Rockpedia : l'API qui fait du bruit

Swagger

Voici les instructions à suivre pour être prêt à coder le plus rapidemment possible.

1. Prérequis

Vous aurez besoin des outils suivants :

- Maven (v3.6.3 ou +)
- JDK8
- Git
- un IDE (nous conseillons Intellij si possible)
- un compte Github (pour pouvoir commit)
- un compte sur la plateforme Jenkins

2. Démarrer

- Clonez le dépot :

```
git clone https://github.com/ALudwig57/Rockpedia
```

- Build avec maven

```
mvn build
```

- Executez les tests

```
mvn test
```

- Build un package

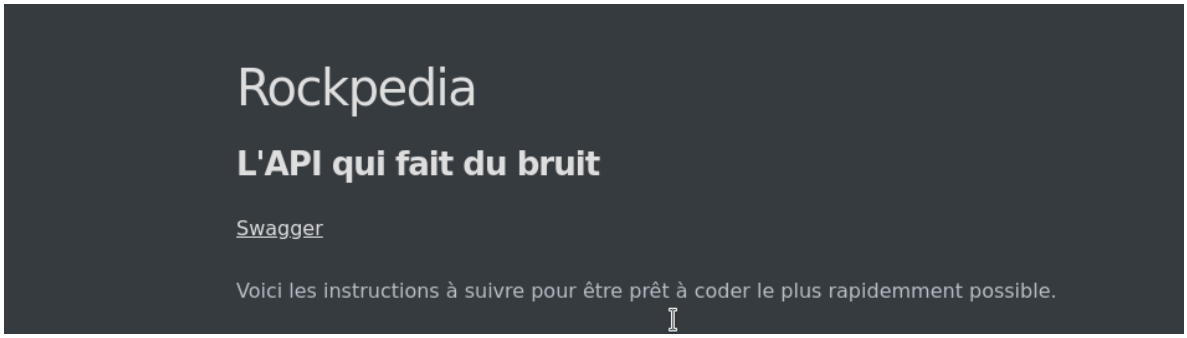
```
mvn package
```

- Démarrez l'application

```
mvn spring-boot:run
```

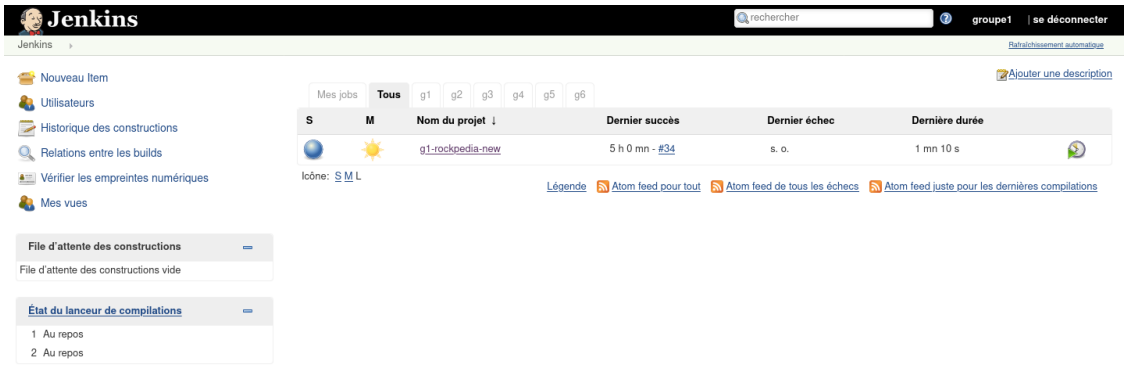
- Rendez vous à l'adresse http://localhost:8080/ avec un navigateur

Vous devriez voir apparaître cette page :



3. Lancer un build Jenkins

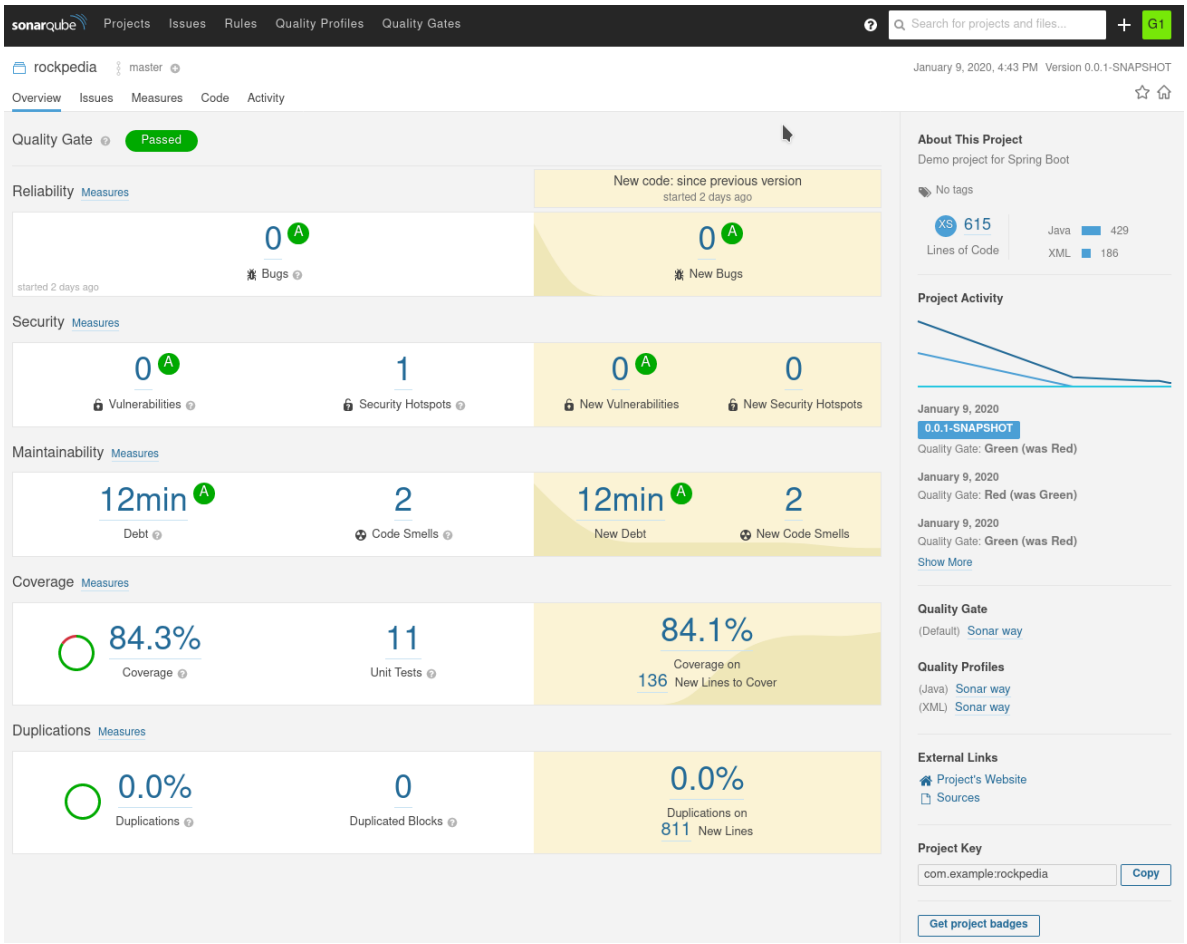
- Rendez vous à l'adresse https://jenkins.m2gi.win/
- Connectez vous. Si tout se passe bien, vous devriez arriver sur un écran comme celui-ci :



- Cliquez sur le nom du projet
- Cliquez sur lancer un build
- Vous devriez voir que quelque chose a démarré. Cliquez sur le #nombre à côté du point clignotant.
- Cliquez sur Console Output pour voir le déroulement du build
- Une fois le build terminé avec succès, passez à l'étape suivante

4. Analyse SonarQube

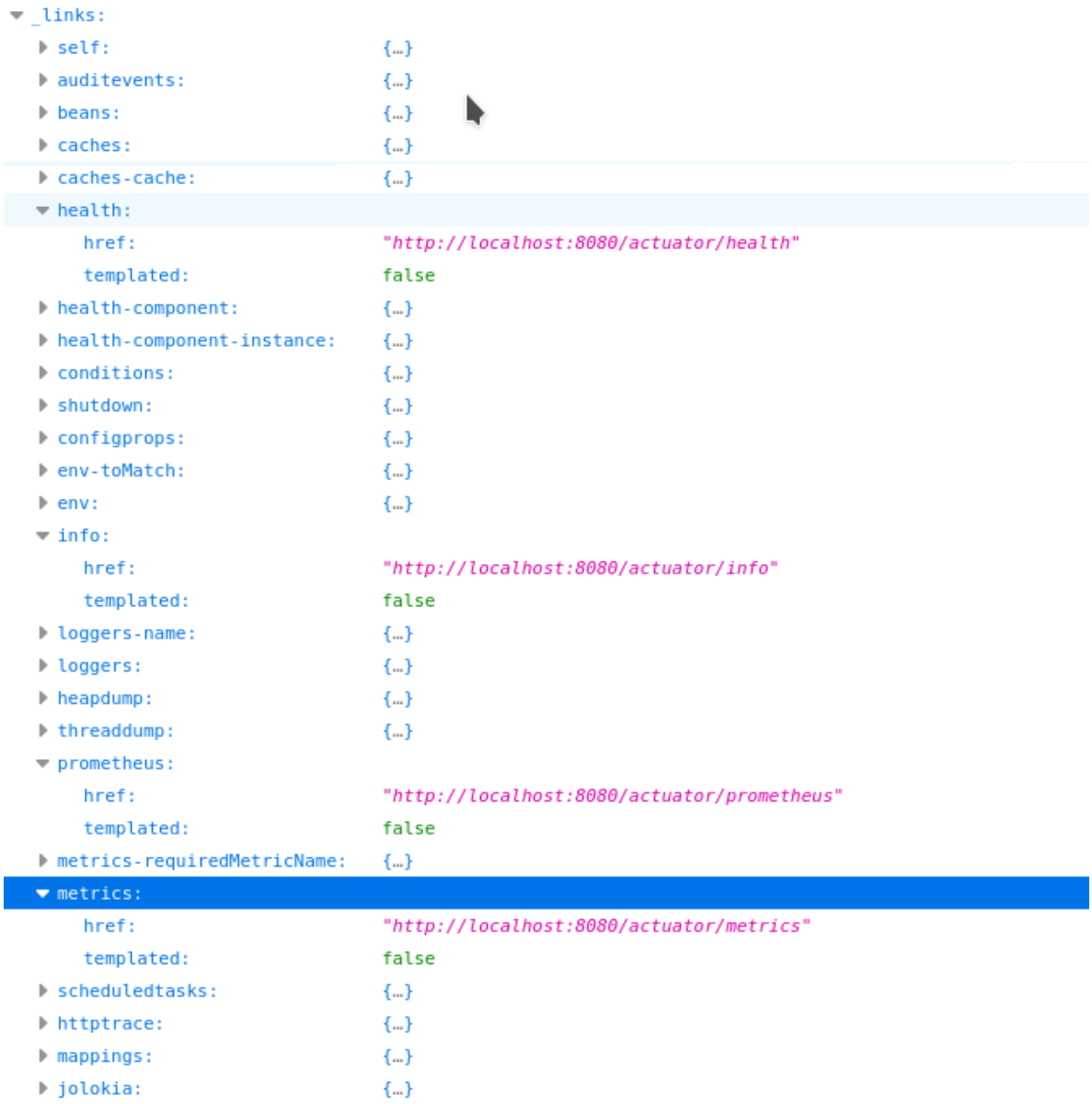
- Rendez vous à l'adresse https://sonarqube.m2gi.win/dashboard?id=com.example%3Arockpedia
- Vous arrivez sur le tableau de bord de l'application



- On peut distinguer les bugs, les failles de sécurités, la dette technique, la couverture de code...

5. Surveillance avec Spring Actuator, Prometheus et Spring Boot Admin

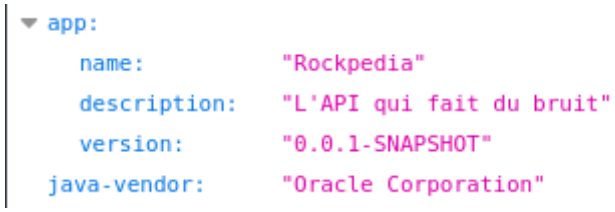
Spring actuator est déjà activé. Cela permet de voir pas mal d'infos relatives à l'application à l'adresse http://localhost:8080/actuator.



J'ai mis en évidence les plus utiles :

- health : permet de savoir si l'application fonctionne (UP) ou pas

- info : informations générales sur l'application



- prometheus : beaucoup d'informations pas très lisibles, surtout utilisé par des applications externes comme graphana ou Spring Boot Admin, que nous allons voir dans un instant.

- metrics : données de surveillance, accessibles à l'adresse http://localhost:8080/actuator/metrics/${DATA} (avec ${DATA} le nom de ce que vous voulez inspecter)


```
▼ names:
    0:  "jvm.memory.max"
    1:  "jvm.threads.states"
    2:  "jdbc.connections.active"
    3:  "process.files.max"
    4:  "jvm.gc.memory.promoted"
    5:  "system.load.average.1m"
    6:  "jvm.memory.used"
    7:  "jvm.gc.max.data.size"
    8:  "jdbc.connections.max"
    9:  "jdbc.connections.min"
    10: "jvm.gc.pause"
    11: "jvm.memory.committed"
    12: "system.cpu.count"
    13: "logback.events"
    14: "http.server.requests"
    15: "tomcat.global.sent"
    16: "jvm.buffer.memory.used"
    17: "tomcat.sessions.created"
    18: "jvm.threads.daemon"
    19: "system.cpu.usage"
    20: "jvm.gc.memory.allocated"
    21: "tomcat.global.request.max"
    22: "hikaricp.connections.idle"
    23: "hikaricp.connections.pending"
    24: "tomcat.global.request"
    25: "tomcat.sessions.expired"
    26: "hikaricp.connections"
    27: "jvm.threads.live"
    28: "jvm.threads.peak"
    29: "tomcat.global.received"
    30: "hikaricp.connections.active"
    31: "hikaricp.connections.creation"
    32: "process.uptime"
    33: "tomcat.sessions.rejected"
    34: "process.cpu.usage"
    35: "tomcat.threads.config.max"
    36: "jvm.classes.loaded"
    37: "hikaricp.connections.max"
    38: "hikaricp.connections.min"
    39: "jvm.classes.unloaded"
    40: "tomcat.global.error"
    41: "tomcat.sessions.active.current"
```

Spring Boot Admin

- Clonez le dépot :

```
git clone https://github.com/SmileEdge/SpringBootAdmin.git
```

- Build avec maven

```
mvn build
```

- Démarrez l'application

```
mvn spring-boot:run
```

- Rendez vous à l'adresse http://localhost:8081/ avec un navigateur, connectez-vous avec le nom d'utilisateur *admin* et le mot de passe *admin* et selectionnez l'application *spring-boot-application* en cliquant une fois dessus, puis sur le numéro sous l'adresse http://localhost:8080/.

Sur la page d'accueil, on peut voir les mêmes infos que par la route info de Spring Actuator, le nombre de threads de l'application, la mémoire utilisée, le nombre de passages du garbage collector...

Aperçus > Metriques permet d'ajouter des métriques visibles sur actuator/metrics.

Aperçus > Envirronement liste les variables d'environnement de l'application.

Aperçus > Propriétés de configuration liste d'autres variables d'environement, plus spécifiques à Spring.

Loggers permet de définir la priorité des logs à afficher (DEBUG < FATAL).