PROMETHEUS AND GRAFANA INSTALLATION

Prometheus and grafana (To the monitoring server)

STEP I Create prometheus system group

```
sudo groupadd --system prometheus
```

```
sudo useradd -s /sbin/nologin --system -g prometheus prometheus
```

STEP II Create data & configs directories for Prometheus

Prometheus needs a directory to store its data. We will create this under */var/lib/prometheus*.

```
sudo mkdir /var/lib/prometheus
```

Prometheus' primary configuration files directory is */etc/prometheus/*. It will have some sub-directories:

```
for i in rules rules.d files_sd; do sudo mkdir -p /etc/prometheus/${i};
done
```

STEP III Download Prometheus

We need to download the latest release of Prometheus archive and extract it to get binary files. You can check releases from Prometheus releases Github page.

Install wget.

```
sudo apt update
sudo apt -y install wget curl vim
```

Then download the latest binary archive for Prometheus.

```
mkdir -p /tmp/prometheus && cd /tmp/prometheus

curl -s https://api.github.com/repos/prometheus/prometheus/releases/latest
| grep browser_download_url | grep linux-amd64 | cut -d '"' -f 4 | wget -qi
-
```

Extract the file:

```
tar xvf prometheus*.tar.gz
cd prometheus*/
```

Move the binary files to */usr/local/bin/* directory.

```
sudo mv prometheus promtool /usr/local/bin/
```

Check installed version:

```
$ prometheus --version
prometheus, version 2.39.1 (branch: HEAD, revision:
dcd6af9e0d56165c6f5c64ebbc1fae798d24933a)
  build user:       root@273d60c69592
  build date:       20221007-15:57:09
  go version:       go1.19.2
  platform:         linux/amd64

$ promtool --version
promtool, version 2.39.1 (branch: HEAD, revision:
dcd6af9e0d56165c6f5c64ebbc1fae798d24933a)
  build user:       root@273d60c69592
  build date:       20221007-15:57:09
  go version:       go1.19.2
  platform:         linux/amd64
```

Move Prometheus configuration template to /etc directory.

```
sudo mv prometheus.yml /etc/prometheus/prometheus.yml
```

Also move consoles and console_libraries to */etc/prometheus* directory:

```
sudo mv consoles/ console_libraries/ /etc/prometheus/
cd $HOME
```

STEP IV Configure Prometheus

```
sudo nano /etc/prometheus/prometheus.yml
```

The template configurations should look similar to below:

```
global:
  Scrape_interval: 15s # By default, scrape targets every 15 seconds.

  # Attach these labels to any time series or alerts when communicating
with
  # external systems (federation, remote storage, Alertmanager).
  # external_labels:
  #  monitor: 'codelab-monitor'

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']

  # Job for node_exporter
  - job_name: 'node_exporter'
    static_configs:
      - targets: [localhost:9100']

    # Example job for postgres_exporter
  - job_name: 'postgres_exporter'
    static_configs:
      - targets: ['postgres_exporter:9187']

  # Example job for cadvisor
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['cadvisor:9120']

  # Example job for adonisjs
  - job_name: 'customer_service'
    static_configs:
      - targets: ['host.docker.internal:3329']
    scrape_interval: 5s


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

  # Example job for alert_manager
alerting:
  alertmanagers:
    - static_configs:
      - targets: ['alertmanager:9093']
```

You can edit the file to your default liking and save it.

STEP V Create a Prometheus systemd Service unit file

To be able to manage Prometheus service with systemd, you need to explicitly define this unit file.

```
sudo tee /etc/systemd/system/prometheus.service<<EOF
[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
EOF
```

STEP VI Change directory permissions.

Change the ownership of these directories to Prometheus user and group.

```
for i in rules rules.d files_sd; do sudo chown -R prometheus:prometheus
/etc/prometheus/${i}; done
for i in rules rules.d files_sd; do sudo chmod -R 775 /etc/prometheus/${i};
done
sudo chown -R prometheus:prometheus /var/lib/prometheus/
```

STEP VII Reload systemd daemon and start the service.

```
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl enable prometheus
```

Check status using *systemctl status prometheus* command:

```
$ systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor
preset: enabled)
   Active: active (running) since Sun 2020-01-19 14:36:08 UTC; 14s ago
     Docs: https://prometheus.io/docs/introduction/overview/
 Main PID: 1397 (prometheus)
    Tasks: 7 (limit: 2377)
   Memory: 21.7M
   CGroup: /system.slice/prometheus.service
           └─1397 /usr/local/bin/prometheus
--config.file=/etc/prometheus/prometheus.yml
--storage.tsdb.path=/var/lib/prometheus --web.console.templates
```

```
Jan 19 14:36:08 deb10 prometheus[1397]: level=info
ts=2020-01-19T14:36:08.959Z caller=main.go:334 vm_limits="(soft=unlimited,
hard=unlimited)"
Jan 19 14:36:08 deb10 prometheus[1397]: level=info
ts=2020-01-19T14:36:08.960Z caller=main.go:648 msg="Starting TSDB ..."
```

If your server has a running firewall service, you'll need to open port *9090*.

```
sudo ufw allow 9090/tcp
```

Confirm that you can connect to port *9090* by accessing the Prometheus server IP address / DNS name in your web browser.

STEP VIII Reload systemd daemon and start the service.

```
sudo apt-get install -y apt-transport-https
sudo apt-get install -y software-properties-common wget
sudo wget -q -O /usr/share/keyrings/grafana.key
https://apt.grafana.com/gpg.key
```

Add this repository for stable releases:

```
echo "deb [signed-by=/usr/share/keyrings/grafana.key]
https://apt.grafana.com stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
```

After you add the repository:

```
sudo apt-get update

# Install the latest OSS release:
sudo apt-get install grafana
```

STEP IX Reload systemd daemon and start the service.

To start the service and verify that the service has started:

```
sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl status grafana-server
```

Configure the Grafana server to start at boot:

```
sudo systemctl enable grafana-server.service
```

To sign in to Grafana for the first time:

1. Open your web browser and go to http://localhost:3000/. The default HTTP port that Grafana listens to is 3000 unless you have configured a different port.

2. On the sign-in page, enter admin for the username and password.

3. Click Sign in. If successful, you will see a prompt to change the password.

4. Click OK on the prompt and change your password.

Install Exporters to servers for Collecting Metrics (Production, Development and Database Server)

STEP I Download Node Exporter for motoring the instance

```
wget
https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_e
xporter-1.3.1.linux-amd64.tar.gz
```

STEP II Extract Node Exporter and move binary

```
tar xvf node_exporter-1.3.1.linux-amd64.tar.gz
```

The content of the zip will be extracted in the current directory, the extracted directory will contain 3 files:

- LICENSE (license text file)
- node_exporter (binary)
- NOTICE (license text file)

You only need to move the binary file *node_exporter* to the */usr/local/bin* directory of your system. Switch to the node_exporter directory:

```
cd node_exporter-1.3.1.linux-amd64
```

And then copy the binary file with the following command:

```
sudo cp node_exporter /usr/local/bin
```

Then you can remove the directory that we created after extracting the zip file content:

```
# Exit current directory
cd ..

# Remove the extracted directory
rm -rf ./node_exporter-1.3.1.linux-amd64
```

STEP III Extract Node Exporter and move binary

```
sudo useradd --no-create-home --shell /bin/false node_exporter
```

And set the owner of the binary node_exporter to the recently created user:

```
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

STEP IV Extract Node Exporter and move binary

The Node Exporter service should always start when the server boots so it will always be available to be scrapped for information. Create the node_exporter.service file with nano:

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

And paste the following content in the file:

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

Close nano and save the changes to the file. Proceed to reload the daemon with:

```
sudo systemctl daemon-reload
sudo systemctl start node_exporter
```

```
sudo ufw allow 9100/tcp
```

As last step, access your server through the web browser at port 9100 and browse the metrics (**http://localhost:9100/metrics**). You should get an output in the browser similar to:

Install cadvisor for monitoring docker

```
sudo apt-get -y install cadvisor
```

Integrate the cadvisor exporter to Prometheus

First, you need to edit the prometheus.yml configuration file.

```
sudo systemctl stop prometheus
sudo nano /etc/prometheus/prometheus.yml
```

Then, add the new targets to scrape

```
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['localhost:9120']
```

Finally, reload the service

```
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl status prometheus
```

Install postgres exporter for monitoring postgres database

Download the Postgres Exporter Binary

```
mkdir /opt/postgres_exporter

cd /opt/postgres_exporter

wget
https://github.com/wrouesnel/postgres_exporter/releases/download/v0.5.1/pos
```

```
tgres_exporter_v0.5.1_linux-amd64.tar.gz

tar -xzvf postgres_exporter_v0.5.1_linux-amd64.tar.gz

cd postgres_exporter_v0.5.1_linux-amd64

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

Prepare the env File

```
cd /opt/postgres_exportersudo nano postgres_exporter.env

# or you can use the following to monitor all the databases available on
localhost
DATA_SOURCE_NAME="postgresql://postgres:postgres@localhost:5432/?sslmode=di
sable"
```

Setup the Postgres Exporter Service

```
sudo useradd -rs /bin/false postgres
sudo nano /etc/systemd/system/postgres_exporter.service
```

Next, put the following inside

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

Finally, enable and start the service

```
sudo systemctl daemon-reload
sudo systemctl start postgres_exporter
sudo systemctl enable postgres_exporter
sudo systemctl status postgres_exporter
```

Integrate the Postgres exporter to Prometheus
First, you need to edit the prometheus.yml configuration file.

```
sudo systemctl stop prometheus
sudo nano /etc/prometheus/prometheus.yml
```

Then, add the new targets to scrape

```
  - job_name: 'postgres_exporter'
    static_configs:
      - targets: ['localhost:9187']
```

Finally, reload the service

```
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl status prometheus
```

Downloading Alertmanager

```
cd ~
curl -LO
https://github.com/prometheus/alertmanager/releases/download/v0.25.0/alertm
anager-0.25.0.linux-amd64.tar.gz


tar xvf alertmanager-0.25.0.linux-amd64.tar.gz


sudo mv alertmanager-0.25.0.linux-amd64/alertmanager /usr/local/bin
sudo mv alertmanager-0.25.0.linux-amd64/amtool /usr/local/bin


sudo chown alertmanager:alertmanager /usr/local/bin/alertmanager
sudo chown alertmanager:alertmanager /usr/local/bin/amtool


rm -rf alertmanager-0.25.0.linux-amd64
alertmanager-0.25.0.linux-amd64.tar.gz
```

Configuring Alertmanager To Send Alerts

```
sudo mkdir /etc/alertmanager
sudo chown alertmanager:alertmanager /etc/alertmanager
```

```
sudo nano /etc/alertmanager/alertmanager.yml
```

```
global:
  resolve_timeout: 1m
  slack_api_url:
'https://hooks.slack.com/services/T01B7P7P6GZ/B048UJCEPT2/1pAxRqDEngYsyIKD0
l1HCYfa'

route:
  receiver: 'slack-notifications'


receivers:
- name: 'slack-notifications'
  email_configs:
    - to: 'omakei96@gmail.com'
      from: 'alertmanager@omakei-system.com'
      smarthost: smtp.mailtrap.io:2525
      auth_username: '131444ba61b99c'
      auth_identity: '131444ba61b99c'
      auth_password: 'caf6d5220683cc'
  slack_configs:
  - channel: '#baridi_api_logs'
    send_resolved: true
    icon_url: https://avatars3.githubusercontent.com/u/3380462
    title:  |-
      [{{ .Status | toUpper }}{{ if eq .Status "firing" }}:{{
.Alerts.Firing | len }}{{ end }}] {{ .CommonLabels.alertname }} for {{
.CommonLabels.job }}
      {{- if gt (len .CommonLabels) (len .GroupLabels) -}}
        {{" "}}(
        {{- with .CommonLabels.Remove .GroupLabels.Names }}
          {{- range $index, $label := .SortedPairs -}}
            {{ if $index }}, {{ end }}
            {{- $label.Name }}="{{ $label.Value -}}"
          {{- end }}
        {{- end -}}
        )
      {{- end }}
    text: >-
      {{ range .Alerts -}}
      *Alert:* {{ .Annotations.title }}{{ if .Labels.severity }} - `{{
.Labels.severity }}`{{ end }}
      *Description:* {{ .Annotations.description }}
      *Details:*
        {{ range .Labels.SortedPairs }} • *{{ .Name }}:* `{{ .Value }}`
        {{ end }}
      {{ end }}
```

Running Alertmanager

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


[Unit]
Description=Alertmanager
Wants=network-online.target
After=network-online.target

[Service]
User=alertmanager
Group=alertmanager
Type=simple
WorkingDirectory=/etc/alertmanager/
ExecStart=/usr/local/bin/alertmanager
--config.file=/etc/alertmanager/alertmanager.yml --web.external-url
http://your_server_ip:9093

[Install]
WantedBy=multi-user.target
```

Integrate the alert manager exporter to Prometheus

First, you need to edit the prometheus.yml configuration file.

```
sudo systemctl stop prometheus
sudo nano /etc/prometheus/prometheus.yml
```

Then, add the new targets to scrape

```
 alerting:
  alertmanagers:
    - static_configs:
      - targets: ['localhost:9093']
```

Finally, reload the service

```
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl status prometheus
```

Creating Alert Rules

```
sudo touch /etc/prometheus/rules.yml
sudo chown prometheus:prometheus /etc/prometheus/rules.yml
sudo nano /etc/prometheus/prometheus.yml
#add this line to prometheus.yml
rule_files:
  - rules.yml

sudo nano /etc/prometheus/rules.yml
# add this lines to rules.yml file
groups:
- name: alert.rules
  rules:
  - alert: EndpointDown
    expr: probe_success == 0
    for: 10s
    labels:
      severity: "critical"
    annotations:
      summary: "Endpoint {{ $labels.instance }} down"

sudo systemctl restart prometheus
sudo systemctl status prometheus
```