

# Documentation d'Architecture Logicielle

---

## Document d'architecture logicielle

### LOG430-01 École de technologie supérieure

Documentation d'Architecture Logicielle (SAD)

LOG430-01 | Hiver 2022

Coordonateurs: Mathieu Massicotte, Olivier Brassard, Philippe Doyon, P-A Heredero, Simon Boyer, Ulysse Corbeil, Charles Brouard, Samuel Krysztofiak, S-O Malouin, Sarah Kadhum

### Table des matières

- ▶ [Portée et Résumé](#)
- ▶ [Contexte Architectural](#)
- ▶ [Vue Architecture globale](#)
- ▶ [Vues et Mappage entre les vues](#)
- ▶ [Guide de déploiement](#)
- ▶ [Interfaces communes](#)
- ▶ [Documents de référence](#)
- ▶ [Glossaire et acronymes](#)

# Overview

---

## Feuille de route et vue d'ensemble de la documentation

Les sous-sections de cette page fournissent des informations qui aideront les lecteurs ou les utilisateurs du document d'architecture logicielle (SAD) à trouver rapidement les informations qui leur sont utiles.

## Portée et résumé du document

Ce document décrit l'architecture du système MontrealLiveTraffic™. Cette application web va permettre de calculer le temps de trajet grâce au capteur de la ville de Montréal ou a des services externes. Pour le moment, le projet est limité a une section de la rue Notre-Dame.

Le projet a une architecture en microservice. Elle est inspiré d'une ébauche fournie par Yvan Ross dans le cadre du cours LOG-430.

## Organisation du document d'architecture

Ce document est divisé en cinq sections principales :

1. [Feuille de route et vue d'ensemble de la documentation](#) : fournit des informations sur ce document et son public cible. Il fournit la feuille de route et un aperçu du document.
2. [Contexte d'architecture](#) : fournit des informations sur l'architecture du logiciel. Il décrit le contexte et la justification de l'architecture logicielle. Il explique les contraintes et les influences qui ont conduit à l'architecture actuelle et décrit les principales approches architecturales qui ont été utilisées dans l'architecture.
3. [Vue et mappages entre les vues](#) : fournit des informations sur les vues architecturales disponibles. Il décrit les vues architecturales disponibles et les mappages entre les vues, c'est à dire les relations existantes entre les différentes vues.
4. [Documents de référence](#): fournissent des informations et des références aux documents qui sont cités ailleurs dans ce SAD.
5. [Glossaire et acronymes](#): contient des définitions de termes techniques utilisés dans la documentation qui peuvent ne pas être familières à tous les lecteurs.

## Comment une vue est documentée

1. **Présentation principale** : Les éléments et les relations de cette vue sont présentés sous forme de diagramme ou de représentation non graphique.
2. **Catalogue d'éléments** : Cette section contient un dictionnaire des éléments de la présentation principale. Elle fournit des informations et propriété supplémentaire pour faciliter la compréhension.
3. **Diagramme de contexte** : Présentation sous forme de diagramme qui montre les limites du système.
4. **Guide de la modifiabilité** : Décrit les mécanismes de modifiabilité utilisés dans la partie du système montrée dans la vue. Cela comprend aussi la manière et le moment où ces mécanismes peuvent être exercés.
5. **Raisonnement** : Justification de toute décision de conception importante dont la portée est limitée à cette vue. Les alternatives importantes sont aussi documentées.
6. **Vue associée** : Lien vers la vue parente et les vues enfants (si elles existent).

# Context

---

## Contexte Architectural

### Contexte du problème

Les sous-sections suivantes expliquent les contraintes et les besoins qui ont influencé l'architecture du système.

### Aperçu du système

Le système **MontrealLiveTraffic™** a comme objectif l'amélioration du calcul d'un temps de trajet entre deux points dans la ville de Montréal. En effet, dans le but de se positionner comme une des villes les plus intelligentes du monde, la ville de Montréal a mis à la disposition du public des données de capteurs en temps réel qui pourraient améliorer la qualité des estimations du temps requis pour effectuer un certain trajet. Dans le cadre de ce projet, le système se limitera dans un premier temps aux capteurs de température et radars situés sur la rue Notre-Dame.

Dans le but de tester différents algorithmes de calcul, le système permet de comparer des trajets provenant de différentes sources de données, que ce soit à partir des données de capteurs ou de services externes tels que Google Maps. Par ailleurs, le système permet de simuler les données des capteurs de la ville de Montréal afin de pouvoir tester les implémentations même si les capteurs ne sont pas disponibles.

L'architecture du système est composée de plusieurs micro-services qui permettent notamment d'accomplir les tâches suivantes :

- ▶ Calcul du temps de trajet entre deux points à l'aide des données de capteurs
- ▶ Calcul du temps de trajet entre deux points à l'aide de services externes (Google Maps, Bing Maps, Mapbox)
- ▶ Simulation des données des capteurs de la ville de Montréal
- ▶ Service de comparaison des temps de trajet entre deux points
- ▶ Interfaces utilisateurs

### Contexte

Ce document a pour but de donner une vue d'ensemble des micro-services composant le système **MontrealLiveTraffic™**. L'objectif caché derrière le choix de décomposer les systèmes en micro-service est de développer un système modulable et fiable. En procédant de cette façon, nous pourrions réaliser plus

facilement les nombreux attributs de qualités requises par le client. De plus, comme nous sommes plusieurs équipes à travailler sur le projet et que nous avons des expériences différentes, le fait de créer plusieurs micro-services plutôt qu'une application monolithique offre à chaque équipe de choisir la pile de technologies la plus appropriée, tout en profitant d'interfaces web communes afin d'interconnecter les micro-services pour former le système complet.

Note référez vous à la section [Vue Globale du Système](#) pour plus de détails sur les micro-services.

## **Exigences principales**

Pour plus de détails sur les attributs de qualité et cas d'utilisation, voir la section [Exigences du projet](#).

## **Principaux cas d'utilisation**

- ▶ CU01. Comparer les temps de trajet
- ▶ CU02. Possibilité de mettre le chaos dans les services en
- ▶ CU04. Perturber le trafic à l'aide d'une interface usager pour simuler un accident ou une période de pointe sur les données de simulation
- ▶ CU05. Simulation de donnée GPS
- ▶ CU06. Simulation usager multiple
- ▶ CU07. Service externes
- ▶ Util: Traidement/filtrage des données de simulation de la ville de Montréal

## **Attribut de qualité**

### **Disponibilité**

- ▶ AQD1. Monitoring
- ▶ AQD2. Ping/écho
- ▶ AQD3. Heartbeat
- ▶ AQD4. Redondance passive
- ▶ ADQ5. Redondance active

### **Modifiabilité**

- ▶ AQM1. Modifier sélection et intersection
- ▶ AQM2. Interface avec autre équipe
- ▶ AQM3. Généralisation redondance passive
- ▶ AQM4. Généralisation redondance active

### **Testabilité**

- AQT1. Interface spécialisée
- AQT2. Abstraire, source de données.
- AQT3. Limiter opération non déterministe
- AQT4. Record/playback
- AQT5. Gestion centralisée des logs + id message

## **Convivialité**

- AQU1. UI
- AQU2. Aggregation

## **Performance**

- AQP1. Optimisation sur la latence
- AQP2. 1000 clients

## **Interopérabilité**

- AQI1. Utilisation des microservices des autres équipes
- AQI2. Découverte dynamique de service

## **Sécurité**

- AQS1. Authentification
- AQS2. Maintien audit trail

## **Contexte de la solution**

Les sous-parties de cette section expliquent les raisons qui ont conduit à la conception de l'architecture telle qu'elle est et fournissent une justification de l'adéquation de l'architecture aux objectifs de qualité qui lui sont fixés.

## **Approche architecturale**

L'application sera divisée en différents services sur la base des fonctionnalités et des responsabilités. Une telle séparation réduit la dépendance entre les modules et nous permet de développer indépendamment. En outre, l'identification au préalable des interfaces et contrats de communication permet le développement des différents services par des équipes différentes. Les services du système sont documentés en détail à la section [Vue Globale du Système](#).

Par ailleurs, cette séparation des responsabilités permet de prioriser le développement des fonctionnalités les plus essentielles et rend chaque module plus générique. Cela facilite la mise en place de tests unitaires et de

tests d'intégration et nous permet de mieux répondre aux attributs de qualité comme la redondance active, la redondance passive et la disponibilité.

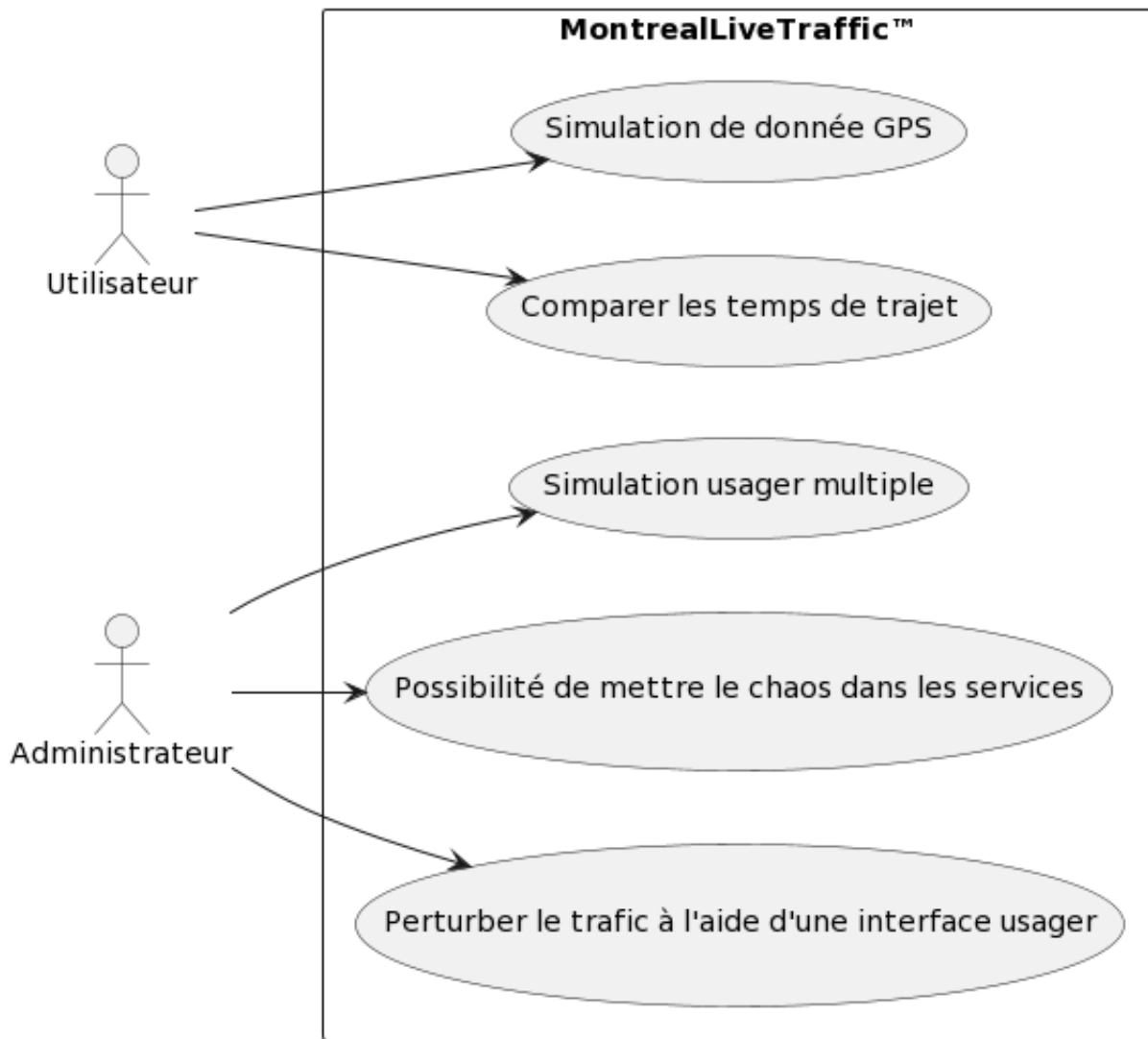
Nous utilisons des conteneurs Docker afin de faciliter le déploiement, la surveillance (monitoring) et la testabilité du système. Chaque service est donc isolé dans son propre conteneur, ce qui nous offre une grande flexibilité pour le développement et la mise en production. C'est aussi pour cette raison que chaque équipe est libre de choisir la stack de technologies la plus adaptée à leurs besoins.

Les déploiement, eux, sont fait à l'aide de docker-compose et sont contrôlés par l'outil d'administration Portainer. L'utilisation de docker-compose permet une grande facilité d'utilisation et permet de se concentrer sur le développement applicatif puisqu'il fonctionne de la même façon sur un ordinateur personnel que sur un serveur. Cela améliore aussi la testabilité de tous les systèmes. Portainer a été choisi comme outil d'administration puisqu'il permet de contrôler dans une interface web intuitive tous les déploiement avec assez de détails.

## **Résultat des analyses**

Il n'y a pas eu d'analyse, le client nous a imposé l'utilisation de l'architecture par micro-services.

## **Mise en correspondance des exigences et de l'architecture**



Ce document se concentre sur les cas d'utilisation suivants. Sous chaque cas d'utilisation, vous trouverez les micro-service qui comblent les besoins associés à ceux-ci.

- ▶ CU01. Comparer les temps de trajet
  - ▶ [RouteComparatorWebApp](#) : Application web qui permet de visualiser dans un graphique les données des différents services.
  - ▶ [RouteComparatorService](#) : Fournit les données à l'application web.
  - ▶ [RouteCalculatorService](#) : Calcule le temps de trajet entre deux point sur la rue Notre-Dame à un moment donné.
  - ▶ [ExternalServices](#) : Calcule le temps de trajet entre deux point sur la rue Notre-Dame à un moment donné.
  - ▶ [ServiceDiscovery](#) : Permet l'interconnection entre les services.
  - ▶ [SimulationDataService](#) : Fournit les données au [RouteCalculatorService](#).
- ▶ CU02. Possibilité de mettre le chaos dans les services
  - ▶ [ChaosMonkey](#) : Permet de tuer ou de changer la latence des micro-services du système.



- [ServiceDiscovery](#) : Permet l'interconnection entre les services.
- CU04. Perturber le trafic à l'aide d'une interface usager
  - PertubateurTrafficWebApp : Permet de simuler un accident, une fermeture de voie...
- CU05. Simulation de donnée GPS
  - GpsSimulatorWebApp : Simule un GPS en déplacement, en affichant en "temps réel" le temps de trajet restant.
  - RouteCalculatorService : Calcule le temps de trajet entre deux point sur la rue Notre-Dame à un moment donné.
  - [SimulationDataService](#) : Fournit les données au RouteCalculatorService.
  - [ServiceDiscovery](#) : Permet l'interconnection entre les services.
- CU06. Simulation usager multiple
  - Tout les services sont mis en oeuvre afin de parvenir à cette tâche.
- CU07. Services externes
  - GoogleMapService: Retourne les données de trajet fournies par Google Map
  - BingMapService: Retourne les données de trajet fournies par Bing Map
  - [MapBoxService](#): Retourne les données de trajet fournies par MapBox

# Exigences du projet

## Cas d'utilisation

### Acteurs

ChargéDeLaboratoire

#### CU01. Veux comparer les temps de trajet:

1. Le CL sélectionne une intersection de départ et une intersection d'arrivée, ainsi que le taux de rafraichissement de la prise de mesure.
2. Le CL sélectionne les sources de données qu'il veut utiliser. Service externe ou de simulation.
3. Le système affiche un graphique du temps de déplacement et met celui-ci à jour selon le taux de rafraichissement.
4. Le CL peut récupérer le fichier de données et générer ses propres graphiques à l'aide d'Excel.

#### Cas alternatifs



- ▶ 2.a Service externe: Utiliser tout les service externe disponible pour faire le comparatif.
- ▶ 2.b1 Simulation: Vous comparer au moins 2 implémentations différentes du microservice TimeTravelCalculator.
- ▶ 2.b2 La comparaison devrait pouvoir s'exécuté sur **toute** les données de simulation.

#### CU02. Veut pouvoir mettre le chaos dans les services en

- ▶ **Option 1: Manuel**
  - 1.1. Le CL consulte la liste des microservices avec leur latence moyenne.
  - 1.2. Le CL change la latence d'un ou plusieurs microservices.
- ▶ **Option 2: Automatique**
  - 2.1. Le CL sélectionne le mode automatique tout en spécifiant la fréquence de la perturbation en seconde.
  - 2.2. Le système détruit<sup>1</sup> un microservice de façon aléatoire a toute les x secondes.
- ▶ Le système conserve un log des différents changements apportés que nous pourrons utiliser pour vérifier les données accumulées.

## Cas alternatifs



- ▶ 1.2.a Le CL détruit un ou plusieurs microservices.
- ▶ 1.2.b Le CL détruit tous les microservices d'une équipe.

### CU04. Perturbateur de trafic

Veux pouvoir perturber le trafic à l'aide d'une interface usager pour simuler un accident ou une période de pointe sur les données de simulation.

1. Le CL indique la date et heure de début de la perturbation.
2. Le CL indique le type<sup>1</sup> et la durée de la perturbation ainsi que la ou les intersections perturbées.
3. Le système perturbe les données de simulation.
4. Le système affiche le graphique du temps de trajet en temps réel avec et sans perturbation.

Note 1: Type de perturbation

- ▶ fermeture d'une voie de circulation
- ▶ en diminution ou augmentation la vitesse de la circulation

### CU05. Simulation donnée GPS

Veux utiliser une ou plusieurs applications pour envoyer les positions GPS et faire le calcul du temps de trajet en temps réel;

1. Le CL sélectionne l'intersection de départ et de destination, il indique la vitesse moyenne qu'il veut simulée ainsi que la fréquence à laquelle il veut une mise à jour des données graphiques.
2. Le CL sélectionne le ou les services externes qu'il désire utiliser.
3. Le système lui indique le temps de trajet estimé s'il n'y a aucune perturbation de trafic.
4. Le CL démarre la simulation.
5. Le système affiche un graphique représentant le temps de trajet de chaque service externe en temps réel.

### CU06. Simulation usager multiple

Simulation du CU05 avec 100 véhicules simultanés ayant tous des vitesses de déplacement différentes (répartie aléatoirement).

Désactiver l'affichage graphique en temps réel et utiliser l'archivage des données pour permettre la réalisation de différents graphiques sur Excel.

Refaire la même simulation en utilisant le CU04 pour perturber le trafic.

# Attribut de qualité

## Disponibilité

### AQD1. Monitoring

Vous devez implémenter un mécanisme de monitoring pour tous les microservices.

### AQD2. Ping/écho,

chaque microservice doit supporter le mode écho (écho).

### AQD3. Heartbeat

chaque microservice doit supporter le mode heartbeat.

### AQD4. Redondance passive

Vous devez implémenter une redondance passive sur le microservice TravelTimeCalculator.

1. Démarrer 1 instance de TravelTimeCalculator qui sera utilisée comme point de référence.
2. Démarrer 1 instance active et une instance passive de TravelTimeCalculator.
3. Afficher le graphique des temps de trajet de ces deux microservices.
4. Utiliser CU02 pour détruire l'instance active de TravelTimeCalculator.
5. Démontrer que la simulation est maintenant réalisée par le microservice passif qui est devenu actif.
6. Démontrer que votre système redémarre une nouvelle instance du serveur passif.



**Refaire** l'étape #4 lorsque le serveur passif est opérationnel (actif).

### ADQ5. Redondance active

Vous devez implémenter une redondance active sur le microservice TimeTravelCalulator .

1. Démarrer une instance de TimeTravelCalulator en configuration de redondance active.
2. Utiliser CU02 pour détruire une des instances du mécanisme de redondance active.

## Modifiabilité:

### AQM1. Modifier sélection et intersection.

Vous devez être en mesure de modifier les intersections utilisées (ajoutez ou retrait) seulement en modifiant un fichier de configuration ou à l'aide de l'interface usager. L'application des changements à votre solution ne devrait pas prendre plus de 15 secondes.

## **AQM2. Interface avec autre équipe**

Vous devez être en mesure d'adapter votre solution pour utiliser les microservices d'une autre équipe en moins d'une heure.

## **AQM3. Généralisation redondance passive**

Démontrer que vous êtes en mesure de généraliser la redondance passive pour n'importe quel composant en moins d'une heure.

1. Quand vous êtes prêt, montrer la liste des microservices que vous avez développés au chargé de laboratoire.
2. Démontrer au chargé de laboratoire que vous avez implémenté des tactiques architecturales vous permettant de facilement généraliser la redondance passive.
3. Le chargé de laboratoire choisi un microservice.
4. Vous avez une heure pour lui montrer que vous avez implémenté la redondance passive pour ce microservice.

## **AQM4. Généralisation redondance active**

Démontrer que vous êtes en mesure de généraliser la redondance active pour n'importe quel composant en moins de 3 heures.

1. Quand vous êtes prêt, montrer la liste des microservices que vous avez développés au chargé de laboratoire.
2. Démontrer au chargé de laboratoire que vous avez implémenté des tactiques architecturales vous permettant de facilement généraliser la redondance active.
3. Le chargé de laboratoire choisi un microservice.
4. Vous avez trois heures pour lui montrer que vous avez implémenté la redondance active pour ce microservice.

## **Testabilité:**

### **AQT1. Interface spécialisée**

Démontrer que vous utilisez la tactique "Interface spécialisée" pour permettre de modifier dynamiquement la latence des microservices de votre architecture.

### **AQT2. Abstraire, source de données.**

Démontrer que vous utilisez la tactique "Abstraire la source de données" pour spécifier les sources de donnée.

### **AQT3. "Limiter opération non déterministe"**

Démontrer que vous utilisez la tactique "Limiter opération non déterministe" pour faciliter l'exécution de vos

tests. Surtout avec les données de simulation

#### **AQT4. Record/playback**

Démontrer que vous utilisez la tactique « Record/playback » pour pouvoir tester à nouveau votre système avec des données déjà reçues.

Démontrer que le playback fonctionne.

#### **AQT5. Gestion centralisée des logs + id message**

Démontrer que vous utilisez la tactique "Interface spécialisée" pour suivre à la trace tous les messages entrant dans les microservices par l'association d'un numéro unique à chaque message entrant. Ceci devrait permettre de faire la trace de tous les microservices actifs ayant été utilisés pour traiter un message.

### **Convivialité:**

#### **AQU1. UI**

Utiliser des interfaces usager au lieu de fichier de configuration pour faciliter la configuration de vos microservices.

#### **AQU2. Aggregation**

Utilisez la tactique "Aggregation" pour applique des commandes ou configuration à de multiples microservices ou intersection simultanément.

### **Performance**

#### **AQP1. Optimisation sur la latence**

Vous devez être en mesure de configurer votre système pour utiliser les microservices ayant les plus faibles latences pour réaliser le calcul du temps de trajet. Sont inclus les microservices développés par toutes les équipes de votre sous-groupe.

#### **AQP2. 1000 clients**

Proposer un scénario de performance pour que votre système soit en mesure de supporter 1000 clients simultanément en utilisant les données des services externes.

- Démontrez **mathématiquement** que la solution proposée est réalisable.
- Vous devez documenter cette solution sans en faire l'implémentation.

### **Interopérabilité:**

#### **AQI1. Utilisation des microservices des autres équipes**

Vous devez pouvoir utiliser les services fournis par les autres équipes du laboratoire uniquement en modifiant

dynamiquement la configuration de votre système.

## **AQI2. Découverte dynamique de service**

Votre système devrait pour récupérer la liste des microservices qu'il a besoin pour s'acquitter de sa tâche et d'utiliser ceux-ci dynamiquement.

# **Sécurité**

## **AQS1. Authentification**

Vous devez mettre en place un mécanisme pour que votre service ne soit accessible que pour les équipes autorisées.

## **AQS2. Mainten audit trail**

Appliquer la tactique "Mainten audit trail" dans un service centralisé pour faciliter la détection de tentative non autorisée de connexion. Vous devriez enregistrer l'adresse IP de l'appelant pour chaque tentative.

## Vues et Mappage entre les vues

# Vues et Mappage entre les vues

## Vues et spécifications remplies

### Vues équipe 2

Vue	Cas d'utilisation	Attributs de qualité
<a href="#">IServiceBaseImplementationView</a>	CU02	AQT1, AQD2, AQD3
<a href="#">Service Manager</a>		AQM2
<a href="#">ChaosMonkey</a>	CU02	
<a href="#">Redondance Passive</a>		AQM3, AQD4
<a href="#">GMaps Service</a>	CU07	
<a href="#">Message Trace</a>		AQT5
<a href="#">Simulator Calculator View</a>	CU01	AQT2
<a href="#">Simulator Calculator Data Pipeline</a>	CU01	AQT2
<a href="#">Service Discovery Library</a>		AQI1, AQI2
<a href="#">AggregationView</a>		AQU2
<a href="#">RecordPlaybackView</a>		AQT4
<a href="#">Redondance Active</a>		AQD5, AQM4

### Vues équipe 3



Vue	Cas d'utilisation	Attributs de qualité
<a href="#">GoogleMapsService</a>	CU07	AQD3, AQT1, AQD2, AQD3
<a href="#">ServiceDiscovery</a>	CU02, CU01	AQD1, AQD2, AQD3, AQI2, AQP1, AQD4, AQS1, AQS2, AQT2, AQT1, AQP2
<a href="#">ServiceManagerWebapp</a>	CU02	AQU1, AQU2
<a href="#">RequestInterceptor</a>	-	AQT5
<a href="#">GoogleMapsProxy</a>	-	AQD5
<a href="#">RouteComparator</a>	CU01	-

#### Vues équipe 4

Vue	Cas d'utilisation	Attributs de qualité
<a href="#">Vue du module BingMapsExternalService C&amp;C</a>	CU07	AQD2, AQD3, AQD4, AQT5, AQI1, AQS1
<a href="#">Vue du module RouteComparatorWebApp</a>	CU01	AQT5, AQM1
<a href="#">Vue du module RouteCalculatorService C&amp;C</a>	CU01	AQD2, AQD3, AQD4, AQI2, AQT5, AQD5,
<a href="#">Vue du module ChaosMonkeyService</a>	CU02	N/A
<a href="#">Vue d'allocation pour RouteCalculatorService</a>	CU01	AQD5

#### Vues équipe 5

Vue	Cas d'utilisation	Attributs de qualité
<a href="#">Interface spécialisée</a>		AQT1

<a href="#">Limiter les opérations non déterministes</a>	AQT3
<a href="#">BingMaps</a>	CU07
<a href="#">RouteComparatorService</a>	CU01

## Vues équipe 6

Vue	Cas d'utilisation	Attributs de qualité
<a href="#">mapbox-top-level-module-uses-view</a>	CU07	AQT1
<a href="#">Montreal-Data-Simulation-Top-Level-Module-Uses-View</a>	Util	AQT1
<a href="#">DeploymentView</a>	-	AQD4 AQD5
<a href="#">RouteCalculator</a>	-	-
<a href="#">LoggingCnCview</a>	-	AQS2
<a href="#">gps-simulation-service-uses</a>	CU05, CU06	-
<a href="#">MapboxActiveRedundancyUsesView</a>	-	AQM4, AQD5

## Vues globales

Vue	Cas d'utilisation	Attributs de qualité
<a href="#">Architecture Globale</a>	N/A	N/A

## Vues et Vues

### Équipe 2

Vue 1	Vue 2	Lien
<a href="#">Service Manager</a>	<a href="#">ChaosMonkey</a>	Parent-Enfant

## Vues équipe 3

Vue 1	Vue 2	Relation
<a href="#">Service Manager WebApp</a>	<a href="#">ServiceDiscovery</a>	Utilisation
<a href="#">Service Manager WebApp</a>	<a href="#">ChaosMonkey</a>	Utilisation
<a href="#">GoogleMapsProxy</a>	<a href="#">GoogleMapsService</a>	Utilisation
<a href="#">RequestInterceptor</a>	<a href="#">GoogleMapsService</a> , <a href="#">ServiceDiscovery</a> , <a href="#">RouteComparator</a>	Aspect (Crosscut)
<a href="#">Service Discovery</a>	<a href="#">Sécurité du Service Discovery</a>	Parent-enfant

## Vues et interfaces

### Vues équipe 2

Vue	Interfaces
<a href="#">IServiceBaseImplementationView</a>	<a href="#">IService</a> (coordonnateurs)
<a href="#">GMapsExternalService</a>	<a href="#">IRouteCalculator</a> (coordonnateurs)
<a href="#">ChaosMonkey</a>	<a href="#">ChaosmonkeyApi</a> (équipe 2)
<a href="#">ServiceManager</a>	<a href="#">IService</a> (coordonnateurs)

### Vues équipe 3

Vue	Interfaces implémentées
<a href="#">GoogleMapsService</a>	<a href="#">IService</a> , <a href="#">IRouteCalculator</a>
<a href="#">ServiceDiscovery</a>	<a href="#">IService</a> , <a href="#">IServiceDiscovery</a>

[RouteComparator](#)[IService](#), [IRouteComparatorBackend](#)

## Vues équipe 4

Vue	Interfaces
<a href="#">Vue du module BingMapsExternalService C&amp;C</a>	CU07
<a href="#">Vue du module RouteComparatorWebApp</a>	CU01
<a href="#">Vue du module RouteCalculatorService C&amp;C</a>	CU01
<a href="#">Vue du module ChaosMonkeyService</a>	CU02
<a href="#">Vue d'allocation pour RouteCalculatorService</a>	CU01
<a href="#">VueRaffinement</a>	CU01

## Vues équipe 5

Vue	Interfaces
<a href="#">Interface spécialisée</a>	<a href="#">BingMaps</a> (équipe 5)
<a href="#">Limiter les opérations non déterministes</a>	<a href="#">RouteComparatorService</a> (équipe 5)
<a href="#">RouteComparatorService</a>	<a href="#">IService</a> (coordonnateurs)
<a href="#">BingMaps</a>	<a href="#">IService</a> (coordonnateurs)

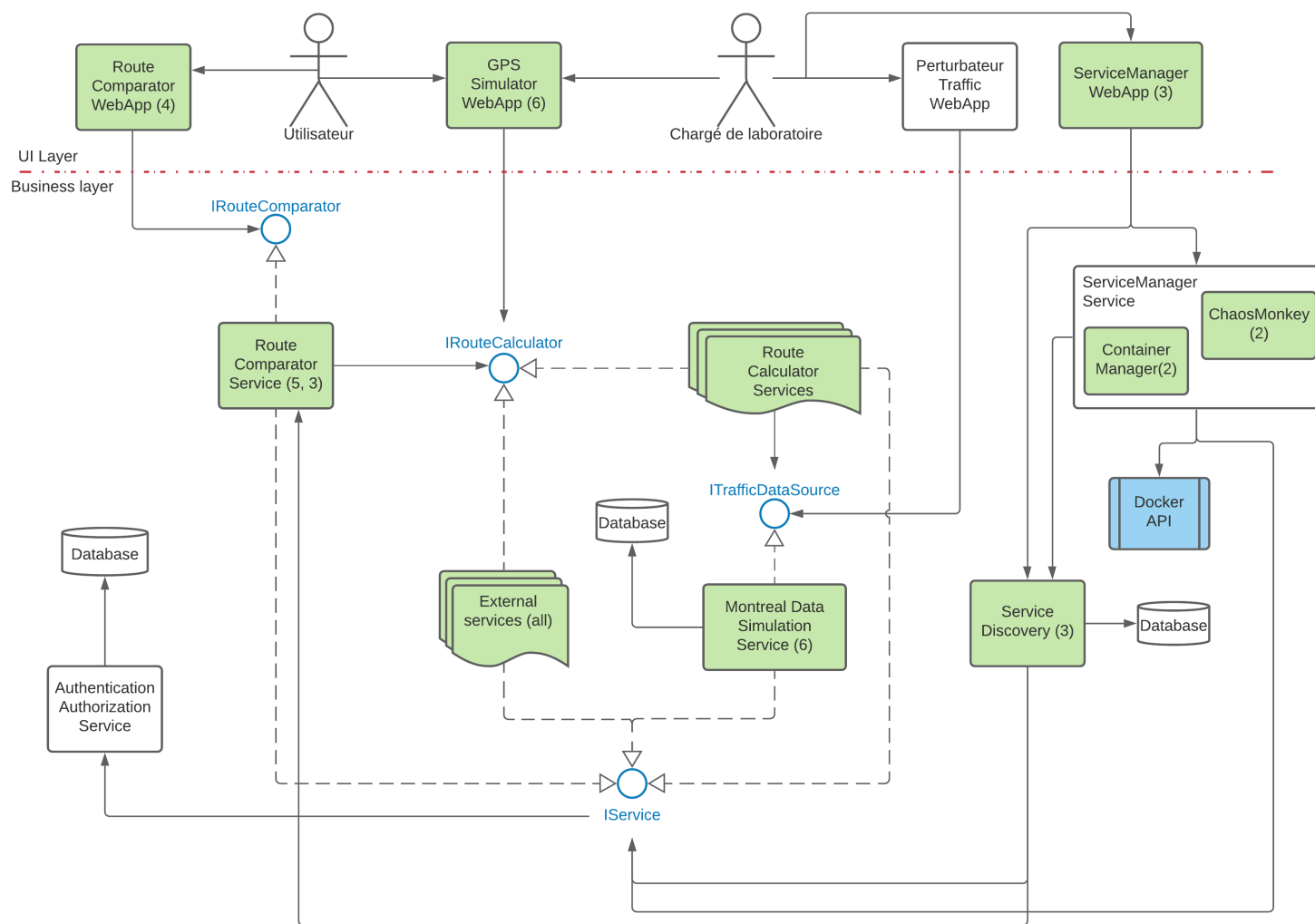
## Vues équipe 6


Vue	Interfaces implémentées
<a href="#">MapBoxService</a>	<a href="#">IService</a> , <a href="#">IRouteCalculator</a>
<a href="#">MontrealSimulationDataService</a>	<a href="#">IService</a> , <a href="#">ITrafficDataSource</a>
<a href="#">RouteCalculator</a>	<a href="#">ITrafficDataSource</a> , <a href="#">IRouteCalculator</a>

# Architecture-globale

## Présentation primaire

La vue suivante montre l'ensemble des services qui seront déployés dans le cadre du projet. Chaque service correspond à un ou plusieurs conteneurs docker.



 Les cases en vert sont des services complétés, les autres sont des services à compléter

## Catalogue des éléments

[RouteComparatorWebApp](#)

Web UI qui affiche et compare les calculs de route d'une origine vers une destination selon des services externes et/ou des services MQTT qui implémentent `IRouteComparator` (sélectionner ce qu'il faut comparer).

## GpsSimulatorWebApp

Web UI qui simule une application GPS selon un fournisseur [IRouteCalculator](#) sélectionné.

## PerturbateurTrafficWebApp

Élément perturbant le trafic pour voir l'impact sur le temps total de trajet. Interagit avec un [ITrafficDataSource](#) pour initier ces événements.

## ServiceManagerWebApp

Gestionnaire de service et générateur de chaos qui peut modifier la latence ou détruire un microservice selon différents critères. Doit aussi pouvoir détruire/faire arrêter/planter/terminer le processus de [ServiceDiscovery](#) pour que celui soit temporairement non disponible.

## [ServiceManager](#)

Gestionnaire de service qui contrôle les services déployés et offre un service de redondance passive pour les autres services.

Contient aussi l'API du générateur de chaos décrit dans ServiceManagerWebApp.

## [RouteComparatorService](#)

Comparateur des calculs du temps de trajet entre deux coordonnées GPS. Le calcul se fait sur le tronçon de la rue Notre-Dame seulement. On lui fournit une liste de services implémentant [IRouteCalculator](#) à comparer et [RouteComparatorService](#) fera les requêtes nécessaires pour comparer les résultats et retourner les informations de comparaison (à préciser).

## RouteCalculatorService

*À implémenter par chaque équipe*

Calculateur du temps de trajet à partir des données MQTT. Elles seront fournies à travers l'interface [ITrafficDataSource](#). L'algorithme est à la discrétion de chaque équipe.

## ExternalServices

Service de calcul de trajet externe. Ce service communiquera avec le service externe et retournera les informations selon le format de l'interface [IRouteCalculator](#). Chaque service sera implémenté à 2 reprises selon la disposition suivante:

Fournisseur	Nom du service	Équipes
Google Maps	GoogleMapsService	<a href="#">2</a> , <a href="#">3</a>

Bing Maps	BingMapsService	<a href="#">4</a> , <a href="#">5</a>
Mapbox	MapBoxService	<a href="#">6</a>

## **Montreal Data Simulation Service**

Service permettant d'analyser les fichiers historiques MQTT de la ville de Montréal, d'enregistrer ces informations dans une base de données puis de servir ces données selon des requêtes à travers l'interface [ITrafficDataSource](#). On précisera une intervalle temporelle lors de ladite requête afin d'identifier quelles données retourner.

## **ServiceDiscovery**

Service recevant des appels d'enregistrement de la part d'un [IService](#). Ensuite vérifie l'état (health checks) de ces services de façon régulière. Permet ensuite aux autres services d'utiliser le [ServiceDiscovery](#) afin d'avoir un répertoire des services actifs et en bonne santé.

## **Interfaces concernées**

Liste des interfaces concernées par la vue.

- [A0.1 IService](#)
- [A0.2 IRouteCalculator](#)
- [A0.3 ITrafficDataSource](#)
- [A0.4 ServiceDiscovery](#)

## **Diagramme de contexte**

Cette vue représentant le système de très haut niveau, la vue principale a ici aussi le rôle de diagramme de contexte.

## **Guide de variabilité**

Les configurations possibles de chacun des services sont décrites dans chacune de leurs vues.

## **Justification**

La séparation en plusieurs petits modules permet de respecter la loi de Conway et de mieux distribuer la charge de travail entre les équipes. De plus, séparer chaque tâche ainsi que de séparer le UI de la logique applicative permet une meilleure séparation des responsabilités (separation of concerns).





# IRouteCalculator

---

## IRouteCalculator

Version : 1.0.0

### Description de l'interface

IRouteCalculator est implémentée par tous les service de calcul de route. Cette interface offre une opération permettant de calculer le temps de trajet entre deux coordonnées GPS.

La description détaillée des **ressources** (endpoints), **types de données** et **gestion des erreurs** pour cette interface est disponible en format OpenAPI (Swagger) à l'adresse suivante: [Interfaces-API](#)

### Variabilité

N/A

### Attributs de qualité

**Modifiabilité** AQM1: Cette interface assure qu'il est possible de modifier les intersections utilisées pour le calcul du temps de trajet. L'opération du calcul de temps de trajet telle que décrite dans la documentation OpenAPI dépend des points d'origine et de destination utilisés. Il est ainsi possible pour tout service implémentant l'interface de modifier les points utilisés dans le calcul du temps de trajet.

### Justification de la conception

L'interface se doit d'avoir des paramètres simples à configurer et modifier. Il devient ainsi facile d'implémenter cette interface peu importe la complexité des calculs de routes ou la provenance des calculs dans le cas ou le service implémentant n'effectue pas lui-même les calculs.

Au niveau de la documentation de l'interface, l'utilisation de OpenAPI permet d'homogénéiser le retour de l'opération du temps de calcul chez les services implémentant l'interface en offrant une définition indépendante du paradigme et du langage.

### Exemples d'utilisation

Voir la page OpenAPI pour des exemples de requêtes et réponses pour chaque endpoint.

## Implémentations

Vue	Équipe
<a href="#">Google Maps Service View</a>	Équipe 2
<a href="#">Google Maps Service</a>	Équipe 3
<a href="#">Vue Bing Maps External Service</a>	Équipe 4
<a href="#">Bing Maps External Service</a>	Équipe 5
<a href="#">Map Decomposition View</a>	Équipe 6

Powered by [Wiki.js](#)

# IService

Documentation de l'interface IService

---

## IService

Version : 1.0.0

### Description de l'interface

Cette interface est utilisé par tous les services du projet. Elle fournis les points de terminaison commun à tous les services. IService permet aussi la communication entre le [ServiceDiscovery](#) et les autres services.

La description détaillée des **ressources** (endpoints), **types de données** et **gestion des erreurs** pour cette interface est disponible en format OpenAPI (Swagger) à l'adresse suivante: [Interfaces-API](#)

### Traçabilité des requêtes

Afin de permettre une meilleure traçabilité des messages à travers les différents services de l'architecture, nous avons mis en place un mécanisme qui permet d'associer un identifiant unique pour chaque flux de requêtes. Ceci permet de faciliter le débogage et les tests en permettant, notamment de filtrer un ensemble de logs par un identifiant de requête précis. Ce mécanisme permet aussi de faciliter le maintien d'une trace des accès, ce qui pourrait être utile pour effectuer des audits de sécurité éventuellement.

Afin que ce mécanisme fonctionne efficacement, chaque équipe doit implémenter la stratégie suivante dans ses microservices:

- ▶ Pour toutes les requêtes HTTP entrantes, si le header http `request_id` est défini, le même identifiant doit être retourné dans le header de la réponse HTTP.
- ▶ Si on reçoit une nouvelle requête HTTP sans header `request_id`, un nouvel identifiant unique (GUID) doit être généré et ajouté dans la réponse.
- ▶ Si un service reçoit une requête entrante et fait un appel à un autre service pour obtenir des données dans le même flux de requête, on doit faire suivre le `request_id` dans toute la séquence d'appel. Les services qui appellent d'autres service, doivent donc ajouter leur `request_id` actuel au header de leur propre requête.

- ▶ Enfin, pour chaque requête entrante, on doit logger un trace de la requête dans Portainer, accompagné de son `request_id` . Ces logs devraient contenir la date et l'heure de la requête, afin de faciliter le monitoring.

Header	Type	Requis	Example
request_id	GUID	Non	20a8dfda-17e6-47f2-bcfd-82c16e162d1e

## Modifiabilité

Non-applicable

## Attributs de qualité

### Disponibilité

- ▶ AQD2 : Grâce à cette interface, il sera possible de vérifier la disponibilité d'un microservice par un ping/écho.

### Testabilité

- ▶ AQT1 : Offre une "Interface spécialisé" pour permettre de modifier dynamiquement la latence des microservices de l'architecture.
- ▶ AQT5. "Interface spécialisé" pour suivre à la trace tous les messages entrant dans les microservices par l'association d'un numéro unique à chaque message entrant

## Justification de la conception

**Utilisation d'OpenAPI** : Nous avons documenté nos interfaces avec OpenApi pour réduire le risque d'erreur humaine lors de l'implémentation.

## Implémentations

Vue	Équipe
<a href="#">IServiceBaseImplementationView</a>	Équipe 2
<a href="#">GoogleMapsService</a>	Équipe 3
<a href="#">ServiceDiscovery</a>	Équipe 3
<a href="#">BingMapsExternalService</a>	Équipe 4

<a href="#">RouteComparatorService</a>	Équipe 5
<a href="#">BingMaps</a>	Équipe 5
<a href="#">Mapbox Decomposition View</a>	Équipe 6
<a href="#">Montreal Data Simulation Decomposition View</a>	Équipe 6

Powered by [Wiki.js](#)

## API Reference

# LOG430

API Version: 1.0.0

Documentation de base pour les micro services du laboratoire de LOG430.  
École de Technologie Supérieure - Hiver 2022.

En cas de question : @Coordonnateur sur discord

# INDEX

1. IROUTECALCULATOR	3
1.1 GET /route/{origin}/{destination}	3
2. ISERVICE	4
2.1 GET /api/info	4
2.2 PUT /api/latency	4

# API

## 1. IROUTECALCULATOR

Service that allow comparison of routes

### 1.1 GET /route/{origin}/{destination}

#### Get travel time between to points

Return the estimated route time between two points for a given provider.

#### REQUEST

##### PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*origin	string	Coordinates of the origin point (Format : LAT, LONG)
*destination	string	Coordinates of the destination point (Format : LAT, LONG)

#### RESPONSE

STATUS CODE - 200: Information about the time required to drive from origin to destination

##### RESPONSE MODEL - application/json

```
{
  date      string
  origin    string
  destination string
  duration  number  Duration in minutes
  distance  number  Distance in meters
}
```



## 2. ISERVICE

Basic endpoints that all micro services should implements

### 2.1 GET /api/info

#### Get API info

Retourne les informations de l'API

#### REQUEST

No request parameters

#### RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{  
  // Api info response which allows to check that the service is still active  
  serviceName    string  The name of the micro service  
  teamNumber     number  Number of the team that developed the service  
  currentLatency string  Current service latency  
}
```

---

### 2.2 PUT /api/latency

#### Set API latency

Permet de mettre à jours la latence pour effectuer des tests.

#### REQUEST

REQUEST BODY - application/json

```
{  
  latency*  number  Permet de modifier la latence (en milliseconde). Pour réinitialiser la latence, mettre la valeur '0'  
}
```

#### RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 400: BAD REQUEST

RESPONSE MODEL - application/json

```
{  
  statusCode  number  
  message     string  
}
```

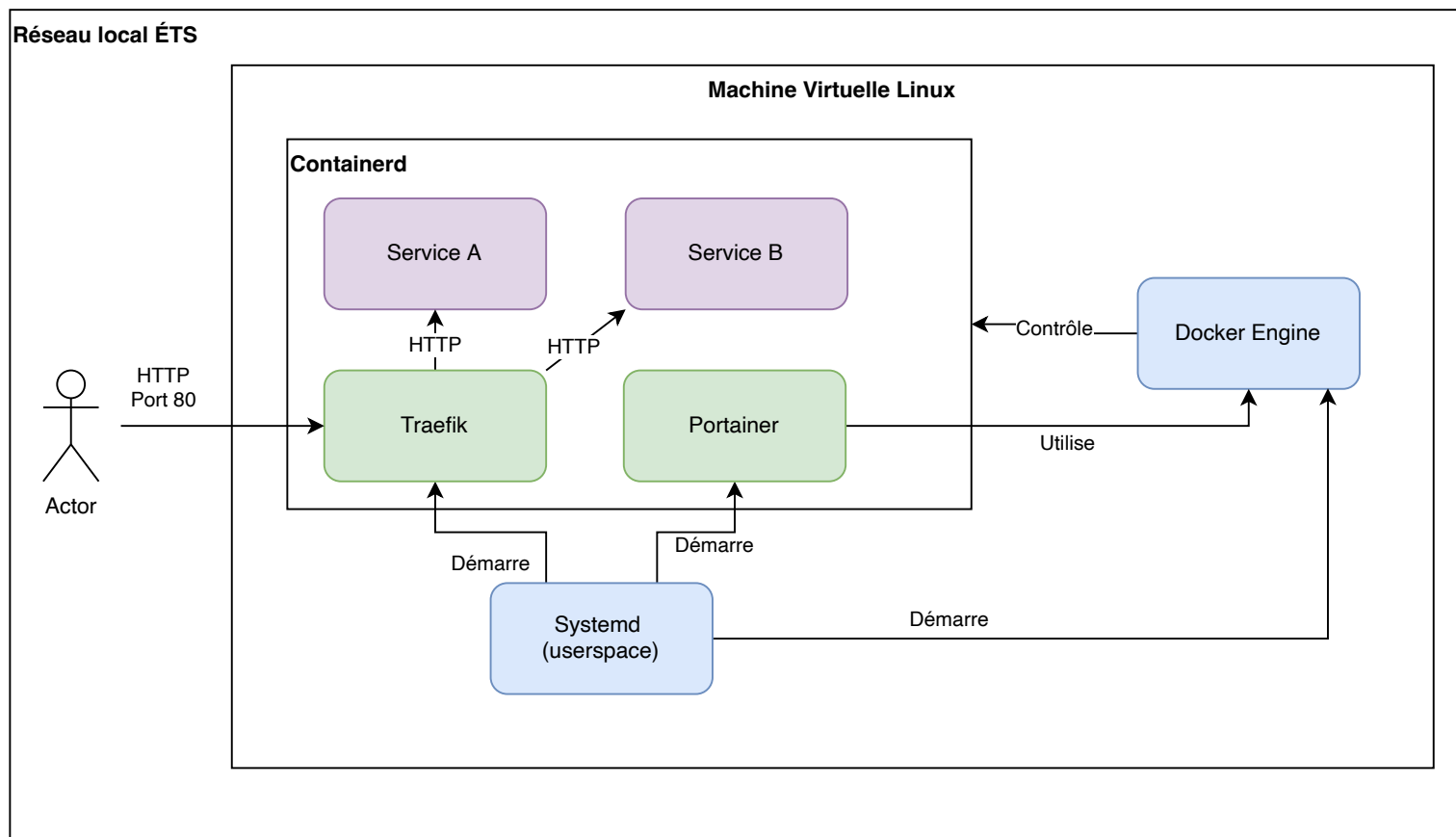
---

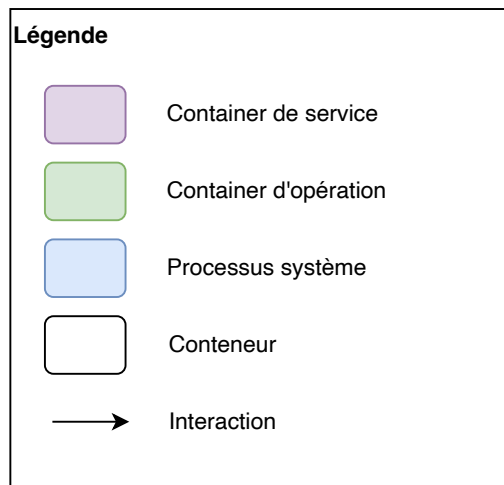
## Guide de déploiement

Nouvelle Section

# Guide de déploiement


## Vue du déploiement






## Catalogue des éléments


### Portainer

UI de gestion des conteneurs: <https://www.portainer.io/> 

### Traefik

Reverse Proxy et Load Balancer pour Docker : <https://traefik.io/traefik/> 

### Docker Engine

Gestionnaire de Containerd habituellement utilis/ en d/veloppement pour faciliter le d/ploiement de services: <https://www.docker.com/> 

## Prérequis

- Docker est installé en *rootless*
- docker-compose est installé
- 1 compte avec accès root et 1 compte sans accès root

## Étapes

1. Configurer les variables d'environnement ( `~/config/environment.d/10-docker.conf` ):

```
1 | DOMAIN=YOUR_DOMAIN_HERE
2 | DOCKER_HOST=unix:///run/user/YOUR_USER_UID_HERE/docker.sock
```

2. Avec un compte administrateur, modifier `/etc/sysctl.conf` pour y ajouter cette ligne:

```
1 | net.ipv4.ip_unprivileged_port_start=80
```

Ensuite exécuter la commande `sudo sysctl --system`

3. Cloner le répertoire de configurations: `git clone https://gitea.codegameeat.com:9443/simon/docker-swarm-configs.git`

4. Modifier `traefikv2/traefikv2.service` et `portainer/portainer.service` pour que les chemins des répertoires correspondent à votre configuration.

5. Créer des symlinks pour créer les services:

```
1 | ln -s ~/chemin/vers/repo/portainer/traefikv2.service ~/.config/systemd/user/traefikv2.service
2 | ln -s ~/chemin/vers/repo/portainer/portainer.service ~/.config/systemd/user/portainer.service
```

5. Ouvrir session screen et démarrer docker:

```
1 | screen
2 | systemctl --user start docker.service
```

6. Créer le network externe:

```
1 | docker network create traefik public
```

7. Démarrer les services:

```
1 | systemctl --user start traefikv2.service
2 | systemctl --user start portainer.service
```

8. Detacher de screen en conservant la session active: `ctrl a d`

9. Vous avez maintenant accès à traefik via [http://traefik.YOUR\\_DOMAIN](http://traefik.YOUR_DOMAIN) et à portainer via [http://portainer.YOUR\\_DOMAIN](http://portainer.YOUR_DOMAIN)

## Déployer un service sur portainer

Chaque service à déployer doit avoir une configuration *docker-compose*. Le format à respecter pour que les requêtes soit routés sur [http://NOM\\_SERVICE.HOST\\_IP.ni.io](http://NOM_SERVICE.HOST_IP.ni.io) est la suivante:

Le format suivant décrit le format [docker-compose](#) à respecter par un service pou avoir du routage. Toutes les valeurs en majuscules sont à personnalise.

```
1 | version: '3.2'
2 |
3 | services:
4 |   app:
5 |     image: VOTREIMAGE:VERSION
6 |     build:
7 |       context: .
8 |       dockerfile: SimulatorCalculator.Dockerfile
9 |     ports:
10 |       - PORT_HTTP # Doit être de ce format, l'exposition
11 |                   # d'un port sur l'hôte (port:port) n'est pas permis
12 |     networks:
13 |       - internal_NOEQUIPE_NOMSERVICE
14 |       - traefik_public
```

```


15     labels:
16         # traefik
17         - traefik.enable=true
18         - traefik.docker.network=traefik_public
19         # traefikv2
20         - "traefik.http.routers.NOEQUIPE_NOMSERVICE.rule=Host(`NOMSERVICE.NOEQUIPE.HOST_DOMAIN`)"
21         - "traefik.http.routers.NOEQUIPE_NOMSERVICE.entrypoints=http"
22         - "traefik.http.services.NOEQUIPE_NOMSERVICE.loadbalancer.server.port=PORT_HTTP"
23
24     networks:
25     traefik_public:
26         external: true
27     internal_NOEQUIPE_NOMSERVICE:

```

Le tableau ci-dessous décrit les détails de ce qui peut être configuré

Texte à modifier	Description	Restrictions
<b>VOTREIMAGE</b>	Nom de l'image à utiliser.	Le registry doit être publique ou enregistré dans l'Engine docker de l'hôte
<b>VERSION</b>	Version de l'image du conteneur	
<b>PORT_HTTP</b>	Port HTTP du conteneur	
<b>NOEQUIPE</b>	Numéro d'équipe dans le format eqX	
<b>NOMSERVICE</b>	Nom du service	<b>Unique pour l'équipe</b>
<b>HOST_DOMAIN</b>	Domaine de l'hôte	Doit pointer vers la machine contenant le Docker Host

Le déploiement sur portainer peut être fait directement en créant un stack sur le logiciel ou en configurant un répertoire git. Voir la documentation

Le déploiement sur portainer peut être fait directement en créant un stack ou le registrar ou en commitant un répertoire git (voir la documentation pour plus d'informations: <https://docs.portainer.io/v/ce-2.11/> 

Si le service est déployé via un répertoire git, il est aussi possible de build l'image docker directement sur portainer. En faisant un *pull & redeploy*, l'image sera automatiquement rebuild **si la version de celle-ci augmente**. Pour ajouter cette fonction, il suffit d'ajouter cela juste avant **image** dans le fichier docker-compose:

```
1 build:
2     context: .
3     dockerfile: Dockerfile
4 image: VOTREIMAGE:VERSION
5     ...
```

## Documents de référence

---

# Documents de référence

Ce document est basé sur un modèle mis à disposition par l'École de science informatique de l'Université de Carnegie Mellon.

<https://wiki.sei.cmu.edu/confluence/display/SAD/Main+Page> 

02 février 2022

## Matériel de référence

Auteur	Référence
Bachmann 2010	Garlan, Bachmann, Bass, Ivers, Little, Stafford, Clements, Merson, Nord, Documenting Software Architectures: Views and Beyond, 2nd_Edition, Addison- Wesley, 2010