

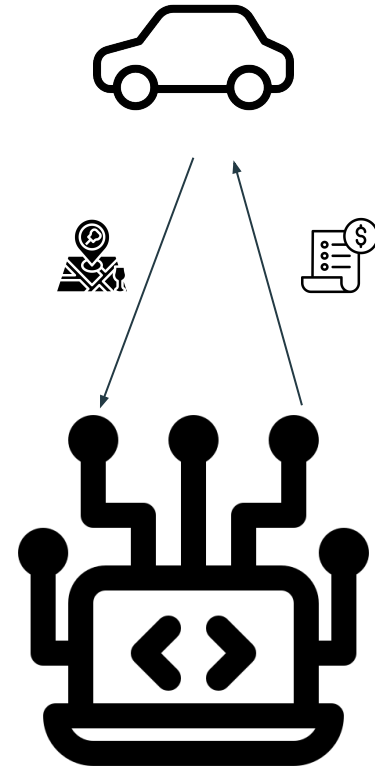
Architecture Logicielle: Evolution

V6 : Pay as you pollute

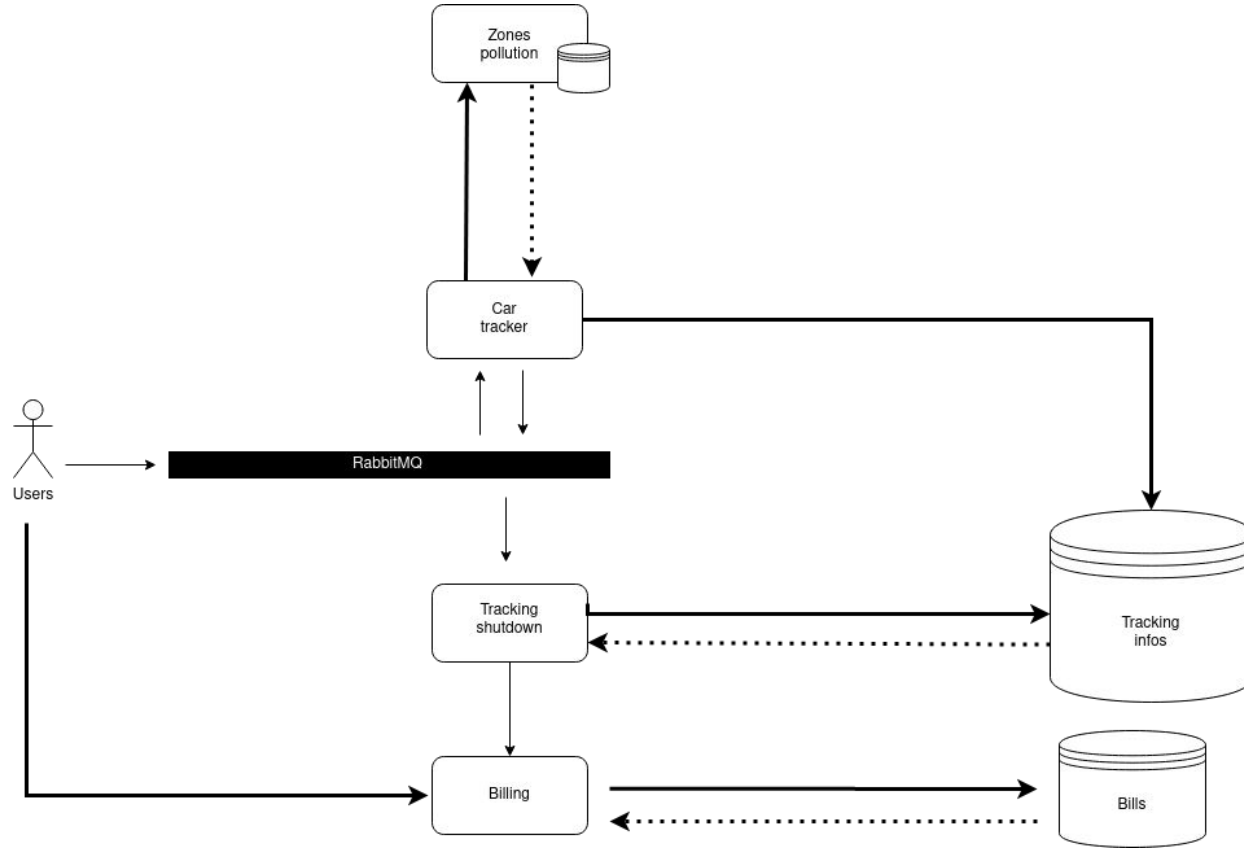
Equipe B : Guillaume Piccina, William D'Andrea, Nicolas Fernandez, Yann Brault

Rappel du sujet

- Suivi en temps réel
- Suivi des zones de pollutions
- Plan de facturation en fonction des zones
- Facturation en temps réel



Architecture logicielle: Construction



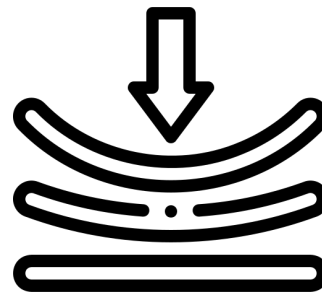
Rappel des problèmes et caractérisations

Volumétrie

- Gros flux de positions aux heures de pointes

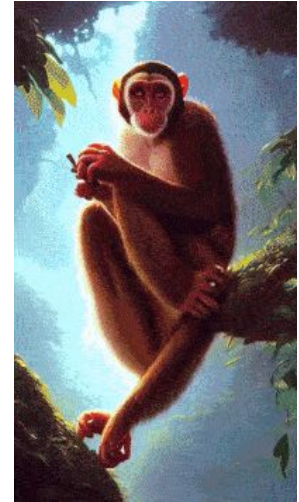
Résilience

- Système cohérent et disponible 24h/24

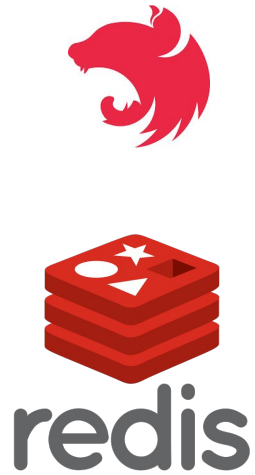


Besoins d'évolution

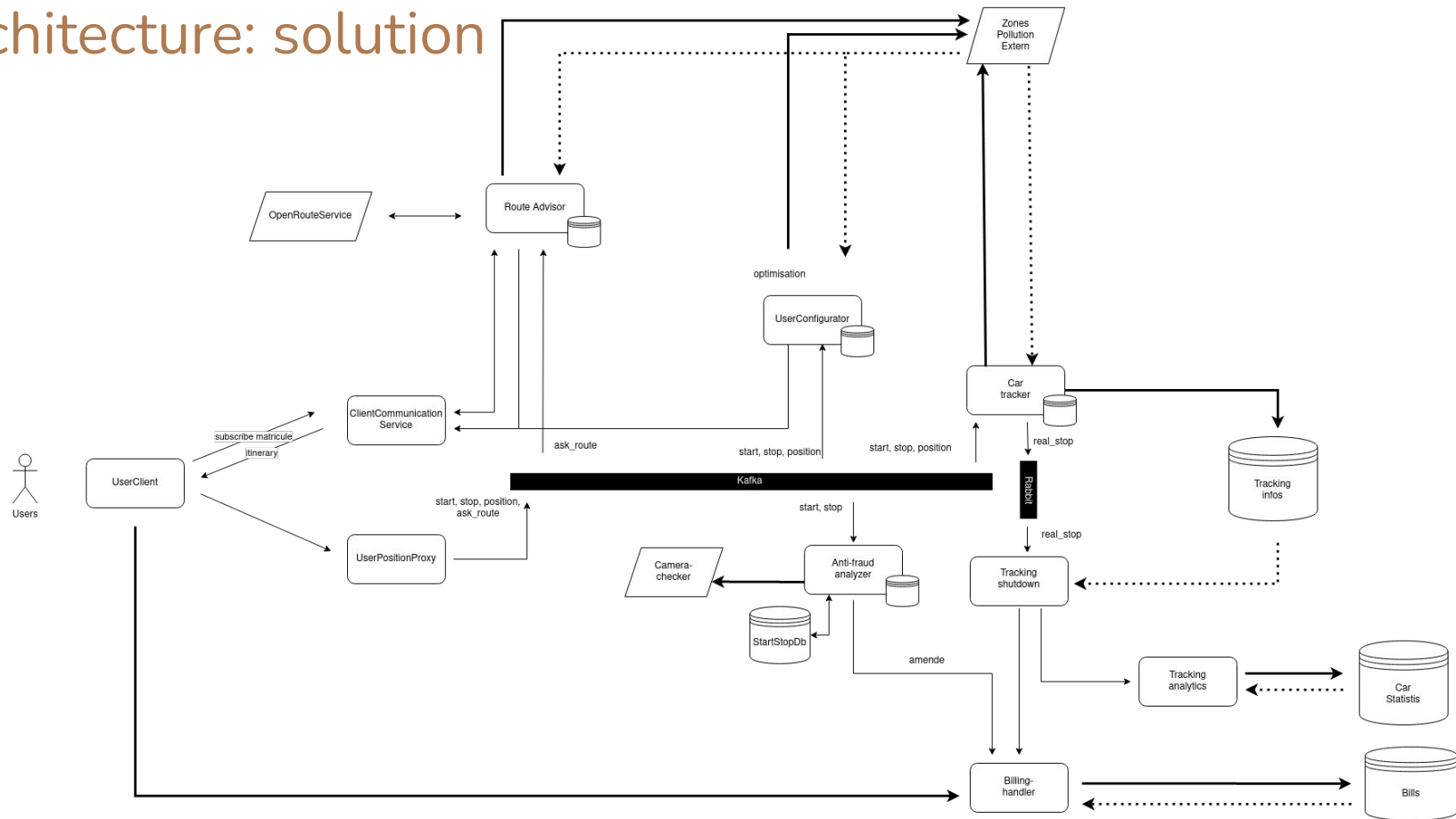
- Retour et de l'aide à l'utilisateur en temps réel via guidage
- Détection des coupures réseau avec détection de fraudes (par une carte des zones blanches connues) donc re calcul des montants après coup.
- Pousser la résilience et le maintien des infos de passage.



Choix technologiques



Architecture: solution



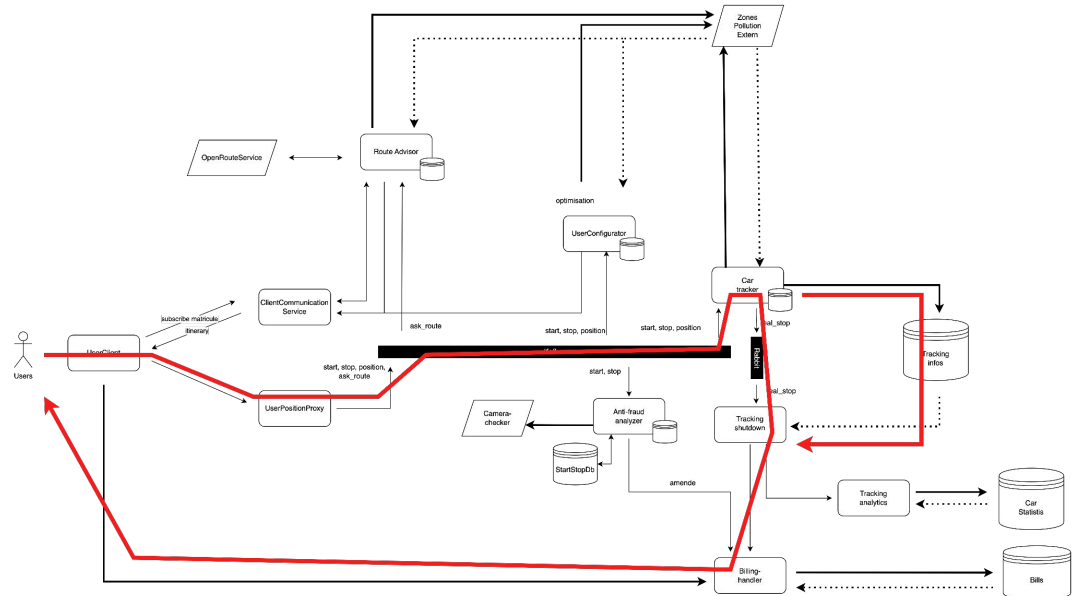
Justification des choix

Nos nouveaux choix technologiques se justifient par les nouveaux besoins fonctionnels:

- **Cache sur route-advisor, user-configurator, car-tracker**
 - Réduction du nombre d'appel a "zones-pollution-externe" (TTL 60s)
- **Queue RabbitMQ ⇒ bus Kafka**
 - Plusieurs micro-services qui écoutent les événements
 - Besoin de topics
 - Technologie que l'on maîtrise
- **Ajout d'une application pour l'utilisateur avec une map**
 - Gestion d'itinéraire
 - Accès aux factures
- **Mise à disposition des données statistiques**
- **Ajout du système d'anti-fraude**

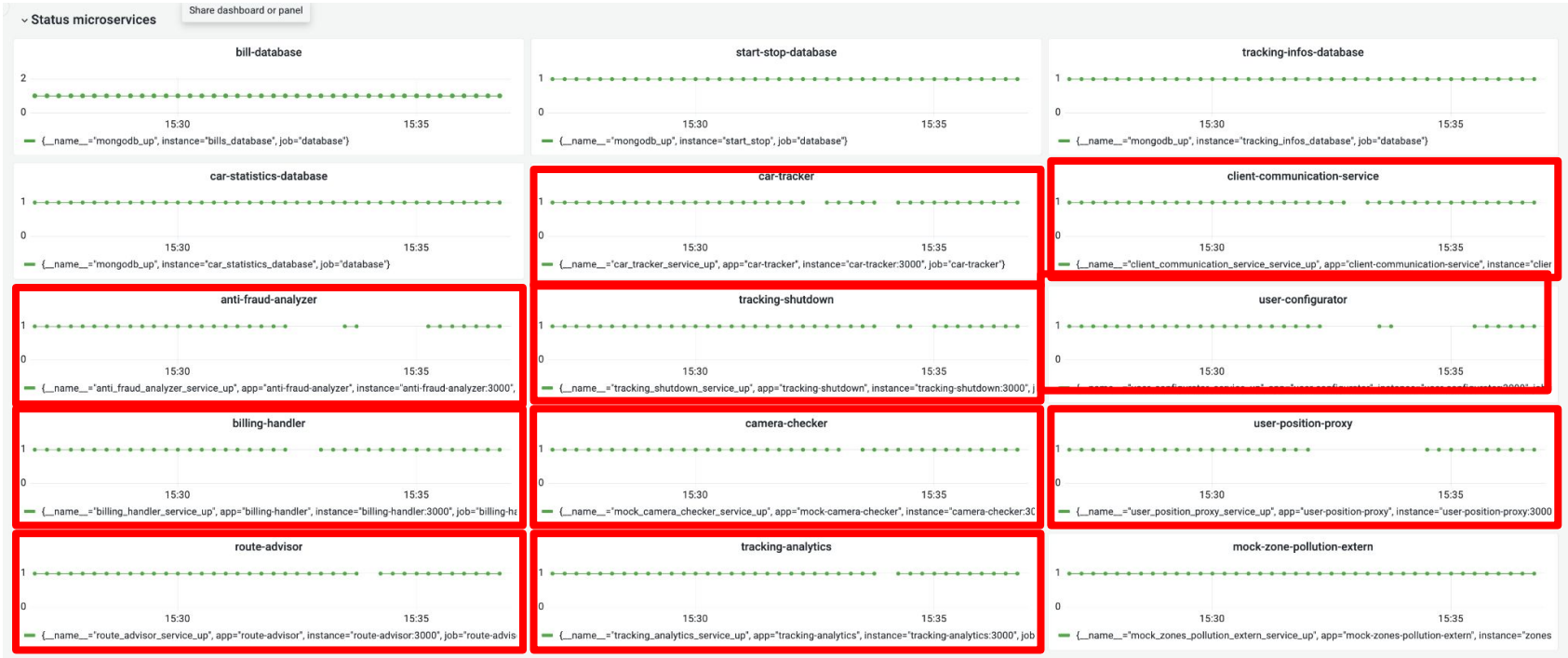
Montée en charge du système

- Chacun de nos micro-services possèdent une limite de mémoire de 128MB et 0.1 coeur
- Circuit “vital” qui doit être très résilient



Montée en charge - test 1

- 100 car starts
- 2000 positions envoyées (200 positions / voiture)
- 100 car stops

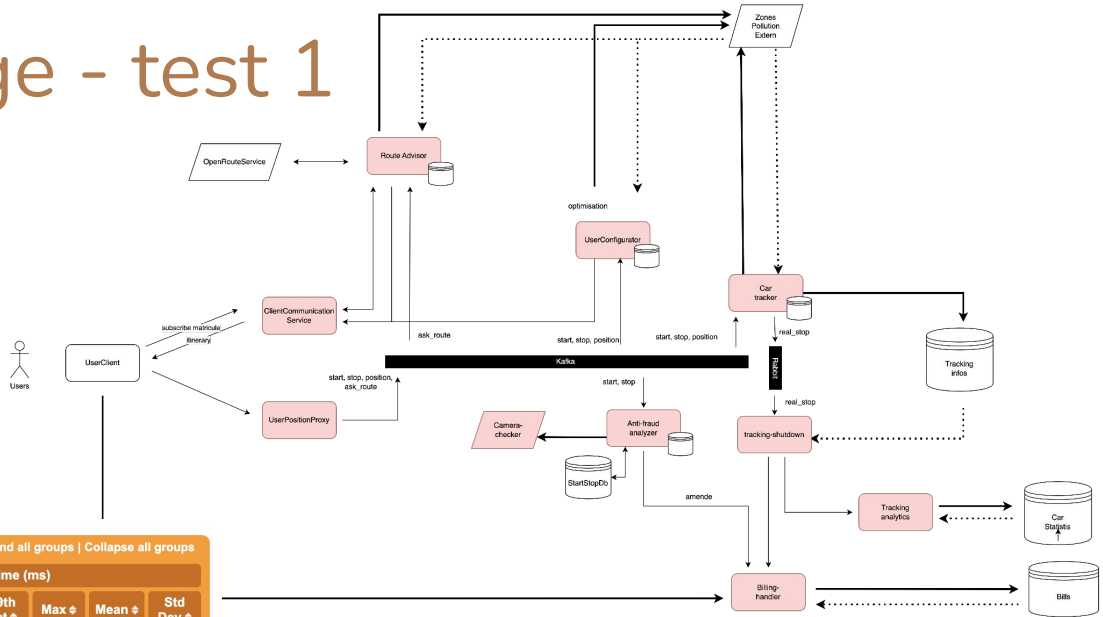


Montée en charge - test 1

Incohérences dans les données stockées

Système non résilient

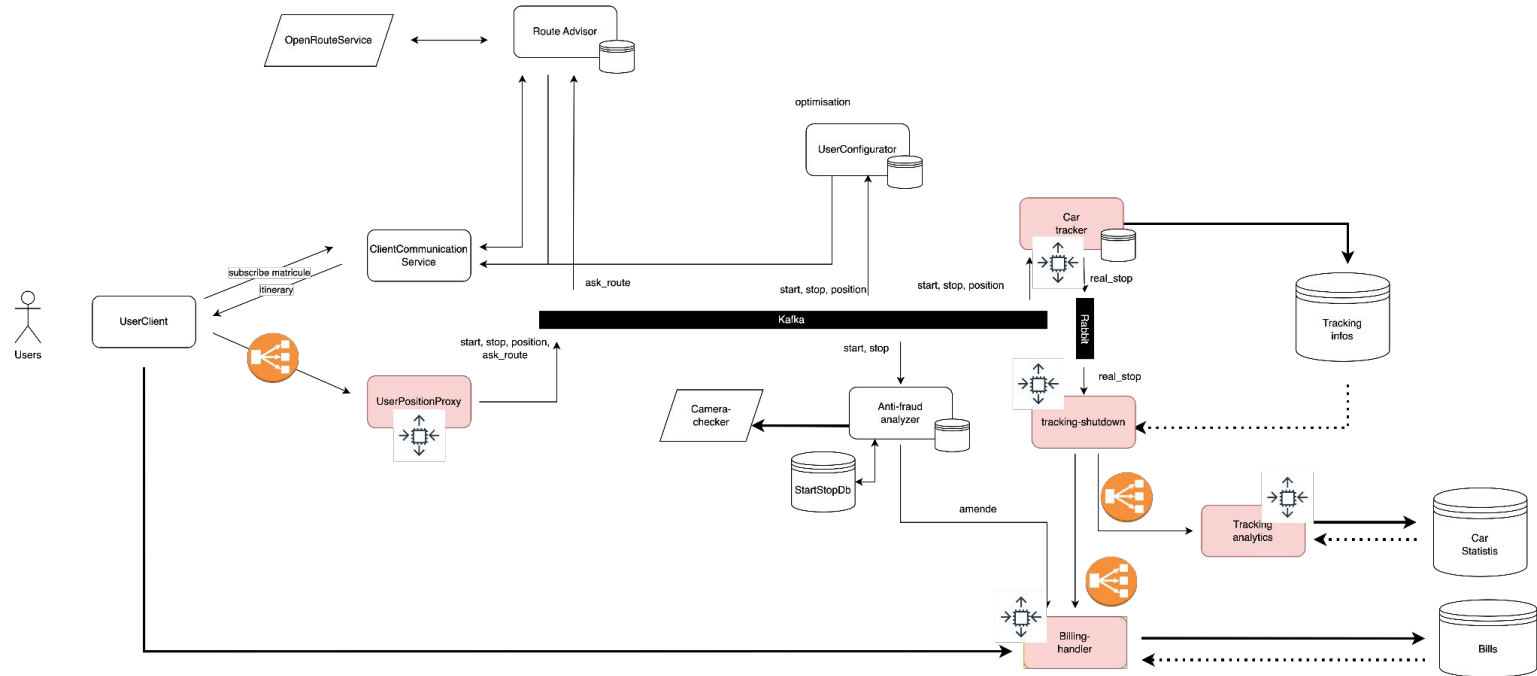
Mauvaise gestion de la volumétrie



STATISTICS														Expand all groups Collapse all groups	
Requests ^	Executions					Response Time (ms)									
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev		
Global Information	2200	913	1287	59%	29.333	132	47311	60159	60412	60467	60512	40216	22706		
car_start	100	100	0	0%	1.333	3743	7829	8810	10108	11188	11392	7875	1568		
car_position	2000	795	1205	60%	26.667	169	50729	60148	60407	60466	60512	41144	22184		
car_stop	100	18	82	82%	1.333	132	60268	60408	60466	60510	60510	54005	15618		

ERRORS				
Error	Count	Percentage		
i.g.h.c.I.RequestTimeoutException: Request timeout to localhost/127.0.0.1:6809 after 60000 ms	526	40.87 %		
i.g.h.c.I.RequestTimeoutException: Request timeout to localhost/0.0.0.0:0.0:1:6809 after 60000 ms	474	36.83 %		
j.l.IOException: Premature close	280	21.756 %		
j.n.SocketException: Connection reset by peer	7	0.544 %		

Montée en charge - modifications apportées



Montée en charge - après modifications

Positions : 100 start + 2000 voitures qui envoient leurs positions même temps



2.1k

DOCUMENTS

1

INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' }

Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 20 of 2100

```
{
  "_id": ObjectId('63c28f5a1d1cfaf676d2f2ff'),
  "license_plate": "DA-104-EF"
}
```

STATISTICS

Expand all groups | Collapse all groups

Requests

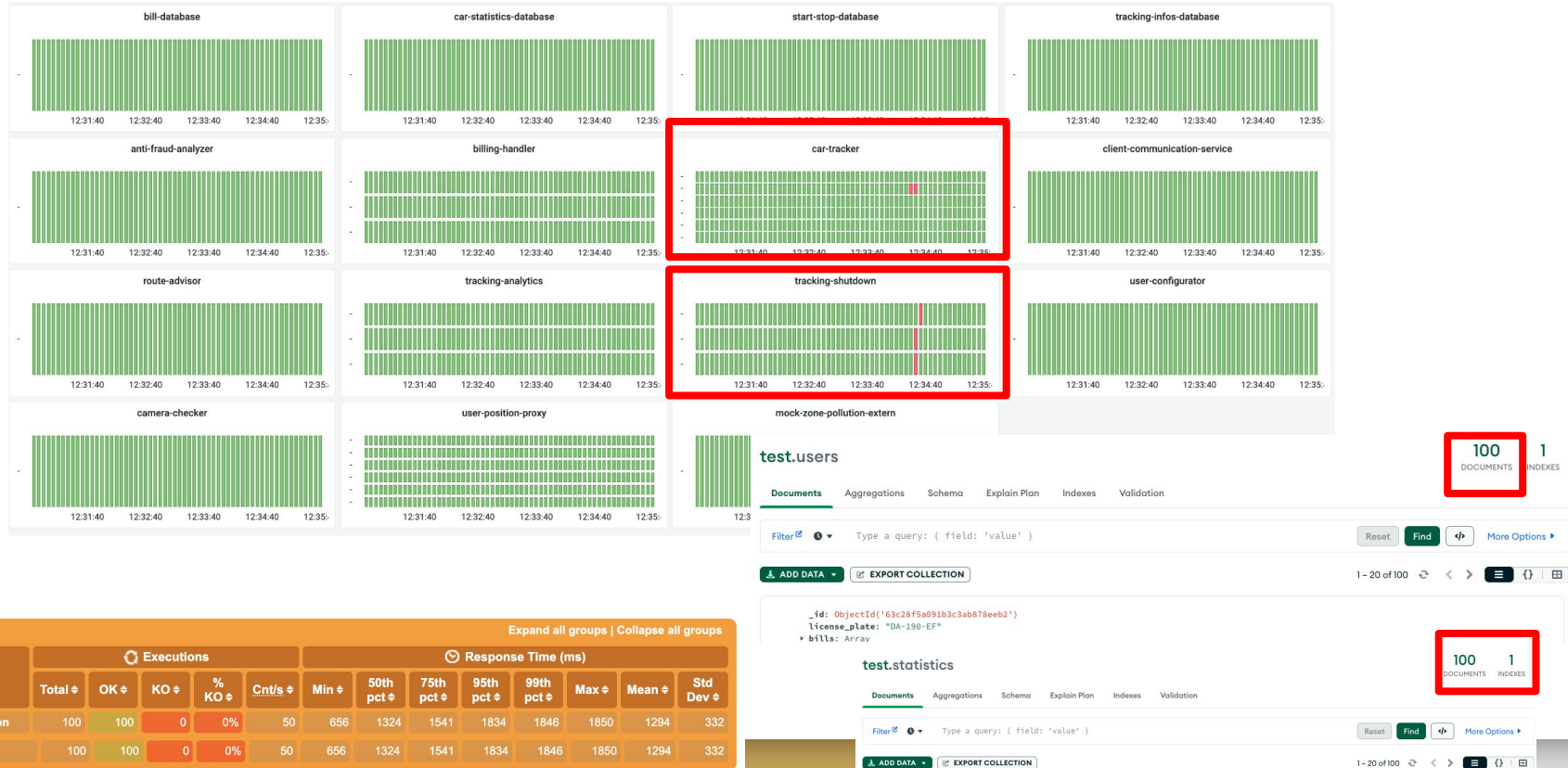
Executions

Response Time (ms)

	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean
Global Information	2000	2000	0	0%	100	441	9268	13579	18089	18781	19504	9748
car_position	2000	2000	0	0%	100	441	9268	13579	18089	18781	19504	9748

Montée en charge - après modifications

Stop : 100 voitures qui envoient leur STOP en même temps

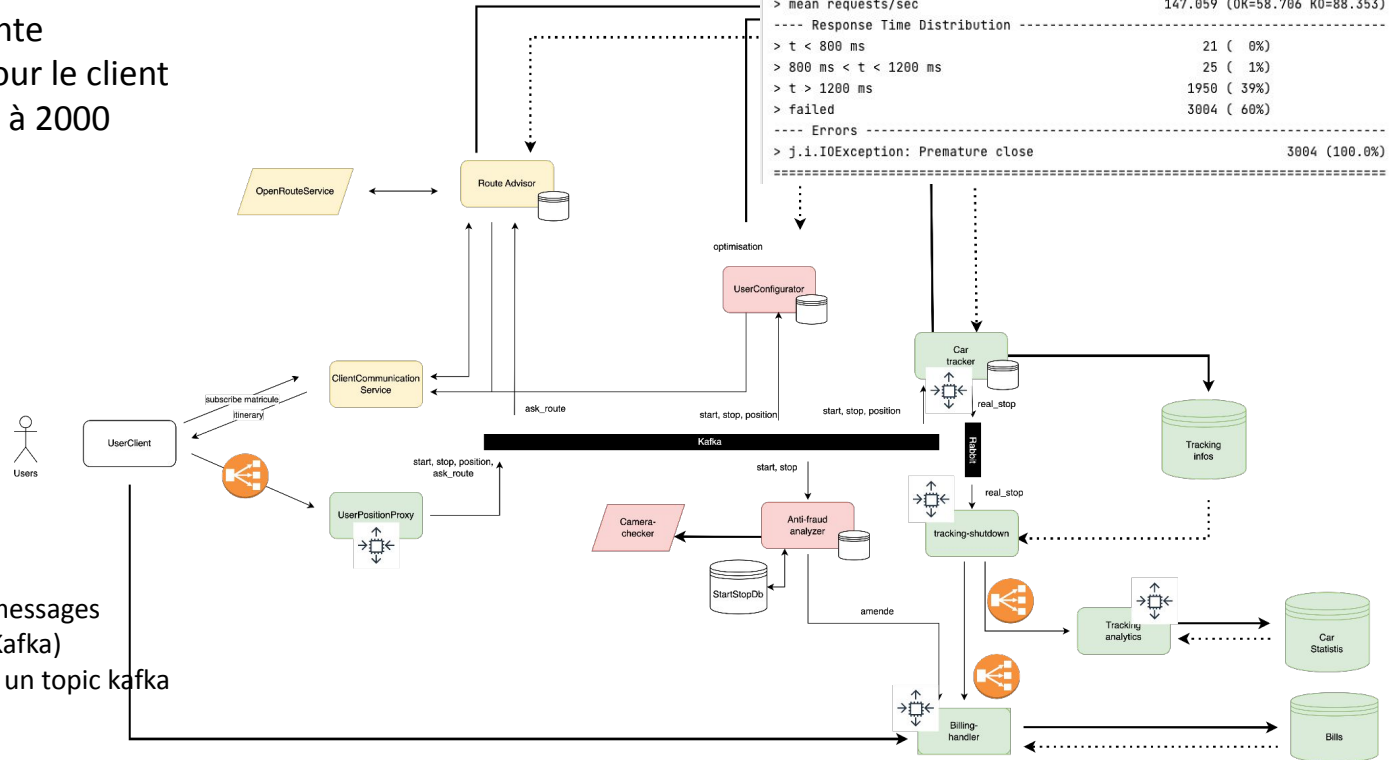


Montée en charge - Ligne primaire

Chaîne principale résiliente
et scalable facilement pour le client
(implémentation limitée à 2000
positions / secondes)

user-configurator et anti-fraud

- Choix 1 : Les scaler
- Choix 2 : Limiter le nombre de messages consommés (via un timing dans Kafka)
- Choix 3 : Le "front" envoie dans un topic kafka séparé



Montée en charge - Optimisations

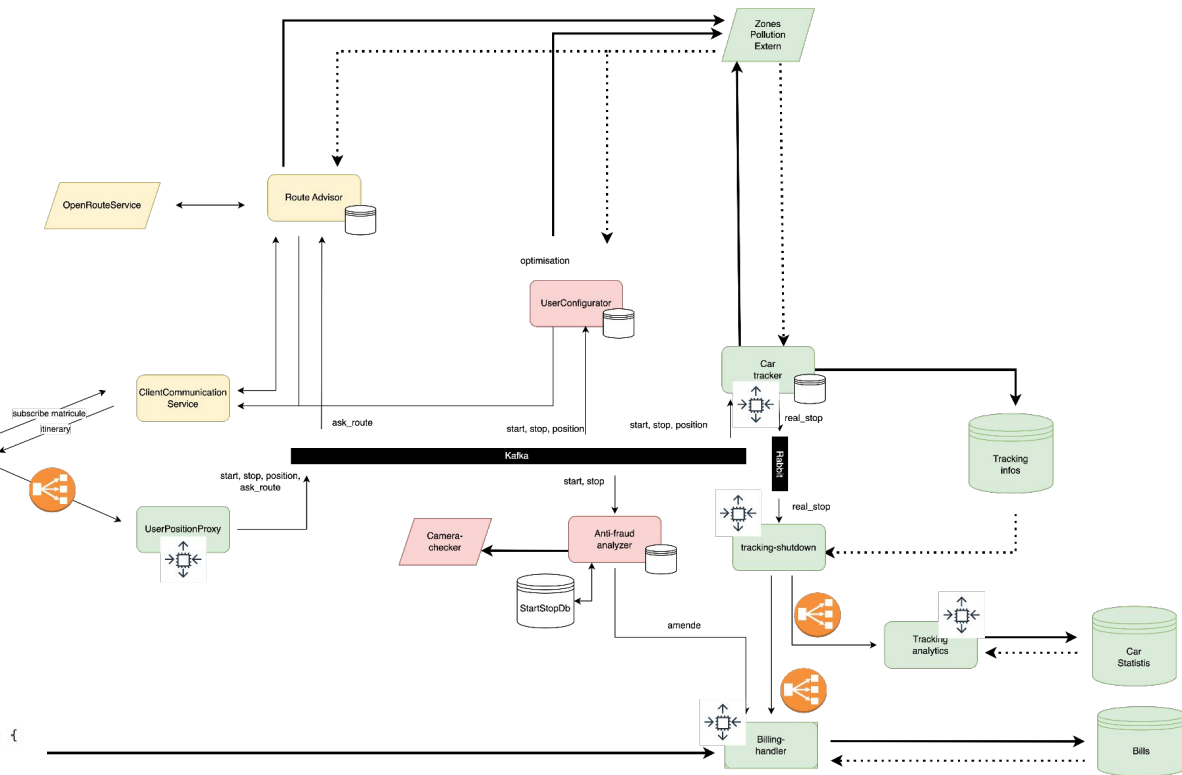
Hypothèse qui a des limites ...

user-configurator et anti-fraud

- Choix 1 : Les scaler
- Choix 2 : Limiter le nombre de messages consommés
- Choix 3 : Le "front" envoie dans un topic kafka séparé

user-position-proxy envoie un message sur 10 dans un topic séparé qui desservira anti-fraud-analyzer & user-configurator

- Si on vient à scaler anti-fraud-analyzer ou user-configurator, on aura qu'un message dans chaque réplique
- Choix plus facile pour l'utilisateur



```
// private PERCENTAGE_OF_OPTIMISATION_MESSAGES: number = 10;  
if (Math.floor( x: Math.random() * 100) < this.PERCENTAGE_OF_OPTIMISATION_MESSAGES) {  
  await this.kafkaClient.emit(  
    pattern: 'car-position-optimisation-and-fraud-topic',  
    data  
  );  
}
```


Montée en charge - Optimisations

STATISTICS

Requests ^

Executions

	Total ▾	OK ▾	KO ▾	% KO ▾	Cnt/s ▾	Min ▾
Global Information	100	100	0	0%	50	111
car_position	100	100	0	0%	50	111

~ anti-fraud-analyzer

Number of input events / minute



Number of output requests / minute



Number of success output requests / minute



Number of database call / minute



Number of car start input events / minute



Number of car position input events / minute

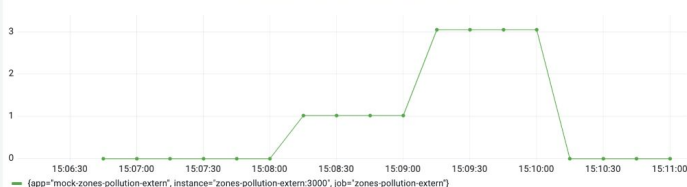


Number of car stop input events / minute

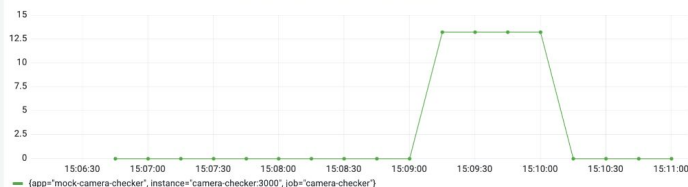


~ mocks

Zone pollution - number of zones requests / min

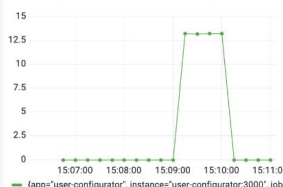


Camera - number of camera check requests / min

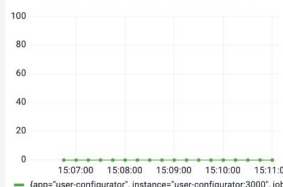


~ user-configurator

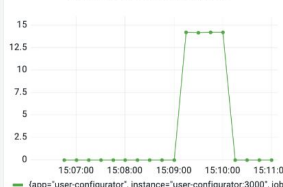
Number of car start events received / min



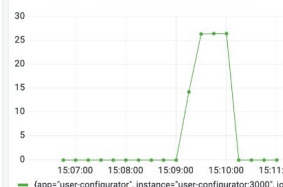
Number of car position events received / min



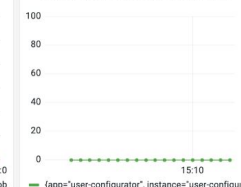
Number of external requests / min



Number of success external requests / min



Number of failed external requests / min



Montée en charge - Optimisations

STATISTICS

Requests ^

Executions

Global Information

car_position

Total ↕

OK ↕

KO ↕

%
KO ↕

Cnt/s ↕

Min ↕

50th
pct ↕

2000

1760

240

12%

80

223

11410

2000

1760

240

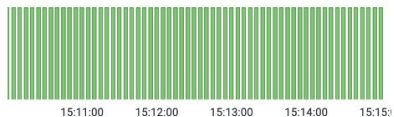
12%

80

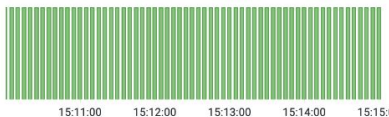
223

11410

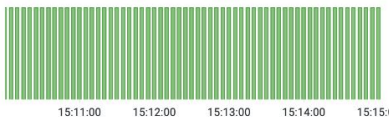
bill-database



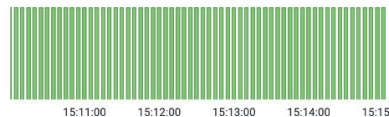
car-statistics-database



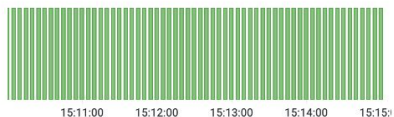
start-stop-database



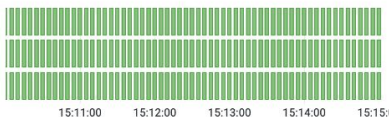
tracking-info-database



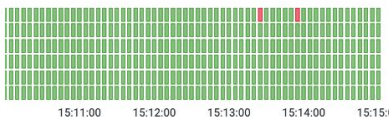
anti-fraud-analyzer



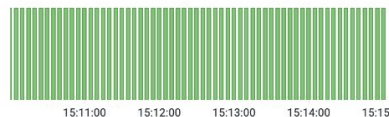
billing-handler



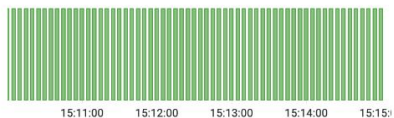
car-tracker



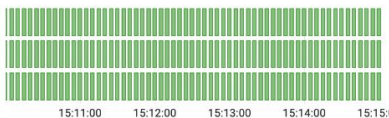
client-communication-service



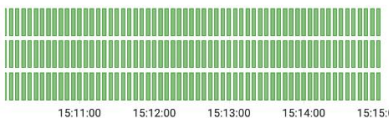
route-advisor



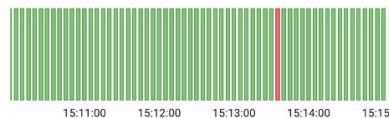
tracking-analytics



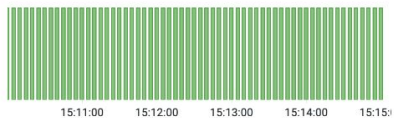
tracking-shutdown



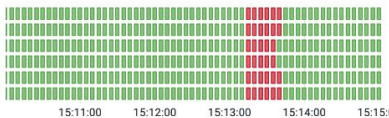
user-configurator



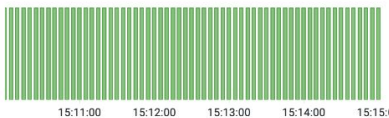
camera-checker



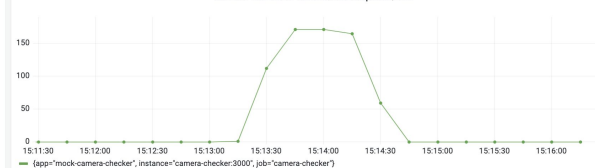
user-position-proxy



mock-zone-pollution-extern



Camera - number of camera check requests / min



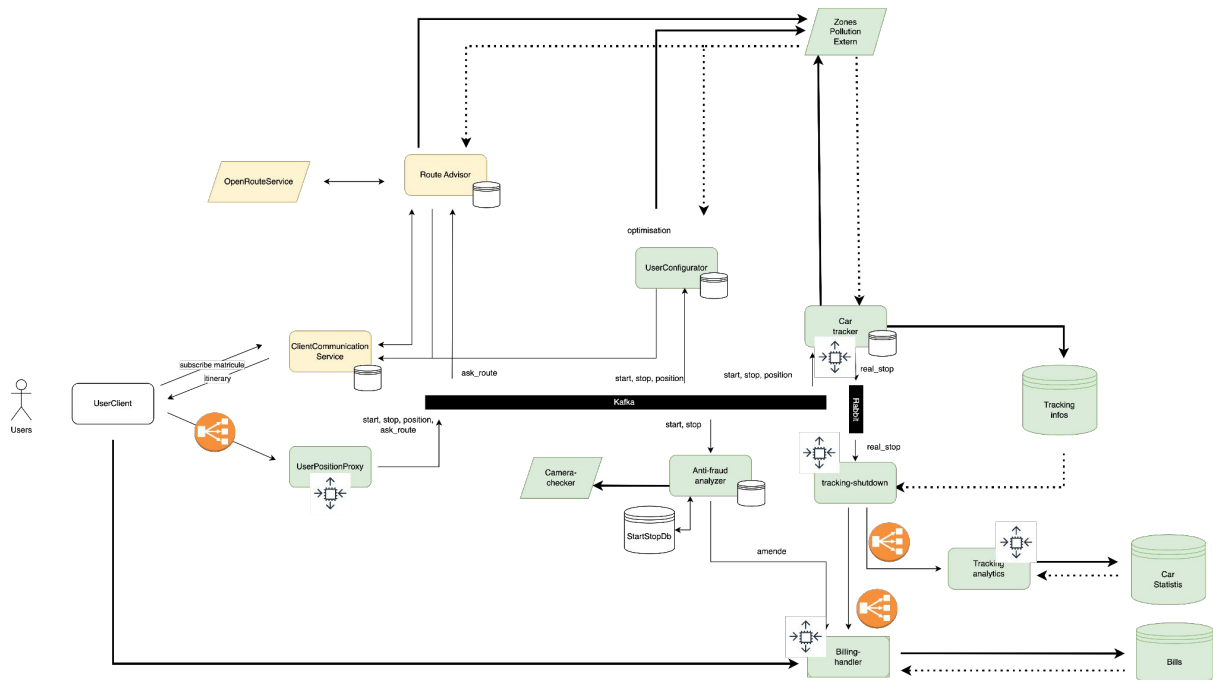
Montée en charge - Résilience de l'implémentation

open-route-service / route-advisor

=> requête sur demande, pas de test de montée en charge mais scaling facile



client-communication-service

=> scale vertical ou socket.io + sticky session
=> Stateful mais optimisé via cache Redis pour permettre un scale horizontal



Prise de recul

- **Beaucoup de load-balancer et scaling non pratique sous docker-compose**
 - Pour ce type de projet, kubernetes serait peut être plus intéressant qu'un docker-compose
- **user-communication-service difficile à scaler**
- **Projet lourd pour les machines**

Nom	Statut	Processe...	Mémoire	Disque
 Vmmon		58,7%	11 159,4 ...	1,4 Mo/s
 System		15,9%	0,1 Mo	7,5 Mo/s

Ce qu'il reste à faire

Majeur:

- Prendre en compte les zones blanches
- Proposition de trajets alternatifs
- Meilleure stratégie d'utilisation de user configurator et anti fraud service

Mineur:

- Scaler davantage le user-position-proxy
- Optimiser user-configurator
- Exploiter tracking-analytics



Organisation

- William : Scaling, test de charge et monitoring
- Tout le monde : Tous les aspects du projet

Merci !

