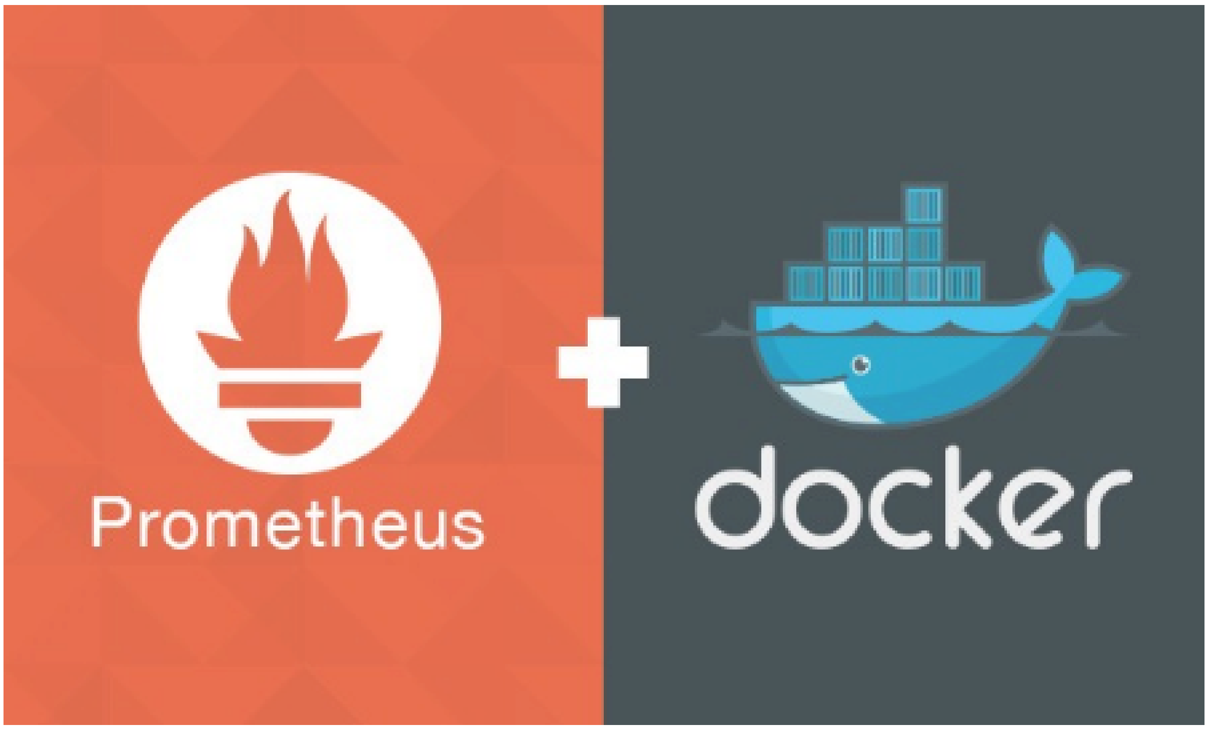
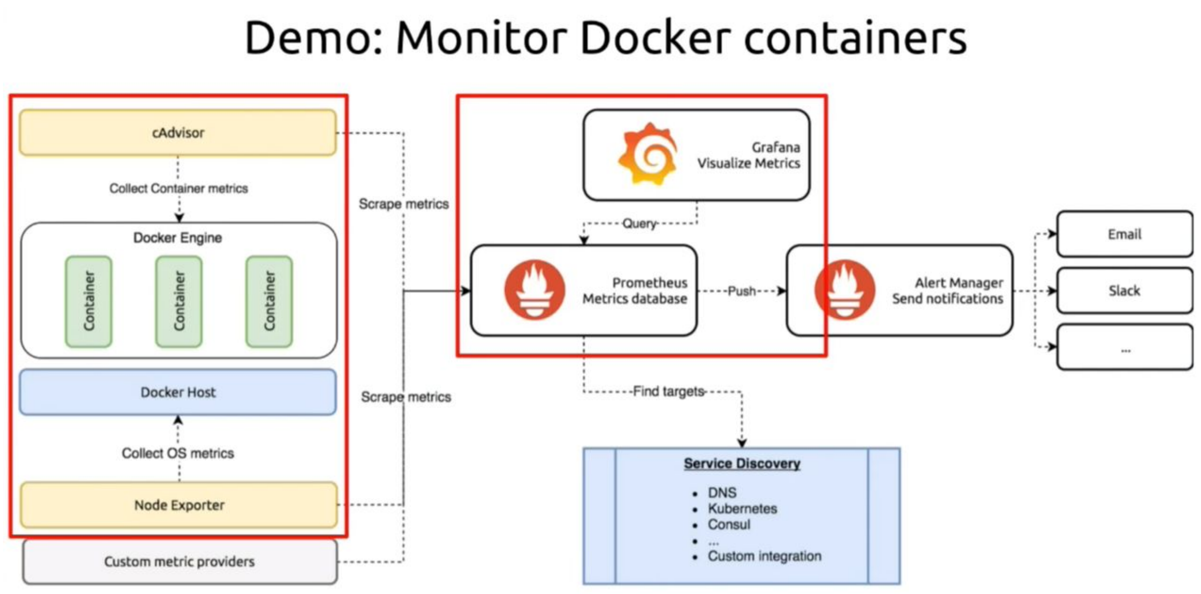


Docker Monitoring

Índice

Project diagram



Concept about software of PI

What is Prometheus?

Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company. To emphasize this, and to clarify the project's governance structure, Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

Grafana

Grafana is the software for time series analytics, with well over 100,000 active installations. Customers turn to Grafana Labs to help bring their disparate data sources together, all through software that is vendor neutral and open source.

Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored.

cAdvisor

cAdvisor (Container Advisor) provides container users an understanding of the resource usage and performance characteristics of their running containers. It is a running daemon that collects, aggregates, processes, and exports information about running containers. Specifically, for each container it keeps resource isolation parameters, historical resource usage, and histograms of complete historical resource usage. This data is exported by container and machine-wide.

Alertmanager

The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of duplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie. It also takes care of silencing and inhibition of alerts.

Installation and configuration

1º Step - Docker

Install docker
→ apt-get install docker.io

2º Step - Docker-Compose

Install docker-compose

- sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
- sudo chmod +x /usr/local/bin/docker-compose
- docker-compose --version

3º Step - Download PI

Download the project and unzip, extract

→ wget https://github.com/juanmi1994/Prometheus-Grafana/archive/master.zip

→ unzip master.zip

→ Move to file of project and "execute docker-compose up -d"

With this steps, we will get started the containers for obtains the metrics, graph and alert.

The project will launch 5 containers

Prometheus → metrics database
AlertManager → alerts management / send notification for example through email
Grafana → visualize metrics
NodeExporter → host metrics collector
CadVisor → container metrics collector
Caddy → reverse proxy and basic auth provider for prometheus and alertmanager

4º Step - Test

Time of trying it in browser, we will need put the ip:port

Example; 192.168.100.1:9090

:9090(Prometheus)

:3000(Grafana)

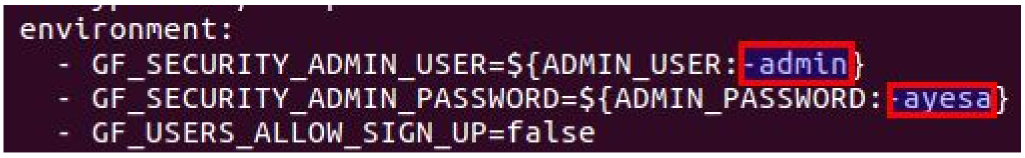
:9093(Alertmanager)

Configuration of project "Prometheus"

"*It is very important to follow the same nomenclature and the same rules of spaces in the configuration files, if they are not well written as it is established, the service will fall automatically*."

Change auth basic

For change the user and password we will have to modify the docker-compose.yml and change the parameters of authentication

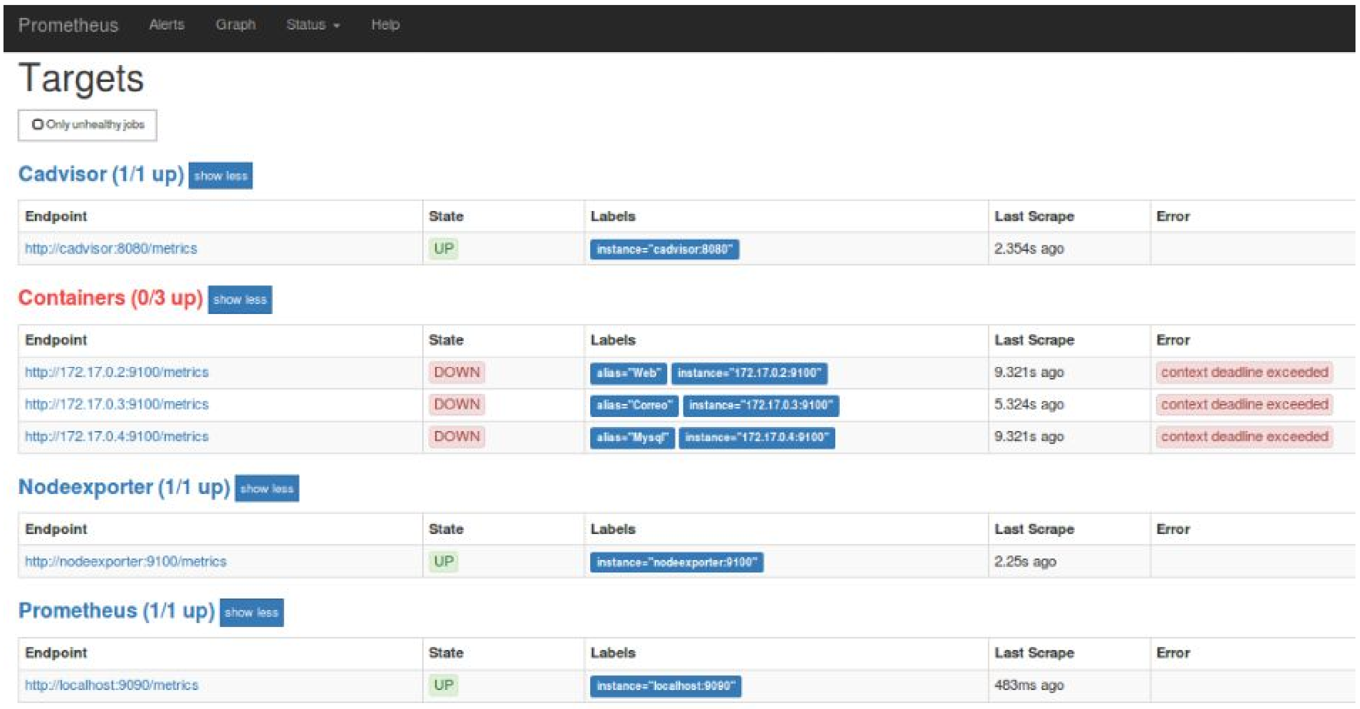


```
environment:
  - GF_SECURITY_ADMIN_USER=${ADMIN_USER:-admin}
  - GF_SECURITY_ADMIN_PASSWORD=${ADMIN_PASSWORD:-ayesa}
  - GF_USERS_ALLOW_SIGN_UP=false
```

Introduction of a container to track alerts

In the project we have a file on the route; /proyecto-Neff/prometheus "prometheus.yml" inside we need to write the following line, this is an example;

```
- job_name: 'Containers'              (Name of target)
    scrape_interval: 5s                (Time interval)
    static_configs:
      - targets: ['172.17.0.2:9100']   (IP+Ports)
        labels:
         alias: 'Web'                  (Alias of container)
```



Targets

Only unhealthy jobs

Cadvisor (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Error
http://cadvisor:8080/metrics	UP	instance="cadvisor:8080"	2.354s ago	

Containers (0/3 up) show less

Endpoint	State	Labels	Last Scrape	Error
http://172.17.0.2:9100/metrics	DOWN	alias="Web" instance="172.17.0.2:9100"	9.321s ago	context deadline exceeded
http://172.17.0.3:9100/metrics	DOWN	alias="Correo" instance="172.17.0.3:9100"	5.324s ago	context deadline exceeded
http://172.17.0.4:9100/metrics	DOWN	alias="Mysql" instance="172.17.0.4:9100"	9.321s ago	context deadline exceeded

Nodeexporter (1/1 up) show less

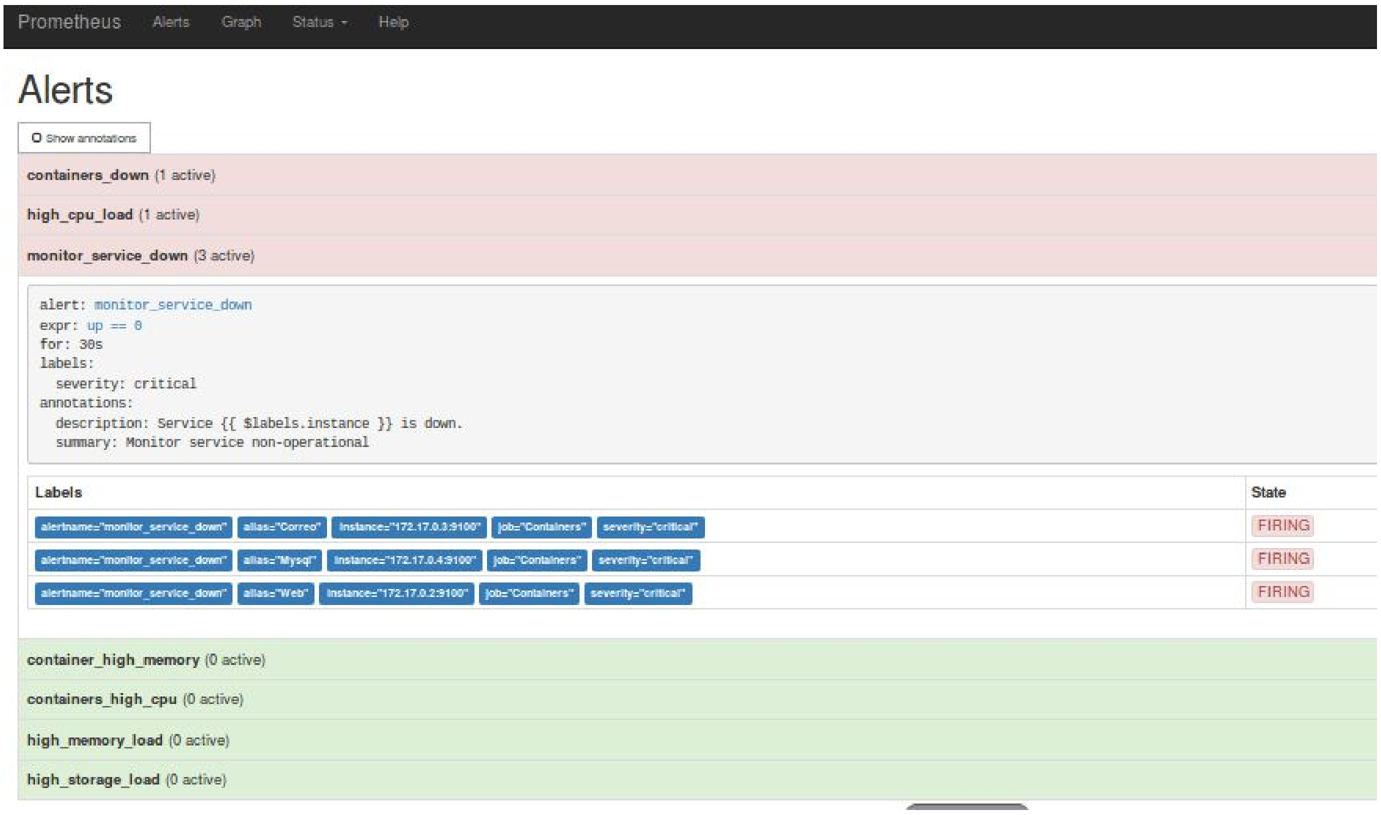
Endpoint	State	Labels	Last Scrape	Error
http://nodeexporter:9100/metrics	UP	instance="nodeexporter:9100"	2.25s ago	

Prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Error
http://localhost:9090/metrics	UP	instance="localhost:9090"	483ms ago	

Introduction and creation of alerts

In the same route as the previous example /proyecto-Neff/prometheus "alert.rules" inside we can create new rules or modify any. This will be displayed;



Alerts

Show annotations

containers_down (1 active)

high_cpu_load (1 active)

monitor_service_down (3 active)

```
alert: monitor_service_down
expr: up == 0
for: 30s
labels:
  severity: critical
annotations:
  description: Service {{ $labels.instance }} is down.
  summary: Monitor service non-operational
```

Labels	State
alertname="monitor_service_down" alias="Correo" instance="172.17.0.3:9100" job="Containers" severity="critical"	FIRING
alertname="monitor_service_down" alias="Mysql" instance="172.17.0.4:9100" job="Containers" severity="critical"	FIRING
alertname="monitor_service_down" alias="Web" instance="172.17.0.2:9100" job="Containers" severity="critical"	FIRING

container_high_memory (0 active)

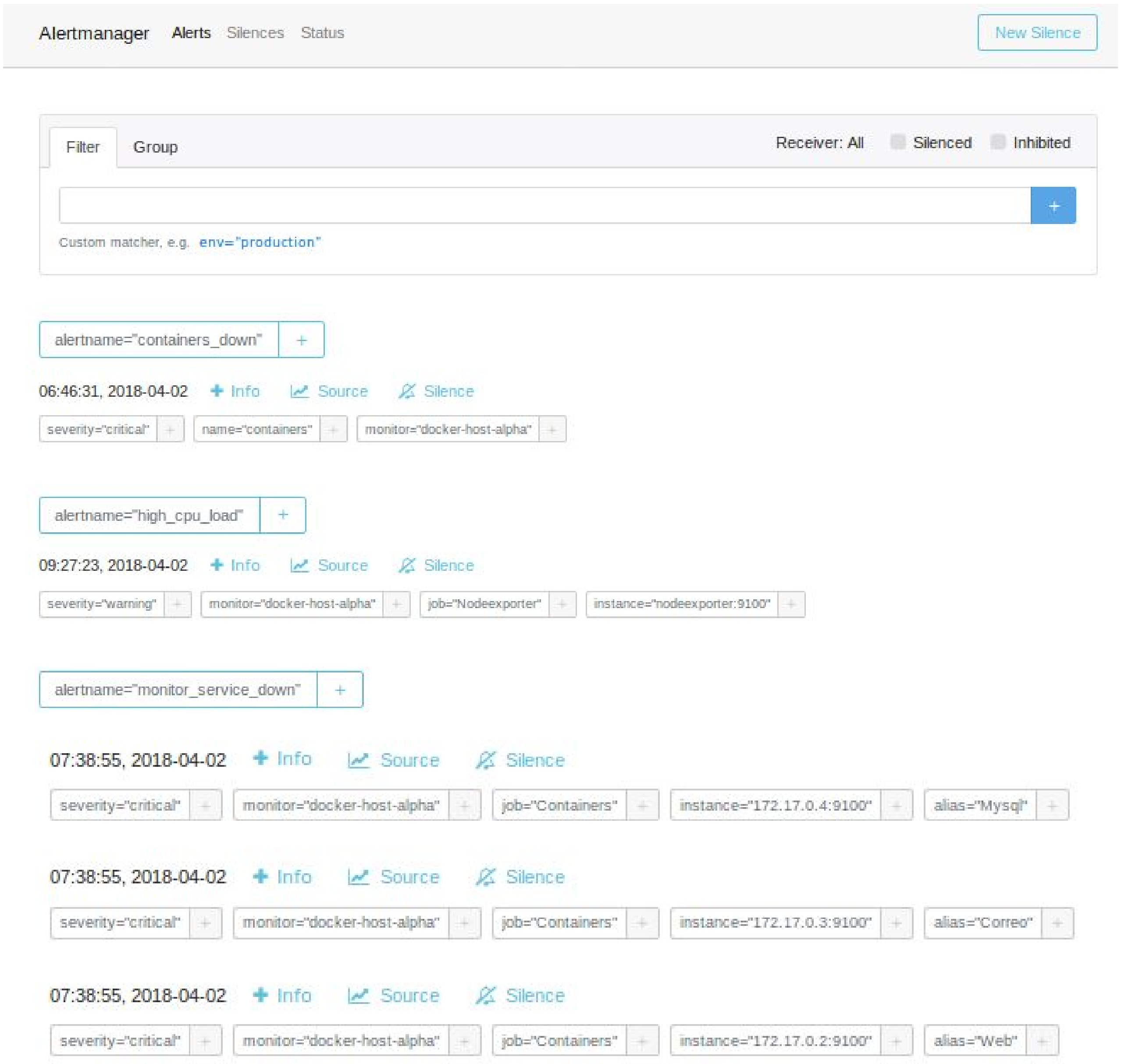
containers_high_cpu (0 active)

high_memory_load (0 active)

high_storage_load (0 active)

Alertmanager

Alertmanager is responsible for duplicating, grouping and routing them to the correct integration of the receiver, such as email. It is also responsible for silencing and inhibiting alerts.



alertname="containers_down"
06:46:31, 2018-04-02 Info Source Silence
severity="critical" name="containers" monitor="docker-host-alpha"

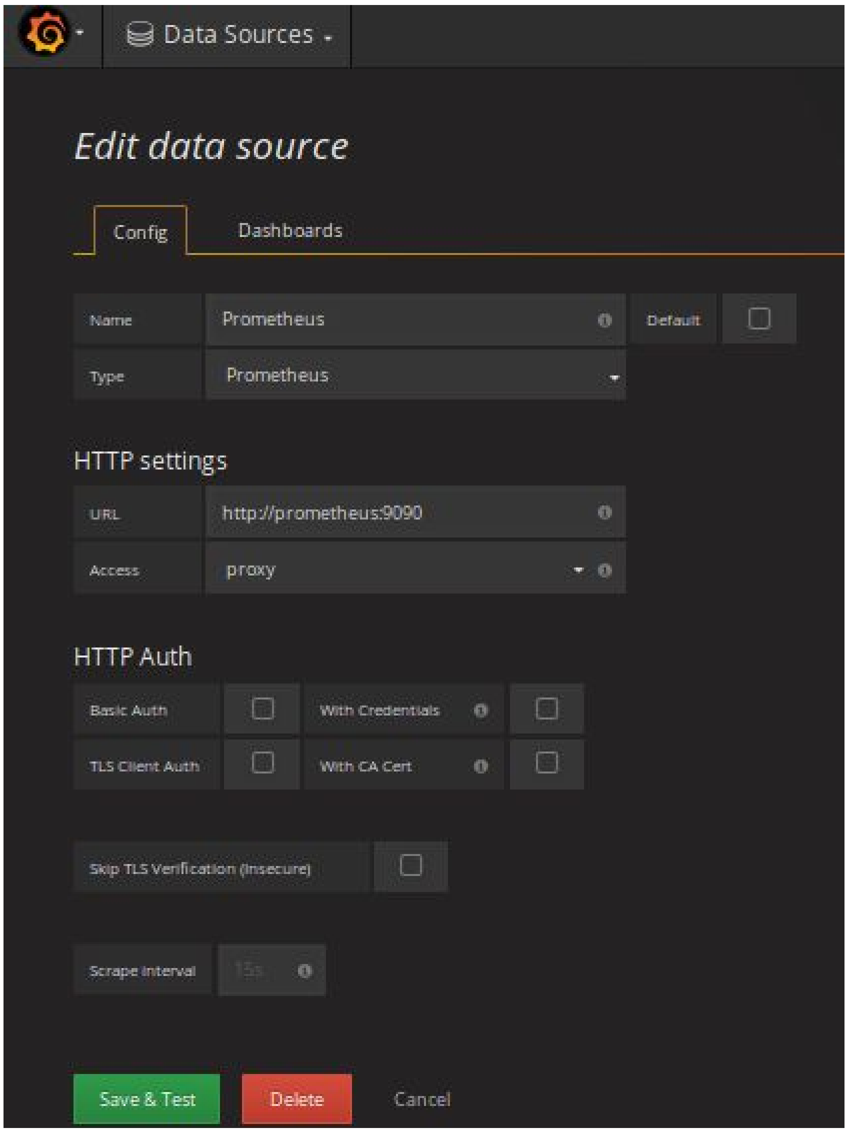
alertname="high_cpu_load"
09:27:23, 2018-04-02 Info Source Silence
severity="warning" monitor="docker-host-alpha" job="Nodeexporter" instance="nodeexporter:9100"

alertname="monitor_service_down"
07:38:55, 2018-04-02 Info Source Silence
severity="critical" monitor="docker-host-alpha" job="Containers" instance="172.17.0.4:9100" alias="Mysql"
07:38:55, 2018-04-02 Info Source Silence
severity="critical" monitor="docker-host-alpha" job="Containers" instance="172.17.0.3:9100" alias="Correo"
07:38:55, 2018-04-02 Info Source Silence
severity="critical" monitor="docker-host-alpha" job="Containers" instance="172.17.0.2:9100" alias="Web"

Grafana

In order for the software to collect the information of the machine and the containers, it is necessary create a new data source, it will collect all information reference to containers and hosts machines

Data source



Edit data source

Name	Prometheus	Default	☐
Type	Prometheus		

HTTP settings

URL	http://prometheus:9090
Access	proxy

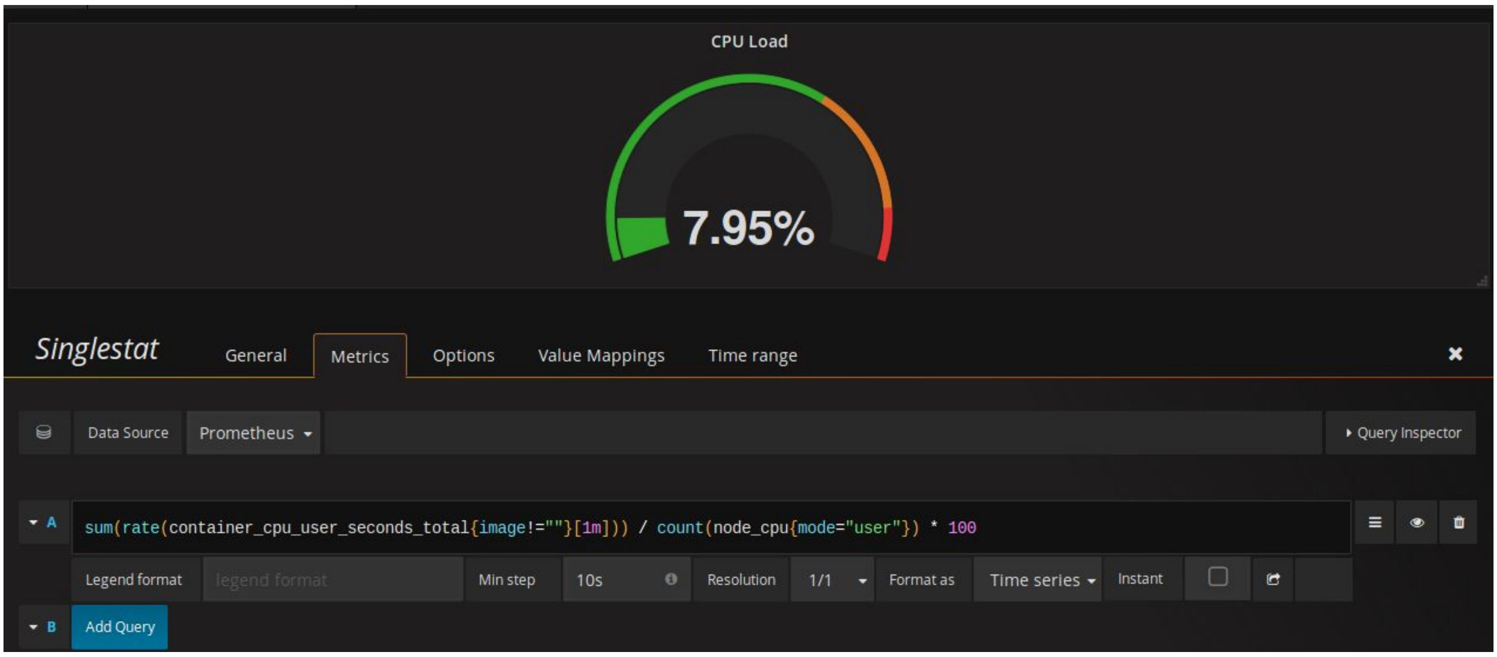
HTTP Auth

Basic Auth	☐	With Credentials	☐
TLS Client Auth	☐	With CA Cert	☐

Skip TLS Verification (Insecure)	☐
Scrape interval	15s

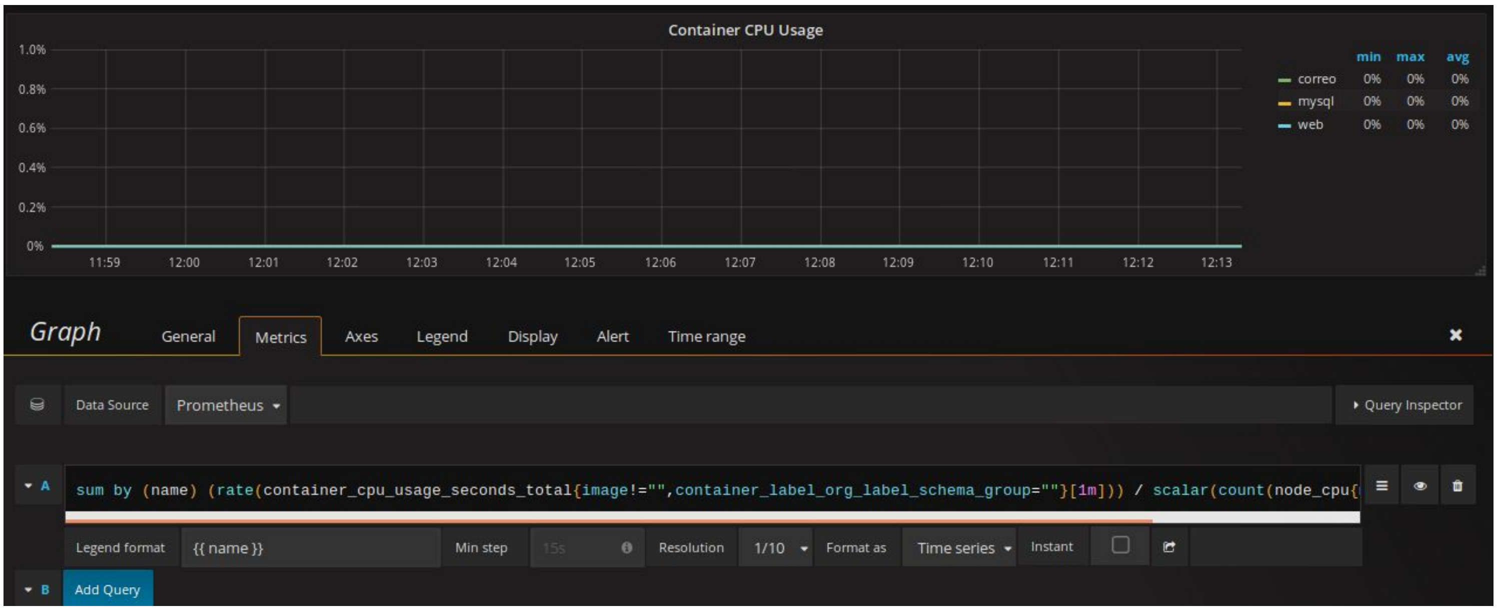
Dashboards

Now we have the information and we will need a list, for this we create the following boards with information regarding the host team and containers.
We already have the dashboards now we are adding the graphics that we need.



```
sum(rate(container_cpu_user_seconds_total{image!=""}[1m])) / count(node_cpu{mode="user"}) * 100
```

Legend format: legend format | Min step: 10s | Resolution: 1/1 | Format as: Time series | Instant

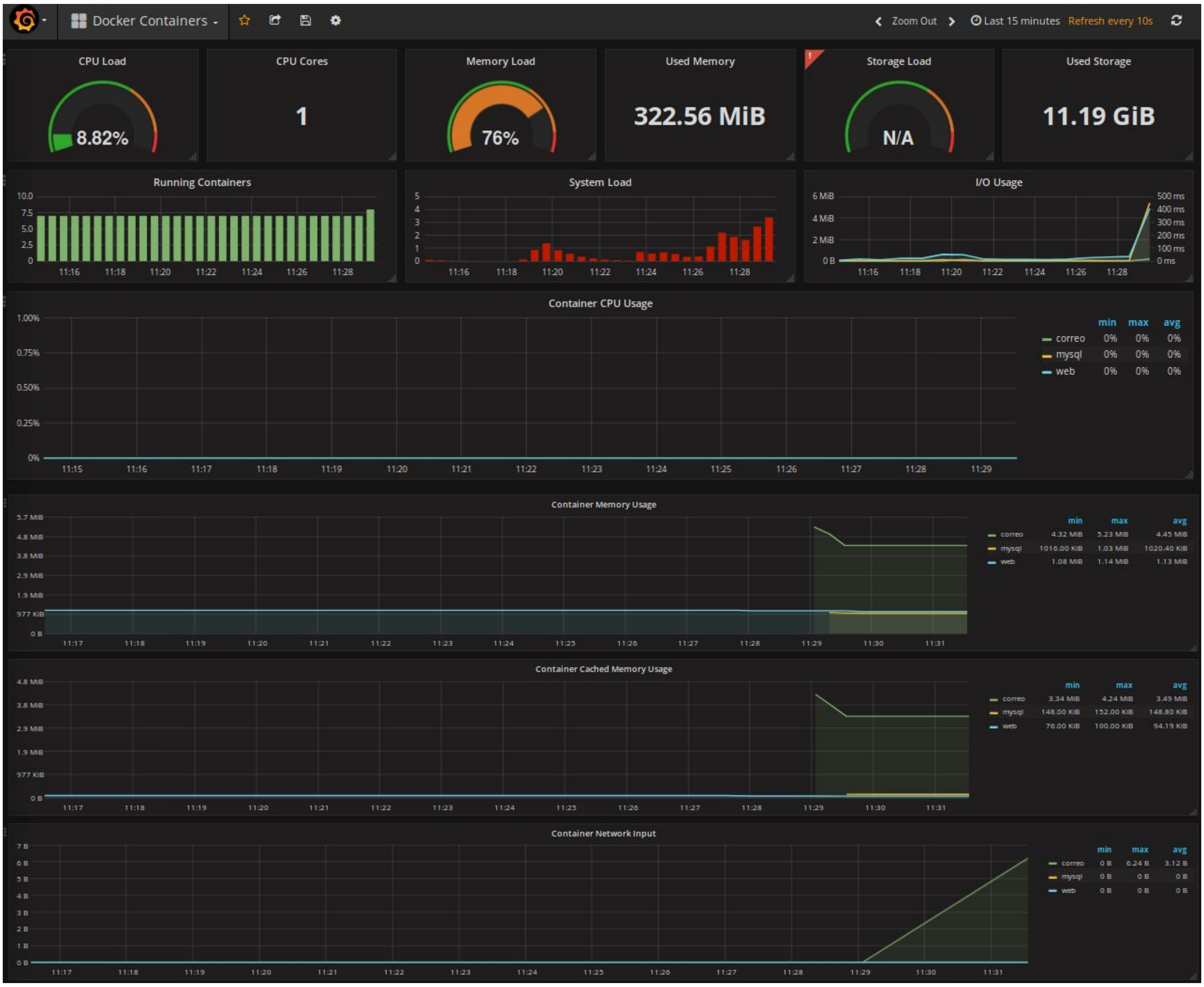


	min	max	avg
correo	0%	0%	0%
mysql	0%	0%	0%
web	0%	0%	0%

```
sum by (name) (rate(container_cpu_usage_seconds_total{image!="",container_label_org_label_schema_group=""}[1m])) / scalar(count(node_cpu{
```

Legend format: {{ name }} | Min step: 15s | Resolution: 1/10 | Format as: Time series | Instant

❖ Docker containers monitoring



Container CPU Usage

	min	max	avg
correo	0%	0%	0%
mysql	0%	0%	0%
web	0%	0%	0%

Container Memory Usage

	min	max	avg
correo	4.32 MiB	5.23 MiB	4.45 MiB
mysql	1016.00 KiB	1.03 MiB	1020.40 KiB
web	1.08 MiB	1.14 MiB	1.13 MiB

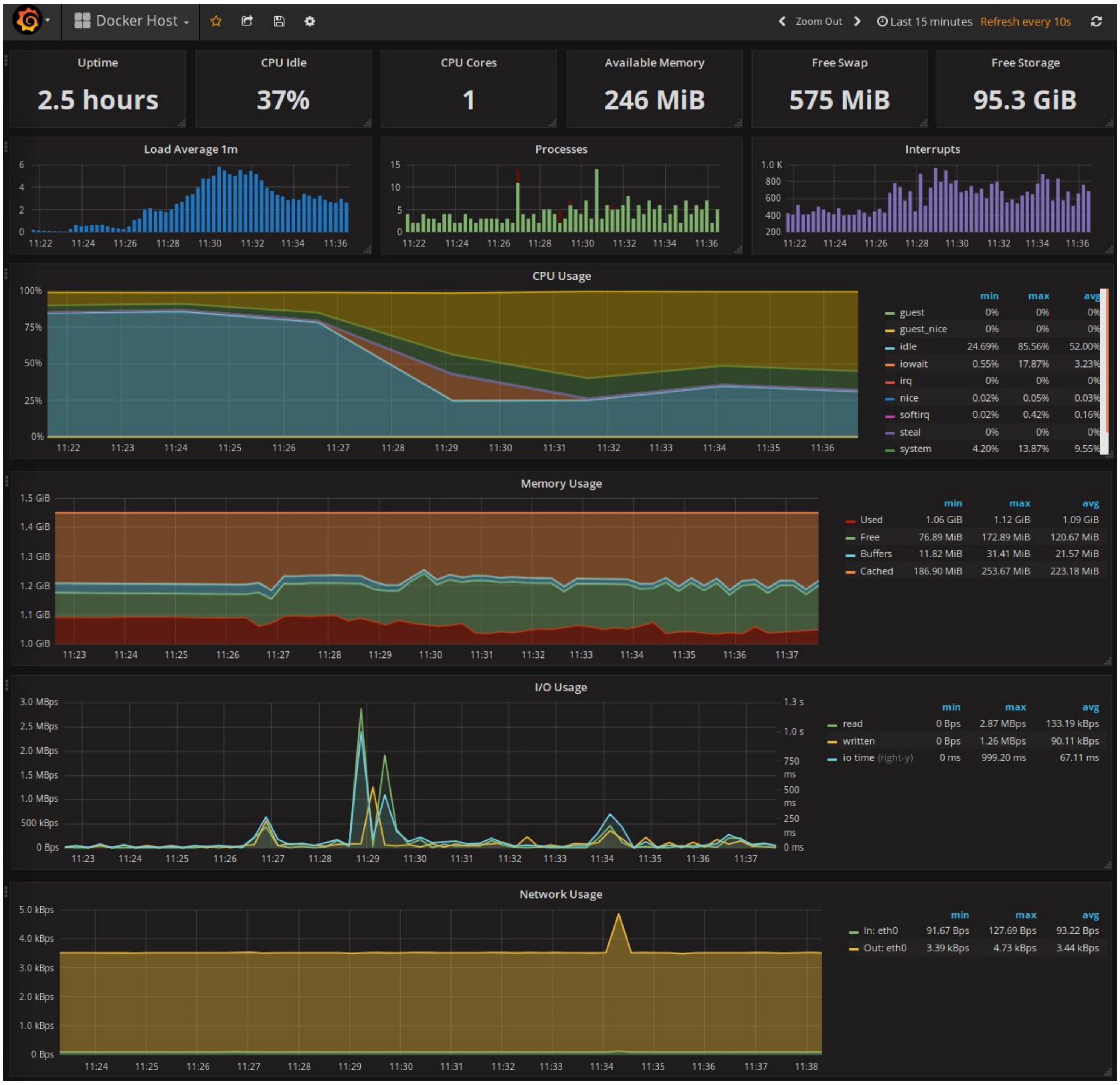
Container Cached Memory Usage

	min	max	avg
correo	3.34 MiB	4.24 MiB	3.49 MiB
mysql	148.00 KiB	152.00 KiB	148.80 KiB
web	76.00 KiB	100.00 KiB	94.19 KiB

Container Network Input

	min	max	avg
correo	0 B	6.24 B	3.12 B
mysql	0 B	0 B	0 B
web	0 B	0 B	0 B

❖ Docker Host monitoring



Uptime	CPU Idle	CPU Cores	Available Memory	Free Swap	Free Storage
2.5 hours	37%	1	246 MiB	575 MiB	95.3 GiB

CPU Usage

	min	max	avg
guest	0%	0%	0%
guest_nice	0%	0%	0%
idle	24.69%	85.56%	52.00%
iowait	0.55%	17.87%	3.23%
irq	0%	0%	0%
nice	0.02%	0.05%	0.03%
softirq	0.02%	0.42%	0.16%
steal	0%	0%	0%
system	4.20%	13.87%	9.55%

Memory Usage

	min	max	avg
Used	1.06 GiB	1.12 GiB	1.09 GiB
Free	76.89 MiB	172.89 MiB	120.67 MiB
Buffers	11.82 MiB	31.41 MiB	21.57 MiB
Cached	186.90 MiB	253.67 MiB	223.18 MiB

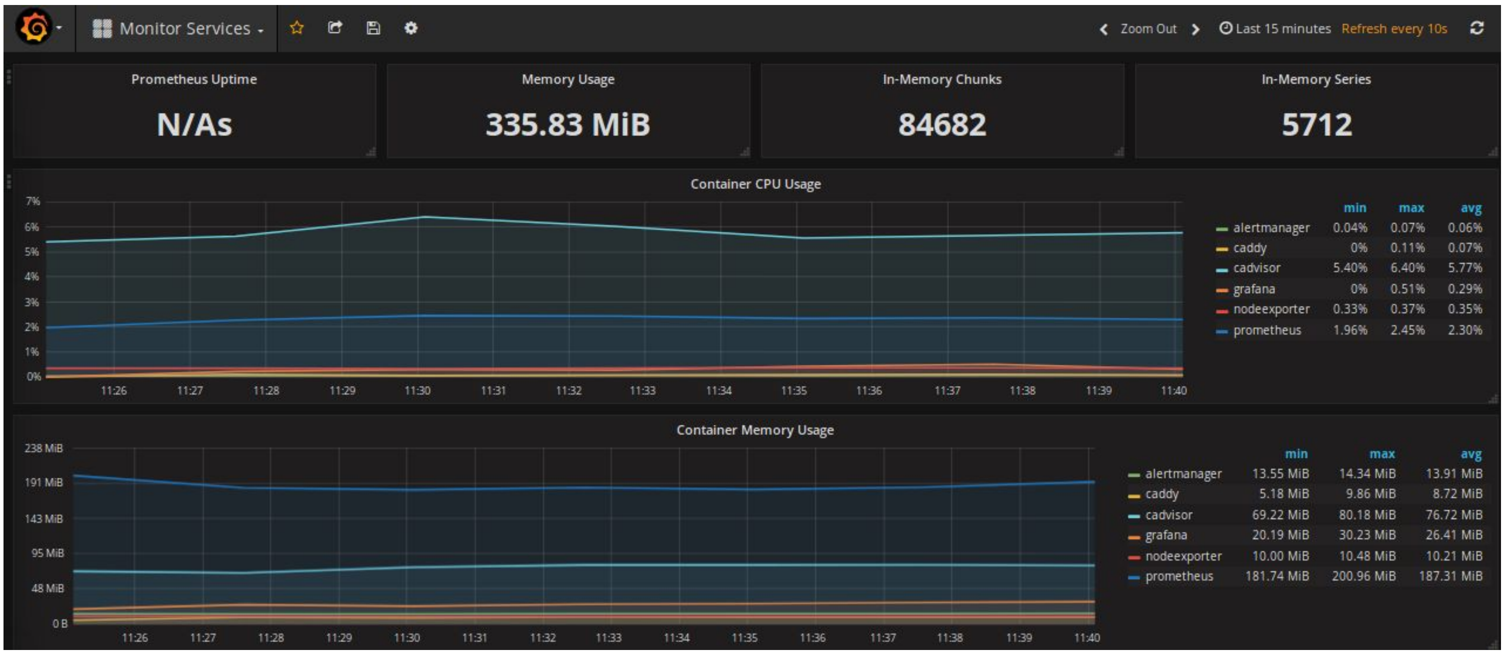
I/O Usage

	min	max	avg
read	0 Bps	2.87 MBps	133.19 kBps
written	0 Bps	1.26 MBps	90.11 kBps
io time (right-y)	0 ms	999.20 ms	67.11 ms

Network Usage

	min	max	avg
In: eth0	91.67 Bps	127.69 Bps	93.22 Bps
Out: eth0	3.39 kBps	4.73 kBps	3.44 kBps

❖ Monitor Service Prometheus



Prometheus Uptime: N/As

Memory Usage: 335.83 MiB

In-Memory Chunks: 84682

In-Memory Series: 5712

Container CPU Usage

	min	max	avg
alertmanager	0.04%	0.07%	0.06%
caddy	0%	0.11%	0.07%
cadvisor	5.40%	6.40%	5.77%
grafana	0%	0.51%	0.29%
nodeexporter	0.33%	0.37%	0.35%
prometheus	1.96%	2.45%	2.30%

Container Memory Usage

	min	max	avg
alertmanager	13.55 MiB	14.34 MiB	13.91 MiB
caddy	5.18 MiB	9.86 MiB	8.72 MiB
cadvisor	69.22 MiB	80.18 MiB	76.72 MiB
grafana	20.19 MiB	30.23 MiB	26.41 MiB
nodeexporter	10.00 MiB	10.48 MiB	10.21 MiB
prometheus	181.74 MiB	200.96 MiB	187.31 MiB