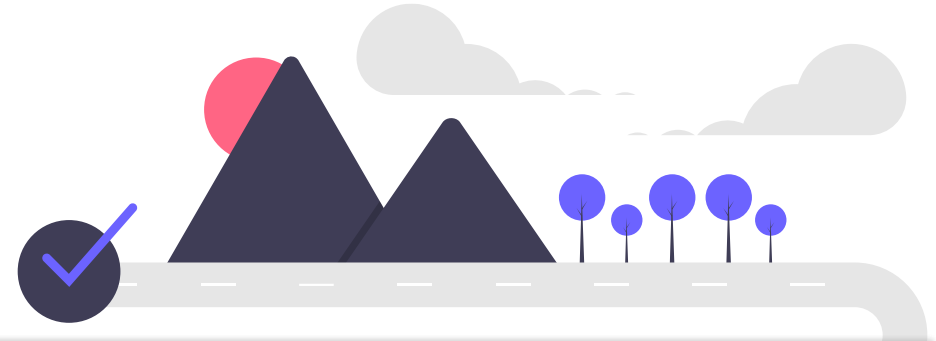




Observability & Monitoring



PROMETHEUS & GRAFANA

Marius N'KOUBA
Data Engineer

May 2024

Plan: Prometheus



Définitions et notions

Installation de Prometheus

Jobs & Instances

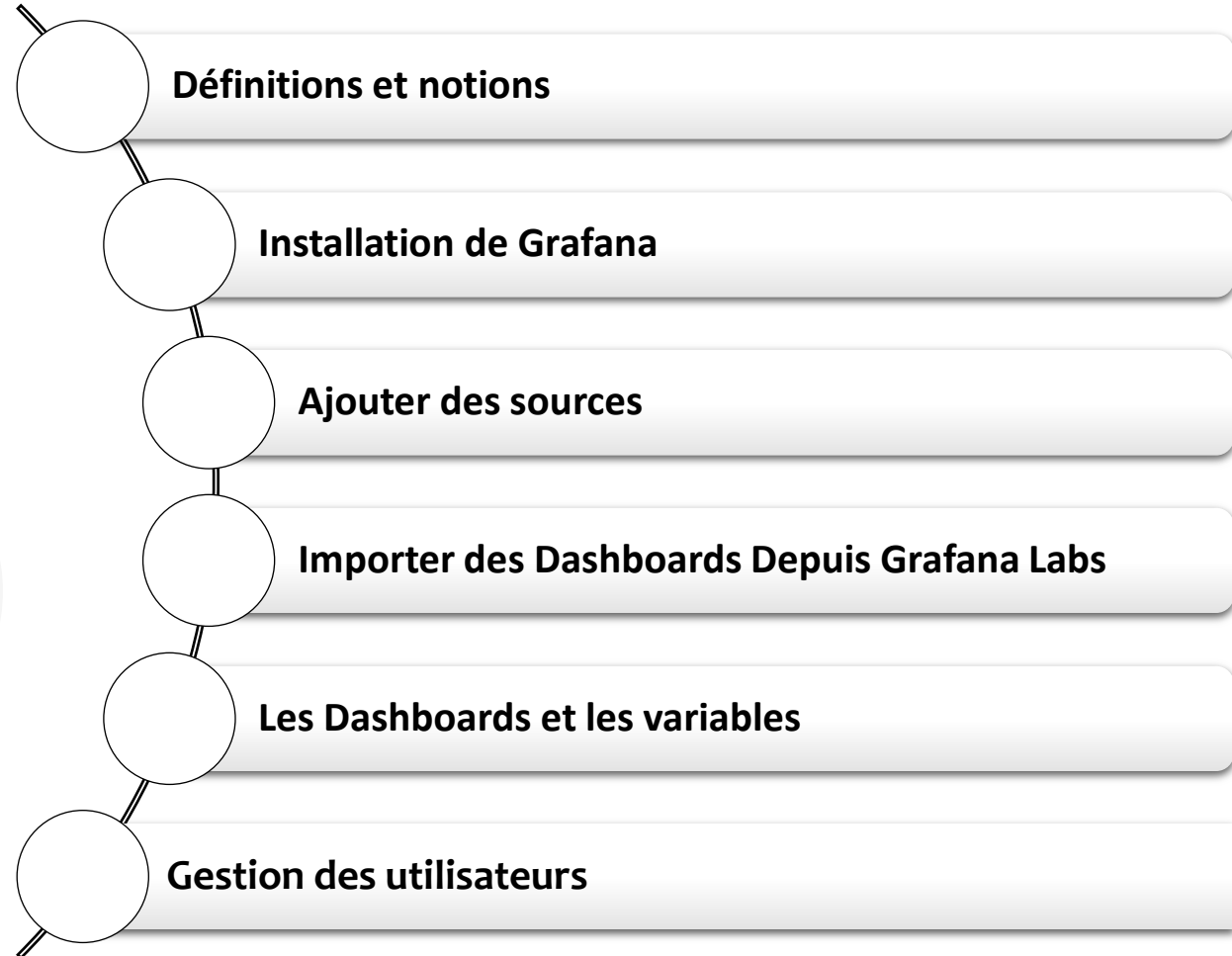
Interface Graphique

Node , PostgreSQL Exporter, Mysql Exporter

PromQL

AlertManager

Plan : Grafana



Monitoring



Prometheus



Définition

Qu'est-ce que Prometheus ? 🤔

Prometheus, est un système de surveillance des systèmes et des services.

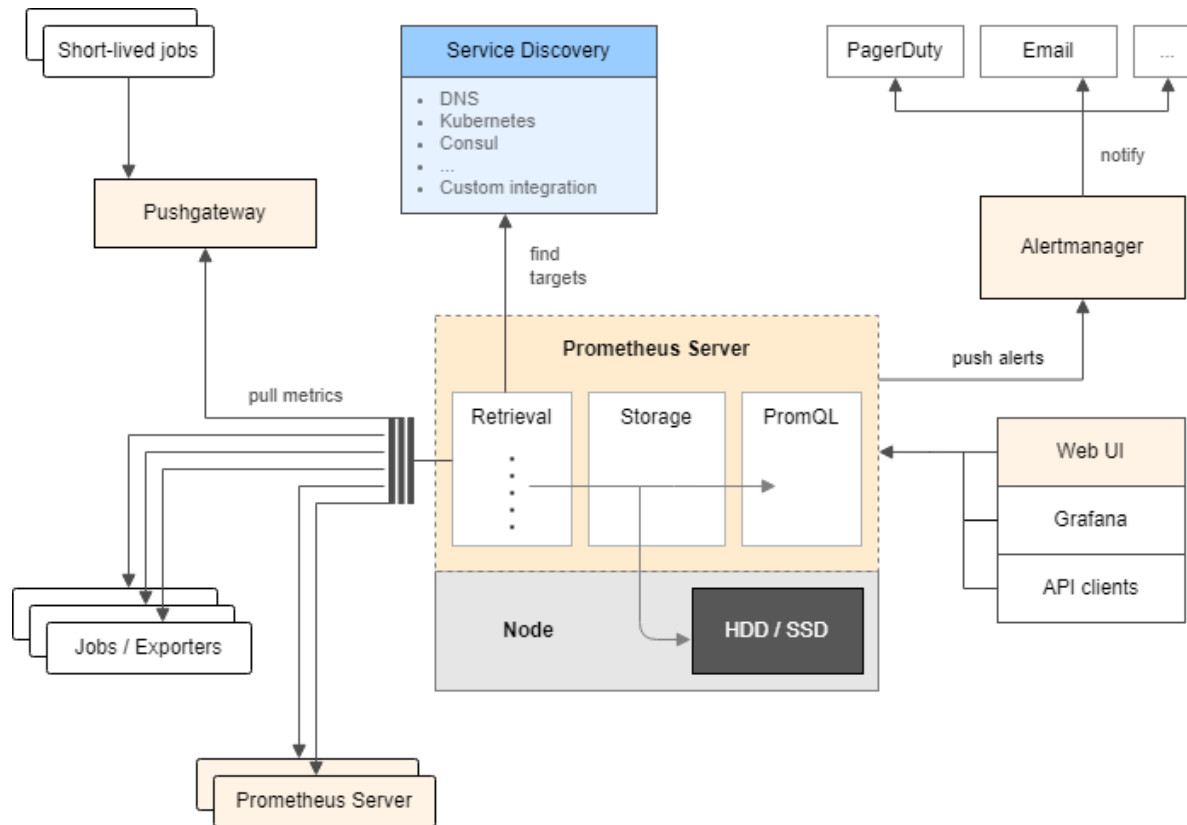
Il collecte des métriques à partir de cibles configurées à des intervalles donnés, évalue les expressions de règles, affiche les résultats et peut déclencher des alertes lorsque des conditions spécifiées sont observées.

- Un modèle de données multidimensionnel (série chronologique définie par le nom de la métrique et un ensemble de dimensions clé/valeur)
- PromQL, un langage de requête puissant et flexible
- Aucune dépendance au stockage distribué
- Un modèle HTTP pull pour la collecte de séries chronologiques
- Les cibles sont découvertes via la découverte de services ou la configuration statique
- Plusieurs modes de prise en charge des graphiques et des tableaux de bord
- Prometheus se base sur le Langage Go
- base de données Base Time series + Serveur Web + Moteur de Base de Données



Définition

Qu'est-ce que Prometheus ? 🤔

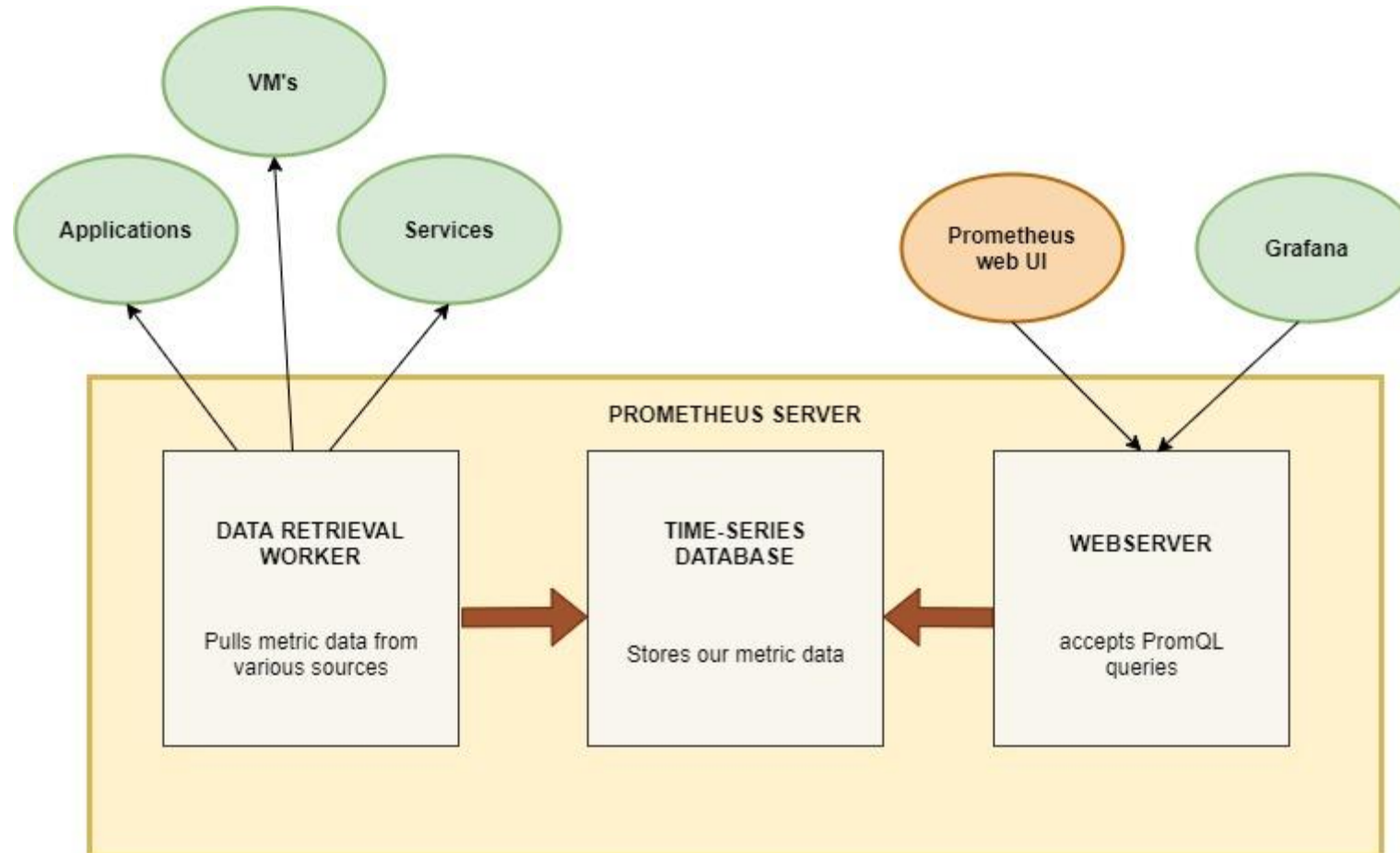


- Il fait essentiellement du Scraping de données à des fréquences régulières
- Principe de stockage : Clé / Valeur / Timestamp
- Il travaille en double Delta:

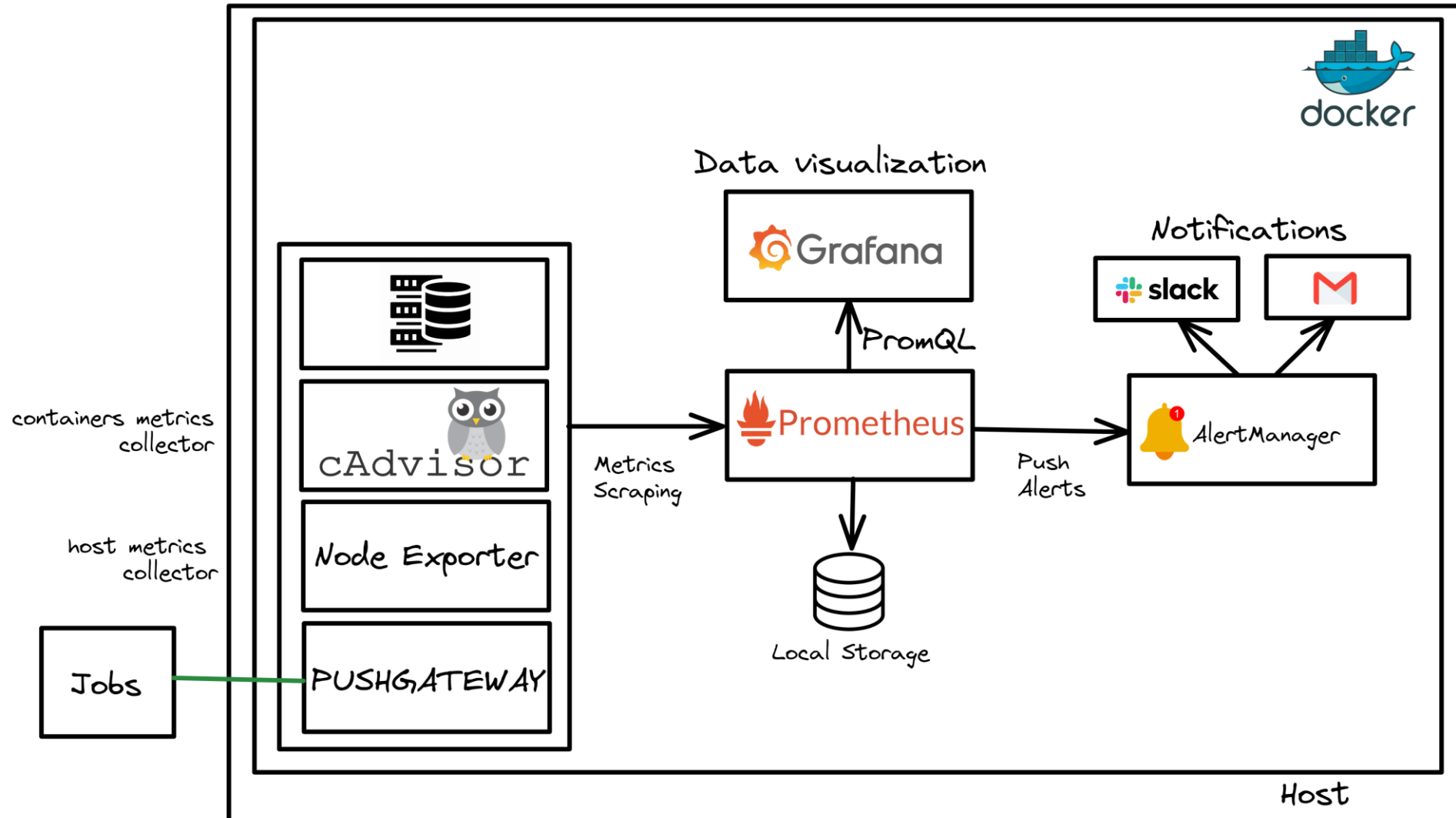
Double delta:

Calcul l'écart entre une valeur et sa valeur précédente, Si ça n'évolue pas : il ne fait pas de modifications sinon

Monitoring using prometheus



Monitoring avec Docker



Types de métriques

Counter (compteur)

Un compteur est une valeur qui augmente de manière monotone. Il est souvent utilisé pour suivre des événements qui se produisent de manière incrémentielle, tels que le nombre de requêtes HTTP reçues ou le nombre de fois qu'une certaine opération a été effectuée.

Gauge (jauge)

Une jauge est une valeur qui peut augmenter ou diminuer de manière arbitraire au fil du temps. Elle est souvent utilisée pour représenter des mesures instantanées, telles que la taille actuelle d'une file d'attente, la quantité de mémoire utilisée ou le nombre d'utilisateurs actuellement connectés.

Types de métriques

Histogram (histogramme)

Un histogramme est utilisé pour suivre la distribution des valeurs d'une variable au fil du temps. Il divise les valeurs en intervalles (appelés "buckets") et compte le nombre d'occurrences dans chaque intervalle.

Cela permet d'analyser la distribution des valeurs, par exemple, le temps passé dans différentes phases d'exécution d'une requête

Summary (résumé)

Un résumé est similaire à un histogramme, mais plutôt que de compter les occurrences dans des intervalles prédéfinis, il calcule des quantiles (par exemple, le 50e, le 90e et le 99e percentile) des valeurs observées.

Cela permet de mieux comprendre la distribution des valeurs et d'identifier les tendances de performance

Installation de Prometheus

Installation paquet Debian

```
sudo apt update  
sudo apt-get install prometheus
```

Configuration de prometheus.yml

```
sudo nano /etc/prometheus/prometheus.yml
```

Prometheus

<http://localhost:9090>

Prometheus Metrics

<http://localhost:9090/metrics>

Node Metrics

<http://localhost:9100/metrics>

Targets

All Unhealthy Collapse All

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

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node"	4.678s ago	44.22ms	

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

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	4.465s ago	8.747ms	

Configuration prometheus.yml

- Configuration
 - Global : Configurations qui s'appliquent en général
 - scrape_interval (ou job): Intervale de récupération des données
 - evelaution_interval : réévaluation des règles (alertes)
 - scrape_timeout: timeout lors du scraping
 - rule_files: configuration des alertes
 - scrape_configs : configuration particulière du scraping
 - job_name : nom du bloc (http://localhost:9090/classic/service-discovery)
 - metric_path: route de scraping (/metrics)
 - static_config :
 - labels: Definition de labels (important pour grafana/ Standardiser)
 - targets : url/ip:port

Interface graphique

Prometheus Alerts Graph Status ▾ Help

☐ Enable query history

Expression (press Shift+Enter for newlines) 1

Execute - insert metric at cursor - ▾ 2

[Remove Graph](#)

Graph Console

◀ 2024-02-20 22:36:20 ▶ 3

Element	Value
no data	4

[Add Graph](#)

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

Installation de quelques Exporters



Node exporter

Agent qui permet de collecter des métriques systèmes :

- disques
- mémoire
- cpu
- load average
- nfs
- time
- uname
- vmstat
- Stats

Nombres de métriques : 500

Exemple de métriques retournés:

https://grafana.com/oss/prometheus/exporters/node-exporter/assets/node_exporter_sample_scrape.txt

<https://grafana.com/oss/prometheus/exporters/node-exporter/>
https://github.com/prometheus/node_exporter



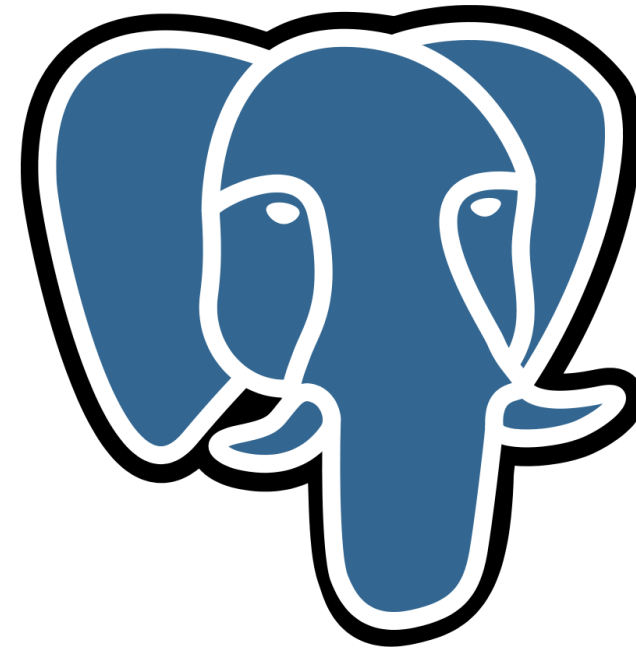
Postgres exporter

Agent qui permet de collecter des métriques d'une base de données Postgres

Nombres de métriques : 450

Exemple de métriques retournés:

https://grafana.com/oss/prometheus/exporters/postgres-exporter/assets/postgres_metrics_scrape.txt



<https://grafana.com/oss/prometheus/exporters/postgres-exporter/>
https://github.com/prometheus-community/postgres_exporter

Mysql exporter

Agent qui permet de collecter des métriques d'une base de données Mysql

Nombres de métriques : 1000

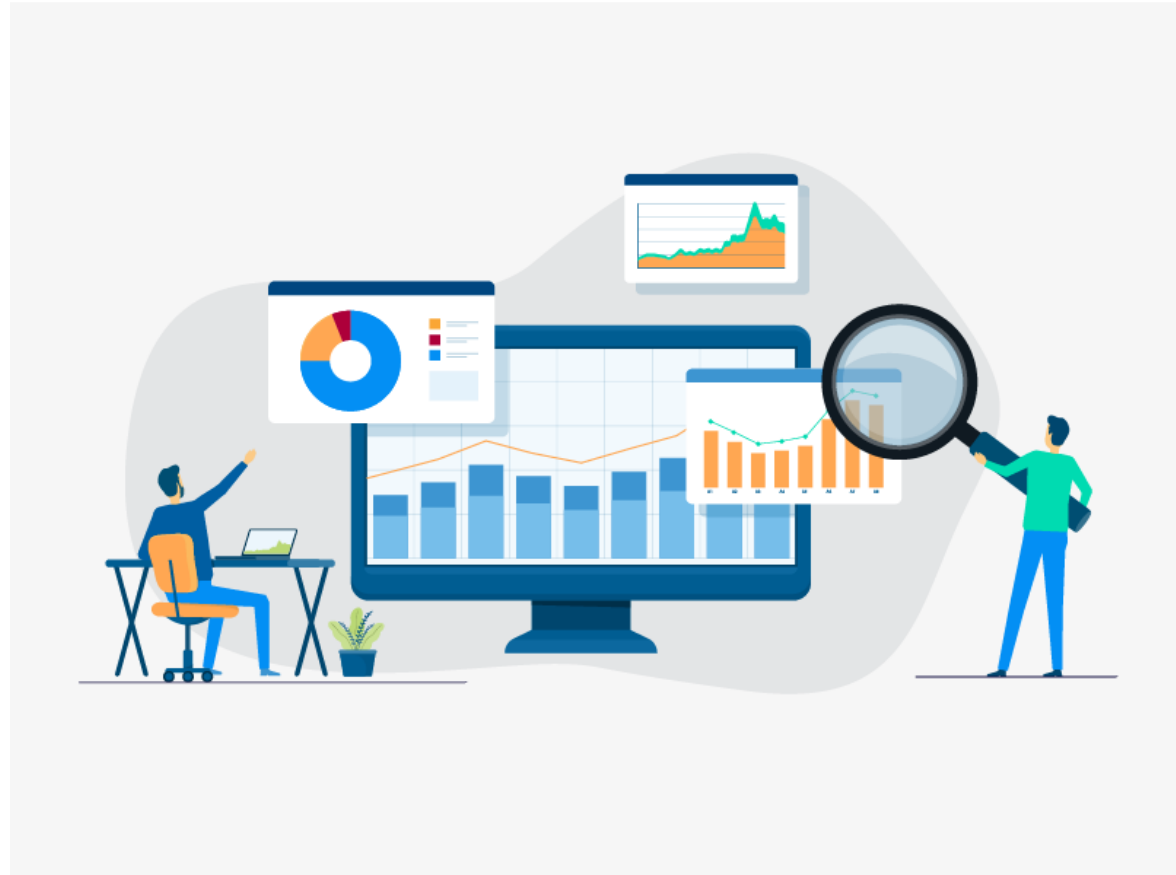
Exemple de métriques retournés:

https://grafana.com/oss/prometheus/exporters/mysql-exporter/assets/mysql_metrics_scrape.txt



https://github.com/prometheus-community/postgres_exporter

PromQL



PromQL

Filtres sur les Labels

- Equivalent à une clause WHERE
- Utilise les regex GO
- <https://prometheus.io/docs/practices/naming/>
(Documentation)

Renommer un Label

```
relabel_configs:  
  - source_labels: [__meta_consul_service]  
    target_label: job
```

WHERE

```
up{job="node"}  
up{job="node_exporter"}  
node_network_receive_bytes_total{device="ens33"}
```

DIFFERENT DE

```
node_network_receive_bytes_total{device!="ens33"}
```

REGEX

```
node_network_receive_bytes_total{device=~"ens.*"}  
node_network_receive_bytes_total{device=~"ens33|lo"}
```

MULTIPLE FILTRES

```
node_network_receive_bytes_total{instance=~"localhost:..+",  
device=~"ens33|lo"}
```

PromQL

Opération d'aggrégation

- `count(node_cpu_seconds_total)`
- `count(node_cpu_seconds_total) by(instance)`
- `count(node_cpu_seconds_total) by(instance, cpu)`
- `count(count(node_cpu_seconds_total) by(instance, cpu)) by(instance)`

Intervalles de temps

Range Vector : `[]`

- cherche tous les timeseries sur un interval de temps
- par défaut par rapport à maintenant

`node_load1[2m]`

Offset : `offset 3m`

- change la date à laquelle on recherche la timeseries

`node_load1[1m] offset 3m`

Conversion

`date -d "@1711004584.27"`

Unités de temps

s : seconds
m : minutes
h : hours
d : days
w : weeks
y : years

PromQL

Opération d'aggrégation

- `sum_over_time(node_cpu_seconds_total[3m])`
- `avg_over_time(node_cpu_seconds_total[3m])`
- `sum_over_time(node_load1[3m])`

Offset

Idéal pour comparer deux périodes

`node_load1 - node_load1 offset 30m`

`node_load1 - sum_over_time(node_load1[5m] offset 5m)`

PromQL

Manipulation de Labels

élément pour filtrer et typer des métriques

Skipper les métriques de nodes commençant par 192.168(.*)

```
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets: ['localhost:9090']
  - job_name: node_exporter
    static_configs:
      - targets: ['localhost:9100']
    relabel_configs:
      - source_labels: [__address__]
        regex: '192.168(.*?)'
        action: drop
```

Garder les métriques de nodes commençant par 192.168(.*)

```
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets: ['localhost:9090']
  - job_name: node_exporter
    static_configs:
      - targets: ['localhost:9100']
    relabel_configs:
      - source_labels: [__address__]
        regex: '192.168(.*?)'
        action: keep
```

Grafana



Grafana

Grafana est une plateforme de visualisation de données interactive Open Source développée par Grafana Labs, qui permet aux utilisateurs de consulter leurs données via des graphiques unifiés dans un ou plusieurs tableaux de bord afin de mieux les interpréter et les comprendre.

C'est un outil pour afficher données chronologiques. A partir d'une série de données collectées, nous obtiendrons un panorama graphique de la situation d'une entreprise ou organisation



Ajouter le DataSource Prometheus

The screenshot displays the Omnishore web interface for configuring a Prometheus data source. The browser address bar shows the URL `192.168.50.101:3000/datasources/edit/fa373d0c-c21b-4a66-87eb-695735a57595`. The left sidebar contains navigation links: Administration, Data sources (selected), Plugins, Users, Teams, Service accounts, Default preferences, Settings, Organizations, and Statistics and licensing. The main content area is titled 'Prometheus' and includes a 'Build a dashboard' button. Below the title, there are tabs for 'Settings' (active) and 'Dashboards'. A green status indicator shows 'Alerting supported'. The 'Name' field is set to 'Prometheus' and is marked as the 'Default' source. The 'HTTP' section contains three settings: 'Prometheus server URL' (http://192.168.50.102:9090), 'Allowed cookies' (New tag (enter key to add) with an 'Add' button), and 'Timeout' (Timeout in seconds). The 'Auth' section includes 'Basic auth' (disabled), 'TLS Client Auth' (disabled), 'Skip TLS Verify' (disabled), and 'Forward OAuth Identity' (disabled). Each of these has associated toggle switches for 'With Credentials' and 'With CA Cert'. At the bottom, there is a 'Custom HTTP Headers' section with an 'Add header' button.

Exporters Dashboard ID

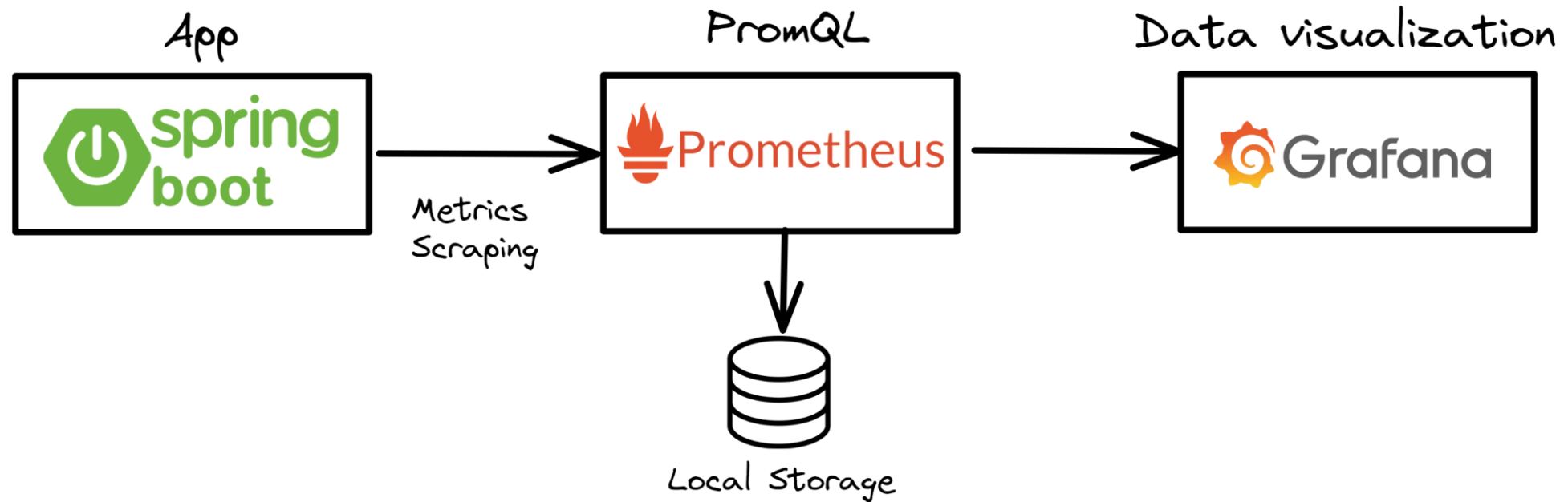
Node Exporter : **1860**

Postgres Exporter : **9628**

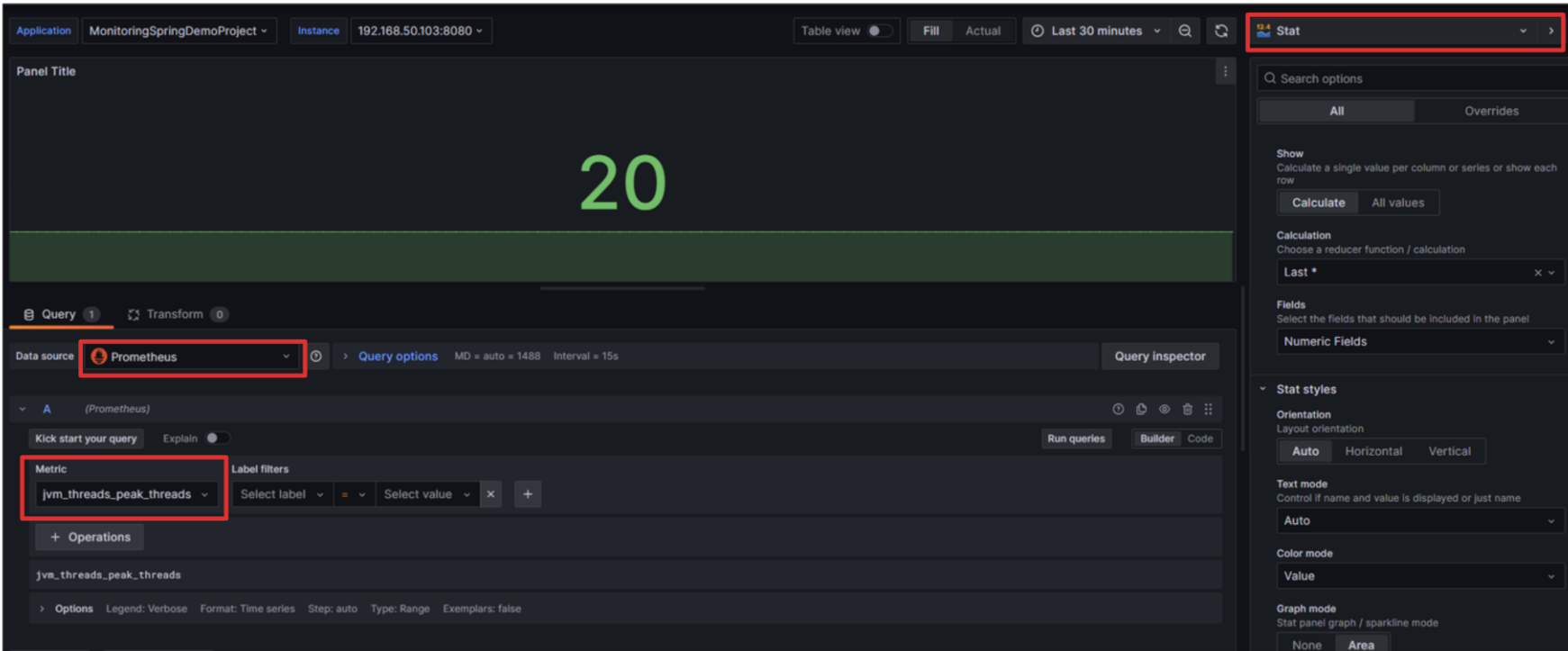
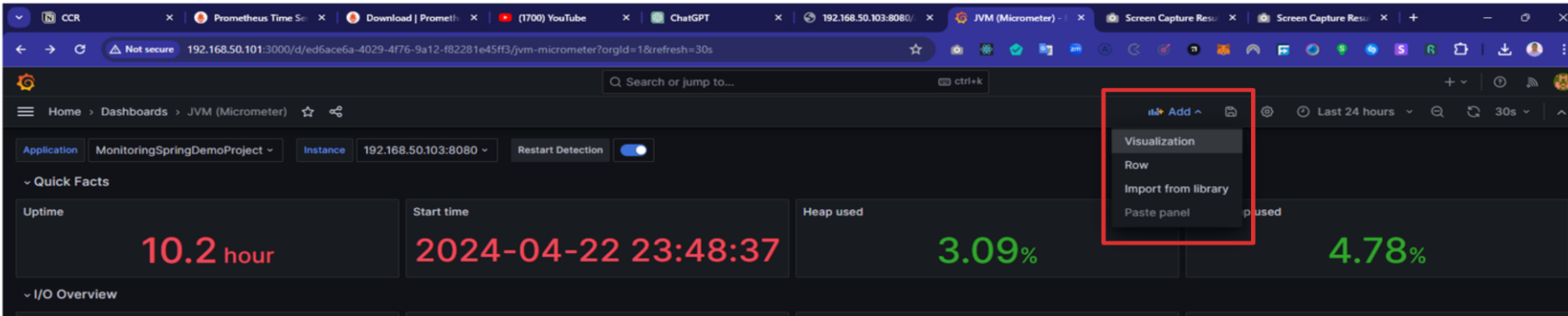
Mysql Exporter : **14057**



Monitoring of SpringBoot App



AJOUT D'UN PANEL DE MÉTRIQUE PERSONNALISÉ



Plugins dans Grafana

Dynamic Text Panel for Grafana

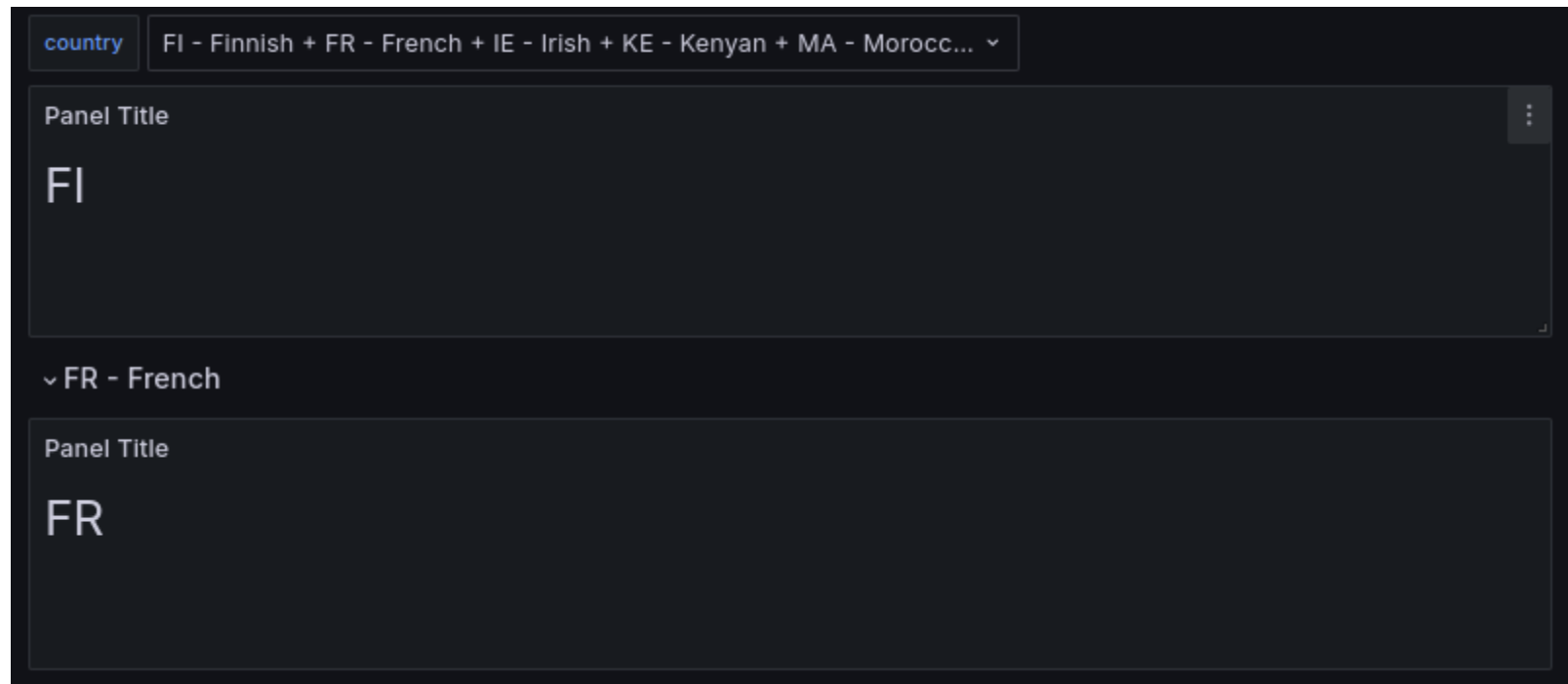
The screenshot shows the Grafana Dynamic Text panel configuration. The main panel displays a table of users with columns for name, login, role, status, and last login date. The table is styled with a blue header and alternating row colors. The configuration sidebar on the right shows the 'Dynamic Text' panel options, including the 'Render template' dropdown set to 'Every row', the 'Editor' with 'HTML' selected, and the 'Content' field containing a table structure using Grafana's templating language. The 'CSS Styles' section is also visible at the bottom of the sidebar.

ID	first_name	last_name	login	last_login_date	status	photo
1	John	Smith	j.smith	2022-11-02 22:11	active	
2	Jessica	Johnson	j.johnson	2022-10-31 14:45	not active	

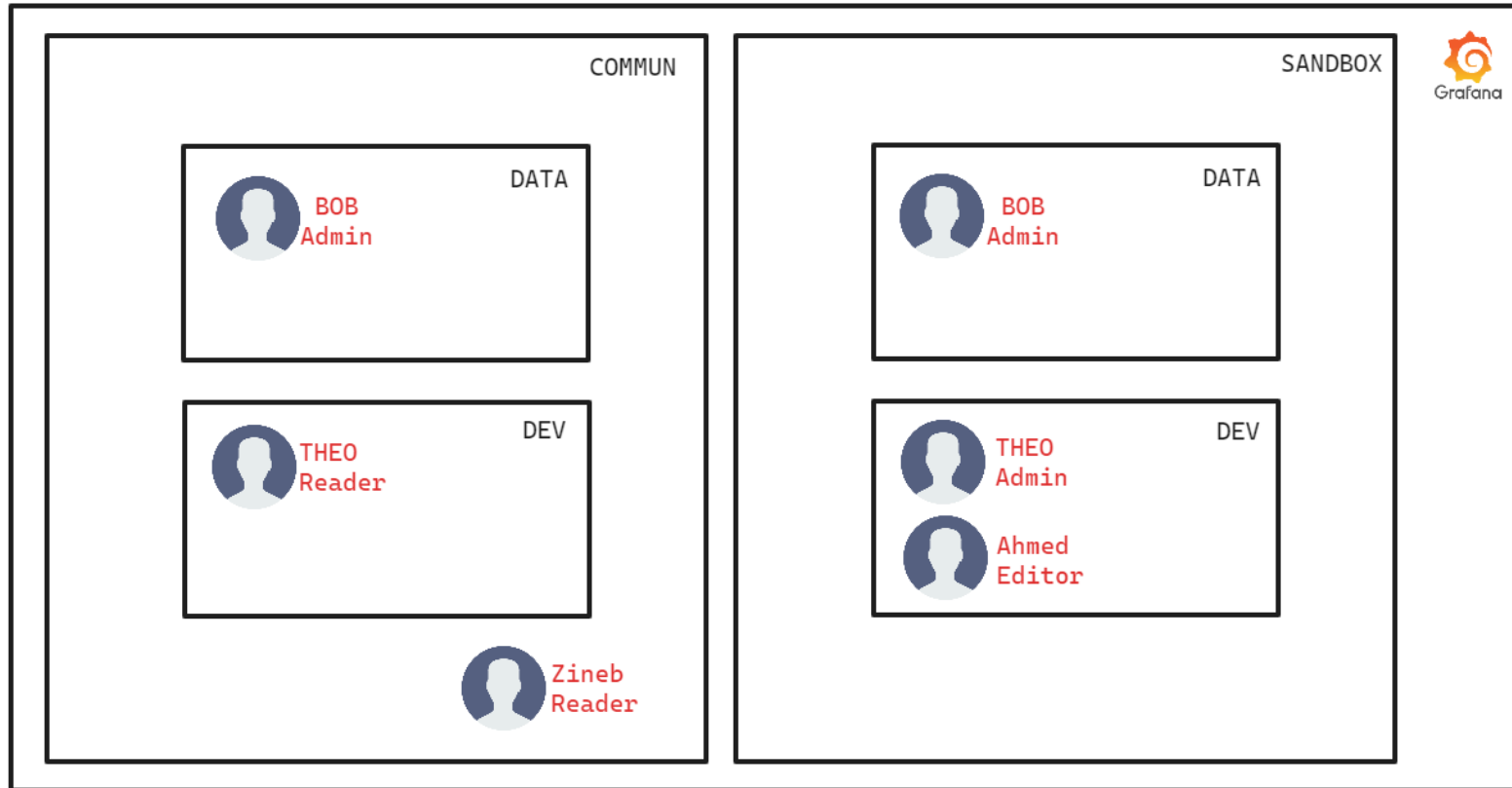


<https://grafana.com/grafana/plugins/all-plugins/>

TP: Variables dans Grafana



TP: Gérer les utilisateurs d'une organisation



TP: Gérer les utilisateurs d'une organisation

Fichiers contenant
la liste des utilisateurs



Script Python pour
automatiser la création
des utilisateurs sur Grafana



Merci

Questions ?