

LOG3000 – Processus du génie logiciel

Hiver 2025

**Travail Pratique No.4 :
Infrastructure en tant que code**

Groupe 70

2194964 – Emir Tuncbilek, 2220131 – Ait Ameer Sami

Soumis à : Zied Kaabi

21 Mars 2025

Section 4

4.1 Questions d'analyse sur la section 3.1

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
6e771e15690e: Pull complete
Digest: sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef88c
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	a8560b36e8b8	5 weeks ago	12.8MB
hello-world	latest	7e1a4e2d11e2	8 weeks ago	17kB
projetlog2990-angular-app	latest	e907740a33b4	3 months ago	89.9MB
projetlog2990-node-server	latest	fac8fbbdbcfa	3 months ago	2.02GB
opensearchproject/opensearch-dashboards	latest	b46f3913acc5	3 months ago	2.48GB
mongo	latest	4f93a84f7d4d	3 months ago	1.12GB

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker run alpine ls -l
total 56
drwxr-xr-x  2 root    root      4096 Feb 13 23:05 bin
drwxr-xr-x  5 root    root      340  Mar 21 14:51 dev
drwxr-xr-x  1 root    root     4096 Mar 21 14:51 etc
drwxr-xr-x  2 root    root     4096 Feb 13 23:05 home
drwxr-xr-x  6 root    root     4096 Feb 13 23:05 lib
drwxr-xr-x  5 root    root     4096 Feb 13 23:05 media
drwxr-xr-x  2 root    root     4096 Feb 13 23:05 mnt
drwxr-xr-x  2 root    root     4096 Feb 13 23:05 opt
dr-xr-xr-x 217 root    root        0 Mar 21 14:51 proc
drwx----- 2 root    root     4096 Feb 13 23:05 root
drwxr-xr-x  3 root    root     4096 Feb 13 23:05 run
drwxr-xr-x  2 root    root     4096 Feb 13 23:05 sbin
drwxr-xr-x  2 root    root     4096 Feb 13 23:05 srv
dr-xr-xr-x 11 root    root        0 Mar 21 14:51 sys
drwxrwxrwt  2 root    root     4096 Feb 13 23:05 tmp
drwxr-xr-x  7 root    root     4096 Feb 13 23:05 usr
drwxr-xr-x 11 root    root     4096 Feb 13 23:05 var
```

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker run alpine echo "hello from alpine"
hello from alpine
```

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6826a794281e	alpine	"echo 'hello from al..."	28 seconds ago	Exited (0) 27 seconds ago		practical_elbakyan
b2cbe4611a0c	alpine	"ls -l"	46 seconds ago	Exited (0) 45 seconds ago		romantic_black
30416a2417b5	hello-world	"/hello"	21 hours ago	Exited (0) 21 hours ago		keen_tharp

1. Expliquez avec vos propres mots ce que fait la commande pull alpine.

Cette commande télécharge l'image Alpine Linux et l'ajoute à la liste des machines disponibles sur la machine locale.

2. Expliquez avec vos propres mots ce qui se passe derrière l'écran lorsque vous exécutez la commande `docker run`.

Docker commence par vérifier si l'image est présente localement, si elle ne l'est pas Docker la télécharge. Docker crée ensuite un conteneur et lance le processus défini par l'image.

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?

Docker crée un conteneur Linux Alpine dans lequel il exécute le processus `echo "hello from alpine"`. La sortie `"hello from alpine"` provient de Linux Alpine, mais est capturée par Docker qui s'occupe de l'afficher sur mon terminal. Une fois le processus `echo` terminé, le conteneur s'arrête.

4. Quelle est la différence entre une image et un conteneur?

Une image est statique et immuable qui indique comment faire fonctionner une application. Un conteneur, lui, est un environnement isolé dans lequel on exécute un processus qui se base sur une image. Une image correspond donc à une description de comment faire exécuter un processus, un conteneur est une instance ou on exécute un processus en se basant sur cette description de comment faire.

5. Quels sont les avantages d'un conteneur par rapport à une machine virtuelle?

Les conteneurs sont plus légers et portables que les machines virtuelles. Étant donné qu'ils ne contiennent pas un OS entier, les conteneurs utilisent moins de RAM et de CPU. Ainsi, ils démarrent plus rapidement et facilitent un déploiement en continu. De plus, les conteneurs fonctionnent de manière identique sur tous les systèmes qui prennent en charge Docker.

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.

- **docker** run crée et exécute un conteneur
- **--name static-site** donne un nom au conteneur (dans ce cas-ci static-site), sinon ça aurait été un identifiant aléatoire
- **-e AUTHOR=Sami** crée une variable d'environnement AUTHOR à laquelle on attribue la valeur "Sami"
- **-d** lance le conteneur en mode détaché. Sans cela, le terminal resterait indéfiniment bloqué sur l'exécution du conteneur.
- **-P** demande à Docker d'associer les ports du conteneur aux ports disponibles sur la machine hôte.
- **dockersamples/static-site** est l'image utilisée pour créer le conteneur

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker run -d dockersamples/static-site
Unable to find image 'dockersamples/static-site:latest' locally
latest: Pulling from dockersamples/static-site
eff3ab7e9c39: Pull complete
a3ed95caeb02: Pull complete
7b10f03a0309: Pull complete
716f7a5f3082: Pull complete
fdd5d7827f33: Pull complete
Digest: sha256:daa686c61d7d239b7977e72157997489db49f316b9b9af3909d9f10fd28b2dec
Status: Downloaded newer image for dockersamples/static-site:latest
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
5b1c1cff328c247fb0afb9f53bd6736a128cf874aa72cc28d490ad6bbd583569

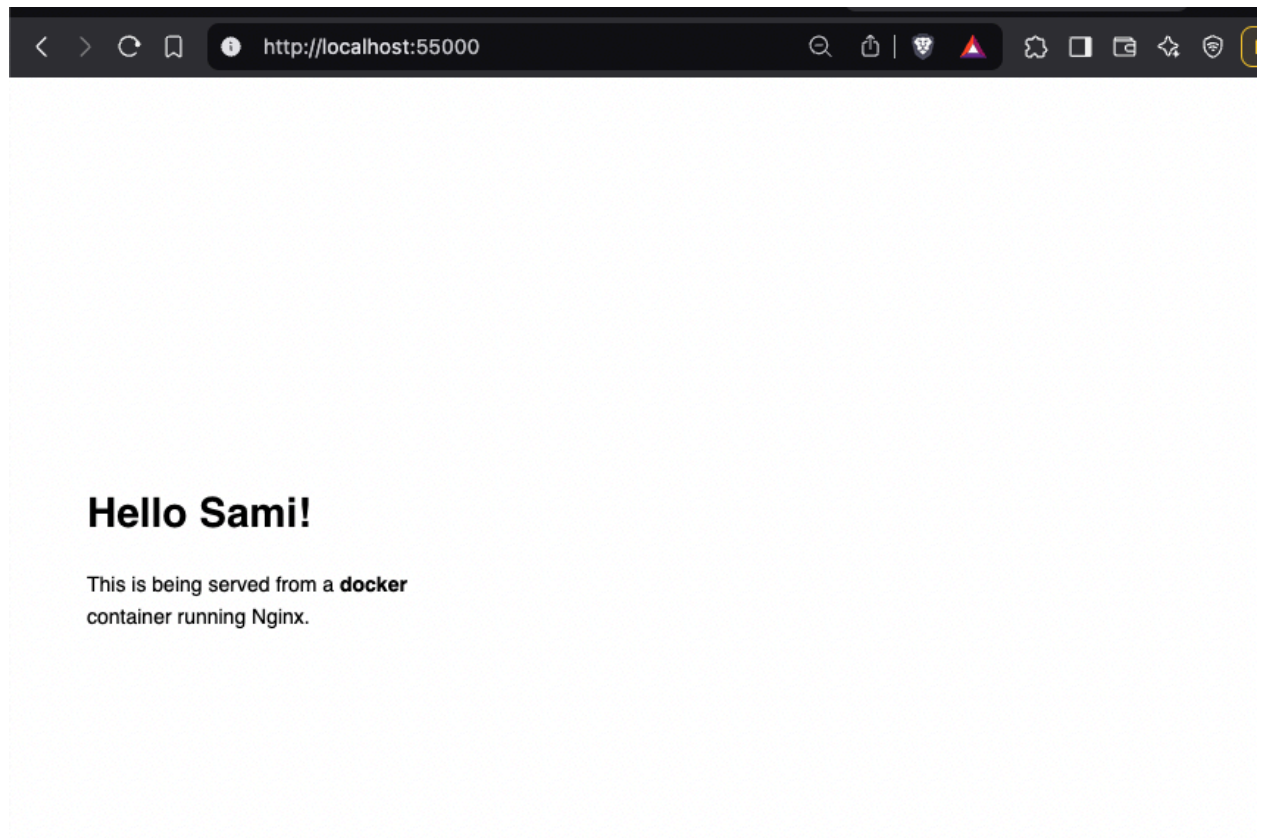
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
5b1c1cff328c   dockersamples/static-site           "/bin/sh -c 'cd /usr_"   About a minute ago    Up About a minute    80/tcp, 443/tcp         sharp_chatelet

(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker stop 5b1c1cff328c
5b1c1cff328c

(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker rm 5b1c1cff328c
5b1c1cff328c

(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker run --name static-site -e AUTHOR=Sami -d -P dockersamples/static-site
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested

(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker port static-site
80/tcp -> 0.0.0.0:55000
443/tcp -> 0.0.0.0:55001
```



4.3 Questions d'analyse sur la section 3.3

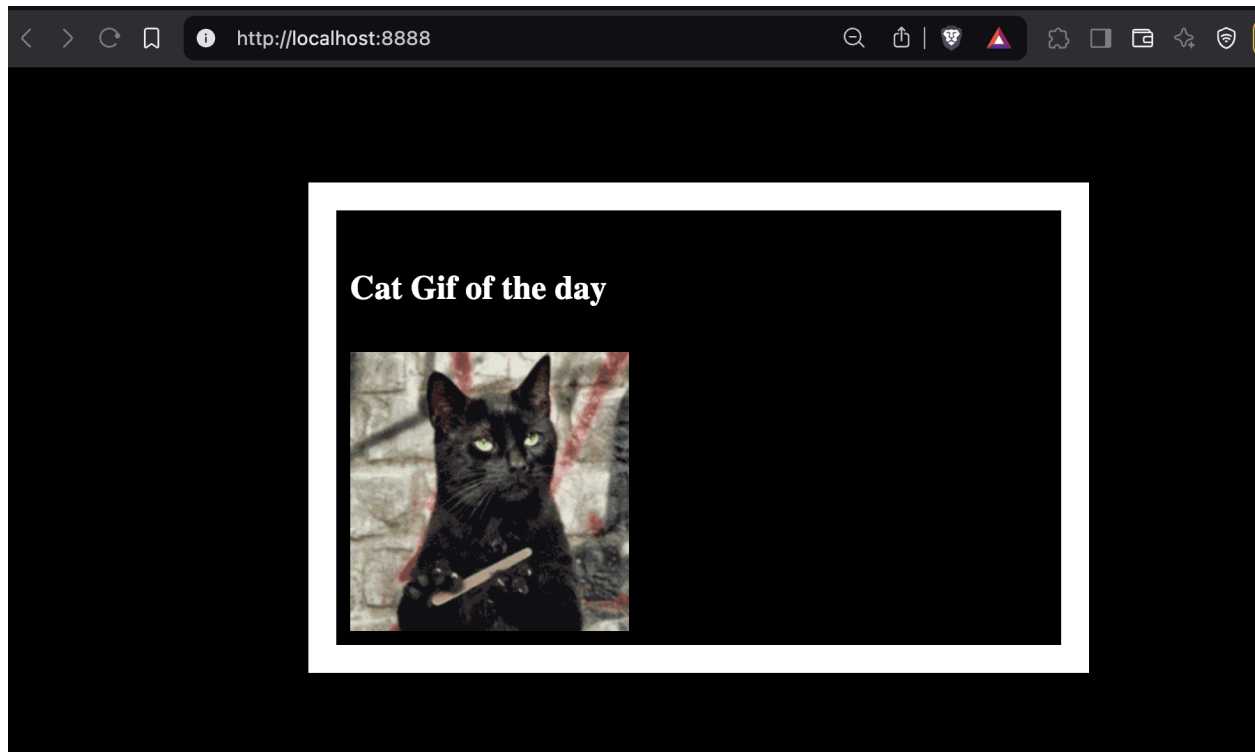
```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	a8560b36e8b8	5 weeks ago	12.8MB
hello-world	latest	7e1a4e2d11e2	8 weeks ago	17kB
projetlog2990-angular-app	latest	e907740a33b4	3 months ago	89.9MB
projetlog2990-node-server	latest	fac8fbdbbcfa	3 months ago	2.02GB
opensearchproject/opensearch-dashboards	latest	b46f3913acc5	3 months ago	2.48GB
mongo	latest	4f93a84f7d4d	3 months ago	1.12GB
dockersamples/static-site	latest	daa686c61d7d	9 years ago	293MB

```
(base) samiait-ameur@MacBook-Air-de-Sami ~ % mkdir flask-app
(base) samiait-ameur@MacBook-Air-de-Sami ~ % cd flask-app
```

```
(base) samiait-ameur@MacBook-Air-de-Sami flask-app % docker build --platform linux/amd64 -t samiaa2805/myfirstapp .
[+] Building 6.5s (12/12) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 341B                                              0.0s
=> [internal] load metadata for docker.io/library/alpine:3.5                    0.6s
=> [auth] library/alpine:pull token for registry-1.docker.io                    0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [1/6] FROM docker.io/library/alpine:3.5@sha256:66952b313e51c3bd1987d7c4ddf5dba9bc0fb6e524eed2448fa660246b3e76ec 0.0s
=> => resolve docker.io/library/alpine:3.5@sha256:66952b313e51c3bd1987d7c4ddf5dba9bc0fb6e524eed2448fa660246b3e76ec 0.0s
=> [internal] load build context                                                  0.0s
=> => transferring context: 132B                                                  0.0s
=> CACHED [2/6] RUN apk add --update py2-pip                                    0.0s
=> CACHED [3/6] COPY requirements.txt /usr/src/app/                             0.0s
=> [4/6] RUN pip install --no-cache-dir --trusted-host pypi.python.org -r /usr/src/app/requirements.txt 4.3s
=> [5/6] COPY app.py /usr/src/app/                                              0.0s
=> [6/6] COPY templates/index.html /usr/src/app/templates/                     0.0s
=> exporting to image                                                            1.4s
=> => exporting layers                                                            1.4s
=> => exporting manifest sha256:275632429f6ba351c9ad7591d8b9bb9778710a68ffa857f35f46da5ecfbd36ce 0.0s
=> => exporting config sha256:5b3b671eb9f36f3391a8b946e8529404a721a42c7f330f3629628499be552882 0.0s
=> => exporting attestation manifest sha256:f0faa6f63bad0bf0b49fecf1c6ebdfe2917da980c8461c9054b0271d71486b77 0.0s
=> => exporting manifest list sha256:b8f026f564023d2ed551a343dfb3429edd091c6d1857bcb16b38d0ac911db0ed 0.0s
=> => naming to docker.io/samiaa2805/myfirstapp:latest                          0.0s
```

```
(base) samiait-ameur@MacBook-Air-de-Sami flask-app % docker run -p 8888:5000 --name flask-app samiaa2805/myfirstapp
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.65.1 - - [21/Mar/2025 16:38:06] "GET / HTTP/1.1" 200 -
192.168.65.1 - - [21/Mar/2025 16:38:07] "GET /favicon.ico HTTP/1.1" 404 -
```



7. Expliquez la sortie de la commande `docker images`. Comment obtenir une version spécifique d'une image?

Cette commande affiche la liste des images présentes localement sur la machine hôte. Elle fournit aussi des informations importantes sur chaque image, comme sa taille ou son id. Afin d'obtenir une version spécifique d'une image, il faut ajouter le tag désiré lors du téléchargement de l'image (`docker pull`). Par exemple: `docker pull alpine:3.18` va télécharger la version 3.18 de Linux Alpine.

8. Expliquez avec vos propres mots la différence entre *base images* et *child images*.

Une *base image* est une image qui sert de fondation afin de créer des *child images*. On se sert de la *base image* (Linux Alpine par exemple), à laquelle on ajoute des applications (Python par exemple) et des configurations. On peut ainsi exécuter une application particulière sur avec différentes personnalisations.

9. Expliquez avec vos propres mots la différence entre *official images* et *user images*.

Les *official images* sont des images qui sont vérifiées et maintenues soit par l'équipe de Docker ou par les développeurs officiels des projets. Ce sont donc généralement des images fiables et sûres. Les *user images*, quant à elles, sont créées et partagées par des utilisateurs. Elles auront donc tendance à être moins fiables que les *official images*. Par contre, elles peuvent contenir des configurations personnalisées qui les rendent plus adaptées à des projets très spécifiques.

10. Expliquez avec vos propres mots ce qu'est un Dockerfile.

Un Dockerfile est un fichier dans lequel on peut inscrire une série d'instructions qui seront exécutées lors de la commande `docker build` (création de l'image Docker). Cela permet d'automatiser la création des images Docker qui pourront ensuite être utilisées pour lancer des conteneurs.

11. Expliquez chaque ligne du Dockerfile créé à l'étape 5 de la section 3.3.

- FROM alpine:3.5
Cette ligne indique que l'image de base utilisée est Alpine Linux version 3.5
- RUN apk add --update py2-pip
Cette ligne exécute une commande pendant la construction de l'image. La commande ici (*apk add*) est utilisée pour ajouter des paquets, *-update* permet de mettre à jours les paquets avant l'installation et *py2-pip* est le paquet qui est installé.
- COPY requirements.txt /usr/src/app/
Cette ligne permet de copier les fichiers de la machine hôte (ici le fichier est *requirements.txt*) vers l'image Docker (ici dans */usr/src/app/* dans le conteneur).
- RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
Cette ligne permet d'exécuter la commande *pip install --no-cache-dir -r /usr/src/app/requirements.txt* dans le conteneur. Cette commande installe toutes les librairies listées dans le fichier *requirements.txt* et *-no-cache-dir* empêche *pip* de stocker les fichiers temporaires (cela permet de réduire la taille de l'image Docker).
- COPY app.py /usr/src/app/
Cette ligne copie le fichier *app.py* depuis la machine hôte vers le répertoire */usr/src/app/* du conteneur Docker.
- COPY templates/index.html /usr/src/app/templates/
Cette ligne copie le fichier *index.html* depuis la machine hôte vers le répertoire */usr/src/app/templates/* du conteneur Docker
- EXPOSE 5000
Cette ligne indique que l'application s'exécutera sur le port 5000.
- CMD ["python", "/usr/src/app/app.py"]
Cette ligne définit la commande qui sera exécutée lorsque le conteneur démarre. La commande qui sera exécutée dans ce cas est *python /usr/src/app/app.py* ce qui lancera l'application *Flask*.


12. Expliquez ce qui se passe lorsque vous exécutez la commande `docker build <YOUR_USERNAME>/flask-app`.

Docker cherche un fichier nommé Dockerfile dans le répertoire courant afin de construire l'image. Docker lit et exécute les instructions contenues dans le Dockerfile ligne par ligne. Docker construit ainsi l'image avec un identifiant unique. Par défaut, cet identifiant est aléatoire, mais dans ce cas précis, l'identifiant devient `<YOUR_USERNAME>/flask-app` et l'image est enregistrée localement sous ce nom.

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.

Soit le Dockerfile suivant:



```
1 FROM nginx:latest
2
3 COPY . /usr/share/nginx/html
4
5 EXPOSE 80
6
7 CMD ["nginx", "-g", "daemon off;"]
```

La première ligne sert à installer la version la plus récente de NGINX, afin d'établir l'environnement de base de l'image.

La deuxième ligne sert à copier tous les fichiers du répertoire dans le dossier `/usr/share/nginx/html`, afin que nginx sache puisse être capable de servir le contenu statique de notre fichier `index.html`.

La troisième ligne permet à notre conteneur de recevoir des requêtes HTTP, en exposant le port 80 (le port pour HTTP).

La quatrième ligne est la déclaration de la commande qui est exécutée quand on démarre le conteneur. Cette ligne s'occupe de partir le serveur NGINX en avant plan (`daemon off;`), afin que Docker garde le conteneur actif.

Une fois le Dockerfile défini, il faut construire une image Docker et partir un conteneur. La première étape peut être faite par la commande `docker build -t tp4 .`, qui va créer une image à partir du Dockerfile dans le répertoire présent, en spécifiant son tag à `tp4`:


```
(pythonProject5) emirtuncbilek@Emirs-MBP 3.4 % docker build -t tp4 .
[+] Building 0.8s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 131B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 3.31kB
=> CACHED [1/2] FROM docker.io/library/nginx:latest@sha256:124b44bfc9ccdf4b592d4d1e8b5ec2ed5056c52d3692baebc19
=> [2/2] COPY . /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:c6a12e4c03acffe3b172dd2e966ff2baeef02d96d72aad8925f9b67dcd6b6acb
=> => naming to docker.io/library/tp4

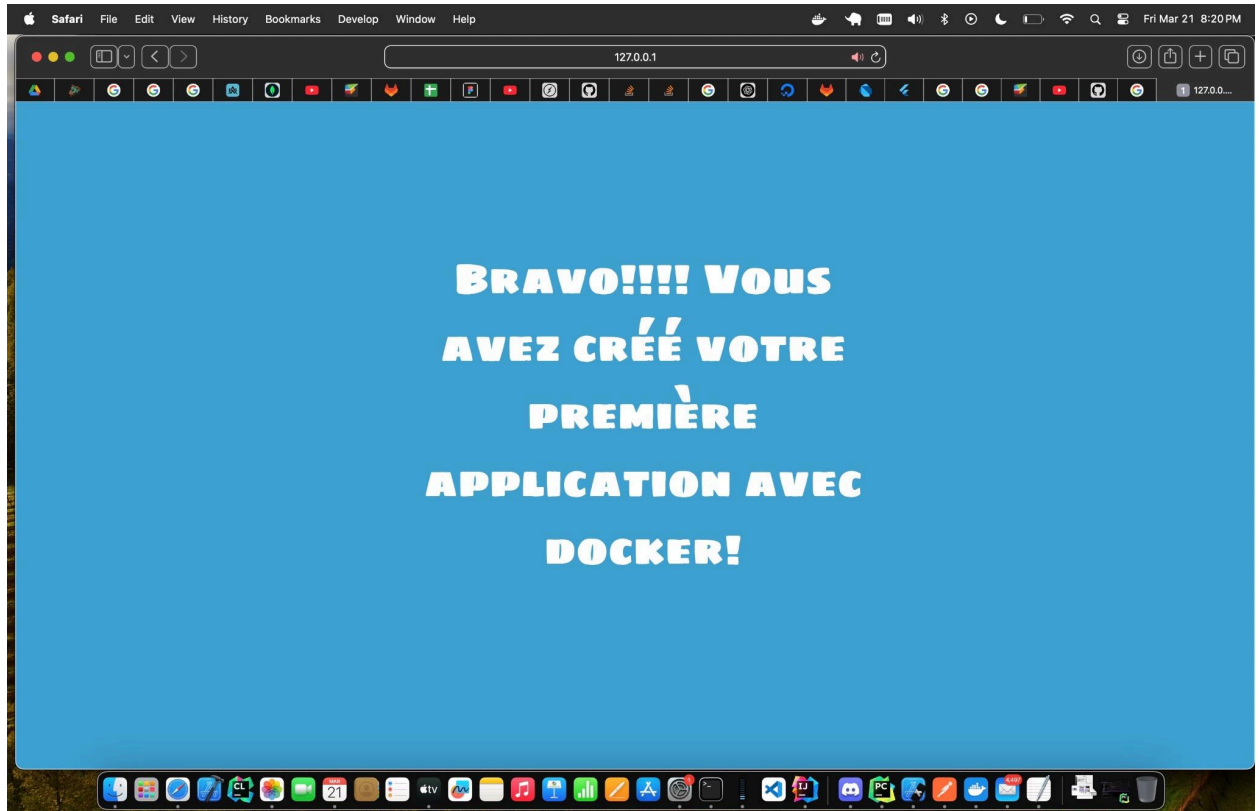
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ejdf3r3fy6mor6tzqytqclfrq

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview
(pythonProject5) emirtuncbilek@Emirs-MBP 3.4 %
```

Maintenant que notre image est construite, on peut partir un conteneur qui va pouvoir servir le contenu statique de notre application. Pour ce faire, il suffit d'exécuter la commande “`docker run -p 8080:80 tp4`”, qui va servir NGINX sur le port 8080:

```
(pythonProject5) emirtuncbilek@Emirs-MBP 3.4 % docker run -d -p 8080:80 tp4
6d8e93a4dff6af6ae316250d5807b0fea6def5dfc26bbfa64fbb480b4360cac7
(pythonProject5) emirtuncbilek@Emirs-MBP 3.4 % docker run -p 8080:80 tp4
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/03/22 00:13:50 [notice] 1#1: using the "epoll" event method
2025/03/22 00:13:50 [notice] 1#1: nginx/1.27.4
2025/03/22 00:13:50 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/03/22 00:13:50 [notice] 1#1: OS: Linux 6.10.14-linuxkit
2025/03/22 00:13:50 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/03/22 00:13:50 [notice] 1#1: start worker processes
2025/03/22 00:13:50 [notice] 1#1: start worker process 29
2025/03/22 00:13:50 [notice] 1#1: start worker process 30
2025/03/22 00:13:50 [notice] 1#1: start worker process 31
2025/03/22 00:13:50 [notice] 1#1: start worker process 32
2025/03/22 00:13:50 [notice] 1#1: start worker process 33
2025/03/22 00:13:50 [notice] 1#1: start worker process 34
2025/03/22 00:13:50 [notice] 1#1: start worker process 35
2025/03/22 00:13:50 [notice] 1#1: start worker process 36
```

Le contenu statique est maintenant disponible sur 127.0.0.1:8080 :



4.5 Question de rétroaction

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 ?***

Ce laboratoire a demandé environ 4 heures-personnes de travail. Il a été significativement plus facile que les autres laboratoires, ce qui a été très apprécié, car la charge de travail pour le cours Projet 3 (corequis) est à son pic actuellement.

2. ***Quelles difficultés avez-vous rencontré lors de ce laboratoire?***

Nous possédons tous les deux des ordinateurs macOS avec des puces Apple Silicon. Ainsi, nous avons dû adapter certaines des commandes à des fins de compatibilité. L'exercice de déterminer la commande équivalente ou d'adapter une commande à une architecture Apple Silicon a parfois été difficile et la recherche sur internet a été parfois frustrante.