2017-09-15 10:54:45	7.13 GB
Free disk space on /boot	2017-09-15 10:54:46	785.12 MB
Total disk space on /	2017-09-15 10:24:49	13.38 GB
Total disk space on /boot	2017-09-15 10:24:50	1014 MB
Used disk space on /	2017-09-15 10:54:51	6.26 GB
Used disk space on /boot	2017-09-15 10:54:52	228.88 MB

Figura F.10: Espacio de disco Zabbix Server

Free memory	2017-09-15 10:54:39	2.09 GB
Free swap space	2017-09-15 10:54:33	1.6 GB
Total memory	2017-09-15 10:54:41	3.7 GB
Total swap space	2017-09-15 10:54:35	1.6 GB
Used memory	2017-09-15 10:54:42	1.61 GB
Used swap space	2017-09-15 10:54:36	0 B

Figura F.11: Memoria Zabbix Server

Figura F.12: Fallo de disponibilidad de Zabbix Server

Figura F.13: Fallo de CPU de Zabbix Server

Figura F.14: Fallo de memoria de Zabbix Server

Figura F.15: Fallo de disco de Zabbix Server