Prometheus Alert Manager

Task requirement:

"Create Email Alert by using Prometheus".

Prerequisite tools:

- Podman
- Prometheus
- Node-Exporter
- Alert Manager

System Configuration:

- OS Name : Ubuntu 20.04.6 LTS
- Podman version:- 3.4.2
- RAM : 5.6 GiB
- CPU : 12
- STORAGE : 512.1 GB

Prometheus Alertmanager Prometheus Alertmanager is an open-source component of the Prometheus monitoring and alerting ecosystem. It is used to manage and handle alerts generated by Prometheus, which is a popular monitoring and alerting toolkit.

Installation Process:

- Step:1 Create Directory for alert manager:

```
mkdir alertmanager
```

- Step:2 Go inside the directory by using "cd" command:

```
cd alertmanager
```

- **Use "ls" command to see the list of files and directories:**

```
ls
```

- Step:3 Now create a configuration file of alertmanager:

```
vim alertmanager.yml
```

- Step:4 Insert data into the configuation file:

```
global:
  resolve_timeout: 10s

route:
  receiver: 'gmail-notifications'
  routes:
    - match:
        severity: 'critical'
      receiver: 'gmail-notifications'

receivers:
- name: 'gmail-notifications'
  email_configs:
    - to: 'km.x.pinki@fosteringlinux.com'
      from: 'pinki.pinki280801@gmail.com'
      smarthost: 'smtp.gmail.com:587'
      auth_username: 'pinki.pinki280801@gmail.com'
      auth_identity: 'pinki.pinki280801@gmail.com'
      auth_password: 'efxujzsvbzbhimqg'
      send_resolved: true
```

- Step:5 Now run alert manager container using podman.

```
podman run -d -p 9093:9093 --name alertmanager -v
/home/pinki/alertmanager/alertmanager.yml:/etc/alertmanager/alertmanager.ym
l  docker.io/prom/alertmanager
```

podman run: This is the command to start a container using Podman, a containerization tool similar to Docker.

-d: This flag stands for "detached" mode, which means the container will run in the background. It's commonly used for running services that should not be tied to the terminal session.

-p 9093:9093: This option maps port 9093 from the host to port 9093 in the container. This is used to expose the Alertmanager's web interface and API to the host system. It means you can access the Alertmanager service running inside the container from your host at http://localhost:9093.

--name alertmanager: This assigns a name to the running container, in this case, "alertmanager." Naming containers can make them easier to manage and interact with, especially when you have multiple containers running.

-v /home/pinki/alertmanager/alertmanager.yml:/etc/alertmanager/alertmanager.yml: This flag is used to bind-mount a configuration file from the host system into the container. It takes the configuration file at **/home/pinki/alertmanager/alertmanager.yml** on your host and mounts it into the container at **/etc/alertmanager/alertmanager.yml.** This allows you to provide a custom configuration for the Alertmanager.

docker.io/prom/alertmanager: This specifies the name of the Docker image to run the container. It pulls the official Prometheus Alertmanager Docker image from the Docker Hub repository (indicated by docker.io).

- Spep:6 Create a Directory of Prometheus:

```
mkdir Prometheus
```

- **Go inside the Directory by using "cd" command:**

```
cd Prometheus
```

- **use "ls" command to see the list of files and ditectories:**

```
ls
```

- Step:7 Create alertrules.yml file inside the prometheus Directory:

```
vim alertrules.yml
```

- Step:8 Insert data into the alertrules.yml:

```
groups:
- name: AllInstances
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 1m
    labels:
      severity: 'critical'
    annotations:
      summary: 'Instance {{ $labels.instance }} down'
      description: '{{ $labels.instance }} of job {{ $labels.job }} has
been down for more than 1 minute.'
```

- Step:9 Create prometheus.yml file into the prometheus directory

```
vim prometheus.yml
```

- Step:10 Insert data into the prometheus.yml

```
global:
  # How frequently to scrape targets
  scrape_interval: 10s
  # How frequently to evaluate rules
  evaluation_interval: 10s

# Rules and alerts are read from the specified file(s)
rule_files:
  - alertrules.yml

# Alerting specifies settings related to the Alertmanager
alerting:
  alertmanagers:
    - static_configs:
      - targets:
        # Alertmanager's default port is 9093
        - 192.168.1.60:9093

# A list of scrape configurations that specifies a set of
# targets and parameters describing how to scrape them.
scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
```

```
      - targets:
        - 192.168.1.60:9090
  - job_name: 'node'
    scrape_interval: 5s
    static_configs:
      - targets:
        - 192.168.1.60:9100
```

Step: 10 Run this command to start Prometheus (alertrules.yml):

```
podman run -d --name prometheus -p 9090:9090 -v
/home/pinki/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml -v
/home/pinki/prometheus/alertrules.yml:/etc/prometheus/alertrules.yml
docker.io/prom/prometheus
```

podman run: This is the command to start a container using Podman, a containerization tool similar to Docker.

-d: This flag stands for "detached" mode, meaning the container will run in the background. It's commonly used for running services that should not be tied to the terminal session.

--name prometheus: This assigns a name to the running container, in this case, "prometheus." Naming containers can make them easier to manage and interact with, especially when you have multiple containers running.

-p 9090:9090: This option maps port 9090 from the host to port 9090 in the container. This is used to expose the Prometheus web user interface and API to the host system. It means you can access the Prometheus service running inside the container from your host at http://localhost:9090.

-v /home/pinki/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml: This flag is used to bind-mount a configuration file from the host system into the container. It takes the Prometheus configuration file at **/home/pinki/prometheus/prometheus.yml** on your host and mounts it into the container at /etc/prometheus/prometheus.yml. This allows you to provide a custom configuration for Prometheus.

-v /home/pinki/prometheus/alertrules.yml:/etc/prometheus/alertrules.yml: Similar to the previous -v flag, this one is used to bind-mount an alert rules configuration file from the host system into the container. It takes the alert rules file at **/home/pinki/prometheus/alertrules.yml** on your host and mounts it into the container at /etc/prometheus/alertrules.yml. This allows you to provide custom alerting rules to be used by Prometheus.

docker.io/prom/prometheus: This specifies the name of the Docker image to run the container. It pulls the official Prometheus Docker image from the Docker Hub repository (indicated by docker.io).

- Get Email Alert