

The background of the ECCAM logo features a stylized, light gray gear or cogwheel. At the top of the gear is a circular element containing a crosshair and an arrow pointing towards the bottom right. The text 'ECCAM' is written in a bold, dark blue, sans-serif font, centered horizontally and partially overlaid by the gear graphic.

# **ECCAM**

**BRUSSELS ENGINEERING SCHOOL**

## **5eiad5L Distributed Systems Project**

**Losseau Baudouin**

**Mezghani Rayhan**

## **Contents**

1. Introduction .....	3
2. Matériel et méthode .....	3
3. Résultats .....	3
4. Discussion .....	5

## 1. Introduction

Ce rapport a pour objectif d'expliquer ce qui a été fait lors du projet ECAMazon 2023. Ce projet consiste à mettre en place des micro-services déployés sur Kubernetes. Le but de cette architecture sous forme de docker permet une flexibilité, scalabilité et une maintenance élevée. L'avantage de fonctionner sous forme de dockerisation permet de s'affranchir du versionnage des programmes propre à l'utilisateur. Chaque micro-service doit pouvoir fonctionner de façon indépendante et propose une documentation sur l'API mis à disposition afin de permettre aux autres groupes de communiquer avec celle-ci.

Nous devons gérer la partie "Dispatching".

Voici les consignes complètes : <https://quentin.lurkin.xyz/courses/scalable/project2023/>

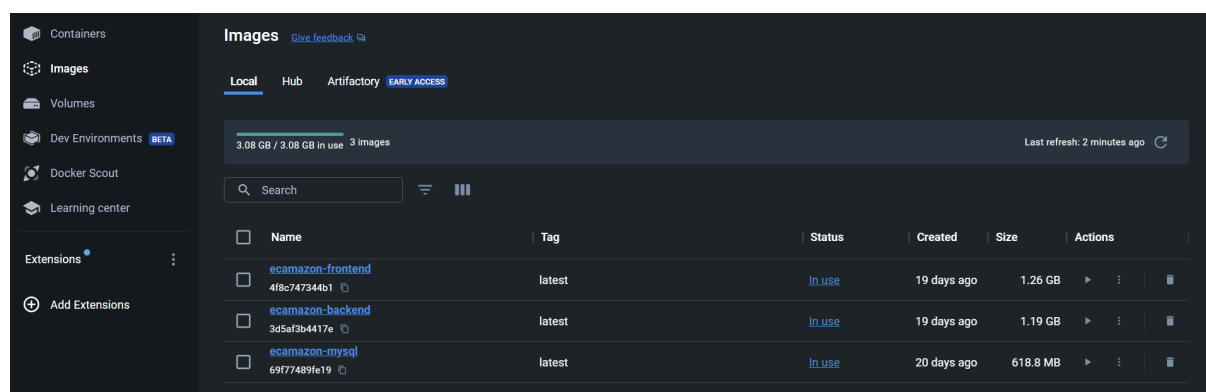
## 2. Matériel et méthode

Dans le but de déployer au final le projet sur la plateforme Teleport-Kubernetes hébergée par l'Ecam nous avons commencé le développement de notre projet en local pour être sûr d'avoir bien compris ce qui était demandé. Par la suite nous nous sommes renseignés sur le fonctionnement de Docker et avons migré notre application sur Docker Hub en local. Afin de pouvoir mettre à jour et plus facilement notre projet nous avons décidé de créer 3 images (et donc donc 3 docker compose). Nous avons donc une image pour le frontend, une pour le backend et finalement une pour la base de données MySQL server. MySQL server n'étant pas prévu pour le sharding nous envisageons de migrer vers MySQL cluster qui est prévu à cet effet. Un autre avantage notable et séparer en plusieurs parties notre projet fut de comprendre le fonctionnement de Docker et de mieux appréhender comment mettre en relation différentes images, ce qui nous semble bénéfique car le but final est de communiquer avec d'autres micro-services.

Le code final est disponible sur ce repository Github : <https://github.com/mezray/ECAMAZON>

## 3. Résultats

En local nous sommes capables de lancer notre micro-service sur DockerHub en local et les 3 images sont liées correctement. Il est possible d'interagir avec celui-ci. La base de données est persistante, si nous arrêtons Docker et le relançons les données précédemment introduites sont toujours présentes.



	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	ecamazon-frontend 4f8c747344b1	latest	In use	19 days ago	1.26 GB	▶ ⋮ ⌵
<input type="checkbox"/>	ecamazon-backend 3d5af2b4417e	latest	In use	19 days ago	1.19 GB	▶ ⋮ ⌵
<input type="checkbox"/>	ecamazon-mysql 69f77489fe19	latest	In use	20 days ago	618.8 MB	▶ ⋮ ⌵

Nous avons tenté de migrer vers Teleport - Kubernetes mais sans succès malheureusement.

L'objectif de notre micro-service "dispatching" est de recevoir les colis à livrer et de créer des listes pour les camions qui partent en livraisons. Il est possible de discuter avec notre micro-service via l'API que voici :

- **POST /add\_colis**

Cette route permet d'ajouter un nouveau colis à une livraison. Si une livraison a déjà 30 colis, un nouveau camion est sélectionné pour la prochaine livraison.

*Paramètres du corps de la requête :*

1. id : L'identifiant du colis.
2. adresse\_x : La coordonnée x de l'adresse de livraison du colis.
3. adresse\_y : La coordonnée y de l'adresse de livraison du colis.

#### • **POST /postPosColisFromDevice**

Cette route permet de mettre à jour l'état d'un colis.

*Paramètres du corps de la requête :*

1. colis\_id : L'identifiant du colis.
2. etat\_colis : Le nouvel état du colis.

#### • **POST /postPosCamionFromDevice**

Cette route permet de mettre à jour la position d'un camion.

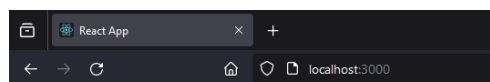
*Paramètres du corps de la requête :*

1. camion\_id : L'identifiant du camion.
2. camion\_pos\_x : La nouvelle coordonnée x de la position du camion.
3. camion\_pos\_y : La nouvelle coordonnée y de la position du camion.

#### • **GET /getLivraison**

Cette route retourne les informations sur les livraisons en cours. Elle retourne l'identifiant de la livraison, l'identifiant du colis, les coordonnées de l'adresse de livraison du colis et l'état du colis.

Un frontend basique a été développé afin de vérifier le bon fonctionnement.



### Ajouter un Colis

ID:   
Adresse X:   
Adresse Y:

### Modifier la Position du Colis

Colis ID:   
Nouvel État du Colis:

### Modifier la Position du Camion

Camion ID:   
Nouvelle Position X du Camion:   
Nouvelle Position Y du Camion:

### Données de Livraison

Livraison	ID Colis	ID Adresse	X	Adresse Y	État du Colis
1	1	1	1	1	
1	2	15	20	0	
1	3	5	2	0	
1	4	1010	541	0	

Les prochaines étapes sont la migration vers Teleport et l'automatisation des tests lors d'un push GitHub afin de mettre à jour automatiquement sur Teleport.

## 4. Discussion

Suite à nos efforts de développement, en local, nous avons réussi à déployer notre micro-services sur Docker Hub avec les trois images correctement liées. L'interaction avec le micro-services est fonctionnelle, et la persistance des données dans la base de données MySQL offre une continuité même après l'arrêt et le redémarrage de Docker.

Cependant, la migration vers Teleport-Kubernetes n'a pas été aussi fructueuse que prévu. Malgré cela, nous avons identifié cette étape comme une priorité future. Les prochaines étapes prévoient également l'automatisation des tests lors d'un push sur GitHub pour faciliter les mises à jour automatiques sur Teleport-Kubernetes.

Notre micro-services de "dispatching" remplit efficacement son objectif de recevoir les colis et de créer des listes pour les camions en partance pour les livraisons. L'API associée offre des points d'interaction bien définis, notamment pour l'ajout de colis, la mise à jour de l'état d'un colis, la mise à jour de la position d'un camion, et la récupération des informations sur les livraisons en cours.

Le développement d'un frontend basique a été ajouté pour permettre une vérification visuelle du bon fonctionnement du micro-services.

En conclusion, ce projet a été une opportunité d'explorer un domaine du développement jusqu'alors inconnu pour nous. Nous aurions aimé voir tous les groupes communiquer entre eux afin de voir les discussions sur un joli tableau blanc porté sur Teleport.