

Rockpedia : l'API qui fait du bruit

[Swagger](#)

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

[1. Prérequis](#) [2. Démarrer l'application](#) [3. Lancer un build Jenkins](#) [4. Analyse SonarQube](#) [5. Surveillance avec Spring Actuator, Prometheus et Spring Boot Admin](#)

1. Prérequis

Vous aurez besoin des outils suivants :

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

2. Démarrer

- Clonez le dépôt :

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

- Build avec maven

```
mvn build
```

- Exécutez 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 :

Rockpedia

L'API qui fait du bruit

[Swagger](#)

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

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 :

The screenshot shows the Jenkins web interface. At the top, there's a search bar and a 'groupe1' user profile. The main area displays a table of jobs. The job 'g1-rockpedia-new' is highlighted, showing a build status of '5 h 0 mn - #34' and a duration of '1 mn 10 s'. Below the table, there are links for 'Légende', 'Atom feed pour tout', 'Atom feed de tous les échecs', and 'Atom feed juste pour les dernières compilations'. On the left sidebar, there are links for 'Nouveau Item', 'Utilisateurs', 'Historique des constructions', 'Relations entre les builds', 'Vérifier les empreintes numériques', and 'Mes vues'. At the bottom, there are sections for 'File d'attente des constructions' and 'Etat du lanceur de compilations'.

- 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

The screenshot shows the SonarQube dashboard for the 'rockpedia' project. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', and 'Quality Gates'. The main area displays a 'Quality Gate' status of 'Passed'. Below this, there are several sections for different metrics: 'Reliability Measures' (0 bugs, 0 new bugs), 'Security Measures' (0 vulnerabilities, 1 security hotspot, 0 new vulnerabilities, 0 new security hotspots), 'Maintainability Measures' (12min debt, 2 code smells, 12min new debt, 2 new code smells), 'Coverage Measures' (84.3% coverage, 11 unit tests, 84.1% coverage on 136 new lines to cover), and 'Duplications Measures' (0.0% duplications, 0 duplicated blocks, 0.0% duplications on 811 new lines). On the right side, there is a 'Project Activity' section with a line graph, an 'About This Project' section with tags and lines of code, a 'Quality Gate' section with a default profile, and a 'Project Key' section with a 'Copy' button.

- On peut distinguer les bugs, les failles de sécurité, 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>.

▼ _links:	
▶ self:	{...}
▶ auditevents:	{...}
▶ beans:	{...}
▶ caches:	{...}
▶ caches-cache:	{...}
▼ health:	
href:	"http://localhost:8080/actuator/health"
templated:	false
▶ health-component:	{...}
▶ health-component-instance:	{...}
▶ conditions:	{...}
▶ shutdown:	{...}
▶ configprops:	{...}
▶ env-toMatch:	{...}
▶ env:	{...}
▼ info:	
href:	"http://localhost:8080/actuator/info"
templated:	false
▶ loggers-name:	{...}
▶ loggers:	{...}
▶ heapdump:	{...}
▶ threaddump:	{...}
▼ prometheus:	
href:	"http://localhost:8080/actuator/prometheus"
templated:	false
▶ metrics-requiredMetricName:	{...}
▼ metrics:	
href:	"http://localhost:8080/actuator/metrics"
templated:	false
▶ scheduledtasks:	{...}
▶ httptrace:	{...}
▶ mappings:	{...}
▶ jolokia:	{...}

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

▼ app:	
name:	"Rockpedia"
description:	"L'API qui fait du bruit"
version:	"0.0.1-SNAPSHOT"
java-vendor:	"Oracle Corporation"

- 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}](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 dépôt :

```
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 sélectionnez 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 > Environnement liste les variables d'environnement de l'application.

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

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