



**POLYTECHNIQUE MONTRÉAL**

**LOG3000**

**PROCESSUS DU GÉNIE LOGICIEL**

*Travail Pratique #4*

***INFRASTRUCTURE EN TANT QUE CODE***

<b><i>Chargé</i></b>	<b><i>Courriel</i></b>
<i>Varun Shiri</i>	<a href="mailto:varun.shiri@polymtl.ca"><i>varun.shiri@polymtl.ca</i></a>

***Automne 2022***

## 1. Travail à effectuer

Dans ce TP, vous allez pratiquer l'approche d'infrastructure en tant que code avec Docker. Pour ça, vous allez apprendre à lancer un conteneur Docker (section 3.1), à exécuter des applications Web avec Docker (sections 3.2 et 3.3), et à «dockerize» une application existante (section 3.4).

Les fichiers à remettre sont les suivants :

- Votre rapport en pdf contenant les réponses aux questions posées dans la section 4. Incluez des captures d'écran pour appuyer vos explications, ainsi que la sortie de toutes les commandes exécutées et le résultat de l'application.
- Le code développé conformément à ce qui est requis dans la section 3.4.

La remise doit se faire sur Moodle avant la date suivante.

Groupe 01 et 02	16 novembre 2022 à 23 h 55
-----------------	----------------------------

Il y aura une pénalité pour les travaux remis en retard.

## 2. Prérequis: Installation et compréhension de Docker




Avant de commencer les tâches, il faut d'abord compléter la section relative aux prérequis.

Tout d'abord, lisez ce document pour comprendre Docker: [Docker overview](#)

### 2.1 Installation de Docker

Notez que vous pouvez sauter cette étape si le Docker est déjà installé sur votre ordinateur.

Choisissez la version de docker appropriée en fonction de votre système d'exploitation:

- Mac: <https://docs.docker.com/docker-for-mac/>
  - Vidéo:  Docker Desktop for macOS Setup and Tips
- Linux: <https://docs.docker.com/engine/install/ubuntu/> (ou choisissez votre distro dans la barre latérale)
- Windows: <https://docs.docker.com/docker-for-windows/>
  - Vidéos:  How to Install Docker Desktop on Windows 11 ,  
 WSL 2 With Docker Getting Started and Docker Desktop Installation

## 2.2 Comprendre Docker

Une fois le Docker installé, dans votre terminal, lancez les commandes suivantes.

```
$ docker help
```

Cette commande affiche toutes les commandes et options de Docker.

Testez votre Docker en exécutant la commande suivante.

```
$ docker run hello-world
```

Voici le résultat:

```
varun@Varun-PC:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:e18f0a777aefabe047a671ab3ec3eed05414477c951ab1a6f352a06974245fe7
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Si vous souhaitez toujours en savoir plus sur docker ou si vous avez encore des questions, consultez les ressources des annexes A et B.

## 3. TP4

### 3.1 Lancer votre premier conteneur

Maintenant que tout est configuré, il est temps d'apprendre comment lancer un conteneur Docker. Pour cela, vous allez exécuter un conteneur [Alpine Linux](#) (une distribution allégée de Linux) sur votre système.

1. Tout d'abord, exécutez la commande suivante sur le terminal: **docker pull alpine**

*Remarque:* selon la façon dont vous avez installé le docker sur votre système, vous pouvez voir une erreur d'autorisation refusée après l'exécution de la commande ci-dessus. Si vous êtes sous Linux, vous devrez peut-être préfixer vos commandes docker avec *sudo*. Vous pouvez également créer un [docker group](#) pour régler ce problème.

2. Ensuite, utilisez la commande **docker images** pour afficher une liste de toutes les images de votre système.

3. Exécutez maintenant un conteneur Docker basé sur cette image. Pour ça, vous allez utiliser la commande **docker run alpine ls -l**.
4. Maintenant, essayez la commande **docker run alpine echo "hello from alpine"**.
5. Essayez la commande **docker ps** pour voir tous les conteneurs en cours d'exécution. Puisqu'aucun conteneur n'est en cours d'exécution, vous devriez voir une ligne vide. Essayez une variante plus utile: **docker ps -a**. Ce que vous verrez est une liste de tous les conteneurs que vous avez exécutés.
6. Ajoutez une capture d'écran des commandes exécutées et des sorties à votre rapport.

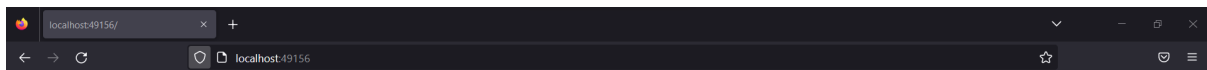
Maintenant, vous devez être familiarisé avec l'exécution d'un conteneur avec **docker run**. C'est probablement la commande que vous utiliserez le plus souvent. Si vous souhaitez en savoir plus sur l'exécution de docker, utilisez **docker run --help** pour voir une liste de tous les indicateurs pris en charge.

### 3.2 Exécuter une application Web statique avec Docker

Ensuite, vous allez utiliser Docker pour exécuter un site Web statique dans un conteneur. Le site Web est basé sur une image existante. Vous allez extraire une image Docker de Docker Store, et exécuter le conteneur.

1. Vous pouvez télécharger l'image du site Web avec la commande suivante:  
**docker run -d dockersamples/static-site**  
Étant donné que l'image static-site n'existe pas sur votre Docker host, le Docker daemon la récupère dans le registry, puis l'exécute en tant que conteneur. Le serveur est en cours d'exécution, mais vous ne pourrez pas encore voir le site Web, car on ne connaît pas encore le port du serveur.
2. Arrêtez le conteneur. Pour ce faire, vous devez trouver l'ID du conteneur avec la commande **docker ps**. Consultez la colonne CONTAINER ID.
3. Maintenant, exécutez les commandes: **docker stop <CONTAINER ID>** et **docker rm <CONTAINER ID>** en utilisant le CONTAINER ID que vous avez noté à la dernière étape.

4. Exécutez la commande: **`docker run --name static-site -e AUTHOR="<YOUR_NAME>" -d -P dockersamples/static-site`**
5. Exécutez la commande: **`docker port static-site`**. Cette commande vous indiquera le port que le serveur exécute.
6. Vous pouvez maintenant consulter le site Web au **`http://localhost:<YOUR_PORT_FOR_TCP>`**, comme par exemple `http://localhost:55001/`. Vous devriez voir sur votre navigateur le site Web qui est similaire à la capture d'écran ci-dessous.
7. Ajoutez une capture d'écran des commandes exécutées et des sorties à votre rapport.



**Hello beautiful person!**

This is being served from a **docker** container running Nginx.

---

### 3.3 Exécuter une application Web dynamique avec docker

Dans la section précédente, vous avez extrait l'image static-site du registry et demandé au client Docker d'exécuter un conteneur basé sur cette image. Pour afficher la liste des images disponibles localement sur votre système, exécutez la commande **`docker images`**.

Le prochain objectif est de créer une image avec l'application [Flask](#). À titre d'exemple, nous allons créer une application Python Flask qui affiche un chat aléatoire .gif à chaque fois que l'application est chargée.

1. Commencez par créer un répertoire *flask-app* avec **mkdir flask-app**. Accédez au dossier avec **cd flask-app**. Vous allez ajouter 4 fichiers au dossier: *app.py*, *requirements.txt*, *templates/index.html*, *Dockerfile*.
2. Créez *app.py* avec le contenu suivant. Lisez le fichier et assurez-vous de bien comprendre le contenu.

```
from flask import Flask, render_template
import random

app = Flask(__name__)

# List of cat images

images = [
    "https://media.giphy.com/media/BzyTuYcmvSORqs1ABM/giphy.gif",
    "https://media.giphy.com/media/xUPGcyi4YxcZp8dWZq/giphy.gif",
    "https://media.giphy.com/media/1iu8uG2cjYFZS6wTxv/giphy.gif",
    "https://media.giphy.com/media/mlvseq9yvZhba/giphy.gif",
    "https://media.giphy.com/media/lJNoBCvQYp7nq/giphy.gif"
]

@app.route('/')
def index():
    url = random.choice(images)
    return render_template('index.html', url=url)

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

3. Afin d'installer les modules Python requis pour l'application, vous devez créer un fichier appelé *requirements.txt* et ajouter la ligne suivante à ce fichier:

```
Flask==0.10.1
```

4. Créez *templates/index.html* avec le contenu suivant:

```
<html>
  <head>
    <style type="text/css">
      body {
        background: black;
        color: white;
      }
      div.container {
        max-width: 500px;
```

```

        margin: 100px auto;
        border: 20px solid white;
        padding: 10px;
        text-align: center;
    }
    h4 {
        text-transform: uppercase;
    }
</style>
</head>
<body>
    <div class="container">
        <h4>Cat Gif of the day</h4>
        
    </div>
</body>
</html>

```

5. Écrivez un *Dockerfile*: Vous allez créer une image Docker avec cette application Web. Toutes les images utilisateur sont basées sur une image de base. Puisque l'application est écrite en Python, vous devriez construire votre propre image Python basée sur Alpine. Vous pouvez le faire en utilisant un *Dockerfile*. Créez un fichier appelé *Dockerfile* et ajoutez le code ci-dessous. Lisez le fichier et assurez-vous de bien comprendre le contenu.

```

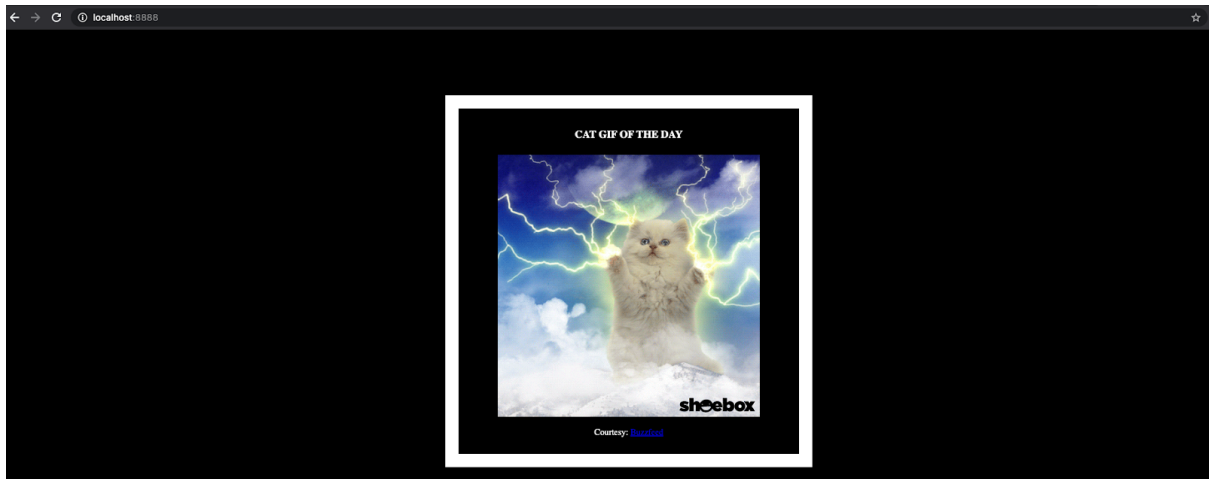
FROM alpine:3.5
RUN apk add --update py2-pip
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/
EXPOSE 5000
CMD ["python", "/usr/src/app/app.py"]

```

6. Maintenant que vous avez créé le *Dockerfile*, vous pouvez créer votre image. Exécutez la commande ci-dessous. Assurez-vous de remplacer `<YOUR_USERNAME>` par votre nom d'utilisateur que vous avez créé lors de l'inscription sur [Docker Cloud](#). Si vous ne l'avez pas encore fait, veuillez créer un compte.

**`docker build -t <YOUR_USERNAME>/myfirstapp .`** (N'oubliez pas le point à la fin)

7. Exécutez l'image créée avec `docker run -p 8888:5000 --name flask-app <YOUR_USERNAME>/myfirstapp`. Vous pouvez accéder au site Web sur votre navigateur: <http://localhost:8888>. Vous devriez obtenir un gif différent à chaque fois que vous rechargez la page. Voici un exemple de l'application en cours d'exécution.



8. Ajoutez une capture d'écran des commandes exécutées et des sorties sur votre rapport.

### 3.4 «Dockerize» une application

Maintenant que vous savez comment exécuter une application Docker, vous devez «dockerize» une application existante (c.t.d créer une application Docker). Votre travail consiste à **créer une image** du site Web à partir du référentiel git ci-dessous, puis à **exécuter l'image**.

\$ <https://github.com/varunks99/LOG3000-Docker-TP>

Vous devez créer l'image du site Web avec le serveur Web NGINX, qui est l'un des serveurs Web les plus utilisés pour les sites Web statiques: [How To Use the Official NGINX Docker Image](#)



## 4. Questions

***Lorsque possible, vous devez appuyer vos réponses avec des captures d'écran des commandes exécutées ainsi que de la sortie.***

### 4.1 Questions d'analyse sur la section 3.1

1. Expliquez avec vos propres mots ce que fait la commande **pull alpine**.
2. Expliquez avec vos propres mots ce qui se passe derrière l'écran lorsque vous exécutez la commande **docker run**.
3. Expliquez avec vos propres mots ce qui se passe derrière l'écran lorsque vous exécutez la commande **docker run alpine echo "hello from alpine"** S'agit-il d'une sortie de Docker ou de Linux Alpine?
4. Quelle est la différence entre une image et un conteneur?
5. Quels sont les avantages d'un conteneur par rapport à une machine virtuelle?

### 4.2 Questions d'analyse sur la section 3.2

6. Expliquez avec vos propres mots chaque paramètre utilisé à l'étape 4 de la section 3.2.

### 4.3 Questions d'analyse sur la section 3.3

7. Expliquez la sortie de la commande **docker images**. Comment obtenir une version spécifique d'une image?
8. Expliquez avec vos propres mots la différence entre *base images* et *child images*.
9. Expliquez avec vos propres mots la différence entre *official images* et *user images*.
10. Expliquez avec vos propres mots ce qu'est un Dockerfile.
11. Expliquez chaque ligne du Dockerfile créé à l'étape 5 de la section 3.3.
12. Expliquez ce qui se passe lorsque vous exécutez la commande **docker build -t <YOUR\_USERNAME>/flask-app**.

### 4.4 Questions d'analyse sur la section 3.4

13. Décrivez chaque ligne du Dockerfile que vous avez créé. Ajoutez des captures d'écran et la sortie des commandes pour appuyer votre explication. Ajoutez également une image de votre navigateur montrant l'URL et le site Web en cours d'exécution.

N'oubliez pas de soumettre le code source que vous avez développé.

## 4.5 Question de rétroaction

Nous travaillons à l'amélioration continue des travaux pratiques de LOG3000. Cette question peut être répondue très brièvement.

1. Combien de temps avez-vous passé au travail pratique, en heures-personnes, en sachant que deux personnes travaillant pendant trois heures correspondent à six heures-personnes. Est-ce que l'effort demandé pour ce laboratoire est adéquat ?
2. Quelles difficultés avez-vous rencontré lors de ce laboratoire?

## Annexe A: Docker

Voici quelques commandes utiles pour utiliser docker et docker-compose.

- Créer une image avec le répertoire courant: `docker build .`
- Lister les images: `docker images`
- Lancer un conteneur: `docker run -d IMAGE`
- Lister les conteneurs actifs: `docker ps`
- Se connecter à un conteneur actif: `docker run -it CONTAINER /bin/sh`
- Clean et build un stack: `docker-compose build --no-cache`
- Lancer un stack: `docker-compose up`
- Supprimer des conteneurs, des images, des volumes et des réseaux Docker: `docker system prune`
  - **Attention!** Supprimer tous les objets non utilisés. La commande d'élagage du système docker supprimera tous les conteneurs arrêtés, toutes les images pendantes et tous les réseaux inutilisés:
- Arrêtez et retirez tous les conteneurs:

```
$ docker container stop $(docker container ls -aq)
```

```
$ docker container rm $(docker container ls -aq)
```

- Supprimer toutes les images inutilisées

Pour supprimer toutes les images qui ne sont référencées par aucun conteneur existant, pas seulement celles qui sont pendantes, utilisez la commande prune avec l'option -a:

```
$ docker image prune -a
```

## Annexe B: Tutoriels et références

Docker for beginners: <https://www.youtube.com/watch?v=fqMOX6JJhGo>

**Dockerfiles:**

- <https://docs.docker.com/get-started/part2/>
- <https://docs.docker.com/engine/reference/builder/#from>
- <https://www.digitalocean.com/community/tutorials/docker-explained-using-dockerfiles-to-automate-building-of-images>

**Docker-compose:**

- <https://docs.docker.com/get-started/part3/#introduction>
- <https://docs.docker.com/compose/compose-file/>
- <https://docs.docker.com/compose/gettingstarted/#step-2-create-a-dockerfile>