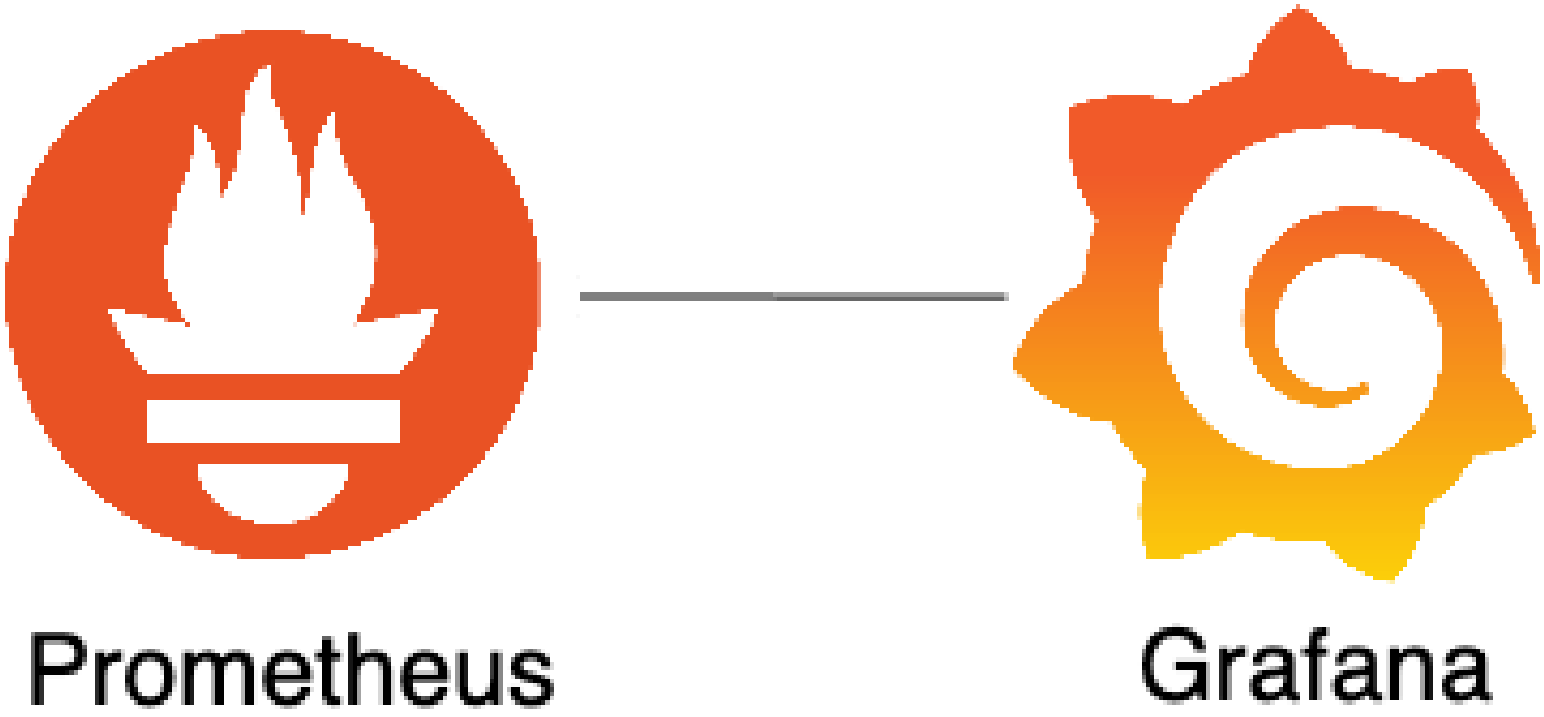
Continuous Monitoring : Grafana + Prometheus



Plan du cours

- Grafana
- Prometheus

- Architecture : Grafana + Prometheus

- Mise en place du Monitoring de l'outil Jenkins, avec Prometheus + Grafana

Grafana

- **Grafana** est un logiciel open source de visualisation et d'analyse.
- Il permet d'interroger, de visualiser, d'alerter et d'explorer des métriques, quel que soit l'endroit où elles sont stockées (donc quel que soit la data source : Prometheus, ElasticSearch, SQL Server, …).
- Il fournit des outils pour transformer des données de base de données en graphiques et visualisations.

Grafana

- Le projet **Grafana** a été lancé par Torkel Odegaard en 2014 et est devenu au cours des dernières années l'un des projets open source les plus populaires sur GitHub.

- " Le but de la création de Grafana était de rendre les choses que je trouvais difficiles, et que d'autres personnes trouvaient difficiles, plus faciles et accessibles. De cette façon, d'avantage de personnes pourraient commencer à instrumenter leurs applications et à créer des tableaux de bord par elles-mêmes. Rendre les outils d'observabilité accessibles à tout le monde dans une organisation, et pas seulement à la seule personne chargée des opérations."

Torkel Odegaard

Grafana

- Projet Open Source.
- Projet utilisé par des millions d'utilisateurs.
- Plus de 750k installation.
- 64k Github stars.
- Projet Grafana OSS (Open Source Softaware).

Prometheus

- **Prometheus** est outil de stockage de métriques (CPU, RAM, …) en temps réel dans une base de données de séries chronologiques (base optimisée pour stockée les couple temps/valeur).
- Il permet aussi de gérer les alertes (mail, ….).
- Prometheus est conçu pour surveiller des cibles: Serveurs / Bases de données / Machines virtuelles / Conteneurs …

Prometheus

Grafana - Prometheus

- **Nous allons surveiller Jenkins avec Prometheus et afficher les métriques avec Grafana.**

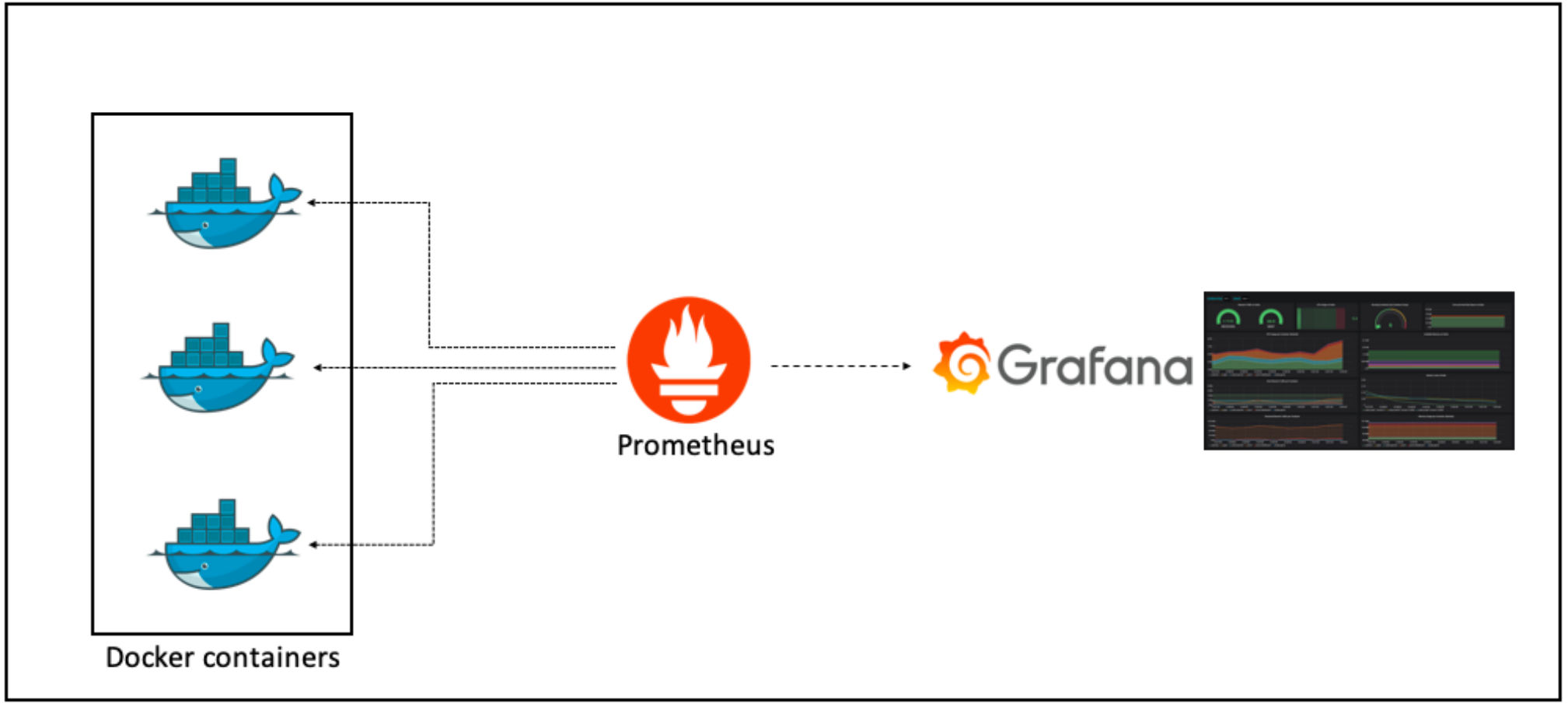
- Vous pouvez surveiller d'autres composants comme le conteneur Spring Boot ou le conteneur de base de données de votre application finale.

Grafana – Prometheus

- Dashboard Grafana : Surveillance de Jenkins par Prometheus.

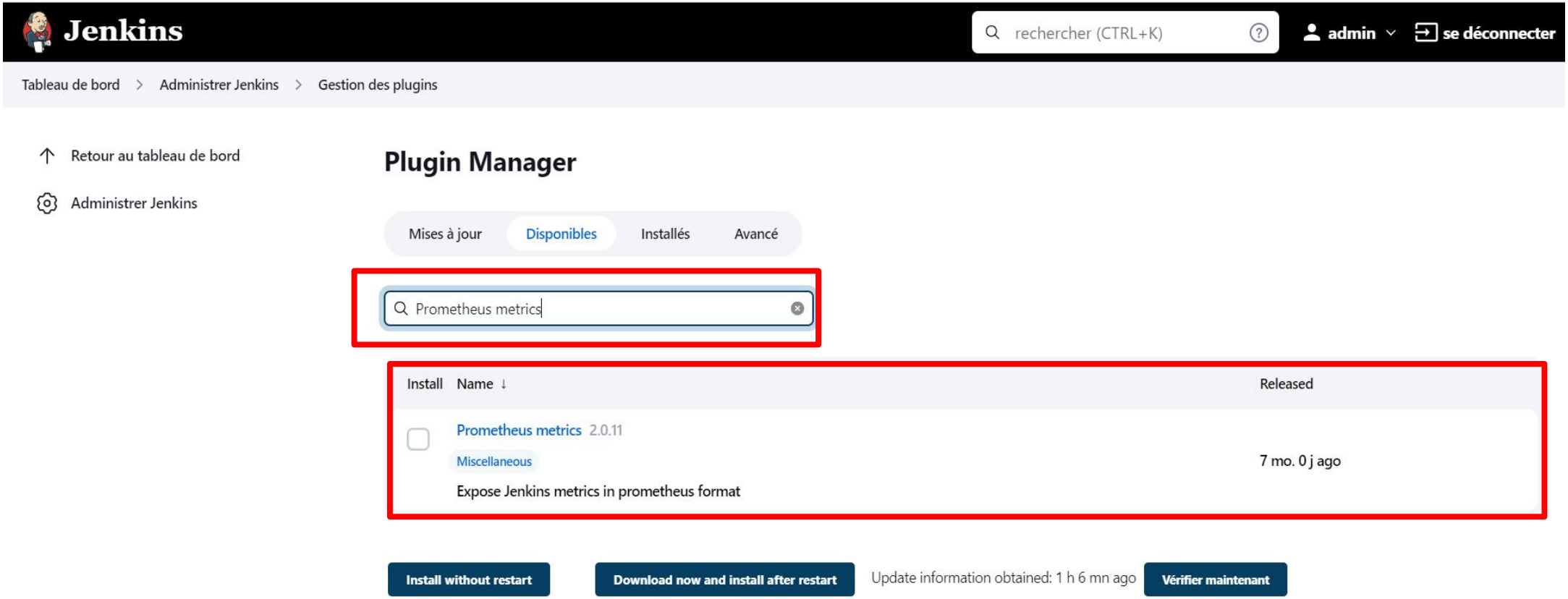
Architecture

- Vous pouvez surveiller les conteneurs de votre application finale (**Backend Spring Boot** + MySQL).
- **Prometheus récupère en temps réel les métriques. Grafana affiche ces métriques.**

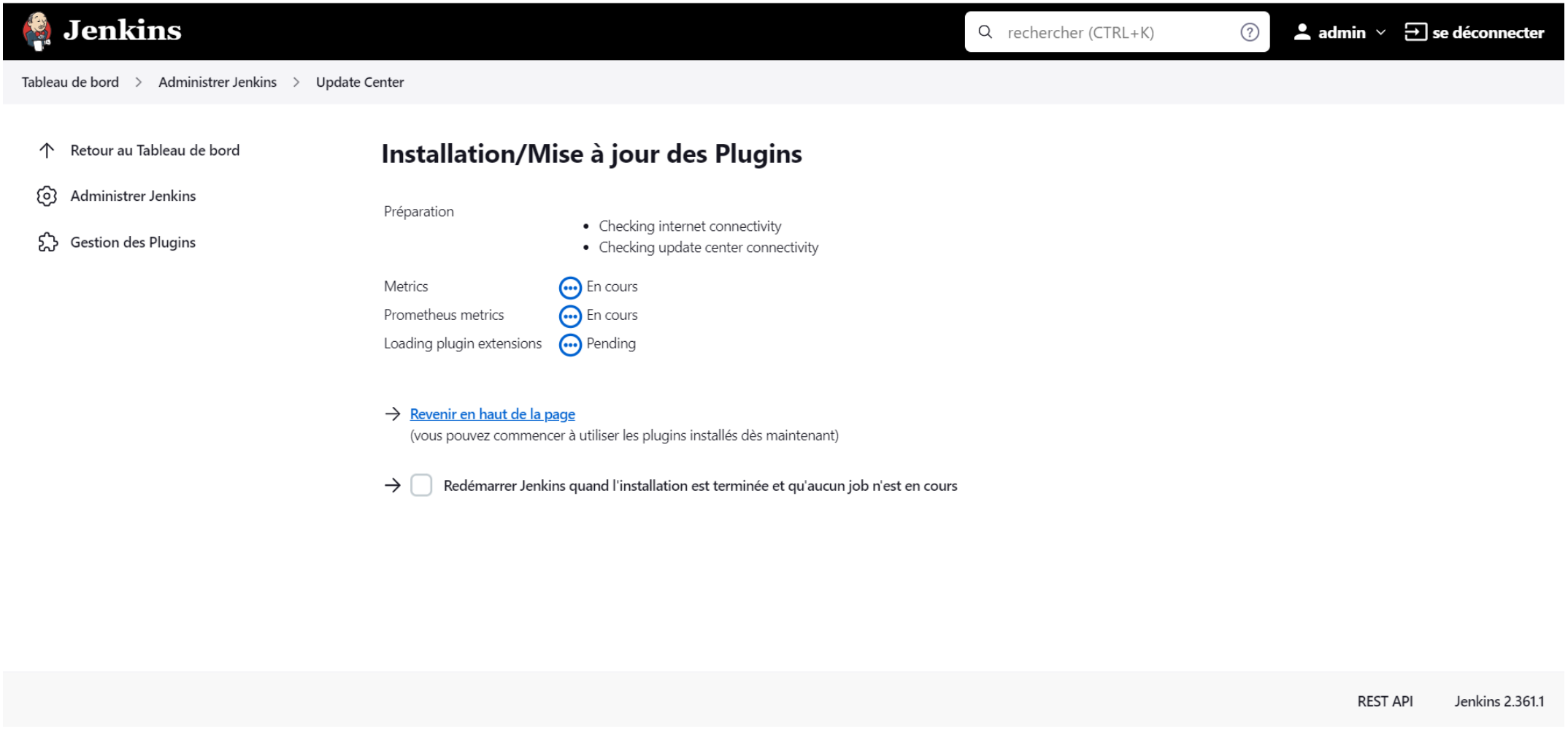


Place à la pratique - Jenkins

- Aller dans "Administrer Jenkins" puis "Gestion des plugins" et installer "Prometheus metrics". Ce plugin permettra **d'exposer** les métriques pour Promotheus sur le endpoint **/prometheus**



Place à la pratique - Jenkins



→ Jenkins doit être redémarré pour que la mise à jour soit effective.

Place à la pratique – Prometheus

- Création d'un conteneur Docker Prometheus.
- **sudo chmod 666 /var/run/docker.sock**
- **docker run -d --name prometheus -p 9090:9090 prom/prometheus**

```
[root@localhost vagrant]# docker run -d --name prometheus -p 9090:9090 prom/prometheus
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
50783e0dfb64: Pull complete
daafb1bca260: Pull complete
bafa8e139cea: Pull complete
0d2e6df8577f: Pull complete
e3d4e14499bc: Pull complete
a3f71f7c721c: Pull complete
aca108eacfe0: Pull complete
b6aee8ea9d2f: Pull complete
950d9a06ee14: Pull complete
d009d09c576e: Pull complete
50100a62d658: Pull complete
34487f1a8146: Pull complete
Digest: sha256:aa1687dd552ed98df598cc0fed2effbc62a0f05236bc2253c65520ddd4f2afce
Status: Downloaded newer image for prom/prometheus:latest
fd0a48c808ac991fb0f4fdcbad3cdb381b3c595846b24facafe1088ddec6e542
```

Place à la pratique – Prometheus

docker exec -it prometheus sh

Puis dans le conteneur prometheus (8080 est le port pour accéder à Jenkins) :

Ne pas faire un copier-coller de la commande ci-dessous, récupérer la commande complète du fichier prometheus.yml.txt sur le **Drive** .

```
tee -a /etc/prometheus/prometheus.yml <<EOF
  - job_name: jenkins
    metrics_path: /prometheus
    static_configs:
      - targets: ['172.17.0.1:8080']
EOF
```

Vérifier le contenu du ficher créer (***cat /etc/prometheus/prometheus.yml*** puis ***exit***

Enfin : ***docker restart prometheus***

Voir capture page suivante pour plus de détails :

Place à la pratique – Prometheus

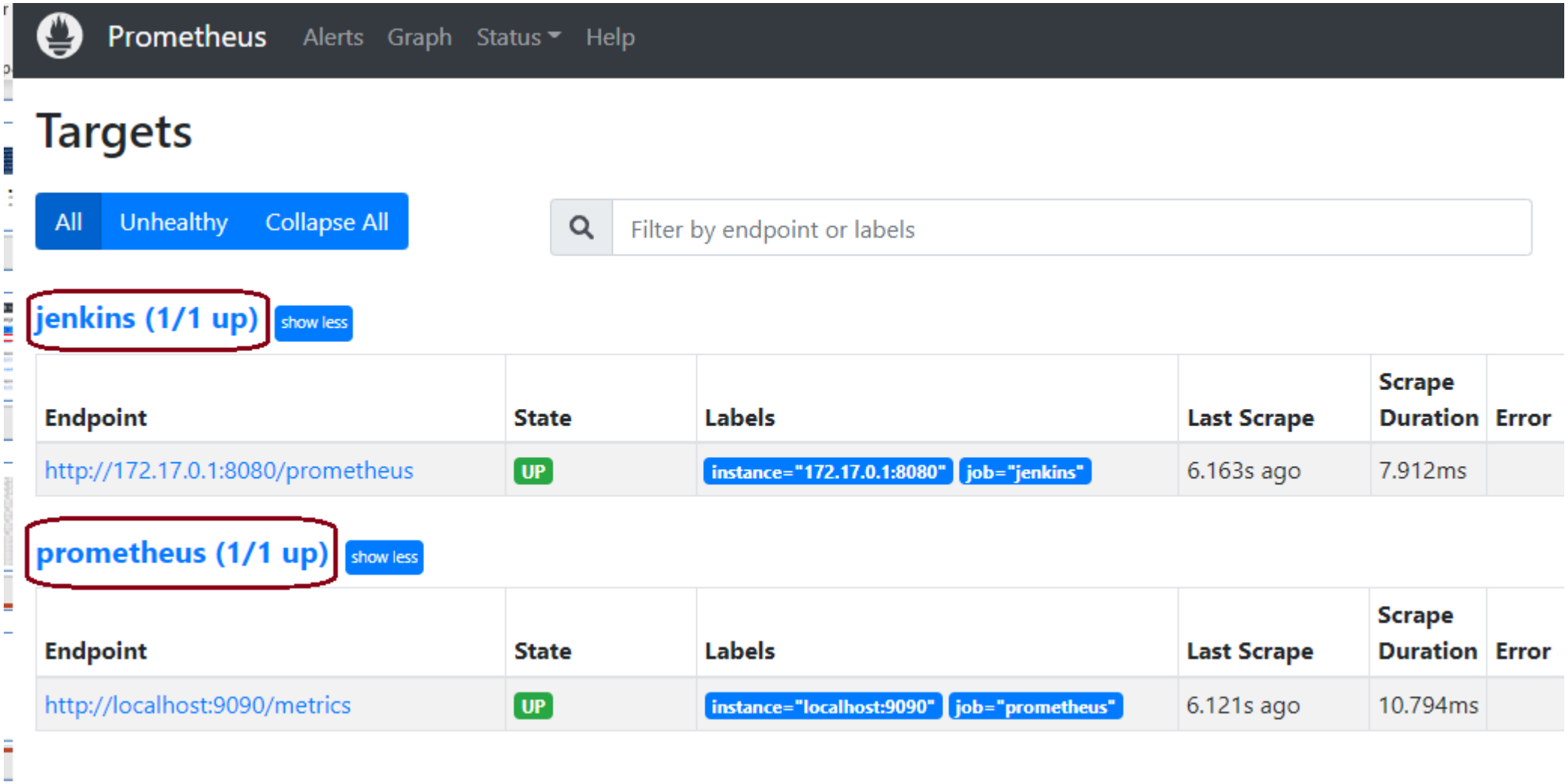
```
[vagrant@localhost ~]$ docker exec -it prometheus sh
/prometheus $ tee -a /etc/prometheus/prometheus.yml <<EOF
>   - job_name: jenkins
>     metrics_path: /prometheus
>     static_configs:
>       - targets: ['172.17.0.1:8080']
> EOF
  - job_name: jenkins
    metrics_path: /prometheus
    static_configs:
      - targets: ['172.17.0.1:8080']
/prometheus $ cat /etc/prometheus/prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every
  evaluation_interval: 15s # Evaluate rules every 15 secon
  # scrape_timeout is set to the global default (10s).
```

Place à la pratique – Prometheus

```
/prometheus $ exit
[vagrant@localhost ~]$ docker restart prometheus
prometheus
[vagrant@localhost ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
130e94221f96        prom/prometheus     "/bin/prometheus -..."   6 minutes ago       Up 2 seconds
```

Place à la pratique – Prometheus

http://@IP_VM:9090/targets



Place à la pratique – Prometheus

http://@IP_VM:9090/metrics

```
# HELP go_memstats_stack_sys_bytes Number of bytes obtained from system for stack allocator.
# TYPE go_memstats_stack_sys_bytes gauge
go_memstats_stack_sys_bytes 655360
# HELP go_memstats_sys_bytes Number of bytes obtained from system.
# TYPE go_memstats_sys_bytes gauge
go_memstats_sys_bytes 3.3113096e+07
# HELP go_threads Number of OS threads created.
# TYPE go_threads gauge
go_threads 7
# HELP net_conntrack_dialer_conn_attempted_total Total number of connections attempted by the given dialer a given na
# TYPE net_conntrack_dialer_conn_attempted_total counter
net_conntrack_dialer_conn_attempted_total{dialer_name="alertmanager"} 0
net_conntrack_dialer_conn_attempted_total{dialer_name="default"} 0
net_conntrack_dialer_conn_attempted_total{dialer_name="jenkins"} 11
net_conntrack_dialer_conn_attempted_total{dialer_name="prometheus"} 1
# HELP net_conntrack_dialer_conn_closed_total Total number of connections closed which originated from the dialer of
# TYPE net_conntrack_dialer_conn_closed_total counter
net_conntrack_dialer_conn_closed_total{dialer_name="alertmanager"} 0
net_conntrack_dialer_conn_closed_total{dialer_name="default"} 0
net_conntrack_dialer_conn_closed_total{dialer_name="jenkins"} 11
net_conntrack_dialer_conn_closed_total{dialer_name="prometheus"} 0
# HELP net_conntrack_dialer_conn_established_total Total number of connections successfully established by the given
# TYPE net_conntrack_dialer_conn_established_total counter
net_conntrack_dialer_conn_established_total{dialer_name="alertmanager"} 0
net_conntrack_dialer_conn_established_total{dialer_name="default"} 0
net_conntrack_dialer_conn_established_total{dialer_name="jenkins"} 11
net_conntrack_dialer_conn_established_total{dialer_name="prometheus"} 1
# HELP net_conntrack_dialer_conn_failed_total Total number of connections failed to dial by the dialer a given name.
# TYPE net_conntrack_dialer_conn_failed_total counter
net_conntrack_dialer_conn_failed_total{dialer_name="alertmanager",reason="refused"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="alertmanager",reason="resolution"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="alertmanager",reason="timeout"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="alertmanager",reason="unknown"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="default",reason="refused"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="default",reason="resolution"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="default",reason="timeout"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="default",reason="unknown"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="jenkins",reason="refused"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="jenkins",reason="resolution"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="jenkins",reason="timeout"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="jenkins",reason="unknown"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="prometheus",reason="refused"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="prometheus",reason="resolution"} 0
net_conntrack_dialer_conn_failed_total{dialer_name="prometheus",reason="timeout"} 0
```

Place à la pratique – Prometheus

http://@IP_VM:8080/prometheus

```
# HELP vm_memory_total_used_window_1h Generated from Dropwizard metric import
(metric=vm.memory.total.used.window.1h, type=jenkins.metrics.util.AutoSamplingHistogram)
# TYPE vm_memory_total_used_window_1h summary
vm_memory_total_used_window_1h{quantile="0.5",} 4.88444384E8
vm_memory_total_used_window_1h{quantile="0.75",} 5.58357134E8
vm_memory_total_used_window_1h{quantile="0.95",} 7.606846732E8
vm_memory_total_used_window_1h{quantile="0.98",} 7.7041319488E8
vm_memory_total_used_window_1h{quantile="0.99",} 7.7043291664E8
vm_memory_total_used_window_1h{quantile="0.999",} 7.72536896E8
vm_memory_total_used_window_1h_count 264.0
# HELP jenkins_queue_blocked_history Generated from Dropwizard metric import
(metric=jenkins.queue.blocked.history, type=jenkins.metrics.util.AutoSamplingHistogram)
# TYPE jenkins_queue_blocked_history summary
jenkins_queue_blocked_history{quantile="0.5",} 0.0
jenkins_queue_blocked_history{quantile="0.75",} 0.0
jenkins_queue_blocked_history{quantile="0.95",} 0.0
jenkins_queue_blocked_history{quantile="0.98",} 0.0
jenkins_queue_blocked_history{quantile="0.99",} 0.0
jenkins_queue_blocked_history{quantile="0.999",} 0.0
jenkins_queue_blocked_history_count 264.0
# HELP vm_memory_pools_Metaspace_used_window_1h Generated from Dropwizard metric import
(metric=vm.memory.pools.Metaspace.used.window.1h, type=jenkins.metrics.util.AutoSamplingHistogram)
# TYPE vm_memory_pools_Metaspace_used_window_1h summary
vm_memory_pools_Metaspace_used_window_1h{quantile="0.5",} 8.6690972E7
vm_memory_pools_Metaspace_used_window_1h{quantile="0.75",} 8.6789596E7
vm_memory_pools_Metaspace_used_window_1h{quantile="0.95",} 8.6840736E7
vm_memory_pools_Metaspace_used_window_1h{quantile="0.98",} 8.692560896E7
vm_memory_pools_Metaspace_used_window_1h{quantile="0.99",} 8.696774368E7
vm_memory_pools_Metaspace_used_window_1h{quantile="0.999",} 8.697656E7
vm_memory_pools_Metaspace_used_window_1h_count 264.0
```

Place à la pratique - Prometheus - Grafana

C'est très difficile d'exploiter et de comprendre ces données textuelles que **Prometheus** nous met à disposition.

Nous allons donc utiliser l'outil **Grafana** qui va récupérer ces données, les mettre en forme et les afficher graphiquement :

Place à la pratique – Grafana

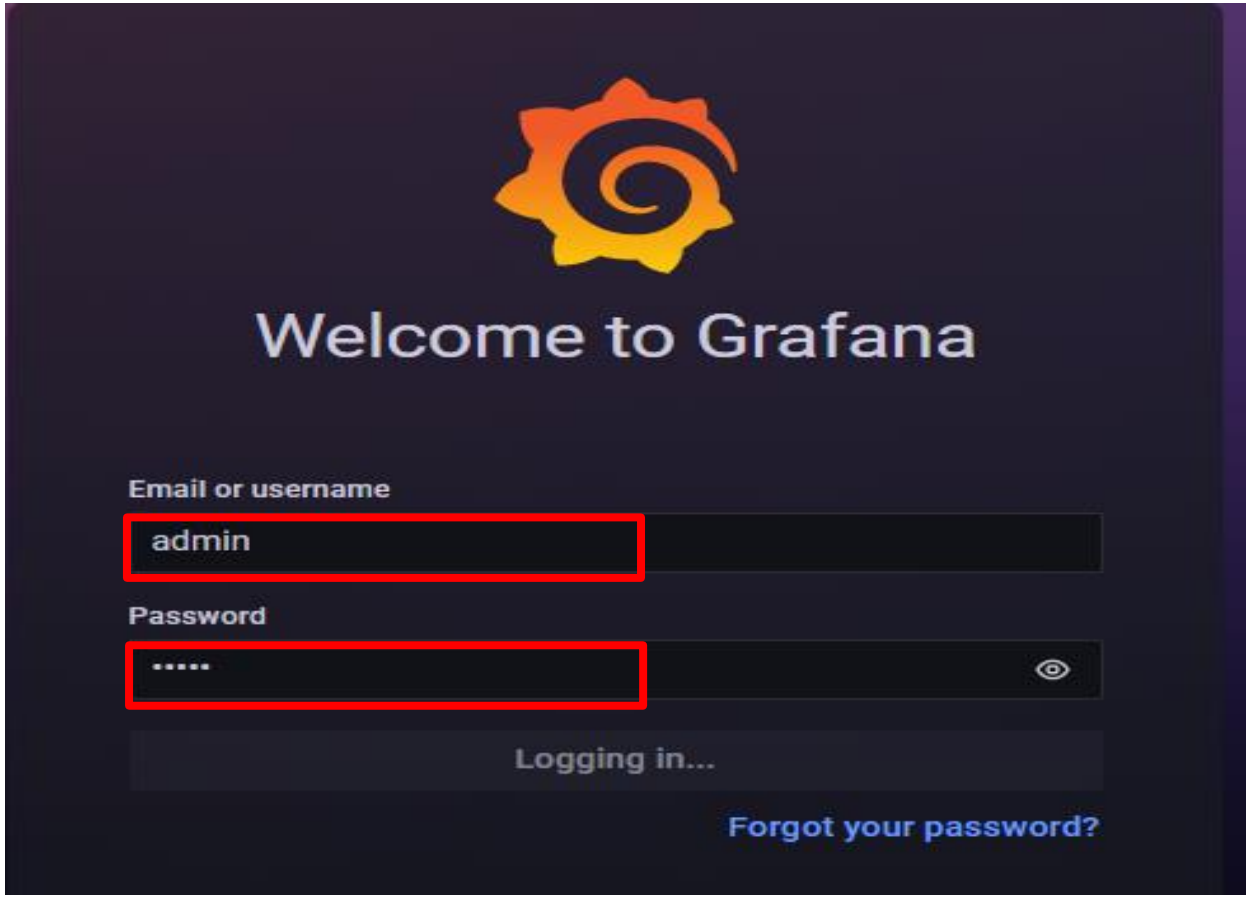
- Création d'un conteneur Docker Grafana.

docker run -d --name grafana -p 3000:3000 grafana/grafana

```
[root@localhost vagrant]# docker run -d --name grafana -p 3000:3000 grafana/grafana
Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
9621f1afde84: Pull complete
8a979fdf9b56: Pull complete
1dfb2bc044fd: Pull complete
83f5d14e4bf0: Pull complete
b6745f3b63b1: Pull complete
c57092a7aaa6: Pull complete
94139446967c: Pull complete
3406d8746525: Pull complete
51ac91216bc8: Pull complete
Digest: sha256:3755790fae9130975b0a778ea7c61e54627550541cf90f0aa5f11fa8936468c9
Status: Downloaded newer image for grafana/grafana:latest
4dda86304b533cc3af4fb4f955d33c0d0581538328777f33f6a91fae009dcaadb
```

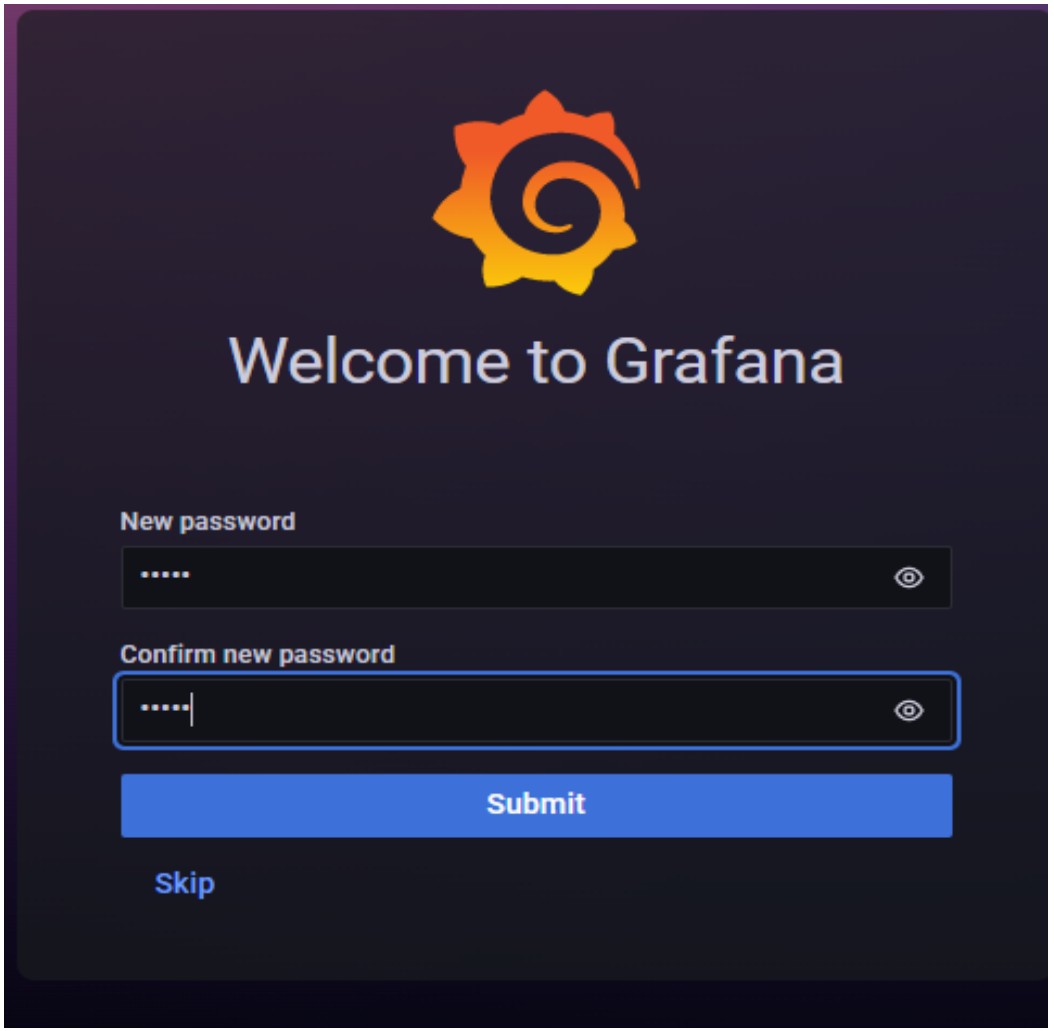
Place à la pratique – Grafana

- http://@IP_VM:3000
- L'utilisateur et le mot de passe par défaut sont "admin/admin".



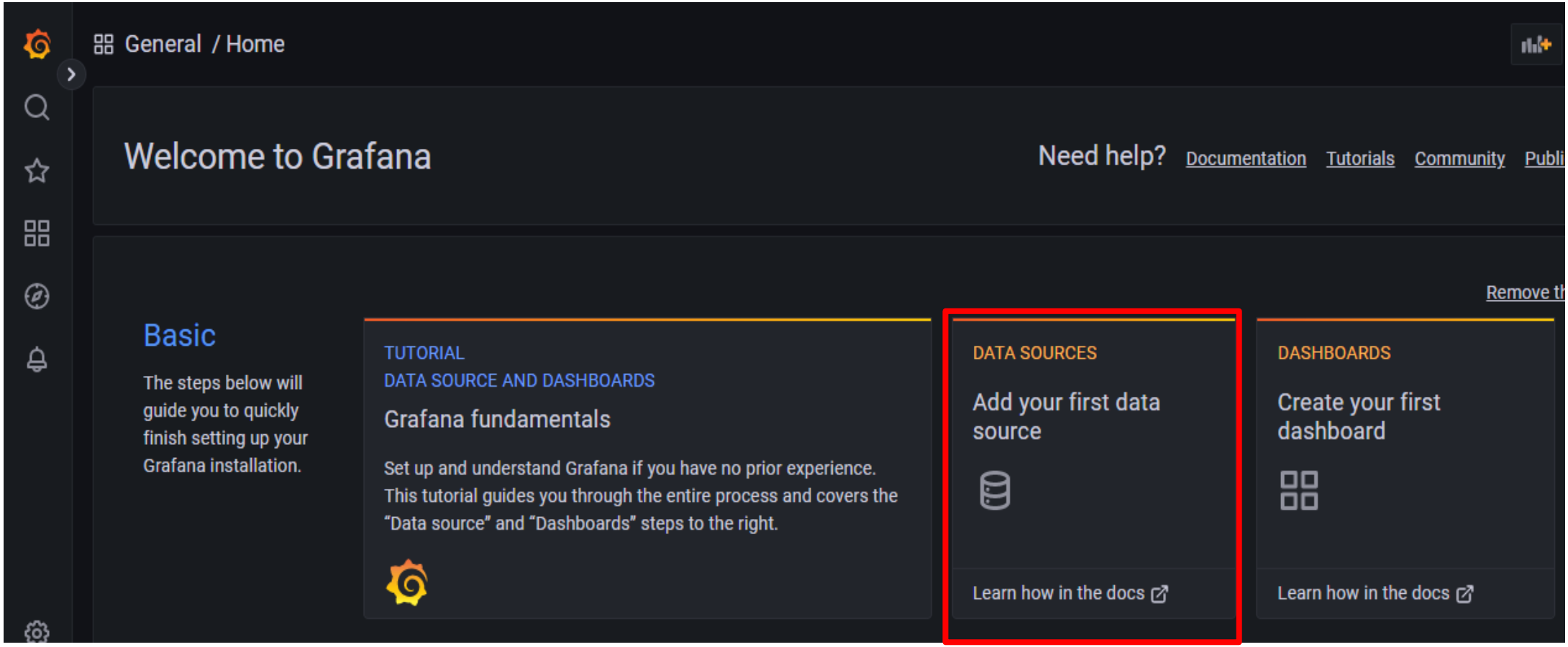
Place à la pratique – Grafana

- Changer le mot de passe
  (par exemple login : **admin** password: **grafana**)

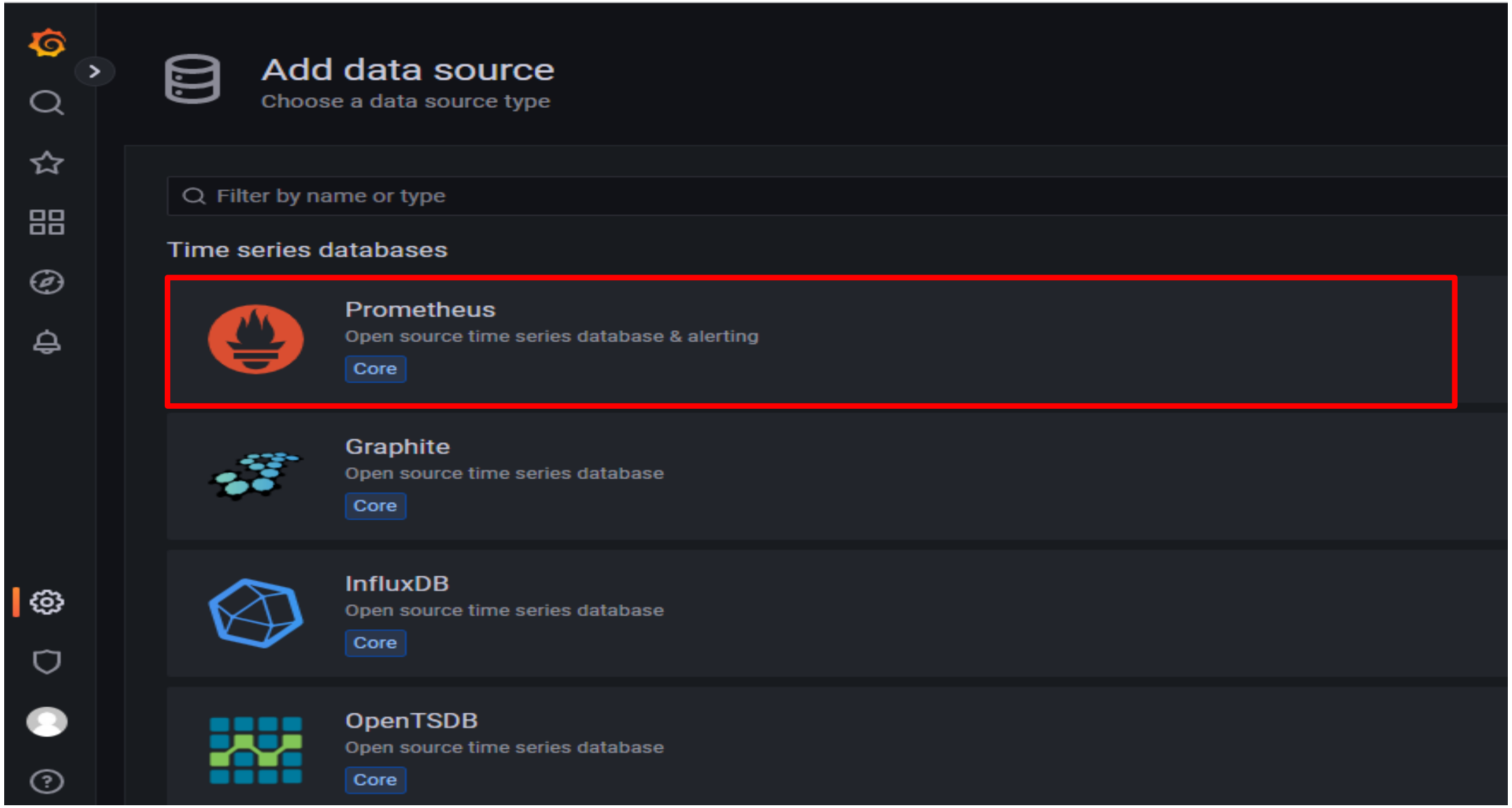


Place à la pratique – Grafana

- Ajouter la source des données ➔ Prometheus

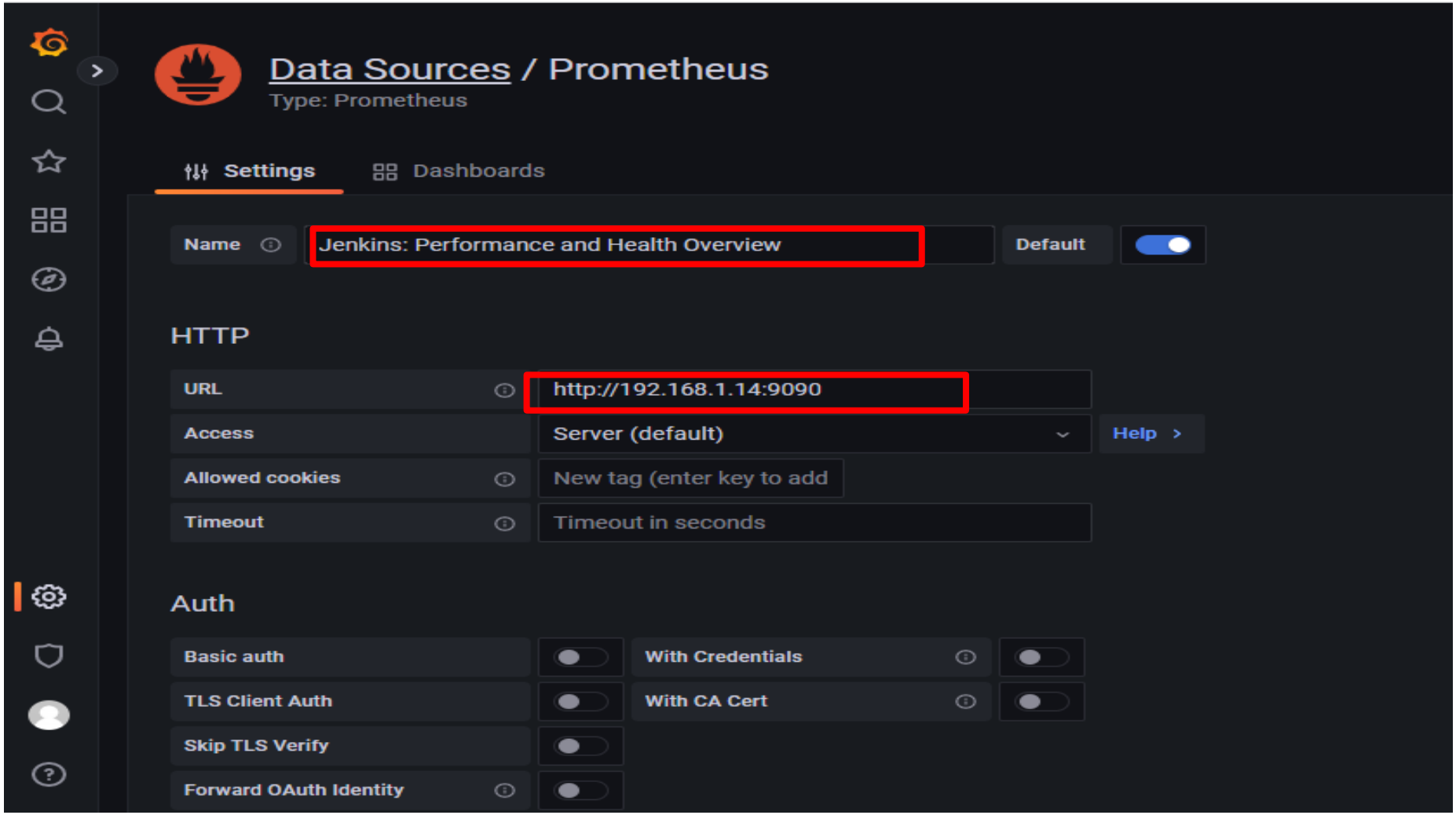


Place à la pratique – Grafana



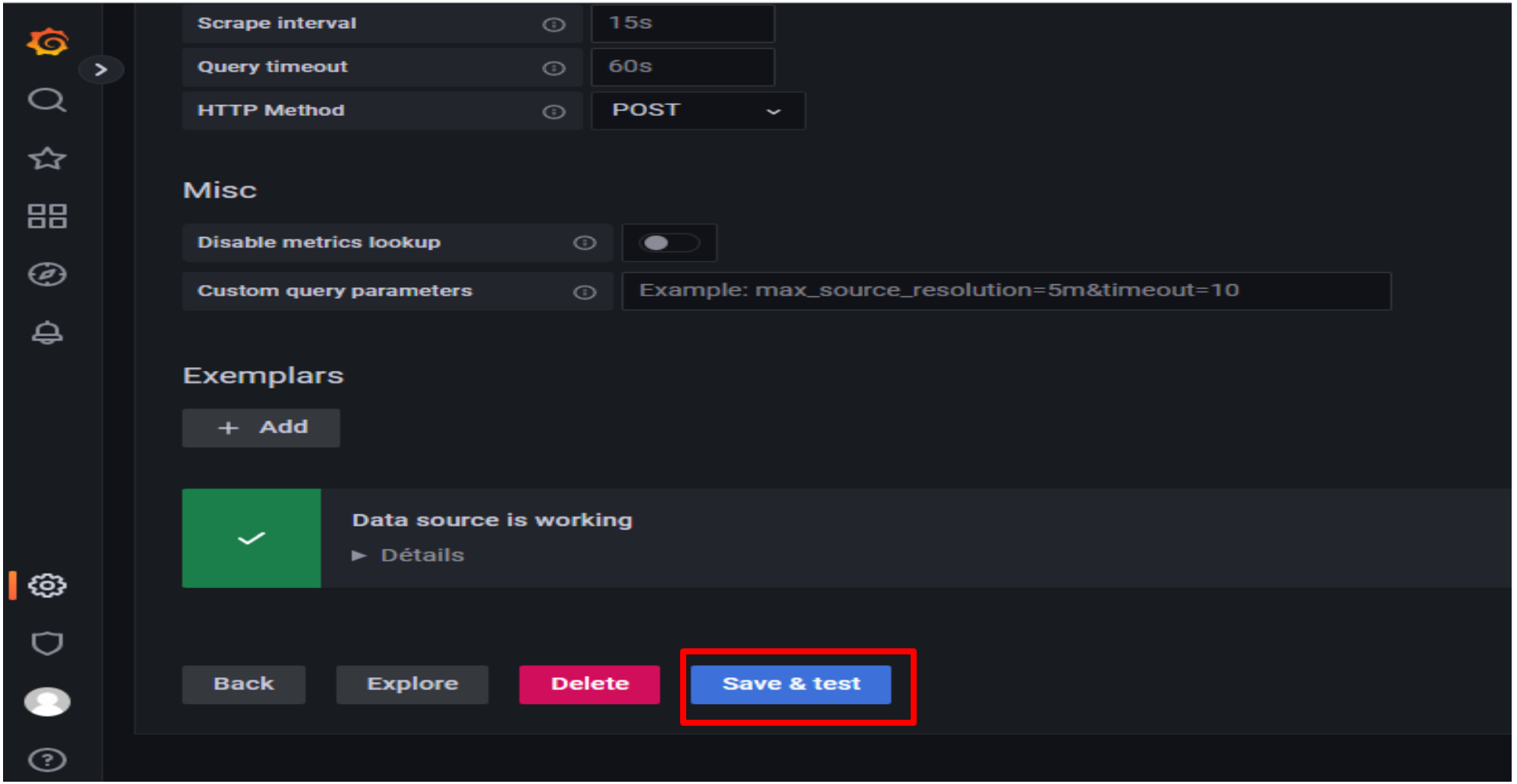
Place à la pratique – Grafana

- Récupérer l'adresse IP du conteneur docker Grafana.



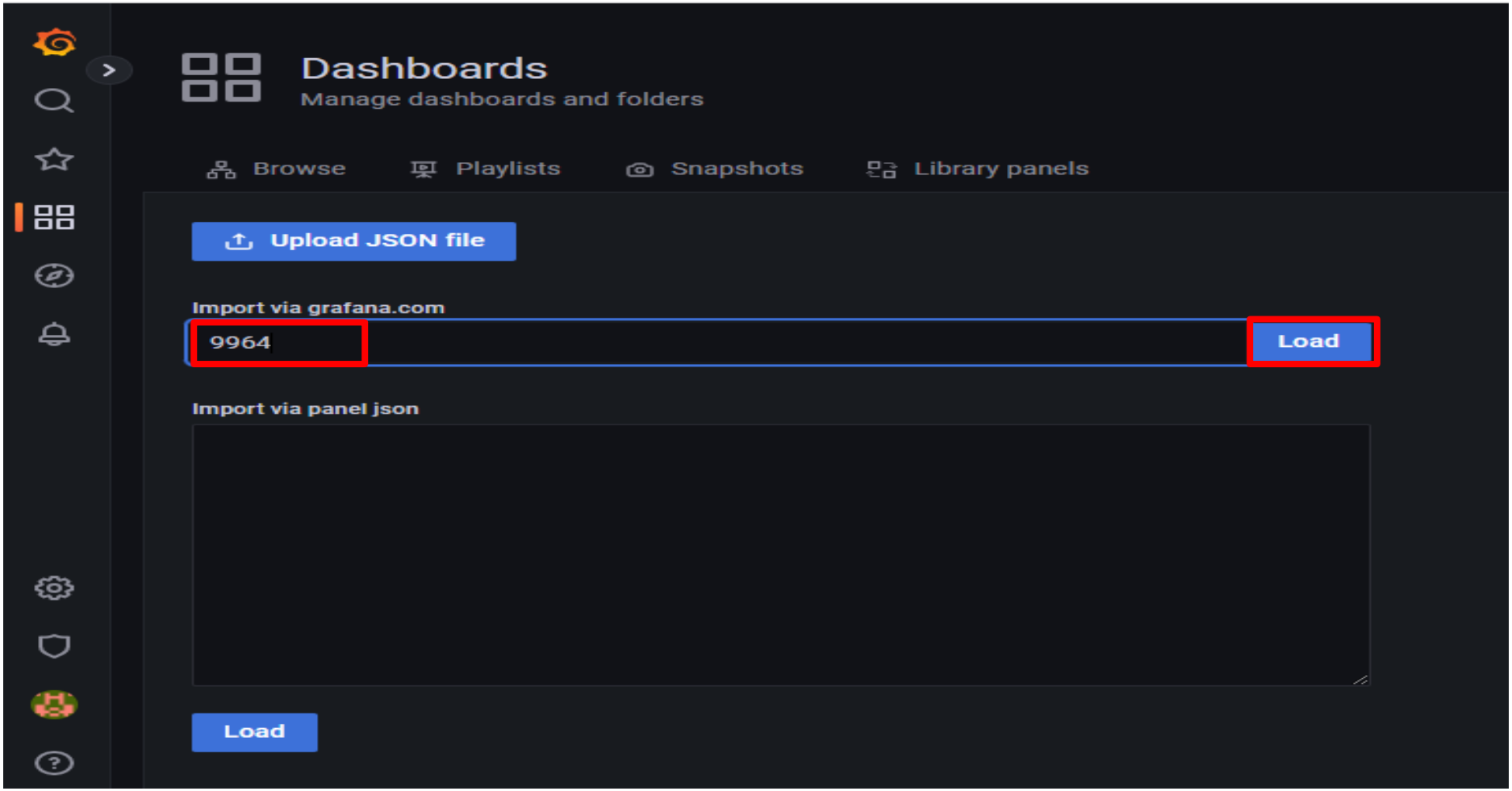
Place à la pratique – Grafana

- Cliquer Save & Test.

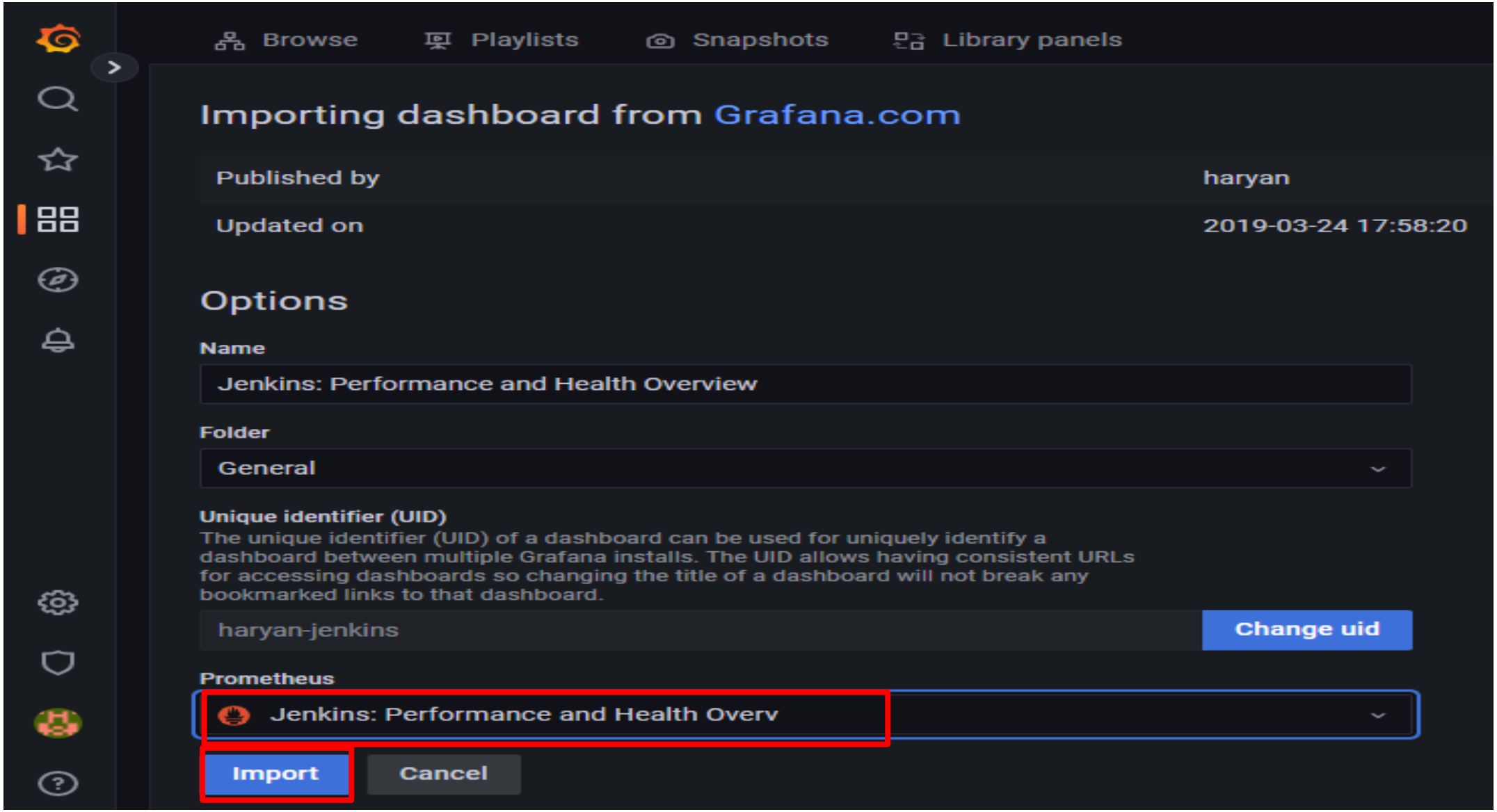


Place à la pratique – Grafana

- Accéder à **http://@IP_VM:3000/dashboard/import** et utiliser 9964 l'identifiant d'un template d'un dashboard.



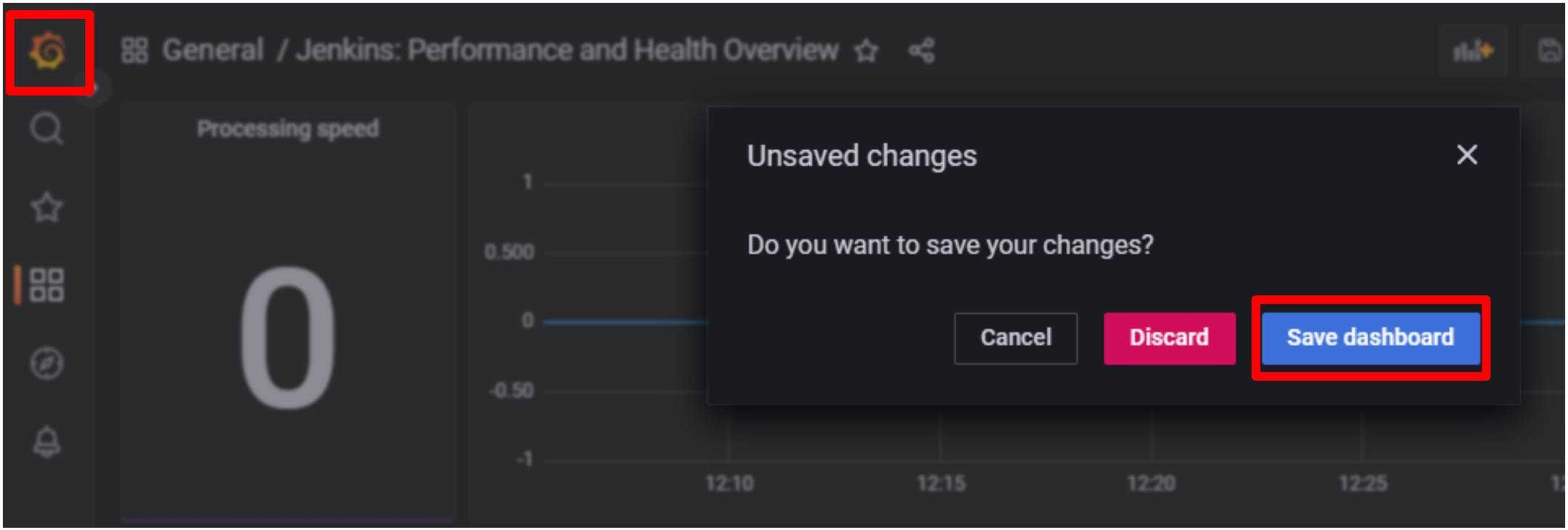
Place à la pratique – Grafana



Place à la pratique – Grafana

Place à la pratique – Grafana

- Sauvegarder le dashboard en cliquant sur le logo de Grafana



TP

1- Installer et Configurer **Prometheus** (Voir étapes ci-dessus)

2- Installer et Configurer **Grafana** (Voir étapes ci-dessus)

3- Surveiller (Monitorez) l'outil **Jenkins** (Voir étapes ci-dessus)

4- Inspirez-vous de la question 3 pour Surveiller votre conteneur de l'application **Backend tp-foyer**.

Continuous Monitoring : Grafana + Prometheus