

2021-2022

## Projet Data - Orchestration de services de données



Professeur : Radouane KARRA

Mohamed DHIFAOU

## Table des matières

1. Introduction .....	2
2. Docker.....	2
3. Postman .....	4

## 1. Introduction

Dans le cadre de ce projet on va conteneuriser un microservice qui consisté à créer un rest api avec un premier endpoint pour va réaliser une authentification avec login et password et qui va générer un jwt. Ensuite, on va créer un deuxième endpoint qui va recevoir le jwt et nous retourner comme résultat les informations qui sont à l'intérieur.

## 2. Docker

On commencer par créer l'image docker qui sera tagée "episen-ms-security" :

```
Mohamed@DESKTOP-NEG6EH1 MINGW64 ~/Desktop/episen-ms-security (main)
$ docker build -t episen-ms-security .
#1 [internal] load build definition from Dockerfile
#1 sha256:c6263b55f9f03edc46f02525ad3328b8168f809156cc9bd7e8a7aea89c93e28c
#1 transferring dockerfile: 363B done
#1 DONE 0.0s

#2 [internal] load .dockerignore
#2 sha256:d9d2703b9266b2d56afaf19a9694cc2f3325e94154084e16f1db06a3e8caf551
#2 transferring context: 2B done
#2 DONE 0.0s

#3 [internal] load metadata for docker.io/library/openjdk:11
#3 sha256:87dc9b3cee4adf6787fd792601b37fcaad0ed6bd5314a02f15c26446f91634ad
#3 ...

#4 [auth] library/openjdk:pull token for registry-1.docker.io
#4 sha256:32b04a5dc37d0934eb92354c0155e282cc865a5eb6f12f04d3419d2004b0be62
#4 DONE 0.0s

#3 [internal] load metadata for docker.io/library/openjdk:11
#3 sha256:87dc9b3cee4adf6787fd792601b37fcaad0ed6bd5314a02f15c26446f91634ad
#3 DONE 2.6s

#6 281.4 [INFO] Installing /episen-ms-security/pom.xml to /root/.m2/repository/com/episen/episen-ms-security/1.0.0-SNAPSHOT/episen-ms-security-1.0.0-SNAPSHOT.pom
#6 281.4 [INFO] -----
#6 281.4 [INFO] BUILD SUCCESS
#6 281.4 [INFO] -----
#6 281.4 [INFO] Total time: 02:52 min
#6 281.4 [INFO] Finished at: 2021-12-15T22:52:45Z
#6 281.4 [INFO] -----
#6 DONE 281.4s

#7 [3/3] WORKDIR episen-ms-security/
#7 sha256:0dbd22dbcb051394d36b9acfbe522a3faaa203fd3b46dcdd19bd64b5d266499
#7 DONE 0.0s

#8 exporting to image
#8 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#8 exporting layers
#8 exporting layers 1.9s done
#8 writing image sha256:9abe228c9e75bd2d089f92dc241ab3a4f5b0f23726e3c91acdd53444f9944090 done
#8 naming to docker.io/library/episen-ms-security done
#8 DONE 1.9s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

Mohamed@DESKTOP-NEG6EH1 MINGW64 ~/Desktop/episen-ms-security (main)
```

Ensuite, on vérifie que l'image a été bien construite et on récupère son ID :

```
Mohamed@DESKTOP-NEG6EH1 MINGW64 ~/Desktop/episen-ms-security (main)
$ docker images | grep episen-ms-security
episen-ms-security   latest          9abe228c9e75    23 minutes ago    1.05GB
```

Après, on lance un conteneur nommé "episen-ms-security" sur le port 9000 en utilisant l'ID de l'image récupéré à l'étape précédente:

```
Mohamed@DESKTOP-NEG6EH1 MINGW64 ~/Desktop/episen-ms-security (main)
$ docker run -d --name episen-ms-security -p 9000:9000 9abe228c9e75
c0de6d569c4c08f8869ee10c8302e7f7263b148f87599fcb78b7c99464780333
```

Maintenant, on peut vérifier que le conteneur a été bien construit et on récupère son ID :

```
Mohamed@DESKTOP-NEG6EH1 MINGW64 ~/Desktop/episen-ms-security (main)
$ docker ps -a | grep episen-ms-security
c0de6d569c4c    9abe228c9e75    "java -jar ./target/..."    About a minute ago    Up About a minute    0.0.0.0:9000->9000/tcp    episen-ms-security
```

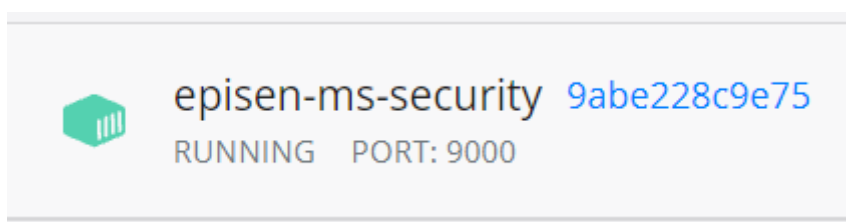
Enfin, on peut lire les logs du conteneur :

```
Mohamed@DESKTOP-NEG6EH1 MINGW64 ~/Desktop/episen-ms-security (main)
$ docker logs -f episen-ms-security

:: Spring Boot ::
(v2.2.4.RELEASE)

2021-12-15 23:17:40.736 INFO 1 --- [main] com.episen.ms.MsSecurityApplication : Starting MsSecurityApplication v1.0.0-SNAPSHOT on c0de6d569c4c with PID 1
2021-12-15 23:17:40.744 INFO 1 --- [main] com.episen.ms.MsSecurityApplication : No active profile set, falling back to default profiles: default
2021-12-15 23:17:42.457 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 9000 (http)
2021-12-15 23:17:42.469 INFO 1 --- [main] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-12-15 23:17:42.470 INFO 1 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.30]
2021-12-15 23:17:42.538 INFO 1 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-12-15 23:17:42.539 INFO 1 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1720 ms
2021-12-15 23:17:43.289 INFO 1 --- [main] o.s.s.web.DefaultSecurityFilterChain : Creating filter chain: any request, [org.springframework.security.web.con
xt.request.async.WebAsyncManagerIntegrationFilter@1698fc68, org.springframework.security.web.context.SecurityContextPersistenceFilter@686449f9, org.springframework.securit
.web.header.HeaderWriterFilter@6035b93b, org.springframework.security.web.authentication.logout.LogoutFilter@2a8d39c4, com.episen.ms.setting.JwtAuthenticationFilter@40844a
b, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@68b6f0d6, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@5c00
84f, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@4504d271, org.springframework.security.web.session.SessionManagementFilter@2974f221, org
springframework.security.web.access.ExceptionTranslationFilter@3e7634b9, org.springframework.security.web.access.intercept.FilterSecurityInterceptor@62163b39]
2021-12-15 23:17:43.458 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-12-15 23:17:43.813 INFO 1 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2021-12-15 23:17:43.908 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 9000 (http) with context path ''
2021-12-15 23:17:43.913 INFO 1 --- [main] com.episen.ms.MsSecurityApplication : Started MsSecurityApplication in 3.712 seconds (JVM running for 4.235)
```

On peut aussi vérifier sur l'application Docker et on trouvera le conteneur :



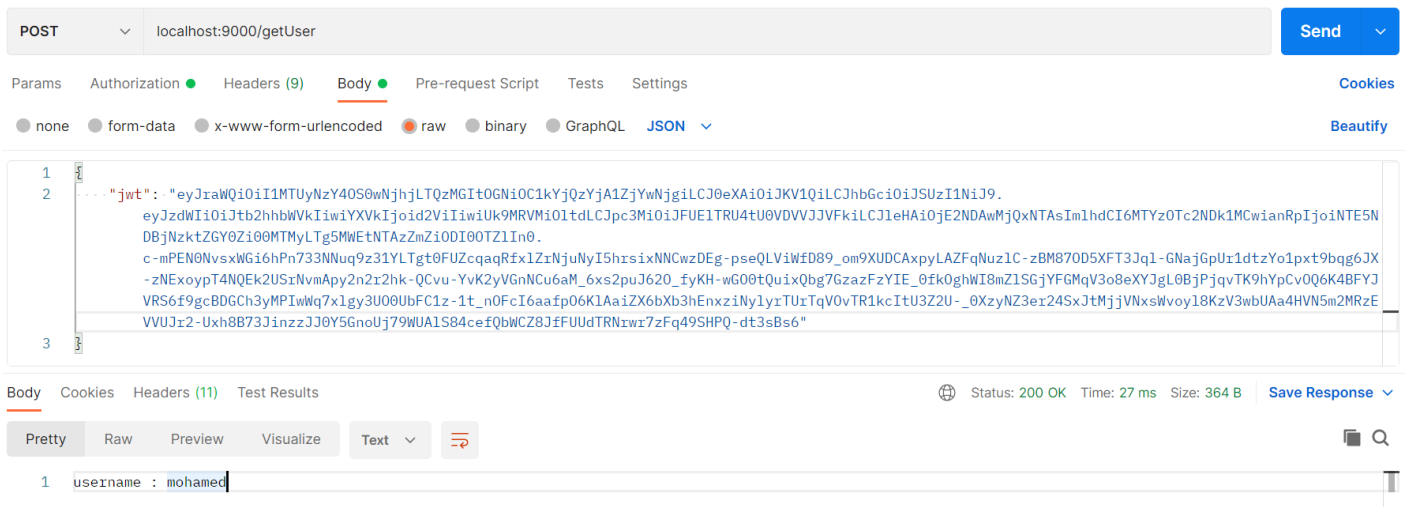
On commence par faire une requête POST qui va tester le premier endpoint qui utilise un username et password pour générer un JWT :

Ensuite on fait une requête POST qui teste le second endpoint qui utilise le JWT généré précédemment pour récupérer le "username" :

Dans la partie "Authorization" on choisit le type "Bearer Token" et on saisit dans la case "Token" : `Header.Payload.Signature` :

Dans la partie "Body" choisir "raw" et le type "json" et on saisit :

```
{ "jwt": "Header.Payload.Signature" }
```



The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** localhost:9000/getUser
- Body Type:** raw (selected)
- Body Content:**

```
{
  "jwt": "eyJraWQioiI1MTUyNzY4OS0wNjhjLTQzMGI0MGNiO01kYjZyYjA1ZjYwNjgiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJtb2hhbWVkaWwiYXVkaWoid2ViIiwiaW9MRVMI01tdCJpc3MiOiJFUElTRU4tU0V0VGVJVFkiLCJleHAiOiJlNDAwMjQxNTAsImh0dCI6MTYzOTc2NDk1MCwianRpIjoieNTE5NDBjNzktZGY0Zi00MTMyLTg5MWEtNTAzZmZiODI0OTZ1In0.c-mPEN0NvsxWGi6hPn733NNuq9z31YLTgt0FUZcqaqRfxlZrNjuNyI5hrsixNNCwzDEg-pseQLViWfD89_om9XUDCAxpyLAZFqNuz1C-zBM870D5XFT3Jq1-GNajGpUr1dtzYo1pxt9bqg6JX-zNExoypT4NQEk2USrNvmApy2n2r2hk-QCvu-YvK2yVGnNCu6aM_6xs2puJ620_fyKH-wG00tQuixQbg7GazFzYIE_0fk0ghWII8mZ1SGjYFGMqV3o8eXYJgLOBjPjqvTK9hYpCv0Q6K4BFYJVR56f9gcBDGCh3yMPIwWq7xlgY3U00UbFC1z-1t_n0FcI6aaFp06K1AaiZX6bXb3hEnxziNy1yrTUrTqV0vTR1kcItU3Z2U-_0XzyNZ3er24SxJtmjjVNxsWvoy18KzV3wbUaa4HVN5m2MRzEVVUJr2-Uxh8B73JinzJJ0Y5GnoUj79WUA1S84cefQbWCZ8JfFUUDTRN1wr7zFq49SHPQ-dt3sBs6"
}
```
- Status:** 200 OK
- Time:** 27 ms
- Size:** 364 B
- Response Body:**

```
1 username : mohamed
```