TP 07 - Monitoring

7.0 - Objectifs

Estimated Time : **60** *minutes*

Le présent TP se veut présenter comment mettre en place spring boot actuator dans une application spring, exposer les end-point en jmx et utiliser jmxtrans et graphite pour effectuer un dashboard

7.1 - Ajouter Spring Boot Actuator

- Dans le pom parent ou dans chaque enfant de votre projet ajouter la dépendance :

```
[...]
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>
</dependencies>
[...]
```

- Vérifier que les métriques montrées dans le cours apparaissent http://localhost:{port}/metrics, en particulier la métrique health sur resanet-catalogue. Magie!

7.2 - Ajouter un health down

- Implémenter l'interface HealthIndicator pour les besoin de démonstration dont la méthode health() retourne dans tout les cas un status DOWN.
- Qu'observe t-on maintenant (remarquez aussi le status HTTP)?

Notes :

- N'oubliez pas de déclarer votre métrique comme une Bean Spring

7.3 - Configurer des dashboard graphite

- Dans la suite du tp, nous allons travailler uniquement avec **resanet-catalogue**

7.3.1 - Endpoint avec jmx

- Démarrez votre application en local (sans utiliser le docker-compose) et constatez avec JConsole les endpoints springboot-actuator

7.3.2 - Configurer une image graphana / graphite

- rajouter le service suivant dans le `docker-compose.yml` :

```
graphite:
    image: kamon/grafana_graphite
    network_mode: "service:resanet-catalogue"
    volumes:
        - ./data/whisper:/opt/graphite/storage/whisper
        - ./data/grafana:/opt/grafana/data
        - ./log/graphite:/opt/graphite/storage/log/webapp
```

- rajouter l'exposition du port de graphite au service resanet-catalogue (il partage le même réseau) :

```
ports:
   - "127.0.0.1:80:80" # graphite
   - "127.0.0.1:81:81" # graphana
```

Nous rattachons cet image au réseau de resanet-catalogue pour simplifier les configurations réseaux de docker. Notes :

- pour éviter que docker créé les dossiers avec les mauvaises permissions créer les avant de démarrer

- vous pouvez rajouter pour les ports suivant dans resanet-catalogue "127.0.0.1:2003:2003" ce qui vous permettra de tester depuis votre IDE

7.3.3 - Ajouter des métriques dropwizard

- Par essence les métriques springboot-actuator ne sont pas faites pour tracer des graphiques.

- Metrics core permet de créer automatiquement des statistiques sur les appels http (https://dropwizard.github.io/metrics/3.1.0/manual/core/)

- Rajouter dans le pom de resanet-catalogue les dépendances suivantes :

```
[...]
<dependencies>
  <dependency>
    <groupId>io.dropwizard.metrics</groupId>
    <artifactId>metrics-core</artifactId>
    <version>3.1.2</version>
  </dependency>
  <dependency>
    <groupId>io.dropwizard.metrics</groupId>
    <artifactId>metrics-graphite</artifactId>
    <version>3.1.2</version>
  </dependency>
</dependencies>
[...]
```

Lancer `docker-compose`

```
docker-compose up
```

7.3.4 - Configurer les métriques dropwizard

Rajouter la classe suivante dans resanet-catalogue :

```
package com.zenika.microservices.resanet.catalog.metrics;

import com.codahale.metrics.MetricRegistry;
import com.codahale.metrics.graphite.Graphite;
import com.codahale.metrics.graphite.GraphiteReporter;
import org.slf4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.net.InetSocketAddress;
import java.util.concurrent.TimeUnit;

@Configuration
public class MonitoringConfig {

  @Autowired
  private MetricRegistry registry;

  @Bean
  public GraphiteReporter graphiteReporter() {
    Graphite graphite = new Graphite(new InetSocketAddress("localhost",
2003));
    GraphiteReporter reporter = GraphiteReporter.forRegistry(registry)
            .prefixedWith("boot").build(graphite);
    reporter.start(500, TimeUnit.MILLISECONDS);
    return reporter;
  }
}
```

7.3.5 - Observer le résultat

- Aller à l'adresse : localhost:80{0,8}1 et stresser l'application (404, controllers), observer le résultat dans graphite ; un dossier boot s'est créé.
- Explorer les possibilité de graphite pour construire des graphes et effectuer des calculs.
- Notes :
  - vous pouvez utiliser la commande suivante `watch curl localhost:8081/transport` qui requetera toute les deux secondes l'application