



Rapport Technique administration système et docker

Encadré par :
Mr.GHAILANI MOHAMED

rédigé par :
Wail Hadad
Jebari Hassani Abdelhak

Rapport Administration Système

Plan :

- **Création d'une application CRUD simple par Laravel et Vuejs pour le test de docker ultérieurement.**
- **Test des différentes requêtes en utilisant Postman.**
- **Test de l'application et son fonctionnement et interaction avec la base de données.**
- **Push dans un dépôt GitHub de mon compte privé.**
- **Continuation du travail : ajout des fichier docker (conteneurisation).**
- **Création des Dockerfiles (backend et frontend).**
- **Création du docker-compose.yml et des différents services.**
- **Configuration DNS POSTFIX ET NGINX dans une machine linux debian**

Image backend

Image frontend

Image postgreSQL

Image Nginx

Ajout des certificats SSL ([http->https](http://>https))

Image pgadmin (UI pour visualiser la base de données)

Image dnsmasq :

(Configuration du DNS pour la résolution du nom de domaine enfant.innova.ma)

Image mailhog (pour la visualisation des emails et notifications)

- Push vers le dépôt GitHub avec tous les fichiers docker et les mises à jour.
- Push vers dockerhub à l'aide des GitHub actions CI.
- Test des images délivrées à dockerhub dans une VM en utilisant SSH.
- Push des fichiers docker dans notre dépôt de production après avoir vérifié l'environnement et la configuration.

1) Création d'une application CRUD simple par Laravel et Vuejs.

```

C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [version 10.0.22631.3447]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\LENOVO>cd \
C:\>cd ./app/api

C:\app\api>php artisan serv
PHP Warning: Module "mysqli" is already loaded in Unknown on line 0
[ WARN ] PHP Warning: Module "mysqli" is already loaded in Unknown on line 0

[ INFO ] Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server

C:\WINDOWS\system32\cmd. x + v
DONE Compiled successfully in 3168ms
App running at:
- Local: http://localhost:8080/
- Network: unavailable

Note that the development build is not optimized.
To create a production build, run npm run build.
  
```

Evaluation de la route de ressource api dans le navigateur :

Route	JSON Response Data
localhost:8000/api/Profs	<pre>[{ "id": 1, "nom": "l3asri", "prenom": "siham", "spécialité": "management", "created_at": "2024-04-30T02:54:07.000000Z", "updated_at": "2024-04-30T02:54:07.000000Z" }, { "id": 3, "nom": "daghmoumi", "prenom": "iliyas", "spécialité": "logistique", "created_at": "2024-04-30T21:02:10.000000Z", "updated_at": "2024-04-30T21:02:10.000000Z" }, { "id": 4, "nom": "allam", "prenom": "oussama", "spécialité": "logistique", "created_at": "2024-05-01T00:11:59.000000Z", "updated_at": "2024-05-01T00:11:59.000000Z" }, { "id": 2, "nom": "isami", "prenom": "taha", "spécialité": "tnawi", "created_at": "2024-04-30T19:44:47.000000Z", "updated_at": "2024-04-30T19:44:47.000000Z" }]</pre>
localhost:8000/api/Etudiants	<pre>[{ "id": 8, "nom": "moussa", "prenom": "oussama", "note": "22", "created_at": "2024-05-01T00:10:14.000000Z", "updated_at": "2024-05-01T00:10:14.000000Z" }, { "id": 9, "nom": "allam", "prenom": "mohammed", "note": "17.5", "created_at": "2024-05-01T00:10:32.000000Z", "updated_at": "2024-05-01T00:10:32.000000Z" }, { "id": 10, "nom": "al haddad", "prenom": "abd arrazzaq", "note": "18", "created_at": "2024-05-01T00:11:02.000000Z", "updated_at": "2024-05-01T00:11:17.000000Z" }]</pre>

localhost8080 :



[**>>Gestion des Étudiant**](#)

[**>>Gestion des Prof**](#)

2024 by Wail Hadad.Thanks for visiting the site.

Connexion réussie avec pgsql

Welcome to Wail Hadad App !!

Page d'accueil about me

Gestion des étudiants

Nom
Entrer le nom de l'étudiant

Prénom
Entrer le prénom étudiant

Note
Entrer la note de l'étudiant

Save

ID	Nom	Prénom	Note	Retirer
9	allam	mohammed	17.5	Edit / Delete
10	al haddad	abd arrazza9	18	Edit / Delete
12	9adiri	imad	18	Edit / Delete

2) Test des Requêtes.

Evaluation de la connexion avec PostgreSQL et test des différentes requêtes http

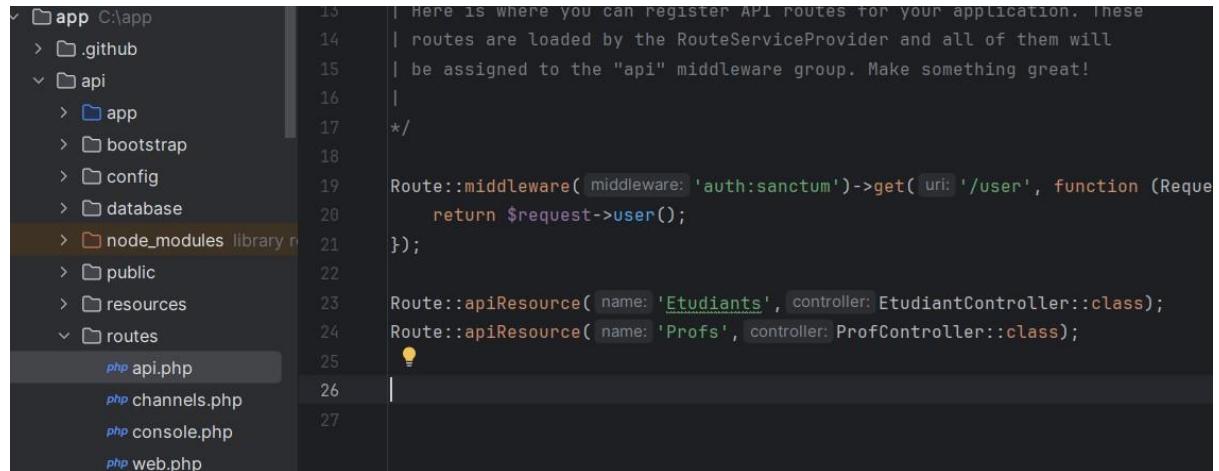
Php artisan route:list

Testons ces routes dans postman :

```
Terminal Local × + ▾
PS C:\app> cd ./api
PS C:\app\api> php artisan route:list
PHP Warning: Module "mysqli" is already loaded in Unknown on line 0

POST          _ignition/execute-solution .....
GET|HEAD      _ignition/health-check .....
POST          _ignition/update-config .....
GET|HEAD      api/Etudiants .....
POST          api/Etudiants .....
GET|HEAD      api/Etudiants/{Etudiant} .....
PUT|PATCH    api/Etudiants/{Etudiant} .....
DELETE        api/Etudiants/{Etudiant} .....
DELETE        api/Etudiants/{Etudiant} .....
GET|HEAD      api/Profs .....
POST          api/Profs .....
GET|HEAD      api/Profs/{Prof} .....
PUT|PATCH    api/Profs/{Prof} .....
DELETE        api/Profs/{Prof} .....
GET|HEAD      api/user .....
GET|HEAD      sanctum/csrf-cookie .....
```

On a 2 contrôleur de ressource api tel que :

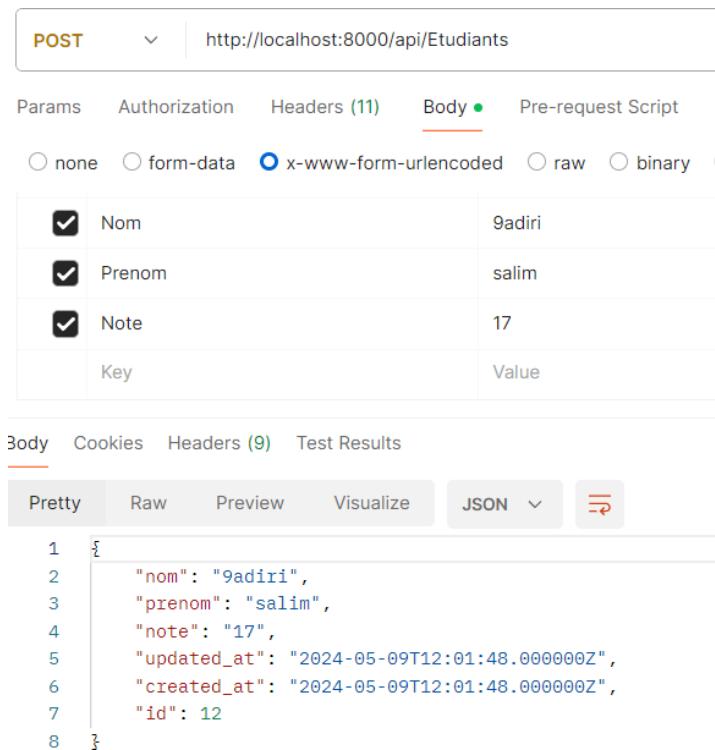


A screenshot of a file explorer window showing the directory structure of a Laravel application. The 'routes/api.php' file is selected and highlighted. The code in this file defines several API routes, including routes for users, etudiants, and profs.

```
13 | Here is where you can register API routes for your application. These
14 | routes are loaded by the RouteServiceProvider and all of them will
15 | be assigned to the "api" middleware group. Make something great!
16 |
17 */
18
19 Route::middleware(['auth:sanctum'])->get('/user', function (Request $request) {
20     return $request->user();
21 });
22
23 Route::apiResource('Etudiants', EtudiantController::class);
24 Route::apiResource('Profs', ProfController::class);
25
26
27
```

Route api/Etudiants

Post / POST <http://localhost:8000/api/Etudiants>

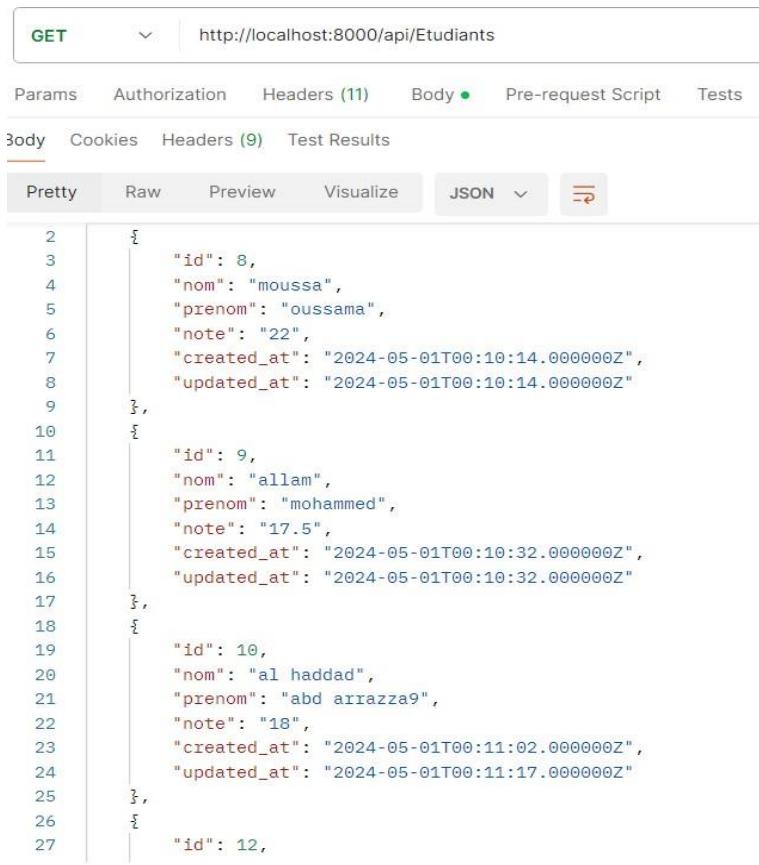


The screenshot shows the Postman application interface. A POST request is being made to the URL <http://localhost:8000/api/Etudiants>. The request body is set to 'x-www-form-urlencoded' and contains three key-value pairs: 'Nom' (value: 9adiri), 'Prenom' (value: salim), and 'Note' (value: 17). The response is displayed in JSON format, showing a single object with keys 'nom', 'prenom', 'note', 'updated_at', 'created_at', and 'id'.

Key	Value
Nom	9adiri
Prenom	salim
Note	17

```
1 {
2     "nom": "9adiri",
3     "prenom": "salim",
4     "note": "17",
5     "updated_at": "2024-05-09T12:01:48.000000Z",
6     "created_at": "2024-05-09T12:01:48.000000Z",
7     "id": 12
8 }
```

Get all / GET <http://localhost:8000/api/Etudiants>



GET <http://localhost:8000/api/Etudiants>

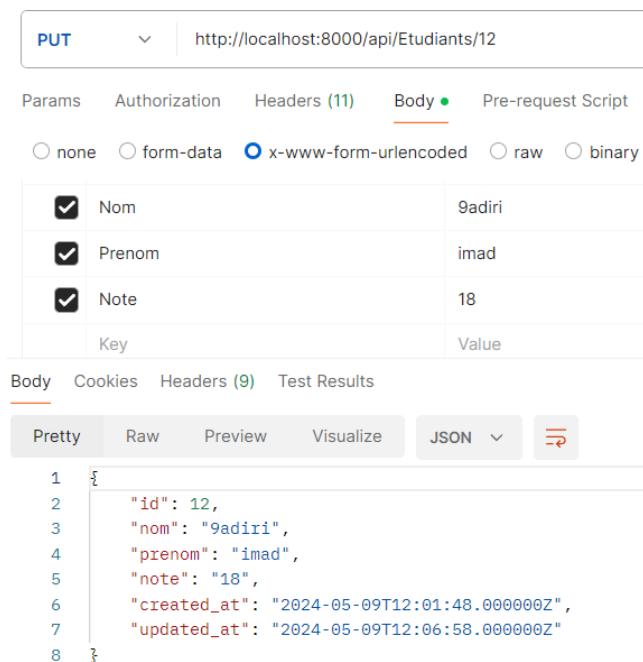
Params Authorization Headers (11) Body Pre-request Script Tests

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON ↻

```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
{
  "id": 8,
  "nom": "moussa",
  "prenom": "oussama",
  "note": "22",
  "created_at": "2024-05-01T00:10:14.000000Z",
  "updated_at": "2024-05-01T00:10:14.000000Z"
},
{
  "id": 9,
  "nom": "allam",
  "prenom": "mohammed",
  "note": "17.5",
  "created_at": "2024-05-01T00:10:32.000000Z",
  "updated_at": "2024-05-01T00:10:32.000000Z"
},
{
  "id": 10,
  "nom": "al haddad",
  "prenom": "abd arrazzaq",
  "note": "18",
  "created_at": "2024-05-01T00:11:02.000000Z",
  "updated_at": "2024-05-01T00:11:17.000000Z"
},
{
  "id": 12,
```

Update : PUT <http://localhost:8000/api/Etudiants/12>



PUT <http://localhost:8000/api/Etudiants/12>

Params Authorization Headers (11) Body Pre-request Script

none form-data x-www-form-urlencoded raw binary

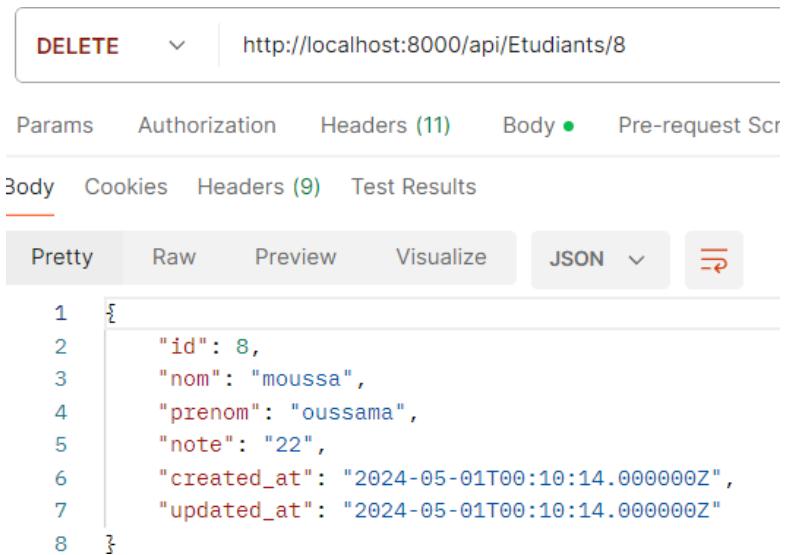
Key	Value
Nom	9adiri
Prenom	imad
Note	18

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1
2
3
4
5
6
7
8
{
  "id": 12,
  "nom": "9adiri",
  "prenom": "imad",
  "note": "18",
  "created_at": "2024-05-09T12:01:48.000000Z",
  "updated_at": "2024-05-09T12:06:58.000000Z"
```

Delete : DELETE http://localhost:8000/api/Etudiants/8



The screenshot shows a Postman request configuration. The method is set to **DELETE** and the URL is <http://localhost:8000/api/Etudiants/8>. The **Body** tab is selected, showing a JSON payload:

```
1 {  
2     "id": 8,  
3     "nom": "moussa",  
4     "prenom": "oussama",  
5     "note": "22",  
6     "created_at": "2024-05-01T00:10:14.000000Z",  
7     "updated_at": "2024-05-01T00:10:14.000000Z"  
8 }
```

Même chose pour la route /api/Profs

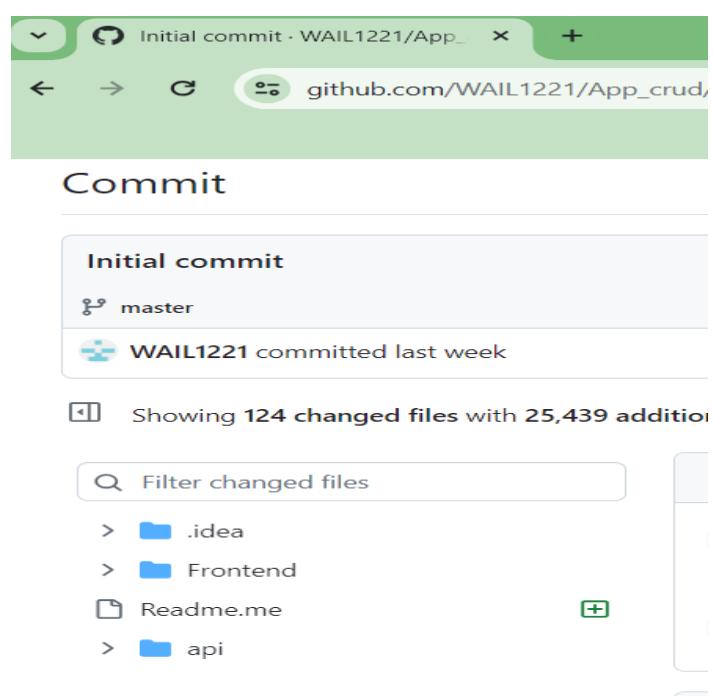
3) Push dans un dépôt GitHub de mon compte privé.

Écrivons ces commandes dans le terminal :

D'abord on copie le dépôt dans un chemin de votre choix :

git clone https://github.com/WAIL1221/App_crud.git

```
cd App_crud  
git add .  
git commit -m "Your  
descriptive commit  
message here"  
git push origin main
```



4) Création des Dockerfiles (backend et frontend).

Dockerfile du backend :

```
Project ▾ Dockerfile x
C:\app
  app
    .github
    api
      app
      bootstrap
      config
      database
      node_modules
      public
      resources
    routes
      api.php
      channels.php
      console.php
      web.php
    storage
    tests
    vendor
    .editorconfig
    .env
    .gitattributes
    .gitignore
    .phpunit.result.cache
    artisan
    composer.json
    composer.lock
    Dockerfile
    package.json
```

```
1 FROM php:8.2-fpm
2
3 # Install system dependencies
4 RUN apt-get update && apt-get install -y libpng-dev libonig-dev libxml2-dev zip curl unzip git libpq-dev
5
6 # Install PHP extensions
7 RUN docker-php-ext-install pdo_pgsql mbstring exif pcntl bcmath gd
8
9 # Get latest Composer
10 COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
11
12 # Set working directory
13 WORKDIR /var/www
14
15 # Copy existing application directory contents
16 COPY . /var/www
17
18 # Set permissions
19 RUN chown -R www-data:www-data /var/www
20
21 # Expose port
22 EXPOSE 8000
23
24 # Start-up script
25 CMD ["php-fpm"]
26
```

1. **FROM php:8.2-fpm** : Nous définissons l'image de base à utiliser pour la construction de notre conteneur Docker. Dans ce cas, nous utilisons l'image PHP version 8.2 avec FPM (FastCGI Process Manager), ce qui signifie que notre application PHP sera exécutée dans un environnement compatible avec PHP 8.2 et configuré pour le traitement des requêtes FastCGI.
2. **RUN apt-get update && apt-get install -y libpng-dev libonig-dev libxml2-dev zip curl unzip git libpq-dev** : Nous mettons à jour les référentiels APT et installons les dépendances système nécessaires à notre application. Cela inclut des bibliothèques et des outils tels que libpng, libonig, libxml2, zip, curl, unzip, git et libpq.
3. **RUN docker-php-ext-install pdo_pgsql mbstring exif pcntl bcmath gd** : Nous installons des extensions PHP spécifiques à notre application à l'aide de l'outil **docker-php-ext-install**. Ces extensions comprennent pdo_pgsql, mbstring, exif, pcntl, bcmath et gd.
4. **COPY --from=composer:latest /usr/bin/composer /usr/bin/composer** : Nous copions l'exécutable Composer depuis une autre image Docker contenant Composer vers l'emplacement approprié dans notre conteneur. Cela nous permet d'utiliser Composer pour gérer les dépendances de notre application PHP.

Dockerfile du frontend :

The screenshot shows a code editor interface with two tabs: "api\Dockerfile" and "Frontend\Dockerfile". The "Frontend\Dockerfile" tab is active, displaying the following Dockerfile content:

```
1 FROM node:16
2
3 # Set working directory
4 WORKDIR /app
5
6 # Copy package.json and package-lock.json
7 COPY ./package*.json .
8
9 # Install app dependencies
10 RUN npm install
11
12 # Bundle app source
13 COPY . .
14
15 # Expose port
16 EXPOSE 8080
17
18 # Start command
19 CMD ["npm", "run", "serve"]
```

The left sidebar shows the project structure under "app C:\app":

- .github
- api
- Frontend
 - node_modules
 - public
 - src
 - assets
 - components
 - router
 - App.vue
 - main.js
 - .gitignore
 - babel.config.js
 - Dockerfile
 - jsconfig.json
 - package.json
 - package-lock.json
 - README.md
 - vue.config.js

5) Création du docker-compose.yml et des différents services.

```

    dockerfile: Dockerfile
    command: npm run serve
    ports:
      - "8080:8080"

    depends_on:
      - backend

nginx:
  image: nginx:alpine

#après le push vers de cette image vers dockerhub sous un tag :
#nginx-custom dans le répertoire wailhadad1221/project_image, on change
#le nom de l'image pour pointer vers le répertoire dockerhub
#wailhadad1221/project_image lors de l'exécution des GitHub actions
# □ image : wailhadad1221/project_image :nginx-custom au lieu de
# □ image: nginx:alpine

  ports:
    - "80:80"
    - "443:443"
  volumes:
    - ./nginx/nginx.conf:/etc/nginx/conf.d/default.conf
    - ./nginx/certs:/etc/ssl/certs:ro
    - ./nginx/private:/etc/ssl
  /private:ro depends_on:
    - frontend
    - backend
    - dnsmasq

dnsmasq:
  image : andyshinn/dnsmasq:2.78

#après le push vers de cette image vers dockerhub sous un tag :
#dnsmasq dans le répertoire wailhadad1221/project_image, on change #le
#nom de l'image pour pointer vers le répertoire dockerhub
#wailhadad1221/project_image lors de l'exécution des GitHub actions
# □ image : wailhadad1221/project_image: dnsmasq au lieu de
# □ image: andyshinn/dnsmasq:2.78

  cap_add:
    - NET_ADMIN
  ports:
    - "53:53/tcp"
    - "53:53/udp"
  volumes:
    - ./hosts:/etc/hosts-custom

db:
  image: postgres:latest

#après le push vers de cette image vers dockerhub sous un tag :
#postgres dans le répertoire wailhadad1221/project_image, on change
#le nom de l'image pour pointer vers le répertoire dockerhub
#wailhadad1221/project_image lors de l'exécution des GitHub actions
# □ image : wailhadad1221/project_image: postgres au lieu de
# □ image: postgres :latest

  environment:
    POSTGRES_DB: myapp

```

Image backend

1. **version: '3.8'** : Nous spécifions la version du format de fichier docker-compose que nous utilisons.
2. **services:** : Nous définissons les services (ou conteneurs) qui composent notre application.
3. **backend:** : Nous nommons notre service backend.
image: voir commentaires du code docker-compose.yml .
4. **build:** : Nous spécifions comment construire l'image Docker pour ce service.
 - **context: ./api** : Nous indiquons le chemin vers le répertoire contenant les fichiers nécessaires à la construction de l'image. Dans ce cas, les fichiers se trouvent dans le répertoire **./api**.
 - **dockerfile: Dockerfile** : Nous spécifions le nom du fichier Dockerfile à utiliser pour la construction de l'image. Dans ce cas, le fichier Dockerfile s'appelle simplement "**Dockerfile**".
5. **volumes:** : Nous montons un volume pour le service.
 - **./api:/var/www** : Nous montons le répertoire local **./api** dans le répertoire **/var/www** à l'intérieur du conteneur. Cela permet de synchroniser les fichiers de l'application entre l'hôte et le conteneur.
6. **depends_on:** : Nous spécifions les dépendances de ce service.
 - **db** : Nous indiquons que ce service dépend du service nommé "**db**". Cela signifie que le service "backend" attendra que le service "**db**" soit démarré avant de démarrer lui-même.
7. **environment:** : Nous configurons les variables d'environnement pour notre service.
 - Les variables d'environnement comme APP_NAME, APP_ENV, APP_KEY, etc., sont configurées pour l'application Laravel qui s'exécutera dans ce conteneur.
8. **command:** : Nous spécifions la commande à exécuter lorsque le conteneur est démarré.
 - Nous utilisons une chaîne de commandes **bash (bash -c)** pour exécuter plusieurs commandes séquentielles. Dans ce cas, nous installons les dépendances avec Composer, générons une clé d'application, migrons la base de données et démarrons le serveur PHP artisan.
9. **ports:** : Nous exposons les ports de notre service.
 - **"8000:8000"** : Nous faisons correspondre le port 8000 de l'hôte avec le port 8000 du conteneur, permettant ainsi d'accéder à l'application Laravel à travers le port 8000 de l'hôte.

Image frontend

1. **frontend:** : Nous nommons notre service frontend.
2. **image:** voir commentaires du code docker-compose.yml
3. **build:** : Nous spécifions comment construire l'image Docker pour ce service.
 - **context: ./Frontend** : Nous indiquons le chemin vers le répertoire contenant les fichiers nécessaires à la construction de l'image. Dans ce cas, les fichiers se trouvent dans le répertoire **./Frontend**.
 - **dockerfile: Dockerfile** : Nous spécifions le nom du fichier Dockerfile à utiliser pour la construction de l'image. Dans ce cas, le fichier Dockerfile s'appelle **Dockerfile**".

4. **commande: npm run serve** : Nous spécifions la commande à exécuter lorsque le conteneur est démarré. Dans ce cas, nous exécutons la commande `npm run server` pour démarrer le serveur de développement du frontend.
5. **ports:** : Nous exposons les ports de notre service.
 - "8080:8080" : Nous faisons correspondre le port 8080 de l'hôte avec le port 8080 du conteneur, permettant ainsi d'accéder au frontend à travers le port 8080 de l'hôte.
6. **depends_on:** : Nous spécifions les dépendances de ce service.
 - **backend** : Nous indiquons que ce service dépend du service nommé "backend". Cela signifie que le service "frontend" attendra que le service "backend" soit démarré avant de démarrer lui-même, ce qui peut être utile si le frontend a besoin de communiquer avec le backend lors de son démarrage.

Image PostgreSQL

Image Nginx

1. **nginx:** : Nous nommons notre service `nginx`.
2. **image: nginx:alpine** : Nous spécifions l'image Docker à utiliser pour ce service. Ici, nous utilisons l'image légère d'nginx basée sur Alpine Linux, qui est une distribution Linux très légère.
3. **ports:** : Nous exposons les ports de notre service.
 - "80:80" : Nous faisons correspondre le port 80 de l'hôte avec le port 80 du conteneur nginx, permettant ainsi d'accéder au serveur HTTP.
 - "443:443" : Nous faisons correspondre le port 443 de l'hôte avec le port 443 du conteneur nginx, permettant ainsi d'accéder au serveur HTTPS.
4. **volumes:** : Nous montons des volumes pour notre service.
 - `./nginx/nginx.conf:/etc/nginx/conf.d/default.conf` : Nous montons notre fichier de configuration nginx personnalisé depuis le répertoire local `./nginx/nginx.conf` vers le répertoire de configuration nginx dans le conteneur, remplaçant ainsi le fichier de configuration par défaut. Cela nous permet de personnaliser la configuration nginx selon nos besoins.

- `./nginx/certs:/etc/ssl/certs:ro` : Nous montons le répertoire contenant nos certificats SSL depuis le répertoire local `./nginx/certs` vers le répertoire des certificats SSL nginx dans le conteneur en lecture seule.
- `./nginx/private:/etc/ssl/private:ro` : Nous montons le répertoire contenant nos clés privées SSL depuis le répertoire local `./nginx/private` vers le répertoire des clés privées SSL nginx dans le conteneur en lecture seule.

5. `depends_on` : Nous spécifions les dépendances de ce service.

- **frontend** : Nous indiquons que ce service dépend du service nommé "frontend". Cela signifie que le service "nginx" attendra que le service "frontend" soit démarré avant de démarrer lui-même, ce qui peut être utile si le nginx doit rediriger le trafic vers le frontend.
- **backend** : Nous indiquons que ce service dépend du service nommé "backend". Cela signifie que le service "nginx" attendra que le service "backend" soit démarré avant de démarrer lui-même, ce qui peut être utile si le nginx doit rediriger le trafic vers le backend.
- **dnsmasq** : Nous indiquons que ce service dépend du service nommé "dnsmasq". Cela signifie que le service "nginx" attendra que le service "dnsmasq" soit démarré avant de démarrer lui-même.

Ajout des certificats SSL ([http->https](#))

Voici la structure de notre projet :

Lors de l'inclusion du certificat SSL

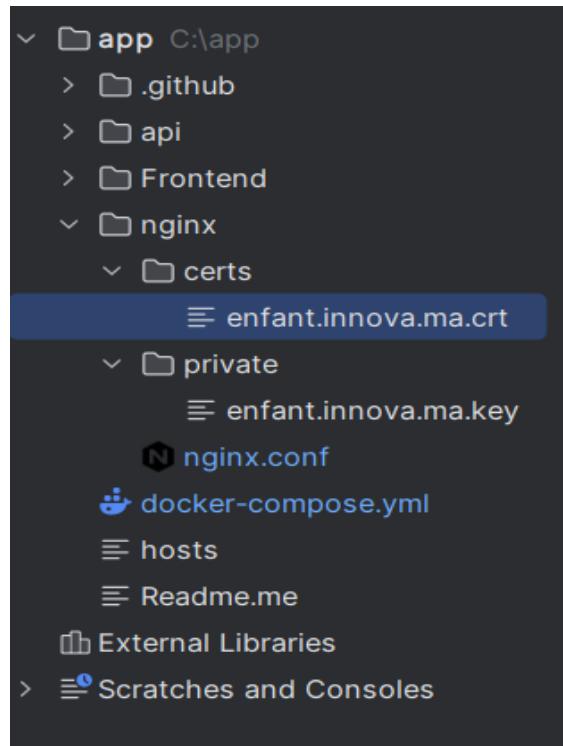
**Il faut absolument respecter les
chemin relatifs vers**

[enfant.innova.ma.crt](#)

Et

[enfant.innova.ma.key](#)

**[dans nginx.config et](#)
[docker-compose.yml](#)**



Génération du certificat SSL (self-signed) . Ce genre de certificats, n'est pas validé par les navigateurs et les autorités de certification CA ex : CloudFlare, Go Daddy.

Mais désormais il permet l'envoie des requêtes chiffrées en https.

Une autorité de certification (CA) valide les certificats SSL pour garantir l'authenticité et la sécurité des sites web. Cependant, une auto-signature peut être utile pour des tests locaux ou des intranets où la validation d'une CA tierce n'est pas nécessaire. Cela permet de chiffrer les données échangées sans avoir à passer par une CA externe.

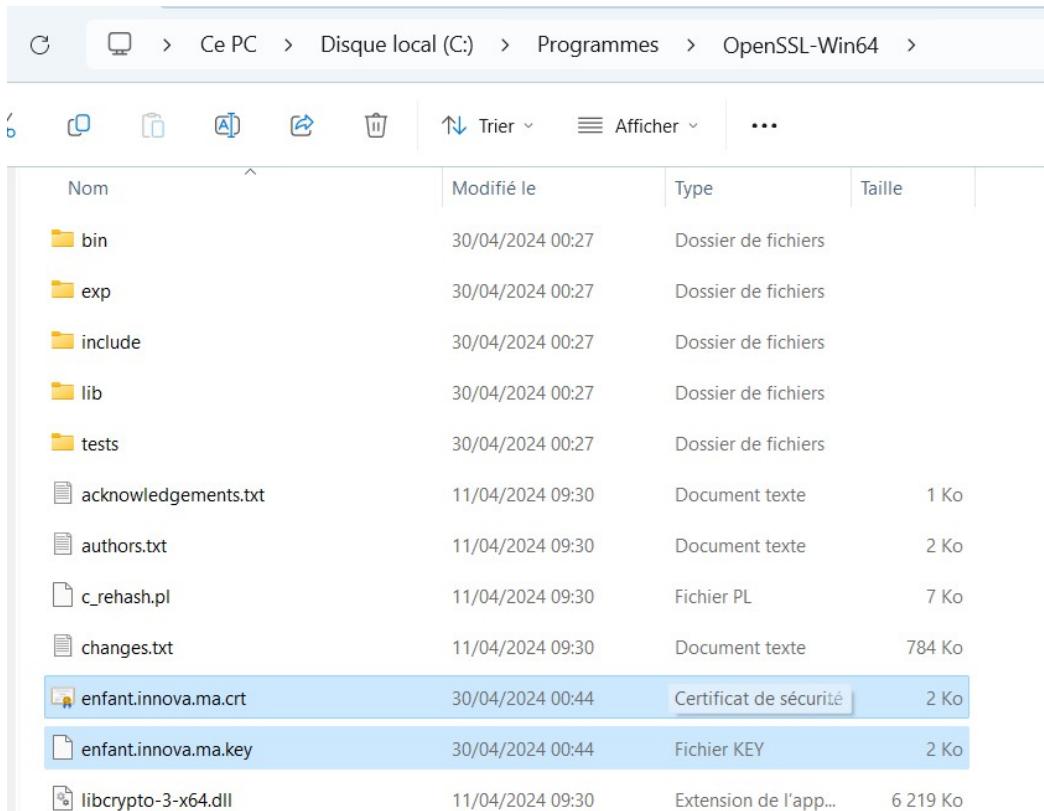
Processus de génération du certificat :

Il faut exécuter les commandes suivantes dans la terminale, il vaut mieux les exécuter en tant qu'administrateur.

- 1) Installer OpenSSL**
- 2) Localiser le dossier OpenSSL installé : cd C:\Program Files\OpenSSL-Win64**
- 3) Génération du certificat :**
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout enfant.innova.ma.key -out enfant.innova.ma.crt -subj "/CN=enfant.innova.ma" -addext "subjectAltName=DNS:enfant.innova.ma,DNS:www.enfant.innova.ma"
- 4) Exemple de commande :**

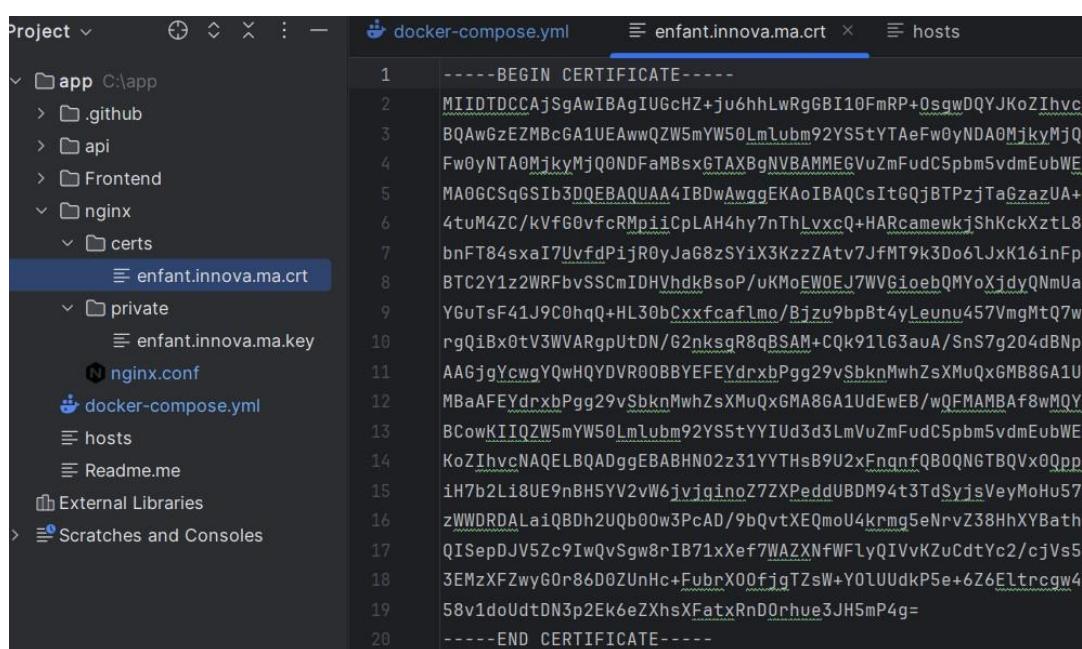
```
PS C:\Users\LENOVO> cd "C:\Program Files\OpenSSL-Win64"
PS C:\Program Files\OpenSSL-Win64> openssl req -x509 -nodes -days 365 -newkey
rsa:2048 -keyout enfant.innova.ma.key -out enfant.innova.ma.crt -subj
"/CN=enfant.innova.ma" -addext
"subjectAltName=DNS:enfant.innova.ma,DNS:www.enfant.innova.ma"
...+...+...+.....+....+.....+.+.+....+.+.+.....+.....+.....+....+++++++
+++++++=.....*..+.....+.....+.+.+.....+.+.+.....+.....+.....+
.....+.....*..+.....+
...+.....+.....+..+.....+.....+.....+.....+.....+.....+.....+.....+
++*.+...+.....+...+...+.....+.....+.....+.....+.....+.....+.....+.....+
+...+.....+.....+..+.....+.....+.....+.....+.....+.....+.....+.....+
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
-----+
PS C:\Program Files\OpenSSL-Win64>
```

Tu peux visualiser le certificat et sa clé dans puis les copier dans le code comme ci-dessous .



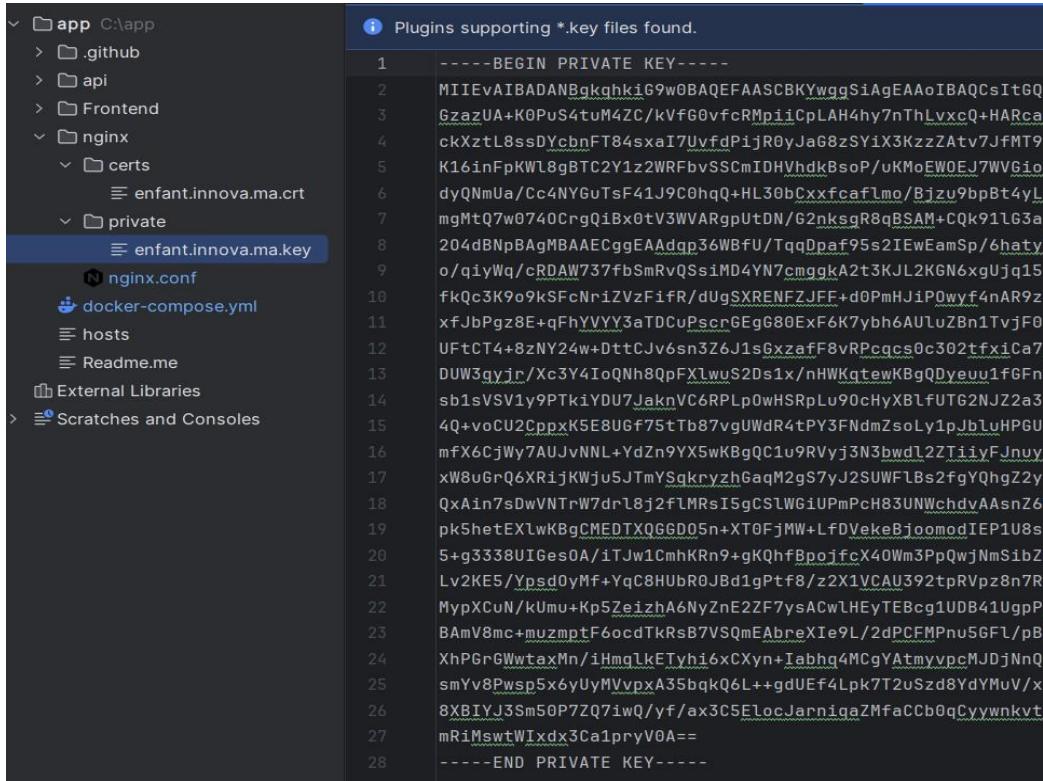
Nom	Modifié le	Type	Taille
bin	30/04/2024 00:27	Dossier de fichiers	
exp	30/04/2024 00:27	Dossier de fichiers	
include	30/04/2024 00:27	Dossier de fichiers	
lib	30/04/2024 00:27	Dossier de fichiers	
tests	30/04/2024 00:27	Dossier de fichiers	
acknowledgements.txt	11/04/2024 09:30	Document texte	1 Ko
authors.txt	11/04/2024 09:30	Document texte	2 Ko
c_rehash.pl	11/04/2024 09:30	Fichier PL	7 Ko
changes.txt	11/04/2024 09:30	Document texte	784 Ko
enfant.innova.ma.crt	30/04/2024 00:44	Certificat de sécurité	2 Ko
enfant.innova.ma.key	30/04/2024 00:44	Fichier KEY	2 Ko
libcrypto-3-x64.dll	11/04/2024 09:30	Extension de l'app...	6 219 Ko

Clé publique : code de la certificat :



```
1 -----BEGIN CERTIFICATE-----  
2 MIIDTDCCAJsgAwIBAgIUGchZ+ju6hhLwRgGBI10FmRP+0sqwDQYJKoZIhv  
3 BQAwGzEZMBcGA1UEAwwQZW5mYW50Lmlubm92YS5tYTAeFw0yNDA0MjkyMjQ  
4 Fw0yNTA0MjkyMjQ0NDFaMBsxGTAXBgNVBAMMEGVuZmFudC5pbm5vdmEubWE  
5 MA0GCSqSIB3DQEBAQUAA4IBDwAwggEKAoIBAQCsItGqjBTPzjTaGzazUA+  
6 4tuM4ZC/kVfG0vfcRMpiiCpLAH4hy7nThLvcxQ+HArcameWkjShKckXztL8  
7 bnFT84sxaI7UvfPijR0yJaG8zSYiX3KzzZAtv7JfMT9k3Do6lJxK16inFp  
8 BTC2Y1z2WRFbvSSCMIDHVhdkBsoP/uKM0EW0EJ7WVGioebQMYoXjdyQNmuA  
9 YGuTsF41J9C0hqQ+HL30bCxxfcflmo/Bjzu9bpBt4yLeunu457VmgtQ7w  
10 rgQiBx0tV3WVARgpUtDN/G2nksqR8qBSAM+CQk91lG3auA/SnStg204dBnp  
11 AAGjgYcwgYQwHQYDVRO0BBYEFYdrxbPgg29vSbknMwhZsXMuQxGMA8GA1UdEwEB/wQFMAMBaf8wMQY  
12 MBaAFEdrxbPgg29vSbknMwhZsXMuQxGMA8GA1UdEwEB/wQFMAMBaf8wMQY  
13 BCowKIIQZW5mYW50Lmlubm92YS5tYYIUD3d3LmVuZmFudC5pbm5vdmEubWE  
14 KoZIhvcNAQELBQADggEBABHN02z31YYTHsB9U2xFnanfQB0QNGTBQVx0Op  
15 iH7b2Li8UE9nBH5YV2vW6jvjgjnoZ7ZXPeddUBDM94t3TdSyjsVeyMoHu57  
16 zWWDRDALaiQBh2UQb00w3PcAD/9bQvtXEQmoU4krmg5eNrvZ38HhXYBath  
17 QISepDJV5Zc9IwQvSgw8rIB71xXef7WAZXNfWFlyQIVvKZuCdtYc2/cjVs5  
18 3EMzXFZwyGOr86D0ZUnHc+FubrX00fjgTzsW+Y0LUUdkP5e+6Z6Eltrcgw4  
19 58v1doUdtDN3p2Ek6eZXhsXfatxRnD0rhue3JH5mP4g=  
20 -----END CERTIFICATE-----
```

Clé privée : code privée de la certificat :



The screenshot shows a file explorer on the left and a terminal window on the right.

File Explorer:

- app C:\app
 - .github
 - api
 - Frontend
 - nginx
 - certs
 - enfant.innova.ma.crt
 - enfant.innova.ma.key
 - private
 - enfant.innova.ma.key
 - nginx.conf
 - docker-compose.yml
 - hosts
 - Readme.me
- External Libraries
- Scratches and Consoles

Terminal Window:

```
Plugins supporting *.key files found.

1 -----BEGIN PRIVATE KEY-----
2 MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCsItGQ
3 GzazUA+KOPuS4tuM4ZC/kVfG0vfcRMpniCpLAH4hy7nThLvcxQ+HARca
4 ckXztL8ssDYcbnFT84sxaI7UvfdPijR0yJaG8zSYiX3KzzZAtv7JfMT9
5 K16inFpKWL8gBTC2Y1z2WRFbvSScmIDHVhdkBsoP/uKM0EW0EJ7WVGio
6 dyQNmu/Cc4NYGuTsF41J9C0nqQ+HL30bCxXfcalfmo/Bjzu9bpBt4yL
7 mgMtQ7w0740CrgQiBx0tV3WVARgpUtDN/G2nksgR8qBSAM+CQk91LG3a
8 204dBnpBAgMBAECCgEAAdqp36WBfU/TqqDpaf95s2IEwEamSp/6haty
9 o/qiyWq/cRDAW737fbSmRvQSsiMD4YN7cmggkA2t3KJL2KGN6xgUjq15
10 fkQc3K9o9kSFcNrizVzF1fR/dUgSXRENfZJFF+d0PmHJiPOwyf4nAR9z
11 xfJbPgzbE+qfhYY3aTDCuPscrGEgG80ExF6K7ybh6AUluZBn1TvJF0
12 UFtCT4+8zNY24w+DttCJv6sn3Z6J1sGxzafF8vRPcqcs0c302tfxiCa7
13 DUW3gyj.../Xc3Y4IoNnh8QpFXlwus2Ds1x/nHWKgtewKBgQDyeuu1GFn
14 sb1sVSV1y9PTkiYDU7J...aknVC6RPl0wHSRpLu90cHyXB1fUTG2NJZ2a3
15 4Q+v0CU2CpxK5E8UGf75tTb87vgUWdR4tPY3FNd...ZsoLy1pJbluHPGU
16 mfx6CjWy7AUJvNNL+YdZn9YX5wBqgQC1u9RVyj3N3bwdl2ZTiiyJnuy
17 xWBUGrQ6XRijKWjusJTmYsqkr...zhGa...M2gS7yJ2SUWF...Bs2fgYQhgZ2y
18 QxAin7sDwVNTrW7dr18j2fLMRsI5gCSlWG...iUpmPch83UNWchdyAA...nZ6
19 pk5hetEXlwKBgCMEDTXQGGD05n+XT0FjMW+LfDVekeBjo...odIEP1U8s
20 5+g3338UI...es0A/iTJw1CmhKRn9+gKQhfBpojfcX40Wm3PpQwjNmSibZ
21 Lv2KE5/Ypsd0yMf+YqC8HUbR0JBd1gPtf8/z2X1VCAU392tpRVpz8n7R
22 MypXCuN/kUmu+Kp5ZeizhA6NyZnE2Zf7ysAcwlHEyTEBcg1UDB41UgpP
23 BAmV8mc+muzmptF6ocdT...RsB7VS...neAbreXi...e9L/2dPCFMPnu5GFL/pB
24 XhPGrGWtaxMn/iHmq...lkETyhi6xCXyn+Iab...hq4MCgYAtmyvpcMJDjNnQ
25 smYv8Pwsp5x6yUyMVvpxA35bqkQ6L++gdUEf4Lpk7T2uSz...dYMuV/x
26 8XB...YJ3Sm50P7ZQ7iwQ/yf/ax3C5ElocJarniqaZMfaCCb0dCyywnkv...
27 mR1Ms...wtW...txdx3Ca1pryVOA=-
28 -----END PRIVATE KEY-----
```

Nginx.config :

```

        return 301 https://$host$request_uri;
    }

# HTTPS Server block to handle secure requests
server {
    listen 443 ssl;
    server_name enfant.innova.ma www.enfant.innova.ma;

    ssl_certificate /etc/ssl/certs/enfant.innova.ma.crt;
    ssl_certificate_key /etc/ssl/private/enfant.innova.ma.key;

    location / {
        proxy_pass http://frontend:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

#This is optional to see the JSON fetching response ,
#you can adjust the paths according to your project
## Ces 2 champs sont pour le test Get JSON dans le navigateur

    location /api/prof {
        proxy_pass http://backend:8000/api/Profs;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api/etudiant {
        proxy_pass http://backend:8000/api/Etudiants;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

Image pgadmin (UI pour visualiser la base de données, Facultatif)

Voir commentaires du code pour l'explication

Image dnsmasq (Configuration du DNS pour la résolution du nom de domaine enfant.innova.ma)

dnsmasq est un serveur DNS (Domain Name System) léger et polyvalent, conçu pour fournir des services de résolution de noms de domaine dans un réseau local ou sur une machine individuelle. Voici une définition et ses utilisations :

Définition :

dnsmasq est un logiciel open source qui agit à la fois comme un serveur DNS et un serveur DHCP (Dynamic Host Configuration Protocol). Il est conçu pour être simple à

configurer et à utiliser, tout en offrant des fonctionnalités avancées pour la résolution de noms de domaine et la gestion des adresses IP dans un réseau.

Utilités de dnsmasq :

1. **Résolution de noms de domaine** : dnsmasq fonctionne comme un serveur DNS, permettant de résoudre les noms de domaine en adresses IP et vice versa. Cela facilite l'accès aux services et aux ressources du réseau local en utilisant des noms de domaine plutôt que des adresses IP.
2. **Cache DNS** : dnsmasq maintient un cache DNS local pour améliorer les performances de résolution des noms de domaine. Il stocke les réponses DNS précédemment récupérées, ce qui réduit le temps nécessaire pour résoudre les mêmes noms de domaine à l'avenir.
3. **Serveur DHCP** : En plus de la résolution DNS, dnsmasq peut également agir en tant que serveur DHCP pour attribuer des adresses IP et d'autres informations de configuration réseau aux appareils connectés à un réseau local. Cela simplifie la gestion des adresses IP dans un environnement réseau domestique ou de petite entreprise.
4. **Blocage des publicités** : dnsmasq peut être configuré pour bloquer les requêtes DNS vers des domaines associés à des publicités ou à du contenu indésirable. Cela permet de créer un réseau plus sécurisé et de fournir une meilleure expérience de navigation en ligne en bloquant les publicités indésirables.
5. **Filtrage de contenu** : En plus du blocage des publicités, dnsmasq peut être utilisé pour filtrer les requêtes DNS basées sur des listes de domaines spécifiques. Cela peut être utile pour restreindre l'accès à certains sites Web ou pour protéger les utilisateurs contre les contenus inappropriés.

Dans le contexte du fichier docker-compose.yml que on a fourni, le service dnsmasq est utilisé pour fournir des services DNS au sein du réseau Docker. Il est configuré pour écouter sur les ports TCP et UDP 53 et utilise un volume pour monter un fichier personnalisé `hosts` pour une configuration spécifique des hôtes. De plus, il bénéficie des priviléges réseau avec la capacité NET_ADMIN pour gérer les configurations réseau.

Pourquoi utiliser dnsmasq pour résoudre le nom de domaine si on ajuste le champs server_name dans nginx par :

server_name enfant.innova.ma www.enfant.innova.ma

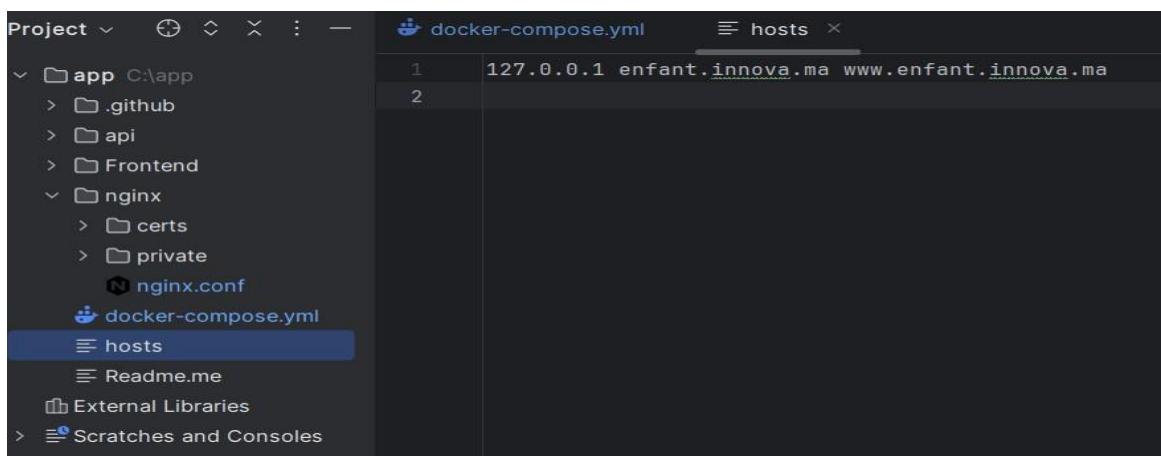
?????

En résumé, si vous utilisez un nom de domaine spécifique dans la configuration de votre serveur nginx et que vous n'avez pas dnsmasq pour la résolution DNS locale,

votre application dépendra des serveurs DNS externes pour la résolution des noms de domaine. Cela peut entraîner des problèmes potentiels de latence, de disponibilité et de sécurité, en fonction de la fiabilité et de la performance des serveurs DNS externes utilisés.

1. **dnsmasq:** : Nous nommons notre service dnsmasq.
 2. **image:** voir commentaires du code docker-compose.yml
 3. **cap_add:** : Nous ajoutons des capacités spéciales au conteneur Docker. Dans ce cas, nous ajoutons la capacité **NET_ADMIN**, ce qui donne au conteneur le droit de modifier la configuration réseau.
 4. **ports:** : Nous exposons les ports de notre service.
 - "**53:53/tcp**" : Nous faisons correspondre le port 53 de l'hôte avec le port 53 du conteneur pour le protocole TCP, permettant ainsi au service dnsmasq de recevoir des requêtes DNS via TCP.
 - "**53:53/udp**" : Nous faisons correspondre le port 53 de l'hôte avec le port 53 du conteneur pour le protocole UDP, permettant ainsi au service dnsmasq de recevoir des requêtes DNS via UDP
5. **volumes:** : Nous montons des volumes pour notre service.
 - **./hosts:/etc/hosts-custom** : Nous montons le fichier hosts personnalisé depuis le répertoire local ./hosts vers le répertoire /etc/hosts-custom dans le conteneur. Cela permet de fournir une configuration personnalisée pour le service dnsmasq en utilisant un fichier hosts personnalisé.

Fichier hosts :



The screenshot shows a code editor interface with a dark theme. On the left, there is a project tree labeled "Project". It contains a folder named "app" which has subfolders ".github", "api", "Frontend", "nginx" (which contains "certs" and "private" folders), and "docker-compose.yml". Below "nginx" is a file named "hosts" which is currently selected. Other files visible in the project tree include "nginx.conf", "Readme.me", "External Libraries", and "Scratches and Consoles". On the right, there is a code editor window titled "hosts" with the following content:

```
1 127.0.0.1 enfant.innova.ma www.enfant.innova.ma
```

- **Image mailhog** (pour la visualisation des emails et notifications Facultatif au cas d'utilisation de Gmail)

MailHog est un outil de test de messagerie électronique qui permet de capturer, de visualiser et de tester les e-mails envoyés par une application. Voici ses principales utilisations et fonctionnalités :

- 1. Capturer les e-mails : MailHog intercepte les e-mails sortants d'une application et les stocke temporairement dans une interface web conviviale, plutôt que de les envoyer réellement à leurs destinataires prévus.**
- 2. Visualiser les e-mails : Il offre une interface web où les e-mails capturés peuvent être visualisés, ce qui permet aux développeurs de vérifier le contenu, la mise en forme et les en-têtes des e-mails envoyés par leur application.**
- 3. Test des e-mails dans un environnement de développement : En utilisant MailHog, les développeurs peuvent tester l'envoi d'e-mails depuis leur application dans un environnement de développement, sans risque que les e-mails soient envoyés à de vraies adresses e-mail.**
- 4. Débogage et développement : MailHog est utile pour le débogage et le développement d'applications qui envoient des e-mails, comme les applications web, les applications mobiles, etc. Il permet de vérifier rapidement si les e-mails sont correctement générés et envoyés.**

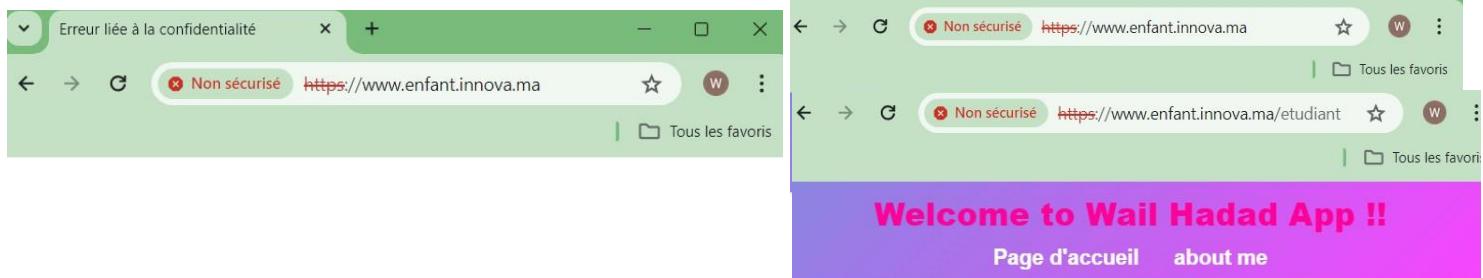
En ce qui concerne Laravel, voici comment vous pouvez utiliser MailHog avec Laravel :

- 1. Configuration dans Laravel : Dans le fichier de configuration `config/mail.php`, vous pouvez spécifier les paramètres de configuration de votre serveur de messagerie, notamment le protocole, l'hôte, le port, le nom d'utilisateur, le mot de passe, etc. Vous pouvez configurer Laravel pour utiliser MailHog comme serveur de messagerie local pour le développement.**
- 2. Utilisation avec Laravel Sail : Si vous utilisez Laravel Sail (un environnement de développement Docker pour Laravel), vous pouvez facilement intégrer MailHog en ajoutant simplement le service MailHog à votre configuration Docker Compose. Vous pouvez ensuite configurer Laravel pour utiliser le service MailHog pour l'envoi d'e-mails dans votre environnement de développement.**
- 3. Test d'e-mails : Une fois MailHog configuré avec Laravel, vous pouvez envoyer des e-mails depuis votre application et les vérifier dans l'interface web de MailHog pour vous assurer qu'ils sont correctement générés et envoyés.**

En résumé, MailHog est un outil précieux pour le développement et le test des fonctionnalités d'envoi d'e-mails dans les applications, offrant une manière pratique de capturer, visualiser et tester les e-mails sortants. Avec Laravel, il est souvent utilisé dans les environnements de développement pour simplifier le processus de test des fonctionnalités d'e-mail.

Afin de visualiser l'application dans le navigateur sous le nom de domaine enfant.innova.ma dans le port 80 ou bien localhost avec ssl et tous les configurations

On va au terminale dans le chemin root de notre application et on exécute les commandes : docker compose build et docker-compose up -d



Votre connexion n'est pas privée

Des individus malveillants tentent peut-être de subtiliser vos informations personnelles sur le site **www.enfant.innova.ma** (mots de passe, messages ou numéros de carte de crédit, par exemple). [En savoir plus](#)

NET::ERR_CERT_AUTHORITY_INVALID

Paramètres avancés

Revenir en lieu sûr

Gestion des étudiants

ID	Nom	Prénom	Note	Retirer
9	allam	mohammed	17.5	Edit / Delete
10	al haddad	abd arrazzaq	18	Edit / Delete
12	9adiri	imad	18	Edit / Delete

Nom
Entrer le nom de l'étudiant

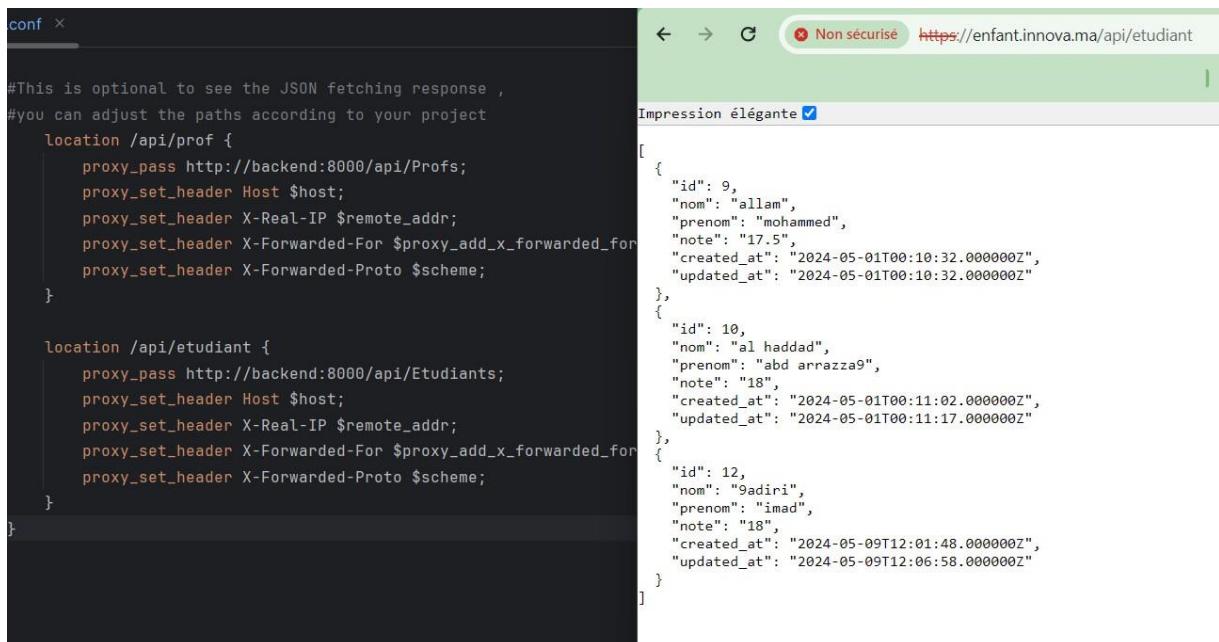
Prénom
Entrer le prénom étudiant

Note
Entrer la note de l'étudiant

Save

Connexion réussie avec la base de données

S'assurer des chemins configurer dans nginx.conf

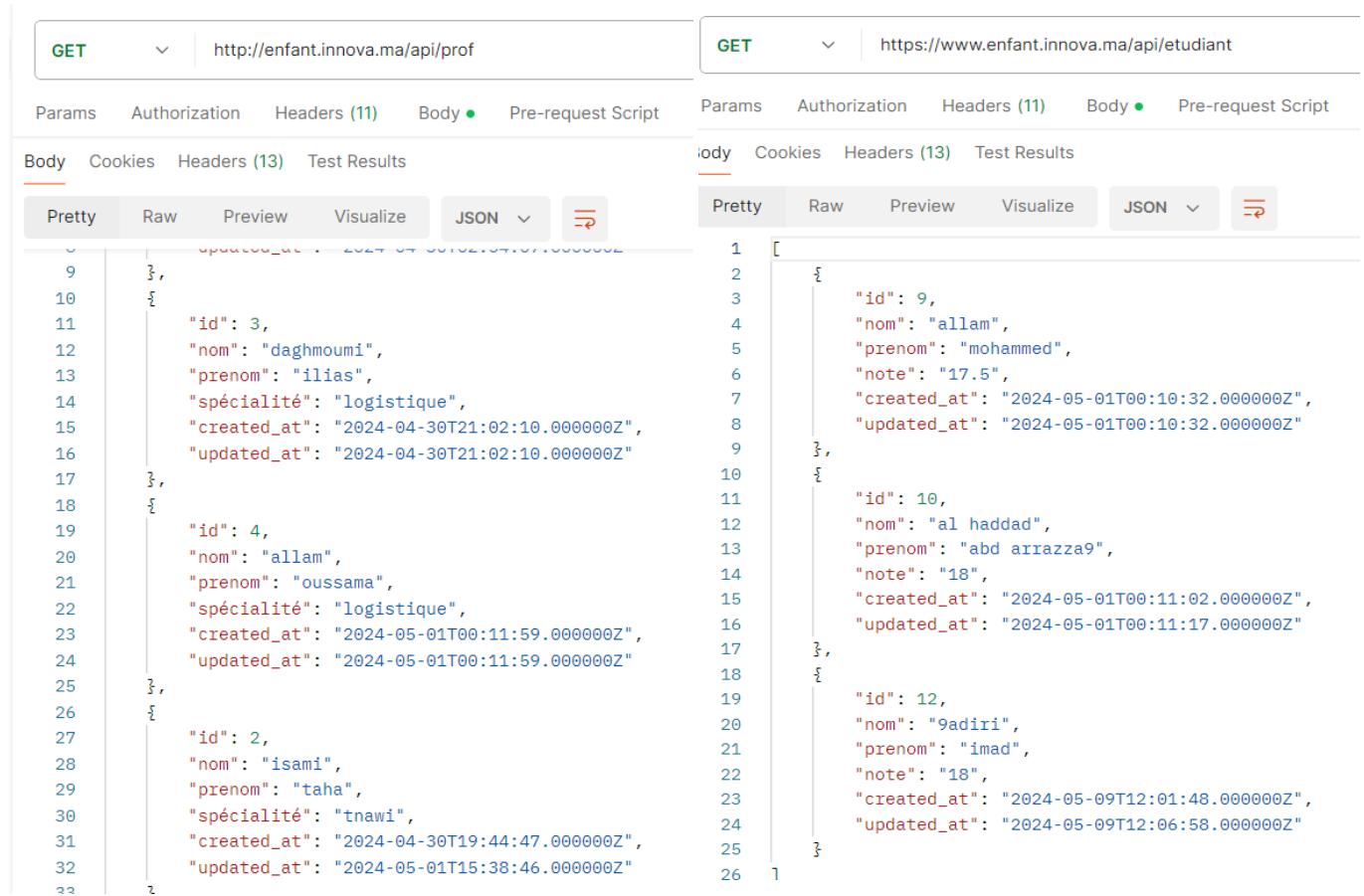


The terminal shows the nginx configuration file 'conf' with two location blocks for '/api/prof' and '/api/etudiant'. The browser window shows the JSON response for the '/api/etudiant' endpoint, listing three student records with fields: id, nom, prenom, note, created_at, and updated_at.

```
#This is optional to see the JSON fetching response ,  
#you can adjust the paths according to your project  
location /api/prof {  
    proxy_pass http://backend:8000/api/Profs;  
    proxy_set_header Host $host;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_set_header X-Forwarded-Proto $scheme;  
}  
  
location /api/etudiant {  
    proxy_pass http://backend:8000/api/Etudiants;  
    proxy_set_header Host $host;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_set_header X-Forwarded-Proto $scheme;  
}
```

```
[  
  {  
    "id": 9,  
    "nom": "allam",  
    "prenom": "mohammed",  
    "note": "17.5",  
    "created_at": "2024-05-01T00:10:32.000000Z",  
    "updated_at": "2024-05-01T00:10:32.000000Z"  
  },  
  {  
    "id": 10,  
    "nom": "al haddad",  
    "prenom": "abd arrazzaq",  
    "note": "18",  
    "created_at": "2024-05-01T00:11:02.000000Z",  
    "updated_at": "2024-05-01T00:11:17.000000Z"  
  },  
  {  
    "id": 12,  
    "nom": "9adiri",  
    "prenom": "imad",  
    "note": "18",  
    "created_at": "2024-05-09T12:01:48.000000Z",  
    "updated_at": "2024-05-09T12:06:58.000000Z"  
}  
]
```

Vérification avec postman :



Two GET requests are shown in Postman. The first request is for 'http://enfant.innova.ma/api/prof' and the second is for 'https://www.enfant.innova.ma/api/etudiant'. Both requests return JSON arrays representing student records with fields: id, nom, prenom, note, created_at, and updated_at.

Request 1: GET http://enfant.innova.ma/api/prof

Request 2: GET https://www.enfant.innova.ma/api/etudiant

JSON Response for Request 1:

```
1 | [  
2 |   {  
3 |     "id": 9,  
4 |     "nom": "allam",  
5 |     "prenom": "mohammed",  
6 |     "note": "17.5",  
7 |     "created_at": "2024-04-30T21:02:10.000000Z",  
8 |     "updated_at": "2024-04-30T21:02:10.000000Z"  
9 |   },  
10 |   {  
11 |     "id": 3,  
12 |     "nom": "daghmoumi",  
13 |     "prenom": "iliyas",  
14 |     "spécialité": "logistique",  
15 |     "created_at": "2024-04-30T21:02:10.000000Z",  
16 |     "updated_at": "2024-04-30T21:02:10.000000Z"  
17 |   },  
18 |   {  
19 |     "id": 4,  
20 |     "nom": "allam",  
21 |     "prenom": "oussama",  
22 |     "spécialité": "logistique",  
23 |     "created_at": "2024-05-01T00:11:59.000000Z",  
24 |     "updated_at": "2024-05-01T00:11:59.000000Z"  
25 |   },  
26 |   {  
27 |     "id": 2,  
28 |     "nom": "isami",  
29 |     "prenom": "taha",  
30 |     "spécialité": "tnawi",  
31 |     "created_at": "2024-04-30T19:44:47.000000Z",  
32 |     "updated_at": "2024-05-01T15:38:46.000000Z"  
33 |   }  
34 | ]
```

JSON Response for Request 2:

```
1 | [  
2 |   {  
3 |     "id": 9,  
4 |     "nom": "allam",  
5 |     "prenom": "mohammed",  
6 |     "note": "17.5",  
7 |     "created_at": "2024-05-01T00:10:32.000000Z",  
8 |     "updated_at": "2024-05-01T00:10:32.000000Z"  
9 |   },  
10 |   {  
11 |     "id": 10,  
12 |     "nom": "al haddad",  
13 |     "prenom": "abd arrazzaq",  
14 |     "note": "18",  
15 |     "created_at": "2024-05-01T00:11:02.000000Z",  
16 |     "updated_at": "2024-05-01T00:11:17.000000Z"  
17 |   },  
18 |   {  
19 |     "id": 12,  
20 |     "nom": "9adiri",  
21 |     "prenom": "imad",  
22 |     "note": "18",  
23 |     "created_at": "2024-05-09T12:01:48.000000Z",  
24 |     "updated_at": "2024-05-09T12:06:58.000000Z"  
25 |   }  
26 | ]
```

6) Push vers le dépôt GitHub après les mises à jour.

The screenshot shows a GitHub repository named "App_crud" which is public. The repository has 1 branch and 0 tags. The commit history is displayed, showing 11 commits from user "WAIL1221". The commits include changes to ".github/workflows", ".idea", "Frontend", "api", "nginx", "Readme.me", "docker-compose.yml", and "hosts". The last commit was made by "ba366e3" last week. The interface includes standard GitHub navigation and search tools.

Push vers dockerhub à l'aide des GitHub actions CI :

on remplis le champ actions dans secrets and variables dans GitHub par les nom des secrets mis dans build_and_push.yml un champs nommé DOCKER_HUB_USERNAME : et un autre nommé DOCKER_HUB_TOCKEN

The screenshot shows the "General" settings page for the "App_crud" repository. The left sidebar lists various settings categories: Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security, and Actions. The main area displays the General settings, including the repository name (set to "App_crud"), a checkbox for "Template repository" (unchecked), a checkbox for "Require contributors to sign off on web-based commits" (unchecked), and the "Default branch" set to "master". Below this, there is a "Social preview" section with instructions to upload an image for social media preview.

Access

Collaborators

Moderation options

Code and automation

- Branches
- Tags
- Rules
- Actions
- Webhooks
- Environments
- Codespaces
- Pages

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets Variables

Environment secrets

This environment has no secrets.

Manage environment secrets

Repository secrets

New repository secret

Name	Last updated
DOCKER_HUB_TOKEN	last week
DOCKER_HUB_USERNAME	last week

Actions Codespaces Dependabot

Username dockerhub : nom du compte ? wailhadad1221

Génération du Tocken docker hub :

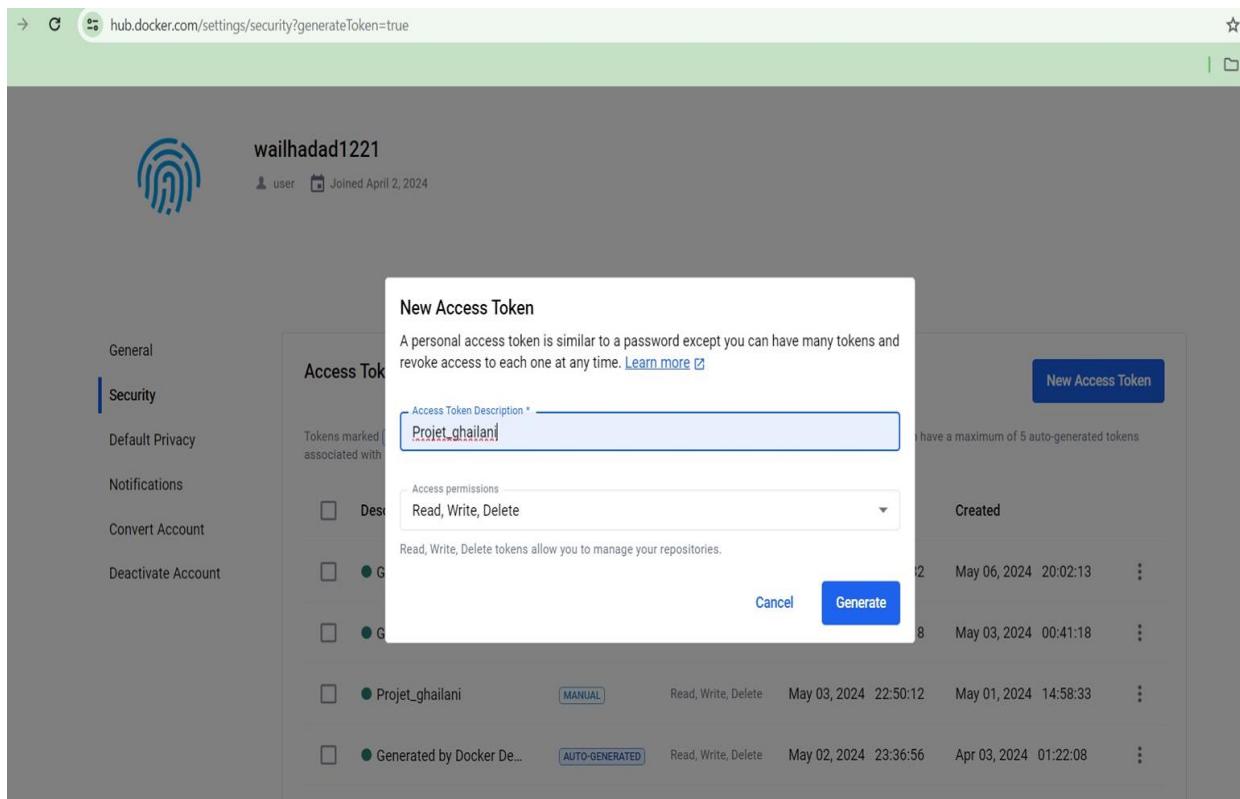
On va au profile ?my account?security?New access tocken

The screenshot shows the Docker Hub homepage with the user's repositories listed:

- wailhadad1221 / project_image (Security unknown, 0 stars, 17 forks, Public)
- wailhadad1221 / app_crud_image (Security unknown, 0 stars, 0 forks, Public)

A dropdown menu is open on the right side, showing the following options:

- What's New
- My Profile
- My Account** (highlighted in grey)
- Billing
- Sign out



Et Maintenant GitHub actions marche bien et le workflow est dirigé automatiquement pour construire une image dans mon répertoire dans dockerhub :

[Wailhadad1221/projet_image](#)

Les image sont construit sans problème et trouvé dans docker hub avec les tags affectés dans docker-compose.yml

The screenshot shows the Docker Hub repository page for 'wailhadad1221/project_image'. The repository has 7 tags: 'frontend-latest', 'backend-latest', 'mailhog', and 'dnsmasq'. It includes a 'Docker commands' section with a 'Public View' button and a command to push a new tag. The 'Automated Builds' section is available with Pro, Team, and Business subscriptions.

Tag	OS	Type	Pulled	Pushed
frontend-latest		Image	5 days ago	6 days ago
backend-latest		Image	5 days ago	6 days ago
mailhog		Image	5 days ago	7 days ago
dnsmasq		Image	5 days ago	7 days ago

Tags

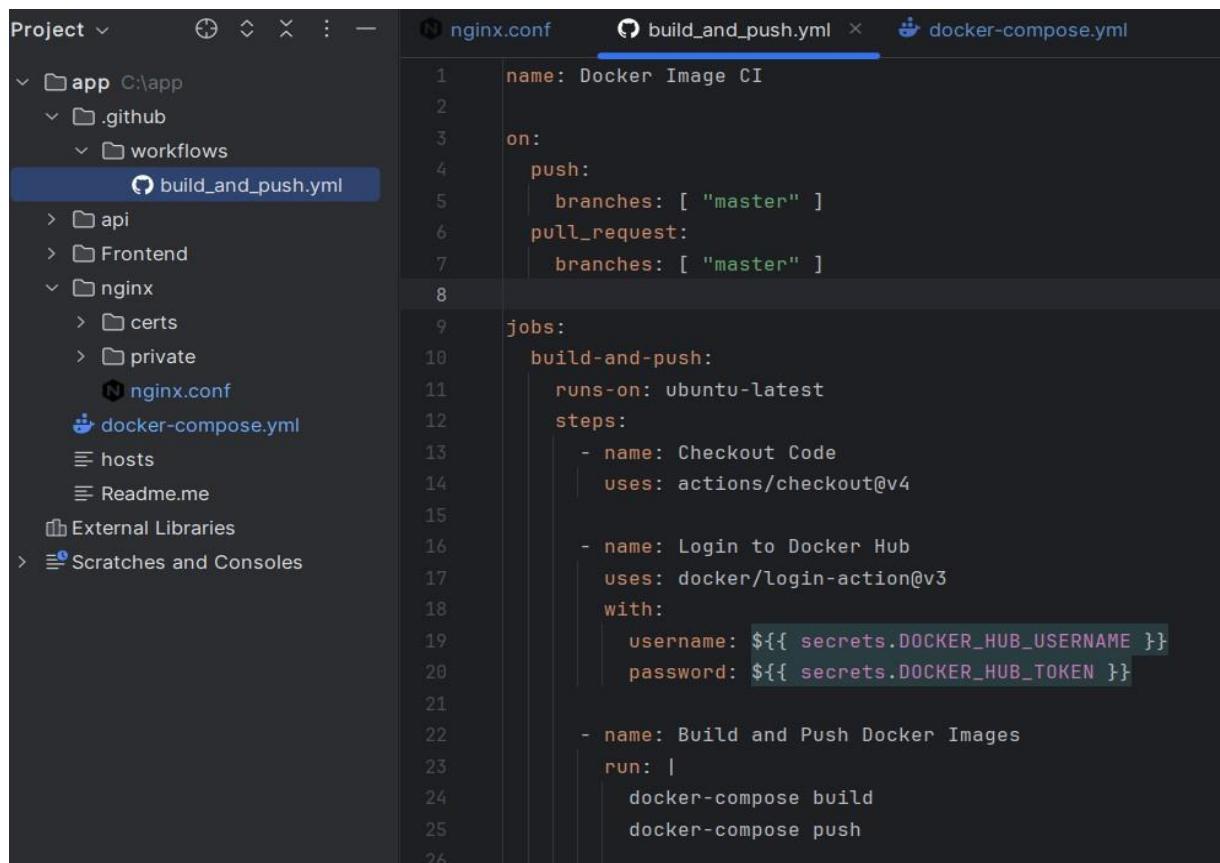
This repository contains 7 tag(s).

Tag	OS	Type	Pulled	Pushed
frontend-latest		Image	5 days ago	6 days ago
backend-latest		Image	5 days ago	6 days ago
mailhog		Image	5 days ago	7 days ago
dnsmasq		Image	5 days ago	7 days ago
postgres		Image	5 days ago	7 days ago

[See all](#)

Script Github actions :

On effectue un autre commit avec Github actions et les autres mises à jour.



The screenshot shows a GitHub repository interface. On the left, there's a sidebar with project navigation and a file tree. The file tree includes an 'app' folder containing '.github' (with 'workflows'), 'api', 'Frontend', 'nginx' (containing 'certs' and 'private'), 'nginx.conf', 'docker-compose.yml', 'hosts', and 'Readme.me'. Below these are 'External Libraries' and 'Scratches and Consoles'. The main area shows three tabs: 'nginx.conf', 'build_and_push.yml' (which is currently selected), and 'docker-compose.yml'. The 'build_and_push.yml' tab displays the following YAML code:

```
name: Docker Image CI
on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]
jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v4
      - name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_HUB_USERNAME }}
          password: ${{ secrets.DOCKER_HUB_TOKEN }}
      - name: Build and Push Docker Images
        run:
          docker-compose build
          docker-compose push
```

GitHub commits ran successfully

The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with 'Actions', 'All workflows', 'Docker Image CI', 'Management', 'Caches', 'Attestations', and 'Runners'. The main area is titled 'All workflows' and shows '3 workflow runs'. Each run is listed with a green checkmark, the name ('commit 11', 'commit 10', 'commit 8'), a brief description ('Docker Image CI #3: Commit ba366e3 pushed by WAIL1221'), the branch ('master'), and the time it was run ('last week', '3m 2s'). There are also 'Give feedback' and '... more' buttons.

7) Test des images délivrées à dockerhub dans une VM en utilisant SSH.

J'ai un Linux machine fixe , mais pour des raison de pratique d'SSH
J'ai installé une machine ubuntu virtuelle ou je vais installer docker et télécharger les images que j'ai dans docker hub.

1) Installer openssh-server dans la VM

Sudo apt update (bonne pratique)

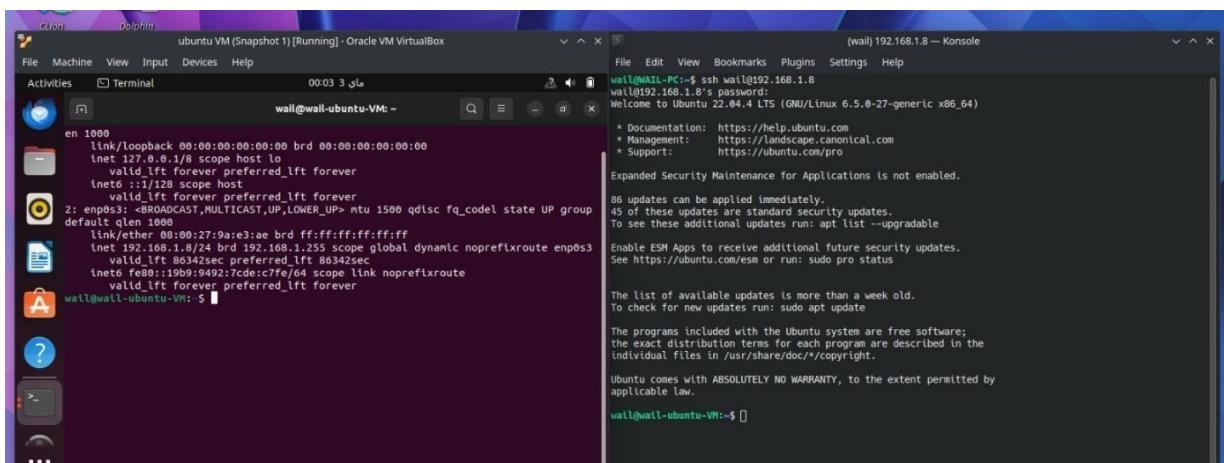
Sudo apt install openssh-server

On peut maintenant nous connecter à cette VM depuis une autre machine quelconque en savant le mot de passe de la VM et son ip (privé si local, publique si distant)

Pour savoir l'adresse ip de la VM en tape : ip a (shortcut of ip address)

Puis depuis une autre machine on accède à la VM par :

ssh nom_utilisateur_VM@ip_address

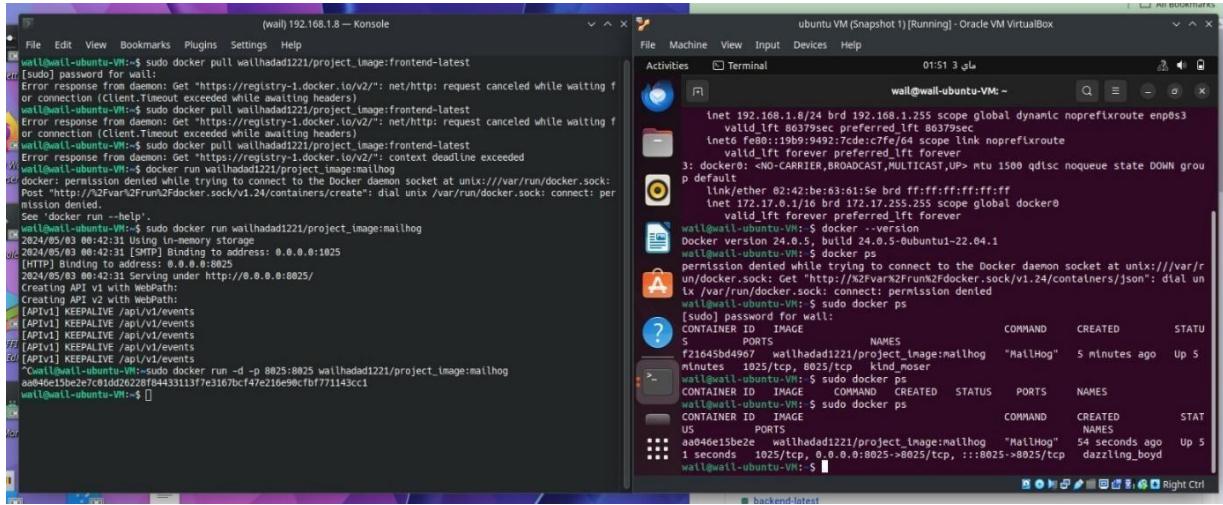


Quant à moi ssh wail@ip_address (elle n'est pas statique selon dhcp et le fournisseur d'accès internet)

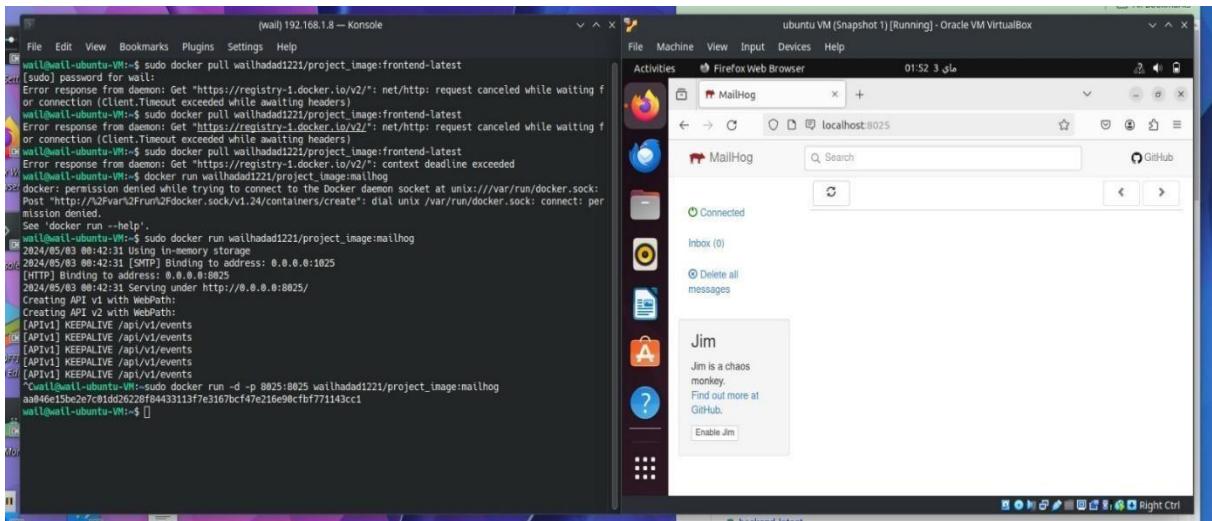
2) Installer docker engine dans la VM depuis l'autre machine

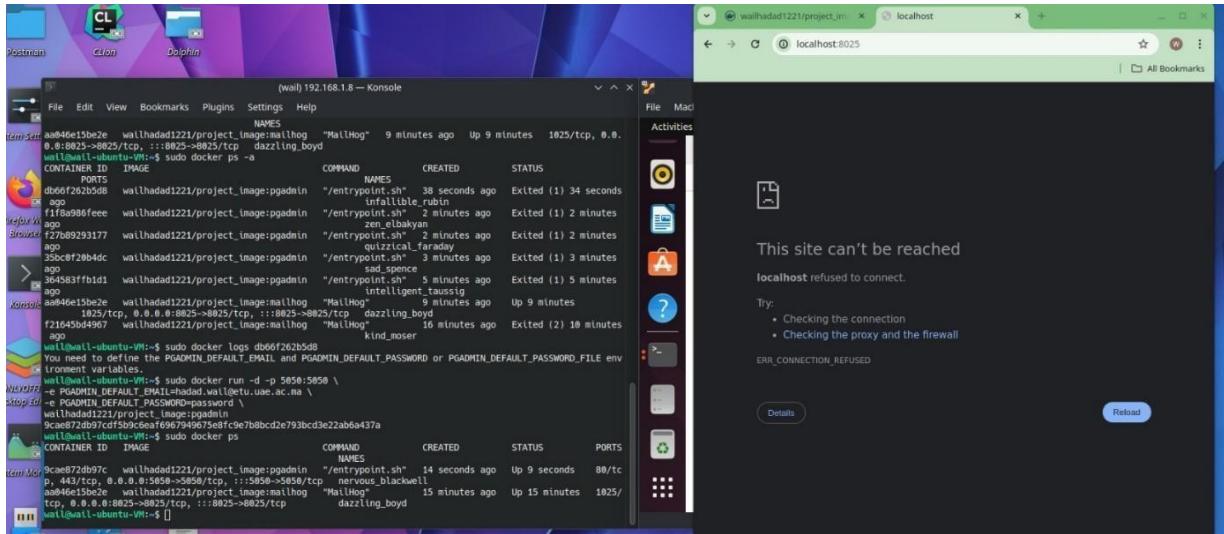
Sudo apt install docker.io

3) Puis vous pouvez tester par une image quelconque dans dockerhub ou pour notre cas les images dans notre répertoire wailhadad1221/projet_image



4) Visualiser le résultat dans le navigateur de la VM





On remarque que les conteneurs ne fonctionnent pas dans le navigateur de la machine statique tandis qu'ils fonctionnent dans la VM malgré que les commandes sont au terminal de la machine statique, car ssh te déconnecte de la machine actuelle et tout modification, installation, suppression est faite dans la VM

Pour revenir à ton terminal il suffit d'écrire « exit »

8) Push des fichiers docker dans notre dépôt de production.

Après être sûre de nos images et notre environnement on est prêt à contribuer dans le dépôt de production, on push nos fichiers docker dans la branche `develop` créée dans le dépôt GitHub de notre projet
On push un fichier `.dockerignore` contenant `docker-compose.yml`, les 2 Dockerfiles `backend` et `frontend`, fichier `hostset` le dossier `nginx` contenant SSL certificat et `nginx.conf`

Le responsable effectue l'emplacement juste de chaque fichier pour aboutir finalement à la structure finale du projet :

<code>.github/workflows</code>	commit 7	last week
<code>idea</code>	Added/Modified files and folders	last week
<code>Frontend</code>	commit 8	last week
<code>api</code>	commit 11	last week
<code>nginx</code>	commit 7	last week
<code>Readme.me</code>	Initial commit	last week
<code>docker-compose.yml</code>	commit 11	last week
<code>hosts</code>	Added/Modified files and folders	last week

Rapport Administration Système et Docker

1) Conteneurisation de l'application suivant les étapes précédentes

(Même étapes précédentes juste le contenu de l'application qui diffère)

2) Déploiement CI/CD de conteneurs dans un serveur

En résumé de CI/CD le serveur prend nos conteneurs pushés dans docker hub et les démarre. Tout ce processus est fait via GitHub actions. L'envoi des images vers le dépôt Docker hub spécifique lors de chaque commit en GitHub, ce qui permet d'avoir la dernière version mise à jour de l'application, puis le serveur prend les images et le code GitHub en utilisant les commandes git pull et docker pull

Pour avoir la dernière modification et l'application finale.

Le script de GitHub actions est le suivant :

Voici l'explication ligne par ligne sans les blocs de code :

- **`name: Docker Image CI`** : Définit le nom du workflow comme "Docker Image CI".

Déclencheurs :

- **`on: push: branches: ["master"]`** : Déclenche l'exécution du workflow lorsqu'un push est effectué sur la branche "master".
- **`on: pull_request: branches: ["master"]`** : Déclenche l'exécution du workflow lorsqu'une pull request est ouverte sur la branche "master".

Jobs

- **`jobs: build-and-push:`** : Définit un job nommé "build-and-push".
- **`runs-on: ubuntu-latest`** : Spécifie que le job s'exécute sur une machine virtuelle Ubuntu la plus récente.

Étape 1: Checkout Code

- `steps: - name: Checkout Code` : Nom de l'étape, "Checkout Code".
- `uses: actions/checkout@v4` : Utilise l'action `actions/checkout@v4` pour récupérer le code source du dépôt GitHub.

Étape 2: Login to Docker Hub

- `- name: Login to Docker Hub` : Nom de l'étape, "Login to Docker Hub".
- `uses: docker/login-action@v3` : Utilise l'action `docker/login-action@v3` pour se connecter à Docker Hub.
- `with: username: ${{ secrets.DOCKER_HUB_USERNAME }}` : Utilise le nom d'utilisateur Docker Hub stocké dans les secrets GitHub.
- `password: ${{ secrets.DOCKER_HUB_TOKEN }}` : Utilise le mot de passe Docker Hub stocké dans les secrets GitHub.

Étape 3: Build and Push Docker Images

- `- name: Build and Push Docker Images` : Nom de l'étape, "Build and Push Docker Images".
- `run: docker-compose build` : Exécute la commande pour construire les images Docker.
- `docker-compose push` : Exécute la commande pour pousser les images Docker sur Docker Hub.

Job: deploy

- `deploy:` : Définit un job nommé "deploy".
- `needs: build-and-push` : Indique que ce job dépend du job "build-and-push". Il ne s'exécutera que si "build-and-push" réussit.
- `runs-on: ubuntu-latest` : Spécifie que le job s'exécute sur une machine virtuelle Ubuntu la plus récente.

Étape 1: Setup SSH

- `- name: Setup SSH` : Nom de l'étape, "Setup SSH".

- **`uses: webfactory/ssh-agent@v0.5.3`** : Utilise l'action `webfactory/ssh-agent@v0.5.3` pour configurer l'agent SSH.
- **`with: ssh-private-key: \${{ secrets.EC2_SSH_KEY }}`** : Utiliser la clé SSH privée stockée dans les secrets GitHub.

Étape 2: Deploy to EC2

- **`name: Deploy to EC2`** : Nom de l'étape, "Deploy to EC2".
- **`run: ssh -o StrictHostKeyChecking=no ubuntu@13.51.206.218 << 'EOF'"** : Exécute une commande SSH pour se connecter à l'instance EC2 avec l'utilisateur "ubuntu" à l'adresse IP spécifiée.

Commandes à exécuter sur l'instance EC2

- **`cd /home/ubuntu/my-docker-compose-project || git clone https://github.com/WAIL1221/CD1_APP.git my-docker-compose-project && cd my-docker-compose-project`** : Change de répertoire pour le projet ou clone le dépôt si nécessaire.
- **`sudo docker-compose down`** : Arrête les conteneurs Docker en cours d'exécution.
- **`git pull`** : Récupère les dernières modifications du dépôt Git.
- **`find . -name "*.sh" -exec chmod +x {} \;`** : Assure que les scripts d'entrée sont exécutables.
- **`sudo docker-compose pull`** : Récupère les dernières images Docker depuis Docker Hub.
- **`sudo docker-compose up -d --remove-orphans`** : Démarrer les conteneurs Docker en arrière-plan en supprimant les conteneurs orphelins.
- **`sudo docker-compose exec backend composer install`** : Installe les dépendances PHP à l'intérieur du conteneur en cours d'exécution.
- **`sudo docker-compose exec backend php artisan config:cache`** : Met en cache la configuration Laravel à l'intérieur du conteneur.
- **`sudo docker-compose exec backend php artisan route:cache`** : Met en cache les routes Laravel à l'intérieur du conteneur.
- **`sudo docker system prune -af`** : Nettoie les images Docker inutilisées.
- **`sudo docker ps`** : Liste les conteneurs Docker en cours d'exécution pour vérifier que tout fonctionne correctement.

Afin de se connecter à la base de données , ces routes ne sont plus valables :

Api/.env :

Il faut remplacer localhost par l'adresse IP publique du serveur

Aussi dans Axios il remplacer localhost dans tous les component et axios.js aussi.

On démarre le serveur :

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (selected), AMIs, and Elastic Block Store. The main content area has a header 'Instances (1/1) Informations' with filters for Name, ID d'instance, État de l'instance, Type d'instance, Contrôle des statut, Statut d'alarme, Zone de disponibilité, and DNS IPv4 publique. A table lists one instance: 'server_1' (ID: i-049acba4abe25264c). It shows the state as 'En cours d...' (Running), type as 't3.micro', 2/2 verifications passed, alarm status as 'Afficher les alarmes', availability zone as 'eu-north-1b', and public DNS as 'ec2-13-53-168-139.eu-north-1.compute.amazonaws.com'. Below this, a detailed view for 'i-049acba4abe25264c (server_1)' is shown with tabs for Détails, Statuts et alarmes, Surveillance, Sécurité, Mise en réseau, Stockage, and Balises. The Détails tab displays the instance summary, including its ID, public and private IP addresses, and DNS name.

Il faut ensuite installer git et docker engine et docker compose dans le serveur afin de pouvoir utiliser les commandes git et docker compose.

La configuration se fait comme suite :

```
ubuntu@ip-172-31-32-19:~  
LENOVO@DESKTOP-EBDMUFT MINGW64 ~/Desktop  
$ cd C:/Users/LENOVO/Downloads  
  
LENOVO@DESKTOP-EBDMUFT MINGW64 ~/Downloads  
$ ls  
'Docker Desktop Installer.exe'* aws_ssh_key_ginf1.pem desktop.ini ideaIU-2024.1.3.exe* jdk-22_windows-x64_bin.exe*  
  
LENOVO@DESKTOP-EBDMUFT MINGW64 ~/Downloads  
$ ssh -i aws_ssh_key_ginf1.pem ubuntu@13.53.168.139  
The authenticity of host '13.53.168.139 (13.53.168.139)' can't be established.  
ED25519 key fingerprint is SHA256:c9ffPc5/uq2pCqqwaCSVXYTyZdvVy2cHIZrynP7OSPI.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '13.53.168.139' (ED25519) to the list of known hosts.  
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1008-aws x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/pro  
  
System information as of Sun Jun 16 11:41:28 UTC 2024  
  
System load: 0.08 Temperature: -273.1 C  
Usage of /: 23.2% of 6.71GB Processes: 107  
Memory usage: 23% Users logged in: 0  
Swap usage: 0% IPv4 address for ens5: 172.31.32.19  
  
Expanded Security Maintenance for Applications is not enabled.  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-32-19:~$ sudo apt-get update  
sudo apt-get install -y docker.io  
sudo systemctl start docker  
sudo systemctl enable docker  
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose  
# check the installation  
docker --version  
docker-compose --version
```

Puis on remplace GitHub actions SSH connexion par l'adresse IP public du serveur, et de même les components du frontend et .env du backend, et on push vers la branche master de GitHub afin de démarrer le workflow.

Project ▾

build_and_push

Terminal Local × + ▾

```

PS C:\CD1_APP> git add .
PS C:\CD1_APP> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
    modified:   .github/workflows/build_and_push.yml
    modified:   Frontend/.env
    modified:   Frontend/nginx.conf
    modified:   api/.env

PS C:\CD1_APP> git commit -m "132th commit"
[master 5014ec6] 132th commit
 4 files changed, 7 insertions(+), 7 deletions(-)
PS C:\CD1_APP> git push -u origin master
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 920 bytes | 115.00 KiB/s, done.
Total 10 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/WAIL1221/CD1_APP.git
 5b8b00d1..5014ec67 master -> master
branch 'master' set up to track 'origin/master'.
PS C:\CD1_APP>

```

Workflow du 132 commit est en cours d'exécution :

The screenshot shows the GitHub repository interface for the 'CD1_APP' repository. The top navigation bar includes links for Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main area displays the workflow runs for the '132th commit'. A sidebar on the left allows for creating new workflows.

Workflow Run	Event	Status	Branch	Actor
132th commit	now	In progress	master	Docker Image CI #129: Commit 5014ec6 pushed by WAIL1221
Delete api/vendor directory	5 days ago	4m 32s	master	Docker Image CI #128: Commit 5b8b00d pushed by WAIL1221
126th Commit From Wail	5 days ago	4m 12s	master	Docker Image CI #127: Commit 1bfd905 pushed by WAIL1221
125th Commit From Wail	5 days ago	4m 6s	master	Docker Image CI #126: Commit b7d60de pushed by WAIL1221

Et voici le résultat final en navigant dans [http:// 13.53.168.139](http://13.53.168.139)

The screenshot shows a web browser window with the URL 13.53.168.139. The page has a purple header with a logo, navigation links for Welcome, About Us, How to use, and Contact, and user icons for notifications and profile. The main content area features a large blue banner with the text "Apprenez vos enfants Sans Limites". Below it, a message encourages users to sign up for free or explore courses. To the right is a photograph of a teacher in a pink blazer interacting with a student in a classroom setting.

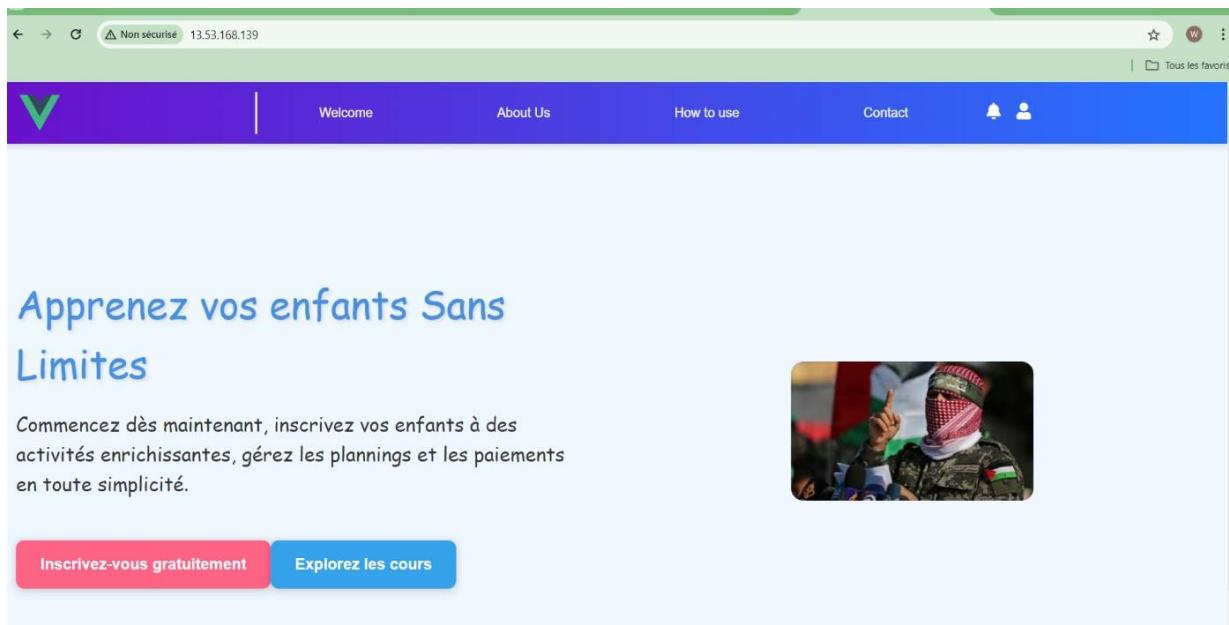
Two side-by-side screenshots show success messages: "13.53.168.139 indique Inscription réussie!" and "13.53.168.139 indique Connexion réussie!". Below these are the registration and login forms. The registration form (left) includes fields for Nom d'utilisateur (Wail_hadad), Email (wailw2445@gmail.com), and Mot de passe, with a "S'inscrire" button. The login form (right) includes fields for Email (wailw2445@gmail.com), Mot de passe (represented by dots), and a "Se connecter" button. A link "Mot de passe oublié ?" is also present.

Si on souhaite effectuer des modifications au niveau du Frontend ou du backend il seront modifiés automatiquement dans le serveur. Grâce au script GitHub actions.

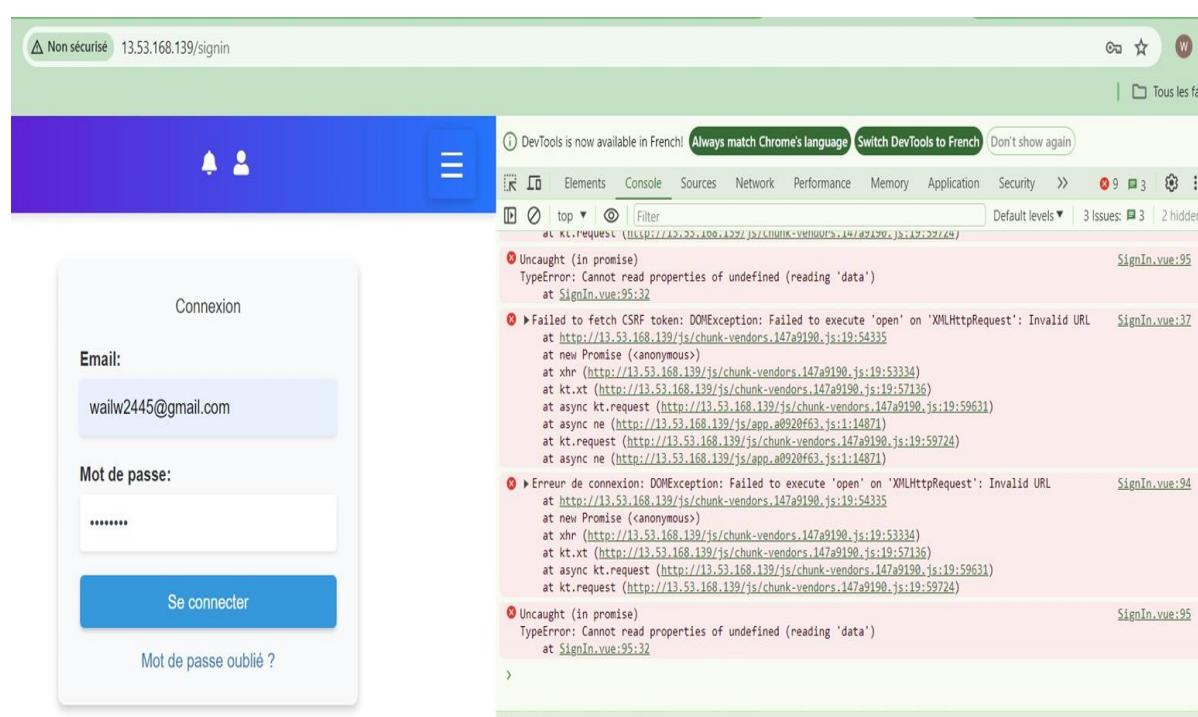
Si vous observer après la modification, on navigue dans la même adresse IP que précédemment :

On a changé la photo dans la page d'accueil et des modifications aussi au niveau des contrôleurs :

Et voici le résultat dans la même adresse IP :



The screenshot shows a web browser window with the URL 13.53.168.139. The page has a purple header with a logo and navigation links for Welcome, About Us, How to use, and Contact. Below the header, the main content area has been modified. The title 'Apprenez vos enfants Sans Limites' is now displayed in blue text. To the right of the title is a new image of a person wearing a balaclava and military-style clothing, pointing upwards. At the bottom left of the page, there are two buttons: 'Inscrivez-vous gratuitement' in red and 'Explorez les cours' in blue.



The screenshot shows a web browser window with the URL 13.53.168.139/signin. The page displays a login form with fields for 'Email' containing 'wailw2445@gmail.com' and 'Mot de passe:' with masked input. Below the form is a 'Se connecter' button. To the right, the Chrome DevTools are open, specifically the 'Console' tab. The console shows multiple error messages in red, indicating issues with promises, XMLHttpRequests, and CSRF tokens. One error message is highlighted in green, likely representing the successful modification made via GitHub Actions.

Il s'agit maintenant d'une erreur au niveau du sign in et sign up, ce qui est prévu lors des modifications faites.

Configuration statatique :

DNS

Postfixe

Nginx

dans une machine linux debian .

configuration DNS



Qu'est ce que DNS

Le DNS, ou système de noms de domaine (Domain Name System), est un système qui traduit les noms de domaine compréhensibles par les humains (comme www.exemple.com) en adresses IP (comme 192.0.2.1) que les ordinateurs utilisent pour s'identifier sur le réseau. Essentiellement, il fonctionne comme un annuaire téléphonique pour Internet, permettant aux utilisateurs d'accéder à des sites web en utilisant des noms faciles à retenir au lieu d'adresses IP numériques.

Etape 1: installation de bind9

Info: BIND9 peut fonctionner comme un serveur DNS autoritaire, qui fournit des réponses définitives pour les noms de domaine sous sa gestion, ou comme un serveur DNS récursif, qui recherche des informations sur Internet pour répondre aux requêtes des clients.

```
khalid@debian:~$ sudo apt install bind9 bind9utils bind9-doc -y  
[sudo] Mot de passe de khalid :
```

Etape 2: change directory to /etc/bind/:

```
khalid@debian:~$ cd /etc/bind/
```

```
khalid@debian:/etc/bind$ ls
db.0                      db.nurul.local          named.conf.options.orig
db.0.16.172                named.conf             named.conf.options.save
db.127                     named.conf.default-zones resolv.conf
db.255                     named.conf.local        rndc.conf
db.empty                   named.conf.local.orig   rndc.key
db.enfants.innova.ma      named.conf.options     zones
db.example.com            named.conf.options.m    zones.rfc1918
db.local                   named.conf.options.,mm
db.nural.local            named.conf.options.m.save
```

Etape 3: modification du fichier named.conf.options:

```
khalid@debian:/etc/bind$ sudo vim named.conf.options
```

```
// Define LAN network
acl MYLAN {
    192.168.1.0/24;
};

options {
    // Default directory
    directory "/var/cache/bind";
    // Allow queries from localhost and LAN network
    allow-query {
        localhost;
        MYLAN;
    };
    // Use Google DNS as a forwarder
    forwarders{
        8.8.8.8 ;
        8.8.4.4 ;
    };
    // Allow recursive queries
    recursion yes;
};
```

Etape 4: verification de la configuration :

```
khalid@debian:/etc/bind$ sudo named-checkconf named.conf.options
```

Si aucun message donc la configuration est correcte.

Etape 5 : modification sur named.conf.local:

```
khalid@debian:/etc/bind$ sudo vim named.conf.local
```

```
// Define the Forward zone
// My domain: enfants.innova.ma
// Forward file called forward.enfants.innova.ma
zone "enfants.innova.ma" IN {
    type master;
    // Path of Forward file
    file "/etc/bind/totatca/forward.enfants.innova.ma";
};

// Define the Reverse zone
// Reverse file called: reverse.enfants.innova.ma
zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/totatca/reverse.enfants.innova.ma";
};

~
```

```
khalid@debian:/etc/bind$ sudo named-checkconf named.conf.local
```

Etape 6: création du répertoire totatca:

```
khalid@debian:/etc/bind$ sudo mkdir totatca
```

Etape 7: modification sur le fichier forward.enfants.innova.ma :

```
khalid@debian:/etc/bind/totatca$ sudo vim forward.enfants.innova.ma
```

```
$TTL      604800
; SOA record with MNAME and RNAME updated
@       IN      SOA      enfants.innova.ma. root.enfants.innova.ma. (
                      3           ; Serial Note: increment after each
                      604800      ; Refresh
                      86400       ; Retry
                     2419200     ; Expire
                     604800 )     ; Negative Cache TTL
; Name server record
@       IN      NS       dns-1.enfants.innova.ma.
; A record for name server
dns-1   IN      A        192.168.1.8
www     IN      A        192.168.1.21
mail    IN      A        192.168.1.15

; Mail handler or MX record for the domain
ttc.local.  IN      MX    10    mail.enfants.innova.ma.

; A record for clients
client1  IN      A        192.168.1.111
client2  IN      A        192.168.1.112

~
```

verification

```
khalid@debian:/etc/bind/totatca$ sudo named-checkzone ttc.local forward.ttc.local
zone ttc.local/IN: loaded serial 3
OK
```

sudo vim reverse.enfants.innova.ma

```
$TTL    604800
; SOA record with MNAME and RNAME updated
@       IN      SOA    enfants.innova.ma. root.enfants.innova.ma. (
                        2           ; Serial Note: increment after each change
                        604800      ; Refresh
                        86400       ; Retry
                        2419200     ; Expire
                        604800 )     ; Negative Cache TTL
; Name server record
@       IN      NS     dns-1.enfants.innova.ma.
; A record for name server
dns-1   IN      A      192.168.1.8
www     IN      A      192.168.1.21
mail    IN      A      192.168.1.10
; PTR record for name server
8       IN      PTR    dns-1.enfants.innova.ma
21     IN      PTR    www.enfants.innova.ma
15     IN      PTR    mail.enfants.innova.ma
; PTR record for clients
111   IN      PTR    client1.enfants.innova.ma
112   IN      PTR    client2.enfants.innova.ma

~
```

redémarrage

```
khalid@debian:/etc/bind/totatca$ sudo systemctl restart bind9
```

```
khalid@debian:/etc/bind/totatca$ sudo named-checkzone enfants.innova.ma reverse.enfan
ts.innova.ma
zone enfants.innova.ma/IN: loaded serial 2
OK
```

```
khalid@debian:/etc/bind/totatca$ sudo systemctl status bind9
● named.service - BIND Domain Name Server
   Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
   Active: active (running) since Sun 2024-06-16 00:21:59 +01; 16s ago
     Invocation: d2891cb0a49241dca301db281cc485b0
       Docs: man:named(8)
 Main PID: 3426 (named)
   Status: "running"
    Tasks: 18 (limit: 9399)
  Memory: 74.0M (peak: 75.4M)
    CPU: 65ms
   CGroup: /system.slice/named.service
           └─3426 /usr/sbin/named -f -u bind
```

il est actif !

Test :

```
khalid@debian:~$ nslookup innova.ma
Server:          192.168.56.160
Address:         192.168.56.160#53

Non-authoritative answer:
Name: innova.ma
Address: 176.9.12.78
Name: innova.ma
Address: 64:ff9b::b009:c4e
```

POSTFIX

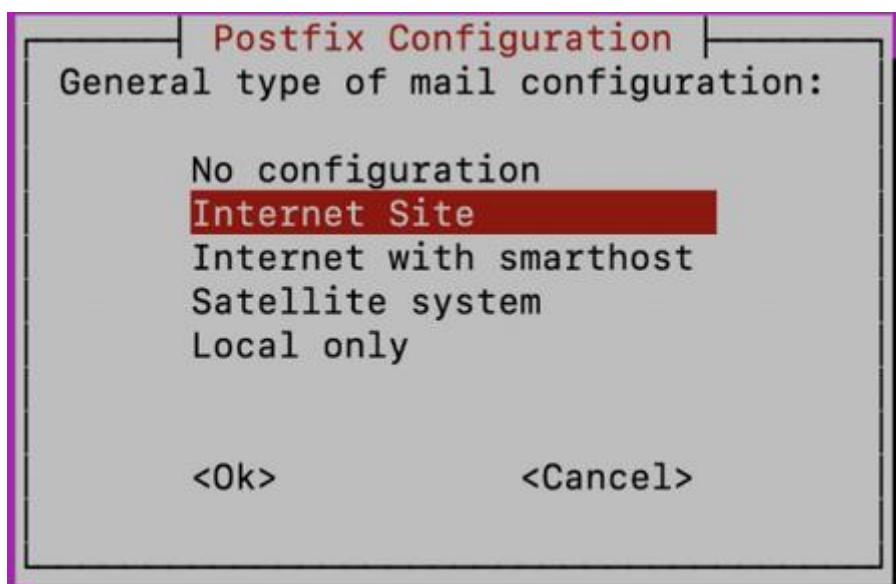


Postfix est un agent de transfert de courriel (MTA - Mail Transfer Agent) utilisé pour envoyer, recevoir et relayer des courriels. Il est couramment utilisé sur les serveurs de messagerie Unix/Linux. Postfix est conçu pour être sécurisé, rapide et facile à configurer et à administrer.

Installation de postfix:

```
khalid@debian:~$ sudo apt-get install postfix
```

Choisir l'option Internet site :



Qu'est-ce que SASL ??

SASL (Simple Authentication and Security Layer) est un framework d'authentification utilisé dans les protocoles de communication informatique. Il est un moyen universel qui assure l'authentification mutuelle entre le client et le serveur et la négociation des mécanismes de sécurité pour protéger leurs communications.

SASL est largement utilisé dans divers protocoles Internet, tels que SMTP (Simple Mail Transfer Protocol).

Qu'est ce que SMTP ?

SMTP est un protocole standard utilisé pour le transfert d'e-mails sur Internet. Il définit la manière dont les serveurs de messagerie envoient et reçoivent des e-mails, ainsi que les règles de communication entre ces serveurs.

Ouvrir le fichier sasl_passwd

```
khalid@debian:~$ ls /etc/postfix/sasl  
sasl_passwd  sasl_passwd.db
```

on ajoute ici le nom du protocol et le num du port 587

```
khalid@debian:~$ sudo vim /etc/postfix/sasl/sasl_  
passwd
```

```
relayhost = [smtp.gmail.com]:587  
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::  
1]/128  
mailbox_size_limit = 0  
recipient_delimiter = +  
inet_interfaces = all  
inet_protocols = all  
# Enable SASL authentication  
smtp_sasl_auth_enable = yes  
smtp_sasl_security_options = noanonymous  
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sa  
sl_passwd  
smtp_tls_security_level = encrypt  
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.cr  
t  
~  
~  
/relayh
```

ici . On écrit l'adresse gmail que j'ai créé et l'app password pour une couche de sécurité ajoutée

```
[smtp.gmail.com]:587 enfant.innova@gmail.com:hrvc e  
rjj jazq nttr  
]
```

on doit tester le service de messagerie qu'on a créé:

```
khalid@debian:~$ echo "This is a test email to local domain" | mail -s "Local Domain Test" enfant.innova@gmail.com  
khalid@debian:~$ telnet localhost 25  
bash: telnet : commande introuvable
```

swaks est l'outil pour tester le service dans ma machine debian !

```
khalid@debian:~$ sudo apt-get install swaks
```

```
swaks (2.12.1-1) ...  
khalid@debian:~$ swaks --to enfant.innova@gmail.com --from abdelhakjebari2003@gmail.com --server localhost
```

voila le service fonctionne !

The screenshot shows a Gmail inbox with one message. The message is titled "test" and was sent on Wednesday, May 22, 2024, at 23:59:28 +0100. The recipient is "enfant.innova@gmail.com" (to me). The message body contains the text "This is a test mailing". The inbox also shows other items like "Starred", "Snoozed", "Sent", and "Drafts".



NGINX

NGINX est un serveur web open-source performant, connu pour sa rapidité, sa scalabilité et son efficacité. Voici quelques-unes de ses fonctionnalités clés et ses utilisations courantes :

Fonctionnalités Clés de NGINX

1. Haute Performance : NGINX est conçu pour gérer un grand nombre de connexions simultanées, ce qui le rend idéal pour les sites web à fort trafic.
2. Proxy Inverse : NGINX peut agir comme un proxy inverse, distribuant les requêtes des clients à plusieurs serveurs backend, améliorant ainsi l'équilibrage de charge et la tolérance aux pannes.
3. Équilibrage de Charge : NGINX distribue le trafic réseau ou applicatif entre plusieurs serveurs pour éviter qu'un seul serveur ne soit surchargé.
4. Cache HTTP : NGINX peut mettre en cache les réponses des serveurs backend, réduisant ainsi la charge et améliorant les temps de réponse.
5. Termination SSL/TLS : NGINX gère le chiffrement et le déchiffrement SSL/TLS, déchargeant cette tâche gourmande en ressources des serveurs backend.
6. Contenu Statique et Dynamique : NGINX sert efficacement le contenu statique comme les fichiers HTML, CSS, JavaScript et les images, tout en agissant comme une passerelle pour le contenu dynamique généré par des applications telles que PHP, Python ou Node.js.
7. Sécurité : NGINX offre des fonctionnalités de sécurité, notamment la limitation du taux de requêtes, le blocage d'IP, etc.
8. Support WebSockets et HTTP/2 : NGINX prend en charge les technologies web modernes, y compris les WebSockets pour la communication en temps réel et HTTP/2 pour des performances améliorées.

Installation :

```
khalid@debian:~$ sudo apt install nginx
```

démarrage du Nginx:

```
khalid@debian:~$ sudo systemctl start nginx
```

modification sur /etc/nginx/sites-available/enfants.innova.ma

```
khalid@debian:~$ sudo nano /etc/nginx/sites-available/enfants.innova.ma
```

listen 80 : Le port 80 est le numéro de port attribué au protocole de communication Internet couramment utilisé, le protocole de transfert hypertexte (HTTP). Il s'agit du port réseau par défaut utilisé pour envoyer et recevoir des

```
server {  
    listen 80;  
    server_name enfants.innova.ma www.enfants.innova.ma;  
  
    root /var/www/laravel/public;  
    index index.php index.html index.htm;  
  
    location / {  
        try_files $uri $uri/ /index.php?$query_string;  
    }  
  
    location ~ \.php$ {  
        include snippets/fastcgi-php.conf;  
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock; # Adjust PHP version if necessary  
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
        include fastcgi_params;  
    }  
  
    location ~ /\.ht {  
        deny all;  
    }  
}
```

redemarrer nginx :

```
khalid@debian:~$ sudo systemctl reload nginx
```

test :

```
khalid@debian:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```