**\*\*\* How to set-up prometheus in local vm (Ubuntu) \*\*\***

**Prometheus:**

==========

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud.Prometheus collects and stores its metrics as time series data, i.e. metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

**What are metrics ?**

===================

metrics are numeric measurements.Time series means that changes are recorded over time.For a web server it might be request times, for a database it might be number of active connections or number of active queries.

Metrics play an important role in understanding why your application is working in a certain way.

* **Steps to follow the prometheus set-up in local vm/ubuntu**

**Setup Prometheus Binaries on the system.**

========================================

**Step 1: Update the yum package repositories.**

**======**

# sudo apt update -y

**\*\*\*** Download the official Prometheus downloads page and get the latest download link for the Linux binary: <https://prometheus.io/download/>

**\*\*\*** Download the source using curl, untar it, and rename the extracted folder to Prometheus-files.

# curl -LO https://github.com/prometheus/prometheus/releases/download/v2.38.0/prometheus-2.38.0.linux-amd64.tar.gz

**\*\*\*** After Downloading that extract and run it by following commands

# tar xvfz prometheus-\*.tar.gz **(**use this,if u download by official site**)**

# tar -xvf prometheus-2.38.0.linux-amd64.tar.gz **(**use this,if u download by using above curl command**)**

**\*\*\*** Move the files

# mv prometheus-2.38.0.linux-amd64 prometheus-files

**\*\*\*** Create a Prometheus user, required directories, and make Prometheus the user as the owner of those directories.

# sudo useradd --no-create-home --shell /bin/false prometheus

# sudo mkdir /etc/prometheus

# cd

# sudo mkdir /var/lib/prometheus

# sudo chown prometheus:prometheus /etc/prometheus

# sudo chown prometheus:prometheus /var/lib/prometheus

**\*\*\*** Copy Prometheus and promtool binary from Prometheus-files folder to /usr/local/bin and change the ownership to Prometheus user.

# sudo cp prometheus-files/prometheus /usr/local/bin/

# sudo cp prometheus-files/promtool /usr/local/bin/

# sudo chown prometheus:prometheus /usr/local/bin/prometheus

# sudo chown prometheus:prometheus /usr/local/bin/promtool

Move the consoles and console\_libraries directories from Prometheus-files to /etc/Prometheus folder and change the ownership to Prometheus user.

# sudo cp -r prometheus-files/consoles /etc/prometheus

**\*\*\***

# sudo cp -r prometheus-files/console\_libraries /etc/prometheus

# sudo chown -R prometheus:prometheus /etc/prometheus/consoles

# sudo chown -R prometheus:prometheus /etc/prometheus/console\_libraries

**Setup Prometheus Configuration**

==============================

**\*\*\*** All the Prometheus configurations should be present in the/etc/prometheus/prometheus.yml file. If you want to monitor any system metric, you need to configure it through this yaml file and need to give the source URL from where you want to collect the data.

Step 1: Create the prometheus.yml file.

# sudo vi /etc/prometheus/prometheus.yml

**\*\*\*** In the above file, We have to add source receiver URL under the remote\_write section for receiving the metrics. Copy the following contents to the prometheus.yml file.

—---------------

**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.**

**#remote\_write:**

**# - url: http://< Source-receiver host name >/api/v1/receive**

**# - url: https://www.google.com/**

**# Alertmanager configuration**

**alerting:**

**alertmanagers:**

**- static\_configs:**

**- targets:**

**# - alertmanager:9093**

**# Load rules once and periodically evaluate them according to the global 'evaluation\_interval'.**

**rule\_files:**

**# - "first\_rules.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.**

—---------------

**\*\*\*** Change the ownership of the file to the Prometheus user.

# sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml

**\*\*\*** Setup Prometheus Service File

**Step 1: Create a Prometheus service file**.

=======

# sudo vi /etc/systemd/system/prometheus.service

—---------------

**[Unit]**

**Description=Prometheus**

**Documentation=https://prometheus.io/docs/introduction/overview/**

**Wants=network-online.target**

**After=network-online.target[Service]**

**Type=simple**

**User=prometheus**

**Group=prometheus**

**ExecReload=/bin/kill -HUP $MAINPID**

**ExecStart=/usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/var/lib/prometheus --web.console.templates=/etc/prometheus/consoles --web.console.libraries=/etc/prometheus/console\_libraries --web.listen-address=0.0.0.0:9090 --web.external-url=SyslogIdentifier=prometheus**

**Restart=always[Install]**

**WantedBy=multi-user.target**

—---------------

**\*\*\*** Reload the system service to register the Prometheus service and start the Prometheus service.

# sudo systemctl daemon-reload

# sudo systemctl start prometheus

**\*\*\*** Check the Prometheus service status using the following command

# sudo systemctl status prometheus

**\*\*\*** Access Prometheus Web UI

Now you will be able to access the Prometheus UI on the 9090 port of the Prometheus server.

# http://<prometheus-ip/localhost-ip>:9090/graph

## **Install Node Exporter on Ubuntu**

we're going to set up and configure Node Exporter to collect Linux system metrics like CPU load and disk I/O. Node Exporter will expose these as Prometheus-style metrics.

create a system user for Node Exporter by running the following command:

[# sudo useradd \](https://antonputra.com/monitoring/install-prometheus-and-grafana-on-ubuntu/" \l "__codelineno-18-2)

[--system \](https://antonputra.com/monitoring/install-prometheus-and-grafana-on-ubuntu/" \l "__codelineno-18-3)

[--no-create-home \](https://antonputra.com/monitoring/install-prometheus-and-grafana-on-ubuntu/" \l "__codelineno-18-4)

--shell /bin/false node\_exporter

Use wget command to download binary

# wget https://github.com/prometheus/node\_exporter/releases/download/v1.3.1/node\_exporter-1.3.1.linux-amd64.tar.gz

Extract node exporter from the archive

# tar -xvf node\_exporter-1.3.1.linux-amd64.tar.gz

Move binary to the /usr/local/bin.

# sudo mv \

node\_exporter-1.3.1.linux-amd64/node\_exporter \

/usr/local/bin/

Clean up, delete node\_exporter archive and a folder.

# rm -rf node\_exporter\*

Verify that you can run the binary

# node\_exporter --version

Node Exporter has a lot of plugins that we can enable. If you run Node Exporter help you will get all the options

# node\_exporter --help

Next, create similar systemd unit file.

# sudo vim /etc/systemd/system/node\_exporter.service

Node\_exporter.service

—---------------------------

[Unit]

Description=Node Exporter

Wants=network-online.target

After=network-online.target

StartLimitIntervalSec=500

StartLimitBurst=5

[Service]

User=node\_exporter

Group=node\_exporter

Type=simple

Restart=on-failure

RestartSec=5s

ExecStart=/usr/local/bin/node\_exporter \

--collector.logind

[Install]

WantedBy=multi-user.target

—--------------------------

Replace Prometheus user and group to node\_exporter, and update ExecStart command

To automatically start the Node Exporter after reboot, enable the service

# sudo systemctl enable node\_exporter

Then start the Node Exporter.

# sudo systemctl start node\_exporter

Check the status of Node Exporter with the following command:

# sudo systemctl status node\_exporter

## **Download and Install Promtail on ubuntu**

To check the latest version of Promtail, visit the Loki releases page. <https://github.com/grafana/loki/releases/>

# cd /usr/local/bin

# curl -O -L "<https://github.com/grafana/loki/releases/download/v2.4.1/promtail-linux-amd64.zip>"

# unzip "promtail-linux-amd64.zip"

And allow the execute permission on the Promtail binary

# sudo chmod a+x "promtail-linux-amd64"

Now we will create the Promtail config file.

# sudo nano config-promtail.yml ### nano/vi/vim use any of these editors

Add the following script into the config-promtail.yml file

—-------------

server:

http\_listen\_port: 9080

grpc\_listen\_port: 0

positions:

filename: /tmp/positions.yaml

clients:

- url: 'http://localhost:3100/loki/api/v1/push'

scrape\_configs:

- job\_name: system

static\_configs:

- targets:

- localhost

labels:

job: varlogs

\_\_path\_\_: /var/log/\*log

—------------------------------------

Now we will configure Promtail as a service so that we can keep it running in the background.

Create user specifically for the Promtail service

# sudo useradd --system promtail

Create a file called promtail.service

# sudo nano /etc/systemd/system/promtail.service

### nano/vi/vim use any of these editors

Add the following script into the promtail.service

—---------

[Unit]

Description=Promtail service

After=network.target

[Service]

Type=simple

User=promtail

ExecStart=/usr/local/bin/promtail-linux-amd64-config.file /usr/local/bin/config-promtail.yml

[Install]

WantedBy=multi-user.target

—----------------------

Now start and check the service is running.

# sudo service promtail start

# sudo service promtail status

**\*\*\* How to set-up prometheus and Promtail in Windows machine \*\*\***

**Pre-requisites:**

* **install NSSM tool**
* **promtail**
* **node-exporter**
* **Prometheus**

install NSSM tool

<https://nssm.cc/download> ### official site for NSSM

To download the Nssm tool the below is the link

<https://nssm.cc/release/nssm-2.24.zip>

After download the above link and execute it. Later follow some commands.

* Open cmd / powershell
* Go to nssm file path
* Go inside the win32 folder
* Run .\nssm.exe install <the name of your service>
* You will see the panel
* Select the path where the service.exe is present
* Go to config section , write --config.file=<service-file-name>
* And press enter
* Service got installed
* Run .\nssm.exe start <the name of your service>

**\*** Install node-exporter in windows

https://github.com/prometheus-community/windows\_exporter/releases

And download this file

[windows\_exporter-0.22.0-amd64.msi](https://github.com/prometheus-community/windows_exporter/releases/download/v0.22.0/windows_exporter-0.22.0-amd64.msi)

And install it on your system

After installation run this url in your browser to cross check the metrics you are getting or not <http://localhost:9182/metrics>

After installtion of node exporter install prometheus  
goto release section in the github

https://github.com/prometheus/prometheus/releases

Download the zip file name [prometheus-2.44.0.windows-amd64.tar.gz](https://github.com/prometheus/prometheus/releases/download/v2.44.0/prometheus-2.44.0.windows-amd64.tar.gz)

Unzip the file and open the prometheus.yml file

And add the node exporter Scrape config in that

And add the Remote URL if you are sending the metrics to any other thanos

- job\_name: "local\_IIS\_windows\_metrix"

static\_configs:

- targets: ["localhost:9182"]

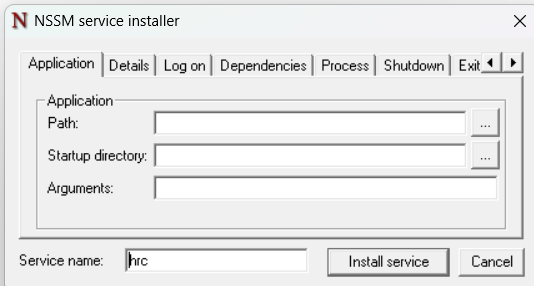
After that use NSSM tool to run the prometheus

Open the powershell as an administrator

Got to NSSM folder win32 section  
run cmd

**->** ./nssm.exe install prometheus

You will get tis page



In the path section provide the path where the config file and prometheus.exe file is present

And in the argument mension the --config.file=<name-of-config>.yml

And click on install and after that run cmd

**->** ./nssm.exe start prometheus

**\*** Setup promtail in the windows

Same as prometheus we will install the promtail file from the relese section from github

https://github.com/grafana/loki/releases

Download [promtail-windows-amd64.exe.zip](https://github.com/grafana/loki/releases/download/v2.8.2/promtail-windows-amd64.exe.zip) file

After installing extract the data you will get promtail.exe  
create config file for promtail

Here is the config file

server:

http\_listen\_port: 9080

grpc\_listen\_port: 0

positions:

filename:C:\Users\Harish\Downloads\prometheus-2.42.0.windows-amd64\prometheus-2.42.0.windows-amd64\position.yaml

clients:

- url: http://<url>/loki/api/v1/push

scrape\_configs:

- job\_name: <Name>

static\_configs:

- targets:

- localhost

labels:

job: <name>

\_\_path\_\_: <Actual-path-of-the-file>

Save It and run this promtail using NSSM tool again as we run for the prometheus.

**\*\*\* How to set-up Thanos, prometheus and Promtail on the cluster \*\*\***

**Prerequisites:**

**Kubernetes cluster**

**Helm**

**Docker**

**\* Install nginx ingress**

**-->** kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml

**\* Thanos Setup in the cluster**

**thanos setup**

**-->** helm repo add bitnami <https://charts.bitnami.com/bitnami>

**-->** helm repo update

**-->** helm install thanos bitnami/thanos -f values.yaml

In values.yaml we need to enable reciver to collect the metrics

receive:

  enabled: true

  persistence:

    enabled: true

    accessModes:

      - ReadWriteOnce

    size: 32Gi

  ingress:

    enabled: true

hostname: thanos-receive.<nginx-ingress-ip>.nip.io

We will create two ingress one is for frontend one is for reciver to collect the data   
  
first is thanos-query-ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  annotations:

    cert-manager.io/cluster-issuer: letsencrypt-prod

  name: query-ingress

spec:

  ingressClassName: nginx

  rules:

  - host: thanos-query.<nginx-ingrss-ip>.nip.io

    http:

      paths:

      - backend:

          service:

            name: thanos-query-frontend

            port:

              number: 9090

        path: /

        pathType: Prefix

Now thanos-reciever-ingress.yaml we will create

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  annotations:

    cert-manager.io/cluster-issuer: letsencrypt-prod

  name: receiver-ingress

spec:

  ingressClassName: nginx

  rules:

  - host: thanos-receive.<nginx-ingrss-ip>.nip.io

    http:

      paths:

      - backend:

          service:

            name: thanos-receive

            port:

              number: 19291

        path: /

        pathType: Prefix

**\* Prometheus Setup in the cluster**

helm repo add prometheus-community \

https://prometheus-community.github.io/helm-charts

helm repo update

helm install prometheus prometheus-community/prometheus -f \

prometheus-values.yaml

In prometheus-values.yaml you need to add the thanos url in the global section

  remoteWrite:

    - url: http://thanos-receive.<nginx-ip>.nip.io/api/v1/receive

**\* Loki with Promtail Setup in the cluster**

**-->** helm repo add grafana https://grafana.github.io/helm-charts

**-->** helm repo update

**-->** helm upgrade --install loki grafana/loki-stack -f loki-values.yaml

In loki-values.yaml

loki:

  enabled: true

  replicas: 6

promtail:

  enabled: true

fluent-bit:

  enabled: false

grafana:

  enabled: false

  sidecar:

    datasources:

      enabled: true

  image:

    tag: 6.7.0

We will create the ingress for it so we can provide it to the grafana as a data source

Loki-ing.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  annotations:

    cert-manager.io/cluster-issuer: letsencrypt-prod

    nginx.ingress.kubernetes.io/auth-type: basic

*# name of the secret that contains the user/password definitions*

    nginx.ingress.kubernetes.io/auth-secret: basic-auth

*# message to display with an appropriate context why the authentication is required*

    nginx.ingress.kubernetes.io/auth-realm: 'Authentication Required - foo'

  name: loki-ingress

spec:

  ingressClassName: nginx

  rules:

  - host: loki.<nginx-ip>.nip.io

    http:

      paths:

      - backend:

          service:

            name: loki

            port:

              number: 3100

        path: /

        pathType: Prefix

As you can see we have basic auth enable in the ingress so no one else can shaire the data in the loki

For basic auth you can follow this doc  
<https://kubernetes.github.io/ingress-nginx/examples/auth/basic/>

**\* Grafana Setup in the cluster with enabling SMTP**

**-->** helm repo add grafana https://grafana.github.io/helm-charts

**-->** helm repo update

**-->** helm install grafana grafana/grafana -n monitoring -f SMTP-grafana.yaml

In SMTP-grafana.yaml

env:

 GF\_SMTP\_ENABLED: "true"

 GF\_SMTP\_HOST: "smtp.sendgrid.net:587"

 GF\_SMTP\_PASSWORD: "" *#16 digit 2FA AppPassword*

 GF\_SMTP\_USER: "apikey"

 GF\_SMTP\_FROM\_ADDRESS: "admin@grafana.localhost"

 GF\_SMTP\_FROM\_NAME: "Grafana"

 GF\_DATABASE\_WAL: "true"

persistence:

  enabled: true

Then create the grafana ingress so we can access it

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: grafana-ing

spec:

  ingressClassName: nginx

  rules:

  - host: grafana.<nginx-ip>.nip.io

    http:

     paths:

     - path: /

       pathType: Prefix

       backend:

         service:

           name: grafana

           port:

             number: 80

**\*\*\*Grafana dashboard setup for metrics and logs of windows vm , linux vm and cluster \*\*\***

First login into grafana and add the data source of loki and thanos  
  
after adding the data sources go to dashboard panel click on and click on the import dashboard   
then write the dashboard no you want to import you can also use pre-build dashboards if metrics you want are present in that dashboard already.   
For serching the dashboard can use this link below:

<https://grafana.com/grafana/dashboards/>

We are using

For linux metrics 11074  
For Windows metrics 13466

For clusters metrics 13332

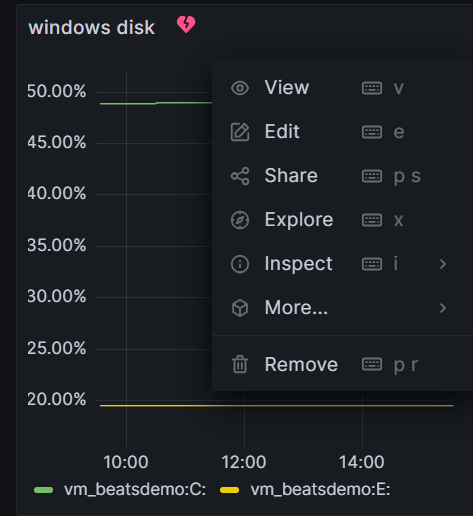
These 3 dashboards we are using for now

After setting up the dashboard lets create the alert on them

Note: you can only create alert on Time serise charts

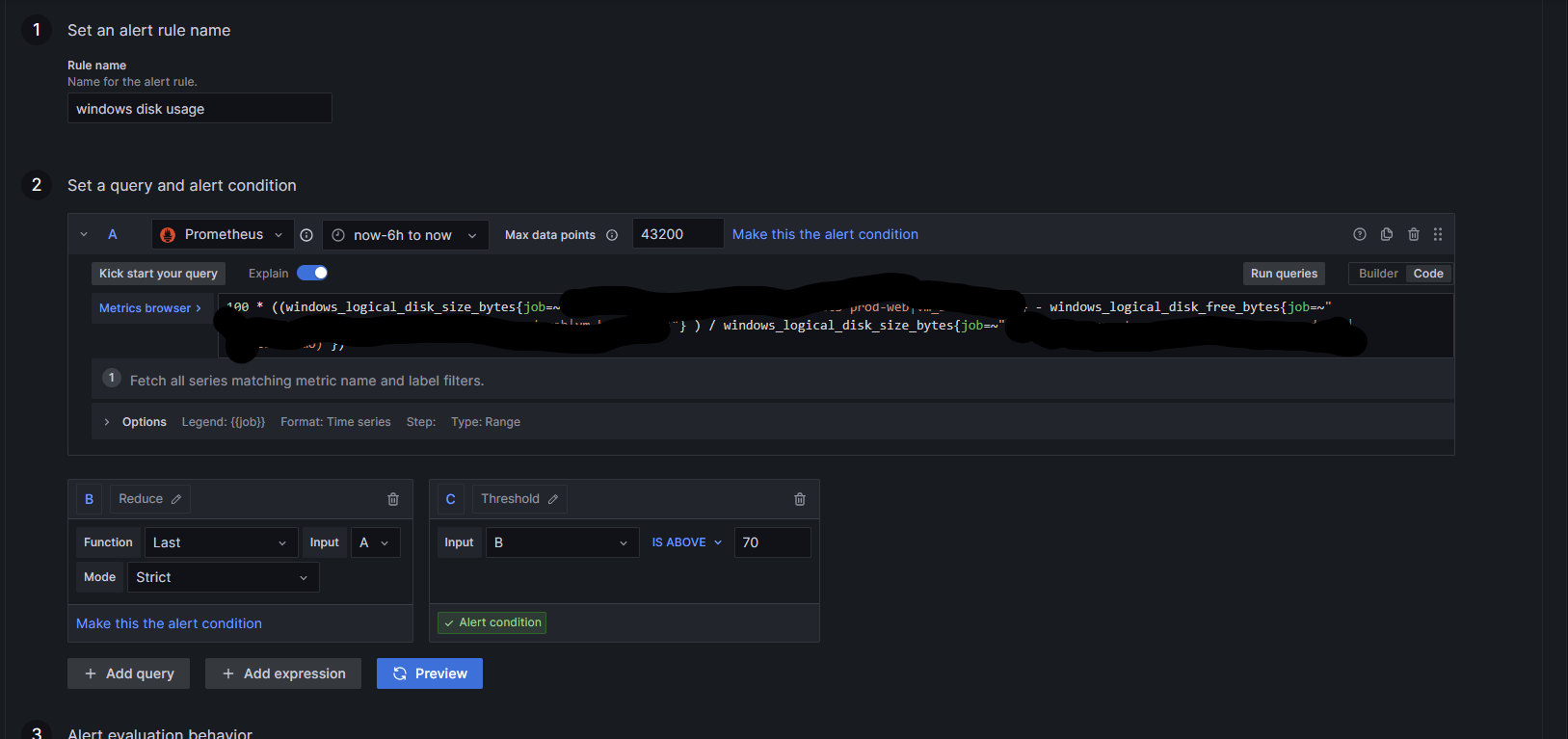
Eg.

First pic the panel where you want to select alert

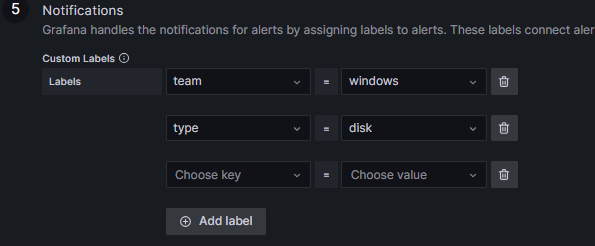
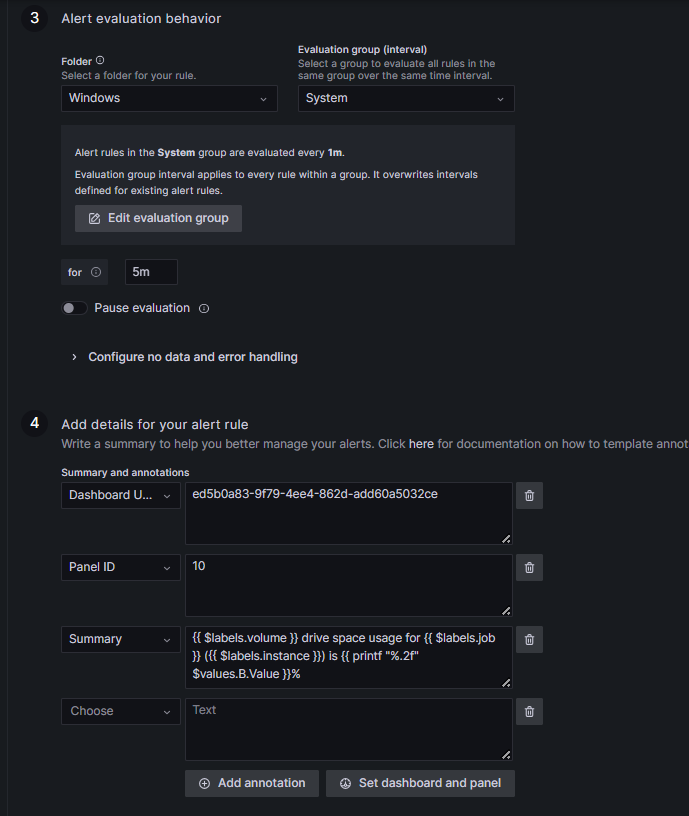


Go to edit section there you will se alert section click on it   


then click on alert rule for this panel



Provide all the details



After that save the alert

Now lets create the alert template and to whom we need to send alert and policy

So as you can see above alert is for windows so all alerts should go to windows people only

So first create template

{{- define "email.message\_alert" -}}

{{- range .Annotations.SortedPairs -}} {{- if eq .Name "summary" -}}{{ .Value }};

{{- end }} {{ end }}

{{- end -}}

{{- define "email.firing" -}}

[{{ .Status | toUpper }}

{{- if eq .Status "firing" -}}:{{ .Alerts.Firing | len }}

{{- if gt (.Alerts.Resolved | len) 0 -}}, RESOLVED:{{ .Alerts.Resolved | len }}

{{- end }}{{ end }}]

{{- end -}}

{{ define "email.sub" }}

{{ if .Alerts.Firing -}}

{{ template "email.firing" . }}

{{- range .Alerts.Firing }}

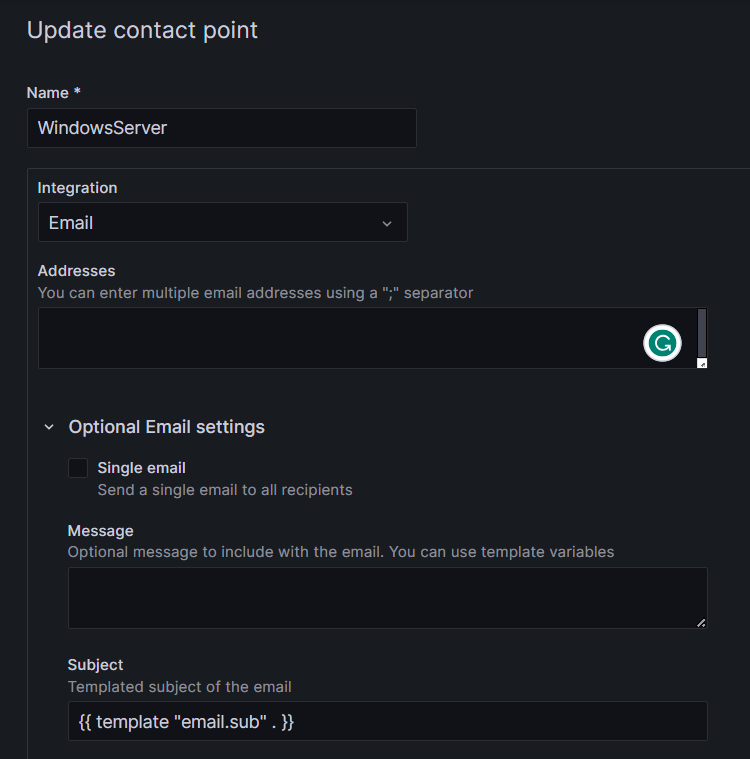
{{ template "email.message\_alert" . }}

{{- end }}

{{- end }}

{{ end }}

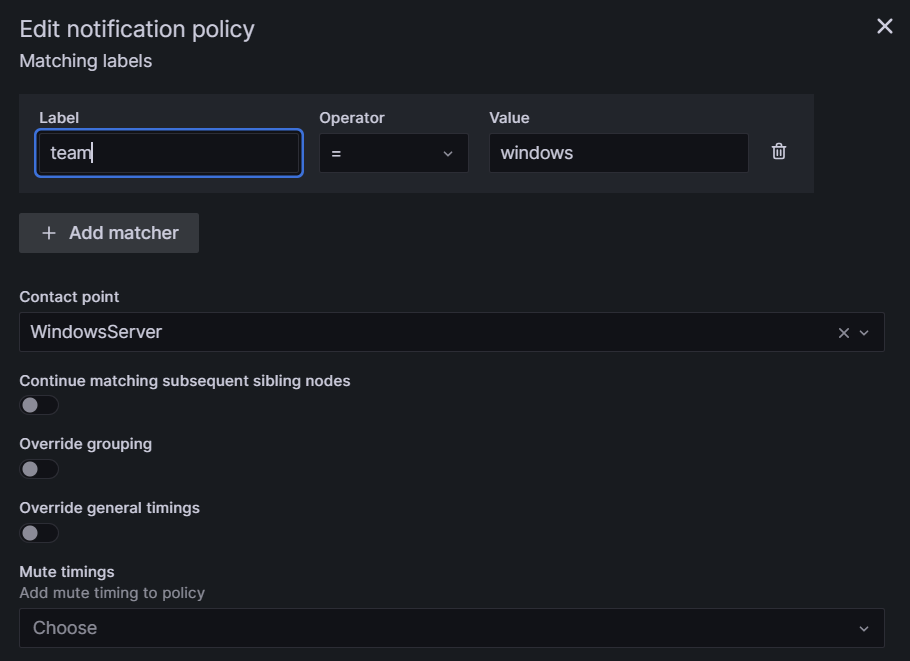
This template we are using now after that we will create the contact point for it



As you can see in the integration section we have to select the type of alert you want we are using emil now after that in address section add the mail id you want to share the alert with and in the subject give the name of the template we created so he will use that and send the alert

After that lets create policy so the win dows team only get this mail

Go to notification policy and click on new nested policy over there details



As you can see we provided the labal name over there and the contact point

And that’s how you will create the alert