Prometheus and Grafana

Introduction to Prometheus and Grafana

Prometheus

Prometheus is a tool used to monitor computer systems and applications. It was created by SoundCloud in 2012 and has since become a popular choice for monitoring due to its strong features and active community.

Key Features of Prometheus:

- **Time-Series Database:** Prometheus stores data as a series of timestamps and values, making it easy to track changes over time.
- **Query Language:** Prometheus uses a language called PromQL to search and analyze the collected data.
- **Metrics Collection:** It gathers metrics from different jobs or applications by regularly checking their status.
- **Service Discovery:** Prometheus can automatically find targets (services to monitor) using various methods like Kubernetes.
- **Alerting:** It can create alerts based on specific conditions and send them to Alertmanager for notification.
- **Basic Visualization:** Prometheus has a simple built-in tool to view and explore the data.

Grafana

Grafana is a tool used to create visual representations of data, making it easier to understand and analyze. It was developed by Grafana Labs and is widely used alongside Prometheus.

Key Features of Grafana:

- **Supports Multiple Data Sources:** Grafana can connect to various data sources like Prometheus, InfluxDB, and Elasticsearch.
- **Custom Dashboards:** Users can create personalized dashboards to display the data they need.
- **Rich Visualizations:** Grafana offers many types of charts and graphs to visualize data effectively.
- **Alerting:** It can set up alerts that notify users through channels like email or Slack.
- **Plugins:** Grafana supports plugins to extend its functionality, allowing for more data sources and visualization options.

How Prometheus and Grafana Work Together

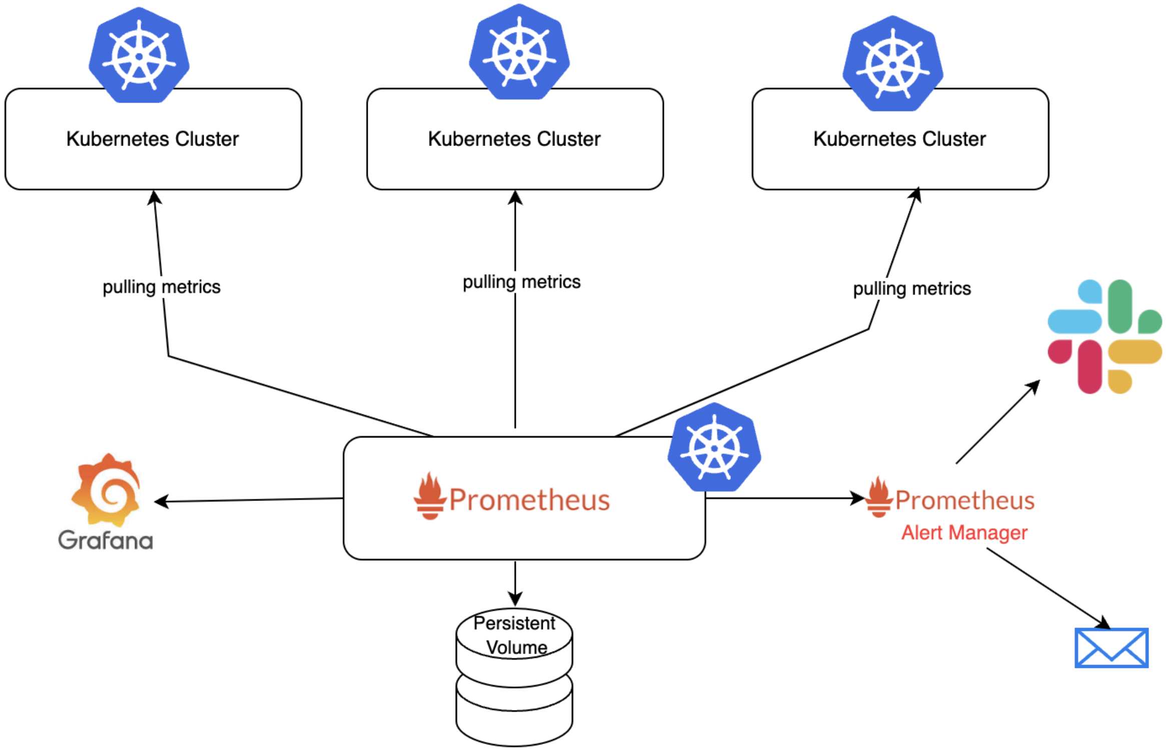
Prometheus and Grafana are often used together to provide a complete monitoring solution:

1. **Data Collection:** Prometheus collects data (metrics) from various sources.
2. **Storage:** Prometheus stores this data in its database.
3. **Visualization:** Grafana retrieves data from Prometheus.
4. **Dashboards:** Users create dashboards in Grafana to visualize the data.
5. **Alerting:** Both tools can send alerts based on the data, ensuring users are notified of important events.

Benefits of Using Prometheus and Grafana

- **Scalability:** They can handle large amounts of data, suitable for small to large environments.
- **Flexibility:** They offer powerful tools for querying, visualizing, and alerting on data.
- **Community and Ecosystem:** A strong community supports both tools, offering regular updates and plugins.
- **Open Source:** Both tools are free to use and customizable, providing a cost-effective monitoring solution.

Prometheus and Grafana Setup for Monitoring Kubernetes Clusters



This diagram illustrates the setup for monitoring Kubernetes clusters using Prometheus and Grafana.

Components:

1. **Kubernetes Clusters:**
   - Represented by the Kubernetes logos, these are the clusters from which Prometheus pulls metrics.
2. **Prometheus:**
   - Central to the setup, Prometheus pulls metrics from the Kubernetes clusters.

   - It stores these metrics in a time-series database.
3. **Persistent Volume:**
   - Used by Prometheus to store data persistently, ensuring that metrics are not lost if Prometheus restarts.
4. **Grafana:**
   - Connects to Prometheus to retrieve the stored metrics.
   - Provides a user interface for visualizing these metrics in customizable dashboards.
5. **Prometheus Alert Manager:**
   - Processes alerts based on rules defined in Prometheus.
   - Sends notifications to various channels (e.g., Slack, Email) when certain conditions are met.

Workflow:

1. **Pulling Metrics:**
   - Prometheus scrapes metrics from the Kubernetes clusters at regular intervals.
2. **Storing Data:**
   - The collected metrics are stored in a time-series database within Prometheus.
   - This data is kept on a persistent volume to prevent data loss during restarts.
3. **Visualization:**
   - Grafana queries Prometheus for the stored metrics data.
   - Users create dashboards in Grafana to visualize and analyze the data.
4. **Alerting:**
   - Prometheus Alert Manager receives alert rules from Prometheus.
   - When conditions defined in the rules are met, alerts are triggered.
   - Notifications are sent through configured channels like Slack and Email.

Notification Channels:

- **Slack:** Alerts can be sent to Slack for real-time notifications within a team chat.
- **Email:** Alerts can also be sent via email to notify administrators or stakeholders.

Lab or Hands-on Activity:

Create 2 Amazon Linux ec2 Servers.

1. K8S Server (to trigger eksctl command to setup EKS cluster)
2. Monitoring server (to setup Prometheus and Grafana)

On k8s-server:

AWS Credentials:

aws configure

AWS_ACCESS_KEY_ID=AKIA6GBMFJPZxxxxxxx
AWS_SECRET_ACCESS_KEY=oVQJav/VTBzqGwfYJxxxxxxxxxxx
region=us-east-1

Kubernetes CLI (kubectl) Setup

Ref:https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/

1. Download the latest version of kubectl:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
```

2. Install kubectl:

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

EKSCTL Setup

Reference: https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-DevelopmentGuide/setting-up-eksctl.html

1. Download and install eksctl:

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

Create cluster:

```
eksctl create cluster test
```

On monitoring server:

Prometheus Setup

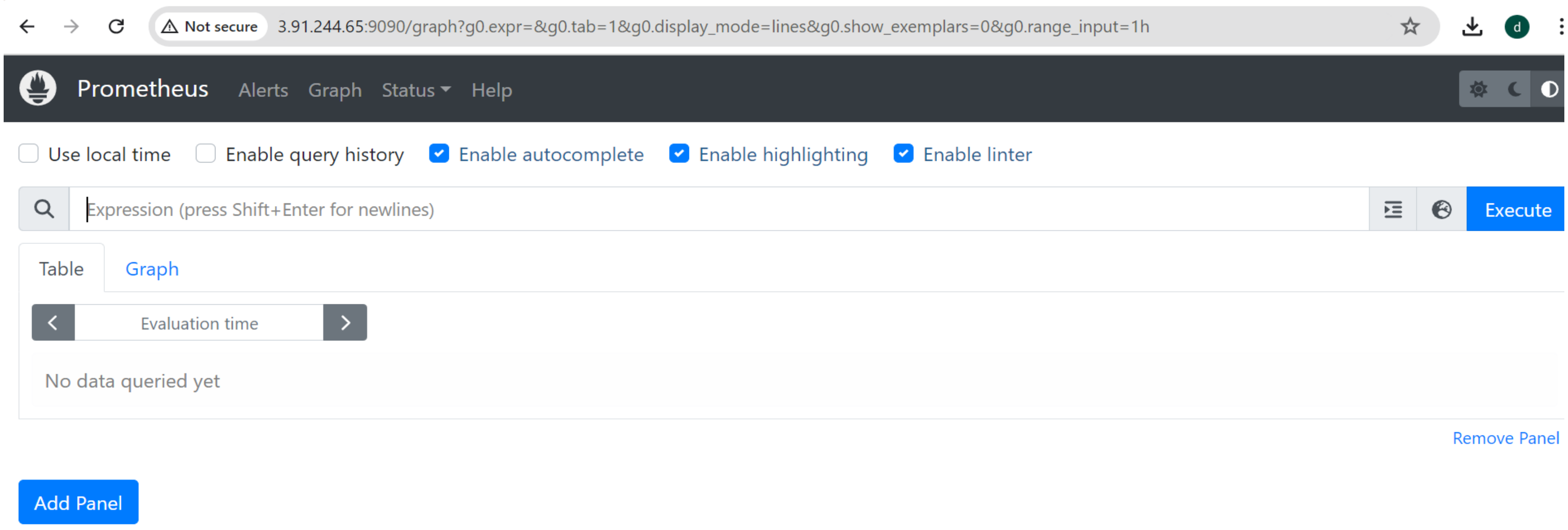
1. Download and extract Prometheus:

```
wget https://github.com/prometheus/prometheus/releases/download/v2.53.1/prometheus-2.53.1.linux-amd64.tar.gz
tar -xvf prometheus-2.53.1.linux-amd64.tar.gz
```

- Start the Prometheus:

```
cd prometheus-2.53.1.linux-amd64/
./prometheus &
```

Access Prometheus on port 9090: public_ip:9090



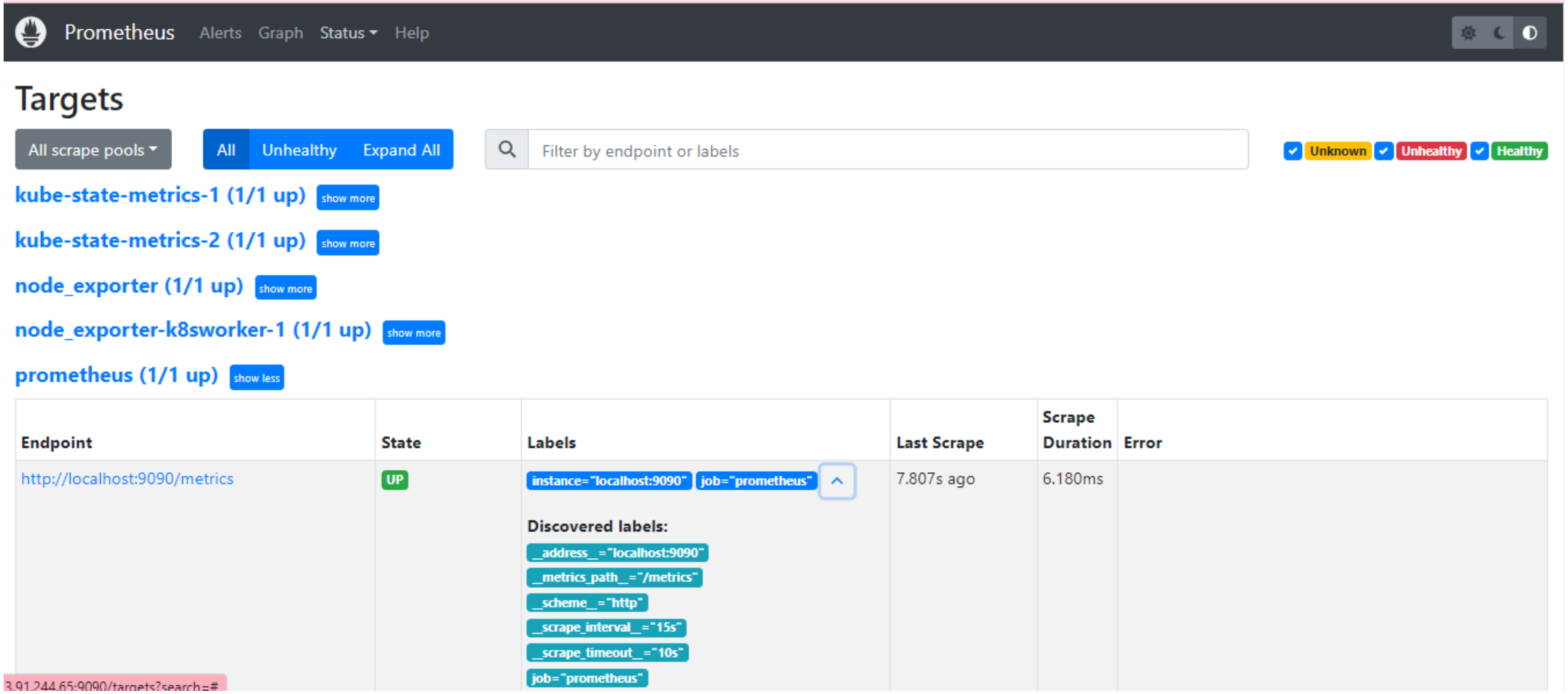
⚠ Not secure 3.91.244.65:9090/graph?g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.range_input=1h

Prometheus

□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

Expression (press Shift+Enter for newlines)

No data queried yet



Prometheus Alerts Graph Status Help

Targets

All scrape pools | All | Unhealthy | Expand All | Filter by endpoint or labels | ☑ Unknown ☑ Unhealthy ☑ Healthy

kube-state-metrics-1 (1/1 up) show more

kube-state-metrics-2 (1/1 up) show more

node_exporter (1/1 up) show more

node_exporter-k8sworker-1 (1/1 up) show more

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus" Discovered labels: __address__="localhost:9090" __metrics_path__="/metrics" __scheme__="http" __scrape_interval__="15s" __scrape_timeout__="10s" job="prometheus"	7.807s ago	6.180ms	

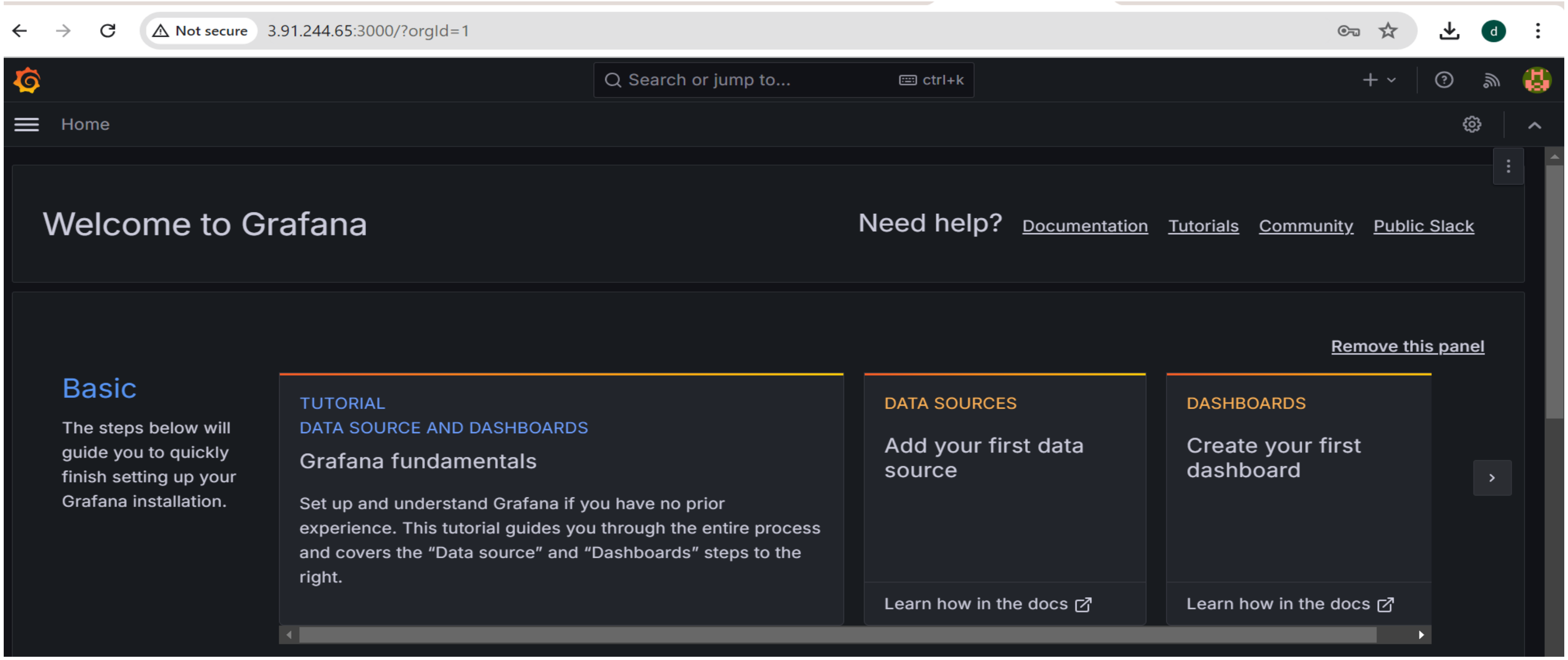
3.91.244.65:9090/targets?search=#

Grafana Setup

1. Download and extract Grafana:

```
wget https://dl.grafana.com/enterprise/release/grafana-enterprise-11.1.0.linux-amd64.tar.gz
tar -xvf grafana-enterprise-11.1.0.linux-amd64.tar.gz
cd grafana-enterprise-11.1.0/
cd bin/
./grafana-server &
```

Access Prometheus on port 3000:public_ip:3000



Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the “Data source” and “Dashboards” steps to the right.

DATA SOURCES
Add your first data source

Learn how in the docs

DASHBOARDS
Create your first dashboard

Learn how in the docs

➢ **Node Exporter & kube-state-metrics**

- **Node Exporter:** Concentrates on gathering metrics about the underlying infrastructure of the cluster, like CPU, memory, disk usage of the physical nodes running Kubernetes.

- **kube-state-metrics:** Focuses on the health and state of Kubernetes objects themselves, such as deployments, pods, services, and their current status (running, pending, etc.).

Node Exporter Cofiguration:

Node Exporter is a Prometheus exporter for hardware and OS metrics. It allows Prometheus to collect metrics from the underlying system, such as CPU usage, memory usage, disk space, and network statistics.

Key Features:

1. **Metric Collection:**
   - Node Exporter gathers a wide range of system-level metrics, including:
     - CPU usage
     - Memory usage
     - Disk I/O
     - Network I/O
     - File system statistics
     - System load
     - Temperature
     - Hardware information (e.g., BIOS, motherboard)
2. **Ease of Use:**
   - Simple to install and configure on various Linux distributions.
   - Exposes metrics in a format that Prometheus understands.

Node Exporter Setup Need to be installed on monitoring server and k8s-worker nodes 1& 2

1. Download and extract Node Exporter:

```
wget https://github.com/prometheus/node_exporter/releases/download/v1.7.0/node_exporter-1.7.0.linux-amd64.tar.gz
```

```
tar -xvf node_exporter-1.7.0.linux-amd64.tar.gz
sudo mv node_exporter-1.7.0.linux-amd64/node_exporter /usr/local/bin/
```

2. Create a system user for Node Exporter:

```
sudo useradd --system --no-create-home --shell /bin/false node_exporter
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
sudo chmod +x /usr/local/bin/node_exporter
```

3. Create a systemd service for Node Exporter:

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

Add the following content to the service file:

```
 [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
```

4. Enable and start the Node Exporter service:

```
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
systemctl status node_exporter.service
```

Prometheus Configuration

Configure Prometheus to Scrape Node Exporter:

1. Edit the Prometheus configuration file:

```
cd prometheus-2.53.1.linux-amd64/
vim prometheus.yml
```

Add the following job configurations:

```
scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "node_exporter"
    static_configs:
      - targets: ["localhost:9100"]

  - job_name: "node_exporter-k8sworker-1"
    static_configs:
      - targets: ["3.81.25.36:9100"] #public-ip k8s-workernode-1

  - job_name: "node_exporter-k8sworker-2"
    static_configs:
      - targets: ["54.167.217.255:9100"]  #public-ip k8s-workernode-2
  - job_name: "kube-state-metrics"
    static_configs:
      - targets: ["public_ip:nodeport"]
```

- Restart the Prometheus:

```
Pgrep prometheus   or ps  -ef |  grep prometheus
Kill pid
./prometheus    &
```

Kube State Metrics

Kube State Metrics (KSM) is a monitoring tool for Kubernetes clusters. It collects and exposes metrics about the state of the various Kubernetes objects (such as Deployments, Nodes, Pods, etc.) and makes these metrics available for monitoring and alerting.

kube-state-metrics Setup

1. Clone the kube-state-metrics repository and apply the standard examples:

```
git clone https://github.com/kubernetes/kube-state-metrics.git
kubectl apply -f kube-state-metrics/examples/standard
```

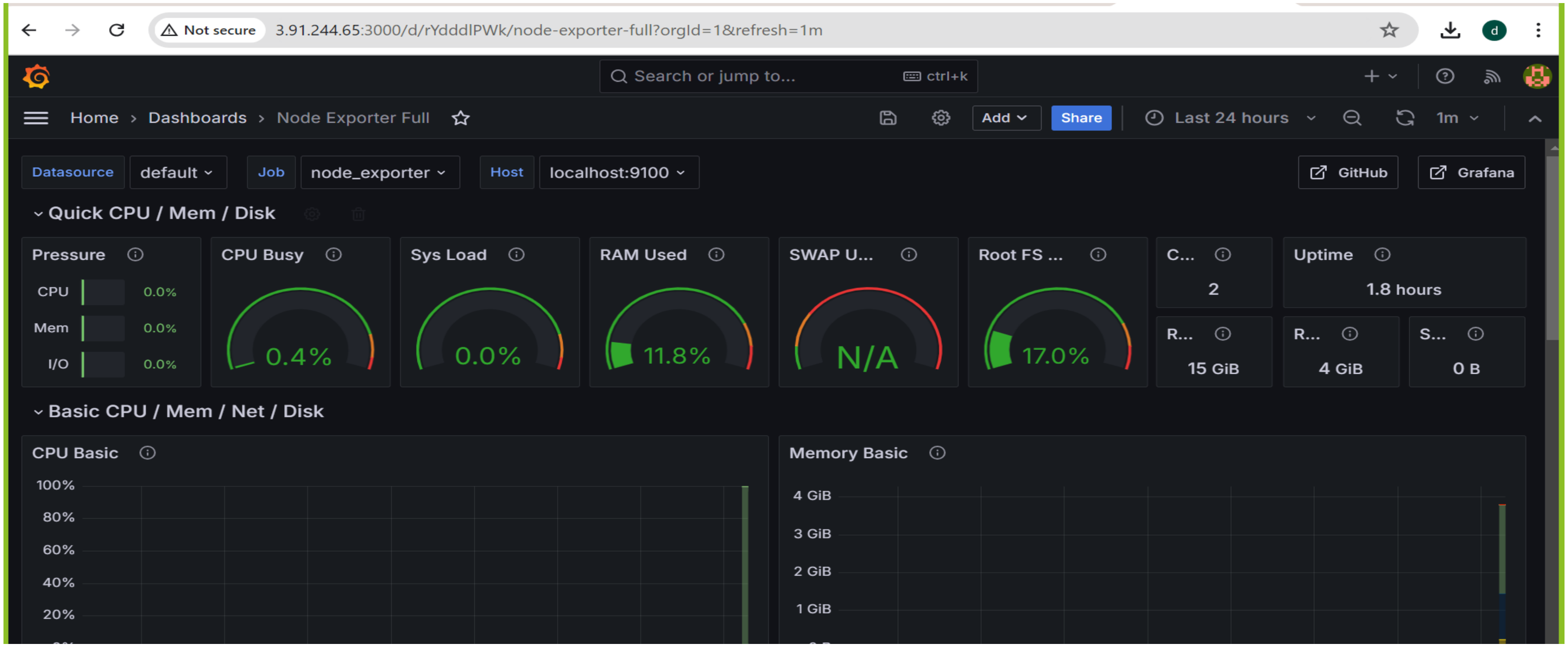
3. Verify the deployment and expose the service:

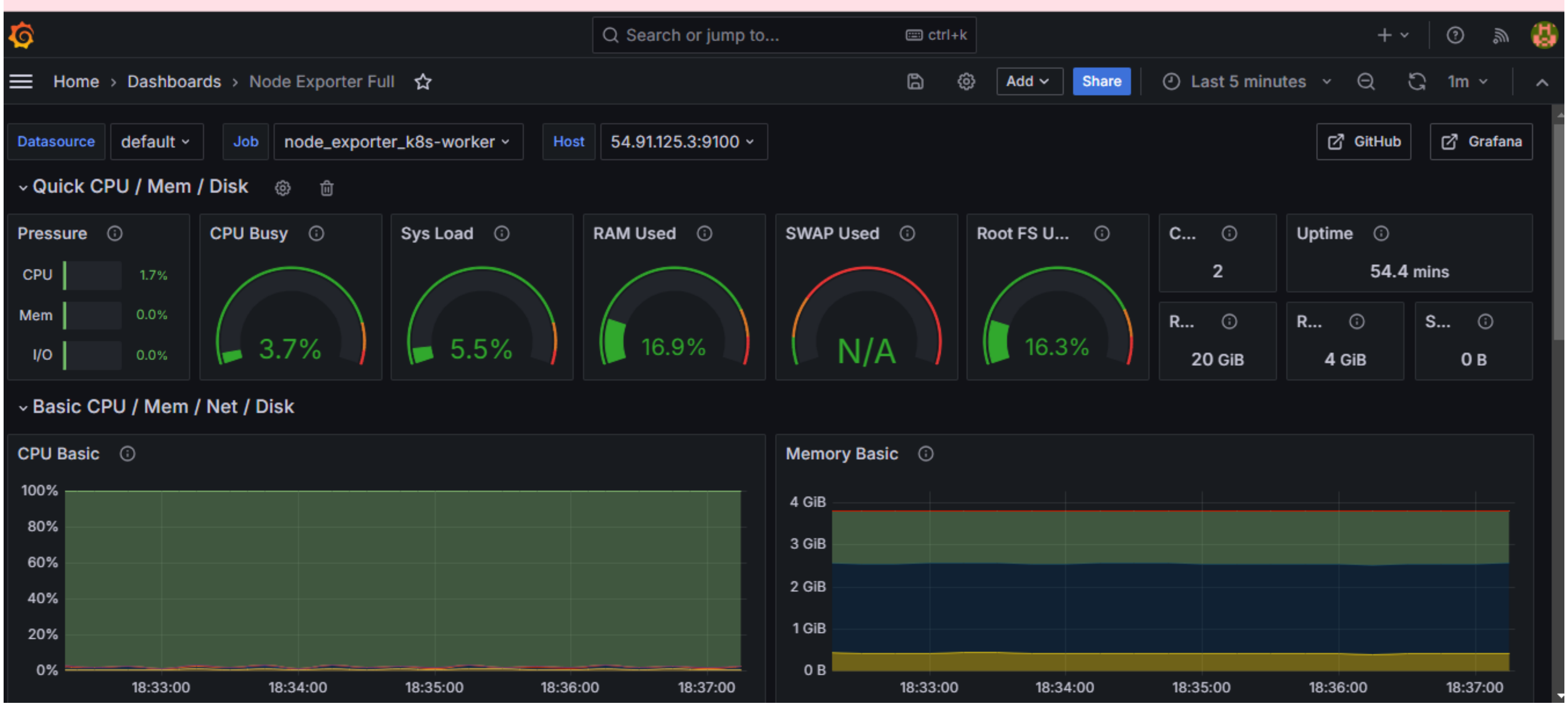
```
kubectl get all -n kube-system
kubectl expose service kube-state-metrics --type=NodePort --name kube-state-metrics-ext -n kube-system
```

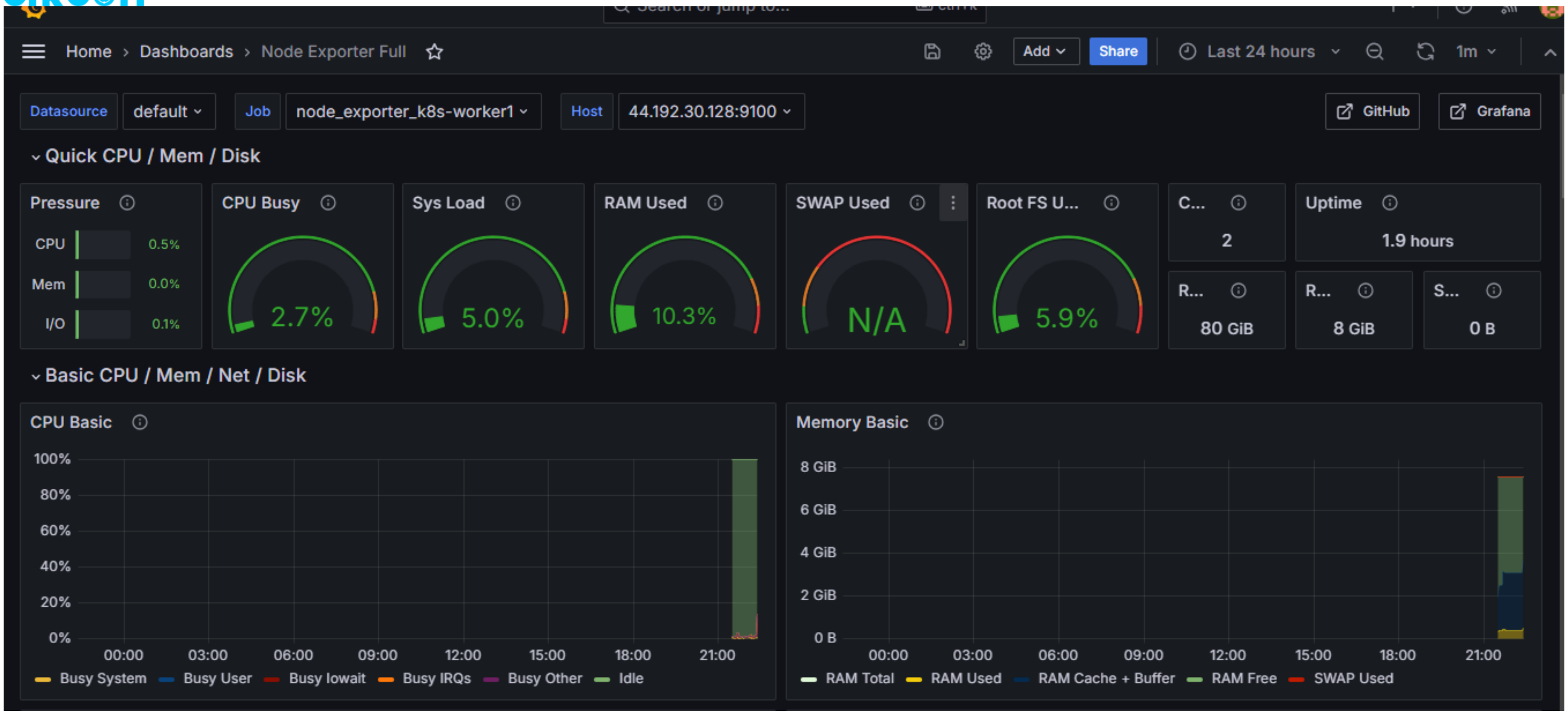
Grafana Dashboards

1860-Node Exporter Full
6417 -Kubernetes Cluster (Prometheus)
15451 -Kubernetes / Compute Resources / Node (Groups)
9964- jenkins dashboard
13105-K8S Dashboard

FINAL OUTPUT:

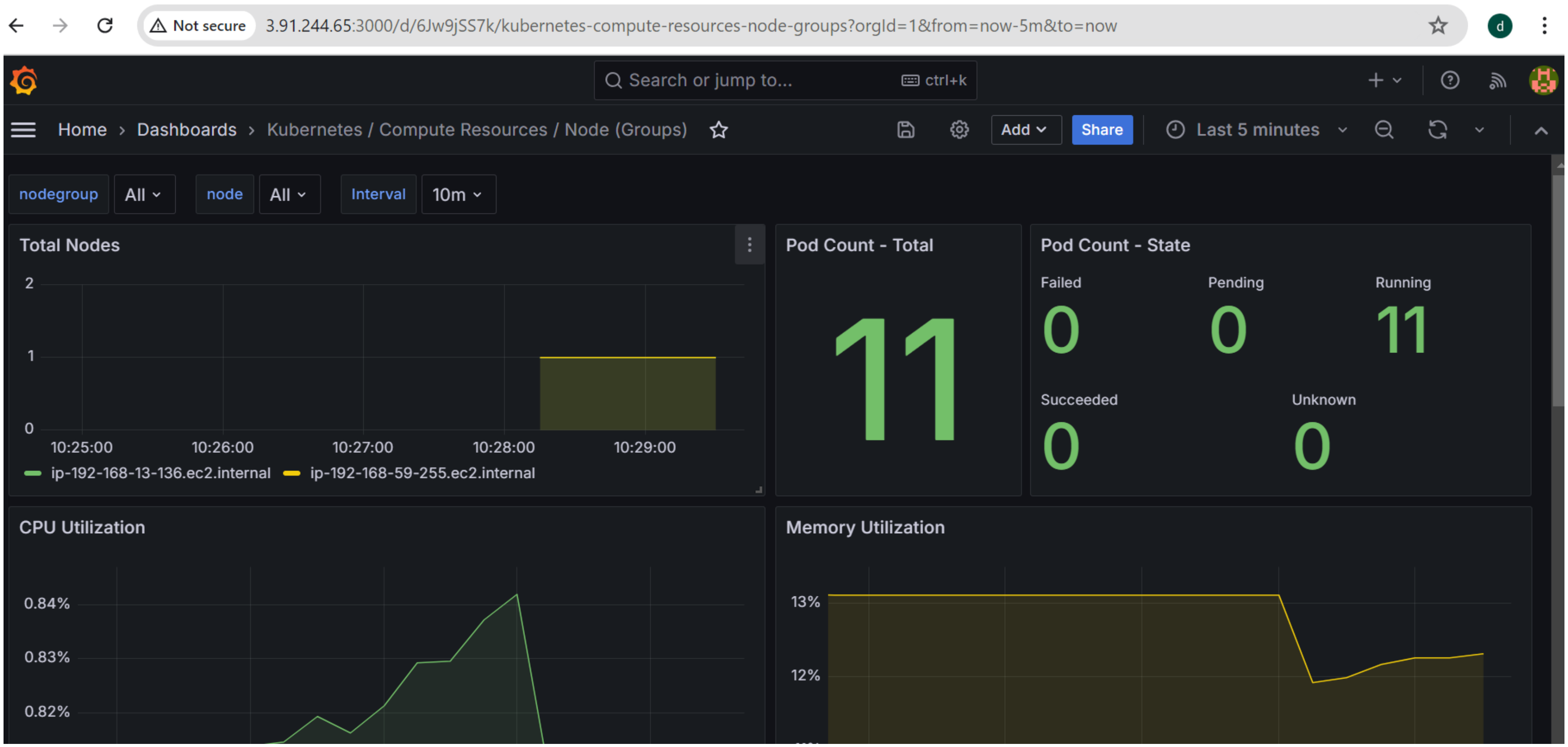



Search or jump to...
ctrl+k
Home › Dashboards › Node Exporter Full
Add
Share
Last 5 minutes
1m
Datasource
default
Job
node_exporter_k8s-worker
Host
54.91.125.3:9100
GitHub
Grafana
Quick CPU / Mem / Disk
Pressure
CPU 1.7%
Mem 0.0%
I/O 0.0%
CPU Busy
3.7%
Sys Load
5.5%
RAM Used
16.9%
SWAP Used
N/A
Root FS U...
16.3%
C...
2
Uptime
54.4 mins
R...
20 GiB
R...
4 GiB
S...
0 B
Basic CPU / Mem / Net / Disk
CPU Basic
100%
80%
60%
40%
20%
0%
18:33:00
18:34:00
18:35:00
18:36:00
18:37:00
Memory Basic
4 GiB
3 GiB
2 GiB
1 GiB
0 B
18:33:00
18:34:00
18:35:00
18:36:00
18:37:00



Home › Dashboards › Node Exporter Full

Datasource: default
Job: node_exporter_k8s-worker1
Host: 44.192.30.128:9100



Not secure 3.91.244.65:3000/d/6Jw9jSS7k/kubernetes-compute-resources-node-groups?orgId=1&from=now-5m&to=now

Home › Dashboards › Kubernetes / Compute Resources / Node (Groups)

nodegroup All · node All · Interval 10m