ISTIC	**Version** : 1 **Date** : 12/14/2022	**Elaboré et enseigné par** : Mme Wided ABIDI **Etudiant** : Yahya Abulhaj – IOT 3A	جامعة قرطاج Université de Carthage

Travaux Pratiques Cloud Computing N°3
Création de vos propres images

Objectifs :

L'objectif principal de ce TP est de se familiariser avec la création de nos propres images.

Boite à outils :

- Les outils qui seront utilisés tout au long de ce TP sont comme suit : Ubuntu desktop.
- Disposer des droits d'administration.
- Disposer d'une connexion à Internet configurée et activée.

Partie I : Création des images en utilisant le Dockerfile

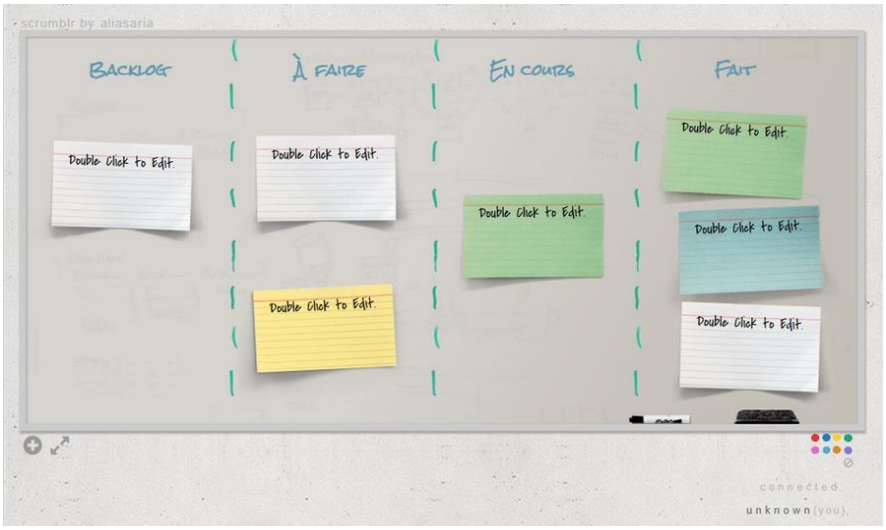
Un **Dockerfile** est un fichier texte avec lequel vous allez pouvoir donner à Docker les instructions nécessaires pour pouvoir créer une image. C'est un peut comme la recette de cuisine d'un plat pour un cuisinier.

Pour commencer il faut bien comprendre comment se rédige un Dockerfile, il y a une syntaxe à respectée avec les instructions suivantes:

- **FROM** permet de définir l'image source
- **LABEL** permet d'ajouter des métadonnées à une image
- **RUN** permet d'exécuter des commandes dans votre conteneur
- **ADD** permet d'ajouter des fichiers dans votre conteneur
- **WORKDIR** permet de définir votre répertoire de travail
- **EXPOSE** permet de définir les ports d'écoute par défaut
- **VOLUME** permet de définir les volumes utilisables
- **CMD** permet de définir la commande par défaut lors de l'exécution de vos conteneurs Docker.

Rédiger son premier Dockerfile

Nous considérons l'exemple suivant : Construire une image pour le logiciel ***SCRUMBLR*** qui permet d'afficher un tableau de suivis des tâches modifiables en temps réel. Ce logiciel utilise ***Redis*** et ***NodeJS*** pour fonctionner. Le travail consiste à créer une image Docker de celui-ci pour obtenir le résultat suivant:



Un fichier **Dockerfile** est toujours rédigé de la même manière. On commence par mettre l'image qui va nous servir de base de travail puis on effectue des actions sur cette image. Pour trouver l'image qui correspond à notre besoin, il faut toujours partir d'une image officielle. Ces images sont régulièrement mises à jour et sont certifiées par Docker.

Les pré-requis :

- Un hôte avec Docker installé ;
- Créer un répertoire de travail pour ce projet ;
- Créer un fichier **Dockerfile** dans le dossier du projet.

L'image de base : Nous allons partir avec une image Ubuntu 18.04 qui est une image officielle.

```
1  FROM ubuntu:18.04
2  LABEL maintainer="kevin@linit.io"
```

Mon Essai :

Dockerfile 1 ●

Dockerfile > ...

```
1   FROM ubunto:18.04
2   LABEL maintainer="Yahya@Itzyahya.tech"
3
```

- Par la directive **FROM** suivi du nom de l'image et sa version, nous déclarons l'image que utilisée dans notre Dockerfile.
- La directive **LABEL** permet d'ajouter des informations à l'image dans notre cas l'adresse mail du mainteneur de l'image.

Installation des pré-requis

```
1  # Installation des prérequis
2  RUN apt-get update && apt-get install -y wget sudo supervisor git redis && \
3      mkdir -p /var/log/supervisor && \
4      mkdir -p /etc/supervisor/conf.d
```

Mon Essai :

```
4
5  RUN apt-get update && apt-get install -y wget sudo supervisor git redis && \
6      mkdir -p /var/log/supervisor && \
7      mkdir -p /etc/supervisor/conf.d
```

Nous commençons la modification de l'image de base. Pour effectuer une commande on utilise la directive **RUN** qui exécute uniquement la commande déclarée après la directive.

Dans cette étape, nous installons les différents pré-requis pour le bon fonctionnement de **SCRUMBLR** à savoir:

- Mise à jour des dépôts

- Installation de paquets (wget, sudo, supervisor, git et redis)
- Création du dossier supervisor dans /var/log/
- Création du dossier conf.d dans /etc/supervisor/

Note : Pour exécuter plusieurs commandes à la suite, il faut les imbriquées les unes à la suite des autres avec « **&&** » qui permet en code bash d'exécuter des commandes en série et « \ » qui permet de faire un retour à la ligne pour y voir plus clair dans le code.

Pour l'étape suivante, nous allons fournir le fichier de configuration du service supervisor à Docker pour qu'il le copie dans notre image.

Pour commencer, créer un fichier supervisor.conf dans le répertoire où est votre Dockerfile et ajouter le contenu suivant.

```
[supervisord]
nodaemon=true
[program:redis-server]
command=redis-server
autostart=true
autorestart=true
user=root
stdout_logfile=/var/log/redis/stdout.log
stderr_logfile=/var/log/redis/stderr.log
[program:scrumblr]
command=node server.js --port 80
autostart=true
autorestart=true
user=root
stdout_logfile=/var/log/supervisor/scrumblr.log
stderr_logfile=/var/log/supervisor/scrumblr_err.log
```

Puis, indiquer à Docker avec la directive **ADD** d'ajouter notre fichier supervisor.conf dans le dossier **/etc/** de notre image.

```
ADD supervisor.conf /etc/supervisor.conf
```

Mon essai :

```
8
9
10  ADD supervisor.conf /etc/supervisor.conf
11
```

Installation de NodeJS

Nous allons installer dans notre image NodeJS qui va nous permettre de compiler et exécuter les sources de SCRUMBLR. Nous allons utiliser la directive **RUN** pour exécuter les actions suivantes.

- Téléchargement du script d'installation de NodeJS
- Exécution du script téléchargé
- Installation du paquet NodeJS

```
# Installation de NodeJS
RUN wget -qO- https://deb.nodesource.com/setup_10.x | sudo -E bash - && \
    apt-get install -y nodejs
```

Mon essai
w/ installing -g pm2 as required.

```
RUN wget -qO- https://deb.nodesource.com/setup_10.x | sudo -E bash - && \
    apt-get install -y nodejs && \
    npm install -g pm2
```

Installation de SCRUMBLR

Nous pouvons maintenant installer l'application **SCRUMBLR** dans notre image toujours en utilisant la directive **RUN** pour réaliser les actions suivantes:

- Clone des sources de SCRUMBLR depuis le dépot GitHub du projet
- Déplacement dans le dossier /scrumblr
- Installation de SCRUMBLR via npm

```
# Installation de SCRUMBLR
RUN git clone https://github.com/aliasaria/scrumblr.git && \
    cd scrumblr && \
    npm install
```

```
RUN git clone https://github.com/aliasaria/scrumblr.git && \
    cd scrumblr && \
     npm install && \
     npm run build
```

Modification du fichier config.js

Pour la dernière étape de configuration nous devons modifier une ligne du fichier **« config.js »** afin de permettre à l'application de pouvoir se connectée à la base de donnée redis.

La ligne suivante qui utilise une nouvelle fois la directive **RUN** utilise la commande « **sed** » afin de replacer la chaine de caractère **127.0.0.1:6379** en **redis://127.0.0.1:6379**.

```
# Modification du fichier config.js
RUN sed -i -e "s/127.0.0.1:6379/redis:\/\/127.0.0.1:6379/g" /scrumblr/config.js
```

```
RUN sed -i -e "s/127.0.0.1:6379/redis:\/\/127.0.0.1:6379/g" /scrumblr/config.js
```

Finalisation du Dockerfile

Pour finir, nous allons fournir à Docker les instructions nécessaires afin d'exécuter notre image dans un conteneur.

- **WORKDIR** indique le répertoire de travail du conteneur ici /scrumblr
- **EXPOSE** indique le port de communication dans notre cas le 80
- **STOPSIGNAL** indique le signal système qui arrête le conteneur
- **CMD** indique la commande à exécuter au démarrage du conteneur

```
WORKDIR /scrumblr
EXPOSE 80
STOPSIGNAL SIGTERM
CMD ["supervisord", "-c", "/etc/supervisor.conf"]
```

Mon Essai :

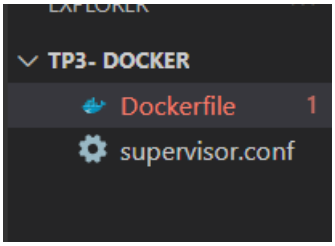
```
RUN sed -i -e "s/127.0.0.1:6379/redis:\/\/127.0.0.1:6379/g" /scrumblr/config.js
WORKDIR /scrumblr
EXPOSE 80
STOPSIGNAL SIGTERM
CMD ["supervisor","-c","/superviosr.conf"]
```

Note : Si la commande dans la directive **CMD** est composée de plusieurs champs il faut impérativement respecter la syntaxe suivante:

[« champs1 », « champs2 », « champs3 », « champs…. »]

Notre **Dockerfile** est maintenant terminé et devrait ressembler à ceci:

```
FROM ubuntu:18.04
LABEL maintainer="kevin@linit.io"
# Installation des prérequis
RUN apt-get update && apt-get install -y wget sudo supervisor git redis && \
    mkdir -p /var/log/supervisor && \
    mkdir -p /etc/supervisor/conf.d
ADD supervisor.conf /etc/supervisor.conf
# Installation de NodeJS
RUN wget -qO- https://deb.nodesource.com/setup_10.x | sudo -E bash - && \
    apt-get install -y nodejs
# Installation de SCRUMBLR
RUN git clone https://github.com/aliasaria/scrumblr.git && \
    cd scrumblr && \
    npm install
# Modification du fichier config.js
RUN sed -i -e "s/127.0.0.1:6379/redis:\/\/127.0.0.1:6379/g" /scrumblr/config.js
WORKDIR /scrumblr
EXPOSE 80
STOPSIGNAL SIGTERM
CMD ["supervisord", "-c", "/etc/supervisor.conf"]
```



Dockerfile

Dockerfile > WORKDIR

```
FROM ubunto:18.04
LABEL maintainer="Yahya@Itzyahya.tech"


RUN apt-get update && apt-get install -y wget sudo supervisor git redis && \
    mkdir -p /var/log/supervisor && \
    mkdir -p /etc/supervisor/conf.d


ADD supervisor.conf /etc/supervisor.conf

RUN wget -qO- https://deb.nodesource.com/setup_10.x | sudo -E bash - && \
    apt-get install -y nodejs && \
    npm install -g pm2


RUN git clone https://github.com/aliasaria/scrumblr.git && \
      cd scrumblr && \
       npm install && \
       npm run build

RUN sed -i -e "s/127.0.0.1:6379/redis:\/\/127.0.0.1:6379/g" /scrumblr/config.js
WORKDIR /scrumblr
EXPOSE 80
STOPSIGNAL SIGTERM
CMD ["supervisor","-c","/superviosr.conf"]
```

Construction et lancement de notre image

Il ne nous reste plus qu'à construire notre image en la nommant correctement. Pour faire ceci déplacer vous dans le répertoire où est contenu votre **Dockerfile** et le ficher **supervisor.conf** et lancer la commande suivante:

```
docker build -t scrumblr .
```

L'argument -t permet de spécifier un tag à notre image.

```
TP3- DOCKER> docker build -t  scrumblr .
```

Success. 😊