Data Service Server Monitoring with Prometheus

User’s Manual

V1.0.2

ADVANTECH WISE-PaaS IoT Edge Intelligence

Revision History

Date	Version	Author	Reviewer	Description
2019-03-05	1.0.0	Evelyn Tang Evelyn.Tang@advantech.com.tw		First version released
2019-03-07	1.0.1	Evelyn Tang Evelyn.Tang@advantech.com.tw		Paragraph adjustment Add architecture image
2019-03-08	1.0.2	Evelyn Tang Evelyn.Tang@advantech.com.tw		Change name to "Data Service Server"
2019-05-02	1.0.3	Evelyn Tang Evelyn.Tang@advantech.com.tw		Add MongoDB exporter and PV dashboards.
2019-06-18	1.0.4	Evelyn Tang Evelyn.Tang@advantech.com.tw		Add AlertManager

Table of Contents

Table of Figures

1 General Introduction

1.1 Data Service Server Architecture

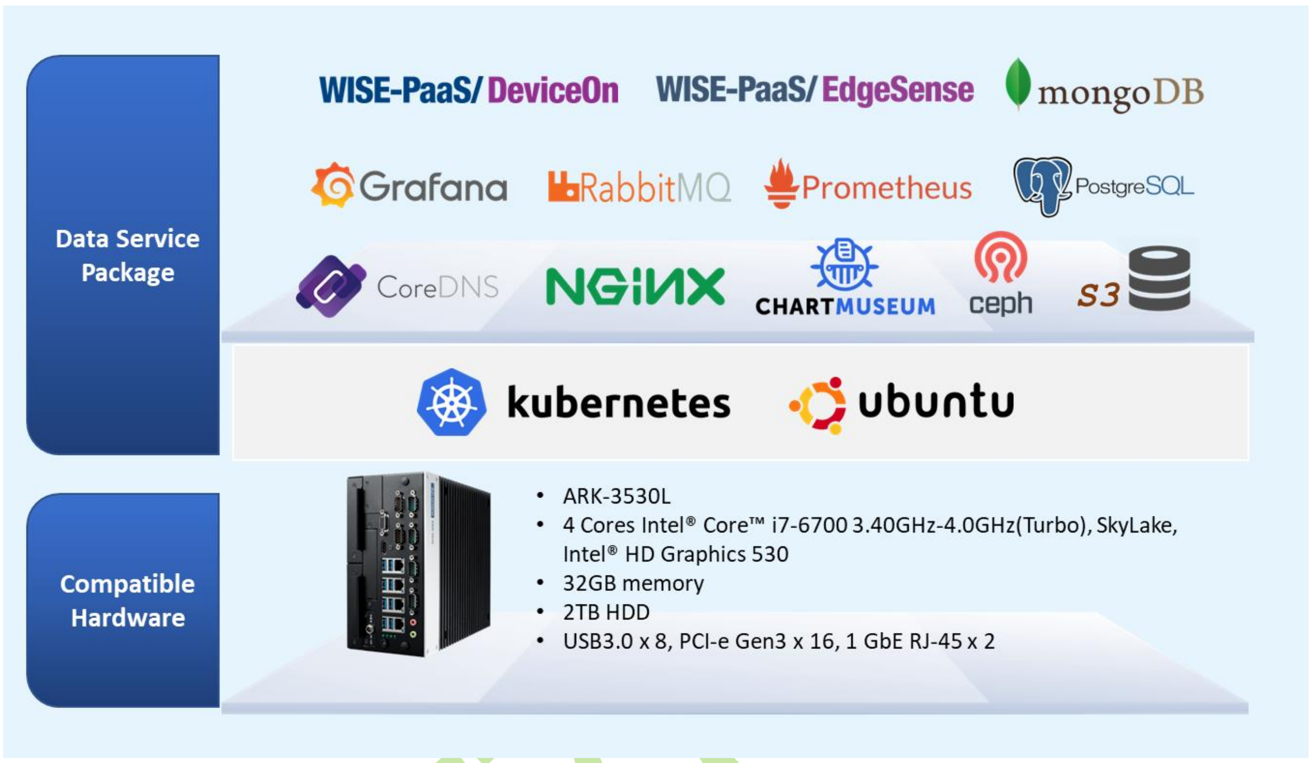


Figure 1 Architecture of Data Service Server

Data Service Server runs Kubernetes for automating deployment, scaling and management of containerized applications. [1] As illustrated in Figure 1, collecting data from an environment composed of so many moving parts is complex and Prometheus is the best monitoring and alerting tool for Kubernetes and Docker.

Prometheus was built specifically to monitor applications and microservices running in containers at scale and is native to containerized environments. Originally developed at Soundcloud, pioneer in the adoption of cloud technology. [2] In 2016, Prometheus project became the second hosted project of the Cloud Native Computing Foundation (CNCF) after Kubernetes.

1.2 Prometheus and Kubernetes

Prometheus is a pull-based monitoring system, which means that central Prometheus servers discover and pull metrics from your services. The discovery and pull system fits well with a dynamic, cloud native environment such as Kubernetes, where Prometheus integrates well with Kubernetes to discover and enumerate the services you have running. As you scale up a service, Prometheus automatically starts pulling metrics from the extra replicas. Similarly as nodes fail and pods are restarted, Prometheus automatically notices and scrapes them. [3] The architecture of Prometheus will be briefly introduced in next section.

2 Prometheus Overview

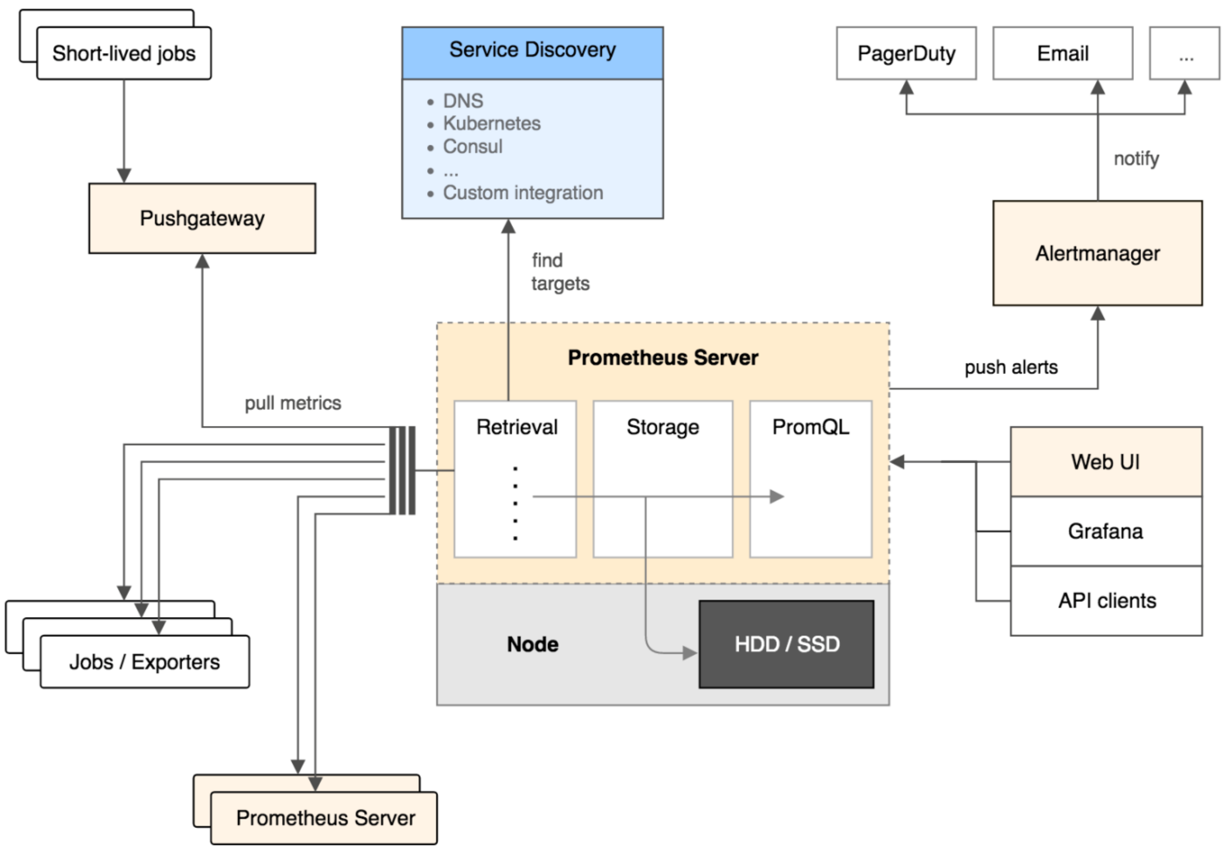


Figure 2 Architecture of Prometheus monitoring system [4]

As shown in Figure 2, Prometheus consists of several components working together. Prometheus server pulls in metrics at regular intervals through HTTP endpoints which provide information about hosts and various applications. These endpoints may be provided by the monitored application itself, or through an "exporter". The endpoint URL is usually /metrics. [5]

Since short-lived jobs not exist long enough to be scraped, they can instead push their metrics to a Pushgateway. Pushgateway then exposes these metrics to

Prometheus server. [6] Last but not least, Prometheus Server also exposes its own metrics and monitors its own metrics. [7]

After Prometheus server scrapes metrics from instrumented jobs, either directly or via an intermediary push gateway for short-lived jobs, it stores time-series data locally. User can use PromQL to query this data or send alerts to the Alertmanager, which will convert them into pages, emails, and other notifications. [8]

Prometheus provides a web interface (Web UI) to run queries. Other applications also can run queries through the HTTP API to retrieve and work with the data; we apply Grafana to visualize the data.

Next section will cover two parts: Grafana dashboards and Prometheus WebUI. It is recommended that users completely read this document before evaluation since you will understand which data visualization method meets your needs.

3 Data visualization

3.1 Grafana Dashboards

3.1.1 Introduction

Grafana is the leading graph and dashboard builder for visualizing time series infrastructure and application metrics. Therefore, we apply Grafana as data visualization tool.

3.1.2 Imported Dashboards

First of all, go to your Grafana main view and login admin account.

We take http://portal.grafana.example.com for example as Figure 3.

Please go to your Grafana domain server.

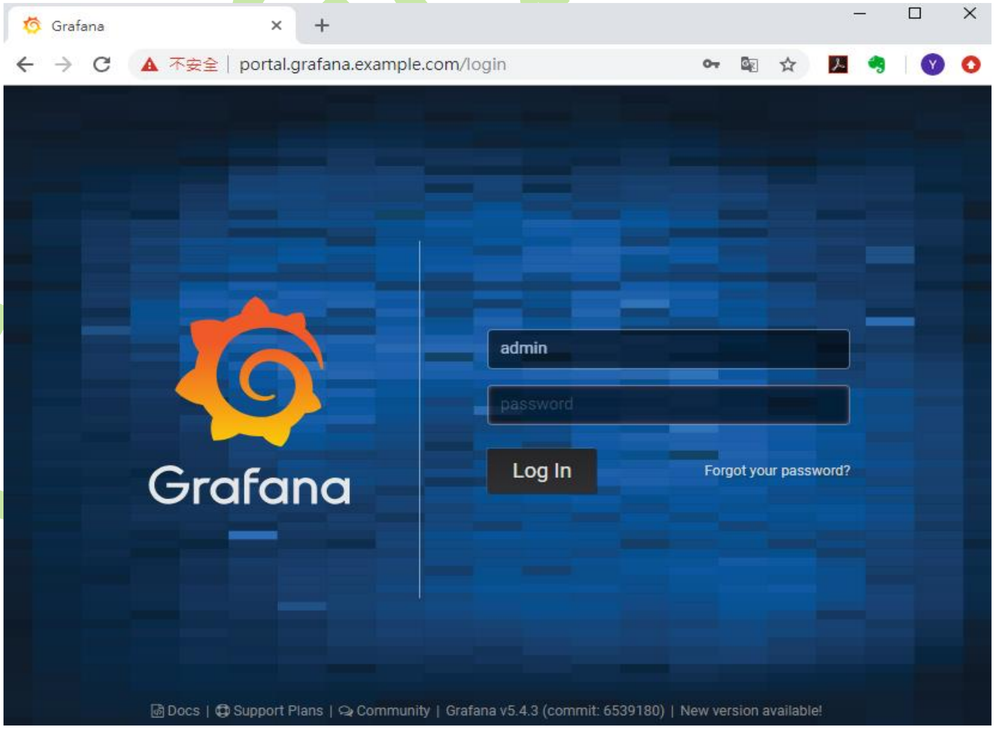

Grafana
不安全 | portal.grafana.example.com/login
Grafana
admin
password
Log In
Forgot your password?
Docs | Support Plans | Community | Grafana v5.4.3 (commit: 6539180) | New version available!

Figure 3 Grafana login page

Please access admin account as below.

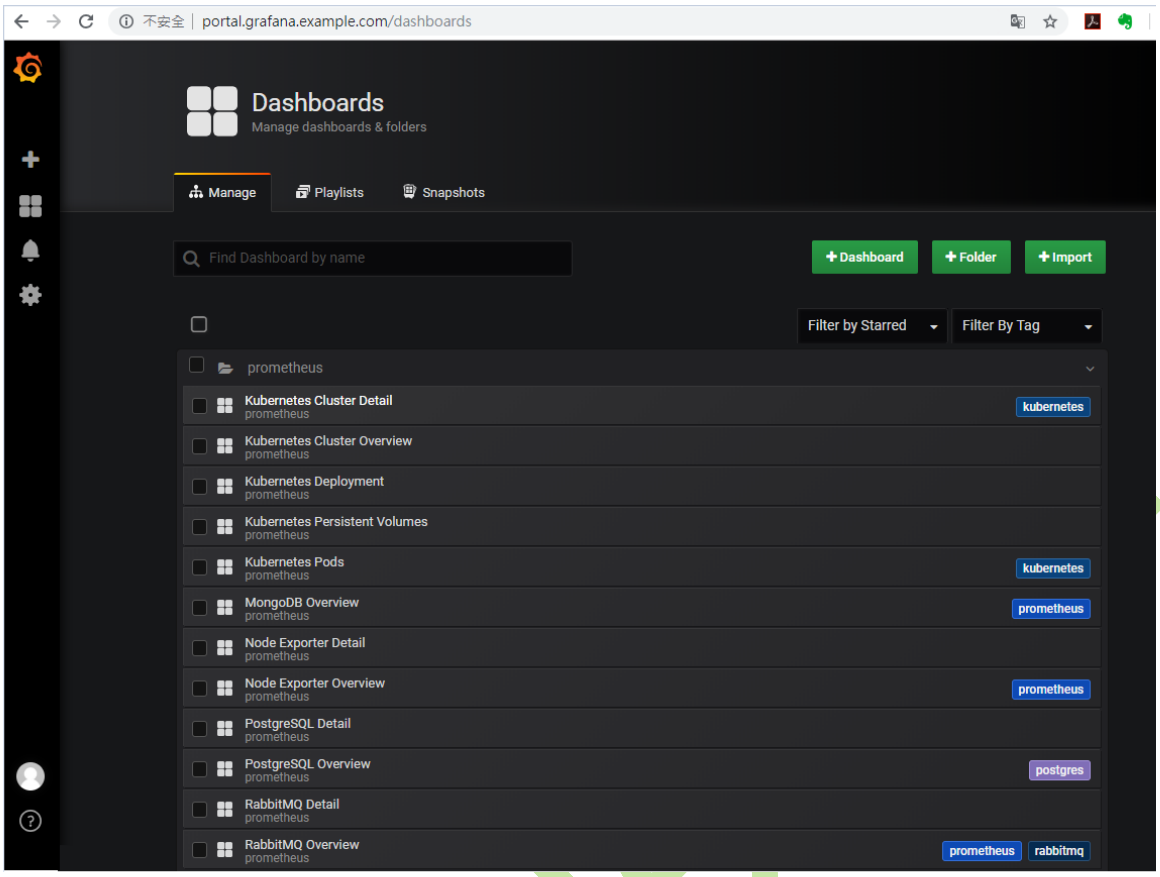
User Name: **admin**

Password: **@dvant1cH**

You can get username and password of admin account by issuing the following kubectl command.

```
✧  Get Username:
# kubectl get secret --namespace monitoring grafana -o
jsonpath="{.data.admin-user}" | base64 --decode ; echo

✧  Get Password:
# kubectl get secret --namespace monitoring grafana -o
jsonpath="{.data.admin-password}" | base64 --decode ; echo
```



Dashboards
Manage dashboards & folders

prometheus

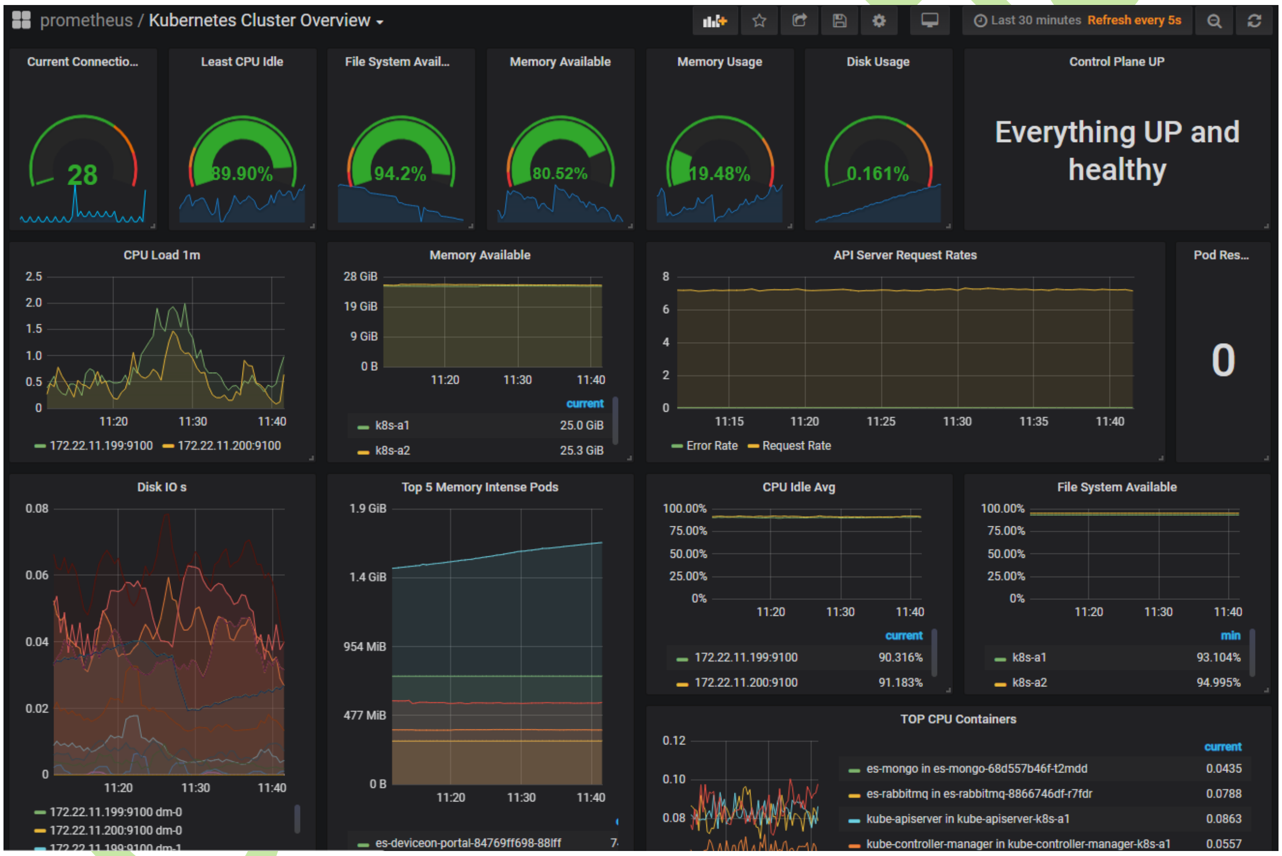
Kubernetes Cluster Detail	prometheus	kubernetes
Kubernetes Cluster Overview	prometheus	
Kubernetes Deployment	prometheus	
Kubernetes Persistent Volumes	prometheus	
Kubernetes Pods	prometheus	kubernetes
MongoDB Overview	prometheus	prometheus
Node Exporter Detail	prometheus	
Node Exporter Overview	prometheus	prometheus
PostgreSQL Detail	prometheus	
PostgreSQL Overview	prometheus	postgres
RabbitMQ Detail	prometheus	
RabbitMQ Overview	prometheus	prometheus rabbitmq

Figure 4 Dashboard overview

Generally speaking, there are several Kubernetes metrics worthy to monitor. As shown in Figure 4, adequate and detailed dashboards have been imported.

- Kubernetes Cluster Overview

For cluster monitoring, the objective is to monitor the health of the entire Kubernetes cluster. As an administrator, we are interested in discovering if all the nodes in the cluster are working properly and at what capacity, how many applications are running on each node, and the resource utilization of the entire cluster. [9]The overview of cluster is shown in Figure 5 below.



```
CPU Load 1m
172.22.11.199:9100   172.22.11.200:9100

Memory Available
                 current
k8s-a1           25.0 GiB
k8s-a2           25.3 GiB

API Server Request Rates
Error Rate   Request Rate

Disk IO s
172.22.11.199:9100 dm-0
172.22.11.200:9100 dm-0
172.22.11.199:9100 dm-1

Top 5 Memory Intense Pods
es-deviceon-portal-84769ff698-88lff

CPU Idle Avg
                      current
172.22.11.199:9100    90.316%
172.22.11.200:9100    91.183%

File System Available
           min
k8s-a1     93.104%
k8s-a2     94.995%

TOP CPU Containers
                                                                  current
es-mongo in es-mongo-68d557b46f-t2mdd                             0.0435
es-rabbitmq in es-rabbitmq-8866746df-r7fdr                        0.0788
kube-apiserver in kube-apiserver-k8s-a1                           0.0863
kube-controller-manager in kube-controller-manager-k8s-a1         0.0557
```

Figure 5 Kubernetes Cluster Overview

There are many measurable metrics worthy of mention, all related to node resource utilization, network bandwidth, disk utilization, CPU, and memory utilization are examples of this. You can navigate to dashboard "**Kubernetes Cluster Detail**" for more cluster information in details.

- Node Exporter Overview

One of the most widely used exporters is the NodeExporter. When NodeExporter runs on a host, it will collect data on I/O, memory, disk and CPU pressure and expose them for scraping. The overview of NodeExporter is as below Figure 6. [10] Also, we have another dashboard "**Node Exporter Detail**", it shows individual details of memory, network and other system metrics.

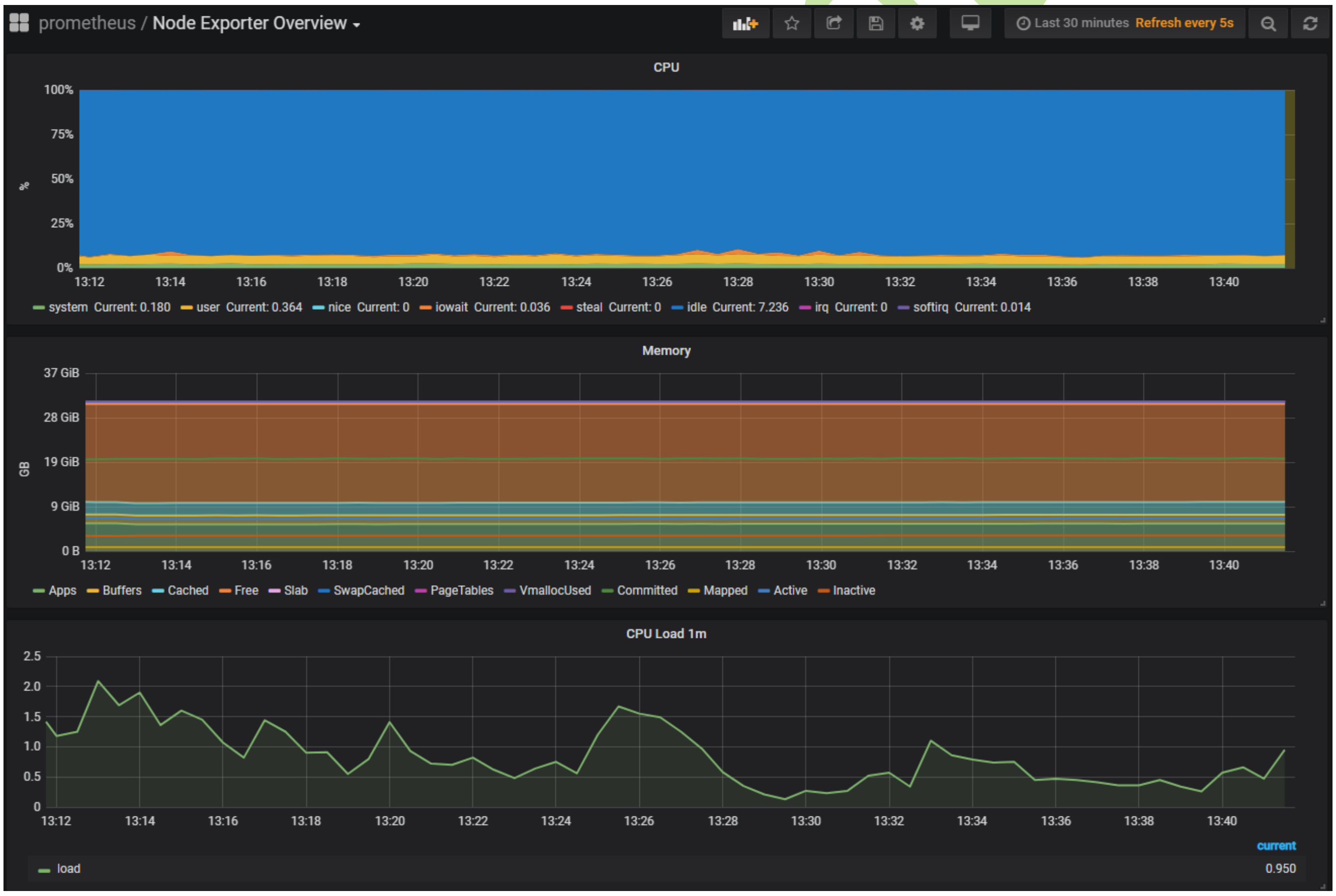
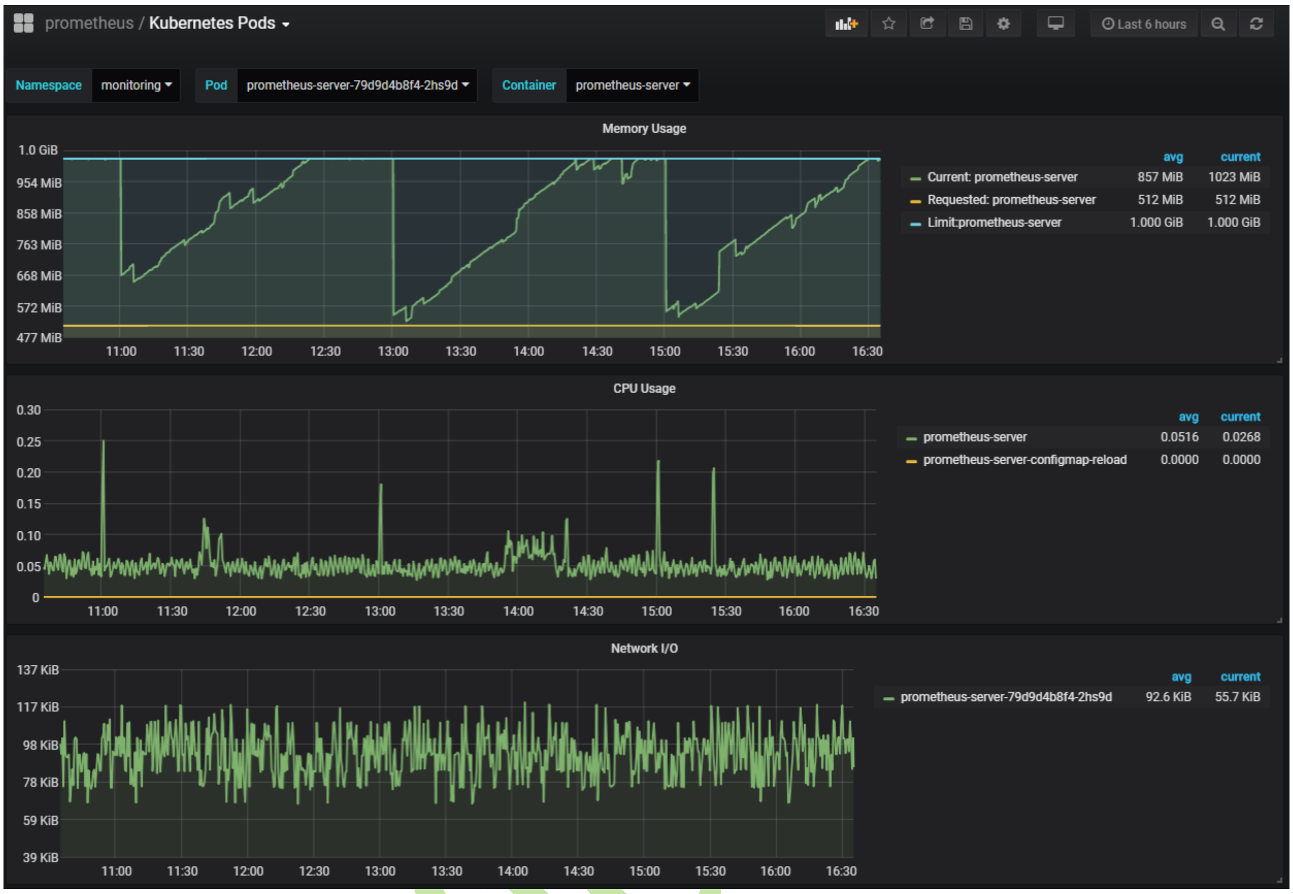


Figure 6 Node Exporter Overview

- Kubernetes Pods


prometheus / Kubernetes Pods

Namespace: monitoring | Pod: prometheus-server-79d9d4b8f4-2hs9d | Container: prometheus-server

Last 6 hours

Memory Usage

	avg	current
Current: prometheus-server	857 MiB	1023 MiB
Requested: prometheus-server	512 MiB	512 MiB
Limit:prometheus-server	1.000 GiB	1.000 GiB

CPU Usage

	avg	current
prometheus-server	0.0516	0.0268
prometheus-server-configmap-reload	0.0000	0.0000

Network I/O

	avg	current
prometheus-server-79d9d4b8f4-2hs9d	92.6 KiB	55.7 KiB

Figure 7 Kubernetes Pods

The act of monitoring a pod can also extend to its containers. You can monitor individual container if there is more than one container in a pod. Container metrics query running container information like CPU, network, and memory usage. As Figure 7 shows above, maximum limitation is set to compare with memory usage.

- Kubernetes Persistent Volumes

A persistent volume (PV) is a piece of storage in the cluster. PVs are volume plugins like Volumes, but have a lifecycle independent of any individual pod that uses the PV.

A persistent volume claim (PVC) is similar to a pod. Pods consume node resources and PVCs consume PV resources. Claims can request specific size and access modes. [11]

Data Service Server implements RBD (Ceph Block Device) as plugins. The disk and inode usage of individual PVCs are shown in Figure 8.

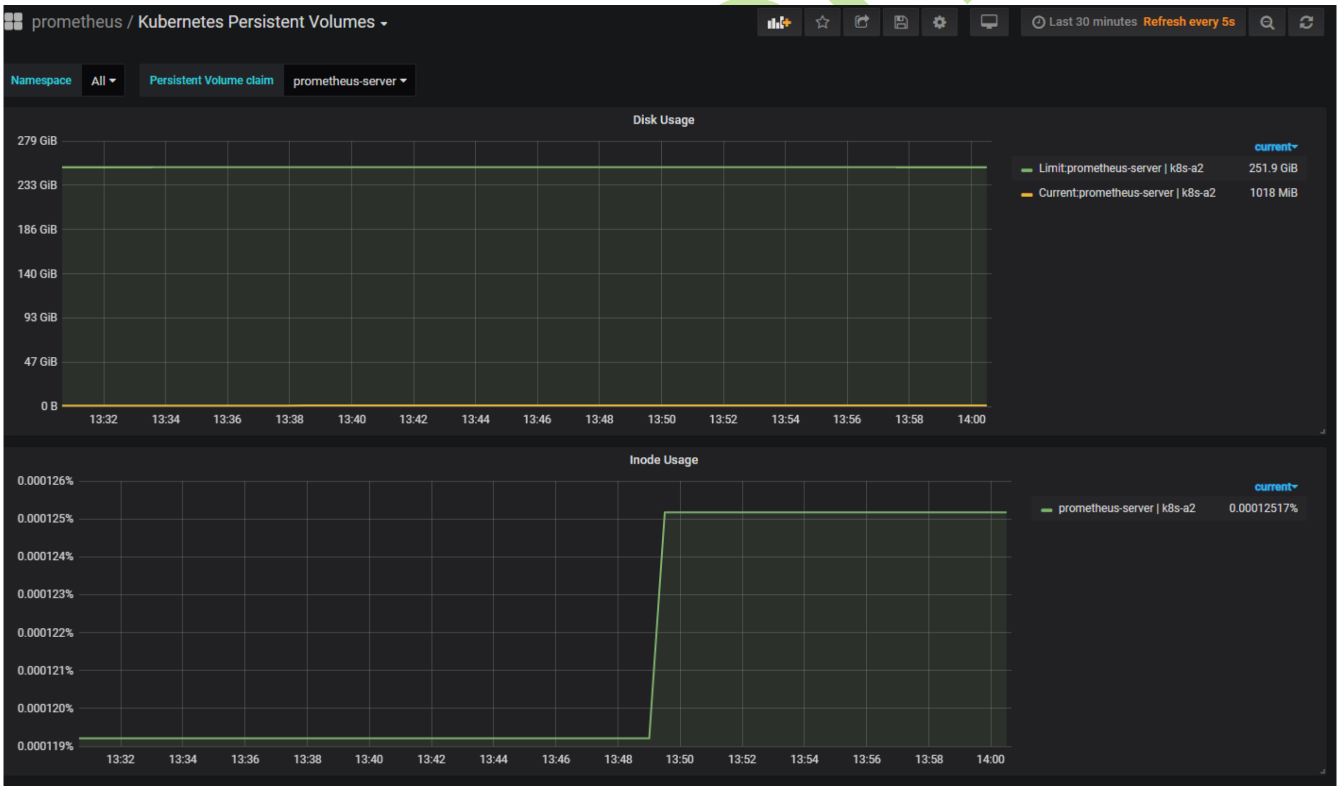


Figure 8 Kubernetes Persistent Volumes

- Other Kubernetes dashboards

"**Kubernetes Deployment**" : Kubernetes deployment is an abstraction layer for the pods. The main purpose of the deployment is to maintain the resources declared in the deployment configuration to be its desired state. [11] User can monitor the status of every deployment in every namespace.

"**Kubernetes StatefulSet**" : Manages the deployment and scaling of a set of Pods, and provides guarantees about the ordering and uniqueness of these Pods. [12] The status overview of StatefulSet will show in this dashboard.

- Third-Party Exporter

Some other services are not natively integrated, but can be easily adapted using an exporter. An exporter is a service that collects service status and translates to Prometheus metrics ready to be scraped. [13] We have three exporters in Data Service Server, they are RabbitMQ, PostgreSQL and MongoDB respectively.

➢ RabbitMQ Overview

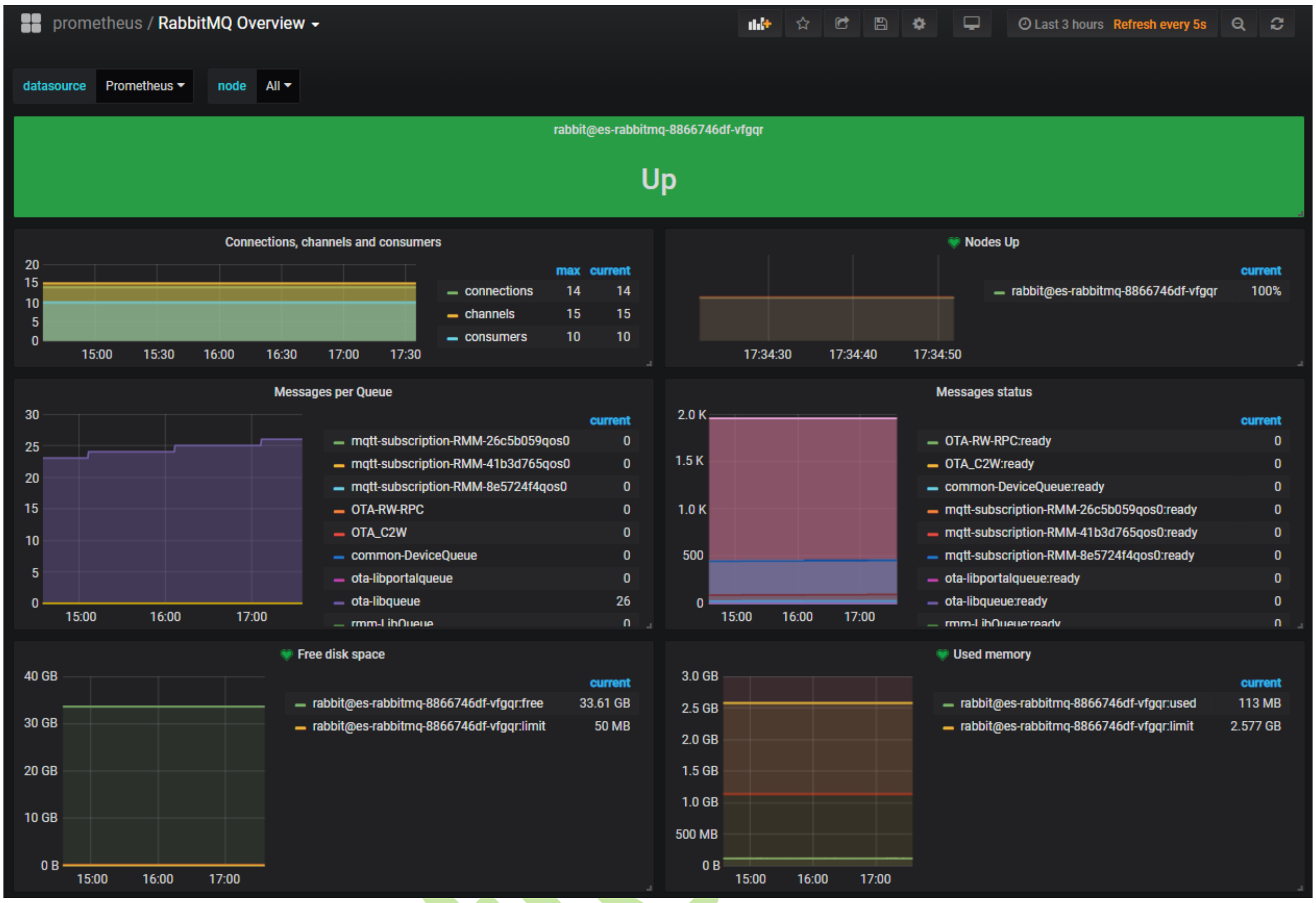
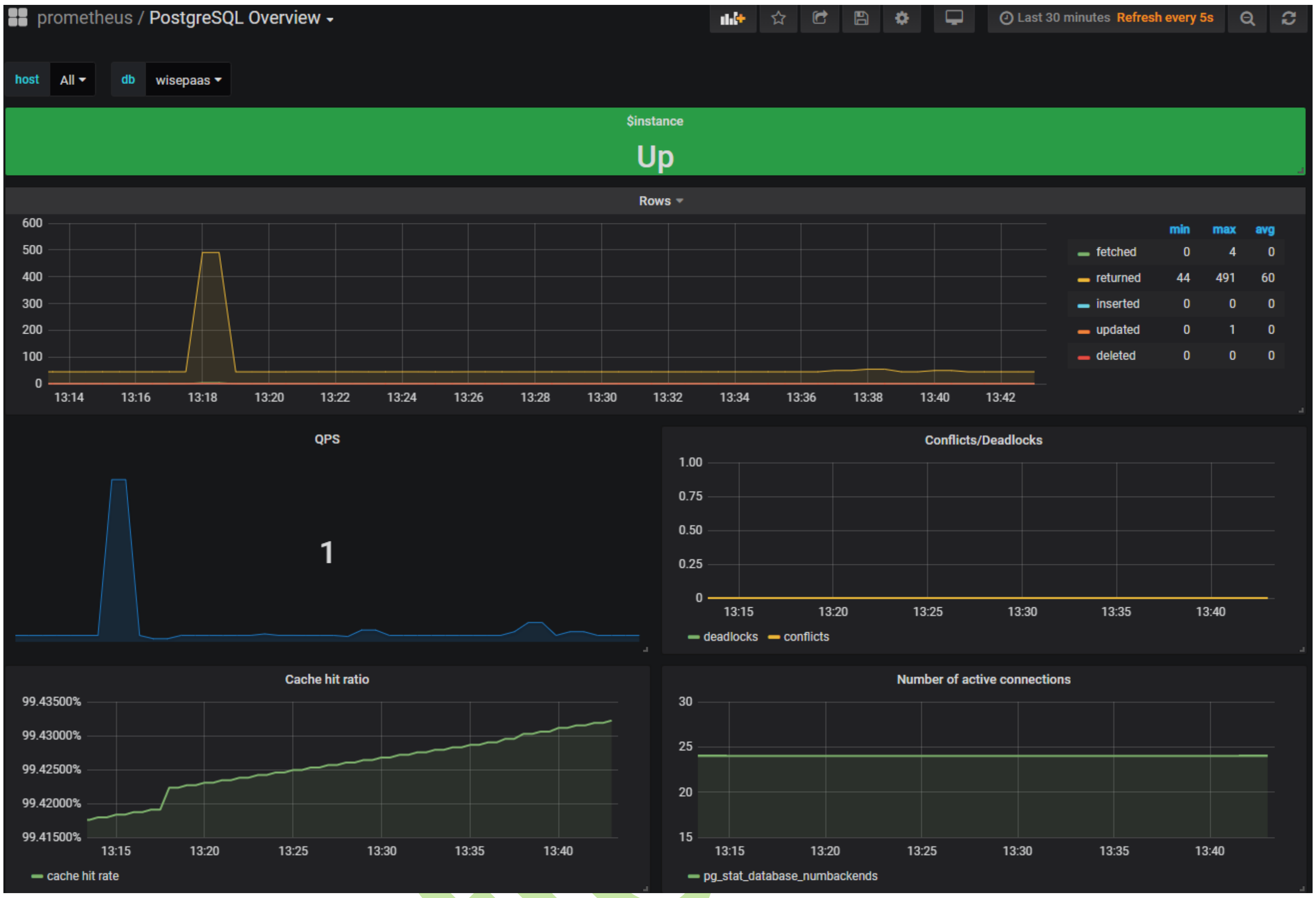

prometheus / RabbitMQ Overview
Last 3 hours Refresh every 5s
datasource Prometheus
node All
rabbit@es-rabbitmq-8866746df-vfgqr
Up
Connections, channels and consumers
max current
connections 14 14
channels 15 15
consumers 10 10
Nodes Up
current
rabbit@es-rabbitmq-8866746df-vfgqr 100%
Messages per Queue
current
mqtt-subscription-RMM-26c5b059qos0 0
mqtt-subscription-RMM-41b3d765qos0 0
mqtt-subscription-RMM-8e5724f4qos0 0
OTA-RW-RPC 0
OTA_C2W 0
common-DeviceQueue 0
ota-libportalqueue 0
ota-libqueue 26
Messages status
current
OTA-RW-RPC:ready 0
OTA_C2W:ready 0
common-DeviceQueue:ready 0
mqtt-subscription-RMM-26c5b059qos0:ready 0
mqtt-subscription-RMM-41b3d765qos0:ready 0
mqtt-subscription-RMM-8e5724f4qos0:ready 0
ota-libportalqueue:ready 0
ota-libqueue:ready 0
Free disk space
current
rabbit@es-rabbitmq-8866746df-vfgqr:free 33.61 GB
rabbit@es-rabbitmq-8866746df-vfgqr:limit 50 MB
Used memory
current
rabbit@es-rabbitmq-8866746df-vfgqr:used 113 MB
rabbit@es-rabbitmq-8866746df-vfgqr:limit 2.577 GB

Figure 9 RabbitMQ Overview

RabbitMQ is the most widely deployed open source message broker, it supports multiple messaging protocols, message queuing, delivery acknowledgment, flexible routing to queues, multiple exchange type. [14] RabbitMQ exporter is applied to provide a starting point for monitoring RabbitMQ metrics as shown in Figure 8. Prometheus is configured 60 seconds interval to collect RabbitMQ status.

➢ PostgresSQL Overview

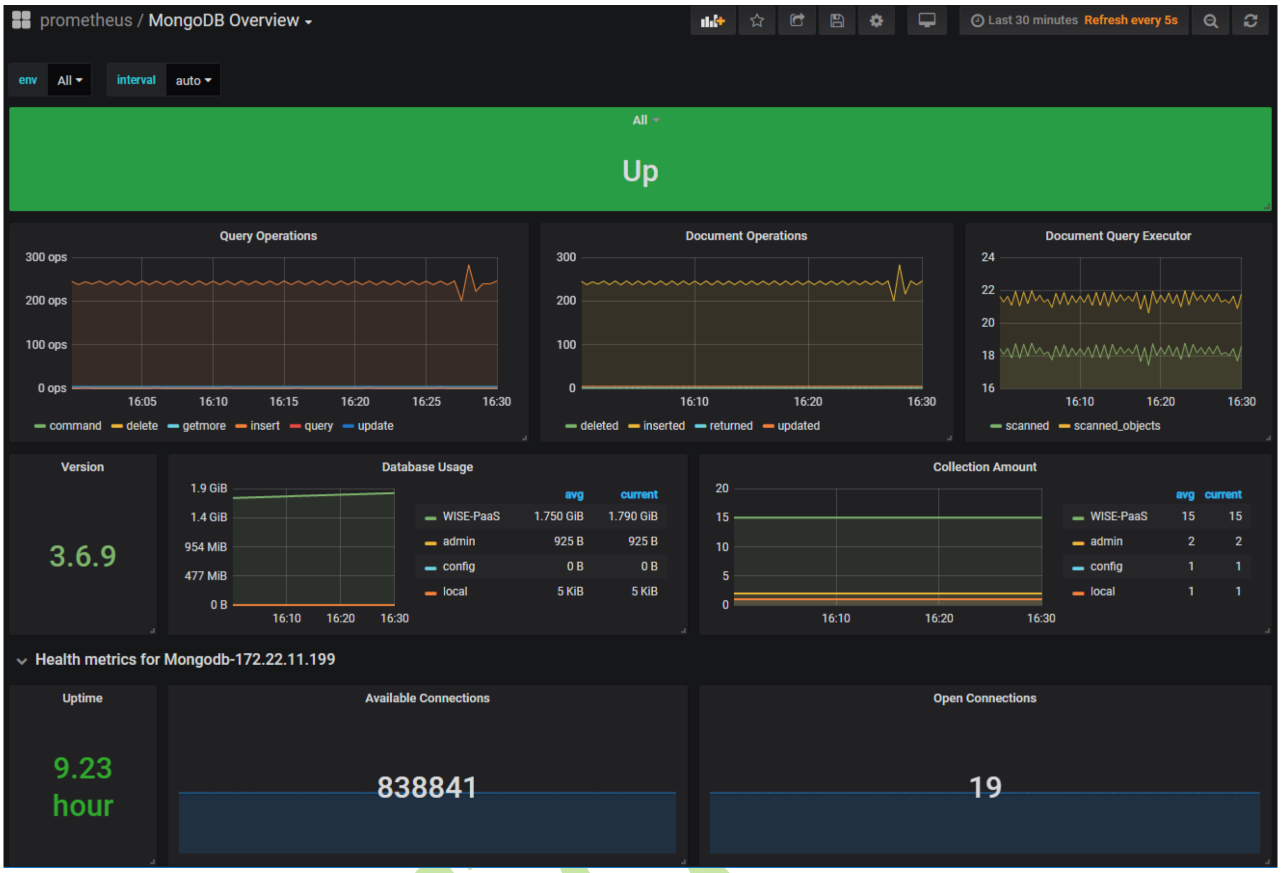

prometheus / PostgreSQL Overview
Last 30 minutes Refresh every 5s
host All
db wisepaas
$instance
Up
Rows
QPS
1
Conflicts/Deadlocks
deadlocks
conflicts
Cache hit ratio
cache hit rate
Number of active connections
pg_stat_database_numbackends

	min	max	avg
fetched	0	4	0
returned	44	491	60
inserted	0	0	0
updated	0	1	0
deleted	0	0	0

Figure 10 PostgresSQL Overview

PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. [15] [16] The user account and connected device info of Data Service Server are stored in PostgresSQL, as shown in Figure 9, you can monitor real-time status of PostgresSQL.

➢ MongoDB Overview



Database Usage

	avg	current
WISE-PaaS	1.750 GiB	1.790 GiB
admin	925 B	925 B
config	0 B	0 B
local	5 KiB	5 KiB

Collection Amount

	avg	current
WISE-PaaS	15	15
admin	2	2
config	1	1
local	1	1

Figure 11 MongoDB Overview

MongoDB is the most popular open-source NoSQL(Not Only SQL) database, it is a document database designed for ease of development and scaling. MongoDB provides high performance data persistence and horizontal scalability. [17]

All the reported data from connected devices in Data Service Server is stored in MongoDB. User can check if data network works normally from MongoDB Overview as shown in Figure 10.

3.2 Prometheus Web UI

3.2.1 Introduction

Prometheus provides a functional query language called PromQL (Prometheus Query Language) that lets user select and aggregate time series data in real time. The result of an expression can either be shown as a graph, viewed as tabular data in Prometheus's expression browser. It will be a good choice if you need to query specific metrics using PromQL.

3.2.2 PromQL

At first, go to your Prometheus Web UI with no username and password.

We take http://portal.prometheus.example.com for example as Figure 10.

Please go to your Prometheus WebUI domain server.

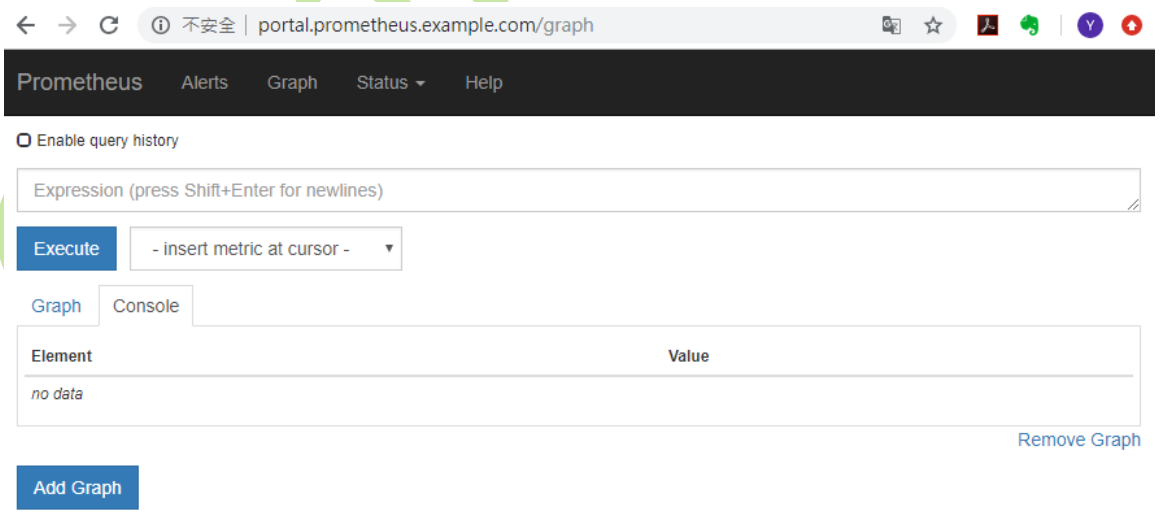
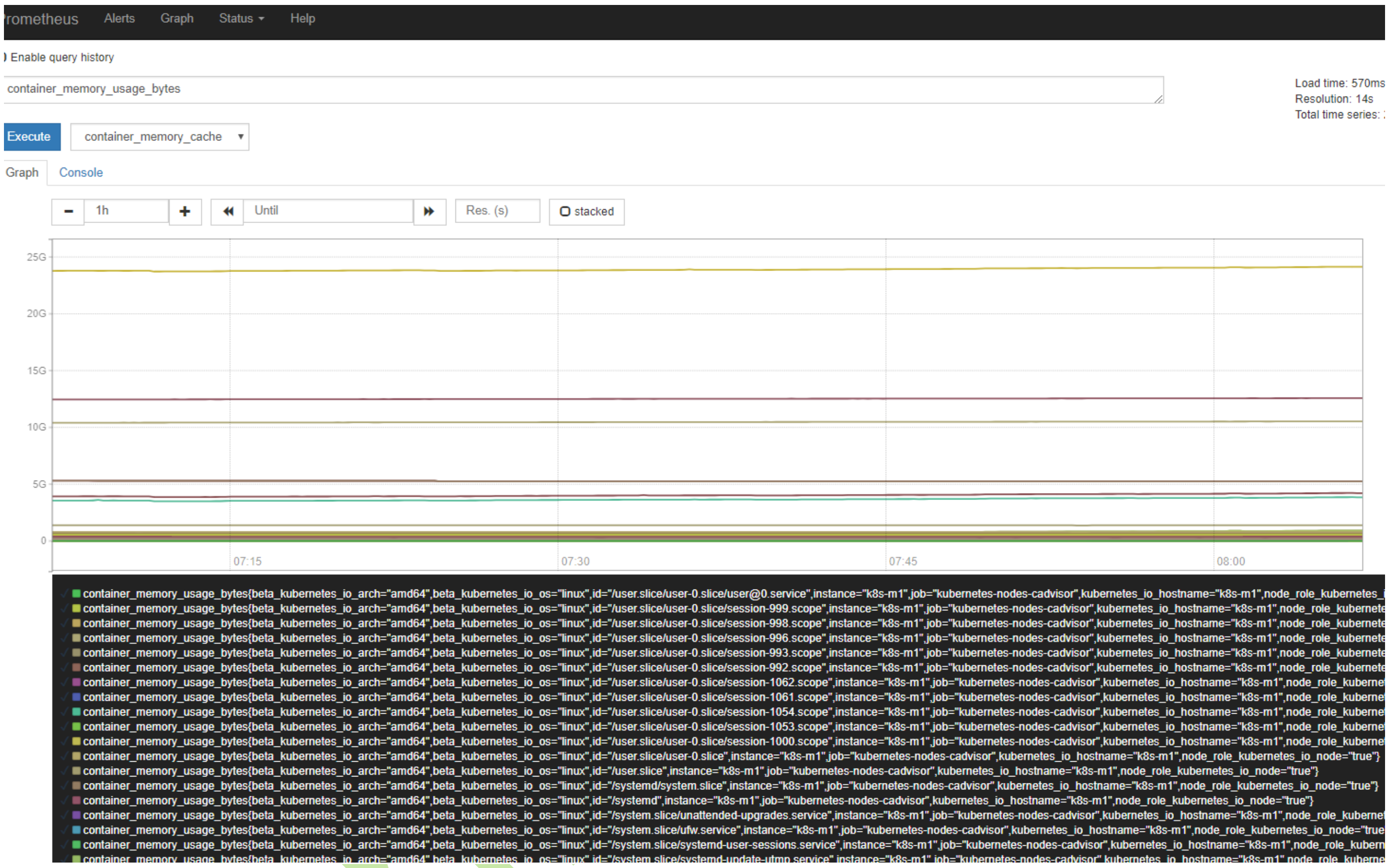

不安全 | portal.prometheus.example.com/graph
Prometheus
Alerts
Graph
Status
Help
Enable query history
Expression (press Shift+Enter for newlines)
Execute
- insert metric at cursor -
Graph
Console
Element
Value
no data
Remove Graph
Add Graph

Figure 12 Prometheus Web UI

Take container_memory_usage_bytes for quick example, it selects all time-series that have the container_memory_usage_bytes metric name. Figure 11 below shows the query results with graph mode.

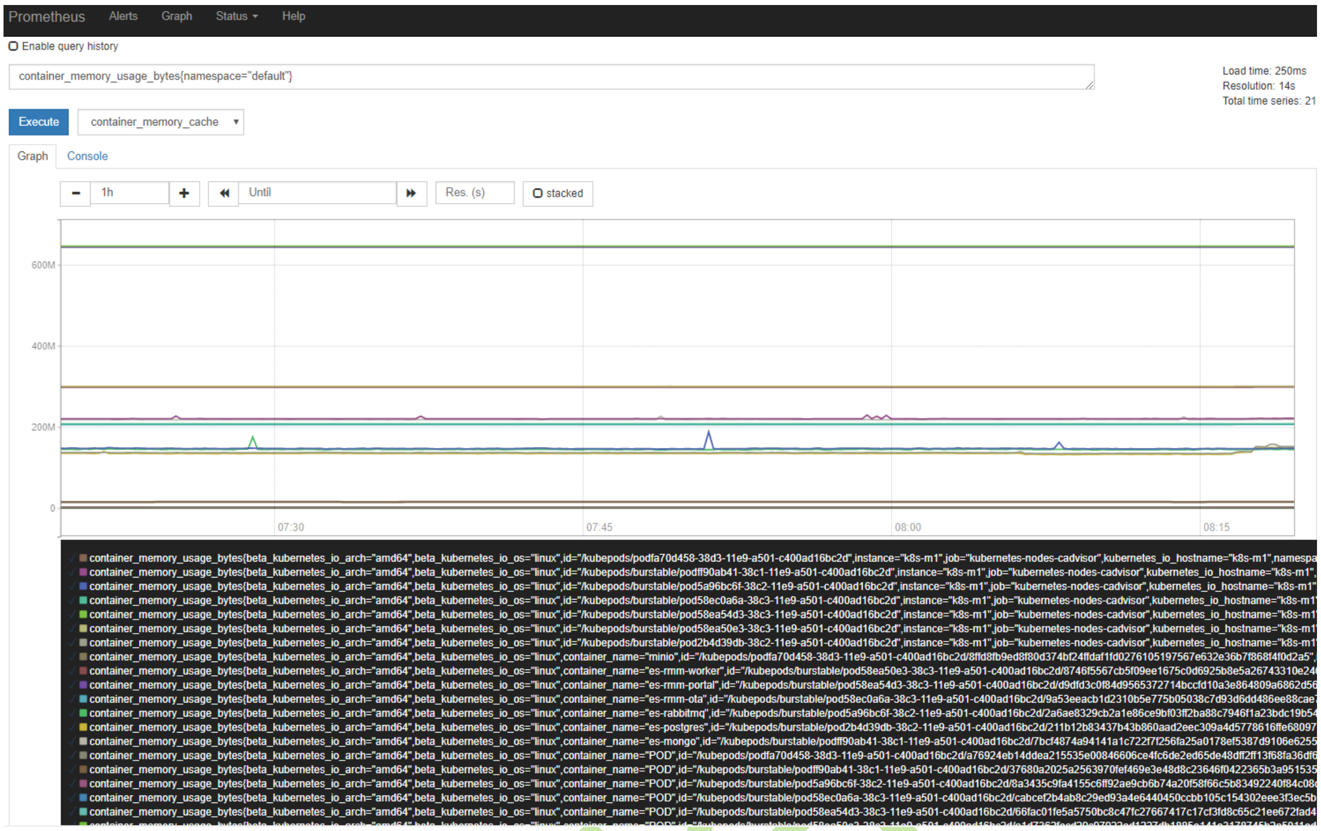


```
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/user@0.service",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernetes_i
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-999.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernete
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-998.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernete
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-996.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernete
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-993.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernete
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-992.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernete
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-1062.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kuberne
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-1061.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kuberne
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-1054.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kuberne
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-1053.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kuberne
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice/session-1000.scope",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kuberne
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice/user-0.slice",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernetes_io_node="true"}
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/user.slice",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernetes_io_node="true"}
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/systemd/system.slice",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernetes_io_node="true"}
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/systemd",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernetes_io_node="true"}
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/system.slice/unattended-upgrades.service",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kuberne
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/system.slice/ufw.service",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubernetes_io_node="true
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/system.slice/systemd-user-sessions.service",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kubern
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/system.slice/systemd-update-utmp.service",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",node_role_kuberne
```

Figure 13 Simple query result

The above results contain all the container_memory_usage_bytes metrics, the data is large. If you want to query more specific data in advanced, append a set of labels to match in curly braces ({}). Take container_memory_usage_bytes for example, append one term in PromQL as below.

```
✧ container_memory_usage_bytes{namespace="default"}
```



```
container_memory_usage_bytes{namespace="default"}
```

Load time: 250ms
Resolution: 14s
Total time series: 21

container_memory_cache

```
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/kubepods/podfa70d458-38d3-11e9-a501-c400ad16bc2d",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",namespa
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/kubepods/burstable/podff90ab41-38c1-11e9-a501-c400ad16bc2d",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1",
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/kubepods/burstable/pod5a96bc6f-38c2-11e9-a501-c400ad16bc2d",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1"
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/kubepods/burstable/pod58ec0a6a-38c3-11e9-a501-c400ad16bc2d",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/kubepods/burstable/pod58ea54d3-38c3-11e9-a501-c400ad16bc2d",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/kubepods/burstable/pod58ea50e3-38c3-11e9-a501-c400ad16bc2d",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",id="/kubepods/burstable/pod2b4d39db-38c2-11e9-a501-c400ad16bc2d",instance="k8s-m1",job="kubernetes-nodes-cadvisor",kubernetes_io_hostname="k8s-m1
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="minio",id="/kubepods/podfa70d458-38d3-11e9-a501-c400ad16bc2d/8ffd8fb9ed8f80d374bf24ffdaf1fd0276105197567e632e36b7f868f4f0d2a5",
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="es-rmm-worker",id="/kubepods/burstable/pod58ea50e3-38c3-11e9-a501-c400ad16bc2d/8746f5567cb5f09ee1675c0d6925b8e5a26743310e24
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="es-rmm-portal",id="/kubepods/burstable/pod58ea54d3-38c3-11e9-a501-c400ad16bc2d/d9dfd3c0f84d9565372714bccfd10a3e864809a6862d56
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="es-rmm-ota",id="/kubepods/burstable/pod58ec0a6a-38c3-11e9-a501-c400ad16bc2d/9a53eeacb1d2310b5e775b05038c7d93d6dd486ee88cae
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="es-rabbitmq",id="/kubepods/burstable/pod5a96bc6f-38c2-11e9-a501-c400ad16bc2d/2a6ae8329cb2a1e86ce9bf03ff2ba88c7946f1a23bdc19b54
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="es-postgres",id="/kubepods/burstable/pod2b4d39db-38c2-11e9-a501-c400ad16bc2d/211b12b83437b43b860aad2eec309a4d5778616ffe68097
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="es-mongo",id="/kubepods/burstable/podff90ab41-38c1-11e9-a501-c400ad16bc2d/7bcf4874a94141a1c722f7f256fa25a0178ef5387d9106e6255
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="POD",id="/kubepods/podfa70d458-38d3-11e9-a501-c400ad16bc2d/a76924eb14ddea215535e00846606ce4fc6de2ed65de48dff2ff13f68fa36df6
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="POD",id="/kubepods/burstable/podff90ab41-38c1-11e9-a501-c400ad16bc2d/37680a2025a2563970fef469e3e48d8c23646f0422365b3a951535
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="POD",id="/kubepods/burstable/pod5a96bc6f-38c2-11e9-a501-c400ad16bc2d/8a3435c9fa4155c6ff92ae9cb6b74a20f58f66c5b83492240f84c08
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="POD",id="/kubepods/burstable/pod58ec0a6a-38c3-11e9-a501-c400ad16bc2d/cabcef2b4ab8c29ed93a4e6440450ccbb105c154302eee3f3ec5b
container_memory_usage_bytes{beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",container_name="POD",id="/kubepods/burstable/pod58ea54d3-38c3-11e9-a501-c400ad16bc2d/66fac01fe5a5750bc8c47fc27667417c17cf3fd8c65c21ee672fad4
```

Figure 14 Advanced query result

The result shows in Figure 12 that container_memory_usage_bytes metrics with namespace “default”. Therefore, user can query any specific metric if in need. Please refer to the document for more detail of PromQL.

https://prometheus.io/docs/prometheus/latest/querying/basics/

4 Alert Manager

The Alertmanager handles alerts sent by client applications such as the Prometheus server. When an alert reaches its threshold, it is forwarded to Alertmanager that acts as a crossroad. Depending on its internal rules, it can forward those alerts further to various destinations like Slack, email, and PagerDuty (only to name a few). [19]

4.1 Rules

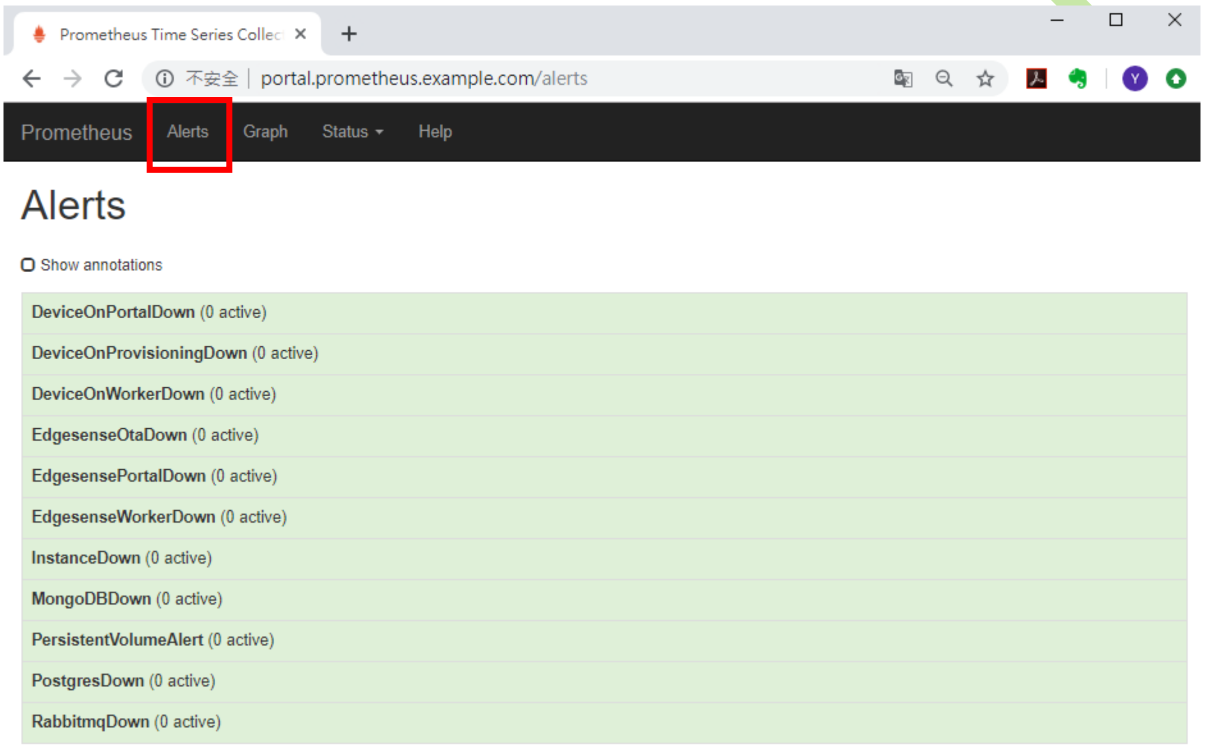
We already define all the necessary alerting rules in Prometheus. Alerting rules define alert conditions based on Prometheus expression language expressions and to send notifications about firing alerts to an external service.

1. Please navigate to Prometheus WebUI as described in Chapter 3.2.

We take http://portal.prometheus.example.com for example.

Please go to your Prometheus WebUI domain server.

2. Navigate to "Alerts" label; you will see all the alerting rules we defined. You might check the expressions by clicking on each alert name. We already make sure that if there's any instance works abnormally, alertmanager will send notification to user at the first time.


Prometheus Time Series Collec
portal.prometheus.example.com/alerts
Prometheus
Alerts
Graph
Status
Help

Alerts

Show annotations

DeviceOnPortalDown (0 active)
DeviceOnProvisioningDown (0 active)
DeviceOnWorkerDown (0 active)
EdgesenseOtaDown (0 active)
EdgesensePortalDown (0 active)
EdgesenseWorkerDown (0 active)
InstanceDown (0 active)
MongoDBDown (0 active)
PersistentVolumeAlert (0 active)
PostgresDown (0 active)
RabbitmqDown (0 active)

Figure 15 Alert rules.

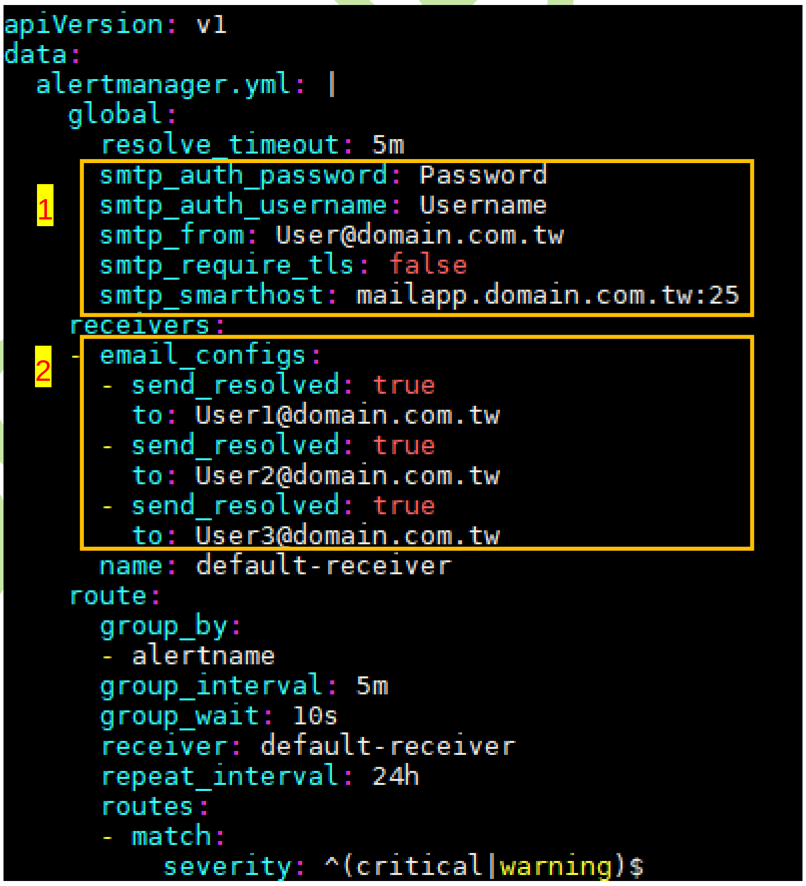
4.2 Configuration

Alertmanager sends notifications to receivers which defined in configuration file. To receive alerting notifications, user needs to modify smtp server and receiver emails in configuration by issuing "kubectl" commands. Alertmanager can reload its configuration at runtime. We set alerting interval to1 day, in other words; the same alerting only send notification once in one day to avoid frequently mailing.

1. Issue command to edit configmap of alertmanager in namespace "monitoring".

```
# kubectl edit -n=monitoring configmaps prometheus-alertmanager
```

2. Configmap is shown as Fig 16. Please modify the **first** part to your smtp server host and login information. **Smtp_from** means the alerting notification sender. **Smtp_require_tls** is default to false; you don't have to change it. **Second** part is receiver's email; please add receivers in this part. If you have more receivers, please feel free to append it.



```
apiVersion: v1
data:
  alertmanager.yml: |
    global:
      resolve_timeout: 5m
      smtp_auth_password: Password
      smtp_auth_username: Username
      smtp_from: User@domain.com.tw
      smtp_require_tls: false
      smtp_smarthost: mailapp.domain.com.tw:25
    receivers:
    - email_configs:
      - send_resolved: true
        to: User1@domain.com.tw
      - send_resolved: true
        to: User2@domain.com.tw
      - send_resolved: true
        to: User3@domain.com.tw
      name: default-receiver
    route:
      group_by:
      - alertname
      group_interval: 5m
      group_wait: 10s
      receiver: default-receiver
      repeat_interval: 24h
      routes:
      - match:
          severity: ^(critical|warning)$
```

Figure 16 Alertmanager configmap

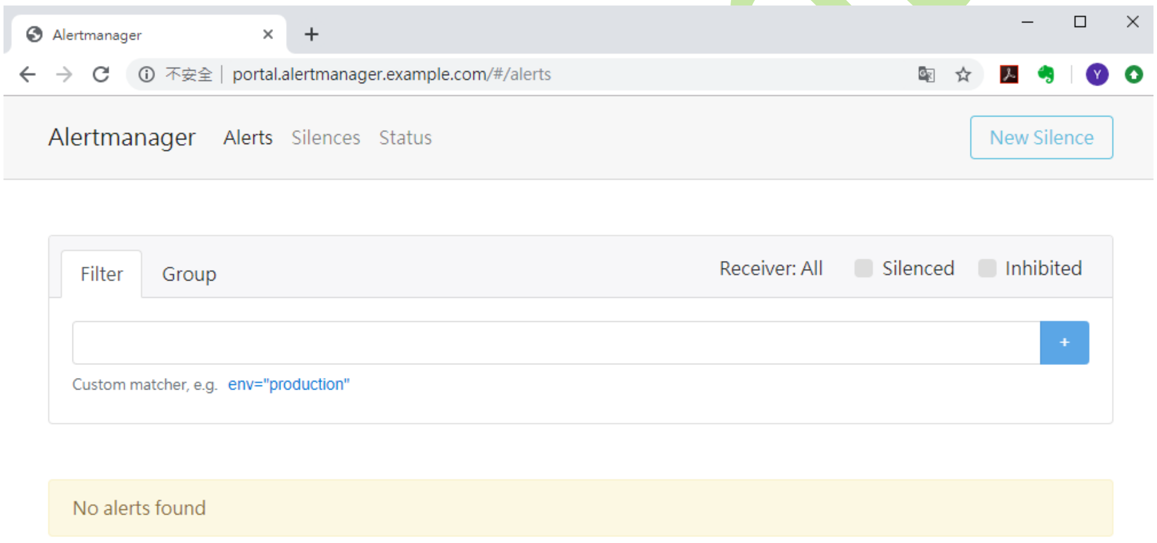
After saving all the modification, alertmanager will reload the configurations immediately. We can check the latest status on Alertmanager WebUI.

4.3 Alertmanager Web UI

1. Please go to your Alertmanager Web UI with no username and password.

We take **http://portal.prometheus.example.com** for example as Figure 17.

Please go to your Alertmanager WebUI domain server.

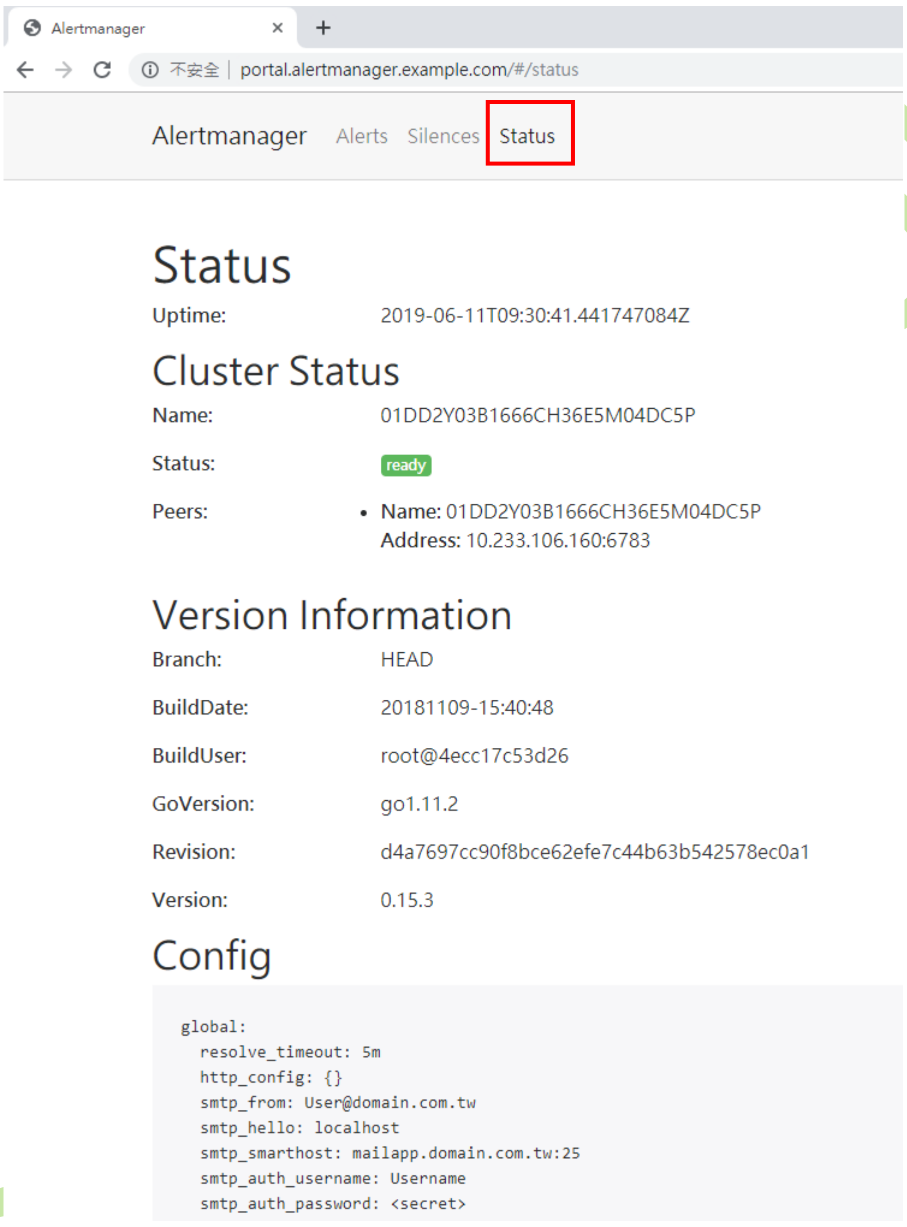


Custom matcher, e.g. env="production"

No alerts found

Figure 17 Alertmanager WebUI

2. Navigate to "Status" label; you will see the configuration you just saved. If smtp server works fine, receivers will receive alerting email when alert occurs. Please make sure that smtp server works fine.



Status

Uptime: 2019-06-11T09:30:41.441747084Z

Cluster Status

Name: 01DD2Y03B1666CH36E5M04DC5P

Status: ready

Peers:
- Name: 01DD2Y03B1666CH36E5M04DC5P
  Address: 10.233.106.160:6783

Version Information

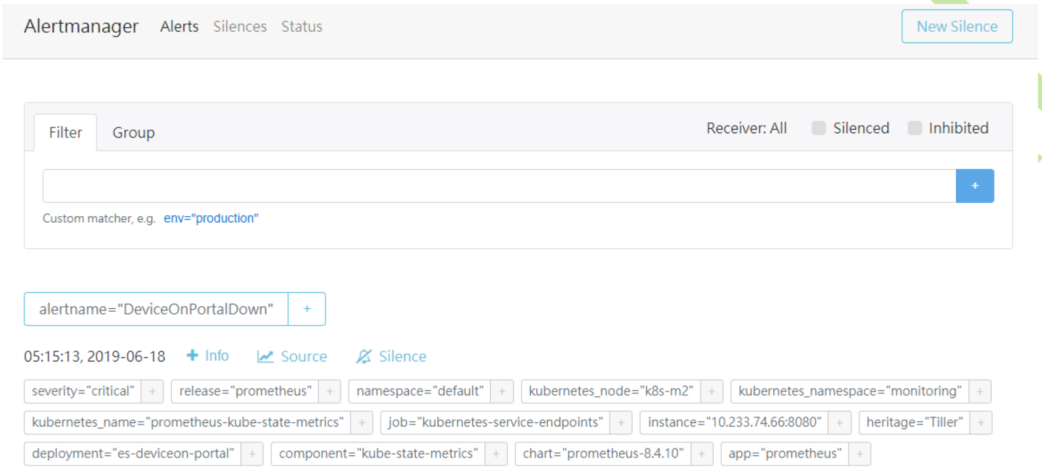
Branch:	HEAD
BuildDate:	20181109-15:40:48
BuildUser:	root@4ecc17c53d26
GoVersion:	go1.11.2
Revision:	d4a7697cc90f8bce62efe7c44b63b542578ec0a1
Version:	0.15.3

Config

```
global:
  resolve_timeout: 5m
  http_config: {}
  smtp_from: User@domain.com.tw
  smtp_hello: localhost
  smtp_smarthost: mailapp.domain.com.tw:25
  smtp_auth_username: Username
  smtp_auth_password: <secret>
```

4.3.1 Alerts

Let’ s forward to “Alerts” label, if there are alerts sent from Alertmanager, the alerts will be listed in this page as Figure 18. Here we take the alert called “DeviceOnPortalDown” for example.



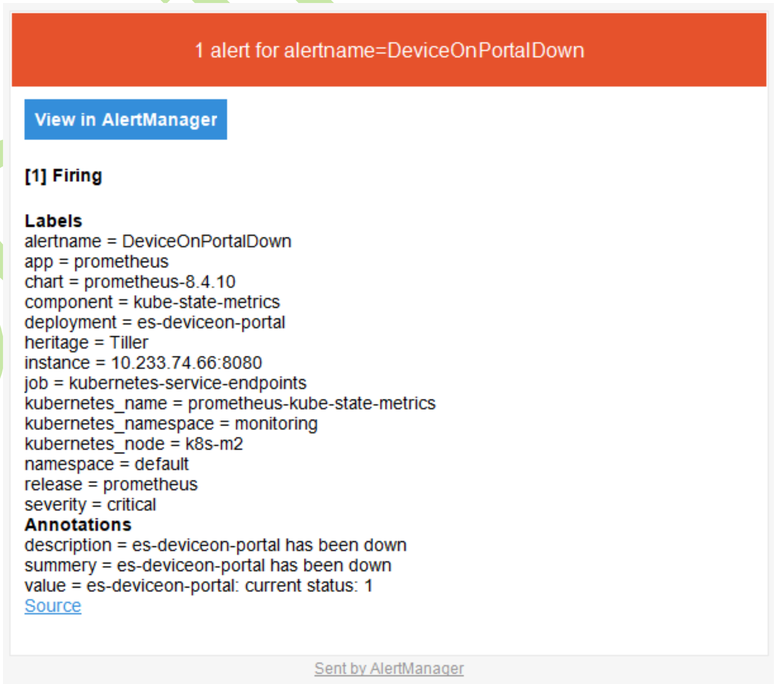
alertname="DeviceOnPortalDown"

05:15:13, 2019-06-18

severity="critical" release="prometheus" namespace="default" kubernetes_node="k8s-m2" kubernetes_namespace="monitoring" kubernetes_name="prometheus-kube-state-metrics" job="kubernetes-service-endpoints" instance="10.233.74.66:8080" heritage="Tiller" deployment="es-deviceon-portal" component="kube-state-metrics" chart="prometheus-8.4.10" app="prometheus"

Figure 18 Alerts page in Alertmanager Web UI

In the same time, receiver will receive the alerting mail of “DeviceOnPortalDown” .



1 alert for alertname=DeviceOnPortalDown

[1] Firing

Labels
alertname = DeviceOnPortalDown
app = prometheus
chart = prometheus-8.4.10
component = kube-state-metrics
deployment = es-deviceon-portal
heritage = Tiller
instance = 10.233.74.66:8080
job = kubernetes-service-endpoints
kubernetes_name = prometheus-kube-state-metrics
kubernetes_namespace = monitoring
kubernetes_node = k8s-m2
namespace = default
release = prometheus
severity = critical
Annotations
description = es-deviceon-portal has been down
summery = es-deviceon-portal has been down
value = es-deviceon-portal: current status: 1
Source

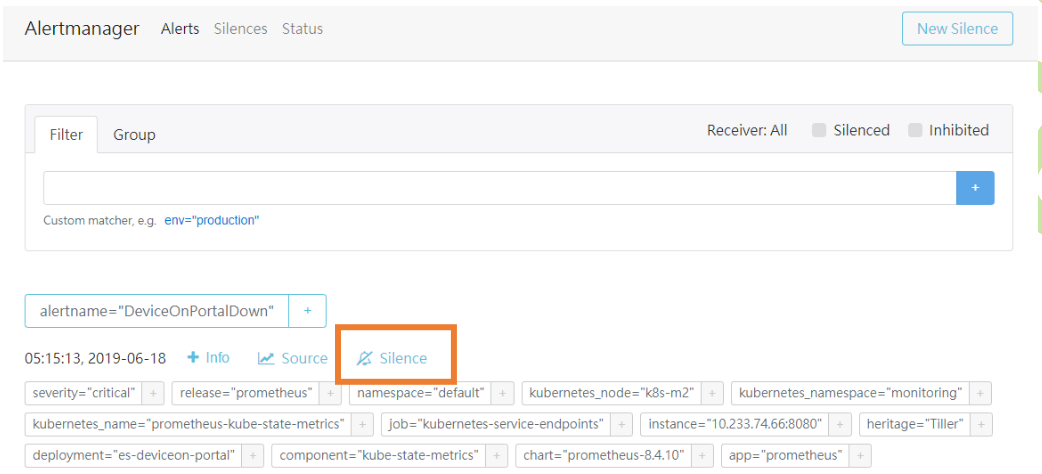
Sent by AlertManager

Figure 19 Alerting email

4.3.2 Silence

Alertmanager offers silence function. User can determine specific alert be silent for an interval not to send alerts.

1. We also take "DeviceOnPortalDown" for example. Click on "Silence".



Alertmanager Alerts Silences Status New Silence

Filter Group Receiver: All Silenced Inhibited

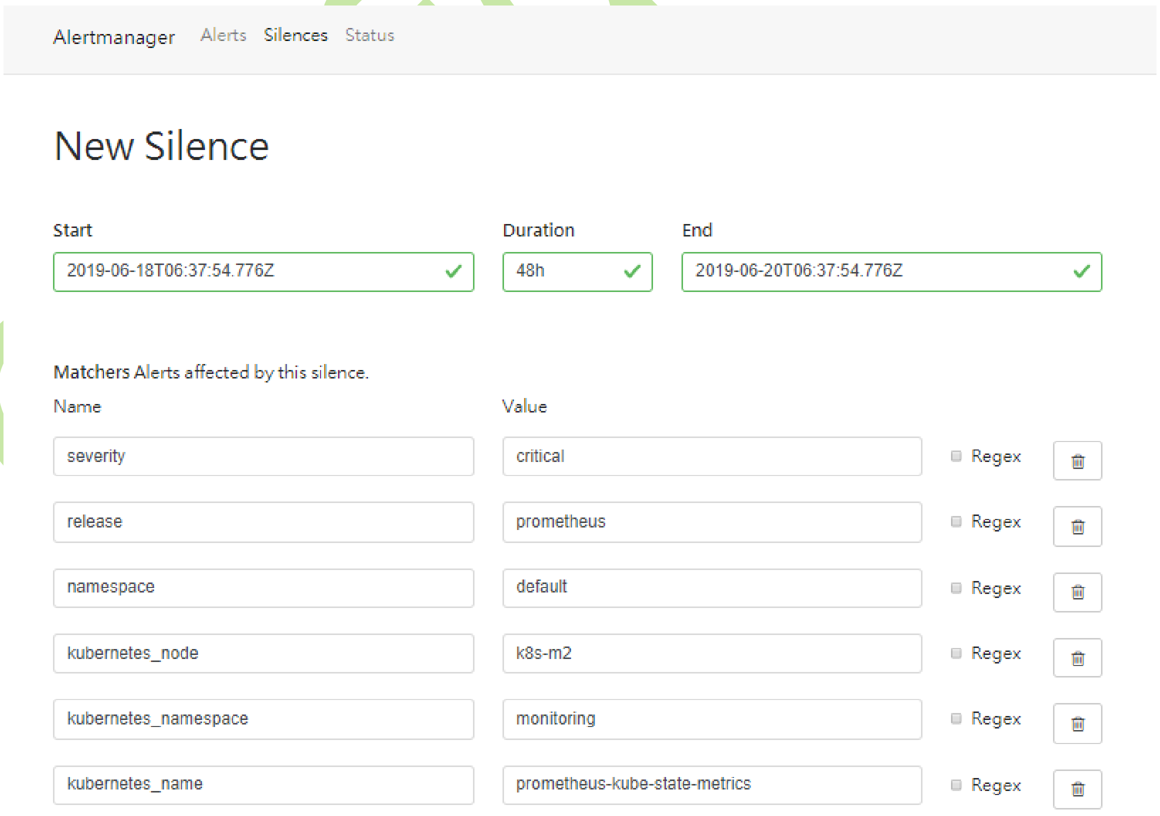
Custom matcher, e.g. env="production"

alertname="DeviceOnPortalDown"

05:15:13, 2019-06-18 Info Source Silence

severity="critical" release="prometheus" namespace="default" kubernetes_node="k8s-m2" kubernetes_namespace="monitoring" kubernetes_name="prometheus-kube-state-metrics" job="kubernetes-service-endpoints" instance="10.233.74.66:8080" heritage="Tiller" deployment="es-deviceon-portal" component="kube-state-metrics" chart="prometheus-8.4.10" app="prometheus"

2. After setting the duration and comment, click on "Create".



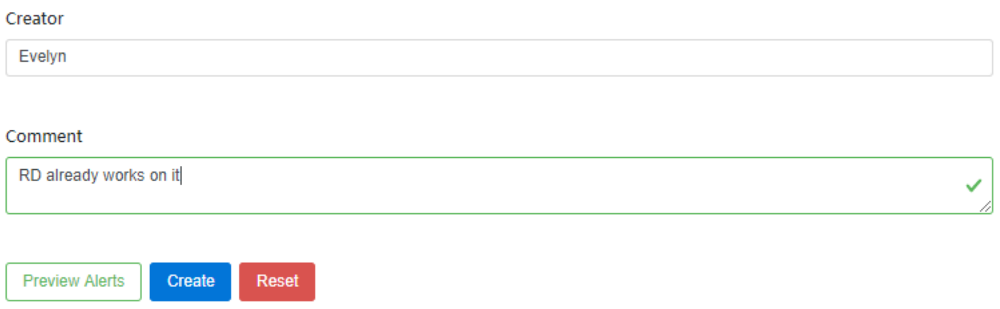
Alertmanager Alerts Silences Status

New Silence

Start	Duration	End
2019-06-18T06:37:54.776Z	48h	2019-06-20T06:37:54.776Z

Matchers Alerts affected by this silence.

Name	Value		
severity	critical	Regex	
release	prometheus	Regex	
namespace	default	Regex	
kubernetes_node	k8s-m2	Regex	
kubernetes_namespace	monitoring	Regex	
kubernetes_name	prometheus-kube-state-metrics	Regex	



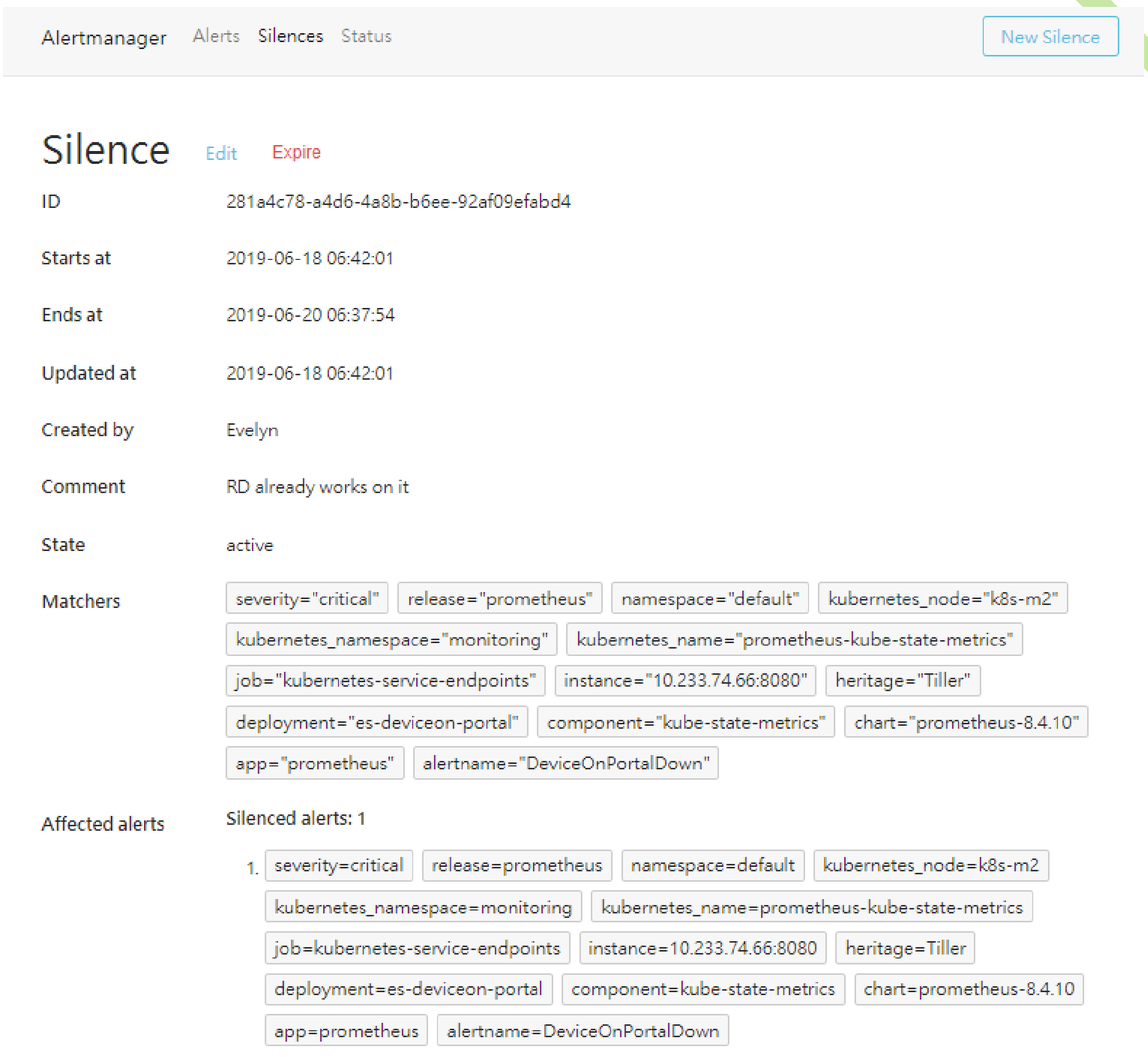
Creator

Evelyn

Comment

RD already works on it

3. User will not receive this alert in the following 48 hours.



Silence Edit Expire

ID	281a4c78-a4d6-4a8b-b6ee-92af09efabd4
Starts at	2019-06-18 06:42:01
Ends at	2019-06-20 06:37:54
Updated at	2019-06-18 06:42:01
Created by	Evelyn
Comment	RD already works on it
State	active
Matchers	severity="critical" release="prometheus" namespace="default" kubernetes_node="k8s-m2" kubernetes_namespace="monitoring" kubernetes_name="prometheus-kube-state-metrics" job="kubernetes-service-endpoints" instance="10.233.74.66:8080" heritage="Tiller" deployment="es-deviceon-portal" component="kube-state-metrics" chart="prometheus-8.4.10" app="prometheus" alertname="DeviceOnPortalDown"
Affected alerts	Silenced alerts: 1 1. severity=critical release=prometheus namespace=default kubernetes_node=k8s-m2 kubernetes_namespace=monitoring kubernetes_name=prometheus-kube-state-metrics job=kubernetes-service-endpoints instance=10.233.74.66:8080 heritage=Tiller deployment=es-deviceon-portal component=kube-state-metrics chart=prometheus-8.4.10 app=prometheus alertname=DeviceOnPortalDown

Note:

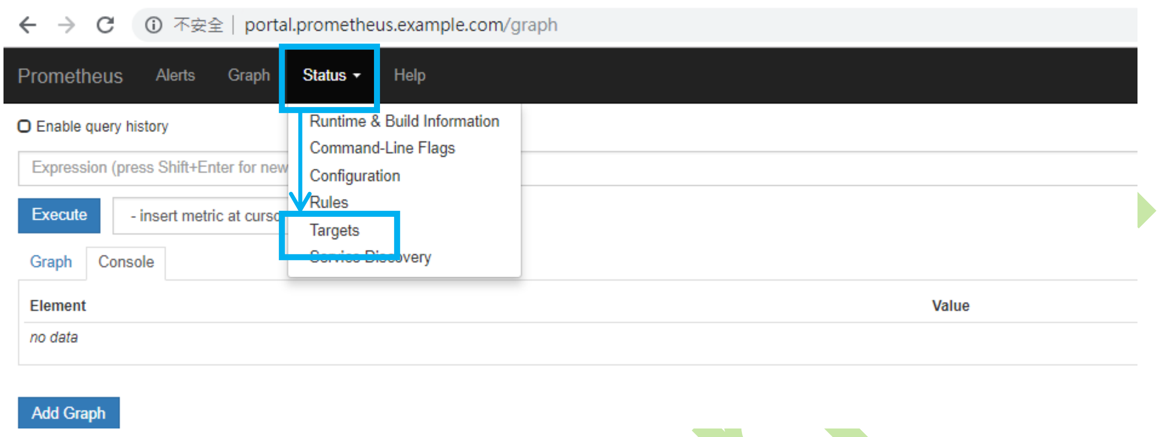
1. User can edit silence anytime, user can make it expire or extend the duration.

2. You won’ t receive alerting if silence is set, please use it carefully.

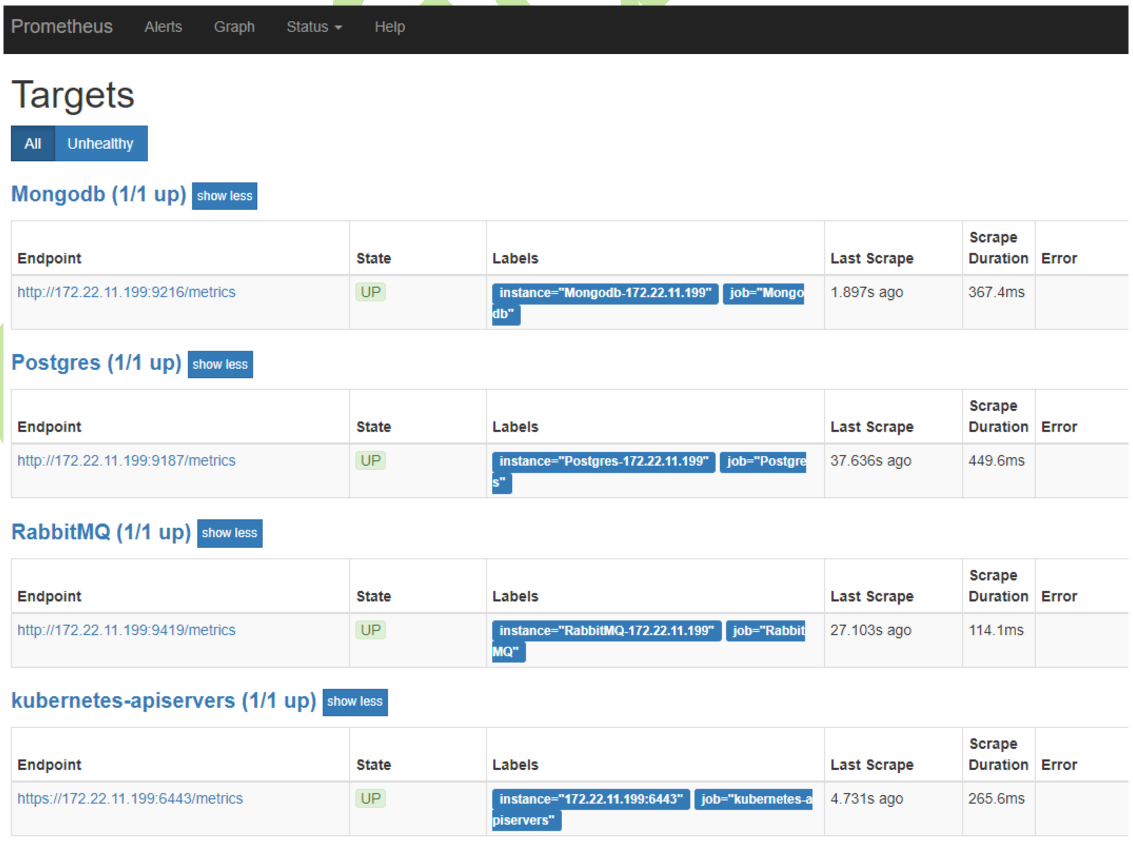
FAQ

✧ How to simply check if all the targets are under monitored?

1. Go to your Prometheus Web UI and navigate to “Targets” page.



2. All the monitored targets will be listed as below, the targets will show “up” if their status are healthy.


Prometheus Alerts Graph Status Help

Targets

All | Unhealthy

Mongodb (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.22.11.199:9216/metrics	UP	instance="Mongodb-172.22.11.199" job="Mongodb"	1.897s ago	367.4ms	

Postgres (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.22.11.199:9187/metrics	UP	instance="Postgres-172.22.11.199" job="Postgres"	37.636s ago	449.6ms	

RabbitMQ (1/1 up) show less

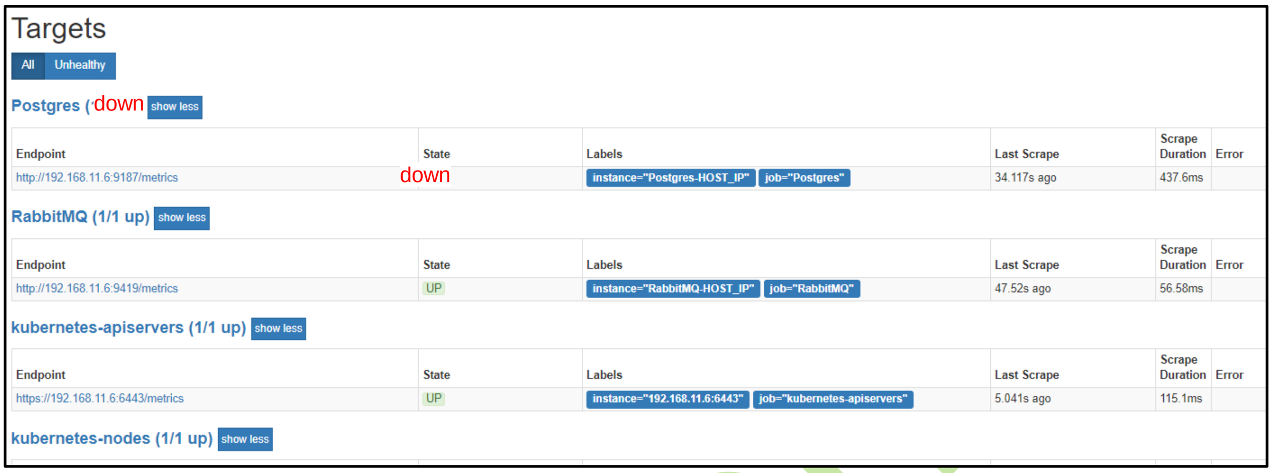
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.22.11.199:9419/metrics	UP	instance="RabbitMQ-172.22.11.199" job="RabbitMQ"	27.103s ago	114.1ms	

kubernetes-apiservers (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://172.22.11.199:6443/metrics	UP	instance="172.22.11.199:6443" job="kubernetes-apiservers"	4.731s ago	265.6ms	

✧ As mentioned above, if there is a target with "down" status and error message, how to check the error in detail?

Take "Postgres" for example and assume "Postgres" is down.



Targets

All | Unhealthy

Postgres ('down show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.11.6:9187/metrics	down	instance="Postgres-HOST_IP" job="Postgres"	34.117s ago	437.6ms	

RabbitMQ (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.11.6:9419/metrics	UP	instance="RabbitMQ-HOST_IP" job="RabbitMQ"	47.52s ago	56.58ms	

kubernetes-apiservers (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.11.6:6443/metrics	UP	instance="192.168.11.6:6443" job="kubernetes-apiservers"	5.041s ago	115.1ms	

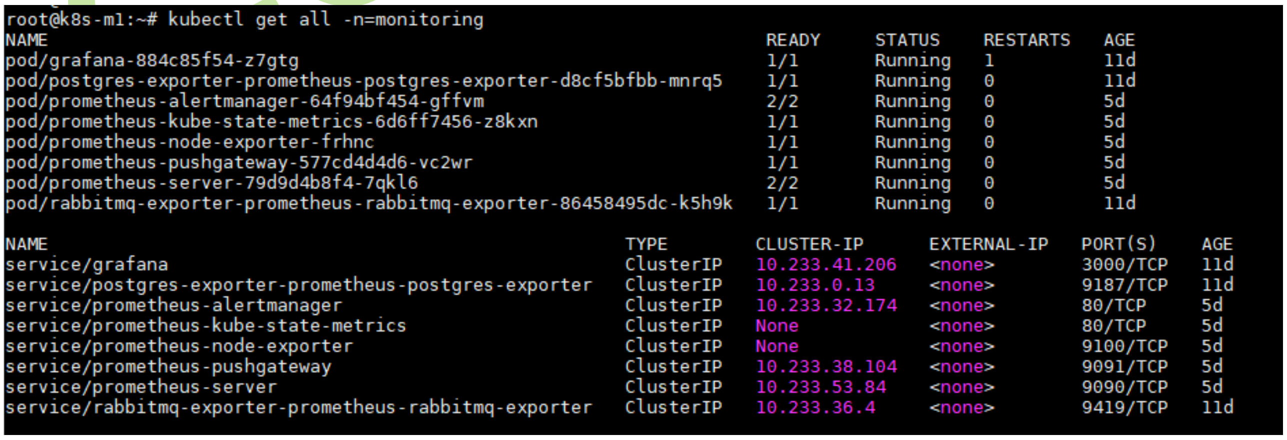
kubernetes-nodes (1/1 up) show less

Please use command kubectl to get more information.

3. Issue command to get all resources in namespace "monitoring".

```
# kubectl get all -n=monitoring
```

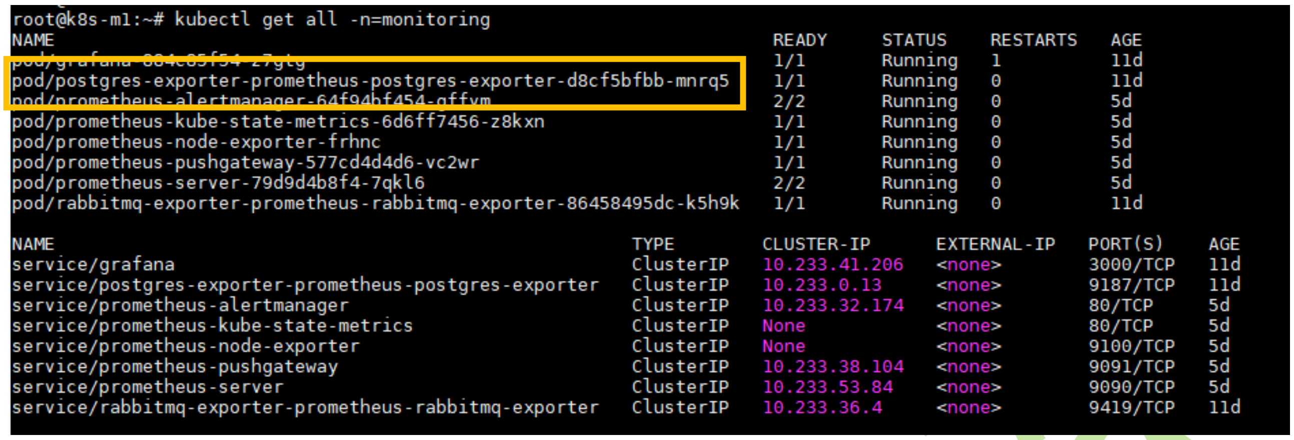
All the resources in "monitoring" will show as below.



```
root@k8s-m1:~# kubectl get all -n=monitoring
NAME                                                         READY   STATUS    RESTARTS   AGE
pod/grafana-884c85f54-z7gtg                                  1/1     Running   1          11d
pod/postgres-exporter-prometheus-postgres-exporter-d8cf5bfbb-mnrq5   1/1     Running   0          11d
pod/prometheus-alertmanager-64f94bf454-gffvm                 2/2     Running   0          5d
pod/prometheus-kube-state-metrics-6d6ff7456-z8kxn            1/1     Running   0          5d
pod/prometheus-node-exporter-frhnc                           1/1     Running   0          5d
pod/prometheus-pushgateway-577cd4d4d6-vc2wr                  1/1     Running   0          5d
pod/prometheus-server-79d9d4b8f4-7qkl6                       2/2     Running   0          5d
pod/rabbitmq-exporter-prometheus-rabbitmq-exporter-86458495dc-k5h9k  1/1     Running   0          11d

NAME                                                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/grafana                                         ClusterIP   10.233.41.206   <none>        3000/TCP   11d
service/postgres-exporter-prometheus-postgres-exporter  ClusterIP   10.233.0.13     <none>        9187/TCP   11d
service/prometheus-alertmanager                         ClusterIP   10.233.32.174   <none>        80/TCP     5d
service/prometheus-kube-state-metrics                   ClusterIP   None            <none>        80/TCP     5d
service/prometheus-node-exporter                        ClusterIP   None            <none>        9100/TCP   5d
service/prometheus-pushgateway                          ClusterIP   10.233.38.104   <none>        9091/TCP   5d
service/prometheus-server                               ClusterIP   10.233.53.84    <none>        9090/TCP   5d
service/rabbitmq-exporter-prometheus-rabbitmq-exporter  ClusterIP   10.233.36.4     <none>        9419/TCP   11d
```

4. We need to get more information from "Postgres" Pod.



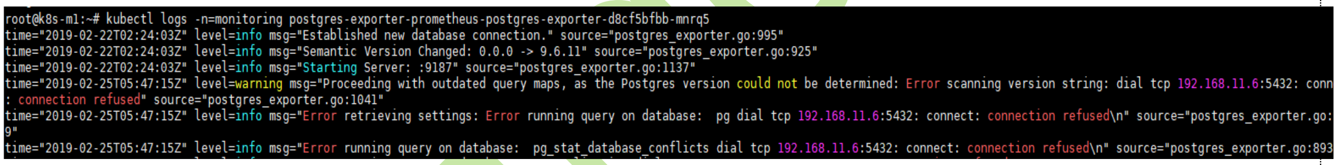
```
root@k8s-m1:~# kubectl get all -n=monitoring
NAME                                                              READY     STATUS    RESTARTS   AGE
pod/grafana-884c85f54-z7gtg                                       1/1       Running   1          11d
pod/postgres-exporter-prometheus-postgres-exporter-d8cf5bfbb-mnrq5   1/1       Running   0          11d
pod/prometheus-alertmanager-64f94bf454-qffvm                      2/2       Running   0          5d
pod/prometheus-kube-state-metrics-6d6ff7456-z8kxn                 1/1       Running   0          5d
pod/prometheus-node-exporter-frhnc                                1/1       Running   0          5d
pod/prometheus-pushgateway-577cd4d4d6-vc2wr                       1/1       Running   0          5d
pod/prometheus-server-79d9d4b8f4-7qkl6                            2/2       Running   0          5d
pod/rabbitmq-exporter-prometheus-rabbitmq-exporter-86458495dc-k5h9k   1/1       Running   0          11d

NAME                                                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/grafana                                           ClusterIP   10.233.41.206   <none>        3000/TCP   11d
service/postgres-exporter-prometheus-postgres-exporter    ClusterIP   10.233.0.13     <none>        9187/TCP   11d
service/prometheus-alertmanager                           ClusterIP   10.233.32.174   <none>        80/TCP     5d
service/prometheus-kube-state-metrics                     ClusterIP   None            <none>        80/TCP     5d
service/prometheus-node-exporter                          ClusterIP   None            <none>        9100/TCP   5d
service/prometheus-pushgateway                            ClusterIP   10.233.38.104   <none>        9091/TCP   5d
service/prometheus-server                                 ClusterIP   10.233.53.84    <none>        9090/TCP   5d
service/rabbitmq-exporter-prometheus-rabbitmq-exporter    ClusterIP   10.233.36.4     <none>        9419/TCP   11d
```

```
# kubectl logs -n=monitoring ${POD_NAME}
```

Please fill in your pod name here and you will see the logs of this pod.

i.e. kubectl logs -n=monitoring postgres-exporter-prometheus-postgres-exporter-d8cf5bfbb-mnrq5



```
root@k8s-m1:~# kubectl logs -n=monitoring postgres-exporter-prometheus-postgres-exporter-d8cf5bfbb-mnrq5
time="2019-02-22T02:24:03Z" level=info msg="Established new database connection." source="postgres_exporter.go:995"
time="2019-02-22T02:24:03Z" level=info msg="Semantic Version Changed: 0.0.0 -> 9.6.11" source="postgres_exporter.go:925"
time="2019-02-22T02:24:03Z" level=info msg="Starting Server: :9187" source="postgres_exporter.go:1137"
time="2019-02-25T05:47:15Z" level=warning msg="Proceeding with outdated query maps, as the Postgres version could not be determined: Error scanning version string: dial tcp 192.168.11.6:5432: conn
: connection refused" source="postgres_exporter.go:1041"
time="2019-02-25T05:47:15Z" level=info msg="Error retrieving settings: Error running query on database:  pg dial tcp 192.168.11.6:5432: connect: connection refused\n" source="postgres_exporter.go:
9"
time="2019-02-25T05:47:15Z" level=info msg="Error running query on database:  pg_stat_database_conflicts dial tcp 192.168.11.6:5432: connect: connection refused\n" source="postgres_exporter.go:893
```

5. By the two steps above, if you can correct the error successfully, congrats!

But if you still have no ideas, please issue command below.

```
# kubectl describe pod -n=monitoring ${POD_NAME}
  Please fill in your pod name here.
```

i.e. kubectl describe pod -n=monitoring postgres-exporter-prometheus-postgres-exporter-d8cf5bfbb-mnrq5

Please copy and paste all the information from "kubectl logs" and "kubectl describe" commands and contact Advantech support team. Thanks!

Reference

[1] "Kubernetes," [Online]. Available: https://kubernetes.io/.

[2] Weaveworks, "Monitoring Kubernetes with Prometheus," [Online]. Available: https://www.weave.works/technologies/monitoring-kubernetes-with-prometheus/.

[3] Tom, "Prometheus and Kubernetes: A Perfect Match," [Online]. Available: https://www.weave.works/blog/prometheus-kubernetes-perfect-match/.

[4] 殷. 蔡. 吴莉, "Prometheus 入门与实践," [Online]. Available: https://www.ibm.com/developerworks/cn/cloud/library/cl-lo-prometheus-getting-started-and-practice/index.html.

[5] S. BISWAS. [Online]. Available: https://www.booleanworld.com/install-use-prometheus-monitoring/.

[6] "Github of Prometheus/pushgateway," [Online]. Available: https://github.com/prometheus/pushgateway.

[7] K. Aziz. [Online]. Available: http://weblogs.com.pk/khurram/archive/2018/04/16/prometheus.aspx.

[8] B. Brazil, Prometheus: Up & Running, O'Reilly Media, Inc., 2018.

[9] D. Berman, "Kubernetes Monitoring: Best Practices, Methods, and Existing Solutions," [Online]. Available: https://logz.io/blog/kubernetes-monitoring/.

[10] A. Ellis, "Monitor your applications with Prometheus," [Online]. Available: https://blog.alexellis.io/prometheus-monitoring/.

[11] [Online]. Available: https://kubernetes.io/docs/concepts/storage/persistent-volumes/.

[12] devopscube, "Kubernetes Deployment Tutorial For Beginners," [Online]. Available: https://devopscube.com/kubernetes-deployment-tutorial/.

[13] "Kubernetes concepts: StatefulSets," [Online]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/.

[14] M. Burillo, "Kubernetes Monitoring with Prometheus -The ultimate guide," [Online]. Available: https://sysdig.com/blog/kubernetes-monitoring-prometheus/.

[15] Kubedex, "Prometheus-Rabbitmq-Exporter," [Online]. Available: https://kubedex.com/resource/prometheus-rabbitmq-exporter/.

[16] "Github Docs of Postgres exporter," [Online]. Available: https://docs.gitlab.com/ee/administration/monitoring/prometheus/postgres_exporter.html.

[17] P. org, "About PostgreSQL," [Online]. Available: https://www.postgresql.org/about/.

[18] "mongoDB," [Online]. Available: https://docs.mongodb.com/manual/introduction/.

[19] [Online]. Available: https://prometheus.io/docs/alerting/alertmanager/.