# A Guide to use Prometheus (public)

# 

**1.Introduction**

**2.Components of Prometheus**

**3.Architecture**

**4.Deployment process**

**5.Patching Process**

**Introduction**

Prometheus is an open-source system monitoring and alerting toolkit

Prometheus collects and stores its metrics as time series data, i.e. metrics information is stored with e timestamp at which it was recorded, alongside optional key-value pairs called labels

### Components

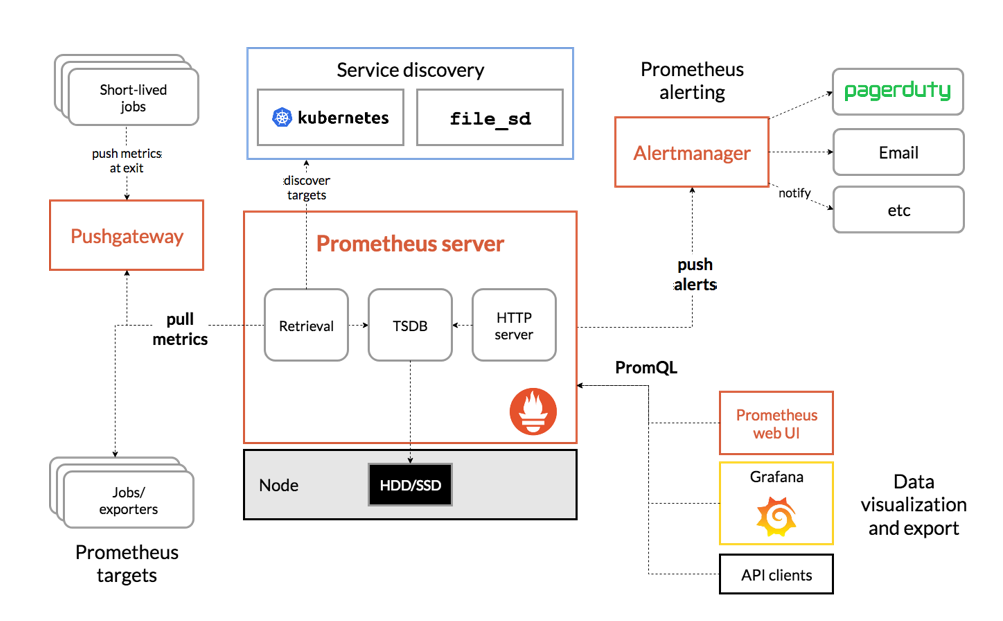
The Prometheus ecosystem consists of multiple components, many of which are optional:

* the main [Prometheus server](https://github.com/prometheus/prometheus) which scrapes and stores time series data
* [client libraries](https://prometheus.io/docs/instrumenting/clientlibs/) for instrumenting application code
* a [push gateway](https://github.com/prometheus/pushgateway) for supporting short-lived jobs
* special-purpose [exporters](https://prometheus.io/docs/instrumenting/exporters/) for services like HAProxy, StatsD, Graphite, etc.
* an [alertmanager](https://github.com/prometheus/alertmanager) to handle alerts

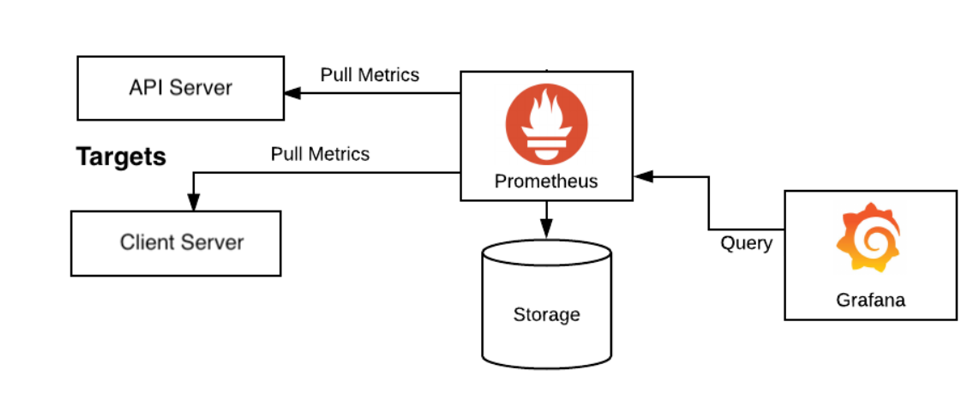
Prometheus runs as container in a HaCT managed Kubernetes cluster.

### Architecture

This diagram illustrates the architecture of Prometheus and some of its ecosystem components:



**Flow Concepts of Prometheus**



**Prometheus installation using Helm chart :**

**Monitoring Deployment process**

We are having 2 Monitoring Solution

1.Prometheus

2.Grafana

**Deployment Process**

We are using Helm to deploy the solution



        helm search repo  <chart\_name)



       helm repo list

**To add Prometheus Repo**

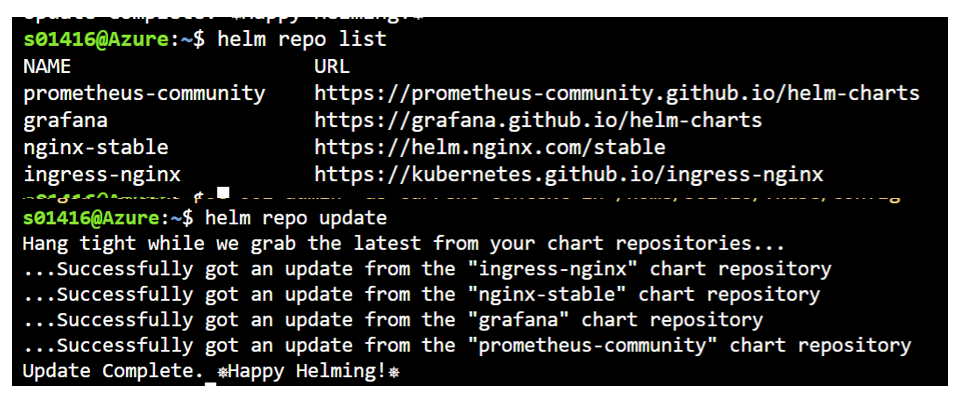
    helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

**To Install Prometheus**

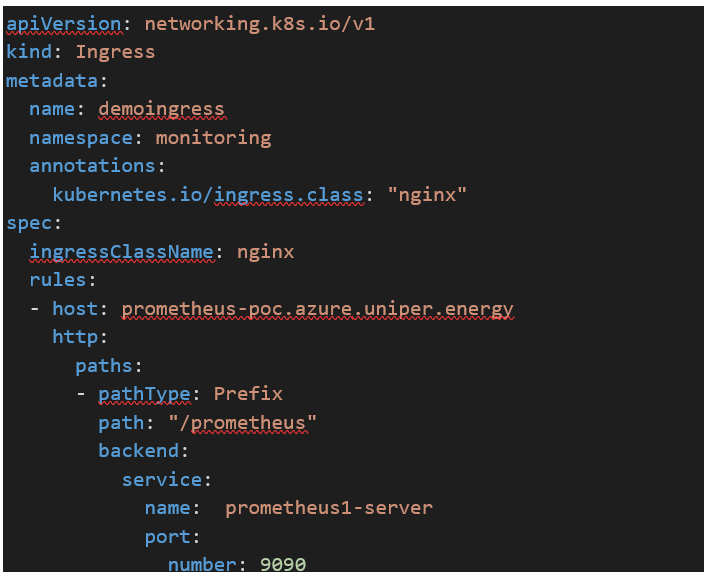
helm install prometheus prometheus-community/prometheus --namespace monitoring



helm repo update

****

Using Ingress we can configure Prometheus as below**:**



In order to deploy ingress.yaml , we need valid  host name and certificate so submit a request to Add certificate / host name entry .

Upload the certificate in your local and generate secret as below .

**kubectl create secret tls grafana-tls --key grafana-poc.key --cert grafana-poc\_azure\_uniper\_energy.crt -n monitoring**

Once deployment is completed , output as below .

Use Prometheus as data source in Grafana for further process improvements**.**

A sample query output from Prometheus :

**Prometheus Management**

**Patching Activity :**

1.Check for the release / latest version in helm community for prometheus on monthly basis .

2. Add the corresponding helm chart with latest version

a**.Get Repo Info**

helm repo add  prometheus-community <https://prometheus-community.github.io/helm-charts>

helm repo update

b.**Do the helm upgrade and update the repo list** .

Upgrade the Prometheus Helm as below

helm upgrade -f [your\_release\_name] prometheus-community/prometheus

Replace [your\_release\_name] with the name of the release you used to install Prometheus.

c**. Output after upgrade**:

After running helm upgrade, you shouldss see the following output:

Release "[your\_release\_name]" has been upgraded. Happy Helming!

NAME: [your\_release\_name]

LAST DEPLOYED: Thu Dec 10 16:41:33 2020

NAMESPACE: default

STATUS: deployed

REVISION: 2

TEST SUITE: None

NOTES:

The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:

[your\_release\_name]-prometheus-server.default.svc.cluster.local

**How to add  Targets for Prometheus** :

$ kubectl get cm prometheus-server -o yaml

- job\_name: 'prometheus'

# metrics\_path defaults to '/metrics'

# scheme defaults to 'http'.

static\_configs:

- targets: ['localhost:9090']

Let’s add another static endpoint:

$ kubectl edit cm prometheus-server

Add a new job:

- job\_name: 'traefik'

static\_configs:

- targets: ['traefik-prometheus:9100]

annotations:

prometheus.io/port: 9216

prometheus.io/scrape: true

We must tell Prometheus to scrape the service or pod and provide data about the port that exposes metrics.

**Sample Metrics to get from Promethues :**

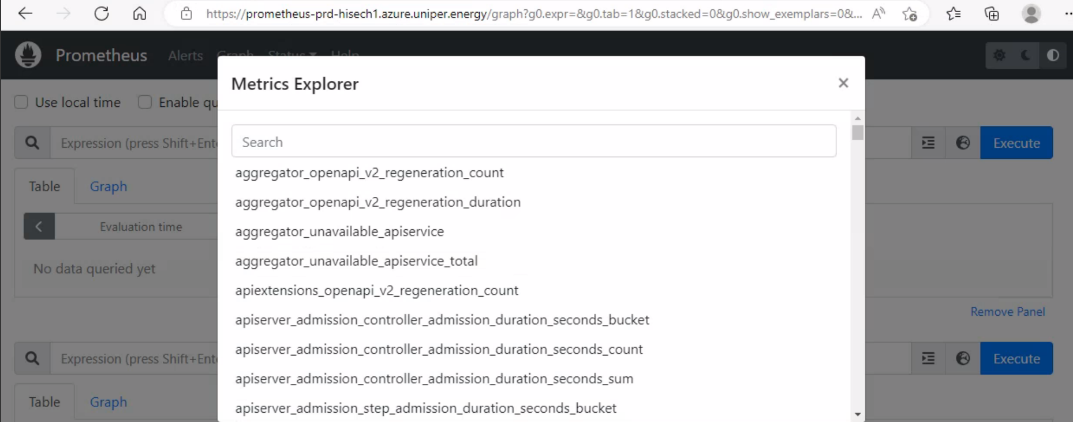
CPU Usage : sum by (cpu)(node\_cpu\_seconds\_total{mode!="idle"})

Memory Usage : node\_memory\_Active\_bytes/node\_memory\_MemTotal\_bytes\*100

Refer <https://www.tigera.io/learn/guides/prometheus-monitoring/prometheus-metrics/> for more metrics

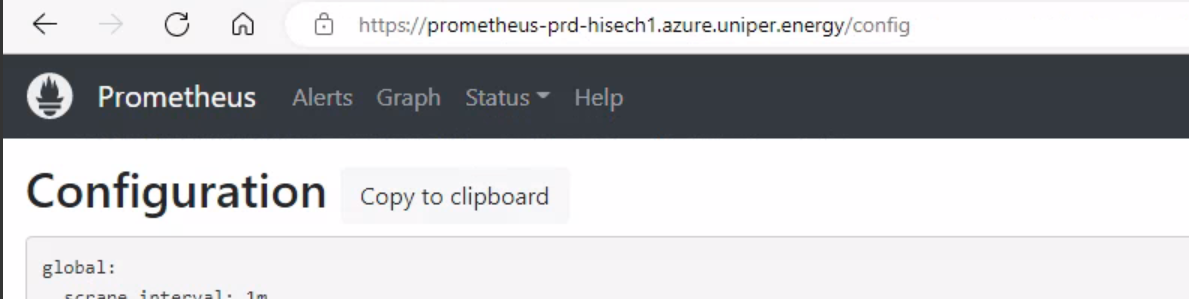
**Available Metrics in Prometheus :**

**Click on the button behind execute to check the available metrics .**



**To verify newly added Targets :**

Click on the configuration and search for Scrape Config to verify the same .



Check for the file name



**How can I use Prometheus?**

HaCT team owns the responsibility of installing / deleting Prometheus in requested shared clusters .

Application team can access Prometheus by below options.

1.Application team can use the existing Prometheus available in all shared clusters , else can request a new one in their own namespace , **but recommended is to use the central Prometheus instance than own setup .**

2.HaCT team will be responsible for managing the Prometheus within shared clusters and application team should own and manage the Prometheus if it comes as a new installation .

3.Application team can send a mail to Hact AKS support to use existing Prometheus URL’s .

With the help of below URL’s they can login and check the Prometheus else a new Prometheus can also be added in their requested namespace also .

**NOTE** : No user id and password is required expect the URL hit .

**Where do I find the technical details to connect?**

Every cluster contains a namespace “**monitoring**” where Prometheus is installed .

Below are the URL list .

|  |  |
| --- | --- |
| **Cluster Name** | **Prometheus URL** |
| cmcaks-pre-001 | https://prometheus-pre.azure.uniper.energy/ |
| cmcaks-pre-hisech-001 | https://prometheus-pre-hisech1.azure.uniper.energy/ |
| cmcaks-pre-hisech-002 | https://prometheus-pre-hisech2.azure.uniper.energy/ |
| cmcaks-pre-hisech-003 | <https://prometheus-pre-hisech3.azure.uniper.energy/> |
| cmcaks-pre-hisech-004 | <https://prometheus-pre-hisech4.azure.uniper.energy/> |
| cmcaks-prd-hisec-001 | https://prometheus-prd-hisech1.azure.uniper.energy/ |
| cmcaks-prd-hisec-002 | <https://prometheus-prd-hisech2.azure.uniper.energy/> |
| cmcaks-prd-hisec-003 | <https://prometheus-prd-hisech3.azure.uniper.energy/> |
| cmcaks-prd-hisec-004 | <https://prometheus-prd-hisech4.azure.uniper.energy/> |
| cmcaks-prd-001 | https://prometheus-prd.azure.uniper.energy/ |
|  |  |

**Example on how to use Prometheus**.

A sample query output from Prometheus :

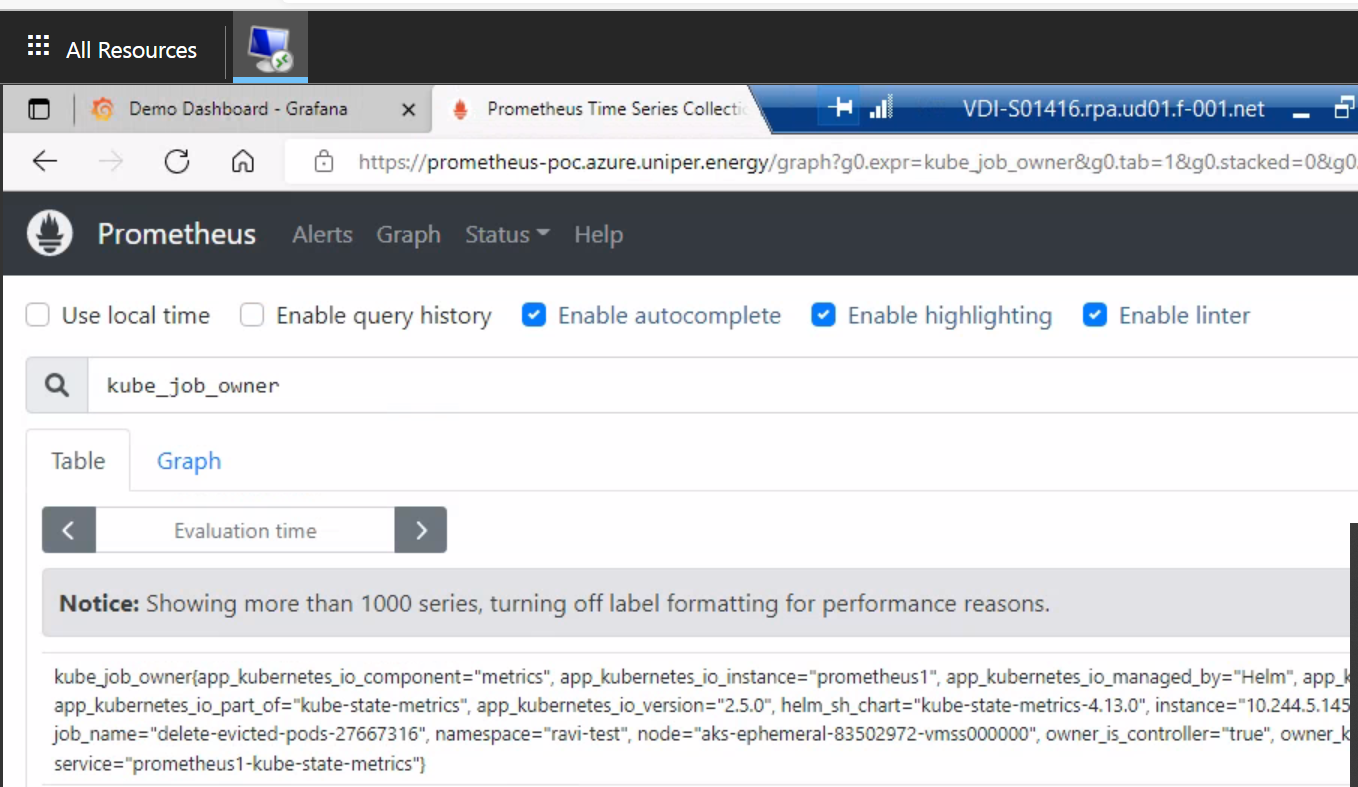
1.Click on the Prometheus URL based on cluster .

2.Click on Graph and provide the required query as needed .

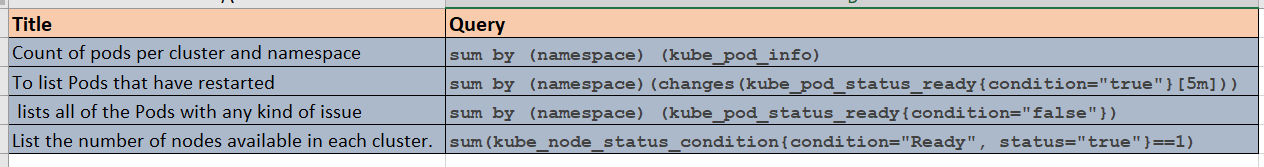
3.Click on execute to get the required output .

Using Prometheus Monitoring on a Kubernetes cluster, we can track the overall health, performance, and behavior of our system as below .





Some of the examples :



**Authentication** :

As Prometheus has no AD authentication , we can provide the URL to the application owners and they can create their own panel .