Supervision des applications

1 Contexte Spring

Historiquement, les applications Spring Boot peuvent être supervisées au travers de l'outil Spring Boot Actuator via JMX ou des *endpoints* HTTP.

Spring Boot Actuator met à disposition, au niveau de chaque application, un certain nombre de *endpoints* standards préfixés par défaut par `/actuator` qu'il est possible d'enrichir.

La société Codedentric propose une solution de centralisation du monitoring des applications Spring avec Spring Boot Admin (Server & Client).

Attention

Le - mal nommé (!) - projet *Spring Boot Admin* est un projet communautaire qui n'est pas soutenu par VMware Tanzu (ex Pivotal) et ne fait donc pas officiellement partie de l'écosystème Spring.

La publication de la version 2 de Spring Boot a introduit (en ce qui concerne le monitoring) :

- un *refactoring* du projet Spring Boot Actuator pour
- le support de micrometer (cf. Micrometer: Spring Boot 2's new application metrics collector).

intégration de micrometer dans les dépendances Actuator par défaut dans Spring Boot 2 :

```
[INFO] +- org.springframework.boot:spring-boot-starter-
actuator:jar:2.3.3.RELEASE:compile
[INFO] |  +- (...)
[INFO] |  \- io.micrometer:micrometer-core:jar:1.5.4:compile
[INFO] |     +- (...)
```

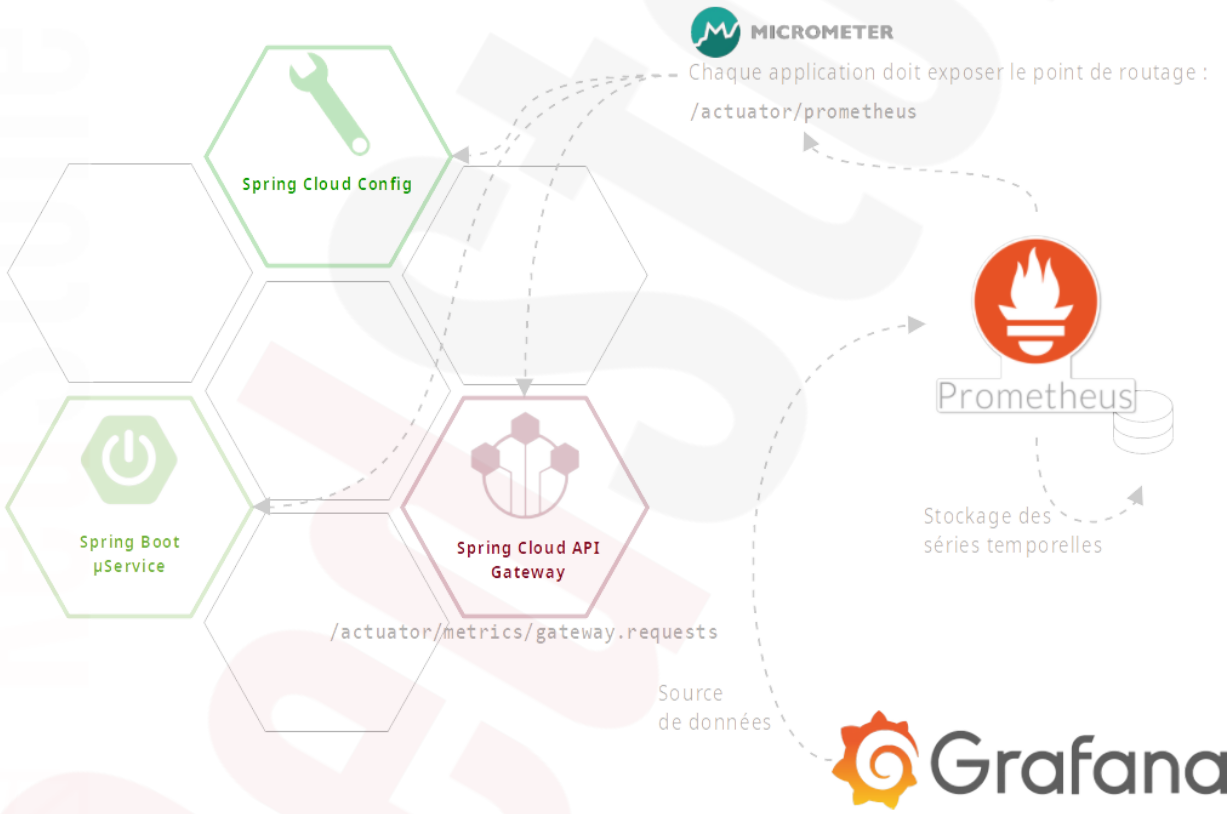
De facto, de nombreux sous-projets Spring, dont Spring Cloud Gateway, intègrent des métriques au format micrometer.

Micrometer est spécialisé dans la collecte des métriques des applications tournant sur une JVM pouvant alimenter différents systèmes de supervision :

- Azure Monitor (cloud Microsoft),
- CloudWatch (AWS),
- Prometheus,
- ...

Il apporte la notion de dimension dans les métriques, en particulier la dimension de temps .

2 Architecture cible



2.1 Fonctionnement

Chaque application quelle qu'elle soit expose le *endpoint* `/actuator/prometheus`.

Les applications qui dispose d'une implémentation spécialisée de Micrometer exposent les *endpoints* associés, par exemple, pour Spring Cloud Gateway : `/actuator/metrics/gateway.requests`

L'instance Prometheus est configurée pour récupérer les informations et les stocker.

Exemple (fichier `prometheus.yml`) :

```
(...)
scrape_configs:
- job_name: spring-cloud-gateway
```

```
    metrics_path: /actuator/prometheus
    static_configs:
    - targets: ['spring.cloud.gateway.url:port']
  (...)
```

L'instance Grafana utilise l'instance Prometheus comme source de données pour réaliser le monitoring de manière visuelle, gérer les alertes, etc.

2.2 le trio gagnant

Sur la base de la recherche des meilleurs candidats selon les critères :

- Collecte des données
- Stockage des données
- Edition et visualisation des tableaux de bord
- Extensibilité du produit
- Compatibilité Cloud

Le critère Open source vs Offre commerciale n'est pas prépondérant car tous les produits cités sont *full featured* en version Open Source.

2.2.1 Micrometer

Micrometer est, depuis Spring Boot 2, l'outil de collecte de métriques dimensionnelles par défaut.

Il est donc plus prudent de se conformer aux évolutions du framework dans son ensemble.

2.2.2 Prometheus

Prometheus permet de collecter des données depuis de nombreuses sources, cf. Prometheus, exporters and integrations. Si ces derniers ne suffisent pas, des librairies sont fournies dans de nombreux langages pour instrumenter le code afin de fournir les métriques voulues.

Prometheus dispose de son propre format de stockage optimisé pour les séries de données temporelles.

Pour la constitution des graphiques et des tableaux de bord, Prometheus semble plus compliqué à prendre en main et n'offre pas autant de souplesse que Grafana.

2.2.3 Grafana

Grafana ne stocke pas les séries de données.

L'outil peut être étendu par l'utilisation de plugins et dispose d'un catalogue de tableaux de bord prêts à l'emploi. Il est par ce biais le plus complet en terme de visualisation des informations collectées.

Grafana offre par ailleurs la possibilité d'effectuer une visualisation en cycle de différents tableaux de bord.

Pour les solutions cloud, le choix entre Prometheus et Grafana repose principalement sur :

- la nécessité de faire la collecte des informations, et
- la facilité d'édition des tableaux de bord.

Dans la mesure où la collecte est souvent fournie *out of the box* dans les offres cloud, Grafana semble être la meilleure solution.

exemple de tableau de bord Grafana :

En conclusion

le choix de :

- Micrometer est fait en cohérence avec l'évolution de la *stack* Spring Boot / Spring Cloud,
- Prometheus pour ses capacités de collecte et de stockage des informations, et
- Grafana pour sa souplesse dans la constitution des tableaux de bord.

Ce choix permet également de :

- Rationaliser les outils de supervision entre les DEV et les OPS.
- Remettre éventuellement en cause l'utilisation de Spring Boot Admin.