



mastery program

> DevOps

Monitoreo con Prometheus

> Chuy Lerma

empezando con prometheus

- > introducción
- > instalación
- > node_exporter
- > ejercicio
- > alerting



empezando con prometheus

> introducción



¿por qué prometheus?

- > Alta escalabilidad.
- > Alta disponibilidad.
- > Despliegue sencillo.
- > Muchas integraciones existentes.



¿por qué prometheus?

- > Descubrimiento de servicios flexible.
- > Se adapta a código existente.
- > Production-ready.
- > Diseñado para microservicios.



¿por qué prometheus?

- > Documentación extensa.
- > Comunidad Activa.
- > Soporte comercial.



¿qué es prometheus?

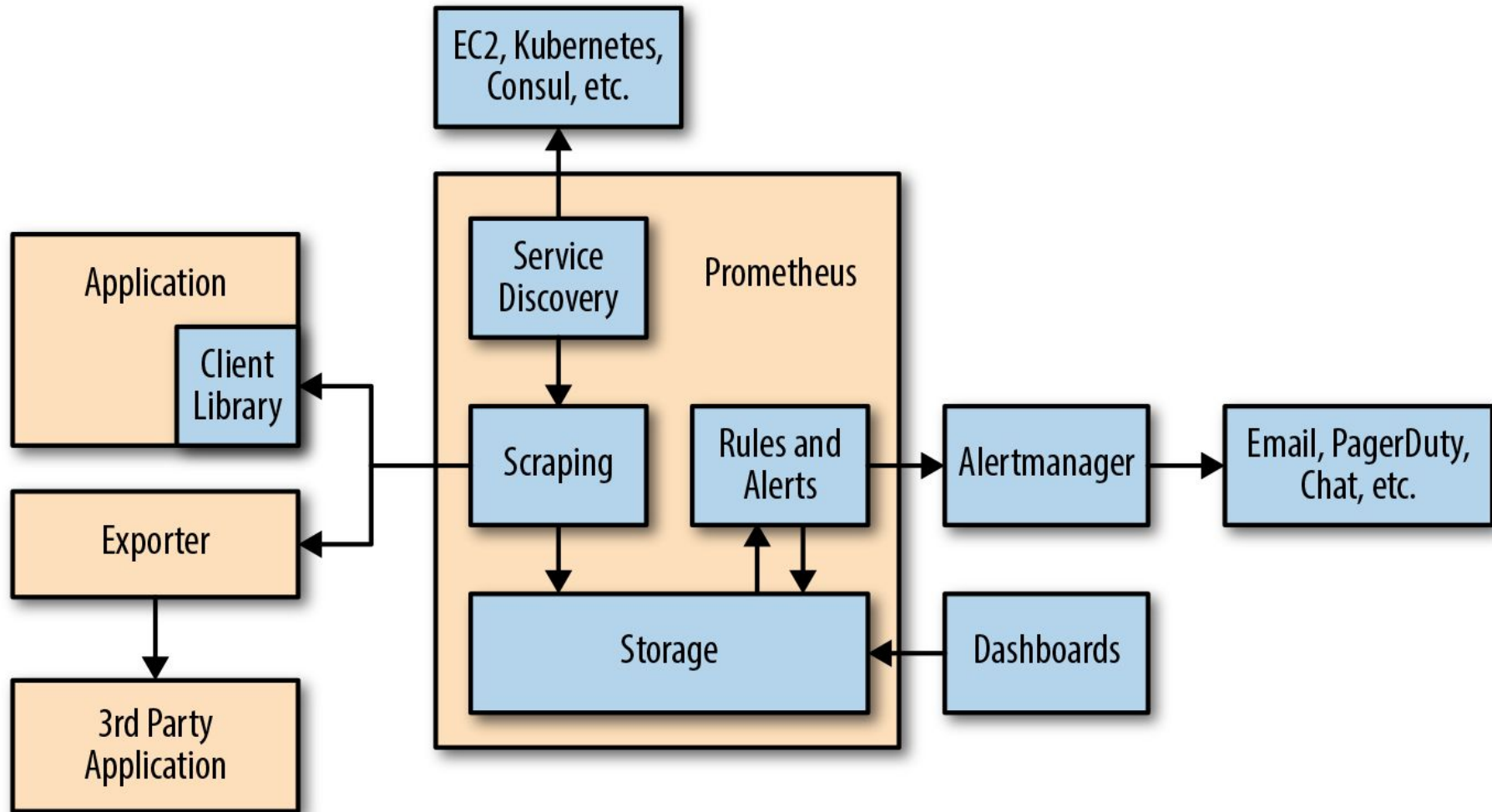
Un sistema de monitoreo de métricas y alertas para sistemas distribuidos e infraestructura.



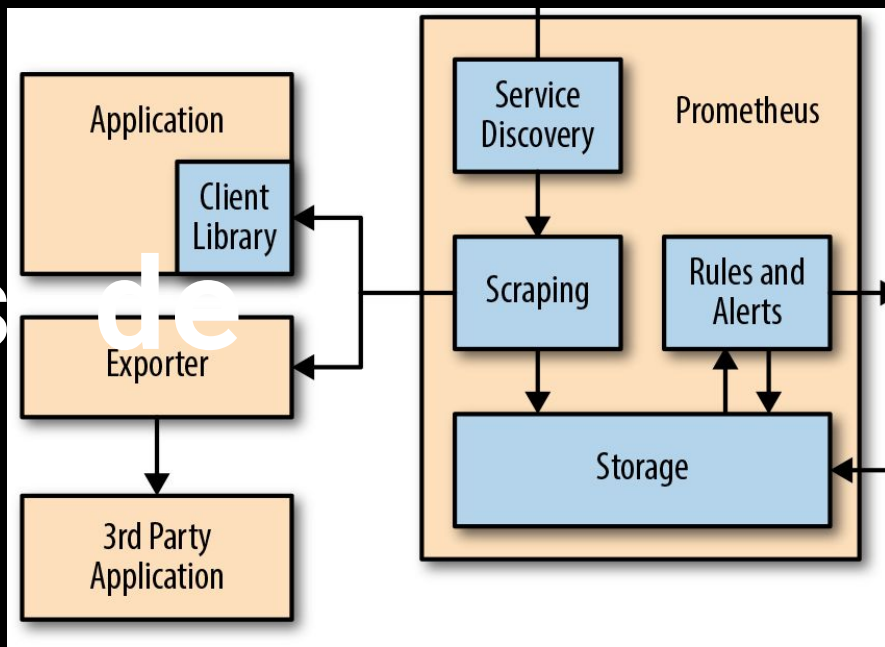
¿qué es prometheus?

Un poderoso modelo de datos y lenguaje de búsqueda que permite analizar cómo se desempeñan tus aplicaciones e infraestructura.





bibliotecas de cliente



Con solo dos o tres líneas de código cualquier aplicación puede añadir métricas a prometheus.



bibliotecas de cliente

A esto se le conoce como
instrumentación directa (direct
instrumentation)



bibliotecas de cliente oficiales

- > Go.
- > Python.
- > Java/JVM.
- > Ruby.

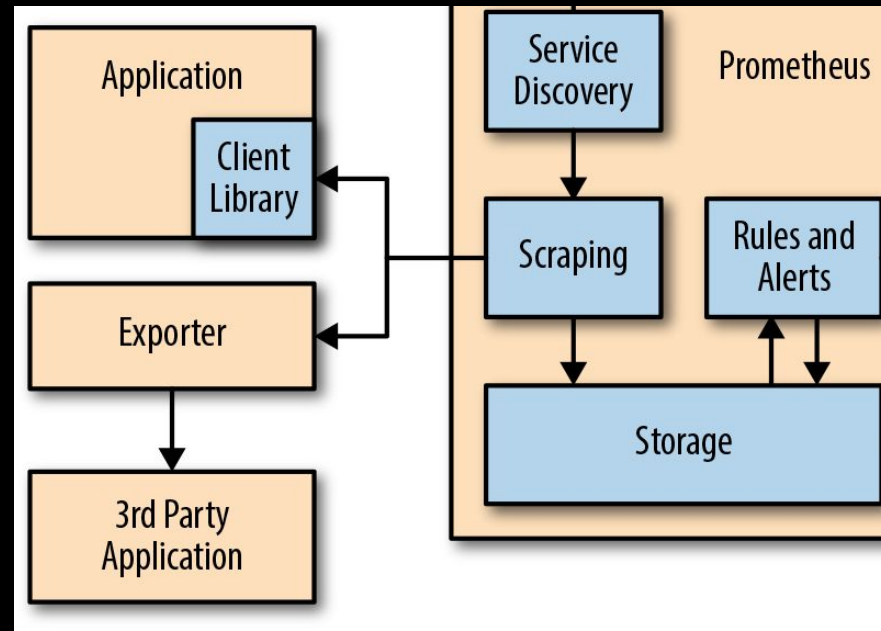


bibliotecas de cliente NO oficiales

- > C#/.Net.
- > Node.js
- > Haskell.
- > Erlang.
- > Rust.




Exporters



Son piezas de código que despliegas justo a lado de la aplicación de la cual necesitas obtener métricas.

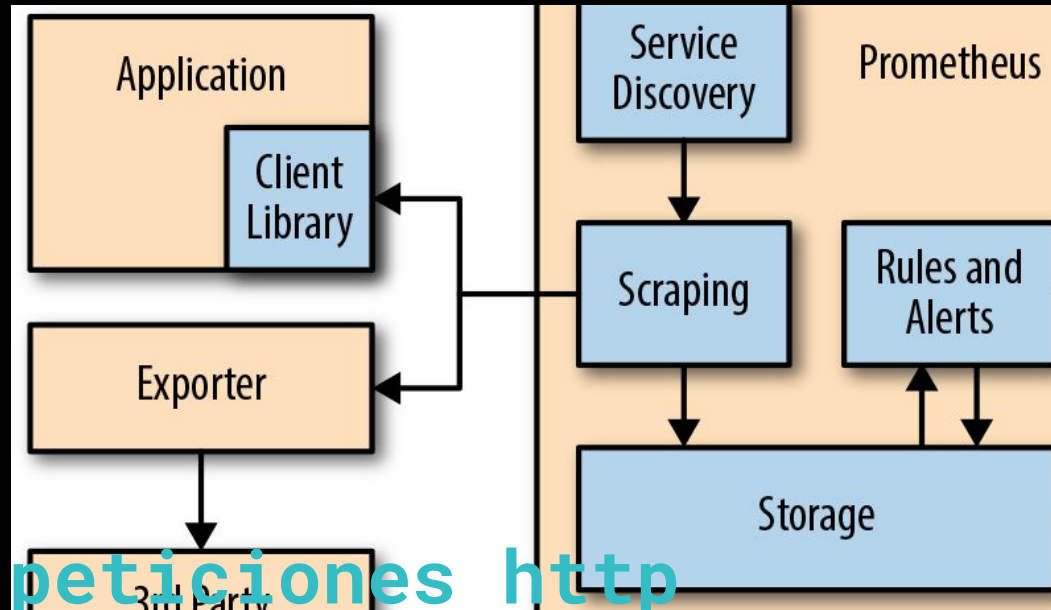
Exporters

Toma peticiones de prometheus, recoge la información requerida, la transforma al formato correcto y lo envía como respuesta a la petición de prometheus.



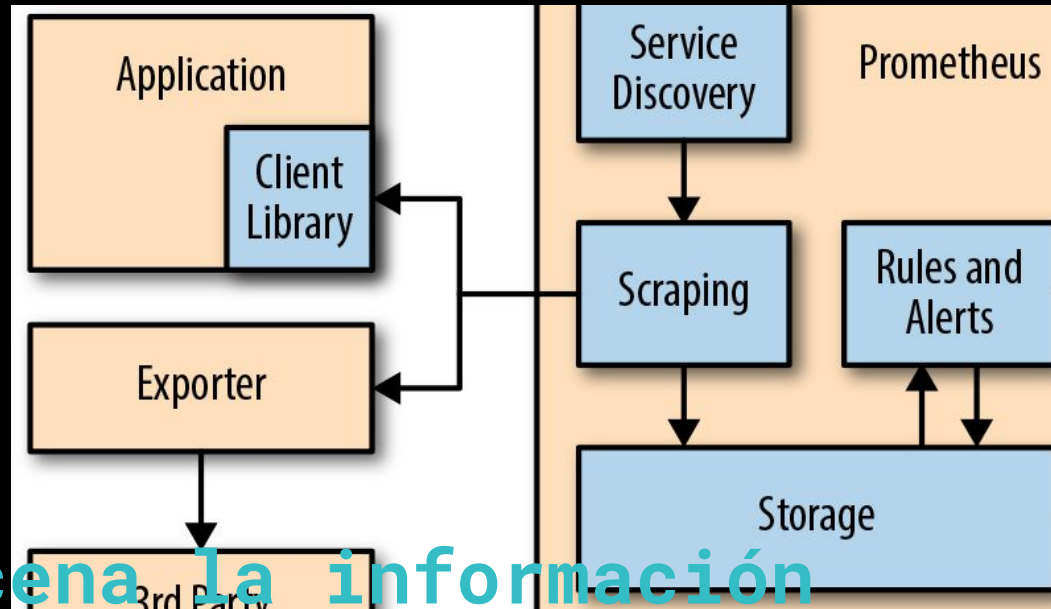
Scraping

Prometheus hace peticiones http nombradas scrape a los exporters cada cierto tiempo. Esta respuesta es analizada y almacenada en storage.



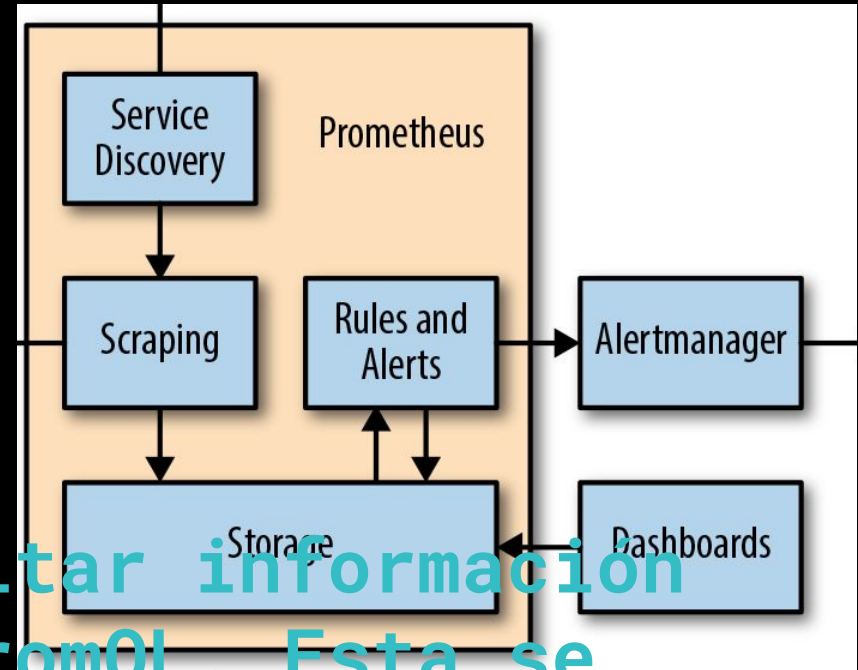
Storage

Prometheus almacena la información de forma local en una base de datos de series de tiempo personalizada.



Dashboards

API Http para solicitar información y evaluar queries PromQL. Esta se utiliza para producir gráficas y dashboards.





Prometheus Benchmark - 2.0.x



Zoom Out



Last 6 hours

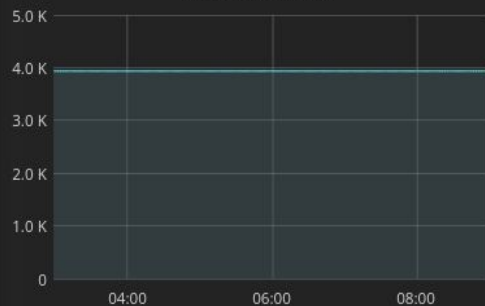
Refresh every 1m



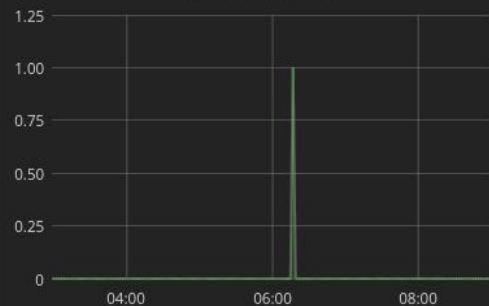
Prometheus

demo.robustperception.io

Head Time series



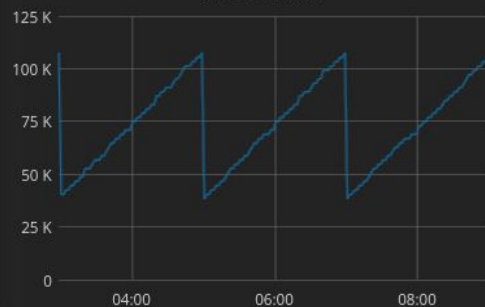
Head Active Appenders



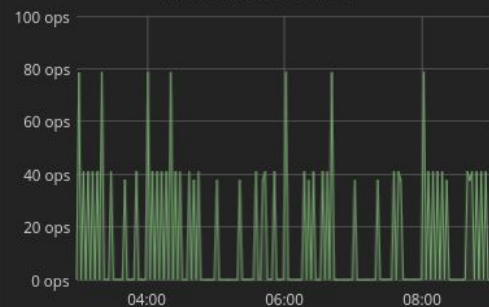
Samples Appended/s



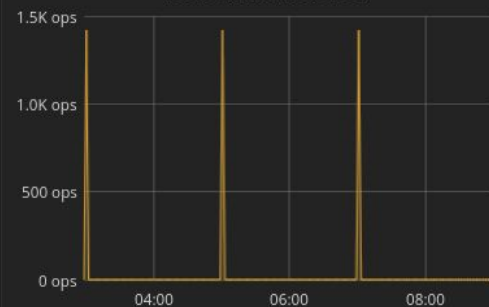
Head Chunks



Head Chunks Created

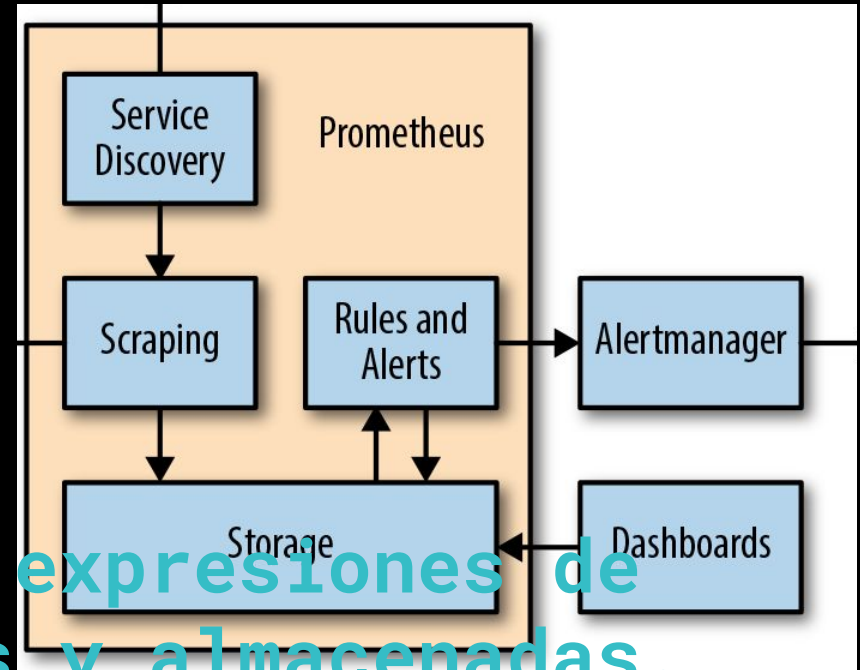


Head Chunks Removed



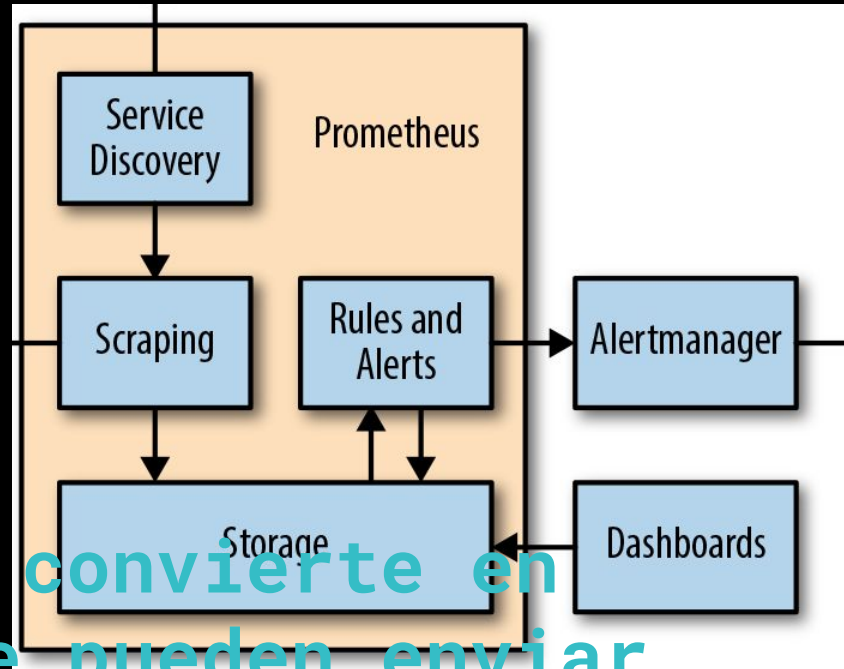
Reglas y Alertas

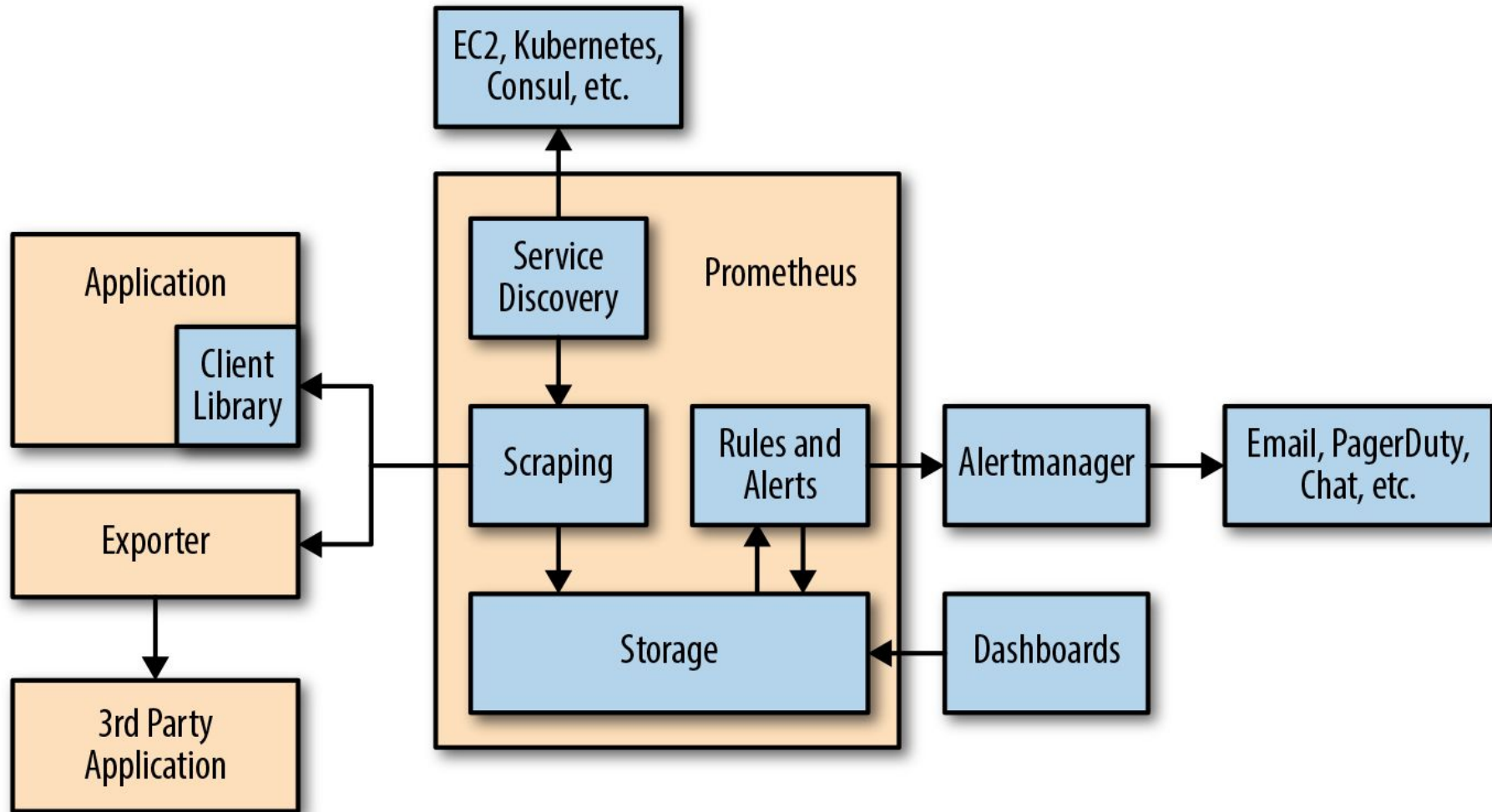
Las reglas permiten expresiones de PromQL ser evaluadas y almacenadas. Las alertas se evalúan mediante reglas y se envían cada que se cumplen.



Alertmanager

Recibe alertas y las convierte en notificaciones que se pueden enviar por email, slack o PagerDuty.





¿qué NO es prometheus?

- > Un sistema de archivos long-term.
- > Un sistema de reportes de inteligencia de negocios.
- > Un backend de minería de datos.



empezando con prometheus

> **instalación**



Machine 1
prometheus
Ip: 10.6.6.1

Machine 2
node_exporter
Ip: 10.6.6.2

Machine 3
alerting
Ip: 10.6.6.3



```
$ mkdir prometheus && cd prometheus  
$ touch Vagrantfile  
$ vagrant up  
$ vagrant ssh
```



```
$ wget
https://github.com/prometheus/prometheus/releases/download/v2.8.0/prometheus-2.8.0.linux-amd64.tar.gz
$ tar -xzf prometheus-2.8.0.linux-amd64.tar.gz
$ cd prometheus-2.8.0.linux-amd64
$ cat << EOF > prometheus.yml
global:
  scrape_interval:10s
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets:
        - localhost:9090
EOF
$ ./prometheus &
```



<https://prometheus.io/download/>

Ir a localhost:9090

[Prometheus](#) [Alerts](#) [Graph](#) [Status ▾](#) [Help](#)

☐ Enable query history

Execute

- insert metric at cursor - ▾

Graph

Console

Element	Value
no data	

[Remove Graph](#)

Add Graph



Ir a localhost:9090

Prometheus Alerts Graph Status ▾ Help

Targets

☐ Only unhealthy jobs

prometheus (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Error
http://localhost:9090/metrics	UP	instance="localhost:9090"	9.896s ago	



Ir a localhost:9090/metrics

```
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.8479e-05
go_gc_duration_seconds{quantile="0.25"} 6.2474e-05
go_gc_duration_seconds{quantile="0.5"} 9.5289e-05
go_gc_duration_seconds{quantile="0.75"} 0.000230219
go_gc_duration_seconds{quantile="1"} 0.000652444
go_gc_duration_seconds_sum 0.002677241
go_gc_duration_seconds_count 17
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 112
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.6763616e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.59820128e+08
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.475242e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 884863
```



empezando con prometheus

> **metricas**



Usamos el Expression Browser

> up

```
up{instance="localhost:9090",job="prometheus"}
```

Instance y Job son labels.

> process_resident_memory_bytes

El valor está en bytes. Los valores que usa prometheus son bytes y seconds.

El valor que se muestra es la cantidad de memoria usada por prometheus.

Podemos seleccionar graph para ver como esta métrica cambia sobre el tiempo.

Al ejecutar en EB nos muestra el valor en ese momento.



Usamos el Expression Browser

El valor que acabas de ver son Gauges. Los Gauges muestran un snapshot de un estado en el momento de ejecución.

Otro tipo de valor son countes:

```
> prometheus_tsdb_head_samples_appended_total
```

Muestra el numero (counter) de ejemplos que prometheus ha añadido.

```
> rate(prometheus_tsdb_head_samples_appended_total[1m])
```

Rate calcula que tan rapido un counte es incrementado por segundo.

Al pasarle el valor 1m mostramos el rate cada minuto.



empezando con prometheus

> `node_exporter`



<https://prometheus.io/download/>

```
$ cd ~
```

```
$ wget
```

```
https://github.com/prometheus/node\_exporter/releases/download/v0.17.0/node\_exporter-0.17.0.linux-amd64.tar.gz
```

```
$ tar -xzf node_exporter-0.17.0.linux-amd64.tar.gz
```

```
$ cd node_exporter-0.17.0.linux-amd64
```

```
$ ./node_exporter &
```

Machine 2
node_exporter
ip: 10.0.0.2



```
$ cd ~/prometheus/prometheus-2.8.0.linux-amd64
$ cat << EOF > prometheus.yml
global:
  scrape_interval:10s
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets:
        - localhost:9090
  - job_name: node
    static_configs:
      - targets:
        - localhost:9100
EOF
$ ./prometheus &
```



<https://prometheus.io/download/>

empezando con prometheus

> `metricas node_exporter`



Usamos el Expression Browser

```
> up
```

Ahora vemos corriendo dos procesos. Si matamos el proceso de `node_explorer` podemos ver que deja de mostrarse.

```
> process_resident_memory_bytes{job="node"}
```

El `job=node` está haciendo un match con el label "node" y restringe las metricas que veremos.

```
> rate(node_network_receive_bytes_total[1m])
```

Ahora veremos el rate de el uso de red que el servidor recibe el último minuto.



Usamos el Expression Browser

```
> rate(node_cpu_seconds_total{mode="system"}[1m])
```

Ahora vemos la cantidad promedio de CPU usado por segundo en el último minuto.

```
> node_filesystem_avail_bytes
```

La cantidad de espacio disponible en usuarios no root.

```
> rate(node_network_receive_bytes_total[1m])
```

Ahora veremos el rate de el uso de red que el servidor recibe por minuto.

```
> node_filesystem_size_bytes
```

```
> node_filesystem_avail_bytes /  
node_filesystem_size_bytes
```

Calcula la cantidad usada del disco.



Usamos el Expression Browser

> node_uname_info



Ejercicio 1

- > Crea dos máquinas virtuales con vagrant e instala node_exporter.
- > Agrega ambas maquinas como targets de prometheus.



empezando con prometheus

> alerting



Alertmanager



```
$ wget
```

```
https://github.com/prometheus/alertmanager/releases/download/v0.16.1/alertmanager-0.16.1.linux-amd64.tar.gz
```

```
$ tar -xzf alertmanager-0.16.1.linux-amd64.tar.gz
```

```
$ cd alertmanager-0.16.1.linux-amd64
```

generamos un app password

```
https://support.google.com/accounts/answer/185833?hl=en
```

Machine 3
alerting
Ip: 10.0.0.3



<https://prometheus.io/download/>

```
$ cat << EOF > alertmanager.yml
route:
  group_by: [Alertname]
  receiver: email-me

receivers:
- name: email-me
  email_configs:
  - to: $GMAIL_ACCOUNT
    from: $GMAIL_ACCOUNT
    smarthost: smtp.gmail.com:587
    auth_username: "$GMAIL_ACCOUNT"
    auth_identity: "$GMAIL_ACCOUNT"
    auth_password: "$GMAIL_AUTH_TOKEN"
EOF
```



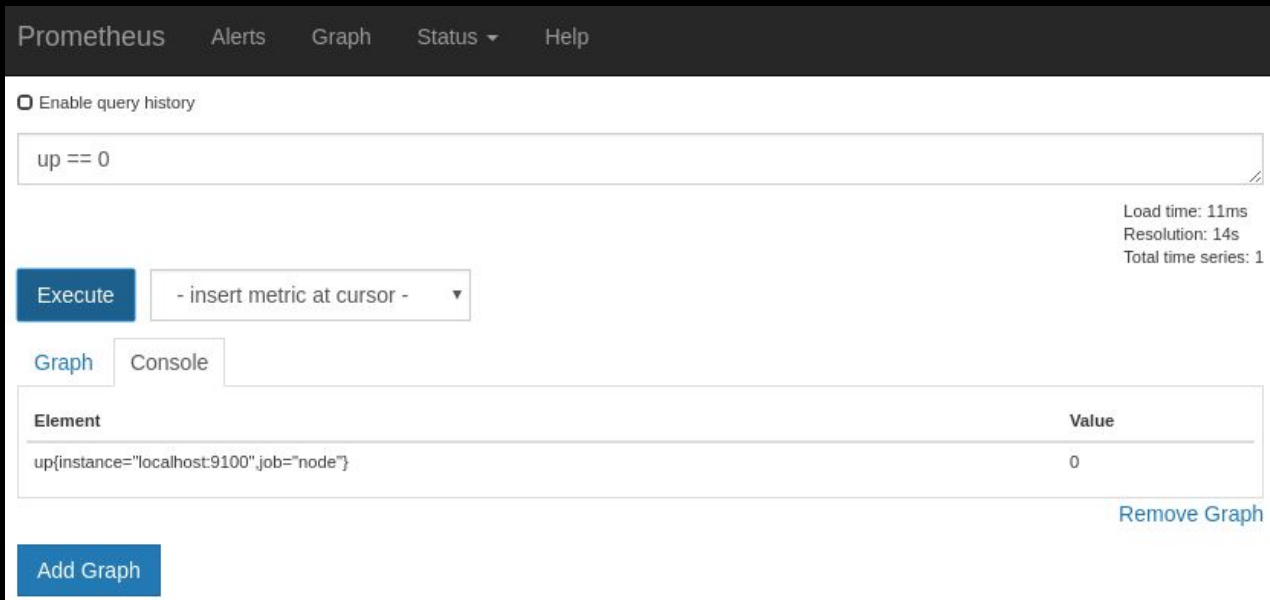
```
$ GMAIL_ACCOUNT=me@example.com  
$ GMAIL_AUTH_TOKEN=XXXX  
$ ./alertmanager &
```



<https://prometheus.io/download/>


```
$ ps ax | grep node_exporter
$ kill -9
```

Ahora debemos de ir a localhost:9090 y escribir en la consola: `up == 0`



The screenshot shows the Prometheus web interface. At the top is a navigation bar with links for Prometheus, Alerts, Graph, Status, and Help. Below this is a section with a checkbox for 'Enable query history' and a text input field containing the query 'up == 0'. To the right of the input field, performance metrics are displayed: 'Load time: 11ms', 'Resolution: 14s', and 'Total time series: 1'. Below the input field is a blue 'Execute' button and a dropdown menu showing '- insert metric at cursor -'. Underneath are two tabs: 'Graph' (selected) and 'Console'. The 'Graph' tab displays a table with two columns: 'Element' and 'Value'. The table contains one row with the element 'up{instance="localhost:9100",job="node"}' and the value '0'. At the bottom left is a blue 'Add Graph' button, and at the bottom right is a blue 'Remove Graph' link.

Element	Value
up{instance="localhost:9100",job="node"}	0

```
$ cd ~/prometheus/prometheus-2.8.0.linux-amd64
$ cat << EOF > prometheus.yml
global:
  scrape_interval:10s
  evaluation_interval: 10s
rule_files: ["rules.yml"]
alerting:
  alertmanagers:
    - static_configs:
      - targets:
        - localhost:9093
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets:
        - localhost:9090
EOF
```



<https://prometheus.io/download/>

```
$ cd ~/prometheus/prometheus-2.8.0.linux-amd64
```

```
$ cat << EOF > rules.yml
```

```
groups:
```

```
  - name: example
```

```
    rules:
```

```
      - alert: InstanceDown
```

```
        expr: up == 0
```

```
        for: 1m
```

```
EOF
```



<https://prometheus.io/download/>

```
$ cd ~/prometheus/prometheus-2.8.0.linux-amd64
```

```
$ cat << EOF > rules.yml
```

```
groups:
```

```
  - name: example
```

```
    rules:
```

```
      - alert: InstanceDown
```

```
        expr: up == 0
```

```
        for: 1m
```

```
EOF
```



<https://prometheus.io/download/>