



KodeKloud

What is Observability

What is Observability

Observability – The ability to understand and measure the state of a system based upon data generated by the system

Observability allows you to generate actionable outputs from **unexpected** scenarios in dynamic environments

Observability will help:

1. Give better insight into the internal workings of a system/application
2. Speed up troubleshooting
3. Detect hard to catch problems
4. Monitor performance of an application
5. Improve cross-team collaboration

Observability

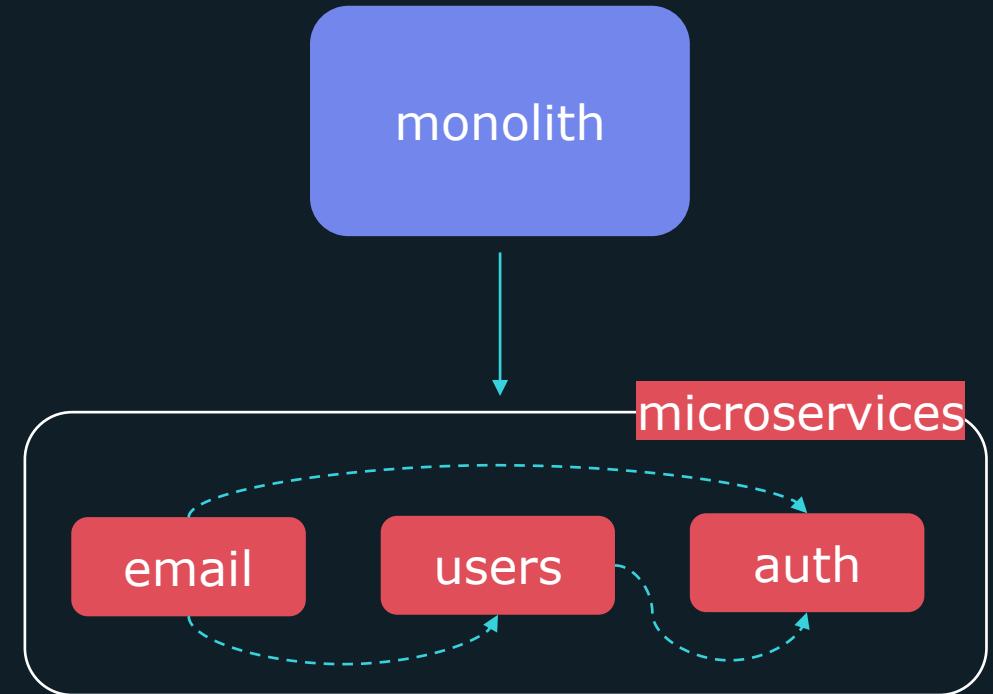
The main purpose of **observability** it to better understand the internals of your system

System/Application

Observability

As system architectures continue to get more and more complex, new challenges arise as tracking down issues become far more challenging

There's a greater need for **observability** as we move towards distributed systems & microservices based application



When it comes to troubleshooting issues, we need more information than just what is wrong.

We need to know why our application entered a specific state, what component is responsible and how we can avoid it in the future

- Why are error rates rising
- Why is there high latency
- Why are services timing out



Observability gives you the flexibility to understand unpredictable events

3 pillars of Observability

How do we accomplish observability?



Logging



Metrics



Traces

Logging

Logs are records of events that have occurred and encapsulate information about the specific event

Logs are comprised of:

- Timestamp of when the log occurred
- Message containing information

```
Oct 26 19:35:00 ub1 kernel: [37510.942568] e1000: enp0s3  
NIC Link is Down
```

```
Oct 26 19:35:00 ub1 kernel: [37510.942697] e1000  
0000:00:03.0 enp0s3: Reset adapter
```

```
Oct 26 19:35:03 ub1 kernel: [37513.054072] e1000: enp0s3  
NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
```

Logs are the most **common** form of observation produced by systems

However, they can be difficult to use due to the verbosity of the logs outputted by systems/applications

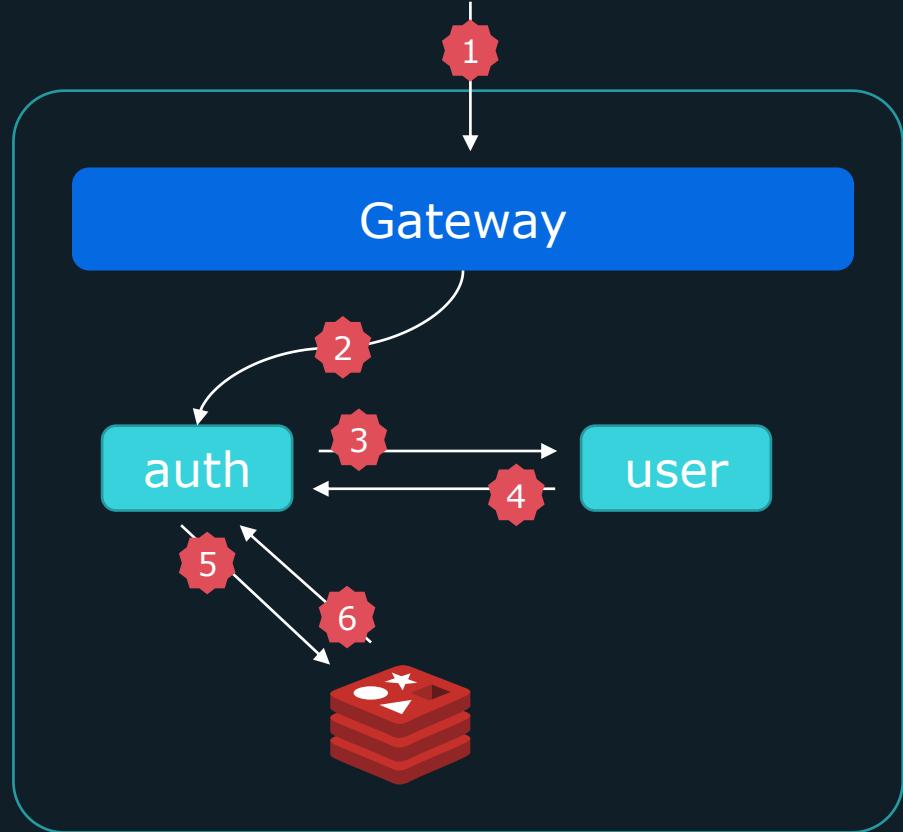
Logs of processes are likely to be interwoven with other concurrent processes spread across multiple systems

Traces

Traces – allow you to follow operations as they traverse through various systems & services

So we can follow an individual request and see it flow through our system hop by hop

Traces help us connect the dots on how processes and services work together



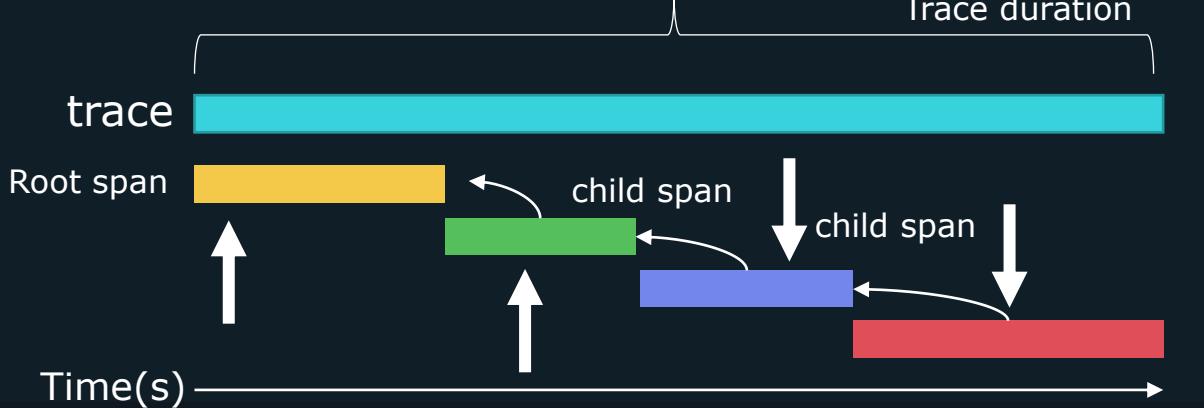
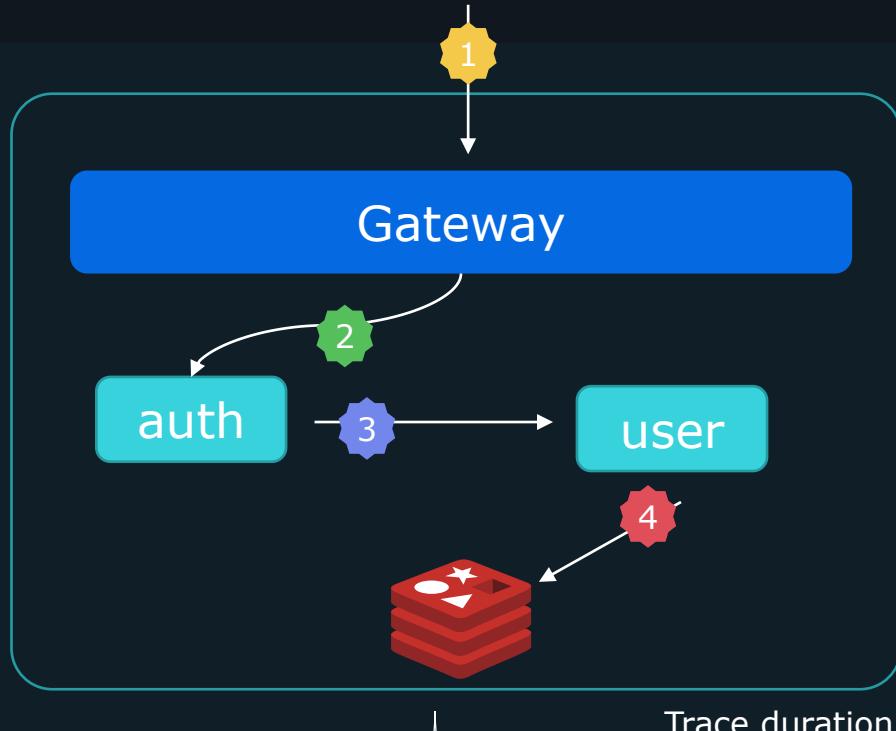
Traces

Each **trace** has a trace-id that can be used to identify a request as it traverses the system

Individual events forming a trace are called **spans**

Each span tracks the following:

- Start time
- Duration
- Parent-id



Metrics provide information about the state of a system using numerical values

- CPU Load
- Number of open files
- HTTP response times
- Number of errors

The data collected can be aggregated over time and graphed using visualization tools to identify trends over time



Metrics

Metrics contain 4 pieces of information:

1. Metric name
2. Value – most recent or current value of the metric
3. Timestamp for the metric
4. Dimensions – additional information about the metric

Metric name	Dimensions	Value	Timestamp
node_filesystem_avail_bytes{fstype="vfat", mountpoint="/home"}		5000	4:30AM 12/1/22

Prometheus



Logs

Metrics

Traces

Prometheus is a monitoring solution that is responsible for collecting and aggregating **metrics**



KodeKloud

SLI/SLO/SLA

When designing a system or applications, it's important for teams to set specific measurable targets/goals to help organizations strike the right balance between product development and operation work.

These targets help customers & end users quantify the level of reliability they should come to expect from a service

"Application should have 97% uptime in a rolling 30 day window"

Service Level Indicator(SLI) – quantitative measure of some aspect of the level of service that is provided

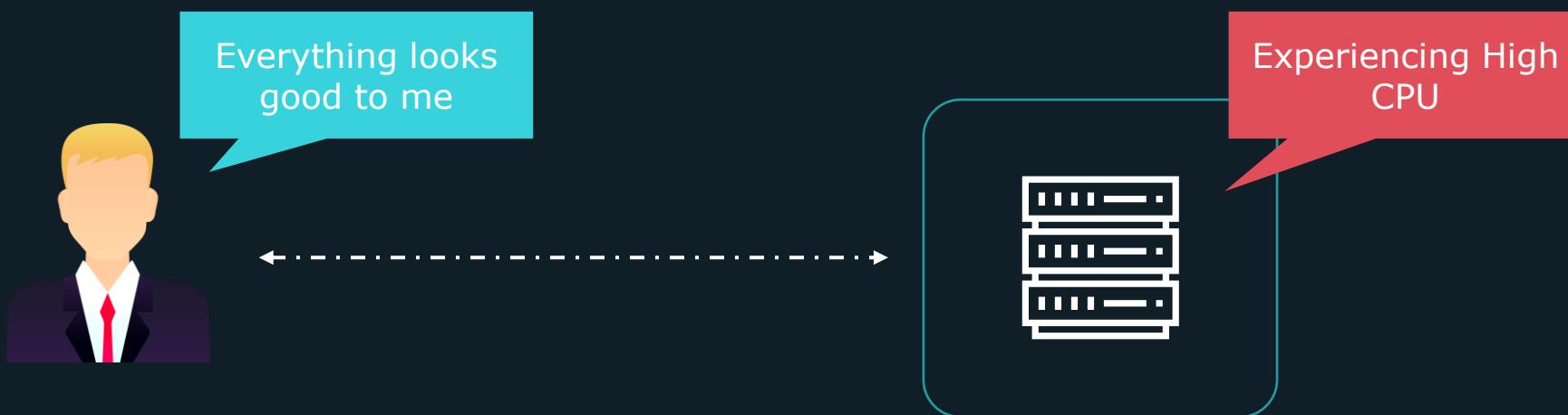
Common SLIs:

- Request Latency
- Error Rate
- Saturation
- Throughput
- Availability

Service Level indicator

Not all metrics make for good SLIs. You want to find metrics that accurately measure a user's experience.

Things like high-cpu/high-memory make for a poor SLI as a user might not see any impact on their end during these events



Service Level Object

Service Level Object(SLO) – target value or range for an SLI

SLI - Latency

SLO - Latency < 100ms

SLI - availability

SLO - 99.9% uptime

SLOs should be directly related to the **customer experience**. The main purpose of the SLO is to quantify reliability of a product to a customer

For SLOs it maybe tempting to set them to aggressive values like 100% uptime however this will come at a higher cost

The goal is not to achieve perfection but instead to make customers happy with the right level of reliability

If a customer is happy with 99% reliability increasing it any further doesn't add any other value

Service Level Agreement(SLA) – contract between a vendor and a user that guarantees a certain SLO



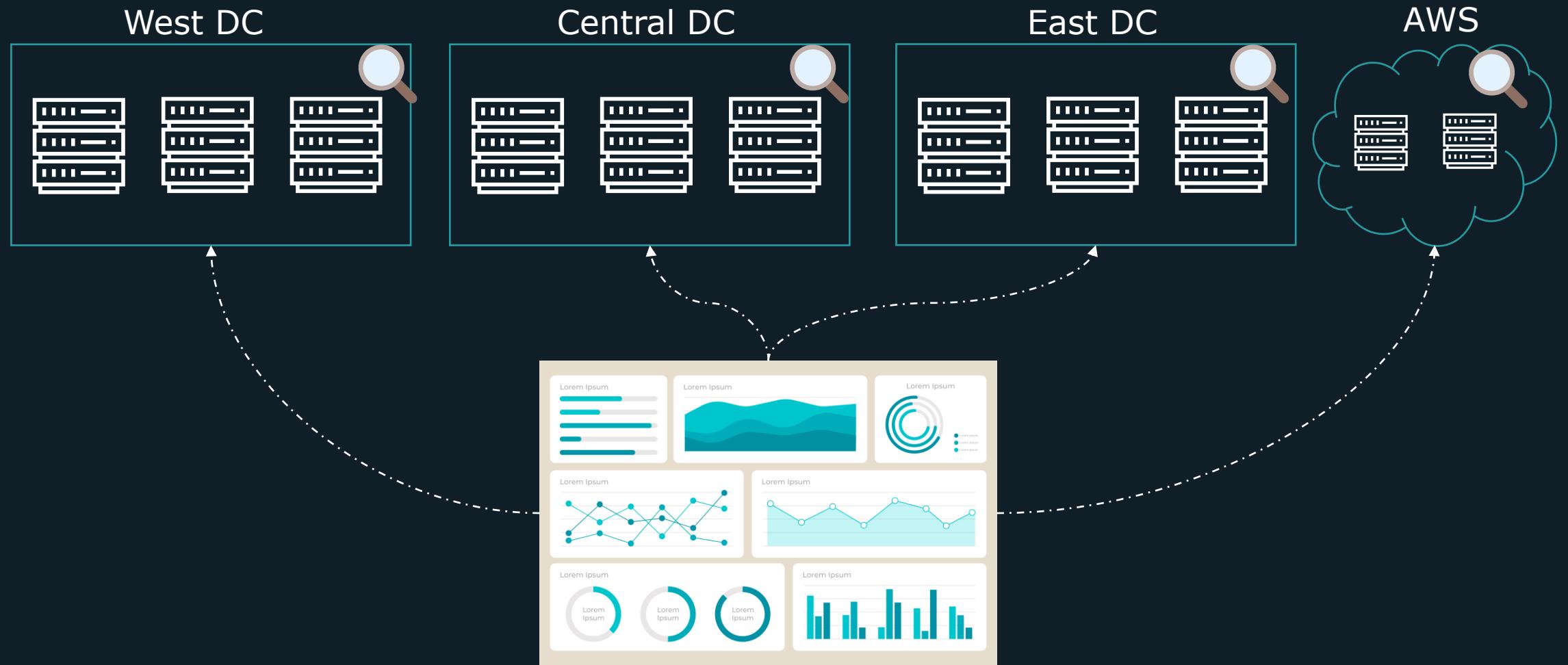
The consequences for not meeting any SLO can be financial based but can also be variety of other things as well



KodeKloud

Prometheus Use Cases

Use Case #1



Usecase #2

Several outages have occurred due to high memory on the server hosting a MySql database.

Operations team would like to be notified through email when the memory gets to 80% max capacity so that proactive measure can be taken



Usecase #3

A new video upload feature has been added to the website, however there is some concerns about users uploading too large of videos.

The team would like to find out at which video length the applications starts to degrade and to do this, they need a chart plotting the avg file size of upload and the average latency per request





KodeKloud

Prometheus Basics

What is Prometheus

Prometheus is an open-source monitoring tool that collects metrics data, and provide tools to visualize the collected data

In addition, Prometheus allows you to generate alerts when metrics reach a user specified threshold

Prometheus collects metrics by scraping targets who expose metrics through an HTTP endpoint

Scraped metrics are then stored in a time series database which can be queried using Prometheus' built-in query language PromQL

So what kind of metrics can Prometheus Monitor?

- CPU/Memory Utilization
- Disk space
- Service Uptime
- Application specific data
 - Number of exceptions
 - Latency
 - Pending Requests

Prometheus is designed to monitor time-series data
that is **numeric**

What type of data should Prometheus **not** monitor

- Events
- System logs
- Traces

Prometheus Background

Prometheus was originally sponsored by SoundCloud but in 2016 it joined the Cloud Native Computing Foundation



Prometheus is primarily written GoLang



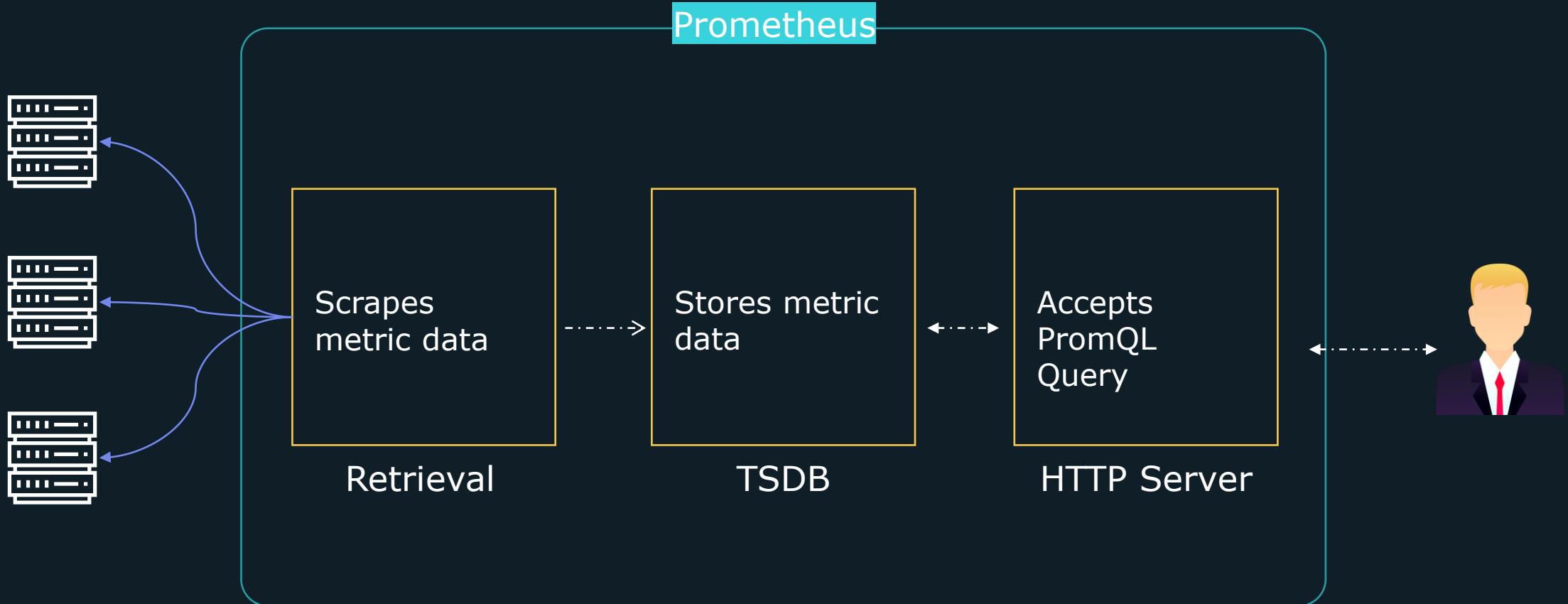
Documentation for Prometheus can be found at prometheus.io/docs



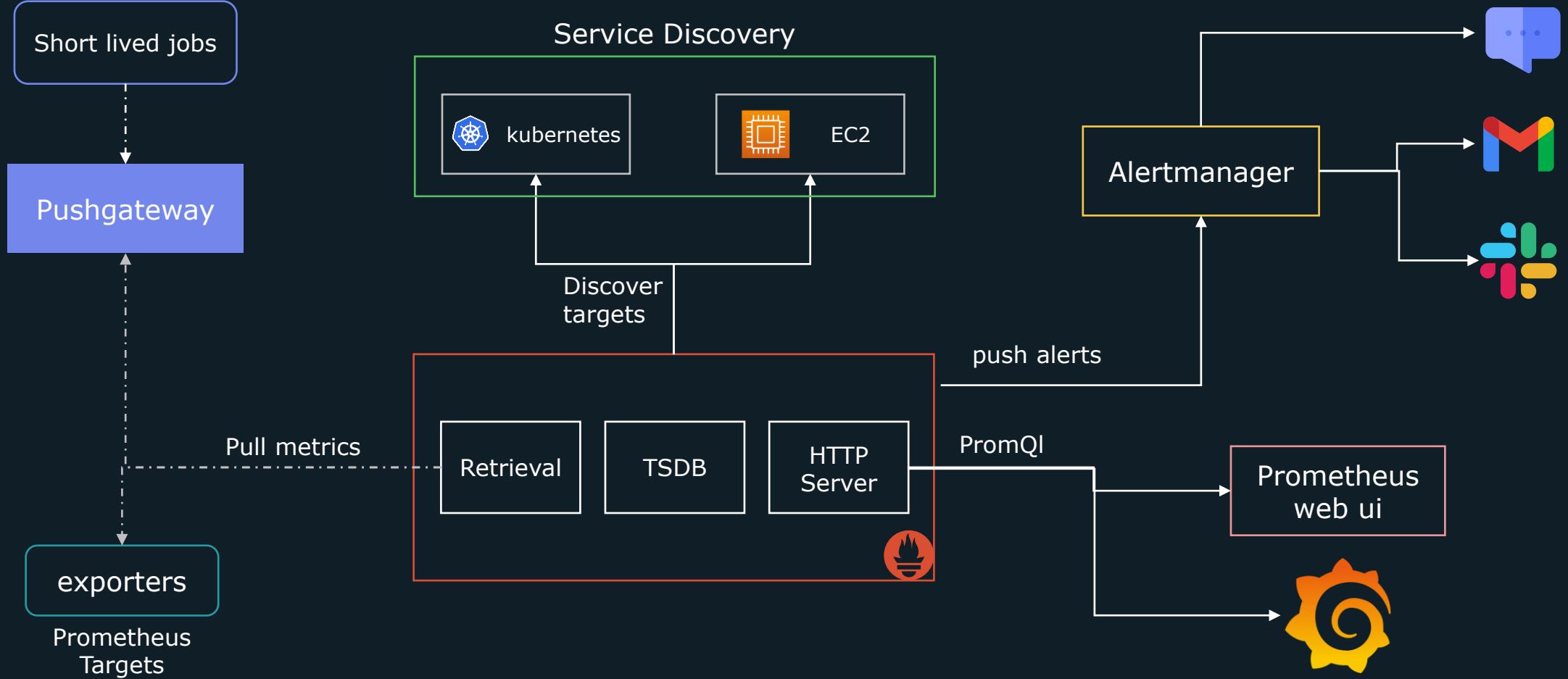
KodeKloud

Prometheus Architecture

Prometheus Architecture



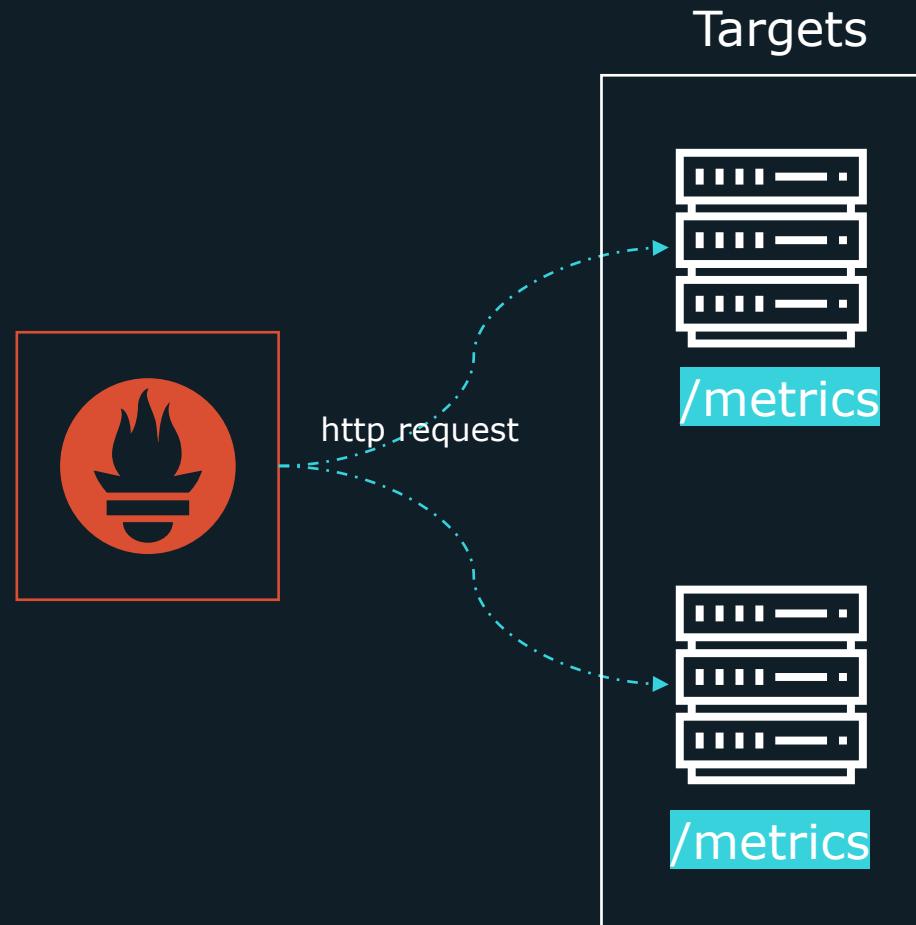
Prometheus Architecture



Collecting Metrics

Prometheus collects metrics by sending http requests to `/metrics` endpoint of each target

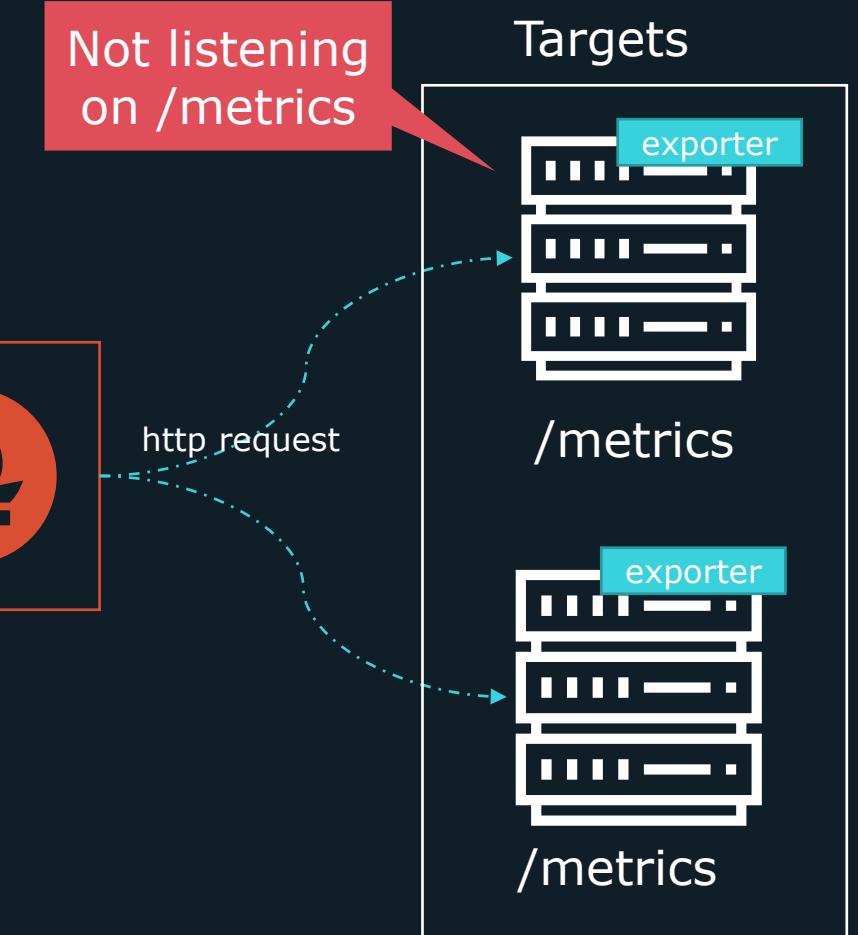
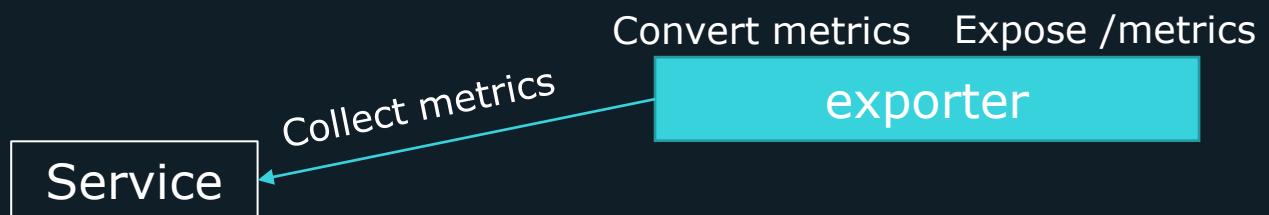
Prometheus can be configured to use a different path other than `/metrics`



Exporters

Most systems by default don't collect metrics and expose them on an HTTP endpoint to be consumed by a Prometheus server

Exporters collect metrics and expose them in a format Prometheus expects



Exporters

Prometheus has several **native** exporters

- Node exporters(Linux servers)
- Windows
- MySQL
- Apache
- HAProxy

EXPORTERS AND INTEGRATIONS

There are a number of libraries and servers which help in exporting existing metrics from third-party systems as Prometheus metrics. This is useful for cases where it is not feasible to instrument a given system with Prometheus metrics directly (for example, HAProxy or Linux system stats).

Third-party exporters

Some of these exporters are maintained as part of the official [Prometheus GitHub organization](#), those are marked as *official*, others are externally contributed and maintained.

We encourage the creation of more exporters but cannot yet vet all of them for [best practices](#). Commonly, those exporters are hosted outside of the Prometheus GitHub organization.

The [exporter default port](#) wiki page has become another catalog of exporters, and may include exporters not listed here due to overlapping functionality or still being in development.

The [JMX exporter](#) can export from a wide variety of JVM-based applications, for example [Kafka](#) and [Cassandra](#).

Databases

- Aerospike exporter
- ClickHouse exporter
- Consul exporter ([official](#))
- Couchbase exporter
- CouchDB exporter
- Druid Exporter
- Elasticsearch exporter
- EventStore exporter

- Third-party exporters
 - Databases
 - Hardware related
 - Issue trackers and continuous integration
 - Messaging systems
 - Storage
 - HTTP
 - APIs
 - Logging
 - Other monitoring systems
 - Miscellaneous
- Software exposing Prometheus metrics
- Other third-party utilities

Can we monitor application metrics

- Number of errors/exceptions
- Latency of requests
- Job execution time

Prometheus comes with **client libraries** that allow you to expose any application metrics you need Prometheus to track

Language support:

- Go
- Java
- Python
- Ruby
- Rust

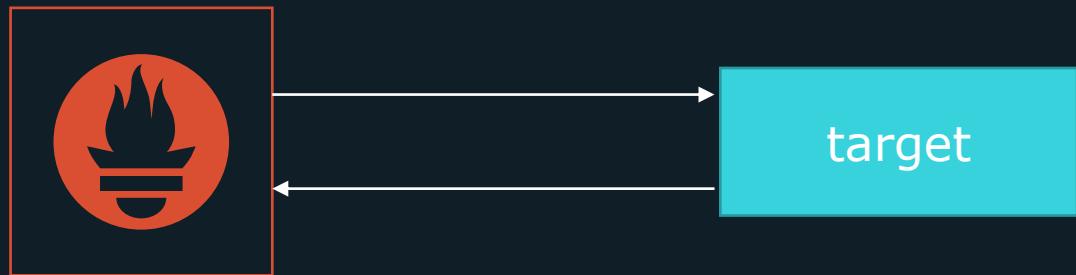


KodeKloud

Push Vs Pull

Pull Based Model

Prometheus follows a pull based model

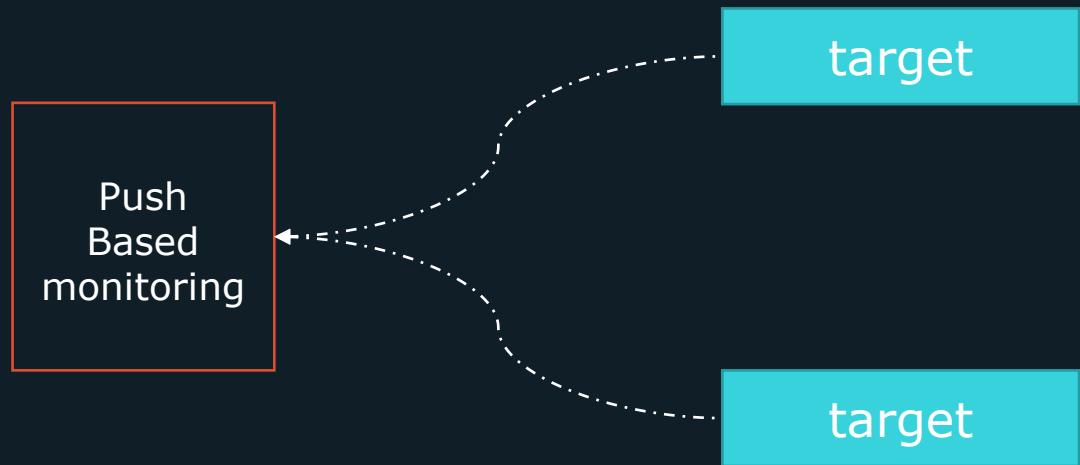


Prometheus needs to have a list of all targets it should scrape

Other Pull based monitoring solutions include:

- Zabbix
- nagios

Push Based Model



In a pushed based model, the targets are configured to push the metric data to the metrics server

Pushed based systems include:

- Logstash
- Graphite
- OpenTSDB

Why Pull?

Some of the benefits of using a pull based system:

- Easier to tell if a target is **down**
 - In a pushed based system we don't know if its down or has been decommissioned
- Push based systems could potentially **overload** metrics server if too many incoming connections get flooded at same time
- Pull based systems have a definitive list of targets to monitor, creating a central source of truth

Why Push?

Push based monitoring excels in the following areas:

- event-based systems, pulling data wouldn't be a viable option
 - However, Prometheus is for collecting metrics and not monitoring events
- Short lived jobs, as they may end before a pull can occur
 - Prometheus has feature called Pushgateway to handle this situation



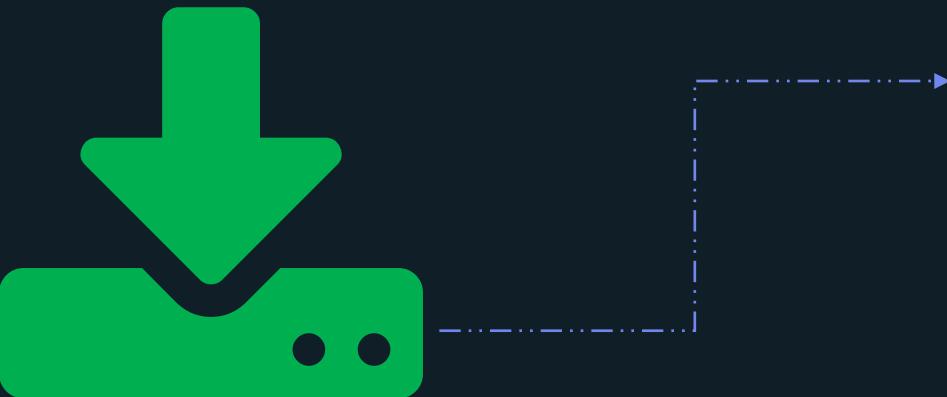
KodeKloud

Prometheus Installation

Installation Bare-Metal/VM

<http://prometheus.io/download>

Get URL



The screenshot shows the Prometheus download page. At the top, there's a navigation bar with links for Prometheus, Docs, Download, Community, Support & Training, and Blog. Below the navigation, there's a section titled "DOWNLOAD" which lists precompiled binaries and Docker images for most officially maintained Prometheus components. A sidebar on the right lists various exporters and integrations. The main content area shows two tables of files for the "prometheus" component, one for version 2.38.0 and one for 2.37.0. A context menu is open over the "alertmanager" link in the second table, with options like "Save link as...", "Copy link address", and "Inspect".

File name	OS	Arch	Size	SHA256 Check
prometheus-2.38.0.darwin-amd64.tar.gz	darwin	amd64	80.40 MiB	9e353da091f01a0c9...
prometheus-2.38.0.linux-amd64.tar.gz	linux	amd64	80.27 MiB	458951e6ef7c28949...
prometheus-2.38.0.windows-amd64.zip	windows	amd64	81.85 MiB	8c21a0a01051a010...

File name	OS	Arch	Size	SHA256 Check
prometheus-2.37.0.darwin-amd64.tar.gz	darwin	amd64	80.01 MiB	ba7900a22e5888622...
prometheus-2.37.0.linux-amd64.tar.gz	linux	amd64	79.90 MiB	c39f5a2a2e1540a9...
prometheus-2.37.0.windows-amd64.zip	windows	amd64	81.47 MiB	0fa21a0c01051a01...

Installation Bare-Metal/VM

>_

```
$ wget https://github.com/prometheus/prometheus/releases/download/v2.37.0/prometheus-2.37.0.linux-amd64.tar.gz  
saving to: 'prometheus-2.37.0.linux-amd64.tar.gz'
```

```
prometheus-2.37.0.linux- 100%[=====>] 79.90M  
36.8MB/s   in 2.2s
```

```
2022-09-01 22:54:38 (36.8 MB/s) - 'prometheus-2.37.0.linux-amd64.tar.gz' saved  
[83782323/83782323]
```

```
FINISHED --2022-09-01 22:54:38--
```

```
Total wall clock time: 2.9s
```

```
Downloaded: 1 files, 80M in 2.2s (36.8 MB/s)
```

Installation Bare-Metal/VM

>_

```
$ ls -l
total 173344
-rwxrwxrwx 1 user1 user1      1112 Sep  1 00:42 ca.crt
drwxr-xr-x 2 user1 user1     4096 Aug 16 11:58 Desktop
drwxr-xr-x 3 user1 user1     4096 Aug 19 01:28 Documents
drwxr-xr-x 4 user1 user1     4096 Aug 23 20:25 Downloads
drwxr-xr-x 2 user1 user1     4096 Aug 16 11:58 Music
drwxr-xr-x 2 user1 user1     4096 Aug 16 11:58 Pictures
-rw-rw-r-- 1 user1 user1 83782323 Jul 14 11:38 prometheus-2.37.0.linux-amd64.tar.gz
drwxr-xr-x 2 user1 user1     4096 Aug 16 11:58 Templates
drwxr-xr-x 2 user1 user1     4096 Aug 16 11:58 Videos
```



Installation Bare-Metal/VM

>_

```
$ tar xvf prometheus-2.37.0.linux-amd64.tar.gz
```

```
user1@user1:~$ tar xvf prometheus-2.37.0.linux-amd64.tar.gz
```

```
prometheus-2.37.0.linux-amd64/
```

```
prometheus-2.37.0.linux-amd64/consoles/
```

```
prometheus-2.37.0.linux-amd64/consoles/index.html.example
```

```
prometheus-2.37.0.linux-amd64/consoles/node-cpu.html
```

```
$ ls -l
```

```
-rwxrwxrwx 1 user1 user1 1112 Sep 1 00:42 ca.crt
```

```
drwxr-xr-x 2 user1 user1 4096 Aug 16 11:58 Desktop
```

```
drwxr-xr-x 3 user1 user1 4096 Aug 19 01:28 Documents
```

```
drwxr-xr-x 4 user1 user1 4096 Jul 14 11:35 prometheus-2.37.0.linux-amd64
```

```
-rw-rw-r-- 1 user1 user1 83782323 Jul 14 11:38 prometheus-2.37.0.linux-amd64.tar.gz
```



Untarred Directory

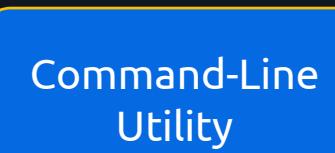
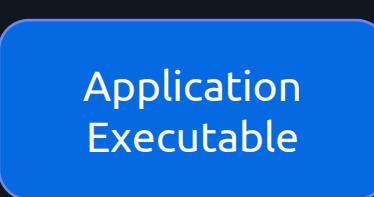
Installation Bare-Metal/VM

>_

```
$ cd prometheus-2.37.0.linux-amd64/
```

```
$ ls -l
```

```
total 206216
drwxr-xr-x 2 user1 user1      4096 Jul 14 11:30 console_libraries
drwxr-xr-x 2 user1 user1      4096 Jul 14 11:30 consoles
-rw-r--r-- 1 user1 user1    11357 Jul 14 11:30 LICENSE
-rw-r--r-- 1 user1 user1     3773 Jul 14 11:30 NOTICE
-rwxr-xr-x 1 user1 user1 109655433 Jul 14 11:16 prometheus
-rw-r--r-- 1 user1 user1      934 Jul 14 11:30 prometheus.yml
-rwxr-xr-x 1 user1 user1 101469395 Jul 14 11:18 promtool
```



Installation Bare-Metal/VM

>_

```
$ ./prometheus
ts=2022-09-02T03:25:25.910Z caller=main.go:996 level=info msg="TSDB started"

ts=2022-09-02T03:25:25.910Z caller=main.go:1177 level=info msg="Loading configuration file" filename=prometheus.yml

ts=2022-09-02T03:25:25.910Z caller=main.go:1214 level=info msg="Completed loading of configuration file"
filename=prometheus.yml totalDuration=308.821µs db_storage=496ns remote_storage=809ns web_handler=206ns query_engine=477ns
scrape=108.631µs scrape_sd=13.469µs notify=16.746µs notify_sd=7.923µs rules=1.049µs tracing=3.23µs

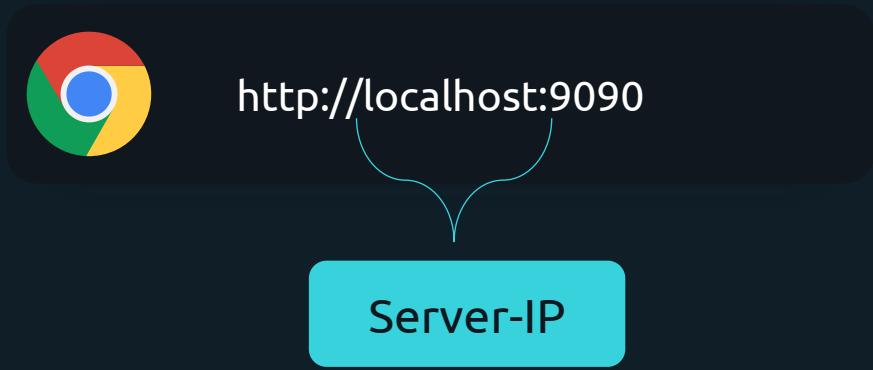
ts=2022-09-02T03:25:25.910Z caller=main.go:957 level=info msg="Server is ready to receive web requests."

ts=2022-09-02T03:25:25.910Z caller=manager.go:941 level=info component="rule manager" msg="Starting rule manager..."
```



Validate no Errors

Installation Bare-Metal/VM



A screenshot of the Prometheus Time Series Graph interface. The top navigation bar shows the title "Prometheus Time Series" and the URL "localhost:9090/graph". The main header includes the Prometheus logo and links for Alerts, Graph, Status, and Help. Below the header are several configuration checkboxes: "Use local time" (unchecked), "Enable query history" (unchecked), "Enable autocomplete" (checked), "Enable highlighting" (checked), and "Enable linter" (checked). A search bar labeled "Expression (press Shift+Enter for newlines)" is present. Below the search bar are two tabs: "Table" (selected) and "Graph". A date range selector shows "Evaluation time" with arrows for navigating between dates. The main content area displays the message "No data queried yet". At the bottom is a blue "Add Panel" button.

Installation Bare-Metal/VM

Prometheus Is configured to monitor itself by default

The screenshot shows the Prometheus web interface with the following details:

- Header:** Prometheus, Alerts, Graph, Status, Help.
- Top Controls:** Use local time (unchecked), Enable query history (unchecked), Enable autocomplete (checked), Enable highlighting (checked), Enable linter (checked).
- Search Bar:** A search bar containing the text "up". A red arrow points to the left side of the search bar.
- Buttons:** Execute (blue button) and a refresh icon.
- Graph Selection:** Graph tab selected.
- Time Range:** Evaluation time range with arrows for navigation.
- Result Panel:** A single data series table:

up{instance="localhost:9090", job="prometheus"}	1
up{instance="localhost:9090", job="prometheus"}	1

A red box highlights the entire result panel area.
- Metrics:** Load time: 60ms, Resolution: 14s, Result series: 1.
- Actions:** Remove Panel.



KodeKloud

Prometheus Installation SystemD

Prometheus Installation Systemd

>_

```
$ ./prometheus
```



Runs in Foreground

```
$ systemctl start prometheus
```



Doesn't start on-boot

Prometheus Installation Systemd

>_

```
$ sudo useradd --no-create-home --shell /bin/false prometheus
```



User Can't log in

Prometheus Installation Systemd

>_

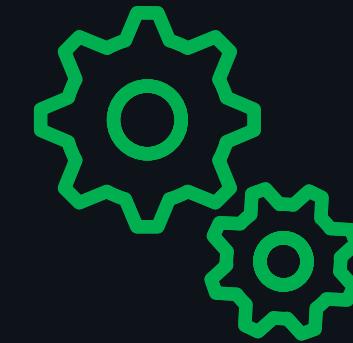
```
$ sudo mkdir /etc/prometheus
```

```
$ sudo mkdir /var/lib/prometheus
```

Permissions

```
$ sudo chown prometheus:prometheus /etc/prometheus
```

```
$ sudo chown prometheus:prometheus /var/lib/prometheus
```



prometheus.yaml

data



Prometheus Installation Systemd

>_

```
$ wget https://github.com/prometheus/prometheus/releases/download/v2.37.0/prometheus-  
2.37.0.linux-amd64.tar.gz  
saving to: 'prometheus-2.37.0.linux-amd64.tar.gz'
```

```
prometheus-2.37.0.linux- 100%[=====] 79.90M  
36.8MB/s   in 2.2s
```

```
2022-09-01 22:54:38 (36.8 MB/s) - 'prometheus-2.37.0.linux-amd64.tar.gz' saved  
[83782323/83782323]
```

```
FINISHED --2022-09-01 22:54:38--
```

```
Total wall clock time: 2.9s
```

```
Downloaded: 1 files, 80M in 2.2s (36.8 MB/s)
```

Prometheus Installation Systemd

>_

```
$ tar xvf prometheus-2.37.0.linux-amd64.tar.gz
user1@user1:~$ tar xvf prometheus-2.37.0.linux-amd64.tar.gz
prometheus-2.37.0.linux-amd64/
prometheus-2.37.0.linux-amd64/consoles/
prometheus-2.37.0.linux-amd64/consoles/index.html.example
prometheus-2.37.0.linux-amd64/consoles/node-cpu.html
```

Prometheus Installation Systemd

>_

```
$ sudo cp prometheus /usr/local/bin/
```

```
$ sudo cp promtool /usr/local/bin/
```

Permissions

```
$ sudo chown prometheus:prometheus /usr/local/bin/prometheus
```

```
$ sudo chown prometheus:prometheus /usr/local/bin/promtool
```

Prometheus Installation Systemd

>_

```
$ sudo cp -r consoles /etc/prometheus  
$ sudo cp -r console_libraries /etc/prometheus
```

Permissions

```
$ sudo chown -R prometheus:prometheus /etc/prometheus/consoles  
$ sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
```

Change permissions for directory and
all of the files within it

Prometheus Installation Systemd

>_

```
$ sudo cp prometheus.yaml /etc/prometheus/prometheus.yml
```

Permissions

```
$ sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

Prometheus Intallation Systemd

>_

```
$ sudo -u prometheus /usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries
```

Prometheus Installation Systemd

>_

```
$ sudo vi /etc/systemd/system/prometheus.service
```

```
>_          Prometheus.service

[Unit]
Description=Prometheus ...
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target
```

Start service after
network is up

Start service as part of normal
system start-up, whether or not
local GUI is active

Prometheus Installation Systemd

>_

```
$ sudo systemctl daemon-reload
```

Prometheus Installation Systemd

>_

```
$ sudo systemctl start prometheus
```

```
$ sudo systemctl status prometheus
```

```
prometheus.service - Prometheus
```

```
Loaded: loaded (/etc/systemd/system/prometheus.service; disabled; vendor preset: enabled)
```

```
Active: active (running) since Fri 2022-09-02 01:27:59 EDT; 3min 43s ago
```

```
Main PID: 6951 (prometheus)
```

```
Tasks: 6 (limit: 7940)
```

```
Memory: 18.1M
```

```
CPU: 143ms
```

```
CGroup: /system.slice/prometheus.service
```

```
└─6951 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --  
storage.tsdb.path /var/lib/prometheus/ --web.console.templates=/etc/pro
```

Prometheus Intallation Systemd

>_

```
$ sudo systemctl enable prometheus
```

Start service on-boot

```
$ sudo systemctl status prometheus
```

prometheus.service - Prometheus

Loaded: loaded (/etc/systemd/system/prometheus.service; disabled; vendor preset: enabled)

Active: active (running) since Fri 2022-09-02 01:27:59 EDT; 3min 43s ago

Main PID: 6951 (prometheus)

Tasks: 6 (limit: 7940)

Memory: 18.1M

CPU: 143ms

CGroup: /system.slice/prometheus.service

└─6951 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --
storage.tsdb.path /var/lib/prometheus/ --web.console.templates=/etc/pro

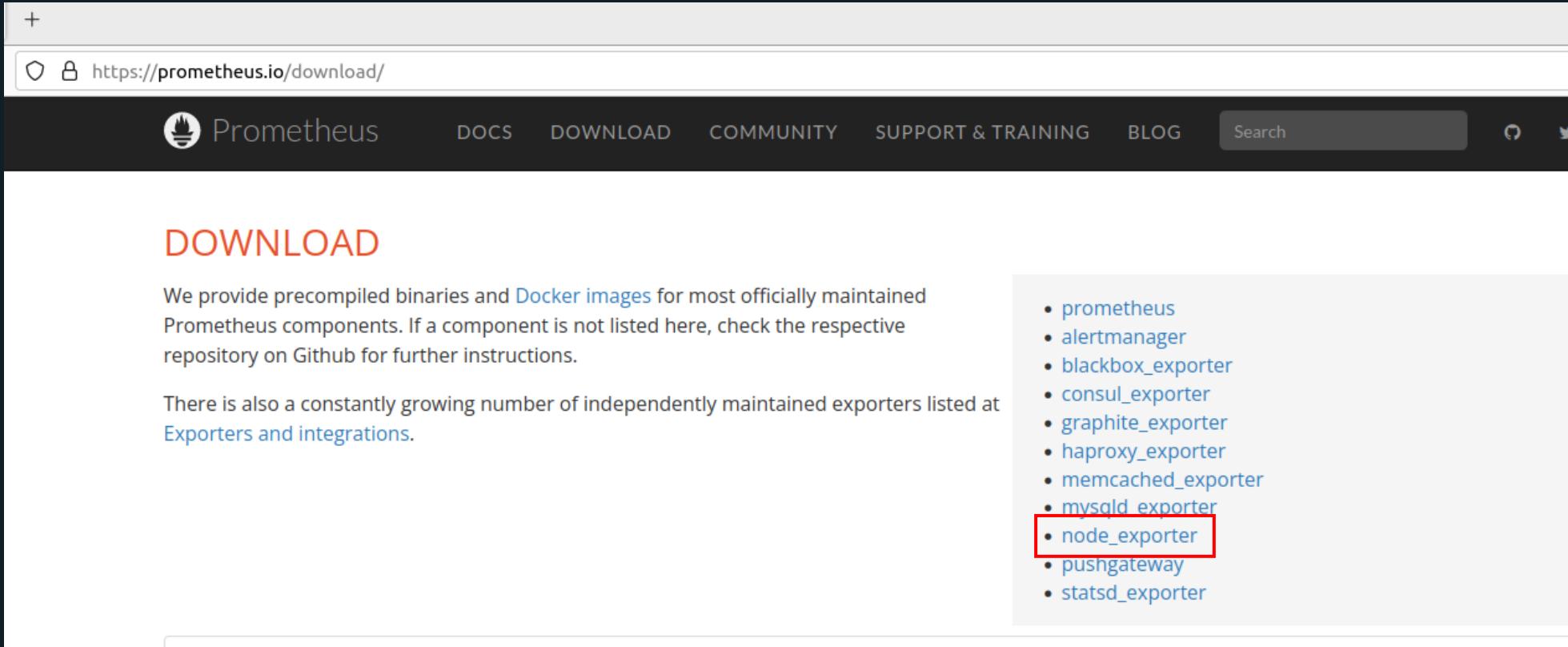


KodeKloud

Node Exporter Installation

Node Exporter Installation

http://prometheus.io/download



The screenshot shows a web browser window with the URL <https://prometheus.io/download/> in the address bar. The page has a dark header with the Prometheus logo and navigation links for DOCS, DOWNLOAD, COMMUNITY, SUPPORT & TRAINING, and BLOG. A search bar and social media icons for GitHub and Twitter are also present. The main content area has a red header "DOWNLOAD". Below it, text states: "We provide precompiled binaries and [Docker images](#) for most officially maintained Prometheus components. If a component is not listed here, check the respective repository on Github for further instructions." Another section says: "There is also a constantly growing number of independently maintained exporters listed at [Exporters and integrations](#)." To the right, a sidebar lists various exporters with the "node_exporter" link highlighted by a red rectangle.

- [prometheus](#)
- [alertmanager](#)
- [blackbox_exporter](#)
- [consul_exporter](#)
- [graphite_exporter](#)
- [haproxy_exporter](#)
- [memcached_exporter](#)
- [mysqld exporter](#)
- [**node_exporter**](#)
- [pushgateway](#)
- [statsd_exporter](#)

Node Exporter Installation

Download binary our copy url

node_exporter
Exporter for machine metrics [prometheus/node_exporter](#)

1.4.0-rc.0 / 2022-07-27 [Pre-release](#) [Release notes](#)

File name	OS	Arch	Size	SHA256 Checksum
node_exporter-1.4.0-rc.0.darwin-amd64.tar.gz	darwin	amd64	4.31 MiB	6a890622c47dabf16278bc0b473ddc478cdc5e7448b170de7cd5000e6a3bd45b
node_exporter-1.4.0-rc.0.linux-amd64.tar.gz	linux	amd64	9.28 MiB	5069b12547b16b7272365476805cfb1f24ea514ac31e86c764eca3a62fae7446

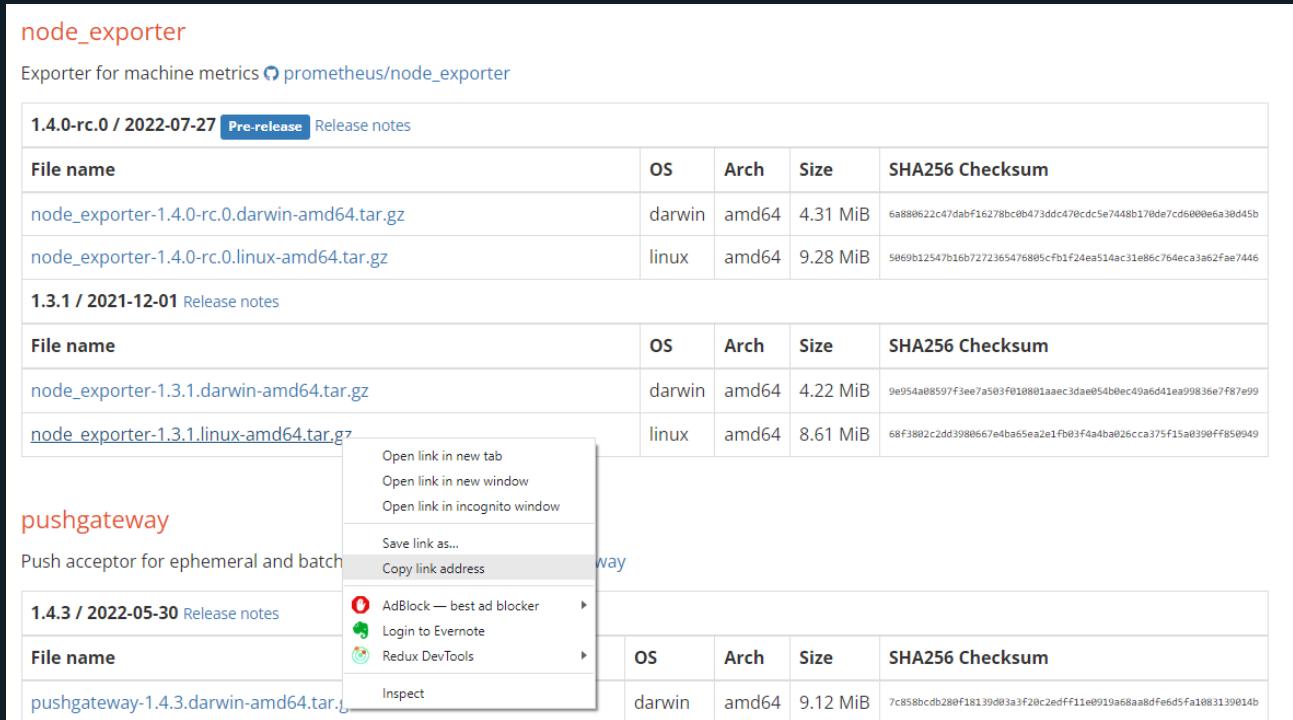
1.3.1 / 2021-12-01 [Release notes](#)

File name	OS	Arch	Size	SHA256 Checksum
node_exporter-1.3.1.darwin-amd64.tar.gz	darwin	amd64	4.22 MiB	9e954a08597f3ee7a503f0108801aaec3dae054bdec49a6d41ea99836e7f87e09
node_exporter-1.3.1.linux-amd64.tar.gz	linux	amd64	8.61 MiB	68f3802c2dd3980667e4ba65ea2efb03f4a4ba026cca375f15a0390ff850949

pushgateway
Push acceptor for ephemeral and batch

1.4.3 / 2022-05-30 [Release notes](#)

File name	OS	Arch	Size	SHA256 Checksum
pushgateway-1.4.3.darwin-amd64.tar.gz	darwin	amd64	9.12 MiB	7c858bcd280f18139d03a3f20c2edff11e0919a68aa8dfe6d5fa1083139014b



Node Exporter Installation

>_

```
$ wget https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_exporter-1.3.1.linux-amd64.tar.gz
HTTP request sent, awaiting response... 200 OK
Length: 9033415 (8.6M) [application/octet-stream]
Saving to: ‘node_exporter-1.3.1.linux-amd64.tar.gz’

node_exporter-1.3.1.linux-amd64.tar.gz
100%[=====] 8.61M 12.4MB/s    in 0.7s

2022-09-02 15:04:10 (12.4 MB/s) - ‘node_exporter-1.3.1.linux-amd64.tar.gz’
saved [9033415/9033415]
```

Node Exporter Installation

>_

```
$ tar -xvf node_exporter-1.3.1.linux-amd64.tar.gz
node_exporter-1.3.1.linux-amd64/
node_exporter-1.3.1.linux-amd64/LICENSE
node_exporter-1.3.1.linux-amd64/NOTICE
node_exporter-1.3.1.linux-amd64/node_exporter
```

```
$ cd node_exporter-1.3.1.linux-amd64
```

```
$ ls -l
```

```
-rw-r--r-- 1 user2 user2 11357 Dec  5 2021 LICENSE
-rwxr-xr-x 1 user2 user2 18228926 Dec  5 2021 node_exporter
-rw-r--r-- 1 user2 user2      463 Dec  5 2021 NOTICE
```

Node Exporter Installation

>_

```
$ ./node_exporter
ts=2022-09-05T16:51:59.947Z caller=node_exporter.go:115 level=info collector=vmstat
ts=2022-09-05T16:51:59.947Z caller=node_exporter.go:199 level=info msg="Listening on" address=:9100
ts=2022-09-05T16:51:59.947Z caller=tls_config.go:195 level=info msg="TLS is disabled." http2=false
```

```
$ curl localhost:9100/metrics
# TYPE promhttp_metric_handler_requests_in_flight gauge
promhttp_metric_handler_requests_in_flight 1

# HELP promhttp_metric_handler_requests_total Total number of scrapes by HTTP status code.

# TYPE promhttp_metric_handler_requests_total counter
promhttp_metric_handler_requests_total{code="200"} 0
promhttp_metric_handler_requests_total{code="500"} 0
promhttp_metric_handler_requests_total{code="503"} 0
```



Metrics



KodeKloud

Node Exporter Installation Systemd

Node Exporter Installation Systemd

>_

```
$ ./node_exporter
```



Runs in Foreground

```
$ systemctl start node_exporter
```



Doesn't start on-boot

Node Exporter Installation Systemd

>_

```
$ sudo cp node_exporter /usr/local/bin
```

Create User

```
$ sudo useradd --no-create-home --shell /bin/false node_exporter
```

Permissions

```
$ sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

Node Exporter Installation Systemd

>_

```
$ sudo vi /etc/systemd/system/node_exporter.service
```

```
>_          node_exporter.service

[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

Start service after
network is up

Start service as part of normal
system start-up, whether or not
local GUI is active

Node Exporter Installation Systemd

>_

```
$ sudo systemctl daemon-reload
```

Node Exporter Installation Systemd

>_

```
$ sudo systemctl start node_exporter
```

```
$ sudo systemctl status node_exporter
```

```
● node_exporter.service - Node Exporter
```

```
    Loaded: loaded (/etc/systemd/system/node_exporter.service; disabled; vendor preset: enabled)
```

```
    Active: active (running) since Mon 2022-09-05 13:29:45 EDT; 23s ago
```

```
      Main PID: 20222 (node_exporter)
```

```
      Tasks: 3 (limit: 8247)
```

```
      Memory: 2.1M
```

```
      CPU: 7ms
```

```
      CGroup: /system.slice/node_exporter.service
```

```
          └─20222 /usr/local/bin/node_exporter
```

Node Exporter Installation Systemd

>_

```
$ sudo systemctl enable node_exporter
```

Start service on-boot

```
$ sudo systemctl status node_exporter
```

- node_exporter.service - Node Exporter

 Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)

 Active: active (running) since Mon 2022-09-05 13:29:45 EDT; 23s ago

 Main PID: 20222 (node_exporter)

 Tasks: 3 (limit: 8247)

 Memory: 2.1M

 CPU: 7ms

 CGroup: /system.slice/node_exporter.service

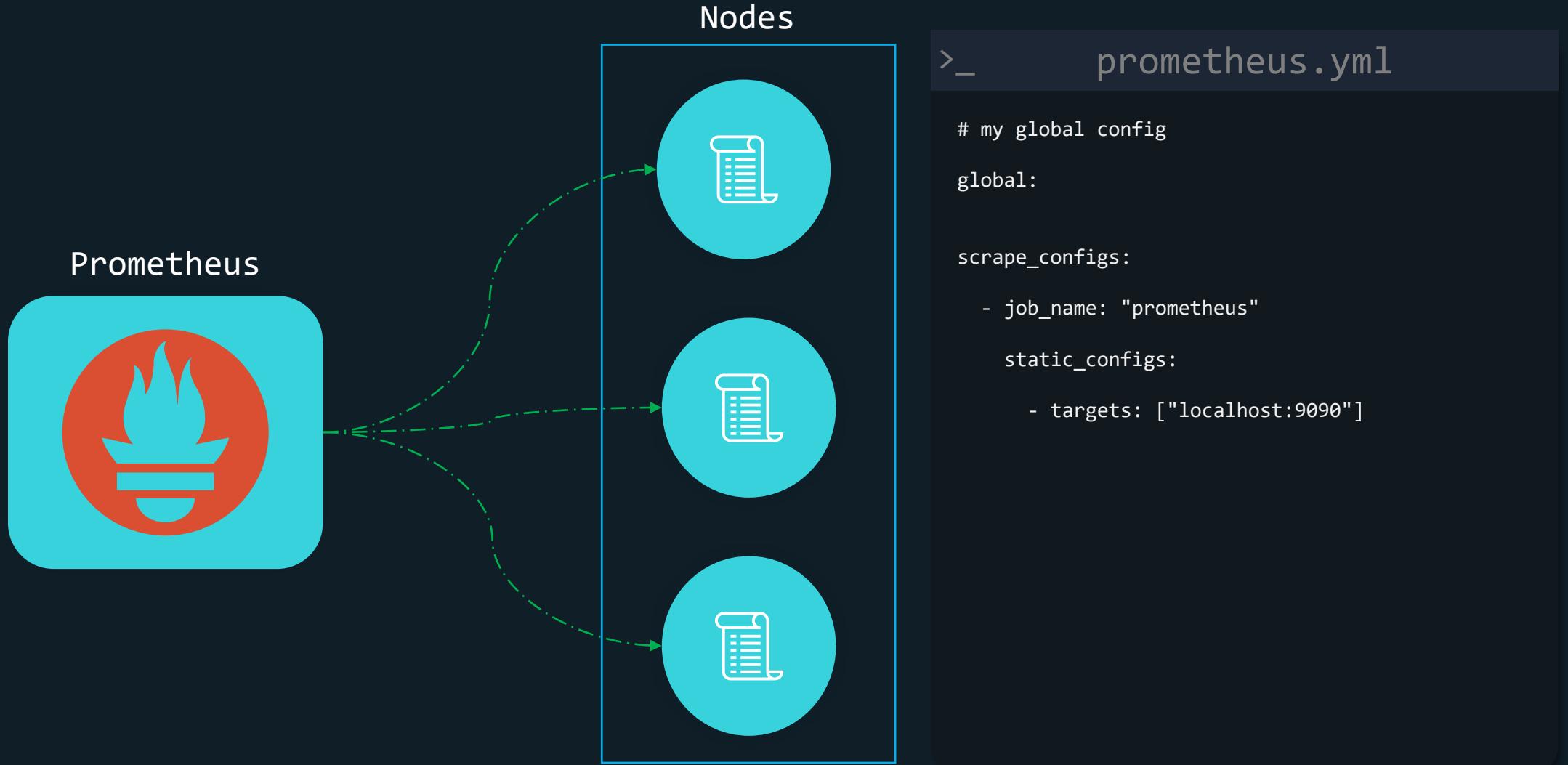
 └─20222 /usr/local/bin/node_exporter



KodeKloud

Prometheus Configuration

Prometheus Configuration



Prometheus Configuration

>_ prometheus.yml

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s

scrape_configs:
  - job_name: 'node'
    scrape_interval: 15s
    scrape_timeout: 5s
    sample_limit: 1000
    static_configs:
      - targets: ['172.16.12.1:9090']

# Configuration related to AlertManager
alerting:

# Rule files specifies a list of files rules are read from
rule_files:

# Settings related to the remote read/write feature.
remote_read:
remote_write:

# Storage related settings
storage:
```

Default parameters for all other config sections

Define targets & configs for metrics collection

A collection of instances that need to be scraped

Configs for scrape job. Takes precedence over global config

Set of targets to scrape

Prometheus Configuration

1. Define a Job with a name of nodes
2. Metrics should be scraped every 30s
3. Timeout of a scrape is 3s
4. Use HTTPS instead of default HTTP
5. Scrape path should be changed from default /metrics to /stats/metrics
6. Scrape two targets at the following IP
 1. 10.231.1.2:9090
 2. 192.168.43.8:9090

```
>_          prometheus.yml
scrape_configs:
  - job_name: 'nodes'
    scrape_interval: 30s
    scrape_timeout: 3s
    scheme: https
    metrics_path: /stats/metrics
    static_configs:
      - targets: ['10.231.1.2:9090', '192.168.43.9:9090']
```

Scrape Config Options

>_

prometheus.yml

```
scrape_configs:  
  # How frequently to scrape targets from this job.  
  [ scrape_interval: <duration> | default = <global_config.scrape_interval> ] ←  
  
  # Per-scraper timeout when scraping this job.  
  [ scrape_timeout: <duration> | default = <global_config.scrape_timeout> ] ←  
  
  # The HTTP resource path on which to fetch metrics from targets.  
  [ metrics_path: <path> | default = /metrics ] ←  
  
  # Configures the protocol scheme used for requests.  
  [ scheme: <scheme> | default = http ] ←  
  
  # Sets the `Authorization` header on every scrape request with the  
  # configured username and password.  
  # password and password_file are mutually exclusive.  
  basic_auth:  
    [ username: <string> ] ←  
    [ password: <secret> ] ←  
    [ password_file: <string> ] ←
```

Restart Prometheus

```
>_
```

```
$ ctrl+c -> ./prometheus
```

```
$ kill -HUP <pid>
```

```
$ systemctl restart prometheus
```



KodeKloud

Reloading Configuration

After making changes to the Prometheus configs, a **reload** of the configuration needs to take place for changes to take affect

1. Restart Prometheus

```
# systemctl restart prometheus
```

2. Send a SIGHUP signal to the Prometheus process

```
# sudo killall -HUP prometheus
```

3. Send a POST/PUT request to http://<prometheus>/-/reload

- This functionality not enabled by default
- Need to start prometheus with --web.enable-lifecycle flag

Reloading Configuration

```
>_          Prometheus.service

[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
    --config.file /etc/prometheus/prometheus.yml \
    --storage.tsdb.path /var/lib/prometheus/ \
    --web.console.templates=/etc/prometheus/consoles \
    --web.console.libraries=/etc/prometheus/console_libraries \
    --web.enable-lifecycle
[Install]
WantedBy=multi-user.target
```

Reload Configuration

```
>_
```

```
$ sudo systemctl daemon-reload
```

```
$ sudo systemctl restart prometheus
```

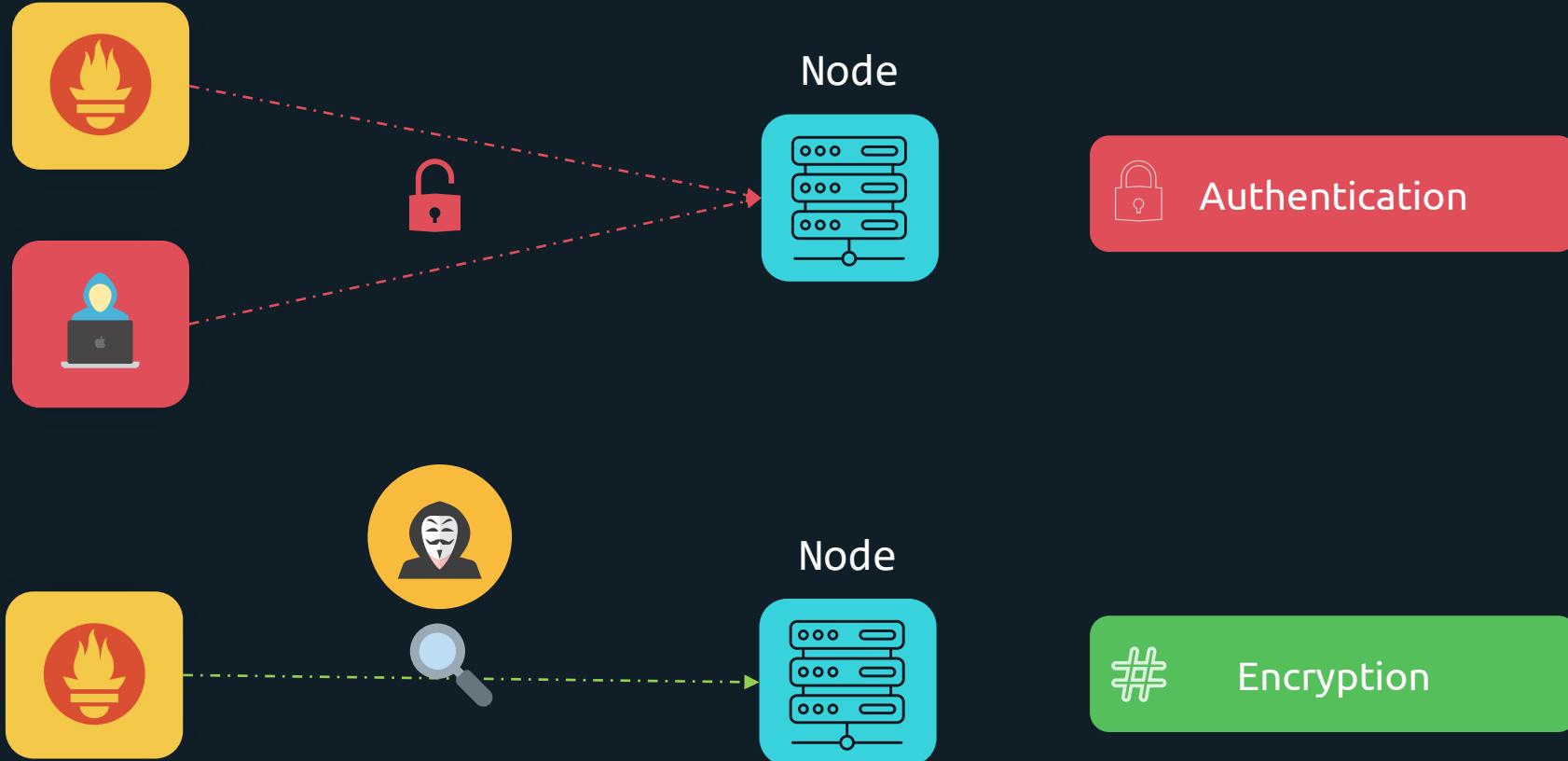
```
$ curl -X POST http://<prometheus>/-/reload
```



KodeKloud

Authentication & Encryption

Authentication & Encryption



Node Exporter TLS

```
>_ $ sudo openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout  
node_exporter.key -out node_exporter.crt -subj  
"/C=US/ST=California/L=Oakland/O=MyOrg/CN=localhost" -addext  
"subjectAltName = DNS:localhost"
```

```
$ ls -l  
-rw-r--r-- 1 user2 user2 11357 Dec 5 2021 LICENSE
```

```
-rwxr-xr-x 1 user2 user2 18228926 Dec 5 2021 node_exporter
```

```
-rw-r--r-- 1 root root 1326 Sep 5 18:04 node_exporter.crt
```

```
-rw----- 1 root root 1700 Sep 5 18:04 node_exporter.key
```

Node Exporter TLS

```
$ vi config.yml  
  
$ ./node_exporter --web.config=config.yml  
ts=2022-09-05T22:26:53.965Z caller=node_exporter.go:115 level=info  
collector=zfs  
  
ts=2022-09-05T22:26:53.966Z caller=node_exporter.go:199 level=info  
msg="Listening on" address=:9100  
  
ts=2022-09-05T22:26:53.966Z caller=tls_config.go:228 level=info msg="TLS  
is enabled." http2=true
```

```
>_ config.yml  
  
tls_server_config:  
  cert_file: node_exporter.crt  
  key_file: node_exporter.key
```

Node Exporter TLS

```
>_
```

```
$ sudo mkdir /etc/node_exporter
```

```
$ mv node_exporter.* /etc/node_exporter
```

```
$ sudo cp config.yml /etc/node_exporter
```

```
$ chown -R node_exporter:node_exporter  
/etc/node_exporter
```

Node Exporter TLS

```
>_
$ vi /etc/system/system/node_exporter.service
$ systemctl daemon-reload
$ systemctl restart node_exporter
```

```
>_ node_exporter.service
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter --web.config=/etc/node_exporter/config.yml

[Install]
WantedBy=multi-user.target
```

Node Exporter TLS

>_

```
$ curl https://localhost:9100/metrics
curl: (60) SSL certificate problem: self-signed certificate
More details here: https://curl.se/docs/sslcerts.html
curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

Caused by self signed
certificate

```
$ curl -k https://localhost:9100/metrics
# HELP promhttp_metric_handler_requests_total Total number of scrapes by
HTTP status code.
# TYPE promhttp_metric_handler_requests_total counter
promhttp_metric_handler_requests_total{code="200"} 10217
promhttp_metric_handler_requests_total{code="500"} 0
promhttp_metric_handler_requests_total{code="503"} 0
```

Allows Insecure
Connection

Prometheus TLS Config

>_

Copy node_exporter.crt to Prometheus server

```
$ scp  
username:password@node:/etc/node_exporter/node_expo  
rter.crt /etc/prometheus
```

```
$ chown prometheus:prometheus node_exporter.crt
```

```
$ vi /etc/prometheus/prometheus.yml
```

```
$ systemctl restart prometheus
```

>_

prometheus.yml

```
scrape_configs:  
  - job_name: "node"  
    scheme: https ←  
    tls_config:  
      ca_file: /etc/prometheus/node_exporter.crt ←  
      insecure_skip_verify: true ←  
    static_configs:  
      - targets: ["192.168.1.168:9100"]
```

Only needed for self
signed certs

Prometheus Authentication



Generate Hash Password

my-password



\$2y\$10\$EYxs8lOG46m9CtpB/XIPxO1ei7i7mD4keR/8JO6m

Hashing



Install apache2-utils or httpd-tools

```
$ sudo apt install apache2-utils
```



```
$ htpasswd -nBC 12 "" | tr -d ':\\n'
```

```
New password:
```

```
Re-type new password:
```

```
$2y$12$gfAopKV008KK063rJe0Z9efGRx30qJEZ9vC8IxBP9.cXkurgug
```

```
c6
```



Use Preferred Programming language

```
>_
```

```
import getpass
import bcrypt

password = getpass.getpass("password: ")
hashed_password =
bcrypt.hashpw(password.encode("utf-8"),
bcrypt.gensalt())
print(hashed_password.decode())

$2y$12$gfAopKV008KK063rJe0Z9efGRx3xhBP9
```

Hashing

>_

```
$ sudo apt install apache2-utils  
$ htpasswd -nBC 12 "" | tr -d ':\\n'  
New password:  
Re-type new password:  
$2y$12$gfAopKV008KK063rJe0Z9efGRx30qJEZ9vC8IxP9.cXkurgugc6
```

Auth Configuration Node Exporter

```
>_
```

```
$ vi /etc/node_exporter/config.yml
```

```
$ systemctl restart node_exporter
```

```
>_
```

```
tls_server_config:  
  cert_file: node_exporter.crt  
  key_file: node_exporter.key  
basic_auth_users:  
  prometheus: $2y$12$dCqkk9uah20wF
```



username



password

Auth Configuration



Prometheus Server will now show as Unauthorized

Prometheus Alerts Graph Status ▾ Help

Targets

All Unhealthy Collapse All Filter by endpoint or labels

node (0/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.1.168:9100/metrics	DOWN	instance="192.168.1.168:9100" job="node"	4.830s ago	0.754ms	server returned HTTP status 401 Unauthorized

Auth Configuration Prometheus

```
>_  
$ vi /etc/prometheus/prometheus.yml  
  
$ systemctl restart prometheus  
  
>_  
- job_name: "node"  
  scheme: https  
  basic_auth:  
    username: prometheus  
    password: password  
  
↑  
plain-text  
password
```

Auth Configuration Prometheus



Node will be in UP state

Prometheus Alerts Graph Status ▾ Help

Targets

All Unhealthy Collapse All

Filter by endpoint or labels

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

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.1.168:9100/metrics	UP	instance="192.168.1.168:9100" job="node"	15.45s ago	218.460ms	



KodeKloud

Promtools

Promtools is a utility tool shipped with Prometheus that can be used to:

- Check and validate configuration
 - Validate Prometheus.yml
 - Validate rules files
- Validate metrics passed to it are correctly formatted
- Can perform queries on a Prometheus server
- Debugging & Profiling a Prometheus server
- Perform unit tests against Recording/Alerting rules

promtools

>_

```
$ promtool check config /etc/prometheus/prometheus.yml
```

Checking prometheus.yml

SUCCESS: prometheus.yml is valid prometheus config file syntax

```
$ promtool check config /etc/prometheus/prometheus.yml
```

Checking prometheus.yml

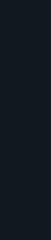
FAILED: parsing YAML file prometheus.yml: yaml: unmarshal errors:
line 24: field metric_path not found in type config.ScrapeConfig



>_

```
scrape_configs:
```

- job_name: "node"
metric_path: "/metrics"
static_configs:
- targets: ["node1:9100"]



Should be "metrics_path"

Prometheus configs can now be validated **before** applying them to a production server

This will prevent any unnecessary downtime while config issues are being identified

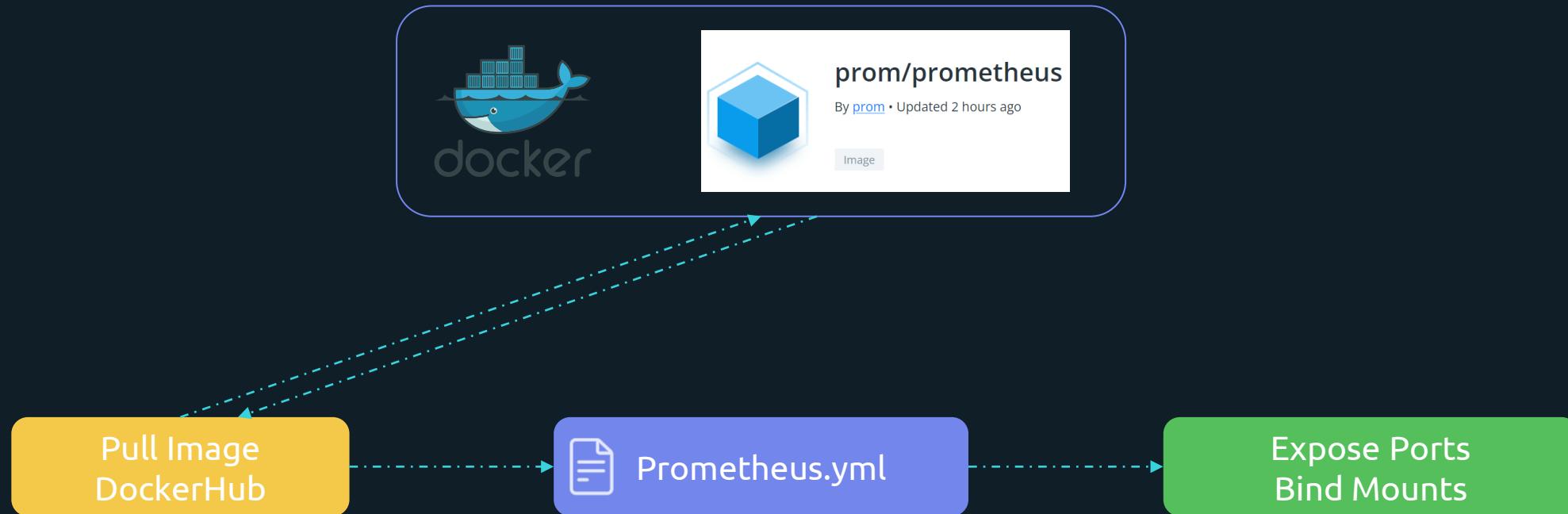
As we progress through this course, we will cover all of the other features Promtools has to offer



KodeKloud

Prometheus on Docker

Prometheus Docker



Prometheus Docker

```
>_
$ vi prometheus.yml

$ docker run -d /path-
to/prometheus.yml:/etc/prometheus/prometheus.yml -p
9090:9090 prom/prometheus
```

```
>_          prometheus.yml
```

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



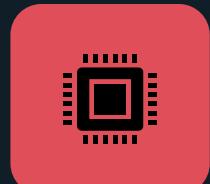
KodeKloud

Metrics

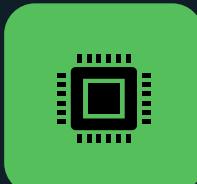
Prometheus Metrics

<metric_name>[{\<label_1>="value_1"}, {\<label_N>="value_N"}] <metric_value>

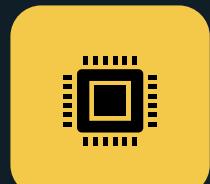
```
node_cpu_seconds_total{cpu="0",mode="idle"} 258277.86
```



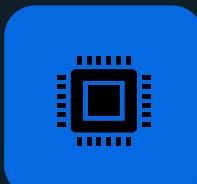
CPU 0



CPU 1



CPU 2



CPU 3



Labels provide us information on which cpu this metric is for and for what cpu state(idle)

```
node_cpu_seconds_total{cpu="0",mode="idle"} 258244.86
node_cpu_seconds_total{cpu="1",mode="idle"} 427262.54
node_cpu_seconds_total{cpu="2",mode="idle"} 283288.12
node_cpu_seconds_total{cpu="3",mode="idle"} 258202.33
```

Prometheus Metrics

```
node_cpu_seconds_total{cpu="0",mode="idle"} 258277.86
node_cpu_seconds_total{cpu="0",mode="iowait"} 61.16
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 61.12
node_cpu_seconds_total{cpu="0",mode="softirq"} 6.63
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 372.46
node_cpu_seconds_total{cpu="0",mode="user"} 3270.86
```

top

```
top - 14:28:10 up 3 days, 1:49, 1 user, load
average: 0.24, 0.09, 0.09
```

```
%Cpu(s): 32.6 us, 1.4 sy, 0.0 ni, 66.0 id,
0.0 wa, 0.0 hi, 0.0 si, 0.0 st
```

Timestamp

When Prometheus scrapes a target and retrieves metrics, it also stores the time at which the metric was scraped as well.

The **timestamp** will look like this:

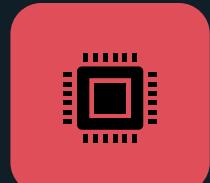
1668215300

This is called a **unix timestamp**, which is the number of seconds that have elapsed since Epoch(January 1st 1970 UTC)

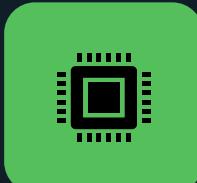
Prometheus Metrics

<metric_name>[{\<label_1>="value_1"}, {\<label_N>="value_N"}] <metric_value>

```
node_cpu_seconds_total{cpu="0",mode="idle"} 258277.86 Jan 1, 12:51
```



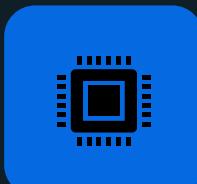
CPU 0



CPU 1



CPU 2



CPU 3



Labels provide us information on which cpu this metric is for and for what cpu state(idle)

```
node_cpu_seconds_total{cpu="0",mode="idle"} 258244.86
node_cpu_seconds_total{cpu="1",mode="idle"} 427262.54
node_cpu_seconds_total{cpu="2",mode="idle"} 283288.12
node_cpu_seconds_total{cpu="3",mode="idle"} 258202.33
```

Prometheus Time Series

Stream of timestamped values sharing the same metric and set of labels

```
node_filesystem_files{device="sda2", instance="server1"}  
node_filesystem_files{device="sda3", instance="server1"}  
node_filesystem_files{device="sda2", instance="server2"}  
node_filesystem_files{device="sda3", instance="server2"}
```

```
node_cpu_seconds_total{cpu="0", instance="server1"}  
node_cpu_seconds_total{cpu="1", instance="server1"}  
node_cpu_seconds_total{cpu="0", instance="server2"}  
node_cpu_seconds_total{cpu="1", instance="server2"}
```



There are two metrics(`node_filesystem_files`, `node_cpu_seconds_total`)

There are 8 total time series(unique combination of metric and set of labels)

Metric Attributes

Metrics have a TYPE and HELP attribute

```
# HELP node_disk_discard_time_seconds_total This is the total number of seconds spent by all discards.  
# TYPE node_disk_discard_time_seconds_total counter  
node_disk_discard_time_seconds_total{device="sda"} 0  
node_disk_discard_time_seconds_total{device="sr0"} 0
```



HELP – description of what the metric is



TYPE – Specifies what type of metric(counter, gauge, histogram, summary)

Metric Types



Counter



Gauge



Histogram



Summary

Counter



Counter

- ✓ How many times did X happen
- ✓ Number can only increase

Total #
requests

Total #
Exceptions

Total # of
job
executions

Gauge



Gauge

- ✓ What is the current value of X
- ✓ Can go up or down

Current CPU
Utilization

Available
System
Memory

Number of
concurrent
requests

Histogram



Histogram

- ✓ How long or how big something is
- ✓ Groups observations into configurable bucket sizes

Response Time

< 1s
< 0.5s
< 0.2s

Request size

< 1500Mb
< 1000Mb
< 800Mb

Response Time



Summary



Summary

- ✓ Similar to histograms (track how long or how big)
- ✓ How many observations fell below x
- ✓ Don't have to define quantiles ahead of time

Response Time

20% = .3s
50% = 0.8s
80% = 1s

Request size

20% = 50Mb
50% = 200Mb
80% = 500Mb



Metric Rules



Metric name specifies a general feature of a system to be measured



May contain ASCII letters, numbers, underscores, and colons



Must match the regex `[a-zA-Z_][a-zA-Z0-9_]*`



Colons are reserved only for recording rules

Labels



Labels are Key-Value pairs associated with a metric



Allows you to split up a metric by a specified criteria



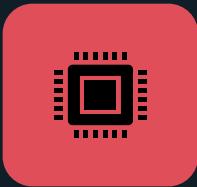
Metric can have more than one label



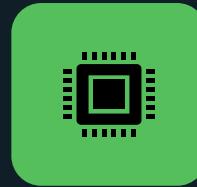
ASCII letters, numbers, underscores



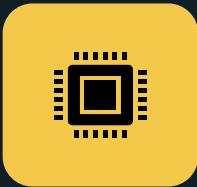
Must match regex [a-zA-Z0-9_]*



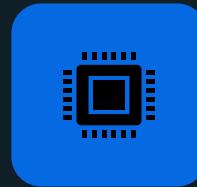
CPU 0



CPU 1



CPU 2



CPU 3



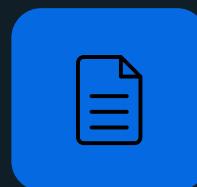
/data



/root



/dev



/run

cpu=0

cpu=1

cpu=2

cpu=3

fs=/data

fs=/root

fs=/dev

fs=/run

Why Labels



API for e-commerce app



Difficult to calculate total requests across all paths



Sum all requests: `sum(requests_total)`

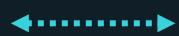
/auth



requests_auth_total

`requests_total{path=/auth}`

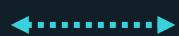
/user



requests_user_total

`requests_total{path=/user}`

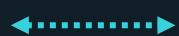
/products



requests_products_total

`requests_total{path=/products}`

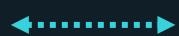
/cart



requests_cart_total

`requests_total{path=/cart}`

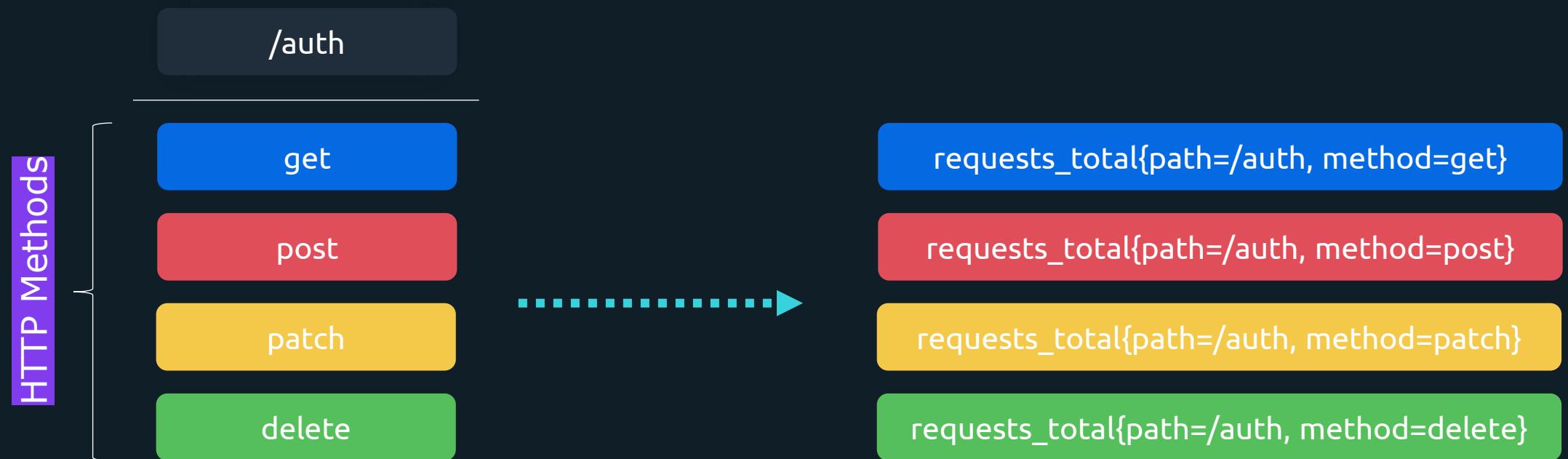
/orders



requests_orders_total

`requests_total{path=/orders}`

Multiple Labels



Internal Labels



Metric name is just another label

```
node_cpu_seconds_total{cpu=0}
```



```
{__name__=node_cpu_seconds_total, cpu=0}
```

Labels surrounded by __ are considered
internal to prometheus

Labels



Every metric is assigned 2 labels by default(**target** and **job**)

node_boot_time_seconds

Table Graph

Evaluation time

node_boot_time_seconds{**instance**=“192.168.1.168:9100”, **job**=“node”}

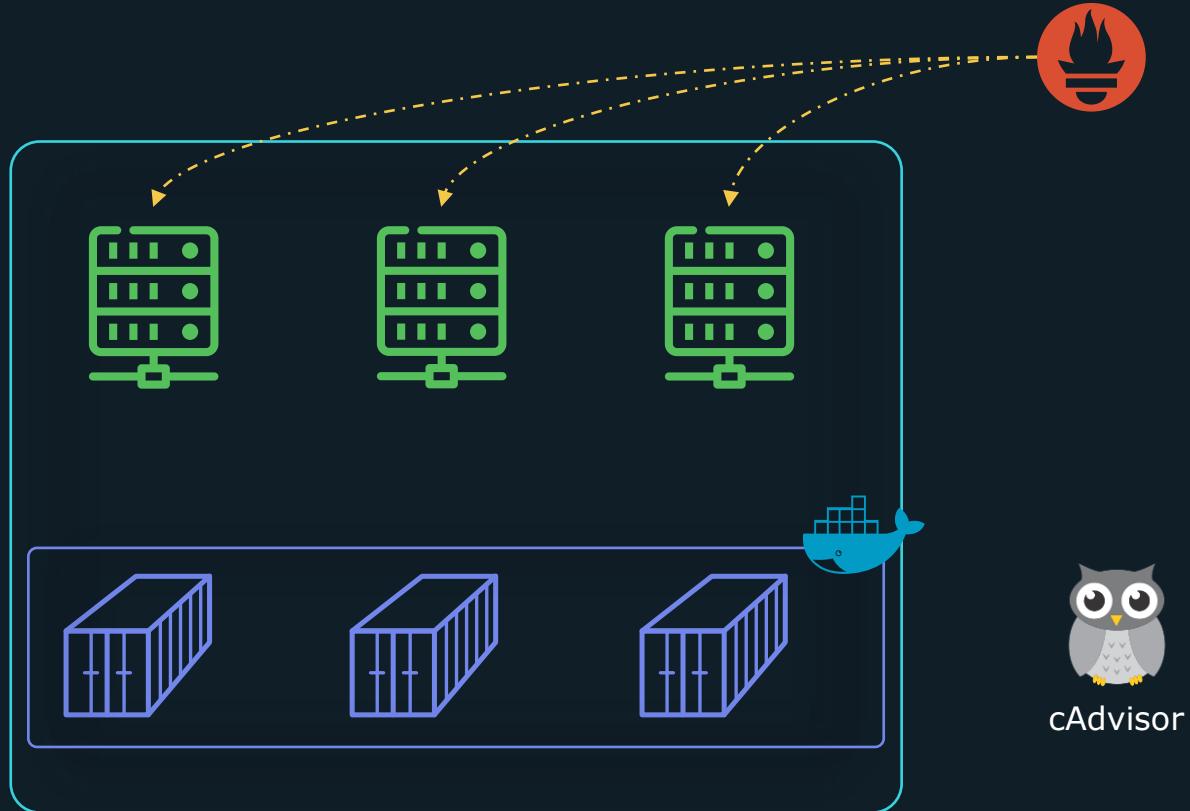
```
>_
  job_name: "node"
  scheme: https
  basic_auth:
    username: prometheus
    password: password
  static_configs:
    - targets:
        - "192.168.1.168:9100"
```



KodeKloud

Monitoring Containers

Container Metrics



- ✓ Metrics can also be scraped from **containerized** environments
- ✓ Docker **Engine** Metrics
- ✓ Container metrics using cAdvisor



Docker Engine metrics

```
>_
$ vi /etc/docker/daemon.json
$ sudo systemctl restart docker
$ curl localhost:9323/metrics
```

```
>_          daemon.json
{
  "metrics-addr" : "127.0.0.1:9323",
  "experimental" : true
}
```

Docker Engine Metrics

>_

Prometheus.yml

```
scrape_configs:  
  - job_name: "docker"  
    static_configs:  
      - targets: ["<ip-docker-host>:9323"]
```

cAdvisor Metrics

```
>_
$ vi docker-compose.yml
$ docker-compose up
$ curl localhost:8080/metrics
```

More Info

<https://github.com/google/cadvisor>

```
>_      docker-compose.yml
version: '3.4'
services:
  cadvisor:
    image: gcr.io/cadvisor/cadvisor
    container_name: cadvisor
    privileged: true
    devices:
      - "/dev/kmsg:/dev/kmsg"
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:ro
      - /sys:/sys:ro
      - /var/lib/docker/:/var/lib/docker:ro
      - /dev/disk/:/dev/disk:ro
    ports:
      - 8080:8080
```

cAdvisor Metrics

>_

Prometheus.yml

```
scrape_configs:  
  - job_name: "cAdvisor"  
    static_configs:  
      - targets: ["<docker-host-ip>:8080"]
```

Docker Metrics vs cAdvisor Metrics



Docker Engine metrics

- ✓ How much cpu does `docker` use
- ✓ **Total** number of failed image builds
- ✓ Time to process container `actions`
- ✓ No metrics `specific` to a container



cAdvisor metrics

- ✓ How much cpu/mem does **each** container use
- ✓ Number of `processes` running inside a container
- ✓ `Container` uptime
- ✓ Metrics on a `per container` basis



KodeKloud

PromQL

Q | What is PromQL



- ✓ Short for Prometheus Query Language
- ✓ Main way to **query** metrics within Prometheus
- ✓ Data returned can be **visualized** in dashboards
- ✓ Used to build **alerting** rules to notify administrators

Section Outline

- Expression Data Structure
- Selectors & modifiers
- Operators & Functions
- Vector Matching
- Aggregators
- Subqueries
- Histogram/Summary



KodeKloud

PromQL Data Types

PromQL Data Types



A PromQL expression can evaluate to one of four types:

- ✓ String – a simple `string` value (currently unused)
- ✓ Scalar – a simple numeric `floating point` value
- ✓ Instant Vector – set of time series containing a single sample for each time series, all sharing the `same` timestamp
- ✓ Range Vector – set of time series containing a `range` of data points over time for each time series

String



String – a simple string value (currently unused)

“some random text”

“This is a string”

Scalar



Scalar – a simple numeric floating point value

54.743

127.43

Instant Vector



Instant Vector – set of time series containing a single sample for each time series, all sharing the same timestamp

\$ node_cpu_seconds_total

Time Series			
Metric	Labels	value	TimeStamp
node_cpu_seconds_total	{cpu="0", instance="server1"}	258277.86	March 3 rd 11:05AM
node_cpu_seconds_total	{cpu="1", instance="server1"}	448430.21	March 3 rd 11:05AM
node_cpu_seconds_total	{cpu="0", instance="server2"}	941202.32	March 3 rd 11:05AM
node_cpu_seconds_total	{cpu="1", instance="server2"}	772838.83	March 3 rd 11:05AM

Same timestamp

Returns Metrics at one single point in time

Range Vector



Range Vector – set of time series containing a range of data points over time for each time series

```
$ node_cpu_seconds_total[3m]
```

Time Series			
Metric	Labels	value	TimeStamp
node_cpu_seconds_total	{cpu="0", instance="server1"}	674478.07	March 3 rd 08:05AM
		674626.76	March 3 rd 08:06AM
		566873.04	March 3 rd 08:07AM
node_cpu_seconds_total	{cpu="1", instance="server2"}	884597.02	March 3 rd 08:05AM
		540071.18	March 3 rd 08:06AM
		944799.49	March 3 rd 08:07AM

Contains all data during 3 minute window

Returns metrics over the course of a certain time period



KodeKloud

Selectors

Selectors

```
$ node_filesystem_avail_bytes
```

```
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node2", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/" }  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node2", mountpoint="/" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/user/1000"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/user/1000"}
```

A query with just the metric name will return all time series with that metric

Matchers



What if we only want to return a subset of times series for a metric

Label Matchers



Exact match on a Label value



Negative equality matcher – return time series that don't have the label



Regular expression matcher – matches time series with labels that match regex



Negative regular expression matcher

Equality Matcher



Return all time series from node1

```
$ node_filesystem_avail_bytes{instance="node1"}
```

```
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node2", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/" }  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node2", mountpoint="/" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/user/1000" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/lock" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/snapd/ns" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/user/1000" }
```

instance="node1" will match all time series from
node1

Negative equality Matcher



Return all time series where device is not equal to "tmpfs"

```
$ node_filesystem_avail_bytes{device!="tmpfs"}
```

```
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node2", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/" }  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node2", mountpoint="/" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/user/1000"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/user/1000"}
```

`!=` matcher will ensure that only devices that are not
"tmpfs"

Regex Matcher



Return all time series where device starts with "/dev/sda" (sda2 & sda3)

```
$ node_filesystem_avail_bytes{device=~"/dev/sda.*"}
```

```
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node2", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/" }  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node2", mountpoint="/" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/user/1000"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/user/1000"}
```

Learn more about regex

<https://github.com/google/re2/wiki/Syntax>

To match anything that starts with /dev/sda, a regex
"/dev/sda.*" will need to be used

Negative Regex Matcher



Return all time series where mountpoint does not start with "/boot"

```
$ node_filesystem_avail_bytes{mountpoint!~/boot.*"}
```

```
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node2", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/" }  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node2", mountpoint="/" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/user/1000"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/user/1000"}
```

To match anything that starts with "/boot", a regex
"/boot.*" will need to be used

Multiple Selectors



Return all time series from node1 without a "device=tmpfs"

```
$ node_filesystem_avail_bytes{instance="node1", device!="tmpfs"}
```

```
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node2", mountpoint="/boot/efi"}  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/" }  
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node2", mountpoint="/" }  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/user/1000"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/lock"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/snapd/ns"}  
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node2", mountpoint="/run/user/1000"}
```

Multiple selectors can be used by separating them by
a comma

Range Vector Selectors



Returns all the values for a metric over a period of time

```
$ node_arp_entries{instance="node1"}[2m]
```

```
8 @1669253129.609  
2 @1669253144.609  
3 @1669253159.609  
1 @1669253174.609  
7 @1669253189.609  
7 @1669253204.609  
7 @1669253219.609  
6 @1669253234.609
```

2min

Returns node_arp_entries metric data for the past 2 minutes



KodeKloud

Offset Modifier

Offset Modifier

When performing a query, it returns the **current** value of a metric

```
$ node_memory_Active_bytes{instance="node1"}
```

```
22259302
```



Most recent value

Offset Modifier

To get historic data use an **offset modifier** after the label matching

```
$ node_memory_Active_bytes{instance="node1"} offset 5m      22259302
```



Value 5 minutes
ago

Time Units

Suffix	Meaning
ms	Milliseconds
s	Seconds
m	Minutes
h	Hours
d	Days
w	Weeks
y	Years, which have 365 days

Offset Modifier

5 days ago

```
$ node_memory_Active_bytes{instance="node1"} offset 5d 22259302
```

2 weeks ago

```
$ node_memory_Active_bytes{instance="node1"} offset 2w 44823311
```

1.5 hours ago

```
$ node_memory_Active_bytes{instance="node1"} offset 1h30m 11864917
```

Offset Modifier

To go back to a specific point in time use the @ modifier

```
Unix timestamp  
↓  
$ node_memory_Active_bytes{instance="node1"} @1663265188 22259302
```

```
↑  
September 15,  
2022 6:06:28 PM  
GMT
```

Offset Modifier

The **offset** modifier can be combined with the **@** modifier

5 minutes before



```
$ node_memory_Active_bytes{instance="node1"} @1663265188 offset 5m
```

22259302



September 15, 2022 6:06:28 PM
GMT

Offset Modifier

Order does not matter when combining @ modifier and
offset modifier

```
$ node_memory_Active_bytes{instance="node1"} @1663265188 offset 5m
```

=

```
$ node_memory_Active_bytes{instance="node1"} offset 5m @1663265188
```

Offset Modifier

The **offset** and **@ modifier** also work with range vectors

September 15, 2022 6:06:28 PM
GMT

\$ node_memory_Active_bytes{instance="node1"}[2m] @1663265188 offset 10m

Get 2 minutes worth of data 10 minutes
before September 15, 2022 6:06:28 PM GMT



5:54:36 PM GMT



2217328640 @1663264476.952
2217328640 @1663264491.952
2217328640 @1663264506.952
2217328640 @1663264521.952
2217328640 @1663264536.952
2217328640 @1663264551.952
2217328640 @1663264566.952
2217328640 @1663264581.952

5:56:21 PM GMT





KodeKloud

Operators

Arithmetic Operators

Arithmetic operators provide the ability to perform basic math operations

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo
^	Power

Arithmetic Operators

The `+` operator will add x amount to the result

```
$ node_memory_Active_bytes{instance="node1"} 2204815360
```

```
$ node_memory_Active_bytes{instance="node1"} + 10 2204815370
```



Add 10 to result

Arithmetric operators

>_

```
$ node_memory_Active_bytes
```

```
node_memory_Active_bytes{instance="node1", job="node"}  
node_memory_Active_bytes{instance="node1", job="node"}
```

```
2204844032  
2204887040
```

bytes

```
$ node_memory_Active_bytes / 1024
```

```
{instance="node1", job="node"} 2153168  
{instance="node1", job="node"} 2153210
```

kilobytes

Metric Name is dropped.
No longer a metric

Comparison Operators

Operator	Description
<code>==</code>	Equal
<code>!=</code>	Not Equal
<code>></code>	Greater than
<code><</code>	Less than
<code>>=</code>	Greater or equal
<code><=</code>	Less or equal

Comparison Operators

>_

```
$ node_network_flags
```

node_network_flags{device="enp0s3", instance="node1", job="node"}	5000
node_network_flags{device="enp0s3", instance="node2", job="node"}	4800
node_network_flags{device="lo", instance="node1", job="node"}	77
node_network_flags{device="lo", instance="node2", job="node"}	84

Filter results for anything **greater** than 100

```
$ node_network_flags > 100
```

node_network_flags{device="enp0s3", instance="node1", job="node"}	5000
node_network_flags{device="enp0s3", instance="node2", job="node"}	4800

Comparison Operators

>_

```
$ node_network_receive_packets_total
```

node_network_receive_packets_total{device="enp0s3", instance="node1", job="node"}	542
node_network_receive_packets_total{device="lo", instance="node1", job="node"}	220
node_network_receive_packets_total{device="enp0s3", instance="node2", job="node"}	8
node_network_receive_packets_total{device="lo", instance="node2", job="node"}	2

Filter all interfaces that have received 220 or greater network packets

```
$ node_network_receive_packets_total >= 220
```

node_network_receive_packets_total{device="enp0s3", instance="node1", job="node"}	542
node_network_receive_packets_total{device="lo", instance="node1", job="node"}	220

Comparison Operators

>_

```
$ node_filesystem_avail_bytes
```

node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}	53371
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/"}	18771
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run"}	421
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock"}	80012
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns"}	872

Bool Operator can be used to return a **true(1)** or **false(0)** result. To find which filesystems have less than 1000 bytes available:

```
$ node_filesystem_avail_bytes < bool 1000
```

node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1", mountpoint="/boot/efi"}	0
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1", mountpoint="/"}	0
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run"}	1
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/lock"}	0
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1", mountpoint="/run/snapd/ns"}	1

Bool Operators are mostly used for generating **alerts**

Less than 1000 bytes available

Binary operator precedence

When an PromQL expression has multiple binary operators, they follow an order of **precedence**, from highest to lowest:

1. \wedge
2. $*$, $/$, $\%$, atan2
3. $+$, $-$
4. $==$, $!=$, $<=$, $<$, $>=$, $>$
5. `and`, `unless`
6. `or`

Operators on the same precedence level are **left-associative**.

For example, $2 * 3 \% 2$ is equivalent to $(2 * 3) \% 2$.

However \wedge is right associative, so $2 \wedge 3 \wedge 2$ is equivalent to $2 \wedge (3 \wedge 2)$

Logical Operators



PromQL has 3 logical operators



OR



AND



UNLESS

AND Operator

>_

```
$ node_filesystem_avail_bytes
```

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 53371
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/var"} 1771
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/etc"} 421
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 80012
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 2872

Return all time series greater than 1000 **and** less than 3000

```
$ node_filesystem_avail_bytes > 1000 and node_filesystem_avail_bytes < 3000
```

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/var"} 1771
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 2872

OR Operator

>_

```
$ node_filesystem_avail_bytes
```

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"}	53371
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/var"}	18771
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/etc"}	421
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/opt"}	80012

Return all time series less than 500 **or** greater than 70000

```
$ node_filesystem_avail_bytes < 500 or node_filesystem_avail_bytes > 70000
```

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/etc"}	421
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/opt"}	80012

UNLESS Operator

>_

Unless operator results in a vector consisting of elements on the left side for which there are no elements on the right side

\$ node_filesystem_avail_bytes

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 53371
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/var"} 1771
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/etc"} 421
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 80012
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 2872

Return all vectors greater than 1000 unless they are greater than 30000

\$ node_filesystem_avail_bytes > 1000 unless node_filesystem_avail_bytes > 30000

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/var"} 1771
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 2872



KodeKloud

vector matching

Vector Matching

Operators between and instant vectors and scalars

```
$ node_filesystem_avail_bytes < 1000
```

Vector Matching

Operators between and 2 instant vectors

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"}	512
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/var"}	484
node_filesystem_size_bytes{instance="node1", job="node", mountpoint="/home"}	1024
node_filesystem_size_bytes{instance="node1", job="node", mountpoint="/var"}	2048

```
$ node_filesystem_avail_bytes / node_filesystem_size_bytes * 100
```

{instance="node1", job="node", mountpoint="/home"}	50
{instance="node1", job="node", mountpoint="/var"}	23.6328125

Vector Matching



One-To-One



One-To-Many/Many-To-One

Vector Matching

Samples with exactly the **same** labels get matched together

```
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} 512  
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/var"} 484  
node_filesystem_size_bytes{instance="node1", job="node", mountpoint="/home"} 1024  
node_filesystem_size_bytes{instance="node1", job="node", mountpoint="/var"} 2048
```

```
$ node_filesystem_avail_bytes / node_filesystem_size_bytes * 100
```

Vector Matching

All labels **must** be same for samples to match

```
$ node_filesystem_avail_bytes / node_filesystem_size_bytes * 100
```

node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} node_filesystem_size_bytes{instance="node2", job="node", mountpoint="/home"}	512 1024	No match
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home"} node_filesystem_size_bytes{instance="node1", job="node", mountpoint="/var"}	512 1024	No match
node_filesystem_avail_bytes{instance="node1", job="node", mountpoint="/home", device="/dev/sda1"} node_filesystem_size_bytes{instance="node1", job="node", mountpoint="/var"}	512 1024	No match

Vector Matching

There may be certain instances where an operation needs to be performed on 2 vectors with **differing** labels

```
$ http_errors
```

http_errors{method="get", code="500"}	40
http_errors{method="get", code="404"}	77
http_errors{method="put", code="501"}	23
http_errors{method="post", code="500"}	61
http_errors{method="post", code="404"}	42

```
$ http_requests
```

http_requests{method="get"}	421
http_requests{method="del"}	288
http_requests{method="post"}	372

2 labels: {method, code}

1 label: {method}

```
$ http_errors{code="500"} / http_requests
```



No match

Vector Matching Keywords

Ignoring keyword can be used to “ignore” labels to ensure there is a match between 2 vectors

```
$ http_errors
```

http_errors{method="get", code="500"} 40
http_errors{method="put", code="501"} 23
http_errors{method="post", code="500"} 60

```
$ http_requests
```

http_requests{method="get"} 421
http_requests{method="del"} 288
http_requests{method="post"} 372

```
$ http_errors{code="500"} / ignoring(code) http_requests
```

{method="get"} 0.0950 // 40 / 421
{method="post"} 0.1612 // 60 / 372

The entries with methods put and del have no match and will not show up in the results

Vector Matching Keywords

Ignoring keyword is used to ignore a label when matching, the **on** keyword is to specify exact list of labels to match on

```
$ http_errors
```

http_errors{method="get", code="500"}	40
http_errors{method="get", code="404"}	77
http_errors{method="put", code="501"}	23
http_errors{method="post", code="500"}	61
http_errors{method="post", code="404"}	42

```
$ http_requests
```

http_requests{method="get"}	421
http_requests{method="del"}	288
http_requests{method="post"}	372

```
$ http_errors{code="500"} / ignoring(code) http_requests
```

or

```
$ http_errors{code="500"} / on(method) http_requests
```

List of all labels to match
on

Vector Matching Keywords

vector1

```
{cpu=0, mode=idle} 4  
{cpu=1, mode=iowait} 7  
{cpu=2, mode=user} 2
```

+

vector2

```
{cpu=1, mode=steal} 4  
{cpu=2, mode=user} 7  
{cpu=0, mode=idle} 2
```

Doesn't exist
in vector1

=

```
{cpu=0} 6  
{cpu=1} 11  
{cpu=2} 9
```

Vector1{}

+ on(cpu) Vector2{}

Vector1{}

+ ignoring(mode) Vector2{}

Resulting vector will
have matching elements
with all labels listed in
“on” or all labels NOT
listed in “ignoring”

One-To-One

One-To-One vector matching – Every element in the vector on the left of the operator tries to find a single matching element on the right

```
{cpu=0, mode=idle} 2  
{cpu=0, mode=user} 5  
{cpu=0, mode=user} 1  
{cpu=0, mode=user} 7
```

Vector1



```
{cpu=0, mode=idle} 4  
{cpu=0, mode=user} 6  
{cpu=0, mode=user} 3  
{cpu=0, mode=user} 3
```

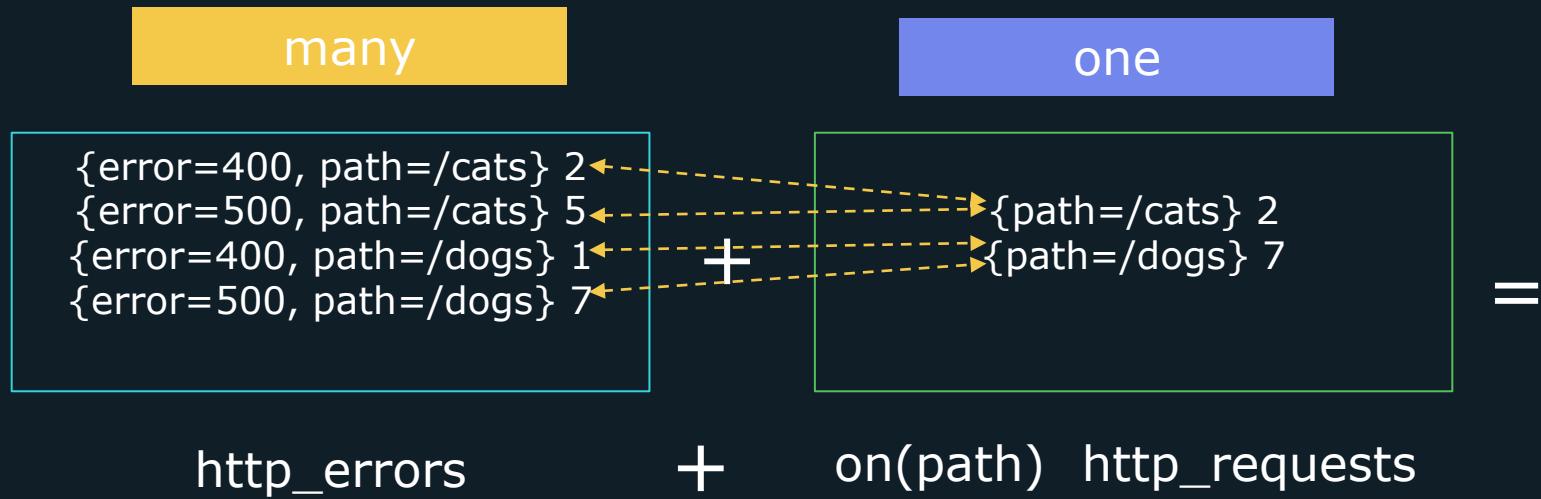
Vector2

=

```
{cpu=0, mode=idle} 6  
{cpu=0, mode=user} 11  
{cpu=0, mode=user} 4  
{cpu=0, mode=user} 10
```

Many-To-One

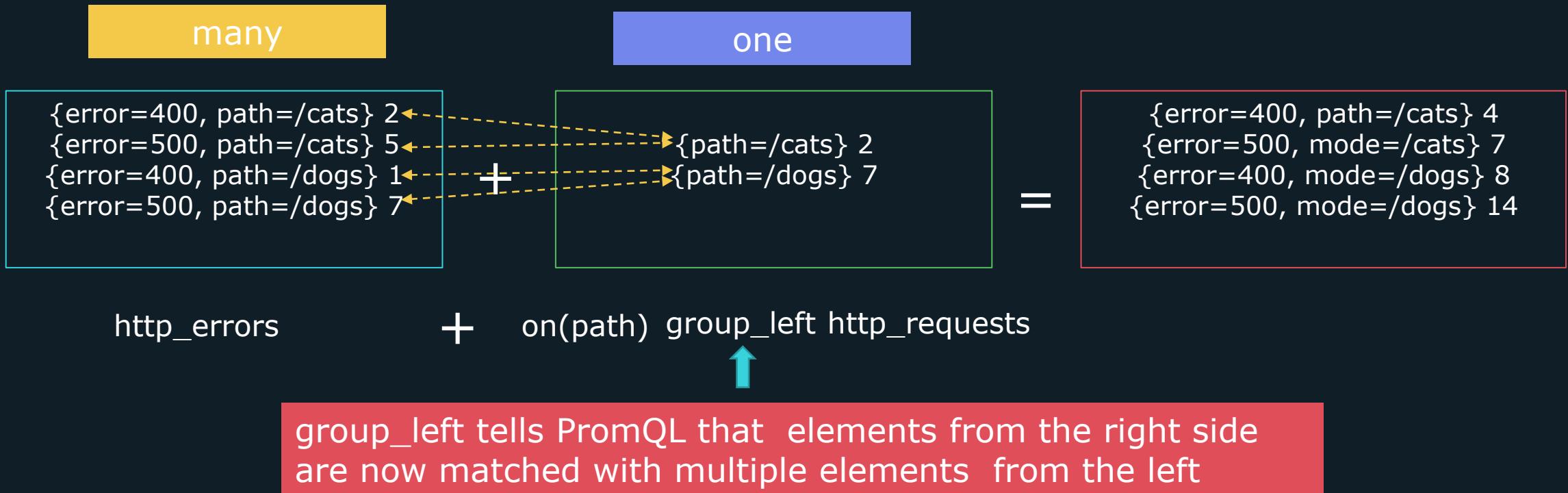
Many-To-One vector matching – each vector elements on the **one** side can match with multiple elements on the **many** side



Error executing query: multiple matches for labels:
many-to-one matching must be explicit
(group_left/group_right)

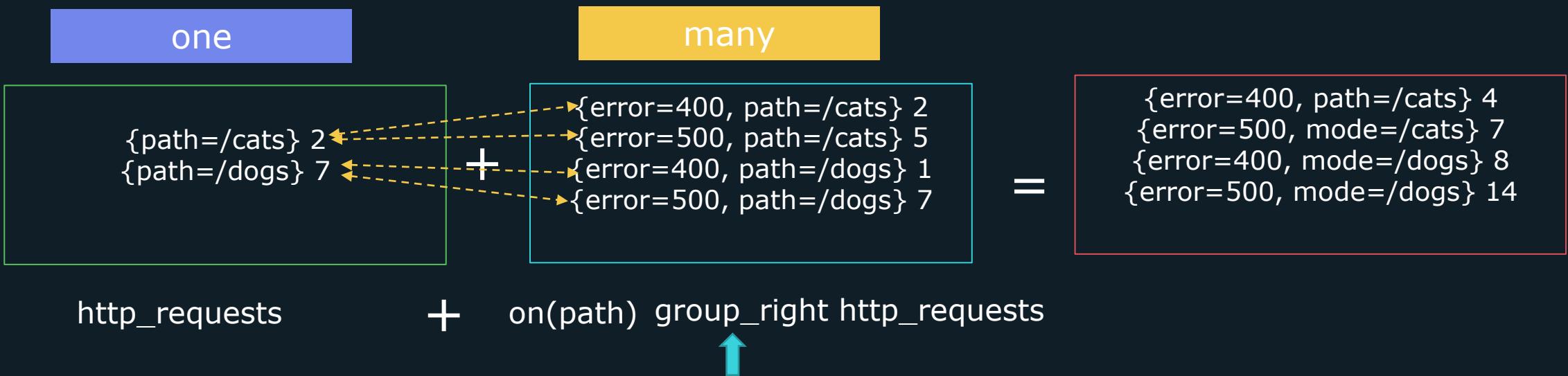
Many-To-One

Many-To-One is when each vector element on the one side can match with multiple elements on the many side



Many-To-One

Group_right is the opposite of group_left, tells PromQL that elements from the left side are now matched with multiple elements from the right





KodeKloud

Aggregation operators

Aggregation operators

Aggregation operators, allow you to take an instant vector and **aggregate** its elements, resulting in a new instant vector, with **fewer** elements

Aggregator	Description
Sum	Calculate sum over dimensions
Min	Select minimum over dimensions
Max	Select maximum over dimensions
Avg	Average over dimensions
Group	All values in the resulting vector are 1
Stddev	Calculate population standard deviations over dimensions
Stdvar	Calculate population standard variance over dimensions
Count	Count number of elements in the vector
Count_values	Count number fo elements with same value
Bottomk	Smallest k elements by sample value
Topk	Largest k elements by sample value
Quantile	calculate ϕ -quantile ($0 \leq \phi \leq 1$) over dimensions

Aggregation Operators

>_

\$ http_requests

```
http_requests{method="get", path="/auth"} 3
http_requests{method="post", path="/auth"} 1
http_requests{method="get", path="/user"} 4
http_requests{method="post", path="/user"} 8
http_requests{method="post", path="/upload"} 2
http_requests{method="get", path="/tasks"} 4
http_requests{method="put", path="/tasks"} 6
http_requests{method="post", path="/tasks"} 1
http_requests{method="get", path="/admin"} 3
http_requests{method="post", path="/admin"} 9
```

\$ sum(http_requests)

```
{} 41 // 3+1+4+8+2+4+6+1+3+9
```

\$ max(http_requests)

```
{} 9
```

\$ avg(http_requests)

```
{} 4.1
```

by clause

>_

```
$ http_requests
```

```
http_requests{method="get", path="/auth"} 3
http_requests{method="post", path="/auth"} 1
http_requests{method="get", path="/user"} 4
http_requests{method="post", path="/user"} 8
http_requests{method="post", path="/upload"} 2
http_requests{method="get", path="/tasks"} 4
http_requests{method="put", path="/tasks"} 6
http_requests{method="post", path="/tasks"} 1
http_requests{method="get", path="/admin"} 3
http_requests{method="post", path="/admin"} 9
```

The **by** clause allows you to choose which labels to aggregate along

```
$ sum by(path) (http_requests)
```

```
{path="/auth"} 4    // 3+1
{path="/user"} 12   // 4+8
{path="/upload"} 2   // 2
{path="/tasks"} 11   // 4+6+1
{path="/admin"} 12   // 3+9
```

Aggregation Operators

>_

\$ http_requests

```
http_requests{method="get", path="/auth"} 3
http_requests{method="post", path="/auth"} 1
http_requests{method="get", path="/user"} 4
http_requests{method="post", path="/user"} 8
http_requests{method="post", path="/upload"} 2
http_requests{method="get", path="/tasks"} 4
http_requests{method="put", path="/tasks"} 6
http_requests{method="post", path="/tasks"} 1
http_requests{method="get", path="/admin"} 3
http_requests{method="post", path="/admin"} 9
```

\$ sum by(method) (http_requests)

```
{method="get"} 14    // 3+4+4+3
{method="post"} 21   // 1+8+2+1+9
{method="put"} 6     // 6
```

Aggregation Operators

>_

\$ http_requests

```
http_requests{method="get", path="/auth", instance="node1"}      3
http_requests{method="post", path="/auth", instance="node1"}       1
http_requests{method="get", path="/user", instance="node1"}        4
http_requests{method="post", path="/user", instance="node1"}       8
http_requests{method="post", path="/upload", instance="node1"}    2
http_requests{method="get", path="/tasks", instance="node1"}       4
http_requests{method="put", path="/tasks", instance="node1"}       6
http_requests{method="post", path="/tasks", instance="node1"}      1
http_requests{method="get", path="/admin", instance="node1"}        3
http_requests{method="post", path="/admin", instance="node1"}      9
http_requests{method="get", path="/auth", instance="node2"}       13
http_requests{method="post", path="/auth", instance="node2"}      11
http_requests{method="get", path="/user", instance="node2"}       14
http_requests{method="post", path="/user", instance="node2"}      18
http_requests{method="post", path="/upload", instance="node2"}    12
http_requests{method="get", path="/tasks", instance="node2"}      14
http_requests{method="put", path="/tasks", instance="node2"}      16
http_requests{method="post", path="/tasks", instance="node2"}     11
http_requests{method="get", path="/admin", instance="node2"}      13
http_requests{method="post", path="/admin", instance="node2"}     19
```

\$ sum by(instance) (http_requests)

```
{instance="node1"} 41 // 3+1+4+8+2+4+6+1+3+9
{instance="node2"} 14 // 13+11+14+18+12+14+16+11+13+19
```

Aggregation Operators

>_

\$ http_requests

```
http_requests{method="get", path="/auth", instance="node1"}  
http_requests{method="post", path="/auth", instance="node1"}  
http_requests{method="get", path="/user", instance="node1"}  
http_requests{method="post", path="/user", instance="node1"}  
http_requests{method="post", path="/upload", instance="node1"}  
http_requests{method="get", path="/tasks", instance="node1"}  
http_requests{method="put", path="/tasks", instance="node1"}  
http_requests{method="post", path="/tasks", instance="node1"}  
http_requests{method="get", path="/admin", instance="node1"}  
http_requests{method="post", path="/admin", instance="node1"}  
http_requests{method="get", path="/auth", instance="node2"}  
http_requests{method="post", path="/auth", instance="node2"}  
http_requests{method="get", path="/user", instance="node2"}  
http_requests{method="post", path="/user", instance="node2"}  
http_requests{method="post", path="/upload", instance="node2"}  
http_requests{method="get", path="/tasks", instance="node2"}  
http_requests{method="put", path="/tasks", instance="node2"}  
http_requests{method="post", path="/tasks", instance="node2"}  
http_requests{method="get", path="/admin", instance="node2"}  
http_requests{method="post", path="/admin", instance="node2"}
```

\$ sum by(instance, method) (http_requests)

3	{instance="node1", method="get"} 14 // 3+4+4+3
1	{instance="node1", method="post"} 21 // 1+8+2+1+9
4	{instance="node1", method="put"} 6 // 6
8	{instance="node2", method="get"} 54 // 13+14+14+13
2	{instance="node2", method="post"} 71 // 11+18+12+11+19
4	{instance="node2", method="put"} 16 // 16
6	
1	
3	
9	
13	
11	
14	
18	
12	
14	
16	
11	
13	
19	

Aggregation Operator

The `without` keyword does the opposite of `by` and tells the query which labels not to include in the aggregation

```
$ http_requests
```

```
http_requests{method="get", path="/auth", instance="node1"}      3
http_requests{method="post", path="/auth", instance="node1"}       1
http_requests{method="get", path="/tasks", instance="node1"}       4
http_requests{method="put", path="/tasks", instance="node1"}       6
http_requests{method="post", path="/tasks", instance="node1"}      1
http_requests{method="get", path="/auth", instance="node2"}       13
http_requests{method="post", path="/auth", instance="node2"}       11
http_requests{method="get", path="/tasks", instance="node2"}       14
http_requests{method="put", path="/tasks", instance="node2"}       16
http_requests{method="post", path="/tasks", instance="node2"}      11
```

```
$ sum without(path) (http_requests)
```

```
{instance="node1", method="get"} 7 // 3+4
{instance="node1", method="post"} 2 // 1+1
{instance="node1", method="put"} 6 // 6
{instance="node2", method="get"} 27 // 13+14
{instance="node2", method="post"} 71 // 11+18+12+11+19
{instance="node2", method="put"} 22 // 11+11
```

Aggregate on every label except `path`. The equivalent to `by(instance, method)`



KodeKloud

Functions

PromQL has several different function for a variety of use cases including sorting, math, label transformation, metric manipulation and many more

Math Functions

>_

```
$ node_cpu_seconds_total
```

{cpu="0", mode="idle"}	115.12
{cpu="0", mode="irq"}	87.4482
{cpu="0", mode="steal"}	44.245

```
$ ceil(node_cpu_seconds_total)
```

{cpu="0", mode="idle"}	116
{cpu="0", mode="irq"}	88
{cpu="0", mode="steal"}	45

ceil rounds up to the closest integer

```
$ floor(node_cpu_seconds_total)
```

{cpu="0", mode="idle"}	115
{cpu="0", mode="irq"}	87
{cpu="0", mode="steal"}	44

floor rounds down to the closest integer

```
$ abs(1-node_cpu_seconds_total)
```

{cpu="0", mode="idle"}	115.12
{cpu="0", mode="irq"}	87.4482
{cpu="0", mode="steal"}	44.245

abs returns absolute value

Date & Time Functions

>_

```
$ time()  
1663872361.957
```

time function return current
time

```
$ time() - process_start_time_seconds  
process_start_time_seconds{instance="node"} 1109175.01
```

If you want to see how long
your processes have been
running.

Date & Time Functions

Prometheus provides functions to provide users information on what day of the week it is, what month, year etc.

If the current time is 15:07 Thursday September 22nd, 2022 UTC timezone

Expression	result
Minute()	{}07
Hour()	{}15
Day_of_week()	{}4
Day_of_month()	{}22
Days_in_month()	{}30
Month()	{}09
Year()	{}2022

Changing type

>_

The vector function takes a scalar value and converts it into an instant vector

```
$ vector(4)  
{} 4
```

Changing type

>_

Given a single element input vector, scalar function returns the sample value of the single element as a **scalar**. If the input vector does not have exactly one element, it returns a **NaN**

```
$ process_start_time_seconds
```

```
process_start_time_seconds{instance="node1"} 1662763800
```

```
$ scalar(process_start_time_seconds)
```

```
scalar 1662763800
```

```
$ node_cpu_seconds_total
```

```
node_cpu_seconds_total{instance="node1"} 228  
node_cpu_seconds_total{instance="node2"} 482
```

```
$ scalar(node_cpu_seconds_total)
```

```
scalar NaN
```

Sorting Functions

>_

Elements can be sorted with sort and sort_desc

```
$ sort(node_filesystem_avail_bytes)
```

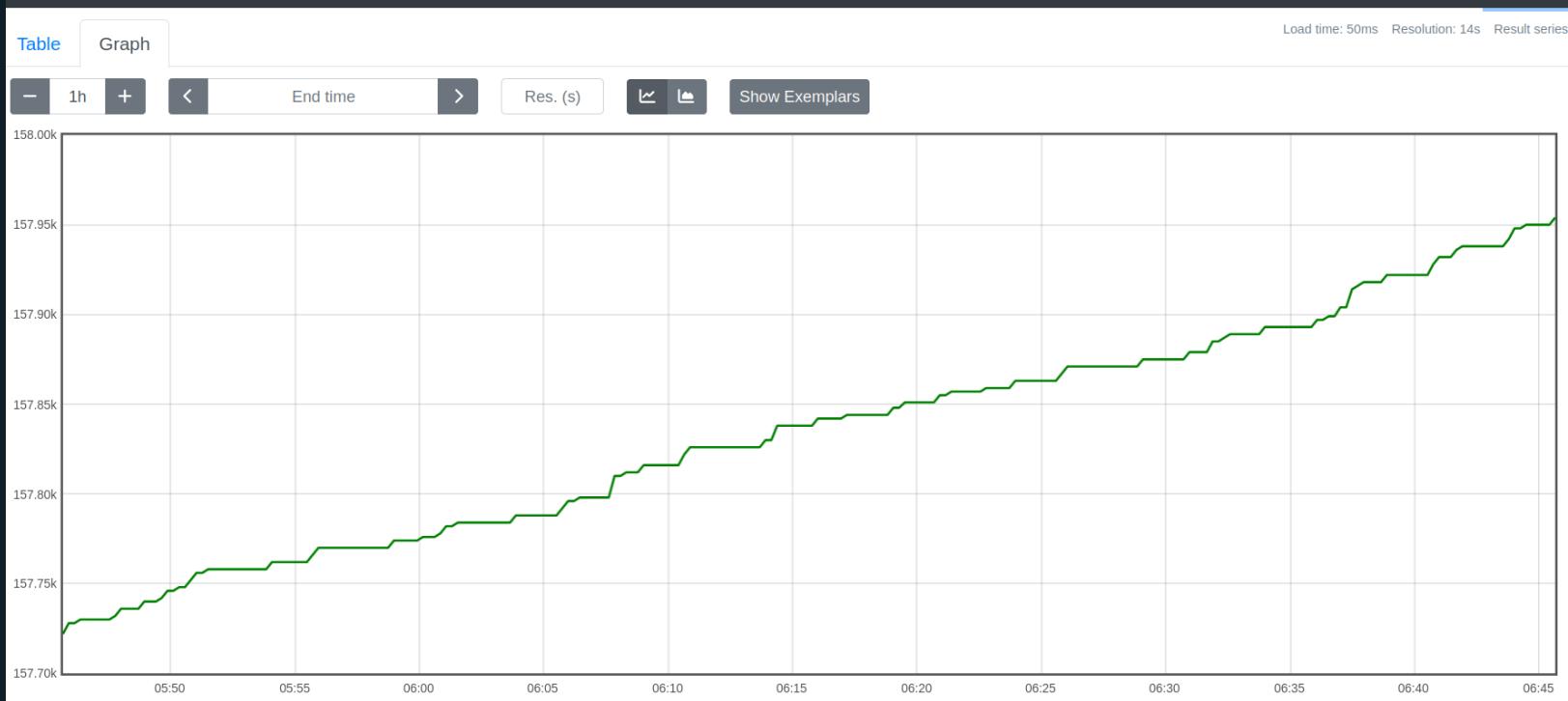
node_filesystem_avail_bytes{device="gvfsd-fuse", fstype="fuse.gvfsd-fuse", instance="node1"}	0
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	5238784
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1"}	531341312
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	725422080
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	726319104
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	726319104
node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1"}	1784819712

```
$ sort_desc(node_filesystem_avail_bytes)
```

node_filesystem_avail_bytes{device="/dev/sda3", fstype="ext4", instance="node1"}	1784819712
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	726319104
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	726319104
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	725422080
node_filesystem_avail_bytes{device="/dev/sda2", fstype="vfat", instance="node1"}	531341312
node_filesystem_avail_bytes{device="tmpfs", fstype="tmpfs", instance="node1"}	5238784
node_filesystem_avail_bytes{device="gvfsd-fuse", fstype="fuse.gvfsd-fuse", instance="node1"}	0

Rate

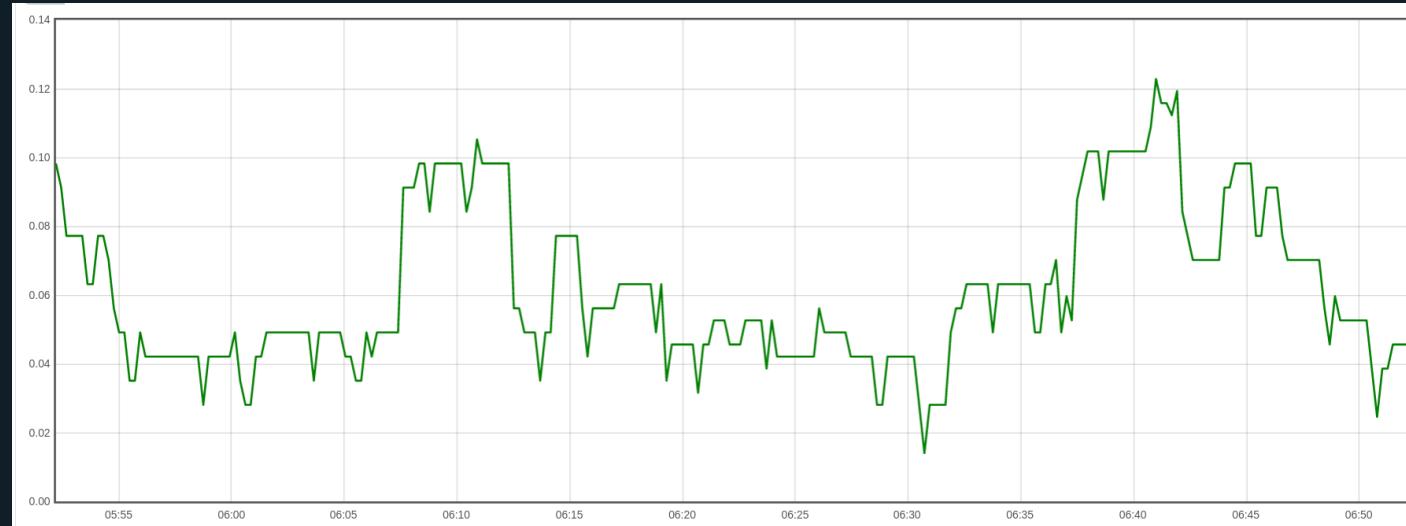
A plot of a counter metric does not show anything useful. We expect counters to **increase** over time.



Rate

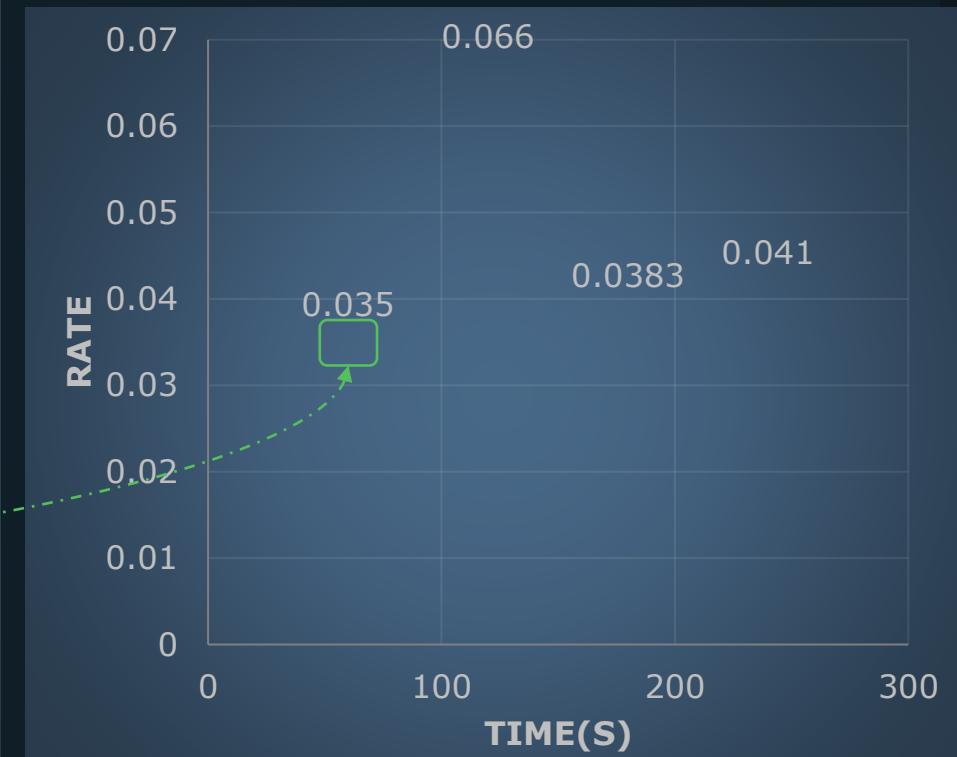
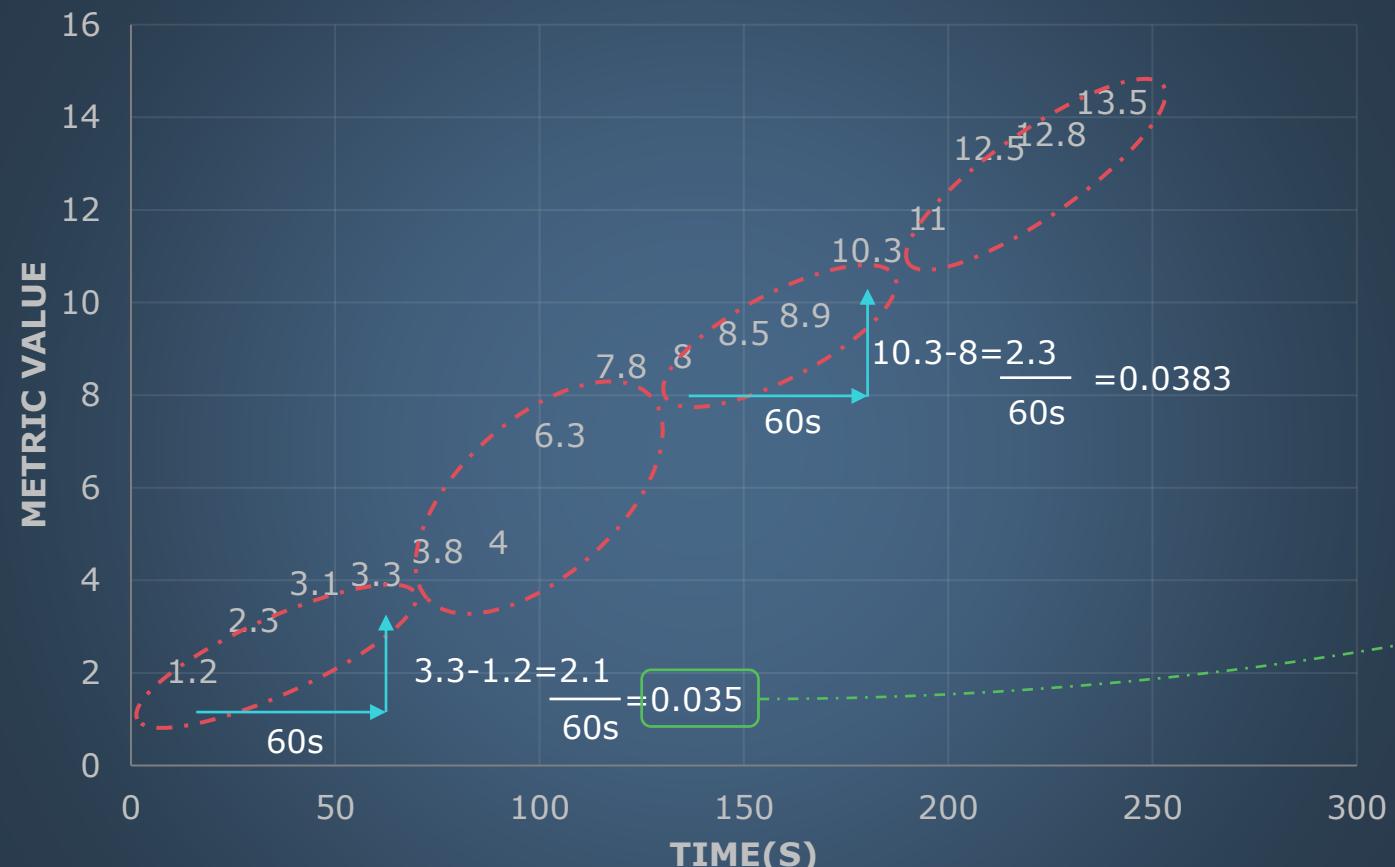
Instead we are more interested in the rate at which a counter metric increases.

The `rate()` and `irate()` function provides the per second average rate of change



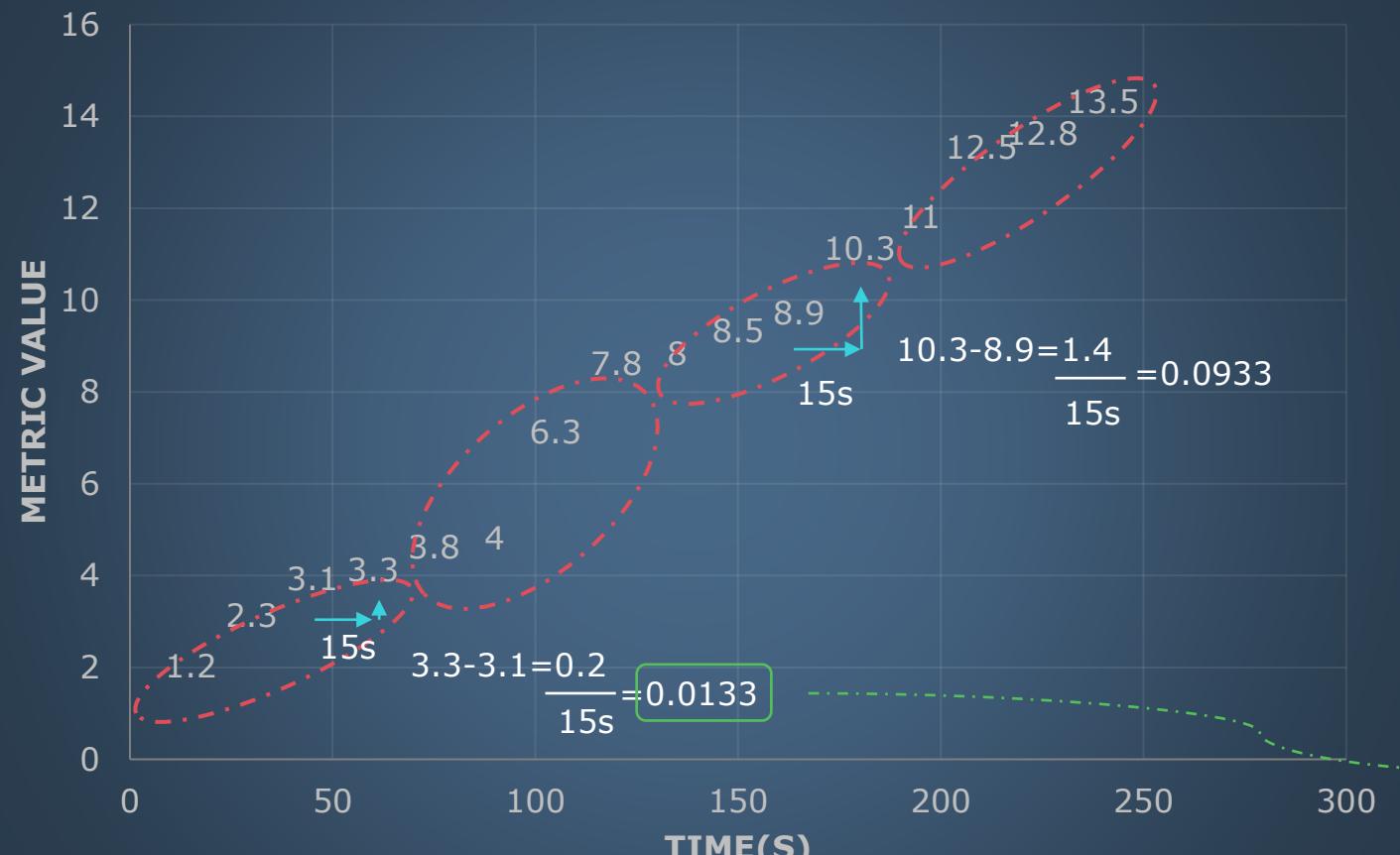
rate

```
$ rate(http_errors[1m])
```



irate()

```
$ irate(http_errors[1m])
```



rate vs irate



rate

- Looks at the first and last data points within a range
- Effectively an average rate over the range
- Best used for slow moving counters and alerting rules



irate

- Looks at the last two data points within a range
- Instant rate
- Should be used for graphing volatile, fast-moving counters

- Make sure there is **atleast** 4 samples within a given time range, ideally more.
 - 15s scrape interval 60s window gives 4 samples
- When combining rate with an aggregation operator, always take **rate()** **first**, then aggregate
 - This is so rate() function can detect counter resets
 - `$ sum without(code,handler)(rate(http_requests_total[24h]))`



KodeKloud

Subqueries

Subqueries

maximum value over a 10m of a gauge metric

```
$ max_over_time(node_filesystem_avail_bytes[10m])
```

For counter metrics we need to find the maximum value of the rate over the past 10 minutes

```
$ max_over_time(rate(http_requests_total[10m]))
```

↑
Expects a range vector

↑
Returns an instant vector



Subquery format

```
$ <instant_query> [<range>:<resolution>] [offset <duration>]
```

```
$ rate(http_requests_total[1m]) [5m:30s]
```

1m – sample range

5m – query range(get data from last 5 minutes)

30s – query step for subquery

```
$ max_over_time(rate(http_requests_total[1m]) [5m:30s])
```

Maximum rate of requests from the last 5 minutes with a 30s query interval and a sample range of 1m

subqueries

```
$ rate(node_cpu_seconds_total[1m]) [2m:10s]  
0.991777777804683 @1664226130  
0.991777777804683 @1664226140  
0.991333333304358 @1664226150  
0.991111111131807 @1664226160  
0.991111111131807 @1664226170  
0.991555555528651 @1664226180  
0.989555555561764 @1664226190  
0.989555555561764 @1664226200  
0.989333333337472 @1664226210  
0.989333333337472 @1664226220  
0.989333333337472 @1664226230  
0.991111111131807 @1664226240
```

} Sample every 10s

{ 2 minute range



KodeKloud

Histogram

Histogram

```
request_latency_seconds_bucket{le="+Inf", path="/articles"} 100
request_latency_seconds_bucket{le="0.01", path="/articles"} 5
request_latency_seconds_bucket{le="0.02", path="/articles"} 17
request_latency_seconds_bucket{le="0.03", path="/articles"} 30
request_latency_seconds_bucket{le="0.04", path="/articles"} 44
request_latency_seconds_bucket{le="0.05", path="/articles"} 50
request_latency_seconds_bucket{le="0.06", path="/articles"} 64
request_latency_seconds_bucket{le="0.07", path="/articles"} 72
request_latency_seconds_bucket{le="0.08", path="/articles"} 79
request_latency_seconds_bucket{le="0.09", path="/articles"} 86
request_latency_seconds_bucket{le="0.1", path="/articles"} 100
request_latency_seconds_count{path="/articles"} 100
request_latency_seconds_sum{path="/articles"} 5.034496069
```

Total number of samples

Sum of all of the samples

Multiple buckets with a label le which contains all samples whose values are less than or equal to the value of the le label

Buckets are cumulative: all requests in the le="0.03" will include all requests less than 0.03s which includes all requests that fall into the buckets below it(le="0.02", le="0.01")

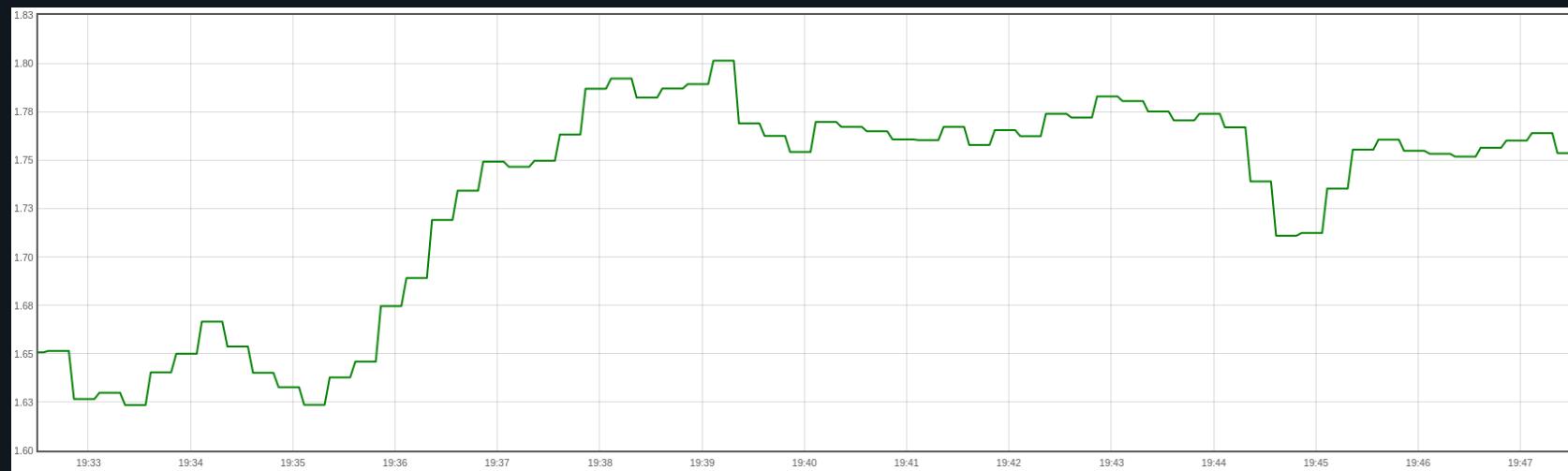
le= "+Inf" is a count for all samples less than positive infinity which in this case is the same as the _count. However in some cases they can be different because in histograms samples can contain negative values as well

Histogram

>_

For the `requests_latency_seconds_count` metric, we don't really care about the number as it's a counter which means it should go up with time. Instead we want to get the rate

```
$ rate(requests_latency_seconds_count[1m]) [5m:30s]
```

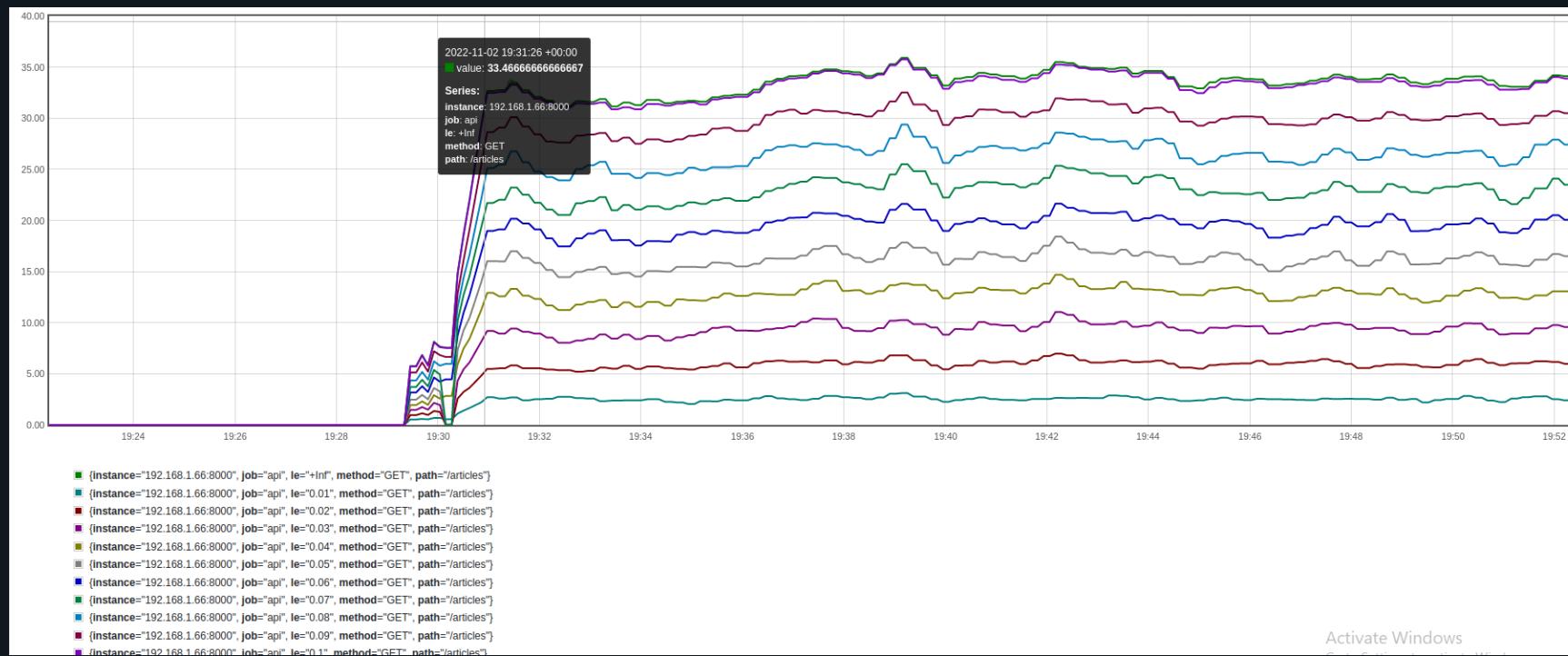


Histogram

>_

We can get the rate of requests for each bucket as well

\$ rate(requests_latency_seconds_bucket[1m]) [5m:30s]



Histogram

>_

To get the total sum of latency across all requests:

```
$ requests_latency_seconds_sum
```

Not necessarily that useful for us in this case. However if we have a metric to track size of files being uploaded the sum metric would be more useful as it would tell us total bytes that have been uploaded

To get the rate of increase of the sum of latency across all requests:

```
$ rate(requests_latency_seconds_sum[1m])
```

To calculate the average latency of a request over the past 5m

```
$ rate(requests_latency_seconds_sum[5m]) / rate(requests_latency_seconds_count[5m])
```

Histogram

>_

To calculate percentage of requests that fall into a specific bucket

```
$ rate(request_latency_seconds_bucket{path="/articles", le="0.06"}[1m]) /  
ignoring(le) rate(request_latency_seconds_count{path="/articles"}[1m])
```

Histogram

>_

To calculate the number of observations between two buckets

```
$ request_latency_seconds_bucket{path="/articles", le="0.06"} - ignoring(le)  
request_latency_seconds_bucket{path="/articles", le="0.03"}
```

Quantiles - determine how many values in a distribution are above or below a certain limit

Quantiles represent percentiles.

90% quantile would mean at what value is 90 percent of the data less than

Quantile

>_

Histograms have `histogram_quantile()` function to easily calculate quantile values

percentile Histogram metric

```
$ histogram_quantile(0.75, request_latency_seconds_bucket)  
{instance="192.168.1.66:8000", job="api", method="GET", path="/articles"} .076
```

75% of all requests had a latency of 0.076s or less

Histogram **quantiles** can be used to accurately measure if a specific SLO is being met and can be used to generate alerts if a SLO is exceeded

For example, we may have an SLO that states 95% of requests cannot exceed 0.5s we can measure that easily with

```
$ histogram_quantile(0.95, request_latency_seconds_bucket)
```

If the value returned is greater than 0.5s than we know that the SLO was missed

The `histogram_quantile()` function makes an **approximation** of the value of a specific quantile by using linear interpolation

This can be a problem when we are measuring SLOs as we want an accurate value

To get an accurate value make sure there is a bucket at the **specific** SLO value that needs to be met

For example, if the SLO is 95% of all requests must have a latency below 0.5s then we would need to make sure one of the buckets is set to 0.5

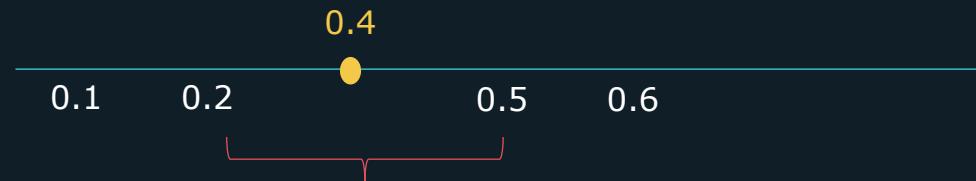
This would accurately tell us whether or not 95% of requests were below the 0.5s mark

Quantiles

However its important to understand that it would not accurately tell us by how much we missed it or how close we were

So we wouldn't be able to accurately say if its 0.3 or 0.47(which would be very close to the SLO)

For example if we have buckets are 0.1, 0.2, and 0.5 and the quantile reports a value of 0.4, then the actual value can be anywhere between the two buckets of 0.2 and 0.5



If you wanted more accurate value, more buckets would need to be added between 0.2 and 0.5

Each histogram bucket is stored as a separate time series, so having too many buckets will result:

- High Cardinality
- High Ram usage
- High disk space
- Slower performance for inserts in Prometheus database



KodeKloud

Summary

Summary

```
>_  
  
request_latency_seconds{path="/articles", quantile="0.01"} 0.004  
request_latency_seconds{path="/articles", quantile="0.1"} 0.0114  
request_latency_seconds{path="/articles", quantile="0.2"} 0.0156  
request_latency_seconds{path="/articles", quantile="0.3"} 0.02  
request_latency_seconds{path="/articles", quantile="0.4"} 0.0252  
request_latency_seconds{path="/articles", quantile="0.5"} 0.0296  
request_latency_seconds{path="/articles", quantile="0.6"} 0.0333  
request_latency_seconds{path="/articles", quantile="0.7"} 0.0365  
request_latency_seconds{path="/articles", quantile="0.8"} 0.0442  
request_latency_seconds{path="/articles", quantile="0.9"} 0.05325  
request_latency_seconds{path="/articles", quantile="0.95"} 0.06  
request_latency_seconds_count{path="/articles"} 100  
request_latency_seconds_sum{path="/articles"} 3.144
```

Total number of samples

Sum of all of the samples

Multiple buckets with a label percentile which reports what percentage of samples were fell below the given value

```
$ request_latency_seconds{quantile="0.5"}
```

Histogram vs Summary

Histogram

- Bucket sizes can be picked
- Less taxing on client libraries
- Any quantile can be selected
- Prometheus server must calculate quantiles

Summary

- Quantile must be defined ahead of time
- More taxing on client libraries
- Only quantiles predefined in client can be used
- Very minimal server-side cost

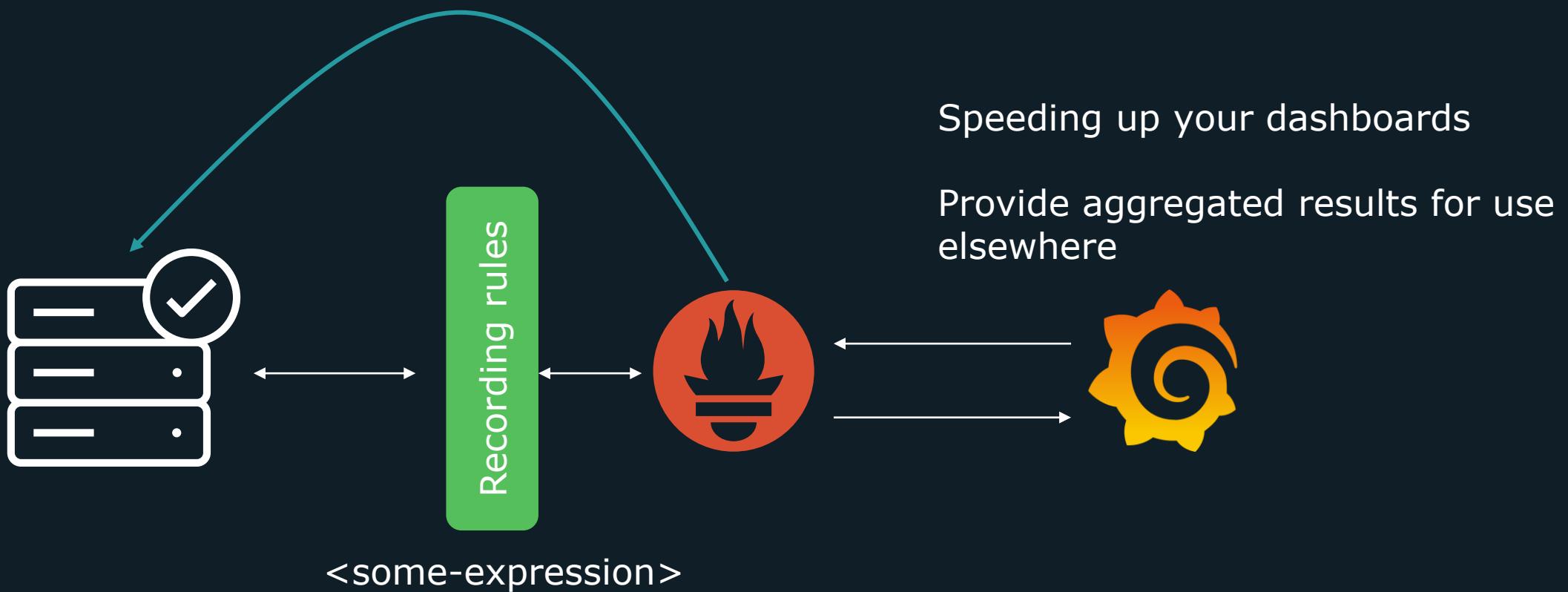


KodeKloud

Recording Rules

Recording Rules

Recording Rules allow Prometheus to periodically **evaluate** PromQL expression and **store** the resulting times series generated by them



Recording rules go in a separate file called a rule files

```
>_      Prometheus.yml

global:
  scrape_interval: 5s
  evaluation_interval: 5s
rule_files:
  - rules.yml
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets:
          - localhost:9090
```

Globs can be used to match file-names:
/etc/Prometheus/rules/*.yml

Changes in rule files is not automatically
picked up by Prometheus, so restarting
Prometheus is required

Rule File

>_

rule.yml

```
[groups]
- name: <group name 1>
  interval: <evaluation Interval>
  rules:
    - record: <rule name 1>
      expr: <promql expression 1>
      labels:
        - <label name>: <label value>
    - record: <rule name 2>
      expr: <promql expression 2>
      labels:
- name: <group name 2>
  rules:
```

Each file defines one or more rule groups under the **groups** key

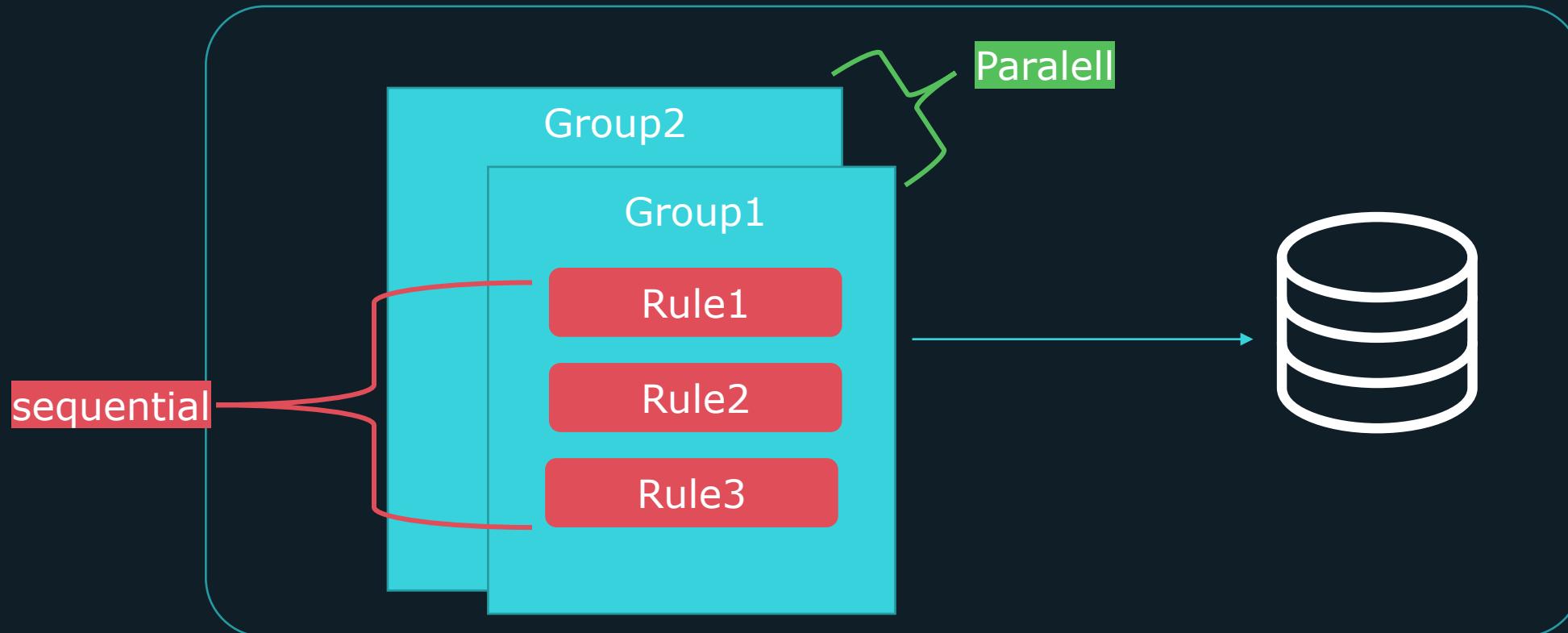
Evaluation interval – defaults to the global evalution interval

Each rule instructs Prometheus to evaluate the PromQL expression defined in **expr**

The label field can optionally add or remove labels before storing the result

Rules in a group will be evaluated sequentially in the order they are declared

Rules



Recording Rule example

Percentage of free memory available on a node

```
$ 100 - (100 * node_memory_MemFree_bytes/node_memory_MemTotal_bytes)
```

Percentage of free space available on filesystem

```
$ 100 * node_filesystem_free_bytes/node_filesystem_avail_bytes
```

If we didn't want to manually run this expression everytime,
we can setup a recording rule to periodically gather this
information

Recording Rule Example

```
>_          rule.yml

groups:
  - name: example1
    interval: 15s
    rules:
      - record: node_memory_memFree_percent ←
        expr: 100 - (100 * node_memory_MemFree_bytes / ←
        node_memory_MemTotal_bytes)
      - record: node_filesystem_free_percent
        expr: 100 * node_filesystem_free_bytes / ←
        node_filesystem_size_bytes
```

Recording Rules Example

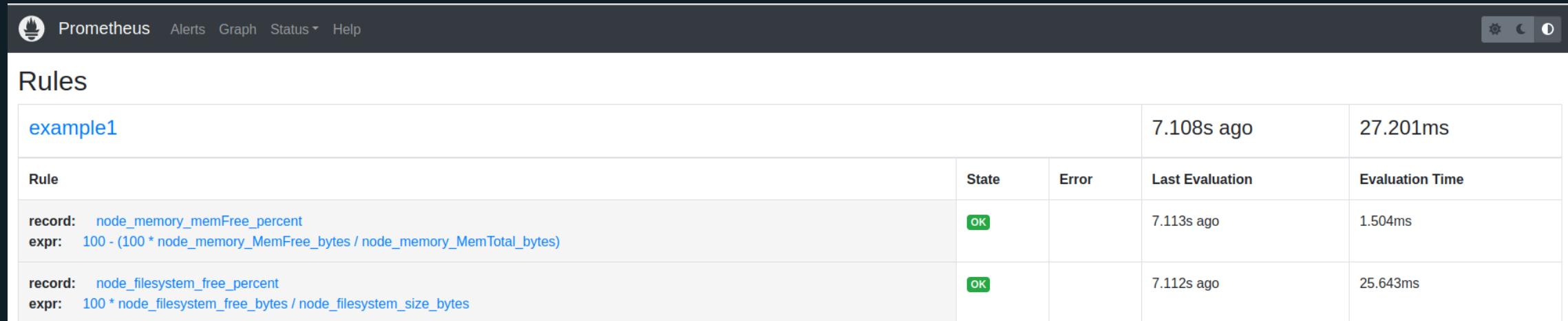
The screenshot shows the Prometheus web interface with the "Rules" page selected. A context menu is open over a specific rule named "example1". The menu items are:

- Runtime & Build Information
- TSDB Status
- Command-Line Flags
- Configuration
- Rules (highlighted with a red dashed box)
- Targets
- Service Discovery

The rule "example1" is defined with the following configuration:

```
record: node_memory_memFree_p  
expr: 100 - (100 * node_memory_M
```

Recording Rules Example



The screenshot shows the Prometheus web interface with the 'Rules' section selected. There are two rules listed:

Rule	State	Error	Last Evaluation	Evaluation Time
example1	OK		7.108s ago	27.201ms
record: node_memory_memFree_percent expr: 100 - (100 * node_memory_MemFree_bytes / node_memory_MemTotal_bytes)	OK		7.113s ago	1.504ms
record: node_filesystem_free_percent expr: 100 * node_filesystem_free_bytes / node_filesystem_size_bytes	OK		7.112s ago	25.643ms

>_

```
$ node_memory_memFree_percent
```

```
node_memory_memFree_percent{instance="192.168.1.168:9100", job="node"}      25
```

```
$ node_filesystem_free_percent
```

```
node_filesystem_free_percent{device="/dev/sda2", instance="node1", job="node", mountpoint="/boot/efi"}      20
node_filesystem_free_percent{device="/dev/sda3", instance="node1", job="node", mountpoint="/"}            72
node_filesystem_free_percent{device="tmpfs", instance="node1", job="node", mountpoint="/run"}           44
node_filesystem_free_percent{device="tmpfs", instance="node1", job="node", mountpoint="/run/lock"}        12
node_filesystem_free_percent{device="tmpfs", instance="node1", job="node", mountpoint="/run/snapd/ns"}   33
node_filesystem_free_percent{device="tmpfs", instance="node1", job="node", mountpoint="/run/user/1000"}  97
```

Rules

```
>_          rule.yml

groups:
  - name: example1
    interval: 15s
    rules:
      - record: node_filesystem_free_percent
        expr: 100 * node_filesystem_free_bytes{job="node"} /
node_filesystem_size_bytes{job="node"}

      - record: node_filesystem_free_percent_avg
        expr: avg by(instance)(node_filesystem_free_percent)
```

level:metric_name:operations

Level – indicates the aggregation level of the metric by the labels it has. This will always include the “job” label + any other relevant target labels

Metric_name – metric/time-series name

Operations – list of functions & aggregators that have been applied to the metric(i.e. sum/max/avg)

Record Rule Naming

Let's say we have a http_errors counter with two instrumentation labels "method" and "path"

- record: job_method_path:http_errors:rate5m
expr: sum without(instance) (rate(http_errors{job="api"}[5m]))

Since both the method and path labels are used, the aggregation level is
job_method_path

The metric name is **http_errors**

The operations that are performed is **rate** & **[5m]**

- record: job_method:http_errors:rate5m
expr: sum without(instance, path) (rate(http_errors{job="api"}[5m]))

Since the path label is removed using a without clause, the aggregation level is just **job_method**. If method is also removed them it would just be **job**

Best Practices

All the rules for a specific job should be contained in a single group

```
>_                               rule.yml

groups:
  - name: node # all the rules for job="node"
    interval: 15s
    rules:
      - record: node_memory_memFree_percent
        expr: 100 - (100 * node_memory_MemFree_bytes{job="node"} /
node_memory_MemTotal_bytes{job="node"})

      - name: docker # all the rules for job="docker"
        interval: 15s
        rules:
```



KodeKloud

HTTP API

Prometheus has an [HTTP API](#) that can be used to execute queries as well as gather information on alert, rules, and service-discovery related configs

This is useful for situations where using the built-in web gui isn't an option

- Building custom tools
- 3rd party integrations

To run a query through the API send a **POST** request to:

`http://<prometheus-ip>/api/v1/query`

The query is then passed into the **query** parameter

```
$ node_arp_entries{instance="192.168.1.168:9100"}
```

To send this query through the API:

```
$ curl <prometheus>:9090/api/v1/query  
--data 'query=node_arp_entries{instance="192.168.1.168:9100"}'
```

>_

Perform query at a specific time

```
$ curl localhost:9090/api/v1/query --data  
'query=node_arp_entries{instance="192.168.1.168:9100"}' --data 'time=1670380680.132'
```

Return a range Vector

```
$ curl localhost:9090/api/v1/query --data  
'query=node_arp_entries{instance="192.168.1.168:9100"}[5m]' --data  
'time=1670380680.132'
```

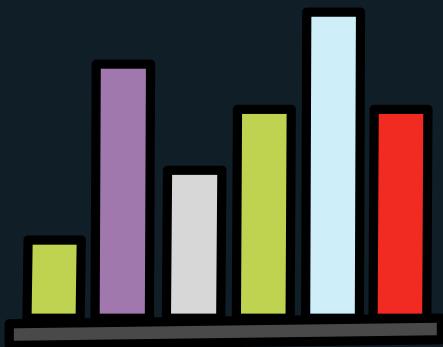


KodeKloud

Dashboarding & Visualization

Visualization

Visualization is all about visually representing Prometheus metrics using graphs, tables, charts, and dashboards



There are several different ways to **visualize** Prometheus metric data

Tools built into Prometheus:

- Expression browser
- Console Templates

3rd Party Tools

- Grafana



KodeKloud

Expression Browser

Expression Browser

Expression browser is a built in web UI that can be used to execute queries and simple graphs

Has limited functionality and should mainly be used for ad-hoc queries and quick debugging

Has no ability to build custom dashboards and is not ideal for day-to-day monitoring



KodeKloud

Console Templates

Console Templates

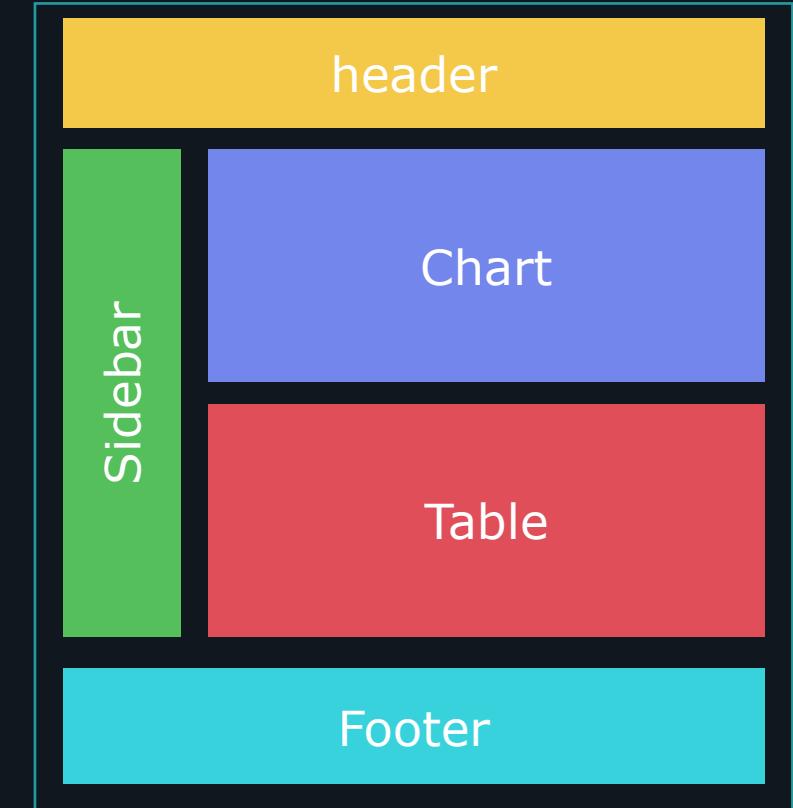
Console templates allow you to create your own custom html pages using Go templating language

Prometheus metrics, queries and charts can be embedded in the templates

Console Templates

>_

```
$ ls /etc/prometheus/consoles
drwxr-xr-x 2 user1 user1 4096 Oct  7 12:24 .
drwxr-xr-x 5 user1 user1 4096 Oct 17 22:23 ..
-rw-r--r-- 1 user1 user1  616 Oct  7 12:24 index.html.example
-rw-r--r-- 1 user1 user1 2675 Oct  7 12:24 node-cpu.html
-rw-r--r-- 1 user1 user1 3522 Oct  7 12:24 node-disk.html
-rw-r--r-- 1 user1 user1 1453 Oct  7 12:24 node.html
-rw-r--r-- 1 user1 user1 5783 Oct  7 12:24 node-overview.html
-rw-r--r-- 1 user1 user1 1334 Oct  7 12:24 prometheus.html
-rw-r--r-- 1 user1 user1 4103 Oct  7 12:24 prometheus-overview.html
```



Console Templates

>_

index.html.example

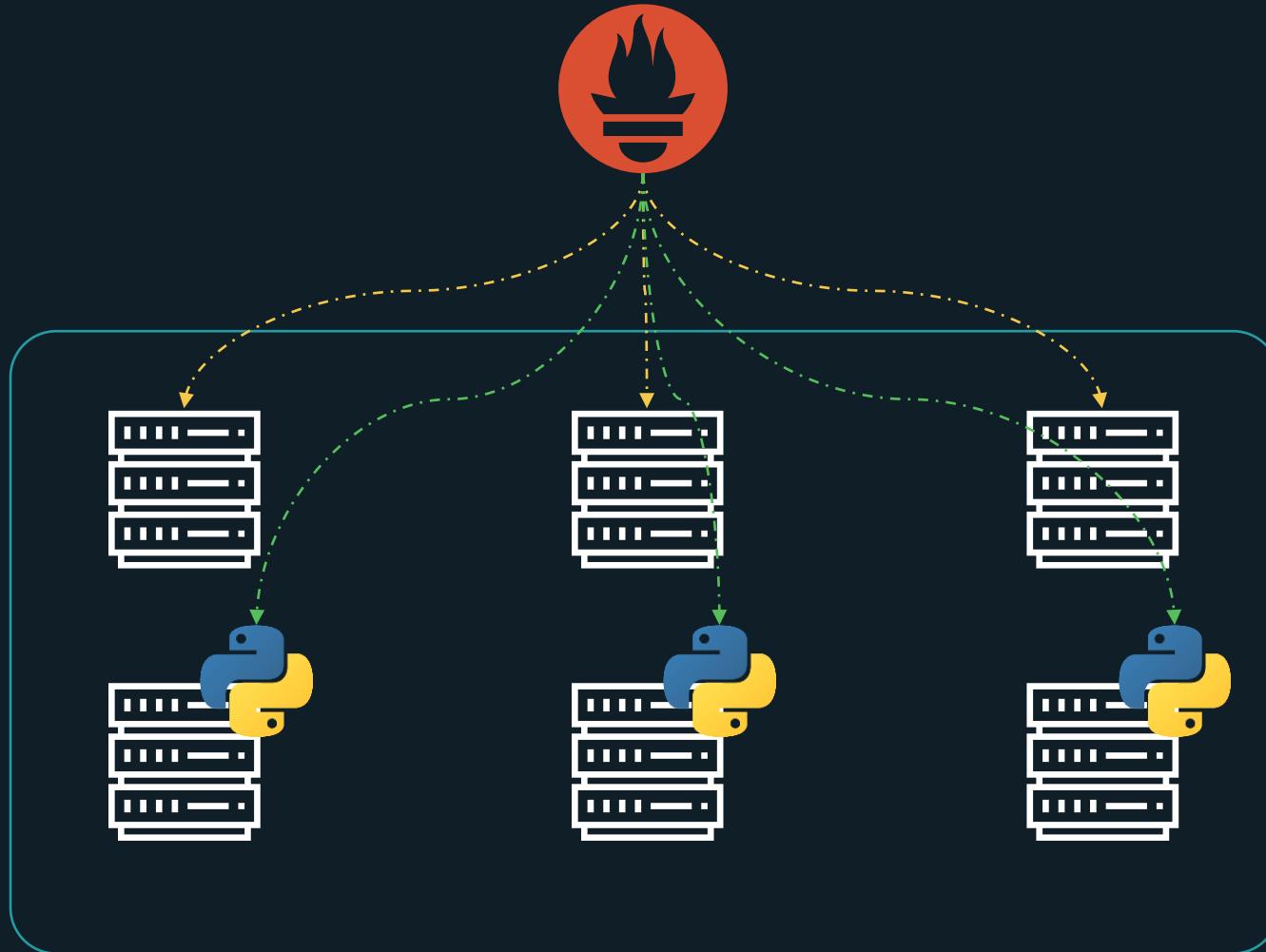
```
 {{ template "head" . }}  
 {{ template "prom_right_table_head" }}  
 {{ template "prom_right_table_tail" }}  
 {{ template "prom_content_head" . }}  
  
<h1>Overview</h1>  
<p>These are example consoles for Prometheus.</p>  
<p>These consoles expect exporters to have the following job labels:</p>  
  
<table class="table table-sm table-striped table-bordered" style="width: 0%">  
 <tr>  
   <th>Exporter</th>  
   <th>Job label</th>  
 </tr>  
 <tr>  
   <td>Node Exporter</td>  
   <td><code>node</code></td>  
 </tr>  
 <tr>  
   <td>Prometheus</td>  
   <td><code>prometheus</code></td>  
 </tr>  
</table>  
  
 {{ template "prom_content_tail" . }}  
 {{ template "tail" }}
```



KodeKloud

Instrumentation

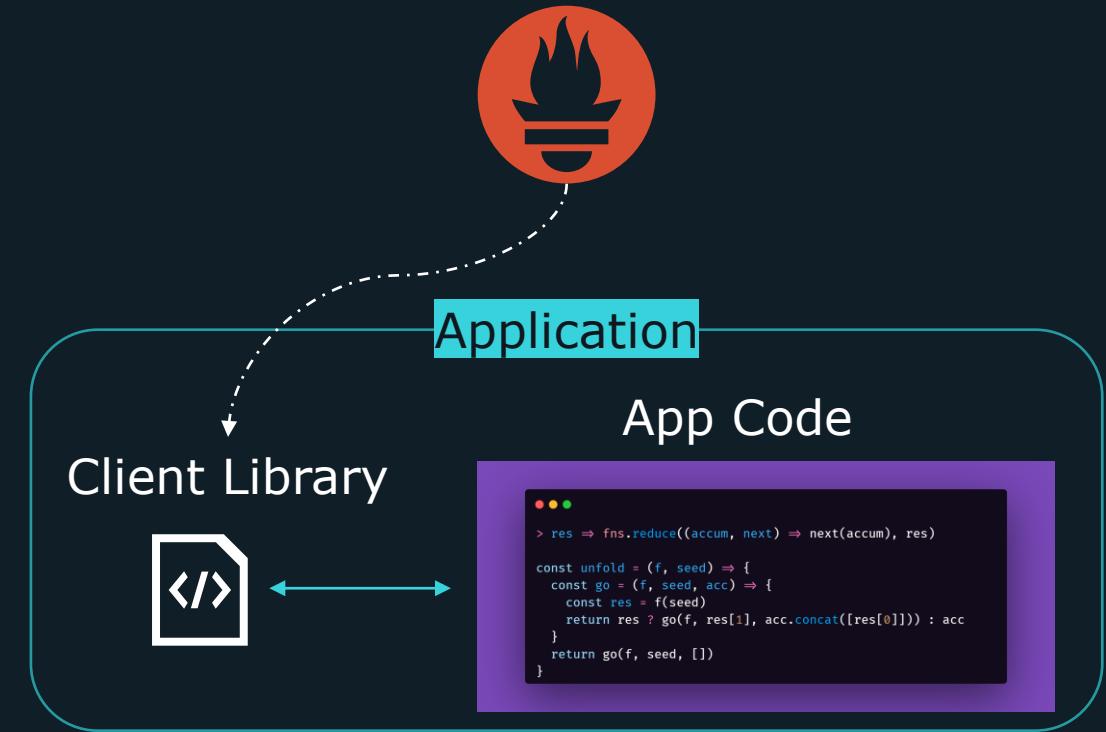
Instrumentation



Client Libraries

The Prometheus client libraries provide an easy way to add **instrumentation** to your code in order to track and expose metrics for Prometheus

1. Track metrics in the Prometheus expected format
2. Expose metrics via /metrics path



Prometheus has official client libraries for the following languages:

- Go
- Java/Scala
- Python
- Ruby
- Rust

You can even write your own client library:

- Unsupported language
- Avoid dependencies

Unofficial third-party client libraries:

- Bash
- C
- C++
- Common Lisp
- Dart
- Elixir
- Erlang
- Haskell
- Lua
- C#
- Node.js
- Ocaml
- Perl
- PHP
- R

Python Client Library

```
>_ Main.py
```

```
from flask import Flask ← Import flask package
```

```
app = Flask(__name__) ← Initialize a flask instance
```

```
@app.get("/cars") ← Creates an API endpoint for  
def get_cars(): ← GET requests to the url /cars  
    return ["toyota", "honda", "mazda",  
"lexus"]
```

```
if __name__ == '__main__': ← Starts API on port 5001  
    app.run(port='5001')
```

Python Client Library

>_

```
$ pip install prometheus_client
```

```
Collecting prometheus_client
  Downloading prometheus_client-0.14.1-py3-none-any.whl (59 kB)
    |████████████████████████████████| 59 kB 1.2 MB/s
Installing collected packages: prometheus-client
Successfully installed prometheus-client-0.14.1
```

Client Counter

>_ Main.py

```
from prometheus_client import Counter  
  
REQUESTS = Counter('http_requests_total',  
                    'Total number of requests')  
  
app = Flask(__name__)  
  
@app.get("/cars")  
def get_cards():  
    REQUESTS.inc()  
    return ["toyota", "honda", "mazda",  
"lexus"]
```

Import Counter class
from prom client

Create a Counter metric with
a name and a description

Call the inc() method on the
counter to increase the
counter +1

How do we access the metrics endpoint?

>_ Main.py

```
from prometheus_client import Counter,  
start_http_server
```

Import start_http_server

```
if __name__ == '__main__':  
    start_http_server(8000)  
    app.run(port='5001')
```

Start the metrics server on port 8000

Flask App: 5001
Prometheus: 8000

Client Endpoint

>_

```
$ curl 127.0.0.1:8000
```

```
# HELP python_gc_objects_uncollectable_total Uncollectable object found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 77.0
python_gc_collections_total{generation="1"} 7.0
python_gc_collections_total{generation="2"} 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="9",patchlevel="5",version="3.9.5"} 1.0
# HELP http_requests_total Total number of requests
# TYPE http_requests_total counter
http_requests_total 5.0
# HELP http_requests_created Total number of requests
# TYPE http_requests_created gauge
http_requests_created 1.6654499091926205e+09
```

Expose metrics from Flask Route

>_ Main.py

```
from flask import Flask
from prometheus_client import make_wsgi_app
from werkzeug.middleware.dispatcher import
DispatcherMiddleware

app = Flask(__name__)

# Add prometheus middleware to export metrics
# at /metrics
app.wsgi_app =
DispatcherMiddleware(app.wsgi_app, {
    '/metrics': make_wsgi_app()})
```

Flask App: http://localhost:5001
Prometheus: http://localhost:5001/metrics

Adding Routes

Main.py

```
>_          Main.py

@app.get("/cars")
def get_cars():
    return ["toyota", "honda", "mazda", "lexus"]

@app.get("/cars/<int:id>")
def get_car():
    return "Single car"

@app.post("/cars")
def create_cars():
    return "Create Car"

@app.patch("/cars/<int:id>")
def update_cars():
    return "Updating Car"

@app.delete("/cars/<int:id>")
def get_cars():
    return "Deleting Car"
```

Get request to /cars/1234

Post request to /cars

Patch request to /cars/1234

Delete request to /cars/1234

Instrumentation

```
@app.get("/cars")
def get_cars():
    REQUESTS.inc()
    return ["toyota", "honda", "mazda", "lexus"]

@app.get("/cars/<int:id>")
def get_car():
    REQUESTS.inc()
    return "Single car"

@app.post("/cars")
def create_cars():
    REQUESTS.inc()
    return "Create Car"

@app.patch("/cars/<int:id>")
def update_cars():
    REQUESTS.inc()
    return "Updating Car"

@app.delete("/cars/<int:id>")
def get_cars():
    REQUESTS.inc()
    return "Deleting Car"
```

The `http_requests_total` counter
should report total number of requests

We need to increment the REQUESTS
counter for **every route**



KodeKloud

Labels

Labels

>_ Main.py

```
@app.get("/cars")
def get_cars():
    REQUESTS.inc()
    return ["toyota", "honda", "mazda", "lexus"]

@app.post("/cars")
def create_cars():
    REQUESTS.inc()
    return "Create Car"

@app.get("/boats")
def get_boats():
    REQUESTS.inc()
    return ["boat1", "boat2", "boat3"]

@app.post("/boats")
def create_boat():
    REQUESTS.inc()
    return "Create Boat"
```

This allows us to count total requests across application.

What if we also want to get total requests per path?

Labels

Main.py

```
>_          Main.py

CAR_REQUESTS = Counter('requests_cars_total',
                      'Total number of requests for /cars path')
BOATS_REQUESTS = Counter('requests_boats_total',
                         'Total number of requests for /boats path')

@app.get("/cars")
def get_cars():
    CAR_REQUESTS.inc()
    return ["toyota", "honda", "mazda", "lexus"]

@app.post("/cars")
def create_cars():
    CAR_REQUESTS.inc()
    return "Create Car"

@app.get("/boats")
def get_boats():
    BOATS_REQUESTS.inc()
    return ["boat1", "boat2", "boat3"]

@app.post("/boats")
def create_boat():
    BOATS_REQUESTS.inc()
    return "Create Boat"
```

Create a separate counter for each path

This is **not** the recommended approach as now it is very difficult to find the total number of requests across all paths.

We would have to add up total requests per path and know all of the paths ahead of time.

Labels

>_

Main.py

```
REQUESTS = Counter('http_requests_total',
                    'Total number of requests',
                    Labelnames=['path'])  
  
@app.get("/cars")
def get_cars():
    REQUESTS.labels('/cars').inc()
    return ["toyota", "honda", "mazda", "lexus"]  
  
@app.post("/cars")
def create_cars():
    REQUESTS.labels('/cars').inc()
    return "Create Car"  
  
@app.get("/boats")
def get_boats():
    REQUESTS.labels('/boats').inc()
    return ["boat1", "boat2", "boat3"]  
  
@app.post("/boats")
def create_boat():
    REQUESTS.labels('/boats').inc()
    return "Create Boat"
```

We can create one counter and use a **label** to distinguish between the different paths

When we call the `inc()` method, add the **labels()** method and specify the path

Labels

>_

```
$ http_requests_total{path="/cars"}
```

```
http_requests_total{path="/cars"} 5.0
```

```
$ http_requests_total{path="/boats"}
```

```
http_requests_total{path="/cars"} 2.0
```

```
$ http_requests_total
```

```
http_requests_total{path="/cars"} 5.0  
http_requests_total{path="/boats"} 2.0
```

```
$ sum(http_requests_total)
```

```
{} 7.0
```

Sum total requests across
all paths

Multiple Labels

>_ Main.py

```
REQUESTS = Counter('http_requests_total',
                    'Total number of requests',
                    Labelnames=['path', 'method'])  
  
@app.get("/cars")
def get_cars():
    REQUESTS.labels('/cars', ['get']).inc()
    return ["toyota", "honda", "mazda", "lexus"]  
  
@app.post("/cars")
def create_cars():
    REQUESTS.labels('/cars', ['post']).inc()
    return "Create Car"  
  
@app.get("/boats")
def get_boats():
    REQUESTS.labels('/boats', 'get').inc()
    return ["boat1", "boat2"]  
  
@app.post("/boats")
def create_boat():
    REQUESTS.labels('/boats', 'post').inc()
    return "Create Boat"
```

Can specify multiple labels,
one for **path** and one for the
method

Multiple Labels

>_

```
$ http_requests_total
```

```
http_requests_total{method="get",path="/cars"} 2.0
http_requests_total{method="get",path="/boats"} 3.0
http_requests_total{method="post",path="/boats"} 1.0
http_requests_total{method="post",path="/cars"} 6.0
```

```
$ http_requests_total{method="get"}
```

```
http_requests_total{method="get",path="/cars"} 2.0
http_requests_total{method="get",path="/boats"} 3.0
```

```
$ http_requests_total{path="/cars"}
```

```
http_requests_total{method="get",path="/cars"} 2.0
http_requests_total{method="post",path="/boats"} 1.0
```



KodeKloud

Histogram

Histogram

>_ Main.py

```
LATENCY = Histogram('request_latency_seconds', 'Request  
Latency', Labelnames=['path', 'method'])  
  
def before_request():  
    request.start_time = time.time()  
  
  
def after_request(response):  
    request_latency = time.time() - request.start_time  
    LATENCY.labels(  
        request.method,  
        request.path).observe(request_latency)  
    return response  
  
if __name__ == '__main__':  
    start_http_server(8000)  
    app.before_request(before_request)  
    app.after_request(after_request)
```

Add a **histogram** metric to track latency/response-time for each request

Histogram

>_

\$ request_latency_seconds

```
request_latency_seconds_bucket{le="0.005",method="/cars",path="GET"} 0.0
request_latency_seconds_bucket{le="0.01",method="/cars",path="GET"} 0.0
request_latency_seconds_bucket{le="0.025",method="/cars",path="GET"} 0.0
request_latency_seconds_bucket{le="0.05",method="/cars",path="GET"} 1.0
request_latency_seconds_bucket{le="0.075",method="/cars",path="GET"} 3.0
request_latency_seconds_bucket{le="0.1",method="/cars",path="GET"} 3.0
request_latency_seconds_bucket{le="0.25",method="/cars",path="GET"} 4.0
request_latency_seconds_bucket{le="0.5",method="/cars",path="GET"} 6.0
request_latency_seconds_bucket{le="0.75",method="/cars",path="GET"} 6.0
request_latency_seconds_bucket{le="1.0",method="/cars",path="GET"} 8.0
request_latency_seconds_bucket{le="2.5",method="/cars",path="GET"} 8.0
request_latency_seconds_bucket{le="5.0",method="/cars",path="GET"} 8.0
request_latency_seconds_bucket{le="7.5",method="/cars",path="GET"} 8.0
request_latency_seconds_bucket{le="10.0",method="/cars",path="GET"} 8.0
request_latency_seconds_bucket{le="+Inf",method="/cars",path="GET"} 8.0
request_latency_seconds_count{method="/cars",path="GET"} 8.0
```

Histogram

Buckets for a histogram can be manually set

>_

Main.py

```
LATENCY = Histogram('request_latency_seconds', 'Flask  
Request Latency', Labelnames=['path', 'method'],  
buckets=[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08,  
0.09, 0.1])
```

Summary

Let's convert the Histogram metric to a Summary metric

```
>_
```

```
Main.py
```

```
LATENCY = Summary('request_latency_seconds', 'Flask  
Request Latency', Labelnames=['path', 'method'])  
  
LATENCY.labels(  
    request.method,  
    request.path).observe(request_latency)
```

>_

\$ http_requests_total

request_latency_seconds_count{method="/cars", path="GET"}	166783
request_latency_seconds_sum{method="/cars", path="GET"}	9013

_count - is the number of observe calls, which in this case is the number of requests

_sum - is the sum of the values passed to observe, which is the amount of time spent responding to requests

rate(request_latency_seconds_count[1m]) returns the per-second rate of requests

Rate(request_latency_seconds_sum[1m]) return the amount of time spent responding to requests per second

rate(request_latency_seconds_count[1m]) / Rate(request_latency_seconds_sum[1m]) Returns average latency over the last minute



KodeKloud

Gauge

Gauge

>_ Main.py

```
IN_PROGRESS = Gauge('inprogress_requests',
                     'Total number of requests in progress',
                     Labelnames=['path', 'method'])

def before_request():
    IN_PROGRESS.labels(request.method, request.path).inc()
    request.start_time = time.time()

def after_request(response):
    IN_PROGRESS.labels(request.method, request.path).dec()
    return response
```

Let's create a **gauge** metric that will track the number of active requests getting processed

There is also a **set()** method to set the metric to a specific value

Gauge

>_

```
$ inprogress_requests
```

```
inprogress_requests{method="/cars",path="GET"} 3.0
inprogress_requests{method="/cars",path="POST"} 0.0
inprogress_requests{method="/boats",path="POST"} 18.0
inprogress_requests{method="/boats",path="GET"} 7.0
```



KodeKloud

Best Practice

Naming Metrics

library_name_unit_suffix

Metrics should follow **snake_case** – lowercase
with words separated by _

http_requests_total

The first word of the metric should be the
application/library the metric is used for.



postgres_

The **name** portion of the metric name should
provide a description of what the metric is
used for. More than one word can be used

postgres_queue_size

Naming Metrics

library_name_unit_suffix

The unit should always be included in the metric name, so there's no second guessing which units its using

Make sure to use unprefixed base units like **seconds** and **bytes**, and **meters**. We don't want to use microseconds or kilobytes

postgres_queue_size_bytes

_total, _count, _sum, and _bucket suffixes are used by various metrics so its best to avoid adding those yourselves

Only exception is to add **_total** for counter metrics

Naming Metrics



Proper Metric Names

process_cpu_seconds

http_requests_total

redis_connection_errors

node_disk_read_bytes_total



Incorrect Metric Names

container_docker_restarts

http_requests_sum

nginx_disk_free_kilobytes

dotnet_queue_waiting_time

What to Instrument

Three types of services



Online-serving systems



Offline processing



Batch jobs

What to Instrument

Online-serving system is one where an **immediate** response is expected. Databases and web-servers would fall into this category

1. Number of Queries/Requests
2. Number of Errors
3. Latency
4. Number of in progress requests

What to Instrument

Offline processing service is one where no one is actively waiting for a response.
Usually patch up work and have multiple stages in a pipeline

For each stage you want to get:

1. Amount of queued work
2. Amount of work in progress
3. Rate of processing
4. errors

What to Instrument

Batch Jobs are similar to offline-serving systems with the main difference being Batch Jobs are run on a regular schedule whereas offline-serving systems run continuously

Will require a pushGateway as it isn't continuously running to be scraped

1. Time processing each stage of job
2. Overall runtime
3. Last time job completed

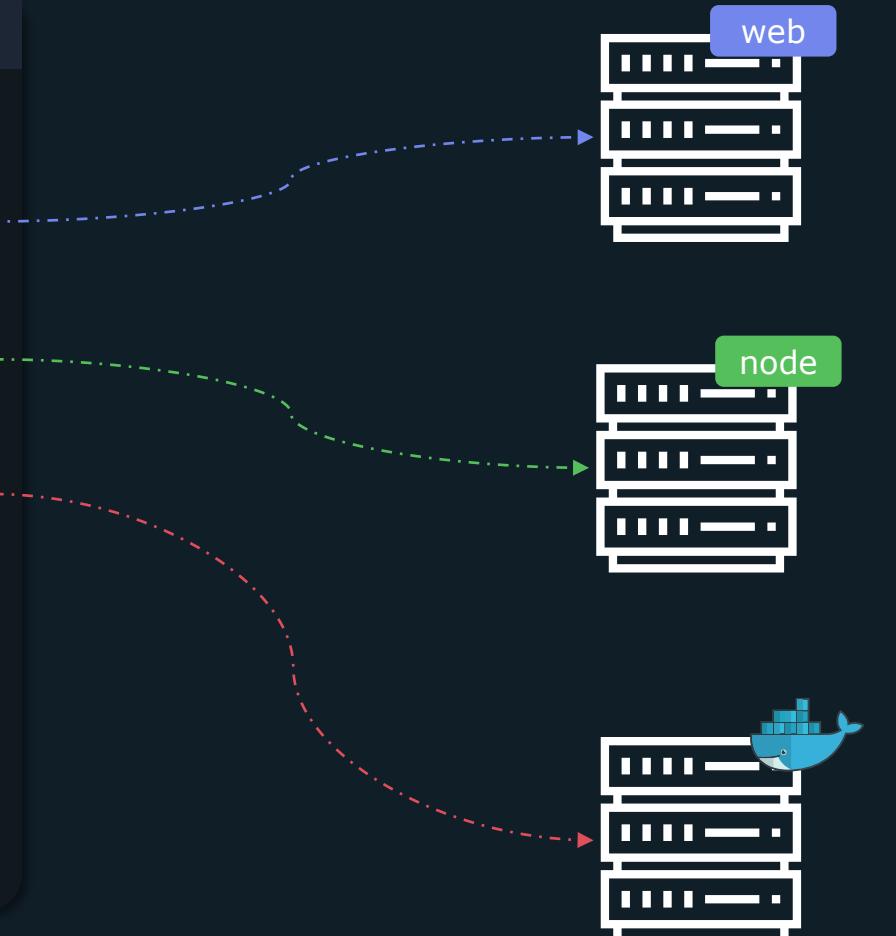


KodeKloud

Service Discovery

Service Discovery

```
>_      prometheus.yml
scrape_configs:
  - job_name: "web"
    static_configs:
      targets: ["localhost:9090"]
  - job_name: "node"
    static_configs:
      - targets: ["192.168.1.168:9100"]
  - job_name: "docker"
    static_configs:
      - targets: ["localhost:9323"]
  - job_name: "database"
    static_configs:
      - targets: ["localhost:9090"]
```



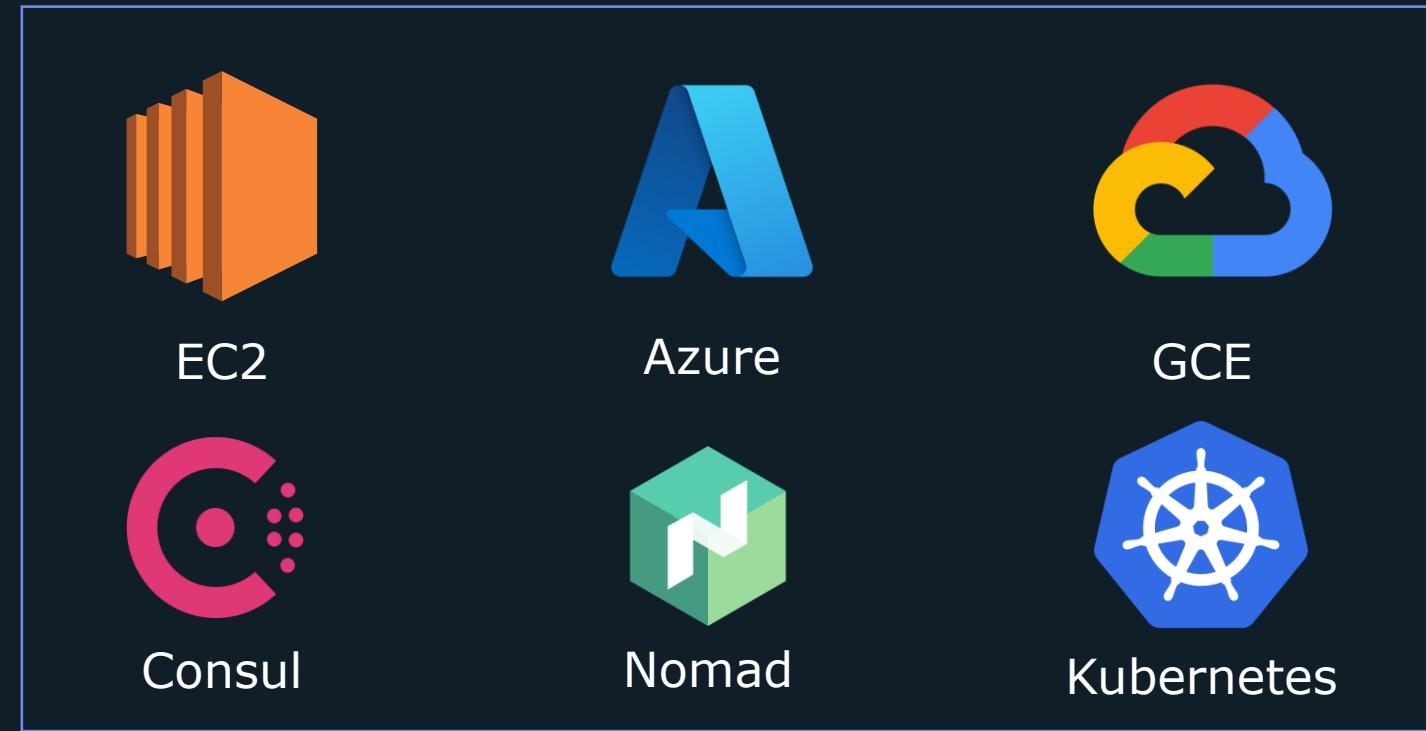


| what is service discovery?

Service Discovery allows Prometheus to populate a list of endpoints to scrape that can get dynamically updated as new endpoints get created & destroyed

Service Discovery

Prometheus has **built-in** support for several service discovery mechanisms



Static

```
>_      prometheus.yml

scrape_configs:
  - job_name: "web"
    static_configs:
      - targets: ["localhost:9090"]
  - job_name: "node"
    static_configs:
      - targets: ["192.168.1.168:9100"]
  - job_name: "docker"
    static_configs:
      - targets: ["localhost:9323"]
```



KodeKloud

File Service Discovery

File Service Discovery

A list of jobs/targets can be **imported** from a file

This allows you to integrate with service discovery
systems Prometheus doesn't **support** out of the box

Supports **json** and **yaml** files



File Service Discovery

```
>_ file-sd.json
```

```
[  
  {  
    "targets": [ "node1:9100", "node2:9100" ],  
    "labels": {  
      "team": "dev",  
      "job": "node"  
    }  
  },  
  {  
    "targets": [ "localhost:9090" ],  
    "labels": {  
      "team": "monitoring",  
      "job": "prometheus"  
    }  
  },  
  {  
    "targets": [ "db1:9090" ],  
    "labels": {  
      "team": "db",  
      "job": "database"  
    }  
  }  
]
```

```
>_ Prometheus.yml
```

```
scrape_configs:  
  - job_name: file-example  
    file_sd_configs:  
      - files:  
        - 'file-sd.json'  
        - '*.json'
```

File Service Discovery

Targets

All Unhealthy Collapse All



Filter by endpoint or labels

file-example (0/4 up)

[show less](#)

Endpoint	State	Labels
http://node1:9100/metrics	UNKNOWN	instance="node1:9100" job="node" team="dev"
http://node2:9100/metrics	DOWN	instance="node2:9100" job="node" team="dev"
http://localhost:9090/metrics	UNKNOWN	instance="localhost:9090" job="prometheus" team="monitoring"
http://db1:9090/metrics	DOWN	instance="db1:9090" job="database" team="db"

File Service Discovery

The screenshot shows the Prometheus web interface with the following details:

- Header:** Prometheus, Alerts, Graph, Status ▾, Help.
- Main Title:** Service Discovery
- Search Bar:** A magnifying glass icon.
- Target List:** A bulleted list of active targets:
 - node (1 / 1 active targets)
 - pushgateway (1 / 1 active targets)
 - prometheus (0 / 1 active targets)
- Filter:** node, show more.
- Dropdown Menu (Status ▾):** A list of options with "Service Discovery" highlighted by a red border:
 - Runtime & Build Information
 - TSDB Status
 - Command-Line Flags
 - Configuration
 - Rules
 - Targets
 - Service Discovery

File Service Discovery

Service Discovery	
file-example (4 / 4 active targets)	
file-example <small>show less</small>	
Discovered Labels	Target Labels
<pre>_address_ = "node1:9100" .meta_filepath= "/etc/prometheus/file-sd.json" .metrics_path_ = "/metrics" .scheme_ = "http" .scrape_interval_ = "15s" .scrape_timeout_ = "10s" .job="node" .team="dev"</pre>	<pre>instance="node1:9100" .job="node" .team="dev"</pre>
<pre>_address_ = "node2:9100" .meta_filepath= "/etc/prometheus/file-sd.json" .metrics_path_ = "/metrics" .scheme_ = "http" .scrape_interval_ = "15s" .scrape_timeout_ = "10s" .job="node" .team="dev"</pre>	<pre>instance="node2:9100" .job="node" .team="dev"</pre>
<pre>_address_ = "localhost:9090" .meta_filepath= "/etc/prometheus/file-sd.json" .metrics_path_ = "/metrics" .scheme_ = "http" .scrape_interval_ = "15s" .scrape_timeout_ = "10s" .job="prometheus" .team="monitoring"</pre>	<pre>instance="localhost:9090" .job="prometheus" .team="monitoring"</pre>
<pre>_address_ = "db1:9090" .meta_filepath= "/etc/prometheus/file-sd.json" .metrics_path_ = "/metrics" .scheme_ = "http" .scrape_interval_ = "15s" .scrape_timeout_ = "10s" .job="database" .team="db"</pre>	<pre>instance="db1:9090" .job="database" .team="db"</pre>

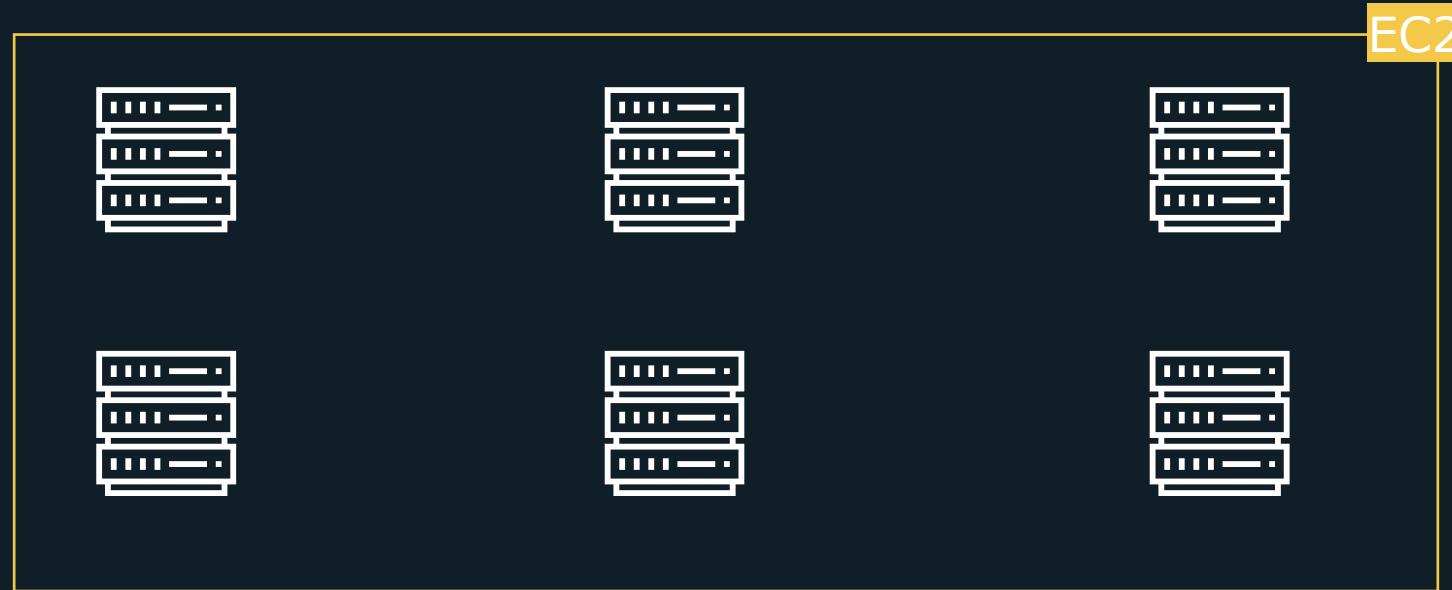


KodeKloud

AWS

Cloud infrastructure is very **dynamic** especially when autoscaling is enabled

EC2 Service Discovery makes it so Prometheus has a list of all available **EC2** instances to scrape



```
>_          Prometheus.yml
```

```
scrape_configs:  
- job_name: EC2  
  ec2_sd_configs:  
    - region: <region>  
      access_key: <access key>  
      secret_key: <secret key>
```

Credentials should be setup with an IAM user with the [AmazonEC2ReadOnlyAccess](#) policy

EC2 Service Discovery

Service Discovery



Filter by labels

- EC2 (2 / 2 active targets)

EC2

[show less](#)

Discovered Labels

`_address = "172.31.1.156:80"`
`_meta_ec2_ami="ami-026b57f3c383c2eec"`
`_meta_ec2_architecture="x86_64"`
`_meta_ec2_availability_zone="us-east-1b"`
`_meta_ec2_availability_zone_id="use1-az4"`
`_meta_ec2_instance_id="i-0e23a91e6c6a198db"`
`_meta_ec2_instance_state="running"`
`_meta_ec2_instance_type="t2.micro"`
`_meta_ec2_owner_id="040497317401"`
`_meta_ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41"`
`_meta_ec2_private_dns_name="ip-172-31-1-156.ec2.internal"`
`_meta_ec2_private_ip="172.31.1.156"`
`_meta_ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com"`
`_meta_ec2_public_ip="3.80.117.102"`
`_meta_ec2_subnet_id="subnet-0fde5f37cd6c80f41,"`
`_meta_ec2_tag_Name="web"`
`_meta_ec2_vpc_id="vpc-294f5553"`
`_metrics_path_ ="/metrics"`
`_scheme_ ="http"`
`_scrape_interval_ ="15s"`
`_scrape_timeout_ ="10s"`
`job="EC2"`

Target Labels

`instance="172.31.1.156:80"`
`job="EC2"`

EC2 Service Discovery

The EC2 service discovery was able to extract metadata for each EC2 instance it found:

- `__meta_ec2_tag_Name="web"`: is the Name tag for the instance
- `__meta_ec2_instance_type="t2.micro"`: Is the instance type
- `__meta_ec2_vpc_id="vpc-294f5553"`: vpc the instance is located in
- `__meta_ec2_private_ip="172.31.1.156"`: Private IP of instance

The instance label defaults to using the private ip. This is done due to the assumption that Prometheus will be running as “close” as possible to what it is monitoring.

In addition, not all EC2 instances have public IPs



KodeKloud

Re-labeling

Re-labeling

Re-labeling – allows you to classify/filter Prometheus targets and metrics by rewriting their label set



Re-labeling

>_ Prometheus.yml

```
scrape_configs:  
  - job_name: EC2  
    relabel_configs:  
    metric_relabel_configs:  
      ec2_sd_configs:  
        - region: <region>  
          access_key: <access key>  
          secret_key: <secret key>
```

Relabeling occurs before scrape occurs
and only has access to labels added by
Service Discovery

Relabeling occurs after the scrape

Re-Labeling

- ec2 (2 / 2 active targets)
- pushgateway (1 / 1 active targets)
- prometheus (0 / 1 active targets)

ec2 [show less](#)

Discovered Labels

```
address_ = "172.31.1.156:80"  
meta_ec2_ami="ami-026b57f3c383c2eec"  
meta_ec2_architecture="x86_64"  
meta_ec2_availability_zone="us-east-1b"  
meta_ec2_availability_zone_id="use1-az4"  
meta_ec2_instance_id="i-0e23a91e6c6a198db"  
meta_ec2_instance_state="running"  
meta_ec2_instance_type="t2.micro"  
meta_ec2_owner_id="040497317401"  
meta_ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41"  
meta_ec2_private_dns_name="ip-172-31-1-156.ec2.internal"  
meta_ec2_private_ip="172.31.1.156"  
meta_ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com"  
meta_ec2_public_ip="3.80.117.102"  
meta_ec2_region="us-east-1"  
meta_ec2_subnet_id=",subnet-0fde5f37cd6c80f41,"  
meta_ec2_tag_Name="web"  
meta_ec2_tag_env="dev" ← Red arrow points here  
meta_ec2_vpc_id="vpc-294f5553"  
metrics_path_ = "/metrics"  
scheme_ ="http"  
scrape_interval_ ="15s"  
scrape_timeout_ ="10s"  
job="ec2"  
  
address_ = "172.31.1.34:80"  
meta_ec2_ami="ami-026b57f3c383c2eec"  
meta_ec2_architecture="x86_64"
```

Target Labels

```
instance="172.31.1.156:80"  
job="ec2"
```

```
instance="172.31.1.34:80"  
job="ec2"
```

`meta_ec2_tag_env= dev | prod`

A screenshot of a search interface. At the top left is a magnifying glass icon. To its right is a search bar containing the text "up". Below the search bar are two tabs: "Table" (which is greyed out) and "Graph" (which is blue). Underneath the tabs is a horizontal slider with arrows on both ends, labeled "Evaluation time". Below the slider, there are two lines of search results:

```
up{instance="172.31.1.156:80", job="ec2"}  
up{instance="172.31.1.34:80", job="ec2"}
```

Re-label

Prometheus.yml

```
scrape_configs:  
- job_name: example  
  relabel_configs:  
    - source_labels: [__meta_ec2_tag_env]  
      regex: prod  
      action: keep|drop|replace
```

Source_labels – array of labels to match on

Regex – uses regular expressions to match on a specific value

Keep means we will scrape target if it has the source_label, drop means we will not scrape target

Doing a keep here means any ec2 instance where this label is not set to prod will **not** be scraped

Re-label

```
>_      Prometheus.yml
```

```
scrape_configs:  
  - job_name: example  
    relabel_configs:  
      - source_labels: [__meta_ec2_tag_env]  
        regex: dev  
        action: drop
```

Drop any targets where
`{__meta_c2_tag_env=dev}`

```
>_      Prometheus.yml  
  
scrape_configs:  
  - job_name: example  
    relabel_configs:  
      - source_labels: [env, team]  
        regex: dev;marketing  
        action: keep
```

When there are more than 1 source labels selected, they will be joined by a ;

To match `{env="dev"}` and `{team="marketing"}` use:

`dev;marketing`

>_ Prometheus.yml

```
scrape_configs:  
- job_name: example  
  relabel_configs:  
    - source_labels: [env, team]  
      regex: dev-marketing  
      action: keep  
      separator: "-"
```

To change the delimiter between labels use the **separator** property

dev-marketing

Target Labels

Target labels - are labels that are added to the labels of **every** time series returned from a scrape

Discovered Labels	Target Labels
_address = "172.31.1.156:80" _meta_ec2_ami="ami-026b57f3c383c2eec" _meta_ec2_architecture="x86_64" _meta_ec2_availability_zone="us-east-1b" _meta_ec2_availability_zone_id="use1-az4" _meta_ec2_instance_id="i-0e23a91e6c6a198db" _meta_ec2_instance_state="running" _meta_ec2_instance_type="t2.micro" _meta_ec2_owner_id="040497317401" _meta_ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41" _meta_ec2_private_dns_name="ip-172-31-1-156.ec2.internal" _meta_ec2_private_ip="172.31.1.156" _meta_ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com" _meta_ec2_public_ip="3.80.117.102" _meta_ec2_region="us-east-1" _meta_ec2_subnet_id="subnet-0fde5f37cd6c80f41," _meta_ec2_tag_Name="web" _meta_ec2_tag_env="dev" _meta_ec2_vpc_id="vpc-294f5553" _metrics_path_ ="/metrics" _scheme_ ="http" _scrape_interval_ ="15s" _scrape_timeout_ ="10s" job="ec2"	instance="172.31.1.156:80" job="ec2"

Target Labels

>_ Prometheus.yml

```
scrape_configs:  
- job_name: example  
  relabel_configs:  
    - source_labels: [__address__]  
      regex: (.*):.*  
      target_label: ip  
      action: replace  
      replacement: $1
```

EC2 instances have label:

__address__=192.168.1.1:80

Let's say we want to convert this to the following label

{ip="192.168.1.1"}

The **action** property should be set to **replace**

target_label should be the name of the new label

regex is set to `(.*):.*` which will assign everything before the `:` into a group(can be referenced with `$1`)

replacement is what the new value of the label should be which is set to `$1` which will grab the group from the regex

Dropping Label

```
>_      Prometheus.yml  
  
scrape_configs:  
- job_name: example  
  relabel_configs:  
    - regex: __meta_ec2_owner_id  
      action: labeldrop
```

To drop the `__meta_ec2_owner_id` discovery label:

specify the `regex` to match the desired labels to drop in this case we want to match on one label so we can just put the name of the label here

For `action` set it to `labeldrop`

Dropping Label

```
>_      Prometheus.yml  
  
scrape_configs:  
  - job_name: example  
    relabel_configs:  
      [- regex: instance|job  
        action: labelkeep]
```

labelkeep can be used to specify labels that should be kept.

All other labels will be **dropped**

In this example only the **instance** & **job** labels will be kept all other labels will be dropped

Target Labels

Discovered Labels	Target Labels
<code>_address = "172.31.1.156:80"</code> <code>_meta_ec2_ami="ami-026b57f3c383c2eec"</code> <code>_meta_ec2_architecture="x86_64"</code> <code>_meta_ec2_availability_zone="us-east-1b"</code> <code>_meta_ec2_availability_zone_id="use1-az4"</code> <code>_meta_ec2_instance_id="i-0e23a91e6c6a198db"</code> <code>_meta_ec2_instance_state="running"</code> <code>_meta_ec2_instance_type="t2.micro"</code> <code>_meta_ec2_owner_id="040497317401"</code> <code>_meta_ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41"</code> <code>_meta_ec2_private_dns_name="ip-172-31-1-156.ec2.internal"</code> <code>_meta_ec2_private_ip="172.31.1.156"</code> <code>_meta_ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com"</code> <code>_meta_ec2_public_ip="3.80.117.102"</code> <code>_meta_ec2_region="us-east-1"</code> <code>_meta_ec2_subnet_id="subnet-0fde5f37cd6c80f41,"</code> <code>_meta_ec2_tag_Name="web"</code> <code>_meta_ec2_tag_env="dev"</code> <code>_meta_ec2_vpc_id="vpc-294f5553"</code> <code>_metrics_path ="/metrics"</code> <code>_scheme_= "http"</code> <code>_scrape_interval = "15s"</code> <code>_scrape_timeout = "10s"</code> <code>job="ec2"</code>	<code>instance="172.31.1.156:80"</code> <code>job="ec2"</code>

All labels beginning with `_` will be discarded at the end of re-labeling

That is why they don't show up as target labels

labelmap

```
>_          Prometheus.yml  
  
scrape_configs:  
- job_name: example  
  relabel_configs:  
    - regex: __meta_ec2_(.*)  
      action: labelmap  
      replacement: ec2_$1
```

If we want to keep one or more of these labels, we can use the **labelmap** action

Unlike the replace action, the labelmap replacement property modifies the label name not the value

This example will take all labels that start with `__meta_ec2_` and map them to target labels and prefix them with the `ec2`

labelmap

Discovered Labels	Target Labels
<code>_address_ = "172.31.1.156:80"</code>	
<code>meta_ec2_ami="ami-026b57f3c383c2eec"</code>	<code>ec2_ami="ami-026b57f3c383c2eec"</code>
<code>meta_ec2_architecture="x86_64"</code>	<code>ec2_architecture="x86_64"</code>
<code>meta_ec2_availability_zone="us-east-1b"</code>	<code>ec2_availability_zone="us-east-1b"</code>
<code>meta_ec2_availability_zone_id="use1-az4"</code>	<code>ec2_availability_zone_id="use1-az4"</code>
<code>meta_ec2_instance_id="i-0e23a91e6c6a198db"</code>	<code>ec2_instance_id="i-0e23a91e6c6a198db"</code>
<code>meta_ec2_instance_state="running"</code>	<code>ec2_instance_state="running"</code>
<code>meta_ec2_instance_type="t2.micro"</code>	<code>ec2_instance_type="t2.micro"</code>
<code>meta_ec2_owner_id="040497317401"</code>	<code>ec2_owner_id="040497317401"</code>
<code>meta_ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41"</code>	<code>ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41"</code>
<code>meta_ec2_private_dns_name="ip-172-31-1-156.ec2.internal"</code>	<code>ec2_private_dns_name="ip-172-31-1-156.ec2.internal"</code>
<code>meta_ec2_private_ip="172.31.1.156"</code>	<code>ec2_private_ip="172.31.1.156"</code>
<code>meta_ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com"</code>	<code>ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com"</code>
<code>meta_ec2_public_ip="3.80.117.102"</code>	<code>ec2_public_ip="3.80.117.102"</code>
<code>meta_ec2_region="us-east-1"</code>	<code>ec2_region="us-east-1"</code>
<code>meta_ec2_subnet_id="subnet-0fde5f37cd6c80f41"</code>	<code>ec2_subnet_id="subnet-0fde5f37cd6c80f41,"</code>
<code>meta_ec2_tag_Name="web"</code>	<code>ec2_tag_Name="web"</code>
<code>meta_ec2_tag_env="dev"</code>	<code>ec2_tag_env="dev"</code>
<code>meta_ec2_vpc_id="vpc-294f5553"</code>	<code>ec2_vpc_id="vpc-294f5553"</code>
<code>metrics_path_ = "/metrics"</code>	<code>instance="172.31.1.156:80"</code>
<code>scheme_ ="http"</code>	
<code>scrape_interval_ ="15s"</code>	
<code>scrape_timeout_ ="10s"</code>	
<code>job="ec2"</code>	<code>job="ec2"</code>

Table **Graph**

Evaluation time

```
up{ec2_ami="ami-026b57f3c383c2eec", ec2_architecture="x86_64", ec2_availability_zone="us-east-1b", ec2_availability_zone_id="use1-az4", ec2_instance_id="i-0e23a91e6c6a198db", ec2_instance_state="running", ec2_private_ip="172.31.1.156", ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com", ec2_public_ip="3.80.117.102", ec2_region="us-east-1", ec2_subnet_id="subnet-0fde5f37cd6c80f41", ec2_tag_Name="web", ec2_tag_env="dev", ec2_vpc_id="vpc-294f5553"}  
up{ec2_ami="ami-026b57f3c383c2eec", ec2_architecture="x86_64", ec2_availability_zone="us-east-1b", ec2_availability_zone_id="use1-az4", ec2_instance_id="i-0f15968d505d7bf59", ec2_instance_state="running", ec2_private_ip="172.31.1.34", ec2_public_dns_name="ec2-54-242-72-200.compute-1.amazonaws.com", ec2_public_ip="54.242.72.200", ec2_region="us-east-1", ec2_subnet_id="subnet-0fde5f37cd6c80f41", ec2_tag_Name="web", ec2_tag_env="prod", ec2_vpc_id="vpc-294f5553"}
```

Metric_relabel_config

```
>_      Prometheus.yml  
  
scrape_configs:  
  - job_name: example  
    metric_relabel_configs:  
      - source_labels: [__name__]  
        regex: http_errors_total  
        action: drop
```

metric_relabel_configs occurs after the scrape and has access to scraped metrics

Configuration is identical to relabel_configs

To drop a metric called:

http_errors_total

Metric_relabel_config

>_ Prometheus.yml

```
scrape_configs:  
  - job_name: example  
    metric_relabel_configs:  
      - source_labels: [__name__]  
        regex: http_errors_total  
        action: replace  
        target_label: __name__  
        replacement: http_failures_total
```

To rename a metric from `http_errors_total` to
`http_failures_total`

Metric_relabel_config

```
>_          Prometheus.yml  
  
scrape_configs:  
  - job_name: example  
    metric_relabel_configs:  
      [- regex: 'code'  
       action: labeldrop ]
```

To drop a label named **code**

Metric_relabel_config

>_ Prometheus.yml

```
scrape_configs:  
  - job_name: example  
    metric_relabel_configs:  
      - source_labels: [path]  
        regex: \/(.*)  
        action: replace  
        target_label: endpoint  
        replacement: $1
```

{path="/cars"}

Change it to

{endpoint="cars"}

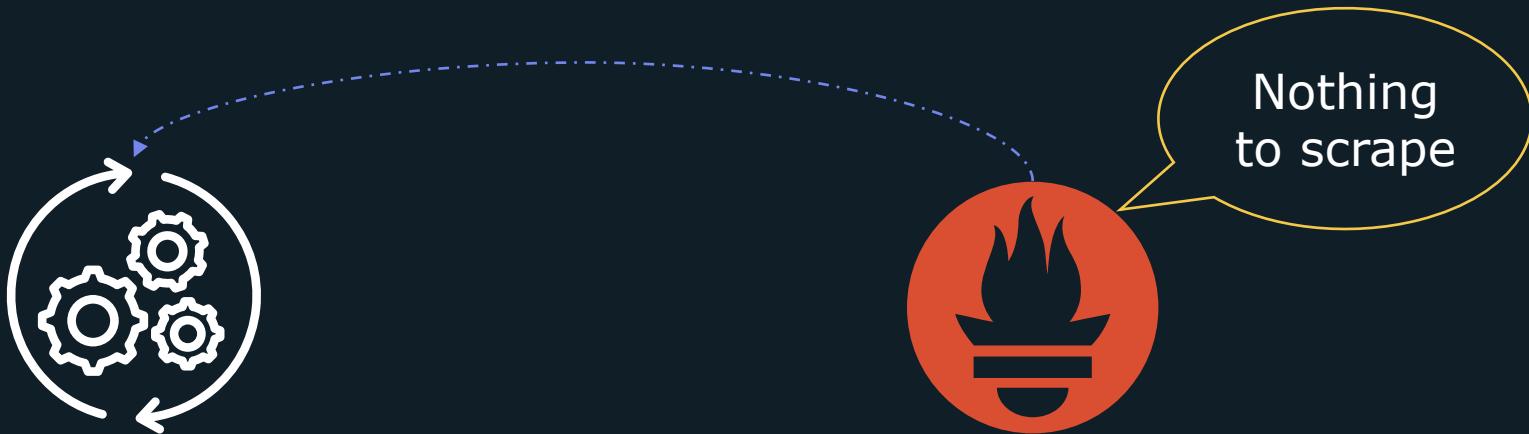


KodeKloud

Push Gateway

Push Gateway

Batch Jobs only run for a short period of time and then exit which makes it difficult for Prometheus to scrape



Push Gateway

Push Gateway acts as the middleman between the batch job and the Prometheus server





KodeKloud

Installation

Installation

The screenshot shows a web browser window with the title "Download | Prometheus". The address bar contains the URL <https://prometheus.io/download/#pushgateway>. The main content area displays information about the "pushgateway" component, including its description as a "Push acceptor for ephemeral and batch jobs" and a link to the GitHub repository [prometheus/pushgateway](#). A table lists the available releases, specifically version 1.4.3 from May 30, 2022. The table includes columns for File name, OS, Arch, Size, and SHA256 Checksum.

File name	OS	Arch	Size	SHA256 Checksum
pushgateway-1.4.3.darwin-amd64.tar.gz	darwin	amd64	9.12 MiB	7c858bcdb280f18139d03a3f20c2edff11e0919a68aa8dfe6d5fa1083139014b
pushgateway-1.4.3.linux-amd64.tar.gz	linux	amd64	9.25 MiB	a8aa4bc0612b44c700454357fd4b23d687ff727f5fdc32e1fd9d48b9ce9a9301
pushgateway-1.4.3.windows-amd64.zip	windows	amd64	9.43 MiB	da83efc91d8244921ebb4b412c3992b9ff1c3a657a44b35ddd34bd215764b0e2

>_

```
$ wget  
https://github.com/prometheus/pushgateway/releases/download/v1.4.3/pushgateway-  
1.4.3.linux-amd64.tar.gz  
  
$ tar xvf pushgateway-1.4.3.linux-amd64.tar.gz  
  
$ cd pushgateway-1.4.3.linux-amd64  
  
$ ./pushgateway  
ts=2022-10-14T03:56:37.848Z caller=main.go:85 level=info msg="starting pushgateway"  
version="(version=1.4.3, branch=HEAD, revision=f9dc1c8664050edbc75916c3664be1df595a1958)"  
ts=2022-10-14T03:56:37.848Z caller=main.go:86 level=info build_context="(go=go1.18.2,  
user=root@75e397dd33fe, date=20220530-19:02:00)"  
ts=2022-10-14T03:56:37.848Z caller=main.go:139 level=info listen_address=:9091  
ts=2022-10-14T03:56:37.849Z caller=tls_config.go:195 level=info msg="TLS is disabled." http2=false
```

Installation

>_

```
$ sudo useradd --no-create-home --shell /bin/false pushgateway  
$ sudo cp pushgateway /usr/local/bin  
$ chown pushgateway:pushgateway /usr/local/bin/pushgateway
```

>_

```
$ vi /etc/systemd/system/pushgateway.service
```

>_

pushgateway.service

[Unit]

Description=Prometheus Pushgateway
Wants=network-online.target
After=network-online.target

[Service]

User=pushgateway
Group=pushgateway
Type=simple
ExecStart=/usr/local/bin/pushgateway

[Install]

WantedBy=multi-user.target

>_

```
$ sudo systemctl daemon-reload
```

```
$ sudo systemctl restart pushgateway
```

```
$ sudo systemctl enable pushgateway
```

```
$ systemctl status pushgateway
```

- pushgateway.service - Prometheus Pushgateway
 Loaded: loaded (/etc/systemd/system/pushgateway.service; enabled; vendor preset: enabled)
 Active: active (running) since Fri 2022-10-14 00:19:52 EDT; 5s ago
 Main PID: 536942 (pushgateway)
 Tasks: 5 (limit: 8247)
 Memory: 3.8M
 CPU: 7ms
 CGroup: /system.slice/pushgateway.service
 └─536942 /usr/local/bin/pushgateway

Installation

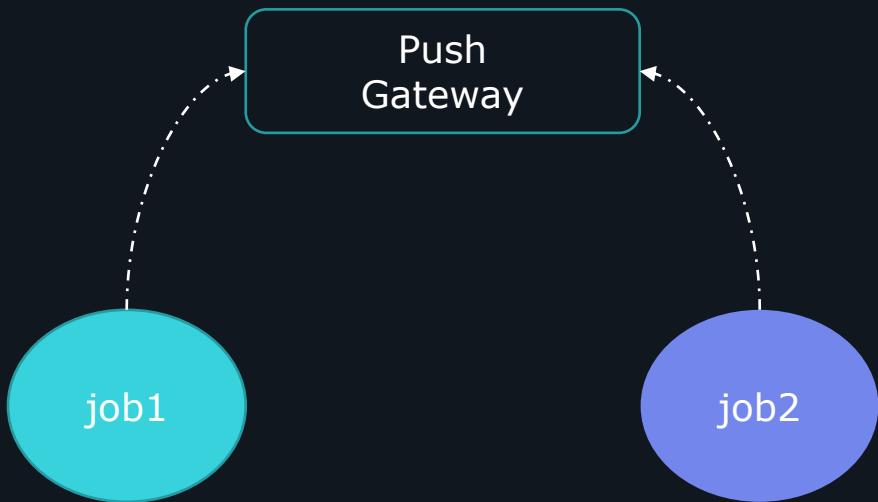
>_

```
$ curl localhost:9091/metrics
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.66572119292e+09
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 7.3271296e+08
# HELP process_virtual_memory_max_bytes Maximum amount of virtual memory available in bytes.
# TYPE process_virtual_memory_max_bytes gauge
process_virtual_memory_max_bytes 1.8446744073709552e+19
# HELP pushgateway_build_info A metric with a constant '1' value labeled by version, revision, branch, and goversion from which pushgateway was built.
# TYPE pushgateway_build_info gauge
pushgateway_build_info{branch="HEAD",goverversion="go1.18.2",revision="f9dc1c8664050edbc75916c3664be1df595a1958",version="1.4.3"} 1
```

Installation

>_

```
$ vi prometheus.yml
```



>_

```
pushgateway.service
```

```
scrape_configs:  
  - job_name: pushgateway  
    honor_labels: true  
    static_configs:  
      - targets: ["192.168.1.168:9091"]
```



KodeKloud

Pushing Metrics

Metrics can be **pushed** to the Pushgateway using one of the following methods:

- Send HTTP Requests to Pushgateway
- Prometheus Client Libraries

Send HTTP POST request using the following URL :

```
http://<pushgateway_address>:<port>/metrics/job/<job_name>/<label1>/<value1>/<label2>/<value2>
```

<job_name> will be the job label of the metrics pushed

<label>/<value> in the url path is used as a grouping key – allows for grouping metrics together to update/delete multiple metrics at once

Push metric "example_metric 4421" with a job label of {job="db_backup"}

```
$ echo "example_metric 4421" | curl --data-binary @-  
http://localhost:9091/metrics/job/db_backup
```

Metric data has to be passed in as binary data with the **-data-binary** flag

@- tells curl to read the binary data from std-in which is the output of the echo command

Pushing Metrics

>_

```
$ curl localhost:9091/metrics
```

```
example_metric{instance="",job="db_backup"} 4421
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.0373e-05
go_gc_duration_seconds{quantile="0.25"} 2.5597e-05
go_gc_duration_seconds{quantile="0.5"} 2.7005e-05
go_gc_duration_seconds{quantile="0.75"} 3.787e-05
```

Pushing Metrics

>_

```
$ cat <<EOF | curl --data-binary @-
http://localhost:9091/metrics/job/job1/instance/instance1
# TYPE my_job_duration gauge
my_job_duration{label="val1"} 42
# TYPE another_metric counter
# HELP another_metric Just an example.
another_metric 12
EOF
```



KodeKloud

Grouping

URL

```
http://<pushgateway_address>:<port>/metrics/job/<job_name>/<label1>/<value1>/<label2>/<value2>
```

The URL path(job_name + labels) acts as a **grouping** key

All metrics sent to a specific URL path are part of a “group”

Groups allow you to update/delete multiple metrics at once

Grouping

>_

```
$ cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/archive/db/mysql
# TYPE metric_one counter
metric_one{label="val1"} 11
# TYPE metric_two gauge
# HELP metric_two Just an example.
metric_two 100
EOF
```

The code block shows a single group of metrics. A yellow bracket labeled "Group 1" covers the entire block from the first metric to the EOF. A yellow arrow points from the "metric_two" line to the bracket.

```
$ cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/archive/app/web
# TYPE metric_one counter
metric_one{label="val1"} 22
# TYPE metric_two gauge
# HELP metric_two Just an example.
metric_two 200
EOF
```

The code block shows two groups of metrics. The first group (blue) contains metric_one and metric_two. The second group (blue) also contains metric_one and metric_two. Both groups are enclosed in blue brackets labeled "Group 1" and "Group 2" respectively. Blue arrows point from the "metric_two" lines to their respective brackets.

Grouping

>_

```
$ curl localhost:9091/metrics | grep archive
```

```
metric_one{db="mysql",instance="",job="archive",label="val1"} 11
metric_two{db="mysql",instance="",job="archive"} 100
```

```
metric_one{app="web",instance="",job="archive",label="val1"} 22
metric_two{app="web",instance="",job="archive"} 200
```

POST

When sending a POST request, only metrics with the same name as the newly pushed metrics are replaced(this only applies to metrics in same group)

```
metric_one{db="mysql",instance="",job="archive",label="val1"} 11
metric_two{db="mysql",instance="",job="archive"} 100
metric_one{app="web",instance="",job="archive",label="val1"} 22
metric_two{app="web",instance="",job="archive"} 200
```

```
$ cat <<EOF | curl --data-binary @- http://localhost:9091/metrics/job/archive/app/web
# TYPE metric_one counter
metric_one{label="val1"} 44
EOF
```

POST

```
metric_one{db="mysql",instance="",job="archive",label="val1"} 11
metric_two{db="mysql",instance="",job="archive"} 100
metric_one{app="web",instance="",job="archive",label="val1"} 44
metric_two{app="web",instance="",job="archive"} 200
```

Updated **metric_one** in the app/web group and kept all other metrics in that group in place

PUT

When sending a PUT request, all metrics within a specific group get replaced by the new metrics being pushed

```
metric_one{db="mysql",instance="",job="archive",label="val1"} 11
metric_two{db="mysql",instance="",job="archive"} 100
metric_one{app="web",instance="",job="archive",label="val1"} 22
metric_two{app="web",instance="",job="archive"} 200
metric_three{app="web",instance="",job="archive"} 300
```

Removed

```
$ cat <<EOF | curl -X PUT --data-binary @- http://localhost:9091/metrics/job/archive/app/web
# TYPE metric_one counter
metric_one{label="val1"} 44
EOF
```

PUT

```
metric_one{db="mysql",instance="",job="archive",label="val1"} 11
metric_two{db="mysql",instance="",job="archive"} 100
metric_one{app="web",instance="",job="archive",label="val1"} 44
```

Replaced **all** metrics within the app/web group with the metrics that were sent in PUT request

DELETE

Delete request will delete all metrics within a group

```
metric_one{db="mysql",instance="",job="archive",label="val1"} 11
metric_two{db="mysql",instance="",job="archive"} 100
metric_one{app="web",instance="",job="archive",label="val1"} 22
metric_two{app="web",instance="",job="archive"} 200
metric_three{app="web",instance="",job="archive"} 300
```

```
$ curl -X DELETE http://localhost:9091/metrics/job/archive/app/web
```

DELETE

```
metric_one{db="mysql",instance="",job="archive",label="val1"} 11
metric_two{db="mysql",instance="",job="archive"} 100
```

Deleted all metrics within the
app/web group



KodeKloud

Client Library

Metrics can be pushed to the Pushgateway through client libraries

There are 3 functions within a client library to push metrics:

push

Any existing metrics for this job are removed
and the pushed metrics are added

PUT

pushadd

Pushed metrics override existing metrics with same
names. All other metrics in group remain unchanged

POST

delete

All metrics for a group are removed

DELETE

Client Library

>_

Main.py

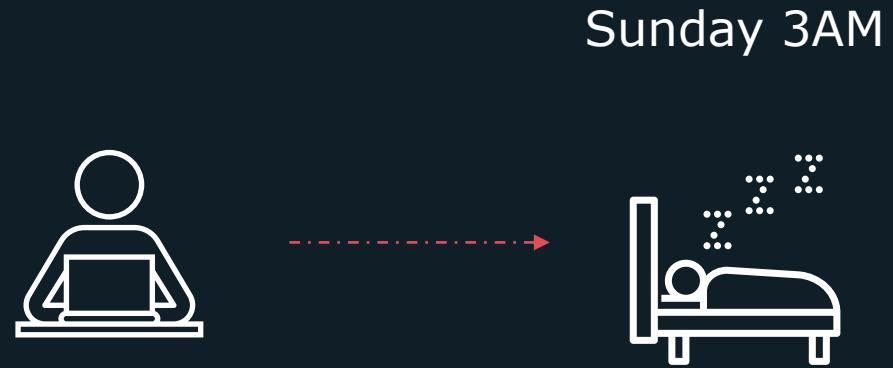
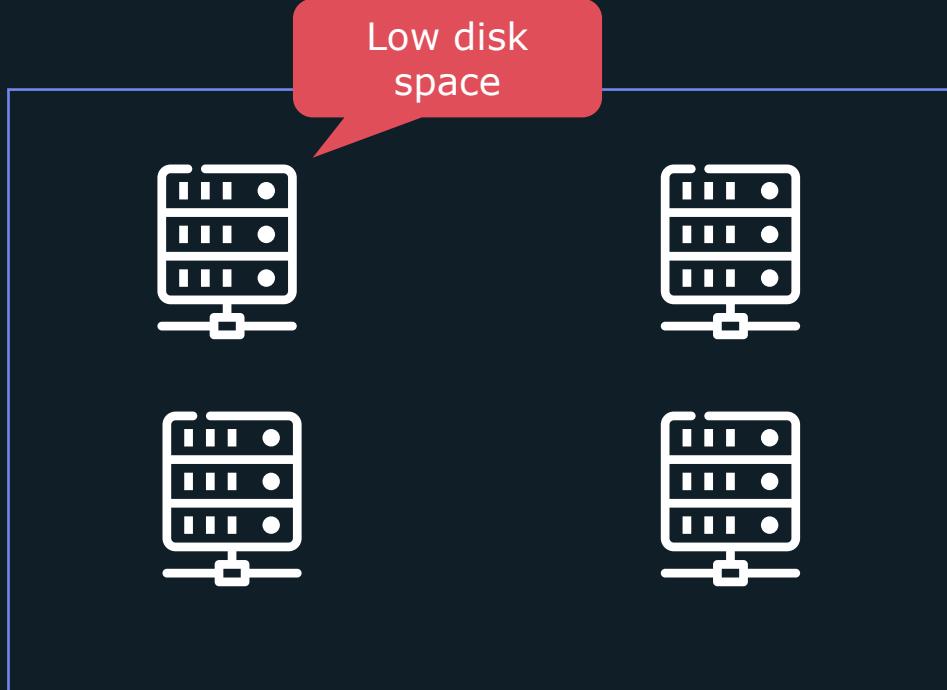
```
from prometheus_client import CollectorRegistry, Gauge,  
pushadd_to_gateway  
  
registry = CollectorRegistry()  
test_metric = Gauge('test_metric',  
                    'This is an example metric',  
                    registry=registry)  
  
test_metric.set(10)  
  
pushadd_to_gateway('user2:9091', job='batch',  
                   registry=registry)
```



KodeKloud

Alerting

Alerting



We need some sort of mechanism to
alert administrators when problems
occur



Alerting

Prometheus allows you to define **conditions**, that if met trigger alerts

These conditions are defined as standard **PromQL** expressions

```
$ node_filesystem_avail_bytes < 1000
```

Trigger alert

```
node_filesystem_avail_bytes{device="tmpfs", instance="node1", mountpoint="/run/lock"} 547
```

Any resulting vectors from the query will trigger an alert

```
$ node_filesystem_avail_bytes < 2000
```

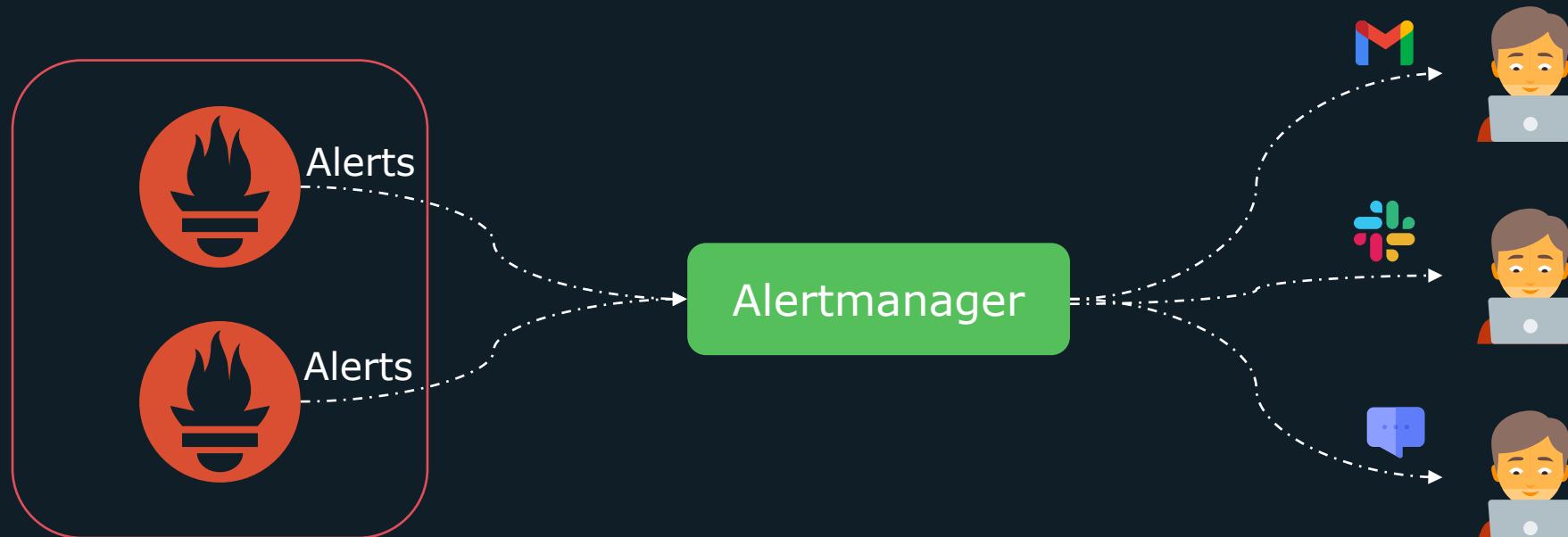
2 alerts

```
node_filesystem_avail_bytes{device="tmpfs", instance="node1", mountpoint="/run/lock"} 547  
node_filesystem_avail_bytes{device="/dev/sda2", instance="node1", mountpoint="/" } 1228
```

Alerting

Prometheus is only responsible for triggering alerts.
It does **not** send notifications such as emails, text messages.

That responsibility is offloaded onto a separate process called **Alertmanager**



Alerting Rules

Alerting rules are similar to recording rules and can be placed right along side recording rules within a rule group

```
>_          rules.yml

groups:
  - name: node
    interval: 15s
    rules:
      - record: node_memory_memFree_percent
        expr: 100 - (100 *
node_memory_MemFree_bytes{job="node"} /
node_memory_MemTotal_bytes{job="node"})
      - alert: LowMemory
        expr: node_memory_memFree_percent < 20
```

```
>_          rules.yml

groups:
  - name: node
    rules:
      - alert: node down
        expr: up{job="node"} == 0
        for: 5m
```

The **for** clause tells Prometheus that an expression must evaluate to true for a specified period of time before firing an alert

This example expects the node to be down for **5 minutes** before firing an alert

Helps prevent against race conditions and scrape timeouts. Network issues could cause an individual scrape to fail, it's best to specify a duration to avoid false alarms

Alerts

The screenshot shows the Prometheus Alerts interface. At the top, there are three filter buttons: "Inactive (2)" (green), "Pending (0)" (yellow), and "Firing (0)" (red). Below the filters, a search bar contains the path "/etc/prometheus/rules.yml > node". The main content area displays two alert rules:

- LowMemory (0 active)**

```
name: LowMemory
expr: node_memory_memFree_percent{job="node"} < 20
for: 3m
```
- Node down (0 active)**

```
name: Node down
expr: up{job="node"} == 0
for: 3m
```

Alert States

```
>_          rules.yml
```

```
groups:  
  - name: node  
    rules:  
      - record: node down  
        expr: up{job="node"} == 0  
        for: 5m
```

Inactive – Expression has not returned any results

Pending – expression returned results but it hasn't been long enough to be considered firing(5m in this case)

Firing – Active for more than the defined for clause(5m in this case)

› LowMemory (0 active)

▼ Node down (1 active)

```
name: Node down  
expr: up{job="node"} == 0  
for: 3m
```

Labels	State
alertname=Node down instance=192.168.1.168:9200 job=node	PENDING

› LowMemory (0 active)

▼ Node down (1 active)

```
name: Node down  
expr: up{job="node"} == 0  
for: 3m
```

Labels	State
alertname=Node down instance=192.168.1.168:9200 job=node	FIRING



KodeKloud

Labels

Labels

Labels can be added to alerts to provide a mechanism to classify and match specific alerts in Alertmanager

```
>_
```

```
rules.yml
```

```
groups:
  - name: node
    rules:
      - alert: Node down
        expr: up{job="node"} == 0
        labels:
          severity: warning

      - alert: Multiple Nodes down
        expr: avg_without(instance)(up{job="node"}) <= 0.5
        labels:
          severity: critical
```

Labels

▼ Node down (1 active)

```
name: Node down
expr: up{job="node"} == 0
for: 3m
labels:
    severity: warning
```

Labels

```
alertname=Node down instance=192.168.1.168:9200 job=node severity=warning
```

▼ Multiple Nodes down (1 active)

```
name: Multiple Nodes down
expr: avg without(instance) (up{job="node"}) <= 0.5
labels:
    severity: critical
```

Labels

```
alertname=Multiple Nodes down job=node severity=critical
```

Annotations can be used to provide additional information, however unlike labels they do no play a part in the alerts identity
So they cannot be used for routing in Alertmanager

Annotations are templated using Go templating language.

To access alert labels use `{{.Labels}}`

To get instance label use `{{.Labels.instance}}`

To get the firing sample value use `{{.Value}}`

Annotations

```
>_                               rules.yml

groups:
  - name: node
    rules:
      - alert: node_filesystem_free_percent
        expr: 100 * node_filesystem_free_bytes{job="node"} /
node_filesystem_size_bytes{job="node"} < 70
        annotations:
          description: "filesystem {{.Labels.device}} on {{.Labels.instance}}
is low on space, current available space is {{.Value}}"
```

Annotations

✓ node_filesystem_free_percent (2 active)

```
name: node_filesystem_free_percent
expr: 100 * node_filesystem_free_bytes{job="node"} / node_filesystem_size_bytes{job="node"} < 70
annotations:
  description: filesystem {{.Labels.device}} on {{.Labels.instance}} is low on space, current available space is {{.Value}}
```

Labels	State	Active Since	Value
alername=node_filesystem_free_percent device=/dev/sda3 fstype=ext4 instance=192.168.1.168:9100 job=node mountpoint=/var/snap/firefox/common/host-hunspell	FIRING	2022-10-04T07:38:39.856695312Z	20.40345164803414

Annotation

Filesystem /dev/sda3 on 192.168.1.168:9100 is low on space, current available space is 20.40345



KodeKloud

Alertmanager

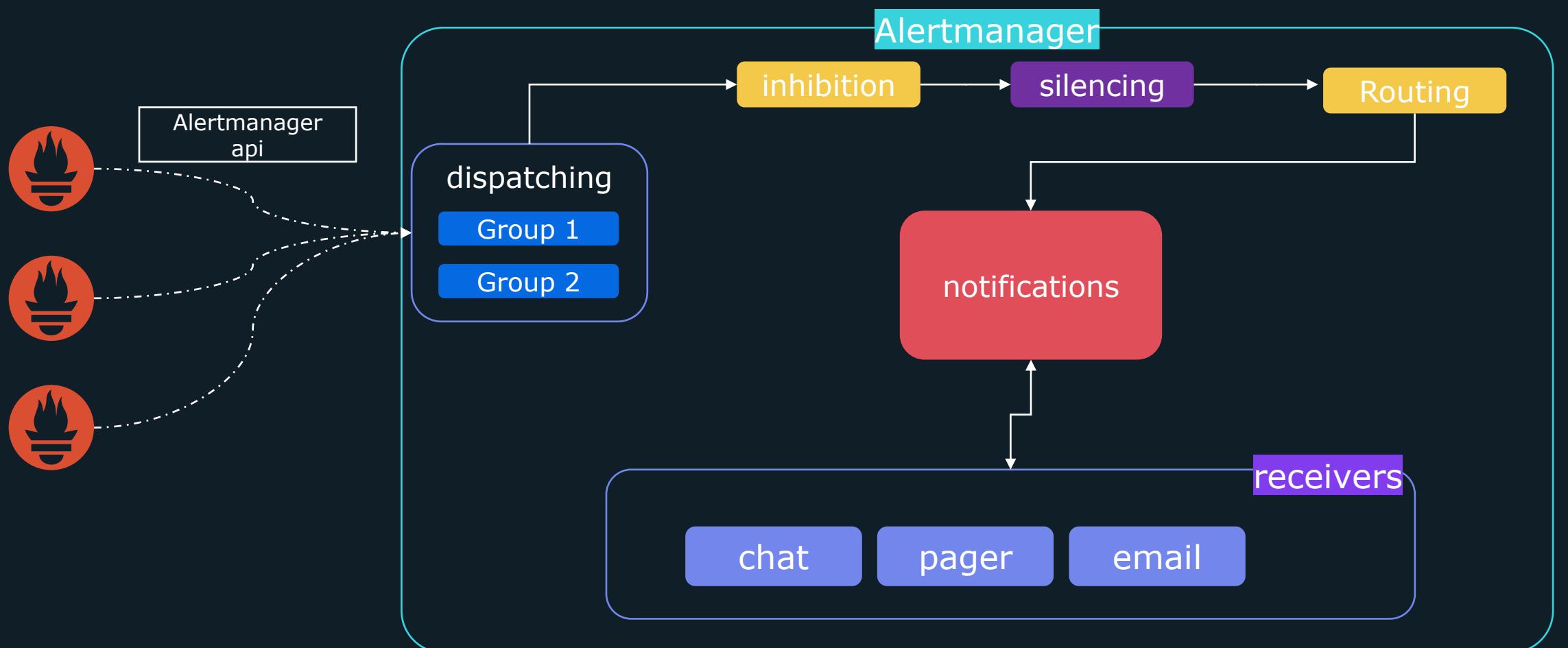


| What is Alertmanager

Alertmanager is responsible for receiving alerts generated from Prometheus and converting them into notifications.

These notifications can include, pages, webhooks, email messages, and chat messages

Alertmanager architecture

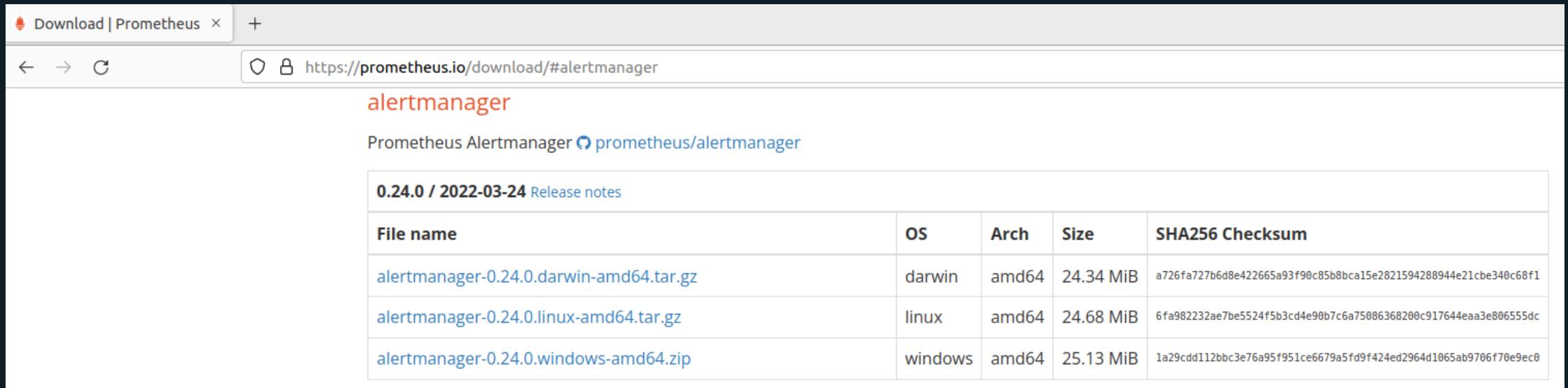




KodeKloud

Alertmanager Installation

Installation



The screenshot shows a web browser window with the following details:

- Title Bar:** Download | Prometheus
- Address Bar:** https://prometheus.io/download/#alertmanager
- Page Content:** alertmanager
- Sub-Content:** Prometheus Alertmanager [prometheus/alertmanager](#)
- Section:** 0.24.0 / 2022-03-24 [Release notes](#)
- Table:** A table listing the available Alertmanager binary releases. The columns are: File name, OS, Arch, Size, and SHA256 Checksum.

File name	OS	Arch	Size	SHA256 Checksum
alertmanager-0.24.0.darwin-amd64.tar.gz	darwin	amd64	24.34 MiB	a726fa727b6d8e422665a93f90c85b8bca15e2821594288944e21cbe340c68f1
alertmanager-0.24.0.linux-amd64.tar.gz	linux	amd64	24.68 MiB	6fa982232ae7be5524f5b3cd4e90b7c6a75086368200c917644eaa3e806555dc
alertmanager-0.24.0.windows-amd64.zip	windows	amd64	25.13 MiB	1a29cdd112bbc3e76a95f951ce6679a5fd9f424ed2964d1065ab9706f70e9ec0

Installation

```
>_  
$ wget  
https://github.com/prometheus/alertmanager/releases/download/v0.24.0/alert  
manager-0.24.0.linux-amd64.tar.gz  
curl: (60) SSL certificate problem: self-signed certificate
```

```
$ tar xzf alertmanager-0.24.0.linux-amd64.tar.gz
```

```
$ cd alertmanager-0.24.0.linux-amd64
```

installation

>_

```
$ ls -l  
total 55752  
-rwxr-xr-x 1 user1 user1 31988661 Mar 25 2022 alertmanager  
-rw-r--r-- 1 user1 user1      356 Mar 25 2022 alertmanager.yml  
-rwxr-xr-x 1 user1 user1 25067944 Mar 25 2022 amtool  
drwxrwxr-x 2 user1 user1     4096 Aug 23 19:21 data  
-rw-r--r-- 1 user1 user1    11357 Mar 25 2022 LICENSE  
-rw-r--r-- 1 user1 user1      457 Mar 25 2022 NOTICE
```

Alertmanager executable

configuration file

command-line utility tool

stores notification states

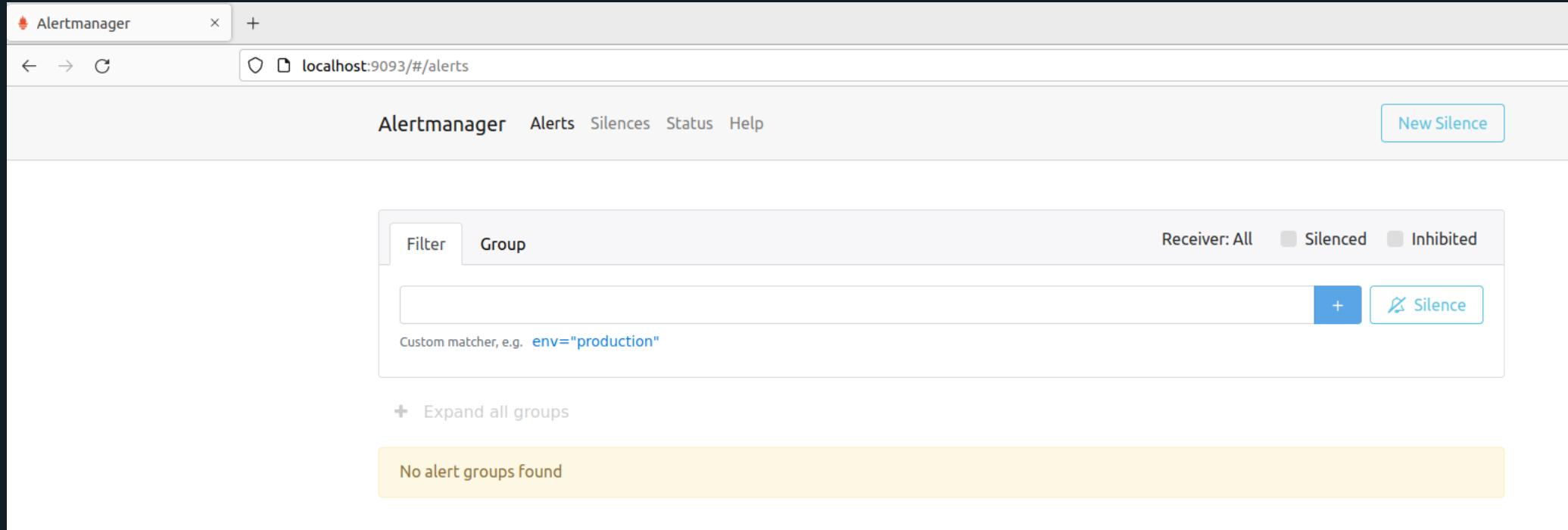
Installation

>_

```
$ ./alertmanager
```

```
ts=2022-10-04T20:57:44.014Z caller=cluster.go:680 level=info component=cluster msg="Waiting for gossip to settle..." interval=2s
ts=2022-10-04T20:57:44.065Z caller=coordinator.go:113 level=info component=configuration msg="Loading configuration file" file=alertmanager.yml
ts=2022-10-04T20:57:44.065Z caller=coordinator.go:126 level=info component=configuration msg="Completed loading of configuration file" file=alertmanager.yml
ts=2022-10-04T20:57:44.068Z caller=main.go:535 level=info msg=Listening address=:9093
ts=2022-10-04T20:57:44.068Z caller=tls_config.go:195 level=info msg="TLS is disabled." http2=false
ts=2022-10-04T20:57:46.014Z caller=cluster.go:705 level=info component=cluster msg="gossip not settled" polls=0 before=0 now=1 elapsed=2.000258318s
ts=2022-10-04T20:57:54.017Z caller=cluster.go:697 level=info component=cluster msg="gossip settled; proceeding" elapsed=10.002673562s
```

Installation



installation

```
>_          prometheus.yml

global:
  scrape_interval: 15s
  evaluation_interval: 15s
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager1:9093
            - alertmanager2:9093
rule_files:
  - "rules.yml"
scrape_configs:
```



KodeKloud

Alertmanager Installation Systemd

Installation

>_

```
$ wget  
https://github.com/prometheus/alertmanager/releases/download/v0.24.0/alert  
manager-0.24.0.linux-amd64.tar.gz
```

```
curl: (60) SSL certificate problem: self-signed certificate
```

```
$ tar xzf alertmanager-0.24.0.linux-amd64.tar.gz
```

```
$ cd alertmanager-0.24.0.linux-amd64
```

Installation

>_

```
$ sudo useradd --no-create-home --shell /bin/false alertmanager
```



User Can't log in

```
$ sudo mkdir /etc/alertmanager
```

```
$ sudo mv alertmanager.yml /etc/alertmanager
```

```
$ sudo chown -R alertmanager:alertmanager /etc/alertmanager
```

Installation

>_

```
$ sudo mkdir /var/lib/alertmanager
```

Alertmanager data

```
$ sudo chown -R alertmanager:alertmanager /var/lib/alertmanager
```

Installation

>_

```
$ sudo cp alertmanager /usr/local/bin  
$ sudo cp amtool /usr/local/bin  
  
$ sudo chown alertmanager:alertmanager /usr/local/bin/alertmanager  
$ sudo chown alertmanager:alertmanager /usr/local/bin/amtool
```

Installation

```
>_  
$ sudo vi /etc/systemd/system/alertmanager.service  
  
$ sudo systemctl daemon-reload  
  
$ sudo systemctl start alertmanager  
  
$ sudo systemctl enable alertmanager
```

>_ Alertmanager.service

```
[Unit]  
Description=Alert Manager  
Wants=network-online.target  
After=network-online.target  
[Service]  
Type=simple  
User=alertmanager  
Group=alertmanager  
ExecStart=/usr/local/bin/alertmanager \  
    --config.file=/etc/alertmanager/alertmanager.yml \  
    --storage.path=/var/lib/alertmanager  
Restart=always  
[Install]  
WantedBy=multi-user.target
```



KodeKloud

Configuration

Configuration

```
>_      alertmanager.yml
```

```
global:  
  smtp_smarthost: 'mail.example.com:25'  
  smtp_from: 'test@example.com'
```

```
route:  
  receiver: staff  
  group_by: ['alertname', 'job']  
  routes:  
    - match_re:  
        job: (node|windows)  
        receiver: infra-email  
    - matchers:  
        job: kubernetes  
        receiver: k8s-slack
```

```
receivers:  
  - name: 'k8s-slack'  
    slack_configs:  
      - channel: '#alerts'  
        text: 'https://example.com/alerts/{{ .GroupLabels.app }}/
```

Global section applies global configuration across **all** sections which can be overwritten

The route section provides a set of **rules** to determine what alerts get matched up with which receivers

A receiver contains one or more notifiers to **forward** alerts to users

Default Route

```
>_      alertmanager.yml
```

```
route:  
  receiver: staff  
  group_by: ['alertname', 'job']  
  
routes:  
- match_re:  
    job: (node|windows)  
    receiver: infra-email  
- matchers:  
    job: kubernetes  
    receiver: k8s-slack
```

At the top level there is a fallback/default route.

Any alerts that don't match any of the other routes will match the default route.

These alerts will be grouped by labels **alertname**, and **job**

These alerts will go to the receiver **staff**

Route

```
>_      alertmanager.yml
```

```
route:  
  routes:  
    - match_re:  
        job: (node|windows)  
        receiver: infra-email  
    - matchers:  
        job: kubernetes  
        severity: ticket  
  receiver: k8s-slack
```

Every route has a list match statement and a receiver for alerts that match

In this example, all alerts with the **job=Kubernetes** & **severity=ticket** labels will match this rule

These alerts that match will get sent to receiver **k8s-slack**

```
>_      alertmanager.yml  
  
route:  
  routes:  
    - match_re:  
        job: (node|windows)  
        receiver: infra-email  
    - matchers:  
        job: kubernetes  
        severity: ticket  
        receiver: k8s-slack
```

If the `match_re` keyword is used instead of `matchers`, then a list of regular expressions can be provided to match specific labels

In this example, the rule will match any alert with a label of `job=node` or `job=windows`

Sub-Routes

```
>_      alertmanager.yml

route:
  routes:
    - matchers:
        job: kubernetes
        receiver: k8s-email
    routes:
      - matchers:
          severity: pager
          receiver: k8s-pager
```

All alerts with label of **job=kubernetes** will be sent to receiver **k8s-email**

However we can configure sub-routes for further route matching.

If the alert has a label **severity=page** then it will be sent to **k8s-pager**.

All other alerts with label **job=kubernetes** will match the main route and be sent to **k8s-email**

>_ alertmanager.yml

```
route:  
  routes:  
    # database team.  
    - match:  
        team: database  
        receiver: database-pager ←  
      routes:  
        - match:  
            severity: page  
            receiver: database-pager  
        - match:  
            severity: email  
            receiver: database-email  
    # api team.  
    - match:  
        team: api  
        receiver: api-pager ←  
      routes:  
        - match:  
            severity: page  
            env: dev  
            receiver: api-ticket  
        - match:  
            severity: page  
            receiver: api-pager  
        - match:  
            severity: ticket  
            receiver: api-ticket
```

Restarting Alertmanager

As with Prometheus, Alertmanager does not automatically pickup changes to its config file. One of the following must be performed, for changes to take effect:

- Restart Alertmanager
- Send a SIGHUP
 - Sudo killall -HUP alertmanager
- HTTP Post to /-/reload endpoint

```
>_ alertmanager.yml
```

```
route:  
  routes:  
    - receiver: alert-logs  
      continue: true  
    - matcher:  
        job: kubernetes  
      receiver: k8s-email
```

What if we want an alert to match two routes.

In this example, we want all alerts to be sent to the receiver **alert-logs** and then if it has the label **jobs=Kubernetes** it should also match the following line and be sent to **k8s-email** receiver

By default, the **first** matching route wins. So a specific alert would stop going down the route tree after the first match

To continue processing down the tree we can add **continue: true**

Grouping

```
>_ alertmanager.yml

route:
  receiver: fallback-pager
  [group_by: [team]]
  routes:
    - match:
        team: infra
        [group_by: [region, env]]
        receiver: infra-email
      routes:
        - match:
            severity: page
        receiver: infra-pager
```

By default Alertmanager will group all alerts for a route into a **single** group, which results in you receiving one big notification

The **group_by** field allows you to specify a list of labels to group alerts by

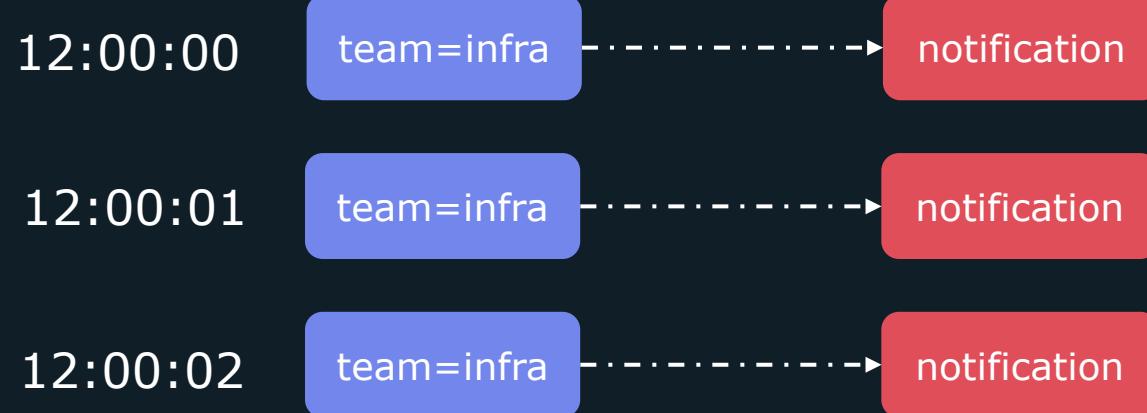
A default route has its alerts grouped by the **team label**

The infra team has alerts grouped based on **region** and **env** labels.

The child routes will inherit the grouping policy and group based on same 2 labels

Batching & throttling

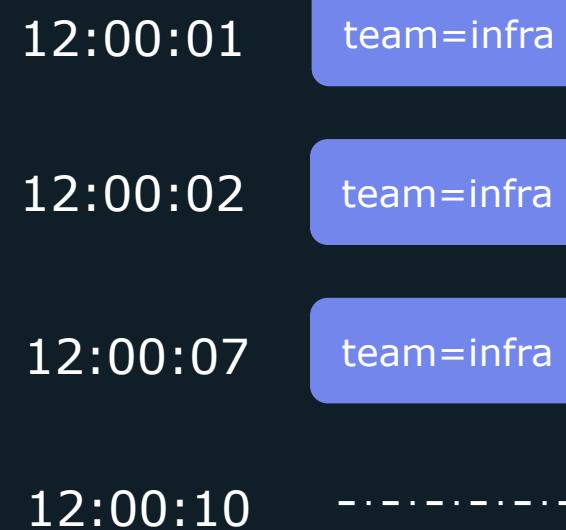
```
>_ alertmanager.yml  
  
route:  
  receiver: fallback-pager  
  group_by: [team]
```



We need some sort of ability to batch and throttle alerts to avoid being slammed with too many notifications

Group wait

```
>_ alertmanager.yml  
  
route:  
  receiver: fallback-pager  
  group_by: [team]  
  group_wait: 10s
```



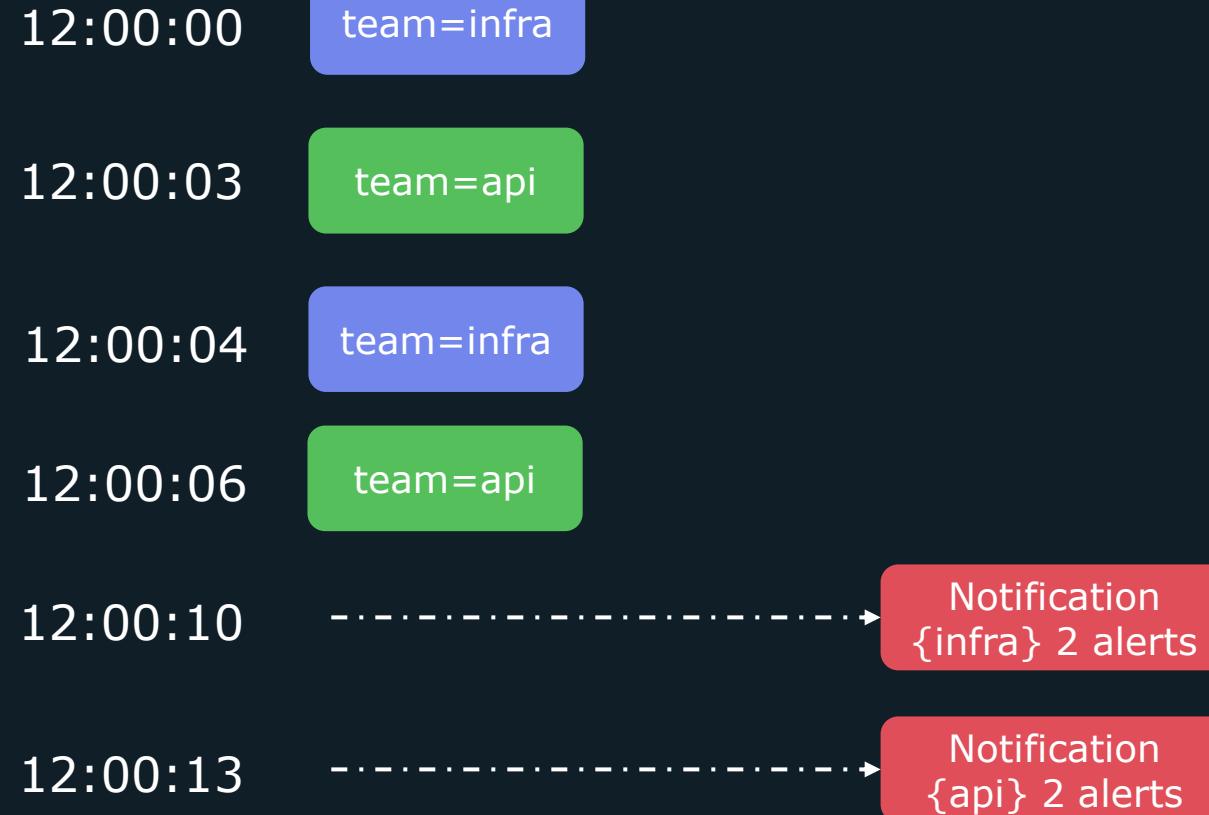
Group_wait – specifies the time alertmanager will wait after receiving an alert for a group before firing off a notification. Default is 30s

This helps reduce the number of notifications for the same problem

Group wait

```
>_ alertmanager.yml
```

```
route:  
  receiver: fallback-pager  
  group_by: [team]  
  group_wait: 10s
```



Group Interval

```
>_ alertmanager.yml
```

```
route:  
  receiver: fallback-pager  
  group_by: [team]  
  group_wait: 10s  
  group_interval: 5m
```



How long till new notification is sent??

Wait 5 minutes



Repeat interval

```
>_ alertmanager.yml
```

```
route:  
  receiver: fallback-pager  
  group_by: [team]  
  group_wait: 10s  
  group_interval: 5m  
  repeat_interval: 4h
```

12:00:00

team=infra

12:00:04

team=infra

12:00:10

Notification
{infra} 2 alerts

There's a possibility that an engineer received a notification but forgot to address it due to being preoccupied with bigger issues.

We can have alertmanager resend notifications periodically

4:00:10

Notification
{infra} 2 alerts



KodeKloud

Receivers & Notifiers

Receivers

Receivers are responsible for taking grouped alerts and producing notifications

Receivers contain various notifiers, which are responsible for handling the actual notifications



Receivers

```
>_          alertmanager.yml

route:
  receiver: infra-pager
receivers:
- name: infra-pager
  slack_configs:
    - api_url: https://hooks.slack.com/services/XXXXXXXXXX
      channel: "#pages"
  email_configs:
    - to: "receiver_mail_id@gmail.com"
      from: "mail_id@gmail.com"
      smarthost: smtp.gmail.com:587
      auth_username: "mail_id@gmail.com"
      auth_identity: "mail_id@gmail.com"
      auth_password: "password"
```

Configuration

<https://prometheus.io/docs/alerting/latest/configuration>

The screenshot shows the Prometheus documentation website with a dark theme. The top navigation bar includes links for DOCS, DOWNLOAD, COMMUNITY, SUPPORT & TRAINING, and BLOG, along with a search bar and social media icons for GitHub and Twitter.

The main content area is titled "CONFIGURATION". It discusses how Alertmanager is configured via command-line flags and a configuration file. It mentions the visual editor for building routing trees and the process for reloading configuration at runtime.

A section titled "Configuration file" provides details on how to specify the configuration file:

```
./alertmanager --config.file=alertmanager.yml
```

The text explains that the file is written in YAML format and follows specific schema rules. It also defines generic placeholders used in the configuration.

The sidebar on the left lists various documentation sections: INTRODUCTION, CONCEPTS, PROMETHEUS, VISUALIZATION, INSTRUMENTING, OPERATING, and ALERTING. Under ALERTING, the "Configuration" section is currently selected. Other visible items in the sidebar include Clients, Notification template reference, Notification template examples, Management API, HTTPS and authentication, and BEST PRACTICES.

Global Config

```
>_      alertmanager.yml

global:
  [victorops_api_key: XXX]

receivers:
- name: infra-pager
  victorops_configs:
    - routing_key: some-route
```

Certain notifier configs/settings will be the same across all receivers and can be moved to **global** config section

Global Config

```
>_          alertmanager.yml

global:
  smtp_smarthost: 'smtp.gmail.com:587'
  smtp_from: 'alertmanager@kodekloud.com'
  smtp_auth_username: xxxx
  smtp_auth_identity: xxxx
  smtp_auth_password: xxxx
receivers:
  - name: 'infra'
    email_configs:
      - to: 'infra@kodekloud.com'
  - name: 'frontend'
    email_configs:
      - to: 'frontend@kodekloud.com'
  - name: 'k8s'
    email_configs:
      - to: 'k8s@kodekloud.com'
```

Notification Templates

Messages from notifiers can be configured by making use of the Go templating system.

GroupLabels – contains the group labels of the notification

CommonLabels – labels that are common across all alerts in the notification

CommonAnnotations – similar to CommonLabels, but for annotations

ExternalURL – URL of this Alertmanager

Status – will be set to firing if at least one alert in the notification is firing, if all are resolved then it will be set to resolved

Receiver – name of the receiver

Alerts – List of all the alerts in the notification

 Labels – labels for the alert

 Annotations – annotation for the alert

 status – firing|resolved

 StartsAt – When the alert started firing

 EndsAt – When the alert has stopped firing

Notification Templates

```
>_          alertmanager.yml

route:
  receiver: 'slack'
receivers:
- name: slack
  slack_configs:
    - api_url: https://hooks.slack.com/xxx
      channel: '#alerts'
      title: '{{.GroupLabels.severity}} alerts
in region {{.GroupLabels.region}}'
```

`{{.GroupLabels.severity}}`
retrieves the group label severity

`{{.GroupLabels.region}}`
retrieves the group label region

This would return:
“critical” alerts in region “west”

Notification Templates

```
>_          alertmanager.yml

route:
  receiver: 'slack'
receivers:
- name: slack
  slack_configs:
    - api_url: https://hooks.slack.com/xxx
      channel: '#alerts'
      title: '{{.GroupLabels.severity}}
alerts in region {{.GroupLabels.region}}'
      text: '{{.Alerts | len}} alerts:
```

.Alerts returns a list of alerts in the notifications and when we do | len it will return the total number of alerts

Notification Templates

```
>_          alertmanager.yml

route:
  receiver: 'slack'
receivers:
- name: slack
  slack_configs:
    - api_url: https://hooks.slack.com/xxx
      channel: '#alerts'
      title: '{{.GroupLabels.severity}}
alerts in region {{.GroupLabels.region}}'
      text: >
        {{.Alerts | len}} alerts:
        {{range .Alerts}}
{{.Annotations.description }}{{"\n"}}{{end}}
```

When you can't squeeze the template string onto one line use > to place the template below

{{ range .Alerts}}{{end}} will loop through the list of alerts

{{.Annotations.description}}

prints out the description for the specific alert we are currently iterating over

{{"\n"}} adds a new line

Notifications

warning alerts in region west

2 alerts: filesystem /dev/sda3 on 192.168.1.168:9100 is low on space, current available space is 19.79372717237496

filesystem /dev/sda3 on 192.168.1.168:9100 is low on space, current available space is 19.79372717237496

critical alerts in region east

1 alerts: More than half of all nodes are currently down

AlertManager

http://localhost:9093

The screenshot shows the AlertManager interface at `http://localhost:9093`. At the top, there are tabs for "Filter" and "Group", with "Group" being active. To the right, it says "Receiver: All" and has three status indicators: "Silenced" (gray), "Inhibited" (orange), and "Unsilenced" (green). Below these are two buttons: a blue "+" button and a green "Silence" button with a megaphone icon.

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

Expand all groups

- + severity="warning" + 1 alert
- + region="east" + severity="critical" + 1 alert
- + region="west" + severity="warning" + 2 alerts

Alertmanager

```
route:  
  group_by: ['severity', 'region']
```

The screenshot shows the configuration interface for an Alertmanager route. At the top, there are two tabs: "Filter" (which is selected) and "Group". Below the tabs is a text input field with placeholder text: "Custom matcher, e.g. env='production'". Underneath this, there is a section titled "+ Expand all groups". This section contains three items, each represented by a blue-bordered box:

- A single item: "severity='warning'" followed by a "+" sign and "1 alert".
- A combined item: "region='east'" followed by a "+" sign, "severity='critical'" followed by a "+" sign, and "1 alert".
- A combined item: "region='west'" followed by a "+" sign, "severity='warning'" followed by a "+" sign, and "2 alerts".

Alertmanager

- region="west" + severity="warning" + 2 alerts

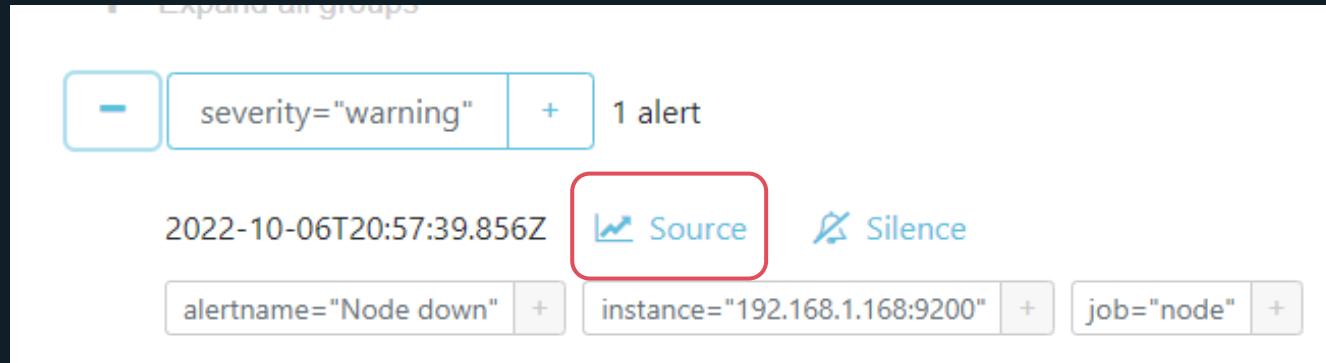
2022-10-06T20:54:39.856Z - Info Source Silence

description: filesystem /dev/sda3 on 192.168.1.168:9100 is low on space, current available space is 19.793696303375587

alertname="node_filesystem_free_percent" + device="/dev/sda3" + fstype="ext4" + instance="192.168.1.168:9100" + job="node" +
mountpoint="/" +

2022-10-06T20:54:39.856Z + Info Source Silence

alertname="node_filesystem_free_percent" + device="/dev/sda3" + fstype="ext4" + instance="192.168.1.168:9100" + job="node" +
mountpoint="/var/snap/firefox/common/host-hunspell" +



The screenshot shows the Prometheus web interface with a dark theme. The top navigation bar includes:

- Prometheus logo
- Alerts
- Graph
- Status
- Help

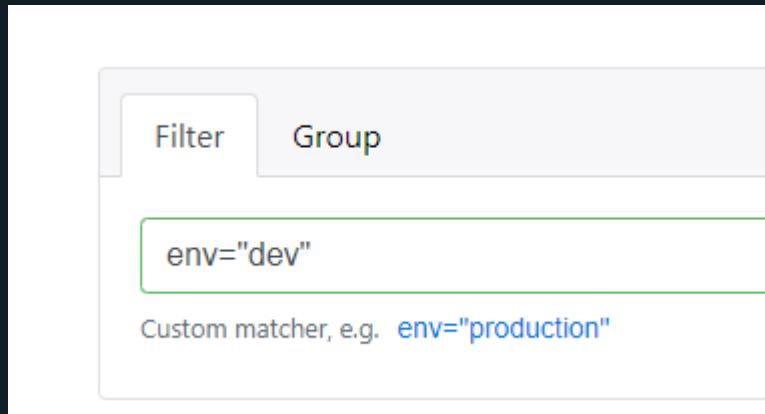
Below the navigation, there are configuration options:

- Use local time
- Enable query history
- Enable autocomplete

The main area displays a search bar with the query: `up{job="node"} == 0`. The results are shown in a table format, with the "Graph" tab selected. A timestamp selector shows "Evaluation time". The result table contains one row:

up{instance="192.168.1.168:9200", job="node"}

At the bottom left is a blue "Add Panel" button.



A screenshot of an alert summary card. The card shows a single alert with the following details:

- Filter applied: region="east" (with a '+' button) and severity="critical" (with a '+' button). To the right of these filters is the text "1 alert".
- Timestamp: 2022-10-06T20:54:39.856Z
- Action buttons: Info, Source, Silence
- Alert labels: alertname="Multiple Nodes down" (with a '+' button), env="dev" (with a '+' button), job="node" (with a '+' button), org="infra" (with a '+' button), and team="infra" (with a '+' button).



KodeKloud

Silences

Silences

Alerts can be **silenced** to prevent generating notifications for a period of time.

This can be useful during maintenance windows when you expect there to be some temporary issues

The screenshot shows two separate alert entries in a monitoring interface. Each entry includes a timestamp, an 'Info' button, a 'Source' link, and a 'Silence' button.

Alert 1 (Top): Occurred at 2022-10-18T00:49:29.499Z. The alert details are:

- Alert name: InstanceDown
- 2 alerts
- Tags: ec2_ami="ami-026b57f3c383c2eec", ec2_architecture="x86_64", ec2_availability_zone="us-east-1b", ec2_availability_zone_id="use1-az4", ec2_instance_id="i-0e23a91e6c6a198db", ec2_instance_state="running", ec2_instance_type="t2.micro", ec2_owner_id="040497317401", ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41", ec2_private_dns_name="ip-172-31-1-156.ec2.internal", ec2_private_ip="172.31.1.156", ec2_public_dns_name="ec2-3-80-117-102.compute-1.amazonaws.com", ec2_public_ip="3.80.117.102", ec2_region="us-east-1", ec2_subnet_id="subnet-0fde5f37cd6c80f41", ec2_tag_Name="web", ec2_tag_env="dev", ec2_vpc_id="vpc-294f5553", instance="172.31.1.156:80", job="ec2", severity="critical".

Alert 2 (Bottom): Occurred at 2022-10-18T00:49:29.499Z. The alert details are:

- Alert name: InstanceDown
- 2 alerts
- Tags: ec2_ami="ami-026b57f3c383c2eec", ec2_architecture="x86_64", ec2_availability_zone="us-east-1b", ec2_availability_zone_id="use1-az4", ec2_instance_id="i-0f15968d505d7bf59", ec2_instance_state="running", ec2_instance_type="t2.micro", ec2_owner_id="040497317401", ec2_primary_subnet_id="subnet-0fde5f37cd6c80f41", ec2_private_dns_name="ip-172-31-1-34.ec2.internal", ec2_private_ip="172.31.1.34", ec2_public_dns_name="ec2-54-242-72-200.compute-1.amazonaws.com", ec2_public_ip="54.242.72.200", ec2_region="us-east-1", ec2_subnet_id="subnet-0fde5f37cd6c80f41", ec2_tag_Name="db", ec2_tag_env="prod", ec2_vpc_id="vpc-294f5553", instance="172.31.1.34:80", job="ec2", severity="critical".

Silences

Alertmanager Alerts **Silences** Status Help

New Silence

Filter

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

Active Pending Expired

No silences found

Silences

New Silence

Start

2022-10-18T01:01:06.080Z

Duration

2h

End

2022-10-18T03:01:06.080Z

**Matchers** Alerts affected by this silence

job="ec2"

Custom matcher, e.g. `env="production"`**Creator**

sanjeev thiagarajan

**Comment**Silencing EC2 alerts during today's maintenance windowPreview AlertsCreateReset

Silences

Filter

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

Active 1 Pending Expired

Ends 2022-10-18T03:01:06.080Z [View](#) [Edit](#) [Expire](#)

`job=ec2`



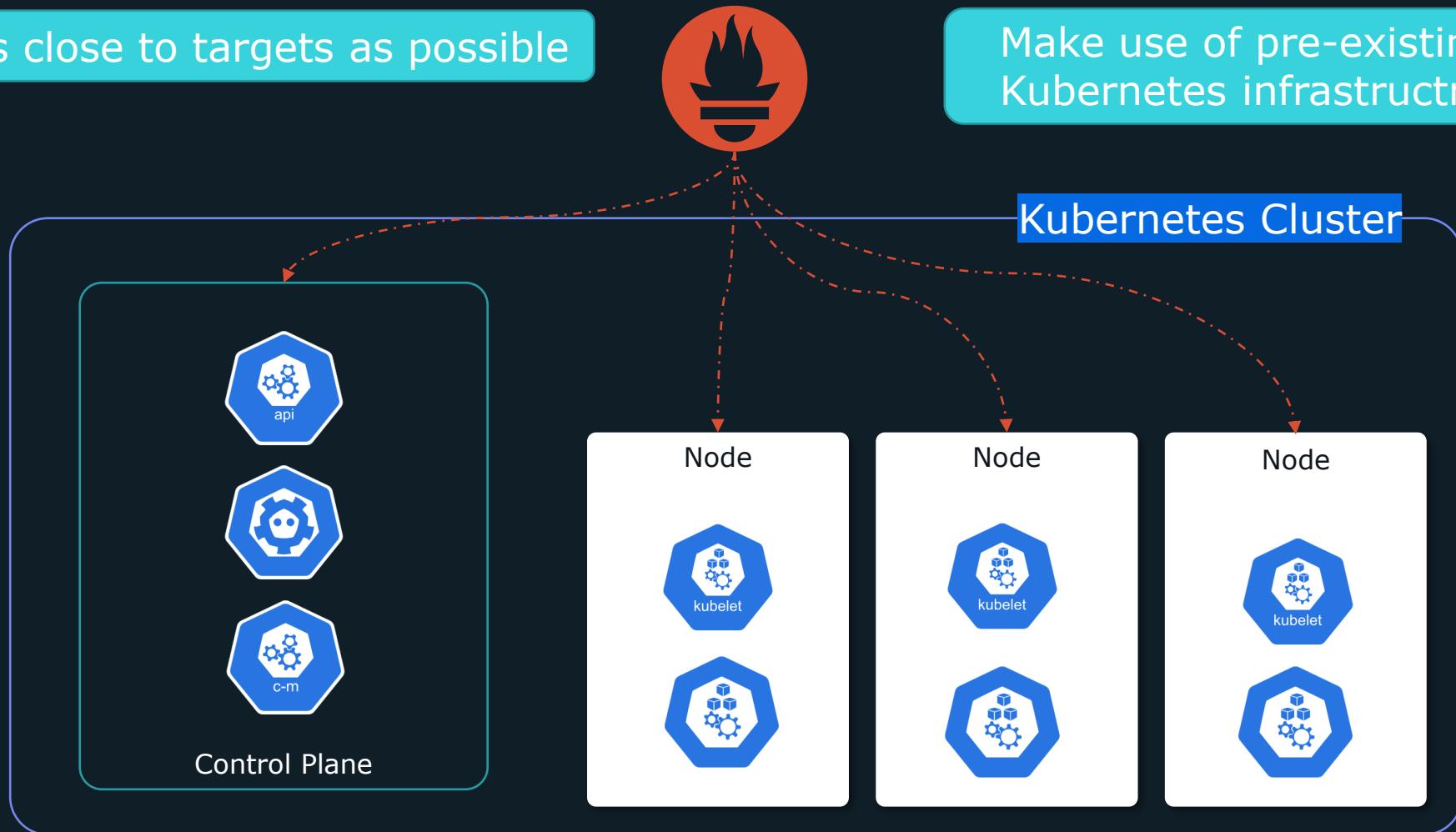
KodeKloud

Kubernetes

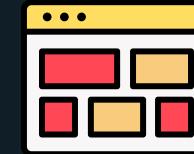
Kubernetes

Deploy as close to targets as possible

Make use of pre-existing
Kubernetes infrastructrue



- Monitor **applications** running on Kubernetes infrastructure
- Monitor Kubernetes Cluster
 - Control-Plane Components(api-server, coredns, kube-scheduler)
 - Kubelet(cAdvisor) – exposing container metrics
 - Kube-state-metrics – cluster level metrics(deployments, pod metrics)
 - Node-exporter – Run on all nodes for host related metrics(cpu, mem, network)



Kube-state-metrics

To collect cluster level metrics(pods, deployments, etc) the kube-state-metrics container must be deployed

Kube-state-metrics

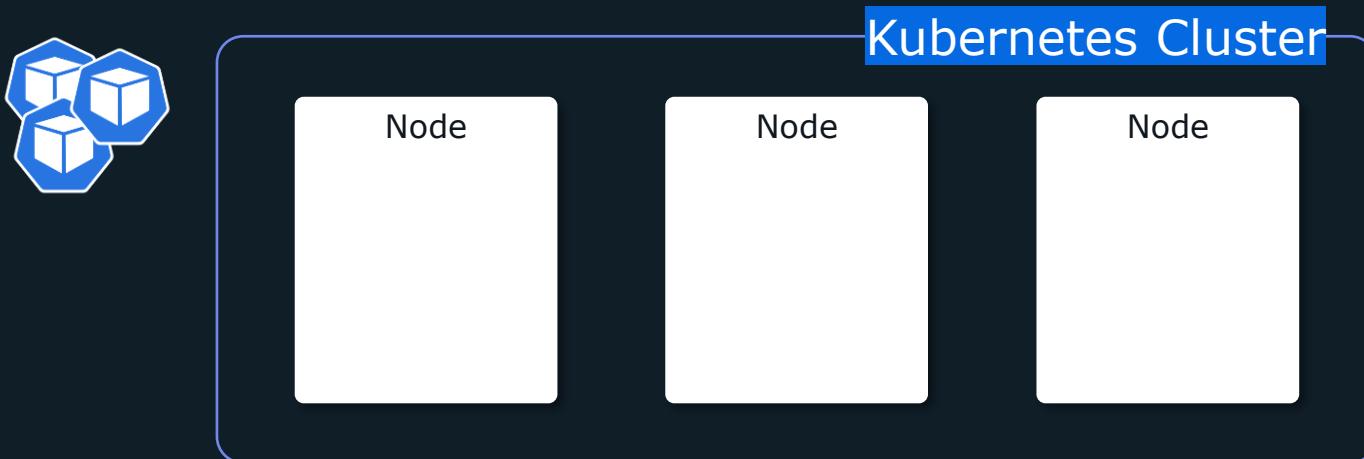
Kubernetes Cluster

Node Exporter

Every host should run a node_exporter to expose cpu, memory, and network stats

We can manually go in and install a node_exporter on every node

Better option is to use a Kubernetes daemonSet - pod that runs on every node in the cluster



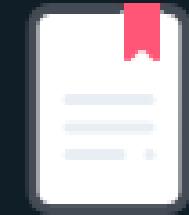
Service Discovery



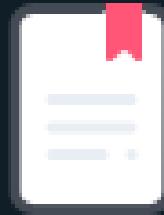
How to deploy

Manually deploy Prometheus on Kubernetes – Create all the deployments, services, configMaps, and secrets

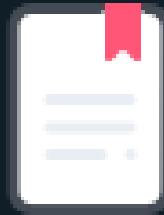
Complex, requires a lot of configuration, not the easiest solution



Deployments



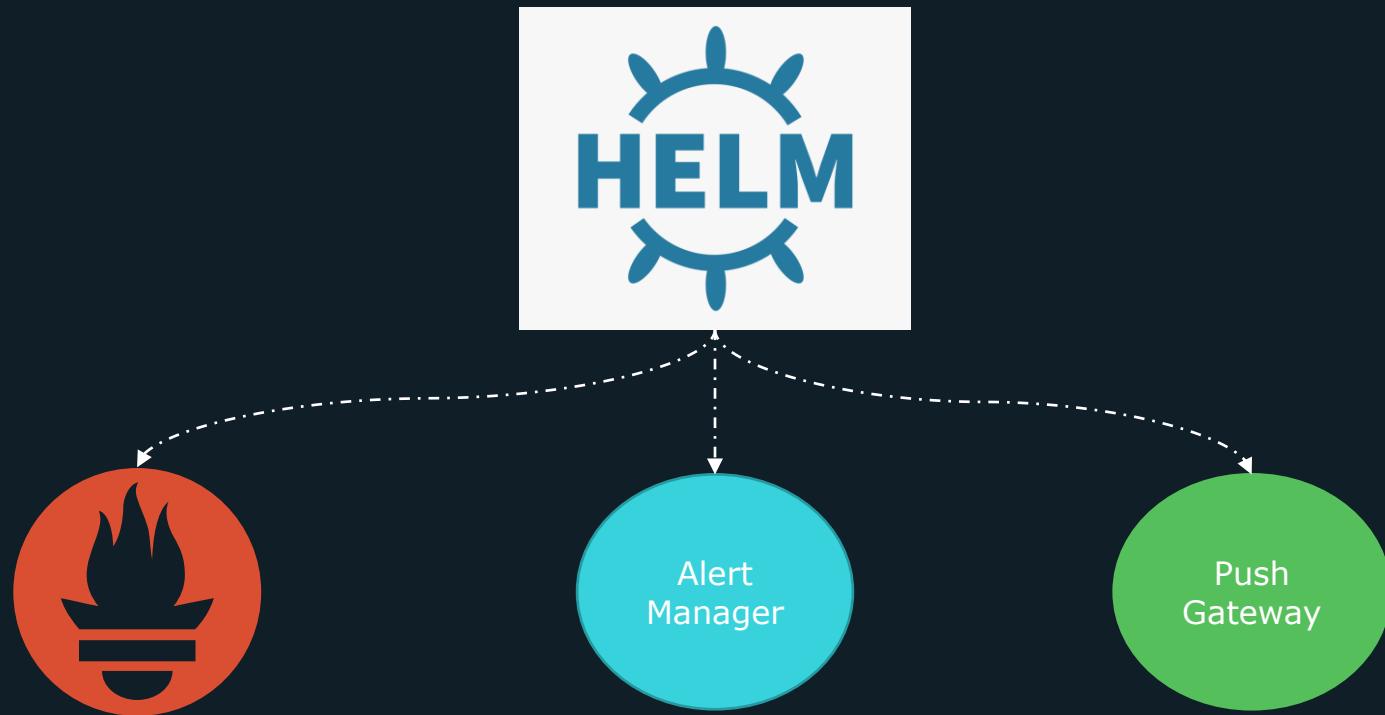
Services



Secrets

How to deploy

Best way to deploy Prometheus is using Helm chart to deploy Prometheus operator



What is Helm

- Helm is a package manager for Kubernetes
- All application and Kubernetes configs necessary for an application can be bundled into a package and easily deployed

```
$ helm install
```



A **helm chart** is a collection of template & YAML files that convert into Kubernetes manifest files

Helm charts can be **shared** with others by uploading a chart to a repository

<https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack>

Repository: Prometheus-community
Chart: kube-Prometheus-stack

The **Kube-Prometheus-stack** chart makes use of the Prometheus Operator

A Kubernetes **operator** is an application-specific controller that extends the K8s API to create/configure/manage instances of complex applications(like Prometheus!)

<https://github.com/prometheus-operator/prometheus-operator>

Prometheus Operator

The Prometheus operator has several **custom resources** to aid the deployment and management of a Prometheus instance

Alertmanager

Prometheus
Rule

Alertmanager
Config

ServiceMonitor

PodMonitor

>_

`prometheus.yaml`

```
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  annotations:
    meta.helm.sh/release-name: prometheus
    meta.helm.sh/release-namespace: default
  creationTimestamp: "2022-11-18T01:19:29Z"
  generation: 1
  labels:
    app: kube-prometheus-stack-prometheus
    name: prometheus-kube-prometheus-prometheus
spec:
  alerting:
    alertmanagers:
    - apiVersion: v2
      name: prometheus-kube-prometheus-alertmanager
      namespace: default
      pathPrefix: /
      port: http-web
```



KodeKloud

Installing Helm Chart

Install Helm

>_

<https://helm.sh/docs/intro/install/>

```
$ curl -fsSL -o get_helm.sh  
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
```

```
$ chmod 700 get_helm.sh
```

```
$ ./get_helm.sh
```

Install Prometheus Chart

>_

```
$ helm repo add Prometheus-community https://prometheus-community.github.io/helm-charts
```

```
$ helm repo update
```

```
$ helm install prometheus prometheus-community/kube-prometheus-stack
```

Release Name – can be anything you want



KodeKloud

Service Monitors

The prometheus operator comes with several **custom resource definitions** that provide a high level abstraction for deploying prometheus as well as configuring it

CRDs

>_

```
$ kubectl get crd
```

NAME	CREATED AT
alertmanagerconfigs.monitoring.coreos.com	2022-11-18T01:18:55Z
alertmanagers.monitoring.coreos.com	2022-11-18T01:18:55Z
eniconfigs.crd.k8s.amazonaws.com	2022-11-18T01:04:11Z
podmonitors.monitoring.coreos.com	2022-11-18T01:18:56Z
probes.monitoring.coreos.com	2022-11-18T01:18:56Z
prometheuses.monitoring.coreos.com	2022-11-18T01:18:56Z
prometheusrules.monitoring.coreos.com	2022-11-18T01:18:56Z
securitygrouppolicies.vpcresources.k8s.aws	2022-11-18T01:04:14Z
servicemonitors.monitoring.coreos.com	2022-11-18T01:18:57Z
thanosrulers.monitoring.coreos.com	2022-11-18T01:18:57Z

Used to create a prometheus instance

Add additional targets for prometheus to scrape

Service Monitors

Service monitors define a set of targets for prometheus to monitor and scrape

They allow you to avoid touching prometheus configs directly and give you a declarative Kubernetes syntax to define targets

Service Monitors

>_ service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: api-service
  labels:
    job: node-api
    app: service-api
spec:
  type: ClusterIP
  selector:
    app: api
  ports:
    - name: web
      protocol: TCP
      port: 3000
      targetPort: 3000
```

>_ Service-monitor.yml

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: api-service-monitor
  labels:
    release: prometheus
    app: prometheus
spec:
  jobLabel: job
  endpoints:
    - interval: 30s
      port: web
      path: /swagger-stats/metrics
  selector:
    matchLabels:
      app: service-api
```

Release Label

>_

```
$ kubectl get prometheuses.monitoring.coreos.com -o yaml
```

```
serviceMonitorNamespaceSelector: {}
  serviceMonitorSelector:
    matchLabels:
      release: prometheus
```

This label allows Prometheus to find **service monitors** in the cluster and register them so that it can start scraping the application the service monitor is pointing to

Release Label

```
>_ Service-monitor.yml
```

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: api-service-monitor
  labels:
    release: prometheus
    app: prometheus
spec:
  jobLabel: job
  endpoints:
    - interval: 30s
      port: web
      path: /swagger-stats/metrics
  selector:
    matchLabels:
      app: service-api
```



KodeKloud

Prometheusrules

Rules

To add rules, the prometheus operator has a CRD called **prometheusrules** which handles registering new rules to a prometheus instance

>_ Prometheus-rule.yml

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  labels:
    release: prometheus
    name: api-rules
spec:
  groups:
    - name: api
      rules:
        - alert: down
          expr: up == 0
          for: 0m
          labels:
            severity: critical
          annotations:
            summary: Prometheus
          target missing {{$labels.instance}}
```

Release Label

>_

```
$ kubectl get prometheuses.monitoring.coreos.com -o yaml
```

```
ruleSelector:  
  matchLabels:  
    release: prometheus
```

This label allows prometheus to find **prometheusRules** in the cluster and register them



KodeKloud

Alertmanager Rules

Rules

To add Alertmanager rules, the prometheus operator has a CRD called **AlertmanagerConfig** which handles registering new rules to Alertmanager

```
>_      Alertmanager-rule.yml
apiVersion: monitoring.coreos.com/v1alpha1
kind: AlertmanagerConfig
metadata:
  name: alert-config
  labels:
    resource: prometheus
spec:
  route:
    groupBy: ["severity"]
    groupWait: 30s
    groupInterval: 5m
    repeatInterval: 12h
    receiver: "webhook"
  receivers:
    - name: "webhook"
      webhookConfigs:
        - url: "http://example.com/"
```

Release Label

>_

```
$ kubectl get alertmanagers.monitoring.coreos.com -o yaml
```

```
spec:
```

```
  alertmanagerConfigNamespaceSelector: {}  
  alertmanagerConfigSelector: {}
```

This label allows Alertmanager to find
AlertmanagerConfig objects in the cluster
and register them

The helm chart by default does not specify a label, so we will have to go into the chart and update this value

Config Differences

>_ alertmanager.yml

```
route:  
  receiver: staff  
  group_by: ['severity']  
  group_wait: 30s  
  group_interval: 5m  
  repeat_interval: 12h  
routes:  
  - matchers:  
    - job: kubernetes  
    receiver: infra  
    group_by: ['severity']
```

snake_case

>_ AlertmanagerConfig.yml

```
spec:  
  route:  
    groupBy: ["alertname"]  
    groupWait: 30s  
    groupInterval: 5m  
    repeatInterval: 12h  
    receiver: "webhook"  
  routes:  
    - matchers:  
      - name: job  
        value: kubernetes  
      receiver: "infra"  
      groupBy: ["severity"]
```

camelCase



KodeKloud