UNIVERSITÉ DE RENNES 1

ESIR ECOLE SUPERIEURE D'INGENIEURS DE RENNES

03/02/2019

TD8

> *« J'atteste que ce travail est original, qu'il indique de façon appropriée tous les emprunts, et qu'il fait référence de façon appropriée à chaque source utilisée »*

Léo Guilpain – IoT – ESIR3

Table des matières

Introduction

Le but de ce TP est de visualiser les données de notre application à l'aide de Grafana. Nous essayerons ensuite de simuler des clients pour augmenter la charge sur cette dernière.

Partie I - Retour sur notre galerie

Question 01

Il peut y avoir plusieurs types de problèmes de façon général :

- **CPU :** Si on fait des gros calculs tels que des graphiques ou alors des calculs scientifiques, cela peut poser problème en termes de performances. Cependant, cela ne devrait pas poser de problèmes pour le déploiement de notre application.
- **Mémoire Disque :** La mémoire disque ne représente pas un problème pour le déploiement des installations. En revanche, pour nous, comme nos machines sont peu performantes et que l'espace disque est réduit, cela a un gros impact sur notre déploiement. Pour lutter contre cela, le seul moyen est d'augmenter l'espace disque sur les machines ou alors d'acquérir des machines plus performantes.
- **Accès disque :** Le problème peut survenir si des clients ajoutent des photos en parallèle. Cela peut avoir un impact sur l'accès disque puisque ces dernières sont stockées que sur un seul disque. Pour lutter contre cela, il existe plusieurs solutions notamment celle d'augmenter le nombre de disque. Cette augmentation peut permettre de répartir la charge sur ces derniers et donc éviter que l'écriture se fasse que sur un seul disque. On peut aussi mettre les données en cache. L'image est stockée dans le cache lorsqu'elle est utilisée puis supprimée lorsqu'elle ne l'est plus. En revanche, la mise en cache des photos va permettre de régler les problèmes de lecture mais pas d'écritures.

Partie II - Collecte et affichage de données

Question 01

Dans un premier temps on crée le fichier “monitor-stack.yml”

```yaml
version: "3"
services:
  prometheus:
    image: prom/prometheus
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
    ports:
      - 9090:9090
    volumes:
      - '/tmp/prometheus.yml:/etc/prometheus/prometheus.yml'
    networks:
      - test_net
  cadvisor:
    image: google/cadvisor:latest
    ports:
      - 8090:8080
    volumes:
      - '/var/lib/docker/:/var/lib/docker'
      - '/dev/disk/:/dev/disk'
      - '/sys:/sys'
      - '/var/run:/var/run'
      - '/:/rootfs:ro'
    deploy:
      mode: global
    networks:
      - test_net
networks:
  test_net:
    external: true
```

Figure 1 : monitor-stack.yml

```
vagrant@manager:/vagrant$ sudo docker stack deploy -c monitor-stack.yml myappstack
Updating service myappstack_prometheus (id: tzknfv0l2c1rrdqmsd15cn3zf)
Creating service myappstack_cadvisor
```

Figure 2 : deploy monitor stack

On déploie donc deux services. Le premier correspond à prometheus et le second à cadvisor.

Les services sont bien lancés correctement. Lorsque l’on se rend sur la page : http://192.168.42.17:8090/containers/

On tombe sur ceci :

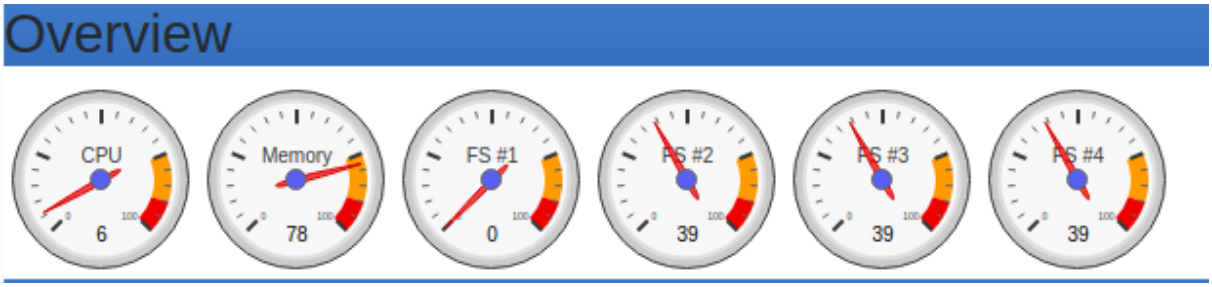

Figure 3 : Overview

On peut voir que la mémoire se trouve dans le orange, donc cela risque de poser problème par la suite. En revanche les autres valeurs sont correctes.

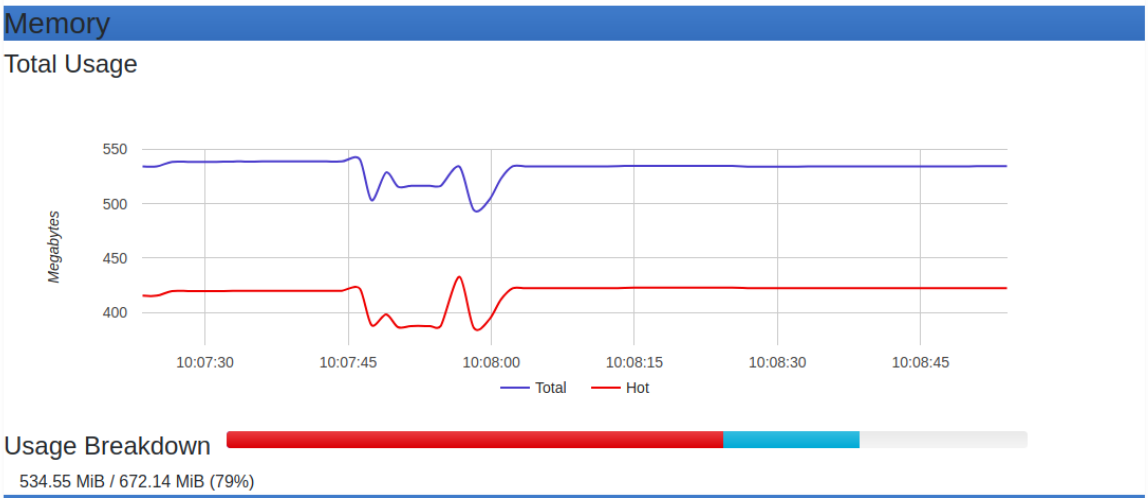

Figure 4 : Memory Usage

La mémoire est utilisée à 79% ce qui est plutôt faible pour la suite.

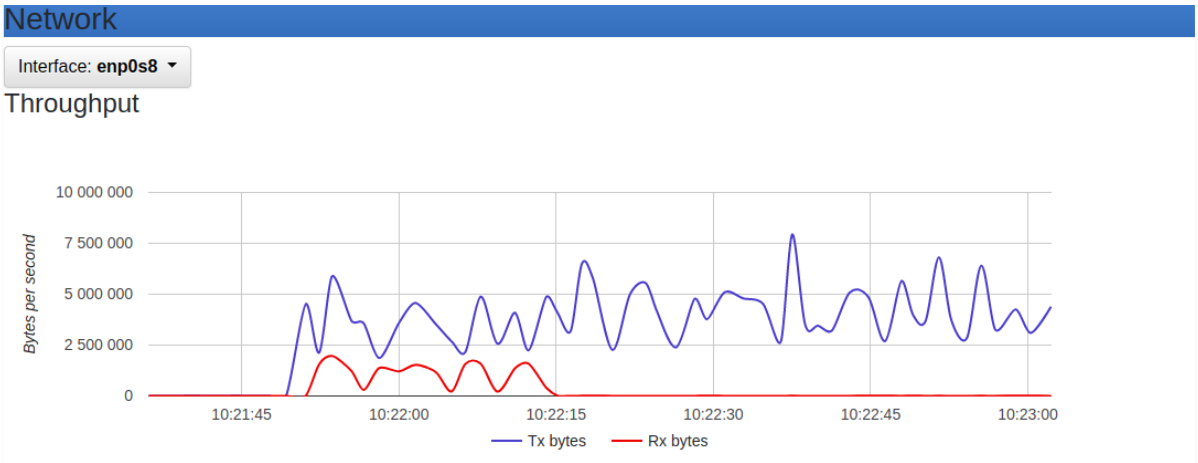

Figure 5 : Throughput

Comme on peut le voir, il y a du trafic.

Tous ces graphiques ont été pris sur cadvisor.

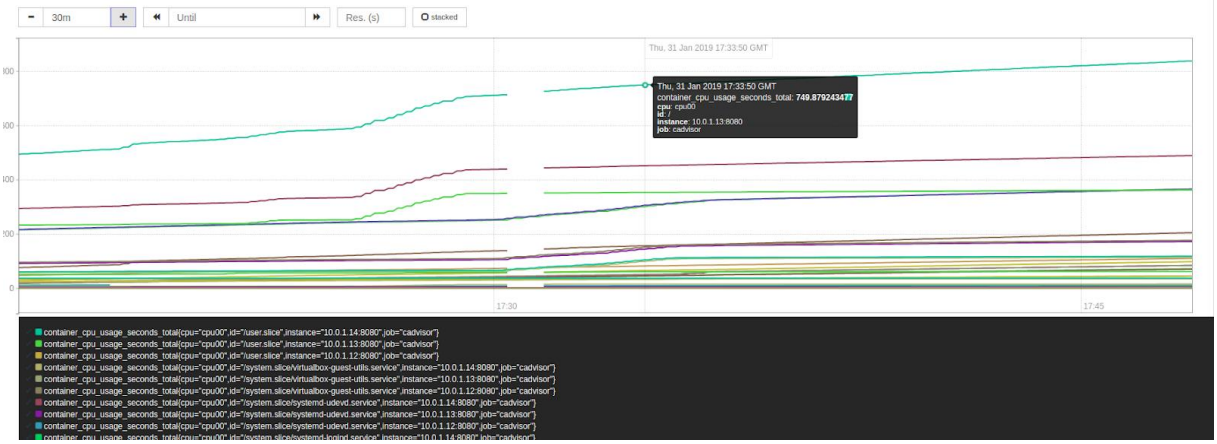

Figure 6 : Prometheus graph

Ce graphique sur prometheus nous permet de visionnage l'utilisation du CPU.

Question 02

On ajoute Graphana au monitor-stack :

```
grafana:
  image: grafana/grafana
  environment:
    - GF_SECURITY_ADMIN_PASSWORD=123456
  depends_on:
    - prometheus
  ports:
    - '9092:3000'
  deploy:
    replicas: 1
    placement:
      constraints: [node.role==manager]
  networks:
    - test_net
```

Figure 7 : grafana

On définit le mot de passe : 123456, puis on rentre l'adresse : http://192.168.42.17:9092 puis on rentre le mot de passe.

On arrive ensuite sur cette page :

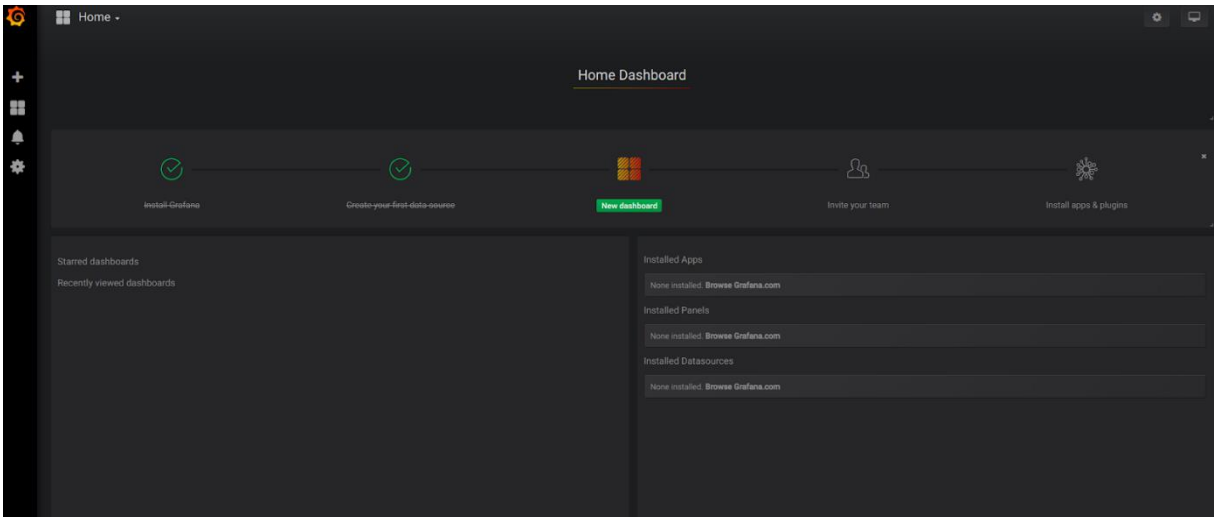

Figure 8 : datasource grafana

Ensuite on récupère les données depuis http://prometheus:9090

Figure 9 : grafana

Pour avoir accès aux données de cadvisor dans prometheus, on ajoute cadvisor :

```
- job_name: 'cadvisor'
  static_configs:
    - targets: ['192.168.42.17:8091']
```

Figure 10 : cadvisor – prometheus

Ensuite on se connecte sur l'adresse http://192.168.42.17:9090 pour regarder vérifier

Targets

All | Unhealthy

cadvisor (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.42.17:8090/metrics	UP	instance="192.168.42.17:8090" job="cadvisor"	7.717s ago	278.2ms	

docker (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.42.17:9323/metrics	UP	instance="192.168.42.17:9323" job="docker"	7.69s ago	69.05ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.42.17:9090/metrics	UP	instance="192.168.42.17:9090" job="prometheus"	2.767s ago	25.68ms	

Figure 11 : target

On se rend ensuite dans grafana, on crée un graphique avec les valeurs « container_cpu_user_seconds_total » pour vérifier :

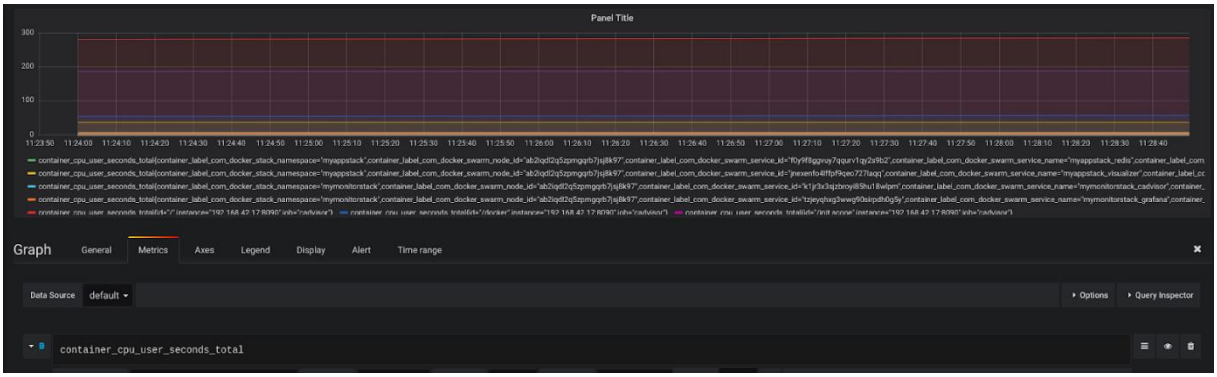

Figure 12 : container_cpu_user_seconds_total

Partie III - Le bazooka

Question 01

Dans un premier temps, après avoir clone le répertoire GitHub, on lance le stack qui permet de déployer tous les services.

```
vagrant@manager:/vagrant/swarmprom$ sudo docker service ls
ID                  NAME                    MODE                REPLICAS            IMAGE
vn1p45cf2u8t        mon_alertmanager        replicated          1/1                 stefanprodan/swarmprom-alertmanager:v0.14.0
mdh2pba4qb3o        mon_caddy               replicated          1/1                 stefanprodan/caddy:latest
kz3jxip63bdc        mon_cadvisor            global              3/3                 google/cadvisor:latest
pzifjedyh5h5        mon_dockerd-exporter    global              3/3                 stefanprodan/caddy:latest
sip71qkhtho9        mon_grafana             replicated          1/1                 stefanprodan/swarmprom-grafana:5.3.4
806odvvowpd3        mon_node-exporter       global              3/3                 stefanprodan/swarmprom-node-exporter:v0.16.0
ljpfqbztfhsr        mon_prometheus          replicated          1/1                 stefanprodan/swarmprom-prometheus:v2.5.0
ujggg3t182o6        mon_unsee               replicated          1/1                 cloudflare/unsee:v0.8.0
gbmkysyyavmj        myappstack_redis        replicated          1/1                 redis:latest
m8oc28s7rib3        myappstack_visualizer   replicated          1/1                 dockersamples/visualizer:stable
znn2a2tokrk9        myappstack_web          replicated          3/3                 localhost:5000/myappgallery:latest
osgdcutvsml9        mypuppeteerstack_puppeteer   replicated     1/3                 localhost:5000/mypuppeteerimage:latest
3lmbydb57nmy        registry                replicated          1/1                 registry:2
vagrant@manager:/vagrant/swarmprom$
```

Figure 13 : service swarmprom

Ensuite on se connecte sur l'adresse http://192.168.42.17:9090 pour regarder vérifier :

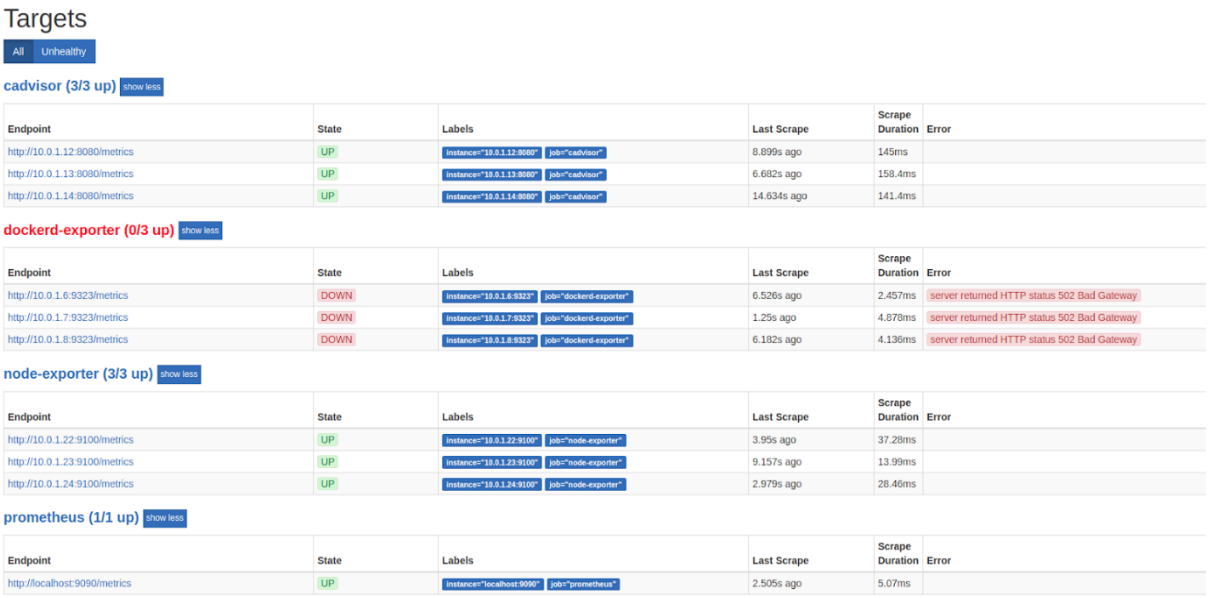

Figure 14 : target

Lorsque l'on se rend sur Grafana on peut analyser plusieurs valeurs. Soit on affiche les statistiques sur les nœuds, soit on affiche les statistiques sur les services.

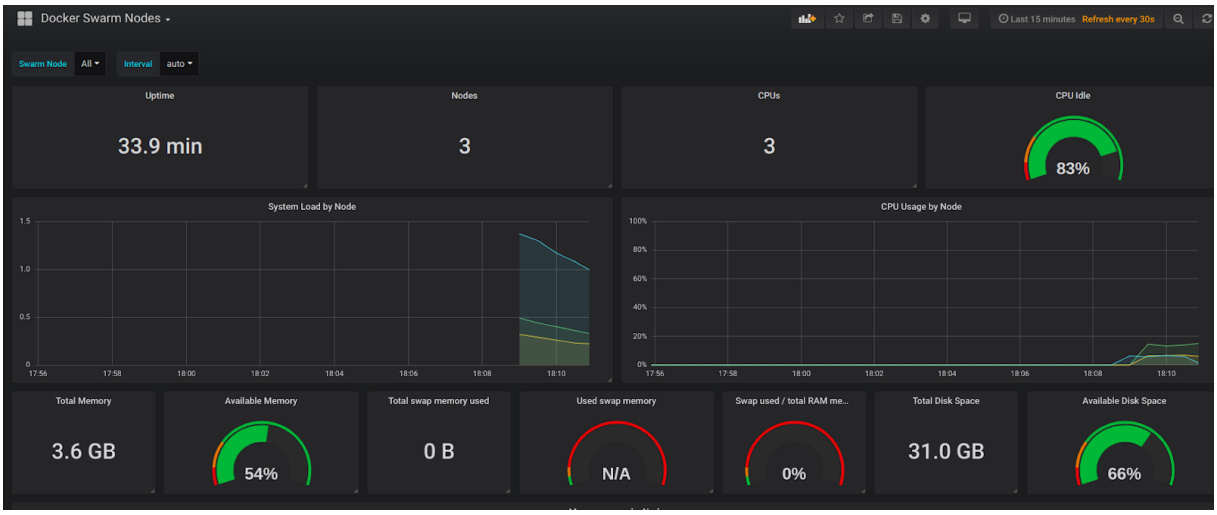

Figure 15 : statistiques sur les nœuds

Ces statistiques sont plutôt correctes puisqu'aucun problème n'est à déclarer.

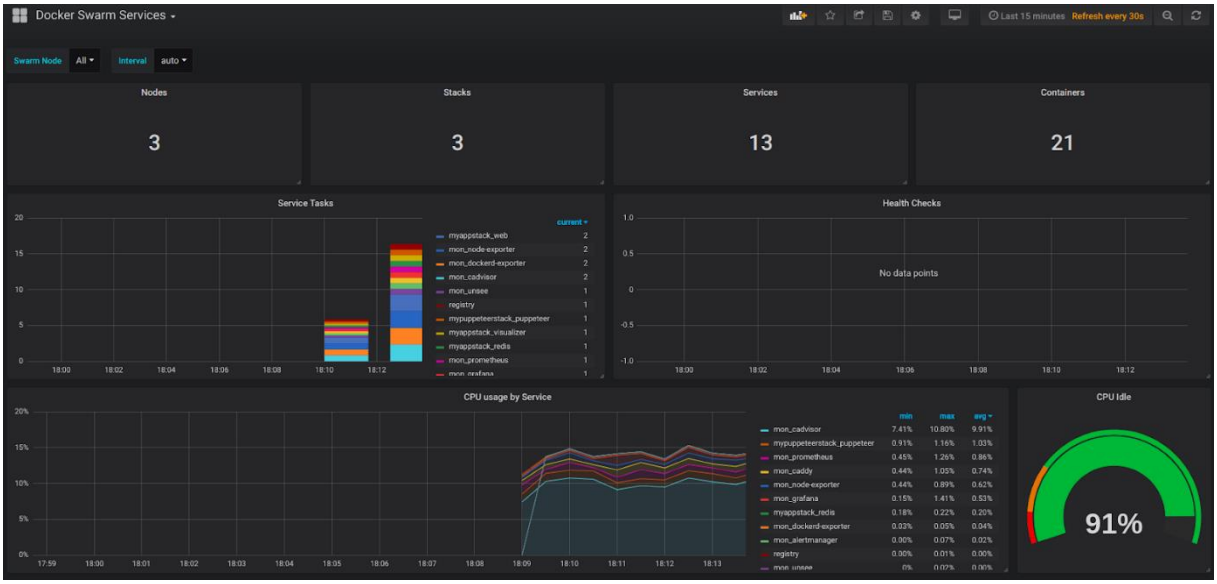

Figure 16 : statistiques sur les services

On peut voir que 13 services sont lancés et les statistiques sont bonnes. Le fonctionnement est normal.

Question 02

Grafana utilise les données de Prometheus. En revanche contrairement à Prometheus l'affichage des graphs est plus joli et plus détaillé.

Question 03

Prometheus se sert des métriques docker, de cadvisor, de node exporter et de dockerd exporter pour avoir ses données et ensuite les afficher dans des graphiques.

Question 04

Le but des alertes est de prévenir l'utilisateur lorsqu'un dysfonctionnement survient sur son installation. Les statistiques du système sont écoutées en permanence et lorsque ces dernières atteignent une valeur limite alors une alerte est envoyée.

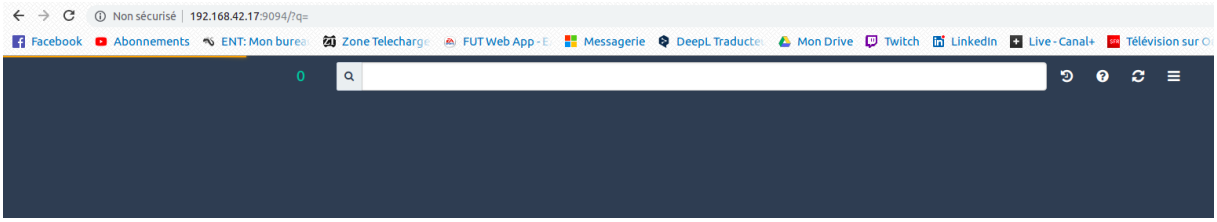

Figure 17 : Pas d'alerte

Lorsque l'on regarde les alertes, avec un seul client, il n'y en a pas.

Si on augmente les requêtes, il y a des alertes qui apparaissent. Il y a des problèmes en termes d'utilisation du CPU.

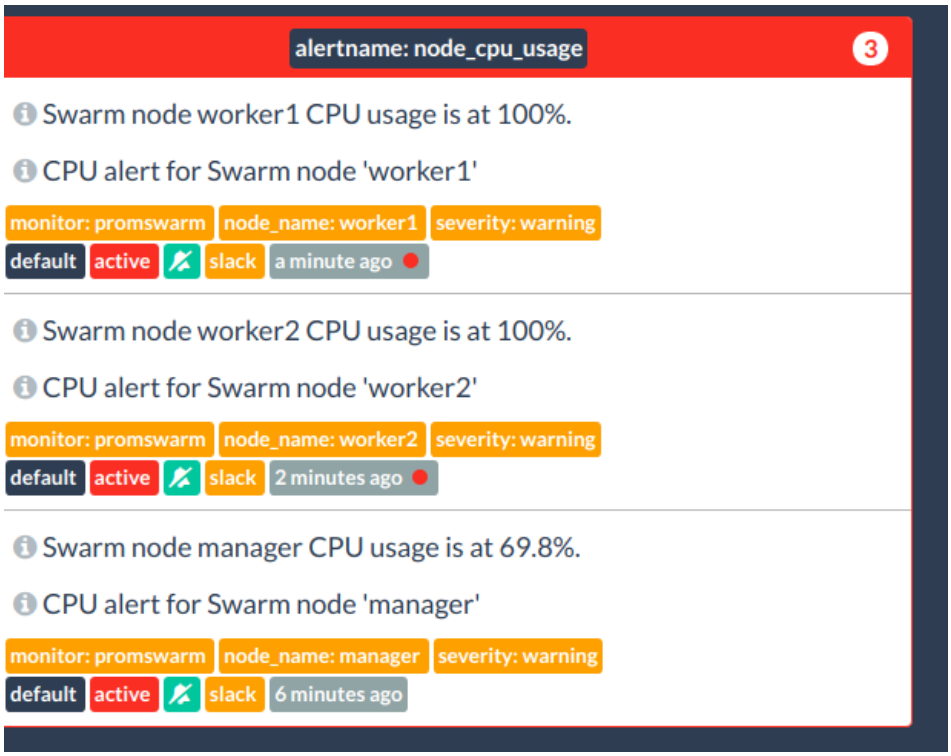

Figure 18 : alert CPU Usage

Conclusion

Dans ce TP nous avons visualisé les données de notre applications grâce à Grafana. Par la suite, pour mieux comprendre le comportement de notre application, nous simulerons un grand nombre de clients pour mieux comprendre le comportement.