Proceso Monitoización de docker con prometheus y slack

El siguente manual es para indicar el prcoedimiento para realizar la instalación de prometheus en un servidor linux para la monitorización y la presentación de alertas.

Prerrequisitos.

- Tener instalado docker en la maquina.

- Conocimientos basicos de linux y docker.

- Acceso al servidor como usuario root.

Paso a paso

Lo primero seria abrir el archivo /etc/docker/daemon.json con tu editor favorito(nano, vi, vim etc..), por lo general este archivo esta vacio en caso de no encontrar el archivo puedes crearlo, lo siguente es agregar esta linea:

```
{
"metrics-addr": "0.0.0.0:9323"
}
```

Esta linea puede depender de los equipos o la configuración de red, por lo general para los servidores o maquinas de ADA es la anteriormente indicada en algunos casos puede que esta no funcione entonces intente con la siguinte.

```
{
"metrics-addr": "127.0.0.1:9323"
}
```

Luego haga el reinicio del servicio del docker.

```
systemctl restart docker
```

En caso de que el servicio no suba revise que no agrego nada mas a ese archivo.

Para validar que todo quedo bien ingrese a la url en el navegador con ip del servidor:9323. Ejemplo:

```
http://ipservidor:9323/metrics
```

Deberia mostrarle algo como lo siguiente.

HELP builder_builds_failed_total Number of failed image builds # TYPE builder_builds_failed_total counter builder_builds_failed_total{reason=“build_canceled”} 0
builder_builds_failed_total{reason=“build_target_not_reachable_error”} 0
builder_builds_failed_total{reason=“command_not_supported_error”} 0
builder_builds_failed_total{reason=“dockerfile_empty_error”} 0

builder_builds_failed_total{reason=“dockerfile_syntax_error”} 0
builder_builds_failed_total{reason=“error_processing_commands_error”} 0
builder_builds_failed_total{reason=“missing_onbuild_arguments_error”} 0
builder_builds_failed_total{reason=“unknown_instruction_error”} 0 # HELP
builder_builds_triggered_total Number of triggered image builds # TYPE builder_builds_triggered_total
counter builder_builds_triggered_total 0 # HELP engine_daemon_container_actions_seconds The
number of seconds it takes to process each container action # TYPE
engine_daemon_container_actions_seconds histogram
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“0.005”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“0.01”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“0.025”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“0.05”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“0.1”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“0.25”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“0.5”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“1”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“2.5”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“5”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“10”} 1
engine_daemon_container_actions_seconds_bucket{action=“changes”,le=“+Inf”} 1
engine_daemon_container_actions_seconds_sum{action=“changes”} 0
engine_daemon_container_actions_seconds_count{action=“changes”} 1

Nota: en caso de que le salga algun error de no esposible conectarse, revise si el puerto 9323 esta expuesto en caso de no estarlo expongalo

Configuración prometheus

Si ya puede ver las metricas es hora de configurar prometheus para el monitoreo, lo primero es levantarel contendor de prometheus, puede hacer el metodo tradicional o compose es opcional, el comando usado en este manual es:

```
docker run -d --name prometheus -p 9090:9090 -v
prometheus_config:/etc/prometheus prom/prometheus
```

Esto levantara el contendor con prometheus.

Recuerde que si esta utilizando una interfaz estricta debe exponer el puerto 9090 solo si tiene reglas estrictas El comando puede ser:

```
sudo firewall-cmd --permanent --zone=public --add-port=9090/tcp
sudo firewall-cmd --reload
```

Para asegurarse que este funcionando bien al igual que con las metricas puede ingresar ip:puerto a esa url y le debe mostrar la interfaz de prometheus.

ejemplo

```
http://ipservidor:9090
```

Luego dirijase a la lista desplegable que dice status, luego click en targets y valide que este en up el endpoint.

Ahora hay que configurar el archivo de prometheus llamado prometheus.yml, la ubicacion de este esta en el volumen creado para el contedor etso tal vez varie dependiendo de la distribución de linux o configuracion de docker pero en ubuntu para ingresar a la locación del archivo seria.

```
cd /var/lib/docker/volumes/prometeus_conf/_data/
```

O simplemente busque a donde apunta su volumen.

Ahora lance el siguiente comando para validar que si se encuentra el archivo

```
ls -l
```

debe aparecer el archivo prometheus.yml ahora abralo con su editor preferido, recuerde que este solo se deja modificar por un usuario del grupo sudo. ejemplo comando.

```
nano prometheus.yml
```

Se le prueden agregar muchas configuraciones a este archivo y declarar muchos parametros pero para este manual deje el archivo con el siguiente contenido.

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds.
Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is
every 1 minute.
  # scrape_timeout is set to the global default (10s).
# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          - 'ipservidor:9093'
# Load rules once and periodically evaluate them according to the global
'evaluation_interval'.
rule_files:
  - "/etc/prometheus/rules-files.yml"
  # - "second_rules.yml"
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries
scraped from this config.
  - job_name: "docker"
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
```

```
      - targets: ["ipservidor:9323"]
```

Lo siguiente seria reiniciar el contendor de prometheus pero si lo hace en este momento la mas probable es que el contenor no suba debido a que el archivo de reglas no esta definido asi que lo siguiente seria crearlo.

En ese msimo direcorio cree un archivo llamado rules-files.yml o el nombre de su preferencia pero recuerde referenciarlo en su archivo prometheus.yml.

Comando.

```
nano rules-files.yml
```

Este archivo es para definir algunas reglas y se puede definir en que caso se levantan alertas. Para esta configuración agregue la siguiente información a este archivo.

```
groups:
- name: docker-containers
  rules:
  - alert: ContainerStoppedAlert
    expr: engine_daemon_container_states_containers{state="stopped"} > 0
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "Detected stopped containers"
      description: "There are one or more containers stopped on the Docker
engine."
```

En resumen esta configuración hace que se levante una alerta cada vez que un contendor se detuvo por almenos 1 minuto.

Lo siguiente seria reinicar el contendor de prmetheus pero existe uan gran probabilidad de que este no suba debido a qaue aún no se ha creado ni configuradoe el aert manager.

Configuración alert manager

El alert manager es el encargado de administrar las alertas y hacer el envío de estas a donde se defina para levantar el alert manager.

Antes de levantar el alert manager se recomienda ya tener su archivo de configuación ya construido para eso digamos que voy a utilizar el directorio /opt/monitoring puede ser el de su preferencia.

Lo siguiente seria crear su archivo alertmanager.yml

Como se ha indicado estos archivos son muy configurables pero para este manual por favor ingrese el seguiente contenido al archivo.

```
global:
  resolve_timeout: 3m
```

```
route:
  group_by: ['alertname']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 2m
  receiver: 'slack-notifications'
receivers:
  - name: 'slack-notifications'
    slack_configs:
      - api_url: 'https://hooks.slack.com/services/your_chanel'
        channel: '#monitoring-portal'
        text: 'Se ha caído el portal {{ .CommonLabels.instance }}.'
        title: 'Portal Caído'
inhibit_rules:
  - source_match:
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['alertname', 'dev', 'instance']
```

Recuerde que debe generar su propio webhook para que conecte a su canal deseado de slack.

Ya con el archivo definido lo siguiente seria inicar el contendor de slack puede utilizar el siguiente comando.

```
docker run -d --name alertmanager -v
/opt/monitoring/alertmanager.yml:/etc/alertmanager/alertmanager.yml -p
9093:9093 prom/alertmanager
```

Ya con este arriba ahora reinicie el contendor de prometheus o levantelo si estaba abajo.

Esta vez el contendor si deberia quedar arriba, si ingresa por el navegador a prometheus y revisa en alert ya debe registrarle un item ya en este momento ya tiene configurado las alertas en caso de que se caiga algun contendor pueda responder a tiempo.

Ya solo tiene que configurar grafana para una monitoria mas comoda de sus contendores o agregar mas reglas para manejar mas factores de riesgo.

Gracias por su atención prestada