**SESSION - 85**

Black Box Monitoring -> as an end user

White Box Monitoring -> inside the system. checking logs, metrics, etc..

**RCA Analysis**

**=============**

P0, P1, P2, P3, P4, etc.

Every alert is an incident.

Why that incident came?

Why we were not able to prevent anticipate it?

What is the effect?

How we cleared?

How can we make sure it will not occur again?

Lessons learnt?

**4 Golden Signals**

**=================**

Latency -> How soon our system is responding, low latency

Traffic --> How many requests per min, per hour

Errors -> 500 400 errors

Saturation -> resources like CPU, memory, Disk, Network, etc.

Prometheus -> TSD server

TSD -> time series data type

capture the metric value against time...

--> Launch server

--> redhat

--> security-group : allow-all

--> connect the server

--> sudo su

--> cd /opt/

--> wget <https://github.com/prometheus/prometheus/releases/download/v3.6.0-rc.0/prometheus-3.6.0-rc.0.linux-amd64.tar.gz>

wget <https://github.com/prometheus/prometheus/releases/download/v3.6.0-rc.0/prometheus-3.6.0-rc.0.linux-amd64.tar.gz>

--> ls -l

--> tar -xf prometheus-3.6.0-rc.0.linux-amd64.tar.gz

--> mv prometheus-3.6.0-rc.0.linux-amd64 prometheus

--> ls -l

--> cd prometheus

--> ls -l

**prometheus/prometheus.service**

**--> vim /etc/systemd/system/prometheus.service**

[Unit]

Description=Prometheus Service

After=network.target

[Service]

ExecStart=/opt/prometheus/prometheus --config.file=/opt/prometheus/prometheus.yml

Restart=on-failure

[Install]

WantedBy=multi-user.target

--> systemctl daemon-reload

--> systemctl restart prometheus

--> netstat -lntp

--> less /var/log/messages

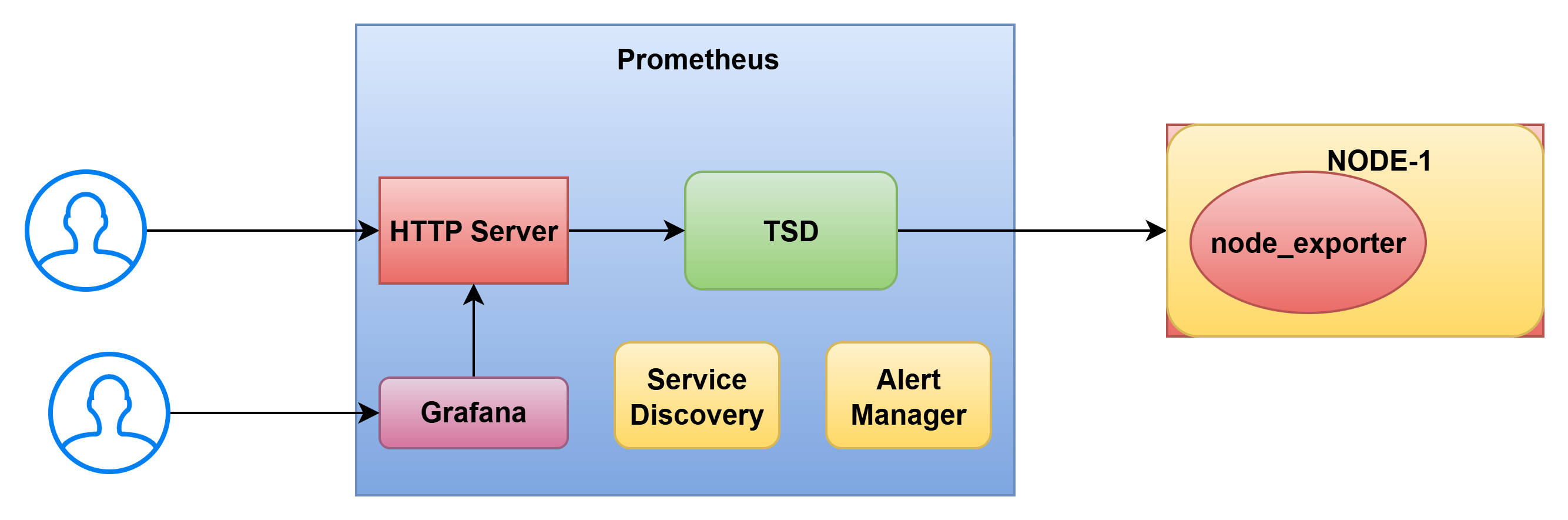
--> ls -l

--> check ip in browser (eg: 54.162.253.247:9090)

--> up is a simple metric whether system is made up or not

--> we need to monitor the servers

--> create a server



--> connect another server

--> sudo su

--> cd /opt/

--> wget <https://github.com/prometheus/prometheus/releases/download/v3.6.0-rc.0/prometheus-3.6.0-rc.0.linux-amd64.tar.gz>

--> tar -xf node\_exporter-1.9.1.linux-amd64.tar.gz

--> ls -l

--> mv node\_exporter-1.9.1.linux-amd64 node\_exporter

--> ls -l

--> cd node\_exporter

--> ls -l

**node\_exporter.service**

**--> vim /etc/systemd/system/node\_exporter.service**

[Unit]

Description=Node Exporter Service

After=network.target

[Service]

ExecStart=/opt/node\_exporter/node\_exporter

Restart=on-failure

[Install]

WantedBy=multi-user.target

--> systemctl daemon-reload

--> systemctl restart node\_exporter

--> netstat -lntp

**CC Camera**

**============**

Central Server --> CC Cameras

https://github.com/prometheus/node\_exporter/releases/download/v1.9.1/node\_exporter-1.9.1.linux-amd64.tar.gz

Prometheus should connect to configured exporter and pull the metrics…

**prometheus.yml**

# 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:

- "localhost:9093"

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

rule\_files:

- "alert-rules/\*.yaml"

# 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: "prometheus"

# metrics\_path defaults to '/metrics'

# scheme defaults to 'http'.

static\_configs:

- targets: ["localhost:9090"]

# The label name is added as a label `label\_name=<label\_value>` to any timeseries scraped from this config.

labels:

app: "prometheus"

- job\_name: 'ec2\_instances'

ec2\_sd\_configs:

- region: us-east-1

port: 9100

filters:

- name: "tag:Monitoring"

values: ["true"]

relabel\_configs:

# Rename the EC2 tag "Name" to the label "ec2\_name"

- source\_labels: [\_\_meta\_ec2\_tag\_Name]

target\_label: name

# Rename the EC2 instance ID to the label "instance\_id"

- source\_labels: [\_\_meta\_ec2\_instance\_id]

target\_label: instance\_id

# - job\_name: "node-1"

# # metrics\_path defaults to '/metrics'

# # scheme defaults to 'http'.

# static\_configs:

# - targets: ["172.31.35.112:9100"]

# # The label name is added as a label `label\_name=<label\_value>` to any timeseries scraped from this config.

# labels:

# app: "node-1"

# name: "node-1"

--> push the code

--> systemctl restart prometheus

--> take ip and check in prowser

--> up [1m]

--> up{name=”node-1”} [1m]

--> launch grafana instances

--> rhel-9

--> instance type : t3.micro

--> security group : allow-all

--> create a instances

--> wget -q -o gpg.key <https://rpm.grafana.com/gpg.key>

**--> curl -o gpg.key <https://rpm.grafana.com/gpg.key>**

**--> sudo rpm --import gpg.key**

**--> vim /etc/yum.repos.d/grafana.repo**

[grafana]

name=grafana

baseurl=https://rpm.grafana.com

repo\_gpgcheck=1

enabled=1

gpgcheck=1

gpgkey=https://rpm.grafana.com/gpg.key

sslverify=1

sslcacert=/etc/pki/tls/certs/ca-bundle.crt

**--> dnf install grafana**

**--> systemctl daemon-reload**

**--> systemctl start grafana-server**

**--> systemctl enable grafana-server**

**--> netstat -lntp**

**-->** take ip and give in browser

--> ip:3000

--> Grafana login page will get

--> username: admin

--> pwd: admin

--> connect <http://ip:9090>

--> save and update

--> create dashboard

--> metric is very import here

--> always we need equal system

--> here also will down the server

--> Configuring one metric will be very difficult due to

--> create new folder and import.

-->