Chapitre 3
Découverte de Prometheus

1. Objectifs du chapitre et prérequis

1.1 Contexte et prérequis

Ce chapitre est une première découverte durant laquelle l'utilisateur va installer et lancer le moteur Prometheus. L'exporteur système ainsi que la configuration nécessaire seront abordés pour intégrer un point de collecte dans Prometheus.

1.2 Fichiers téléchargeables

Vous pouvez récupérer les exemples sur le dépôt GitHub suivant :
https://github.com/EditionsENI/prometheus-grafana

Vous pouvez également récupérer ces fichiers dans l'archive **chapitre-03.tar.gz** sur la page du livre sur le site des Éditions ENI.

2. Lancement de Prometheus sous Linux

2.1 Contexte

Le logiciel Prometheus est disponible sur de nombreuses plateformes. Les binaires sont disponibles sur Linux, Windows ou Mac mais il est également possible de le démarrer à l'aide de containers Docker.

Par la suite, les exemples s'appuieront sur l'utilisation d'une machine Linux Ubuntu 24.04 avec une architecture AMD64. Ces instructions sont très facilement transposables aux autres systèmes.

2.2 Installation de Prometheus

L'installation va se faire en respectant les étapes suivantes :

– Téléchargement d'une archive de Prometheus compatible avec son système.

– Décompression de l'archive dans un répertoire du système.

– Lancement de Prometheus.

Pour la première étape, rendez-vous à l'adresse https://prometheus.io/download/ et téléchargez la dernière version disponible (3.2.1 au moment de l'écriture de ce livre).

En plus de cette version courante, le projet Prometheus met à disposition une version LTS (*Long Term Support*, qu'on peut traduire par "support à long terme"). Cette dernière est utile dans le cas où la stabilité serait déterminante. Cette version s'appuie actuellement sur la version 2.53 avec la révision 4 (soit la version 2.53.4).

Cette opération peut s'effectuer en ligne de commande à l'aide de la commande `wget` suivie de l'archive à télécharger. Voici ci-dessous un exemple avec la dernière version disponible :

```
$ wget
https://github.com/prometheus/prometheus/releases/download/
v3.2.1/prometheus-3.2.1.linux-amd64.tar.gz
```

La commande récupère un fichier prometheus-3.2.1.linux-amd64.tar.gz. La décompression se fait ensuite à l'aide de la commande `tar` et de l'option `xfvz` suivie de l'archive à décompresser.

Voici ci-dessous la commande complète :

```
$ tar xfvz prometheus-3.2.1.linux-amd64.tar.gz
```

La commande doit alors renvoyer le résultat suivant :

```
prometheus-3.2.1.linux-amd64/
prometheus-3.2.1.linux-amd64/prometheus.yml
prometheus-3.2.1.linux-amd64/LICENSE
prometheus-3.2.1.linux-amd64/NOTICE
prometheus-3.2.1.linux-amd64/prometheus
prometheus-3.2.1.linux-amd64/promtool
```

Cette liste est relativement restreinte et se compose des éléments suivants :

– Fichier `NOTICE` : fichier texte contenant une liste de composants utilisés par Prometheus.
– Fichier `LICENSE` : fichier contenant la licence de Prometheus (Apache License).
– Fichier `prometheus.yml` : configuration par défaut pour Prometheus.
– Fichier `promtool` : boîte à outils pour Prometheus (vérification de configuration et d'administration).
– Fichier `prometheus` : binaire d'exécution de Prometheus.

Remarque

Prometheus est écrit en langage Go. Cette caractéristique explique l'absence de librairie partagée (fichier lib.so sous Linux). En effet, ce langage produit par défaut des binaires sans dépendances externes. L'avantage étant que les programmes sont indépendants au prix d'une augmentation de la taille du binaire (144 Mo pour le binaire de Prometheus en version 3.2.1).*

2.3 Configuration de Prometheus

2.3.1 Présentation de la configuration par défaut

Le binaire de Prometheus est prêt avec une configuration par défaut. Cette dernière prend la forme d'un fichier YAML. Afin de consulter le contenu du fichier `prometheus.yml` par défaut, lancez la commande `cat` suivie du nom de fichier :

```
$ cat prometheus-3.2.1.linux-amd64/prometheus.yml
```

Voici ci-dessous un extrait de ce fichier :

```
# my global config
global:
  scrape_interval:     15s
  evaluation_interval: 15s
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093

rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090']
```

2.3.2 Test de la configuration

Lorsque la configuration doit être modifiée, il est recommandé de la tester à froid avant de l'appliquer au moteur Prometheus. Pour cela, l'utilisateur peut faire appel à la commande `promtool` suivie des options suivantes :

– Les mots-clés `check config`.

– Le chemin vers le fichier de configuration.

Voici ci-dessous un exemple de lancement pour tester le fichier `prometheus.yml` par défaut :

```
$ ./promtool check config prometheus.yml
```

Lorsque la configuration est conforme, la commande renvoie la sortie suivante :

```
Checking prometheus.yml
 SUCCESS: prometheus.yml is valid prometheus config file syntax
```

2.3.3 Structure globale de la configuration

La configuration est constituée de champs portant des noms assez explicites :

– `global` : est la configuration globale par défaut de Prometheus.
– `alerting` : permet de configurer les mécanismes d'envoi d'alertes.
– `rule_files` : permet de définir les règles de déclenchement des alertes.
– `scrape_configs` : est la configuration des points de collecte.

Le champ `global` contient deux sous-champs :

– `scrape_interval` : permet de définir la fréquence de collecte par défaut (ici 15 secondes).
– `evaluation_interval` : permet de définir l'intervalle de temps entre deux évaluations de règles.

2.3.4 Définition des points de collecte

Le champ `scrape_configs` est un tableau qui permet de définir la liste des points de collecte. Chaque entrée de ce tableau est constituée d'un enregistrement à la structure suivante :

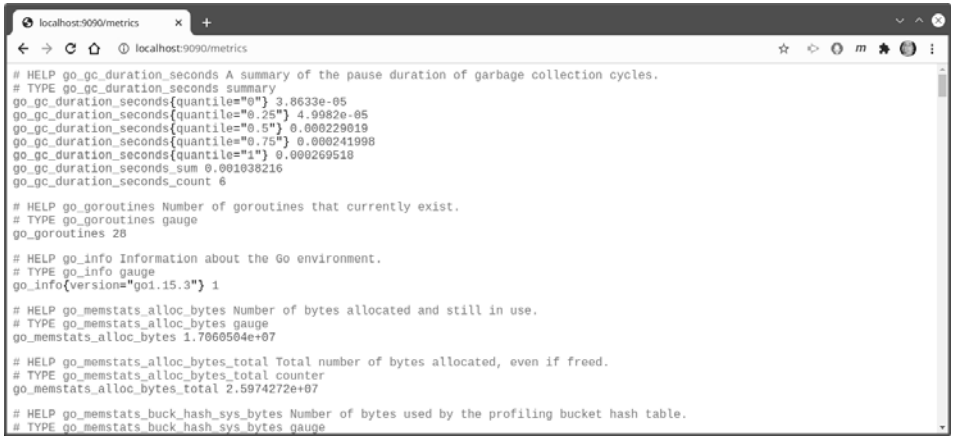
– Le champ `job_name` permettant de définir le travail que doit réaliser Prometheus.
– Le champ `static_configs` définissant la source des points de collecte.

Dans le cas d'une entrée `static_configs`, cette dernière permet de définir directement des points de collecte à l'aide du champ `targets`. Ce champ attend un tableau définissant des points d'entrée.

À noter que le champ `scrape_configs` peut contenir de nombreux autres champs afin de configurer le comportement de Prometheus. Voici ci-dessous quelques exemples de champs intéressants à connaître :

– `basic_auth` : définition d'utilisateur et mot de passe HTTP.
– `tls_config` : configuration de la communication en TLS.
– `proxy_url` : configuration de proxy de communication HTTP.
– `metrics_path` : contexte HTTP permettant d'accéder aux métriques de l'agent (par défaut /metrics).
– `scrape_interval` : définition de configuration d'intervalles de collectes.
– `scrape_timeout` : temps maximum de collecte.

À noter que, par défaut, les métriques de Prometheus sont disponibles à l'adresse suivante : http://localhost:9090/metrics



Point d'entrée des métriques Prometheus

Remarque

Tant que Prometheus n'est pas lancé, ce point d'entrée n'est pas accessible.

Dans le cas d'une définition d'entrée statique ne contenant que la collecte du moteur Prometheus, le champ `scrape_configs` prend la forme suivante :

```
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090']
```

2.4 Lancement de Prometheus

Laissez la configuration de Prometheus par défaut pour passer au lancement. Cette opération consiste à se rendre dans le répertoire du binaire et à lancer la commande `prometheus`.

Voici ci-dessous les commandes correspondantes à lancer :

```
$ cd prometheus-3.2.1.linux-amd64
$ ./prometheus
```

La dernière commande doit alors renvoyer un ensemble d'informations relatives au lancement du moteur Prometheus :

```
level=info ... caller=main.go:315 msg="No time or size retention
was set so using the default time retention" duration=15d
level=info ... caller=main.go:353 msg="Starting Prometheus"
version="(version=...)"
level=info ... caller=main.go:358 build_context="(...)"
level=info ... caller=main.go:359 host_details="(...)"
level=info ... caller=main.go:360 fd_limits="(soft=1024, hard=1048576)"
level=info ... caller=main.go:361 vm_limits="(soft=unlimited,
hard=unlimited)"
level=info ... caller=web.go:516 component=web msg="Start
listening for connections" address=0.0.0.0:9090
level=info ... caller=main.go:712 msg="Starting TSDB ..."
level=info ... caller=head.go:642 component=tsdb msg="Replaying
on-disk memory mappable chunks if any"
level=info ... caller=head.go:656 component=tsdb msg="On-disk
memory mappable chunks replay completed" duration=3.069µs
level=info ... caller=head.go:662 component=tsdb msg="Replaying
WAL, this may take a while"
level=info ... caller=head.go:714 component=tsdb msg="WAL segment
loaded" segment=0 maxSegment=1
level=info ... caller=head.go:714 component=tsdb msg="WAL segment
loaded" segment=1 maxSegment=1
level=info ... caller=head.go:719 component=tsdb msg="WAL replay
```

```
completed" checkpoint_replay_duration=34.84µs
wal_replay_duration=549.339µs total_replay_duration=610.658µs
level=info ... caller=main.go:732 fs_type=EXT4_SUPER_MAGIC
level=info ... caller=main.go:735 msg="TSDB started"
level=info ... caller=main.go:861 msg="Loading configuration file"
filename=prometheus.yml
level=info ... caller=main.go:892 msg="Completed loading of
configuration file" filename=prometheus.yml
totalDuration=599.745µs remote_storage=2.305µs web_handler=416ns
query_engine=882ns scrape=214.705µs scrape_sd=33.631µs
notify=27.631µs notify_sd=16.008µs rules=1.29µs
level=info ... caller=main.go:684 msg="Server is ready to receive
web requests."
```

Certains messages de ce démarrage donnent des indications sur les valeurs par défaut utilisées :

– Le temps de rétention des données, défini à 15 jours par défaut (message `duration=15d`).
– Les informations concernant les limites système (`fd_limits` et `vm_limits`).
– La configuration appliquée par rapport au type de système de fichiers (message `fs_type=EXT4_SUPER_MAGIC`).
– La lecture et l'application du journal d'écritures (`WAL` pour *Write-Ahead Log*).

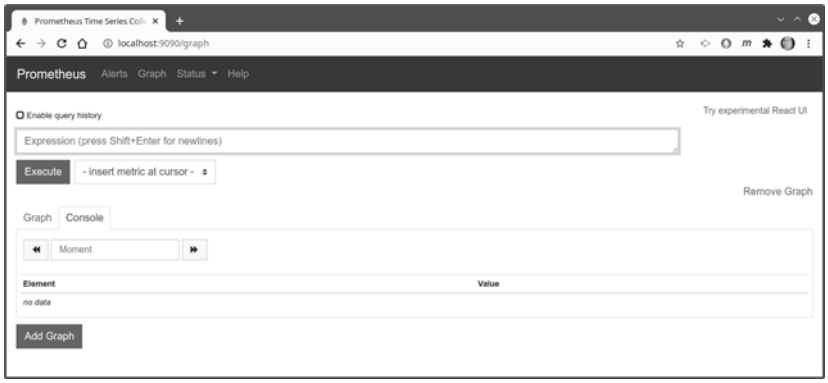
Remarque

Sur une installation vierge, le lancement est quasiment instantané. Dans le cas d'un moteur ayant une activité importante, la gestion du journal d'écritures (WAL) ralentira le temps de démarrage.

Le message *Server is ready to receive web requests* indique que Prometheus est maintenant démarré et accessible.

2.5 Consultation de la console Prometheus

Prometheus est lancé. Il est maintenant possible de consulter sa console en entrant l'adresse http://localhost:9090 dans un navigateur.



Racine par défaut de la console de Prometheus

Depuis cette console, il devient possible d'interroger Prometheus. Pour cela, l'utilisateur doit utiliser le langage PromQL. Un premier exemple de requête peut consister à interroger une métrique existante. Pour ce faire, entrez la requête `up` dans la console.

Cette métrique est un peu particulière puisqu'elle est ajoutée automatiquement par Prometheus pour chacun des points de collecte.