WAZUH

I.E.S. Gonzalo Nazareno
Jonathan Márquez Jiménez
Proyecto Fin de Ciclo

Contenido

Contenido

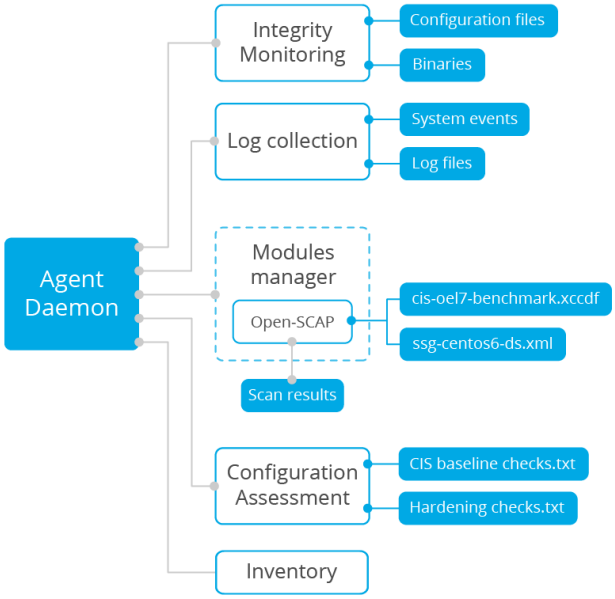
¿QUE ES WAZUH?

Wazuh proporciona una solución de seguridad capaz de monitorear su infraestructura, detectar amenazas, intentos de intrusión, anomalías del sistema, aplicaciones mal configuradas y acciones de usuarios no autorizados.

AGENTE DE WAZUH

El agente ligero de Wazuh está diseñado para realizar una serie de tareas con el objetivo de detectar amenazas y, cuando sea necesario, activar respuestas automáticas. Las capacidades principales del agente son:

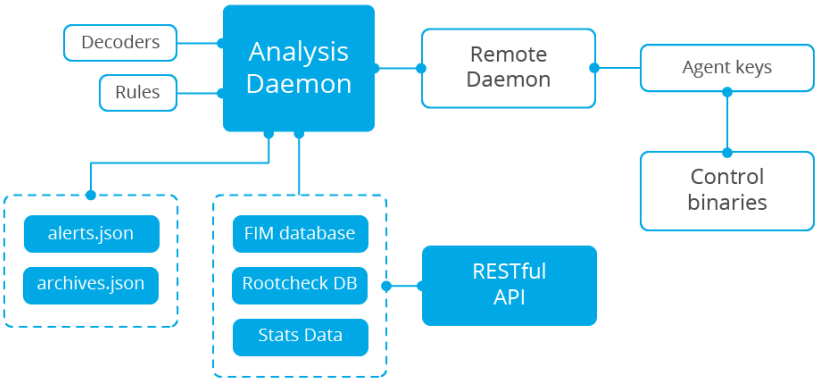
- Recopilación de datos de registros y eventos
- Supervisión de la integridad de las claves de registro y archivos
- Inventario de procesos en ejecución y aplicaciones instaladas
- Monitoreo de puertos abiertos y configuración de red.
- Detección de rootkits o artefactos de malware
- Evaluación de la configuración y seguimiento de políticas
- Ejecución de respuestas activas



SERVIDOR WAZUH

El servidor de Wazuh se encarga de analizar los datos recibidos de los agentes, procesar eventos a través de decodificadores y reglas, y usar inteligencia de amenazas para buscar IOC (Indicadores de Compromiso) conocidos. Un solo servidor de Wazuh puede analizar datos de cientos o miles de agentes.

El servidor también se utiliza para gestionar los agentes, configurándolos y actualizándolos de forma remota cuando sea necesario.



ELASTIC STACK

Las alertas generadas por Wazuh se envían a Elastic Stack, donde se indexan y almacenan. La integración única entre Wazuh y Kibana, proporciona una poderosa interfaz de usuario para la visualización y el análisis de datos, que también se puede utilizar para administrar y monitorear la configuración y el estado de los agentes.

La interfaz de usuario web de Wazuh incluye paneles de control listos para usar para el cumplimiento normativo (por ejemplo, PCI DSS, GDPR, CIS), aplicaciones vulnerables detectadas, monitoreo de integridad de archivos, evaluación de configuración, eventos de seguridad, monitoreo de infraestructura en la nube y otros.

CREACION DEL ESCENARIO PARA WAZUH

Nos descargamos el wazuh del repositorio oficial
git clone https://github.com/wazuh/wazuh-docker.git -b v4.2.4 --depth=1

Dentro encontraremos dos entornos, el “demo” (que es el docker-compose.yml) y el de “producción” (que es el production-cluster.yml), en este caso usaremos el “demo” ya que trae menos contenedores y será más fácil de explicar

En wazuh demo encontraremos varios contenedores:
- wazuh-odfe:4.2.4: Es el encargado de analizar
- opendistro-for-elasticsearch: Sera el encargado de pasar los datos a kibana
- wazuh-kibana-odfe:4.2.4: Sera nuestro panel para visualizar los datos recibidos

Ejecutamos el siguiente comando para aumentar la memoria ya que si no lo hacemos elasticsearch no funcionaría correctamente
sysctl -w vm.max_map_count=262144

Para hacerlo permanente el cambio
echo "vm.max_map_count=262144" >> /etc/sysctl.conf

Para iniciar el entorno “demo”
docker-compose up

Para iniciar el entorno de “producción”
docker-compose -f production-cluster.yml up

CONFIGURACIÓN DE WAZUH MANAGER PARA QUE RECIBA LOG POR EL PUERTO 514/UDP

Lo primero es irnos a la configuración del wazuh manager y dentro debajo de la conexión que ya está creada para los agentes pondremos el siguiente código
sudo nano /var/lib/docker/volumes/wazuh-docker_ossec_etc/_data/ossec.conf
……

```
<!-- Puerto para el agente -->
  <remote>
    <connection>secure</connection>
    <port>1514</port>
    <protocol>tcp</protocol>
    <queue_size>131072</queue_size>
  </remote>
<!-- Puerto que tendremos que habilitar para los log de los contenedores -->
  <remote>
    <connection>syslog</connection>
    <port>514</port>
    <protocol>udp</protocol>
    <allowed-ips>0.0.0.0/0</allowed-ips>
  </remote>
```

……

En el código que pusimos podemos apreciar:
- Tipo de conexión: <connection>syslog</connection>
- Puerto de escucha: <port>514</port>
- Protocolo a usar: <protocol>udp</protocol>
- Rango de IP permitidas, en mi caso todas las IP: <allowed-ips>0.0.0.0/0</allowed-ips>

NOTA: Los archivos están todos montados en volúmenes especificados en el docker-compose
IMPORTANTE: En algunas versiones de sistemas operativos no puedes usar el puerto 514, en su lugar yo e usado el 11514, eso lo especificamos en el docker-compose

Destruimos los contenedores y lo montamos de nuevo para efectuar los cambios
docker-compose down
docker-compose up

CONFIGURAR LOS CONTENEDORES PARA MANDAR LOS LOGS A WAZUH

Creare un docker-compose.yml donde crearemos un contenedor con un servicio web para enviar los logs a wazuh

nano docker-compose.yml

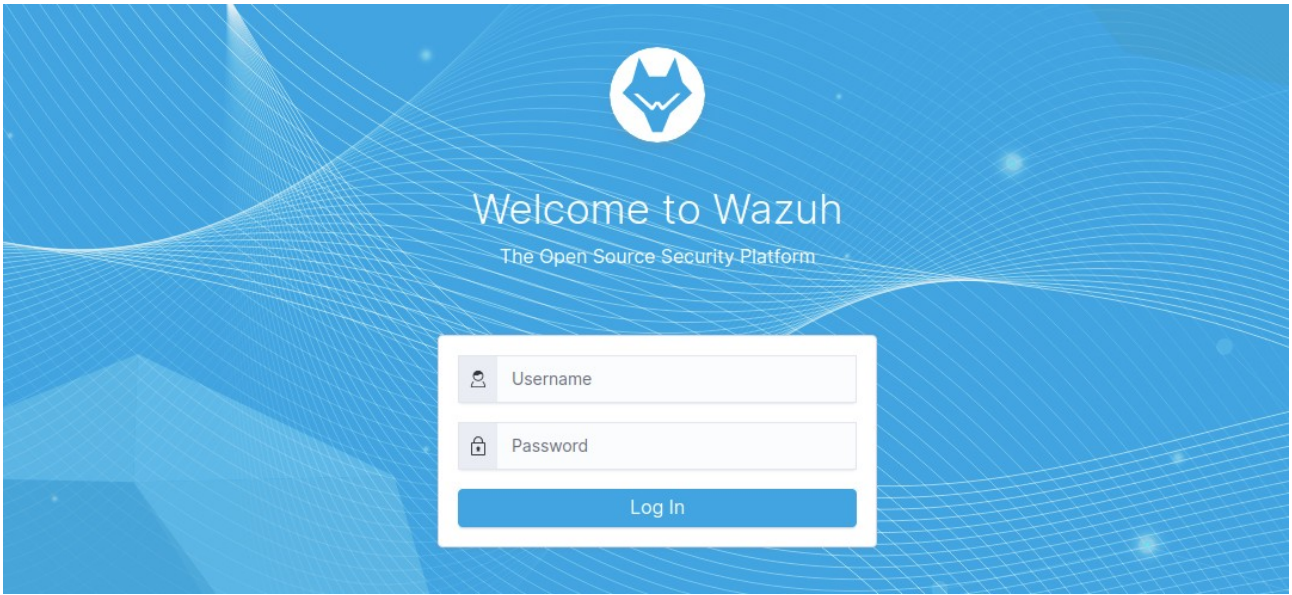
```
version: '3.7'

services:
  miservicio_php:
    image: php:7-apache
    volumes:
      # Montamos nuestra web desde fuera en el directorio web del contenedor
      - ./miweb/:/var/www/html
    ports:
      - "8080:80"
    logging:
      driver: syslog
      options:
        syslog-address: "udp://localhost:514"
        tag: "{{.ImageName}}/{{.Name}}/{{.ID}}"
```

Como se puede apreciar, las ultimas líneas de código es donde se especifica él envió de los logs a wazuh, donde le ponemos
- Tipo: syslog
- IP y protocolo donde enviar los logs: udp://localhost:514
- Etiquetas que ponemos para identificar el contenedor del log: {{.ImageName}}/{{.Name}}/{{.ID}}

Y por último levantamos el contenedor
docker-compose up

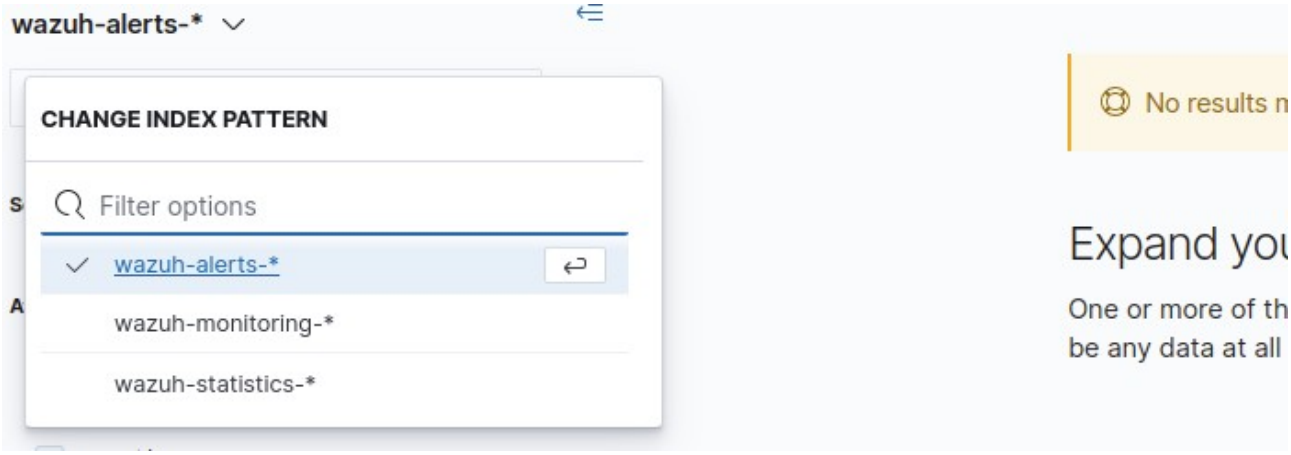
VISUALIZAR LAS ALERTAS EN KIBANA

Para entrar en el panel ponemos en el navegador https://localhost ya que kibana está usando el puerto 443 que es el de https
Usuario y contraseña son “admin”



En el menú le daremos a “Wazuh → Wazuh” para que cargue los módulos si es que no lo ha hecho ya

Después nos iremos a “Kibana → Discover” donde nos aparecerá los logs, dentro le daremos al index de alertas de wazuh



A la derecha saldrá la tabla de las alertas, en el servidor web he puesto un valid-user en él .htaccess, de tal manera que tendremos que poner un usuario y una contraseña para ver la web, en caso de ponerlos mal nos saldrá una alerta en wazuh

Y si le damos a la alerta para ver más información

_index	wazuh-alerts-4.x-2021.10.28	
	agent.id	000
	agent.name	wazuh-manager
	data.id	500
	data.protocol	GET
	data.srcip	192.168.16.1

data.srcip2	php:7-apache/cliente_miservicio_php_1/f67645878204[1615]:
data.url	/favicon.ico
decoder.name	web-accesslog
decoder.parent	web-accesslog
full_log	Oct 28 12:52:01 php:7-apache/cliente_miservicio_php_1/f67645878204[1615]: 192.168.16.1 - asd [28/Oct/2021:10:52:01 +0000] "GET /favicon.ico HTTP/1.1" 500 799 "http://localhost/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36"
id	1635418321.51645
input.type	log
location	192.168.100.21
manager.name	wazuh-manager
predecoder.hostname	php:7-apache/cliente_miservicio_php_1/f67645878204[1615]:
predecoder.timestamp	Oct 28 12:52:01
rule.description	Web server 500 error code (Internal Error).
rule.firedtimes	8
rule.groups	web, accesslog, system_error
rule.id	31122
rule.level	5
rule.mail	false
timestamp	Oct 28, 2021 @ 12:52:01.457

En la tabla nos sale información del error, como
- Imagen, nombre e ID del contenedor: predecoder.hostname: php:7-apache/cliente_miservicio_php_1/f67645878204[1615]:
- Nivel de la alerta: rule.level: 5
- Descripción: rule.description: Web server 500 error code (Internal Error).

ANALIZAR LOG DIRECTAMENTE

Si queremos preguntarle directamente al contenedor por un log que tenemos porque no sabemos si ha saltado la alerta o lo ha ignorado, lo primero es entrar en el contenedor
docker exec -ti wazuh-docker_wazuh_1 bash

Y dentro ejecutamos el siguiente comando
echo '<log para analizar>' | /var/ossec/bin/wazuh-logtest

Ejemplo

[root@wazuh-manager /]# echo '192.168.16.1 - asd [28/Oct/2021:10:52:01 +0000] "GET /favicon.ico HTTP/1.1" 500 799 "http://localhost/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36"' | /var/ossec/bin/wazuh-logtest

Starting wazuh-logtest v4.2.4
Type one log per line

**Phase 1: Completed pre-decoding.
full event: '192.168.16.1 - asd [28/Oct/2021:10:52:01 +0000] "GET /favicon.ico HTTP/1.1" 500 799 "http://localhost/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36"'

**Phase 2: Completed decoding.
name: 'web-accesslog'
id: '500'
protocol: 'GET'
srcip: '192.168.16.1'
url: '/favicon.ico'

**Phase 3: Completed filtering (rules).
id: '31122'
level: '5'
description: 'Web server 500 error code (Internal Error).'
groups: '['web', 'accesslog', 'system_error']'
firedtimes: '1'
mail: 'False'
**Alert to be generated.

Al final de la línea podemos apreciar que ha generado la alerta (**Alert to be generated.)

INSTALACIÓN DE AGENTE DE WAZUH EN EL SISTEMA

El agente lo podemos instalar siguiendo los pasos de la web oficial https://documentation.wazuh.com/current/installation-guide/wazuh-agent/index.html

En linux seria de la siguiente manera
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -

echo "deb https://packages.wazuh.com/4.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list

apt-get update

Y lo iniciamos con el siguiente comando, teniendo en cuenta que la IP que ponemos seria la IP del servidor de wazuh ya que el agente enviara la información a dicho servidor, en mi caso está todo en local

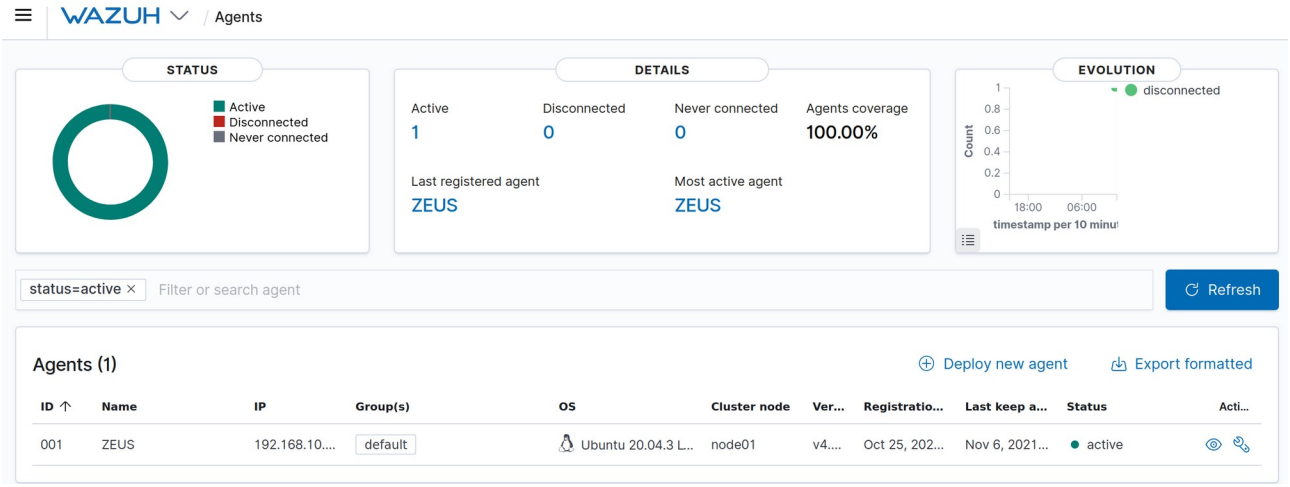
WAZUH_MANAGER="localhost"
apt-get install wazuh-agent

Mas adelante lo podemos modificar en el archivo “/var/ossec/etc/ossec.conf”

Y por último le decimos que se inicie con el sistema y lo iniciamos
systemctl daemon-reload
systemctl enable wazuh-agent
systemctl start wazuh-agent

Es recomendable quitar las actualizaciones del agente, ya que puedes tener problemas en el futuro para posibles cambios
sed -i "s/^deb/#deb/" /etc/apt/sources.list.d/wazuh.list
apt-get update

Cuando lo ejecutemos en el panel del servidor de Wazuh podremos ver que se ha conectado el agente



MONITORIZAR LOS ESTADOS Y LA EJECUCIÓN DE COMANDOS DE LOS CONTENEDORES

IMPORTANTE: Para monitorizar la ejecución de comandos en los contenedores es necesario el uso del agente del wazuh

Necesitaremos instalar algunos paquetes
apt install python3-pip libffi-dev libxml2-dev libxslt1-dev libssl-dev
pip install docker
apt install python-is-python3
python -m easy_install --upgrade pyOpenSSL

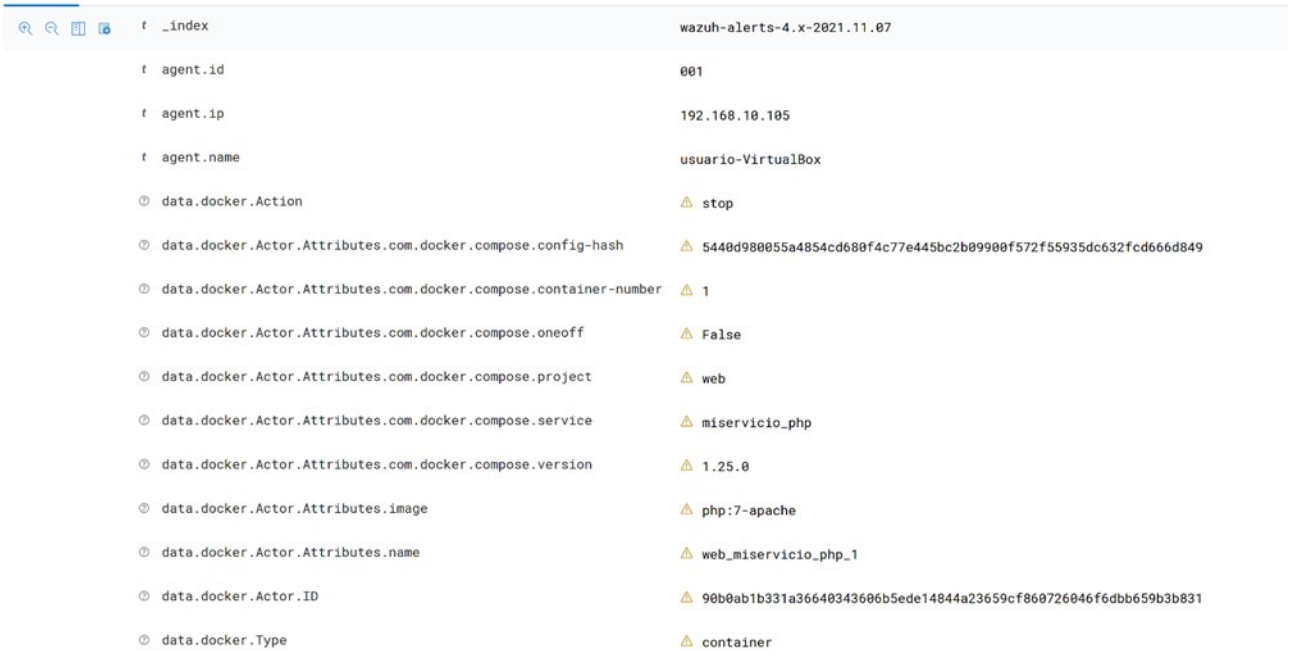
Editaremos el archivo “ossec.conf” del agente y le pondremos los siguientes parámetros
nano /var/ossec/etc/ossec.conf
....

```
  <wodle name="docker-listener">
    <disabled>no</disabled>
  </wodle>
```

….

Por último, reiniciamos el agente
/etc/init.d/wazuh-agent restart

Si nos dirigimos al panel de kibana podemos ver las alertas de los estados de los contenedores “Menu -> Kibana -> Discover”

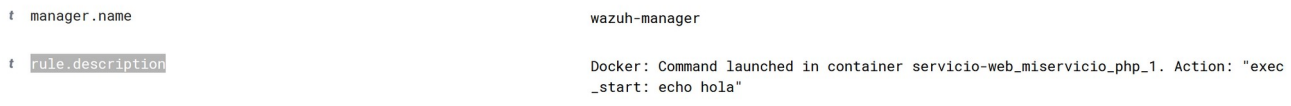


t _index	wazuh-alerts-4.x-2021.11.07
t agent.id	001
t agent.ip	192.168.10.105
t agent.name	usuario-VirtualBox
ⓘ data.docker.Action	⚠ stop
ⓘ data.docker.Actor.Attributes.com.docker.compose.config-hash	⚠ 5440d980055a4854cd680f4c77e445bc2b09900f572f55935dc632fcd666d849
ⓘ data.docker.Actor.Attributes.com.docker.compose.container-number	⚠ 1
ⓘ data.docker.Actor.Attributes.com.docker.compose.oneoff	⚠ False
ⓘ data.docker.Actor.Attributes.com.docker.compose.project	⚠ web
ⓘ data.docker.Actor.Attributes.com.docker.compose.service	⚠ miservicio_php
ⓘ data.docker.Actor.Attributes.com.docker.compose.version	⚠ 1.25.0
ⓘ data.docker.Actor.Attributes.image	⚠ php:7-apache
ⓘ data.docker.Actor.Attributes.name	⚠ web_miservicio_php_1
ⓘ data.docker.Actor.ID	⚠ 90b0ab1b331a36640343606b5ede14844a23659cf860726046f6dbb659b3b831
ⓘ data.docker.Type	⚠ container

En el listado podemos apreciar

- data.docker.Action: Es estado al que paso el contenedor
- agent.name: El usuario que realizo la acción
- data.docker.Actor.Attributes.name: Nombre del contenedor
- data.docker.from: Imagen del contenedor

En el caso de que ejecutemos un comando en un contenedor nos aparecería de la siguiente manera, teniendo en cuenta el campo “rule.description” que es el comando a ejecutar



t manager.name	wazuh-manager
t rule.description	Docker: Command launched in container servicio-web_miservicio_php_1. Action: "exec_start: echo hola"

INTEGRIDAD DE LOS FICHEROS

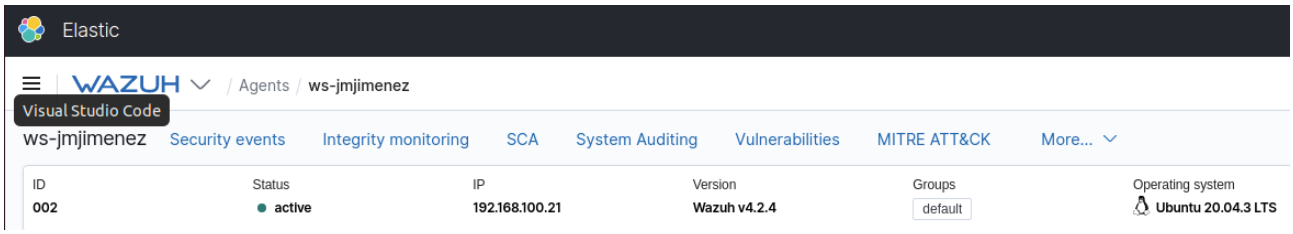
Para monitorizar un directorio tendremos que editar el “ossec.conf” del agente y le especificamos el directorio a monitorizar

sudo nano /var/ossec/etc/ossec.conf
<!-- File integrity monitoring -->
….
….
<!-- ESCANEAR LOS ARCHIDOS -->

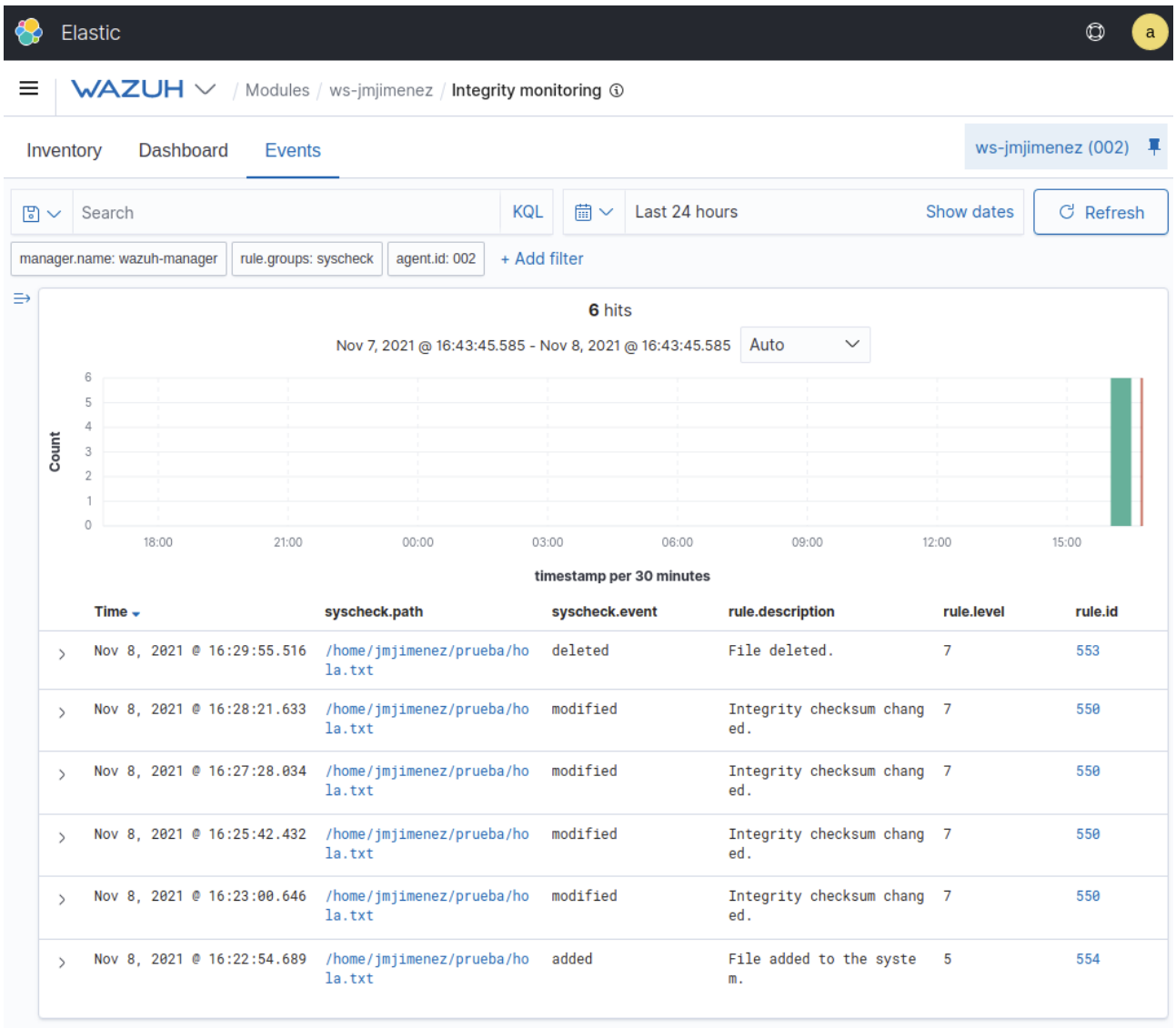
```
  <syscheck>
                                        <directories        check_all="yes"
realtime="yes">/home/jmjimenez/prueba</directories>
  </syscheck>
....
```

Le especificamos varios parámetros en los cuales encontramos
- check_all: Le decimos que monitorice el tamaño del archivo, permisos, propietario, fecha de última modificación, inodo y todas las sumas de hash
- realtime: Este atributo solo funciona con los directorios y no con los archivos individuales, es para que escanee el directorio en tiempo real
- /home/jmjimenez/prueba: Ruta a analizar

Para verlo en el panel nos tendremos que ir al agente y le damos “Integrity monitoring”



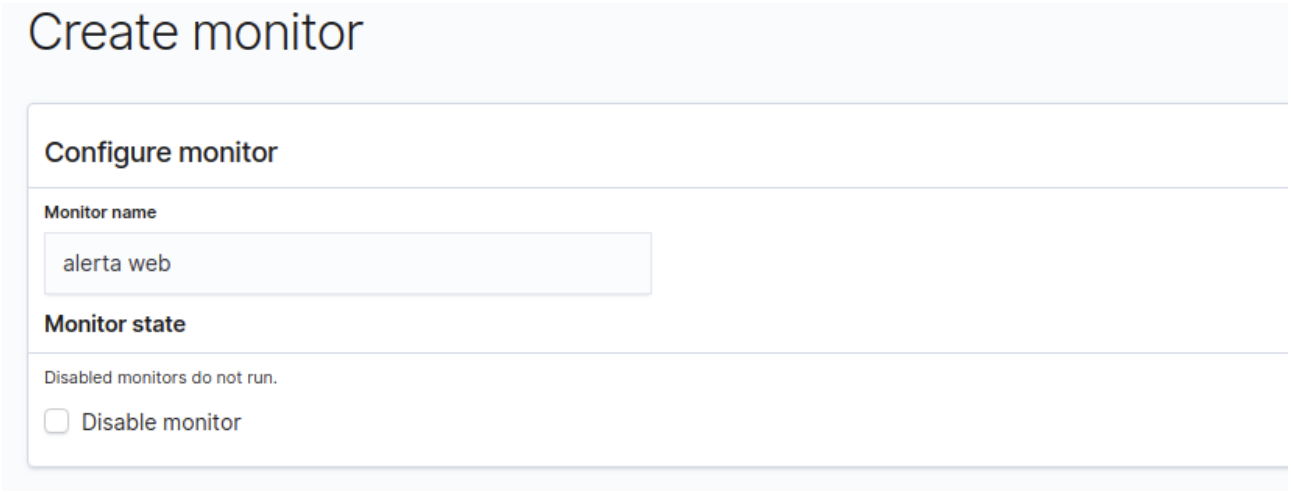
Dentro nos saldrá unos gráficos, pero si queremos ver los registros le podemos dar a “Events”



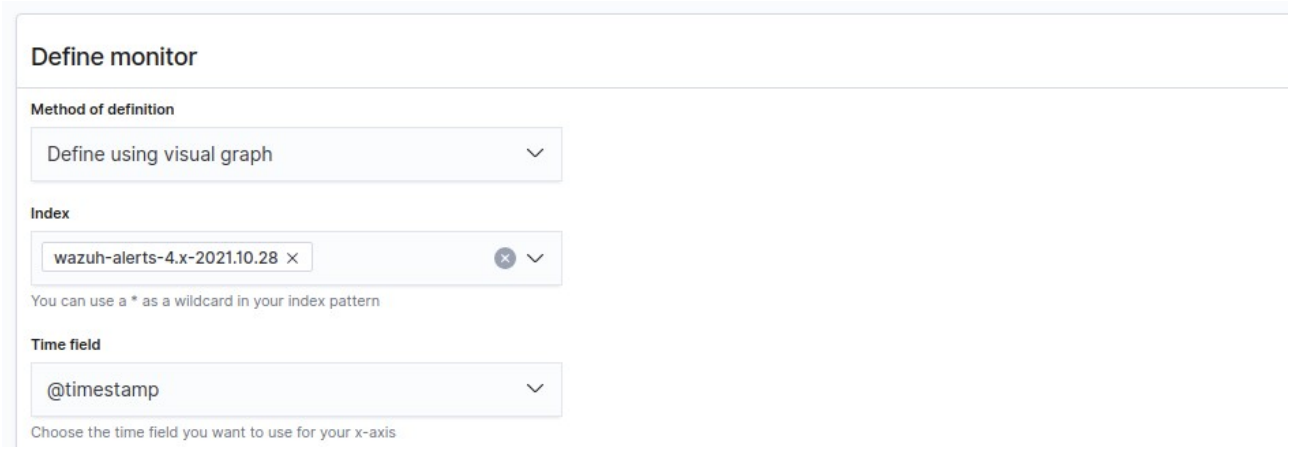
Con esto ya nos monitorizaría el directorio en tiempo real con el agente de wazuh en Linux

CREAR ALERTA EN KIBANA

Crearemos una alerta en kibana para que nos notifique de alguna acción importante, para crearlo nos dirigimos a “Open Distro for Elasticsearch → Alerting”
Lo primero es crear la gráfica para monitorizar, le damos a “Create monitor”
Nos pedirá algunos datos

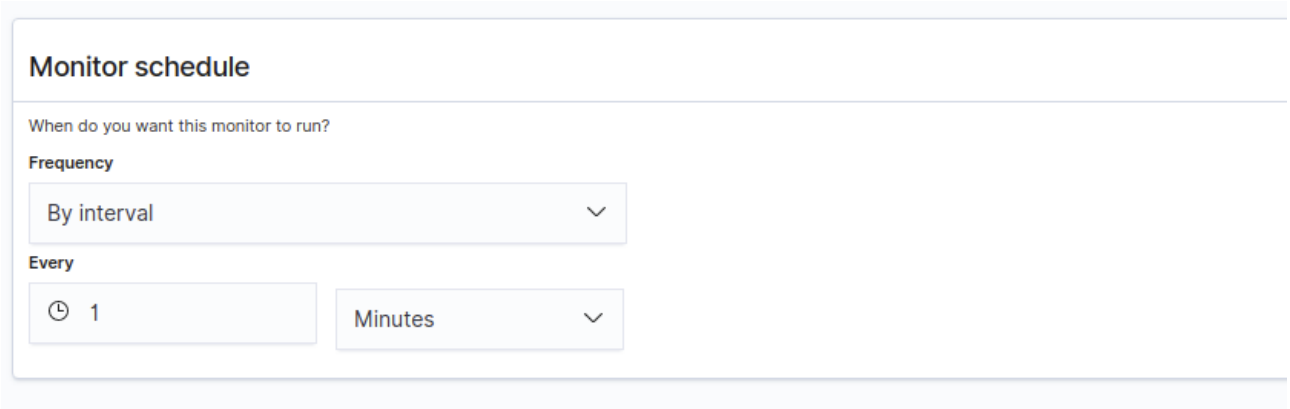


A la hora de definir el monitor le decimos que sea de tipo grafico “Define using visual graph”
En el Index le pondremos donde se genera la o las alerta a monitorizar en mi caso “wazuh-alerts-4.x-2021.10.28”
Le decimos que el grafico que tome como referencia el tiempo “@timestamp”



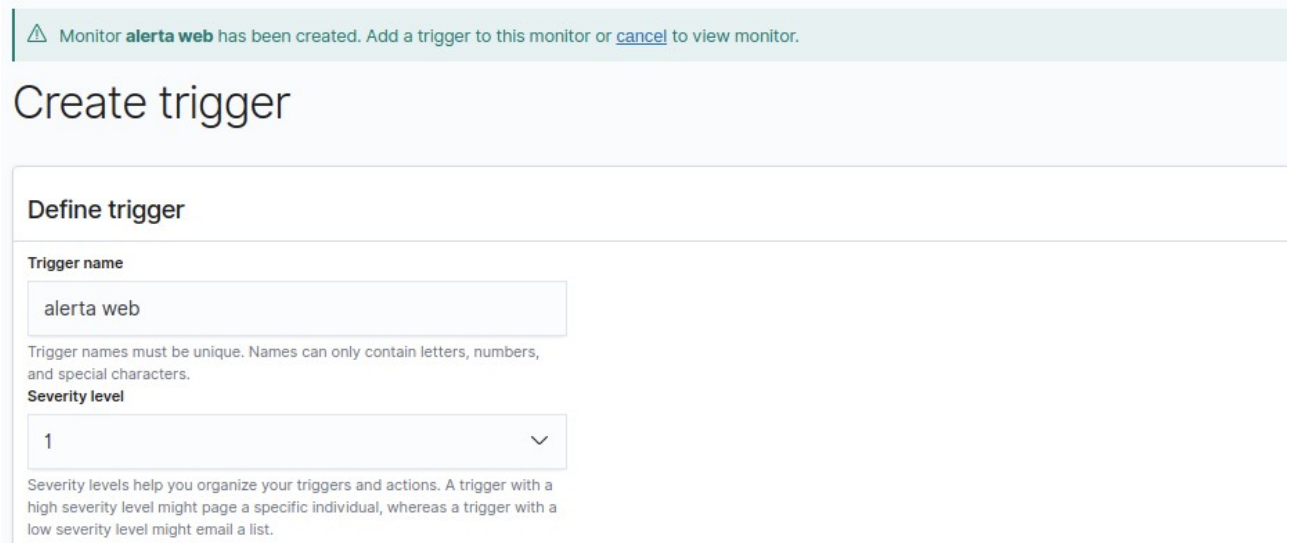
Le decimos la reglas del monitor, queremos que cuente las alertas “WHEN count()”
La franja de hora para mostrar será de 1 hora “FOR THE LAST 1 hour(s)”
Y por último le decimos que el log en el tag “precoder.hostname” tiene que contener “cliente_miservicio_php₁” que quedaría de la siguiente manera “precoder.hostname contains cliente_miservicio_php₁”

Por último, le decimos que el monitorio se ejecute en intervalos “By interval” de 1 minuto

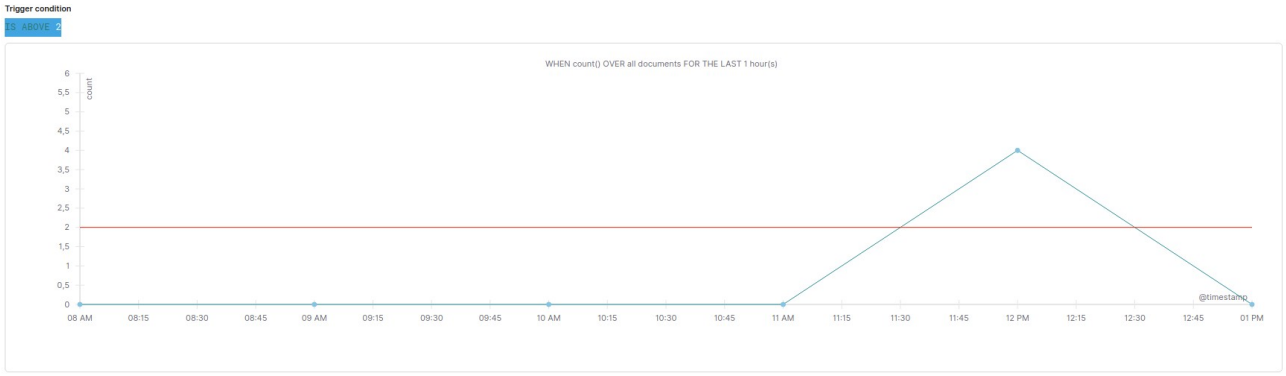


Cuando lo tengamos le damos a crear y en la siguiente página que nos sale es donde configuramos la alerta

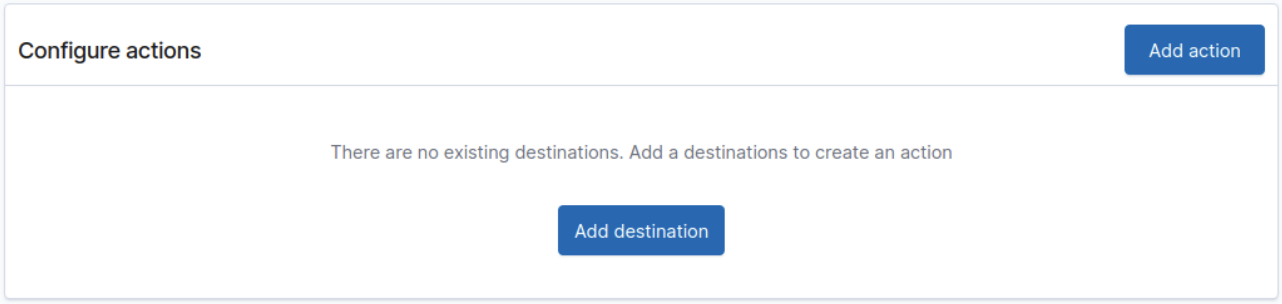
Lo primero es el nombre y después el nivel de la alerta, siendo 5 el nivel alto de gravedad y 1 el más bajo



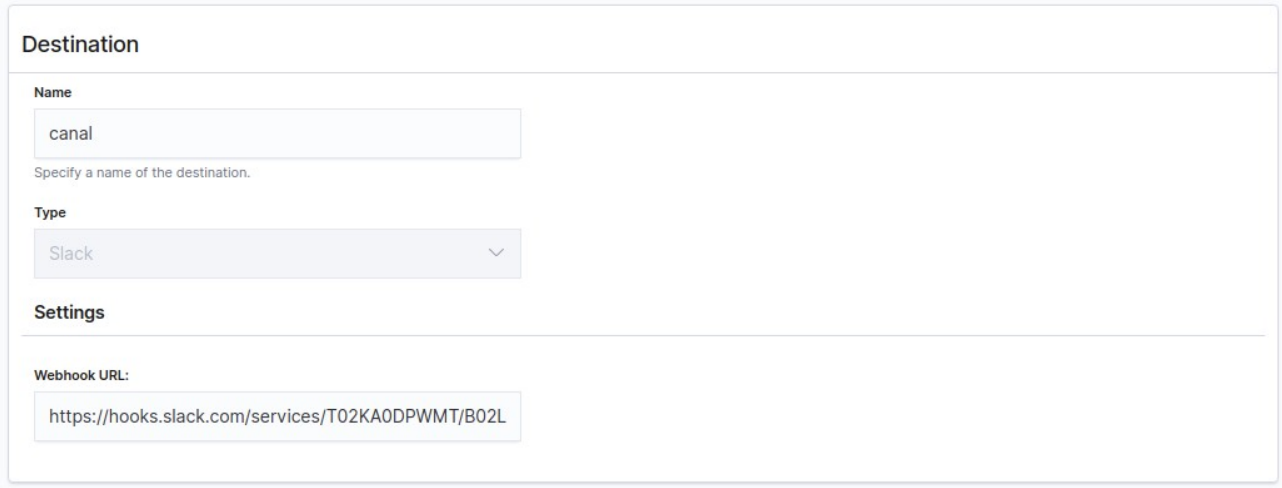
Le decimos que si el contador del grafico supera el 2 que se ejecute la alerta “IS ABOVE 2”



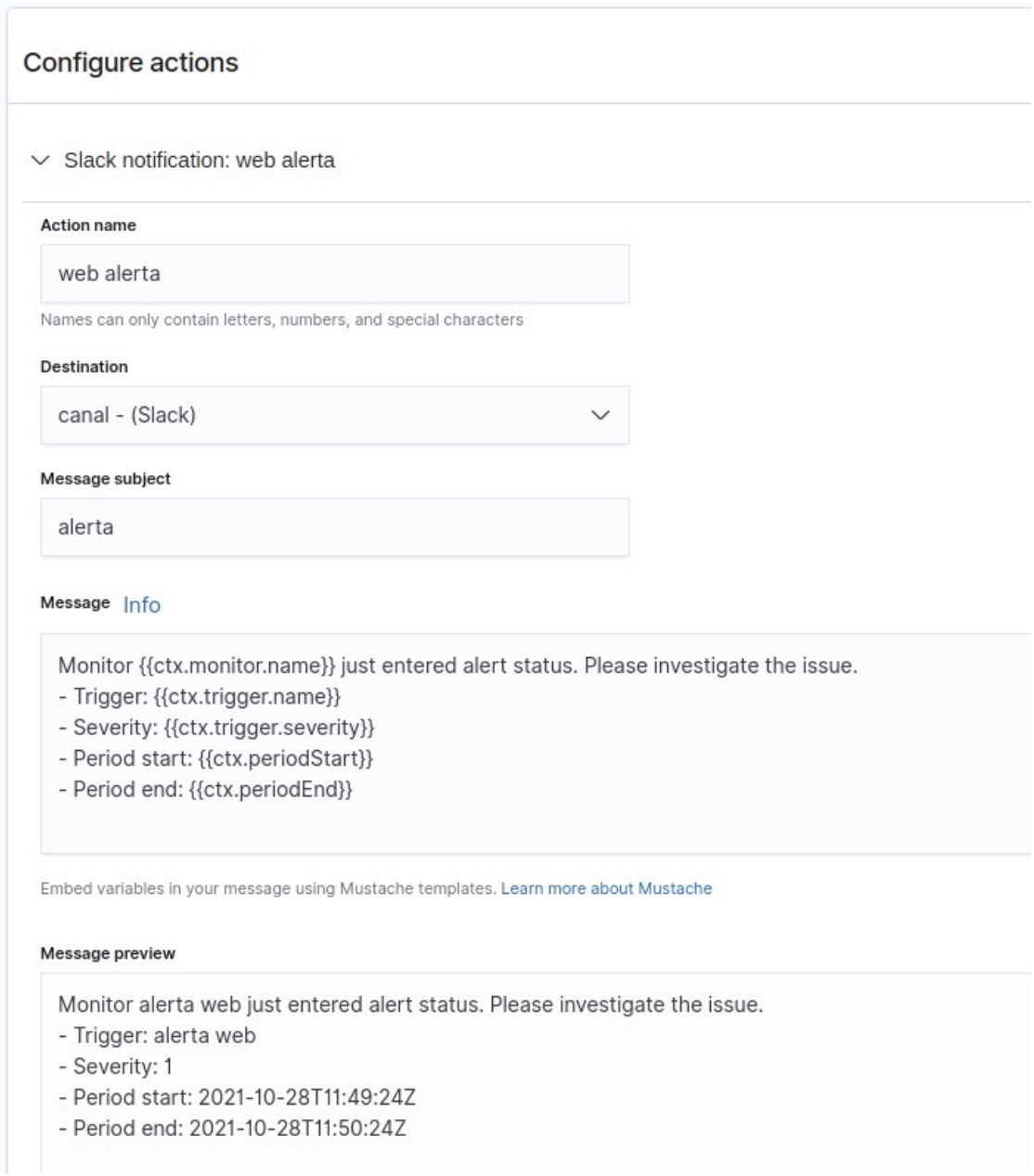
Podemos configurar el método por el cual nos avisa de la alerta, para ello debajo encontraremos la configuración de la acción a ejecutar cuando salta la alarma, lo primero es crear el canal del aviso le damos a “Add destination”



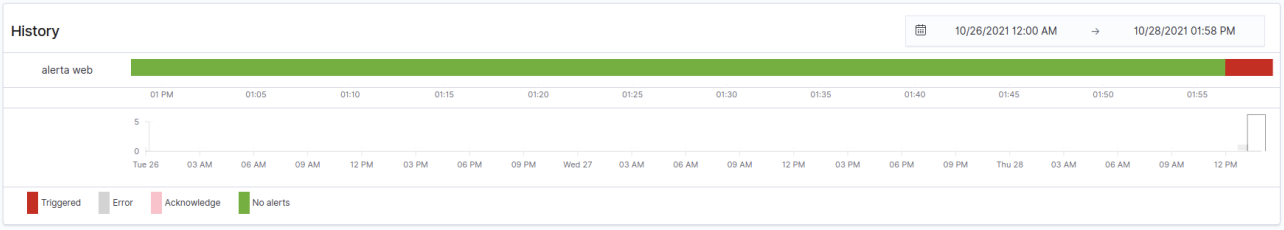
En este caso los mensajes nos llegasen por la aplicación de Slack, para conseguir el “Webhook URL” podéis seguir esta guía https://slack.com/intl/es-es/help/articles/115005265063-Webhooks-entrantes-para-Slack



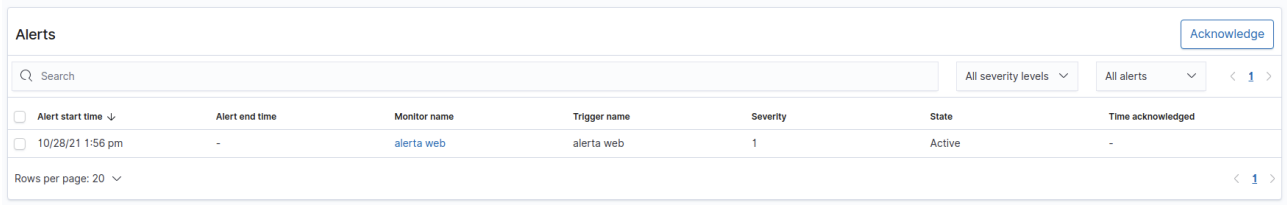
Cuando ya tengamos el canal por el cual avisaremos al usuario, toca crear la acción, le daremos “Add action” y rellenamos los datos



Pasamos a generar la alerta, nos dirigimos al menú “Open Distro for Elasticsearch → Alerting” y dentro le damos a “Monitors” y en la tabla le damos al monitor que creamos “alerta web”
Debajo nos saldrá un pequeño historial en el cual si nos fijamos se a ejecutado una alerta



Y más abajo encontramos la tabla donde nos dice que se ha ejecutado la alerta



Para ver si es cierto, nos podemos ir a Slack y ver si nos ha avisado