Admin User Guide

Version 1.0

Mon Compte Mobilité

Standardisation des MaaS - Gateway

Sommaire

[1. Introduction 4](#_Toc129073891)

[1.1. Définitions et termes 4](#_Toc129073892)

[1.2. Table des révisions 5](#_Toc129073893)

[2. Set-up 6](#_Toc129073894)

[3. CI/CD 7](#_Toc129073895)

[4. Composants 8](#_Toc129073896)

[4.1. Base de données 8](#_Toc129073897)

[4.1. Cache 9](#_Toc129073898)

[4.1. Microservices gateway 9](#_Toc129073899)

[4.1. Gravitee io 11](#_Toc129073900)

[4.2. Elastic stack : elasticsearch, kibana, apm 12](#_Toc129073901)

[4.3. Grafana : monitoring & kpi 22](#_Toc129073902)

Tables des illustrations

**Aucune entrée de table d'illustration n'a été trouvée.**

1. Introduction

Ce document présente l’exploitation du système MCM « Standardisation des MaaS » Gateway dans son écosystème, il sera complété et enrichi de manière itérative. Il fera références à des sections documentaires traitant tout ou parties des points abord.

Vous pouvez vous référer à ce document pour retrouver les changelogs de la Gateway ainsi que toutes les fonctionnalités qui ont été mise en place jusqu’à aujourd’hui en **production.**

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/CHANGELOG.md](https://github.com/fabmob/mcm-gateway/blob/main/CHANGELOG.md)

Vous pouvez également consulter ce document qui rappelle le contexte du programme MCM et de la Gateway, ainsi que le principe et les phases de publication du code source.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/Gouvernance.md](https://github.com/fabmob/mcm-gateway/blob/main/Gouvernance.md)

Vous retrouverez également toutes les informations, fonctionnels et technique, nécessaires à la compréhension de la gateway dans le DAT.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/docs/DAT-MCM\_Std\_MaaS\_GW\_v1.1.pdf](https://github.com/fabmob/mcm-gateway/blob/main/docs/DAT-MCM_Std_MaaS_GW_v1.1.pdf)

* 1. Définitions et termes

|  |  |
| --- | --- |
| Terme | Description |
| MSP | “Mobility Service Provider”, fournisseur d’un service de mobilité. |
| MaaS | “Mobility As A Service”, plateforme réunissant l’information, la réservation et le paiement de l’ensemble de l’offre de mobilité disponible. |
| MCM | Mon Compte Mobilité |
| moB | Produit du projet Mon Compte Mobilité |
| Std MaaS | Standardisation des MaaS |
| Gateway (GW) | Passerelle |
| CMS | Compte Mobilité Standardisé |
| HUB | Plateforme regroupant les services MCM de Gateway et de Compte Mobilité (MOB / CMS) |
| Mobilité durable | Mobilité décarbonée. |
| RGPD | Règlement Général sur la Protection des Données. |
| RGAA | Référentiel Général d'Accessibilité pour les Administrations. |
| CNIL | Commission Nationale de l'Informatique et des Libertés |
| CEE | Certificats d’Économie d’Energie |
| SIS | Système d'Information et de Services pour la billettique |
| SIV | Système d'Information Voyageur |
| TC | Transports en Commun |
| IV | Information Voyageur |
| RI | Recherche d'Itinéraires |
| IVTR | Information Voyageur Temps Réel |
| REF | Référentiel |
| IDFM | Île-de-France Mobilités |
| GART | Groupement des Autorités Responsables de Transport |
| AOM | Autorité Organisatrice de Mobilité |
| PRIM | Plateforme Régionale d’Information pour la Mobilité |
| VE | Véhicule Électrique |
| VLS | Vélo en Libre-Service |
| VTC | Véhicule de Tourisme avec Chauffeur |
| PMV | Produit Minimum Viable (MVP Minimum Viable Product) |

* 1. Table des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Description | Auteur | Date |
| 1.0 | Initialisation du document | Capgemini | 27/02/2023 |

1. Set-up

Pour installer l’environnement requis afin de faire tourner la Gateway en local, suivez les instructions dans ce document.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/docs/Installation\_Environnement.docx](https://github.com/fabmob/mcm-gateway/blob/main/docs/Installation_Environnement.docx)

Une fois l’environnement installé, consultez le lien suivant pour configurer la base de données, le cache redis, le lancement des tests et retrouver comment les erreurs sont gérées et quels standards sont supportés dans chacun de nos endpoints.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/README.md](https://github.com/fabmob/mcm-gateway/blob/main/README.md)

1. CI/CD

La Continuous Integration et Continious Developement mis en place sur le projet est basé sur l’utilisation de gitlab (premium).

Le fonctionnement de la pipeline est décrit dans le readme suivant : <https://github.com/fabmob/mcm-gateway/blob/0.7.2/README_DATAMAPPING.md>

1. Composants
   1. Bases de données
      1. Description

On considère qu’il existe 3 types de bases de données pour faire tourner la gateway et ses composants :

* Une base de données NoSQL utilisée pour le stockage des logs de gravitee ainsi que pour la partie application performance monitoring. La base de données retenue est Elasticsearch déployée sous forme de pods dans le cluster Kubernetes de la solution Gateway.

Lors de la mise en place d’elasticsearch il est recommandé de définir le mot de passe pour le compte kibana\_system en lançant la requête surl suivante sur un des pods elasticsearch (master ou data).

curl -XPOST "http://localhost:9200/\_security/user/kibana\_system/\_password" -H 'Content-Type: application/json' -d'{ "password" : "MYPASSWORD"}' -u "elastic:MYPASSWORD"

* Une base de données relationnelle (SQL) utilisée pour le stockage de la configuration de l’api manager Gravitee. La base de données retenue est postgresql en mode PaaS.
* Une autre base de données relationnelle (SQL) utilisée pour le stockage de la configuration des MSP, MaaS, services, data-mapping etc… qui correspond à tout le cœur du modèle de données de la partie microservices. La base de données retenue est postgresql en mode PaaS.

Le document suivant explique comment fonctionne le data-mapping en relation avec notre modèle de données ainsi que l’utilisation de chaque attribut des tables de la base de données.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/README\_DATAMAPPING.md](https://github.com/fabmob/mcm-gateway/blob/main/README_DATAMAPPING.md)

* + 1. Administration
       1. Access

Elasticsearch :

Afin d’accéder à la base NoSQL il faut se connecter en port forwarding et utiliser les API REST mises à disposition par la solution elasticsearch. Afin de se connecter à elasticsearch un login et mot de passe sont requis, ils correspondent à ceux utilisées lors de l’installation.

Postgresql :

L’accès à la base de données postgresql peut soit se faire indirectement par l’utilisation du microservice data-api (mode recommandé) mais on peut aussi contacter le service PaaS. Pour cela on peut avoir besoin de provisionner un pod technique spécifique aux opérations des ops et ce basé sur une image contenant un client postgres. De ce fait en faisant du port forwarding on peut se placer dans le cluster et accéder à postgres à l’aide de la commande psql si les règles de restriction du serveur de base de données n’accepter comme seule origine le cluster K8S.

Pour accéder à la base de données via data-api en local il suffit de respecter les étapes dans ce document :

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/README.md](https://github.com/fabmob/mcm-gateway/blob/main/README.md)

Note : côté microservices, afin d’accéder à la base de données sur un environnement déployé en mode helm-chart, il faut fournir le mot de passe dans une variable d’environnement nommé POSTGRES\_PASSWORD

* + - 1. Logs

Les logs de base de données sont gérés par la solution PaaS mise en disposition.

Ce qui est plus courant c’est de consulter les logs de data-api afin de consulter les logs du seul applicatif qui peut communiquer avec la base de données postgresql.

* + - 1. Backups

Le backup de la base de données postgresql est à paramétrer et gérer au niveau du service PaaS. Si on souhaitait avoir un backup de la base NoSQL il faudrait procéder au backup des disques utilisés par les pods elasticsearch au travers des storageClass.

* + 1. Opérations de Maintenance en Conditions opérationnelles

Toutes les opérations de mise à jour des données sont effectuées par un admin via data-api.

* + 1. FAQ & astuces

**Est-ce qu’il y a des fichiers de migrations pour reproduire / revenir en arrière ?**

Non, il n’existe à ce jour, pas de fichiers permettant de reproduire ou revenir en arrière. Un retour sur un précédent dump de la base de donnée permet de restaurer l’état du paramétrage de la gateway.

**Quels sont les modules liés à la base de données relationnelle ?**

Gravitee car toute la configuration est stockée en base de données.

Data-api, c’est le seul microservice se connectant à la base de données.

* 1. Cache
     1. Description

Le cache *redis* permet d’accéder aux données beaucoup plus rapidement qu’en utilisant la base de données. Nous stockons certaines réponses des MSPs, ainsi que la liste des partenaires (MSP ou MaaS) sauvegardés en base.

Le cache est actuellement fonctionnel, mais la partie gestion des messages d’erreurs nécessite quelques retouches lorsque ce dernier est activé.

Le cache redis est un composant utilisé en mode PaaS.

* + 1. Administration
       1. Access

On accède au cache en utilisant un mot de passe. Les accès sont limités au paramétrage de sécurité réalisé sur le composant PaaS.

* + - 1. Backups

Il n’existe pas de backup pour le cache.

* + 1. Opérations de Maintenance en Conditions opérationnelles

**Activer/Désactiver le cache**

Pour activer ou désactiver le cache sur un environnement sécurisé, un admin gateway doit mettre à jour la valeur de CACHE\_ACTIVATION présente dans la table *gateway\_params* de la base de données en contactant data-api, en utilisant la bonne url et en fournissant son api-key en tant que header et le body contenant la valeur à mettre à jour. Ce faisant, la Gateway va synchroniser la valeur dans le cache.

**Note :** Il ne s’agit pas d’activer ou désactiver redis, mais plutôt de mettre à jour la valeur stockée dans la base de données ET dans le cache pour savoir si nous devons utiliser le cache, ou non en lecture. Ceci permet de dissocier la partie alimentation du cache de son utilisation. En effet on peut imaginer désactiver la lecture du cache le temps que ce dernier se reconstruise ou en cas d’anomalie de cache détectée, et cela se traduit par l’utilisation de ce paramètre.

* + 1. FAQ & astuces

**Est-ce que le cache utilise la géolocalisation ?**

Les éléments de mobilités mis en cache (principalement issus des endpoints Tomp-api) le sont avec des coordonnées géographique. Lors des requêtes au cache on peut ainsi spécifier une géolocalisation et un rayon et c’est le cache qui au travers de sa fonctionnalité de géolocalisation va sélectionner les éléments correspondant au filtre géographique.

* 1. Microservices gateway
     1. Description

La gateway comporte 7 microservices permettant de traiter différentes données.

Le microservice **adapters** contient toutes les actions supportées par la gateway. Il est en charge de la transformation des requêtes vers le standard attendu par le partenaire.

Plus de détails sous : [https://github.com/fabmob/mcm-gateway/blob/0.7.2/adapters/README.md](https://github.com/fabmob/mcm-gateway/blob/main/adapters/README.md)

Le microservice **api** est le point d’entrée de la Gateway. C’est le seul exposé aux partenaires et regroupe tous les endpoints actuellement supportés.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/api/README.md](https://github.com/fabmob/mcm-gateway/blob/main/api/README.md)

Le module **cache-manager** est un microservice fournissant les opérations de gestion du cache. Les partenaires externes n'ont pas accès à ces services.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/cache-manager/README.md](https://github.com/fabmob/mcm-gateway/blob/main/cache-manager/README.md)

Le module **common-api** n’est pas un microservice en soit mais fait office de librairie commune à tous les autres microservices. Il centralise des dictionnaires de constantes et méthodes communes et réutilisables.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/common-api/README.md](https://github.com/fabmob/mcm-gateway/blob/main/common-api/README.md)

Le module **data-api** est un microservice d'administration des données de la Gateway. En effet il fournit les endpoints des opérations de CRUD (GET, POST, PUT, PATCH, DELETE) de l'ensemble des tables de la base de données.

Seul un admin de la Gateway a accès à cette Data-Api, elle n'est pas exposée à l'extérieur. Via les différentes opérations de CRUD, ce microservice permet d'enregistrer en base et éditer les informations de meta-data, data-mapping et configuration des différents partenaires.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/data-api/README.md](https://github.com/fabmob/mcm-gateway/blob/main/data-api/README.md)

Le module **database** (encadré orange ci-dessus) est le module de configuration de la base de données de la Gateway.  
Il contient les classes de modèles, repository et services associés aux tables en base. Ce n'est pas un microservice REST. La base de données "gateway" est divisée en deux schémas : ‘msp’ et ‘configuration’

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/database/README.md](https://github.com/fabmob/mcm-gateway/blob/main/database/README.md)

Afin de pouvoir mocker les différentes réponses des partenaires extérieurs, le microservice de mocks **mock-api** permet de simuler un MSP ou un MaaS en répondant avec un code retour et un body (facultatif) au choix.  
Il s'agit d'un module additionnel, indépendant de tous les autres et n’est pas livré en production.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/mock-api/README.md](https://github.com/fabmob/mcm-gateway/blob/main/mock-api/README.md)

Le module **request-relay** exécute les requêtes auprès des partenaires interrogés. Il est le seul microservice qui réalise des appels vers l'extérieur de la Gateway.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/request-relay/README.md](https://github.com/fabmob/mcm-gateway/blob/main/request-relay/README.md)

Le module **routing** est un microservice d'aiguillage des requêtes des partenaires.  
Le module api transmet au routing les paramètres de la requête afin qu'il la redirige vers l'adaptateur approprié pour la conversion de la requête au format (standard ou propriétaire) du partenaire appelé.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/routing-api/README.md](https://github.com/fabmob/mcm-gateway/blob/main/routing-api/README.md)

* + - 1. Access

Les microservices ne sont pas accessibles directement avec leurs urls lorsque Gravitee est déployé. Il faut passer par l’APIM, qui filtre les appels et authentifie l’appelant grâce au header ‘x-api-key’.

Il existe un moyen d’accéder directement aux urls des microservices grâce au portforwarding. Seulement accessible par un admin.

Liens pour appeler le **micro-service API** par environnement :

Local : [http://localhost:8080/api/v1/](http://localhost:8080/api/v1/swagger-ui/index.html#/)

Develop : [https://api-develop.preview-gw.{domaineGateway}/api/v1/](https://api-develop.preview-gw.moncomptemobilite.fr/api/v1/swagger-ui/index.html#/)

Via l’APIM Preprod : [https://apim-gateway.preprod-gw.{domaineGateway}/api/v1/](https://apim-gateway.preprod-gw.moncomptemobilite.fr/api/v1/)

Via l’APIM Prod : [https://apim-gateway.gw.{domaineGateway}/api/v1/](https://apim-gateway.gw.moncomptemobilite.fr/api/v1/)

Liens pour appeler le **micro-service data-api** par environnement :

Local : <http://localhost:8081/dataapi/v1/>

Develop : [https://dataapi-develop.preview-gw.{domaineGateway}/dataapi/v1/](https://dataapi-develop.preview-gw.moncomptemobilite.fr/dataapi/v1/)

Via l’APIM Preprod : [https://apim-gateway.preprod-gw.{domaineGateway}/dataapi/v1/](https://apim-gateway.preprod-gw.moncomptemobilite.fr/dataapi/v1/)

Via l’APIM Prod : [https://apim-gateway.gw.{domaineGateway}/dataapi/v1/](https://apim-gateway.gw.moncomptemobilite.fr/dataapi/v1/)

Liens pour appeler les autres micro-services avec ou sans l’APIM, par environnement :

**Local**

Data-api : <http://localhost:8081/dataapi/v1/>

Autres : [http://localhost:{port}/v1/](http://localhost:%7bport%7d/v1/)

{port} :

* cache-manager : 8086
* mock-api : 8084
* default-adapter : 8090
* routing-api : 8082
* request-relay : 8085

**Autres environnements et autres microservices sans APIM**

https://{microservice}-{branche}.{environnement}.{domaineGateway}/{api/dataapi/Rien}/v1/

**Autres environnements et autres microservices avec APIM**

Via portforwarding uniquement

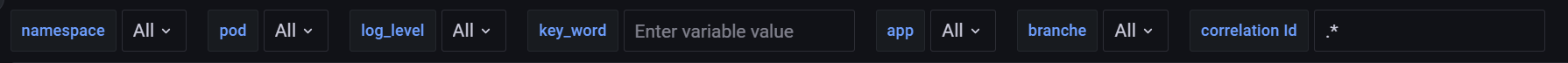
Pour accéder à leur swagger rajouter à la fin de l’url : swagger-ui/index.html#/

**Note** : Il est possible d’accéder au swagger via Gravitee, mais celui-ci sera inutilisable. Il ne sert qu’à la documentation et fourni des exemples d’utilisation.

* + - 1. Logs

Il est possible de retrouver les logs de tous les microservices indexés par Grafana loki. Les logs dans l’APIM Gravitee sont quant à elles, indexés par elasticsearch.

Une fois connecté, il suffit de cliquer sur la loupe située à gauche de l’écran et rechercher ‘logs’. Cliquer sur le seul résultat qui s’affiche.

Il est possible de filtrer les résultats en fonction d’un pod, d’une branche et notamment le correlation Id.  


Ce dernier est particulièrement efficace quand il s’agît de retrouver tous les logs, de tous les microservices liés à **un** appel.



Pour retrouver toute la trace d’un appel particulier, il suffit de récupérer le correlation Id (en jaune) et de l’ajouter au filtre

* + - 1. Backups

Les microservices sont stateless, il n’y a pas de backup nécessaire.

* + 1. Opérations de Maintenance en Conditions opérationnelles

**Rajouter un partenaire en base**

Seul un admin Gateway peut rajouter un partenaire en base. Pour cela, il faut contacter le service data-api via l’APIM (Gravitee) en utilisant une application permettant de tester des API (Postman), en fournissant un header nommé X-API-KEY correspondant à la clé secrète d’un admin, et un body avec toutes les informations nécessaires. Il est possible d’avoir un exemple des informations demandés via le swagger.

Une fois le partenaire ajouté, il faut créer, **dans cet ordre et pour chaque endpoint supporté**, l’action correspondante, créer le call qui permettra de connaitre l’url qui sera appelé en fonction de l’action, et créer le standard suivi par ce partenaire.

**Créer un adapter en base de données (nécessaire quand la bdd vient d’être créée)**

Chaque standard doit pointer vers un adapter. Cela permet à la Gateway de savoir quel adapter utiliser pour transformer le standard en question.

Lorsqu’il s’agit de format standard (Covoiturage, TOMP, GBFS, etc…), il faut pointer ce standard vers un adapter dont le nom est ‘default-adapter’. Pour des formats non-standards, faire pointer la relation ‘standard’ vers un adapter dont le nom correspondra au micro-service, implémenté spécifiquement pour ce format, dans la gateway ; par exemple ‘custom-adapter’.

**Mettre à jour un standard**

Un partenaire pouvant suivre un ou plusieurs standards, ou tout du moins, plusieurs versions d’un même standard, il peut être nécessaire de mettre à jour la version du standard suivi par le partenaire. Cela indiquera à la Gateway sous quel format et sous quelle version il doit transformer la réponse.

Il faudra dans un premier temps qu’un admin ajoute en base ce nouveau standard via data-api avec l’attribut ‘isActive’ à *true*. Dans un second temps, mettre à jour l’ancien standard avec ce même attribut à *false.*

Il faudra créer autant de standard que d’actions/endpoints supportés par le partenaire. En effet, la gateway est conçue telle que toute nouvelle version d’un standard soit retro-compatible. Cela permet, par exemple, à un ou plusieurs SI extérieurs d’être évolutifs et ne dépendent pas l’un de l’autre.

Il faudra créer / mettre à jour les champs qui diffèrent en insérant en base des ‘dataMappers’. S’il s’agit du standard covoiturage, il n’y a pas besoin de faire de transformation car la gateway ‘suis’ ce standard.

* + 1. FAQ & astuces

**Que faire en cas d’erreur de la base données ?**

Seul data-api est en lien avec la base de données

Vérifier log in / mdp utilisé par ce module

Vérifier les logs au lancement de data-api (logs de schéma de la bdd)

Si le login/mdp sont correctes mais que l’applicatif ne fonctionne toujours pas, tester la connexion à la base de données via un autre client (psql, DBeaver, etc…)

**Est-ce qu’il suffit uniquement de mettre à jour les bases de datamapping ?**

La Gateway a été conçue pour que les versions des standards soient retro-compatibles. Cependant, cette fonctionnalité n’a pas été testée.

**A-t-on besoin de créer les adapters de requête et réponses (un exemple : custom-adapter ?). Ou le défaut suffit ? Dans quels cas ?**

L’adapter par défaut (default-adapter) permet de transformer les standards en d’autres standards. Actuellement il supporte le standard covoiturage et partiellement TOMP.

Dans le cas où un MSP ne suivrait aucun standard, alors il faudrait implémenter un ‘custom-adapter’ pour chaque MSP ayant un format propriétaire. Il faudra également penser à insérer dans la base de données un adapter et un ‘standard’ dont la clé primaire sera constituée de l’id de l’adapter.

**Comment le cache est-il mis-à-jour ?**

Nous importons dans la pipeline un outil extérieur appelé Dkron. Cet outil permet d’automatiser le rafraichissement du cache en appelant certains de nos endpoints tous les certains temps. Lorsque le cache est activé, la gateway va consulter redis, récupérer et renvoyer la même réponse à la requête envoyée il y a 5 minutes (par exemple). Notamment sur l’endpoint /Around-me qui récupère la liste des véhicules disponible autour d’une position donnée.

Redis gère automatiquement la durée de validité d’une donnée cachée (hardTTL). La durée de validité softTTL de chaque attribut est enregistré en base de données et fourni à Dkron. Lorsque la donnée arrive à la limite du softTTL, fourni à Dkron, celui-ci va tenter de déclencher un appel à la gateway, afin que celle-ci fasse un appel réel vers les MSPs concernés. Si l’appel est réussi, tant mieux, l’utilisateur recevra de la donnée cachée récemment. Sinon, nous renvoyons quand même les données cachées si nous estimons quelle est encore valide. Le hardTTL, quant à lui, est fourni a Redis et supprime automatiquement les données ayant dépassé la durée de validité maximale.

Lorsque la donnée n’est pas disponible en cache lors d’un appel client, alors le gateway fera un appel réel vers le/les msp concernés par cet endpoint.

Autre point, le cache se rafraichit automatiquement lorsque nous ajoutons/modifions un nouveau partenaire en base.

La valeur du **statut** du cache se met à jour lorsque nous la modifions via le cache-manager ou bien la base de données.

* 1. Graviteeio
     1. Description

Gravitee est notre API Management permettant d’exposer notre API au monde extérieur. Il filtre les appels et authentifie les utilisateurs de la gateway.

Toutes les informations nécessaires à la création et à la configuration d’une API, à la création de plans (subscriptions), à l’intégration d’un partenaire et à la mise en place de Gravitee sont disponibles dans ce document :

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/docs/Gravitee-Doc-configuration-APIM-console-MCM\_Std\_MaaS\_GW\_v1.5.pdf](https://github.com/fabmob/mcm-gateway/blob/main/docs/Gravitee-Doc-configuration-APIM-console-MCM_Std_MaaS_GW_v1.5.pdf)

La configuration des Rôles et des permissions sont partiellement documentés en vue des préférences utilisées par le repreneur de la Gateway.

* + 1. Administration
       1. Access

Lien APIM Console (Administration) par environnement :

* Testing : [https://apim-console.testing-gw.{domaineGateway}/#!/login](https://apim-console.testing-gw.moncomptemobilite.fr/#!/login)
* Preprod : [https://apim-console.preprod-gw.{domaineGateway}/#!/login](https://apim-console.preprod-gw.moncomptemobilite.fr/#!/login)
* Prod : ?

Lien APIM Portal (Client) par environnement :

* Testing : [https://apim-portal.testing-gw.{domaineGateway}/user/login](https://apim-portal.testing-gw.moncomptemobilite.fr/user/login?)
* Preprod : [https://apim-portal.preprod-gw.{domaineGateway}/user/login](https://apim-portal.preprod-gw.moncomptemobilite.fr/user/login)
* Prod ?
  + - 1. Logs

Les logs techniques gravitee sont répartis principalement entre les pods gateway et api des pods apim3x. Les logs fonctionnels (requêtes) sont consultables depuis l’administration gravitee, une fois l’api sélectionné puis être allé sur le menu Analytics, Logs.

* + - 1. Backups

Le backup est celui réalisé sur le serveur de base de données au niveau du service PaaS de postgresql.

* + 1. Opérations de Maintenance en Conditions opérationnelles

Modifier timeout :

* Afin de modifier le timeout d’une api de l’api management, aller sur la console gravitee, sélectionner l’api, Proxy> endpoints > Edit Groups> Configuration > Read timeout.

Consulter les logs :

* Afin de consulter les logs d’une api de l’api management, aller sur la console gravitee, sélectionner l’api, Analytics > logs
  + 1. FAQ & astuces
  1. Elastic stack : elasticsearch, kibana, apm
     1. Description

Pour comprendre comment fonctionne l’APM, configurer des agents, mettre en place l’APM et visualiser les performances de la Gateway, vous pouvez vous référer à ce lien :

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/README\_APM.md](https://github.com/fabmob/mcm-gateway/blob/main/README_APM.md)

* + 1. Administration
       1. Access

Pour accéder à Kibana qui permet notamment de visualiser les résultats de l’APM il faut s’y connecter à l’aide de port forwarding.

* + - 1. Logs

Logs à consulter au niveau des pods.

* + - 1. Backups

Pas de backup mis en place. Un backup des disques utilisés par les storageClass serait à prévoir afin de sauvegarder l’état des bases elasticsearch.

* + 1. Opérations de Maintenance en Conditions opérationnelles
       1. Nettoyage / merge des index

Gravitee créant des index elasticsearch journaliers, quelques opérations de maintenance sont à prévoir afin d’améliorer les performances du search et limiter le nombre de shards. Plus le nombre de shards sollicités lors d’une recherche, plus on prend le risque de saturer elasticsearch et le nombre de sous requêtes qu’il pourra traiter.

Il est donc utile de réaliser quelques opérations de nettoyages ou optimisation des index. Mieux vaut éviter de solliciter plusieurs vingtaines ou centaines de shards pour une même requête utilisateur (dans notre cas ce sont les requêtes des KPI).

Voici quelques actions permettant d’optimiser les index :

* Suppression des logs & request des sondes ayant au moins 1 semaine
* merge des index journaliers en 1 par mois pour les index:

- gravitee-monitor-\*

- gravitee-health-\*

- gravitee-log-\*

- gravitee-request-\*

* une fois mergés suppression des logs journaliers

POST /gravitee-request-\*/\_delete\_by\_query

{

"query": {

"bool": {

"filter": [

{

"bool": {

"should": [

{

"match\_phrase": {

"path": "/database"

}

}

],

"minimum\_should\_match": 1

}

},

{

"range": {

"@timestamp": {

"lte": "now-1d/d"

}

}

}

]

}

}

}

POST /gravitee-request-\*/\_delete\_by\_query

{

"query": {

"bool": {

"filter": [

{

"bool": {

"should": [

{

"match\_phrase": {

"path": "/api"

}

}

],

"minimum\_should\_match": 1

}

},

{

"range": {

"@timestamp": {

"lte": "now-1d/d"

}

}

}

]

}

}

}

POST /gravitee-log-\*/\_delete\_by\_query

{

"query": {

"bool": {

"filter": [

{

"bool": {

"should": [

{

"match\_phrase": {

"client-request.uri.keyword": "/api/v1/api"

}

}

],

"minimum\_should\_match": 1

}

},

{

"range": {

"@timestamp": {

"lte": "now-1d/d"

}

}

}

]

}

}

}

POST /gravitee-request-\*/\_forcemerge

POST \_reindex

{

"source": {

"index": "gravitee-log-2022.11.\*"

},

"dest": {

"index": "gravitee-log-2022.11"

}

}

DELETE gravitee-log-2022.11.\*

POST \_reindex

{

"source": {

"index": "gravitee-log-2022.12.\*"

},

"dest": {

"index": "gravitee-log-2022.12"

}

}

DELETE gravitee-log-2022.12.\*

POST \_reindex

{

"source": {

"index": "gravitee-log-2023.01.\*"

},

"dest": {

"index": "gravitee-log-2023.01"

}

}

DELETE gravitee-log-2023.01.\*

POST \_reindex

{

"source": {

"index": "gravitee-log-2023.02.\*"

},

"dest": {

"index": "gravitee-log-2023.02"

}

}

DELETE gravitee-log-2023.02.0\*,gravitee-log-2023.02.1\*,gravitee-log-2023.02.20,gravitee-log-2023.02.21,gravitee-log-2023.02.22,gravitee-log-2023.02.23,gravitee-log-2023.02.24,gravitee-log-2023.02.25,gravitee-log-2023.02.26

POST \_reindex

{

"source": {

"index": "gravitee-health-2022.11.\*"

},

"dest": {

"index": "gravitee-health-2022.11"

}

}

POST \_reindex

{

"source": {

"index": "gravitee-health-2022.12.\*"

},

"dest": {

"index": "gravitee-health-2022.12"

}

}

DELETE gravitee-health-2022.12.\*

POST \_reindex

{

"source": {

"index": "gravitee-health-2023.01.\*"

},

"dest": {

"index": "gravitee-health-2023.01"

}

}

DELETE gravitee-health-2023.01.\*

POST \_reindex

{

"source": {

"index": "gravitee-health-2023.02.\*"

},

"dest": {

"index": "gravitee-health-2023.02"

}

}

DELETE gravitee-health-2023.02.0\*,gravitee-health-2023.02.1\*,gravitee-health-2023.02.20,gravitee-health-2023.02.21,gravitee-health-2023.02.22,gravitee-health-2023.02.23,gravitee-health-2023.02.24,gravitee-health-2023.02.25,gravitee-health-2023.02.26

POST \_reindex

{

"source": {

"index": "gravitee-monitor-2022.11.\*"

},

"dest": {

"index": "gravitee-monitor-2022.11"

}

}

POST \_reindex

{

"source": {

"index": "gravitee-monitor-2022.12.\*"

},

"dest": {

"index": "gravitee-monitor-2022.12"

}

}

DELETE gravitee-monitor-2022.12.\*

POST \_reindex

{

"source": {

"index": "gravitee-monitor-2023.01.\*"

},

"dest": {

"index": "gravitee-monitor-2023.01"

}

}

DELETE gravitee-monitor-2023.01.\*

POST \_reindex

{

"source": {

"index": "gravitee-monitor-2023.02.\*"

},

"dest": {

"index": "gravitee-monitor-2023.02"

}

}

DELETE gravitee-monitor-2023.02.0\*,gravitee-monitor-2023.02.1\*,gravitee-monitor-2023.02.20,gravitee-monitor-2023.02.21,gravitee-monitor-2023.02.22,gravitee-monitor-2023.02.23,gravitee-monitor-2023.02.24,gravitee-monitor-2023.02.25,gravitee-monitor-2023.02.26

POST \_reindex

{

"source": {

"index": "gravitee-request-2022.11.\*"

},

"dest": {

"index": "gravitee-request-2022.11"

}

}

DELETE gravitee-request-2022.11.\*

POST \_reindex

{

"source": {

"index": "gravitee-request-2022.12.\*"

},

"dest": {

"index": "gravitee-request-2022.12"

}

}

DELETE gravitee-request-2022.12.\*

POST \_reindex

{

"source": {

"index": "gravitee-request-2023.01.\*"

},

"dest": {

"index": "gravitee-request-2023.01"

}

}

DELETE gravitee-request-2023.01.\*

POST \_reindex

{

"source": {

"index": "gravitee-request-2023.02.\*"

},

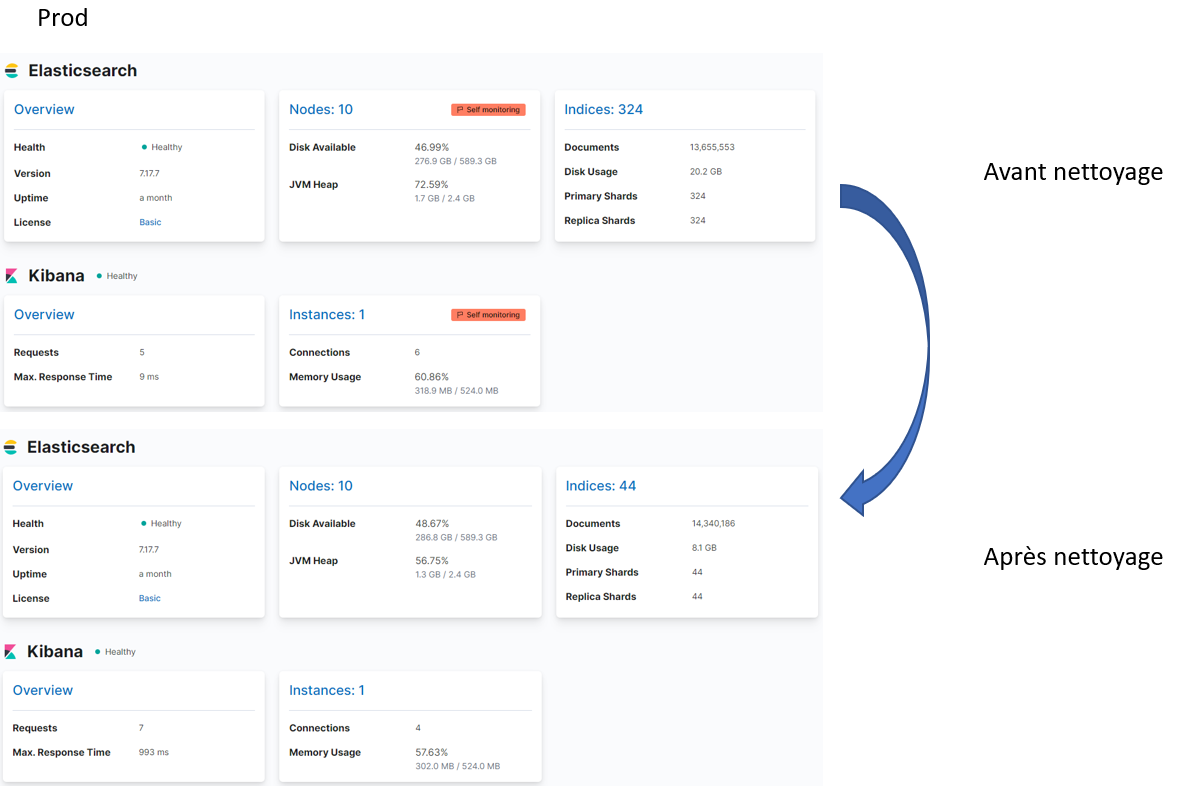
"dest": {

"index": "gravitee-request-2023.02"

}

}

DELETE gravitee-request-2023.02.0\*,gravitee-request-2023.02.1\*,gravitee-request-2023.02.20,gravitee-request-2023.02.21,gravitee-request-2023.02.22,gravitee-request-2023.02.23,gravitee-request-2023.02.24,gravitee-request-2023.02.25,gravitee-request-2023.02.26



* + 1. FAQ & astuces
  1. Grafana : monitoring & kpi
     1. Description

Afin de surveiller les performances de la gateway et surveiller son usage, un board grafana a été créé pour pouvoir remonter les différentes statistiques de l'utilisation de la gateway.

[https://github.com/fabmob/mcm-gateway/blob/0.7.2/kpi/README\_KPI.md](https://github.com/fabmob/mcm-gateway/blob/main/kpi/README_KPI.md)

* + 1. Administration
       1. Access

Connexion via navigateur web soit par login/mot de passe soit via SSO (si paramétré).

* + - 1. Logs

Pour accéder aux logs des microservices se reporter à la section **4.1.1.2**

* + - 1. Backups

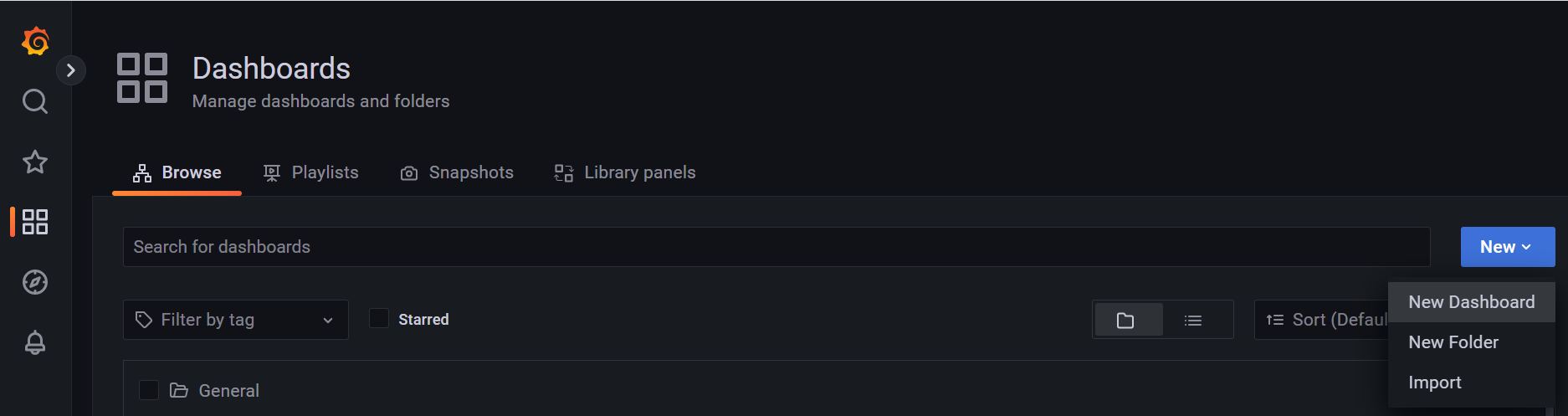
Il existe des backups des dashboards des kpi dans le dossier ‘kpi’ de la Gateway :

[https://github.com/fabmob/mcm-gateway/tree/main/kpi](https://github.com/fabmob/mcm-gateway/tree/release-0.7.2/kpi)

* + 1. Opérations de Maintenance en Conditions opérationnelles

**Pour importer le dashboard des kpi dans grafana :**

1. Cliquer sur l’onglet dashboard
2. Cliquer sur le bouton ‘New’ -> Import



* + 1. FAQ & astuces