ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

UNIFICACIÓN DE HERRAMIENTAS DE MONITOREO DE COMPONENTES Y SERVICIOS DE TI

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

DIEGO ANDRÉS PORTERO LÓPEZ

diego.portero@epn.edu.ec

DIRECTOR: MSc. William Humberto Andrade Hinojosa

william.andrade@epn.edu.ec

Quito, junio 2021

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Diego Andrés Portero López, bajo mi supervisión.

MSc. William Humberto Andrade Hinojosa
DIRECTOR DE PROYECTO

DECLARACIÓN

Yo Diego Andrés Portero López, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Diego Andres Portero Lopez

DEDICATORIA

Para mis padres y toda mi familia, por todo su amor, tiempo y que han alegrado toda mi vida.

A memoria de mi tía Eulalia y mi tío Carlos que siempre estarán en nuestros corazones.

AGRADECIMIENTOS

A mis padres por todo su amor, apoyo y sacrificios, que me ha permitido cumplir este objetivo en mi vida, sin ellos no hubiera sido posible. Una vida no alcanza para agradecerles.

A mis abuelitos, tíos, primos y hermano de Ambato, Quito y Zaruma por todos los valores que me han enseñado y cariño que me han dado.

A mis amigos que han llenado esta etapa de mi vida con momentos inolvidables.

A Lenin, Suas, Cesar y Marco por hacer amena esta triste época de pandemia.

Gracias a mi tutor William por la ayuda para la culminación de este proyecto.

Diego

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

RESUMEN

El monitoreo de los componentes y servicios de una infraestructura de TI es muy importante para que el negocio de una empresa funcione correctamente y conocer el estado actual de la misma. Generalmente, la monitorización de una infraestructura de TI requiere que un administrador u operador este interactuando continuamente con la infraestructura para detectar cualquier problema que se produzca.

El presente trabajo de titulación busca elaborar una consola compuesta por varias herramientas open source para monitorizar el estado de varios componentes y servicios de una infraestructura de TI. Con esta consola se pretende solucionar las necesidades de los administradores y operadores de infraestructuras de TI.

Para el proceso de monitoreo en este trabajo se utilizó tres conjuntos de herramientas, Zabbix, Prometheus y ELK Stack. Para las notificaciones de los problemas que se encuentren en la infraestructura de TI se utilizará Telegram y correo electrónico. Se supervisará y controlará los recursos de varios componentes y servicios. Entre los servicios que se van a supervisar se encuentran: servidor HTTP apache, base de datos MySQL y servidor de correos Postfix. Además, esta solución se implementará en un entorno real de una empresa de desarrollo de software para llevar control de varios de sus recursos.

Palabras Claves: Monitoreo, Infraestructura de TI, Open Source, Métricas, Logs, Alertas.

ABSTRACT

Monitoring the components and services of an IT infrastructure is very important for a company's business to function properly and to know the current status of the infrastructure. Generally, monitoring an IT infrastructure requires that an administrator or operator is continuously interacting with the infrastructure to detect any problems that may occur.

This degree work aims to develop a console composed of several open source tools to monitor the status of various components and services of an IT infrastructure. This console is intended to solve the needs of IT infrastructure administrators and operators.

For the monitoring process in this work, three sets of tools were used, Zabbix, Prometheus and ELK Stack. For notifications of problems encountered in the IT infrastructure Telegram and email will be used. The resources of various components and services will be monitored and controlled. Among the services to be monitored are Apache HTTP server, MySQL database and Postfix mail server. In addition, this solution will be implemented in a real environment of a software development company to keep track of several of its resources.

Keywords: Monitoring, IT Infrastructure, Open Source, Metrics, Logs, Alerts.

CAPÍTULO 1. INTRODUCCIÓN

1.1. Planteamiento del Problema

Los servicios y sistemas de la tecnología de la información o TI de una empresa (mediana y grande) son diversos, con componentes diferentes en hardware, software, bases de datos, sistemas operativos, y cada uno de los mismos tiene un componente de monitoreo de su funcionamiento en plataformas de software propietarias y consolas diferentes.

Para los administradores y supervisores de TI tienen que disponer de varios monitores para visualizar cada consola de los diferentes componentes. Existe una gran cantidad de sistemas que permiten realizar la supervisión de componentes y servicios de una infraestructura de TI.

En caso de que estos sistemas usan software propietario, se requiere pagar una licencia para su uso y no existe posibilidad que el usuario pueda modificar el funcionamiento de este. También existen herramientas de monitoreo basados en software open source, diseñado para que sea accesible al público: todos pueden ver, modificar y distribuir el código fuente. A diferencia de software propietario que es desarrollado por una corporación, el software open source se desarrolla de manera descentralizada y colaborativa, en caso de necesitar ayuda técnica con el software open source existe una gran cantidad de información en foros y documentación mantenidos por la comunidad que lo desarrolla. Para solventar esta necesidad se plantea unificar varias herramientas open source de monitoreo de componentes y servicios de TI en una consola que permita gestionar todos los recursos de una infraestructura de TI.

1.2. Justificación Teórica

La tecnología de la información está ahora en el centro de cualquier negocio y se aplica a cada departamento, cada sección y cada empleado de una organización. Sin embargo, cuanto más grande es el negocio, más difícil puede ser monitorear y mantener la infraestructura de TI, por lo que es indispensable poseer herramientas de monitoreo [1].

Las herramientas de monitoreo se encargan de administrar equipos de alta importancia, permiten a los usuarios verificar y controlar eficazmente las operaciones que estos equipos realizan, a la vez de acceder a la información que estos generan [2]. La utilización de protocolos, variables de entorno, MIBS, agentes SNMP, puertos TCP/UDP son comunes a todos los elementos a ser monitoreados y deben ser agrupados por los diferentes variables y parámetros a monitorear. Una de las limitaciones en la implementación de herramientas de monitoreo de una infraestructura de TI son los costos de licencia para adquirirlas y usarlas. En el caso de herramientas de open source, son en su mayoría gratuitas y permiten modificar el código fuente según sea conveniente [3].

En Ecuador, según un estudio realizado por el INEC en el 2014, el 97,4% de las empresas investigadas utilizan al menos un programa open source, El 60,3% utilizan aplicaciones ofimáticas, el 40,3% sistemas operativos y el 25,8% de otro tipo de software open source. En el país en abril del 2008 se emitió el Decreto No. 1014, basado en cumplimiento de recomendaciones Internacionales que recomiendan el uso de estándares abiertos y software libre como herramientas informáticas con los objetivos de alcanzar la soberanía y autonomía tecnológica y alcanzar un ahorro significativo de recursos públicos [4].

1.3. Justificación Práctica

La aplicación concreta es el de implementar una plataforma de monitoreo, basada en open source, que permita agregar los distintos componentes de TI y sus parámetros a ser monitoreados. La consola permitirá llevar un control óptimo de los recursos de TI implementados en una infraestructura.

1.4. Justificación Metodológica

Existen métodos determinísticos, pruebas de rendimiento, elementos de análisis que se deben presentar para determinar los parámetros comunes a ser monitoreados y disponerlos en consolas unificadas. Se utilizará ITIL qué es el conjunto de buenas prácticas más aceptado y utilizado en el mundo [5].

1.5. Objetivos

1.5.1. Objetivos Generales

Unificar distintas herramientas de monitoreo de componentes y servicios de TI en una consola.

1.5.2. Objetivos Específicos

- Analizar las herramientas Open Source de monitoreo de servicios de TI.
- Establecer los parámetros de monitoreo de los componentes de TI.
- Definir los protocolos, puertos, llamadas de petición y esquemas de recolección de estado de variables de componentes de TI.
- Instalar y configurar herramientas de software de monitoreo que permitan comunicarse con los componentes a monitorear.
- Realizar las pruebas de monitoreo de varios componentes de TI en una consola.

CAPÍTULO 2. MARCO TEÓRICO

2.1. Componentes de un Sistema de Monitoreo

Comprender el estado de su infraestructura y sistemas es esencial para garantizar la confiabilidad y estabilidad de sus servicios. La información sobre el estado y el rendimiento de sus implementaciones no solo ayuda a reaccionar ante los problemas, sino que también brinda la seguridad para realizar cambios con confianza. Una de las mejores formas de obtener esta información es con un sistema de monitoreo robusto que recopila métricas, visualiza datos y alerta a los operadores cuando las cosas parecen estar dañadas [7].

Las métricas, el monitoreo y las alertas son conceptos interrelacionados que juntos forman la base de un sistema de monitoreo. Tienen la capacidad de proporcionar visibilidad sobre el estado de un sistema, ayudando a comprender las tendencias de uso o comportamiento y comprender el impacto de los cambios que realiza. Si las métricas caen fuera de los rangos esperados, estos sistemas pueden enviar notificaciones para pedirle al operador que eche un vistazo, y luego pueden ayudar a sacar información para ayudar a identificar las posibles causas [7].

2.1.1. Métricas

Las métricas representan las medidas sin procesar del uso o comportamiento de los recursos que se pueden observar y recopilar en todo un sistema. Estos pueden ser resúmenes de uso de bajo nivel proporcionados por el sistema operativo, o pueden ser tipos de datos de nivel superior vinculados a la funcionalidad o el trabajo específico de un componente. Algunas métricas se presentan en relación con la capacidad total, mientras que otras se representan como una tasa que indica el "nivel de actividad" de un componente [7].

A menudo, las métricas más fáciles para empezar son las que ya expone su sistema operativo para representar el uso de los recursos físicos subyacentes. Los datos sobre el espacio en disco, la carga de la CPU, el uso de intercambio, etc. Estos datos ya están disponibles, proporcionan valor de inmediato y pueden enviarse a un sistema de monitoreo sin mucho trabajo adicional. Muchos servidores web, servidores de bases de datos y otro software también proporcionan sus propias métricas que también se pueden transmitir [7].

Las métricas son útiles porque brindan información sobre el comportamiento y el estado de los sistemas, especialmente cuando se analizan en conjunto. Representan la materia prima utilizada por su sistema de monitoreo para construir una visión holística de su entorno, automatizar las respuestas a los cambios y alertar a los seres humanos cuando sea necesario. Las métricas son los valores básicos que se utilizan para comprender las tendencias históricas, correlacionar diversos factores y medir los cambios en su rendimiento, consumo o tasas de error [7].

2.1.2. Monitoreo

Si bien las métricas representan los datos en su sistema, el monitoreo es el proceso de recopilar, agrupar y analizar esos valores para mejorar el conocimiento de las características y el comportamiento de sus componentes. Los datos de varias partes de su entorno se recopilan en un sistema de monitoreo que es responsable del almacenamiento, agregación, visualización e inicio de respuestas automáticas cuando los valores cumplen con requisitos específicos [7].

En general, la diferencia entre métricas y monitoreo refleja la diferencia entre datos e información. Los datos se componen de hechos sin procesar, mientras que la información se produce mediante el análisis y la organización de datos para crear un contexto que proporcione valor. El monitoreo toma datos de métricas, los agrega y los presenta de varias maneras que permiten a los humanos extraer información de la colección de piezas individuales [7].

Los sistemas de monitoreo cumplen muchas funciones relacionadas. Su primera responsabilidad es aceptar y almacenar datos entrantes e históricos. Si bien los valores que representan el momento actual son útiles, casi siempre es más útil ver esos números en relación con los valores pasados para proporcionar contexto sobre los cambios y las tendencias. Esto significa que un sistema de monitoreo debe ser capaz de administrar datos durante períodos de tiempo, lo que puede implicar muestrear o agregar datos más antiguos [7].

En segundo lugar, los sistemas de supervisión suelen proporcionar visualizaciones de datos. Si bien las métricas pueden mostrarse y entenderse como valores o tablas individuales, los seres humanos son mucho mejores para reconocer tendencias y comprender cómo encajan los componentes cuando la información se organiza de una manera visualmente significativa. Los sistemas de monitoreo generalmente representan los componentes que miden con gráficos y paneles configurables. Esto hace posible comprender la interacción de variables complejas o cambios dentro de un sistema al echar un vistazo a una pantalla [7].

Una función adicional que proporcionan los sistemas de monitoreo es organizar y relacionar datos de varias entradas. Para que las métricas sean útiles, los administradores deben poder reconocer patrones entre diferentes recursos y entre grupos de servidores. Por ejemplo, si una aplicación experimenta un aumento en las tasas de error, un administrador debería poder usar el sistema de monitoreo para descubrir si ese evento coincide con el agotamiento de la capacidad de un recurso relacionado [7].

2.1.3. Logs

Los logs (en español registros) son cadenas de texto con marcas de tiempo asociadas a ellos para indicar cuando ocurrió un evento. Los logs contienen significativamente más datos que las métricas. Existen dos tipos de logs: Estructurados y No Estructurados [8].

Los logs no estructurados no tienen ninguna asignación explícita de significado a un campo determinado [8].

Los logs estructurados poseen un formato de mensaje predeterminado y coherente para los registros de las aplicaciones, que permite que se traten como conjuntos de datos en lugar de texto [9].

La recopilación de logs se puede realizar de dos formas diferentes, pero la más común es configurar el reenvío de registros en sus sistemas. Todos los principales sistemas operativos y demonios de registro admiten el reenvío de registros, incluido equipos de red. El reenvío de registros permite decirle a un sistema que envíen sus registros a otro lugar en lugar de dejarlos en el sistema localmente. Esto permite que se puedan analizar los registros de muchos sistemas desde un solo lugar en lugar de iniciar sesión en varios sistemas [8].

2.1.4. Alerta

La alerta es el componente de respuesta de un sistema de monitoreo que realiza acciones basadas en cambios en los valores de las métricas. Las definiciones de alertas se componen de dos componentes: una condición o umbral basado en métricas y una acción a realizar cuando los valores caen fuera de las condiciones aceptables [7].

Si bien los sistemas de monitoreo son increíblemente útiles para la interpretación e investigación activa, uno de los principales beneficios de un sistema de monitoreo completo es permitir que los administradores se desconecten del sistema. Las alertas le permiten definir situaciones que tienen sentido para administrar activamente, mientras confía en el monitoreo pasivo del software para observar las condiciones cambiantes [7].

Si bien notificar a las partes responsables es la acción más común para alertar, algunas respuestas programáticas también se pueden activar en función de las violaciones del umbral. Por ejemplo, una alerta que indica que necesita más CPU para procesar la carga actual se puede responder con un script que escala automáticamente esa capa de su aplicación. Si bien esto no es estrictamente una alerta, ya que no da como resultado una notificación, el mismo mecanismo del sistema de monitoreo a menudo también se puede usar para iniciar estos procesos [7].

Sin embargo, el objetivo principal de las alertas sigue siendo atraer la atención humana sobre el estado actual de sus sistemas. La automatización de respuestas es un mecanismo importante para garantizar que las notificaciones solo se activan en situaciones que requieran la consideración de un ser humano informado. La alerta en sí debe contener información sobre lo que está mal y dónde ir para encontrar información adicional. La persona que responde a la alerta puede usar el sistema de monitoreo y las herramientas asociadas, como archivos de registro, para investigar la causa del problema e implementar una estrategia de mitigación [7].

2.1.5. Tipos de Métricas

2.1.5.1. Basadas en Host

Estos serían cualquier dato relacionado con la evaluación del estado o el rendimiento de una máquina individual, sin tener en cuenta por el momento sus pilas de aplicaciones y servicios. Estos se componen principalmente del uso o rendimiento del sistema operativo o hardware, como:

- CPU
- Memoria
- Espacio de Disco
- Red

Estos pueden dar una idea de los factores que pueden afectar la capacidad de una sola computadora para permanecer estable o realizar un trabajo.

2.1.5.2. Aplicación

Estas son métricas relacionadas con unidades de procesamiento o trabajo que dependen de los recursos a nivel de host, como servicios o aplicaciones. Los tipos específicos de métricas a considerar dependen de lo que proporciona el servicio, qué dependencias tiene y con qué otros componentes interactúan. Las métricas en este nivel son indicadores del estado, el rendimiento o la carga de una aplicación:

- Tasa de Errores y Existo
- Fallo del Servicio y Reinicios
- Rendimiento y Latencia de las Respuestas
- El Uso de Recursos

Estos indicadores ayudan a determinar si una aplicación está funcionando correctamente y con eficiencia.

2.1.5.3. Red y Conectividad

Estos son indicadores importantes de disponibilidad externa, pero también son esenciales para garantizar que los servicios sean accesibles a otras máquinas para cualquier sistema que abarque más de una máquina. Las redes deben verificarse por su corrección funcional general y su capacidad para ofrecer el rendimiento necesario al observar:

- Conectividad
- Tasas de errores y pérdidas de paquetes
- Latencia
- Utilización del ancho de banda

Monitorear su capa de red puede ayudarle a mejorar la disponibilidad y la capacidad de respuesta de sus servicios internos y externos.

2.1.5.4. Grupo de Servidores

Si bien las métricas sobre servidores individuales son útiles, a escala, un servicio se representa mejor como la capacidad de una colección de máquinas para realizar un trabajo y responder adecuadamente a las solicitudes. Este tipo de métrica es, en muchos sentidos, una extrapolación de nivel superior de métricas de aplicaciones y

servidores, pero los recursos en este caso son servidores homogéneos en lugar de componentes a nivel de máquina. Algunos datos de los que quizás desee realizar un seguimiento son:

- Uso de recursos agrupados
- Indicadores de ajuste de escala
- Instancias degradadas

La recopilación de datos que resuman el estado de las colecciones de servidores es importante para comprender las capacidades reales de su sistema para manejar la carga y responder a los cambios.

2.1.5.5. Dependencias Externas

A menudo, los servicios proporcionan páginas de estado o una API para descubrir interrupciones del servicio, pero rastrearlas dentro de sus propios sistemas, así como sus interacciones reales con el servicio, puede ayudarle a identificar problemas con sus proveedores que pueden afectar sus operaciones. Algunos elementos que pueden ser aplicables para realizar un seguimiento en este nivel son:

- Estado y Disponibilidad del Servicio
- Tasas de Errores y Éxito
- Tasas de Ejecución y Costos Operativos
- Agotamiento de Recursos

Hay muchos otros tipos de métricas que pueden ser útiles para recopilar. Conceptualizar la información más importante en diferentes niveles de enfoque puede ayudar a identificar los indicadores que son más útiles para predecir o identificar problemas.

2.1.6. Recopilación de Métricas

Las métricas son el material primario procesado por los sistemas de monitoreo para construir una vista cohesiva de los sistemas a los cuales se realiza el seguimiento. Estar seguro de qué componentes vale la pena monitorear y qué características se

deben tomar en cuenta es el primer paso para diseñar un sistema que pueda brindar información confiable y procesable sobre el estado de sus software y hardware.

2.1.6.1. Señales del Monitoreo

Estas señales representan algunos de los factores más importantes a medir en un sistema de cara al usuario.

2.1.6.1.1. Latencia

La latencia es una medida del tiempo que lleva completar una acción. Dependiendo del componente existen varias formas de realizar esta medición, pero comúnmente se toma como referencia el tiempo de procesamiento, tiempo de respuesta o tiempo de viaje [10].

Esta medición brinda una medida concreta de cuánto tiempo tarda en completarse una tarea o acción específica. La captura de la latencia de varios componentes le permite construir un modelo holístico de las diferentes características de rendimiento de un sistema. Esto puede ayudar en encontrar un cuello de botella, comprender qué recursos requieren más tiempo para acceder y detectar cuando las acciones de repente toman más tiempo de lo esperado [10].

2.1.6.1.2. Tráfico

El tráfico mide la actividad de los componentes y sistemas. Se captura la demanda o carga de los servicios para entender cuánto trabajo está realizando actualmente su sistema [10].

Los números de tráfico altos o bajos sostenidos pueden indicar que el proceso necesita más recursos o que un problema impide que el tráfico se enrute correctamente. En la mayoría de los casos, las tasas de tráfico serán más útiles para ayudar a comprender los problemas que surgen a través de otras señales [10].

2.1.6.1.3. Errores

Es imprescindible realizar un seguimiento de los errores para comprender el estado de los componentes y la frecuencia con la que no responden a las solicitudes de manera adecuada [10].

Distinguir entre diferentes tipos de errores pueden facilitar la identificación de la naturaleza exacta de los problemas que afectan sus aplicaciones. Esto también brinda flexibilidad para las alertas. Es posible que se tenga que recibir una alerta de inmediato si aparece un tipo de error, pero para otros, es posible que no sea fundamental [10].

2.1.6.1.4. Saturación

La saturación mide la cantidad de un recurso determinado que se está utilizando. Los porcentajes o fracciones se utilizan con frecuencia con recursos que tiene una capacidad total clara, pero es posible necesitar otro tipo de medidas para otro tipo de recursos con un máximo menos definido [10].

Los datos de saturación proporcionan información sobre los recursos de los que depende un servicio o una aplicación para funcionar de forma eficaz. Ya que un servicio proporcionado por un componente puede ser consumido por otro, la saturación es una de las métricas de cola que muestra los problemas de capacidad de los sistemas subyacentes [10].

2.1.6.1.5. Medición de Datos en Todo el Entorno

Dado que los servicios a menudo se crean agregando capas de abstracción sobre componentes más básicos, las métricas deben diseñarse para agregar información en cada nivel de la implementación [10].

Algunos de los niveles más comunes en aplicaciones distribuidas son:

- Componentes de servidor individual
- Aplicaciones y servicios
- Colecciones de servidores
- Dependencias ambientales
- Experiencia de principio a fin

El orden anterior amplía el alcance y el nivel de abstracción con cada capa posterior.

2.1.6.1.6. Métricas para Componentes de Servidores Individuales

Las métricas más importantes para recopilar son aquellas relevantes para las computadoras subyacentes de las que depende el sistema. A pesar de que se hace un esfuerzo considerable en el desarrollo de software moderno para abstraer los componentes físicos y los detalles del sistema operativo de bajo nivel, cada servicio se basa en el hardware y los sistemas operativos subyacentes para hacer su trabajo. Por lo que, vigilar los recursos fundamentales de sus máquinas es el primer paso para comprender el estado de un sistema [10].

Para medir la CPU, las siguientes medidas pueden apropiarse:

- **Latencia**: Retraso promedio o máximo en el programador de la CPU
- **Tráfico**: Utilización de CPU
- **Errores**: Eventos de error específicos del procesador, CPU con fallas
- **Saturación**: Longitud de la cola de ejecución

Para la memoria:

- **Latencia**: (No existe un buen método de medición)
- **Tráfico**: Cantidad de memoria en uso
- **Errores**: Errores de falta de memoria
- **Saturación**: Eventos asesinos de OOM, uso de intercambio

Para el almacenamiento:

- **Latencia**: Tiempo de espera promedio para lectura y escritura
- **Tráfico**: Leer y escribir niveles de I/O
- **Errores**: Errores de sistema de archivos, errores de disco en /sys/devices
- **Saturación**: Profundidad de la cola de I/O

Para la red:

- **Latencia**: Cola de controladores de red

- **Tráfico**: Bytes o paquetes entrantes y salientes por segundo
- **Errores**: Errores del dispositivo de red. Paquetes descartados
- **Saturación**: Desbordamiento, paquetes descartados, segmentos retransmitidos.

Además, se debe tomar en cuenta las representaciones de recursos físicos, también es una buena idea recopilar métricas relacionadas con las abstracciones del sistema operativo que tienen límites impuestos.

2.1.6.2. Métricas para Aplicaciones y Servicios

En una capa superior nos encontramos con las aplicaciones y servicios que se ejecutan en los servidores. Estos programas utilizan los componentes individuales del servidor de capas más bajas como recursos para realizar los trabajos. Las métricas en este nivel nos ayudan a comprender el estado de las aplicaciones y servicios de host único [10].

Estas métricas muestran que tan bien las aplicaciones pueden realizar el trabajo que se les solicita. También ofrece información sobre qué recursos dependen las aplicaciones y que tan bien se manejan las limitaciones de estos [10].

Las métricas más importantes en esta capa dependen de las características de las aplicaciones, sus configuraciones y la carga de trabajo que se están ejecutando en cada máquina [10].

En caso de que una aplicación sirva a los clientes se puede tener el siguiente caso:

- **Latencia**: El tiempo para completar las solicitudes
- **Tráfico**: Número de solicitudes por segundo atendidas
- **Errores:** Errores de aplicaciones que ocurren al procesar solicitudes de clientes o acceder a recursos
- **Saturación**: El porcentaje o la cantidad de recursos que se utilizan en el momento

2.1.6.3. Métricas para Colecciones de Servidores y su Comunicación

La mayoría de los servicios, especialmente cuando se operan en un entorno de producción, abarcan varias instancias de varios servidores para aumentar el rendimiento y la disponibilidad. Este mayor nivel de complejidad agrega un área de superficie adicional que es importante monitorear. La computación distribuida y los sistemas redundantes pueden hacer que sus sistemas sean más flexibles, pero la coordinación basada en la red es más frágil que la comunicación dentro de un solo host [10].

Algunas señales que se pueden aplicar a sistemas distribuidos son:

- **Latencia**: Tiempo para que el grupo responda a las solicitudes, tiempo para coordinar o sincronizar con los demás dispositivos
- **Tráfico**: Número de solicitudes procesadas por los dispositivos por segundo
- **Errores**: Errores de aplicación que ocurren al procesar solicitudes de cliente, acceder a recursos o comunicarse con pares
- **Saturación**: La cantidad de recursos que se utilizan actualmente, la cantidad de servidores que funcionan actualmente a su capacidad, la cantidad de servidores disponibles.

2.1.6.4. Métricas Relacionadas con las Dependencias Externas y el Entorno de Implementación

Algunas de las métricas más valiosas para recopilar existen en los límites de las aplicaciones o servicios, fuera del control directo de los administradores. Algunas de las dependencias externas están relacionadas con el proveedor de alojamiento y cualquier servicio en el que la aplicación esté deseada para confiar. Son recursos que no se pueden administrar directamente, pero pueden comprometer la capacidad para garantizar el propio servicio [10].

Ya que las dependencias externas representan recursos críticos, una estrategia para mitigar casos de interrupciones totales es cambiar las operaciones a varios proveedores [10].

Algunas de las señales que se pueden tomar en las dependencias externas son:

- **Latencia**: Tiempo que lleva recibir una respuesta del servicio o proporcionar nuevos recursos de un proveedor
- **Tráfico**: Cantidad de trabajo que se envía a un servicio externo, la cantidad de solicitudes que se realizan a una API externa
- **Errores**: Tasas de error para solicitudes de servicio
- **Saturación**: Cantidad de recursos restringidos por cuenta

Estas métricas pueden ayudar a identificar problemas con las dependencias, alertar sobre agotamiento inminente de recursos y ayudar a mantener los gastos bajo control.

2.1.6.5. Métricas que Rastrean la Funcionalidad General de un Extremo a Otro

Las métricas de más alto nivel rastrean las solicitudes a través del sistema en el contexto del componente más externo con el que interactúan los usuarios. Dado que esto representa el primer punto de contacto con el sistema, la recopilación de métricas en este nivel brinda una aproximación de la experiencia general del usuario [10].

En esta capa suele ser más habitual la configuración de alertas. Para evitar un mal funcionamiento de las respuestas del servidor, las alertas, se reservan para escenarios que tienen un efecto negativo reconocible en la experiencia del usuario. Los problemas que surgen con estas métricas se pueden investigar profundamente utilizando las métricas recopiladas en otros niveles [10].

Algunas de las señales que se pueden encontrar en esta capa son similares a los servicios individuales, pero su alcance e importancia es mayor.

- **Latencia**: El tiempo para completar las solicitudes de los usuarios
- **Tráfico**: Número de solicitudes de usuarios por segundo
- **Errores**: Errores que ocurren al procesar solicitudes de clientes o acceder a recursos
- **Saturación**: El porcentaje o la cantidad de recursos que se utilizan

Ya que estas métricas son paralelas a las solicitudes de los usuarios, los valores que quedan fuera de los rangos aceptables para las métricas probablemente indiquen un impacto directo en el usuario.

2.2. Open Source

2.2.1. Definición

El software open source (código abierto) está diseñado de manera que su código sea accesible al público: todos pueden ver, modificar y distribuir el código de la forma que consideren conveniente. El desarrollo del software open source se realiza de forma descentralizada y colaborativa, dependiendo de la revisión y la colaboración de la comunidad. En muchos casos el software open source suele ser más económico, flexible y duradero que sus alternativas propietarias, ya que los encargados de su desarrollo es la comunidad y no solo un autor o una sola empresa [3].

El open source se ha convertido en un movimiento y una forma de trabajo que trasciende la producción del software. Este movimiento utiliza valores y modelos de producción descentralizada del software open source para encontrar soluciones a los problemas de las comunidades y los sectores [3].

2.2.2. Valores del Open Source

- **Revisión entre compañeros:** La comunidad de desarrolladores verifica y mejora los proyectos open source.
- **Transparencia:** Se puede revisar los cambios realizados en una aplicación.
- **Confiabilidad:** Los proyectos open source reciben actualizaciones constantes. Además de tener revisiones continuas entre varios integrantes.
- **Flexibilidad:** No es obligatorio usar el código de una manera específica.
- **Menor costo:** Las soluciones son más económicas y el código en sí es gratuito.
- **Sin dependencia de un solo proveedor**: Se puede trasladar a cualquier parte y usarlo para lo que sea en cualquier momento.
- **Colaboración abierta:** Las comunidades brindan la posibilidad de buscar ayuda, recursos y puntos de vista que trascienden el interés de un grupo o una empresa.

2.2.3. Software Open Source y Software Propietario

Tanto el software open source y el software propietario tienen ventajas y desventajas, por lo que para elegir una de las dos alternativas se debe realizar un análisis de cuál es el más conveniente para solucionar las necesidades de cada empresa. En una investigación realizada por Swati Dhir [11], se analizó las diferentes características y beneficios de estas dos alternativas y se concluyó que en su mayoría el software open source brinda más seguridad, soporte gratuito y facilidad de desarrollo en comparación al software propietario.

Basados en todos los beneficios que brindar, por la gran popularidad que está teniendo, y el menor costo del software open source como alternativa al software propietario se utilizará herramientas open source para desarrollar este proyecto.

2.3. ITIL

2.3.1. Definición

Information Technology Infrastructure Library (en español Biblioteca de Infraestructura de Tecnologías de Información) o ITIL por sus siglas en inglés. ITIL es un conjunto mundialmente reconocido de mejores prácticas para la Gestión de servicios de tecnología de la información (ITSM). Proporciona asesoramiento sobre la provisión de servicios de TI de calidad y de los procesos, funciones y demás capacidades necesarias para darles apoyo [12].

El gobierno británico creó ITIL cuando observó que la creciente dependencia en TI requería un conjunto de prácticas estándar. La norma ahora está publicada y pertenece a una organización conjunta entre una empresa privada, Capita y la Oficina del Gabinete del Reino Unido, denominada Axelos [12].

2.3.2. Importancia

Se depende de la TI para que la organización alcance sus objetivos. Influye sobre el funcionamiento y la comunicación, es un elemento fundamental de su forma de hacer negocios [12].

Se utiliza la TI para superar a los competidores, para llegar a mayores públicos, para ser más productivo y eficiente. De diversas formas, la TI es vital para mejorar los ingresos, reducir costos y mejorar la reputación de una empresa [12].

2.3.3. Ciclo de Vida del Servicio

El ciclo de vida del servicio según ITIL está organizado en las siguientes 5 etapas:

1. **Estrategia del Servicio:** Proporciona orientación sobre los principios que sustentan la práctica de gestión de servicios útiles para desarrollar políticas, directrices y procesos de gestión de servicios.
2. **Diseño del Servicio:** Proporciona orientación para el diseño y desarrollo de servicios y prácticas de gestión de servicios.
3. **Transición del Servicio:** Proporciona orientación para el desarrollo y mejora de las capacidades para la transición de servicios nuevos o modificados a la operación de servicios en ejecución.
4. **Operación del Servicio:** Proporciona orientación sobre cómo mantener la estabilidad en las operaciones de servicio, permitiendo cambios en el diseño, la escala, el alcance y los niveles de servicio.
5. **Mejora Continua del Servicio:** Proporciona una guía práctica para evaluar y mejorar la calidad de los servicios, la madurez general del ciclo de vida del servicio ITSM y sus procesos subyacentes.

2.3.1. Operación del Servicio

2.3.1.1. Monitoreo y Control

La medición y el control de los servicios se basan en un ciclo continuo de seguimiento, información y posterior actuación [13].

También es importante señalar que, aunque este ciclo tiene lugar durante el funcionamiento del servicio, proporciona una base para establecer la estrategia, diseñar y probar los servicios y lograr una mejora significativa. También es la base para la medición de la gestión del nivel de servicio (SLM) [13].

2.3.1.1.1. Definición

2.3.1.1.1.1. Monitoreo

La supervisión se refiere a la actividad de observar una situación para detectar los cambios que se producen a lo largo del tiempo [13].

En el contexto de operación del servicio, esto implica lo siguiente:

- Utilizar herramientas para supervisar el estado de los ítems de configuración clave y las actividades operativas clave
- Garantizar que se cumplen las condiciones especificadas y, en caso contrario, lanzar una alerta al grupo correspondiente
- Garantizar que el rendimiento o la utilización de un componente o sistema está dentro de un rango especificado
- Detectar tipos o niveles anormales de actividad en la infraestructura
- Detectar cambios no autorizados
- Garantizar el cumplimiento de las políticas de la organización
- Hacer un seguimiento de los resultados de la empresa y garantizar que cumplen los requisitos de calidad y rendimiento
- Hacer un seguimiento de cualquier información que se utilice para medir los indicadores clave de rendimiento (KPI).

2.3.1.1.1.2. Informar

La elaboración de informes se refiere al análisis, la producción y la distribución de los resultados de la actividad de supervisión [13].

- Utilizar herramientas para cotejar el resultado de la información de seguimiento que puede difundirse a diversos grupos, funciones o procesos
- Interpretar el significado de esa información
- Determinar el mejor uso de la información
- Garantizar que los responsables de la toma de decisiones tengan acceso a la información que les permita tomarlas
- Dirigir la información comunicada a la persona, grupo o herramienta adecuados.

2.3.1.1.1.3. Control

El control se refiere al proceso de gestionar la utilización o el comportamiento de un dispositivo, sistema o servicio. El control requiere tres condiciones:

- La acción debe garantizar que el comportamiento se ajuste a una norma o estándar definido
- Las condiciones que motivan la acción deben definirse, ser comprendidas y confirmadas
- La acción debe ser definida, aprobada y adecuada a estas condiciones.

En el contexto de la Operación de Servicios, el control implica lo siguiente:

- Utilizar herramientas para definir qué condiciones representan operaciones normales o anormales
- Regular el rendimiento de los dispositivos, sistemas o servicios
- Medir la disponibilidad
- Iniciar acciones correctivas, que podrían automatizarse, o manuales

2.3.1.1.2. Medición, Métricas y KPIs

2.3.1.1.2.1. Medición

La medición se refiere a cualquier técnica que se utilice para evaluar la extensión, dimensión o capacidad de un elemento en relación con una norma o unidad [13].

- La extensión se refiere al grado de cumplimiento o finalización.
- La dimensión se refiere al tamaño de un elemento.
- La capacidad se refiere a la capacidad total de un elemento.

2.3.1.1.2.2. Métrica

Las métricas se refieren a la evaluación cuantitativa y periódica de un proceso, sistema o función, junto con los procedimientos y herramientas que se utilizarán para realizar estas evaluaciones y los procedimientos para interpretarlas [13].

2.3.1.1.2.3. Indicadores Clave de Rendimiento (KPI)

Un KPI se refiere a un nivel de rendimiento específico y acordado que se utilizará para medir la eficacia de una organización o proceso. Los KPI son únicos por cada empresa dependiendo de las necesidades de la misma [13].

2.4. Herramientas de Monitoreo

2.4.1. Zabbix

Figura 1. Logo de la herramienta Zabbix

Zabbix es una solución de monitorización distribuida de clase empresarial de código abierto. Es un software que monitoriza numerosos parámetros de una red y la salud e integridad de servidores, máquinas virtuales, aplicaciones, servicios, bases de datos, sitios web, la nube y más. Zabbix utiliza un mecanismo de notificación flexible que permite a los usuarios configurar alertas por varios medios para prácticamente cualquier evento. Esto permite una rápida reacción a los problemas del servidor. Ofrece excelentes características de informes y visualización de datos. Zabbix es gratuito y se distribuye bajo la licencia pública general GPL versión 2 [14].

2.4.2. Nagios

Figura 2. Logo de la herramienta Nagios

Nagios es una herramienta de código abierto para la monitorización de sistemas. Esto significa que vigila los servidores y otros dispositivos de su red y se asegura de que funcionen correctamente. Comprueba constantemente si otras máquinas están funcionando correctamente. También verifica que varios servicios en esas máquinas

están funcionando correctamente. Se distribuye bajo la Licencia Pública General de GNU (GPL) Versión 2 [15].

2.4.3. MRTG

Figura 3. Logo de la herramienta MRTG

MRTG es el Multi Router Traffic Grapher (Graficador de tráfico de múltiples enrutadores en español), una pieza de open source liberada bajo la Licencia Pública General de GNU. Produce páginas web que muestran gráficos del uso del ancho de banda en los enlaces de red a escala diaria, semanal, mensual y anual. Esto puede ser una herramienta inestimable para diagnosticar problemas de red, ya que no sólo indica el estado actual de la red, sino que también permite compararlo visualmente con el historial de utilización de la red [16].

2.4.4. Pandora FMS

Figura 4. Logo de la herramienta Pandota FMS

Pandora FMS posee una licencia GNU General Public License (GPL). Es una solución de monitorización preparada para la empresa que proporciona una flexibilidad para que el departamento de TI pueda abordar tanto los problemas operativos inmediatos como los imprevistos, incluyendo la infraestructura y los procesos de TI. Permite que el negocio y las TI se adapten a las necesidades cambiantes a través de un enfoque flexible y rápido para la implementación de las TI y el negocio [17].

2.4.5. Net-SNMP

Figura 5. Logo de la herramienta Net

El Protocolo simple de administración de redes (SNMP) es un protocolo ampliamente utilizado para monitorear la salud y el bienestar de los equipos de red, equipos informáticos e incluso dispositivos como UPS. Net-SNMP es un conjunto de aplicaciones que se utilizan para implementar SNMP v1, SNMP v2c y SNMP v3 utilizando tanto IPv4 como IPv6. Esta suit posee una licencia BSD [18].

2.4.6. Prometheus

Figura 6. Logo de la herramienta Prometheus

Prometheus es un conjunto de herramientas de monitorización y alerta de sistemas de código abierto creado originalmente en SoundCloud. Desde su creación en 2012, muchas empresas y organizaciones han adoptado Prometheus. El proyecto cuenta con una comunidad de desarrolladores y usuarios muy activa. Ahora es un proyecto independiente de código abierto y se mantiene independientemente de cualquier empresa. Para enfatizar esto, y para aclarar la estructura de gobierno del proyecto, Prometheus se unió a la Cloud Native Computing Foundation en 2016 como el segundo proyecto alojado, después de Kubernetes [19]. Prometheus se publica bajo la licencia Apache 2.0 [20].

2.5. Zabbix

2.5.1. Host (Anfitrión)

Los Host en Zabbix son los dispositivos que se desean monitorear. La creación de un host es una de las primeras tareas de monitoreo de Zabbix [21].

2.5.2. Item (Elemento)

Los elementos son los que recopilan los datos de un host. Una vez que se haya configurado un host, debe agregar algunos Items de monitoreo para comenzar a obtener los datos reales [22].

Un Item es una métrica individual. Una forma de agregar rápidamente muchos elementos es adjuntar una plantilla predeterminada a un host. No obstante, para un rendimiento óptimo del sistema, se deben ajustar las plantillas para tener tantos elementos y una supervisión tan frecuente como sea realmente necesario [22].

2.5.3. Trigger (Disparador)

Los triggers son expresiones lógicas que evalúan los datos recopilados por los items y representan el estado actual del sistema. Los triggers permiten definir un umbral en el que el estado de los datos es aceptable. Por lo tanto, si un dato de entrada supera el estado aceptable, se ejecuta un trigger o cambia su estado a "problema" [23].

2.5.4. Event (Evento)

Existen varios tipos de eventos generados por Zabbix.

- Eventos Trigger: Cuando un trigger cambie de estado
- Eventos de descubrimiento: Cuando se detecte un host o servicio
- Eventos de registro automático: Cuando los agentes activos se registran automáticamente en el servidor
- Eventos internos: Cuando un elemento deja de ser compatible o un disparador pasa a un estado desconocido.

Los eventos tienen una marca de tiempo y pueden ser la base de acciones como el envío de notificaciones por correo electrónico, o por cualquier medio [24].

2.5.5. Graphs (Gráficos)

Con una gran cantidad de datos fluyendo hacia Zabbix, es mucho más fácil para los usuarios si pueden ver una representación visual de lo que está sucediendo. Para esto es la utilidad de los gráficos. Los gráficos permiten captar el flujo de datos de un vistazo, correlacionar problemas, descubrir cuándo sucedió un evento o hacer una presentación de cuando algo podría convertirse en un problema [25]. Existe varios tipos de gráficos como:

- Gráficos simples incorporados de los datos de un Item
- La posibilidad de crear gráficos personalizados más complejos
- Acceso a una comparación de varios items rápidamente en gráficos

2.5.6. Template (Plantilla)

Un template es un conjunto de entidades que se pueden aplicar cómodamente a varios hosts. Las entidades pueden ser:

- Items
- Trigger
- Gráficos
- Aplicaciones
- Template
- Reglas de Descubrimientos
- Escenarios Web

Como muchos hosts en la vida real son idénticos o bastante similares, naturalmente se deduce que el conjunto de entidades, que se ha creado para un host, puede ser útil para muchos [26].

Otro beneficio de usar plantilla es cuando se debe realizar un cambio para todos los hosts, solo es necesario cambiar algo a nivel de plantilla una vez para propagar el cambio a todos los hosts vinculados [26].

2.5.7. Notification (Notificación)

Una vez configurado algunos Items y Triggers, y en caso de que ocurra algún evento como resultado de un cambio de estado de los Triggers, es el momento de utilizar las

acciones. En el caso de que ocurra problemas, es fundamental que todas las personas interesadas estén informadas [27].

Es por eso por lo que enviar notificaciones es una de las principales acciones que ofrece Zabbix. En las configuraciones se puede definir quién y cuándo se debe notificar sobre un evento determinado [27].

2.6. Prometheus

Prometheus es un sistema de monitoreo open source basado en métricas. Tiene un modelo de datos simple pero poderoso y un lenguaje de consulta que le permite analizar el desempeño de las aplicaciones e infraestructura. Está escrito principalmente en Go y posee una licencia Apache 2.0. Su desarrollo se realiza de forma colaborativa a nivel mundial. Se calcula que, a partir de 2018, decenas de miles de organizaciones están utilizando Prometheus en producción [28].

Para instrumentar un software a medida, existe librerías cliente en todos los lenguajes y runtimes populares, incluido Go, Java/JVM, C#/.Net, Python, Ruby, Node.js, Haskell, Erlang y Rust. Software como Kubernete y Docker ya están equipados con librerías cliente de Prometheus. Para el software de terceros que expone métricas en un formato que no es de Prometheus, existen cientos de integradores disponibles [28].

2.6.1. PromQL

PromQL es el lenguaje de consulta de Prometheus. Las etiquetas con una parte clave de PromQL y se las pueden usar no solo para realizar agregaciones arbitrarias, sino también para unir diferentes métricas para operaciones aritméticas. Operaciones aritméticas permitidas:

- Medir
- Contar
- Resumir
- Histogramas

2.6.2. Exportador

Un exportador es una pieza de software que se implementa junto a la aplicación de la que se debe obtener métricas. Recibe solicitudes de Prometheus, recopila los datos necesarios de la aplicación, los transforma al formato correcto y finalmente los devuelve en una respuesta a Prometheus [28].

Debido al tamaño de la comunidad de Prometheus, existe una gran cantidad de exportadores que se pueden utilizar con poco esfuerzo.

2.6.3. Alertmanager

Prometheus permite definir condiciones en forma de expresiones PromQL que se evalúan continuamente, y cualquier serie de tiempo resultante se convierte en alertas. Una vez que se activa una alerta, Prometheus lo comunica a un Alertmanager [28].

Alertmanager es responsable de recibir todas las alertas de todos los servidores de Promethus y convertirlas en notificaciones como correos electrónicos, mensajes de chat, entre otros medios [28].

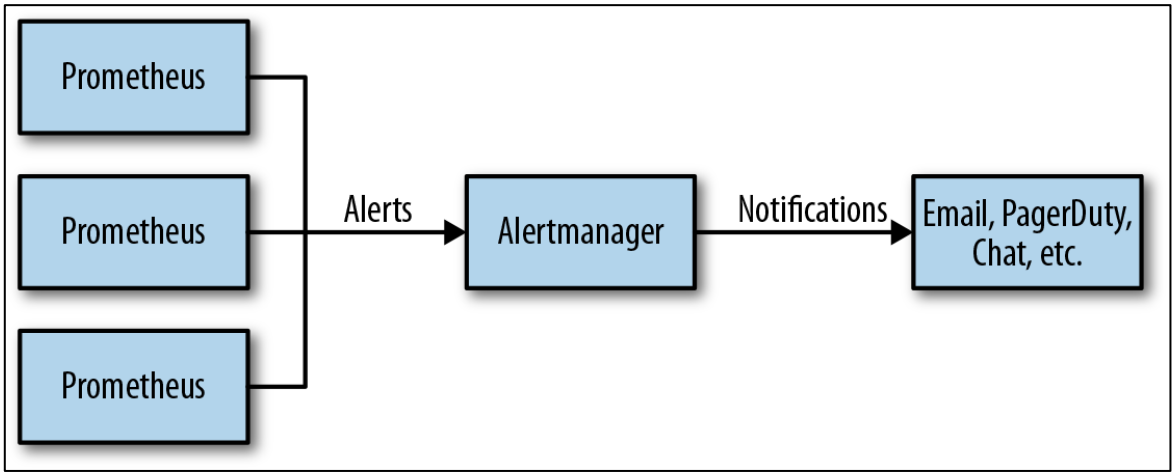


Figura 7. Arquitectura de Alertmanager

2.6.4. Grafana

Grafana es una herramienta popular con la que se puede crear Dashboards (Paneles de Control en español) para muchos sistemas de monitoreo y no monitoreo. Es una herramienta recomendada para crear paneles de control cuando se usa Prometheus, ya que mejora continuamente su compatibilidad con Prometheus.

2.6.5. Arquitectura de Prometheus

En la siguiente imagen se puede ver un diagrama con la arquitectura de Prometheus y algunos de sus componentes de ecosistema:

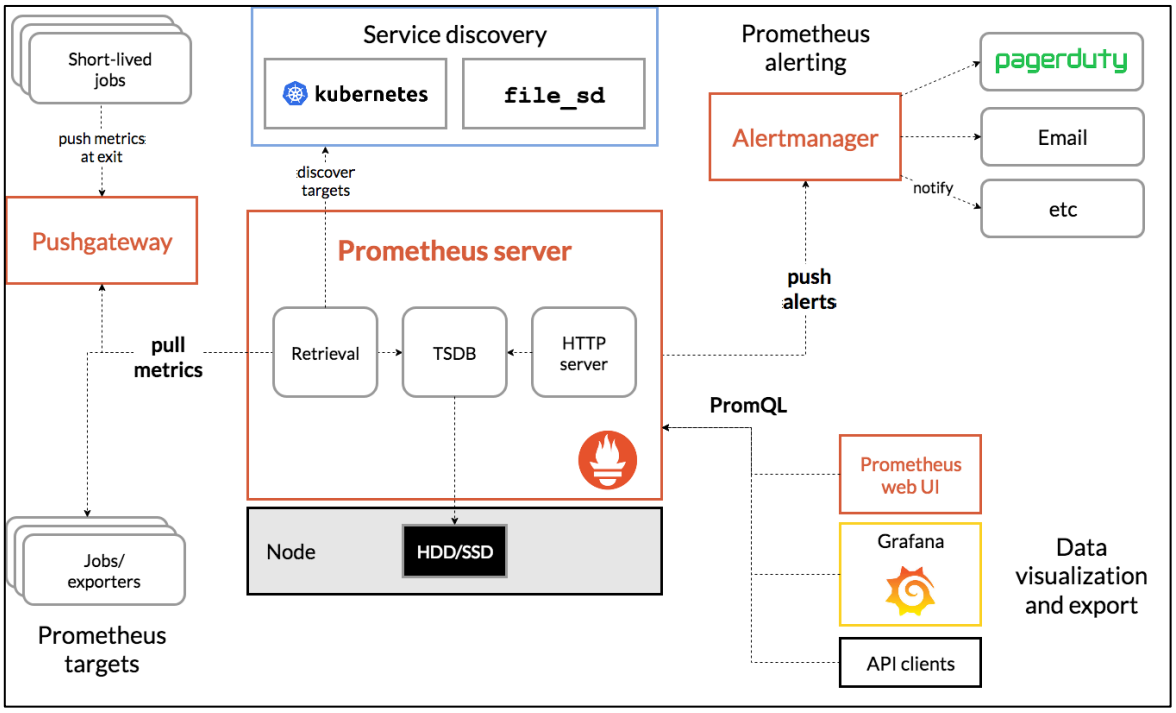


Figura 8. Diagrama con la arquitectura de Prometheus

2.7. ELK Stack

El ELK Stack es una solución completa de análisis de logs, construida sobre una combinación de tres herramientas de código abierto. Elasticsearch, Logstash y Kibana [29]. Estas herramientas intentan abordar algunos de los problemas y desafíos más comunes en el análisis de logs, los cuales son. Algunos de estos problemas son:

- Formato de logs no consistente
- Logs descentralizados
- Alta curva de aprendizaje

El ELK Stack es mantenido y apoyado activamente por la compañía llamada Elastic.

2.7.1. Elasticsearch

Elasticsearch es un motor de búsqueda distribuido de código abierto basado en Apache Lucene y publicado bajo una licencia Apache 2.0. Proporciona escalabilidad horizontal, confiabilidad y capacidad multiusuario para búsquedas en tiempo real. Las funciones de Elasticsearch están disponibles en JSON a través de un API Rest. Las capacidades de búsqueda están respaldadas por un motor Apache Lucene sin esquema, que permite indexar datos dinámicamente sin conocer la estructura de antemano. Elasticsearch puede lograr respuestas de búsqueda rápidas porque utiliza la indexación para buscar en los textos [29].

2.7.2. Logstash

Logstash es un canalizador de datos que ayuda a recopilar, tratar y analizar una gran variedad de datos y eventos estructurados y no estructurados generados en varios sistemas. Proporciona complementos para conectarse a varios tipos de fuentes de entrada y plataformas, y está diseñado para procesar de manera eficiente registros, eventos y fuentes de datos no estructurados para su distribución en una variedad de salidas con el uso de complementos de salida [29].

2.7.3. Kibana

Kibana es una plataforma de visualización de datos con licencia Apache 2.0, que ayuda a visualizar cualquier tipo de datos estructurados y no estructurados almacenados en índices de Elasticsearch. Kibana está escrito completamente en HTML y JavaScript. Utiliza las poderosas capacidades de búsqueda e indexación de Elasticsearch expuestas a través de su API Rest para mostrar gráficos poderosos para los usuarios finales. Desde la inteligencia empresarial básica hasta la depuración en tiempo real, Kibana desempeña su función al exportar datos a través de hermosos histogramas, mapas geográficos, gráficos circulares, tablas, etc [29].

2.7.4. Canalización de Datos ELK

Una típica canalización de datos con ELK se parece a esto

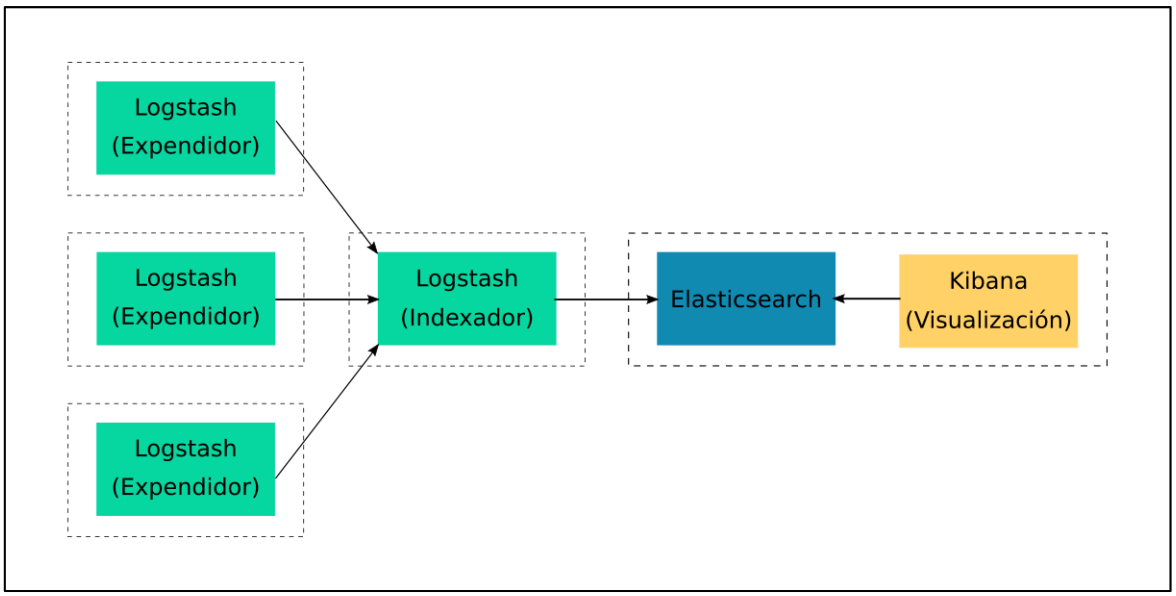


Figura 9. Arquitectura de Logstash

En esta arquitectura, los logs de varios servidores de aplicaciones envían a través del remitente de Logstash a un indexador de Logstash centralizado. El indexador de Logstash enviará datos a un cluster de Elasticsearch, que será consultado por Kibana para mostrar visualizaciones y crear paneles sobre los datos de los logs.

2.8. Aplicaciones

2.8.1. Servidor HTTP Apache

2.8.1.1. HTTP

Los navegadores, servidores y las aplicaciones web relacionadas del mundo se comunican entre sí a través de Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto) o HTTP. Es el lenguaje común de la Internet global moderna [30].

Miles de millones de archivos navegan por internet todos los días. HTTP mueve la mayor parte de esta información de manera rápida, conveniente y confiable desde los servidores web de todo el mundo a los navegadores web en los escritorios de las personas [30].

Debido a que HTTP utiliza protocolos de transmisión de datos confiables, garantiza que sus datos no se dañan o codificaran en tránsito, incluso cuando provengan del otro lado del mundo. Esto es muy bueno para el usuario ya que este no se debe preocupar por la integridad de la información. Y para los desarrolladores también es bueno ya que no se tiene que preocupar de que la comunicación HTTP se destruya, duplique o distorsiones en tránsito [30].

2.8.1.2. Cliente y Servidor Web

El contenido web se almacena en servidores web. Estos servidores web hablan el protocolo HTTP, por lo que también se les llama servidores HTTP. Estos servidores almacenan los datos de internet y proporcionan los datos cuando los solicitan los clientes HTTP. Los clientes envían solicitudes HTTP a los servidores y los servidores devuelven los datos solicitados en respuestas HTTP [30].

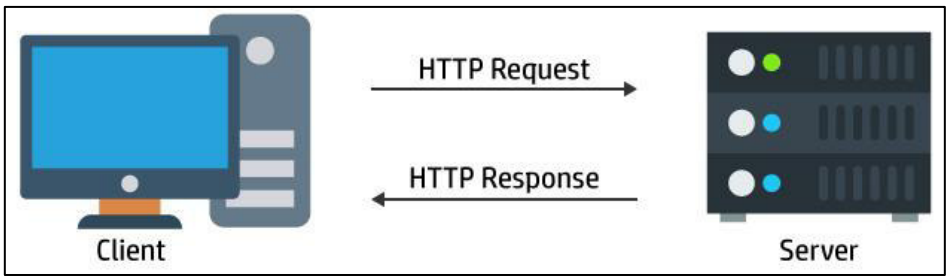


Figura 10. Petición de cliente-servidor

2.8.1.3. Recursos

2.8.1.3.1. URIs

Cada recurso del servidor web tiene un nombre, por lo que los clientes pueden señalar en qué recurso están interesados. El nombre del recurso del servidor se denomina uniform resource identifier (identificador uniforme de recursos en español) o URI [30].

2.8.1.3.2. URLs

El localizador uniforme de recursos (localizador uniforme de recursos en español) o URL es la forma más común de identificador de recursos. La URL describe la ubicación específica de un recurso en un servidor en particular [30].

La mayoría de las URLs sigue un formato estandarizado de tres partes principales:

- La primera parte se llama esquema y describe el protocolo utilizado
- La segunda parte proporciona la dirección de Internet del servidor
- El resto nombre a un recurso en el servidor web

2.8.1.4. Transacciones

2.8.1.4.1. Métodos

HTTP admite varios comandos de solicitudes diferentes, llamados métodos HTTP. Cada mensaje de solicitud HTTP tiene un método. El método le dice al servidor que acción realizar [30].

Método HTTP	Descripción
GET	Enviar el recurso nombrado desde el servidor al cliente
PUT	Almacenar datos del cliente en un recurso del servidor
DELETE	Eliminar el recurso nombrado en un servidor
POST	Envía los datos del cliente a una aplicación del servidor
HEAD	Envía solo los encabezados HTTP de la respuesta para el recurso nombrado

Tabla 1. Métodos HTTP

2.8.1.4.2. Código de estado

Cada mensaje de respuesta HTTP vuelve con un código de estado. El código de estado es un código numérico de tres dígitos que le dice al cliente si la solicitud se realizó correctamente o si se requiere otra acción [30].

Código de estado HTTP	Descripción
200	Documento devuelto correctamente
302	Redireccionamiento
404	No encontrado. No se pudo encontrar este recurso

Tabla 2. Códigos de estado HTTP

2.8.1.5. Mensajes

Los mensajes HTTP son secuencias de caracteres simples y orientadas a líneas. Ya que son textos sin formato, son fáciles de leer y escribir para los humanos.

Los mensajes HTTP enviados desde los clientes web a servidores web se denominan mensajes de solicitud. Los mensajes de los servidores a los clientes se denominan mensajes de respuesta [30].

2.8.1.6. Proyecto Servidor HTTP Apache

Este proyecto es un esfuerzo de desarrollo de software colaborativo destinado a crear una implementación de código fuente sólida, de calidad comercial, con funciones y disponible de forma gratuita de un servidor HTTP. El proyecto es administrado conjuntamente por un grupo de voluntarios ubicados en todo el mundo, que utilizan Internet y la Web para comunicarse, planificar y desarrollar el servidor y su documentación relacionada. Este proyecto es parte de Apache Software Foundation [31].

2.8.1.7. Indicadores de Rendimiento

Las siguientes métricas se las obtuvo de datadoghq, la cual es una plataforma de seguridad y monitoreo para aplicaciones en la nube [32].

2.8.1.7.1. Métricas de Rendimiento y Latencia

Nombre	Descripción	Tipo de Métrica	Disponibilidad
Tiempo de procesamiento de la solicitud	Microsegundos necesarios para procesar una solicitud del cliente	Trabajo: Rendimiento	Apache access log
Tasa de solicitudes	Número medio de solicitudes de clientes por segundo	Trabajo: Rendimiento	mod_status
Bytes	Total, de bytes servidos*	Otros	mod_status

Tabla 3. Métricas de rendimiento y latencia de un servidor HTTP

Alertas:

- Tiempo de procesamiento de la solicitud
- Tasa de solicitudes

2.8.1.7.2. Utilización de Recursos y Métricas de Actividad

Nombre	Descripción	Tipo de Métrica	Disponibilidad

Trabajadores ocupados	Número total de hilos/procesos de trabajo ocupados	Recurso: Utilización	mod_status
Trabajadores ociosos	Número total de hilos/procesos de trabajo inactivos	Recurso: Utilización	mod_status
Conexiones asíncronas: escritura	Número de conexiones asíncronas en estado de escritura	Recurso: Utilización	mod_status
Conexiones asíncronas: keep-alive	Número de conexiones asíncronas en estado de mantenimiento de la vida	Recurso: Utilización	mod_status
Conexiones asíncronas: cierre	Número de conexiones asíncronas en estado de cierre	Recurso: Utilización	mod_status

Tabla 4. Utilización de recursos y métricas de actividades de un servidor HTTP

2.8.1.7.3. Métricas de Recursos a Nivel de Host

Estos se componen principalmente del uso o rendimiento del sistema operativo o hardware, como:

- CPU
- Memoria
- Espacio de Disco
- Red

2.8.1.7.4. Errores

Nombre	Descripción	Tipo de Métrica	Disponibilidad
Tasa de error de los clientes	Índice de errores de cliente 4xx	Trabajo: Errores	Apache access log
Tasa de error del servidor	Tasa de errores 5xx del lado del servidor	Trabajo: Errores	Apache access log

Tabla 5. Errores de un servidor HTTP

2.8.2. Base de Datos MySQL

2.8.2.1. Base de datos

Una base de datos es una colección organizada de información estructurada, o datos, que normalmente se almacenan electrónicamente en un sistema informático. Una base de datos suele estar controlada por un Database Management System (sistema de gestión de bases de datos en español) o DBMS [33].

Actualmente, la forma más común de modelar una base de datos es en filas y columnas en una serie de tablas para que el procesamiento y consultas sean eficientes. La mayoría de las bases de datos utilizan un Structured Query Language (lenguaje de consulta estructurado en español) o SQL [33].

2.8.2.2. SQL

SQL es un lenguaje de programación utilizado por casi todas las bases de datos relacionales para consultar, manipular y definir datos, y proporcionar control de acceso [33]. Ya que SQL incluye declaraciones para modificar la base de datos y para declarar esquemas de la base de datos, sirve como Data-Manipulation Language (lenguaje de manipulación de datos en español) o DLL y Data-Definition Language

(lenguaje de definición de datos en español) o DML. Además, SQL estandariza muchos otros comandos de base de datos [8].

2.8.2.3. DBMS

Un Database Management System o DBMS consiste en una colección de datos interrelacionados y una colección de programas para acceder a esos datos. Su objetivo principal es proporcionar un entorno que sea conveniente y eficiente para que las personas lo utilicen para recuperar y almacenar información [34].

2.8.2.4. Bases de Datos Relacional

El modelo relaciones es el modelo de datos principal en una gran cantidad de aplicaciones comerciales de procesamiento de datos. Alcanzando su posición debido a su simplicidad, lo que facilita el trabajo del programador, en comparación con otros modelos de datos [34].

Una base de datos relacional consta de una colección de tablas, a las cuales se les asigna un nombre único. Por lo generar una fila en una tabla representa una relación entre un conjunto de valores. En el modelo relacional, el término relación se usa para referirse a una tabla, mientras que el término tupla se usa para referirse a una fila [34].

2.8.2.5. MySQL

MySQL es el sistema de administración de bases de datos SQL de código libre más popular, está desarrollado, distribuido y respaldado por Oracle Corporation [43]. Algunas características de MySQL son:

- Es un sistema de gestión de base de datos.
- Es una base de datos relacional.
- Es de código abierto.
- Es muy rápido, confiable, escalable y fácil de usar.
- Funciona en sistemas cliente servidor o embebidos
- Hay una gran cantidad de software desarrollado por la comunidad está disponible.

2.8.2.6. Indicadores de Rendimiento

Las siguientes métricas se las obtuvo de datadoghq, la cual es una plataforma de seguridad y monitoreo para aplicaciones en la nube [35].

2.8.2.6.1. Rendimiento de las consultas

Nombre	Descripción	Tipo de Métrica	Disponibilidad
Consultas	Recuento de sentencias ejecutadas	Trabajo: Rendimiento	Variable de estado del servidor
Com_select	Declaraciones SELECT	Trabajo: Rendimiento	Variable de estado del servidor
Escritura	Inserts, updates, o deletes	Trabajo: Rendimiento	Calculado a partir de las variables de estado del servidor

Tabla 6. Rendimiento de las consultas de una base de datos MySQL

2.8.2.6.2. Rendimiento de la ejecución de la consulta

Nombre	Descripción	Tipo de Métrica	Disponibilidad
Tiempo de ejecución de la consulta	Tiempo medio de ejecución, por esquema	Trabajo: Rendimiento	Consulta del esquema de rendimiento

Errores de consulta	Número de sentencias SQL que generan errores	Trabajo: Error	Consulta del esquema de rendimiento
Slow_queries	Número de consultas que superan el límite de tiempo de consulta configurado	Trabajo: Rendimiento	Variable de estado del servidor

Tabla 7. Rendimiento de la ejecución de las consultas de una base de datos MySQL

2.8.2.6.3. Conexiones

Nombre	Descripción	Tipo de Métrica	Disponibilidad
Threads_connected	Conexiones abiertas actualmente	Recurso: Utilización	Variable de estado del servidor
Threads_running	Conexiones en curso	Recurso: Utilización	Variable de estado del servidor
Connection_errors_internal	Recuento de conexiones rechazadas por error del servidor	Recurso: Error	Variable de estado del servidor

Aborted_connects	Recuento de intentos de conexión fallidos con el servidor	Recurso: Error	Variable de estado del servidor
Connection_errors_max_connections	Recuento de conexiones rechazadas debido al límite de max_connections	Recurso: Error	Variable de estado del servidor

Tabla 8. Conexiones de una base de datos MySQL

2.8.2.6.4. Uso de la reserva de búferes

Nombre	Descripción	Tipo de Métrica	Disponibilidad
Innodb_buffer_pool_pages_total	Número total de páginas en la reserva de búferes	Recurso: Utilización	Variable de estado del servidor
Buffer pool utilization	Relación entre las páginas utilizadas y el total de páginas en la reserva de búferes	Recurso: Utilización	Calculado a partir de las variables de estado del servidor

Innodb_buffer_pool_read_requests	Solicitudes realizadas a la reserva de búferes	Recurso: Utilización	Variable de estado del servidor
Innodb_buffer_pool_reads	Solicitudes que la reserva de búferes no ha podido satisfacer	Recurso: Saturación	Variable de estado del servidor

Tabla 9. Uso de reserva de búferes de una base de datos MySQL

2.8.3. Servidor de Correo Postfix

2.8.3.1. SMTP

Simple Mail Transfer Protocol (Protocolo Simple de Transferencia de Correo en español) o SMTP es un protocolo estándar de Internet para la transmisión de correo electrónico, generalmente se utiliza los puertos 587 o 465 [36].

2.8.3.2. POP

Post Office Protocol (Protocolo de Oficina Postal en español) o POP es un protocolo de capa de aplicación que se utiliza para recuperar mensajes de un servidor de correo. La versión más actual y utilizada del protocolo es POP3, que se basa en TCP y opera en el puerto 110. Con POP3 el correo se entrega y almacena en un servidor de correo hasta que un usuario se conecta, a través de un correo electrónico [37].

2.8.3.3. IMAP

Internet Message Access Protocol (Protocolo de acceso a mensajes de Internet en español) o IMAP es un protocolo de recuperación de correo que se desarrolló como alternativa más sofisticada a POP3. Su versión más actual es IMAP4. IMAP4 puede reemplazar a POP3 sin que el usuario tenga que cambiar los programas de correo electrónico. La mayor ventaja de este protocolo es que los usuarios pueden

almacenar mensajes en el servidor de correo, en lugar de descargarlos en la máquina local [37].

2.8.3.4. MTA

Message Transfer Agent (Agente de Transferencia de Mensajes en español) o MTA es un software que transfiere mensajes de correo electrónico de una computadora a otra usando SMTP [38].

MTA tiene varias formas de comunicarse con otros servidores de correo haciendo uso del SMTP:

- Recibe los mensajes desde otro MTA. Actúa como servidor de otros servidores.
- envía los mensajes hacia otro MTA. Actúa como un cliente de otros servidores.
- Actúa como intermediario entre un Mail Submission Agent y otro MTA.

2.8.3.5. Componentes del Correo Electrónico

La siguiente figura muestra los componentes que intervienen en una simple transmisión de correo electrónico del remitente al destinatario. Los MTA hacen la mayor parte del trabajo para que un mensaje sea entregado de un sistema a otro. Cuando se recibe una solicitud para aceptar un mensaje de correo electrónico, el MTA determina si debe aceptar el mensaje o no. Un MTA suele aceptar mensajes para sus propios usuarios locales; para otros sistemas a los que sabe cómo reenviar; o para mensajes de usuarios, sistemas o redes que están autorizados a retransmitir el correo a otros destinos. Una vez que el MTA acepta un mensaje, tiene que decidir qué hacer con él. Puede que entregue el mensaje a un usuario de su sistema o que tenga que pasarlo a otro MTA. Si el MTA no puede entregar el mensaje o pasarlo, lo devuelve al remitente original o lo notifica al administrador del sistema.

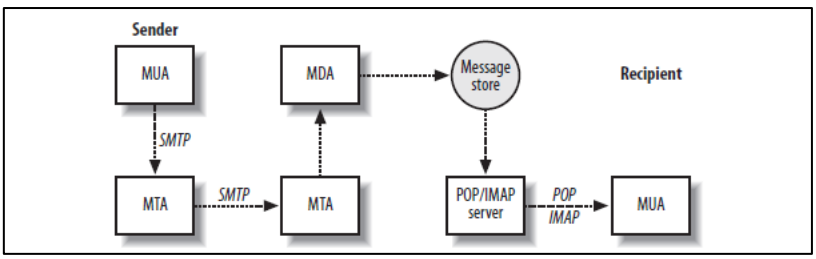


Figura 11. Componentes que intervienen en la transmisión de un correo electrónico

2.8.3.6. Postfix

Postfix es un Mail Transfer Agent (Agente de transferencia de mensajes en español) o MTA. Es decir, el software que utilizan los servidores de correo para enrutar los correos electrónicos. Fue escrito por un conocido experto en ciberseguridad Wietse Vienema [39]. Está publicado bajo la licencia pública IBM 1.0, la cual es una licencia de software libre [40].

Postfix recibe y entrega mensajes de correo electrónico a través de la red mediante el protocolo SMTP. Para la entrega local, el agente de entrega local de Postfix puede depositar los mensajes directamente en un almacén de mensajes o entregar un mensaje a un agente de entrega de correo especializado [39].

2.8.3.7. Dovecot

Dovecot es un servidor de correo electrónico IMAP y POP3 de código abierto para sistemas tipo Linux/UNIX, escrito pensando principalmente en la seguridad. Dovecot es una excelente opción tanto para instalaciones pequeñas como grandes. Es rápido, sencillo de configurar, no requiere ninguna administración especial, además de utilizar muy poca memoria [41].

CAPÍTULO 3. DESARROLLO DE LA SOLUCIÓN

3.1. Arquitectura de la consola

Para realizar el trabajo se escogió 3 herramientas, Zabbix, Prometheus y ELK Stack. Y se definió el diseño de la arquitectura de la consola de monitorización.

3.1.1. Zabbix

La arquitectura que se usó en la implementación de Zabbix en el ambiente simulado cuenta con los siguientes componentes:

- **Agente Zabbix:** Es un agente nativo de Zabbix, desarrollado en lenguaje C, puede ejecutarse en varias plataformas compatibles, como Linux, UNIX y Windows, y recoger datos como el uso de la CPU, la memoria, el disco y la interfaz de red de un dispositivo. Por defecto utiliza el puerto 10050.

- **Servidor Zabbix:** Es el proceso central del software Zabbix. Realiza el sondeo y la captura de datos, calcula los disparadores y envía notificaciones a los usuarios. Es el componente central al que los agentes y proxies de Zabbix reportan datos sobre la disponibilidad e integridad de los sistemas. Por defecto utiliza el puerto 10051.
- **PostgresSQL:** Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en POSTGRES, versión 4.2. Es donde se almacenan todas la configuraciones, métricas y logs de Zabbix, también se puede utilizar una base de datos MySQL. Por defecto utiliza el puerto 5432.
- **Frontend Zabbix:** La interfaz Web de Zabbix está implementada en PHP y requiere de un servidor Apache como contenedor Web.
- **Correo electrónico:** Es un servicio de red que permite a los usuarios enviar y recibir mensajes mediante redes de comunicación electrónica. La instalación y configuración del servicio Postfix se encuentra en el ANEXO 1.
- La comunicación entre cada agente Zabbix y el servidor central se realiza a través de los puertos 10050 y 10051 por medio del protocolo TCP.

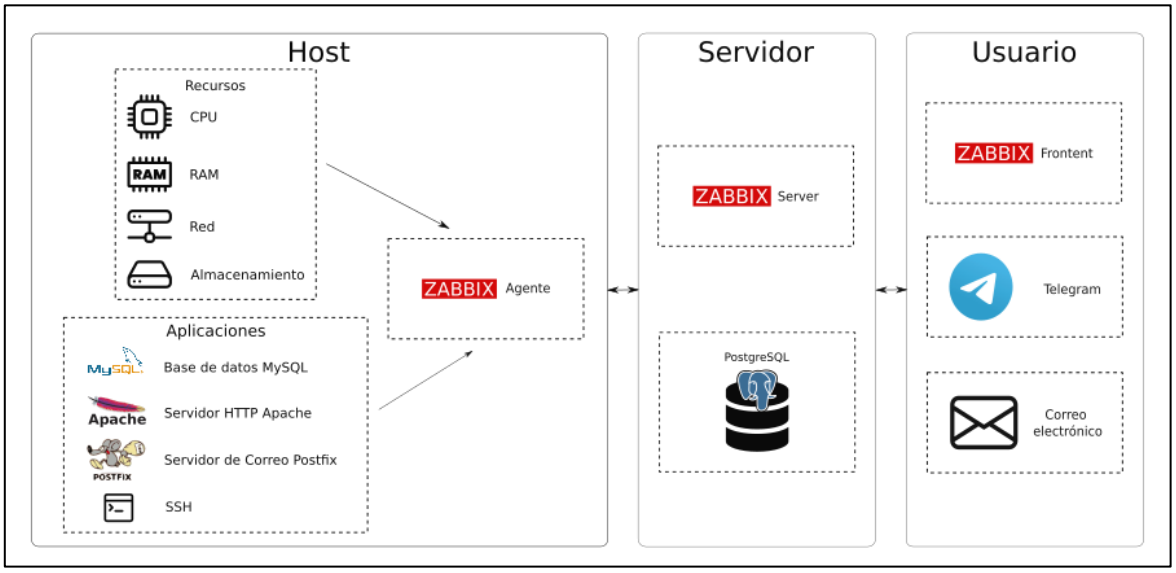


Figura 12. Arquitectura de recolección de datos Zabbix

3.1.2. Prometheus y ELK Stack

3.1.2.1. Prometheus

La arquitectura que se usó en la implementación de Prometheus en el ambiente simulado cuenta con los siguientes componentes:

- **Node exporter:** Exportador de Prometheus para métricas de hardware y SO expuestas por el kernels UNIX. Escrito en GO. Por defecto utiliza el puerto 9100.
- **Apache exporter:** Exporta las estadísticas de apache mod_status vía HTTP. Escrito en GO. Por defecto utiliza el puerto 9117.
- **MySQL Server exporter:** Exportador de Prometheus para las métricas del servidor MySQL. Escrito en Go. Por defecto utiliza el puerto 9104.
- **Postfix export:** Exportador de métricas de Prometheus para el servidor de correo Postfix. Este exportador proporciona métricas de histograma para el tamaño y la edad de los mensajes almacenados en la cola de correo. Por defecto utiliza el puerto 9154.
- **Prometheus:** Es un sistema de monitoreo open source basado en métricas. Por defecto utiliza el puerto 9090.
- **Alertmanager:** Gestiona las alertas enviadas por aplicaciones cliente. Se encarga de duplicarlas, agruparlas y dirigirlas a las integraciones receptoras correctas. Por defecto utiliza el puerto 9093.
- **API Rest:** Es una interfaz de programación de aplicaciones, utilizada para la comunicación con el API de Telegram. Por defecto utiliza el puerto 4000.
- **Correo electrónico:** Es un servicio de red que permite a los usuarios enviar y recibir mensajes mediante redes de comunicación electrónica.
- **Grafana:** Es una aplicación web de visualización interactiva. Proporciona cuadros, gráficos y alertas para la web cuando se conecta a fuentes de datos compatibles. Por defecto utiliza el puerto 3000.

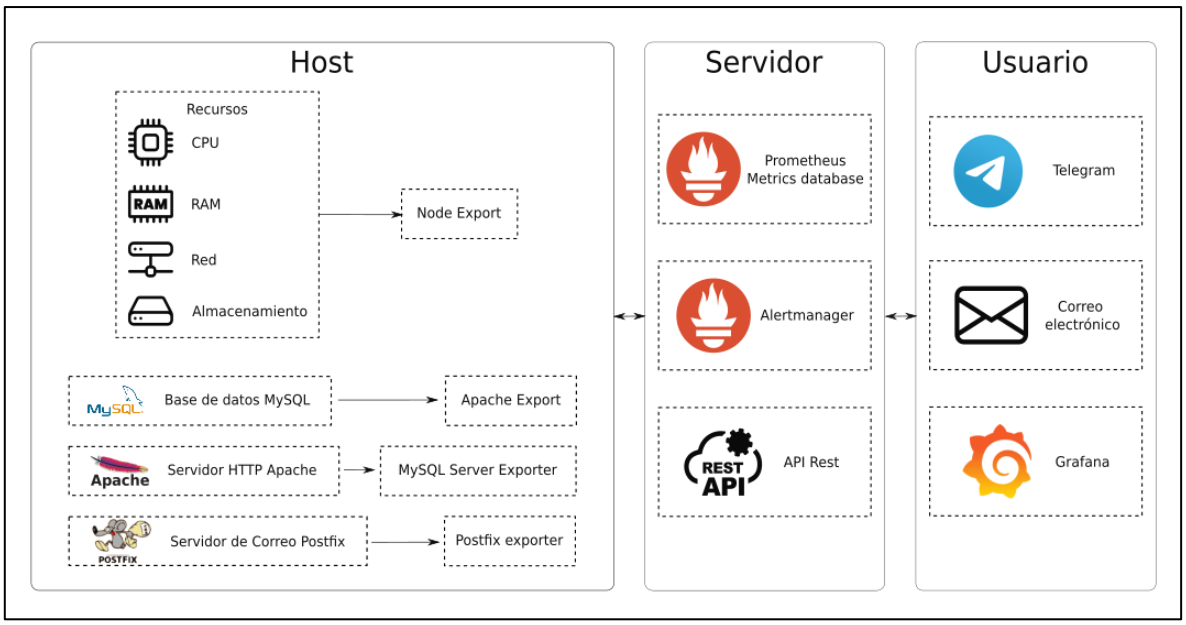


Figura 13. Arquitectura de recolección de datos Prometheus

3.1.2.2. ELK Stack

- **Logstash:** Es un canalizador de datos que ayuda a recopilar, tratar y analizar una gran variedad de datos y eventos estructurados y no estructurados generados en varios sistemas.
- **Elasticsearch:** Es un motor de búsqueda distribuido basado en Apache Lucene. Por defecto utiliza el puerto 9200.
- **Kibana:** Es una plataforma de visualización de datos. Por defecto utiliza el puerto 5601.
- **API Rest:** Es una interfaz de programación de aplicaciones, utilizada para la comunicación con el API de Telegram. Por defecto utiliza el puerto 4000.
- **Correo electrónico:** Es un servicio de red que permite a los usuarios enviar y recibir mensajes mediante redes de comunicación electrónica.
- **Telegram:** Es una aplicación de mensajería centrada en la velocidad y la seguridad, es rápida, sencilla y gratuita.

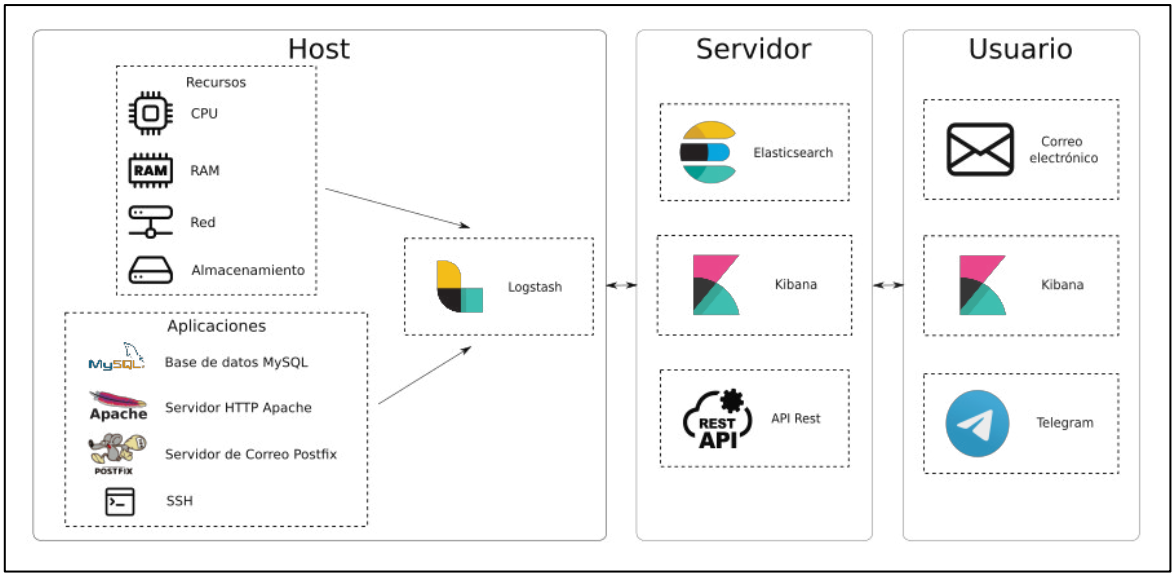


Figura 14. Arquitectura de recolección de datos Stack ELK

3.2. Implementación de Zabbix

En esta sección se va a instalar y configurar un entorno de monitoreo con la herramienta Zabbix. Se lo realizará en un entorno simulado usando dos computadoras personales, el primer host posee un sistema operativo Debian 10 y el segundo Ubuntu 20.

Especificaciones técnicas del Host 1	
Sistema Operativo	Debian 10
RAM	2 GB DDR3 800 Hz
Procesador	AMD Turion(tm) 64 X2 Mobile Technology TL-56
Disco Duro	240 GB
Tipo de Sistema	Sistema Operativo de 64 bits

Tabla 10. Especificaciones técnicas del Host 1

Especificaciones técnicas del Host 2	
Sistema Operativo	Ubuntu 20.10
RAM	16 GB DDR4 1333 Hz
Procesador	Intel Core i5-10400
Disco Duro	932 GB
Tipo de Sistema	Sistema Operativo de 64 bits

Tabla 11. Especificaciones técnicas del Host 2

3.2.1. Instalación del Servidor

Para comenzar el proceso de instalación de la herramienta Zabbix se va a instalar el servidor Zabbix en el host 1. Todos los programas y herramientas necesarios se encuentran en el repositorio oficial https://www.zabbix.com/la/download. Una guía detallada de la instalación y configuración del servidor Zabbix se encuentra en el Anexo 1.

3.2.2. Instalación del Agente en Linux

Para continuar con el proceso se instaló el agente Zabbix en el host 2.

3.2.2.1. Instalación

Se instaló el agente Zabbix para recopilar información del host 2.

```
# apt-get install zabbix-agent
```

Como siguiente paso se inició y habilitó el servicio para que se inicie automáticamente al encender el host 2.

```
# systemctl start zabbix-agent
# systemctl enable zabbix-agent
```

En el archivo de configuración del agente Zabbix se debe agregar la IP del servidor Zabbix, la IP del agente Zabbix y el hostname del servidor Zabbix.

```
# vim /etc/zabbix/zabbix_agentd.conf
Server=192.168.1.15          <----- IP del Servidor Zabbix
ListenIP=192.168.1.10        <----- IP de la máquina del Agente Zabbix
ServerActive=192.168.1.15    <----- IP del Servidor Zabbix
Hostname=Zabbix server       <----- Nombre de la máquina del Agente Zabbix
```

Una vez realizado los cambios en el archivo de configuración se reinició el servicio para que se apliquen los cambios.

```
# systemctl restart zabbix-agent
```

3.2.2.2. Configuración del Host

En Zabbix se le denomina “Host” a cualquier equipo, dispositivo o aplicación que se quiere monitorizar. Para agregar un nuevo host a Zabbix, se tiene que ir al apartado de “Configuración -> Host”.

Figura 15. Módulo "Host" en la barra lateral

En la pantalla de configuraciones se escoge la opción de “Create Host”.

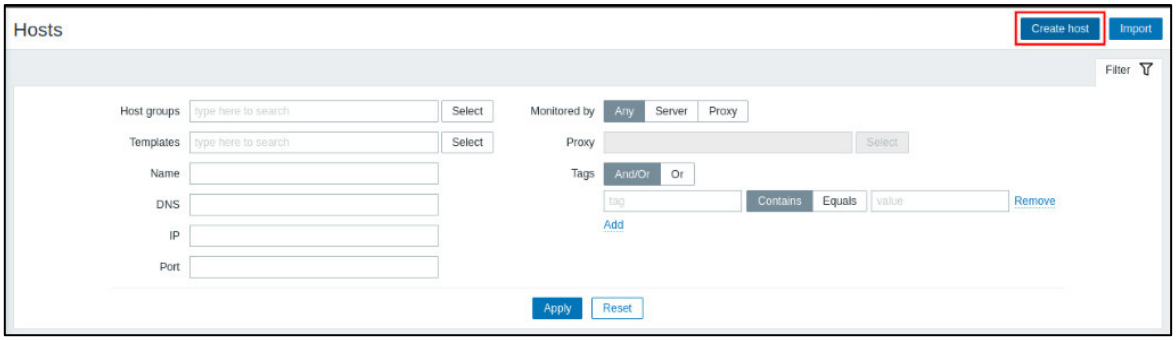


Figura 16. Pantalla de configuración de host

En la pantalla creación de host se tiene que llenar un formulario de configuración, en donde los campos obligatorios son:

- Host Name: Nombre del Host
- Groups: Grupo que tiene aplicado los permisos de acceso
- Interfaces: La dirección IP del Host

Una vez lleno el formulario se debe usar la opción “Add”.

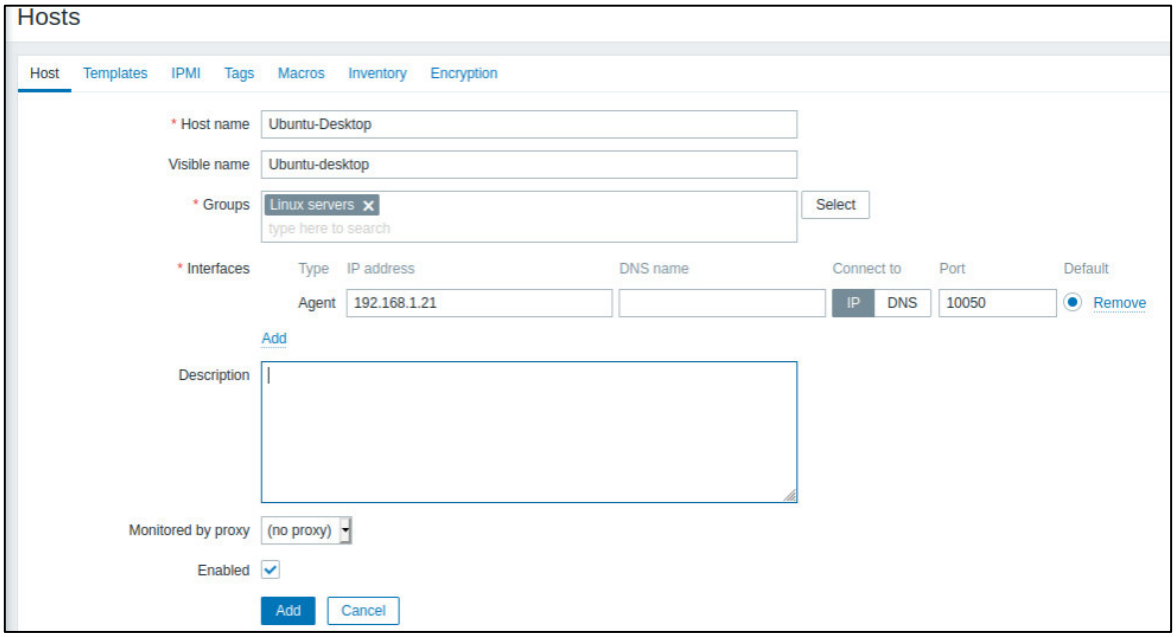


Figura 17. Pantalla de creación de host

Como se puede ver en la imagen el host se ha creado correctamente

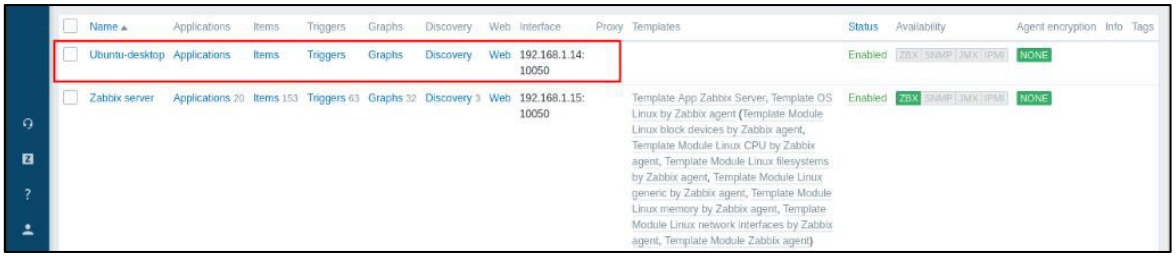


Figura 18. Actualización de la lista de hosts en pantalla de configuración

3.2.2.3. Configuración de Item

Para agregar un Item nuevo hay que ir al apartado “Configuration -> Host”, y se marca el Host al que se quiere agregar el Item.

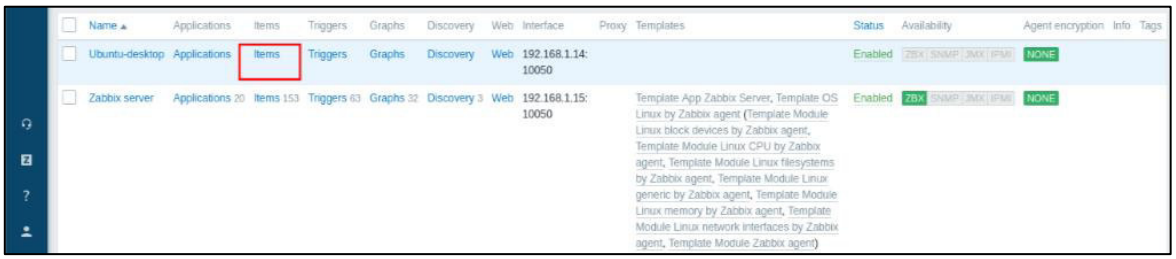


Figura 19. Módulo Items en pantalla de configuración de host

A continuación, se selecciona la opción de “Create item”

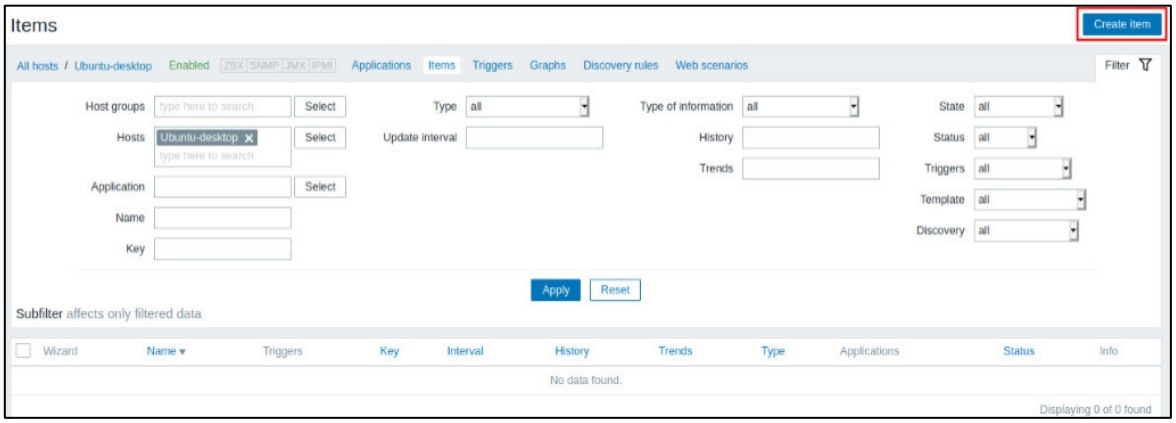


Figura 20. Pantalla de configuración de Item

Y se configura las opciones:

- Name: Nombre que recibirá el Item
- Key: Se indica la información que se va a solicitar al Agente Zabbix

- Type of information: Se indica el formato de los datos que se va a recibir

En este caso como se quiere realizar el monitoreo el rendimiento del CPU, se ingresó los siguientes datos:

- Name: CPU Load
- Key: system.cpu.load[all,avg1]
- Type of information: Numeric (float)

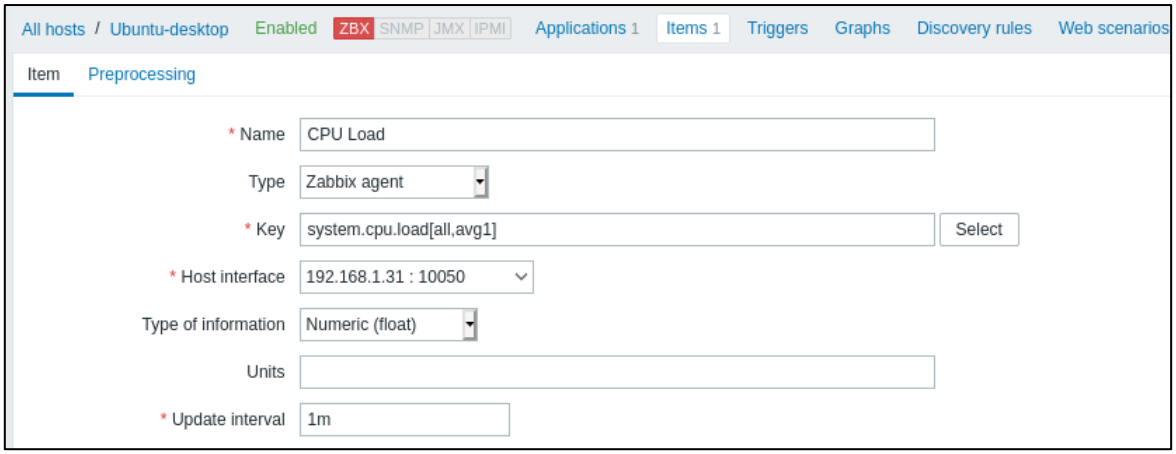


Figura 21. Formulario de creación de Item

Para verificar que los parámetros ingresados son correctos, en la parte de debajo de la pantalla de configuración del Item se tiene una opción de “Test”.

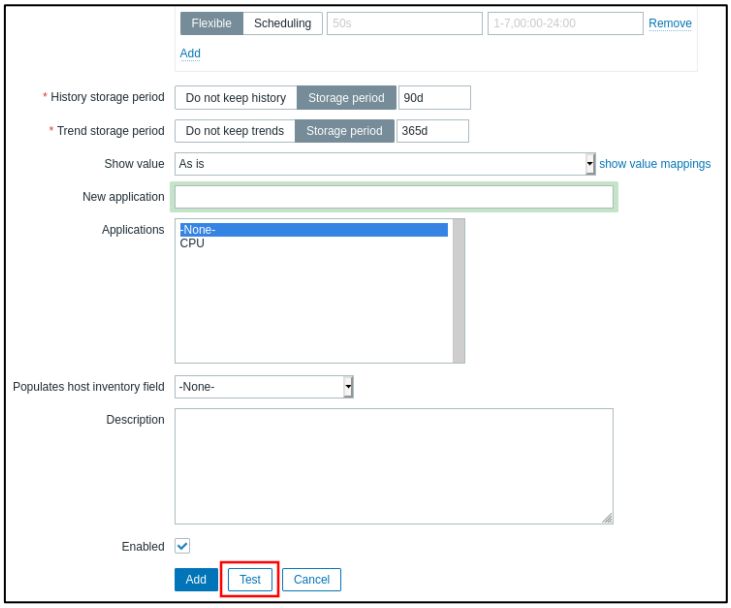


Figura 22. Opción "Test" en formulario de creación de Item

En el Popup de prueba, por medio del botón “Get value” se puede obtener información de la métrica configurada anteriormente, en este caso de la carga del CPU.

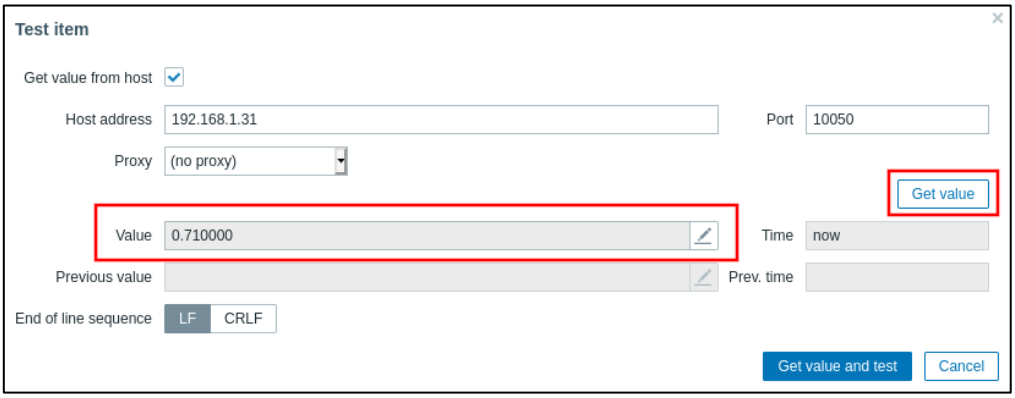


Figura 23. Resultado de la prueba de un Item

Una vez comprobado que la métrica funciona correctamente se utiliza la opción “Add” para agregar el Item.

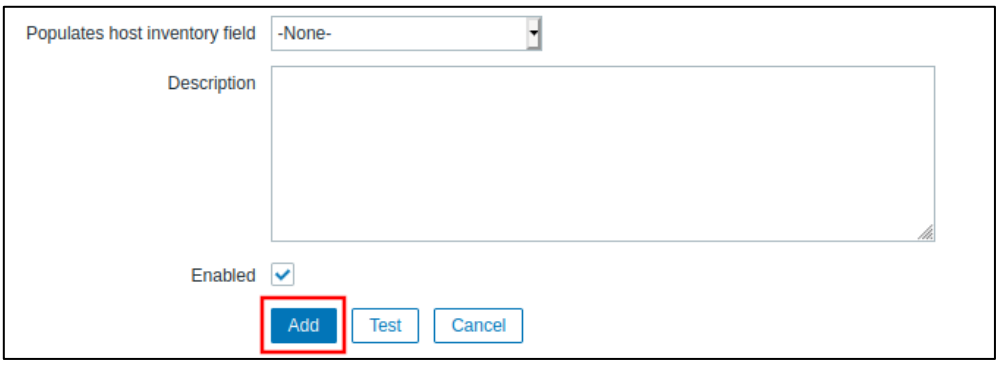


Figura 24. Opción añadir en formulario de creación de Item

Para poder ver los datos recopilados por el Item hay que ir al apartado “Monitoring -> Latest data”.

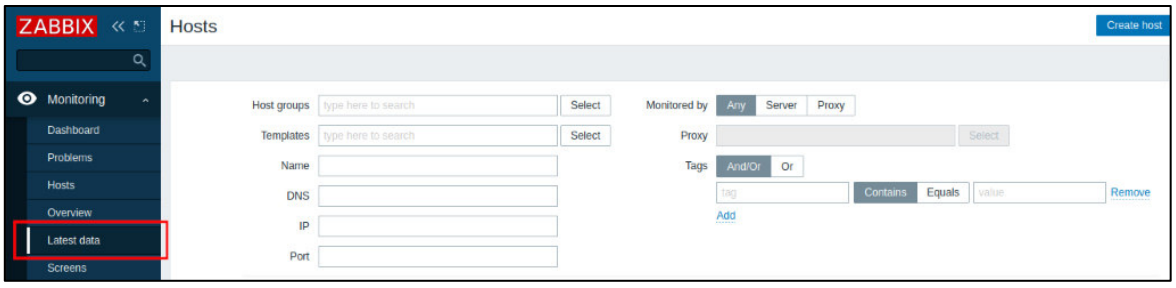


Figura 25. módulo "Lastest data" en barra lateral

Para visualizar la información gráficamente, se tiene que dar clic en la etiqueta “Graph” del Item al cual se desea observar la información.

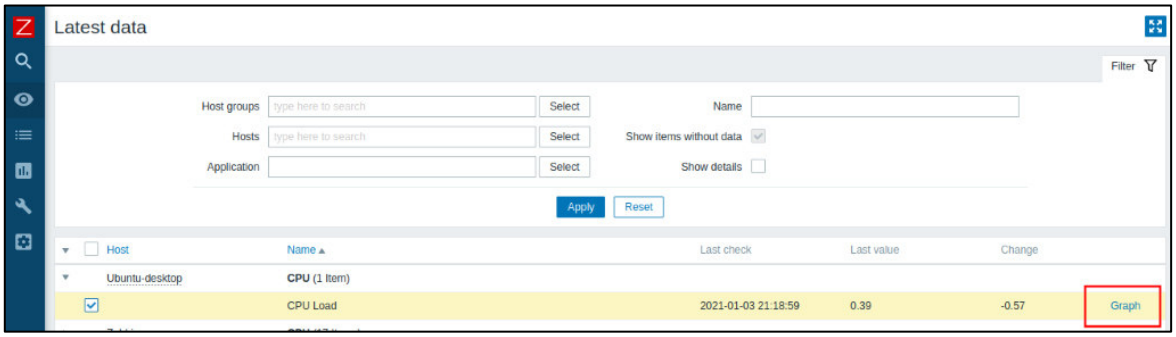


Figura 26. Pantalla de módulo "Lastest data"

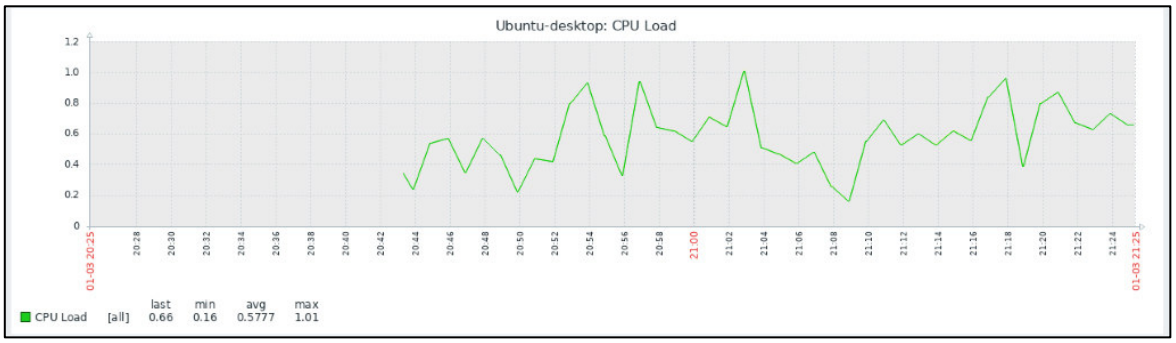


Figura 27. Gráfico de líneas de carga del CPU

3.2.2.4. Configuración de Trigger

Utilizamos los Triggers para comparar los valores recolectados por los Items con condiciones.

Para crear un Item nos dirigimos al apartado de “Configuration -> Host”, se selecciona a un “Host” y luego hacemos clic en “Trigger”.

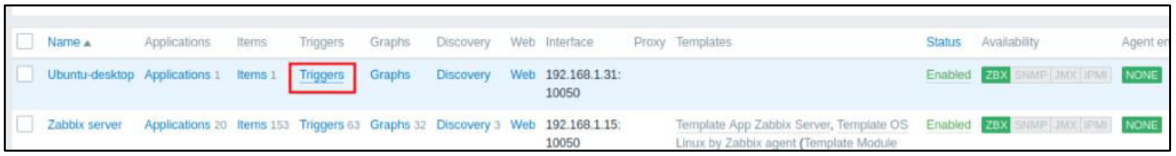


Figura 28. Módulo "Triggers" en pantall de configuración de host

Una vez en la pantalla de triggers se marca la opción de “Create trigger”.

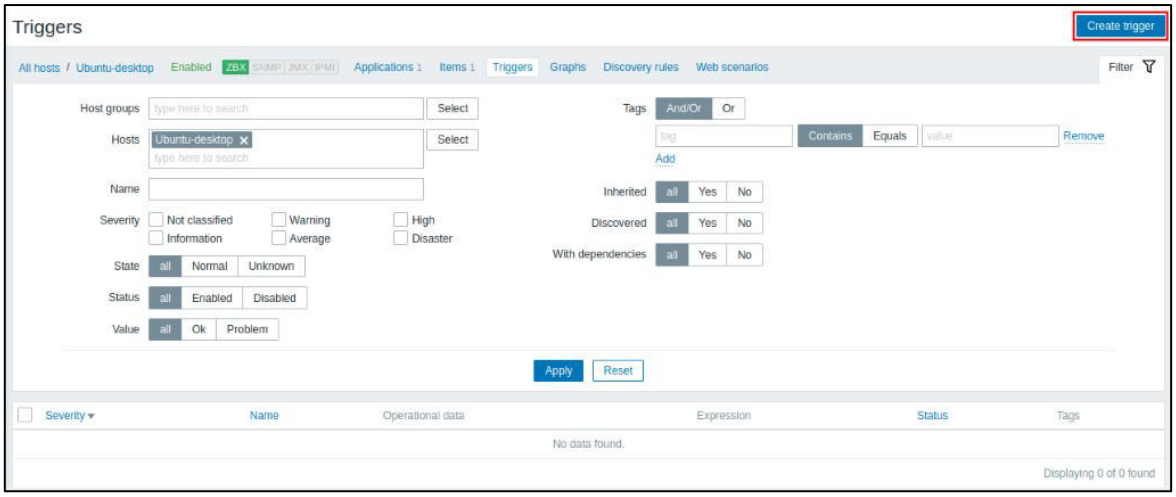


Figura 29. Módulo “creación trigger” en pantalla de configuración de Trigger

La información que se tiene que ingresar obligatoriamente es:

- Name: Nombre que se mostrará cuando se ejecute el Trigger
- Expression: Se indica la condición del Trigger

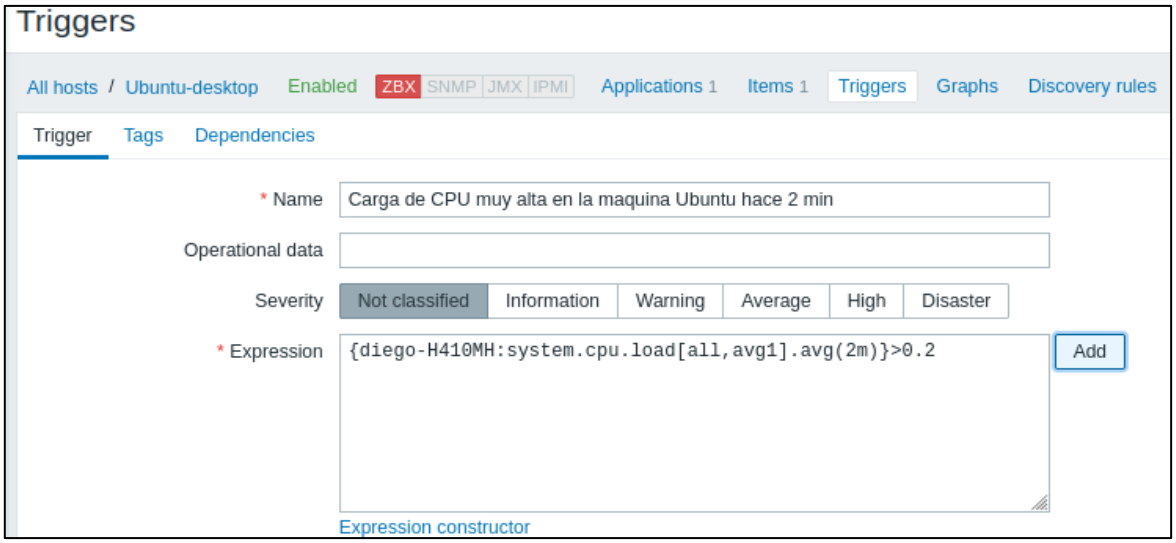


Figura 30. Formulario de creación de Triggers

Para añadir la expresión se utiliza el botón “Add”, el cual abrirá un Popup en donde se debe escoger el Item, una función y un valor con el que se realizará la condición.

Figura 31. Formulario de condición en creación de Trigger

Una vez que se ingrese el Item y la expresión, se da en el botón “Add” que se encuentra al final del formulario para guardar el Trigger.

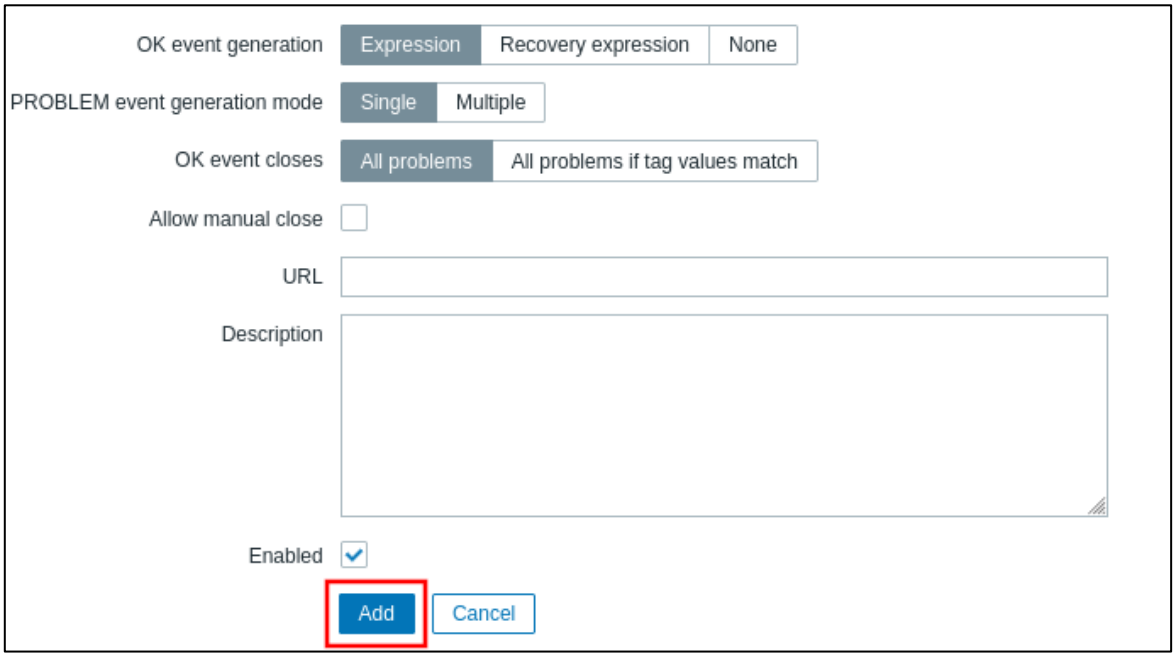


Figura 32. Opción añadir en formulario de creación de host

La información del estado del trigger se encuentra en la parte inferior de la pantalla de configuración. Si se muestra en color verde significa que todo se encuentra correctamente, en caso de que se encuentre en color rojo se ha sobrepasado el valor indicado en las configuraciones del trigger.

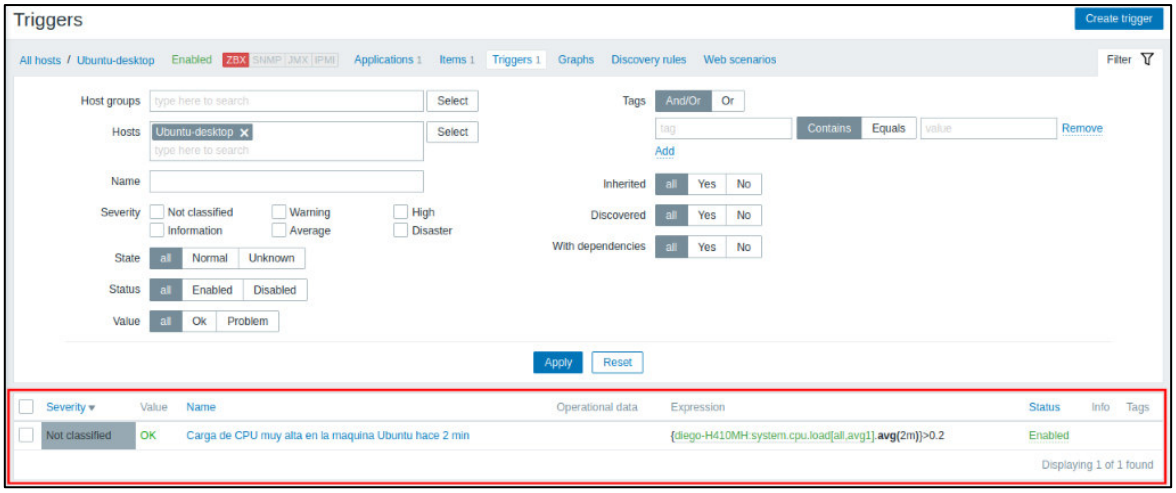


Figura 33. Estado de trigger

3.2.3. Instalación del Agente en Windows

Se instaló el agente Zabbix en un host con sistema operativo Windows 10. El programa se lo puede descargar desde la página web oficial del producto en el siguiente URL https://www.zabbix.com/download_agents?version=5.0+LTS&release=5.0.9&os=Windows&os_version=Any&hardware=i386&encryption=No+encryption&packaging=Archive.

Como siguiente paso se descomprime el archivo descargado en la raíz de los directorios del disco C.

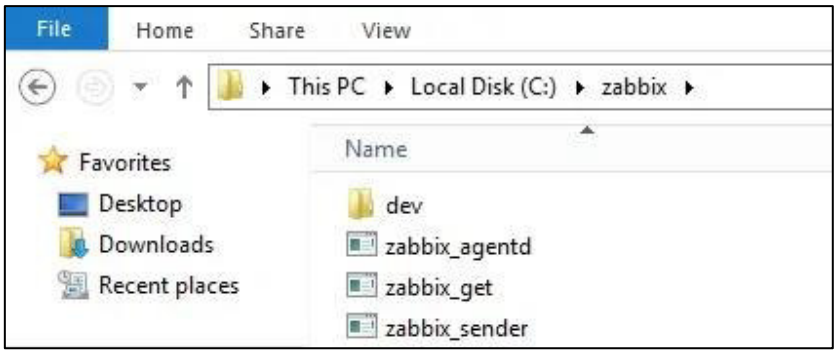


Figura 34. Contenido del archivo de Zabbix

A continuación, se creó un archivo de configuración con el nombre "zabbix_agentd.win.conf", con el siguiente contenido.

```
Server=127.0.0.1
ServerActive=127.0.0.1
Logfile=C:\zabbix\zabbix_agent.log
```

En el documento se debe incluir la IP del servidor Zabbix, tanto en Server y ServerActive. Y se grabó el contenido.

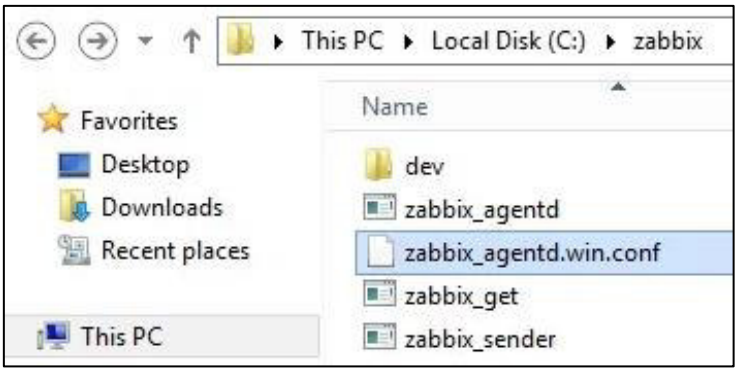


Figura 35. Archivo de configuración de Zabbix

Después de terminar de crear el archivo de configuración, se debe crear un servicio, este procedimiento se puede realizar desde el cmd.

```
c:\zabbix\zabbix_agentd.exe --config c:\zabbix\zabbix_agentd.win.conf --install
```

Si todo sale bien se mostrará el siguiente mensaje.

```
zabbix_agentd.exe [4508]: service [Zabbix Agent] installed successfully
zabbix_agentd.exe [4508]: event source [Zabbix Agent] installed successfully
```

El agente Zabbix se instaló en la computadora, pero aún se necesita que se active el servicio. Para esto nos dirigimos a la interfaz de servicios de Windows, buscar el servicio e iniciarlo.

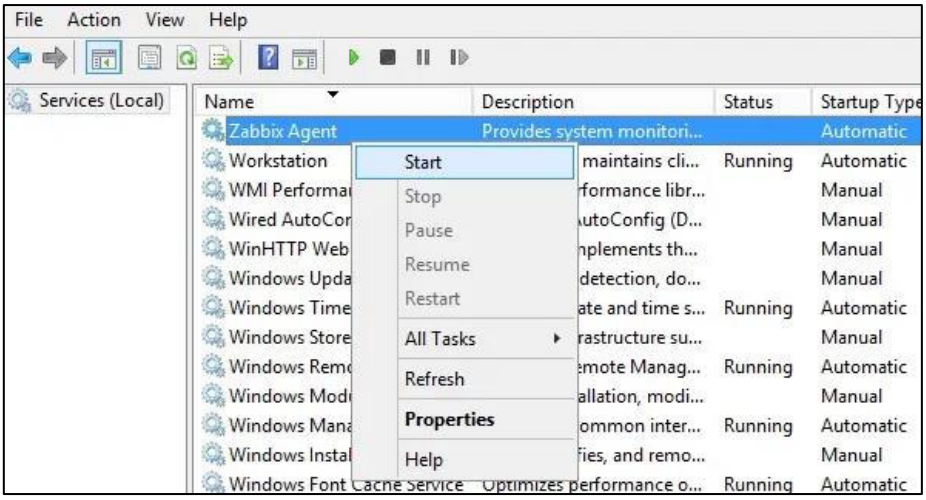


Figura 36. Ejecución del servicio de Zabbix

3.2.4. Creación de un Item Personalizado

Se va a crear un Item en Zabbix que permita monitorear la temperatura de un procesador, para esto se utilizará “Sensors” la cual es una pequeña herramienta que ayuda a saber la temperatura del CPU de un ordenador.

Por medio del lenguaje awk es posible obtener un promedio de la temperatura de todos los núcleos de un host con sistema operativo basado en Linux. El comando se compone de cuatro partes.

El primero es el “sensors” que nos sirve para obtener la información de temperatura de cada núcleo del procesador.

```
# sensors
acpitz-acpi-0
Adapter: ACPI interface
temp1:        +26.0°C  (crit = +127.0°C)
temp2:        +27.8°C  (crit = +119.0°C)
coretemp-isa-0000
Adapter: ISA adapter
Package id 0:  +27.0°C  (high = +84.0°C, crit = +100.0°C)
Core 0:        +25.0°C  (high = +84.0°C, crit = +100.0°C)
Core 1:        +25.0°C  (high = +84.0°C, crit = +100.0°C)
```

```
Core 2:        +24.0°C  (high = +84.0°C, crit = +100.0°C)
Core 3:        +24.0°C  (high = +84.0°C, crit = +100.0°C)
Core 4:        +26.0°C  (high = +84.0°C, crit = +100.0°C)
Core 5:        +25.0°C  (high = +84.0°C, crit = +100.0°C)
```

El segundo comando obtiene la salida del comando anterior y extrae las últimas filas de la impresión. En este caso como el host 2 posee 6 núcleos el parámetro correspondiente para este comando es “tail -n 7”.

```
# sensors | tail -n 7 | head -n 6 | awk -F'[:+°]' '{avg+=$3}END{print avg/NR}'
24.8333
```

Para que el agente Zabbix permita la ejecución de este comando hay que realizar dos cambios en el archivo /etc/zabbix/zabbix_agentd.conf.

```
# vim /etc/zabbix/zabbix_agentd.conf
UnsafeUserParameters=1
UserParameter=cluster1.cpuTemperature, sensors | tail -n 7 | head -n 6 | awk -F'[:+°]' '{avg+=$3}END{print avg/NR}'
```

Para que se apliquen los cambios de configuración hay que reiniciar el servicio zabbix-server.

```
# systemctl restart zabbix-agent.service
```

Desde la interfaz gráfica del servidor Zabbix se crea un nuevo Item en donde se incluye el parámetro creado anteriormente.

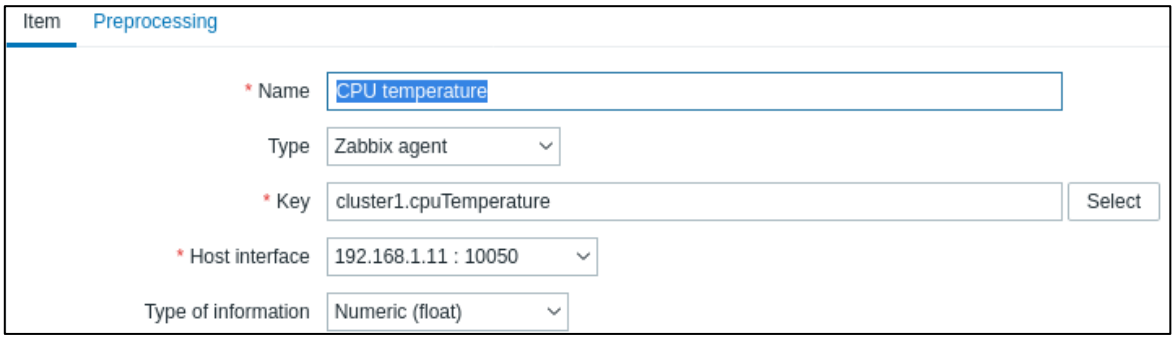


Figura 37. Formulario de creación de un Item personalizado

Para este Item es necesario el preprocesamiento de la información que se va a obtener del comando, en la pestaña de preprocesamiento se crea una función de JavaScript en donde se realiza un parse de un string a un float.

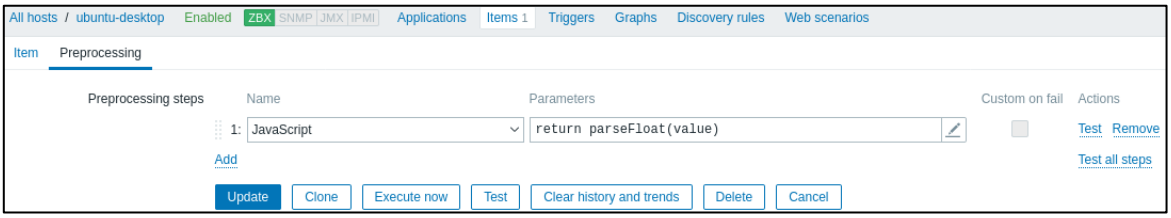


Figura 38. Preprocesamiento de valor de Item

Una vez grabado el Item ya se estará monitoreando la temperatura del procesador de host, esto se puede apreciar en los gráficos de Zabbix.

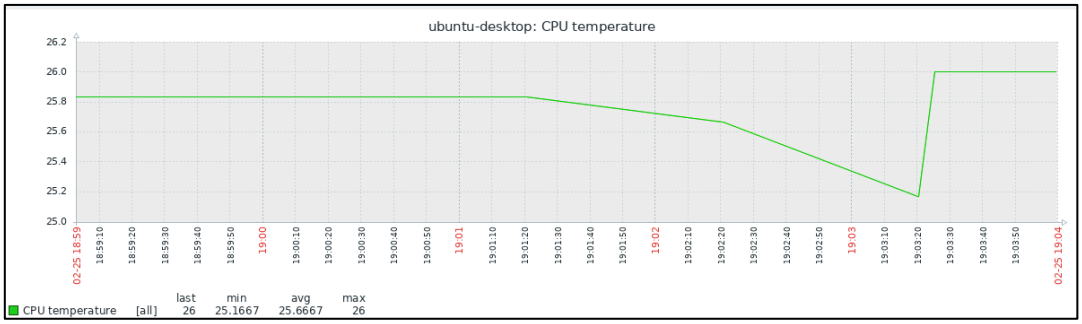


Figura 39. Gráfico de líneas de la temperatura del procesador

3.2.5. Configuración de Alerta en Telegram

El siguiente paso es mostrar cómo Zabbix es capaz de notificar a través de un mensaje por Telegram sobre eventos importantes.

Como primer paso hay que registrar un nuevo bot en Telegram, enviando el mensaje "/newbot" a @BotFather, escoger el nombre del bot. En este caso "zabbix_test_ec_bot". El token proporcionado por @BotFather será necesario en la configuración del servidor Zabbix.

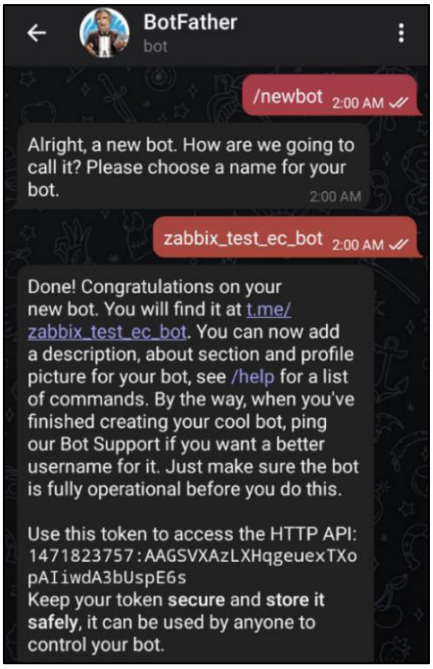


Figura 40. Registrar bot en Telegram

Para que el bot de Telegram pueda enviar mensajes al Usuario, este debe enviarle un mensaje “/start” al bot.

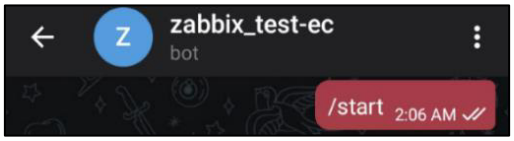


Figura 41. Inicio de bot

Para continuar con la configuración del envío de notificaciones hay que ir al apartado “Administration -> Media Types”. En la pantalla de “Media Type”, se selecciona la opción que se va a utilizar en este caso Telegram.

Figura 42. Pantalla de configuración de Media type

En la configuración del medio se incluye el token del bot creado anteriormente.

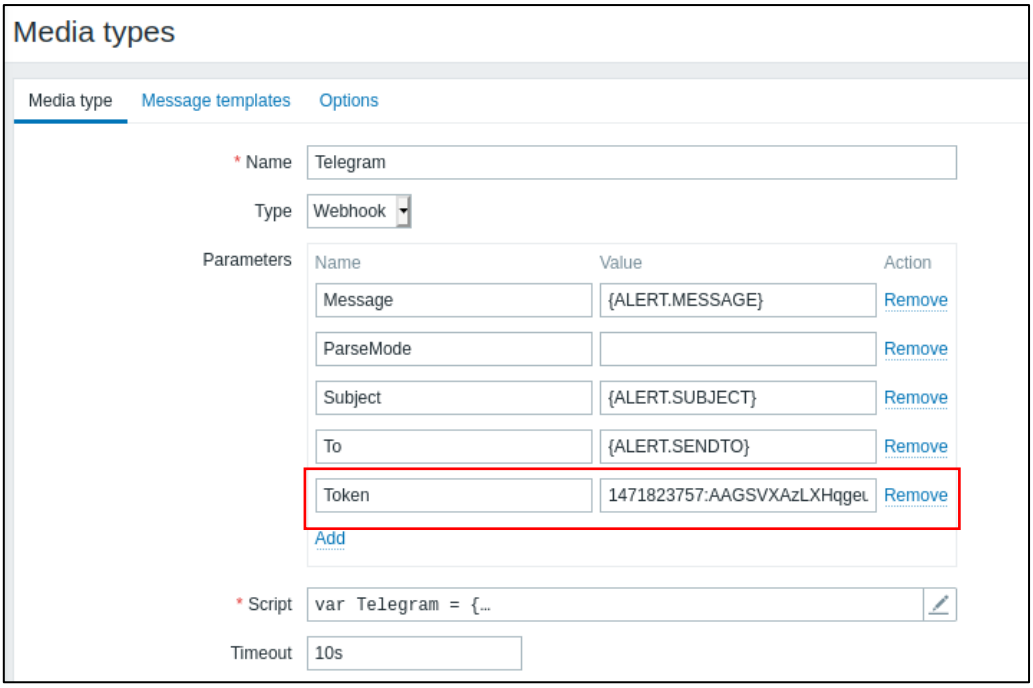


Figura 43. Pantalla de configuración de Media types de Telegram

Para continuar, hay que agregar un medio a los usuarios. Esto se puede hacer en el apartado “Administration -> Users”. Se selecciona el Usuario que se al cual se va a agregar el medio en este caso el usuario “Admón.”.

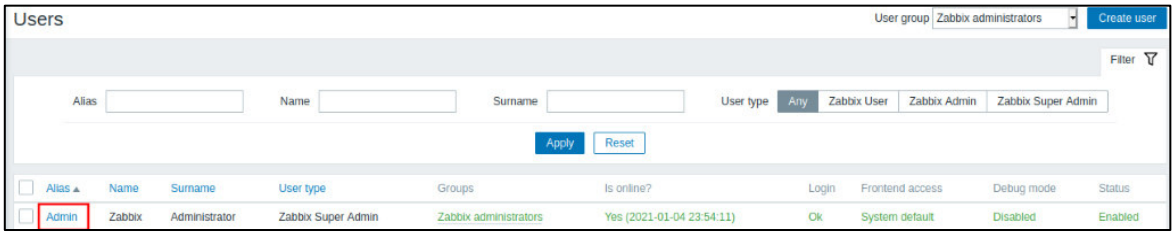


Figura 44. Lista de usuarios registrados

En la opción de “Media” utilizamos la etiqueta “Add” para ingresar la configuración del medio.

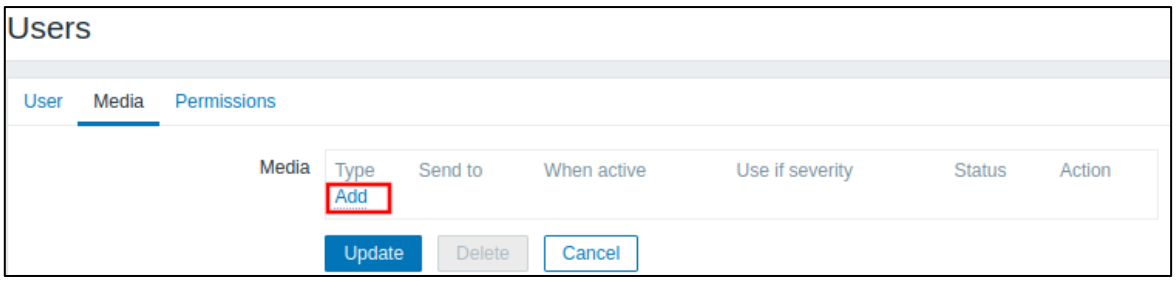


Figura 45. Opción de añadir usuario

Existen varias formas de realizar la configuración de “Telegram”, en este caso el bot va a enviar un mensaje a un usuario en específico para esto se necesita conocer el id del usuario. Esto se puede realizar escribiendo “/getid” en el chat de @IDBot.

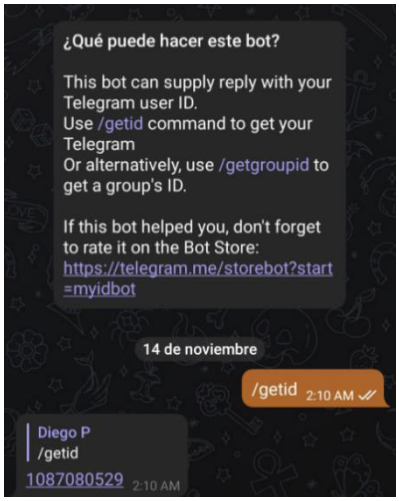


Figura 46. Utilización de bot IDBot

En la pantalla de configuración se especifica el tipo del medio “Telegram” y el id del cliente que va a recibir los mensajes “1087080529”.

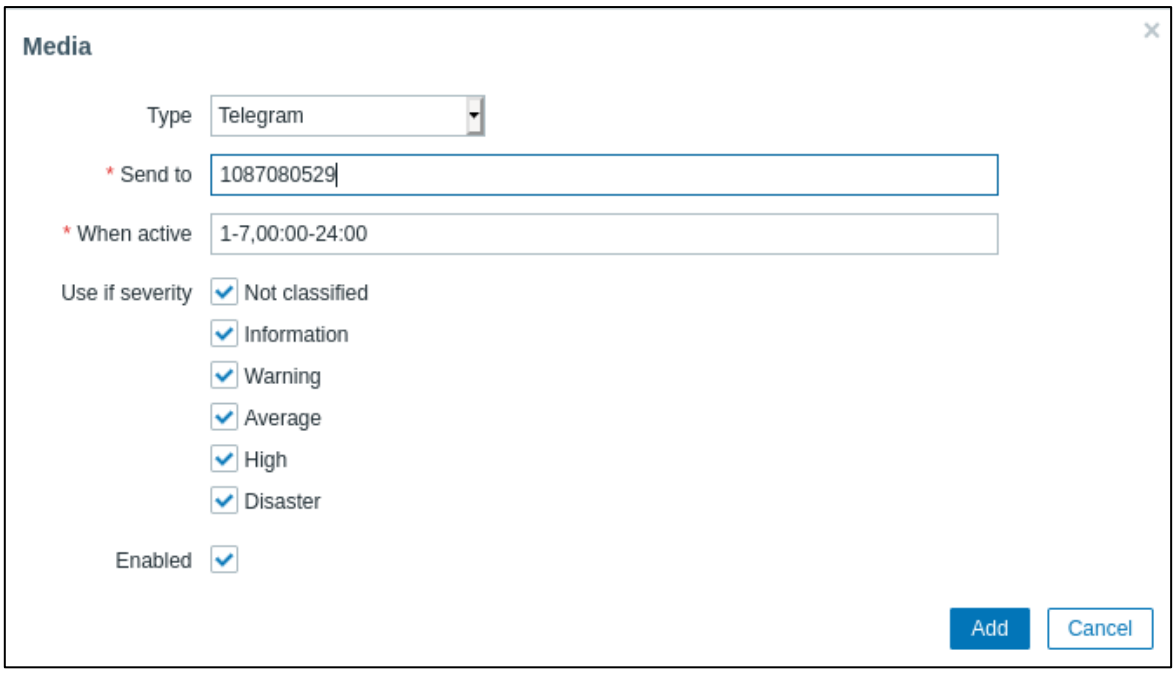


Figura 47. Pantalla de configuración de un medio de comunicación de un usuario

Una vez ingresado el medio se podrá observar el registro en la pantalla de Usuario.

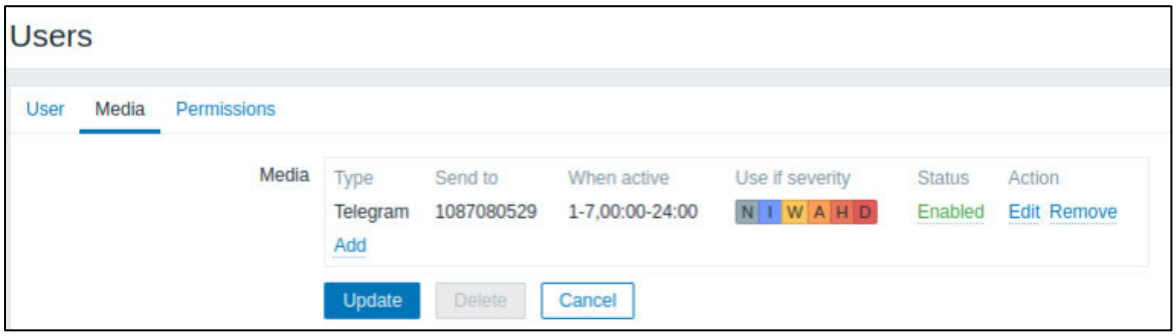


Figura 48. Lista de medios de comunicación de un usuario

A continuación, en el módulo “Configuration -> Actions -> Trigger action”. Se activa el trigger action “Report problems to Zabbix administrators”.

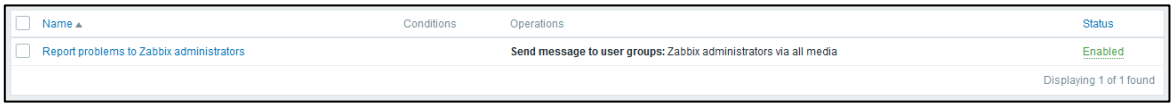


Figura 49. Activación de acción reportar problemas a administradores Zabbix

3.2.6. Prueba de una Alerta

Para comprobar el funcionamiento de una Alerta y el Trigger asociada a ella se va a desconectar de la red a un host que se está monitoreando.

Una vez que el host 2 ha permanecido suficiente tiempo desconectado de la red, en el apartado de problemas de Zabbix se podrá observar una alerta, que describe la ausencia de conexión del host monitoreado, el mensaje que se puede observar en el panel de control es que el agente Zabbix no se encuentra disponible.

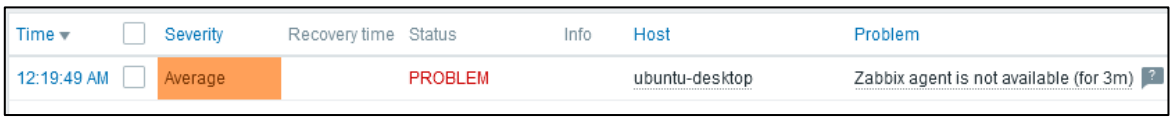


Figura 50. Activación de una alerta

Ya que también se encuentra configurado las notificaciones por Telegram, el problema se notificará a todos los medios implementados.

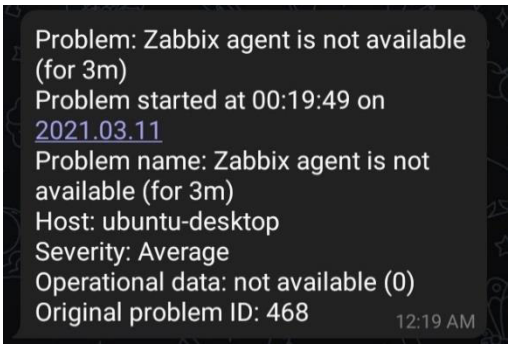


Figura 51. Visualización de alerta en la aplicación móvil de Telegram

3.2.7. Monitorización de Archivos Logs

Para dar un ejemplo de monitoreo de logs se va a usar el servicio SSH. Se va a crear un Trigger en donde se va a monitorear los registros de ingreso al servidor por SSH, en el caso de que las credenciales que se ingresaron no son correctas se va a notificar al administrador.

Para comenzar tenemos que saber dónde se encuentran los logs del servicio SSH. En el caso de Debian 10, esta información se encuentra en “/var/log/auth.log”. Se creó un nuevo Item con la siguiente configuración, en el host que se desea implementar esta funcionalidad.

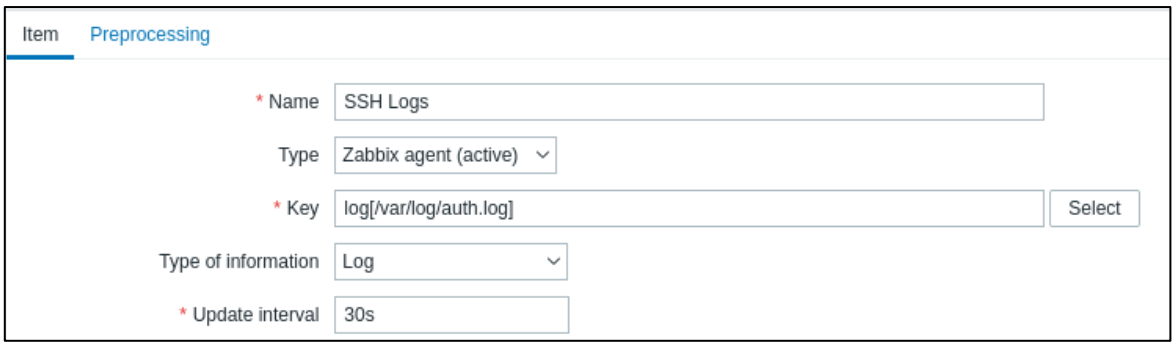


Figura 52. Configuración de Item de los logs del servicio SSH

Una vez creado el Item, se crea el Trigger con la validación que se desea implementar, en este caso se desea monitorear los intentos de acceso fallidos al servidor, para esto se utilizó una expresión regex para obtener esta información.

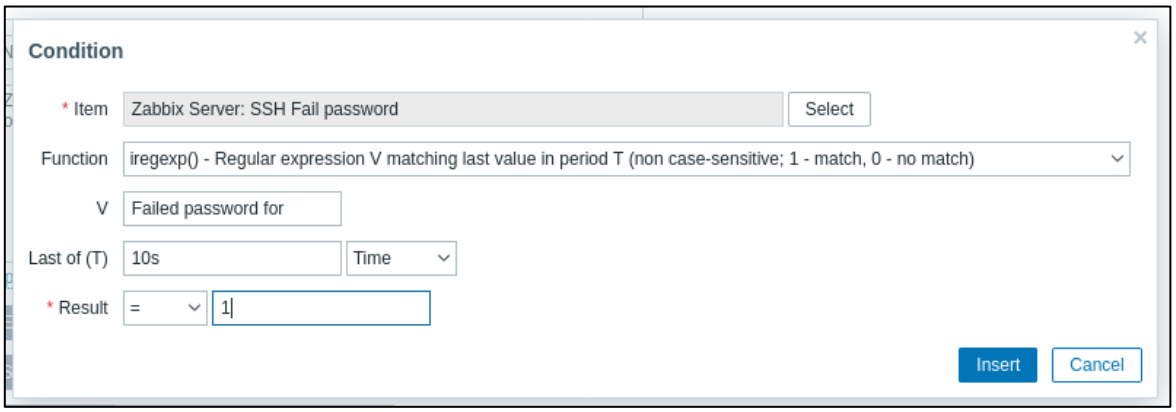


Figura 53. Formulario de configuración de la condición de un Trigger

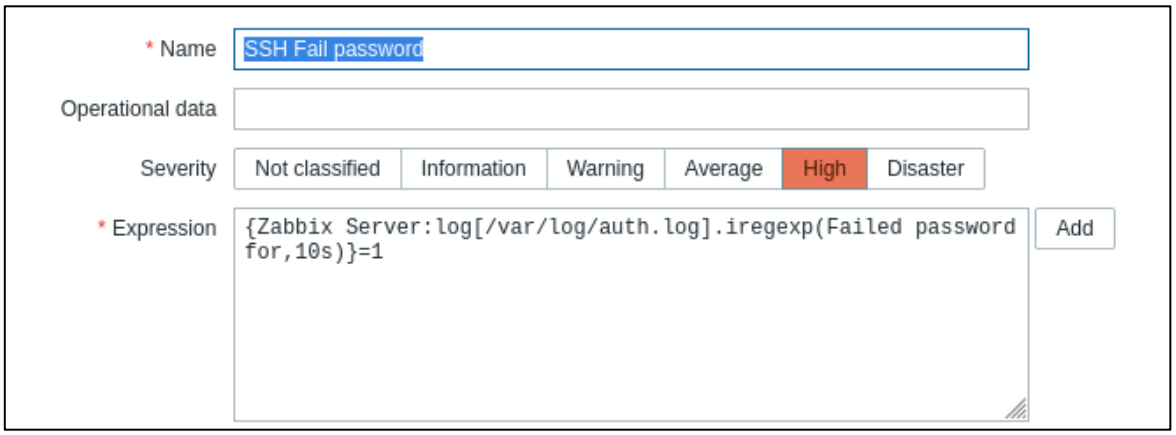


Figura 54. Formulario de configuración de un Trigger

Para validar el funcionamiento del Trigger se ingresó incorrectamente las credenciales de un usuario conectado por SSH. Como se puede ver en la siguiente imagen, el evento se registró en el servidor y se envió una notificación al administrador.

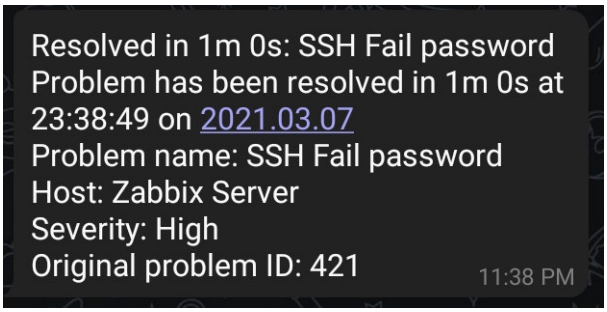


Figura 55. Visualización de alerta en la aplicación móvil de Telegram

3.2.8. Configuración de Template

En el caso de que existan muchos hosts los cuales se deban monitorear y estos posean parámetros similares los Templates son de gran ayuda. Los Templates evitan la tarea tediosa de copiar manualmente elementos de monitoreo. Además de ser útiles cuando se requiera realizar un cambio a todos los hosts monitoreados.

Por defecto Zabbix ya posee configurado ciertos Templates, en este caso se utilizará el Templates “Template OS Linux by Zabbix agent” el cual sirve para monitorear varias entidades de un host que posea una alguna versión de Linux.

En el apartado de configuraciones, se selecciona el host al que se quiere añadir un Templates, en este caso el host 2 ubuntu-desktop y en las opciones de la configuración del host se selecciona Templates.

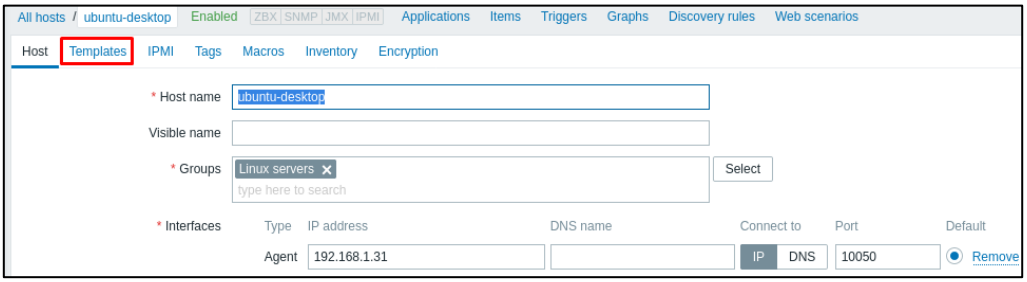


Figura 56. Módulo Templates en pantalla de configuración de un host

En la opción de templetes se tiene una interfaz en donde se puede incluir todos los Templates que sean necesarios.

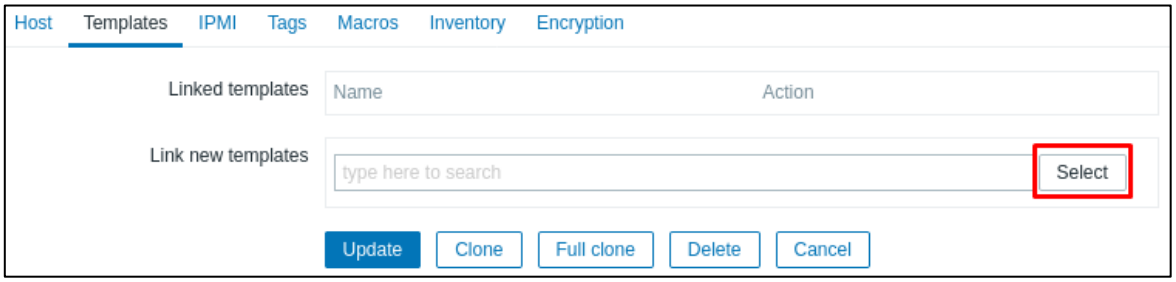


Figura 57. Opción de "Select" en pantalla de Templates

Ya que el monitoreo se realizará en un host que posee un sistema operativo Ubuntu, se seleccionó el Template "Template OS Linux by Zabbix Agent".

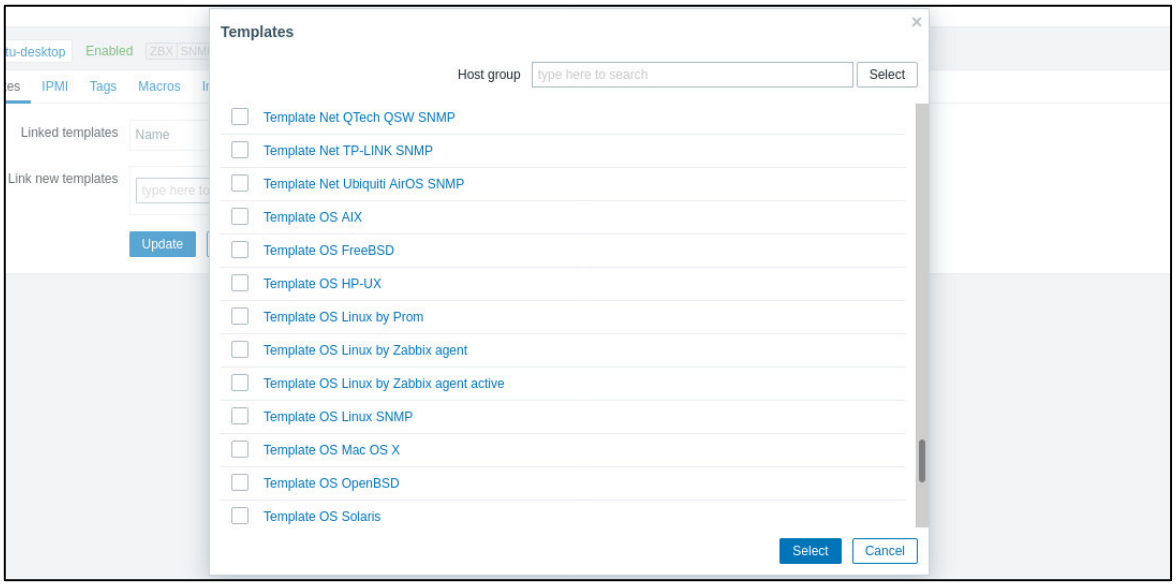


Figura 58. Lista de Template de Zabbix

Una vez seleccionados todos los Template requeridos con el botón "Update" se guarda el estado actual del host.

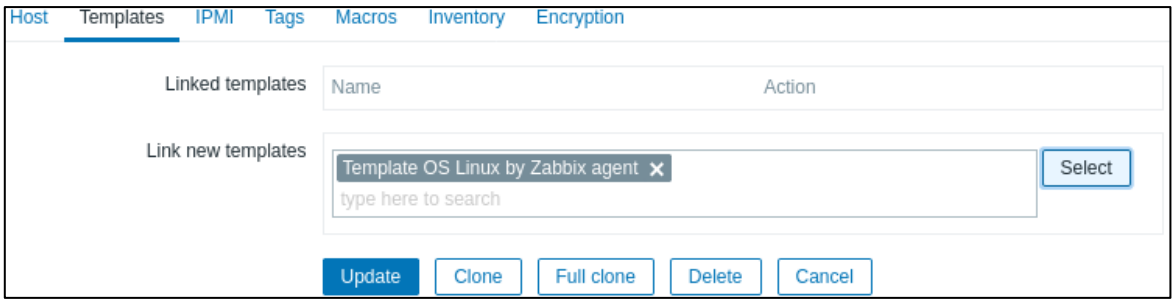


Figura 59. Formulario de Templates después de seleccionar uno

Como se puede ver en la siguiente imagen, la cantidad de items, aplicaciones, triggers, gráficos que se están monitoreando han aumentado, todos estos componentes se han configurado automáticamente con el Template seleccionado.

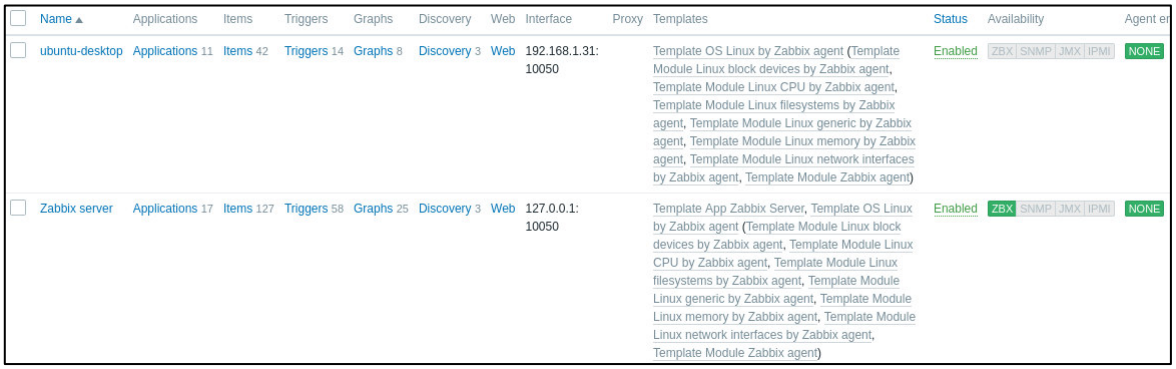


Figura 60. Lista de host una vez añadido un Template

3.3. Implementación de Prometheus

3.3.1. Instalación de Prometheus

La versión precompilada de Prometheus y otros componentes están disponibles en el sitio web de Prometheus https://prometheus.io/download/. En esta ocasión se realizará el monitoreo de un servidor con sistema operativo Linux con arquitectura amd64.

Una vez descargado y descomprimido el archivo de Prometheus ya estará listo para ser ejecutado. Por defecto Prometheus utiliza el puerto TCP 9090. La forma de correr el programa es ejecutando el binario Prometheus.

```
$ ./prometheus
```

Prometheus comenzará a registrar una gran cantidad de información con solo iniciarlo, incluyendo su versión exacta y detalles de la máquina en la que se está ejecutando. Ahora se puede acceder a la interfaz gráfica de Prometheus utilizando el navegador en la dirección http://localhost:9090.

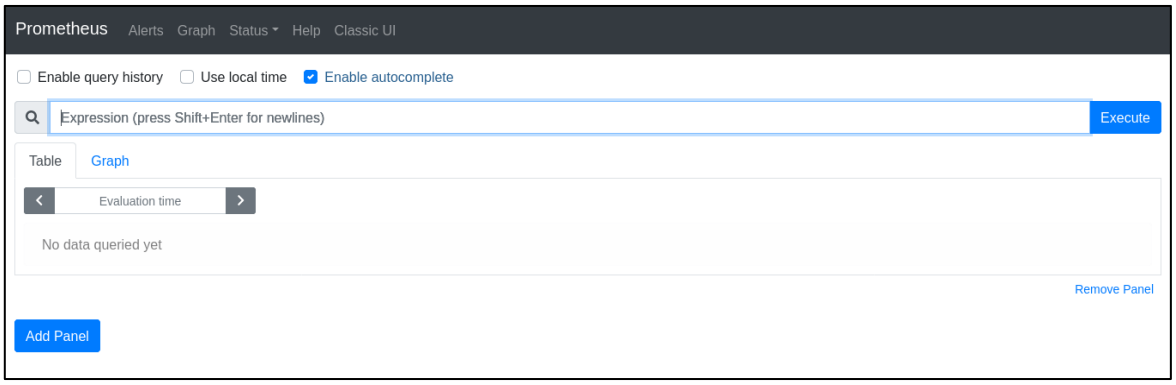


Figura 61. Pantalla de inicio del frontend de Prometheus

Desde esta pantalla se pueden ejecutar consultas PromQL, para obtener una gran variedad de métricas. En la siguiente imagen se utilizará la expresión “up” que sirve para obtener el estado actual de todos los targets de Prometheus.

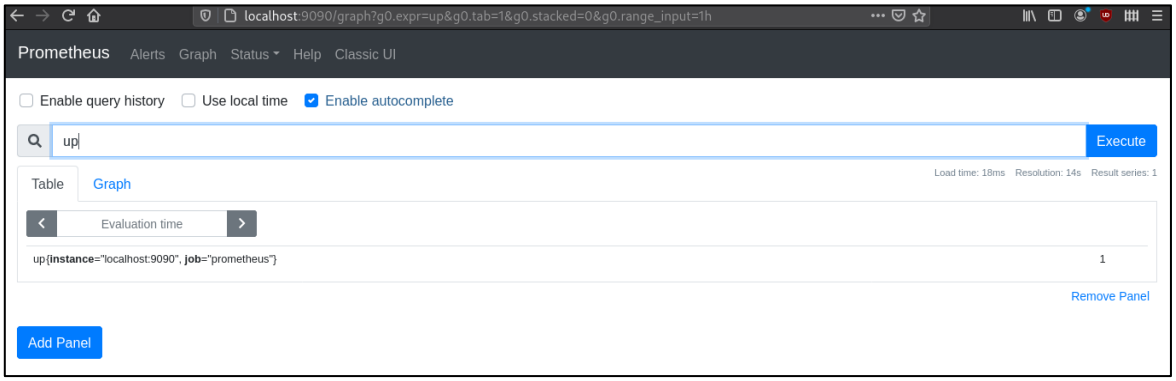


Figura 62. Ejecución de comando de consulta en Prometheus

3.3.2. Instalación de Node Export (Exportador de Nodos)

El exportador de nodo expone métricas a nivel de máquina y de kernel en sistemas Unix, como Linux. Proporciona todas las métricas estándar, como CPU, memoria, espacio en disco, I/O de disco y ancho de banda de red. Además, proporciona una gran cantidad de métricas adicionales expuestas por el kernel, desde el promedio de carga hasta la temperatura de la placa base.

Lo que el Node Export no expone son métricas sobre procesos individuales, ni métricas proxy de otros exportadores o aplicaciones. En la arquitectura de Prometheus, se debe supervisar las aplicaciones y los servicios directamente, en lugar de entrelazarlos con las métricas de la máquina.

Igualmente se puede descargar una versión precompilada del Node Export desde https://prometheus.io/download/, en este caso se va a utilizar para un host con sistema operativo Linux con arquitectura de amd64.

Una vez descargado y descomprimido el archivo. Node Export puede ser ejecutado directamente.

```
$ ./node_export
```

Se puede acceder a la información de node_export a través del puerto TCP 9100.

Para que Prometheus monitoree el node_export es necesario realizar cambios en el archivo de configuración prometheus.yml, añadiendo la siguiente configuración:

```
$ vim prometheus.yml
global:
  scrape_interval:     15s
  evaluation_interval: 15s
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090']
  - job_name: node
    static_configs:
    - targets: ['localhost:9100']
```

Una vez realizado el cambio en el archivo de configuración es necesario reiniciar el Prometheus para que se aplique el cambio. Desde la interfaz gráfica de Prometheus, en la opción de targets se puede observar los nodos que se están monitoreando.

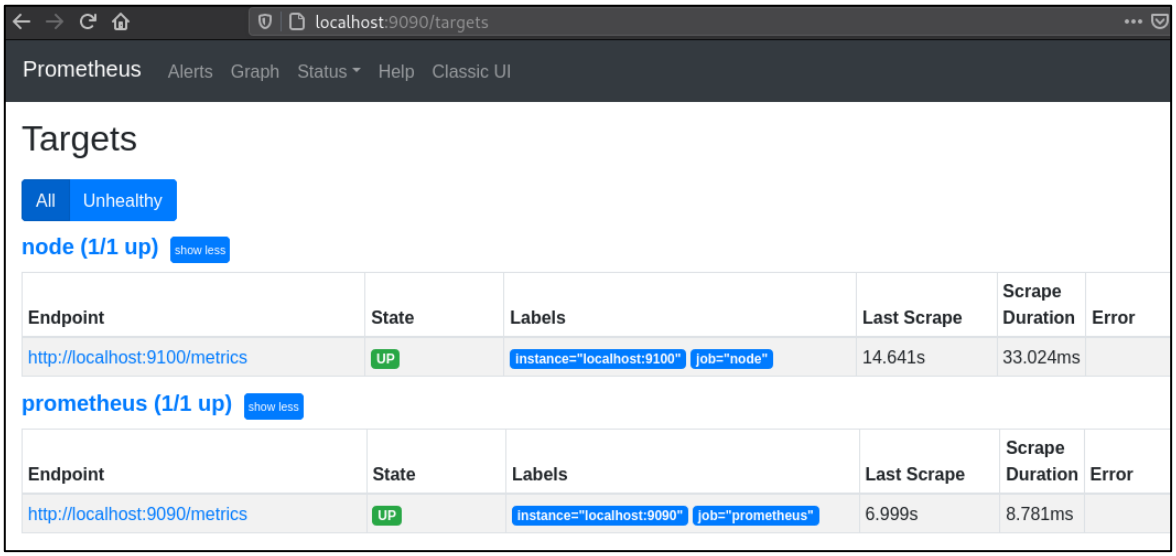


Figura 63. Pantalla de Targets de Prometheus

3.3.3. Alertas

Existen dos partes para la creación de una alerta. El primer paso es agregar la regla de alerta a Prometheus, definiendo la lógica de lo que constituye la alerta. Y, en segundo lugar, configurar el Alertmanager (administrador de alertas en español) que convierte las alertas de activación en notificaciones.

Para la creación de las alertas, se necesita implementar una expresión PromQL que devuelva solo los resultados sobre los que se desea alertar. En este caso solo sería necesario utilizar el operador "==". Con este operador se filtra cualquier serie temporal cuyos valores no coincidan.

A continuación, se debe agregar esta expresión PromQL que se va a utilizar en la regla de las alertas. Además de agregar el Alertmanager que Prometheus va a utilizar. Para esto se debe editar el archivo de configuración "prometheus.yml" y añadir el siguiente texto.

```
$ vim prometheus.yml
alerting:
```

```
  alertmanagers:
  - static_configs:
    - targets:
      - localhost:9093
rule_files:
  - "rules.yml"
```

A continuación, se debe crear un nuevo archivo rules.yml con el siguiente contenido.

```
$ vim rule.yml
groups:
  - name: example
    rules:
    - alert: InstanceDown
      expr: up == 0
      for: 1m
```

La alerta InstanceDown se evaluará cada minuto. Si una serie se devuelve continuamente el valor configurado en la alerta durante al menos un minuto, se considerará que la alerta está activada.

Una vez que se tiene la alerta configurada se necesita una Alertmanager para disparar una acción. Para esto desde https://prometheus.io/download/, se descargó la última versión de Alertmanager para el sistema operativo Linux con arquitectura amd64.

Después de descargado y descomprimido el archivo Alertmanager, se debe configurar, existen varias opciones en las que Alertmanager puede enviar una notificación, para este caso se lo realizará por medio de un mensaje utilizando el API de Telegram.

A continuación, se debe modificar el archivo de configuración “alertmanager.yml”. Y agregar el siguiente texto.

```
$ vim alertmanager.yml
receivers:
```

```
- name: 'Telegram'
  webhook_configs:
  - url: 'http://127.0.0.1:4000/alert/prometheus'
```

3.3.4. Prueba de Alerta

Para comprobar el funcionamiento de la alerta se desconectó un host monitoreado por más de un minuto de la red. Desde la interfaz gráfica se pudo apreciar que se activó la alerta InstanceDown.

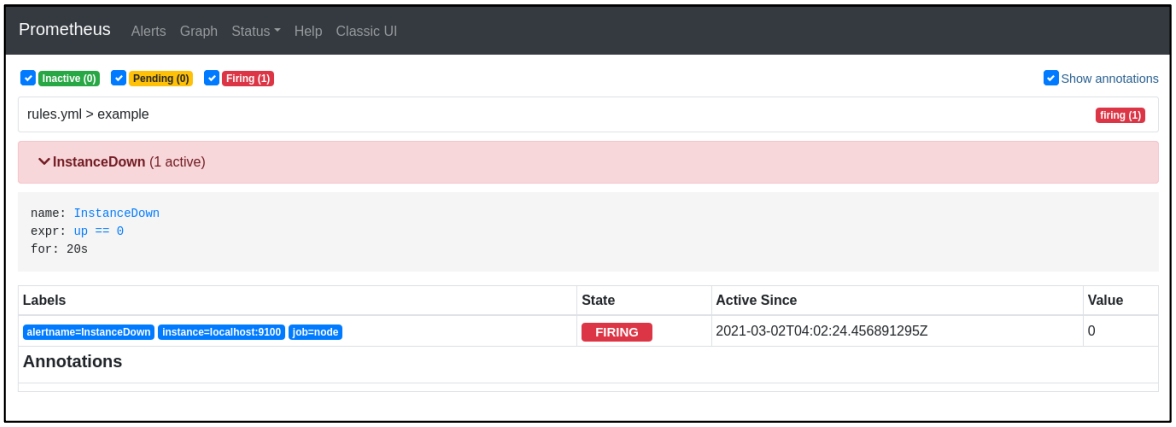


Figura 64. Ejecución de una alerta en Prometheus

3.3.5. Grafana

El navegador de expresiones de Prometheus es una excelente herramienta para gráficos ad hoc, pero no está diseñado para usarse como un panel de control. El panel de control es un conjunto de gráficos, tablas y otras visualizaciones del sistema.

Grafana es una popular herramienta con la cual se puede crear paneles de control. Es la herramienta más recomendada para crear paneles de control cuando se usa Prometheus, ya que mejora continuamente su compatibilidad.

La herramienta se puede descargar desde su web oficial https://grafana.com/grafana/download, una vez descargado y ejecutado se puede acceder a la interfaz gráfica por el puerto 3000, en este caso http://localhost:3000.

Para comenzar a usar Grafana hay que añadir una “Data Source” (fuente de datos), en las configuraciones.

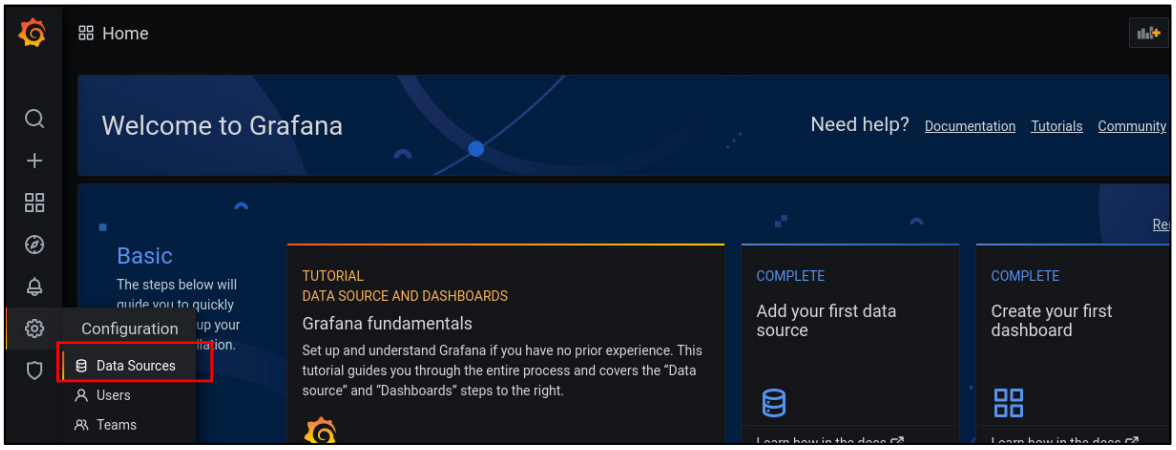


Figura 65. Módulo "Data Source" en barra lateral de Grafana

Una vez nos encontramos en la pantalla de configuración se tiene que utilizar el botón “Add data source” (añadir fuente de datos), y escoger la fuente que se planea utilizar.

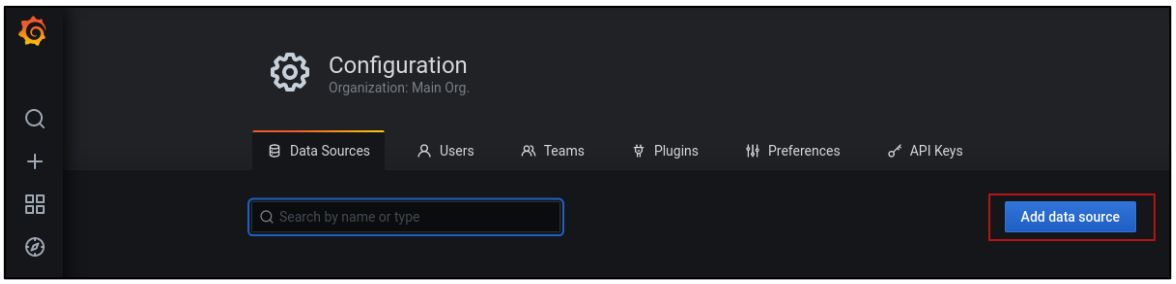


Figura 66. Opción de "Add data source" en módulo de configuración de Data Source

En este caso se va a utilizar la opción de Prometheus, para lo cual se da clic en el botón “Select” (seleccionar).

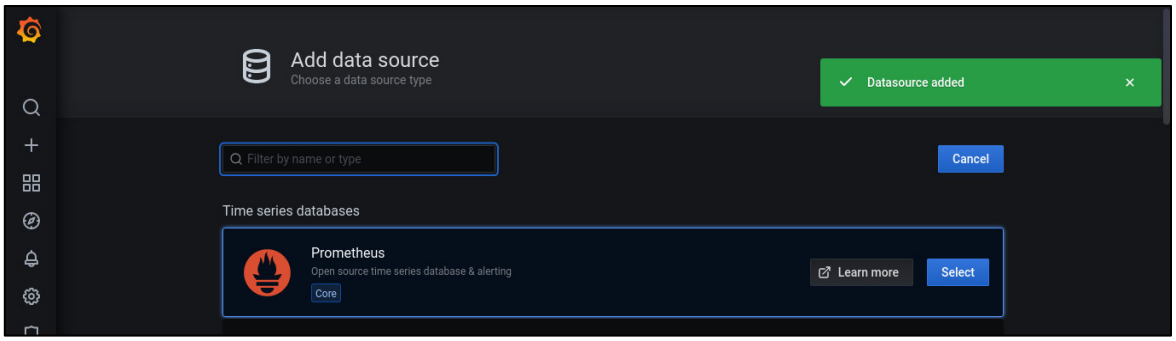


Figura 67. Lista de fuentes de datos de Grafana

En la pantalla de configuración se debe ingresar la URL en la que se está ejecutando el servicio de Prometheus, y grabar la configuración.

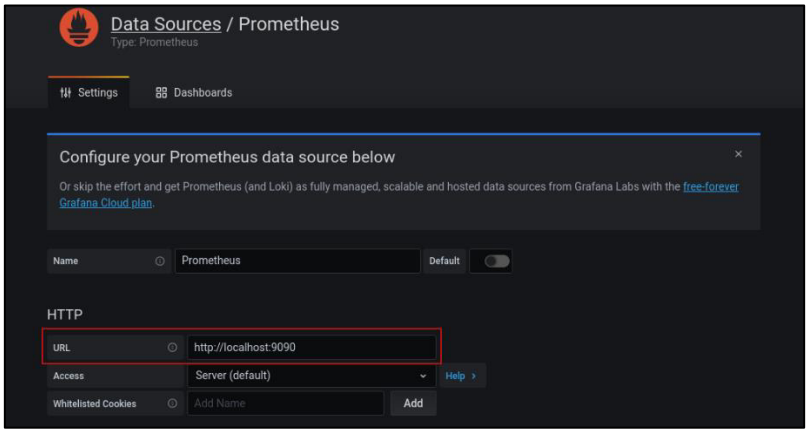


Figura 68. Formulario de configuración de fuente

El siguiente paso es crear el panel de control, Grafana cuenta con una gran cantidad de paneles de control creados por la comunidad, en este caso se utilizará uno diseñado para el módulo node_export, para esto hay que ir a la opción de “Dashboard->Manage” y usar la opción de “Import”.

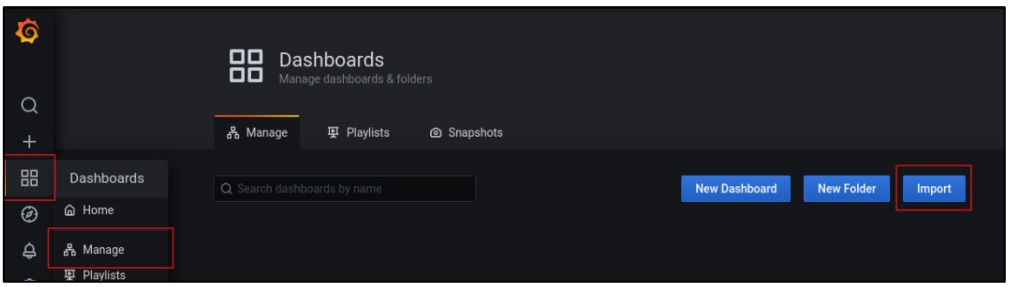


Figura 69. Modulo "Manage" en barra lateral y opción "Import" en Grafana

Para importar el panel de control solo es necesario ingresar el id 1860, y cargarlo.

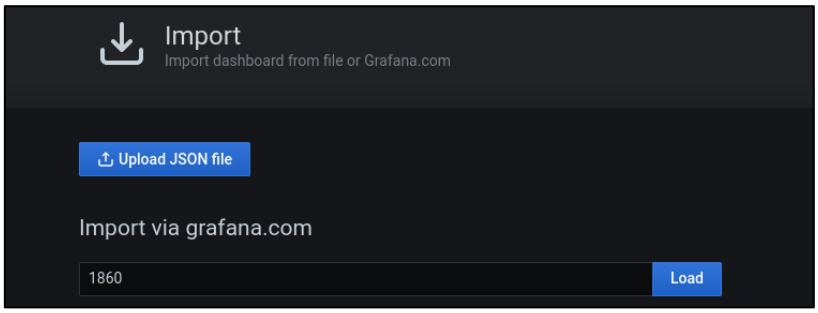


Figura 70. Formulario de importación de panel de control

En la siguiente imagen se puede observar algunas de las métricas que se encuentran configuradas en este panel de control, el cual posee una gran cantidad de métricas, útiles para observar el estado actual del host.

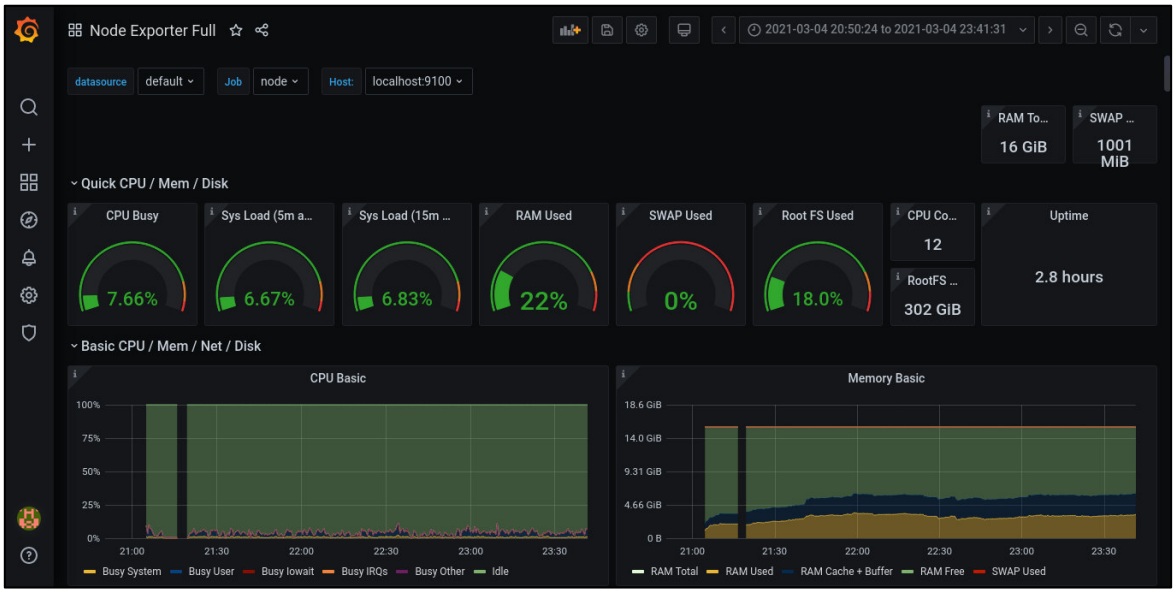


Figura 71. Panel de control de "Node Exporter Full"

3.4. Implementación de ELK Stack

Para instalar el ELK Stack se requiere un entorno de ejecución de java. Se puede verificar que alguna versión de java esté instalando con el siguiente comando.

```
# java -version
```

En caso de no tener una versión de java instalado, se puede instalar con el siguiente comando.

```
# sudo apt-get install default-jdk
```

Si ya se posee alguna versión de Java instalado en el sistema ya se puede proceder con las instalaciones de los diferentes componentes del ELK Stack.

3.4.1. Instalación de Elasticsearch

Se puede instalar la última versión de Elasticsearch desde su web oficial en https://www.elastic.co/downloads/elasticsearch, en esta ocasión se va a utilizar la

versión 7.11.2 para Linux, que se encuentra en el siguiente enlace https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.11.2-linux-x86_64.tar.gz. Una vez descargado el archivo hay que descomprimirlo, y ejecutar el programa.

```
# wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.11.2-linux-x86_64.tar.gz

# tar zxvf elasticsearch-7.11.2-linux-x86_64.tar.gz

# ./elasticsearch-7.11.2/bin/elasticsearch
```

3.4.2. Instalación de Logstash

Igualmente, para Logstash el software se puede descargar desde https://www.elastic.co/downloads/logstash, la última versión disponible al momento de realizar el documento es la 7.11.2 para Linux. Se debe realizar los mismos pasos vistos anteriormente.

```
# wget https://artifacts.elastic.co/downloads/logstash/logstash-7.11.2-linux-x86_64.tar.gz

# tar zxvf logstash-7.11.2-linux-x86_64

# ./logstash-7.11.2/bin/logstash
```

3.4.3. Instalación de Kibana

Este mismo proceso realizado anteriormente se debe realizar para instalar la última versión de Kibana, desde la misma página web https://www.elastic.co/downloads/kibana, la última versión en el momento de realizar este documento es la 7.11.2 para Linux, que se encuentra en el siguiente enlace https://artifacts.elastic.co/downloads/kibana/kibana-7.11.2-linux-x86_64.tar.gz.

```
# wget https://artifacts.elastic.co/downloads/kibana/kibana-7.11.2-linux-x86_64.tar.gz

# tar zxvf kibana-7.11.2-linux-x86_64.tar.gz

# ./kibana-7.11.2-linux-x86_64/bin/kibana
```

Kibana se ejecuta por defecto en el puerto 5601. Todas las configuraciones de Kibana se pueden modificar dentro de la carpeta de Kibana en el archivo config/kibana.yml. Ya que se tiene en ejecución un servicio de Elasticsearch, se modificó el archivo de configuración para añadir URL en la que se encuentra la instancia.

```
# vim config/kibana.yml

elasticsearch.hosts: ["http://localhost:9200"]
```

3.4.4. Ejecutar un Agente Logstash

Una vez se tenga todo el entorno levantado hay que crear un archivo de configuración para Logstash con el siguiente contenido.

Este archivo se divide en tres partes, en el input se especifica el origen de los datos, en este caso se va a leer el archivo de log “auth.log”, en donde se proporciona un registro de todas las actividades que implica un proceso de autenticación. Este archivo se va a utilizar para la creación de un gráfico que resuma las actividades del servicio SSH.

```
input{

file{

path => "/var/log/auth.log"

start_position => "beginning" }}
```

En el filter se realiza un preprocesamiento de los datos, en este caso se usó un plugin de grok que sirve para analizar datos de eventos no estructurados y transformarlo en campos.

```
filter{

grok{

match => {"message" => ['%{SYSLOGTIMESTAMP:system.auth.timestamp}
%{SYSLOGHOST:system.auth.hostname} sshd\[%{POSINT:system.auth.pid}\]:
%{DATA:system.auth.ssh.event} password for %{DATA:system.auth.user} from
%{IPORHOST:system.auth.ip} port %{NUMBER:system.auth.port} ssh2']}
```

```
} }
```

En el output se especifica a donde se va a enviar los datos, es la etapa final de la canalización de eventos, en este caso se va a enviar todos los eventos procesados al servicio de Elasticsearch.

```
output{
elasticsearch{
hosts => ["127.0.0.1:9200"]
} }
```

Una vez se tenga listo el archivo de configuración de Logstash se debe ejecutar el binario de Logstash haciendo referencia el archivo de configuración.

```
# ./logstash -f ssh.conf
```

3.4.5. Creación de Index Pattern (Patrón de índice)

Como siguiente paso hay que crear un “Index Pattern”, este proceso se puede realizar desde Kibana, en el apartado "Stack -> Index Patterns", utilizando el botón "Create index patten".

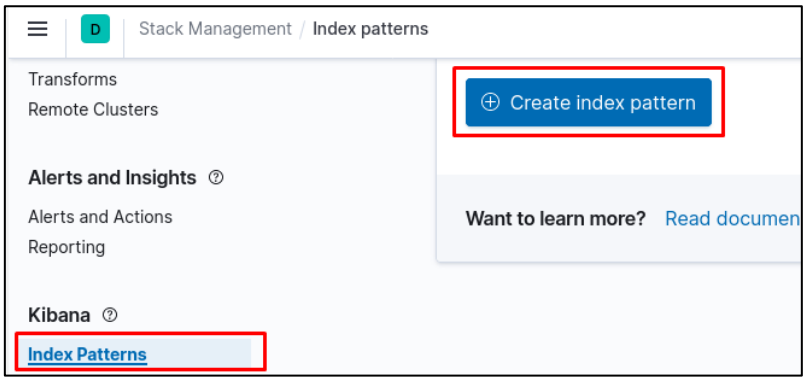


Figura 72. Módulo "Index Patterns" y opción "Create index pattern" en Kibana

En la interfaz de creación hay que escribir el nombre del “Index Patter”, en este caso se usó la palabra “logstash-*” para hacer referencia a todos los datos que provienen de esa fuente.

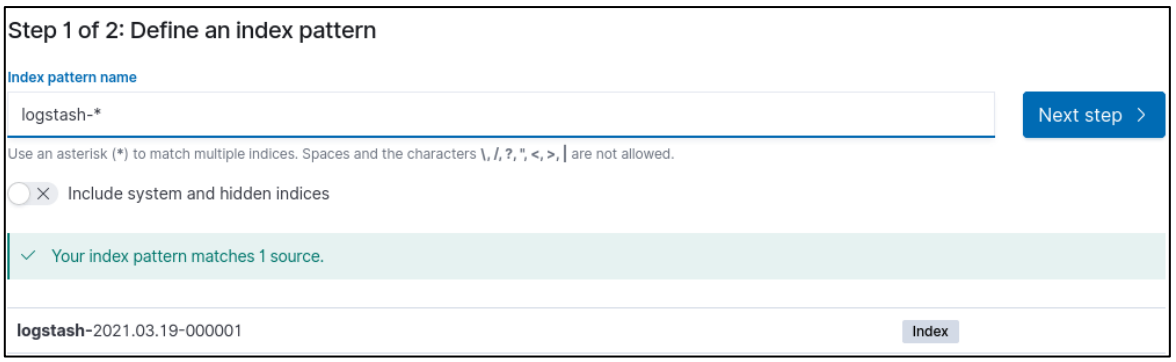


Figura 73. Formulario de definición de un patrón de índice

En el siguiente paso se debe escoger un campo principal para este “Index Pattern”. Para esta ocasión no se va a escoger ningún campo de referencia para el tiempo.

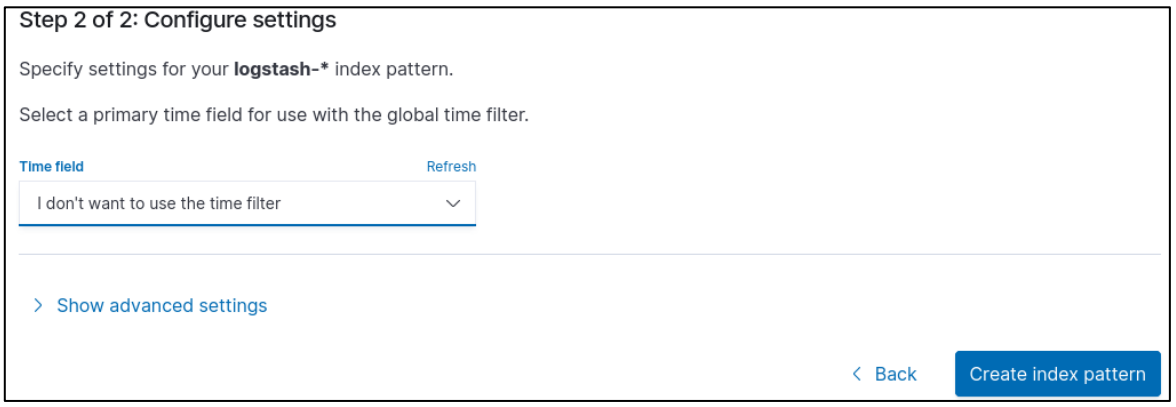


Figura 74. Formulario de configuración de patrón de índice

3.4.6. Visualización de Eventos

A continuación, se va a crear un gráfico en donde se pueda visualizar las autenticaciones realizadas por SSH en el host, en este gráfico se va a distinguir los accesos exitosos de los fallidos. Para esto nos dirigimos al apartado de “Visualize” en el menú.

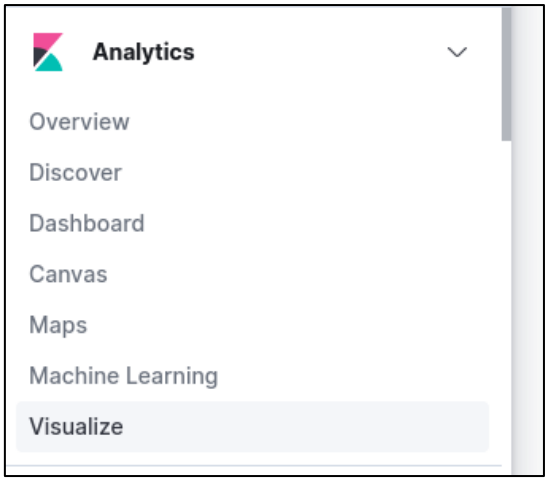


Figura 75. Módulo "Visualize" en barra lateral de Kibana

Se seleccionar la opción de "Create new visualization" (crear una nueva visualización en español).

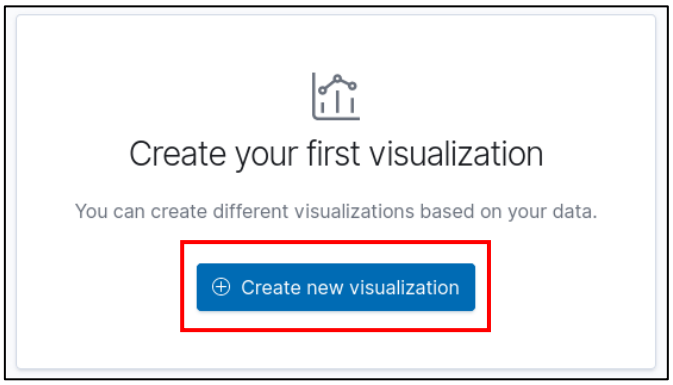


Figura 76. Opción de "Create new visualization" en pantalla de Visualización

Seleccionar la opción de “Lens”.

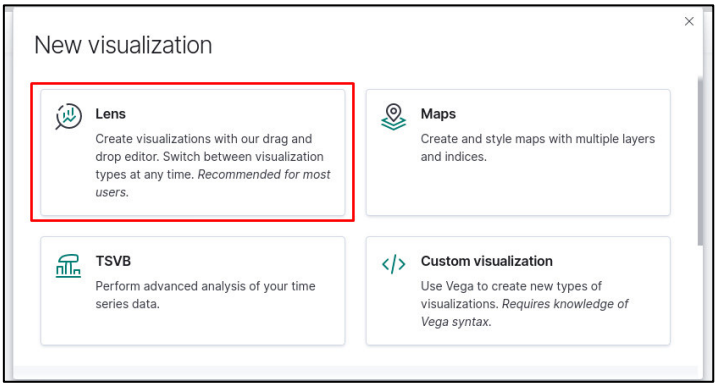


Figura 77. Opción de Lens en formulario de "New visualization"

En la interfaz de visualización se debe escoger los campos que se van a utilizar, para realizar este gráfico, en el eje horizontal se seleccionara @timestamp y en el vertical "auth.ssh.event.keyword", además de escoger los campos se utilizó la funcionalidad de "Break down by" para desglosar los diferentes datos sé que pueden encontrar en el campo “auth.ssh.event.keyword”, como se puede apreciar en el gráfico se tiene una visualización de los intentos de autenticación por SSH, de color verde los que fueron exitoso y de color azul los fallidos.

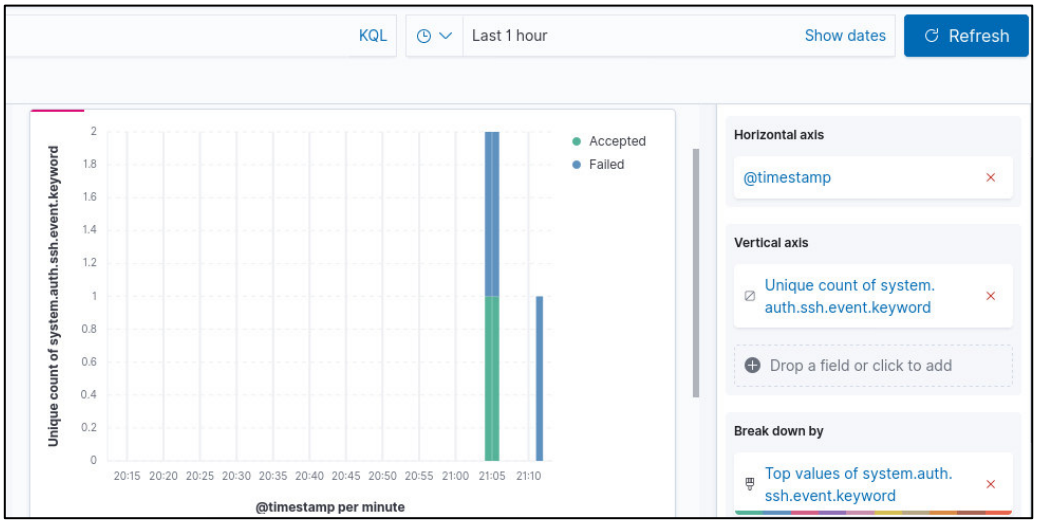


Figura 78. Gráfico de barra de Kibana

3.4.7. Creación de Alerta

Existen varias configuraciones de alerta que ofrece el ELK stack, en este caso se usará un plugin de LogStash para monitorear los intentos de acceso fallido en el sistema. Además, se utilizará el API Rest creado anteriormente para la comunicación con el API de Telegram.

Para esto hay que incluir las siguientes líneas en el archivo de configuración “ssh.conf”.

```
output{
elasticsearch{
hosts => ["127.0.0.1:9200"]
}
if "Failed" in [system.auth.ssh.event] {
```

```
http{
http_method => "post"
url => "http://localhost:4000/api/alert/logstash/ssh"
content_type => "application/json"
}}}
```

En el plugin output se agregó una condición, en el caso de tener la palabra “Failed” en el campo “system.auth.ssh.event” se va a utilizar un plugin http, el cual permite enviar eventos a un extremo http.

En el plugin se debe configurar la forma en la que se va a consumir el método, en este caso va a realizar una petición POST, en donde se va a enviar todos los parámetros del evento.

El código del API Rest se encuentra en el siguiente repositorio en GitHub https://github.com/Dandrezz/Alerts.

Una vez levantado el entorno del ELK stack y el API Rest, se realizó una prueba de su funcionamiento. Para eso se ingresó incorrectamente las credenciales de un usuario por SSH.

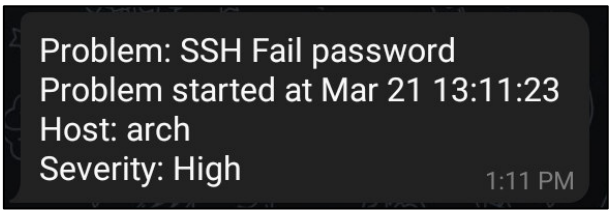


Figura 79. Visualización de alerta en aplicación móvil de Telegram

La alerta se envió exitosamente, y se puede apreciar que se muestran varios datos sobre la alerta.

3.5. Monitoreo de Aplicaciones

3.5.1. Servidor HTTP Apache y API Rest

3.5.1.1. Zabbix

Para implementar un Template de Apache en Zabbix, hay que ir a “Configuraciones -> Host” y seleccionar el host en donde se desea implementar, en este caso en el servidor Zabbix e ir al apartado de Templates. En la interfaz de Template se debe usar la opción de “Select” y escoger el Template, para facilitar la búsqueda se filtró por aplicaciones y se seleccionó el “Template App apache by Zabbix agent”.

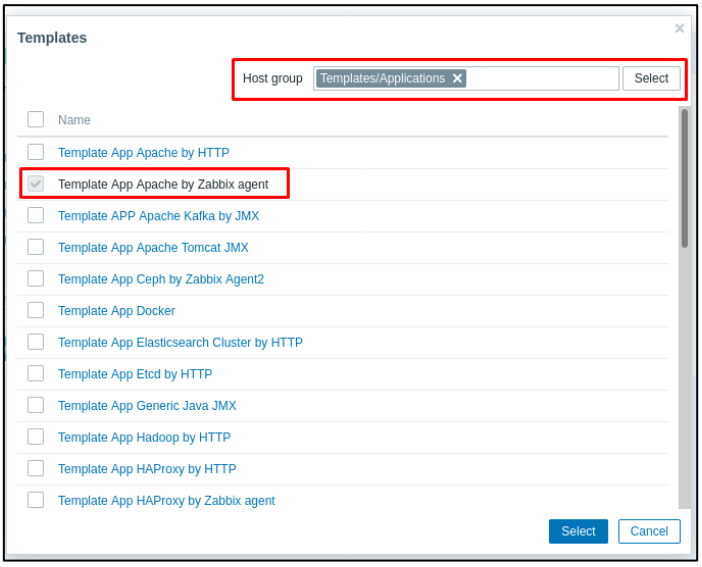


Figura 80. Lista de Templates de Zabbix

En la siguiente imagen se puede observar que el Template está enlazado con el host, solo hace falta usar la opción de “Update” y ya se estarán monitorear todas las métricas que se encuentran en el “Template App apache by Zabbix agent”.

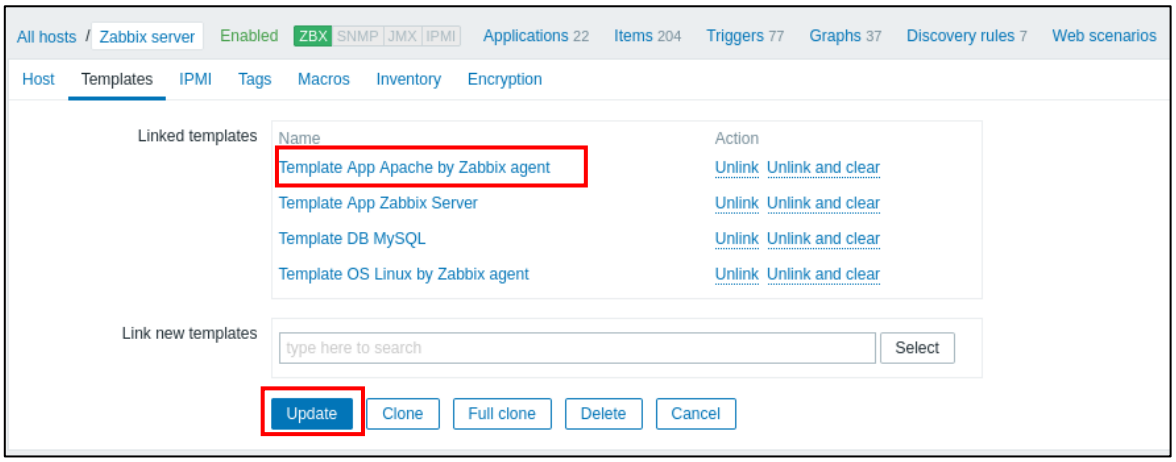


Figura 81. Pantalla de configuración de Templates de Zabbix

Para verificar su funcionamiento podemos ir al gráfico, en “Monitoring -> Lastest Data”, y seleccionar uno de los Items relacionados con Apache, un Item significativo es la cantidad de requests por segundo por lo cual se lo seleccionó. Como se puede ver en el siguiente gráfico el Item funciona correctamente.

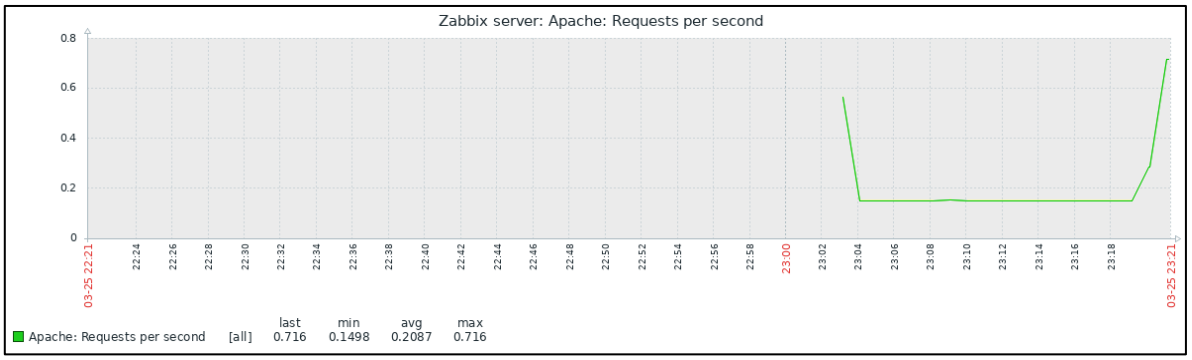


Figura 82. Gráfico de líneas de peticiones por segundo de un servidor HTTP

3.5.1.2. Prometheus

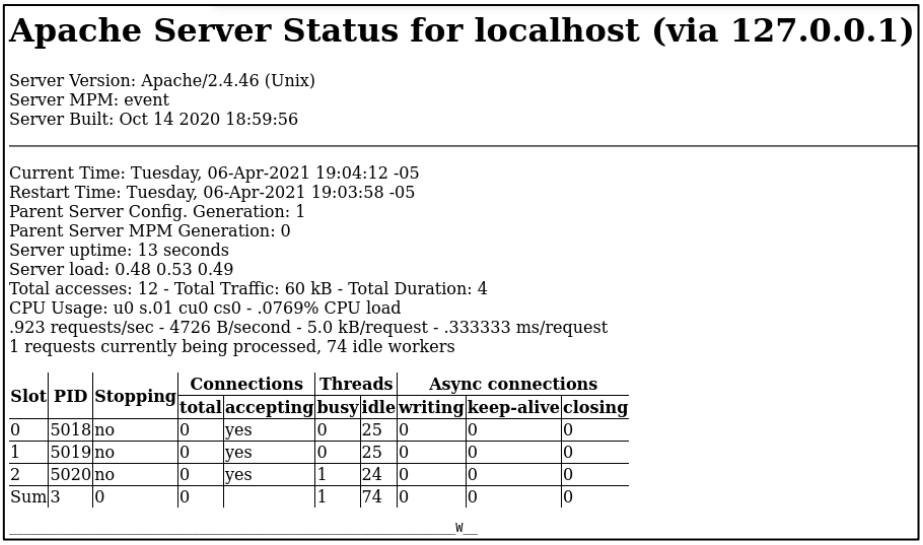
Para exponer varios datos de rendimiento del servidor HTTP Apache es necesario habilitar el módulo “server-status” (página de estado en español). Esto se realizó agregando las siguientes líneas al archivo de configuración de Apache.

```
# vim /etc/apache/conf/httpd.conf
ExtendedStatus on
<Location /server-status>
SetHandler server-status
</Location>
```

Una vez realizado el cambio es necesario reiniciar el servicio Apache.

```
# systemctl restart apache2.service
```

Para comprobar que la información se encuentra disponible nos dirigimos a la dirección http://localhost/server-status. Como se puede ver en la siguiente imagen se tiene acceso a varias métricas del servicio.



Apache Server Status for localhost (via 127.0.0.1)

Server Version: Apache/2.4.46 (Unix)
Server MPM: event
Server Built: Oct 14 2020 18:59:56

Current Time: Tuesday, 06-Apr-2021 19:04:12 -05
Restart Time: Tuesday, 06-Apr-2021 19:03:58 -05
Parent Server Config. Generation: 1
Parent Server MPM Generation: 0
Server uptime: 13 seconds
Server load: 0.48 0.53 0.49
Total accesses: 12 - Total Traffic: 60 kB - Total Duration: 4
CPU Usage: u0 s.01 cu0 cs0 - .0769% CPU load
.923 requests/sec - 4726 B/second - 5.0 kB/request - .333333 ms/request
1 requests currently being processed, 74 idle workers

Slot	PID	Stopping	Connections		Threads		Async connections		
			total	accepting	busy	idle	writing	keep-alive	closing
0	5018	no	0	yes	0	25	0	0	0
1	5019	no	0	yes	0	25	0	0	0
2	5020	no	0	yes	1	24	0	0	0
Sum	3	0	0		1	74	0	0	0

W__

Figura 83. Página de información de Apache

Para que Prometheus recopile las métricas de Apache, se utilizara un exportador para Apache, el cual se encuentra en el siguiente repositorio https://github.com/Lusitaniae/apache_exporter. Para poder ejecutar el exportador es necesario tener instalado Golang en el sistema. Dentro del archivo apache_export se utilizó el siguiente comando.

```
# go run .
```

El siguiente paso es ingresar las fuentes de las métricas en el archivo de configuración de Prometheus “prometheus.yml”.

```
# vim prometheus.tml
scrape_configs:
  - job_name: 'apache'
    static_configs:
    - targets: ['localhost:9117']
```

Una vez realizado el cambio y levantado el servicio de Prometheus, ya se estarán registrando las métricas del servidor HTTP Apache en Prometheus. A continuación, se creará un panel de control para poder ver algunas de estas métricas de forma gráfica.

Para esto nos dirigimos a Grafana en el módulo “Create”, se utiliza la opción de “Import”. Existen varios paneles de control registrados por la comunidad de Grafana, en esta ocasión se usará el que tiene ID “3894”.

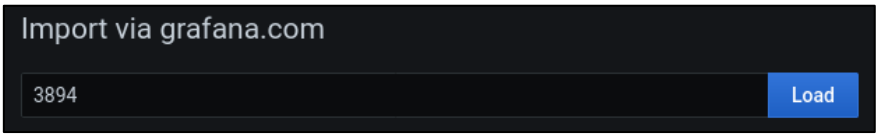


Figura 84. Importación de un panel de control en Grafana

Se escoge la fuente de los datos y se importa el panel de control.

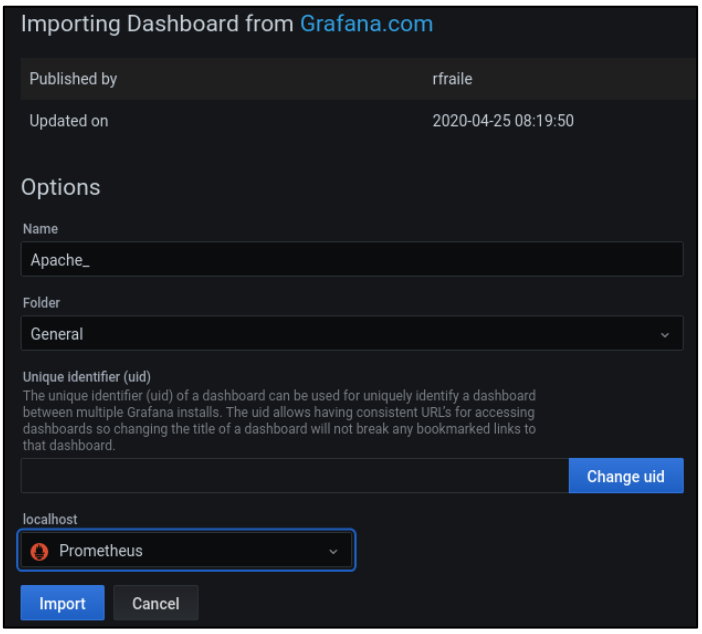


Figura 85. Formulario de importación de panel de control

En la siguiente imagen se puede ver algunos de los datos que se muestran en el panel de control. Se puede visualizar el tiempo de actividad, el estado del servicio a lo largo del tiempo, total de bytes enviados actualmente, entre otros datos.

Figura 86. Panel de control de servidor Apache en Grafana

Uno de los factores fuertes de Prometheus es que existen librerías que se pueden implementar en un software a medida para de esta forma poder acceder a sus métricas. A continuación, se presentará un ejemplo de monitoreo de las métricas de un API Rest escrito en JavaScript y ejecutado con Node.

La librería que se usó en el API Rest fue “prometheus-api-metrics”, se puede encontrar más documentación de esta librería en el siguiente URL https://www.npmjs.com/package/prometheus-api-metrics. Se instaló y se instanció un middleware en las configuraciones del servidor API Rest.

Para poder capturar la nueva fuente de datos en Prometheus se debe editar su archivo de configuración. En donde se debe añadir un nuevo job.

```
# vim prometheus.yml
scrape_configs:
  - job_name: 'apiRest'
    static_configs:
    - targets: ['localhost:4000']
```

Una vez se reinicie el servicio de Prometheus ya se estarán registrando las métricas de la aplicación. En el siguiente paso se va a crear un panel de control con esta información.

Desde Grafana en el módulo “Create” se utilizó la opción “Import”. Y se cargó el panel de control con id “11159”.

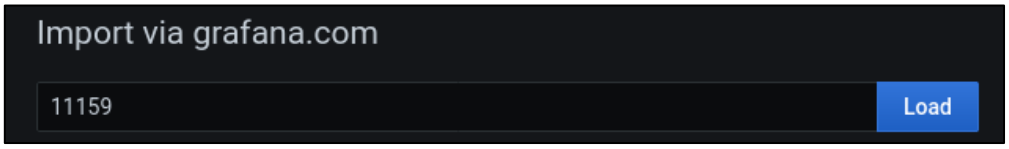


Figura 87. Importación de panel de control

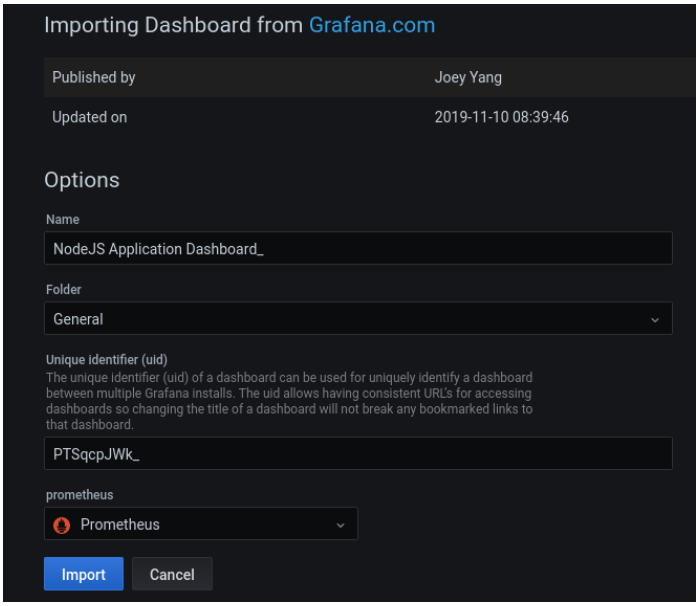


Figura 88. Formulario de importación de Grafana

En la siguiente imagen se puede ver algunos de los datos que se encuentran configurados en el panel de control. Entre los datos que se muestran está la versión de Node, el consumo del procesador, consumo de la memoria, entre otros datos.

Figura 89. Panel de control de aplicación Nodejs

3.5.1.3. ELK Stack

Con el Stack ELK se va a monitorear los datos registrados en los logs de un servidor HTTP Apache. Para esto se creó un API Rest escrita en Javascripts y ejecutada con Nodejs. El servidor Apache va a interactuar como un proxy y balanceador de carga del API Rest. El código del API Rest utilizado se encuentra en el siguiente repositorio https://github.com/Dandrezz/Alerts.

En el servidor Apache se creó un archivo de configuración en donde se especifica el puerto en donde se encuentra ejecutándose la aplicación y el puerto que va a utilizar el servidor Apache para exponerlo. También se configuró los archivos logs que se van a generar.

```
# vim /etc/apache2/sites-available/miapi.conf
Listen 5000
<VirtualHost *:5000>
        ServerName node.api
        ProxyPreserveHost on
        ProxyPass / http://127.0.0.1:4000/
        ProxyPassReverse / http://127.0.0.1:4000/
 ServerAdmin webmaster@localhost
ErrorLog ${APACHE_LOG_DIR}/api_error.log
```

```
CustomLog ${APACHE_LOG_DIR}/api_access.log vhost_combined
</VirtualHost>
```

En este caso se van a crear dos archivos. Uno para los errores y otro para los demás registros. Además, el formato de logs que se va a usar para los registros es el de vhost_combined, la definición de este formato se encuentra en el archivo “apache.conf”.

```
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

Figura 90. Formato de los logs en archivo apache.conf

El formato vhost_combined contiene los siguientes datos:

- IP del cliente
- Identificación del Usuario
- Autentificación
- Tiempo de petición
- Verbo de la petición
- Request
- Versión HTTP
- Código de respuesta
- Número de Bytes
- Una referencia
- User-Agent

Todos estos datos se deben tener en cuenta al momento de realizar el filtrado de los datos en Logstash. El siguiente paso es crear el archivo de configuración de Logstash, en este caso se lo va a llamar “apache.conf”.

El archivo de configuración de logstash va a estar dividido en tres partes, el input en donde se especifica las fuentes de donde va a obtener los datos, en este caso se los va a obtener del archivo “api_access.log”.

```
input {
file {
path => "/var/log/apache2/api_access.log"
start_position => "beginning"
}}
```

El filter es donde se va a realizar el procesamiento de los datos, en este caso se va a usar grok para separar los datos, para esto se va a usar un patrón que se encuentra en el archivo “grok-patterns”, este patrón tiene el nombre de “COMBINEDAPACHELOG”. Una vez que se obtengan los campos se va a aplicar una operación convert para cambiar el tipo de datos de response y bytes a entero y float respectivamente.

```
filter {
grok {
patterns_dir => ["./grok-patterns"]
match => {
"message" => ["%{COMBINEDAPACHELOG}+%{GREEDYDATA:extra_fields}"]
}}
mutate {
convert => ["response", "integer"]
convert => ["bytes", "float"]
}}
```

Y por último el plugin output, en donde se configura la salida de los datos, en este caso se va a utilizar Elasticsearch para almacenar los datos. Se escribe la dirección IP y puerto en donde se encuentra, además se incluyó el nombre del “index” para diferenciar los datos que provengan de los logs de apache de los demás.

```
output{
elasticsearch{
hosts => ["127.0.0.1:9200"]
index => "apache-%{+YYYY.MM.dd}"
}}
```

Para ejecutar Logstash con su archivo de configuración se usa el siguiente comando.

```
# ./logstash -f apache.conf
```

A continuación, hay que ir a crear el index en Kibana para esto hay que ir a “Stack Management -> Index Patterns” y utilizar la opción de “Create index pattern”.

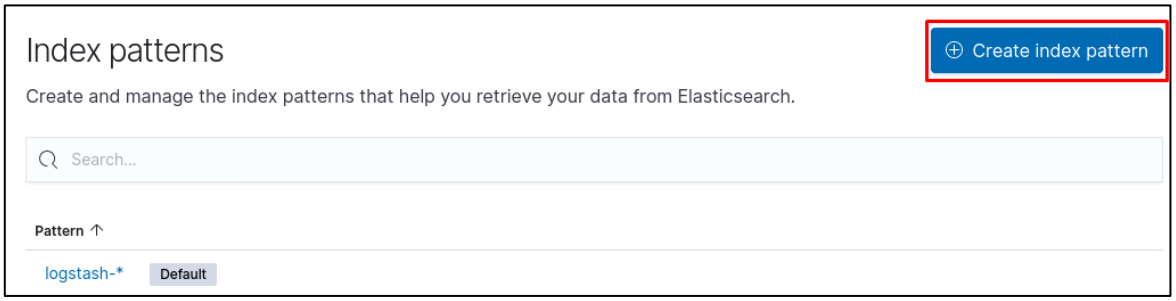


Figura 91. Opción "Create index pattern" en formulario patrón de índice

La cadena de texto que se va a usar para la creación del “Index Patern” es "apache-*" y se continúa con el procedimiento.

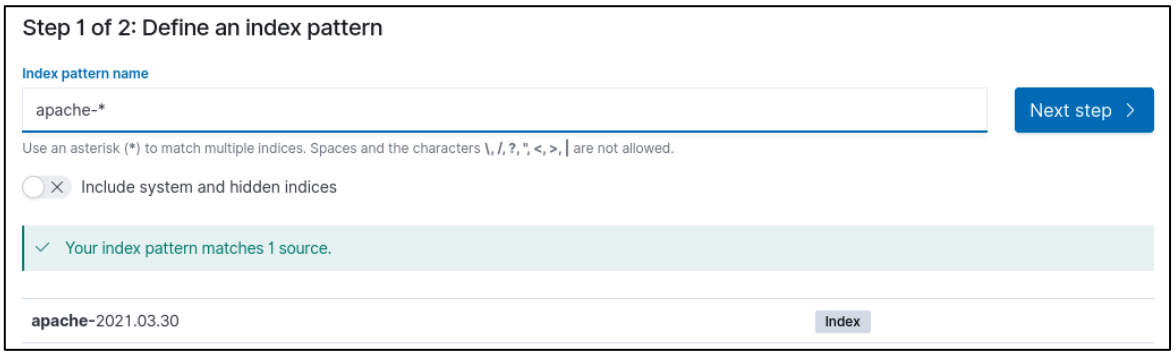


Figura 92. Formulario de definición de patrón de índice

Se escoge el campo en donde se registra el tiempo y se crea el index.

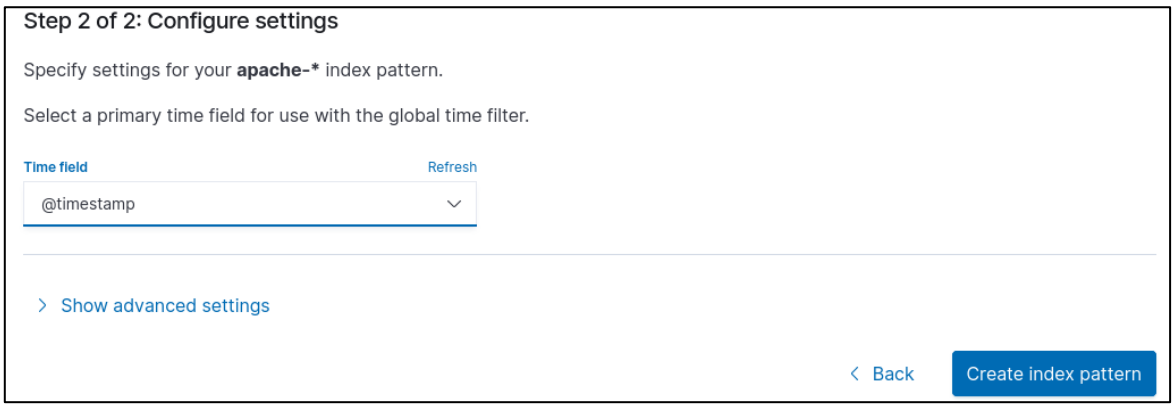


Figura 93. Formulario de configuración de patrón de índice

Una vez creado el “Index Pattern” ya se puede realizar varios gráficos para mejorar la presentación de los datos. Como primer gráfico se creó una tabla con los User-Agent más usados, para esto se seleccionó el campo "Top values of Agent" y se agregó una métrica de conteo.

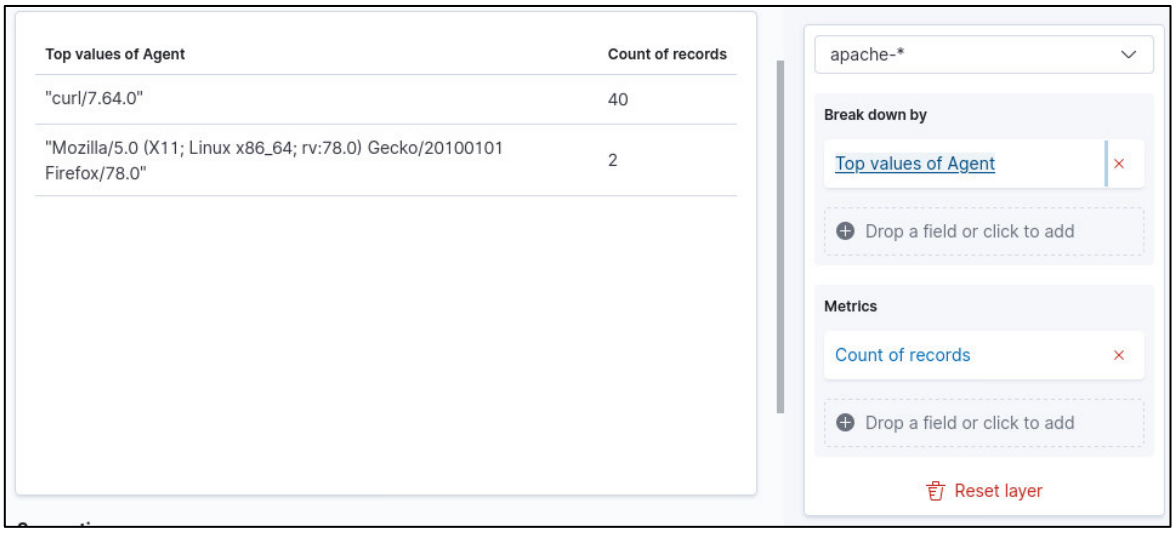


Figura 94. Gráfico de tabla de registros de agentes del servidor Apache

En el siguiente gráfico representa la distribución de los verbos de las peticiones que ha recibido el servidor http, como se puede ver el 33.33% de peticiones que ha recibido el servidor han sido POST mientras que el resto han sido peticiones GET.

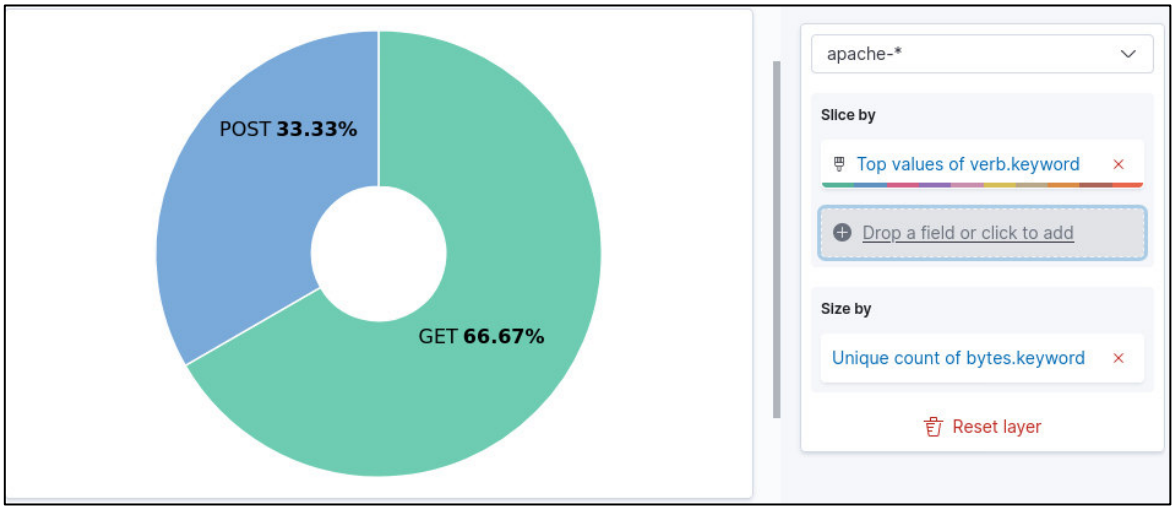


Figura 95. Gráfico de pastel de tipo de peticiones recibidas

También se creó un en donde se muestra la distribución de peticiones ordenadas por request, en este gráfico se puede observar que el método más utilizado en el API Rest es el que se encuentra en la ruta "/api".

Figura 96. Gráfico de barras de los métodos más usados en el servidor Apache

Los gráficos que se mostraron anteriormente son unos pocos ejemplos de lo que se puede hacer con los datos registrados por el servidor http, dependiendo de las necesidades del negocio se puede crear distintos gráficos para mostrar la información más importante.

3.5.1.3.1. Alertas ELK

Para este ejemplo se creará una alerta para llevar un control de la cantidad de peticiones que son respondidas con un Bad Requests en un periodo de tiempo. Existen varias formas de realizar este control para este caso se usará las herramientas que nos ofrece Kibana. Antes de poder utilizar las alarmas de Kibana es necesario configurar el archivo kibana.yml. En donde se debe ingresar un string de 32 o más caracteres para cifrar propiedades confidenciales en las reglas y acciones de alertas antes de que se almacenen en Elasticsearch.

```
# vim config/kibana.yml
xpack.encryptedSavedObjects:
  encryptionKey: "min-32-byte-long-strong-encryption-key"
```

Además, se debe realizar varios cambios al archivo de configuración de Logstash, ya que solo se quiere crear una acción para un exceso de respuestas con códigos de errores. En el plugin de output se agregará una condición en la que todos los http response status mayores a 400 se van a almacenar en un índice con el nombre de apache-error.

```
output{
if [path] =~"api_access"{
if [response] >= 400 {
elasticsearch{
hosts => ["127.0.0.1:9200"]
index => "apache-errors-%{+YYYY.MM.dd}"
}
}else{
elasticsearch{
hosts => ["127.0.0.1:9200"]
index => "apache-%{+YYYY.MM.dd}"
}}}}
```

Una vez se esté ejecutando Logstash se debe crear el índice en Elasticsearch, este proceso se puede realizar desde Kibana en el apartado “Stack Management -> Index patterns”, utilizando la opción de “Create index pattern”.

En la interfaz de “Create index pattern” escribe el patrón que se va a utilizar para obtener los datos del índice, en este caso “apache-error-*”.

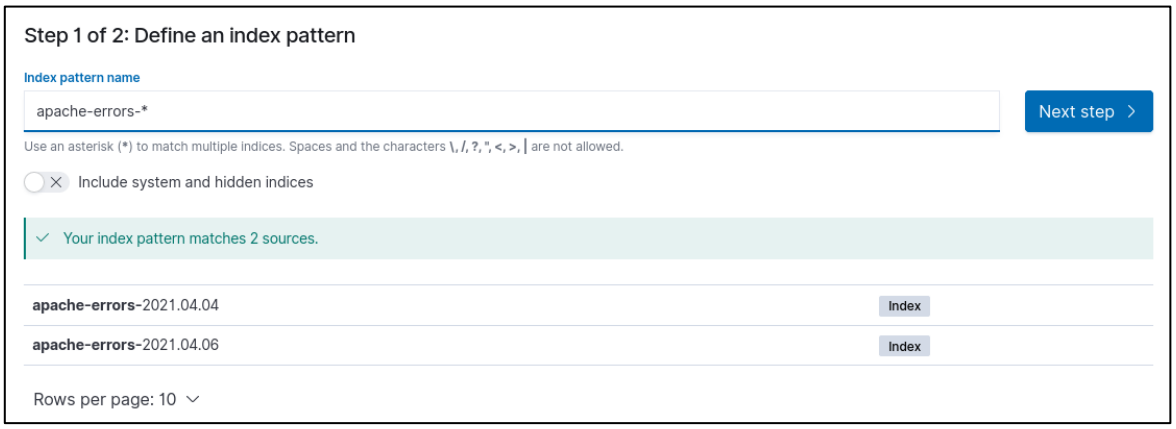


Figura 97. Formulario de definición de índice de patrón

Y se escoge el campo que se usará para los filtros de tiempo, en este caso @timestamp.

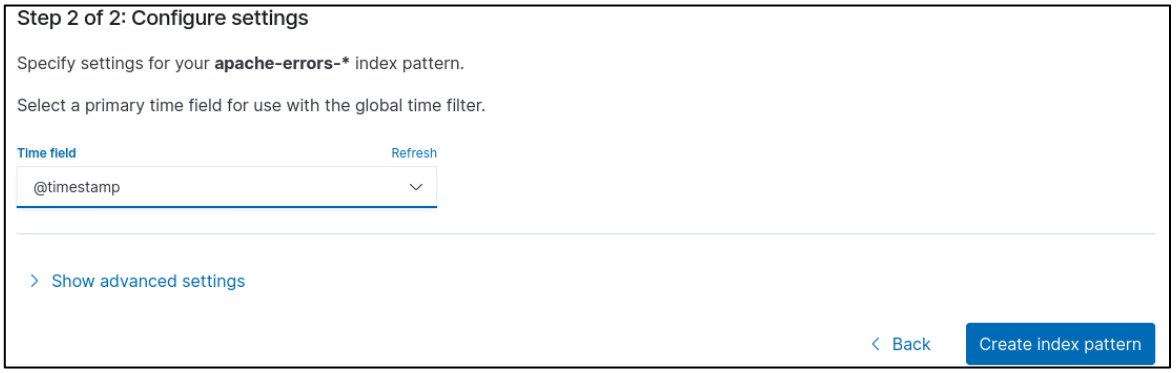


Figura 98. Formulario de configuración de patrón de índice

Una vez creado el índice hay que ir al apartado “Stack Manager-> Alert”, y utilizar la opción de “Create alert” (Crear alerta en español).

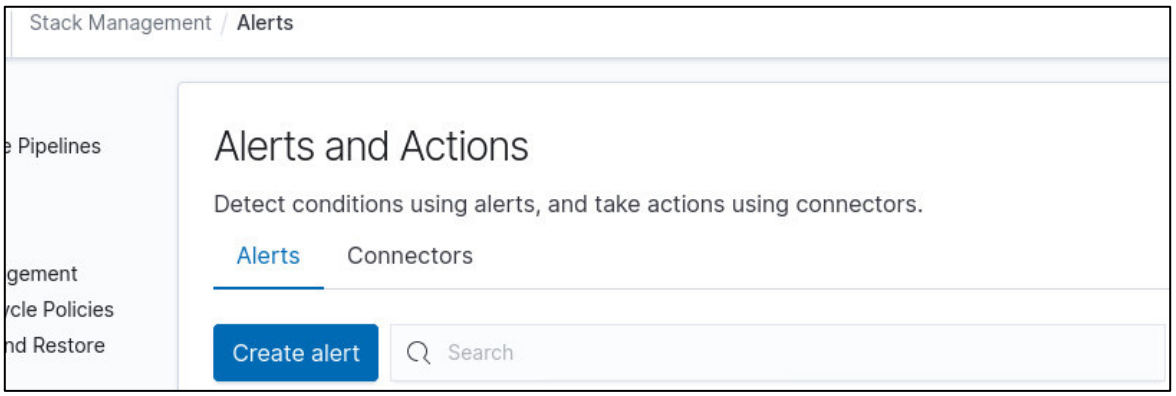


Figura 99. Módulo de alertas y acciones Kibana

Como siguiente paso hay que llenar un formulario con las configuraciones de la alerta. La comprobación se va a realizar una vez cada minuto y la notificación se va a realizar una vez cuando se cambie de estado.

La fuente de información que se va a usar para la alerta se encuentra en un índice por lo que se va a usar la opción de “Index Threshold”.

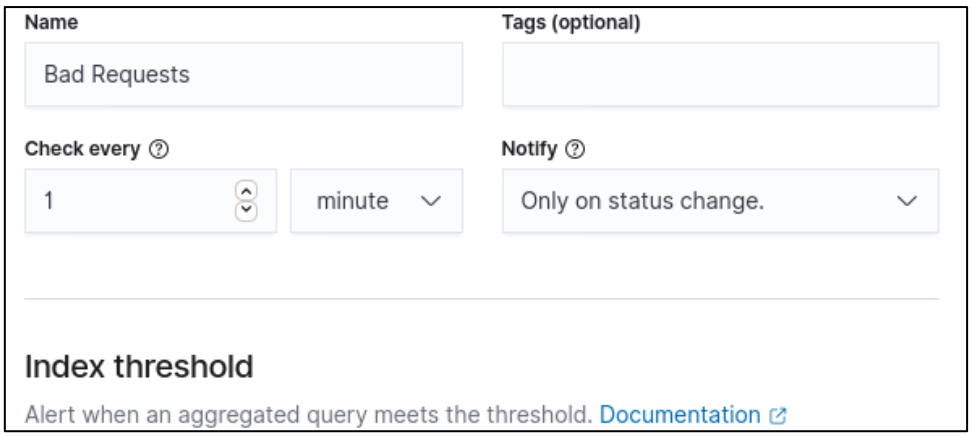


Figura 100. Sección de formulario de creación de alerta Kibana

Como se dijo anteriormente se va a obtener los datos almacenados en el índice apache-error-*, en la cláusula “when” se especifica cómo se va a calcular el valor que se va a comparar con el umbral, en este caso se utilizara “count()” para contabilizar la cantidad de peticiones que se generan en un periodo de tiempo. Y por último se va a dividir todos los documentos en tres grupos, estos grupos son los que posean las cardinalidades más altas de todos los documentos.

Figura 101. Sección de formulario de selección de índice

En la definición de la condición se configura un umbral, un operador de comparación y el periodo de tiempo que se debe tener en cuenta para buscar los documentos. En este caso como umbral se va a poner cuando sea mayor a mil en un periodo de un minuto.

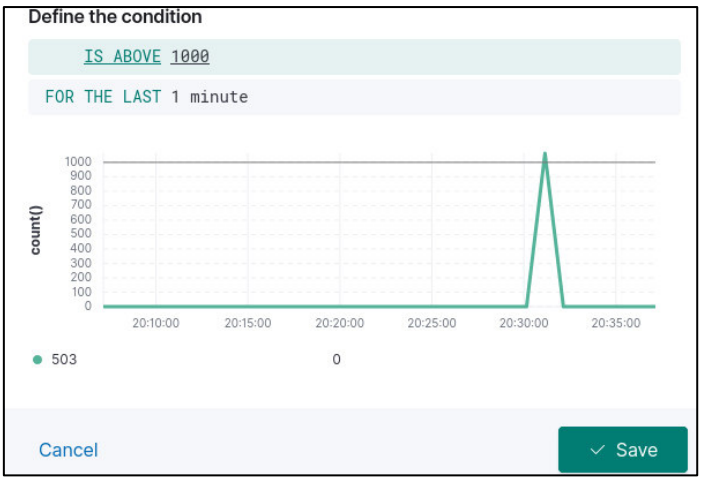


Figura 102. Sección de formulario de definición de condición

A continuación, se va a crear un “connector” en la misma sección de creación de alertas existe un módulo enrutar para acciones a diferentes destinos, como archivos de log. En esta ocasión se va a registrar las alertas en un archivo log, que por medio de Logstash va a servir para enviar notificación por medio de Telegram. Este proceso se puede realizar de forma directa sin usar Logstash en las versiones pagadas de Kibana, pero en este caso solo se tiene acceso a la versión estándar.

En el módulo de “connector” se utiliza la opción de “Create connector”, y en la interfaz de creación se escogió la opción de server log, en la siguiente interfaz se ingresa el nombre del “connector” y se guarda el “connector".

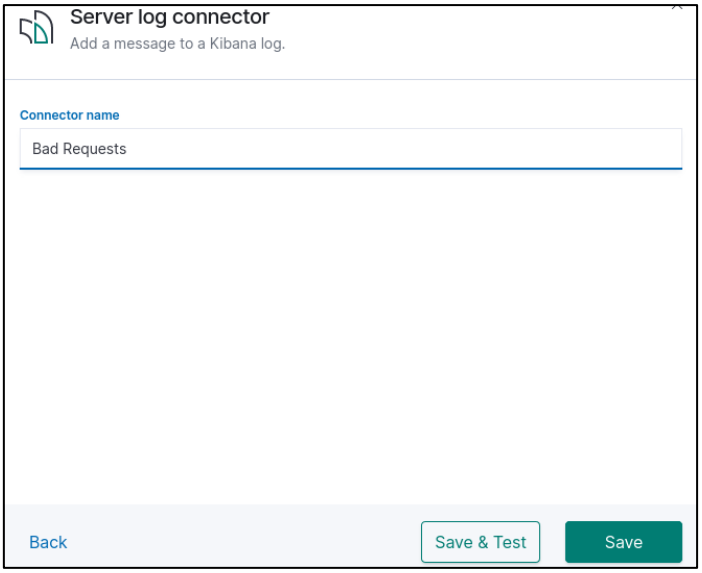


Figura 103. Formulario de "Server log connector"

Para enlazar la alerta con el conector hay que editar la alerta, en parte inferior de la interfaz de edición se escoge el “connector” llamado “Test”, se agrega el mensaje que se va a enviar, se escoge el nivel de la alerta y se guarda la configuración.

```
alert '{{alertName}}' is active for group '{{context.group}}':
- Value: {{context.value}}
- Conditions Met: {{context.conditions}} over
{{params.timeWindowSize}}{{params.timeWindowUnit}}
- Timestamp: {{context.date}}
```

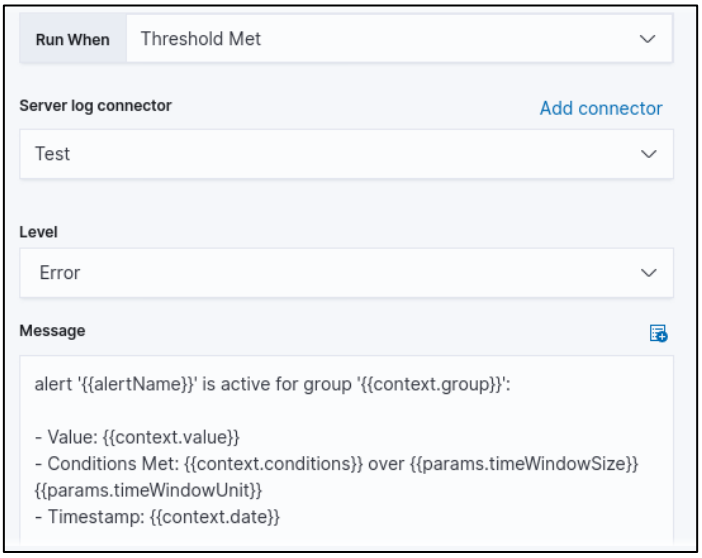


Figura 104. Formulario de configuración de alerta

Para que la acción “Test” se registre en un archivo es necesario agregar los siguientes comandos en el archivo de configuración de Kibana. En logginng.dest se especifica el archivo donde se van a guardar los logs y en logging.quit se especifica que solo se quiere almacenar los mensajes de error.

```
# vim conf/kibana.log
logging.dest: /var/log/kibana.log
logging.quiet: true
```

Para que los cambios se apliquen es necesario reiniciar Kibana.

En el archivo de configuración de Logstash se deben realizar varios cambios para poder leer los logs registrados tanto de Apache como de Kibana. En el plugin de input

se debe agregar el archivo de Kibana, ya que los logs registrados en este archivo están en formato json se debe especificar el codec.

```
input {
file
{
path => "/var/log/httpd/api_access.log"
start_position => "beginning"
}
file
{
path => "/var/log/kibana.log"
start_position => "beginning"
codec => json
}}
```

En el plugin de output se debe discriminar los datos, esto se lo realizó por medio de condiciones evaluando la ruta archivo log, y configurar el destino de los datos, ya que se desea enviar una notificación por medio de Telegram, se utilizará un plugin http para consumir un servicio del API Rest de notificaciones creado anteriormente.

```
output{
if [path] =~"api_access"{
if [response] >= 400 {
elasticsearch{
hosts => ["127.0.0.1:9200"]
index => "apache-errors-%{+YYYY.MM.dd}"
}
}else{
elasticsearch{
hosts => ["127.0.0.1:9200"]
index => "apache-%{+YYYY.MM.dd}"
}}}
if [path] =~ "kibana" {
```

```
http{
http_method => "post"
url => "http://localhost:5000/api/alert/elk"
content_type => "application/json"
}}}
```

Para verificar el funcionamiento se generaron alrededor de mil peticiones con código de respuesta 404 Bad Requests y como se puede ver a continuación el mensaje se envió correctamente por medio de Telegram.

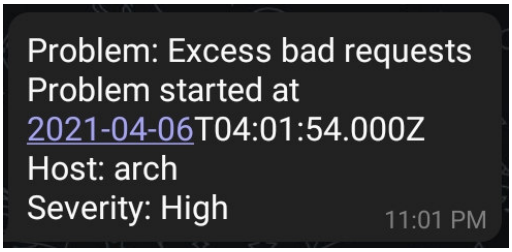


Figura 105. Mensaje de alerta en la aplicación móvil de Telegram

3.5.2. Base de Datos MySQL

3.5.2.1. Zabbix

Existen varios Template creados por la comunidad para distintas bases de datos, en este caso se usará uno para MySQL. Para utilizar este Template se debe realizar una serie de pasos que se explicaran a continuación.

Se debe tener instalado en el host MySQL cliente. Y Tener agregado las utilidades mysql y mysqladmin en las variables de entorno global.

A continuación, se debe crear un archivo "/etc/zabbix/zabbix_agentd.d/template_db_mysql.conf " con el siguiente contenido.

```
# vim template_db_mysql.conf
UserParameter=mysql.ping[*], mysqladmin -h"$1" -P"$2" ping
UserParameter=mysql.get_status_variables[*], mysql -h"$1" -P"$2" -sNX -e "show
global status"
UserParameter=mysql.version[*], mysqladmin -s -h"$1" -P"$2" version
```

```
UserParameter=mysql.db.discovery[*], mysql -h"$1" -P"$2" -sN -e "show databases"
UserParameter=mysql.dbsize[*], mysql -h"$1" -P"$2" -sN -e "SELECT COALESCE(SUM(DATA_LENGTH + INDEX_LENGTH),0) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='$3'"
UserParameter=mysql.replication.discovery[*], mysql -h"$1" -P"$2" -sNX -e "show slave status"
UserParameter=mysql.slave_status[*], mysql -h"$1" -P"$2" -sNX -e "show slave status"
```

Como siguiente paso se debe crear un usuario con los permisos necesarios para acceder a la base de datos y recolectar métricas, como se muestra a continuación.

```
MariaDB [(none)]> CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'password'
MariaDB [(none)]> GRANT REPLICATION CLIENT, PROCESS, SHOW DATABASES, SHOW VIEW ON *.* TO zabbix'@'localhost';
```

Para que Zabbix pueda autenticarse en la base de datos se tiene que crear un archivo ".my.cnf " en el directorio “/var/lib/zabbix”.

```
vim .my.cnf
[client]
user='zabbix'
password='password'
```

Para que las configuraciones se apliquen se debe reiniciar el servicio de zabbix-agent.

```
# systemctl restart zabbix-agent.service
```

Desde la interfaz gráfica de Zabbix hay que ir a configuraciones -> host, seleccionar el host en donde se desea implementar el monitoreo y dar clic en la opción de Template. Dar clic en el botón “Select” y escoger el template de MySQL.

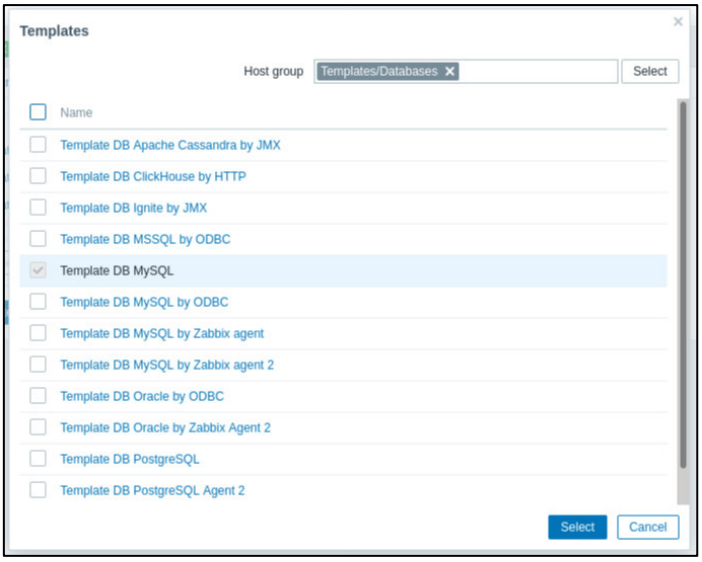


Figura 106. Lista de Templates de Zabbix

Una vez que esté seleccionado el Template hay que actualizar el host con el botón "update". Para comprobar el funcionamiento el Template vamos a ir a "Monitoring -> Lastest data" y seleccionar el Item "Availability: MySQL status, se podrá observar un gráfico del tiempo que la base de datos MySQL ha estado disponible.

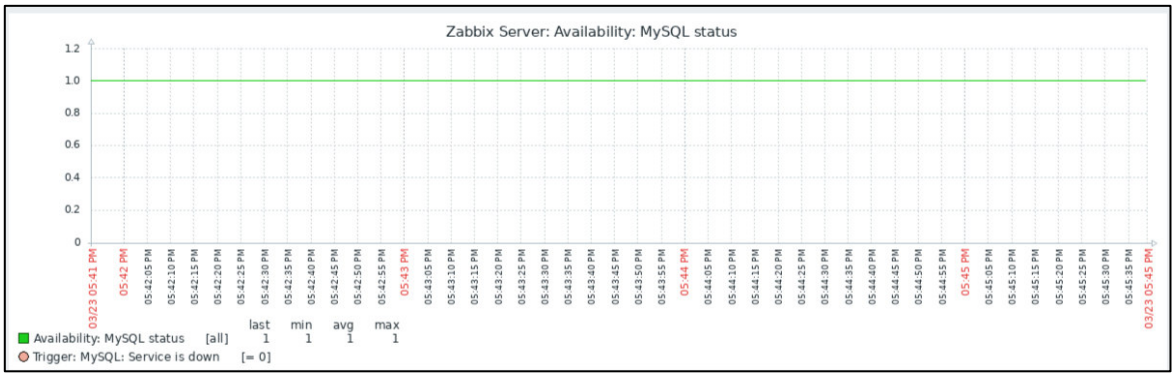


Figura 107. Gráfico de líneas de la disponibilidad de MySQL

3.5.2.2. Prometheus

Se descargo el exportados de datos para MySQL que se puede encontrar en la siguiente dirección https://github.com/prometheus/mysqld_exporter/releases/download/v0.12.1/mysqld_exporter-0.12.1.aix-ppc64.tar.gz.

Antes de recopilar las métricas de MySQL se necesita crear un usuario para acceder a la base de datos.

```
MariaDB [(none)]> CREATE USER 'mysqld_exporter'@'localhost' IDENTIFIED BY
'password' WITH MAX_USER_CONNECTIONS 2;
```

Para limitar la cantidad de conexiones que este usuario va a poder realizar se utilizó el parámetro “MAX_USER_CONNECTIONS” con el valor de dos, esto se realizó para evitar una sobrecarga en el servidor por recopilación de datos en el monitoreo.

El usuario creado debe poseer los permisos necesarios de lectura en la base de datos. Para lo cual se usó el siguiente comando.

```
MariaDB [(none)]> GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO
'mysqld_exporter'@'localhost';
```

El siguiente paso es crear el archivo de configuración que el exportador de mysql va a usar para autenticarse en la base de datos.

```
# vim cnf
[client]
user=mysqld_exporter
password=password
host=localhost
```

Para facilitar la ejecución del exportador se creó un archivo bash con el siguiente contenido. Esto se realizó ya que el comando de ejecución requiere varios parámetros.

```
# vim initExportMySQL.sh
 /home/diego/Documents/Prometheus/mysql/mysqld_exporter \
--config.my-cnf /home/diego/Documents/Prometheus/mysql/cnf \
--collect.global_status \
--collect.info_schema.innodb_metrics \
--collect.auto_increment.columns \
--collect.info_schema.processlist \
--collect.binlog_size \
--collect.info_schema.tablestats \
--collect.global_variables \
--collect.info_schema.query_response_time \
```

```
--collect.info_schema.userstats \
--collect.info_schema.tables \
--collect.perf_schema.tablelocks \
--collect.perf_schema.file_events \
--collect.perf_schema.eventswaits \
--collect.perf_schema.indexiowaits \
--collect.perf_schema.tableiowaits \
--collect.slave_status \
--web.listen-address=0.0.0.0:9104
```

A continuación, se agregará el exportador de MySQL en la configuración de Prometheus.

```
# vim prometheus.yml
 scrape_configs:
  - job_name: 'mysql'
    static_configs:
    - targets: ['localhost:9104']
```

Para poder observar la información de forma más fácil, se va a crear un panel de control en Grafana. Desde el módulo “Create” se utilizó la opción “Import”. Se utilizó un panel de control creado por la comunidad con el id “7362”.

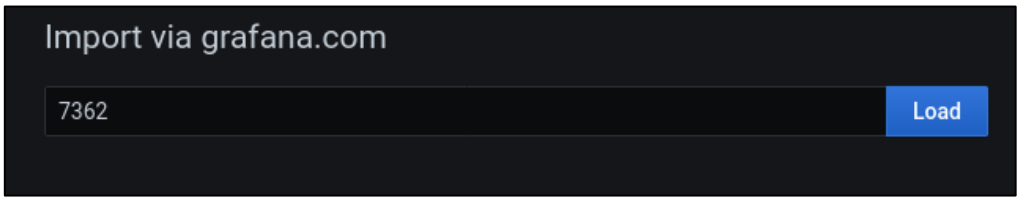


Figura 108. Formulario de importación de panel de control de Grafana

En la interfaz de importación, se escoge la fuente de datos en este caso se eligió Prometheus. Y se importó el panel de control.

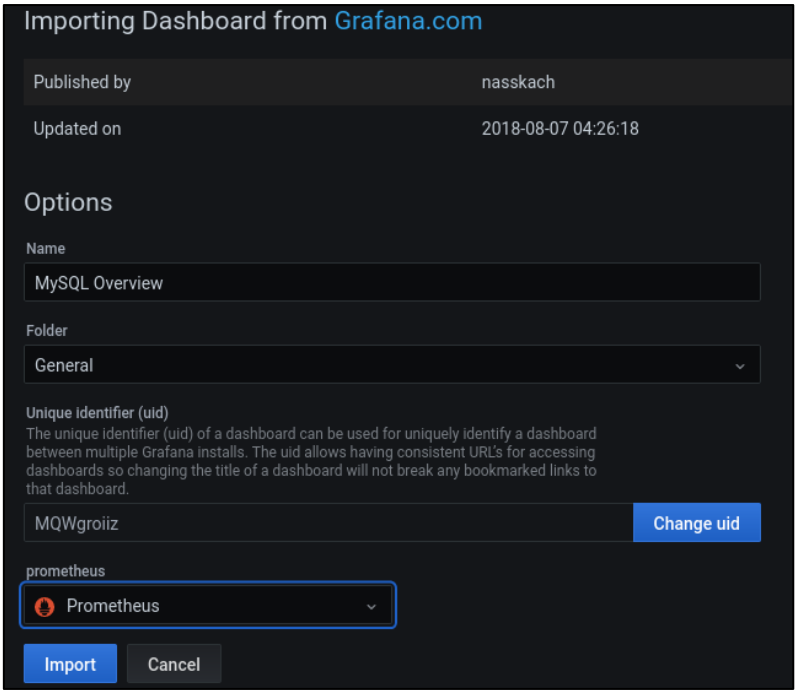


Figura 109. Formulario de configuración de importación de un panel de control en Grafana

Si todas las configuraciones se realizaron de forma correcta se podrá visualizar una gran cantidad de métricas de servidor MySQL. Como el tiempo de actividad, el número de conexiones y las actividades de los hilos del servidor.

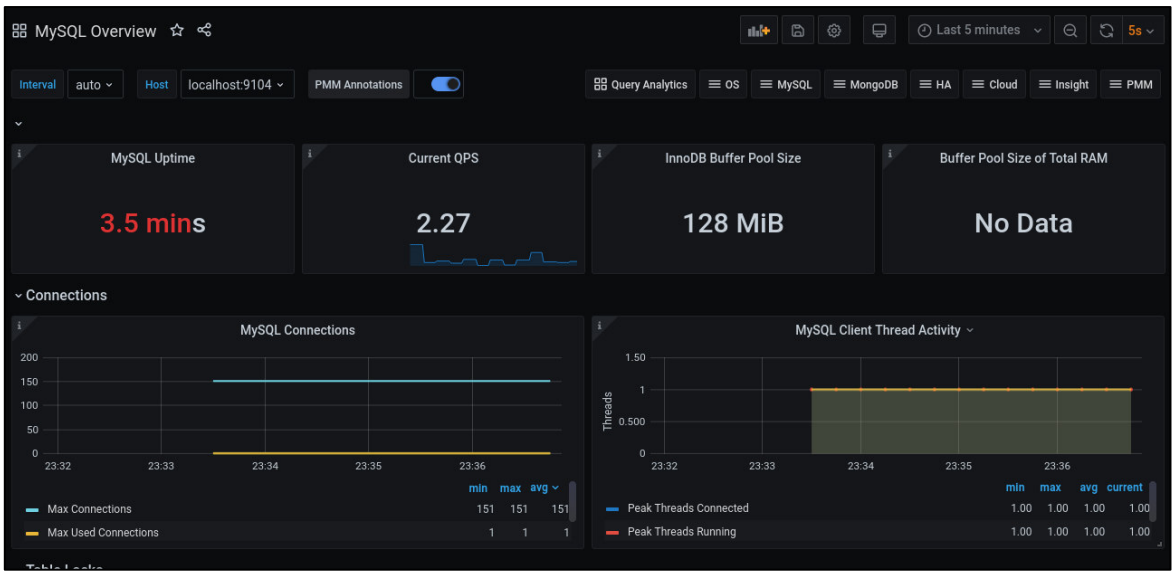


Figura 110. Panel de control de MySQL en Grafana

Se quiere crear una alerta cuando se produzcan intentos de acceso fallidos en la base de datos. Para esto se usará las métricas recopiladas con el export de MySQL, el archivo de

configuraciones “rules.yml” de Prometheus, Alertmanager y el API Rest de comunicación con Telegram.

En el archivo de rules.yml añadió las siguientes líneas que sirven para obtener el promedio de conexiones abortadas que se haya producido en un periodo de un minuto.

```
# vim rules.yml
groups:
  - name: mysql
    rules:
    - alert: Aborted Connections
      expr: rate(mysql_global_status_aborted_connects[1m]) > 0
      for: 60s
```

En el archivo de configuración de Alertmanager se debe incluir la ruta del servicio del API Rest que se va a utilizar.

```
# vim alertmanager.yml
 receivers:
- name: 'telegram'
  webhook_configs:
  - url: 'http://127.0.0.1:4000/api/alert/prometheus'
```

El código del API Rest se encuentra en el siguiente repositorio https://github.com/Dandrezz/Alerts.

Se comprobó el escenario para él envió de alerta y se obtuvo una notificación por medio de Telegram con la siguiente información.

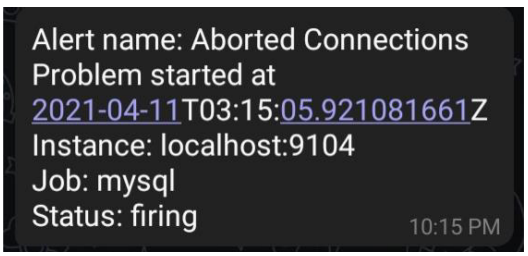


Figura 111. Visualización de alerta en aplicación móvil de Telegram

3.5.2.3. ELK Stack

Un tema crucial en un entorno de base de datos es la optimización de los tiempos de respuesta de las consultas. Para esto primero se tiene que comprender porque el servidor requiere una cierta cantidad de tiempo para responder a una consulta y reducir o eliminar cualquier trabajo innecesario que se esté haciendo para lograr el resultado. Con este fin es necesario medir en que se consume el tiempo.

La carga adicional que significa registrar estas actividades es mínima, en otras palabras, no afecta el rendimiento del servidor de base de datos, pero algo que sí hay que tener en cuenta es el espacio en disco que pueden ocupar los logs.

MySQL cuenta con configuraciones que permiten registrar las consultas que excedan un límite de tiempo. Para esta práctica se configuró el parámetro de tiempo en un segundo. Todas las consultas que tomen más de un segundo se van a registrar en el archivo “log-slow-queries.log”.

En las configuraciones de MySQL se deben agregar las siguientes líneas, en donde se va a especifica el tipo de registro que se va a guardar y su ruta, en estos casos los logs de errores, generales y consultas lentas.

```
#vim /etc/my.cnf
[mysqld]
log_error=/var/log/mysql/mysql_error.log
 general_log_file        = /var/log/mysql/mysql.log
general_log             = 1
[mysqld]
slow-query-log = 1
slow_query_log_file = /var/log/mysql/log-slow-queries.log
long_query_time = 1
```

Para que las configuraciones se implementen se debe reiniciar el servicio de MySQL.

```
# systemctl start mysql.service
```

Como siguiente paso se va a configurar Logstash, para leer el archivo “log-slow-queries.log”. Se implementó toda la lógica de recolección de información en el archivo mysql.conf.

En el plugin del input se especificó que la información se obtendrá del archivo "/var/log/mysql/log-slow-queries.log", ya que los logs que se registran en este archivo tienen un formato de varias líneas se usa el codec "multiline" con los parámetros de "pattern", "negate" y "what".

```
input {
  file {
    path => "/var/log/mysql/log-slow-queries.log"
    codec => multiline{
       pattern => "^# Time:"
      negate => true
      what => previous
}}}
```

En el plugin filter, se usó un filtro grok por cada una de las líneas que componen los registros del archivo, como se puede ver a continuación.

```
filter {
  grok {
    match => { "message" => ["# User@Host:
%{USER:user}\[%{USER:current_user}\] @ %{HOSTNAME:query_host}
\[%{IPORHOST:ip}?\]#
Rows_affected:%{SPACE}%{NUMBER:rows_affected:int}%{SPACE}Bytes_sent:%
{SPACE}%{NUMBER:bytes_sent}%{GREEDYDATA}"]}
  }
  grok {
    match => { "message" => ["# Thread_id:
%{NUMBER:id_thread}%{SPACE}Schema:%{SPACE}QC_hit:%{SPACE}%{USER
:qc_hit}"]}
}
grok {
    match => { "message" => ["# Query_time:
%{NUMBER:query_time:float}%{SPACE}Lock_time:%{SPACE}%{NUMBER:lock_t
```

```
ime}%{SPACE}Rows_sent:%{SPACE}%{NUMBER:rows_sent:int}%{SPACE}Rows
_examined:%{SPACE}%{NUMBER:rows_examined:int}"]}
}
  grok {
    match => { "message" =>
["SET%{SPACE}timestamp=%{NUMBER:timestamp};\n(?<query>(?<action>\w+)\s
+.*)"]}
}}
```

Y por último en el plugin output, se especifica el destino de los datos, en este caso Elasticsearch y el nombre del índice con el que se referencian los datos.

```
output {
elasticsearch {
hosts => ["127.0.0.1:9200"]
index => "mysql-logs-%{+YYYY.MM.dd}"}}
```

Para ejecutar Logstash con su archivo de configuración se usa el siguiente comando.

```
# ./logstash -f mysql.conf
```

Desde Kibana nos dirigimos al módulo “Stack Management -> Index patterns”, y se utilizó la opción de “Create index pattern”. En la primera interfaz de creación se ingresa el patrón que se va a usar para este índice, en este caso “mysql-*”

Figura 112. Formulario de definición de índice de patrón

En la siguiente interfaz de creación se escoge el campo que se va a utilizar como referencia de tiempo.

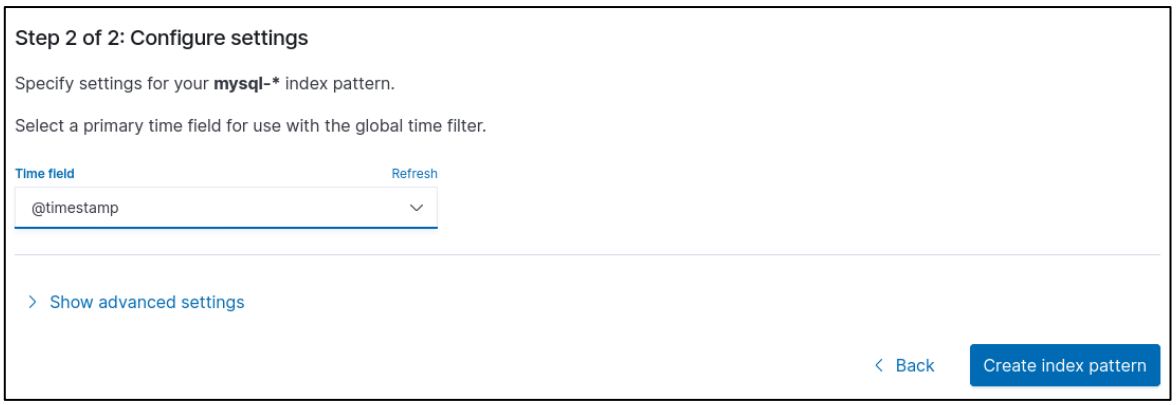


Figura 113. Formulario de configuración de índice de patrón

Desde la pantalla de “Discover” se pueden observar los logs registrados en Elasticsearch.

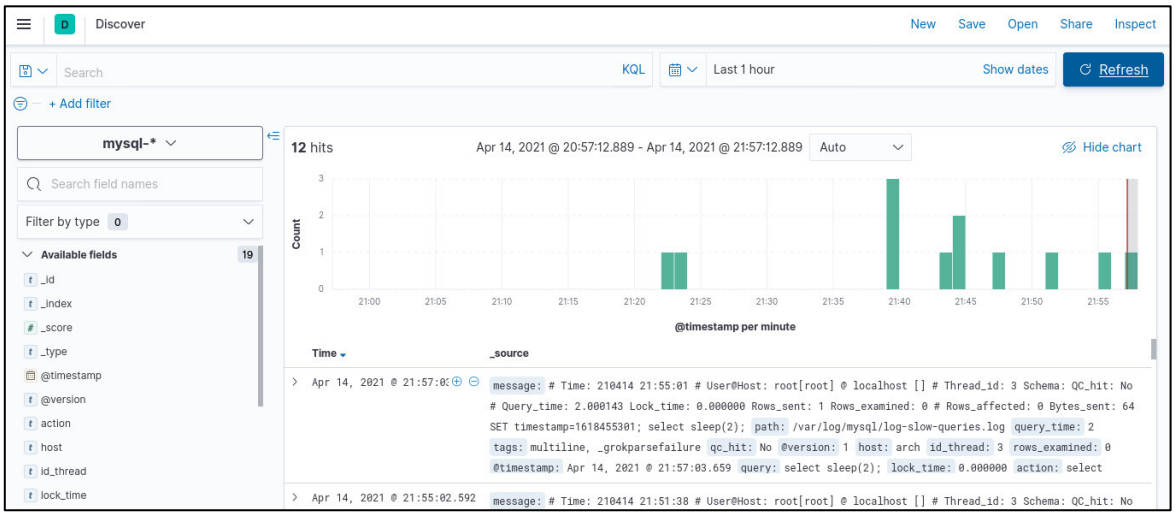


Figura 114. Pantalla "Discover" Kibana

Por medio del lenguaje KQL (Keyword Query Language) se puede implementar filtros de esta información.

Desde el módulo “Visualize” se puede crear distintos gráficos para poder visualizar los datos de una forma más ordenada y sencilla, la siguiente imagen se puede ver una tabla en donde se puede ver la información de las consultas que más tiempo le ha demorado en realizar a la base de datos, se tiene una columna para contar la cantidad de veces que este registro se repite y otra columna para ver la cantidad de columnas que se envían.

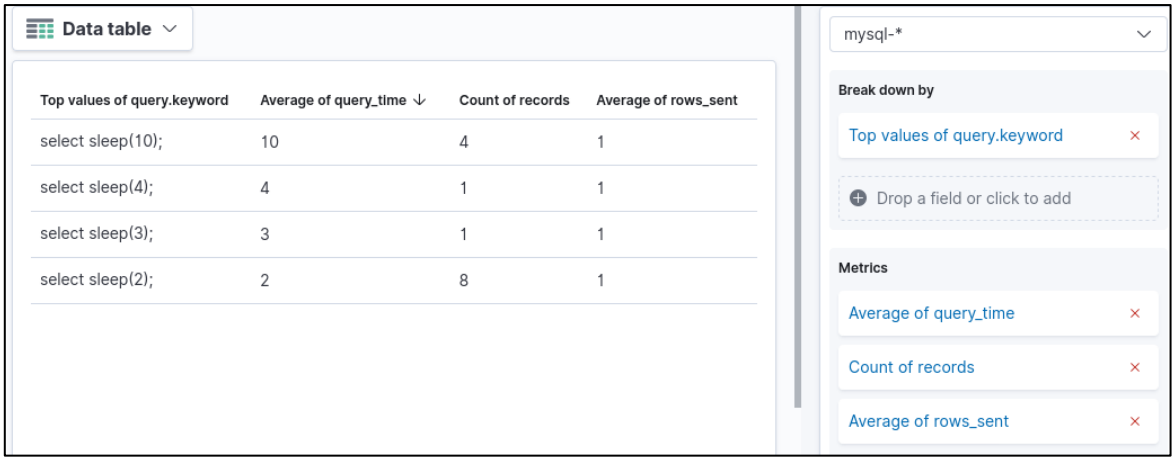


Figura 115. Gráfico de una tabla de las sentencias del archivo log-slow-queries.log

En el siguiente gráfico se puede apreciar que el 100% de los registros del archivo “log-slow-queries.log” son de instrucciones “SELECT”.

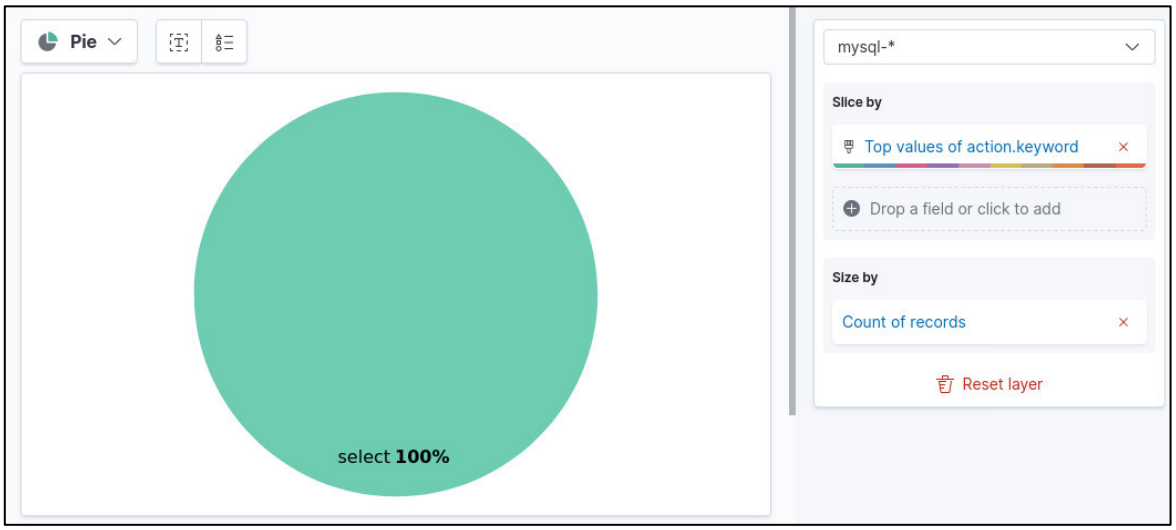


Figura 116. Gráfico de pastel del tipo de sentencia en el archivo log-slow-queries.log

Estos solo son unos pocos ejemplos de algunos gráficos que se pueden crear con los datos de los logs de MySQL.

Un evento no deseado en una base de datos es un “deadlock”, el cual sucede cuando varios procesos están bloqueados porque cada proceso tiene un recurso y espera otro recurso adquirido por algún otro proceso.

En el caso de que un “deadlock” se produzca es de vital importancia que el administrador de la base de datos se entere y lo corrija. En caso de ser necesario tomar futuras medidas para que no se vuelvan a producir.

Para esto se usará la herramienta Logstash, y se creará una condición, en donde si uno de los registros tiene un valor mayor a uno en su campo de “lock_time” se enviará una alerta a los integrantes del equipo responsable del mantenimiento de la base de datos.

Dentro del plugin output se utilizó un plugin http para la comunicación con el API Rest de notificaciones.

```
output {
 if [lock_time] > 0 {
http{
  http_method => "post"
  url => "http://localhost:5000/api/alert/elk/deadlock"
  content_type => "application/json"
}}
 elasticsearch {
hosts => ["127.0.0.1:9200"]
index => "mysql-logs-%{+YYYY.MM.dd}"
}}
```

Y listo, en caso de que se registre un “deadlock” en la base de datos se enviará una notificación por medio de telegrama a los administradores de la base de datos.

Algunas formas de optimizar los tiempos de respuesta son: un buen diseño de la base de datos, un computador debidamente equipado, estructuras lógicas y un computador bien equipado.

3.5.3. Servidor de Correo Postfix

3.5.3.1. Zabbix

En el URL https://github.com/rafael747/zabbix-postfix se encuentra el repositorio con los archivos necesarios para configurar el monitoreo de Postfix desde Zabbix.

Como primer paso se debe instalar cuatro paquetes de software: pflogsumm, bc, zabbix-agent, zabbix-sender. La instalación se realizó con el siguiente comando.

```
# apt-get install pflogsumm bc zabbix-agent zabbix-sender
```

A continuación, se debe clonar el repositorio de git del proyecto zabbix-postfix, existen varios proyectos dedicados a monitorear Postfix, se eligió un proyecto con una buena cantidad de revisiones positivas.

```
# git clone https://github.com/rafael747/zabbix-postfix.git
```

Dentro del archivo del proyecto, se deben cambiar los permisos de ejecución de los archivos “zabbix_postfix.sh” y “pygtail.py”. Y copiarlos en el directorio “/usr/sbin”

```
# chmod +x pygtail.py
# chmod +x zabbix_postfix.sh
# zabbix_postfix.sh /usr/local/sbin/
# pygtail.py /usr/local/sbin/
```

Para que se puedan ejecutar estos binarios de forma automática, se debe copiar el archivo zabbix_postfix al directorio “/etc/sudoers.d/” y cambiar los permisos a solo lectura.

```
# cp zabbix_postfix /etc/sudoers.d/
# chmod 440 /etc/sudoers.d/zabbix_postfix
```

El siguiente paso es copiar el archivo zabbix_postfix.conf a los archivos de configuración del agente Zabbix.

```
# cp zabbix_postfix.conf /etc/zabbix/zabbix_agentd.conf.d/
```

Para que los cambios se apliquen se debe reiniciar el servicio del agente Zabbix.

```
# service zabbix-agent restart
```

Por medio de la utilidad “cron” se va a ejecutar el script “zabbix_postfix.sh” de forma automática cada cinco minutos.

```
# crontab –e
*/5 * * * * /usr/local/sbin/zabbix_postfix.sh
```

A continuación, hay que ir a la interfaz gráfica de Zabbix e importar el template “template_postfix.xml”. Para esto nos dirigimos al módulo "Configuration -> Templates"

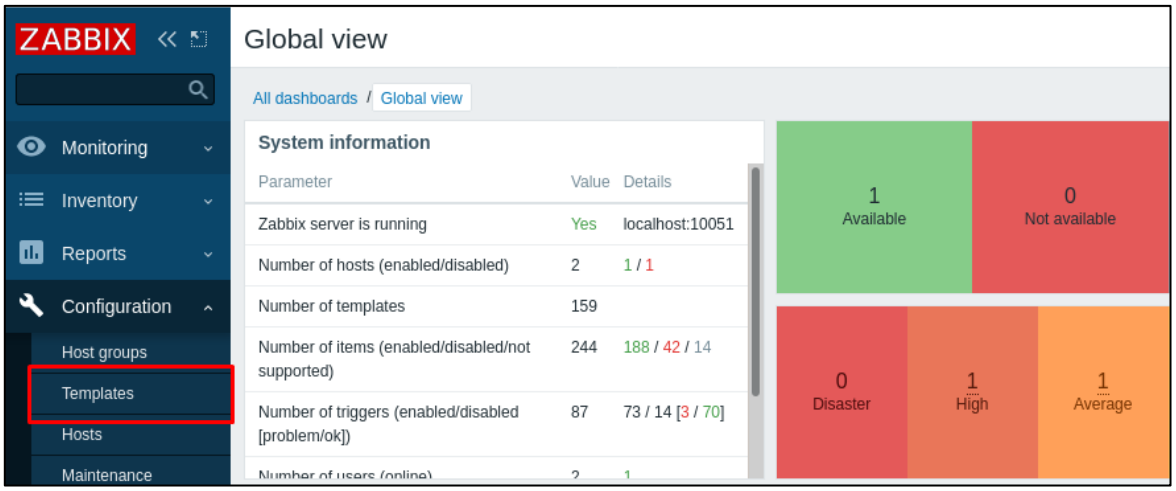


Figura 117. Módulo Tamplates en barra lateral Zabbix

Se utiliza la opción importar en la esquina superior derecha y se selecciona el archivo que se va a usar de template, y se lo importa.

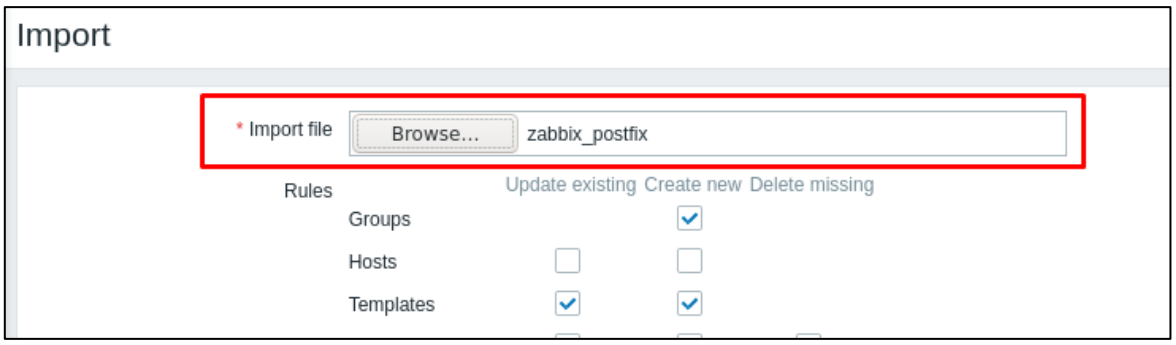


Figura 118. Formulario de importación de Template

Una vez importado el template se lo asigna al host que se va a monitorear en este caso a "Zabbix Server".

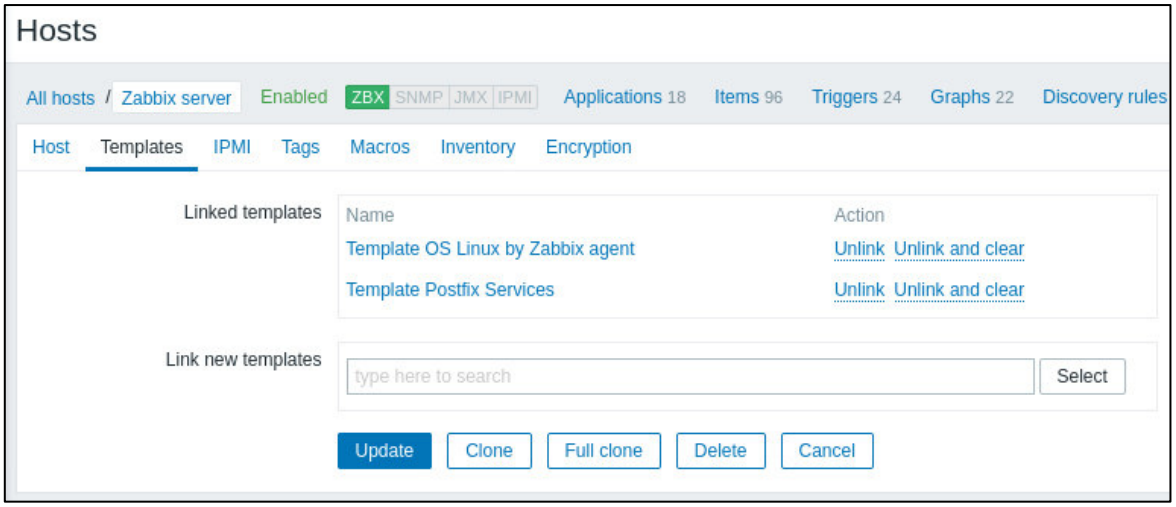


Figura 119. Lista de Templates configurados para un host

Zabbix comenzará a monitorear varias métricas del servidor de correo Postfix, esto se puede verificar en las gráficas en donde se puede observar esta información.

3.5.3.2. Prometheus

Para poder obtener las métricas de Postfix se usará el postfix_exporter, el cual se encuentra en la siguiente URL https://github.com/kumina/postfix_exporter.

Como primer paso se clonó el repositorio del proyecto de postfix_export.

```
# git clone https://github.com/kumina/postfix_exporter.git
```

Dentro del archivo del postfix_export se debe generar el binario ejecutable del exportador, esto se lo realiza con el comando “go build” con la flag “-tags nosystemd” para que no sea requerido los encabezados de systemd.

```
# go build -tags nosystemd
```

Una vez que termine la compilación se tendrá a disposición un archivo con el nombre de “postfix_export”, el exportador obtiene los logs del archivo “/var/log/maillog”, esta dirección es correcta en distribuciones como RHEL pero no para distribuciones Debian, por lo que hay que especificar la ruta en donde se encuentra.

```
# ./postfix_exporter --postfix.logfile_path="/var/log/mail.log"
```

En el archivo de configuración de Prometheus se debe especificar el puerto en donde se encuentran expuestas las métricas de postfix_export.

```
# vim prometheus.yml
scrape_configs:
  - job_name: 'Postfix'
    static_configs:
    - targets: ['localhost:9154']
```

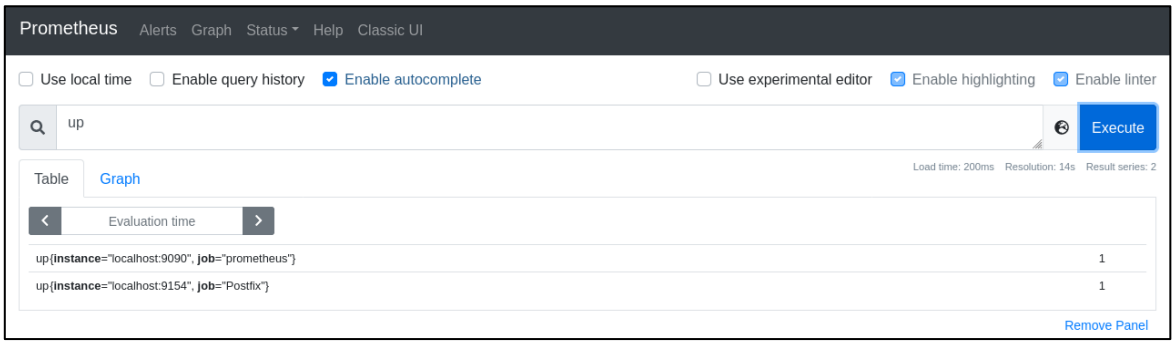


Figura 120. Ejecución de comando "up" en Prometheus

En la siguiente imagen se puede ver que la instancia Postfix se encuentra activa y se está recopilando datos de esta instancia.

A continuación, se va a implementar un panel de control por medio de Grafana para poder observar los indicadores de la instancia Postfix.

En la interfaz de Grafana nos dirigimos al módulo “Create → Import” y se carga el id 10013 el cual es un panel de control hecho por la comunidad.

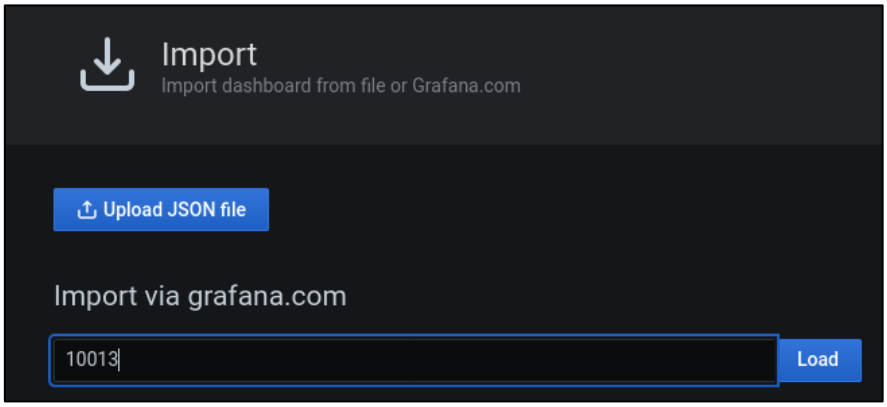


Figura 121. Formulario de importación de paneles de control en Grafana

En la siguiente interfaz se debe seleccionar la fuente de los datos en este caso Prometheus. Se selecciona e importa el panel de control.

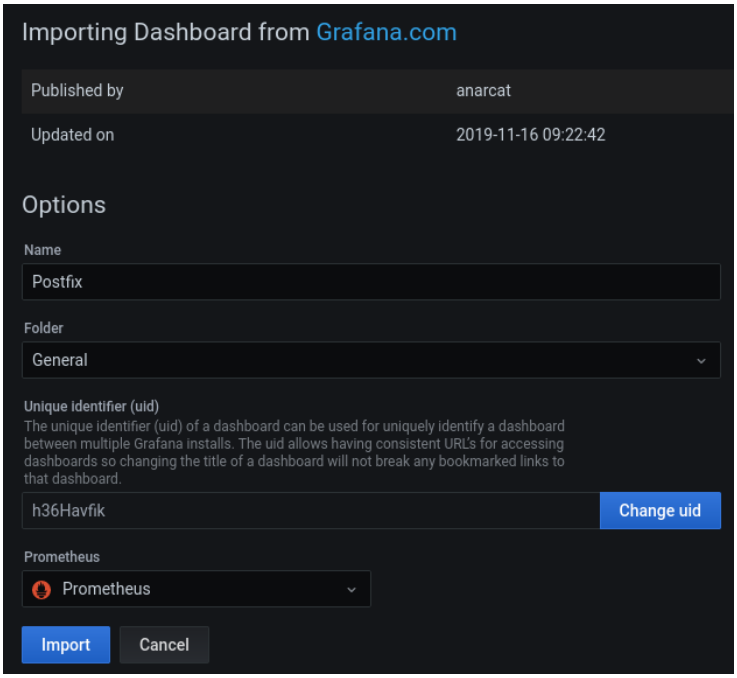


Figura 122. Formulario de importación de paneles de control en Grafana

Una vez importado ya se puede observar las distintas métricas con las que está diseñado el panel de control.

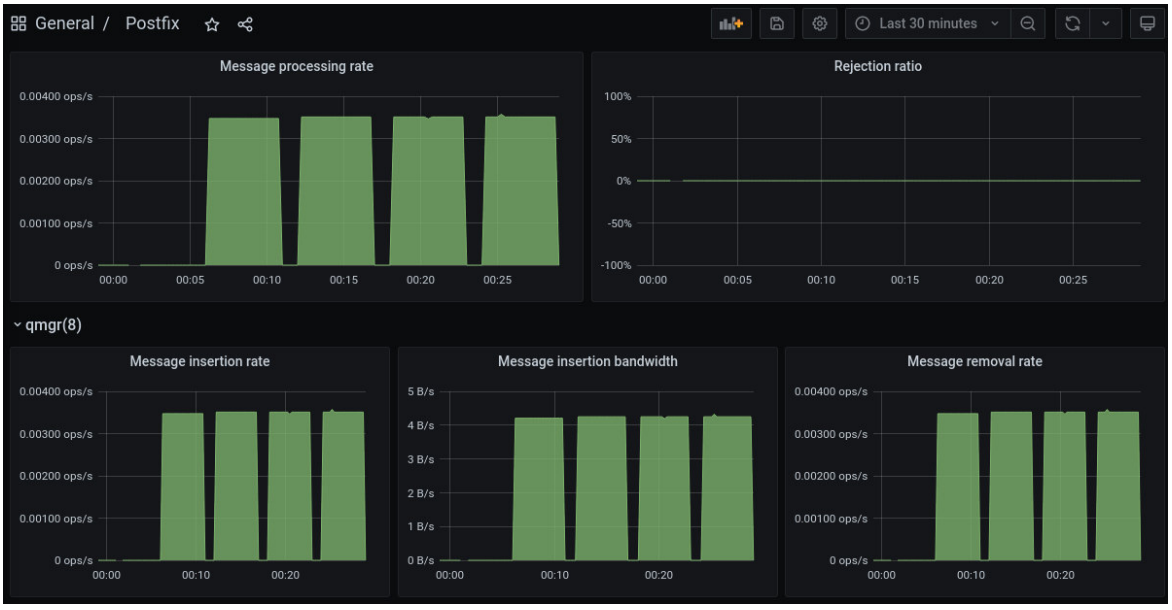


Figura 123. Panel de control de Postfix en Grafana

3.5.3.3. ELK Stack

Para poder obtener los datos que se almacenan en los logs de “/var/log/mail.log” por medio del ELK Stack, se realizaron los siguientes pasos.

Se creó un archivo de configuración de Logstash con los plugins de input, filter, grok y output. En el plugin de input se especificó la fuente de donde se obtendrá los datos en este caso “/var/log/mail.log”.

```
input {
    file {
        type => "postfix"
        path => "/var/log/mail.log"
        start_position => "beginning"
}}
```

En el plugin filter se realizó una discriminación de los logs por una palabra clave. Ya que la estructura de los logs que provienen de Postfix son diferentes a los de Dovecot. Dentro de cada condición se especifica los patrones que se van a usar. Debido a la complejidad de los que se tuvo que usar dos archivos externos de patrones grok, uno para Postfix y otro para Dovecot.

```
filter {
    if "postfix" in [message]{
        grok {
            patterns_dir => [ "./grok-postfix" ]
            match => { "message" =>
["%{SYSLOGTIMESTAMP:timestamp}%{SPACE}%{HOSTNAME:query_host}%{S
PACE}postfix/%{WORD:sufijos}\[%{POSINT:pid}\]:%{SPACE}%{POSTFIXMIO}"]}
 } }
    if "dovecot" in [message]{
        grok{
            patterns_dir => [ "./grok-dovecot" ]
```

```
        match => { "message" =>
["%{SYSLOGTIMESTAMP:timestamp}%{SPACE}%{HOSTNAME:query_host}%{S
PACE}dovecot:%{SPACE}%{DOVECOT}"]}
}}}
```

En el plugin de output se especificó el destino de los datos tratados, ya que el servidor de correos se encuentra en otro host, se escribió la IP del host donde se está ejecutando el servicio de Elasticsearch.

```
output{
   elasticsearch{
   hosts => ["192.168.1.4:9200"]
   index => "postfix-%{+YYYY.MM.dd}"
   }
   stdout {}
}
```

Ya que Logsatsh se encuentra en un servidor externo, se debe configurar que Elasticsearch sea visible desde servidores externos. Para esto se modificó el archivo “elasticsearch.yml” y se ingresó las siguientes líneas, esta configuración no se recomienda en entornos de producción, ya que se permite el acceso de cualquier host.

```
# vim elasticsearch.yml
network.host: 0.0.0.0
transport.tcp.port: 9300
discovery.type: single-node
discovery.seed_hosts: ["0.0.0.0", "[::1]"]
network.bind_host: 0.0.0.0
```

Una vez realizado el cambio y levantado el servicio de Elasticsearch se estarán registrando los datos. Para poder visualizar los datos con distintos tipos de gráficos se utilizará Kibana.

Desde Kibana se entró al módulo “Stack Management -> Index Patterns” y se usó la opción “Create index pattern”. En la interfaz de “Create Index Pattern” se ingresa el texto se usará para crear el índice de estos datos.

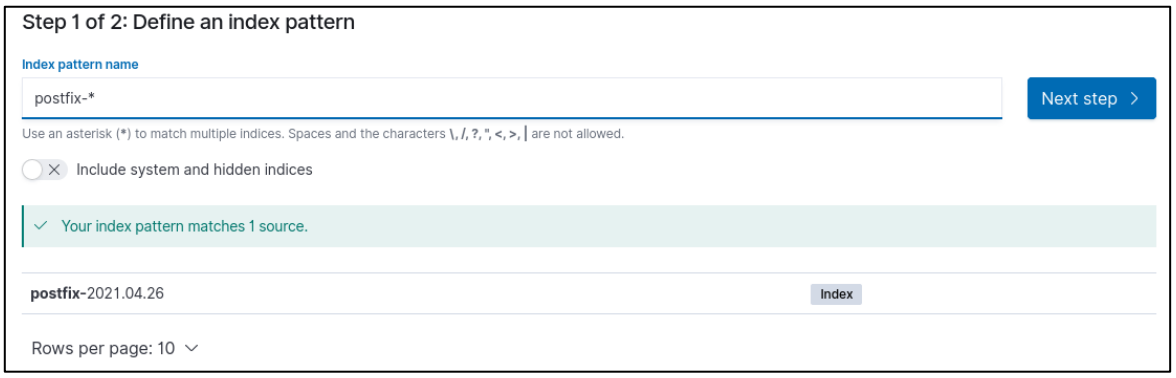


Figura 124. Formulario de definición de índice de patrón

Se elige el campo que se usará como referencia del tiempo y se crea el patrón de índice.

Figura 125. Formulario de configuración de índice de patrón

En el módulo “Discover” se puede observar los datos que provienen de los logs de “mail.log”

Figura 126. Pantalla "Discover" en Kibana

En la siguiente imagen se creó una tabla en donde se puede observar la cantidad de correos en los que un usuario aparece como emisor.

Figura 127. Gráfico de tabla de la cantidad de correos enviados por un emisor

3.6. Implementación en un Entorno Real

Se implementó las tres herramientas seleccionadas en un entorno real de una empresa de desarrollo de software. Por motivos de seguridad no se dará muchos detalles de la infraestructura.

La empresa se dedica al desarrollo de software financiero, posee una gran cantidad de servicios e información. El buen funcionamiento de estos servidores es de gran importancia ya que se brinda estos servicios a varias cooperativas a nivel nacional,

además se usa estos equipos para alojar aplicaciones en estado de pruebas, un posible mal funcionamiento de estos equipos puede significar un gran coste para la empresa y un gran percance en las actividades de los desarrolladores. Por lo que es muy recomendado la implementación de un sistema de monitoreo como se va a mostrar a continuación.

Para la implementación de las consolas de monitoreo se tuvo acceso a dos servidores de producción con las siguientes características.

Especificaciones técnicas del Servidor 1	
Sistema Operativo	Windows Server 2016
RAM	24 GB DDR3 800 Hz
Procesador	Intel Core i7-2600
Disco Duro	932 GB
Tipo de Sistema	Sistema Operativo de 64 bits

Tabla 12. Especificaciones técnicas del Servidor 1

Especificaciones técnicas del Servidor 2	
Sistema Operativo	Windows Server 2016
RAM	32 GB DDR4 1333 Hz
Procesador	Intel Core i7-8700

Disco Duro	1.5 TB
Tipo de Sistema	Sistema Operativo de 64 bits

Tabla 13. Especificaciones técnicas del Servidor 2

3.6.1. Zabbix

Una vez realizado todo el proceso de instalación y configuración del servidor Zabbix y el agente Zabbix en los dos servidores. Se comenzó a recolectar los datos de los servidores por varios días, en la siguiente imagen se puede ver el panel de control de "Global view" en la página inicial de Zabbix, se puede observar varios datos del sistema y en la parte inferior todos los problemas registrados.

Figura 128. Página de inicio de Zabbix

En los dos hosts monitoreados se implementó una recolección de métricas para los recursos de un servidor Windows y una base de datos MySQL.

Figura 129. Configuración de Hosts

3.6.2. Prometheus y ELK Stack

Después de realizar la instalación y configuración de Prometheus y sus exportadores, se procedió a recopilar datos. Se utilizo windows_exporter para las métricas del sistema operativo y mysqld_exporter para las métricas de las instancias MySQL.

En la página de inicio de Prometheus se ejecutó el comando “up” para poder ver el estado de los componentes que se están monitoreando.

Figura 130. Ejecución del comando "up" en Prometheus

En el siguiente panel de control se puede ver varias métricas del sistema operativo del servidor 1. Útil para dar un vistazo al estado actual del servidor.

Figura 131. Panel de control del Servidor 1 en Grafana

Igualmente se creó un panel de control con la misma información para el servidor 2.

Figura 132. Panel de control del Servidor 2 en Grafana

Para las bases de datos MySQL de los dos servidores se realizó el mismo proceso, se configuró su exportador y se creó un panel de control con los datos recopilados.

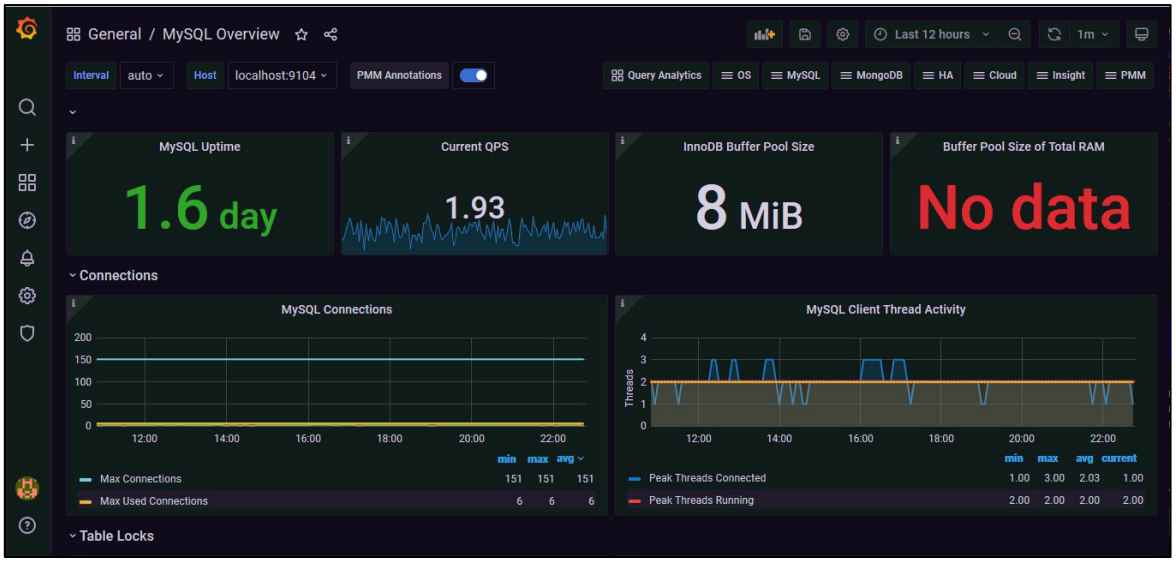


Figura 133. Panel de control de base de datos MySQL del Servidor 1

Para el servidor 2 se realizó el mismo procedimiento. Con esto se puede verificar que se está obteniendo datos tanto del sistema operativo como de los servicios.

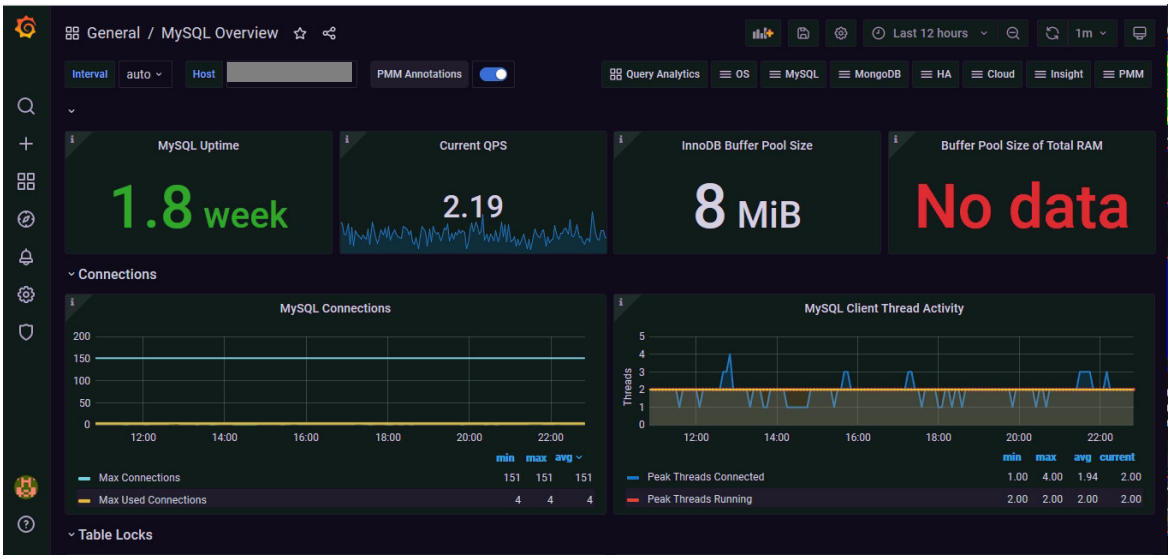


Figura 134. Panel de control de base de datos MySQL del Servidor 2

3.6.2.1. Alertas

Para las alertas se instaló y configuró Alertmaneger y el API Rest de comunicación en el servidor 1, se usó un archivo de reglas con el siguiente contenido.

```
groups:
```

```
  - name: disponibilidad
    rules:
    - alert: InstanceDown
      expr: up == 0
      for: 2m
  - name: SO
    rules:
    - alert: HostHighCpuLoad
      expr: 100 - (avg by (instance)
(rate(windows_cpu_time_total{mode="idle"}[1m])*100)) > 80
      for: 2m
      labels:
        severity: warning
      annotations:
        summary: Host high CPU load (instance {{ $labels.instance }})
        description: "CPU load is > 80%\n  VALUE = {{ $value }}\n  LABELS = {{
$labels }}"
    - alert: MemoryUsage
      expr: 100 - ((windows_os_physical_memory_free_bytes /
windows_cs_physical_memory_bytes) * 100) > 90
      for: 2m
      labels:
        severity: warning
      annotations:
        summary: Windows memory Usage (instance {{ $labels.instance }})
        description: "Memory usage is more than 90%\n  VALUE = {{ $value }}\n
LABELS = {{ $labels }}"
    - alert: DiskSpaceUsage
      expr: 100.0 - 100 * ((windows_logical_disk_free_bytes / 1024 / 1024 ) /
(windows_logical_disk_size_bytes / 1024 / 1024)) > 80
      for: 2m
      labels:
        severity: critical
```

```
    annotations:
      summary: Windows disk Space Usage (instance {{ $labels.instance }})
      description: "Disk usage is more than 80%\n  VALUE = {{ $value }}\n
LABELS = {{ $labels }}"
```

En el módulo de alertas de Prometheus se puede observar el actual estado de estos recursos.

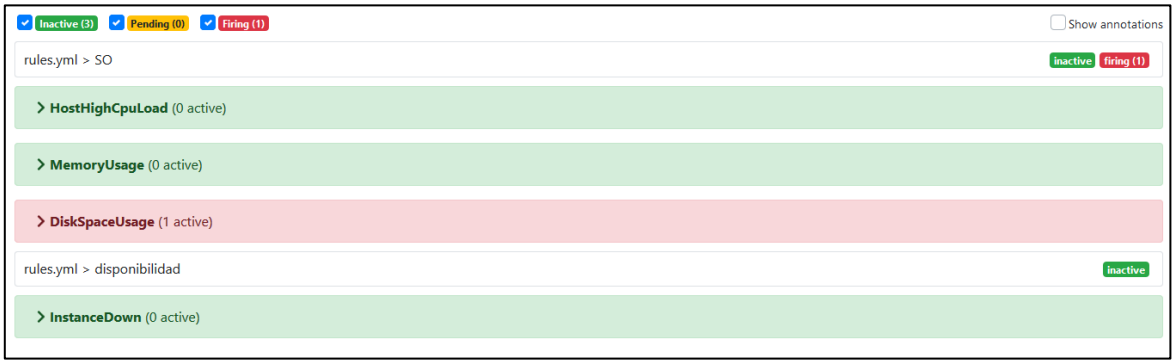


Figura 135. Módulo alertas Prometheus

3.6.2.1. Logs

Después de realizar la instalación y configuración del ELK Stack se procedió a recopilar datos, en este caso se utilizó los logs de la base de datos MySQL, se utilizó los archivos de error “error.log” y de sentencias lentas “mysql-slow.log”. Además, se utilizó un plugin llamado “winlogbeat” para recopilar los logs del sistema operativo de los dos servidores.

En la siguiente captura de pantalla se puede apreciar que se está registando las diferentes entradas que se encuentran en los logs del servidor.

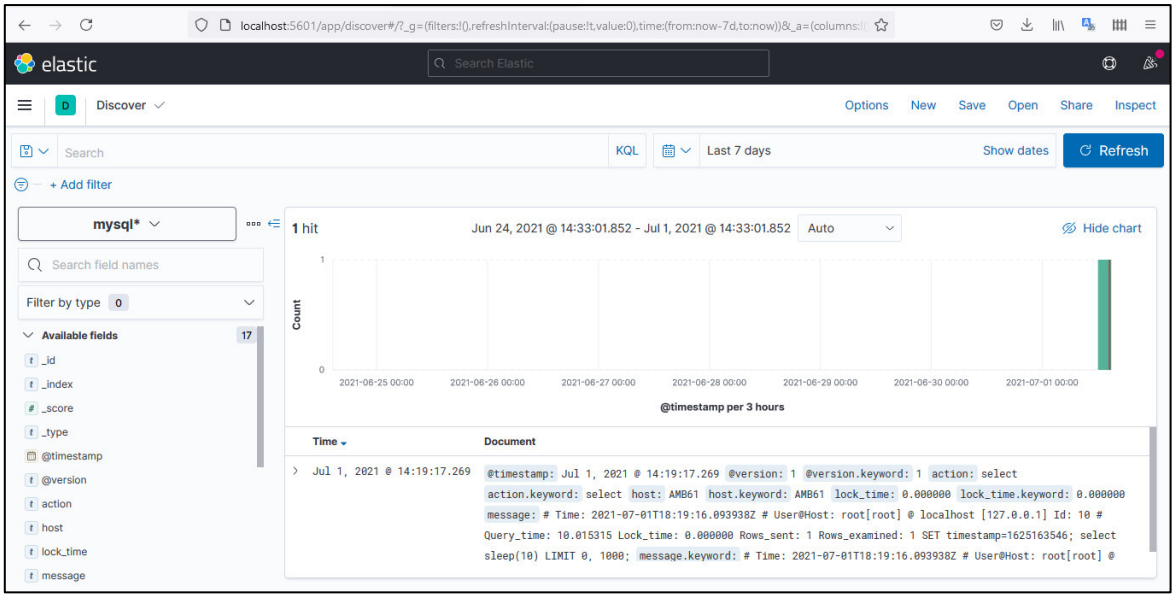


Figura 136. Módulo discovery de Kibana

Winlogbeat es una agente que recolecta varias métricas del sistema operativo Windows. Con los datos que ha recolectado este agente se creó un panel de control para poder ver de forma más ágil algunos de estos datos.

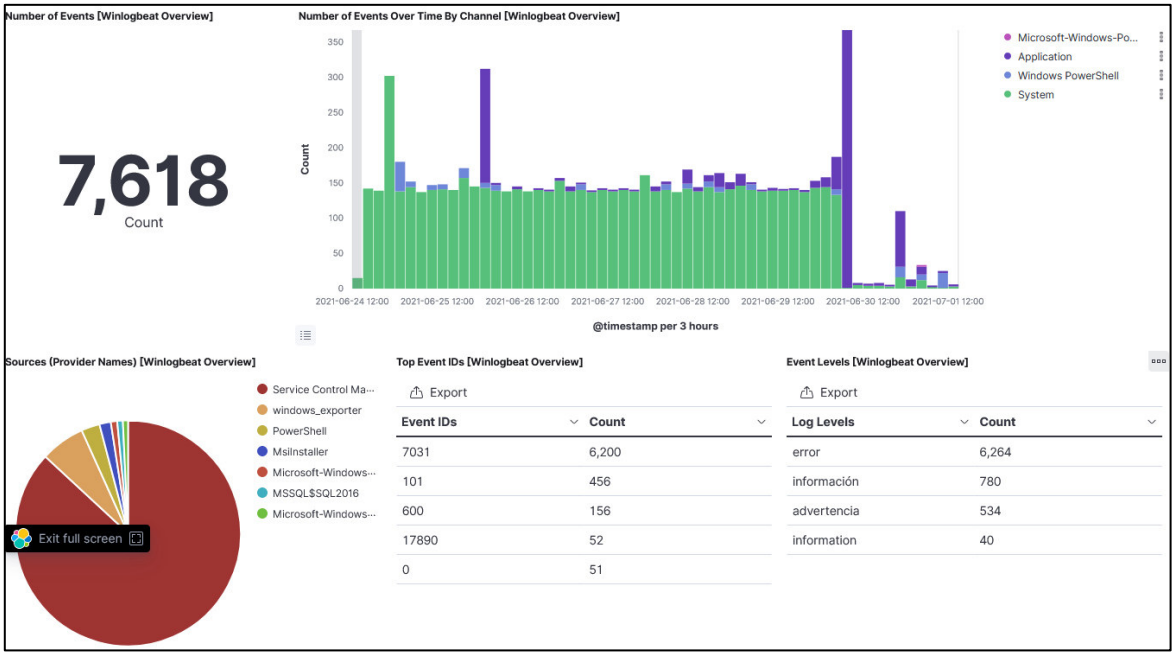


Figura 137. Panel de control de winlogbeat en Kibana

Ya que el archivo mysql-slow.log se encontraba vacío en los dos servidores donde se instaló el Stack ELK se tuvo que ejecutar alguna sentencia de prueba para verificar

que se estén registrando en el sistema. Como se puede ver en la siguiente captura de pantalla estos datos se guardan correctamente en la consola.

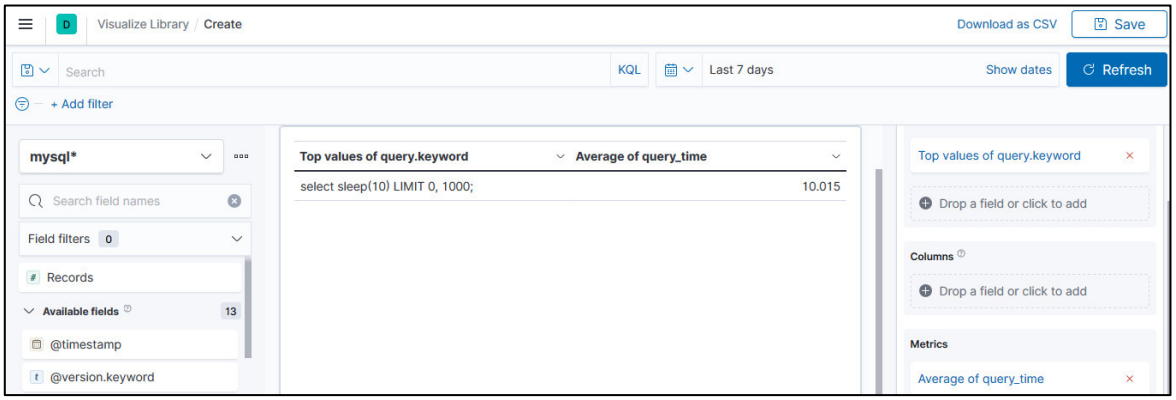


Figura 138. Tabla de las sentencias ordenadas por tiempo

CAPÍTULO 4. RESULTADO Y DISCUSIÓN

Una vez terminado el proceso de instalación y configuración de las consolas de monitoreo se procede a realizar pruebas de funcionalidad en el ambiente simulado y en el real. Esta sección tiene como objetivo determinar que las consolas implementadas cumplan con todas las funcionalidades necesarias.

4.1. Ambiente Simulado

Las métricas y logs que se van a analizar son las siguientes

Métricas y Logs	
Disponibilidad	Capacidad de un sistema en funcionar continuamente
Servidor HTTP Apache	• Peticiones por segundo • Tiempo de procesamiento de solicitud • Trabajadores ocupados • Tabla de agentes

	• Métodos más consumidos
MySQL	• Número de conexiones abortadas • Número de sentencias “Select” por segundo • Sentencias de archivo log-slow-queries.log
Servidor Postfix	• Tabla de la cantidad de correos enviados por un emisor

4.1.1. Servidor HTTP Apaches

A continuación, se mostrará el resultado de registrar métricas con las herramientas Zabbix, Prometheus y Logs de un servidor HTTP Apache.

Peticiones por segundo es un importante indicador del rendimiento y el nivel de tráfico del servidor. En las figuras 139 y 140 se puede apreciar que el servidor se encuentra por debajo de las 150 peticiones por segundo, por lo que el servidor no requiere modificaciones para seguir funcionando correctamente.

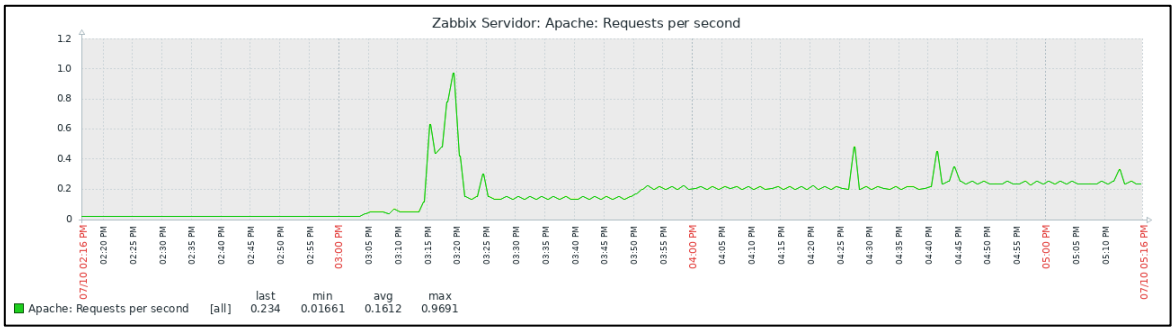


Figura 139. Peticiones por segundo (Zabbix)

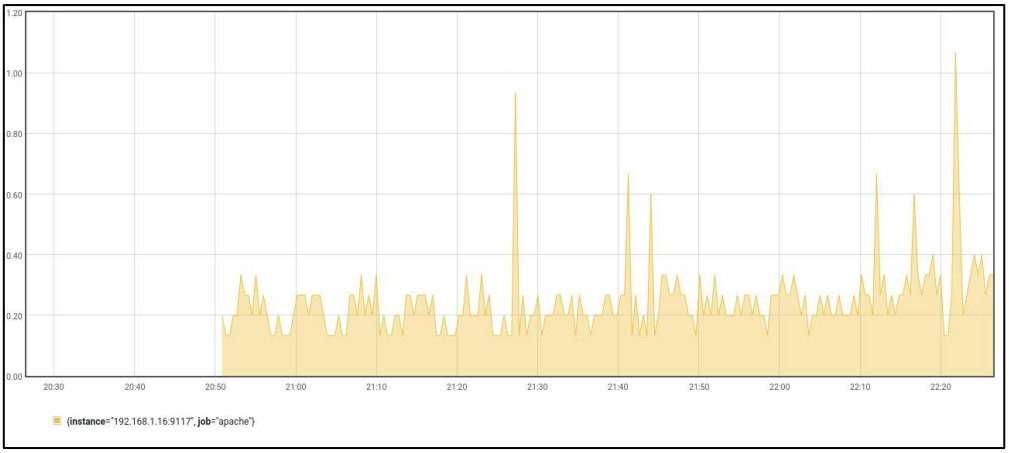


Figura 140. Peticiones por segundo (Prometheus)

La consecuencia de un sitio web lento puede ser perjudicial para las empresas. El tiempo de carga de su sitio web afecta significativamente la experiencia del usuario. Se recomienda que este valor no supere los 200 milisegundos. En la figura 141 y 142 se puede apreciar que no se ha registrado tiempo de respuesta superiores a 2 ms.

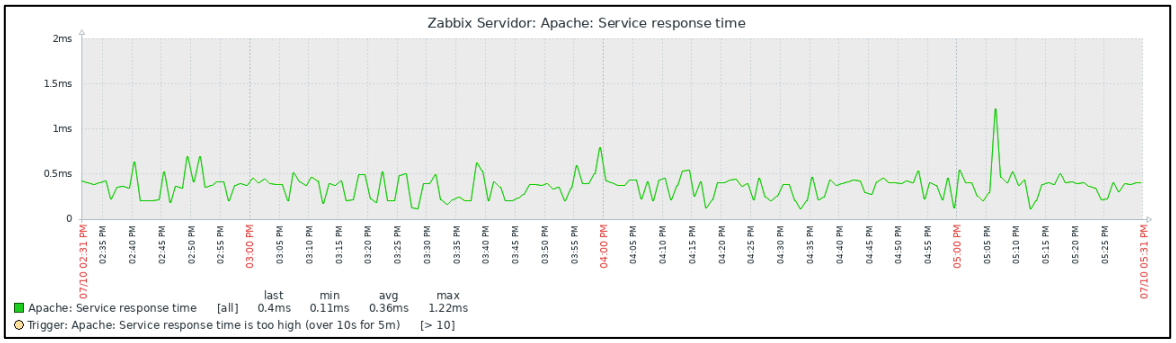


Figura 141. Tiempo de respuesta (Zabbix)

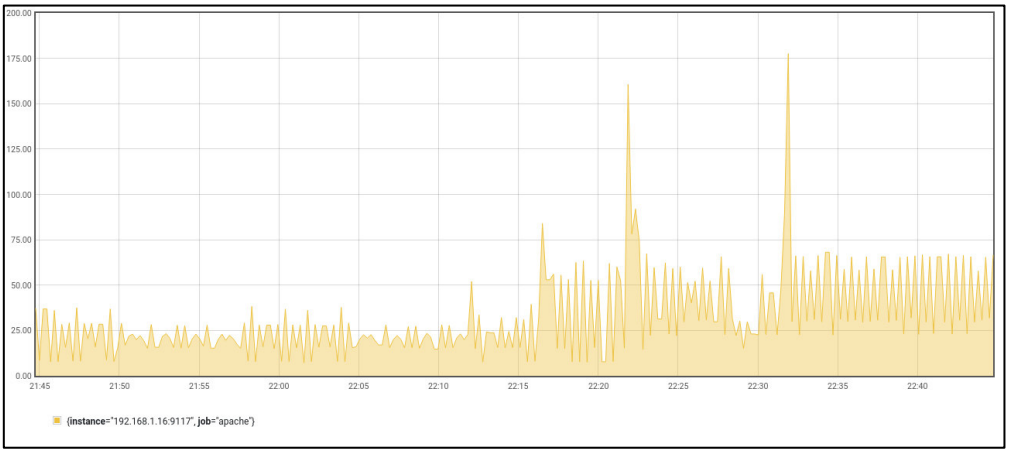


Figura 142. Tiempo de respuesta (Prometheus)

Para no consumir innecesariamente recursos del servidor se debe tener un control de la cantidad de workers que son necesarios. Por otro lado, una cantidad baja de workers puede provocar que la respuesta de las solicitudes tome mucho tiempo. En las figuras 143 y 144 se puede ver que los Workers que se utilizan está entre 1 y 4.

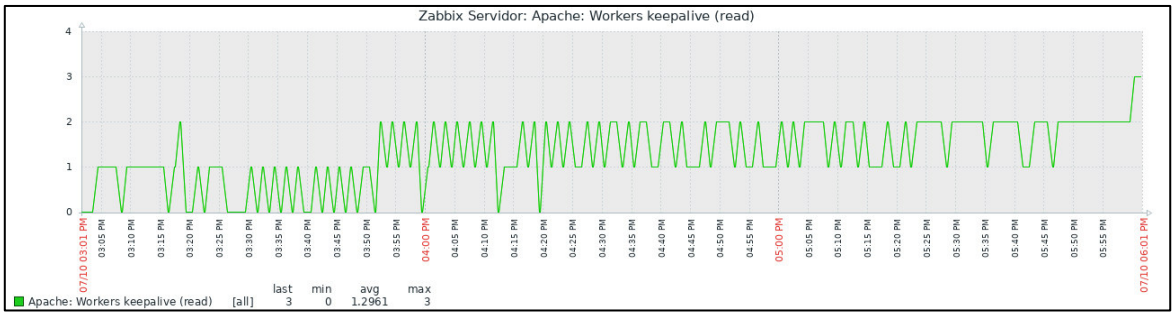


Figura 143. Número de trabajadores que se mantienen vivos (Zabbix)

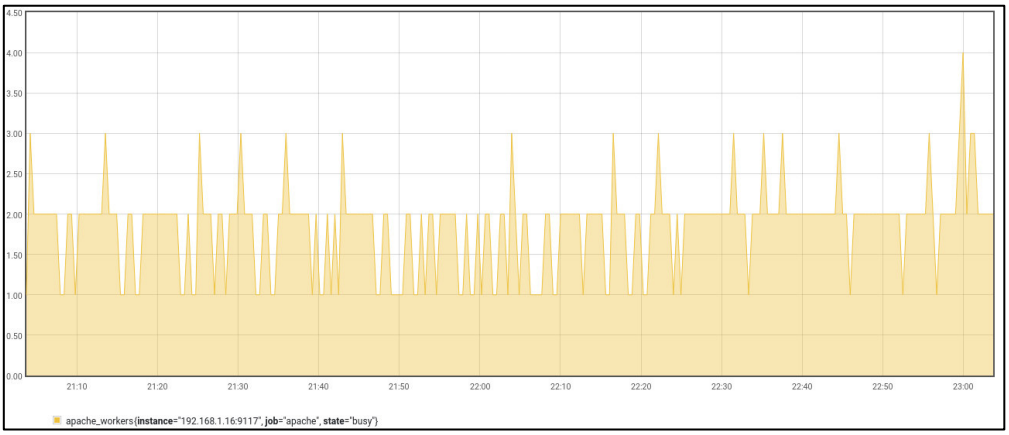


Figura 144. Número de trabajadores (Prometheus)

En la figura 145 se puede ver cuáles son los agentes que con más frecuencia aparecen en los registros del servidor HTTP apache.

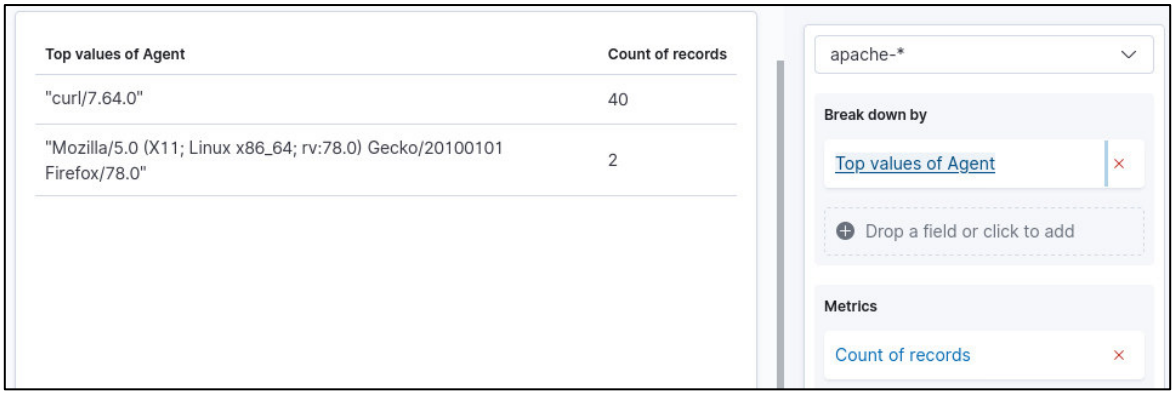


Figura 145. Gráfico de tabla de registros de agentes del servidor Apache (ELK Stack)

En la figura 146 se puede observar cuales son las rutas más consumidas en el servidor HTTP, esta información es útil sí que quiere hacer un estudio de mercado o que planea realizar un balance de carga de algún servicio.

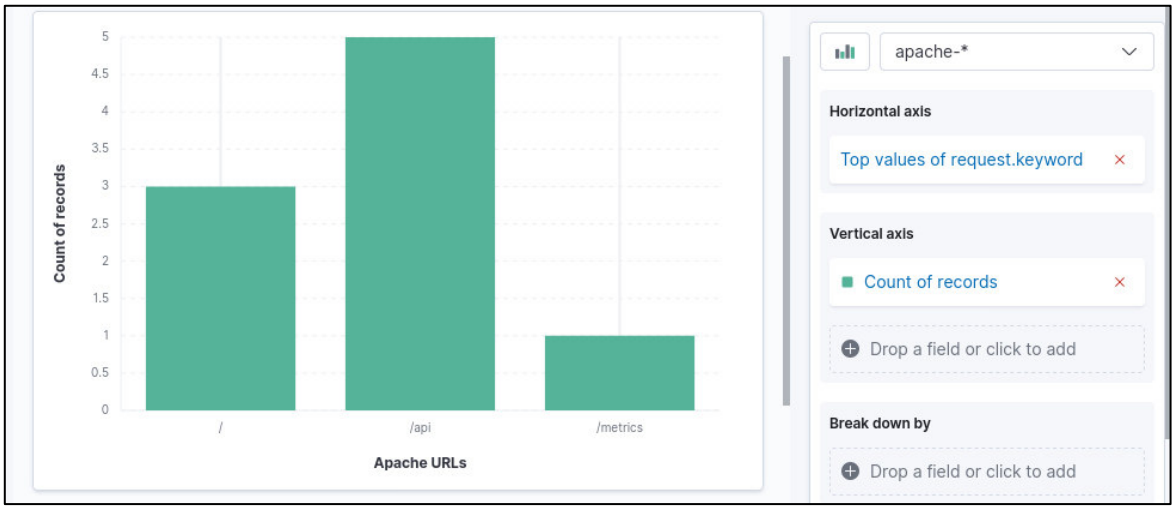


Figura 146. Gráfico de barras de los métodos más usados en el servidor Apache (ELK Stack)

4.1.2. MySQL

En las figuras 147 y 148 se puede ver información sobre las conexiones por segundo registradas en el host. La cantidad de conexiones es muy baja, esto se debe a que el servicio no tiene una gran demanda de uso.

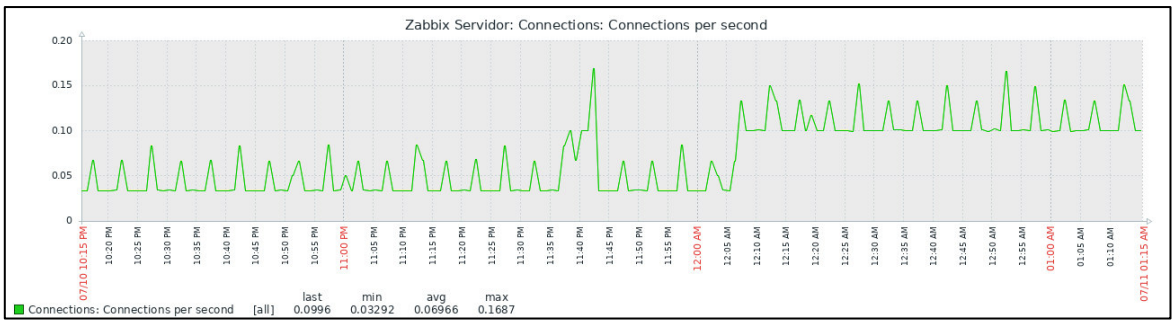


Figura 147. Conexiones por segundo (Zabbix)

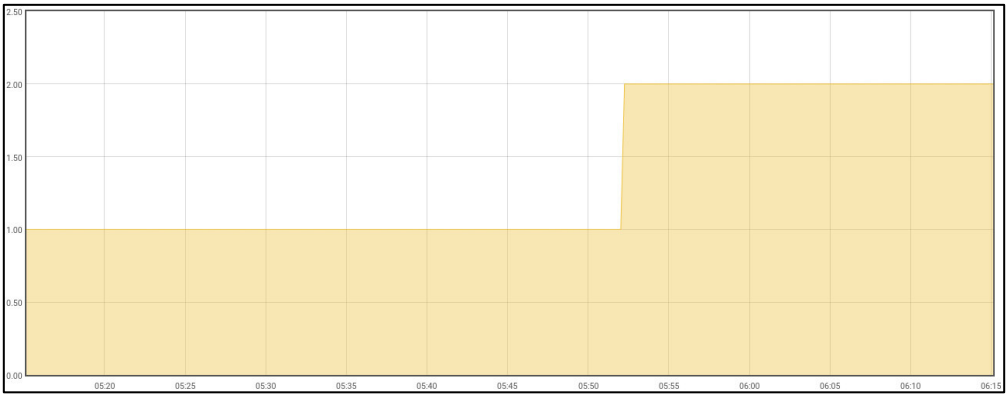


Figura 148. Conexiones por segundo (Prometheus)

Conocer la frecuencia de sentencias de consultas que se ejecutan en el servidor es de utilidad para comprender la demanda de la base de datos, en las figuras 149 y 150 se puede observar los valores registrados de esta métrica.

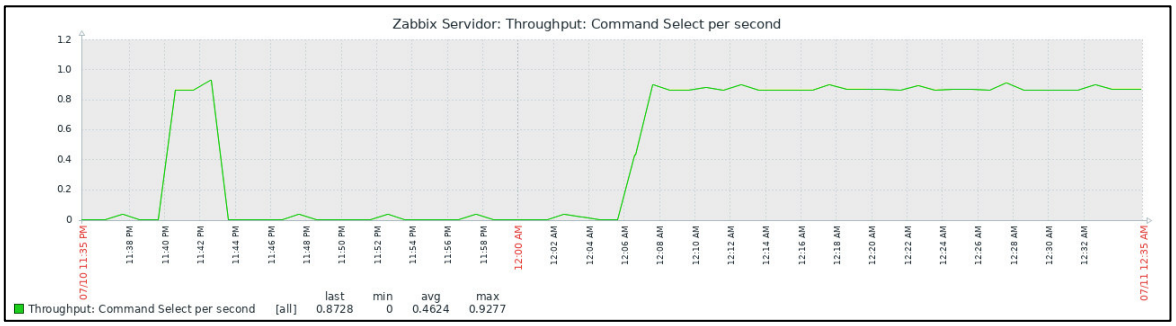


Figura 149. Sentencias "select" por segundo (Zabbix)

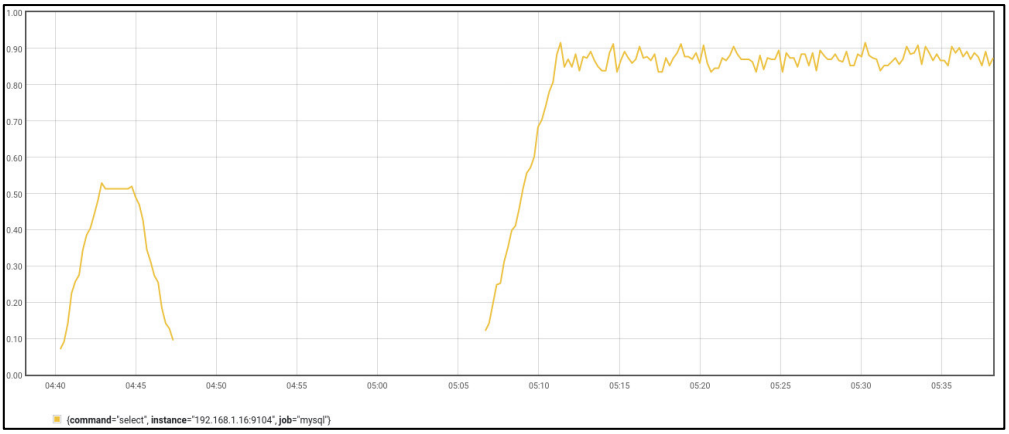


Figura 150. Sentencias "select" por segundo (Prometheus)

En la figura 151 se creó una tabla con todas las sentencias que han superado un umbral de tiempo en responderse. Conocer esta información es de gran utilidad para mejorar el rendimiento de una base de datos, estas demoras pueden ser provocadas por la falta de índices o una consulta mal estructurada.

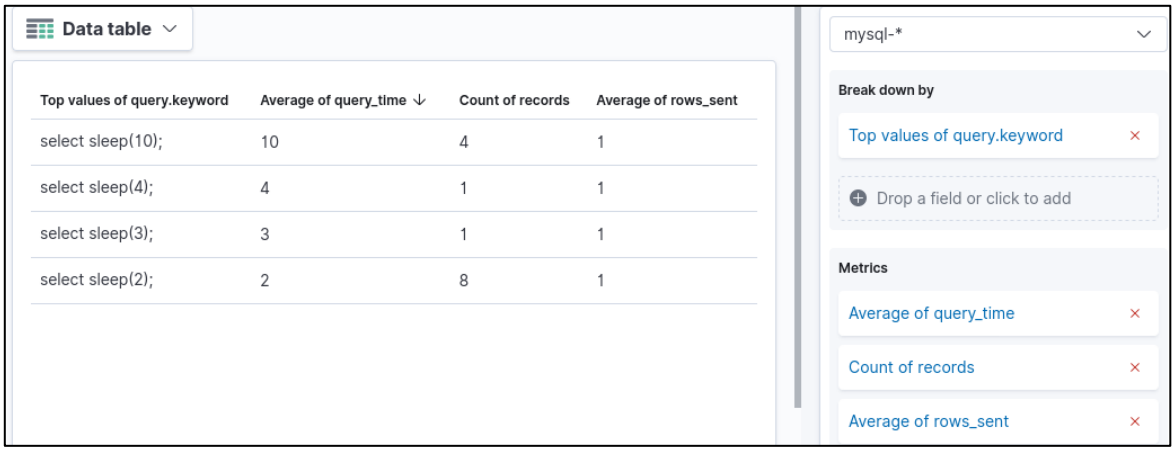


Figura 151. Gráfico de una tabla de las sentencias del archivo log-slow-queries.log

4.1.3. Servidor de Correos Postfix

En la figura 152 se encuentran ordenadas por cantidad de mensajes los emisores que usan el servicio de Postfix.

Figura 152. Gráfico de tabla de la cantidad de correos enviados por un emisor

4.2. Ambiente Real

Las métricas que se van a analizar son las siguientes:

Métricas	
CPU	• Tiempo de usuario
RAM	• Uso de la memoria
Almacenamiento	• Uso de disco dura • Espacio de disco duro
Red	• Bits recibidos • Bits enviados
MySQL	• Número de conexiones abortadas • Número de sentencias “Select” por segundo • Número de sentencias “Insert” por segundo

Tabla 14. Métricas de demostración

4.2.1. Servidor 1

A continuación, se mostrará el resultado de registrar métricas con la herramienta Zabbix por una semana en el servidor 1.

El tiempo del CPU es el principal indicador de actividad del procesador. En las figuras 153 y 154 se puede apreciar que el servidor 1 se encuentra funcionando bajo el umbral del 80% recomendado. En base a esta información este componente funciona correctamente.

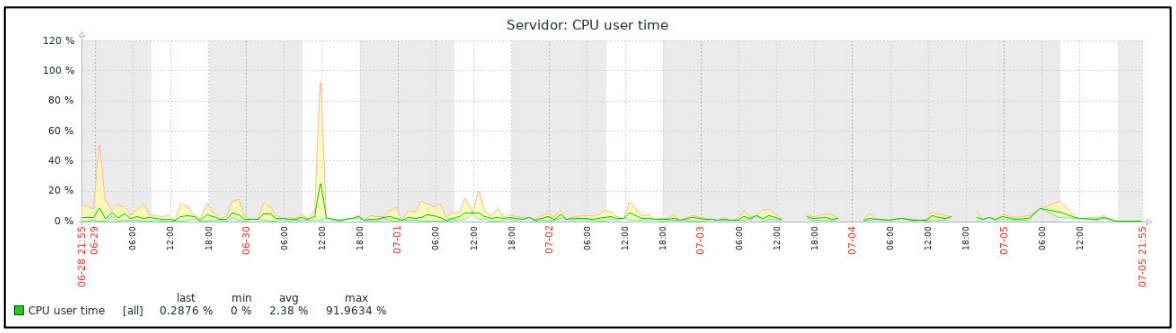


Figura 153. Tiempo de CPU de usuario del Servidor 1 (Zabbix)

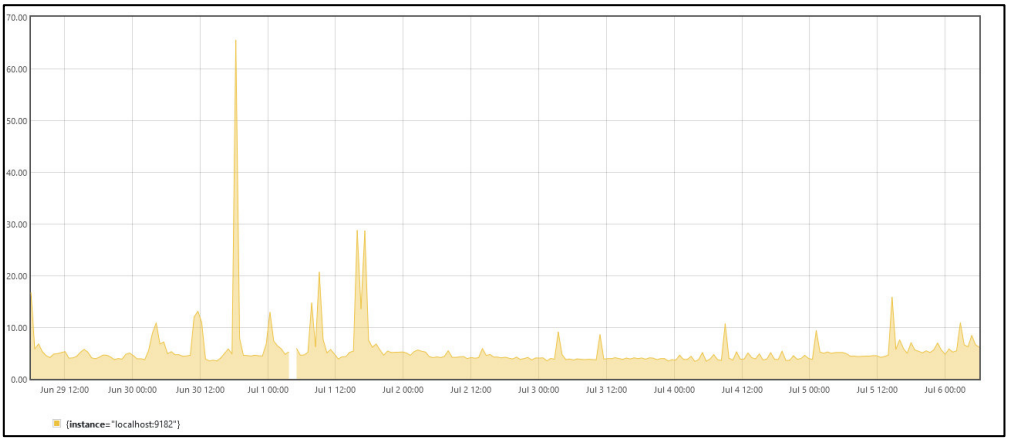


Figura 154. Tiempo de CPU servidor 1 (Prometheus)

Según las figuras 155 y 156 el uso de memoria RAM del servidor 1 es alto, pero sigue siendo menor al umbral de 90%, es recomendable aumentar la capacidad de este recurso o reducir el número de aplicaciones que se estén ejecutando en este servidor.

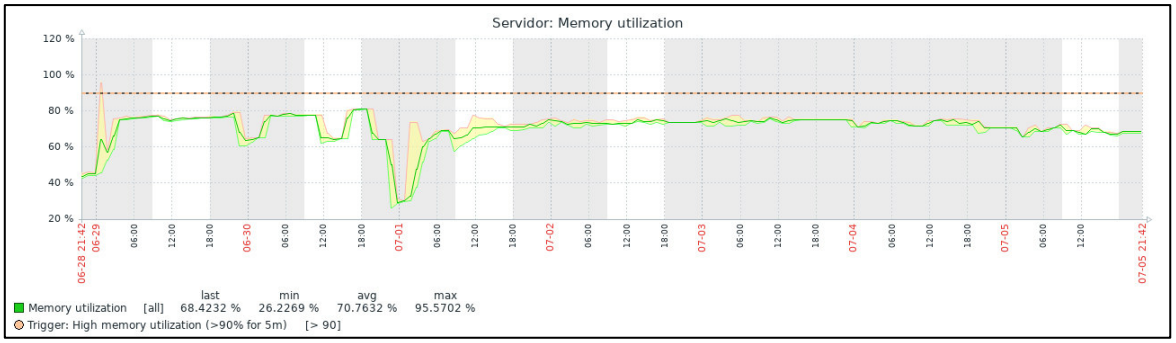


Figura 155. Uso de memoria RAM servidor 1 (Zabbix)

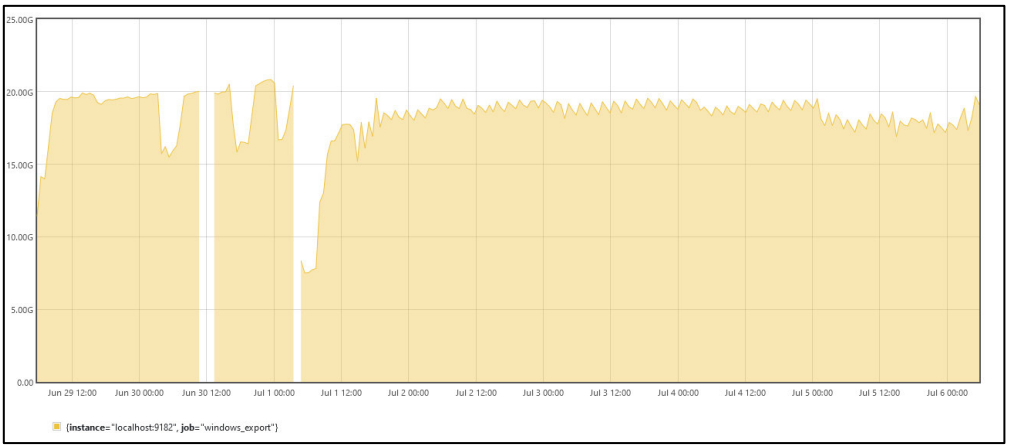


Figura 156. Memoria utilizada servidor 1 (Prometheus)

Como se puede ver en la figura 157 y 158 el servidor 1 ha superado en repetidas ocasiones el 95% de uso de disco duro, esto puede significar cuellos de botella en la respuesta de las aplicaciones, para este problema se cambiar el almacenaje a un SSD para aumentar la velocidad de lectura y escritura.

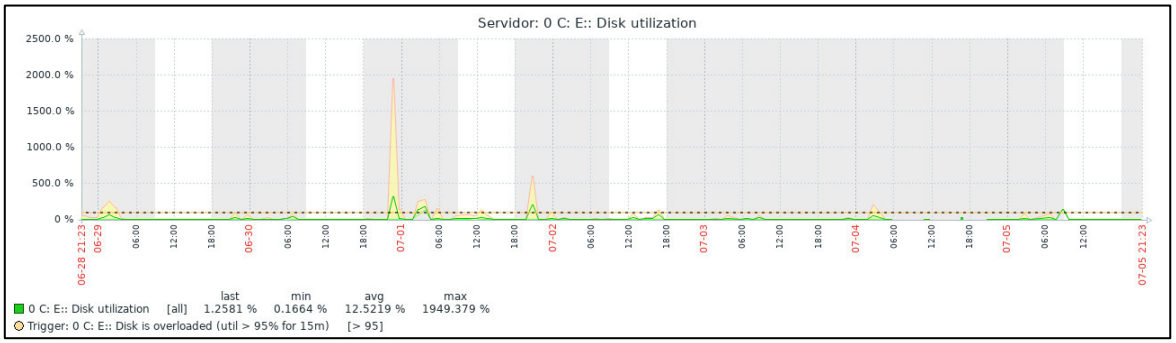


Figura 157. Uso de disco duro servidor 1 (Zabbix)

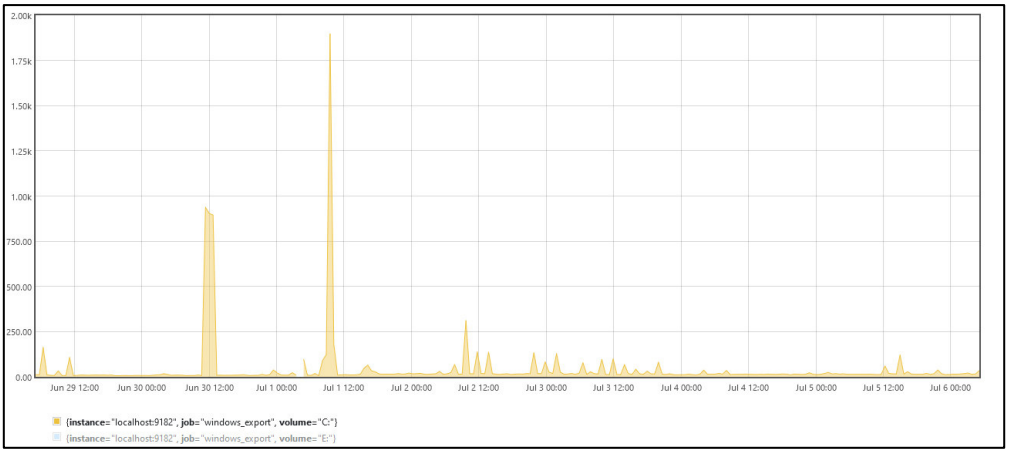


Figura 158. Uso de disco C servidor 1 (Prometheus)

En las figuras 159 y 160 se puede ver el uso del espacio en el disco duro. Se puede apreciar que el espacio libre se ha reducido en los últimos días, casi llegando al máximo de su capacidad.

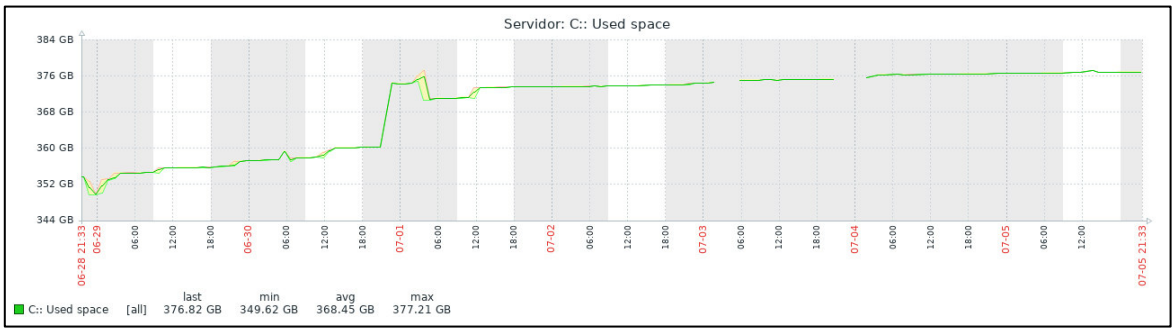


Figura 159. Uso de espacio de disco duro servidor 1 (Zabbix)

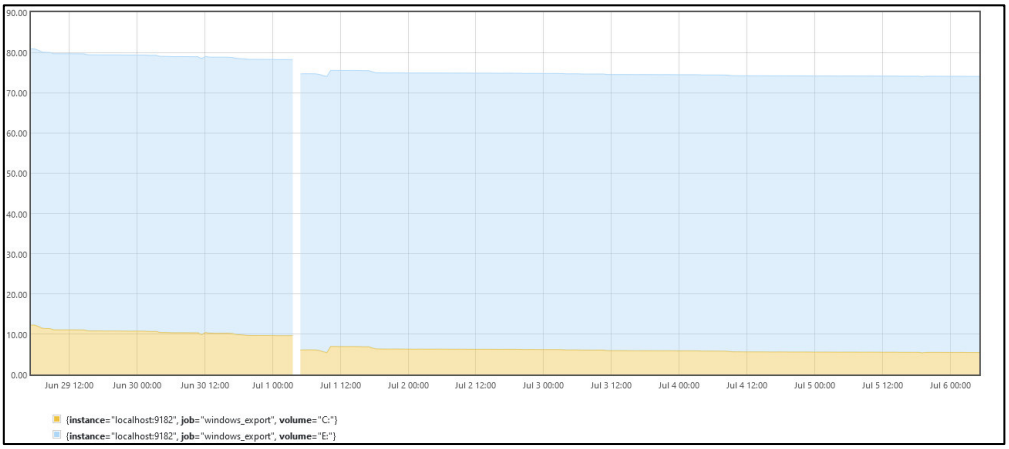


Figura 160. Porcentaje de espacio de disco duro servidor 1 (Prometheus)

En las figuras 161 y 162 se puede ver los bits enviados por la interfaz de red del servidor 1. No se puede apreciar ninguna novedad. El componente parece funcionar correctamente.

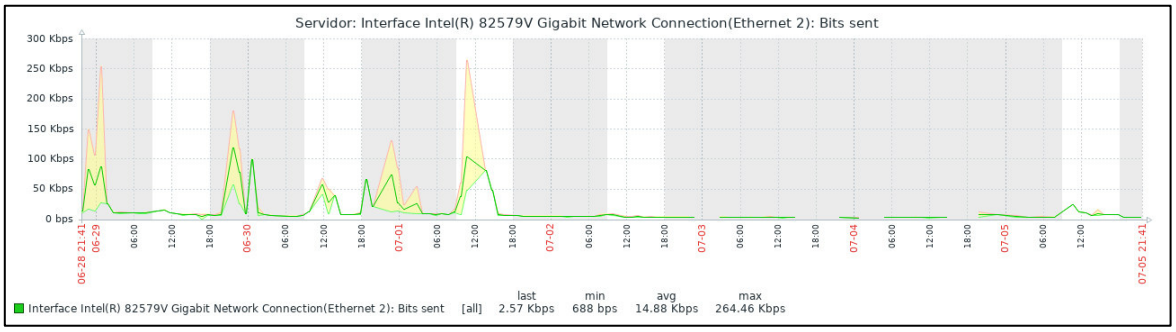


Figura 161. Bits enviados de la interfaz de red servidor 1 (Zabbix)

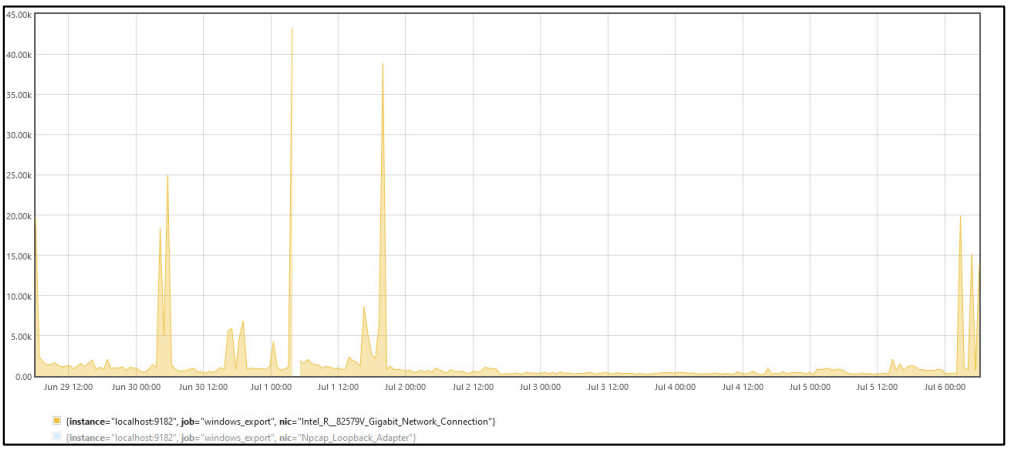


Figura 162. Bits enviados de la interfaz de red servidor 1 (Prometheus)

En las figuras 163 y 164 se puede ver los bits recibidos por la interfaz de red del servidor 1. No se puede apreciar ninguna novedad. El componente parece funcionar correctamente.

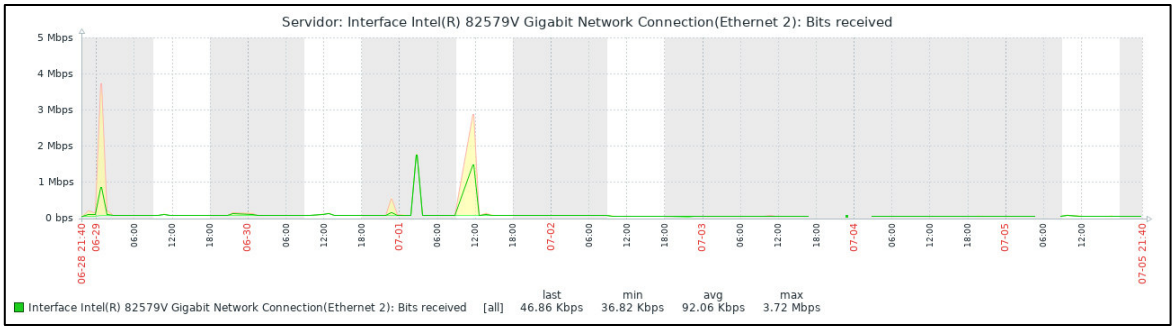


Figura 163. Bits recibidos de la interfaz de red servidor 1 (Zabbix)

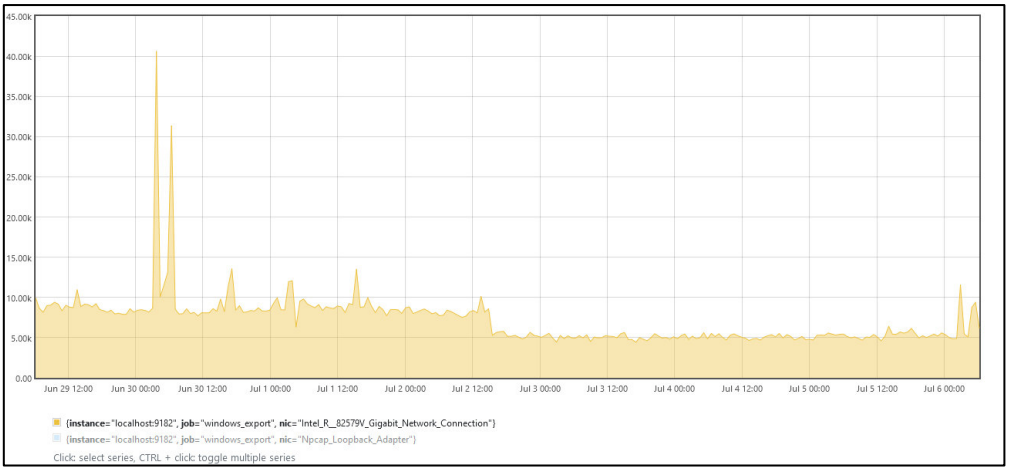


Figura 164. Bits recibidos de la interfaz de red servidor 1 (Prometheus)

En las figuras 165 y 166 se puede apreciar que se ha detectado varios cortes en la disponibilidad del servicio de MySQL en el servidor 1. En caso de que este comportamiento continúe se debe realizar una investigación de este problema.

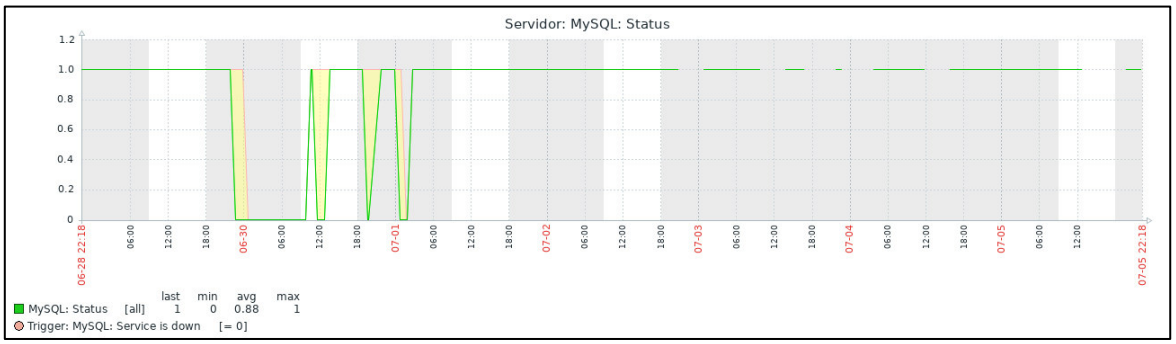


Figura 165. Disponibilidad de la base de datos MySQL servidor 1 (Zabbix)

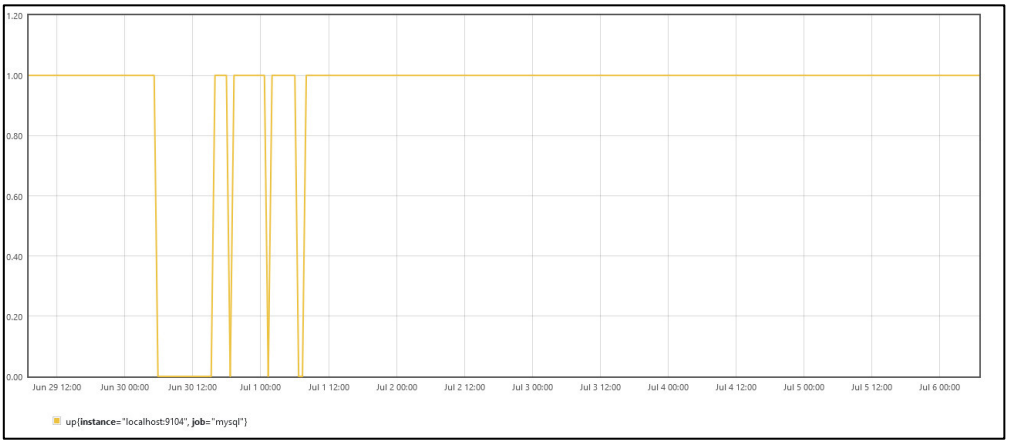


Figura 166. Disponibilidad de la base de datos MySQL servidor 1 (Prometheus)

La supervisión de las conexiones de los clientes es fundamental, ya que una vez que se han agotado las conexiones disponibles, se rechazarán las nuevas conexiones del cliente. Para el servidor 1 no se requiere ningún tipo de tratamiento ya que el número de conexiones es mucho menor a las 151 máximas de la configuración inicial de MySQL, esto se puede ver en las figuras 167 y 168.

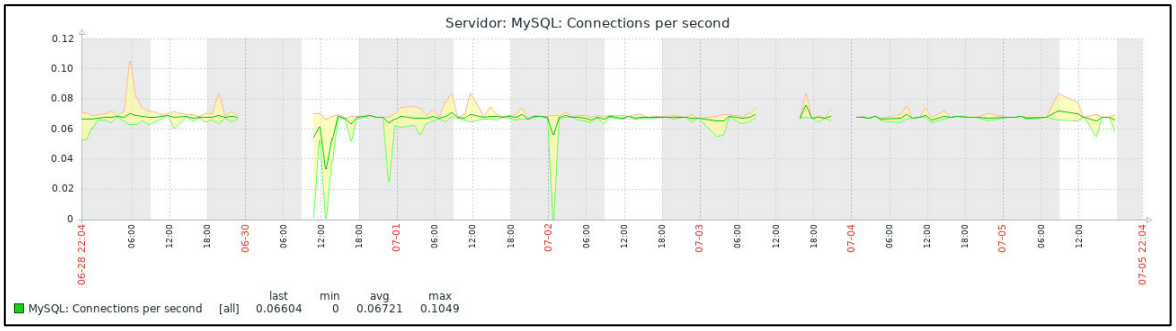


Figura 167. Conexiones por segundo de la base de datos MySQL servidor 1 (Zabbix)

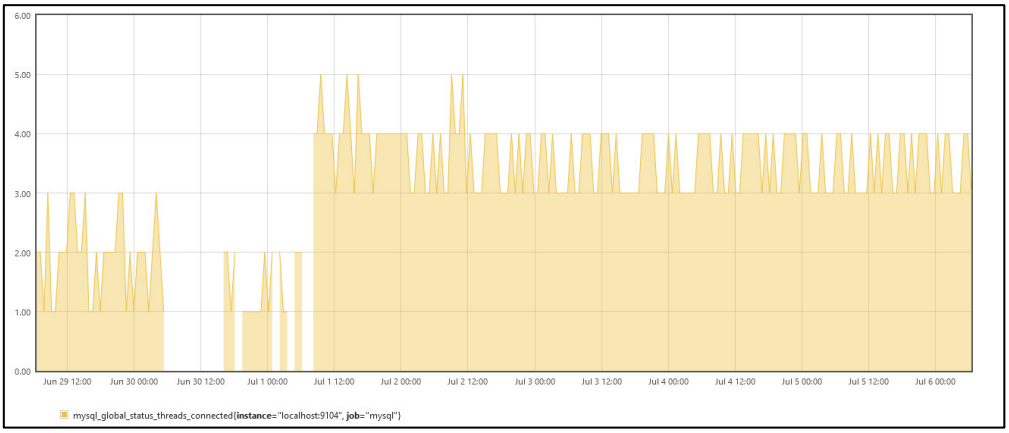


Figura 168. Conexiones por segundo MySQL servidor 1 (Prometheus)

Esta métrica registra la cantidad de clientes que no se han podido conectar a la base de datos. Un aumento de esta métrica tiene un impacto directo en la experiencia de los usuarios. Según la figura 169 y 170 el servidor 1 no tiene problema de conexiones abortadas.

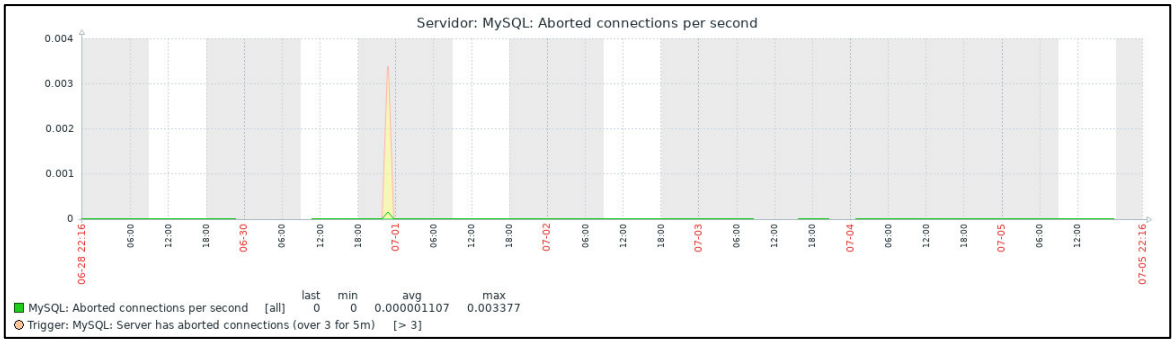


Figura 169. Conexiones abortadas por segundo de la base de datos MySQL servidor 1 (Zabbix)

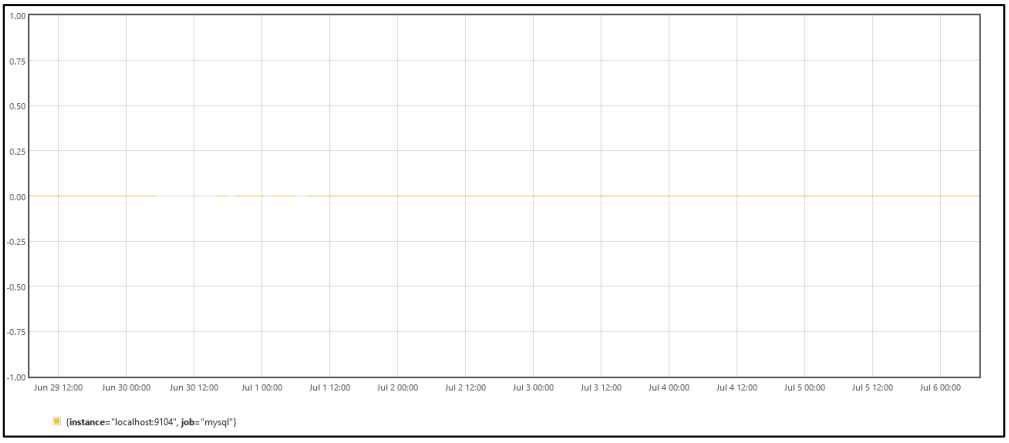
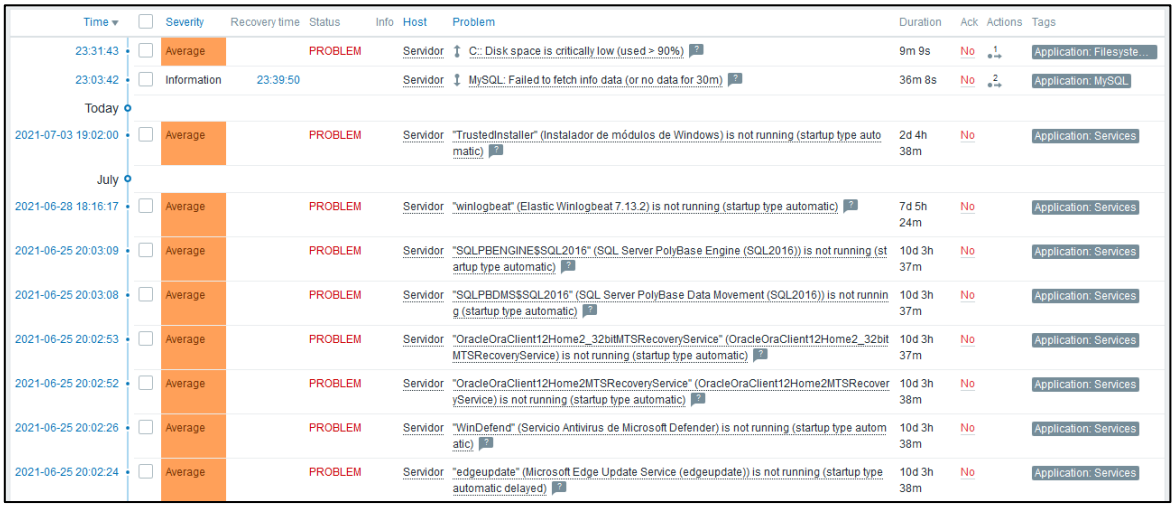


Figura 170. Conexiones abortadas por segundo MySQL servidor 1 (Prometheus)

En la figura 175 se puede ver todos los problemas encontrados por la herramienta Zabbix en el servidor 1.



Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags
23:31:43	Average		PROBLEM		Servidor	C:: Disk space is critically low (used > 90%)	9m 9s	No	1	Application: Filesyste...
23:03:42	Information	23:39:50			Servidor	MySQL: Failed to fetch info data (or no data for 30m)	36m 8s	No	2	Application: MySQL
Today										
2021-07-03 19:02:00	Average		PROBLEM		Servidor	"TrustedInstaller" (Instalador de módulos de Windows) is not running (startup type automatic)	2d 4h 38m	No		Application: Services
July										
2021-06-28 18:16:17	Average		PROBLEM		Servidor	"winlogbeat" (Elastic Winlogbeat 7.13.2) is not running (startup type automatic)	7d 5h 24m	No		Application: Services
2021-06-25 20:03:09	Average		PROBLEM		Servidor	"SQLPBENGINE$SQL2016" (SQL Server PolyBase Engine (SQL2016)) is not running (startup type automatic)	10d 3h 37m	No		Application: Services
2021-06-25 20:03:08	Average		PROBLEM		Servidor	"SQLPBDMS$SQL2016" (SQL Server PolyBase Data Movement (SQL2016)) is not running (startup type automatic)	10d 3h 37m	No		Application: Services
2021-06-25 20:02:53	Average		PROBLEM		Servidor	"OracleOraClient12Home2_32bitMTSRecoveryService" (OracleOraClient12Home2_32bitMTSRecoveryService) is not running (startup type automatic)	10d 3h 37m	No		Application: Services
2021-06-25 20:02:52	Average		PROBLEM		Servidor	"OracleOraClient12Home2MTSRecoveryService" (OracleOraClient12Home2MTSRecoveryService) is not running (startup type automatic)	10d 3h 38m	No		Application: Services
2021-06-25 20:02:26	Average		PROBLEM		Servidor	"WinDefend" (Servicio Antivirus de Microsoft Defender) is not running (startup type automatic)	10d 3h 38m	No		Application: Services
2021-06-25 20:02:24	Average		PROBLEM		Servidor	"edgeupdate" (Microsoft Edge Update Service (edgeupdate)) is not running (startup type automatic delayed)	10d 3h 38m	No		Application: Services

Figura 171. Tabla de problemas encontrados del servidor 1 (Zabbix)

En la figura 172 se puede ver que se las alertas activas en Prometheus.

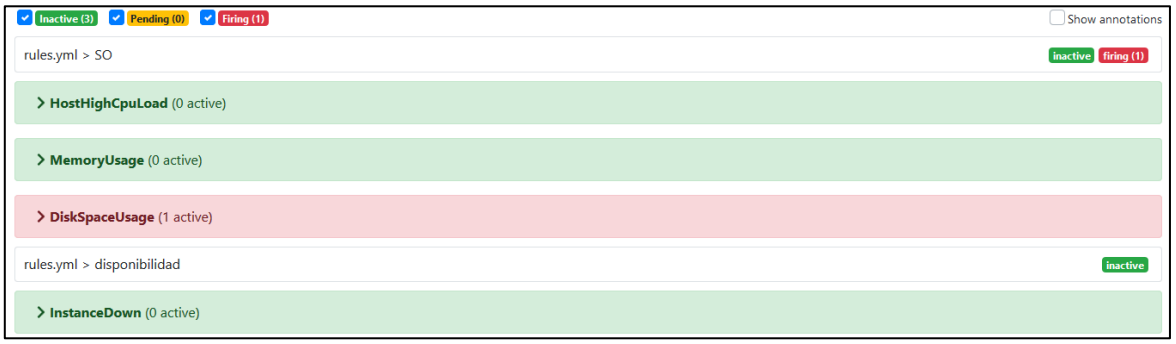


Figura 172. Módulo alertas (Prometheus)

Como se puede ver en la figura 173 no se ha detectado problemas de conexión con el agente Zabbix en el servidor 1.

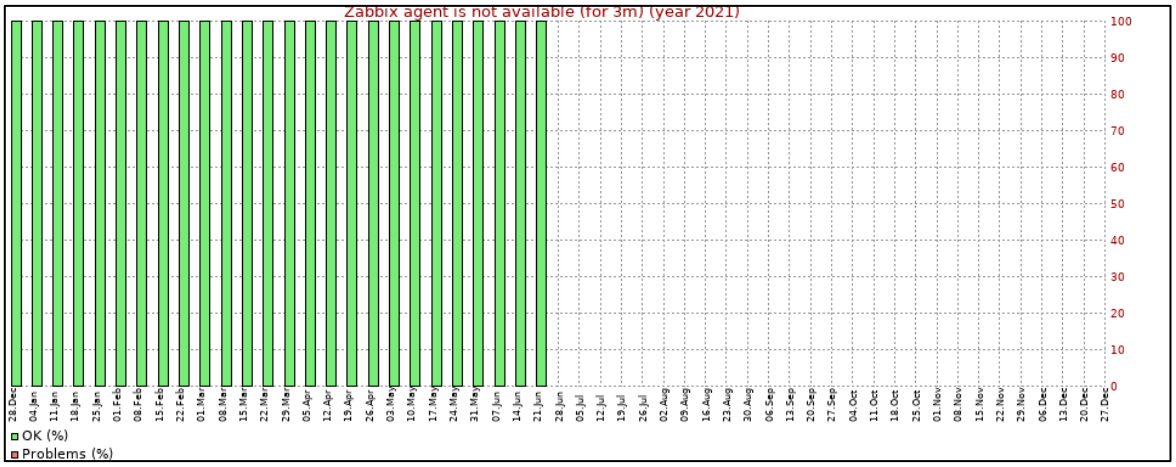


Figura 173. Reporte de disponibilidad de agente Zabbix servidor 1

4.2.2. Servidor 2

A continuación, se mostrará el resultado de registrar métricas con la herramienta Zabbix por una semana en el servidor 2.

En las figuras 174 Y 175 se puede observar que el porcentaje de uso del CPU no pasa el 12%. Muy debajo del umbral del 80%.

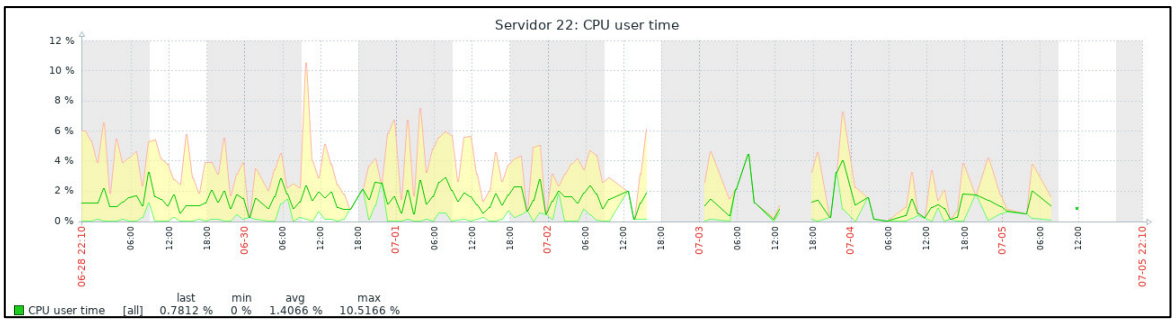


Figura 174. Tiempo de CPU de usuario del servidor 2 (Zabbix)

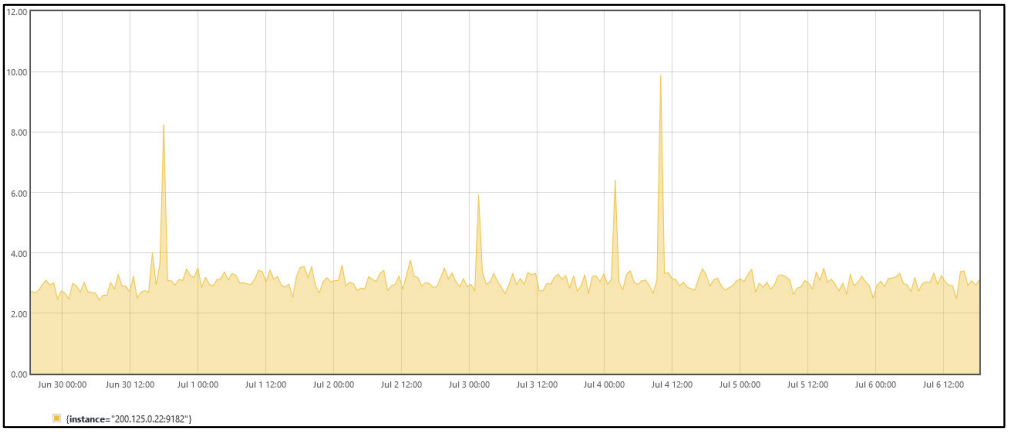


Figura 175. Uso de CPU servidor 2 (Prometheus)

En base a las figuras 176 y 177 el uso de memoria RAM se mantiene por debajo del 55% de uso. No se ha encontrado problemas con este recurso.

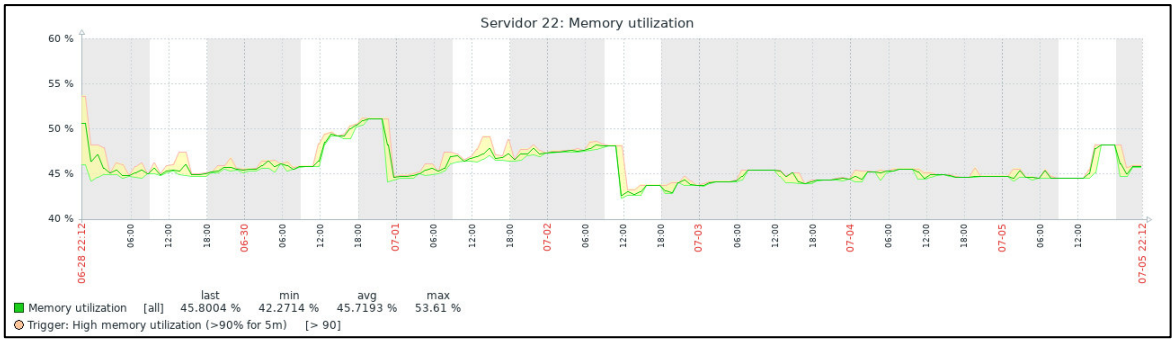


Figura 176. Uso de memoria servidor 2 (Zabbix)

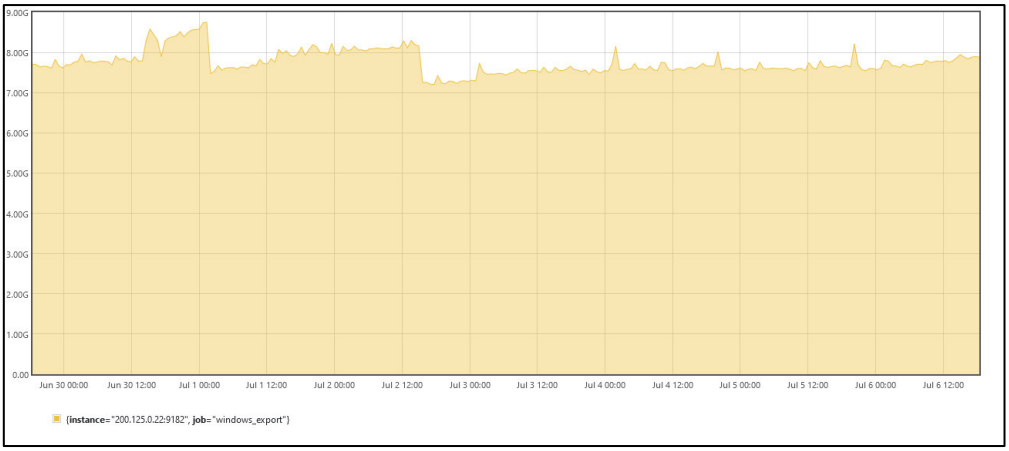


Figura 177. Uso de memoria servidor 2 (Prometheus)

En las figuras 178 y 179 se puede apreciar que en una ocasión se superó el umbral de 95% de uso de disco duro, en caso de que este problema persista se recomienda usar un disco duro SSD.

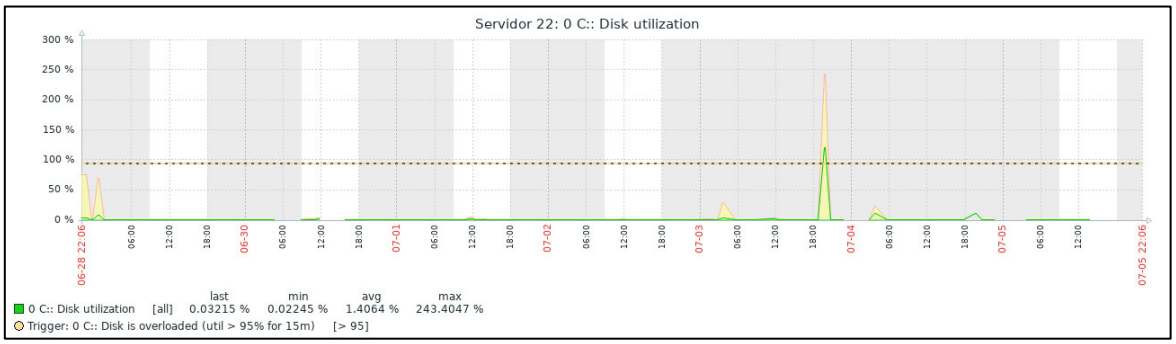


Figura 178. Uso de disco duro servidor 2 (Zabbix)

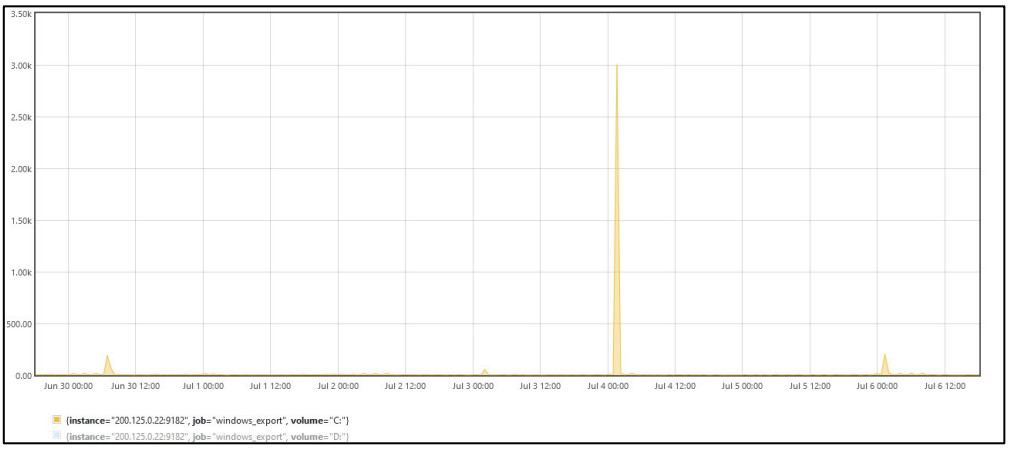


Figura 179. Uso de disco C servidor 2 (Prometheus)

Según lo que se puede ver en las figuras 180 y 181 en espacio ocupado en los discos C es de alrededor de 320 GB, lo que deja un 30% de espacio libre en la partición C. El servidor 2 posee suficiente espacio para las actuales funciones que cumple.

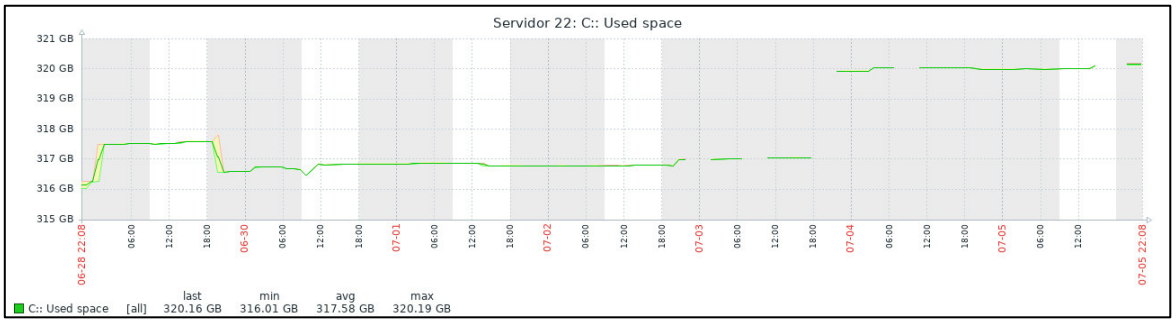


Figura 180. Uso de espacio del disco servidor 2 (Zabbix)

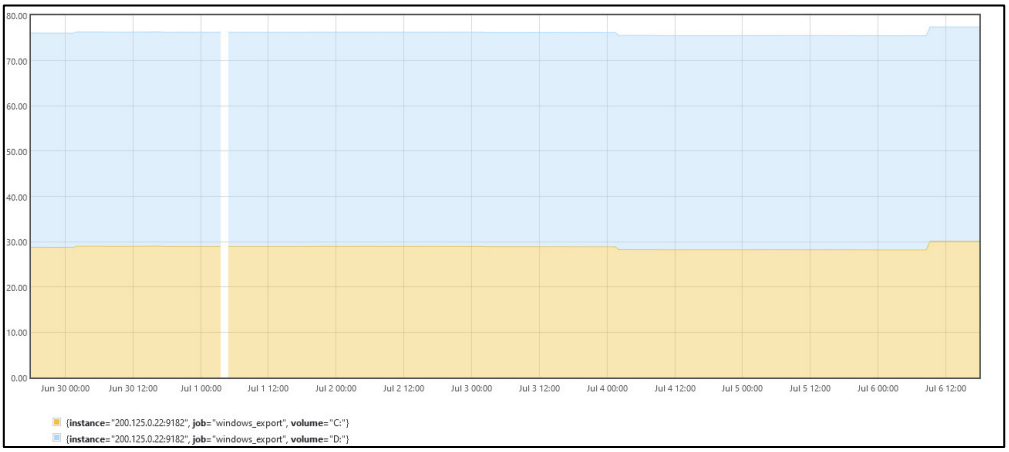


Figura 181. Espacio de disco servidor 2 (Prometheus)

En las figuras 182 y 183 no se puede apreciar ninguna novedad. El componente parece funcionar correctamente.

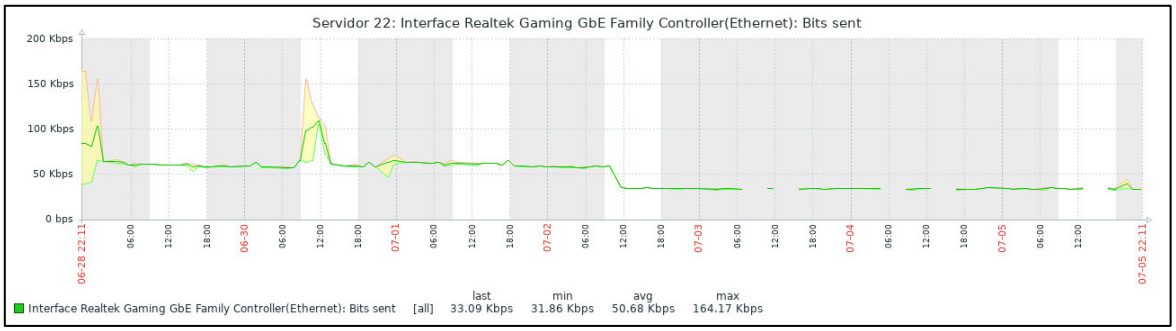


Figura 182. Bits enviados de la interfaz de red servidor 2 (Zabbix)

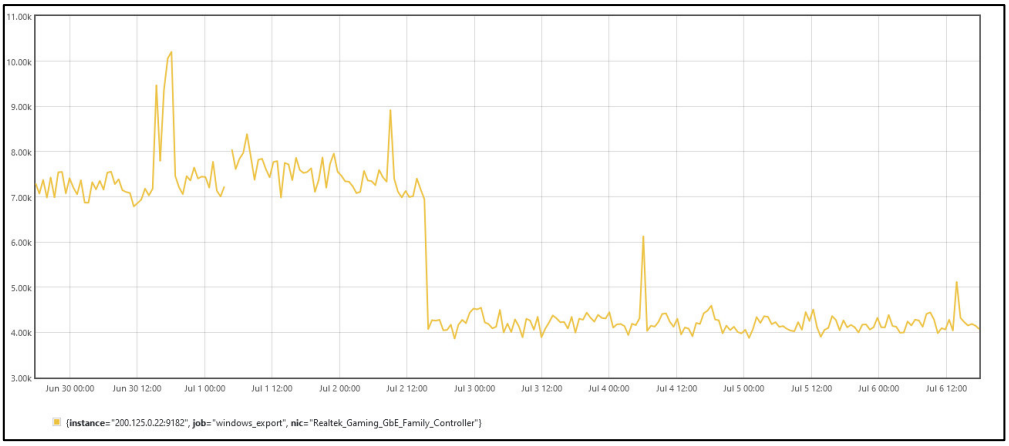


Figura 183. Bits enviados de la interfaz de red servidor 2 (Prometheus)

En las figuras 184 y 185 no se puede apreciar ninguna novedad. El componente parece funcionar correctamente.

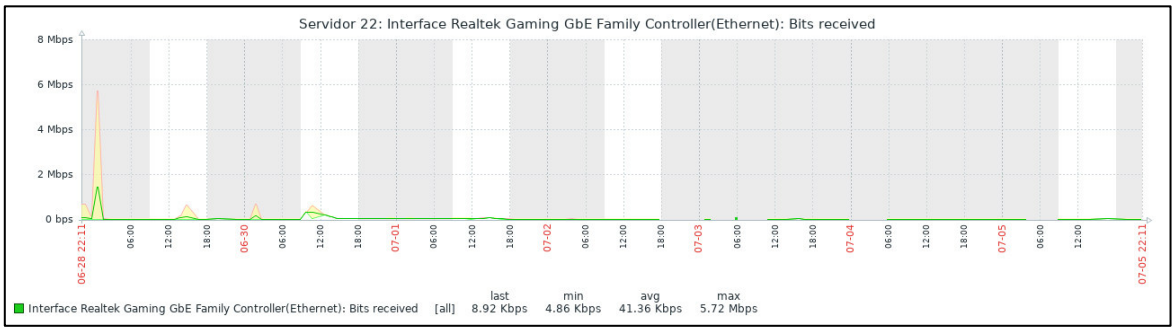


Figura 184. Bits recibidos de la interfaz de red servidor 2 (Zabbix)

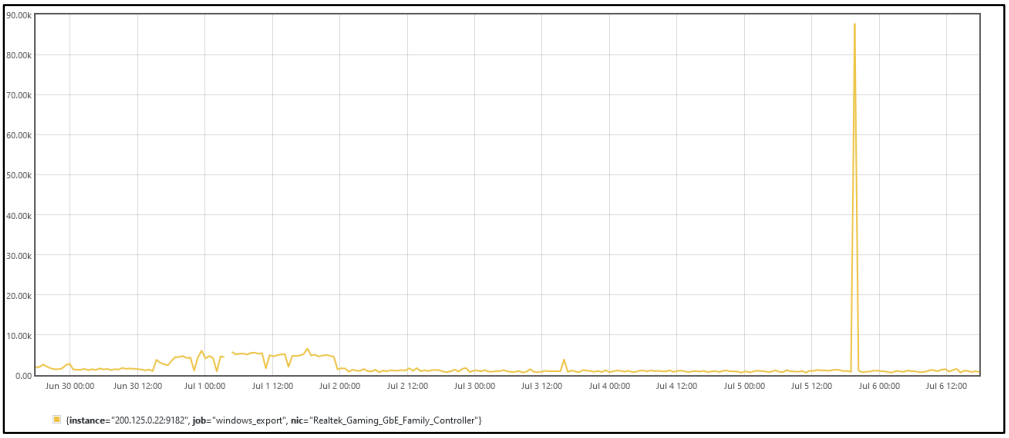


Figura 185. Bits recibidos de la interfaz de red servidor 2 (Prometheus)

En base a las figuras 186 y 187 se puede observar que el servicio de MySQL se ha interrumpido en una sola ocasión por un corto periodo de tiempo, en caso de que este problema continúe se debe investigar la causa.

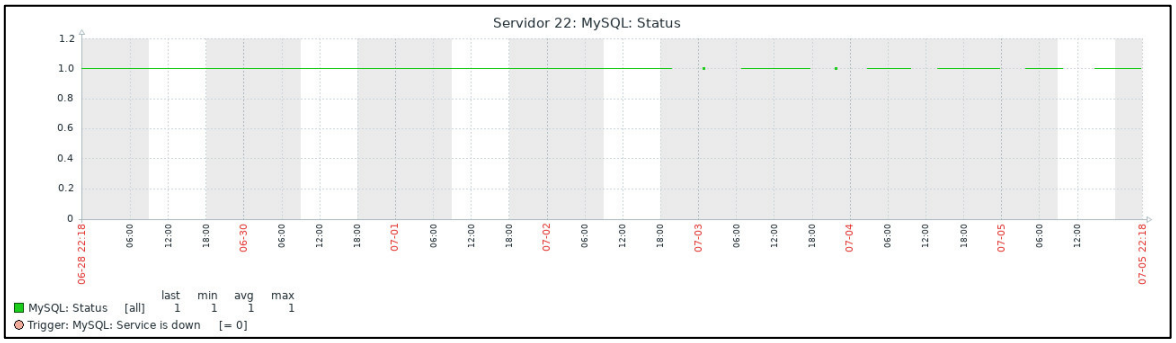


Figura 186. Disponibilidad de la base de datos MySQL servidor 1 (Zabbix)

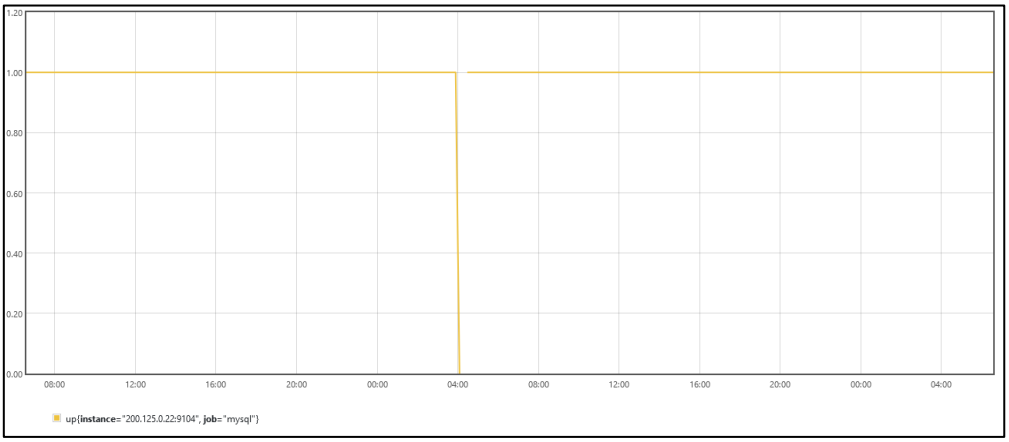


Figura 187. Disponibilidad de MySQL servidor 2 (Prometheus)

Según las figuras 188 y 189 las conexiones por segundo son mucho menor al umbral de 151. El servicio no requiere mantenimiento.

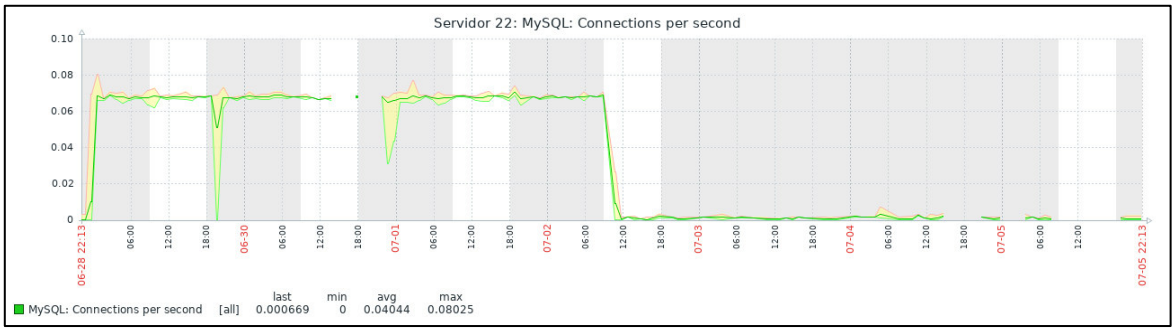


Figura 188. Conexiones por segundo de la base de datos MySQL servidor 2 (Zabbix)

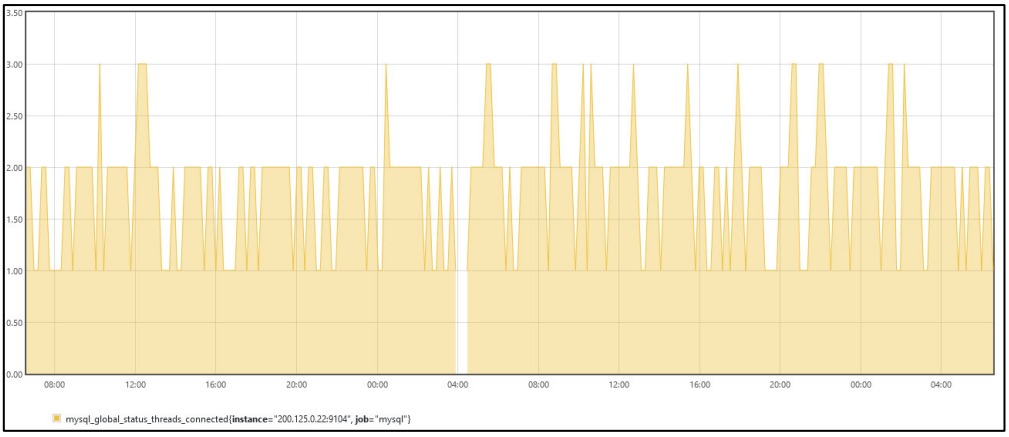


Figura 189. Conexiones por segundo de MySQL servidor 2 (Prometheus)

En las figuras 190 y 191 no se han detectado conexiones abortadas.

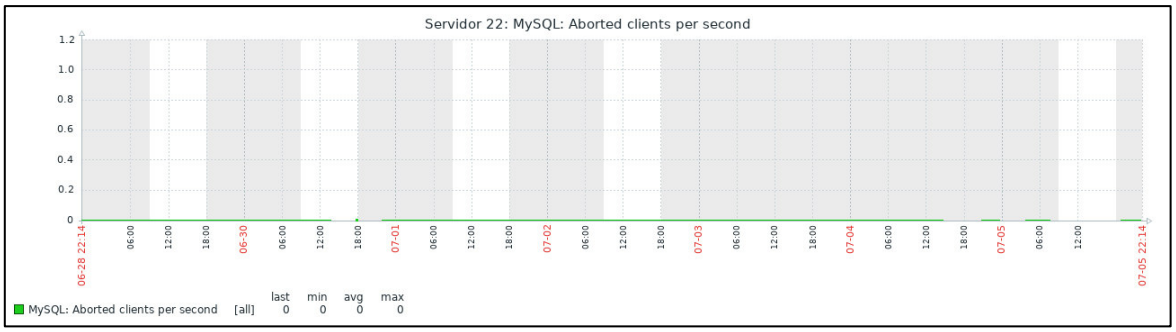


Figura 190. Conexiones abortadas por segundo de la base de datos MySQL servidor 2 (Zabbix)

Como se puede ver en la figura 192 no se ha registrado conexiones abortadas en la base de datos MySQL del servidor 2.

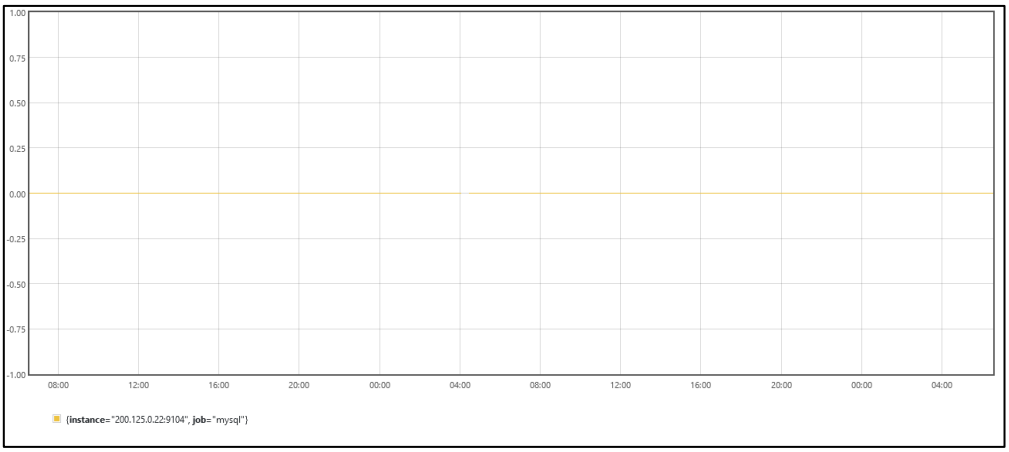
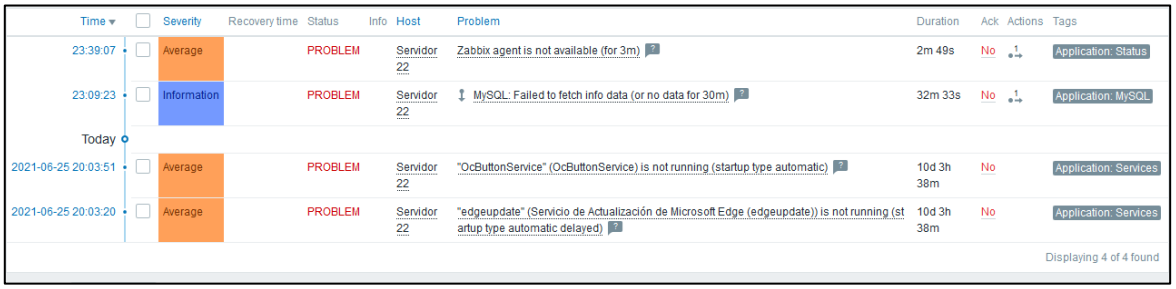


Figura 191. Conexiones abortadas por segundo de MySQL servidor 2 (Prometheus)

En la figura 192 se puede ver todos los problemas encontrados en por la herramienta Zabbix en el servidor 2.



Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags
23:39:07	Average		PROBLEM		Servidor 22	Zabbix agent is not available (for 3m)	2m 49s	No		Application: Status
23:09:23	Information		PROBLEM		Servidor 22	MySQL: Failed to fetch info data (or no data for 30m)	32m 33s	No		Application: MySQL
Today										
2021-06-25 20:03:51	Average		PROBLEM		Servidor 22	"OcButtonService" (OcButtonService) is not running (startup type automatic)	10d 3h 38m	No		Application: Services
2021-06-25 20:03:20	Average		PROBLEM		Servidor 22	"edgeupdate" (Servicio de Actualización de Microsoft Edge (edgeupdate)) is not running (startup type automatic delayed)	10d 3h 38m	No		Application: Services

Displaying 4 of 4 found

Figura 192. Tabla de problemas encontrados del servidor 2 (Zabbix)

Como se puede ver en la figura 193 no se ha detectado problemas de conexión con el agente Zabbix en el servidor 2.

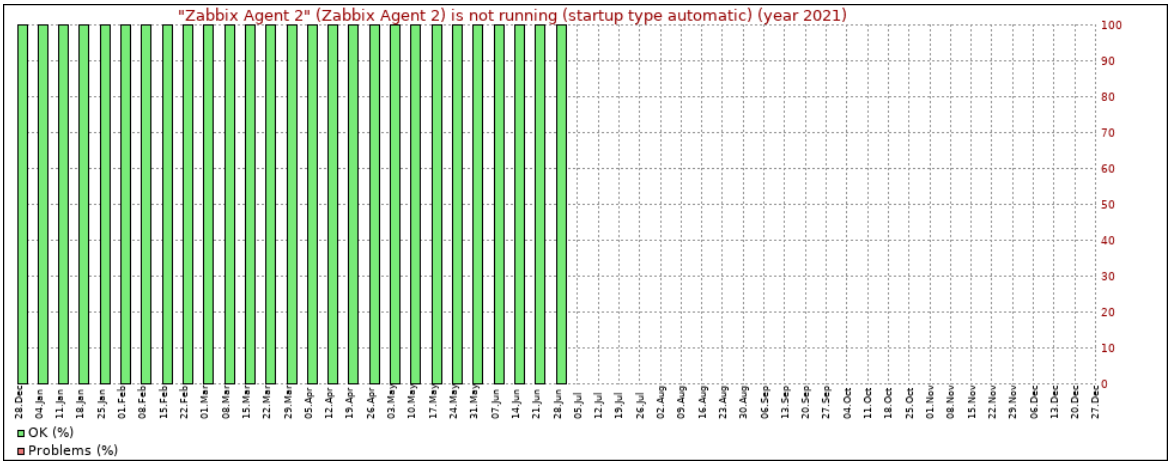


Figura 193. Reporte de disponibilidad de agente Zabbix servidor 2

4.2.3. Pruebas

Se realizó varias pruebas para comprobar el funcionamiento de la consola. Las alarmas que se van a probar son las siguientes:

Alarmas	
Disponibilidad	Capacidad de un sistema en funcionar continuamente

Alto uso de CPU	Esta alerta se genera cuando el uso del CPU supera el umbral del 80%
Alto uso de disco duro	Esta alerta se genera cuando el uso del disco duro supera el umbral del 80%
Alto uso de memoria RAM	Esta alerta se genera cuando el uso de la memoria RAM supera el umbral del 80%

Tabla 15. Alarmas de demostración

Para la primera prueba se detuvo el servicio de la base de datos MySQL, al cabo de unos minutos se pudo ver en el módulo de problemas de Zabbix (Ver figura 194) y el de alertas de Prometheus (Ver figura 195) que esta instancia MySQL se detuvo,

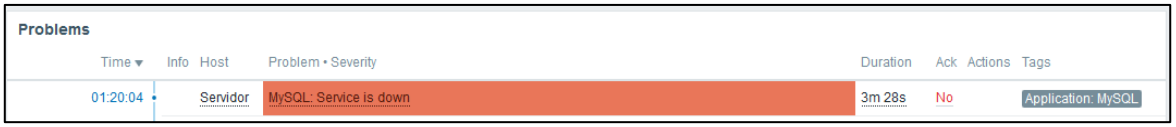


Figura 194. Activación alerta Servicio MySQL no funciona de (Zabbix)

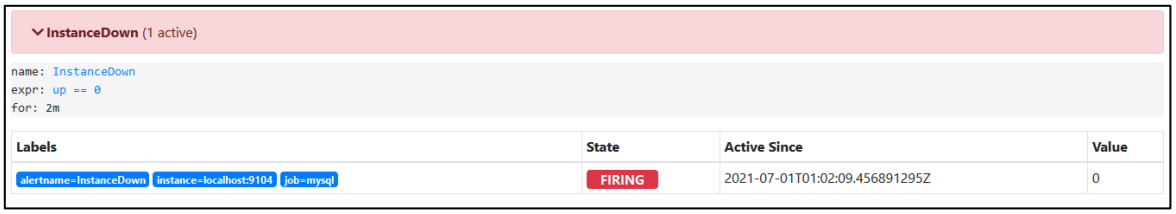


Figura 195. Activación de alerta InstanceDown (Prometheus)

También se recibió una notificación por los medios configurados (Ver en las figuras 196 y 197).

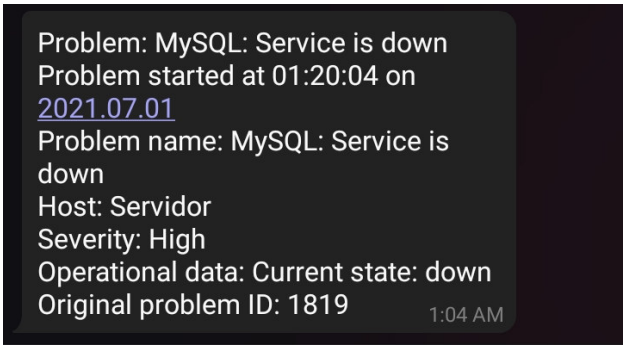


Figura 196. Notificación de alerta servicio MySQL no funciona (Zabbix)

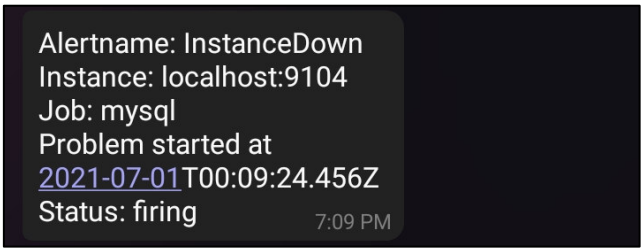


Figura 197. Notificación de alerta InstanceDown (Prometheus)

En el caso del CPU se usó una prueba de estrés la cual se realizó por medio del software CPU-Z.

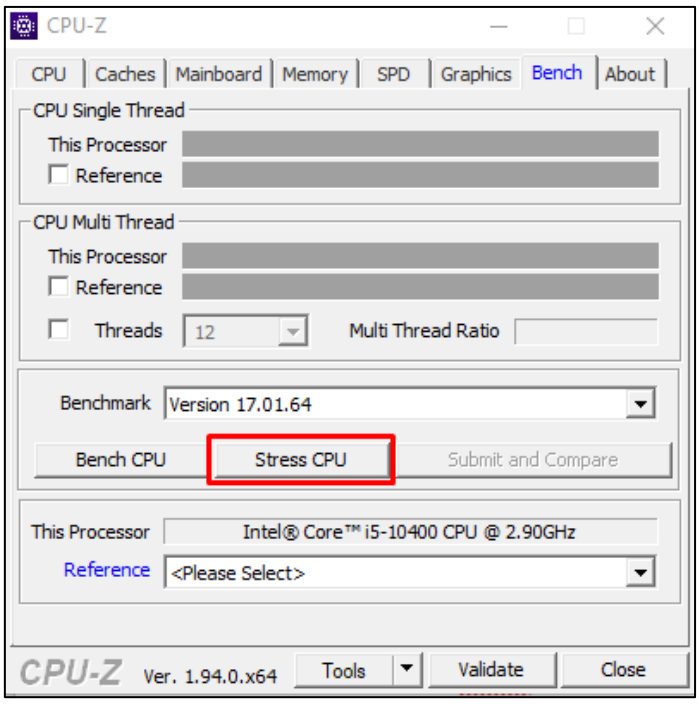


Figura 198. Módulo Bench de CPU-Z

Una vez ejecutada la prueba de estrés al cabo de unos minutos se activó la alerta en las dos herramientas (Ver en la figura 199 y 201).

Figura 199. Activación de alerta Carga de CPU muy alta (Zabbix)

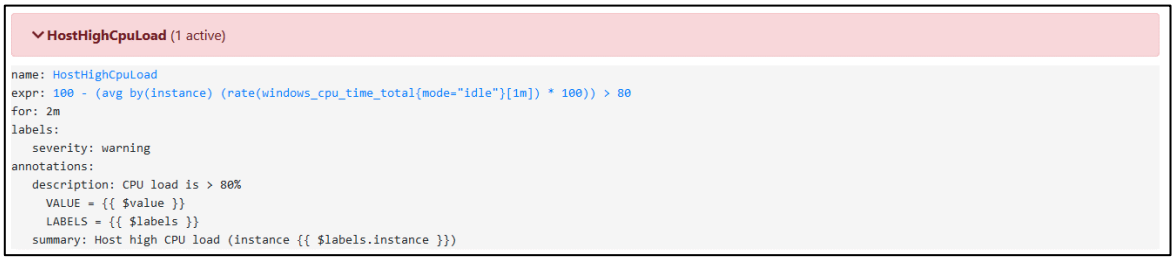


Figura 200. Activación de alerta HostHighCPULoad (Prometheus)

Se recibió las respectivas notificaciones del problema registrado en las herramientas Zabbix y Prometheus (Ver figuras 200 y 201).

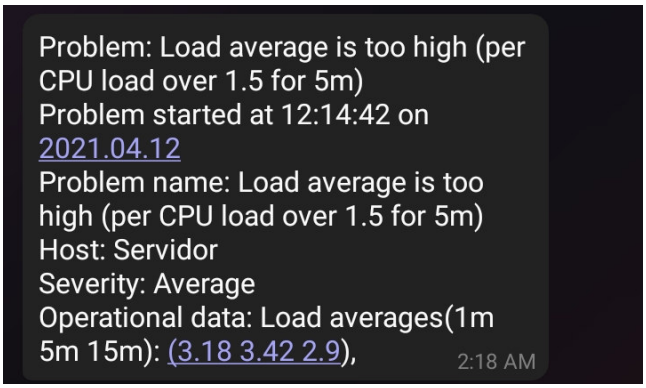


Figura 201. Notificación de carga del CPU muy alta (Zabbix)

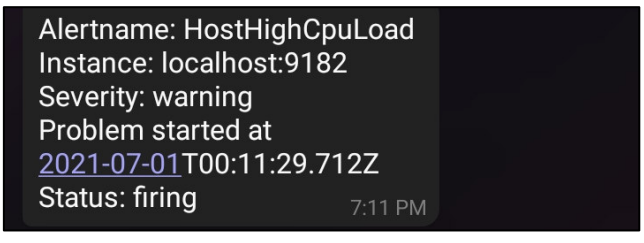


Figura 202. Notificación de alerta HostHighCPULoad (Prometheus)

Para probar la alerta de uso de memoria se modificó temporalmente el umbral de esta, la alerta se activó al cabo de unos minutos (Ver figuras 203 y 204).

Figura 203. Activación alerta alto uso de memoria (Zabbix)

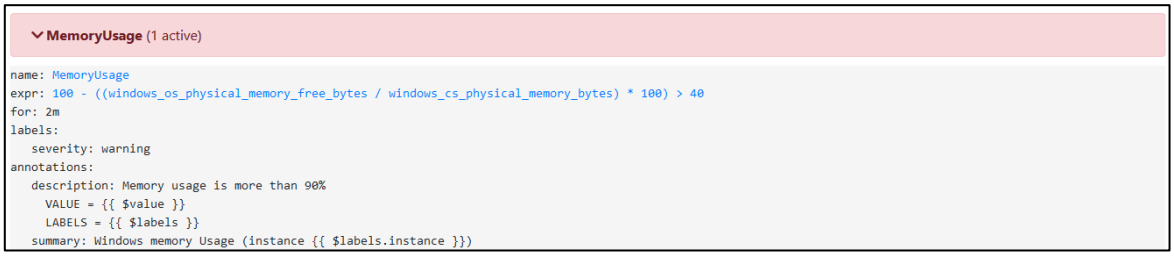


Figura 204. Activación de alerta MemoryUsage (Prometheus)

En la aplicación móvil de Telegram también se observó información sobre esta alerta (Ver en las figuras 205 y 206).

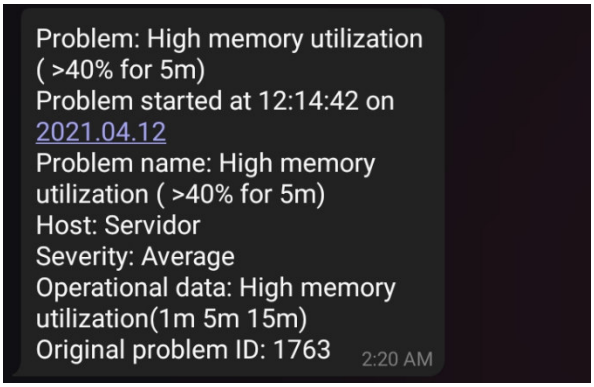


Figura 205. Notificación de alerta alto uso de memoria (Zabbix)

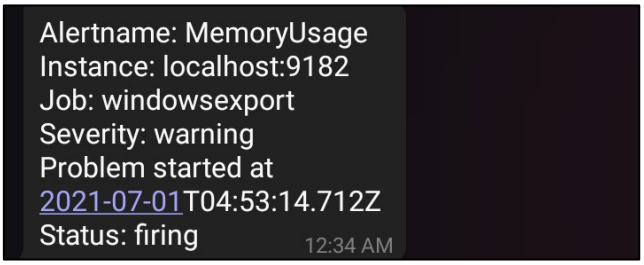


Figura 206. Notificación de alerta MemoryUsage (Prometheus)

El servidor 1 tiene más del 90% de la capacidad del disco duro ocupada esto se ve reflejado en la alerta configurada (Ver figuras 207 y 209).

Figura 207. Activación de alerta espacio de disco críticamente bajo (Zabbix)

Figura 208. Activación de alerta DiskSpaceUsage (Prometheus)

En la aplicación móvil de Telegram también se puedó observar información sobre el problema del espacio libre del disco duro (Ver figuras 209 y 210).

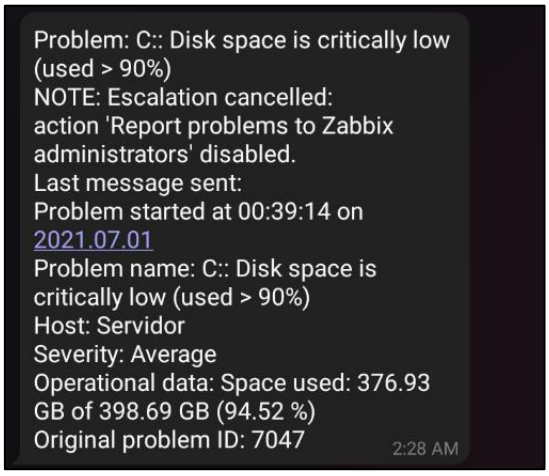


Figura 209. Notificación de espacio de disco críticamente bajo (Zabbix)

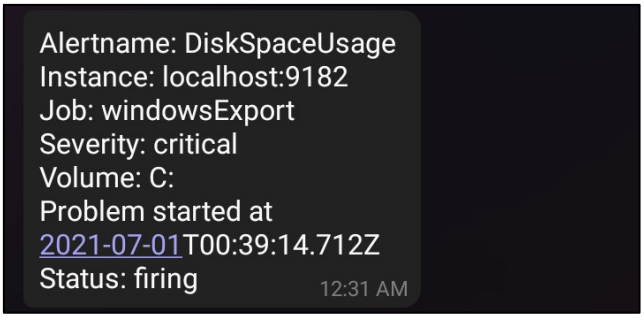


Figura 210. Notificación de alerta DiskSpaceUsage (Prometheus)

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- En el presente trabajo de titulación se ha logrado desarrollar varias consolas que unifican distintas herramientas de monitoreo de TI, para supervisar las actividades que se realizan en la infraestructura y ecosistema de una empresa. Con estas consolas, se pretende aumentar la disponibilidad y seguridad de los distintos componentes de una infraestructura de TI.

- Se escogió herramientas open source debido a que son de mayor calidad, mayor confiabilidad, más flexibilidad y de menores costos en comparación a las herramientas propietarias. La reducción en los costos de implementación es importante para empresas pequeñas que no poseen una gran capacidad económica.

- Una vez realizado el análisis de las distintas herramientas de monitoreo open source, se escogió tres herramientas Zabbix, Prometheus y ELK Stack. Zabbix se escogió por su gran alcance, la cantidad de documentación y curva de aprendizaje que no es tan grande como otras herramientas. Prometheus se escogió ya que es el actual estándar de monitoreo en la nube, la comunidad que lo desarrolla es muy activa y posee una buena documentación. Ya que Prometheus no está optimizada para el manejo de logs se complementó esta herramienta con el Stack ELK, el cual es un conjunto de herramientas que permite administrar logs.

- Antes de comenzar con el despliegue de las consolas de monitoreo se estableció los distintos parámetros a supervisar y se definió el esquema con el que se van a recolección los datos. Se lo realizó en dos ambientes uno simulado con varios dispositivos y servicios, y en un entorno real con dos servidores de producción de una empresa de desarrollo de software.

- Una vez implementado las consolas de monitoreo, se realizó varias pruebas de funcionamiento. Se ejecuto varias pruebas de estrés, además de apagar manualmente varios servicios, y se verifico que las consolas detecten los niveles

anormales de actividad en la infraestructura y la falta de respuesta de los servicios. Se obtuvieron resultados satisfactorios ya que desde la consola se pudo observar todos estos eventos en forma de gráficos, reportes y notificaciones por los medios configurados.

- No se encontró ningún problema critico en los dos servidores de producción, aun así, es recomendable que se actualice el dispositivo de almacenamiento a un disco duro de mayor capacidad y a una tecnología más rápida como un SSD.

- Uno de los aspectos más importantes para evaluar en un sistema de monitoreo es su capacidad de alerta. El sistema de alerta debe ser lo suficientemente flexible como para notificar a los operadores a través de múltiples medios y lo suficientemente potente como para poder crear disparadores de notificación reflexivos y procesables.

- La capacidad de mostrar resúmenes de alto nivel y solicitar más detalles a pedido es una característica importante para garantizar que los datos de métricas sean útiles y consumibles para los operadores humanos. El diseño de paneles que presenten los datos que se ven con mayor frecuencia ayuda a los usuarios a comprender el estado del sistema de un vistazo.

- Un sistema de monitoreo es más útil cuando se tiene un rico historial de datos que puede ayudar a establecer tendencias, patrones y consistencias a lo largo de líneas de tiempo prolongadas. Lo óptimo sería tener la información almacenada con la granularidad original, sin embargo, esto puede consumir muchos recursos de almacenamiento por lo que es recomendable almacenar las métricas más antiguas con una resolución reducida.

- El monitoreo es la base para la medición de la gestión del nivel de servicio (SLM), la cual sirve para garantizar que todos los acuerdos de nivel de servicio (SLA) y contratos de apoyo (UC) se cumplan.

5.2. Recomendaciones

- Las alertas innecesarias hacen que las personas dejen de confiar en los sistemas de monitoreo y comiencen a ignorarlos. Por esto, se debe definir cuidadosamente las alertas que requieran la intervención de personal capacitado.

- Si la acción que es necesaria ejecutar cuando ocurre un problema son una serie de pasos bien conocidos y documentados. La automatización es una gran solución para solucionar el problema y evitar la fatiga de las alertas.

- Un sistema de monitoreo debe ser externo a otros servicios. Debido a la relación con los sistemas observados y su utilidad para diagnosticar problemas, un sistema de monitoreo debe ser accesible de forma independiente de los servicios.

- Ya que inevitablemente el sistema de monitoreo va a afectar el rendimiento de los sistemas monitoreados, se debe tratar de mantener al mínimo el impacto del seguimiento en el rendimiento.

- Un buen punto para comenzar con el monitoreo de una aplicación es el usuario, es decir, los puntos en los que los usuarios interactúan con la aplicación. Ya que produce un punto de vista más eficaz.

- Es recomendado usar herramienta de monitorización ya construidas en lugar de crear una desde cero. Ya que crear un sistema propio desde cero puede requerir mucho tiempo y dinero.

- Se debe garantizar que el personal operativo o administrativo de los diferentes componentes de la infraestructura de TI tengan acceso a la información obtenida por los sistemas de monitoreo.

- Es recomendable que todos los departamentos y equipos de una empresa identifiquen que métricas y logs les ayudaría a ejecutar su función correctamente. Esta información debe ser considerada en el diseño de la consola de monitoreo.

- Es recomendable que una vez que la consola de monitoreo este desplegada en componentes que se usen en producción, se gestione respaldos de las bases de datos y archivos de configuración para evitar la pérdida de datos.

- Un sistema de monitoreo debe poder realizar ajustes a medida que cambien las máquinas y la infraestructura. Se debe tener la capacidad de eliminar fácilmente las máquinas dadas de baja sin destruir los datos recopilados asociadas a ellas. El sistema debe hacer que estas operaciones sean simples para fomentar la configuración del monitoreo parte del proceso de aprovisionamiento o retiro de instancias.

Bibliografía

[1] W. Dalton y B. Turner, “Best network monitoring tools of 2021: RMM software for remote monitoring and management”, *TechRadar*. https://www.techradar.com/uk/best/best-network-monitoring-tools (consultado jun. 04, 2021).

[2] C. Figueroa y A. Liseth, “Estudio de la implementación de redundancia para un sistema de monitoreo en empresas de telecomunicaciones.”, 2016, Consultado: jun. 04, 2021. [En línea]. Disponible en: http://www.dspace.uce.edu.ec/handle/25000/5462

[3] “¿Qué es el open source?” https://www.redhat.com/es/topics/open-source/what-is-open-source (consultado nov. 04, 2020).

[4] I. T. M. Cruz y K. I. C. Tandazo, “Incidencia del software libre en el sector comercial del Ecuador: una revisión literaria.”, *Espirales Rev. Multidiscip. Investig.*, vol. 1, núm. 10, Art. núm. 10, nov. 2017, doi: 10.31876/re.v1i10.121.

[5] “ITIL”, *AXELOS*. https://www.axelos.com/best-practice-solutions/itil (consultado jun. 07, 2021).

[6] D. R. Mauro y K. J. Schmidt, *Essential SNMP*, 2nd ed. Sebastopol, CA: O’Reilly, 2005.

[7] “An Introduction to Metrics, Monitoring, and Alerting”, *DigitalOcean*. https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting (consultado nov. 09, 2020).

[8] M. Julian, *Practical monitoring: effective strategies for the real world*, First edition. Beijing: O'Reilly, 2018.

[9] "What is Structured Logging?", *Sumo Logic*. https://www.sumologic.com/glossary/structured-logging/ (consultado may 04, 2021).

[10] "Gathering Metrics from Your Infrastructure and Applications", *DigitalOcean*. https://www.digitalocean.com/community/tutorials/gathering-metrics-from-your-infrastructure-and-applications (consultado may 11, 2021).

[11] S. Dhir y S. Dhir, "Adoption of open-source software versus proprietary software: An exploratory study", *Strateg. Change*, vol. 26, núm. 4, pp. 363–371, jul. 2017, doi: 10.1002/jsc.2137.

[12] "¿Qué es ITIL?" https://advisera.com/20000academy/es/que-es-itil/ (consultado may 13, 2021).

[13] D. Cannon, D. Wheeldon, S. Taylor, y Großbritannien, Eds., *ITIL: IT service management practices; ITIL v3 core publications. 4: Service operation*, 3. impression. London: TSO, 2010.

[14] "2 What is Zabbix [Zabbix Documentation 5.0]". https://www.zabbix.com/documentation/5.0/manual/introduction/about (consultado may 18, 2021).

[15] W. Kocjan y P. Beltowski, *Learning Nagios*. 2016. Consultado: may 18, 2021. [En línea]. Disponible en: http://sbiproxy.uqac.ca/login?url=https://international.scholarvox.com/book/88843369

[16] J. M. Kretchmar, *Open source network administration*. Upper Saddle River, N.J: Prentice Hall Professional Technical Reference, 2004.

[17] "Pandora FMS: Flexible Monitoring System", *SourceForge*. https://sourceforge.net/projects/pandora/ (consultado may 26, 2021).

[18] "Net-SNMP". http://www.net-snmp.org/about/license.html (consultado may 26, 2021).

[19] Prometheus, "Overview | Prometheus". https://prometheus.io/docs/introduction/overview/#architecture (consultado may 26, 2021).

[20] Prometheus, “FAQ | Prometheus”. https://prometheus.io/docs/introduction/faq/#what-license-is-prometheus-released-under (consultado may 26, 2021).

[21] “1 Hosts and host groups [Zabbix Documentation 5.4]”. https://www.zabbix.com/documentation/current/manual/config/hosts (consultado may 26, 2021).

[22] “2 Items [Zabbix Documentation 5.4]”. https://www.zabbix.com/documentation/current/manual/config/items (consultado may 26, 2021).

[23] “3 Triggers [Zabbix Documentation 5.4]”. https://www.zabbix.com/documentation/current/manual/config/triggers (consultado may 26, 2021).

[24] “4 Events [Zabbix Documentation 5.4]”. https://www.zabbix.com/documentation/current/manual/config/events (consultado may 26, 2021).

[25] “1 Graphs [Zabbix Documentation 5.4]”. https://www.zabbix.com/documentation/current/manual/config/visualization/graphs (consultado may 26, 2021).

[26] “8 Templates [Zabbix Documentation 5.4]”. https://www.zabbix.com/documentation/current/manual/config/templates (consultado may 26, 2021).

[27] “10 Notifications upon events [Zabbix Documentation 5.4]”. https://www.zabbix.com/documentation/current/manual/config/notifications (consultado may 26, 2021).

[28] B. Brazil, *Prometheus: up & running: infrastructure and application performance monitoring*, First edition. Sebastopol, CA: O’Reilly Media, 2018.

[29] S. Chhajed, *Learning ELK stack: build mesmerizing visualizations, and analytics from your logs and data using Elasticsearch, Logstash, and Kibana*.

[30] D. Gourley y B. Totty, *HTTP: the definitive guide*, 1st ed. Beijing ; Sebastopol, CA: O’Reilly, 2002.

[31] “Welcome! - The Apache HTTP Server Project”. https://httpd.apache.org/ (consultado may 06, 2021).

[32] Datadog, “Monitoring Apache web server performance”, *Monitoring Apache web server performance*. https://www.datadoghq.com/blog/monitoring-apache-web-server-performance/ (consultado jun. 01, 2021).

[33] “What is a database?” https://www.oracle.com/database/what-is-database/ (consultado may 08, 2021).

[34] A. Silberschatz, H. F. Korth, y S. Sudarshan, *Database system concepts*, Seventh edition. New York, NY: McGraw-Hill, 2020.

[35] Datadog, “Monitoring MySQL performance metrics”, *Monitoring MySQL performance metrics*. https://www.datadoghq.com/blog/monitoring-mysql-performance-metrics/ (consultado jun. 01, 2021).

[36] J. B. Postel, “Simple Mail Transfer Protocol”, RFC Editor, STD 10, ago. 1982. [En línea]. Disponible en: http://www.rfc-editor.org/rfc/rfc821.txt

[37] T. Dean, *Network+ guide to networks*, 5th ed. Boston, Mass: Course TechnologyCengage Learning, 2010.

[38] D. Crocker, “Internet Mail Architecture”, RFC Editor, RFC 5598, jul. 2009. [En línea]. Disponible en: https://datatracker.ietf.org/doc/html/rfc5598

[39] K. D. Dent, *Postfix: the definitive guide*, 1st ed. Sebastopol, CA: O’Reilly, 2004.

[40] “Postfix stable release 3.2.5, and legacy releases 3.1.8, 3.0.12, and 2.11.11”. http://www.postfix.org/announcements/postfix-3.2.5.html (consultado may 27, 2021).

[41] “Dovecot manual — Dovecot documentation”. https://doc.dovecot.org/ (consultado may 27, 2021).

ANEXOS

Anexo 1: Instalación y configuración de servidor Zabbix

Se adjunto el documento en forma digital.

Enlace de Descarga: https://epnecuador-my.sharepoint.com/:w:/g/personal/diego_portero_epn_edu_ec/ET0SSOb32GJJkweyWwcZbjgBlcr3LFj7YI8aVZ3AXhrBZw?e=KhbcX4

Anexo 2: Instalación y configuración de DNS y Postfix

Se adjunto el documento en forma digital.

Enlace de Descarga: https://epnecuador-my.sharepoint.com/:w:/g/personal/diego_portero_epn_edu_ec/Edj_zsRox5NNpdza9seEmsoBi-ZIjKJ8VvrEjxaJNineyg?e=X4f89G