Advanced Software Engineering for Digital Services (A.S.E.D.S)	TAOUFIQ Fatima INE2

Conteneurisation Des Applications

Rapport de Mini-projet

Année universitaire:

2021-2022

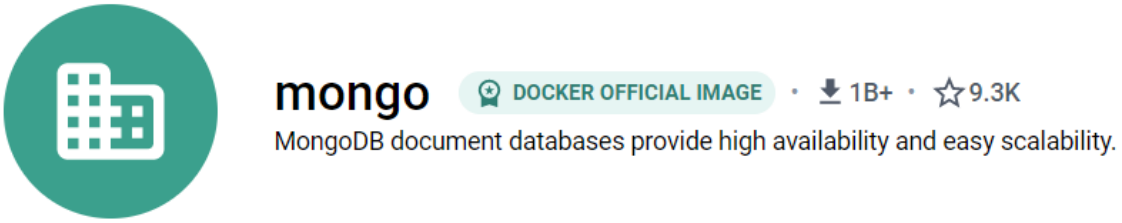
Considérons une application web 3-tiers (composée de frontend, backend et base de données) développée avec le MERN stack (MongoDB, Express, React, Node.js).

Le code source de cette application est disponible sur ces repos : frontend et backend.

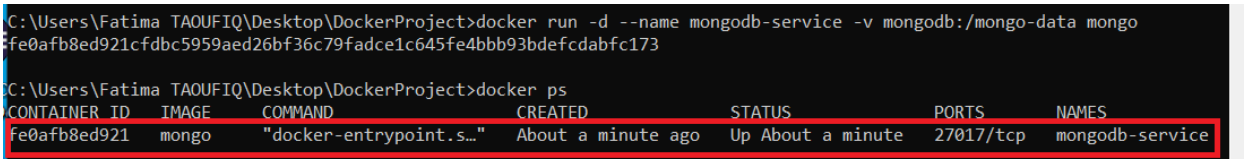
Partie 1 :

Dans cette partie, il est demandé de réaliser les manipulations suivantes en utilisant un outil de conteneurisation comme Docker.

1. La création d'un conteneur nommé **"*mongodb-service*"** pour la base de donne mongoDB en instanciant l'image officielle *mongo*:
   - Premièrement en cherche l'image office de mongo dans Docker Hub:



   - On utilise la commande **docker run** pour créer notre conteneur:



   - Concernant l'option de stockage j'ai choisi **volume** pour plusieurs raisons:
     Les volumes sont plus faciles à sauvegarder ou à migrer que Bind Mount
     Vous pouvez gérer les volumes à l'aide des commandes Docker CLI ou de l'API Docker.
     Les volumes fonctionnent sur les conteneurs Linux et Windows.
     Les volumes peuvent être partagés de manière plus sûre entre plusieurs conteneurs.
     …

2. La création d'un réseau docker:
   - On utilise la commande **{ docker network create }** avec d'autre option avec le choix de drive convenable:

```
C:\Users\Fatima TAOUFIQ\Desktop\DockerProject>docker network create --driver bridge --subnet 192.168.157.2/24 netMongo
78da4d9f09b7b8daaf11384ce222c58c11456267bdba9eb750b3f3091d748510
```

   - On vérifiait si notre réseau a bien été créé :

```
C:\Users\Fatima TAOUFIQ\Desktop\DockerProject>docker network ls
NETWORK ID     NAME       DRIVER    SCOPE
6405c0611c34   bridge     bridge    local
b588bb527dfc   host       host      local
5bad7ce59e63   minikube   bridge    local
78da4d9f09b7   netMongo   bridge    local
73443f9b7e15   none       null      local
```

   - On connecte le conteneur mongodb-service avec ce réseau avec l'utilisation de la commande **{ docker network connect nom_network nom_conteneur }** :

```
C:\Users\Fatima TAOUFIQ\Desktop\DockerProject>docker network connect netMongo mongodb-service
```

   - On suite On fait une **inspect** pour voir si le réseau a bien été ajouté au conteneur:

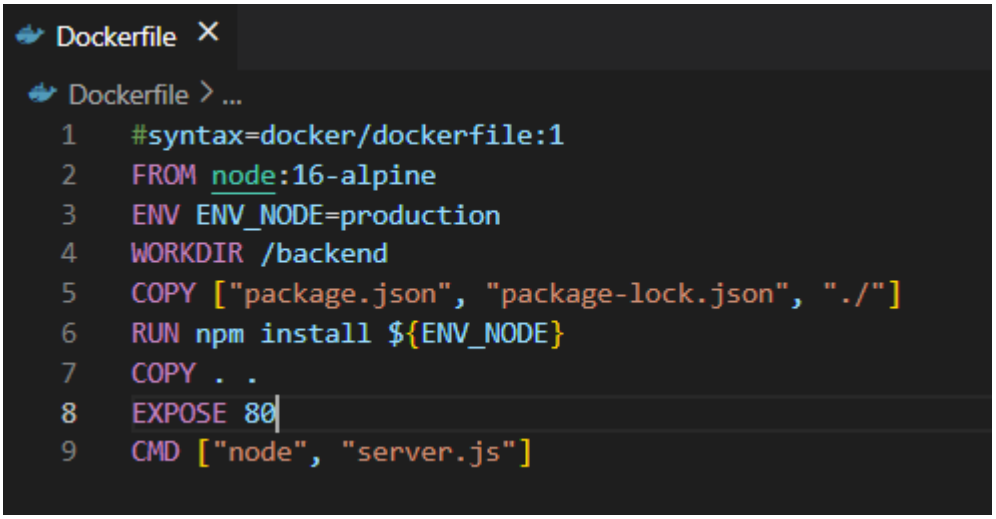

```
"netMongo": {
    "IPAMConfig": {},
    "Links": null,
    "Aliases": [
        "fe0afb8ed921"
    ],
    "NetworkID": "78da4d9f09b7b8daaf11384ce222c58c11456267bdba9eb750b3f3091d748510",
    "EndpointID": "1730429f080e2d078e1fb8fba81ef3f641b01640447df5eb1d36869904f5151e",
    "Gateway": "192.168.157.1",
    "IPAddress": "192.168.157.2",
    "IPPrefixLen": 24,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "MacAddress": "02:42:c0:a8:9d:02",
    "DriverOpts": {}
}
```


A. Backend

3. la création d'une fichier Dockerfile pour notre projet Backend:

node DOCKER OFFICIAL IMAGE · 1B+ · 10K+

Node.js is a JavaScript-based platform for server-side and networking applications.



4. L'explication en détails des bonnes pratiques utilisées pour l'écriture de ce fichier Dockerfile pour backend.
- L'instruction FROM indique que l'image de base à utiliser est **node:16-alpine**. Cette image est une image de base minimaliste basée sur Alpine Linux qui contient une version de Node.js préinstallée.

- L'instruction **ENV** définit une variable d'environnement appelée **ENV_NODE** avec la valeur production. Cette variable d'environnement peut être utilisée dans d'autres instructions du fichier Dockerfile ou dans les commandes exécutées dans le conteneur.

- L'instruction **WORKDIR** définit le répertoire de travail du conteneur comme étant /backend. Cela signifie que toutes les commandes suivantes seront exécutées à partir de ce répertoire.

- L'instruction **COPY** copie les fichiers **package.json** et **package-lock.json** du système de fichiers local dans le répertoire de travail du conteneur. Ces fichiers sont utilisés pour gérer les dépendances de l'application Node.js.

- L'instruction **RUN** exécute la commande **npm install ${ENV_NODE}**, qui installe les dépendances de l'application en utilisant la variable d'environnement **ENV_NODE** comme argument.

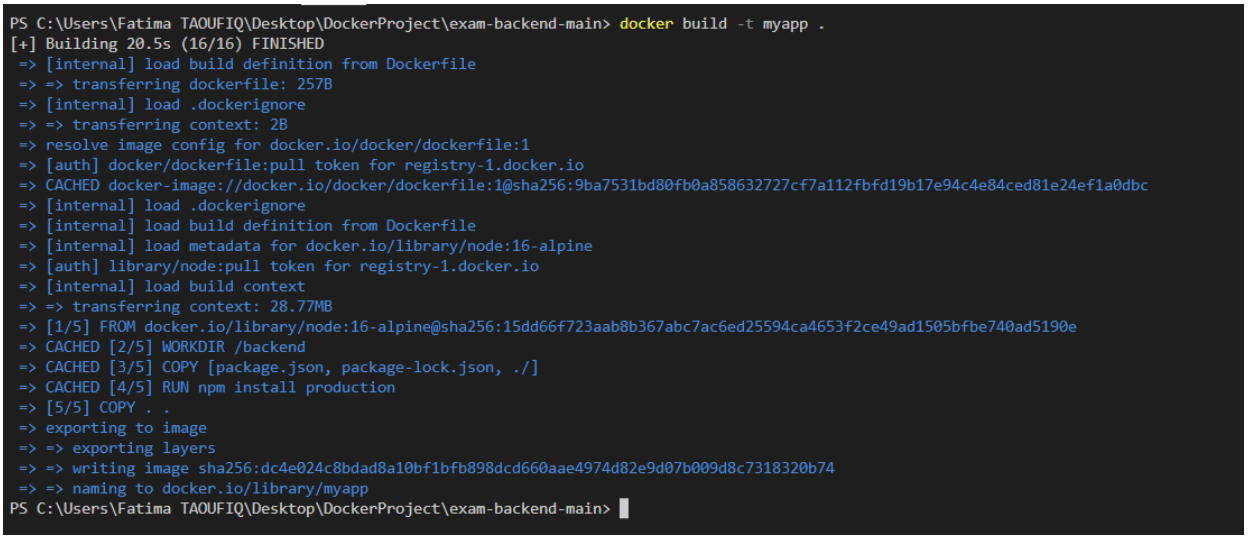
- L'instruction **COPY** copie tous les fichiers du répertoire courant (indiqué par le point final) dans le répertoire de travail du conteneur.

- L'instruction **EXPOSE** indique que le conteneur est écouté sur le port **80**. Cela signifie que les applications exécutées dans le conteneur pourront être accessibles via ce port depuis l'extérieur du conteneur.

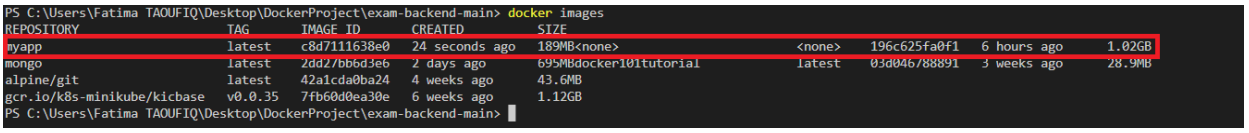
- L'instruction **CMD** définit la commande à exécuter lorsque le conteneur est lancé. Dans ce cas, la commande node server.js lance l'application Node.js contenue dans le fichier server.js.

➔ En général, il est recommandé de maintenir l'ordre des instructions de manière à minimiser le nombre de couches de l'image finale et à rendre le fichier Dockerfile plus lisible et facile à comprendre. Il est également important de choisir une image de base appropriée et de ne pas inclure des fichiers inutiles dans l'image. En utilisant les instructions ARG et ENV, vous pouvez rendre votre image plus flexible et réutilisable en permettant de passer des arguments ou de définir des variables d'environnement lors de la construction de l'image.

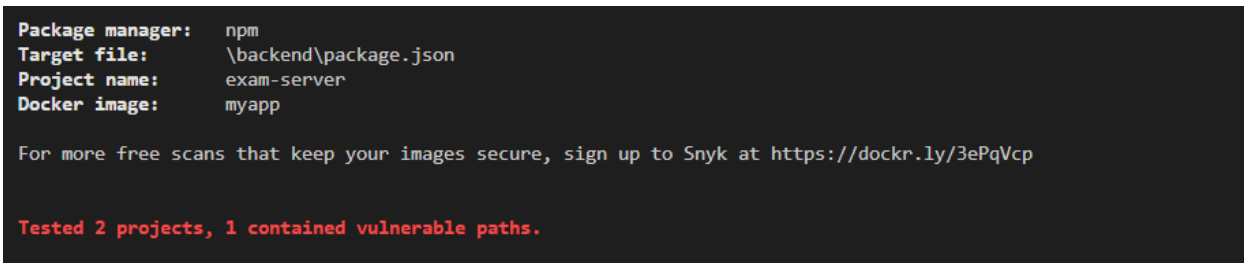
5. Les commandes docker nécessaires pour :
   a. La création d'une image docker a partir de ce Dockerfile :
      - On utilise la commande **{ docker build -t image_Name }**
      - **-t** : spécifie le nom de l'image à crée
      - **.** : indique au build de se baser sur les fichiers et répertoires présents dans le répertoire courant.



- On utilise la commande On vérifie que l'image a été créée avec succès en utilisant la commande **{ docker images}** , Elle va lister toutes les images créées.



   b. On scanne l'image des vulnérabilités qu'elle peut contenir.
      - On utilise la commande **{ docker scan backend }** afin de vérifier qu'elles ne contiennent pas de vulnérabilités connues ou de logiciels malveillants.



c. On Publie cette image dans le Docker Hub.
   - On ajoute un tag à notre image myapp on utilisant la commande **{ docker tag }**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-backend-main> docker tag myapp fatimataaoufiq/dockerproject:myapp-v1
```

   - On publie l'image dans Docker Hub en utilisant la commande **{ docker push }**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-backend-main> docker push fatimataaoufiq/dockerproject:myapp-v1
The push refers to repository [docker.io/fatimataaoufiq/dockerproject]
f88ff6d12378: Pushed
3604f169f95a: Layer already exists
e40aced459f7: Layer already exists
d87c4627757f: Layer already exists
48bc9cd339ab: Layer already exists
170bc007c2d3: Layer already exists
a30c73fdccc6: Layer already exists
e5e13b0c77cb: Layer already exists
myapp-v1: digest: sha256:06d6371f80c63488940aa6acf73eddf67001890d7683760aad9825bf3523d65e size: 1997
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-backend-main>
```

d. On instancie cette image en créant un conteneur (nommé backend) qui s'exécute en local et qui partage le même réseau avec le conteneur de la base de données MongoDB

{ docker build }

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-backend-main> docker run --name backend -itd --network=netMongo -p 5000:5000 fatimataaoufiq/dockerproject:myapp-v1
74cf24fc82a28f910e889486ca31305a44e29b71969a072654225030400a56b5
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-backend-main>
```

- **run**: indique à Docker de créer et exécuter un conteneur à partir d'une image.

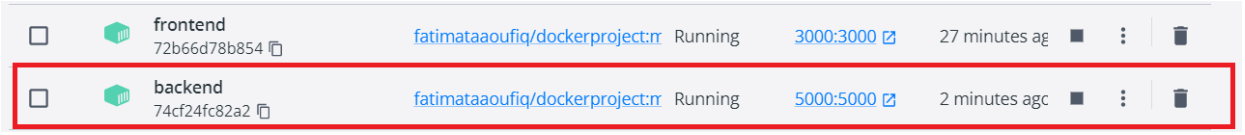
- **--name backend**: donne un nom au conteneur qui sera créé. Le nom de ce conteneur sera "backend".

- **itd**: ces options sont utilisées pour configurer le mode d'exécution du conteneur. **-i** indique que le conteneur doit être exécuté en mode interactif, c'est-à-dire que l'utilisateur peut entrer des commandes dans le conteneur. **-t** indique que le conteneur doit avoir un terminal associé, ce qui permet à l'utilisateur de saisir des commandes dans le conteneur. **-d** indique que le conteneur doit être exécuté en arrière-plan, c'est-à-dire qu'il continuera à s'exécuter même si l'utilisateur quitte le terminal ou la console.

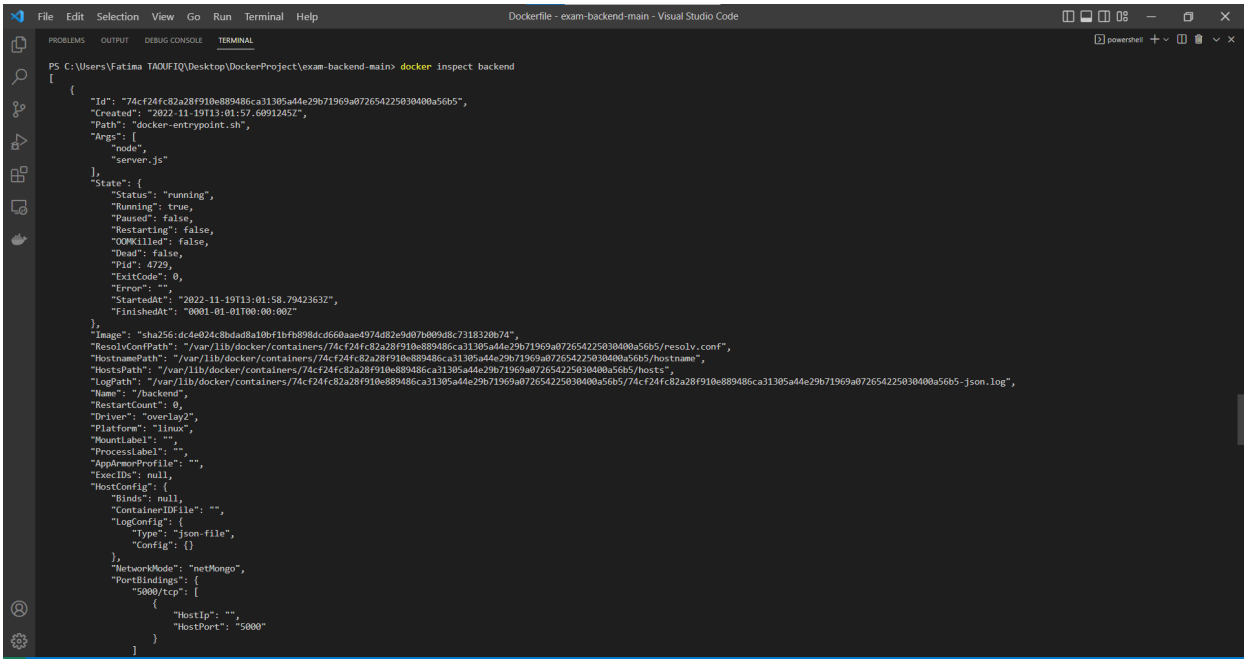
- **--network=netMongo**: cette option indique que le conteneur doit être connecté à un réseau Docker nommé "netMongo". Cela permet au conteneur de communiquer avec d'autres conteneurs sur ce réseau.

- **-p 5000:5000**: cette option indique que le port 5000 du conteneur doit être mappé sur le port 5000 de l'hôte. Cela signifie que les données envoyées au port 5000 de l'hôte seront acheminées vers le port 5000 du conteneur, et vice versa.

En résumé, cette commande exécute un conteneur Docker nommé "**backend**" à partir de l'image " **fatimataaoufiq/dockerproject:myapp-v1** ", en mode interactif avec un terminal associé et en arrière-plan. Le conteneur est connecté au réseau Docker "**netMongo**" et le port 5000 du conteneur est mappé sur le port 5000 de l'hôte.



e. On inspecte le conteneur backend.



f. Pour afficher les logs liés à ce conteneur backend on utilisant la commande

{ docker logs }

➔ **docker logs** est une commande Docker qui permet d' afficher les journaux d'un conteneur. Les journaux incluent les sorties standard (stdout) et d'erreur (stderr) de toutes les commandes exécutées dans le conteneur, ainsi que tout autre message envoyé par le système au niveau du noyau.

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-backend-main> docker logs backend
Server v8 up and running on port 5000 !
MongoDB successfully connected
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-backend-main>
```

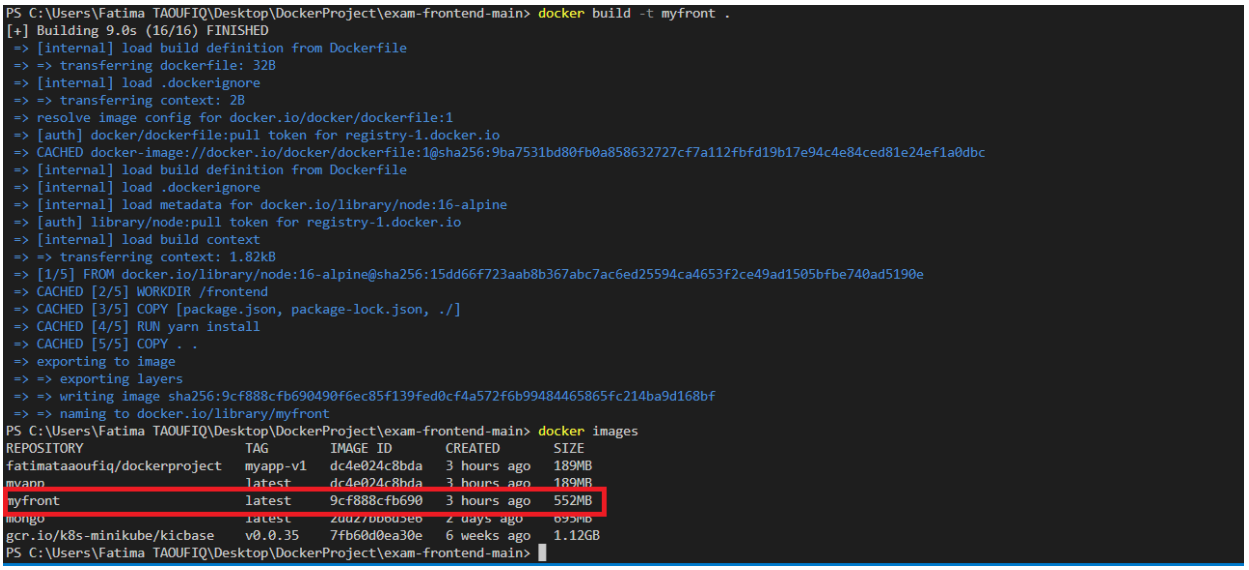
B. Frontend

6. On refaire le même travail pour la partie Frontend:
   - La création du fichier dockerfile:

```
exam-frontend > Dockerfile > ...
1   FROM node:16.18.1-alpine AS builder
2   ENV CLOUDL_SERVER=localhost
3   WORKDIR /usr/src/app
4   COPY package.json .
5   RUN npm install
6   COPY . .
7   RUN npm run build
8   FROM nginx:1.23.2-alpine
9   WORKDIR /usr/share/nginx/html
10  RUN rm -rf ./*
11  COPY --from=builder /usr/src/app/build .
12  COPY --from=builder /usr/src/app/nginx/nginx.conf /etc/nginx/conf.d/default.conf
13  EXPOSE 80
14  CMD ["nginx", "-g", "daemon off;"]
```

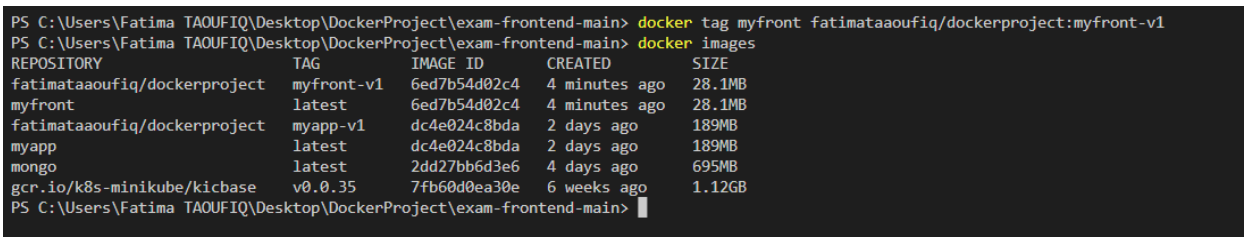
- La création du l'image a partir de ce dockerfile on utilisant la commande

  { docker build }



- On ajoutant un tag à notre image:

  { docker tag}



- On scanne l'image des vulnérabilités qu'elle peut contenir la commande

  { docker scan container_Name }

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker scan myfront
WARNING!  Partially defined environment, please ensure to provide both SNYK_INTEGRATION_NAME and SNYK_INTEGRATION_VERSION together
!

Testing myfront...

Package manager:   apk
Project name:      docker-image|myfront
Docker image:      myfront
Platform:          linux/amd64
Base image:        node:16.18.1-alpine3.16

✓ Tested 16 dependencies for known vulnerabilities, no vulnerable paths found.

According to our scan, you are currently using the most secure version of the selected base image

For more free scans that keep your images secure, sign up to Snyk at https://dockr.ly/3ePqVcp


PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main>
```

- On publie cette image dans Docker Hub on utilisant la commande

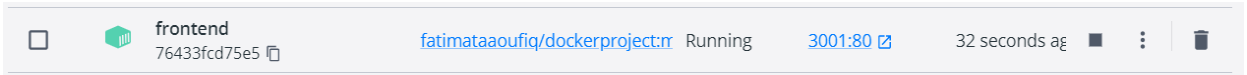
  { docker push}

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker push fatimataaoufiq/dockerproject:myfront-v1
The push refers to repository [docker.io/fatimataaoufiq/dockerproject]
490a32f3e1e7: Pushed
a42ecf4dd6af: Pushed
1024efd2d6bd: Pushed
5f70bf18a086: Pushed
bd502c2dee4c: Mounted from library/nginx
9365b1fffb04: Mounted from library/nginx
6636f46e559d: Mounted from library/nginx
fcf860bf48b4: Mounted from library/nginx
07099189e7ec: Mounted from library/nginx
e5e13b0c77cb: Layer already exists
myfront-v1: digest: sha256:8f3e6d26d9bd9aa137061ade1fb67a7a5f5c17eab2889590d5920f04078961a1 size: 2399
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main>
```

- On instancie cette image en créant un conteneur (nommé frontend) qui s'exécute en local et qui partage le même réseau avec le conteneur de la base de données MongoDB

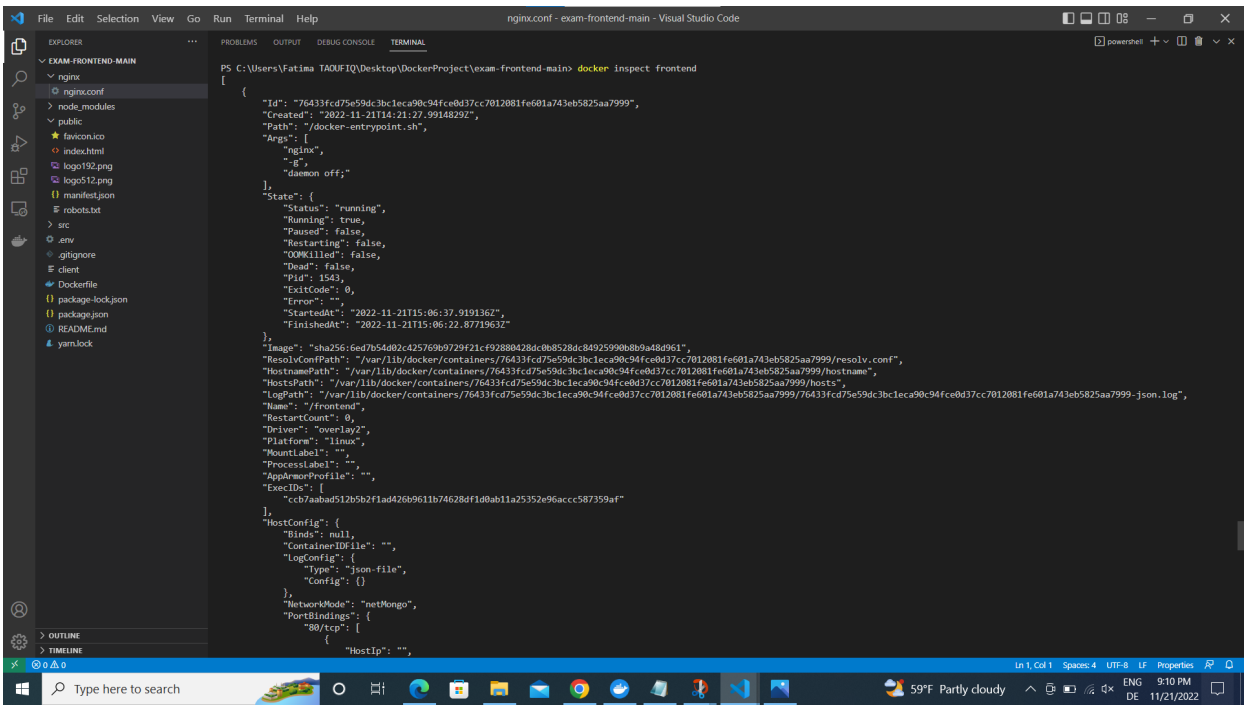
  On utilise la commande **{ docker run}** avec les options nécessaires.

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker run --name frontend -itd --network=netMongo -p 3001:80 fatimataaoufiq/dockerproject:myfront-v1
76433fcd75e59dc3bc1eca90c94fce0d37cc7012081fe601a743eb5825aa7999
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main>
```



- On inspecte ce conteneur frontend

  { docker inspect }

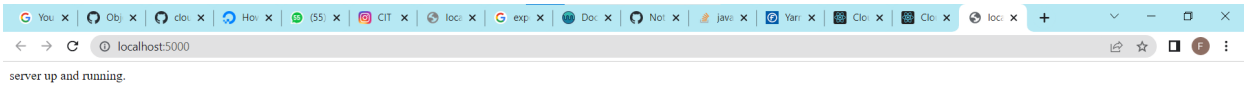


- On affiche les logs liés à ce conteneur frontend

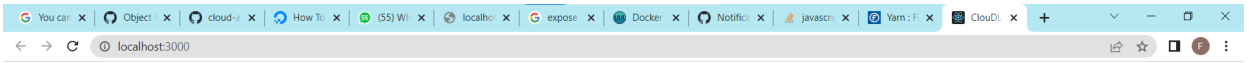
{ docker logs}

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker logs frontend
/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: /etc/nginx/conf.d/default.conf differs from the packaged version
/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
2022/11/21 14:21:29 [notice] 1#1: using the "epoll" event method
2022/11/21 14:21:29 [notice] 1#1: nginx/1.23.2
2022/11/21 14:21:29 [notice] 1#1: built by gcc 11.2.1 20220219 (Alpine 11.2.1_git20220219)
2022/11/21 14:21:29 [notice] 1#1: OS: Linux 5.10.16.3-microsoft-standard-WSL2
2022/11/21 14:21:29 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/11/21 14:21:29 [notice] 1#1: start worker processes
2022/11/21 14:21:29 [notice] 1#1: start worker process 29
2022/11/21 14:21:29 [notice] 1#1: start worker process 30
2022/11/21 14:21:29 [notice] 1#1: start worker process 31
2022/11/21 14:21:29 [notice] 1#1: start worker process 32
2022/11/21 14:21:29 [notice] 1#1: start worker process 33
2022/11/21 14:21:29 [notice] 1#1: start worker process 34
2022/11/21 14:21:29 [notice] 1#1: start worker process 35
2022/11/21 14:21:29 [notice] 1#1: start worker process 36
```

7. Assurent que l'application a été bien conteneurisée et qu'elle fonctionne correctement:
   - **Backend**



   - **Frontend**

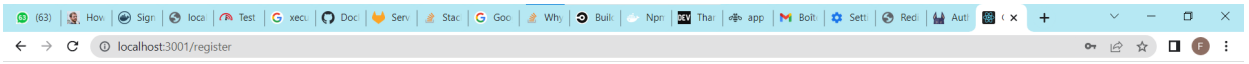


ASEDS-Exam

The **online** file downloader 3

Let me handle your slow downloads and torrents!

REGISTER LOG IN



ASEDS-Exam

BACK TO HOME

Register below

Already have an account? Log in

Name

Fatima Taoufiq

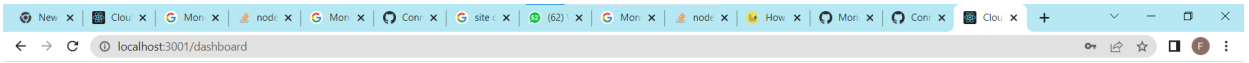
Email

fatimatq352@gmail.com

Password

Confirm Password

SIGN UP



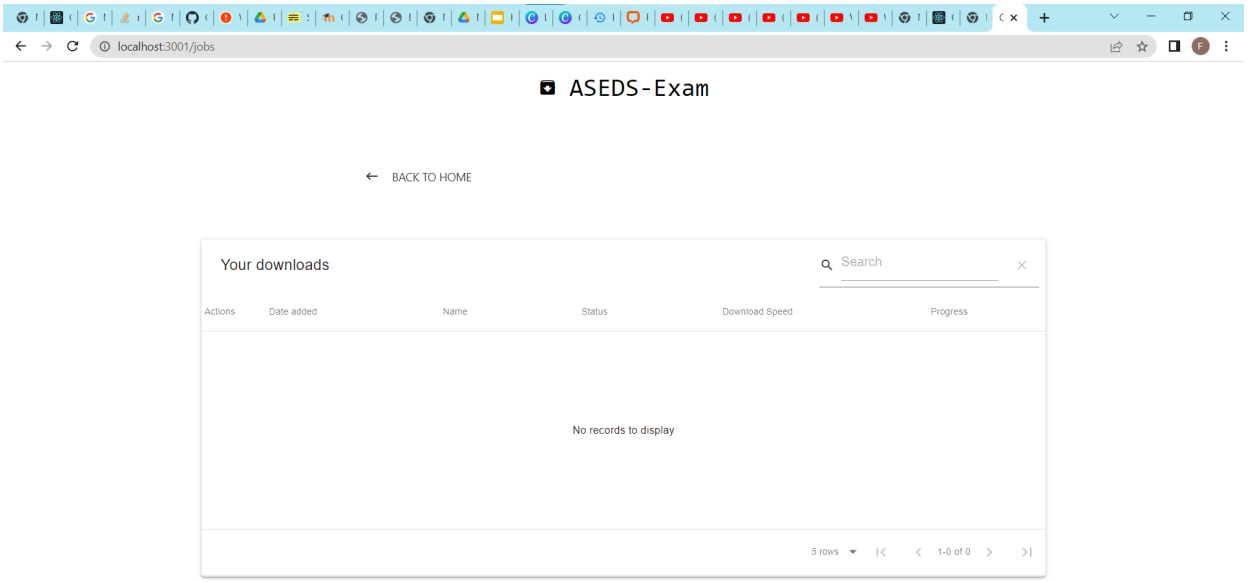
ASEDS-Exam

Hey there, Fatima!

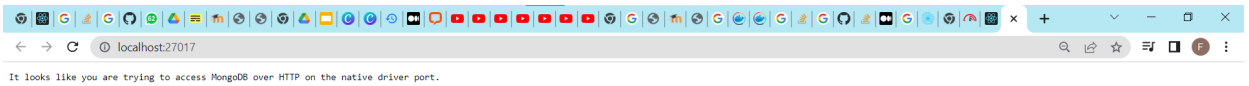
What do you need me to download today?

paste your link here

DOWNLOAD



- **Mongodb**



8. On supprime les 3 conteneurs en exécution:
   - On utilise la commande **{ docker rm -f Name_Container }**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker container ls
CONTAINER ID   IMAGE                                  COMMAND                  CREATED       STATUS              PORTS                                       NAMES
76433fcd75e5   fatimataaoufiq/dockerproject:myfront-v1   "/docker-entrypoint.…"   6 hours ago   Up About a minute   0.0.0.0:3001->80/tcp                        frontend
74cf24fc82a2   fatimataaoufiq/dockerproject:myapp-v1     "docker-entrypoint.s…"   2 days ago    Up About a minute   80/tcp, 0.0.0.0:5000->5000/tcp              backend
fe0afb8ed921   mongo                                  "docker-entrypoint.s…"   3 days ago    Up 5 hours          27017/tcp                                   mongodb-service
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker rm -f frontend
frontend
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker rm -f backend
backend
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main> docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS       PORTS       NAMES
fe0afb8ed921   mongo     "docker-entrypoint.s…"   3 days ago    Up 5 hours   27017/tcp   mongodb-service
PS C:\Users\Fatima TAOUFIQ\Desktop\DockerProject\exam-frontend-main>
```

```
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker rm -f mongodb-service
mongodb-service
PS C:\Users\Fatima TAOUFIQ\Desktop\docker>
```

   - On remarque que toutes les conteneurs sont supprimés

```
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\Fatima TAOUFIQ\Desktop\docker>
```

9. Redéploiement de l'application (frontend, backend et base de données) en utilisant docker-compose.
   a. On commence tout d'abord par la création du fichier docker-compose:

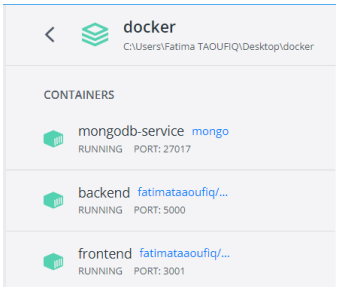
docker-compose.yaml

```yaml
version: '3'
services:
  mongodb-service:
    image: mongo
    volumes:
    - data:/data/mongodb
    container_name: mongodb-service
    ports:
    - 27017:27017
    networks:
    - netMongo

  backend:
    image: fatimataaoufiq/dockerproject:myapp-v1
    container_name: backend
    ports:
    - 5000:5000
    depends_on:
    - mongodb-service
    networks:
    - netMongo

  frontend:
    image: fatimataaoufiq/dockerproject:myfront-v1
    container_name: frontend
    ports:
    - 3001:80
    depends_on:
    - backend
    networks:
    - netMongo

volumes:
  data:
networks:
  netMongo:
    driver: bridge
```

b. On exécute notre fichier:

- On utilise la commande **{ docker compose up}**

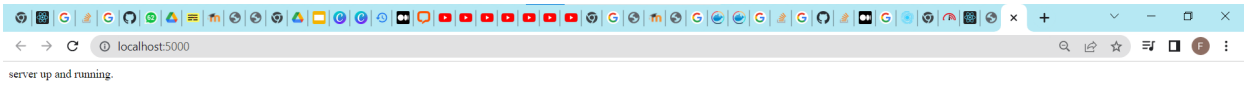
```
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker-compose up
[+] Running 3/3
 - Container mongodb-service  Created
 - Container backend          Created
 - Container frontend         Created
```



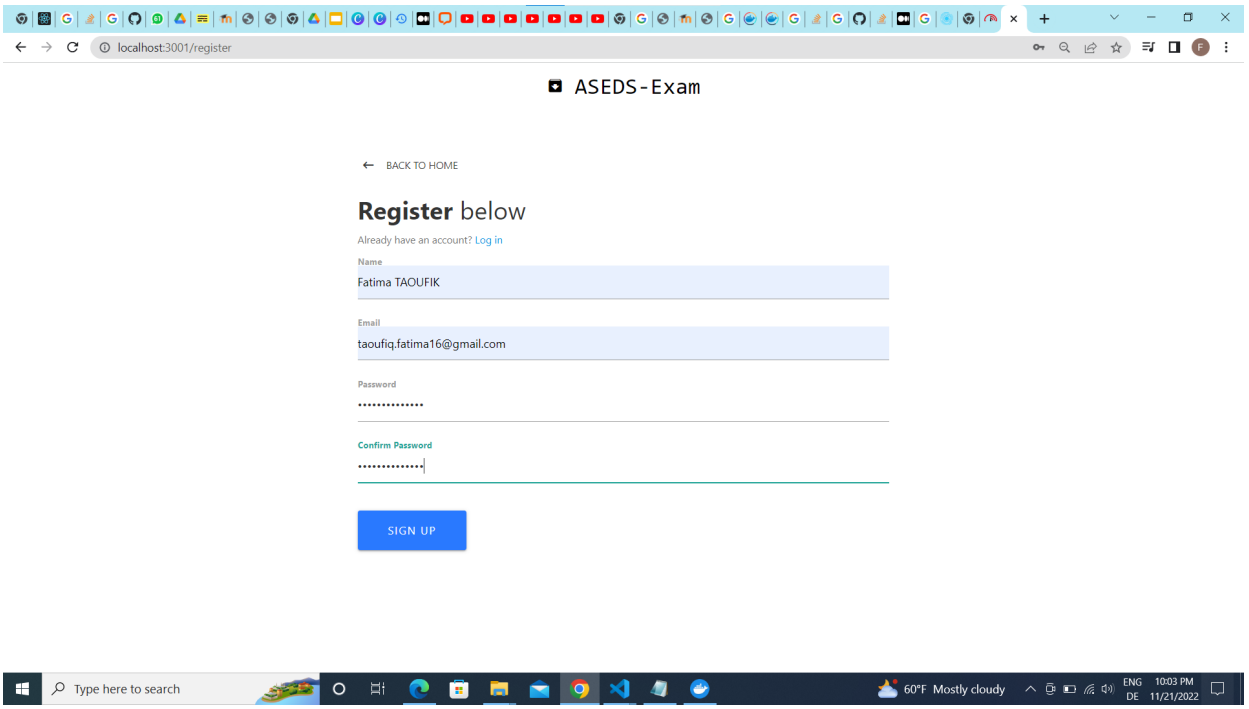

```
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker container ls
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORTS                              NAMES
942d6d343381   fatimataaoufiq/dockerproject:myfront-v1   "/docker-entrypoint.…"   21 minutes ago   Up About a minute   0.0.0.0:3001->80/tcp               frontend
0c00e907e99b   fatimataaoufiq/dockerproject:myapp-v1     "docker-entrypoint.s…"   21 minutes ago   Up About a minute   80/tcp, 0.0.0.0:5000->5000/tcp     backend
ac6e50b23ffa   mongo                                  "docker-entrypoint.s…"   21 minutes ago   Up About a minute   0.0.0.0:27017->27017/tcp           mongodb-service
PS C:\Users\Fatima TAOUFIQ\Desktop\docker>
```

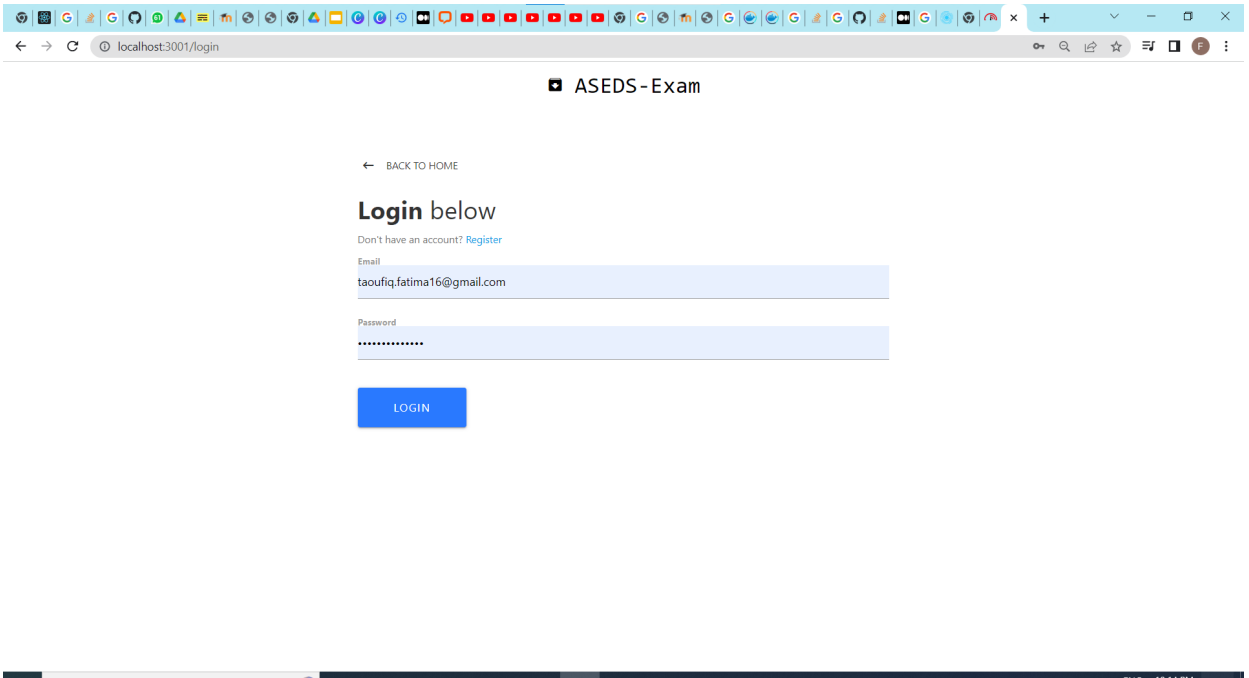
c. Assurent que l'application fonctionne correctement:

- **Backend**

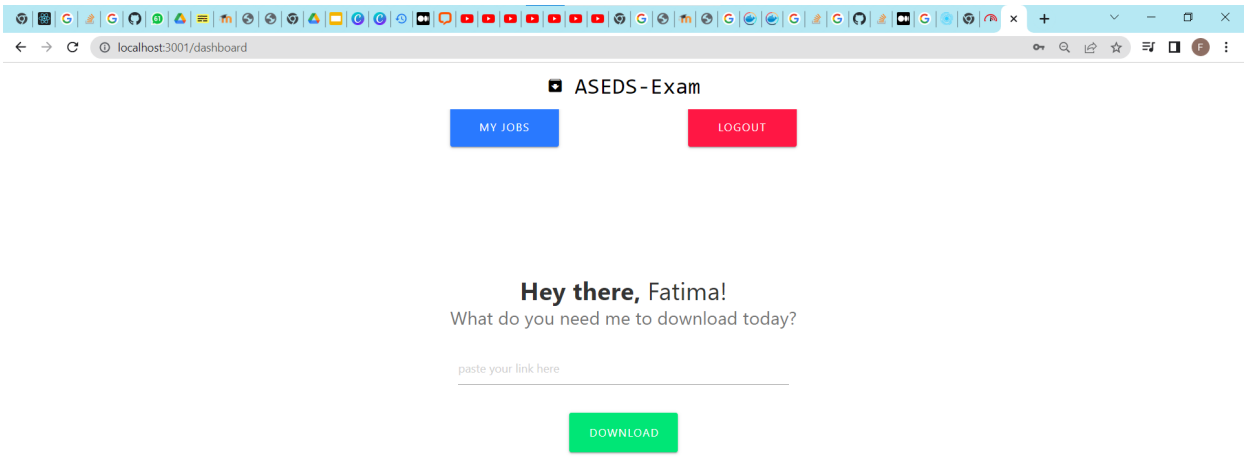


- **Frontend**

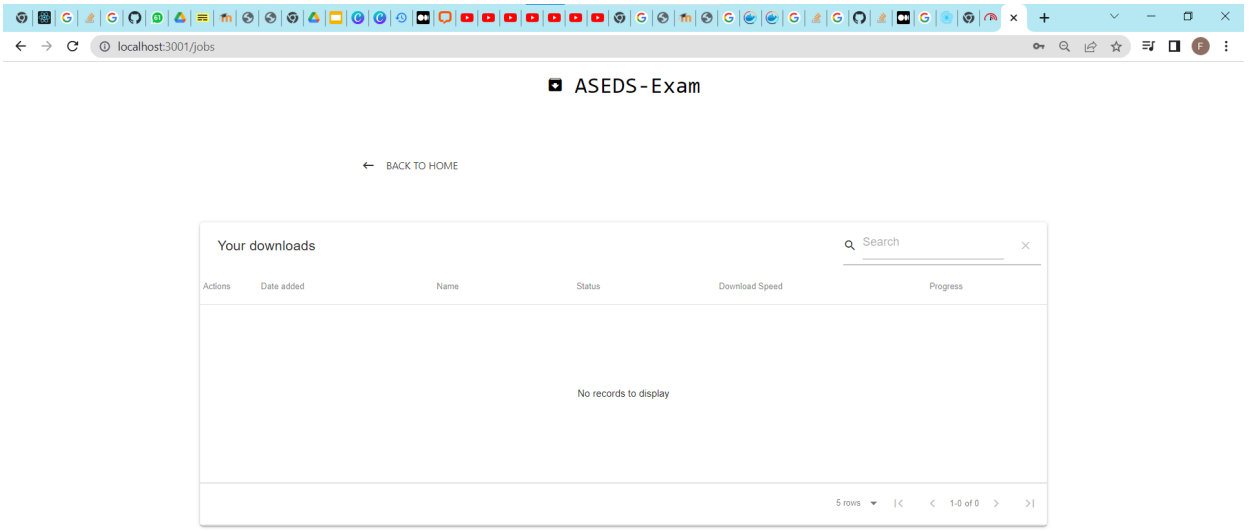



localhost:3001/login
ASEDS-Exam
BACK TO HOME
Login below
Don't have an account? Register
Email
taoufiq.fatima16@gmail.com
Password
LOGIN


Type here to search
60°F Mostly cloudy
ENG
DE
10:14 PM
11/21/2022

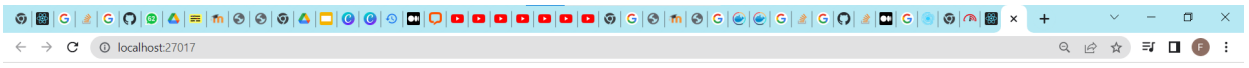

localhost:3001/dashboard
ASEDS-Exam
MY JOBS
LOGOUT
Hey there, Fatima!
What do you need me to download today?
paste your link here
DOWNLOAD


Type here to search
60°F Mostly cloudy
ENG
DE
10:14 PM
11/21/2022


localhost:3001/jobs
ASEDS-Exam
BACK TO HOME
Your downloads
Search
Actions
Date added
Name
Status
Download Speed
Progress
No records to display
5 rows
1-0 of 0


Type here to search
60°F Mostly cloudy
ENG
DE
10:15 PM
11/21/2022

- **Mongodb**


localhost:27017
It looks like you are trying to access MongoDB over HTTP on the native driver port.


Type here to search
60°F Mostly cloudy
ENG
DE
10:20 PM
11/21/2022

10. On supprime les conteneurs qui s'exécutent ainsi que les images qui se trouvent dans le docker host.
    - **suppression des conteneurs :**
        - Avec l'utilisation de la commande **docker rm -f** :

```
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker container ls
CONTAINER ID   IMAGE                                  COMMAND                  CREATED         STATUS         PORTS                              NAMES
942d6d343381   fatimataaoufiq/dockerproject:myfront-v1   "/docker-entrypoint.…"   23 minutes ago   Up 3 minutes   0.0.0.0:3001->80/tcp               frontend
0c00e907e99b   fatimataaoufiq/dockerproject:myapp-v1     "docker-entrypoint.s…"   23 minutes ago   Up 3 minutes   80/tcp, 0.0.0.0:5000->5000/tcp     backend
ac6e50b23ffa   mongo                                  "docker-entrypoint.s…"   23 minutes ago   Up 3 minutes   0.0.0.0:27017->27017/tcp           mongodb-service
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker rm -f frontend, backend, mongodb-service
frontend
backend
mongodb-service
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\Fatima TAOUFIQ\Desktop\docker>
```

        - On constate que, après l'exécution de la commande **docker container ls**, rien n'est affiché.

    - **suppression des images :**
        - Avec l'utilisation de la commande **docker rmi -f** :

```
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker images
REPOSITORY                      TAG          IMAGE ID       CREATED       SIZE
myfront                         latest       6ed7b54d02c4   7 hours ago   28.1MB
fatimataaoufiq/dockerproject    myfront-v1   6ed7b54d02c4   7 hours ago   28.1MB
fatimataaoufiq/dockerproject    myapp-v1     dc4e024c8bda   2 days ago    189MB
myapp                           latest       dc4e024c8bda   2 days ago    189MB
mongo                           latest       2dd27bb6d3e6   5 days ago    695MB
gcr.io/k8s-minikube/kicbase     v0.0.35      7fb60d0ea30e   7 weeks ago   1.12GB
PS C:\Users\Fatima TAOUFIQ\Desktop\docker> docker rmi myfront, mapp, mongo, fatimataaoufiq/dockerproject
Untagged: myfront:latest
Untagged: mongo:latest
Untagged: mongo@sha256:8bed0be3e86595283d67836e8d4f3f08916184ea6f2aac7440bda496083ab0c8
Deleted: sha256:2dd27bb6d3e6ef0dc97e0ea1e29ec2213e6c384ff69340f553d08a4f76cc0623
Deleted: sha256:ae74493b71541b59ef312d8c187fdb662663ce1209013ebf31d0d990a5d083a3
Deleted: sha256:3f38c035db1bf16806f2201b7095b83698f47f2efd19045f352b0b2192cd7795
Deleted: sha256:da8473fc7076989e9775fa6820966fecf5d771154d5bfb652b31bd7d5dfbcba4
Deleted: sha256:8a173111c463364755ff78f0bd0df77ddff0bd8995432061ad56980ccf141b5d
Deleted: sha256:e23880b5b37f569f84e026ca4d0e3c437d527891c3dc9b444b1a7f0af7f8ae14
Deleted: sha256:4bda15efcd182d03bbfc0114ce25201895288f3e5824f3b4a6333db2bb8111d8
Deleted: sha256:4528d86ac792ddc8e58b057a0ab5c27cfab00906cd0b04382f8d6bb5dee4e1eb
Deleted: sha256:ac7b321428fef5521b304f44cde7a6889a9b0220a30ea3691a089eafc18d4aff
Deleted: sha256:f4462d5b2da2985f37409c9b257afd2b9fb82356ce4e43e804ee34214242e34a
Error: No such image: mapp
Error: No such image: fatimataaoufiq/dockerproject
```

11. On crée un registre privé d'images docker en local :
    a. **Création d'un registre local**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project> docker run -d -p 5000:5000 --restart=always --name registry registry:2
f3a240eeb559bfba50bde3c4a0d9926a886e24dbe0e7cf87277f5b60007daf64
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project>
```

➔ cette commande exécute un conteneur Docker en arrière-plan à partir de l'image "**registry:2**", en mappant le port **5000** du conteneur sur le port 5000 de l'hôte et en configurant le conteneur pour qu'il soit toujours redémarré en cas d'arrêt ou de crash. Le conteneur est nommé "**registry**".

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project> docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS          PORTS                    NAMES
f3a240eeb559   registry:2   "/entrypoint.sh /etc…"   59 seconds ago   Up 58 seconds   0.0.0.0:5000->5000/tcp   registry
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project>
```

➔ Pour stocker les images Docker, il est nécessaire de changer leur tag de manière à ce qu'elle ressemble à **localhost:<port>/<repository>** en utilisant la commande **docker tag**.

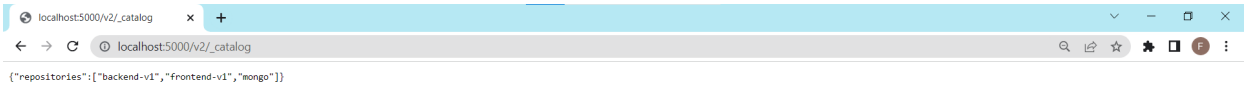
```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project\exam-backend> docker tag back localhost:5000/backend-v1
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project\exam-backend> docker push localhost:5000/backend-v1
Using default tag: latest
The push refers to repository [localhost:5000/backend-v1]
0b2b5b8d872b: Pushed
98b1a2b9cc0c: Pushed
bd8cb60f907b: Pushed
ef376ec63e0f: Pushed
65fd22078896: Pushed
069592e4e25c: Pushed
73f654397d17: Pushed
ded7a220bb05: Pushed
latest: digest: sha256:04daf947dabf3f14aa2b25b0018fe2847e3f3dfa8e5bfb09cd4655b9a28e14d3 size: 1995
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project\exam-backend>
```

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project\exam-frontend> docker tag myfront localhost:5000/frontend-v1
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project\exam-frontend> docker push localhost:5000/frontend-v1
Using default tag: latest
The push refers to repository [localhost:5000/frontend-v1]
cab6eb2f98b8: Pushed
09ae6814f52f: Pushed
aba42e6183de: Pushed
5f70bf18a086: Pushed
bd502c2dee4c: Pushed
9365b1fffb04: Pushed
6636f46e559d: Pushed
fcf860bf48b4: Pushed
07099189e7ec: Pushed
e5e13b0c77cb: Pushed
latest: digest: sha256:16f2fd9ebf1c25837800d768518b464b9ddf5f309a381ec8c44878d4d5460517 size: 2399
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project\exam-frontend>
```

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project> docker tag mongo localhost:5000/mongo
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project> docker push localhost:5000/mongo
Using default tag: latest
The push refers to repository [localhost:5000/mongo]
36cf62b9c104: Pushed
8cb70249ec8b: Pushed
6eb53606d2f0: Pushed
790676c6bc4c: Pushed
9e00d642bd32: Pushed
bd57488bf0dd: Pushed
7cc0e6d35a6f: Pushed
0dba6a2e4907: Pushed
0002c93bdb37: Pushed
latest: digest: sha256:695e5eb141aa8516bd4857ee987d65401a424d0e5f8e0244ab89f6da981ddf65 size: 2202
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\Docker-project>
```

b. Visualiser les repositories et images créés via une UI :

- On accede a URL **localhost:8080/v2/_catalog**



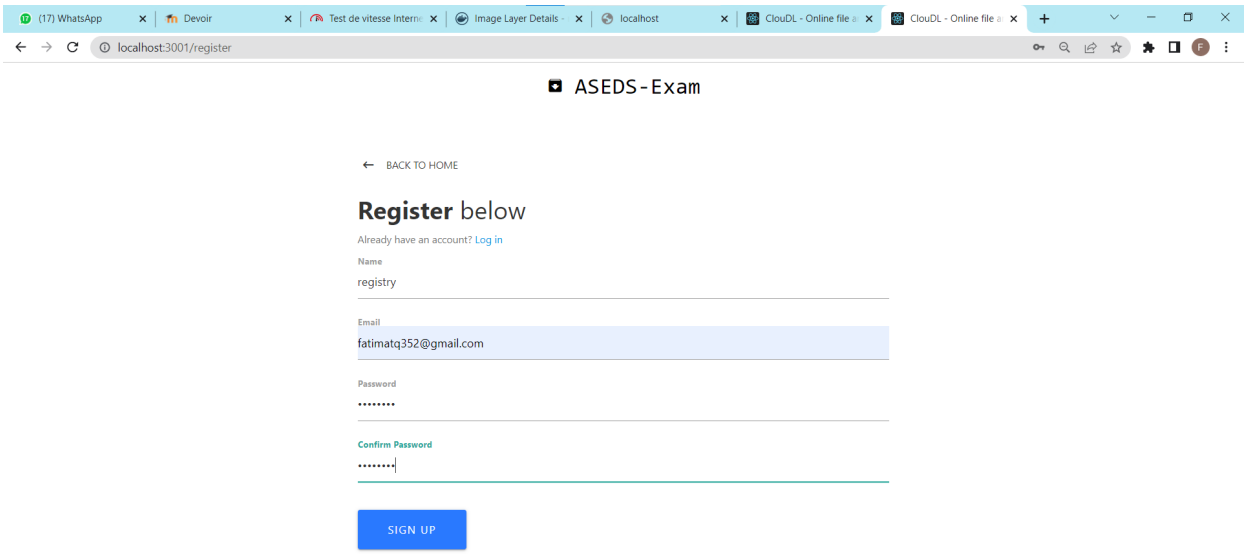
12. Redéployer l'application (frontend, backend et base de données) en utilisant docker-compose tout en récupérant l'image des conteneurs depuis ce nouveau registre privé.
    NB : S'assurer que l'application fonctionne correctement (prises d'écran du navigateur)

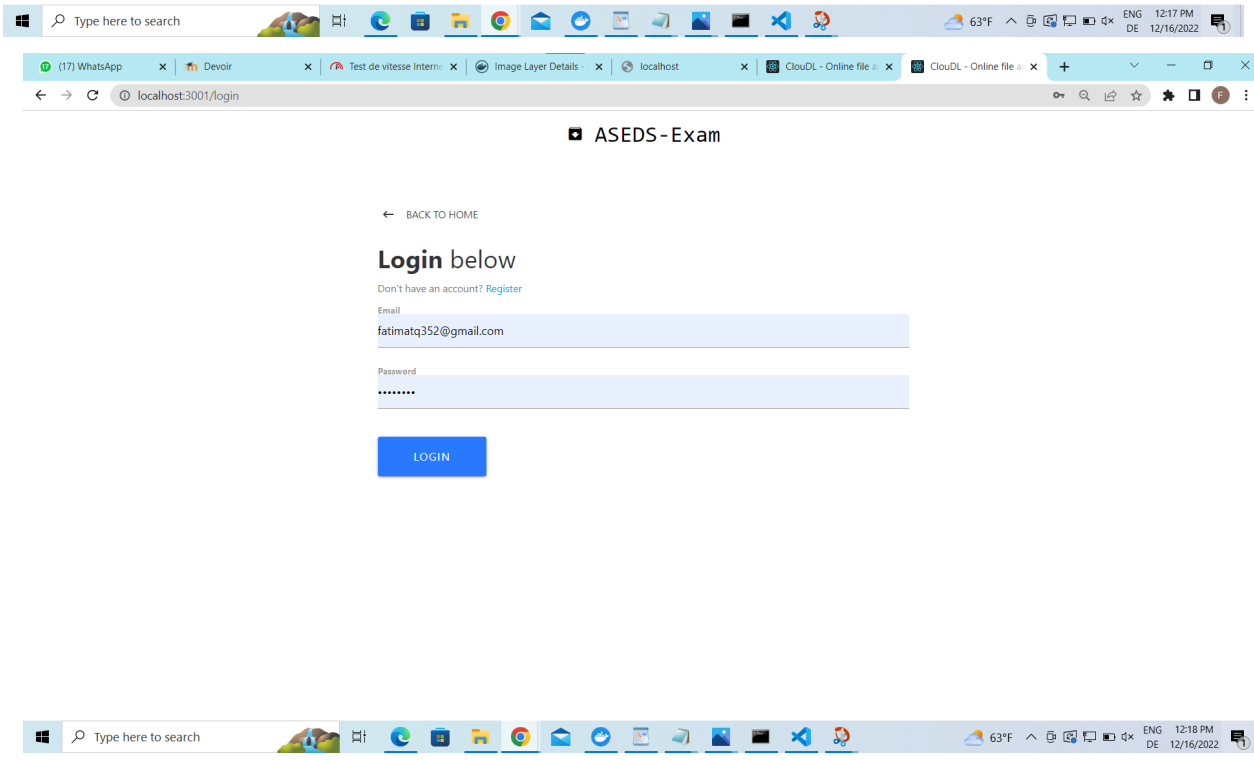
- **Création du fichier docker-compose.yml :**

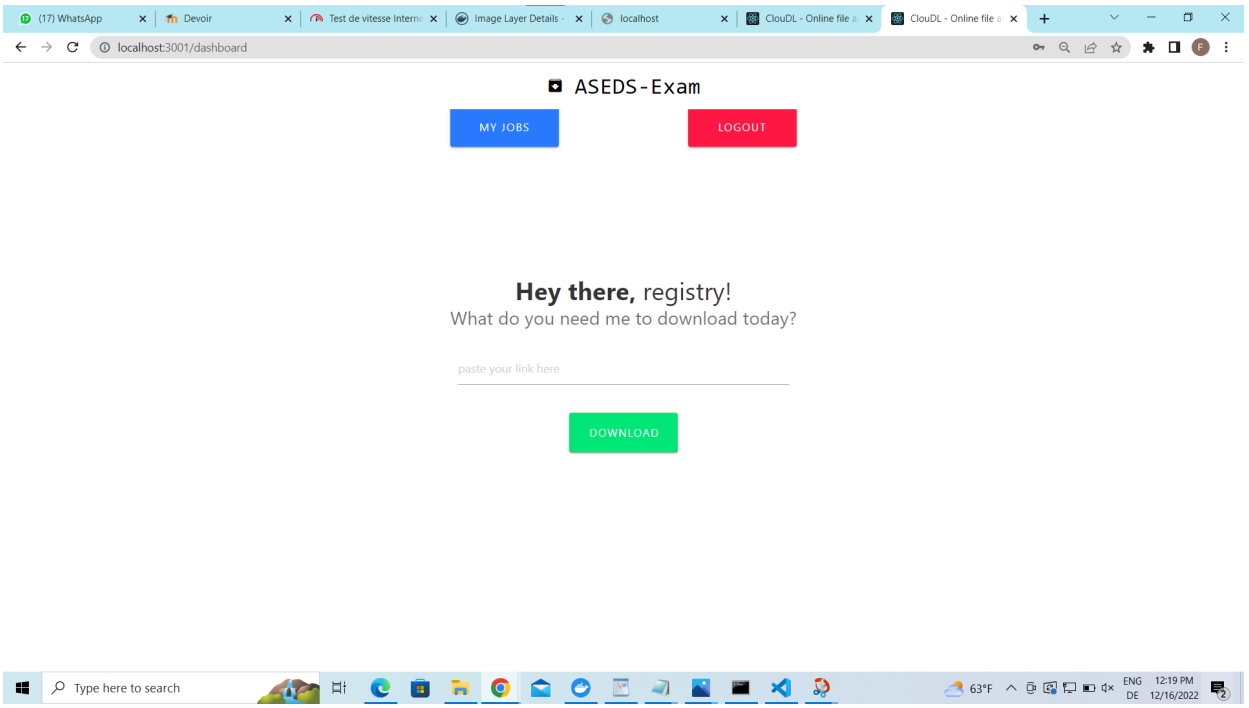
docker-compose.yaml

```yaml
version: '3'
services:
  mongo:
    image: localhost:5000/mongo
    volumes:
    - data:/data/mongodb
    container_name: mongodb-service
    ports:
    - 27017:27017
    networks:
    - netMongo

  backend:
    image: localhost:5000/backend-v1
    container_name: backend
    ports:
    - 5001:5001
    networks:
    - netMongo

  frontend:
    image: localhost:5000/frontend-v1
    container_name: frontend
    ports:
    - 3001:80
    networks:
    - netMongo

volumes:
   data:
networks:
  netMongo:
    driver: bridge
```

- On exécute la commande **docker compose up**







Partie 2 :

Pour commencer cette section, nous avons besoin d'un cluster Kubernetes. Dans notre cas, nous allons utiliser Minikube.

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\K8s> minikube start
😄  minikube v1.27.1 on Microsoft Windows 10 Pro 10.0.19045 Build 19045
✨  Using the docker driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🏃  Updating the running docker "minikube" container ...
🐳  Preparing Kubernetes v1.25.2 on Docker 20.10.18 ...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
🌟  Enabled addons: storage-provisioner, dashboard, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by defaul
```

1. La création des fichiers YAML nécessaires:

- **Créer un namespace nommé exam**
  - On utilisant la commande **kubectl create namespace exam**
  - la 2eme commande utilisée pour changer le namespace dans lequel les commandes **kubectl** s'exécutent, ce qui peut être utile lorsqu' on travaille sur plusieurs environnements ou projets différents et que vous souhaitez limiter l'exécution des commandes à un namespace spécifique.

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\K8s> kubectl create namespace exam
namespace/exam created
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\K8s> kubectl config set-context --current --namespace=exam
Context "minikube" modified.
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\K8s>
```

- **Backend**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl apply -f backend.yaml
service/backend-service created
deployment.apps/backend-deployement created
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
backend-deployement-6bb7b65cbb-8ljw6   1/1     Running   0          6s
backend-deployement-6bb7b65cbb-gghq4   1/1     Running   0          6s
mongodb-deployment-7cbd55d5d4-prwbs    1/1     Running   0          2m
mongodb-deployment-7cbd55d5d4-s6dhg    1/1     Running   0          2m
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

- On lance la commande **kubectl get service** pour récupérer le Cluster-IP du service backend et l'ajouter dans le fichier yaml backend-configmap

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get service
NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
backend-service   ClusterIP   10.108.114.122   <none>        5000/TCP    34s
mongodb-service   ClusterIP   10.101.227.182   <none>        27017/TCP   2m29s
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl apply -f backend-configmap.yaml
configmap/backend-conf created
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

- **MongoDb**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl apply -f mongodb-deployment.yaml
service/mongodb-service created
persistentvolume/mongodbpv created
persistentvolumeclaim/mongodbvolclaim created
deployment.apps/mongodb-deployment created
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-7cbd55d5d4-prwbs   1/1     Running   0          78s
mongodb-deployment-7cbd55d5d4-s6dhg   1/1     Running   0          78s
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

- **Frontend**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl apply -f frontend.yaml
service/frontend-service created
deployment.apps/frontend-deployment created
```

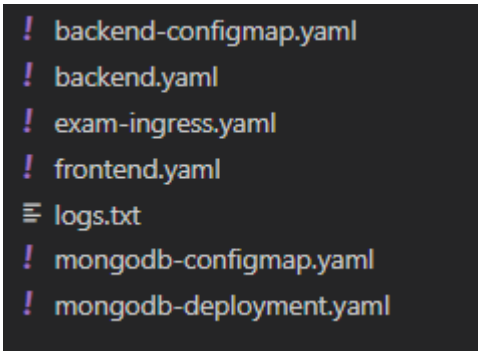
```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
backend-deployement-7d4fcb96b8-sbb4h   1/1     Running   0          4m17s
backend-deployement-7d4fcb96b8-sr65n   1/1     Running   0          4m17s
frontend-deployment-847776b7c8-267tr   1/1     Running   0          3m14s
frontend-deployment-847776b7c8-4gj2d   1/1     Running   0          3m14s
mongodb-deployment-6447c4995f-knpk4    1/1     Running   0          5m3s
mongodb-deployment-6447c4995f-zr949    1/1     Running   0          5m3s
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

- On constate que tout fonctionne bien après avoir lancé la commande.
  - **minikube dashboard**

Pods

	Nom	Images	Étiquettes	Noeud	Statut	Redémarr	Utilisation CPU (coeurs)	Utilisation mémoire (octets)	Date de création
●	frontend-deployment-847776b7c8-267tr	fatimataaoufiq/dockerproject:myfront-v1	app: frontend pod-template-hash: 847776b7c8	minikube	Running	0	-	-	47 seconds ago
●	frontend-deployment-847776b7c8-4gj2d	fatimataaoufiq/dockerproject:myfront-v1	app: frontend pod-template-hash: 847776b7c8	minikube	Running	0	-	-	47 seconds ago
●	backend-deployement-7d4fcb96b8-sbb4h	abdorty/backend-app:back-app-v1	app: back-service pod-template-hash: 7d4fcb96b8	minikube	Running	0	-	-	a minute ago
●	backend-deployement-7d4fcb96b8-sr65n	abdorty/backend-app:back-app-v1	app: back-service pod-template-hash: 7d4fcb96b8	minikube	Running	0	-	-	a minute ago
●	mongodb-deployment-6447c4995f-knpk4	mongo:latest	app: mongodb pod-template-hash: 6447c4995f	minikube	Running	0	-	-	2 minutes ago
●	mongodb-deployment-6447c4995f-zr949	mongo:latest	app: mongodb pod-template-hash: 6447c4995f	minikube	Running	0	-	-	2 minutes ago

- **Tous les fichiers yaml créés**



- **Deployment**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get deployment
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
backend-deployement   2/2     2            2           3m53s
frontend-deployment   2/2     2            2           2m50s
mongodb-deployment    2/2     2            2           4m39s
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

- **service**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get service
NAME               TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
backend            ClusterIP      10.99.250.79     <none>        5000/TCP       3m16s
frontend-service   LoadBalancer   10.100.252.212   <pending>     80:32189/TCP   2m14s
mongodb-service    ClusterIP      None             <none>        27017/TCP      4m2s
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

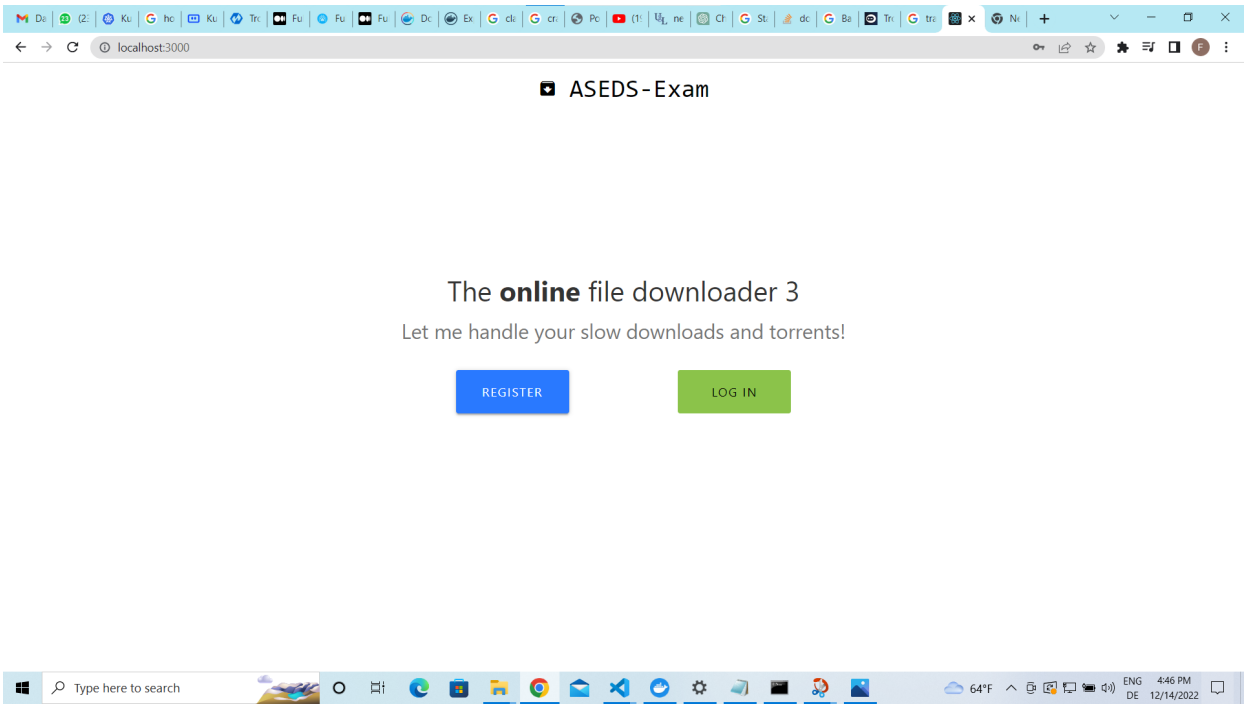
2. Créer et configurer pour cette application un :
    a. **Liveness Prob**
    b. **Readiness Prob**
    c. **Startup Prob**

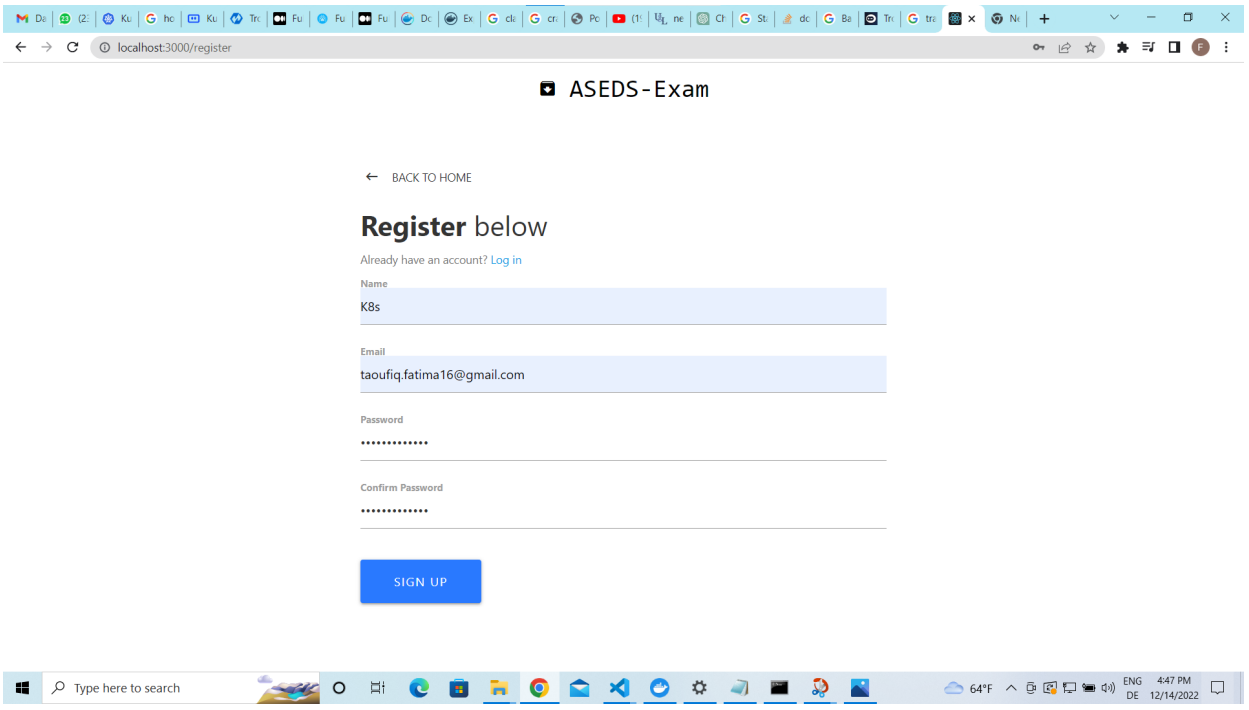
```yaml
livenessProbe:
  tcpSocket:
    port: 80
  initialDelaySeconds: 300
  periodSeconds: 200
readinessProbe:
  tcpSocket:
    port: 80
  initialDelaySeconds: 50
  periodSeconds: 100
startupProbe:
  tcpSocket:
      port: 80
  periodSeconds: 100
  failureThreshold: 3
```

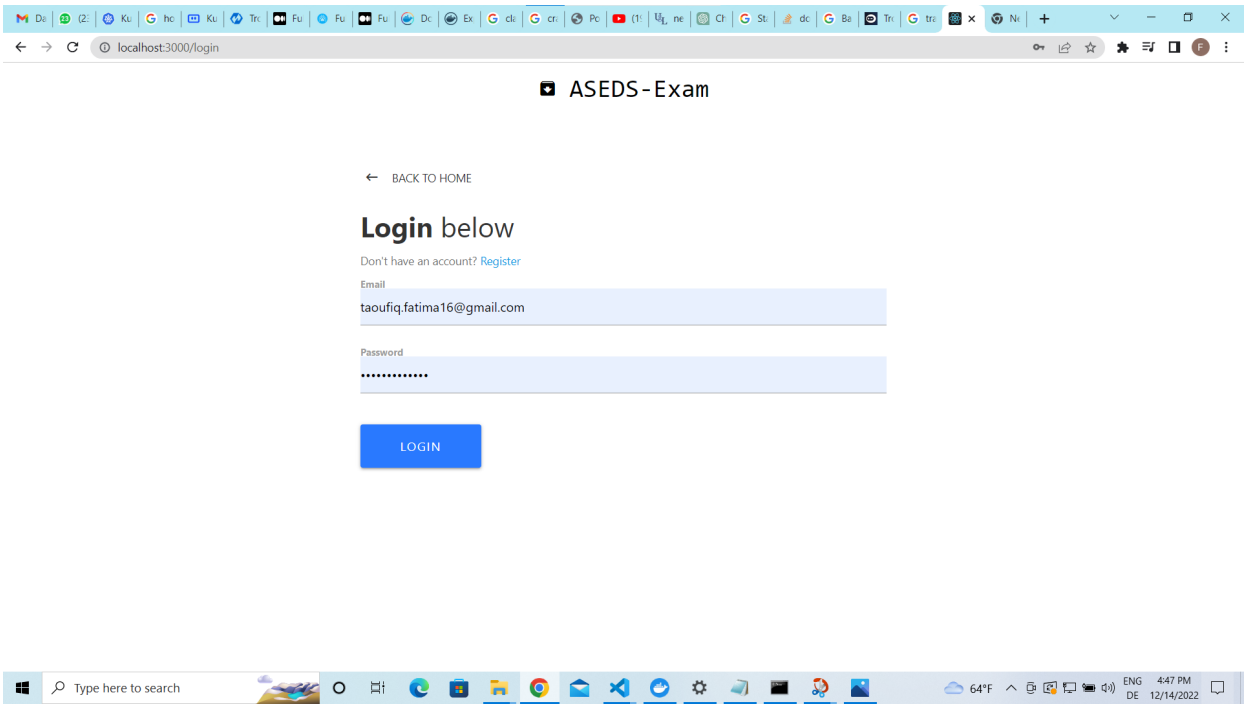
★ Pour s'assurer que tout fonctionne bien après le déploiement de tous les fichiers YAML, on teste si l'application fonctionne.

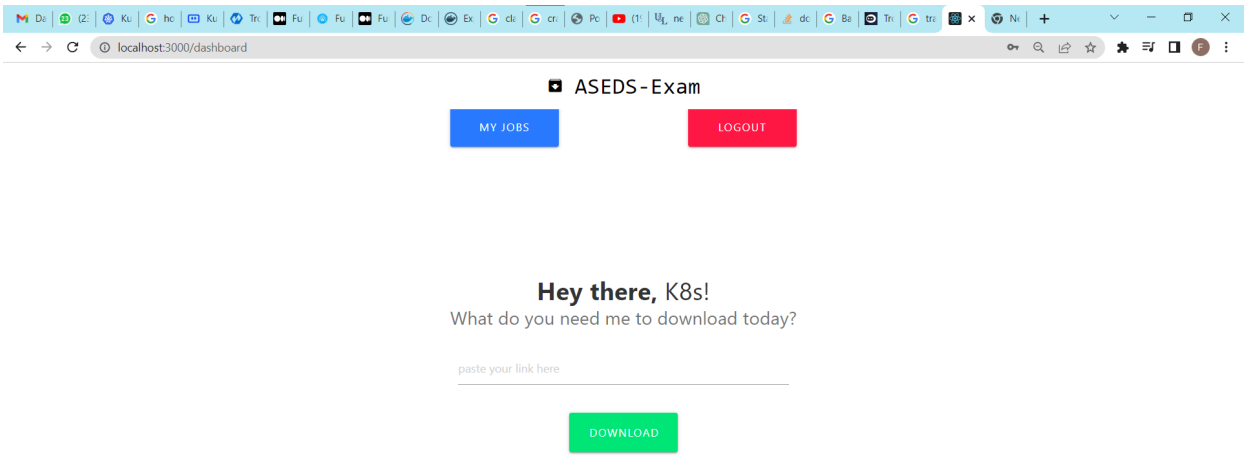
- La commande **kubectl port-forward** permet de rediriger les requêtes vers un port de votre ordinateur vers un port de votre cluster Kubernetes. Cette commande est utile lorsque vous voulez accéder à une application ou un service de votre cluster depuis votre ordinateur local, mais que ce service n'est pas accessible en dehors du cluster.

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl port-forward frontend-deployment-7f75c94b6b-mrkkn 3000:80
Forwarding from 127.0.0.1:3000 -> 80
Forwarding from [::1]:3000 -> 80
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
```


localhost:3000
ASEDS-Exam
The online file downloader 3
Let me handle your slow downloads and torrents!
REGISTER
LOG IN
Type here to search
64°F
ENG
DE
4:46 PM
12/14/2022


localhost:3000/register
ASEDS-Exam
BACK TO HOME
Register below
Already have an account? Log in
Name
K8s
Email
taoufiq.fatima16@gmail.com
Password
Confirm Password
SIGN UP
Type here to search
64°F
ENG
DE
4:47 PM
12/14/2022





3. La creation d'un Ingress pour acceder à l'application depuis un navigateur moyennant un nom de domaine:

exam-ingress.yaml

C: > Users > Fatima TAOUFIQ > Desktop > ASEDS > S3 > P1 > Docker > DockerProject > exam-ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: exam-ingress
  namespace: exam
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: "aseds-exam.com"
    http:
      paths:
        - pathType: Prefix
          path: /
          backend:
            service:
              name: frontend-service
              port:
                number: 80
```

4. Assuront que l'application a été bien conteneurisée et qu'elle fonctionne correctement:

- Pour utiliser l'addon Ingress, vous devez d'abord l'activer en utilisant la commande **minikube addons enable ingress**. Une fois l'addon activé, on crée des ressources Ingress dans notrecluster en utilisant la commande **kubectl apply -f exam-ingress.yaml**

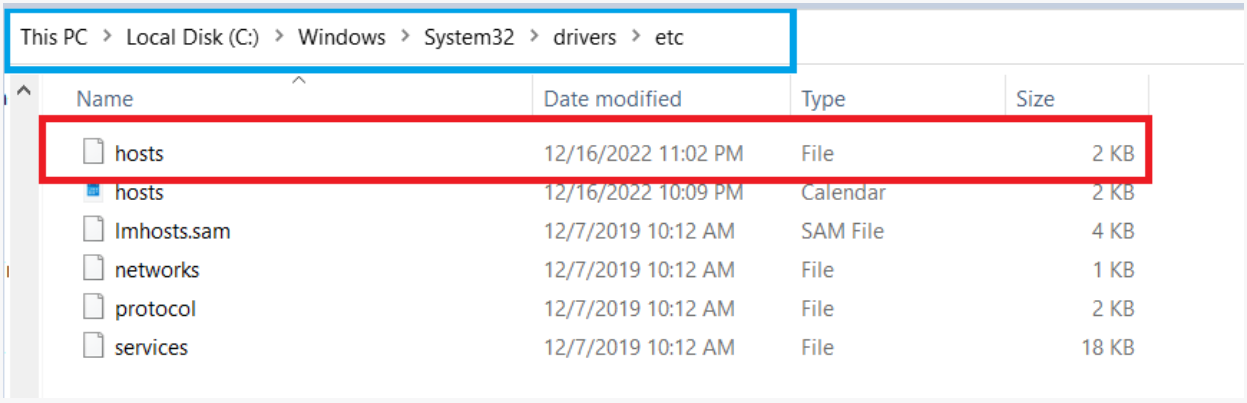
```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
💡  After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
    ▪ Using image k8s.gcr.io/ingress-nginx/controller:v1.2.1
    ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
    ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

- On déploie notre fichier yaml ingress on utilisant la commande **kubectl apply -f exam-ingress.yaml**

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl apply -f .\exam-ingres
s.yaml
ingress.networking.k8s.io/exam-ingress created
```

```
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get ingress
NAME           CLASS   HOSTS            ADDRESS   PORTS   AGE
exam-ingress   nginx   aseds-exam.com             80      10s
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject> kubectl get ingress
NAME           CLASS   HOSTS            ADDRESS        PORTS   AGE
exam-ingress   nginx   aseds-exam.com   172.21.65.58   80      46s
PS C:\Users\Fatima TAOUFIQ\Desktop\ASEDS\S3\P1\Docker\DockerProject>
```

- On accède au répertoire **C:\Windows\System32\drivers\etc**



- On ajoute notre **adresse** et l'hôte (**hostname**):

hosts - Notepad

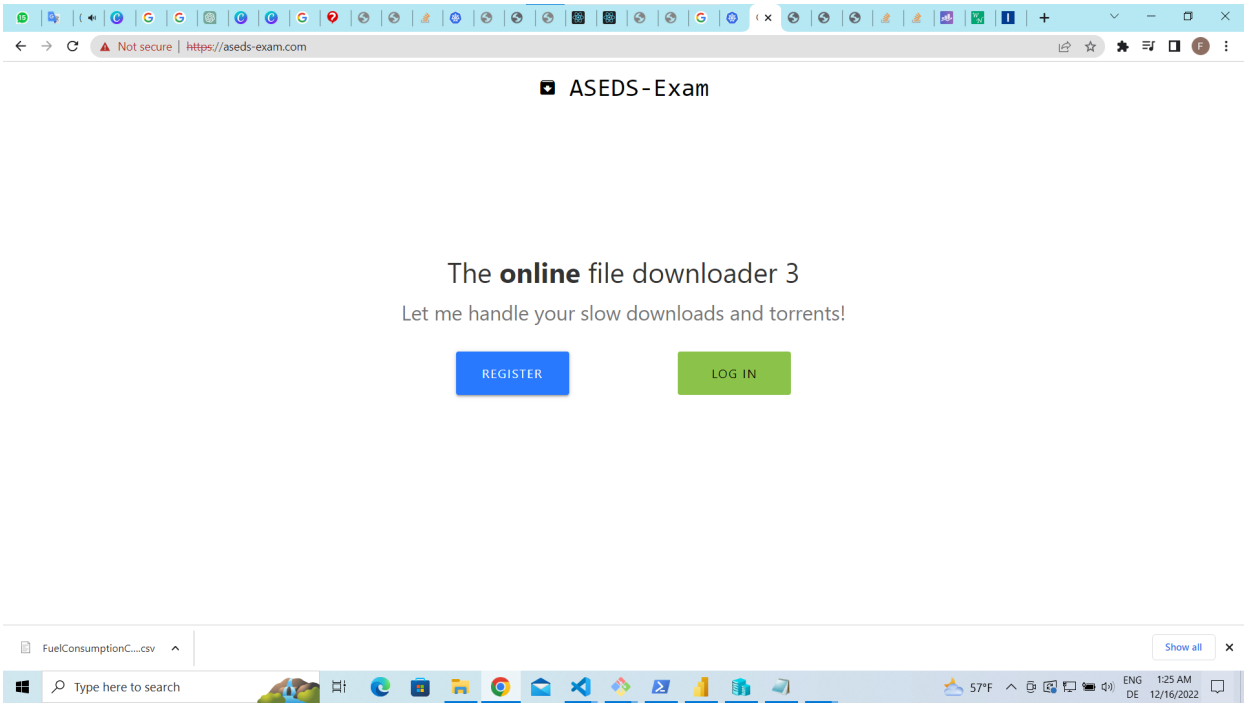
File Edit Format View Help

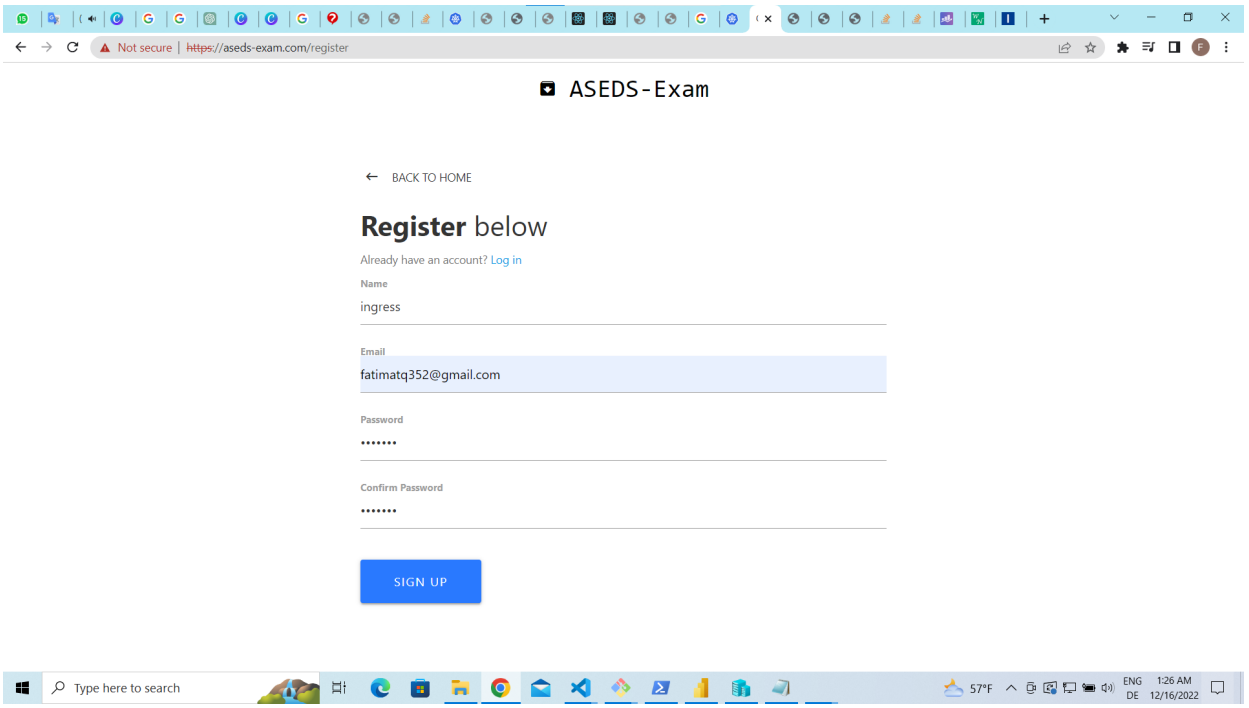
```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host
       172.21.65.58     aseds-exam.com
# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost

# Added by Docker Desktop
172.16.108.146 host.docker.internal
172.16.108.146 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
```

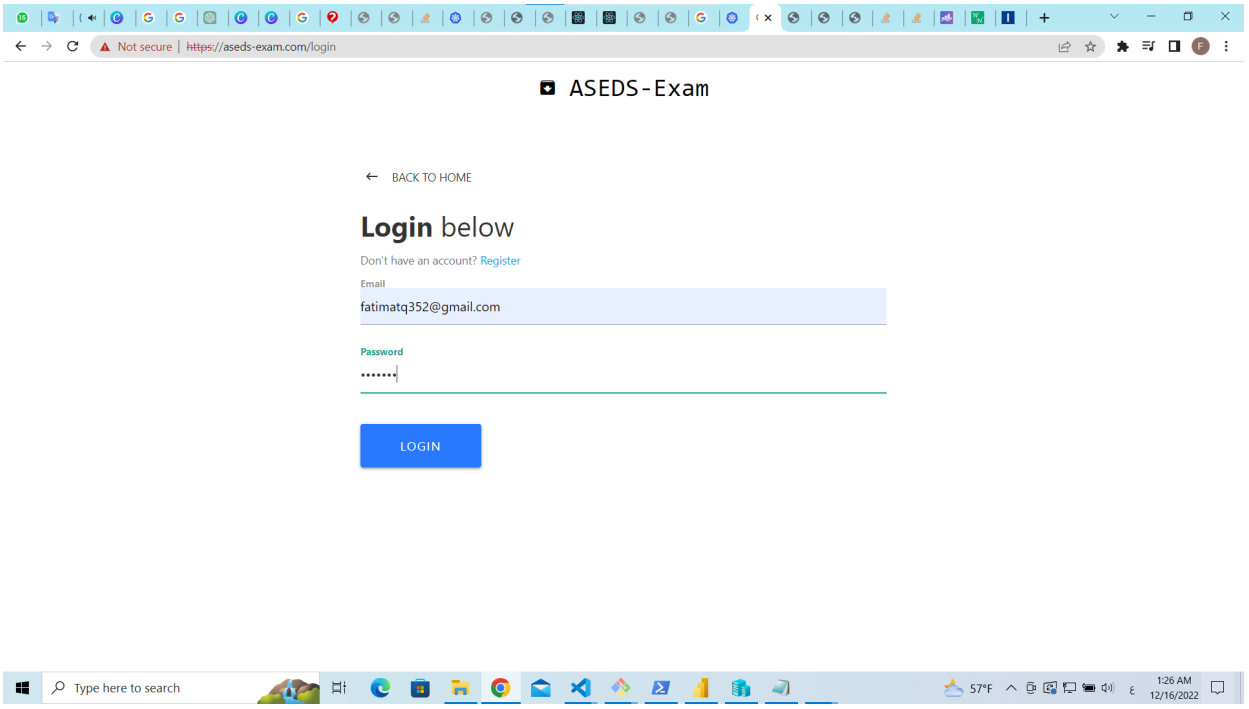
NB: Dans cette partie, j'ai travaillé avec un nouveau Minikube dans Hyper-V

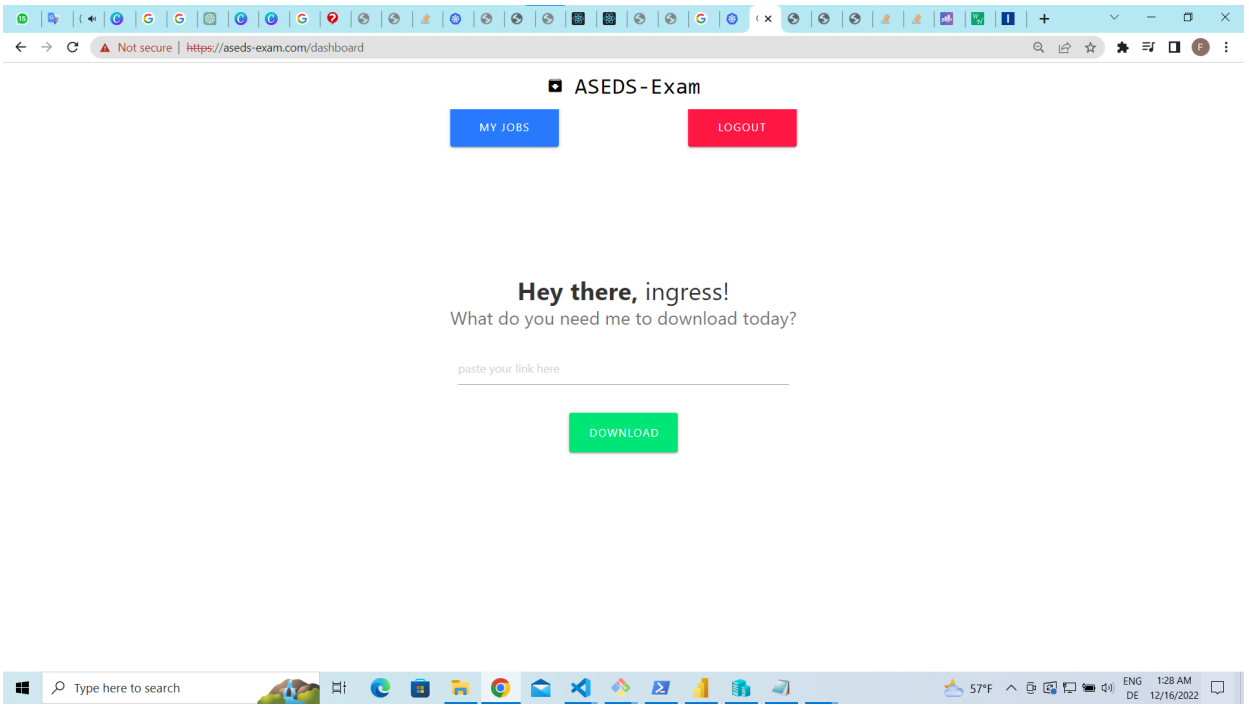
```
C:\Windows\system32>minikube start -p v2 --driver=hyperv
* [v2] minikube v1.27.1 on Microsoft Windows 10 Pro 10.0.19045 Build 19045
* Using the hyperv driver based on user configuration
* Starting control plane node v2 in cluster v2
* Creating hyperv VM (CPUs=2, Memory=6000MB, Disk=20000MB) ...
* Preparing Kubernetes v1.25.2 on Docker 20.10.18 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "v2" cluster and "default" namespace by default

C:\Windows\system32>
```

- TEST




Not secure | https://aseds-exam.com/register
ASEDS-Exam
BACK TO HOME
Register below
Already have an account? Log in
Name
ingress
Email
fatimatq352@gmail.com
Password
Confirm Password
SIGN UP
Type here to search
57°F
ENG
DE
1:26 AM
12/16/2022


Not secure | https://aseds-exam.com/login
ASEDS-Exam
BACK TO HOME
Login below
Don't have an account? Register
Email
fatimatq352@gmail.com
Password
LOGIN
Type here to search
57°F
1:26 AM
12/16/2022



➔ **Lien repository Github:**

https://github.com/Fatimatq/Docker-exam