

# Prometheus & Grafana

OMNISHORE®



Marius N'KOUBA  
Data Engineer

May 2024

- 
- [Outils à installer](#)
- [Architecture](#)
  - [Configuration des VMs](#)
  - [Mise en place de l'environnement de travail](#)
- [Prometheus](#)
  - [Les types de métriques](#)
  - [Installer prometheus](#)
  - [Interface Graphique](#)
- [Exporters](#)
  - [Installer l'exporter node\\_exporter sur les VMs grafana et app](#)
  - [PostgreSql Exporter](#)
  - [Mysql Exporter](#)
- [PromQL](#)
  - [PromQL: Filtres sur les Labels](#)
  - [Requêtes:](#)
  - [Les opérateurs](#)

- [PromQL: Group By](#)
- [PromQL: Les Intervalles de temps](#)
- [PromQL : Offset](#)
- [Grafana](#)
  - [Installer Grafana](#)
  - [Ajouter des Métriques sur Grafana](#)
    - [Node Exporter Dashboard ID : 1860](#)
    - [Postgres Exporter Dashboard ID : 9628](#)
    - [Mysql Exporter Dashboard ID : 14057](#)
- [Monitorer une application Springboot](#)
  - [SETUP SPRING BOOT](#)
  - [Scrape Springboot App Metrics](#)
  - [AJOUT D'UN PANEL DE MÉTRIQUE PERSONNALISÉ](#)
- [Plugins dans Grafana](#)
  - [Dynamic Text](#)
    - [TP pratique](#)
- [TP : Utilisation des Variables dans Grafana](#)
- [Gérer les utilisateurs d'une organisation](#)
  - [Automatiser la création des utilisateurs sur Grafana](#)
- [TP Haproxy Exporter](#)
  - [Install Haproxy](#)

## Outils à installer

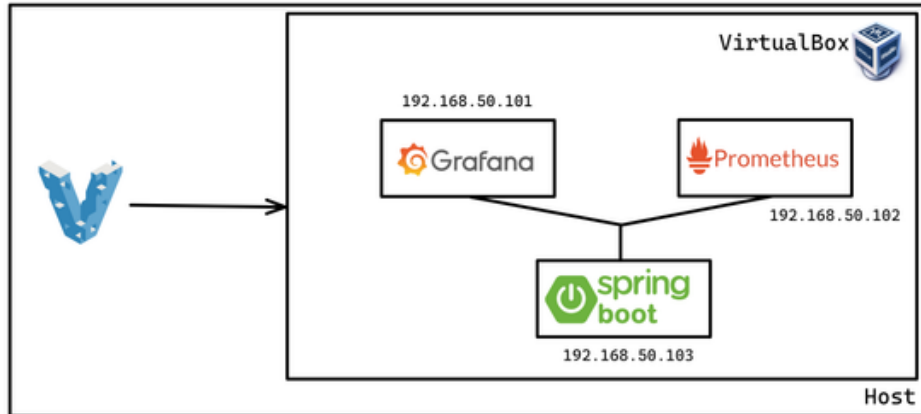
Pour le bon déroulement de cette formation, nous aurons besoin de quelques outils sur nos machines.

- Vs Code <https://code.visualstudio.com/download>
- Vagrant <https://developer.hashicorp.com/vagrant/downloads>
- VirtualBox [Downloads – Oracle VM VirtualBox](#)

## Architecture

Notre architecture sera essentiellement composée de trois VMs

- grafana (192.168.50.101) : Cette VM va héberger notre service Grafana.
- prometheus (192.168.50.102) : On aura ici le service prometheus
- app (192.168.50.103) : Sur cette dernière, nous aurons une application Spring Boot et une base de données.



## Configuration des VMs

URL	Software	Hardwares	Installed services
grafana 192.168.50.101	<ul style="list-style-type: none"> <li>• Ubuntu= "22.04.2"</li> <li>• Grafana</li> </ul>	<ul style="list-style-type: none"> <li>• RAM: 2</li> <li>• CPU: 2</li> <li>• Data Storage: <b>Dynamic</b></li> </ul>	<ul style="list-style-type: none"> <li>• Grafana</li> <li>• node-exporter</li> </ul>
prometheus 192.168.50.102	<ul style="list-style-type: none"> <li>• Ubuntu= "22.04.2"</li> <li>• prometheus</li> </ul>	<ul style="list-style-type: none"> <li>• RAM: 2</li> <li>• CPU: 2</li> <li>• Data Storage: <b>Dynamic</b></li> </ul>	<ul style="list-style-type: none"> <li>• Prometheus</li> <li>• node-exporter</li> </ul>
app 192.168.50.103	<ul style="list-style-type: none"> <li>• Ubuntu= "22.04.2"</li> </ul>	<ul style="list-style-type: none"> <li>• RAM: 2</li> <li>• CPU: 2</li> <li>• Data Storage: <b>Dynamic</b></li> </ul>	<ul style="list-style-type: none"> <li>• Java</li> <li>• node-exporter</li> </ul>

Vagrant sera utilisé pour créer nos machines virtuelles en local sur VirtualBox.

## Mise en place de l'environnement de travail

- Créer un dossier `training`, demarrer Vs Code depuis le repertoire , créer un fichier de configuration nommé `Vagrantfile` puis coller la config suivante
- `Vagrant.configure(2) do |config|`

- etcHosts = ""
- config.vm.box = "ubuntu/jammy64"
- config.vm.box\_url = "ubuntu/jammy64"
- config.vm.boot\_timeout = 6000
- 
- NODES = [
- { :hostname => "grafana", :ip => "192.168.50.101", :cpus => 2,
- :mem => 2048 },
- { :hostname => "prometheus", :ip => "192.168.50.102", :cpus =>
- 2, :mem => 2048 },
- { :hostname => "app", :ip => "192.168.50.103", :cpus => 2, :mem
- => 2048 }
- ]
- 
- # Define /etc/hosts for all servers
- NODES.each do |node|
- etcHosts += "echo '#{node[:ip]}   #{node[:hostname}}' >>
- /etc/hosts\n"
- end
- 
- NODES.each do |node|
- config.vm.define node[:hostname] do |cfg|
- cfg.vm.hostname = node[:hostname]
- cfg.vm.network "private\_network", ip: node[:ip]
- cfg.vm.provider "virtualbox" do |v|
- v.customize [ "modifyvm", :id, "--cpus", node[:cpus] ]
- v.customize [ "modifyvm", :id, "--memory", node[:mem] ]
- v.customize [ "modifyvm", :id, "--natdnshostresolver1", "on" ]
- v.customize [ "modifyvm", :id, "--natdnspoxy1", "on" ]
- v.customize [ "modifyvm", :id, "--name", node[:hostname] ]
- end
- end
- cfg.vm.provision :shell, :inline => etcHosts
- end
- end
- end

- Demarrer l'environnement de travail

```
vagrant up
```

Une fois l'exécution de la commande terminée,

```
vagrant ssh grafana
ping -c 5 prometheus
exit
```

```
vagrant ssh prometheus
ping app
```

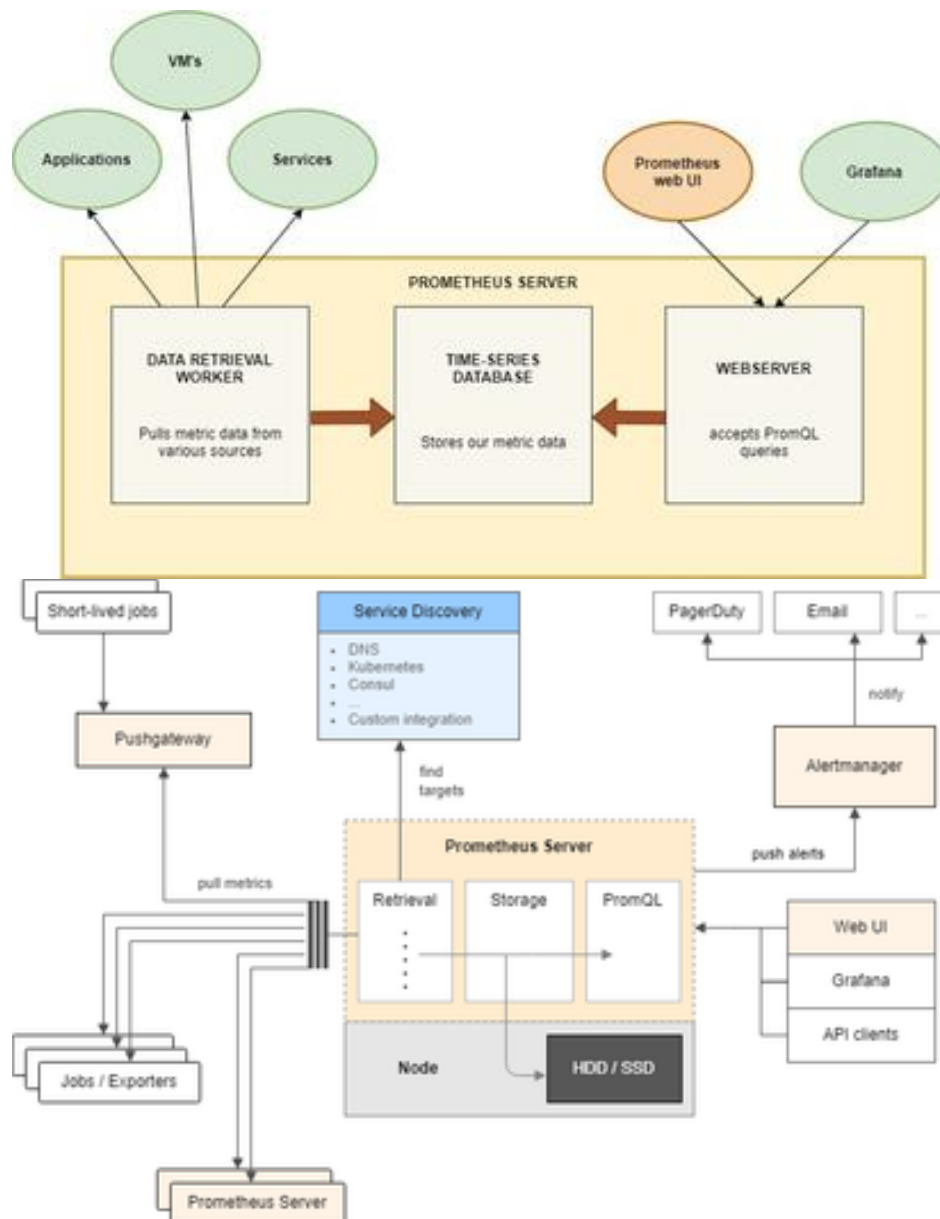
```
exit
```

```
vagrant ssh app  
ping grafana  
exit
```

# 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.



Les caractéristiques qui distinguent Prometheus des autres systèmes de métriques et de surveillance sont :

- 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 pour exploiter cette dimensionnalité
- Aucune dépendance au stockage distribué ; les nœuds de serveur unique sont autonomes
- 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

- C'est une base de données Base Time series + Serveur Web + Moteur de Base de Données
- Il fait essentiellement du Scraping de données à des fréquences régulières (C'est lui qui va chercher l'information)
- 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

## Les types de métriques

1. **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.
2. **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.
3. **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.
4. **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.

## Installer prometheus

```
vagrant ssh prometheus
```

- Mettre à jour les packages système

```
sudo apt update
```

- Créer un utilisateur système pour Prometheus

```
sudo groupadd --system prometheus
sudo useradd -s /sbin/nologin --system -g prometheus prometheus
```

- Créer des répertoires pour Prometheus

```
sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
```

- **Téléchargez et extraire Prometheus**

```
wget
https://github.com/prometheus/prometheus/releases/download/v2.51.1/prometheus-2.51.1.linux-amd64.tar.gz

tar vxf prometheus*.tar.gz

cd prometheus*/
```

- **Déplacer les fichiers binaires**

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

- **Déplacer les fichiers de configuration**

```
sudo mv consoles /etc/prometheus
sudo mv console_libraries /etc/prometheus
sudo mv prometheus.yml /etc/prometheus

sudo chown prometheus:prometheus /etc/prometheus
sudo chown -R prometheus:prometheus /etc/prometheus/consoles
sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
sudo chown -R prometheus:prometheus /var/lib/prometheus

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

- **Contenu du fichier de configuration**

```
- Configuration
  - Global : Configurations qui s'applique en général
    - scrape_interval (ou job): Intervale de récupération des données
    - evaluation_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: Définition de labels (important pour grafana/
Standardiser )
    - targets : url/ip:port
```

- **Créer le service Prometheus Systemd**

```
sudo nano /etc/systemd/system/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
```

- Reload Systemd

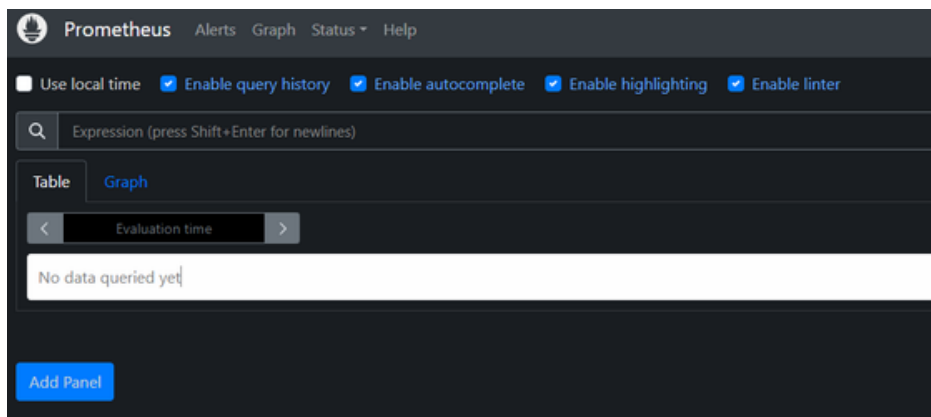
```
sudo systemctl daemon-reload
```

- Start Prometheus Service

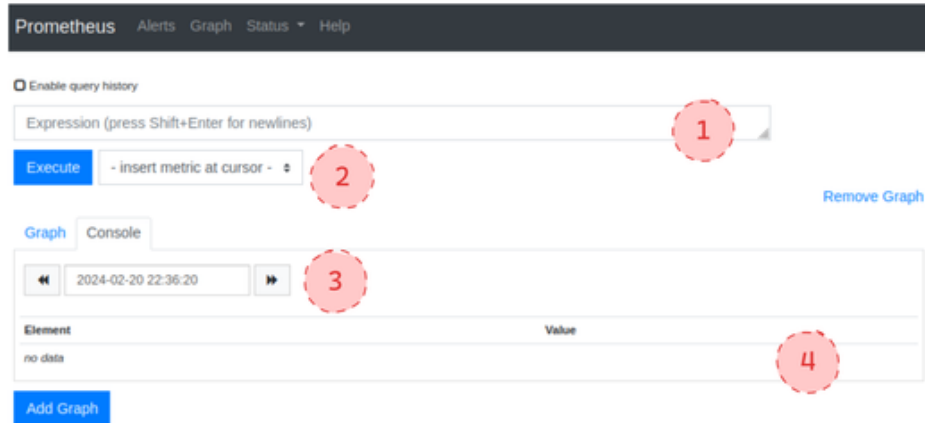
```
sudo systemctl enable prometheus
sudo systemctl start prometheus
```

```
sudo systemctl status prometheus
```

Le Link vers Prometheus → <http://192.168.50.102:9090/>



## Interface Graphique



1. Effectuer des requêtes
2. Liste déroulante contenant toutes les métriques
3. Intervalle de temps pour les données retournées
4. Section pour les résultats des requêtes

# Exporters

## Installer l'exporter node\_exporter sur les VMs grafana et app

Agent qui permet de collecter des métriques systèmes :

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

Node Exporter : [https://github.com/prometheus/node\\_exporter](https://github.com/prometheus/node_exporter)

[https://prometheus.io/download/#node\\_exporter](https://prometheus.io/download/#node_exporter)

vagrant ssh app

- Préparation

```
sudo useradd -rs /bin/false node_exporter
```

```
wget
https://github.com/prometheus/node_exporter/releases/download/v1.7.0/node_exporter-1.7.0.linux-amd64.tar.gz
```

```
tar -xvzf node_exporter-1.7.0.linux-amd64.tar.gz
sudo mv node_exporter-1.7.0.linux-amd64/node_exporter /usr/local/bin/
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

- Installer le service systemd pour notre node Exporter

```
sudo nano /etc/systemd/system/node_exporter.service
[Unit]
Description=Node Exporter
After=network-online.target
[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter --web.listen-address=:7676
[Install]
WantedBy=multi-user.target
```

- Demarrer le service Node-Exporter

```
sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
```

- Ajouter la config de Scraping a prometheus

Connectez-vous a la VM prometheus

```
vagrant ssh prometheus
sudo nano /etc/prometheus/prometheus.yml
- job_name: node-exporter
  static_configs:
    - targets: ['192.168.50.103:7676']
sudo systemctl restart prometheus
```

## PostgreSql Exporter

Doc : <https://grafana.com/oss/prometheus/exporters/postgres-exporter/?tab=installation>

- Installer PostgreSql

```
sudo apt install postgresql
systemctl status postgresql
```

- Ajout d'un mot de passe pour l'utilisateur postgres

```
sudo su - postgres
psql
alter user postgres encrypted password 'password';
exit
```

```
psql -h 127.0.0.1 -U postgres
# Enter 'password' as password
```

- **Installer postgres Exporter**

```
wget https://github.com/prometheus-
community/postgres_exporter/releases/download/v0.9.0/
postgres_exporter-0.9.0.linux-amd64.tar.gz
```

```
tar xvfz postgres_exporter-*.linux-amd64.tar.gz
cd postgres_exporter-*.linux-amd64
```

```
export DATA_SOURCE_NAME='postgresql://postgres:password@127.0.0.1:5432/
postgres?sslmode=disable'
```

```
./postgres_exporter
```

- **Mettre a jour le fichier de configuration Prometheus**

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

```
- job_name: postgres
  static_configs:
    - targets: ['192.168.50.103:9187']
```

```
sudo systemctl restart prometheus
```

```
http://192.168.50.103:9187/metrics
```

## **Mysql Exporter**

Doc : <https://grafana.com/oss/prometheus/exporters/mysql-exporter/?tab=installation>

- **Install Mysql**

```
sudo apt update
sudo apt install mysql-server
sudo systemctl start mysql.service
sudo mysql_secure_installation
sudo mysql
```

- **Installer Mysql Exporter**

```
sudo useradd -rs /bin/false exporter
```

```
wget https://github.com/prometheus/mysqld_exporter/releases/download/v0.12.1/
mysqld_exporter-0.12.1.linux-amd64.tar.gz
```

```
tar xvfz mysqld_exporter-*.tar.gz
cd mysqld_exporter-*.tar.gz
```

Avant d'exécuter l'exportateur MySQL, vous devez d'abord créer l'utilisateur MySQL qu'il utilisera pour récupérer les métriques de la base de données. Connectez-vous à votre base de données MySQL et exécutez la commande suivante en tant qu'utilisateur disposant de privilèges administratifs :

```
CREATE USER 'exporter'@'127.0.0.1' IDENTIFIED BY 'Password#2024' WITH
MAX_USER_CONNECTIONS 3;

GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'127.0.0.1';
GRANT SUPER ON *.* TO 'exporter'@'127.0.0.1';

FLUSH PRIVILEGES;
export DATA_SOURCE_NAME='exporter:Password#2024@(127.0.0.1:3306)/'
```

Exécutez l'exporter :

```
./mysqld_exporter
```

<http://192.168.50.103:9104/metrics>

- Mettre à jour le fichier de configuration Prometheus

```
vagrant ssh prometheus
sudo nano /etc/prometheus/prometheus.yml
- job_name: mysql
  static_configs:
  - targets: ['192.168.50.103:9104']
sudo systemctl restart prometheus
```

Dashboard Grafana : **14057**

# PromQL

## PromQL: Filtres sur les Labels

les labels permettent de faire du filtrage

- Equivalent à une clause WHERE
- utilise les Regexp GO
- Conseils : <https://prometheus.io/docs/practices/naming/>

## Rename labels

```
scrape_configs:
- job_name: Consul
```

```

consul_sd_configs:
  - server: '192.168.57.10:8500'
    datacenter: 'mydc'
relabel_configs:
  - source_labels: [__meta_consul_service]
    target_label: job

```

## Requêtes:

```

up
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"}

```

## Les opérateurs

```

== (equal)
!= (not equal)
> greater
< less than
= greater or equal
<= less or equal
and = intersection
or = union
unless = complement
node_load1 > 1.0 and node_load1 < 1.6

```

## Creer de la charge pour le CPU

```

while true; do df ;done

```

## PromQL: Group By

Operation d'aggregation: count , sum

```

clause = by ()
(cpu, instance)

```

```
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)
```

## PromQL: Les Intervalles de temps

Prometheus scrape que la dernière value connue

- Range Vector : []
  - cherche tous les timeseries sur un interval de temps
  - par défaut par rapport a maintenant

```
node_load1[1m]
```

- Offset : offset 3m
- change la date a laquelle on recherche la timeseries

```
node_load1[1m] offset 3m
```

## PromQL : Offset

```
node_load1 - node_load5 offset 10m
```

- Offset permet de deplacer son point de référence
- ideal pour comparer 2 periodes
- `node_load1 - node_load1 offset 30m`  
`node_load1 - sum_over_time(node_load1[5m] offset 5m)`

# 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.

## Installer Grafana

- Grafana from APT repository

<https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/>

- Install the prerequisite packages

```
sudo apt-get install -y apt-transport-https software-properties-common wget
```

- Import the GPG key

```
sudo mkdir -p /etc/apt/keyrings/
```

```
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo  
tee /etc/apt/keyrings/grafana.gpg > /dev/null
```

- To add a repository for stable releases, run the following command

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com  
stable main"  
| sudo tee -a /etc/apt/sources.list.d/grafana.list
```

- To add a repository for beta releases, run the following command:

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com  
beta main"  
| sudo tee -a /etc/apt/sources.list.d/grafana.list
```

- Run the following command to update the list of available packages:

```
sudo apt-get update
```

- To install Grafana OSS, run the following command:

```
sudo apt-get install grafana=10.0.0
```

- To install Grafana Enterprise, run the following command:

```
sudo apt-get install grafana-enterprise=10.0.0
```

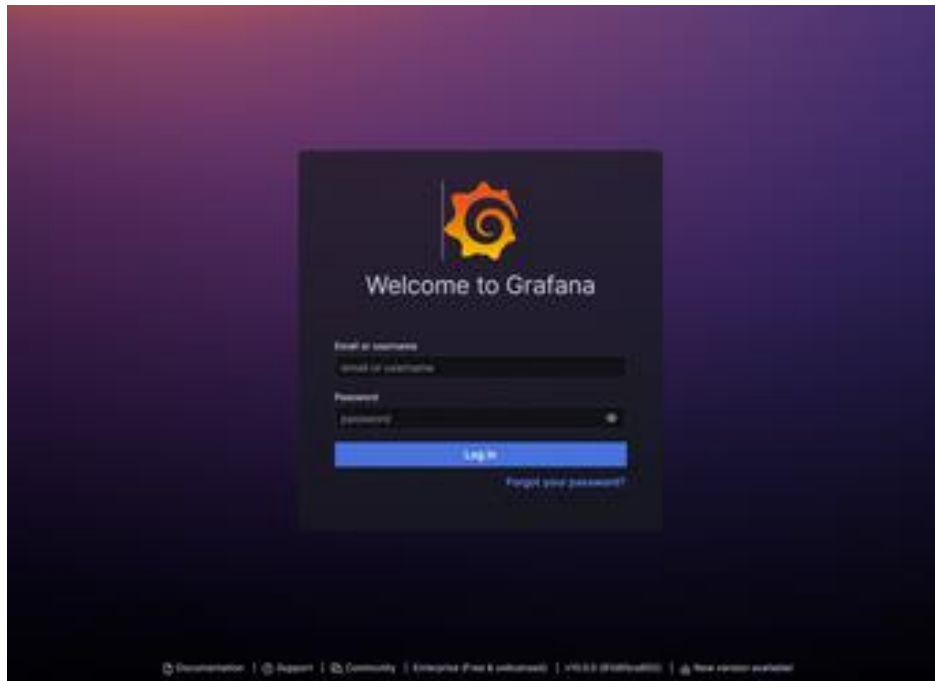
- Start the service

```
sudo systemctl daemon-reload  
sudo systemctl start grafana-server  
sudo systemctl status grafana-server
```

- Configure the Grafana server to start at boot using systemd

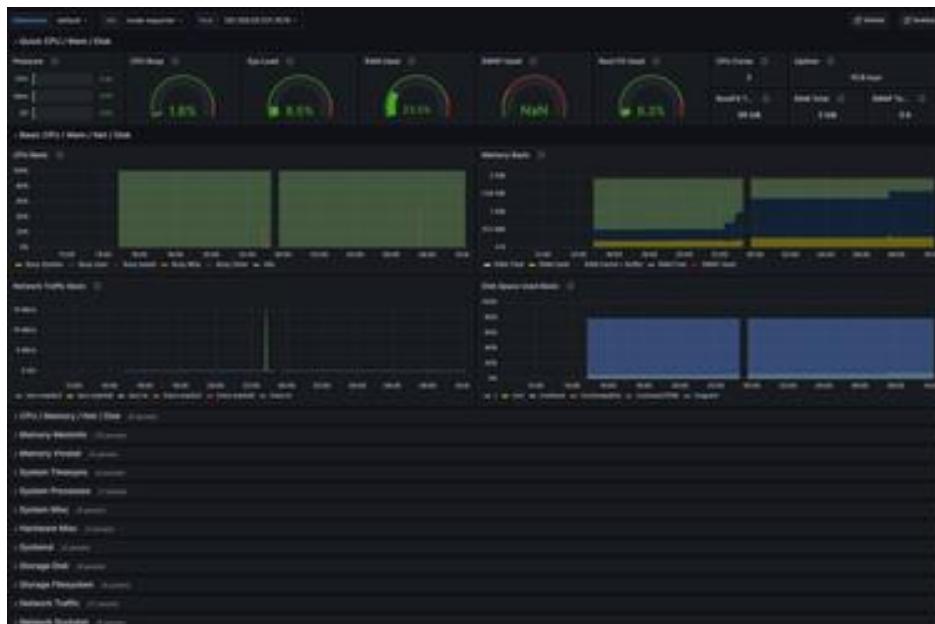
```
sudo systemctl enable grafana-server.service
```



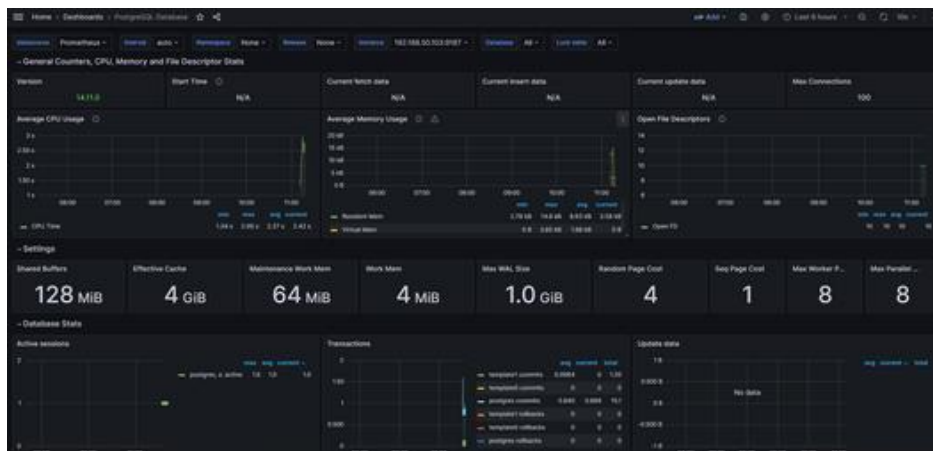


## Ajouter des Métriques sur Grafana

Node Exporter Dashboard ID : 1860



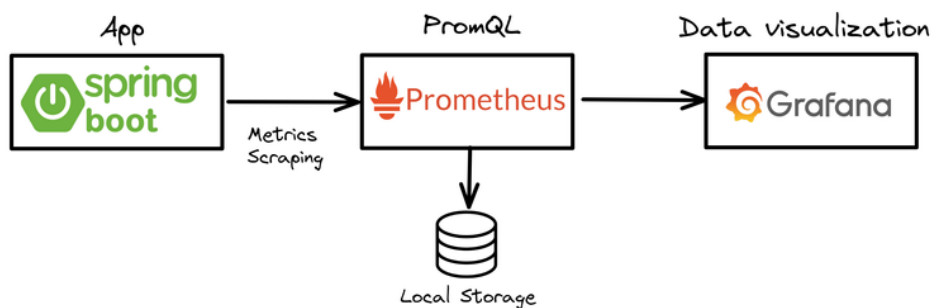
Postgres Exporter Dashboard ID : 9628



MySQL Exporter Dashboard ID : 14057



## Monitorer une application Springboot



## SETUP SPRING BOOT

Je vais vous guider à travers les étapes pour mettre en place une application Spring Boot de base que nous surveillerons en utilisant des images Docker de Prometheus et Grafana.

1. Set up a regular Spring Boot application by using [Spring Initializr](#).

Dependencies :

- Spring web
  - Spring Boot DevTools
2. Add dependency for Actuator
  3. `<dependency>`
  4.     `<groupId>org.springframework.boot</groupId>`
  5.     `<artifactId>spring-boot-starter-actuator</artifactId>`  
   `</dependency>`
  6. Add dependency for Micrometer
  7. `<dependency>`
  8.     `<groupId>io.micrometer</groupId>`
  9.     `<artifactId>micrometer-registry-prometheus</artifactId>`
  10.    `<version>1.5.5</version>`  
   `</dependency>`
  11. Expose our needed Prometheus endpoint in the application.properties file
  12.    `management.endpoints.web.exposure.include=prometheus`
  13.    `management.endpoint.health.show-details=always`  
      `management.metrics.tags.application= MonitoringSpringDemoProject`

Or Clone the repository containing an Example Project

```
git clone https://github.com/nkoubamarius/prometheus_training.git
cd /home/vagrant/prometheus_training/springboot_tp/demo
```

5. Start the application

```
sudo apt-get install maven
```

Compile the app

```
mvn clean install
cd target
java -jar demo-0.0.1-SNAPSHOT.jar
```

<http://192.168.50.103:8080/>

Springboot Application Metrics → <http://192.168.50.103:8080/actuator/prometheus>

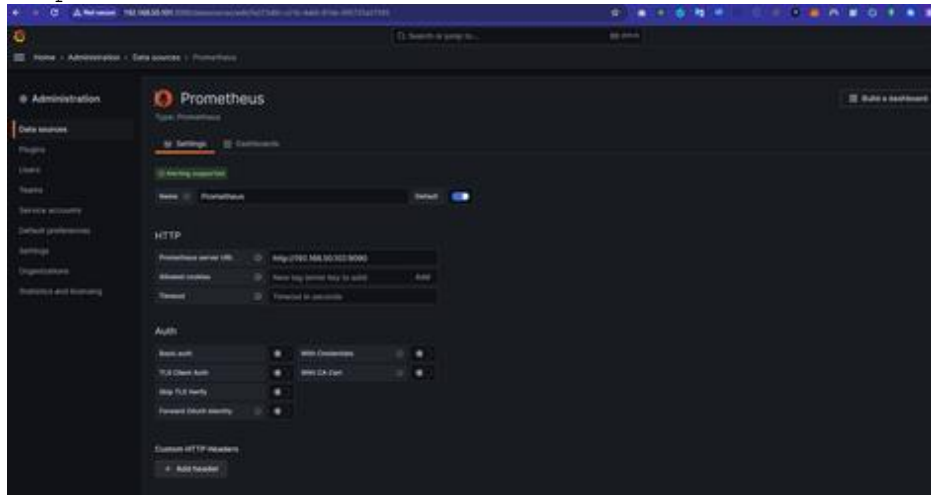
## Scrape Springboot App Metrics

```
sudo nano /etc/prometheus/prometheus.yml
- job_name: 'spring-actuator'
  metrics_path: '/actuator/prometheus'
  scrape_interval: 5s
  static_configs:
```

```
- targets: ['192.168.50.103:8080']  
sudo systemctl restart prometheus
```

- Ouvrez Grafana et connectez vous avec le user `admin` et le mot de passe `admin`
- Naviguez vers Configuration > Sources de données, ajoutez une source de données Prometheus et configurez-la comme dans l'exemple ci-dessous

<http://192.168.50.102:9090>



Pour cet exemple, j'ai utilisé l'un des tableaux de bord prédéfinis que vous pouvez trouver sur la page [Tableaux de bord Grafana](#). Le tableau de bord que j'ai utilisé pour surveiller notre application est le tableau de bord JVM Micrometer avec l'identifiant d'importation : 4701.

Home

Dashboards

Import dashboard

Dashboards

Playlists

Snapshots

Library panels

Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse

Accepted file types: json, txt

Import via grafana.com

4701

Load

Import via panel json

Load

Cancel

Import dashboard

Import dashboard from file or Grafana.com

Importing dashboard from Grafana.com

Published by

rweltrauch

Updated on

2023-05-04 23:00:06

Options

Name

JVM (Micrometer)

Folder

General

Unique Identifier (UID)

The unique identifier (UID) of a dashboard can be used to uniquely identify a dashboard between multiple Grafana instances. The UID allows having consistent URLs for accessing dashboards as changing the title of a dashboard will not break any bookmarked links to that dashboard.

Change uid

Prometheus

Prometheus

Import

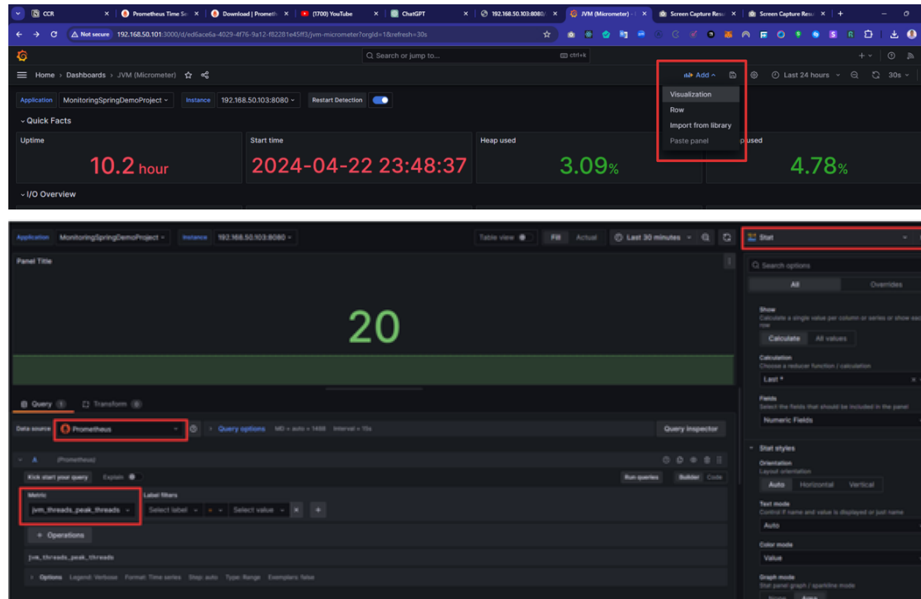
Cancel



**AJOUT D'UN PANEL DE MÉTRIQUE PERSONNALISÉ**

Pour démontrer comment créer un panel pour l'une de nos propres métriques personnalisées, je vais énumérer les étapes nécessaires ci-dessous.

Tout d'abord, nous devons ajouter un panel en cliquant sur "Add" en haut de la page, puis à nouveau sur "Visualization".



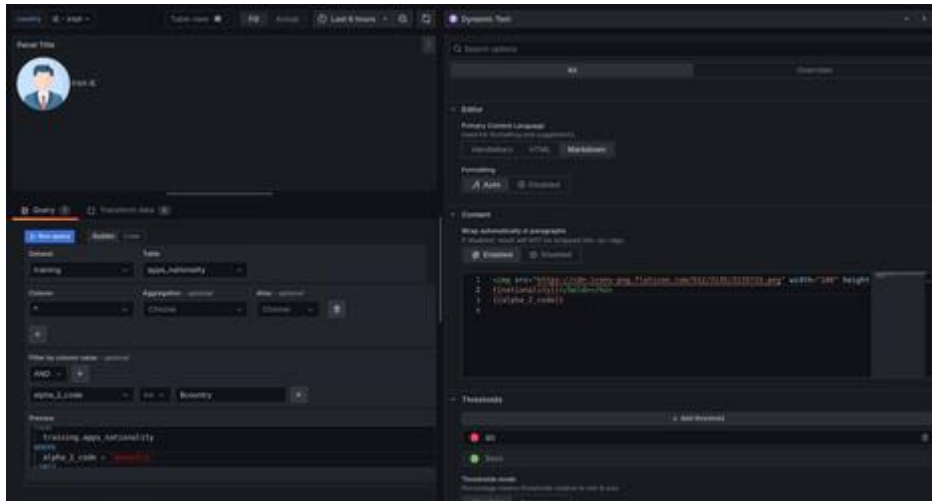
# Plugins dans Grafana

## Dynamic Text

Doc: <https://volkovlabs.io/plugins/volkovlabs-dynamic-text-panel/>

```
sudo grafana-cli plugins install marcusolsson-dynamic-text-panel
sudo systemctl start grafana-server
```

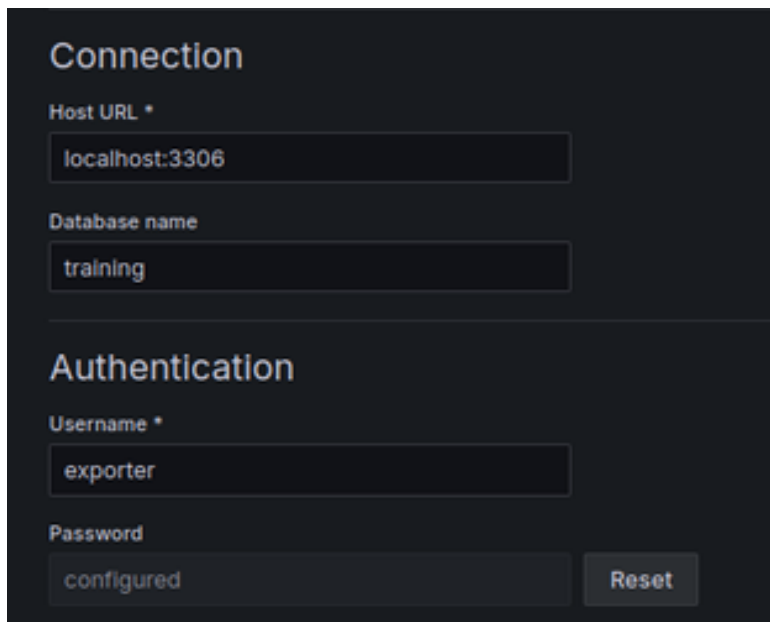
## TP pratique



## TP : Utilisation des Variables dans Grafana

- Créer une table dans MySql
- `CREATE database training;`
- `use training;`
- 
- `CREATE TABLE IF NOT EXISTS `apps_nationality` (`
- ``id` int(11) NOT NULL auto_increment,`
- ``num_code` int(3) NOT NULL DEFAULT '0',`
- ``alpha_2_code` varchar(2) DEFAULT NULL,`
- ``alpha_3_code` varchar(3) DEFAULT NULL,`
- ``en_short_name` varchar(52) DEFAULT NULL,`
- ``nationality` varchar(39) DEFAULT NULL,`
- `PRIMARY KEY (`id`),`
- `UNIQUE KEY `alpha_2_code` (`alpha_2_code`),`
- `UNIQUE KEY `alpha_3_code` (`alpha_3_code`)`
- `);`
- `INSERT INTO `apps_nationality` (`num_code`, `alpha_2_code`,`
- ``alpha_3_code`, `en_short_name`, `nationality`) VALUES`
- `( "246", "FI", "FIN", "Finland", "Finnish"),`
- `( "250", "FR", "FRA", "France", "French"),`
- `( "372", "IE", "IRL", "Ireland", "Irish"),`
- `( "404", "KE", "KEN", "Kenya", "Kenyan"),`
- `( "504", "MA", "MAR", "Morocco", "Moroccan"),`
- `( "508", "MZ", "MOZ", "Mozambique", "Mozambican"),`
- `( "104", "MM", "MMR", "Myanmar", "Burmese");`
- Ajouter un plugin Mysql dans Grafana





The image shows a dark-themed web interface for database connection and authentication. It is divided into two main sections: 'Connection' and 'Authentication'. In the 'Connection' section, there is a 'Host URL \*' field with the value 'localhost:3306' and a 'Database name' field with the value 'training'. The 'Authentication' section contains a 'Username \*' field with the value 'exporter' and a 'Password' field with the value 'configured'. A 'Reset' button is located to the right of the password field.

### Connection

Host URL \*

localhost:3306

Database name

training

---

### Authentication

Username \*

exporter

Password

configured

Reset

- Créer une variable contenant la liste des pays
- ```
SELECT CONCAT(`alpha_2_code`, ' - ', `nationality`) AS __text,  
       `alpha_2_code` AS __value FROM apps_nationality
```

## country

Select variable type

Query

### General

Name  
The name of the template variable. (Max: 30 characters)

country

Label  
Optional display name

country

Description

Descriptive text

Show on dashboard

Label and value Value Nothing

### Query options

Data source

mysql

### Query

```
SELECT CONCAT(alpha_2_code, ' - ', nationality) AS _text,
alpha_2_code AS _value FROM apps.nationality
```

- Build a panel

country KE - Kenyan

Panel Title

KE

- Loop Row for each Country

Row options

Title

\$country

Repeat for

country

Cancel Update

country FI - Finnish + FR - French + IE - Irish + KE - Kenyan + MA - Morocc...

Panel Title

FI

FR - French

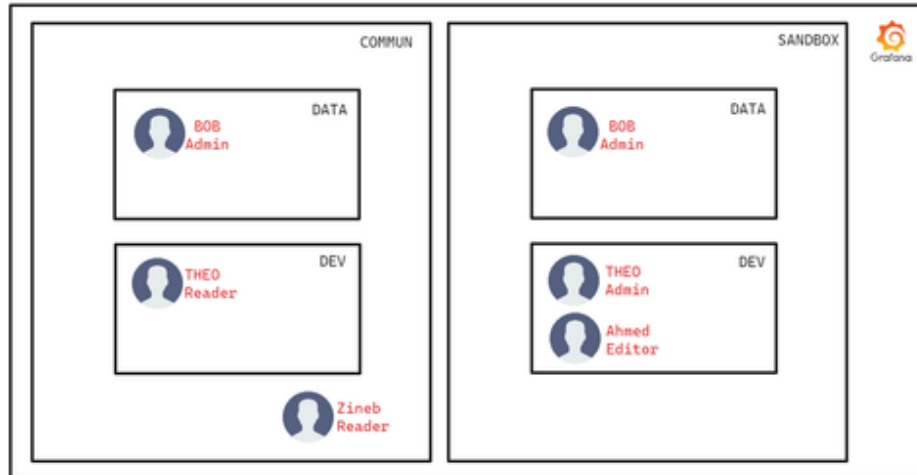
Panel Title

FR

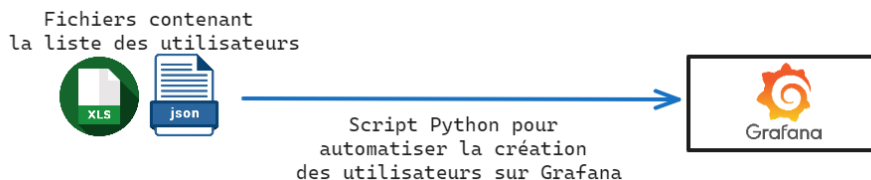
## Gérer les utilisateurs d'une organisation

Les administrateurs d'organisations peuvent inviter des utilisateurs à rejoindre leur organisation. Les utilisateurs de l'organisation ont accès aux ressources de l'organisation en fonction de leur rôle, à savoir Administrateur, Éditeur ou Lecteur.

Les autorisations associées à chaque rôle déterminent les tâches qu'un utilisateur peut effectuer dans le système.



## Automatiser la création des utilisateurs sur Grafana



## TP Haproxy Exporter

HAProxy is a free, very fast and reliable reverse-proxy offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for very high traffic web sites and powers a significant portion of the world's most visited ones

- Proxy : met en relation entre client/serveur
- load balancing: repartition de charge entre plusieurs machines cibles
- protège d'attaques directes
- écoute sur des adresses ou ports (eventuellement translation)
- configuration des timeout (regles) un serveur est done

## Install Haproxy

```
sudo apt install haproxy
```

```
sudo nano /etc/haproxy/haproxy.cfg
frontend stats
    bind *:8181
    stats enable
    stats uri /stats
    stats refresh 10s
```

```
frontend app1
    bind *:1234
    use_backend back_app1

backend back_app1
    server srv1 127.0.0.1:8086
    server srv2 127.0.0.1:8888
service haproxy restart
service haproxy status
```

## SimpleHTTPServer

### Lancer deux servers Python

```
python3 -m http.server 8086
python3 -m http.server 8888
```

Statistiques = <http://localhost:8181/stats>