

Le Dockerfile et ses instructions

Table des matières

I. Tout sur le Dockerfile	3
A. Introduction.....	3
II. Exercice : Quiz	7
III. Exemple : Mon image docker de DL	8
A. Créer son image docker spéciale deep learning	8
IV. Exercice : Quiz	10
V. Essentiel	11
VI. Auto-évaluation	11
A. Exercice	11
B. Test.....	11
Solutions des exercices	13

I. Tout sur le Dockerfile

Durée : 1 h 30

Environnement de travail : Visual studio / Powershell (Windows) ou Terminal (Mac OS)

Prérequis : avoir quelques connaissances de docker

Contexte

Supposons que vous devez utiliser la dernière image Ubuntu pour l'environnement de développement de votre solution d'intelligence artificielle. Pour pouvoir utiliser cet environnement compartimenté, il vous faudra paramétrer l'image. Pour ce faire, vous devez créer un *Dockerfile*, et sans doute autant de Dockerfiles qu'il y aura de variations de l'environnement considéré.

Une fois l'image créée, vous n'avez qu'à lancer automatiquement vos *containers* et vous pouvez vous focaliser sur votre projet. Ce cours sert d'introduction à la réalisation de Dockerfile et de création d'image docker d'une application Python qui entraîne un réseau de neurones. Le cours fonctionne si vous possédez Windows, et dans ce cas, vous utiliserez Powershell. Si vous travaillez sur macOS ou Ubuntu, vous utiliserez le terminal ou shell.

A. Introduction

Windows vs Mac

Pour exécuter vos commandes Docker vous pouvez utiliser sur Windows : Powershell ou sur Mac : le terminal. Vous écrivez alors de la même manière les commandes telles que ci-dessous :

```
1 docker build <flags> .  
2 docker run <flags> <image>  
3
```

Image de base

Il est plutôt rare de construire une image Docker de zéro. Vous trouverez la plupart du temps une image de base dont elles héritent. Dans le cas de l'image qui n'a pas de parent, l'expression **FROM** est suivie du mot-clé « *scratch* » écrit dans le Dockerfile.

```
1 # Maintainer : "assansanogo@gmail.com"  
2 # ici, l'image parente s'appelle "ubuntu" avec le tag "18.04"  
3 FROM ubuntu:18.04  
4  
5 CMD [ "/bin/bash" ]  
6
```

Commande RUN

L'instruction RUN vous permet d'installer votre application et les packages nécessaires en créant une nouvelle image en validant les résultats. Il est possible que vous rencontriez plusieurs lignes avec les instructions RUN dans un Dockerfile mais jamais une commande RUN par librairie installée.

RUN existe sous deux formes :

- Commande : **RUN** <commande> (forme shell directe)
- Commande : **RUN ["exécutable", "param1", "param2"]** (forme executable)

Le Dockerfile ci-dessous illustre ces commandes en installant plusieurs librairies : à la fois Linux et Python.

```

1 # Maintainer : "assansanogo@gmail.com"
2 # download ubuntu image
3 FROM ubuntu:18.04
4
5 RUN apt-get update
6 RUN DEBIAN_FRONTEND=noninteractive apt-get update && apt-get install -y python3.8 python3-pip
7 # crée le répertoire predicteev dans l'image
8 RUN mkdir /predicteev
9
10 # installation des librairies via requirements.txt
11 RUN pip3 install -r /predicteev/requirements.txt
12
13 # exécute l'application
14 ENTRYPOINT [ "python3" ]
15 CMD [ "/predicteev/app.py" ]
16

```

Commande COPY et ADD

Si vous désirez uniquement copier des fichiers et / ou répertoires, vous pouvez utiliser de manière interchangeable **ADD** (ajouter) et **COPY** (copier).

D'un point de vue syntaxe : les fonctions ADD / COPY sont suivies d'abord par la source que vous souhaitez copier (<src>) suivie de la destination où vous souhaitez la stocker (<dest>). Si la source est un répertoire, ADD en copie la totalité.

Par exemple, si le fichier est disponible localement et que vous voulez l'ajouter au répertoire d'une image, alors le fichier sera copié et ses metadata également :

```

1 # copie les fichiers .py predicteev dans l'image
2 ADD ./python_code/trucmuche.zip /local_folder

1 # copie les fichiers .py predicteev dans l'image
2 ADD http://www.predicteev.com/trucmuche.zip /local_folder

```

La commande COPY a été introduite après la commande ADD.

À l'inverse de ADD, il est impossible de télécharger et de copier des fichiers externes dans votre conteneur par la commande COPY.

Pour utiliser l'instruction COPY, la syntaxe est similaire à celle de ADD :

```

1 COPY <source> <destination>

```

Le Dockerfile ci-dessous :

- Met à jour et installe le gestionnaire de paquets **apt-get**.
- Installe Python 3.8.
- Installe le gestionnaire de paquets **pip3**.

La commande ci-dessous empêche une utilisation interactive du shell. Cette commande est nécessaire pour empêcher le shell de demander à l'utilisateur de valider ou de choisir tel ou tel critère et de bloquer l'installation de l'image.

```

1 RUN DEBIAN_FRONTEND=noninteractive

1 # Maintainer : "assansanogo@gmail.com"
2 # download ubuntu image
3 FROM ubuntu:18.04
4
5 RUN apt-get update
6 RUN DEBIAN_FRONTEND=noninteractive apt-get update && apt-get install -y python3.8 python3-pip
7 # crée le repertoire predicteev dans l'image

```

```

8 RUN mkdir /predicteev
9
10 # copie les fichiers .py predicteev dans l'image
11 ADD https://upload.wikimedia.org/wikipedia/commons/3/3c/Predict_Logo.jpg /predicteev
12 COPY ./python_code/* /predicteev
13
14 # copie les fichiers concernant le modèle prédictif dans l'image
15 # extrait les fichiers tout en les copiant de local vers l'image
16 COPY ./finalized_model.pkl /predicteev
17 ADD ./myfile.zip /predicteev
18 #installation des librairies via requirements.txt
19 RUN pip3 install -r /predicteev/requirements.txt
20
21 # execute application
22 ENTRYPOINT [ "python3" ]
23 CMD [ "/predicteev/app.py" ]
24

```

Commande VOLUME

Une image docker exécute la succession d'installation des logiciels dans une image. Cette image est définie par le Dockerfile.

On retiendra que les images docker sont en lecture seule, et seuls les containers autorisent l'écriture de données dans des fichiers.

Ainsi, toute modification de fichier à l'intérieur du conteneur crée une copie provisoire qui sera détruite avec le conteneur lorsqu'il est arrêté ou supprimé, cette couche de lecture-écriture est perdue. *C'est pourquoi il nous faut créer des volumes.*

Par ce mécanisme, l'hôte (= votre ordinateur) va pouvoir partager son propre système de fichiers avec le conteneur.

Cela nous permet d'utiliser un conteneur pour exécuter des outils ou librairies que nous voulons uniquement installer dans l'image, tout en travaillant avec les fichiers présents sur la machine hôte (votre ordinateur). Par exemple, si nous voulons utiliser une version personnalisée de Python pour un script particulier, nous pourrions exécuter ce script dans un conteneur Python modifié, monté dans notre répertoire de travail actuel :

```
1 docker run -v /doesnt/exist_but_will:/foo -w /foo -i -t ubuntu bash
```

Commande ENTRYPOINT

L'instruction **ENTRYPOINT** fait fonctionner votre conteneur en tant qu'exécutable.

Comme CMD, ENTRYPOINT peut être configuré sous deux formes :

Mode executable

ENTRYPOINT ["exécutable", "paramètre1", "paramètre2"]

Mode shell

Commande ENTRYPOINT paramètre1 paramètre2

Attention, ENTRYPOINT représente la commande par défaut, donc si une image possède un ENTRYPOINT et que vous lui transmettez un argument, lors de l'exécution du conteneur, l'argument passé sera ajouté comme argument supplémentaire au code par défaut.

```

1 FROM ubuntu:latest
2 # récupère image ubuntu
3 RUN apt-get update && \
4     apt-get install -y apache2-utils && \
5     rm -rf /var/lib/apt/lists/*
6 # update de apt-get et installation de la librairie apache2-utils

```

```
7 # suppression des fichiers /var/lib/apt/lists/*
8 ENTRYPOINT ["python3"]
9 # exécution de python 3
10
```

Commande CMD

L'instruction CMD vous permet de définir une commande par défaut, qui sera exécutée uniquement lorsque vous exécutez le conteneur sans spécifier de commande. Si le conteneur docker s'exécute avec une commande, la commande par défaut sera ignorée. Si Dockerfile a plus d'une instruction CMD, toutes les instructions CMD sauf la dernière sont ignorées. Enfin, on retiendra que CMD et ENTRYPOINT sont des commandes qui travaillent en synergie.

CMD peut revêtir trois formes :

- Commande **CMD ["parametre1","parametre2"]** (définit des **paramètres par défaut** supplémentaires pour ENTRYPOINT sous forme exécutable),
- Commande **CMD ["executable","parametre1","parametre2"]** (forme exécutable, préférée),
- Commande **CMD parametre1 parametre2** (forme shell).

La seconde forme est un peu plus complexe : elle est utilisée avec l'instruction ENTRYPOINT sous forme exécutable. Dans ce scénario, on définit les paramètres par défaut qui seront ajoutés après les paramètres ENTRYPOINT.

Voyons comment fonctionne l'instruction CMD dans l'extrait suivant du Dockerfile :

```
1 # Maintainer : "assansanogo@gmail.com"
2 # download ubuntu image
3 FROM ubuntu:18.04
4
5 RUN apt-get update
6 RUN DEBIAN_FRONTEND=noninteractive apt-get update && apt-get install -y python3.8 python3-pip
7 # créé le répertoire predicteev dans l'image
8 RUN mkdir /predicteev
9
10 # copie les fichiers .py predicteev dans l'image
11 COPY ./python_code/* /predicteev
12
13 # copie les fichiers concernant le modèle prédictif dans l'image
14 # extrait les fichiers tout en les copiant de local vers l'image
15 COPY ./finalized_model.pkl /predicteev
16 ADD ./myfile.zip /predicteev
17 #installation des librairies via requirements.txt
18 RUN pip3 install -r /predicteev/requirements.txt
19
```

En définitive, les commandes CMD, ENTRYPOINT et RUN peuvent donc être exécutées :

De manière exécutable :

```
1 RUN ["apt-get", "install", "python3"]
2 CMD ["/bin/echo", "Hello world"]
3 ENTRYPOINT ["/bin/echo", "Hello world"]
4
```

De manière shell :

```
1 RUN apt-get install python3
2 CMD echo "Hello world"
3 ENTRYPOINT echo "Hello world"
4
```

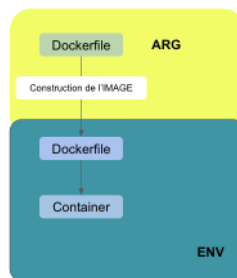
Commande ENV

Lors de l'utilisation de docker, nous distinguons deux types de variables différents - **ARG** et ENV.

ARG est le mot-clé pour des variables qui ont une durée de vie limitée : elles ne sont disponibles qu'à partir du moment où elles sont écrites et déclarées dans le Dockerfile avec une instruction ARG. Leur durée de vie se termine à la construction de l'image. Les conteneurs en cours d'exécution ne peuvent pas accéder aux valeurs des variables ARG (définies à la création d'image).

On retiendra que les valeurs ARG peuvent cependant être facilement inspectées après la création d'une image, il ne faut pas les utiliser pour stocker des informations sensibles.

Les variables ENV sont également accessibles par quelqu'un de mal intentionné, dès que vous introduisez les variables d'environnement. Il convient d'utiliser des fichiers d'environnement et un cryptage pour garantir la sécurité.



Commande WORKDIR

La commande **WORKDIR** effectue en réalité la combinaison des deux commandes mkdir et cd. Voici un exemple de Dockerfile où WORKDIR désigne le répertoire courant dans le container appelé / project.

```
1 FROM ubuntu:16.04
2
3 WORKDIR /project
4
5 RUN npm install
6
```

Exercice : Quiz

[solution n°1 p.15]

Question 1

Vous pouvez exécuter la commande suivante pour changer de répertoire :

```
1 FROM ubuntu:18.04
2 RUN mkdir /mon_folder
3 RUN COPY /another_folder /mon_folder
4 RUN chdir /another_folder
5 RUN apt-get update
6 ENTRYPOINT ["/bin/echo", "Hello world"]
7
```

- ☐ Vrai
- ☐ Faux

Question 2

La commande qui sera exécutée sera « *Bonjour à tous* ».

```
1 FROM ubuntu:18.04
2 RUN apt-get update && apt-get install python3-opencv
3 ENTRYPOINT ["/bin/echo", "Hello world"]
4 CMD echo "Bonjour à tous"
5
```

- ☐ Vrai
- ☐ Faux

Question 3

Le Dockerfile ci-dessous est erroné.

```
1 FROM ubuntu:18.04
2 RUN apt-get update && apt-get install python3-opencv
3 WORKDIR /telecharments
4 RUN wget https://upload.wikimedia.org/wikipedia/commons/3/3c/Predict_Logo.jpg -o .
5 CMD echo "Bonjour je suis un Dockerfile plein de bugs"
6
```

- ☐ Vrai
- ☐ Faux

Question 4

Vous pouvez construire une image docker (docker build) en spécifiant les variables d'environnement.

- ☐ Vrai
- ☐ Faux

Question 5

Vous pouvez exécuter un conteneur docker (docker run) en spécifiant les variables d'environnement et les arguments adaptés.

- ☐ Vrai
- ☐ Faux

III. Exemple : Mon image docker de DL

A. Créer son image docker spéciale deep learning

Problématique

Très souvent, vous devez avoir un environnement où les librairies de deep learning sont déjà présentes. Nous allons nous placer dans le cas de Yolo v.5 qui est un modèle de deep learning très connu. Le principe reste le même lorsque vous voulez réaliser cette opération avec un autre Jupyter notebook.

Qu'est-ce que yolov5 ?

Yolo est un algorithme de deep learning, qui permet de faire de la détection d'objets en temps réel. Il existe une multitude de détecteurs d'objets, qui sont plus ou moins rapides. YOLO est la référence en termes de temps d'inférence. Il est à sa cinquième itération, il est actuellement utilisé pour les applications de deep learning de type embarqué. Ce projet est maintenu par la compagnie Ultralytics. Vous trouverez plus d'informations ici : YOLOv5 Documentation¹

Méthode Définir le Dockerfile

Si votre machine possède un GPU, vous pouvez utiliser le code ci-dessous :

```
1 # commencer par une image de base
2 # qui permet de configurer facilement un GPU
3 # lorsqu'il est de type nvidia
4 FROM nvidia/cuda:10.2-devel-ubuntu18.04 as intermediate
5 # upd
6 RUN apt-get update && DEBIAN_FRONTEND=noninteractive \
7 && apt-get install git \
8 # git clone yolov5
9 && git clone https://github.com/ultralytics/yolov5 \
10 && cd yolov5 \
11 && pip install -qr ./requirements.txt \
12 && pip install jupyter
13 # run jupyter notebook
14 WORKDIR /yolov5
15 CMD ["jupyter notebook -i 0.0.0.0 --allow-root --port=8888"]
16
```

Sinon :

```
1 # commencer par une image de base
2
3 FROM ubuntu:20.04 as intermediate
4 # upd
5 RUN apt-get update && DEBIAN_FRONTEND=noninteractive \
6 && apt-get install git \
7 # git clone yolov5
8 && git clone https://github.com/ultralytics/yolov5 \
9 && cd yolov5 \
10 && pip install -qr ./requirements.txt \
11 && pip install jupyter
12 # run jupyter notebook
13 WORKDIR /yolov5
14 CMD ["jupyter notebook -i 0.0.0.0 --allow-root --port=8888"]
15
```

Méthode Créer l'image Docker

Pour créer l'image docker, vous pouvez exécuter la commande ci-dessous dans un environnement type macOS , Linux ou Powershell.

```
1 sudo docker build -t docker_dl .
```

¹ <https://docs.ultralytics.com/>

Méthode Exécuter le container d'entraînement

```
1 sudo docker run -it docker_dl -p 8888:8888
```

Pour pouvoir exécuter ce container, il faut mapper le port du container avec celui de la machine hôte.

Méthode

Exercice : Quiz

[solution n°2 p.16]

Question 1

Lorsque vous voulez exécuter une application du type jupyter (qui fonctionne en mode serveur) vous devez faire un mapping de port container → host.

- ☐ Vrai
- ☐ Faux

Question 2

Le mapping de ports se fait à la création de l'image.

- ☐ Vrai
- ☐ Faux

Question 3

La commande CMD prend la préséance sur la commande ENTRYPOINT.

- ☐ Vrai
- ☐ Faux

Question 4

Un Dockerfile peut être dépourvu d'une commande de type ENTRYPOINT.

- ☐ Vrai
- ☐ Faux

Question 5

Le dépôt github ne nous appartient pas. On aurait pu cependant modifier le fichier requirements.txt et y ajouter la version de jupyter désirée.

```
1 # commencer par une image de base
2 # qui permet de configurer facilement un GPU
3 # lorsqu'il est de type nvidia
4 FROM nvidia/cuda:10.2-devel-ubuntu18.04 as intermediate
5 # upd
6 RUN apt-get update && DEBIAN_FRONTEND=noninteractive \
7 && apt-get install git \
8 # git clone yolov5
9 && git clone https://github.com/ultralytics/yolov5 \
10 && cd yolov5 \
11 && pip install -qr ./requirements.txt \
12 && pip install jupyter
13 # run jupyter notebook
14 WORKDIR /yolov5
```

```
15 CMD ["jupyter notebook -i 0.0.0.0 --allow-root --port=8888"]
```

```
16
```

- ☐ Vrai
- ☐ Faux

V. Essentiel

Le Dockerfile contient les instructions de construction et d'exécution de l'image Docker.

L'avantage d'un Dockerfile par rapport au simple stockage de l'image binaire (ou d'un instantané/modèle dans d'autres systèmes de virtualisation) est la facilité de mise à jour des images mises à disposition sur Docker Hub. C'est une bonne chose du point de vue de la sécurité, car vous voulez vous assurer que vous n'installez aucun logiciel vulnérable.

Lorsque vous exécutez un entraînement de machine learning (sur le cloud ou localement), l'idéal est de pouvoir entraîner vos modèles sans vous soucier des conflits de bibliothèque. Vous souhaitez également une certaine reproductibilité de vos résultats. Enfin, parfois, les librairies qui ont permis d'écrire un code intéressant sont obsolètes et incompatibles avec votre machine.

C'est pourquoi les conteneurs Docker sont indiqués afin de mettre terme à ces conflits et long débogage.

Vous pouvez même avoir plusieurs containers sur la même machine qui possèdent PyTorch, Tensorflow et même MXNET. Vous pouvez aussi spécifier à docker quels GPU utiliser, etc.

Par défaut, les conteneurs sont dans des environnements séparés. Ainsi, si l'un des conteneurs est compromis ou doit être arrêté, seul ce dernier sera débranché puis rebranché.

De cette manière, si un entraînement avait lieu, les autres GPU ne sont pas impactés.

Vous pouvez combiner docker à github et l'ajouter à l'image docker principale.

Chaque fois qu'un container est créé, vous récupérez le code sur votre dépôt github.

Les dossiers / fichiers locaux peuvent être également montés facilement lors du démarrage du conteneur. Docker vous permet de gagner en productivité !

VI. Auto-évaluation

A. Exercice

Question 1

[solution n°3 p.17]

On vous demande d'écrire un Dockerfile qui permet d'installer :

- Nano - paquet Linux
- Sklearn - paquet Python
- tqdm - paquet Python

Question 2

[solution n°4 p.17]

Comment allez-vous exécuter ce Jupyter Notebook ? Comment allez-vous vous y connecter dans votre navigateur web ?

B. Test

Exercice 1 : Quiz

[solution n°5 p.19]

Question 1

Vous pouvez également définir un volume dans le Dockerfile.

Si vous ne créez pas ce volume au niveau du dockerfile, vous devez exécuter la commande :

docker run -v <hostfolder>:/var/www/html

```
1 # volume dans le container
2 VOLUME /var/www/html
3
```

- ☐ Vrai
- ☐ Faux

Question 2

Dans la commande CMD, app.py est le paramètre qui sera utilisé par Entrypoint.

```
1 FROM ubuntu:16.04
2
3 RUN apt-get update -y && \
4     apt-get install -y python-pip python-dev
5
6 COPY ./requirements.txt /app/requirements.txt
7
8 WORKDIR /app
9
10 RUN pip install -r requirements.txt
11
12 COPY . /app
13
14 ENTRYPOINT [ "python" ]
15
16 CMD [ "app.py" ]
17
```

- ☐ Vrai
- ☐ Faux

Question 3

Vous pouvez exécuter la commande CMD sans Entrypoint.

```
1 FROM ubuntu:16.04
2
3 RUN apt-get update -y && \
4     apt-get install -y python-pip python-dev
5
6 COPY ./requirements.txt /app/requirements.txt
7
8 WORKDIR /app
9
10 RUN pip install -r requirements.txt
11
12 COPY . /app
13
14 CMD [ "python" , "app.py" ]
15
```

- ☐ Vrai
- ☐ Faux

Question 4

Vous pouvez optimiser l'écriture :

```
1 FROM ubuntu
2 RUN apt-get update
3 RUN apt-get install -y apache2
4 RUN apt-get install -y apache2-utils
5 RUN apt-get clean
6 EXPOSE 80
7 CMD ["apache2ctl", "-D", "FOREGROUND"]
8
```

Comme ci-dessous :

```
1 FROM ubuntu
2 RUN apt-get update
3 RUN apt-get install -y apache2 && \
4 apt-get install -y apache2-utils && \
5 apt-get clean
6 EXPOSE 80
7 CMD ["apache2ctl", "-D", "FOREGROUND"]
8
```

☐ Vrai

☐ Faux

Question 5

Ces lignes de commandes sont-elles correctes ?

```
1 # construire l'image
2 docker build -t first-dockerfile -f Dockerfile1 .
3 # lister les images
4 docker images
5 # exécuter le container associé à l'image
6 docker run -d --name first-container first-dockerfile
7 # use exec for interaction avec le container dont l'id est : f1edbfca3eac
8 docker exec -it f1edbfca3eac bash
9
```

☐ Vrai

☐ Faux

Solutions des exercices


Exercice p. 7 Solution n°1**Question 1**

Vous pouvez exécuter la commande suivante pour changer de répertoire :

```
1 FROM ubuntu:18.04
2 RUN mkdir /mon_folder
3 RUN COPY /another_folder /mon_folder
4 RUN chdir /another_folder
5 RUN apt-get update
6 ENTRYPOINT ["/bin/echo", "Hello world"]
7
```

☐ Vrai

☒ Faux

 Pour changer de répertoire, vous devez utiliser WORKDIR. Dans le cas présent, vous changez de répertoire dans la commande RUN associée, mais pas pour les prochaines commandes.


Question 2

La commande qui sera exécutée sera « *Bonjour à tous* ».

```
1 FROM ubuntu:18.04
2 RUN apt-get update && apt-get install python3-opencv
3 ENTRYPOINT ["/bin/echo", "Hello world"]
4 CMD echo "Bonjour à tous"
5
```

☒ Vrai

☐ Faux

 La commande CMD a précédence sur la commande ENTRYPOINT.


Question 3

Le Dockerfile ci-dessous est erroné.

```
1 FROM ubuntu:18.04
2 RUN apt-get update && apt-get install python3-opencv
3 WORKDIR /telecharments
4 RUN wget https://upload.wikimedia.org/wikipedia/commons/3/3c/Predict_Logo.jpg -o .
5 CMD echo "Bonjour je suis un Dockerfile plein de bugs"
6
```


☒ Vrai

☐ Faux

 Vous devez préciser le flag -y dans apt-get install <package> -y. Sans quoi, le shell va vous demander de valider une instruction, chose impossible quand vous créez votre image.


Question 4

Vous pouvez construire une image docker (docker build) en spécifiant les variables d'environnement.

- ☐ Vrai
- ☒ Faux
-  Les variables d'environnement sont spécifiées pour les containers ! Dans ce cas, il faut spécifier des arguments par le mot-clé ARG.

Question 5


Vous pouvez exécuter un conteneur docker (docker run) en spécifiant les variables d'environnement et les arguments adaptés.

- ☐ Vrai
- ☒ Faux
-  Les variables d'environnement sont spécifiées pour les containers ! (Les arguments sont spécifiés pour les images).

Exercice p. 10 Solution n°2

Question 1


Lorsque vous voulez exécuter une application du type jupyter (qui fonctionne en mode serveur) vous devez faire un mapping de port container → host.

- ☒ Vrai
- ☐ Faux
-  Le mapping se fait comme ci-dessus :

```
1 sudo docker run -it docker_dl -p 8888:8888
```


Question 2

Le mapping de ports se fait à la création de l'image.

- ☐ Vrai
- ☒ Faux
-  Le mapping se fait à la création du container et non à la création de l'image.

Question 3

La commande CMD prend la préséance sur la commande ENTRYPOINT.

- ☐ Vrai
- ☒ Faux
-  La commande ENTRYPOINT ne peut pas être remplacée par CMD. CMD ajoute des arguments à la commande ENTRYPOINT

Question 4

Un Dockerfile peut être dépourvu d'une commande de type ENTRYPOINT.

- ☒ Vrai
- ☐ Faux

- Vous pouvez avoir un Dockerfile qui n'exécute pas de commandes par défaut via ENTRYPOINT. Vous pouvez alors spécifier la commande au niveau de CMD.

Question 5

Le dépôt github ne nous appartient pas. On aurait pu cependant modifier le fichier requirements.txt et y ajouter la version de jupyter désirée.

```
1 # commencer par une image de base
2 # qui permet de configurer facilement un GPU
3 # lorsqu'il est de type nvidia
4 FROM nvidia/cuda:10.2-devel-ubuntu18.04 as intermediate
5 # upd
6 RUN apt-get update && DEBIAN_FRONTEND=noninteractive \
7 && apt-get install git \
8 # git clone yolov5
9 && git clone https://github.com/ultralytics/yolov5 \
10 && cd yolov5 \
11 && pip install -qr ./requirements.txt \
12 && pip install jupyter
13 # run jupyter notebook
14 WORKDIR /yolov5
15 CMD ["jupyter notebook -i 0.0.0.0 --allow-root --port=8888"]
16
```

☐ Vrai

☒ Faux

- Vous ne pouvez pas modifier le dépôt github¹.
Vous devez alors ajouter votre propre fichier requirements.txt.

p. 11 Solution n°3

On installe nano via apt-get.

On installe les packages python via python3-pip.

Dockerfile

```
1 FROM ubuntu:devel
2 RUN DEBIAN_FRONTEND=noninteractive apt-get update && DEBIAN_FRONTEND=noninteractive apt-get
  install -y nano python3 python3-pip
3 RUN pip3 install jupyter sklearn tqdm
4 CMD ["jupyter", "notebook", "--allow-root", "--port=8866", "--ip=0.0.0.0", "--allow-root", "--
  no-browser"]
5
```

On crée alors l'image jup3_img

```
1 docker build -t jup3_img.
```

p. 11 Solution n°4

¹ <https://github.com/ultralytics/yolov5>

- Vous devez maîtriser les -flags utilisés :
 - -it : mode interactif
 - -d : mode détaché (vous ne verrez pas la sortie des commandes)
- Vous devez maîtriser le mapping de port :
On mappe le port 8866 de la machine hôte au port 8866 du container.
- Enfin, vous devez spécifier l'adresse dans votre browser :
 - **localhost:8866** ou **127.0.0.1:8866**
 - Le token est spécifié dans les messages liés à Jupyter

```
1 docker run -it -p 8866:8866 jup3_img
```

Vous obtenez ceci dans votre shell :

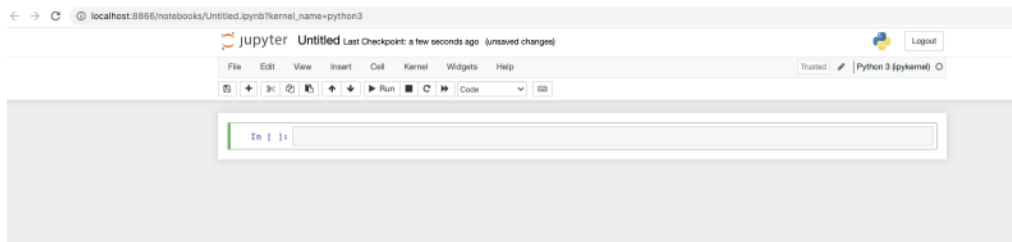
```
(base) AMBP:docker_studi assansanogo$ docker run -it -p 8866:8866 jup3_img
[I 20:26:53.891 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
[I 20:26:53.420 NotebookApp] Serving notebooks from local directory: /
[I 20:26:53.421 NotebookApp] Jupyter Notebook 6.4.0 is running at:
[I 20:26:53.421 NotebookApp] http://66b03e9a97de:8866/?token=478f8176af3de0b1f4d7ebc9a74e7d47e8f5dfce960b74c4
[I 20:26:53.421 NotebookApp] or http://127.0.0.1:8866/?token=478f8176af3de0b1f4d7ebc9a74e7d47e8f5dfce960b74c4
[I 20:26:53.421 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:26:53.425 NotebookApp]

To access the notebook, open this file in a browser:
file:///root/.local/share/jupyter/runtime/nbserver-1-open.html
Or copy and paste one of these URLs:
http://66b03e9a97de:8866/?token=478f8176af3de0b1f4d7ebc9a74e7d47e8f5dfce960b74c4
or http://127.0.0.1:8866/?token=478f8176af3de0b1f4d7ebc9a74e7d47e8f5dfce960b74c4
```

Vous reportez alors le token dans votre navigateur :

Ici le token fut : 478f8176af3de0b1f4d7ebc9a74e7d47e8f5dfce960b74c4

Vous voici capable d'utiliser Jupyter notebook :



Exercice p. 11 Solution n°5

Question 1

Vous pouvez également définir un volume dans le Dockerfile.

Si vous ne créez pas ce volume au niveau du dockerfile, vous devez exécuter la commande :

docker run -v <hostfolder>:/var/www/html

```
1 # volume dans le container
2 VOLUME /var/www/html
3
```

☒ Vrai

☐ Faux

☒ Vous pouvez déclarer le mapping de port au démarrage du container.

Question 2

Dans la commande CMD, app.py est le paramètre qui sera utilisé par Entrypoint.

```
1 FROM ubuntu:16.04
2
3 RUN apt-get update -y && \
4     apt-get install -y python-pip python-dev
5
6 COPY ./requirements.txt /app/requirements.txt
7
8 WORKDIR /app
9
10 RUN pip install -r requirements.txt
11
12 COPY . /app
13
14 ENTRYPOINT [ "python" ]
15
16 CMD [ "app.py" ]
17
```

☒ Vrai

☐ Faux

☒ Dans la commande CMD, app.py est le paramètre qui sera utilisé par Entrypoint. Il peut être remplacé par une autre valeur à l'exécution du container.

Question 3

Vous pouvez exécuter la commande CMD sans Entrypoint.

```
1 FROM ubuntu:16.04
2
3 RUN apt-get update -y && \
4     apt-get install -y python-pip python-dev
5
6 COPY ./requirements.txt /app/requirements.txt
7
8 WORKDIR /app
9
10 RUN pip install -r requirements.txt
11
12 COPY . /app
13
14 CMD [ "python" , "app.py" ]
15
```

☒ Vrai

☐ Faux

 Vous pouvez exécuter **CMD "python app.py" ou CMD ["python" , "app.py"]**

Question 4

Vous pouvez optimiser l'écriture :


```
1 FROM ubuntu
2 RUN apt-get update
3 RUN apt-get install -y apache2
4 RUN apt-get install -y apache2-utils
5 RUN apt-get clean
6 EXPOSE 80
7 CMD ["apache2ctl", "-D", "FOREGROUND"]
8
```

Comme ci-dessous :

```
1 FROM ubuntu
2 RUN apt-get update
3 RUN apt-get install -y apache2 && \
4     apt-get install -y apache2-utils && \
5     apt-get clean
6 EXPOSE 80
7 CMD ["apache2ctl", "-D", "FOREGROUND"]
8
```

☒ Vrai

☐ Faux

 Vous devez éviter d'ajouter trop de commandes RUN. Chaque commande run crée une image intermédiaire.

Question 5

Ces lignes de commandes sont-elles correctes ?

```
1 # construire l'image
2 docker build -t first-dockerfile -f Dockerfile1 .
3 # lister les images
4 docker images
5 # exécuter le container associé à l'image
6 docker run -d --name first-container first-dockerfile
7 # use exec for interaction avec le container dont l'id est : f1edbfca3eac
8 docker exec -it f1edbfca3eac bash
9
```

☒ Vrai

☐ Faux



- Dans la première commande, vous spécifiez le nom de fichier (Dockerfile)
- Dans la seconde commande, vous listez les images
- Dans la troisième commande, vous exécutez le container en mode interactif
- Dans la dernière commande, vous pouvez interagir avec le container dont l'id est spécifié