TP3 – Création d'images

Brahim Hamdi

Objectifs :

- Créer l'image en utilisant *Dockerfile*

Préparation de l'environnement et compilation/test du microservice

Avant de conteneuriser une application, on doit la compiler et la tester. Dans cette partie, nous allons compiler et tester un microservice écrit en java.

1. Sur votre hôte Docker, installez les paquets *openjdk-11-jdk* et *maven*.

```
vagrant@Manager:~$ sudo apt install openjdk-11-jdk maven -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fonts-dejavu-extra
  gsettings-desktop-schemas java-common libaopalliance-java libapache-pom-java libasound2 libasound2-data
  libatinject-jsr330-api-java libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data
  libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcdi-api-java libcommons-cli-java libcommons-io-java
  libcommons-lang3-java libcommons-parent-java libcups2 libdconf1 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1
  libfontenc1 libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libgif7 libgl1 libgl1-amber-dri
  libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libguava-java libguice-java libharfbuzz0b
```

...

2. Git clonez le dépôt *https://github.com/brahimhamdi/demoapp.git*

```
vagrant@Manager:~$ git clone https://github.com/brahimhamdi/demoapp.git
Cloning into 'demoapp'...
remote: Enumerating objects: 260, done.
remote: Counting objects: 100% (260/260), done.
remote: Compressing objects: 100% (142/142), done.
remote: Total 260 (delta 85), reused 242 (delta 69), pack-reused 0 (from 0)
Receiving objects: 100% (260/260), 98.62 KiB | 343.00 KiB/s, done.
Resolving deltas: 100% (85/85), done.
vagrant@Manager:~$
```

3. Sous le répertoire *demoapp*, créez le package ***demoapp.jar*** avec la commande ***mvn clean package***.

```
vagrant@Manager:~$ cd demoapp/
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ mvn clean package
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for com.dduportal.voxxed2016lu:demoapp:jar:1.0.0-SNA
PSHOT
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-surefire-plugin is missing. @ line 85, column 21
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-failsafe-plugin is missing. @ line 91, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----------------< com.dduportal.voxxed2016lu:demoapp >-----------------
[INFO] Building Demo DropWizard Application 1.0.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-failsafe-plugin/maven-metadata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-failsafe-plugin/maven-metadata.xml
(2.0 kB at 946 B/s)
```

...

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:58 min
[INFO] Finished at: 2024-10-19T13:52:07Z
[INFO] ------------------------------------------------------------------------
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ ls target/
classes      generated-sources       maven-archiver  original-demoapp.jar  test-classes
demoapp.jar  generated-test-sources  maven-status     surefire-reports
vagrant@Manager:~/demoapp$
```

Après l'exécution de la dernière commande, Maven a fait plusieurs tâches :

- Créer répertoire ***target***
- Compiler le code source et génèrer les classes sous ***target/classes***
- Faire les test et mettre le résultat sous ***target/surfire-reports***
- Archiver le microservice et ces dépendances dans le fichier ***target/demoapp.jar***

Création usuelle d'images

Une fois compiler et tester avec succès, l'étape suivante est de conteneuriser le microservice. Dans cette partie, nous verrons la méthode manuelle de création de l'image.

4. Toujours sous le répertoire *demoapp* de la machine hôte de Docker :
   - Démarrez le conteneur *demoapp* (créé dans le TP précédent) et qui exécute l'image *openjdk:8-jre-alpine* :

```
vagrant@Manager:~/demoapp$ docker ps -a
CONTAINER ID   IMAGE                COMMAND     CREATED       STATUS                     PORTS     NAMES
c0aa0eb94faf   openjdk:8-jre-alpine   "/bin/sh"   3 hours ago   Exited (0) 3 hours ago             demoapp
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker start demoapp
demoapp
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker ps -a
CONTAINER ID   IMAGE                COMMAND     CREATED       STATUS         PORTS                    NAMES
c0aa0eb94faf   openjdk:8-jre-alpine   "/bin/sh"   3 hours ago   Up 2 seconds   0.0.0.0:8080->8080/tcp   demoapp
vagrant@Manager:~/demoapp$
```

   - Copiez le paquet jar vers *app/app.jar* sous le conteneur *demoapp*. Vérifier que le fichier a été bien copié.

```
vagrant@Manager:~/demoapp$ ls
dependency-reduced-pom.xml  hello-world.yml  pom.xml             pom.xml.ori  src
Dockerfile                  jenkinsfile      pom.xml-checkstyle  README.md    target
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker cp target/demoapp.jar demoapp:/app/app.jar
Successfully copied 17MB to demoapp:/app/app.jar
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker exec -t demoapp /bin/ls /app
app.jar
vagrant@Manager:~/demoapp$
```

   - Copiez le fichier *hello-world.yml* vers */app/config.yml* sous le conteneur *demoapp*

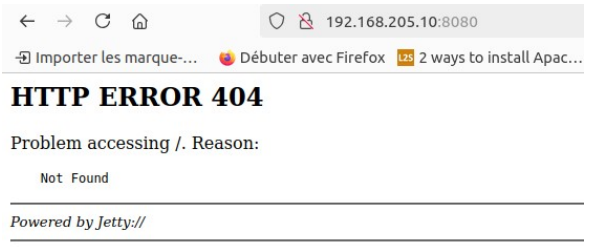
```
vagrant@Manager:~/demoapp$ docker cp hello-world.yml demoapp:/app/config.yml
Successfully copied 2.05kB to demoapp:/app/config.yml
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker exec -t demoapp /bin/ls /app
app.jar     config.yml
vagrant@Manager:~/demoapp$
```

5. Connectez-vous maintenant au conteneur *demoapp* pour lancer le microservice.

```
vagrant@Manager:~/demoapp$ docker exec -ti demoapp sh
/ #
/ # java -jar /app/app.jar server /app/config.yml
INFO  [2024-10-19 14:08:38,172] org.eclipse.jetty.util.log: Logging initialized @1813ms
INFO  [2024-10-19 14:08:38,257] io.dropwizard.assets.AssetsBundle: Registering AssetBundle with name: assets for path /*
INFO  [2024-10-19 14:08:38,287] io.dropwizard.server.ServerFactory: Starting hello-world
INFO  [2024-10-19 14:08:38,297] io.dropwizard.server.DefaultServerFactory: Registering jersey handler with root path prefix: /
INFO  [2024-10-19 14:08:38,310] io.dropwizard.server.DefaultServerFactory: Registering admin handler with root path prefix: /
INFO  [2024-10-19 14:08:38,377] org.eclipse.jetty.setuid.SetUIDListener: Opened application@674658f7{HTTP/1.1}{0.0.0.0:8080}
INFO  [2024-10-19 14:08:38,378] org.eclipse.jetty.setuid.SetUIDListener: Opened admin@5c8eee0f{HTTP/1.1}{0.0.0.0:8081}
INFO  [2024-10-19 14:08:38,382] org.eclipse.jetty.server.Server: jetty-9.2.z-SNAPSHOT
INFO  [2024-10-19 14:08:38,860] io.dropwizard.jersey.DropwizardResourceConfig: The following paths were found for the configured
```

...

6. En utilisant le navigateur de la machine hôte, affichez l'interface web du microservice.

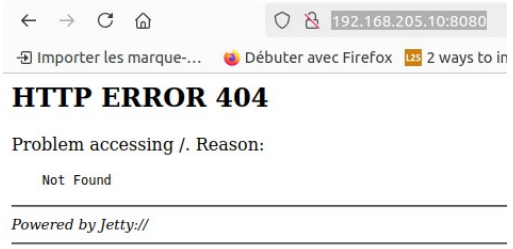


7. Quittez le conteneur et convertissez-le en une nouvelle image docker nommée *demoapp:1.0-usel*

```
vagrant@Manager:~/demoapp$ docker commit demoapp demoapp:1.0-usuel
sha256:aa65e16535f99ea8fee830f063519b67e7af01bf2a6ddb5c55cfb5cb71d5e912
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker images
REPOSITORY   TAG            IMAGE ID       CREATED         SIZE
demoapp      1.0-usuel      aa65e16535f9   6 seconds ago   113MB
nginx        latest         3b25b682ea82   2 weeks ago     192MB
mariadb      latest         4b8711c6c639   6 weeks ago     407MB
openjdk      8-jre-alpine   f7a292bbb70c   5 years ago     84.9MB
vagrant@Manager:~/demoapp$
```

8. Testez la nouvelle image *demoapp:1.0-usuel*

```
vagrant@Manager:~/demoapp$ docker run -d --name demoapp-usuel -p 8080:8080 demoapp:1.0-usuel java -jar /app/app.jar server /app/config.yml
5bd2e3589cfa2d9380c18019fa1381aea7614689410c787a9d738f0588551e94
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS                    NAMES
5bd2e3589cfa   demoapp:1.0-usuel  "java -jar /app/app.…"   4 seconds ago   Up 2 seconds   0.0.0.0:8080->8080/tcp   demoapp-usuel
vagrant@Manager:~/demoapp$ _
```



Création automatisée d'image

Pour créer des images Docker, nous recommandons la méthode automatique. Dans cette partie on va automatiser la conteneurisation du microservice en utilisant un fichier Dockerfile qui contient la liste des étapes sous forme d'instructions.

9. Arrêtez et supprimez tous les conteurs et toutes les images.

```
vagrant@Manager:~/demoapp$ docker ps -a
CONTAINER ID   IMAGE               COMMAND                  CREATED          STATUS          PORTS                    NAMES
5bd2e3589cfa   demoapp:1.0-usuel   "java -jar /app/app.…"   10 minutes ago   Up 10 minutes   0.0.0.0:8080->8080/tcp   demoapp-us
uel
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker stop  demoapp-usuel
demoapp-usuel
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker rm demoapp-usuel
demoapp-usuel
vagrant@Manager:~/demoapp$
```

10. Le ficher *Dockerfile* (sous le dossier demoapp) décrit toutes les opérations qu'on a fait dans la partie précédente, interprêtez les instructions de ce fichier.

```
vagrant@Manager:~/demoapp$ cat Dockerfile
FROM openjdk:8-jre-alpine
MAINTAINER Damien DUPORTAL <dduportal@cloudbees.com>

COPY ./target/demoapp.jar /app/app.jar
COPY hello-world.yml /app/config.yml
EXPOSE 8080

ENTRYPOINT ["java","-jar","/app/app.jar"]
CMD ["server","/app/config.yml"]
vagrant@Manager:~/demoapp$
```

11. Lancez le build de la nouvelle image *demoapp:1.0-auto*. Quelle est le nombre de couches de la nouvelle image ?

```
vagrant@Manager:~/demoapp$ docker build -t demoapp:1.0-auto .
[+] Building 0.2s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 283B
 => [internal] load .dockerignore
 => => transferring context: 150B
 => [internal] load metadata for docker.io/library/openjdk:8-jre-alpine
 => [1/3] FROM docker.io/library/openjdk:8-jre-alpine
 => [internal] load build context
 => => transferring context: 102B
 => CACHED [2/3] COPY ./target/demoapp.jar /app/app.jar
 => CACHED [3/3] COPY hello-world.yml /app/config.yml
 => exporting to image
 => => exporting layers
 => => writing image sha256:c9e3d3dd96705d658e9ae364b8de4dafdaf35c919018ea929e500c34838ba762
 => => naming to docker.io/library/demoapp:1.0-auto
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker inspect demoapp:1.0-auto | grep Layers -A6
            "Layers": [
                "sha256:f1b5933fe4b5f49bbe8258745cf396afe07e625bdab3168e364daf7c956b6b81",
                "sha256:9b9b7f3d56a01e3d9076874990c62e7a516cc4032f784f421574d06b18ef9aa4",
                "sha256:edd61588d12669e2d71a0de2aab96add3304bf565730e1e6144ec3c3fac339e4",
                "sha256:178c73a76e197dfd71b22834ff48c7143f6491bb8acaca973b6f2c0edf509f63",
                "sha256:ca5700d5c1bfb5b845e2bddbbf88141736b516e9635e1169a51681c9a0fef40f"
            ]
vagrant@Manager:~/demoapp$
```

12. Testez la nouvelle image *demoapp:1.0-auto*

```
vagrant@Manager:~/demoapp$ docker images
REPOSITORY    TAG            IMAGE ID       CREATED          SIZE
demoapp       1.0-auto       c9e3d3dd9670   9 minutes ago    102MB
demoapp       1.0-usuel      aa65e16535f9   29 minutes ago   113MB
nginx         latest         3b25b682ea82   2 weeks ago      192MB
mariadb       latest         4b8711c6c639   6 weeks ago      407MB
openjdk       8-jre-alpine   f7a292bbb70c   5 years ago      84.9MB
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker run -d --name demoapp-auto -p 8080:8080 demoapp:1.0-auto
914b637f7eb81dc658acee894345577db021083805b3a6fc6b3c56152aa4c64a
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                    NAMES
914b637f7eb8   demoapp:1.0-auto   "java -jar /app/app.…"   11 seconds ago   Up 10 seconds   0.0.0.0:8080->8080/tcp   demoapp-aut
o
vagrant@Manager:~/demoapp$
```

Sauvegarder l'image dans une registry

Dans cette dernière partie, nous sauvegarderons notre microservice conteneurisé dans le registre Docker Hub.

13. Créez un compte sur Docker Hub (https://hub.docker.com/)

14. Avant de pousser l'image vers le Docker Hub, vous devez changer le nom de l'image et de vous vous authentifier sur DockerHub.

```
vagrant@Manager:~/demoapp$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker push demoapp:1.0-auto
The push refers to repository [docker.io/library/demoapp]
ca5700d5c1bf: Preparing
178c73a76e19: Preparing
edd61588d126: Preparing
9b9b7f3d56a0: Preparing
f1b5933fe4b5: Preparing
denied: requested access to the resource is denied
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker tag  demoapp:1.0-auto brahimhamdi/demoapp:latest
vagrant@Manager:~/demoapp$
vagrant@Manager:~/demoapp$ docker images
REPOSITORY             TAG            IMAGE ID       CREATED          SIZE
brahimhamdi/demoapp    latest         c9e3d3dd9670   10 minutes ago   102MB
demoapp                1.0-auto       c9e3d3dd9670   10 minutes ago   102MB
demoapp                1.0-usuel      aa65e16535f9   31 minutes ago   113MB
nginx                  latest         3b25b682ea82   2 weeks ago      192MB
mariadb                latest         4b8711c6c639   6 weeks ago      407MB
openjdk                8-jre-alpine   f7a292bbb70c   5 years ago      84.9MB
vagrant@Manager:~/demoapp$
```

15. Une fois authentifier, vous pouvez poussez l'image vers le register Docker Hub avec le nom approprié.

```
vagrant@Manager:~/demoapp$ docker push brahimhamdi/demoapp:latest
The push refers to repository [docker.io/brahimhamdi/demoapp]
ca5700d5c1bf: Pushed

178c73a76e19: Pushing [================>                                  ]  5.309MB/16.97MB

edd61588d126: Pushing [==========>                                        ]  15.38MB/79.39MB

178c73a76e19: Pushed
edd61588d126: Pushed
f1b5933fe4b5: Pushed

latest: digest: sha256:485ca883887e902ad7398a35a493854f266a95f6394f79a6af7d09e0e7b20ec7 size: 1366
vagrant@Manager:~/demoapp$
```