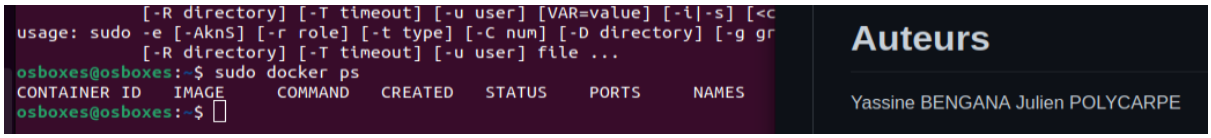
Technologie des conteneurs

TP1 : Mise en route

Dans ce TP vous allez manipuler Docker, et écrire les Dockerfiles nécessaires aux microservices de votre application.

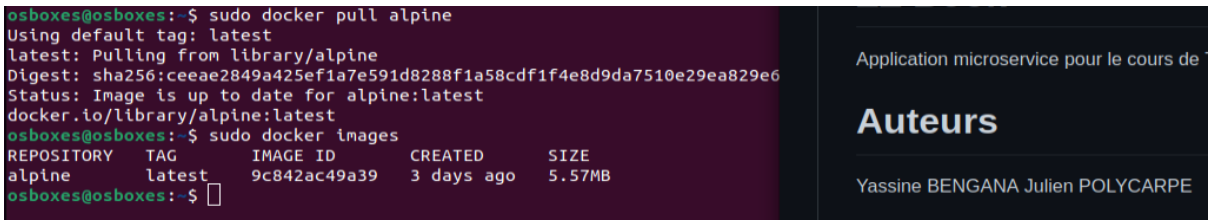
Partie 1 : Docker

Q1 / Installez docker sur votre machine : https://docs.docker.com/install/
Validez l'installation en exécutant la commande *docker ps*



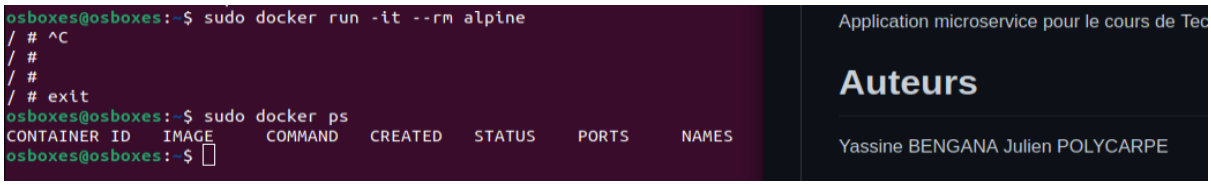
Q2 / Récupérez l'image officielle "alpine" sur votre machine, quelle commande utilisez-vous ? Listez les images afin de vérifier qu'alpine est bien présente

docker pull alpine
docker images



Q3 / Démarrez un conteneur alpine en mode interactif. Vérifiez que vous pouvez utiliser les commandes standards. Quittez votre conteneur (*exit*), que se passe t'il si vous faites *docker ps* ?

docker run -it --rm alpine
docker ps -> conteneur éteint



Q4 / Démarrez un conteneur mariadb, appelez le "mariadbtest". Que faut-il ajouter à la ligne de commande Docker pour démarrer la base correctement ? Utilisez *docker ps* pour vérifier que votre conteneur est en cours d'exécution.

Technologie des conteneurs

```
[root@patient14 student]# docker run --detach --name mariadbtest --env MARIADB_USER=root --env MARIADB_PASSWORD=root --env MARIADB_ROOT_PASSWORD=root  mariadb
b2f9567bfb8d41703b13e750759cea05dfeda5c5f3b58e2f59befcd026eca70e
[root@patient14 student]#
[root@patient14 student]#
[root@patient14 student]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
b2f9567bfb8d        mariadb             "docker-entrypoint..."   2 seconds ago       Up 2 seconds        3306/tcp            mariadbtest
[root@patient14 student]#
```

docker run --detach --name mariadbtest --env MARIADB_USER=root --env MARIADB_PASSWORD=root --env MARIADB_ROOT_PASSWORD=root mariadb b2f9567bfb8d41703b13e750759cea05dfeda5c5f3b58e2f59befcd026eca70e

Q5 / Quelle commande utiliser pour obtenir un shell dans le conteneur “mariadbtest” ?

docker exec -it mariadbtest /bin/bash

Q6 / Exécutez la commande “$ docker top mariadbtest”, comparez le résultat avec la commande ps de votre machine (ex. “$ ps -aux”). Que constatez-vous ?

```
[root@patient14 student]# docker top mariadbtest
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
polkitd             27545               27524               0                   15:55               ?                   00:00:00            mariadbd
```

```
[root@patient14 student]# ps -aux | grep docker
root     25365  0.9  0.1 1196688 40776 ?       Ssl  15:40   0:09 /usr/bin/dockerd-current --add-runtime docker-runc=/usr/libexec/docker/docker-runc-current --default-runtime=docker-runc
 --exec-opt native.cgroupdriver=systemd --userland-proxy-path=/usr/libexec/docker/docker-proxy-current --init-path=/usr/libexec/docker/docker-init-current --seccomp-profile=/etc/docker/
seccomp.json --selinux-enabled --log-driver=journald --signature-verification=false --storage-driver overlay2
root     25376  0.1  0.0 534844 17696 ?        Ssl  15:40   0:01 /usr/bin/docker-containerd-current -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --metrics-interval=0 -
-start-timeout 2m --state-dir /var/run/docker/libcontainerd/containerd --shim docker-containerd-shim --runtime docker-runc --runtime-args --systemd-cgroup=true
root     27508  0.0  0.0 248568 13364 pts/0    Sl+  15:55   0:00 /usr/bin/docker-current run --name mariadbtest --env MARIADB_USER=aaaa --env MARIADB_PASSWORD=aaaa --env MARIADB_ROOT_PA
SSWORD=aaaa mariadb
root     27524  0.0  0.0 349960  5008 ?        Sl   15:55   0:00 /usr/bin/docker-containerd-shim-current a1f4062197dc5e1b7502e8cedce8814006a55c1a5a2d802b90b23495214a0836 /var/run/docker
/libcontainerd/a1f4062197dc5e1b7502e8cedce8814006a55c1a5a2d802b90b23495214a0836 /usr/libexec/docker/docker-runc-current
root     27947  0.0  0.0 112836   980 pts/1    S+   15:58   0:00 grep --color=auto docker
[root@patient14 student]#
```

les PID sont différents grâce à l’isolation fournie par Docker

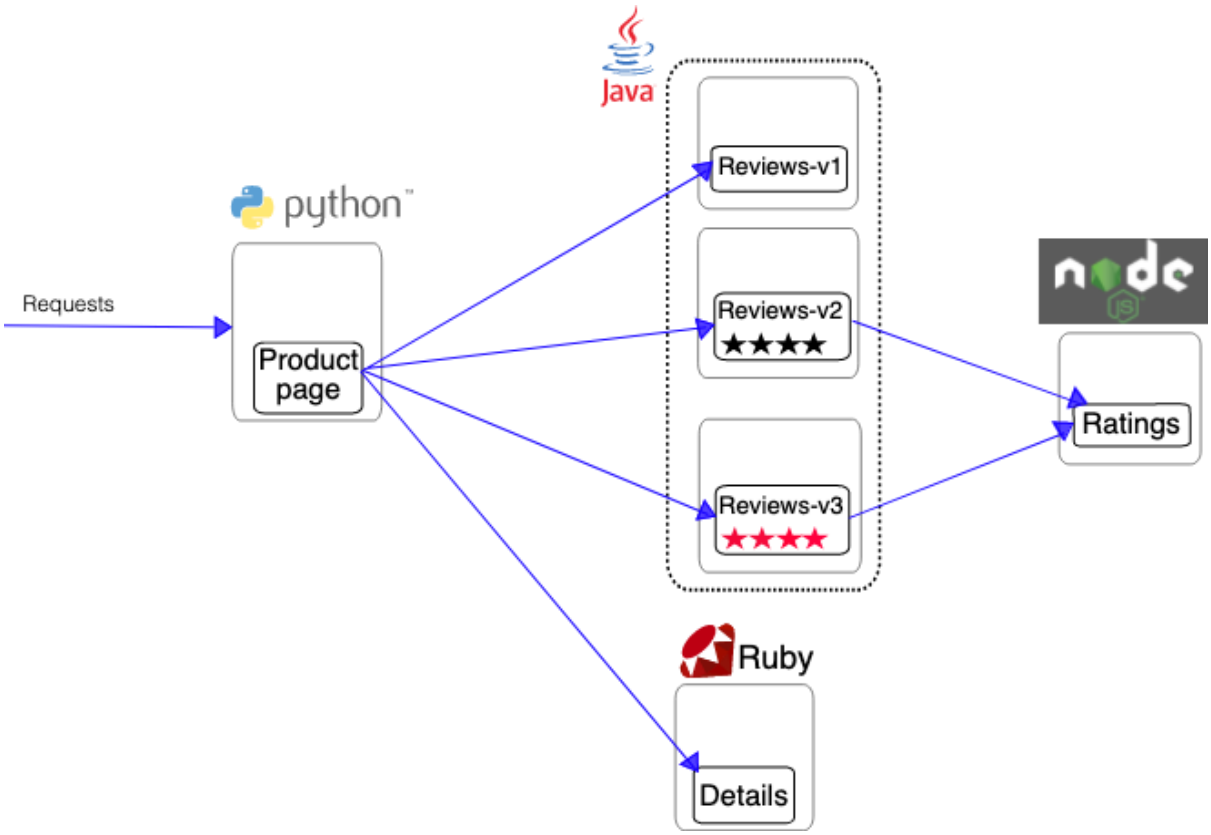
Q7 / Démarrez un conteneur, en respectant les spécifications suivantes :

- **Image : dockersamples/static-site**
- **Nom du conteneur : my-site**
- **Variables d'environnement : AUTHOR="You"**
- **Le port 80 du conteneur doit être accessible sur votre machine (au port de votre choix).**

docker run --name my-site --env AUTHOR=you -p 8080:80 dockersamples/static-site

Technologie des conteneurs

Partie 2 : Projet en groupe



Q1 / Initialiser un dépôt Git sur votre machine et le pusher sur Github/Gitlab.

Q2 / Récupérez les sources des microservices (fichier zz-book.zip) et décompressez-les dans le dossier de votre projet. Éditez le fichier README.md en ajoutant la liste des membres de votre équipe ainsi que vos mails.

Q3 / Écrivez le Dockerfile pour vos microservices (details, productpage et ratings). Le Dockerfile de reviews est déjà présent. Utilisez des images officielles. Aidez-vous des informations ci-dessous :

- details
  - Utilisez ruby 2.7.1
  - Définissez deux variables d'environnement
    - SERVICE_VERSION (valeur par défaut v1)
    - ENABLE_EXTERNAL_BOOK_SERVICE (valeur par défaut false)
  - Faites en sorte que ces deux variables soient personnalisables lors du docker build
  - Le dossier de travail est /opt/microservices (là ou vous copierez les sources)
  - Pour exécuter votre service, la commande est "ruby details.rb 9080"
  - N'oubliez pas d'indiquer le port d'écoute
- productpage
  - Utilisez python 3.7.7

    - Définisez la variable FLOOD_FACTOR (valeur par défaut 0, modifiable lors du build Docker)
    - Installez en premier les dépendances du code python, utilisez la commande “pip install –no-cache-dir -r <fichier-requirements.txt>”.
    - Copiez le code restant dans /opt/microservices
    - La commande de démarrage est python productpage.py 9080
- ratings
    - Utilisez node 12.18.1
    - Installez les dépendances via “npm install” et le fichier package.json
    - Copiez le code dans /opt/microservices
    - Ajoutez les dépendances systèmes grâce à la commande “apt-get update && apt-get install curl --no-install-recommends -y && rm -rf /var/lib/apt/lists/*”
    - Ajoutez la variable SERVICE_VERSION (valeur par défaut v1, éditable au docker build)
    - Commande de démarrage “node ratings.js 9080”

N’oubliez pas de bien choisir l’ordre de vos instructions Dockerfile, afin de bénéficier au maximum du cache de Docker.

Q4 / Buildez vos conteneurs, et tentez de les démarrer un par un pour valider leur fonctionnement. Vous aurez peut-être besoin de démarrer les bases mongodb et mysql, présentes dans src/