Scope

Scope of this poc is:
- install and config postgres-exporter
- install and config one Prometheus-Server
- install and config alertmanager
- install and config Grafana

not in scope is:
- high availability
- fault tolerance
- custom Grafana dashboard
- Prometheus federation
- Security and ssl

Versions

Versions used:
- rhel 8.7
- postgresql-Server 13.9
- postgres-exporter 0.11.1 (latest)
- prometheus 2.40.4 (latest)
- alertmanager 0.24.0 (latest)
- Grafana 9.2.6 OSS Version (latest)

1. Architecture

The Prometheus/Grafana Monitoring Stack is built of 4 different parts

1) PostgresExporter

The exporter is a systemd binary which runs on each host that has a postgres-server instance running which should be monitored.
So for example in a cluster of 5 postgres-servers there should be 5 exporters for every server instance.
Its purpose is to connect to a specific postgres-server and scrapes the server. The exporter is stateless and doesnt store any data.
The exporter connects itself to the database with a special user (in this documentation pgexporter) that has the pgmonitor role.

2) Prometheus

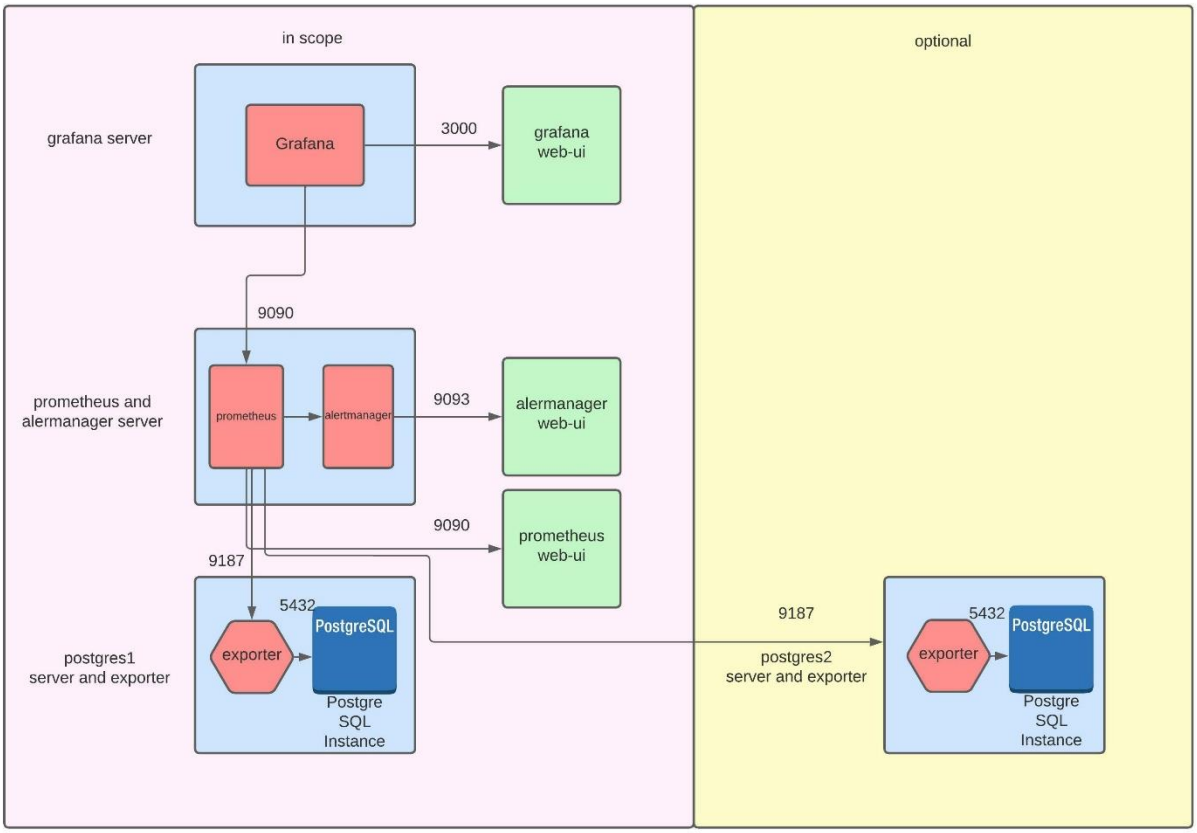
Prometheus is a time series database which scrapes and stores the data from the exporters.
It also provides a simple web gui for basic monitoring and handles alerts that than will be sent to the alertmanager.
Service discovery and alert rules are done through config files.
In this setup prometheus is not HA and also no federation is included now but both can be achieved later.

3) Alertmanager

Alertmanager receives configured alerts from Prometheus and forwards them to different channels such as email, on-call notification systems, and chat platforms.
It also supports grouping from similar events into one event, and silencing. It has a basic web-gui but config is also done with a config file.

4) Grafana

The purpose of Grafana is to visulaize the data from Prometheus.
It is possible for Grafana to include multiple datasources what in this context means Prometheus servers however in this documentation we are only dealing with one datasource.
It is accessible through a web-gui which also supports some basic user management in the OSS version.



2. Configuration

2.1 PostgresExporter

Download the exporter binary from the prometheus community github in opt directory (directory can be changed)

```
Sudo mkdir /opt/postgres-exporter

cd /opt/postgres-exporter
sudo wget https://github.com/prometheus-community/postgres_exporter/releases/download/v0.11.1/postgres_exporter-0.11.1.linux-amd64.tar.gz
sudo tar -xvzf postgres_exporter-0.11.1.linux-amd64.tar.gz

cd postgres_exporter-0.11.1.linux-amd64

sudo cp postgres_exporter /usr/local/bin
```

Adding user and set permissions

```
sudo useradd -rs /bin/false pgexporter
sudo chown -R pgexporter:pgexporter /opt/postgres-exporter
```

Creating user pgexporter in the Postgres database and grant monitoring permissions

```
CREATE ROLE pgexporter WITH NOSUPERUSER NOCREATEDB NOCREATEROLE
NOREPLICATION LOGIN PASSWORD 'secretpassword';
GRANT pg_monitor TO pgexporter;
```

adding pgexporter to pg_hba.conf

```
local   postgres    pgexporter                      password
host    postgres    pgexporter     127.0.0.1/32     password
host    postgres    pgexporter     ::1/128          password
```

In PostgreSQL, views run with the permissions of the user that created them so they can act as security barriers. Functions need to be created to share this data with the non-superuser. To do so apply the changes in pgexporter.sql file

Adding config file and set permissions

```
sudo mkdir /etc/postgres-exporter
sudo nano /etc/postgres-exporter/postgres-exporter.env
# inside the file add the following to monitor all the databases
available on localhost or specific ip of postgres server

DATA_SOURCE_NAME="postgresql://pgexporter:password@localhost:5432/postgres?sslmode=disable"
chown pgexporter:pgexporter /etc/postgres-exporter/postgres-
exporter.env
```

Create the postgres-exporter.service

```
sudo nano /etc/systemd/system/postgres_exporter.service

[Unit]
Description=Prometheus exporter for Postgresql
Wants=network-online.target
After=network-online.target
[Service]
User=pgexporter
Group=pgexporter
WorkingDirectory=/opt/postgres-exporter
EnvironmentFile=/etc/postgres-exporter/postgres-exporter.env
ExecStart=/usr/local/bin/postgres_exporter --web.listen-
address=:9187 --web.telemetry-path=/metrics
Restart=always
[Install]
WantedBy=multi-user.target

sudo systemctl daemon-reload

sudo systemctl enable --now postgres_exporter
```

Adding firewall rules

```
sudo firewall-cmd --add-port 9187/tcp
sudo firewall-cmd --add-port 9187/tcp --permanent
```

The exporter should now be up and scraping the local postgresql database and serving as an endpoint for prometheus on port 9187

2.2 Prometheus

Cause of the nature of prometheus as a time series database, correct time is important so it should be considered to set up ntp servers!

Installing prometheus binary from the prometheus github in var directory (directory can be changed)

```
sudo mkdir /var/lib/prometheus
for i in rules rules.d files_sd; do sudo mkdir -p
/etc/prometheus/${i}; done
sudo mkdir /opt/prometheus
cd /opt/prometheus
sudo curl -s
https://api.github.com/repos/prometheus/prometheus/releases/latest
\
| grep browser_download_url \
| grep linux-amd64 \
| cut -d '"' -f 4 \
| sudo wget -qi -
Sudo tar -xzvf prometheus-*.tar.gz
cd prometheus-*/
sudo cp prometheus promtool /usr/local/bin/
sudo cp -r prometheus.yml consoles/ console_libraries/
/etc/prometheus/
```

Creating config file where in static config all exporter has to be included with the ip addresses and port. In this example its 192.168.122.20:5432. Also custom labels for each exporter can be defined as well as the ip and port of the alertmanager in this case 192.168.122.21:9093

```
sudo nano /etc/prometheus/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:
          - 192.168.122.21:9093
# Load rules once and periodically evaluate them according to the
global 'evaluation_interval'.
  rule_files:
    - /etc/prometheus/rules/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.
  - job_name: "prometheus"
```

```
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
    - targets: ["localhost:9090"]
  - job_name: postgresql1
    metrics_path: /metrics
    static_configs:
    - targets:
      - 192.168.122.20:9187
      labels:
      cluster: cluster1
```

Adding user and set permissions

```
sudo useradd -s /sbin/nologin --system prometheus
sudo chown -R prometheus:prometheus /etc/prometheus
sudo chmod -R 775 /etc/prometheus/
sudo chown -R prometheus:prometheus /var/lib/prometheus/
```

Create systemd unit file

```
sudo nano /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
sudo systemctl daemon-reload
sudo systemctl enable --now prometheus.service
```

Adding firewall rules

```
sudo firewall-cmd --add-port 9090/tcp
sudo firewall-cmd --add-port 9090/tcp --permanent
```

Prometheus should now be up and scraping the exporters and providing a web-gui on port 9090

2.3 Alertmanager

Download alertmanager binary from the prometheus github in opt directory (directory can be changed)

```
sudo mkir /opt/alertmanager
cd /opt/alertmanager

sudo wget
https://github.com/prometheus/alertmanager/releases/download/v0.24.
0/alertmanager-0.24.0.linux-amd64.tar.gz

sudo tar -xvzf alertmanager-*.tar.gz
cd alertmanager-*/
sudo mv amtool alertmanager /usr/local/bin
sudo mkdir -p /etc/alertmanager
sudo mv alertmanager.yml /etc/alertmanager
```

Create user and set permissions

```
sudo useradd -rs /bin/false alertmanager
sudo chown -R alertmanager:alertmanager /opt/alertmanager
/etc/alertmanager/*
```

Create systemd unit file

```
sudo nano /etc/systemd/system/alertmanager.service
  [Unit]
  Description=Alert Manager
  Wants=network-online.target
  After=network-online.target
  [Service]
  Type=simple
  User=alertmanager
  Group=alertmanager
  ExecStart=/usr/local/bin/alertmanager \
  --config.file=/etc/alertmanager/alertmanager.yml \
  --storage.path=/opt/alertmanager
  Restart=always
  [Install]
  WantedBy=multi-user.target
sudo systemctl daemon-reload

sudo systemctl enable --now alertmanager.service
```

Adding firewall rules

```
sudo firewall-cmd --add-port 9093/tcp
sudo firewall-cmd --add-port 9093/tcp --permanent
```

Adding basic alerting rules to prometheus. Alerting rules are defined on the prometheus server under /etc/prometheus/rules/rules.yml

```
sudo nano /etc/prometheus/rules/rules.yml
  groups:
  - name: AllInstances
  rules:
  - alert: InstanceDown
      # Condition for alerting
      expr: up == 0
      for: 1m
      # Annotation - additional informational labels to store more
information
      annotations:
      title: 'Instance {{ $labels.job }} down'
      description: '{{ $labels.job }} of job {{ $labels.job }} has
been down for more than 1 minut$
      # Labels - additional labels to be attached to the alert
      labels:
      severity: 'critical'

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

sudo systemctl restart prometheus.service
```

Alertmanager should now be up waiting for alerts from prometheus on port 9093. Also the web gui can be accessed on port 9093.

2.4 Grafana

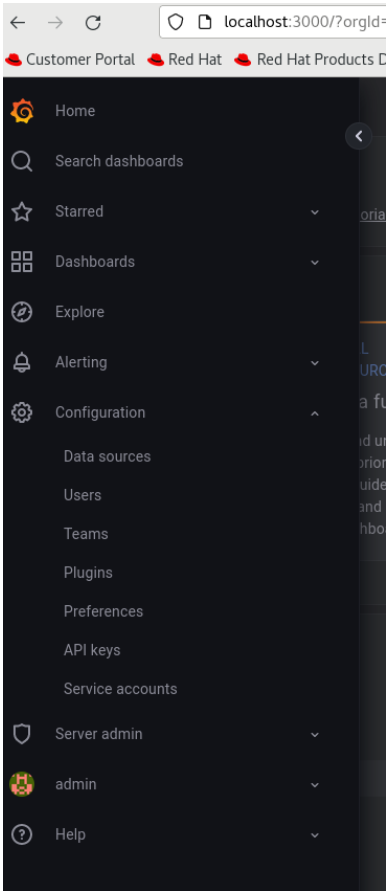
Grafana can be downloaded on the official grafana website as a rpm package and be installed locally.

```
wget https://dl.grafana.com/oss/release/grafana-9.2.6-1.x86_64.rpm
sudo yum install grafana-9.2.6-1.x86_64.rpm
sudo systemctl enable --now grafana.service
```

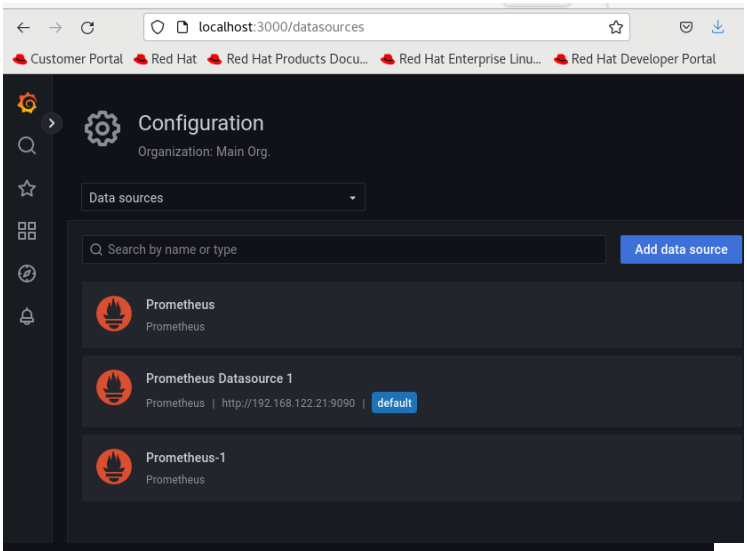
add ports to firewall if necessary

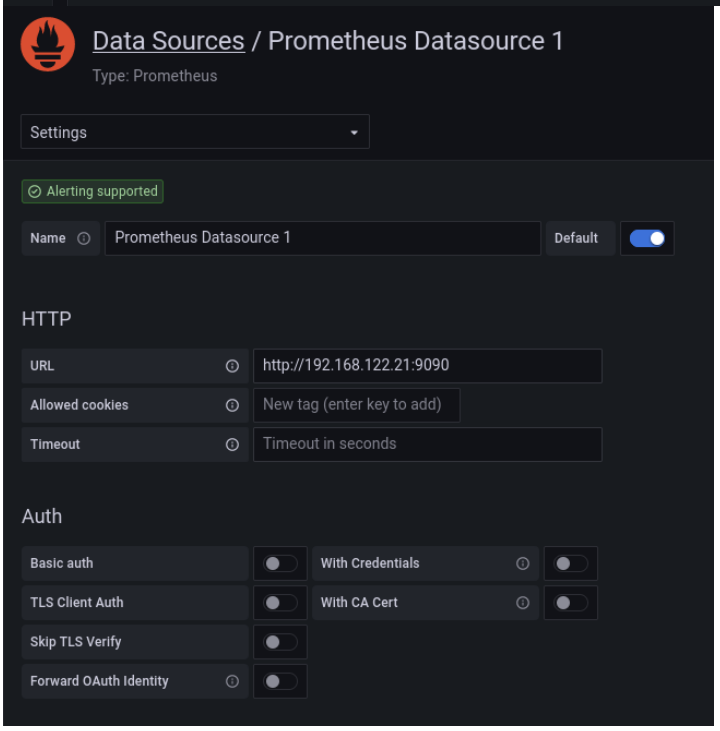
```
sudo firewall-cmd --add-port 3000/tcp
sudo firewall-cmd --add-port 3000/tcp --permanent
```

access Grafana dashboard on localhost or remote on port 3000 default user and pw is admin admin. To add Prometheus as a Datasource for Grafana, navigate in the top left to home expand the tab with the little arrow and under configuration select data sources



In the new Window click on add datasource. A list of different datasource providers appear. Select Prometheus and then enter a name for the datasource and the IP and port of the Prometheus server.





To add a dashboard into Grafana navigate to the dashboard icon and select import there you can upload and the example dashboard as a json file which is included in the git repo