

Módulo de Herramientas de monitoreo

Práctico 3

Vamos instalar el sistema de alertas de prometheus y agregar reglas y notificaciones a la suite de prometheus

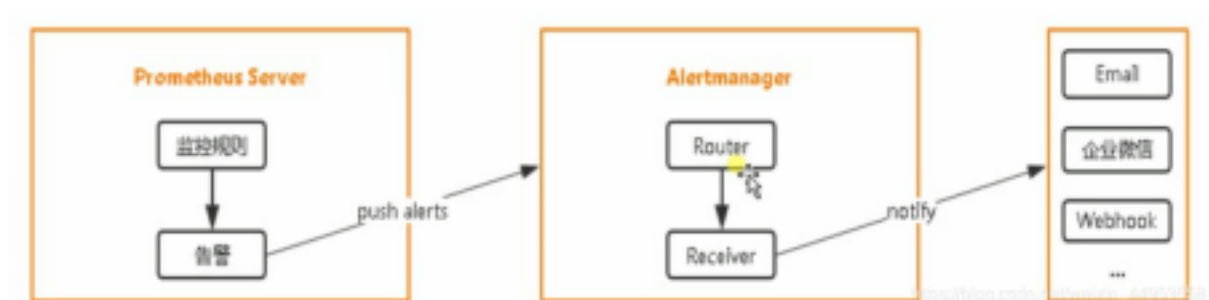
Requisito Previo

Tener configurado Grafana, NodeExport y Prometheus según el practico 2

Configuración de alertas

Prometheus en sí no tiene la capacidad de generar alarmas, por lo que debe combinarse con un programa de alarma de terceros para lograr las alarmas del indicador de monitoreo **AlertManager** es un buen programa de alerta.

Primero, configurar en prometheus las reglas de alerta. Cuando se activa la regla de alerta, se enviará la información de alerta al altermanager. Después de que AlertManager reciba la alerta, la enviará a diferentes alarmas según la ruta configurada y los diferentes niveles de alerta. Recibir (destinatario), AlertManager puede implementar correo electrónico, Slack y otros destinos



Instalación de Alertmanager

alertmanager

Prometheus Alertmanager  [prometheus/alertmanager](https://prometheus.io/alertmanager/)

0.23.0 / 2021-08-25 Release notes			
File name	OS	Arch	Size
alertmanager-0.23.0.darwin-amd64.tar.gz	darwin	amd64	23.62 MiB
alertmanager-0.23.0.linux-amd64.tar.gz	linux	amd64	23.95 MiB
alertmanager-0.23.0.windows-amd64.zip	windows	amd64	24.40 MiB

Descargamos

› **wget**

<https://github.com/prometheus/alertmanager/releases/download/v0.23.0/alertmanager-0.23.0.linux-amd64.tar.gz>

› **tar xvf alertmanager-0.23.0.linux-amd64.tar.gz**

> **cd alertmanager-0.23.0.linux-amd64**

> **ls -l**

En la carpeta debemos encontrar los siguientes archivos

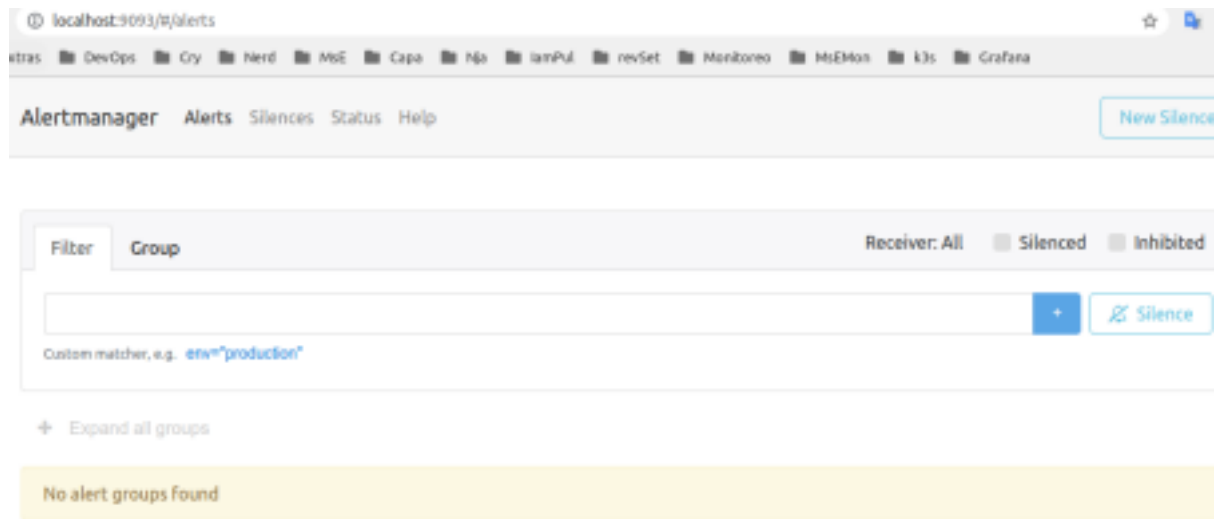
- **amtool**: Utilitario para modificar el estado actual de las alertas, cambiandolas, silenciandolas sin tocar el archivo de configuracion
- **alertmanager**: El ejecutable para iniciar el servicio
- **alertmanager.yml**: Archivo de configuracion con las rutas

Levantamos el servicio

› **./alertmanager --config.file=alertmanager.yml**

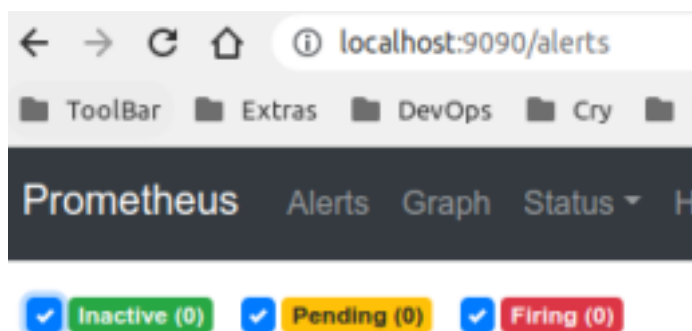
```
> ./alertmanager --config.file=alertmanager.yml
level=info ts=2021-09-13T15:46:04.259Z caller=main.go:225 msg="Starting Alertmanager" version="(version=0.23.0, branch=HEAD, revision=61046b17771a57cf4c4a51be370ab930a4d7d54)"
level=info ts=2021-09-13T15:46:04.259Z caller=main.go:226 build_context="(go=go1.16.7, user=root@e21a959be8d2, date=20210825-10:48:55)"
level=info ts=2021-09-13T15:46:04.268Z caller=cluster.go:184 component=cluster msg="setting advertise address explicitly" address=192.168.0.130 port=9094
level=info ts=2021-09-13T15:46:04.271Z caller=cluster.go:671 component=cluster msg="Waiting for gossip to settle..." interval=2s
level=info ts=2021-09-13T15:46:04.299Z caller=coordinator.go:113 component=configuration msg="Loading configuration file" file=alertmanager.yml
level=info ts=2021-09-13T15:46:04.308Z caller=coordinator.go:126 component=configuration msg="Completed loading of configuration file" file=alertmanager.yml
level=info ts=2021-09-13T15:46:04.303Z caller=main.go:518 msg="Listening address=:9093"
level=info ts=2021-09-13T15:46:04.303Z caller=tls_config.go:191 msg="TLS is disabled." http2=false
level=info ts=2021-09-13T15:46:06.272Z caller=cluster.go:696 component=cluster msg="gossip not settled" polls=8 before=8 now=1 elapsed=2.00066367s
```

Ahora vamos al 9093 a ver el sistema



Editamos el **prometheus.yml** indicando el puerto de alerta
(Ojo reemplazar el dominio alertmanager:9093 por localhost:9093)

```
! prometheus.yml
home > marce > proj > projMsE > labGraf1 > prometheus-2.30.0-rc.0.linux-amd64 > ! prometheus.yml
1 # my global config
2 global:
3   scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
4   evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
5   # scrape_timeout is set to the global default (10s).
6
7 # Alertmanager configuration
8 alerting:
9   alertmanagers:
10    - static_configs:
11      - targets:
12        - alertmanager:9093
13
14 # Load rules once and periodically evaluate them according to the global 'evaluation_interval'
15 rule_files:
16   # - "first_rules.yml"
17   # - "second_rules.yml"
18
```

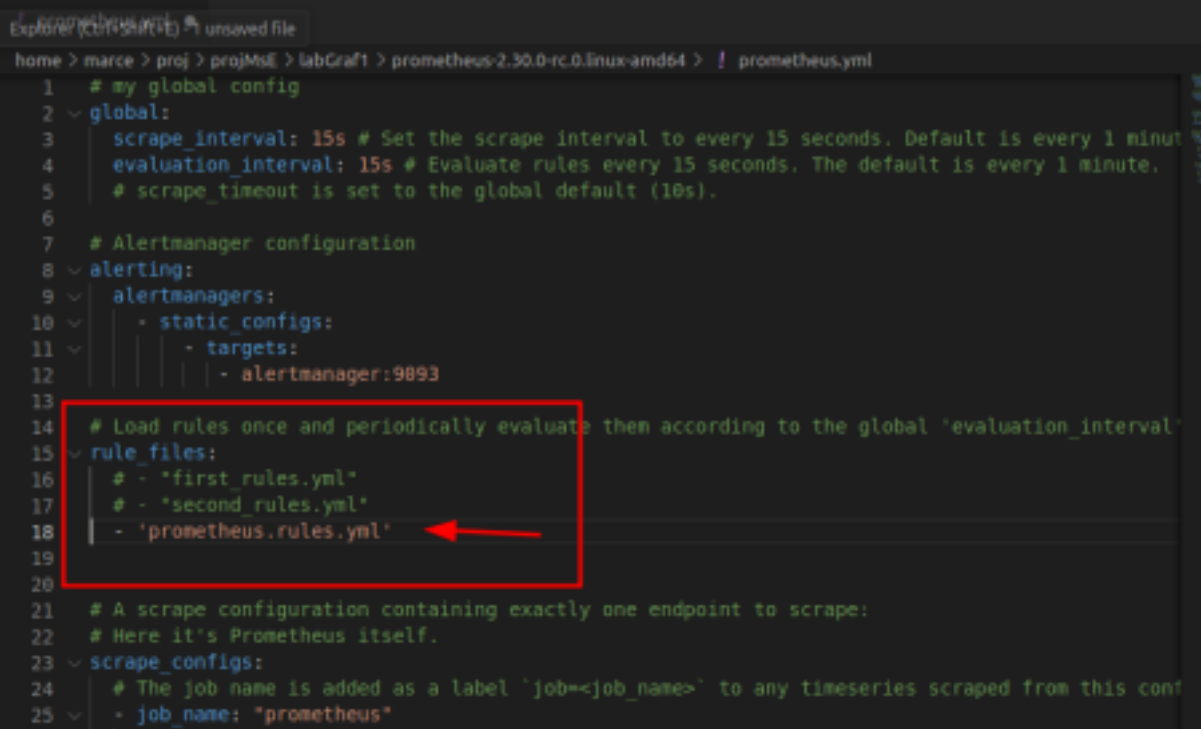


Configuración de Reglas para las alertas

Tenemos dos tipos de reglas, las **recording** rules y las **alerting** rules.

Recording Rules

Estas reglas nos sirven para ir almacenando resultados que pueden tener un alto coste computacional, así se irán ejecutando periódicamente y guardando como una métrica que podemos usar, lo cual hará que sea mucho mas rápido a la hora de consultarla o usarla en cálculo de otras consultas. Són muy útiles para usar en paneles que se refrescan automáticamente, ya que se generará el calculo por detrás y no tendrá que ser el panel el que espere a la ejecución del cálculo.



```
1 # my global config
2 global:
3   scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
4   evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
5   # scrape_timeout is set to the global default (10s).
6
7 # Alertmanager configuration
8 alerting:
9   alertmanagers:
10    - static_configs:
11      - targets:
12        - alertmanager:9093
13
14 # Load rules once and periodically evaluate them according to the global 'evaluation_interval'
15 rule_files:
16   - 'first_rules.yml'
17   - 'second_rules.yml'
18   - 'prometheus.rules.yml'
19
20 # A scrape configuration containing exactly one endpoint to scrape:
21 # Here it's Prometheus itself.
22 scrape_configs:
23   # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
24   - job_name: 'prometheus'
```

Este ejemplo lo que hace es calcular el porcentaje de uso de CPU de todas las instancias para poder consultarlo luego, no es una consulta pesada pero como ejemplo es bastante útil.

```
Explorer (Ctrl+Shift+E) yml x
home > marce > proj > projMsE > labGraf1 > prometheus-2.30.0-rc.0.linux-amd64 > ! prometheus.rules.yml
1 | groups:
2 | #   - name: nombre del grupo
3 | #   rules:
4 | #     - record: nombre:de:la:regla
5 | #       expr: expresión de consulta.
6 |
7 |
8 | groups:
9 | - name: Systems CPU Percentage
10 |   rules:
11 |   - record: systems.cpu_percentage_usage
12 |     expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{node="idle"}[1m])) * 100)
13 |
```

Verificamos

Prometheus Alerts Graph Status Help Classic UI

Rules

Systems CPU Percentage			4.584s ago	0.459ms
Rule	State	Error	Last Evaluation	Evaluation Time
record: systems.cpu_percentage_usage expr: 100 - (avg by(instance) (irate(node_cpu_seconds_total{node="idle"}[1m])) * 100)	OK		4.587s ago	0.459ms

Ejecutamos

Prometheus Alerts Graph Status Help Classic UI

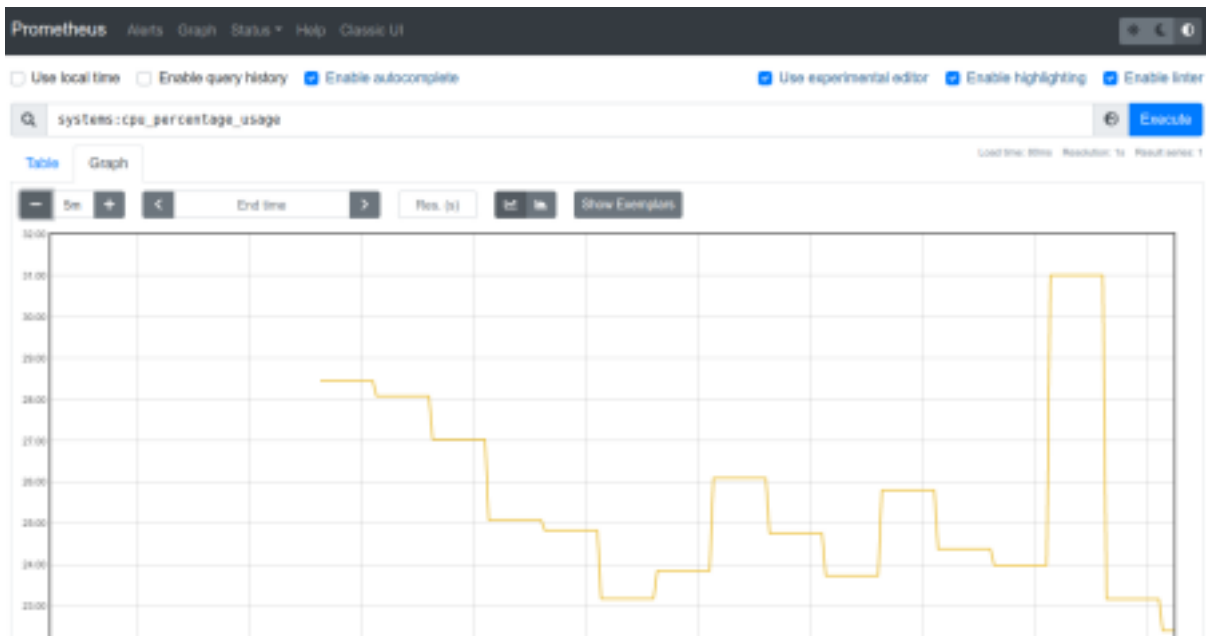
☐ Use local time ☐ Enable query history ☒ Enable autocomplete ☒ Use experimental editor ☒ Enable highlighting

Q systems.cpu_percentage_usage

Table Graph Load time: 23ms Reset

< Evaluation time >

systems.cpu_percentage_usage(instance="localhost:9100")	26.85633333334206
---	-------------------



Alerting Rules

Son reglas que queremos que disparen alertas si se cumplen las condiciones que establezcamos. Para crear estas reglas se hace de la misma forma que con las recording, con un fichero de reglas y añadiendo al fichero de configuración de Prometheus. Creamos una

```
! prometheus.rules.yml x
home > marce > proj > projMsE > labGraft > prometheus-2.30.0-rc.0.linux-amd64 > ! prometheus.rules.yml
1  # groups:
2  #   - name: nombre del grupo
3  #     rules:
4  #       - record: nombre de la regla
5  #         expr: expresión de consulta.
6
7
8  groups:
9  - name: Systems CPU Percentage
10     rules:
11     - record: systems:cpu_percentage_usage
12       expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[1m])) * 100)
13
14
15     # - name: Nombre
16     #   rules:
17     #     - alert: NombreAlerta
18     #       expr: Expresión de consulta
19     #       for: 5m #tiempo en el que tiene que cumplirse la regla para saltar
20     #       labels:
21     #         severity: page
22     #       annotations:
23     #         summary: Texto de resumen
24     - alert: HighCPUUsage
25       expr: systems:cpu_percentage_usage > 20
26       for: 1m
27       labels:
28         severity: page
29       annotations:
30         summary: High CPU Usage
31
```

Validar archivo de reglas

Si el fichero de reglas no es válido nos dará un error y no podremos ejecutar el servidor de Prometheus, así que existe una herramienta para validar los ficheros, eso sí, necesitaremos tener go instalado en la máquina en la que la ejecutemos. Se usa de la siguiente manera:

```
> ./promtool check rules prometheus.rules.yml
```

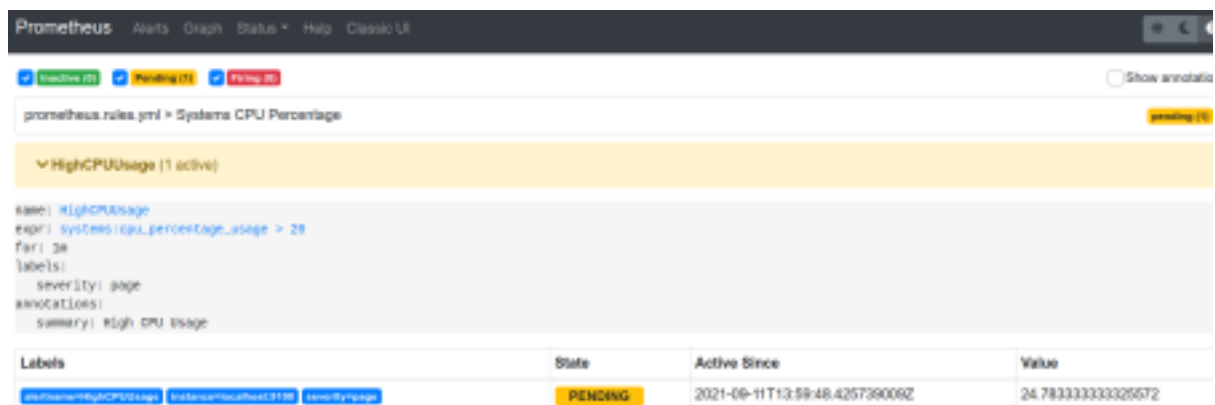
```
Checking prometheus.rules.yml
SUCCESS: 2 rules found
```

```
> ./promtool check rules cpu_alert.yml
```

```
Checking cpu_alert.yml
FAILED:
cpu_alert.yml: yaml: unmarshal errors:
  line 12: field rules not found in type rulefmt.RuleGroups
```

Ahora vamos a ver cómo funcionan las alertas, para ello reiniciamos el servicio de prometheus y controlamos desde la web

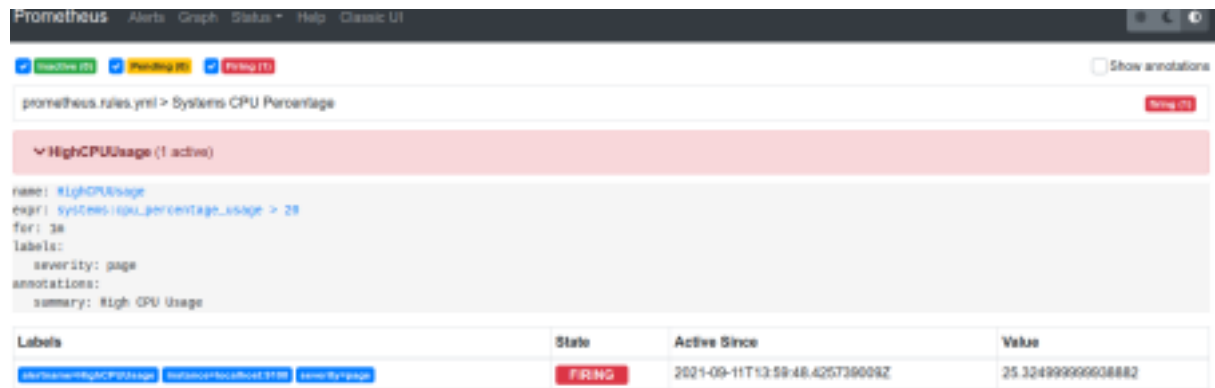
Podremos ver que está en amarillo porque está pendiente. Ha encontrado una instancia que cumple la condición y está esperando los 5 minutos que le hemos dicho para ver si dispararse o no.



The screenshot shows the Prometheus Alerting interface. At the top, there are tabs for Alerts, Graph, Status, and Help. Below the tabs, there are filters for Inactive (0), Pending (1), and Firing (0). The main section shows a rule named 'HighCPUUsage' with a severity of 'page'. The alert is currently in a 'PENDING' state. The table below shows the details of the alert.

Labels	State	Active Since	Value
environment:staging instance:localhost:9090 job:prometheus	PENDING	2021-09-11T13:59:48.425739006Z	24.783333333326672

Cuando se dispara



Prometheus Alerts Graph Status Help Classic UI

1 Inactive (0) 1 Pending (0) 1 Firing (1) Show annotations

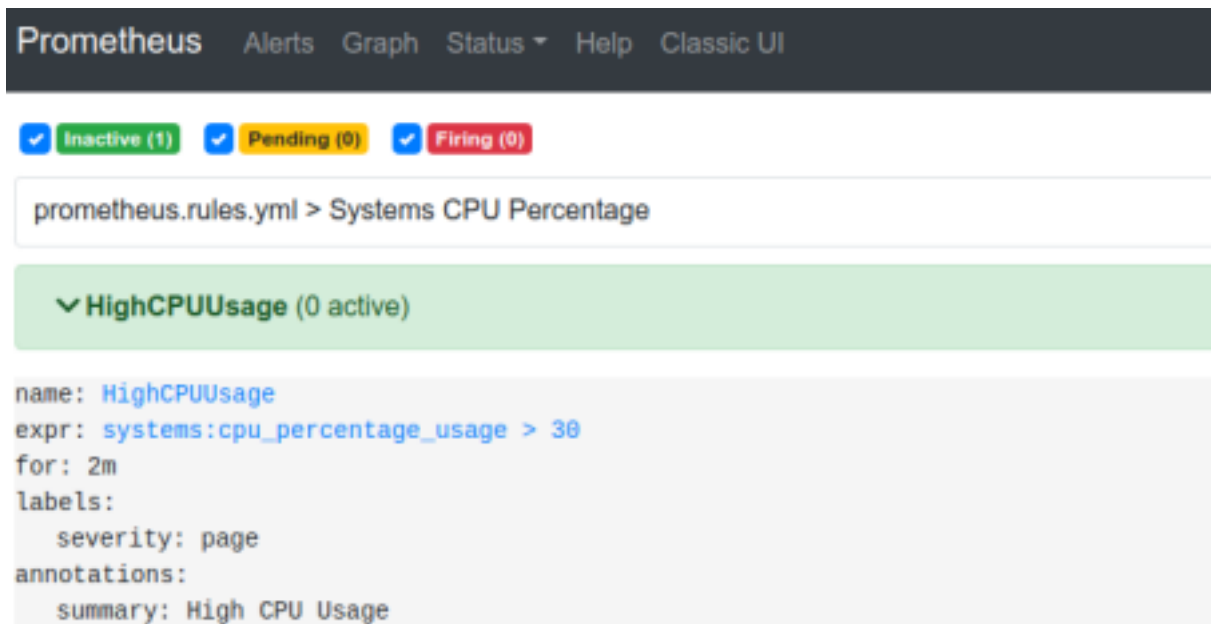
prometheus.rules.yml > Systems CPU Percentage Firing (1)

▼ HighCPUUsage (1 active)

name: HighCPUUsage
expr: systems:cpu_percentage_usage > 28
for: 3s
labels:
 severity: page
annotations:
 summary: High CPU Usage

Labels	State	Active Since	Value
instance=host01 CPUUsage=25.324999999999998 severity=page	FIRING	2021-09-11T13:58:48.425739009Z	25.324999999999998

Normal



Prometheus Alerts Graph Status Help Classic UI

1 Inactive (1) 1 Pending (0) 1 Firing (0)

prometheus.rules.yml > Systems CPU Percentage

▼ HighCPUUsage (0 active)

name: HighCPUUsage
expr: systems:cpu_percentage_usage > 30
for: 2m
labels:
 severity: page
annotations:
 summary: High CPU Usage

Estresando el sistema - generando carga Utilizamos

una aplicación para generar carga de forma simple y probar nuestras alertas

La instalamos con:

> sudo apt install stress

Ahora vamos a enviar 8 procesos intensivos en cada CPU, 4 procesos de entrada y salida, y 6 procesos directos a la memoria RAM , cada uno de 256Mb , durante un tiempo de 150 segundos

> stress -c 4 -i 4 -m 6 --vm-bytes 256M -t 150s --verbose

El comando anterior genera una carga inmediata en nuestro sistema para disparar las alertas

Configuración de destinos para nuestras alertas

No es práctico tener a alguien viendo los paneles de forma intensiva, si un control general pero no particular, para eso necesitamos que si se produce una alerta nos envíe la misma por algún canal configurado. Veremos como configurar para que nos avise por slack que es una herramienta muy difundida

Slack

Debemos tener instalado slack y ser parte de un “workspace” donde crearemos un canal para que lleguen los avisos

Creamos un workspace si no lo tenemos desde

<https://slack.com/create>

Luego necesitamos configurar una api url para recibirlas mediante un web-hook

Esto lo hacemos desde

Slack -> Administration -> Manage apps

Buscamos los webhooks y le indicamos el canal de destino

Anotamos la URL del webhook que nos da slack

<https://hooks.slack.com/services/T02E4GT6282/B02E4H034AW/rj7ZfBUhPMJcpAkPEFw9B6T>

Ahora vamos al archivo de configuración del alertmanager y actualizamos la sección global, rules y receivers

Reiniciamos el servicio de alertmanager y ahora podemos correr nuestro comando para generar carga en el sistema y comprobar que se genere la alerta y nos llegue a slack

```
> stress -c 4 -i 4 -m 6 --vm-bytes 256M -t 150s --verbose
```

Comprobamos la recepción de los avisos en Slack, en nuestro canal de monitoreo

También podemos agregar otras “reglas” para generar nuevas alertas

<https://rakeshjain-devops.medium.com/prometheus-alerting-most-common-alert-rules-e9e219d4e949>