esprit So former autrement	**Applications web Distribuées** **Année universitaire 2023/2024** **UP WEB**

Atelier

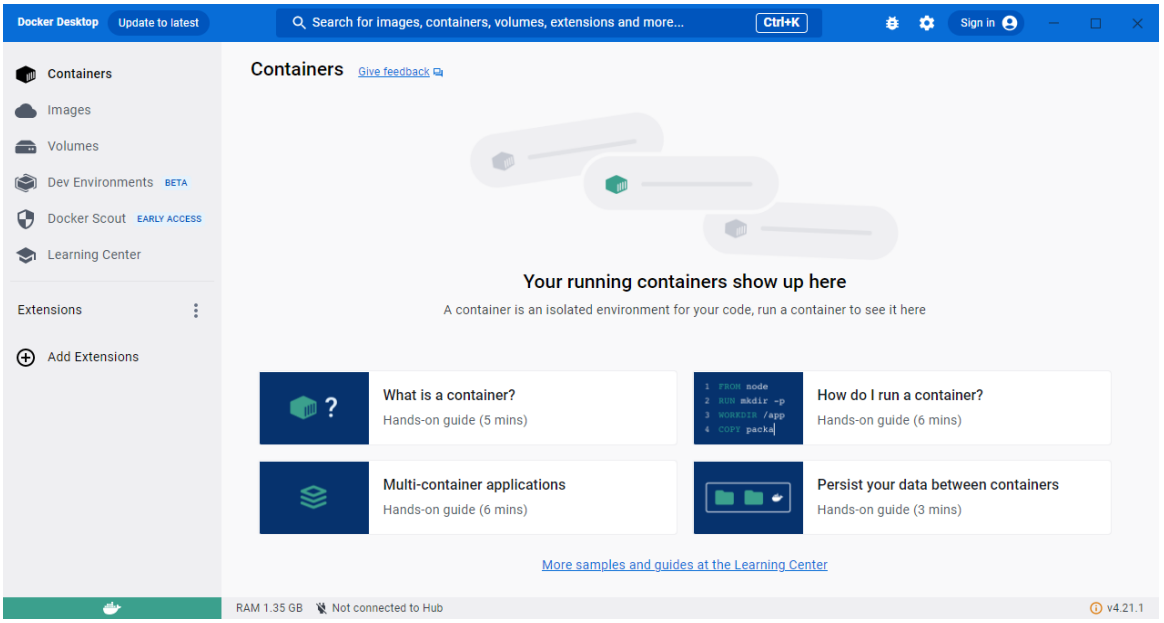
Micro Service avec Docker

Objectifs

- Déploiement d'un Micro service Spring Boot avec Docker.
- Découvrez comment héberger un Micro service sur un conteneur docker.

1- Installation Docker

- Avant de déployer l'application sur Docker, assurez-vous d'avoir installé Docker. Dans cet exemple, nous avons utilisé **Docker Desktop** pour Windows.

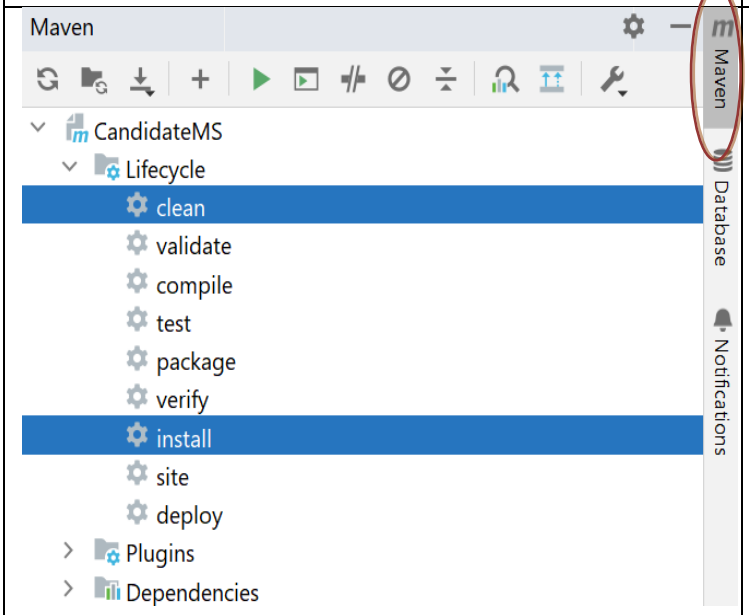
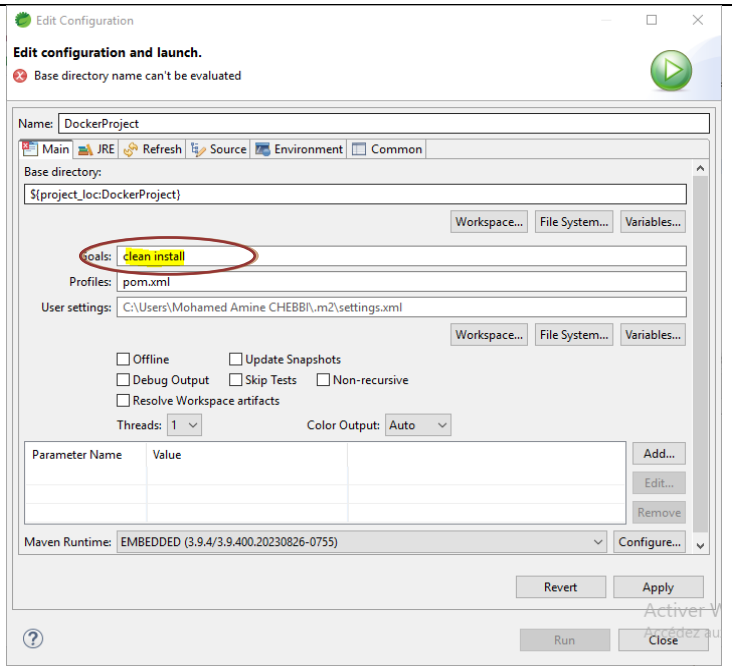


2. Déployer une application Spring Boot avec Docker

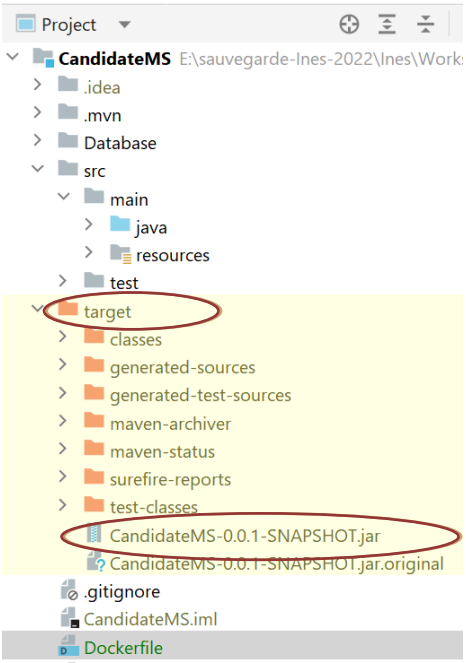
a- Dockerfile

- Les images Docker sont un élément important du travail avec le moteur Docker.
- Dans la structure de projet ci-dessous, nous avons créé un nouveau fichier « **Dockerfile.txt** » sous la racine du projet.
- Maintenant, avant de remplir le fichier **Dockerfile** avec les instructions nécessaires, et avant d'exécutez la commande Docker pour construire l'image et la déployez-la sur Docker, nous devons générer le fichier .jar.

Pour ce faire, utiliser la commande **mvn clean install.** Pour la créer en **Eclipse** → Run AS➔ Maven build➔ **clean install** comme goals

IntelliJ	Eclipse
	

Le fichier « **CandidatMS-0.0.1-SNAPSHOT.jar** » est généré sous le dossier **target** comme indiqué dans la figure suivante :



- Le fichier Docker ci-dessous contient les commandes permettant de créer l'image :

Dockerfile | CandidateMsApplication.java | console | CandidateMS.iml | pom.xml

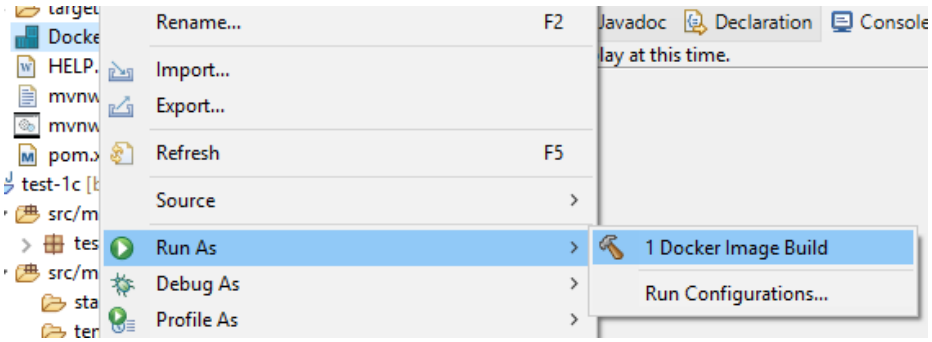
```
1  FROM openjdk:8
2  EXPOSE 8082
3  ADD target/CandidateMS-0.0.1-SNAPSHOT.jar CandidatMS-docker.jar
4  ENTRYPOINT ["java", "-jar", "CandidatMS-docker.jar"]
```

- **FROM :** Doit être la première dans le fichier Docker. Cette commande crée un calque à partir de l'image Docker. Dans notre cas, nous avons utilisé openjdk: 8, ce qui signifie que cette application fonctionnera sous Java 8.
- **EXPOSE** : Exposer le port pour le noeud final. Dans cet exemple, nous avons configuré 8089.
- **ADD** : Cette commande permet de prendre une source et une destination.
- **ENTRYPOINT :** C'est semblable à CMD, où le fichier de commande / jar sera exécuté.

b- Création de l'image Docker

En utilisant « **Eclipse Docker Toolings** » lancer la commande « **Docker Image Build** » pour la création de l'image.

Docker lit les commandes et instructions de "Dockerfile.txt" et construit l'image.



Le résultat de l'exécution sera comme suit :

```
Step 1/4 : FROM openjdk:8
 ---> b273004037cc
Step 2/4 : EXPOSE 8082
 ---> Using cache
 ---> ad642356cc66
Step 3/4 : ADD target/CandidateMS-0.0.1-SNAPSHOT.jar CandidatMS-docker.jar
 ---> Using cache
 ---> 7f7430aed33e
Step 4/4 : ENTRYPOINT ["java", "-jar", "CandidatMS-docker.jar"]
 ---> Using cache
 ---> e2e4ce873465

Successfully built e2e4ce873465
```

Pour déployer l'image à tavers un terminal, il faut exécuter la commande suivante :

- **Docker build -t dockerms .**

Vous pouvez vérifier l'image soit via docker desktop, soit via le terminal avec la commande ci-dessous.

- **$ docker images**

Pour déployer l'image il faut lancer la commande :

- **docker run -p 8082:8089 -t dockerms**

```
C:\Users\Mohamed Amine CHEBBI\Documents\workspace-spring-tool-suite-4-4.16.0.RELEASE>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED      STATUS                  PORTS                    NAMES
66ac57d66254   dockerms       "docker-entrypoint.s…"   2 days ago   Exited (0) 2 days ago   0.0.0.0:8089/tcp         kind_curran
```

En fin pour accéder à l'application il suffit de taper « http://localhost :8089 »