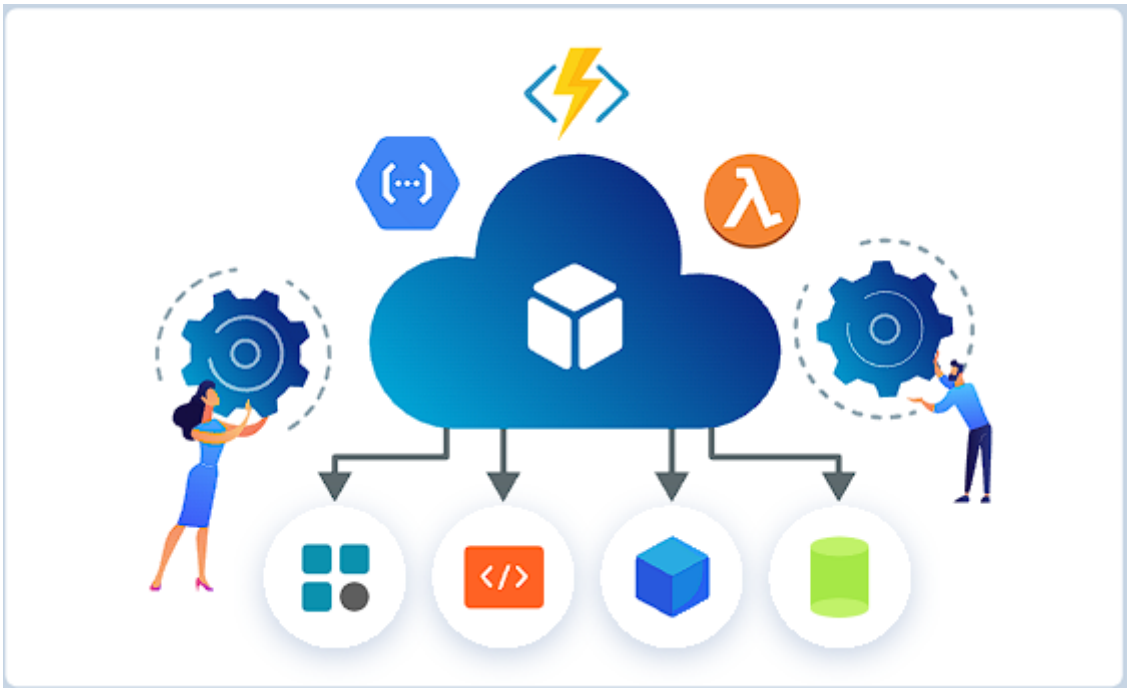
 Light



Kubernetes Observability

Project for learning about Kubernetes observability.
Explore the docs »

View Demo - Report Bug - Request Feature

▶ Table of Contents

About The Project

This project is for learning about kubernetes observability.

(back to top)

Built With

Lig

- 
- 
- 
- 
- 
- 

(back to top)

Getting Started

This project is for getting started with kubernetes observability tools and best practices.

Some tools for learning:

- Prometheus
- Alertmanager
- Grafana
- Grafana Loki
- Grafana Tempo
- Grafana Alloy

Prerequisites

- Linux System Up
- Cluster Kubernetes Up
- Git

Installation

Clone the repo

```
git clone https://github.com/marcossilvestrini/kubernetes-observability.git
cd kubernetes-observability || exit
```

(back to top)

 Light

Usage

I publish some exemples for use in this repository.

(back to top)

Roadmap

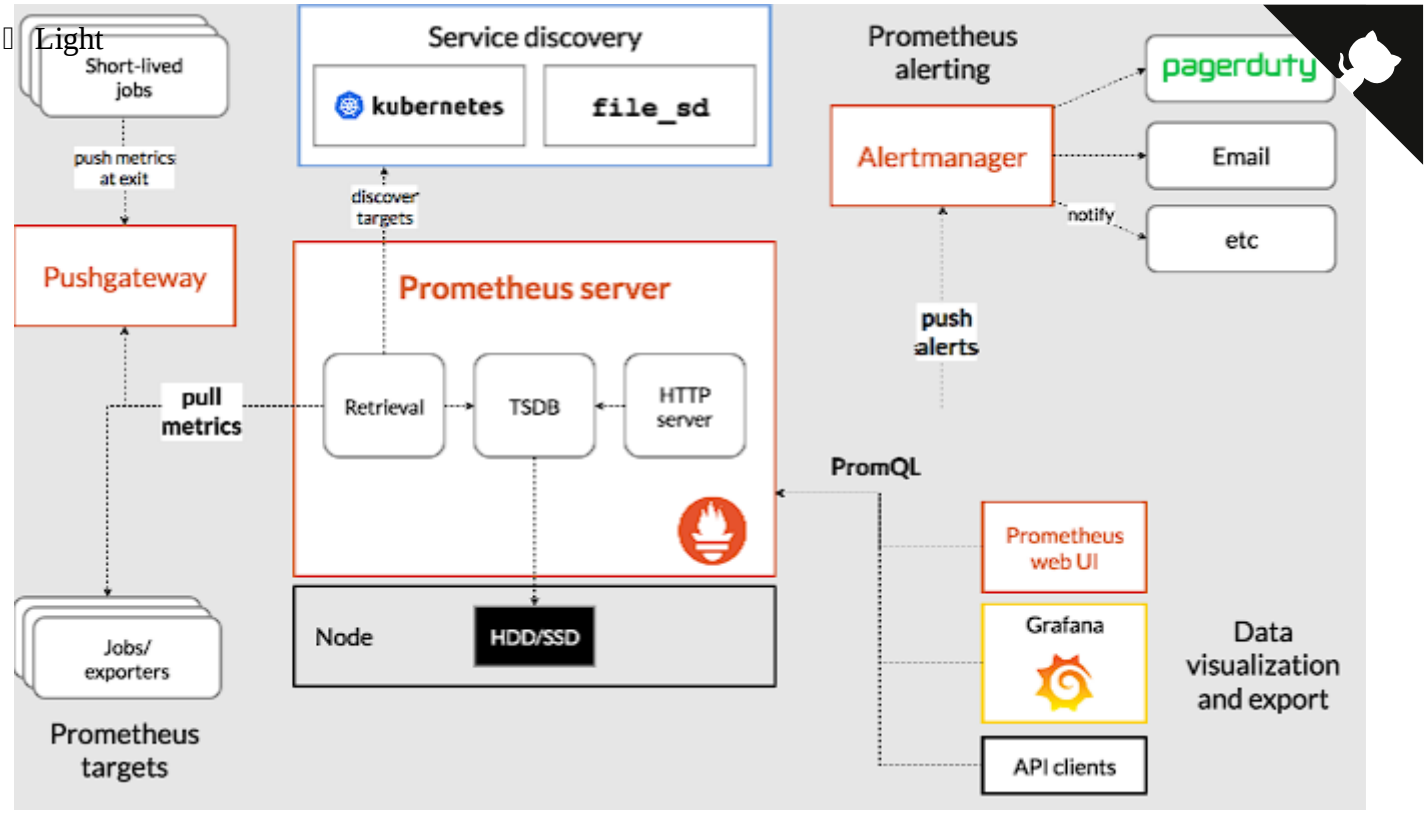
- [x] Create repositoty
- [ ] Prometheus
- [ ] Alertmanager
- [ ] Grafana
- [ ] Grafana Loki
- [ ] Grafana Tempo
- [ ] Grafana Alloy
- [ ] Others tools

See the open issues for a full list of proposed features (and known issues).

(back to top)

Prometheus

Light



Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud.

Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community.

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

- the main Prometheus server which scrapes and stores time series data
- client libraries for instrumenting application code
- a push gateway for supporting short-lived jobs
- special-purpose exporters for services like HAProxy, StatsD, Graphite, etc.
- an alertmanager to handle alerts
- various support tools

Metric names and labels

Example metric name:

```
<metric name>{<label name>=<label value>, ...}
```

Example metric name with labels:

```
api_http_requests_total{method="POST", handler="/messages"}
```

Light

For more information about Prometheus access official documentation:
https://prometheus.io/docs/introduction/overview/

Install Prometheus

```
# Download files - https://prometheus.io/download/
wget https://github.com/prometheus/prometheus/releases/download/v2.51.2/prometheus-2

# Extract files
tar xvfz prometheus-*.tar.gz
rm  prometheus-*.tar.gz
cd prometheus-*

# Check version
./prometheus --version
```

Configure Prometheus

See my configuration file prometheus.yaml

```
vim prometheus.yaml
```

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is eve
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
```

```
static_configs:
  - targets: ["localhost:9090"]
```

Start Prometheus

```
# Start
./prometheus --config.file=prometheus.yml

# Start with PM2 - npm install pm2@latest -g
pm2 start prometheus --name prometheus-server -- --config.file=prometheus.yml
```

Important Endpoints

```
http://localhost:9090 # all endpoints
http://localhost:9090/graph # PromQL expressions
http://localhost:9090/metrics # metrics
http://localhost:9090/targets # scrape_configs jobs
```

Using the expression browser

You can use the expression in Table or Graph mode.

Open the page http://localhost:9090

```
# Check all http metrics
promhttp_metric_handler_requests_total

# Check http metrics with http status code 200
promhttp_metric_handler_requests_total{code="200"}

# Count http metrics
count(promhttp_metric_handler_requests_total)

# Rate function
rate(promhttp_metric_handler_requests_total{code="200"}[1m])
```

Prometheus Exporters

An exporter is a binary running alongside the application you want to obtain metrics from.
The exporter exposes Prometheus metrics, commonly by converting metrics that are exposed in a non-Prometheus format into a format that Prometheus supports.

Node Exporter

The Prometheus Node Exporter exposes a wide variety of hardware- and kernel-related metrics.

Instal Node Exporter

```
# Download - https://prometheus.io/download#node_exporter
wget https://github.com/prometheus/node_exporter/releases/download/v1.7.0/node_expor

# Extract
tar xvfz node_exporter-*.*-amd64.tar.gz
cd node_exporter-*.*-amd64
```

Start Node Exporter

```
# Start
./node_exporter

# Start with PM2 - npm install pm2@latest -g
pm2 start node_exporter --name node_exporter
```

Endpoints Node Exporter

```
# Access metrics
http://localhost:9100/metrics
```

Configure Node Exporter

For enable scrap for node exporter, you can configure prometheus.

```
# Edit prometheus file and add job node
vim prometheus.yaml
```

```
...
scrape_configs:
- job_name: node
  static_configs:
  - targets: ['localhost:9100']
...
```

Restart prometheus service for apply new job.

PushGateway

The Prometheus Pushgateway is an intermediary service which allows ephemeral and batch jobs to expose their metrics to Prometheus.
Since these kinds of jobs may not exist long enough to be scraped, they can instead push their

metrics to a Pushgateway.
The Pushgateway then acts as a temporary metrics store which Prometheus scrapes.

This setup is particularly useful for capturing the outcome of a job that does not run continuously, such as a batch job in a CI system, or a backup script running at a scheduled time.
It simplifies monitoring these kinds of jobs without needing to run a long-lived Prometheus instance that might outlive the jobs themselves.

Install PushGateway

```
# Download
wget -q https://github.com/prometheus/pushgateway/releases/download/v1.8.0/pushgatew

# Extract
tar xvfz pushgateway-*.*-amd64.tar.gz
cd pushgateway-*.*-amd64

# Start
# Start with PM2 - npm install pm2@latest -g
pm2 start pushgateway --name pushgateway -- --web.listen-address "192.168.0.130:9091
```

Configure PushGateway

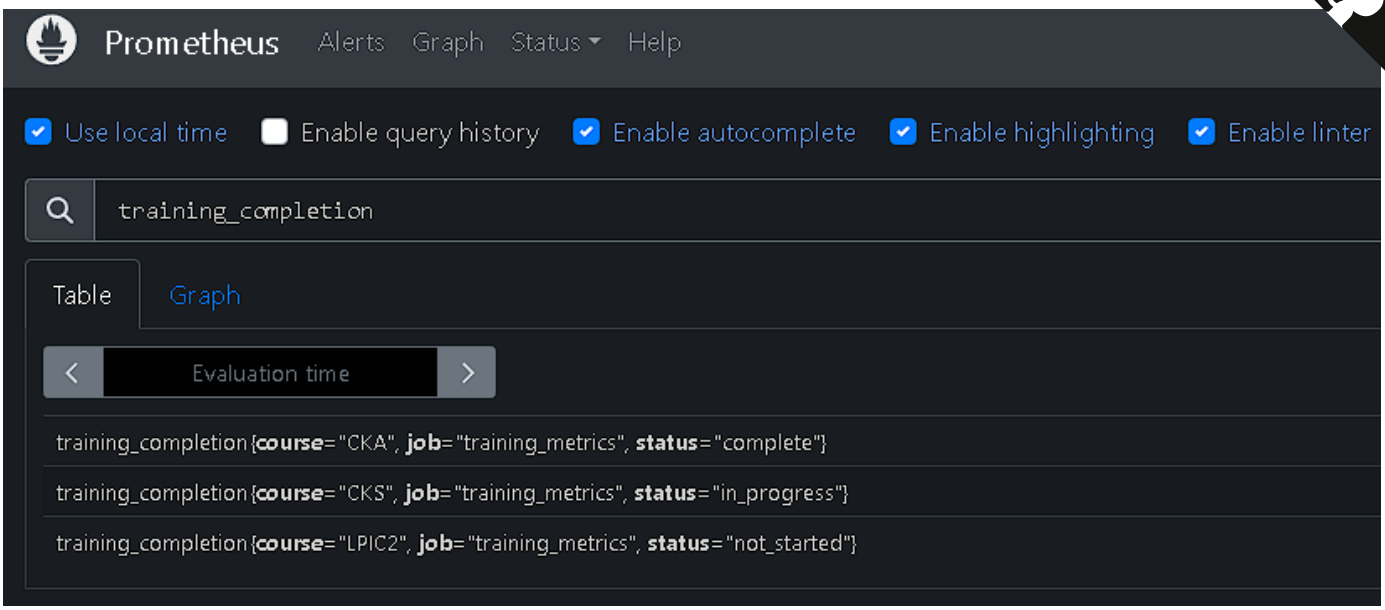
```
# Edit prometheus file and add job pushgateway
vim prometheus.yaml



scrape_configs:
  - job_name: 'pushgateway'
    honor_labels: true
    static_configs:
      - targets: ['192.168.0.130:9091'] # prometheus server for scraping



# Restart prometheus

# restart with pm2
pm2 restart prometheus-server
```

Create metrics for test pushgateway

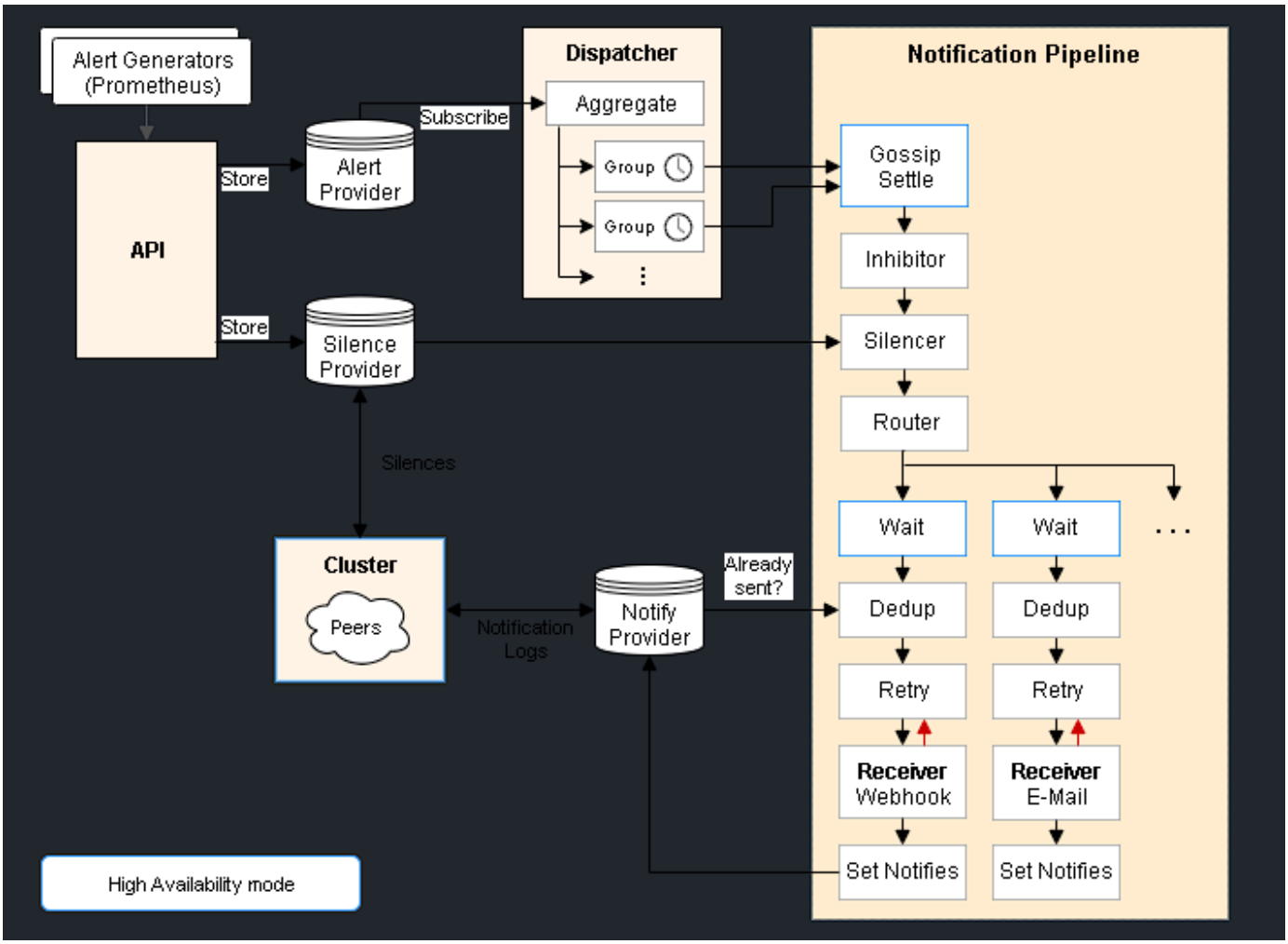
```
echo 'training_completion{course="CKA", status="complete"} 1' > metrics.txt
echo 'training_completion{course="CKS", status="in_progress"} 0.5' >> metrics.txt
echo 'training_completion{course="LPIC2", status="not_started"} 0' >> metrics.txt
curl --data-binary @metrics.txt http://192.168.0.130:9091/metrics/job/training_metri
```

Use PromQL for find metrics pushgateway target

Light



Alertmanager



For more information about Alertmanager access official documentation:
https://github.com/prometheus/alertmanager

(back to top)

 Light

Grafana

Grafana Loki

Grafana Tempo

Grafana Alloy

(back to top)

Contributing

Contributions are what make the open source community such an amazing place to learn, inspire, and create. Any contributions you make are **greatly appreciated**.

If you have a suggestion that would make this better, please fork the repo and create a pull request. You can also simply open an issue with the tag "enhancement". Don't forget to give the project a star! Thanks again!

1. Fork the Project
2. Create your Feature Branch (`git checkout -b feature/AmazingFeature`)
3. Commit your Changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the Branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

(back to top)

License

Distributed under the MIT License. See `LICENSE` for more information.

(back to top)

Contact

Light

- Marcos Silvestrini - @mrsilvestrini
- marcos.silvestrini@gmail.com

Project Link: https://github.com/marcossilvestrini/kubernetes-observability

(back to top)

Acknowledgments

- Prometheus
- Node Exporter
- Prometheus Default port allocations
- Pushgateway
- Kube Prometheus Stack Article

(back to top)