LOGS APIs

**Architecture technique :**

Schema

API Talend -> Apache ActiveMQ -> middleware -> (Logstash -> ElasticSearch -> Kibana)ELK

Définir chaque modules de l’architecture

A message broker enables applications to communicate with each other to share information. Apache ActiveMQ is a popular open-source, multi-protocol, Java message broker. It supports industry-standard protocols such as AMQP (Advanced Message Queuing Protocol) to integrate multi-platform applications, STOMP (Simple Text Orientated Messaging Protocol) for exchanging messages between web applications over websockets, and MQTT (Message Queuing Telemetry Transport) to manage IoT devices. In this article, you will learn how to install Apache ActiveMQ on Ubuntu 20.04 server.

**Infrastructure proposée**

1. **API Talend** :
   * Capture les informations de log (requêtes, timestamp, utilisateur, etc.).
   * Envoie les logs formatés en JSON à une queue ActiveMQ de manière asynchrone.
2. **ActiveMQ** :
   * Reçoit et stocke les messages de log provenant de l'API Talend.
   * Assure la persistance et la fiabilité des messages en cas de panne.
3. **Script de traitement des logs** :
   * Un script (par exemple, en Python ou Java) est développé pour consommer les messages de la queue ActiveMQ.
   * Le script traite les logs et les envoie à Logstash.
4. **Logstash** :
   * Reçoit les logs du script de traitement, les filtre, et les envoie à Elasticsearch.
5. **Kibana** :
   * Utilisé pour visualiser et analyser les logs.

**Avantages de l'utilisation d'ActiveMQ :**

Couplage lâche : ActiveMQ permet de découpler la génération de logs dans l’API de leur traitement et de leur stockage. Cela signifie que ton application Talend n'a pas besoin d'attendre que les logs soient stockés avant de continuer à traiter les requêtes.

Fiabilité : ActiveMQ offre des fonctionnalités de persistance et de redondance, ce qui peut être utile pour s'assurer que les logs ne sont pas perdus. ACID ?

Scalabilité : Il est possible de traiter les messages de la queue de manière distribuée, ce qui peut aider à gérer un volume important de logs.

**Prérequis** :

A compléter later.

Server Linux Ubuntu avec Java et Docker installés.

**Mise en place POC and POV**

**Partie I. Installation de Apache Active MQ**

1. Install Java

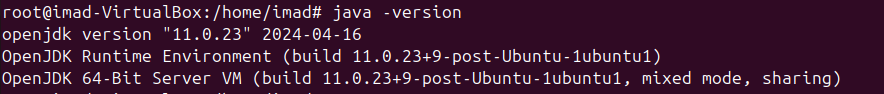
Update the system.

root@imad-VirtualBox:/home/imad# sudo apt update

Apache ActiveMQ requires Java to run. Install Java.

root@imad-VirtualBox:/home/imad# sudo apt install openjdk-11-jre –y

Verify the Java installation.



2. Install and Configure Apache ActiveMQ

Download ActiveMQ from the Apache. To find the latest version of this software, you can visit the download page : <https://archive.apache.org/dist/activemq/>

$ wget https://archive.apache.org/dist/activemq/5.16.3/apache-activemq-5.16.3-bin.tar.gz

Extract the downloaded file.

$ sudo tar -xvzf apache-activemq-5.16.3-bin.tar.gz

Create a directory named /opt/activemq.

$ sudo mkdir /opt/activemq

Move the extracted files to the /opt/activemq directory.

$ sudo mv apache-activemq-5.16.3/\* /opt/activemq

Create a group account activemq to run Apache ActiveMQ.

$ sudo addgroup --quiet --system activemq

Créer l'utilisateur activemq-talend dans le groupe activemq avec un mot de passe

$ sudo adduser --quiet --system --ingroup activemq --no-create-home activemq-talend

Définir le mot de passe pour l'utilisateur activemq-talend

$ echo "activemq-talend:\*\*\*\*\*\*\*\* | sudo chpasswd

Change the permissions of the /opt/activemq directory.

$ sudo chown -R activemq-talend:activemq /opt/activemq

This section above is ignored for this time, maybe install it later

Create an ActiveMQ systemd service file to control the Apache ActiveMQ service.

$ sudo nano /etc/systemd/system/activemq.service

Add the below code into the file. Save and close the file.

[Unit]

Description=Apache ActiveMQ

After=network.target

[Service]

Type=forking

User=activemq-talend

Group=activemq

ExecStart=/opt/activemq/bin/activemq start

ExecStop=/opt/activemq/bin/activemq stop

[Install]

WantedBy=multi-user.target

Edit the jetty.xml configuration file to change the host.

$ sudo nano /opt/activemq/conf/jetty.xml

Find the line below:

<property name="host" value="127.0.0.1"/>

Change it to:

<property name="host" value="0.0.0.0"/>

Save and close the file.

Reload the system daemon.

$ sudo systemctl daemon-reload

Start the Apache ActiveMQ service.

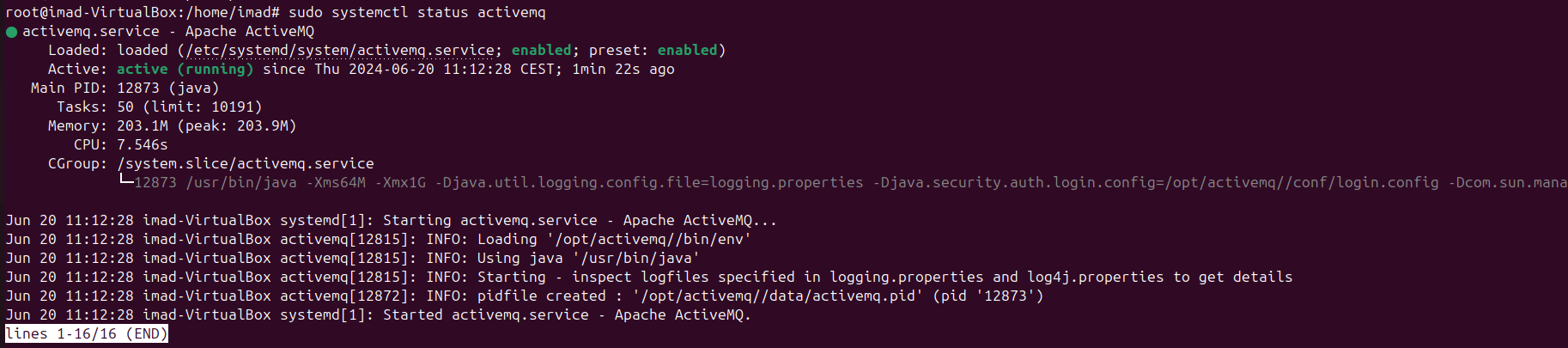
$ sudo systemctl start activemq

Enable the Apache ActiveMQ service to run at system startup.

$ sudo systemctl enable activemq

Verify the status of the service.

$ sudo systemctl status activemq



Restart ActiveMQ service.

$ sudo systemctl restart activemq

3. Access Apache ActiveMQ Web Interface

Open your web browser and access the Apache ActiveMQ web UI using the URL <http://localhost:8161/admin>

Par défaut, le compte admin du mot de passe est admin avec le mot de passe admin. Ce compte admin a des permissions étendues.

Nous allons modifier le mot de passe du compte admin.

Mettre à jour le fichier users.properties

admin=<new password>

activemq-talend=T@lendAMQ!

Mettre à jour le fichier groups.properties

admins=admin

users=activemq-talend

Ensure the jetty-realm.properties file reflects the same information:

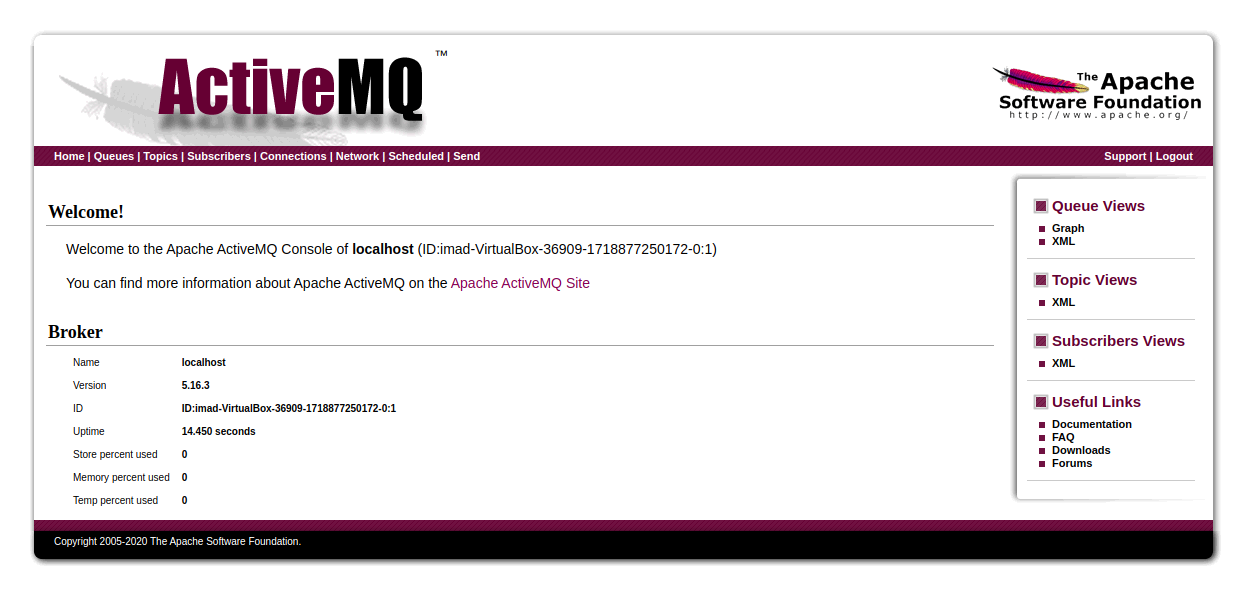
# username: password [,rolename ...]

admin: <new password>, admin

user: <new password>, user

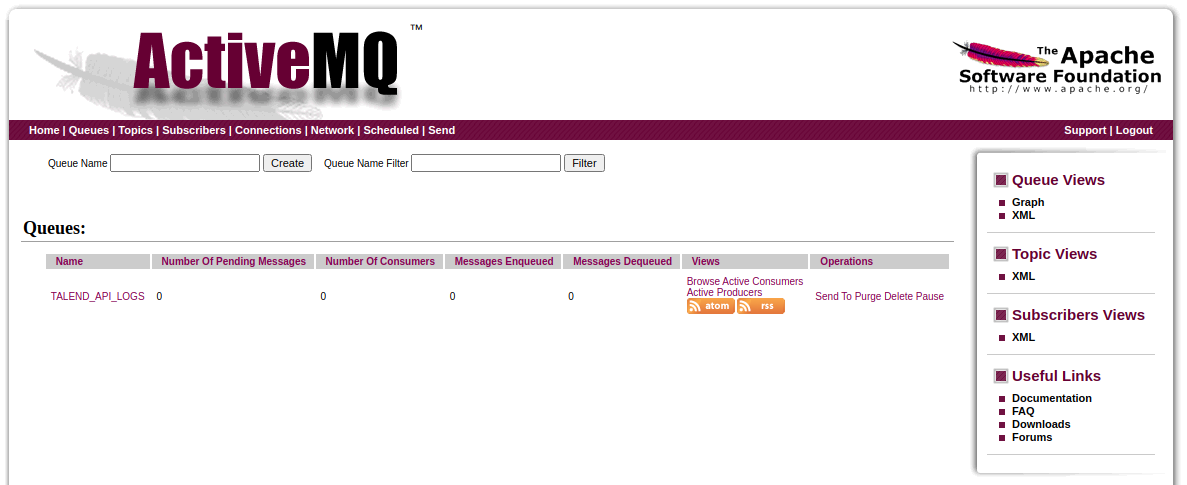
activemq-talend: T@lendAMQ!, user

On peut maintenant se connecter à ActiveMQ avec les comptes admins ou activemq-talend

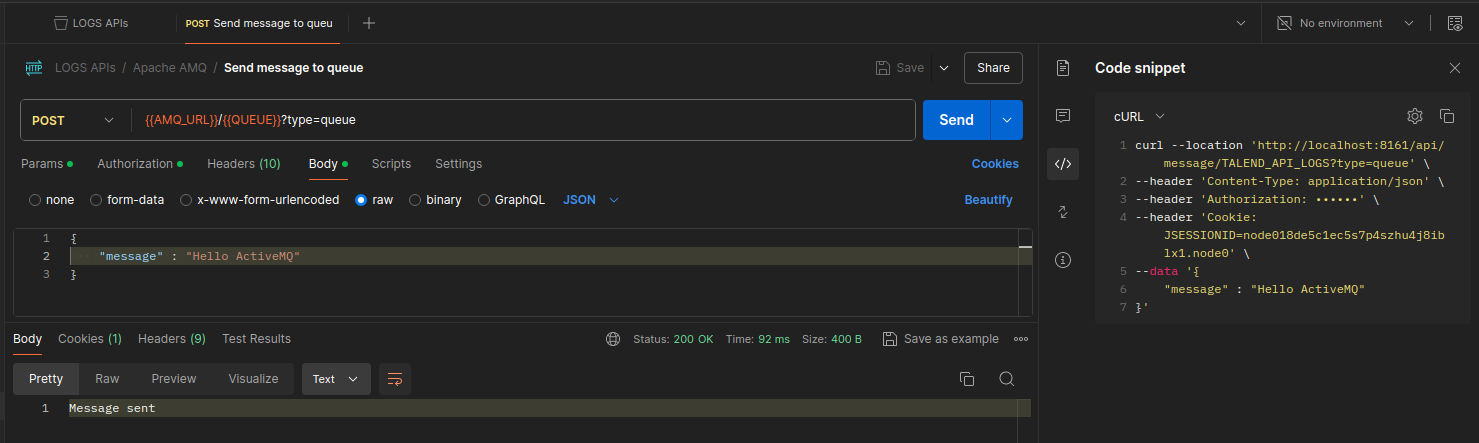


1. Création de la queue

Aller sur Queues et créer la queue TALEND\_API\_LOGS



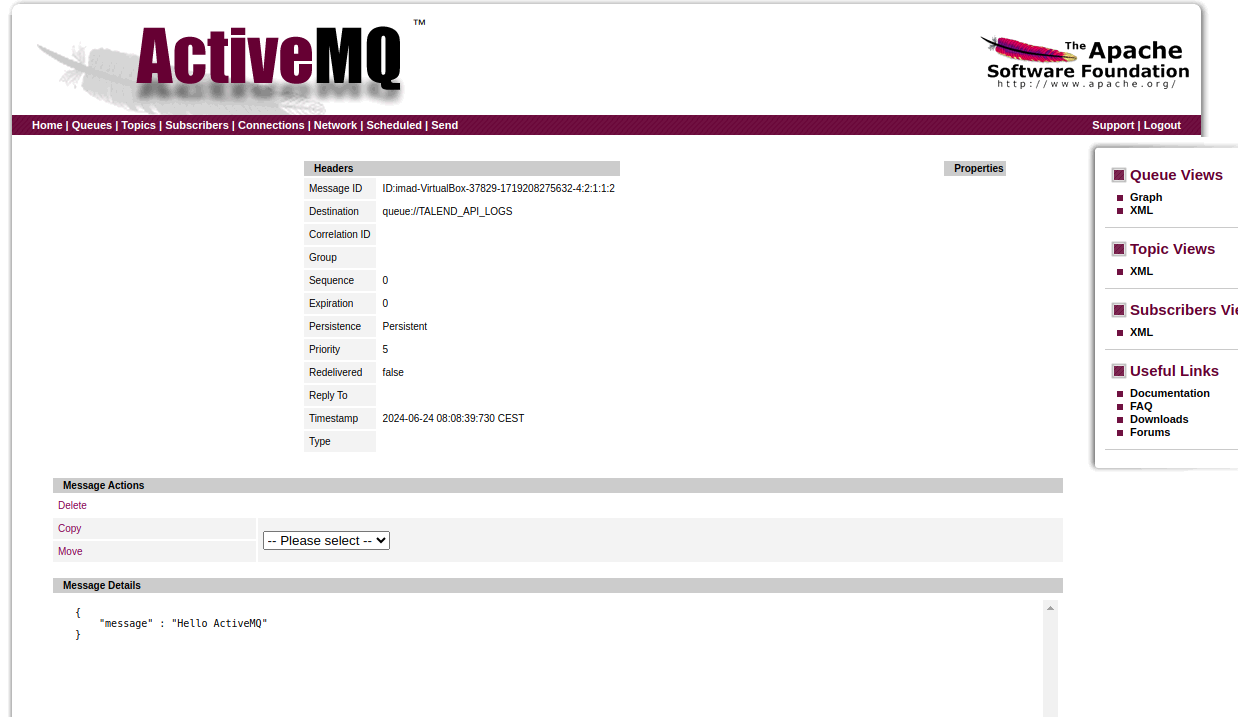
Tout est prêt ! Nous pouvons désormais envoyer des messages à notre queue de plusieurs façons comme JMS (Java Message Service), ou directement via un call api.



Suite à cet appel api, le message envoyé atterrit dans la queue et attend d’être consommé.



Cliquer dessus pour le consulter



**Partie II. Installation de la Stack ELK**

1. **Installation Docker**

**ELK Stack avec Docker**

ELK v7.15.2

pour lancer Elasticsearch, Logstash et Kibana en utilisant Docker, en prenant en compte le réseau et les volumes pour la persistance des données. Assurez-vous d'avoir Docker installé sur votre machine avant de commencer.

**Etape 1 : Installation de Docker**

sudo apt-get remove containerd.io

sudo apt-get update

sudo apt install docker.io

sudo apt install curl

sudo systemctl enable docker

sudo systemctl status / start docker

sudo docker –v

**Étape 2 : Création d'un réseau Docker**

Créez un réseau Docker pour permettre la communication entre les conteneurs Elasticsearch, Logstash et Kibana.

docker network create elk\_network

**Étape 3 : Démarrage d'Elasticsearch**

Lancez Elasticsearch en utilisant Docker avec un conteneur nommé "elasticsearch", en reliant le conteneur au réseau précédemment créé et en montant un volume pour persister les données.

docker run -d \

--name elasticsearch \

--net elk\_network \

-p 9200:9200 \

-p 9300:9300 \

-v es\_data:/usr/share/elasticsearch/data \

-e "discovery.type=single-node" \

docker.elastic.co/elasticsearch/elasticsearch:7.15.2

**Étape 4 : Démarrage de Logstash**

Lancez Logstash en utilisant Docker, en reliant le conteneur au réseau précédemment créé.

docker run -d \

--name logstash \

--net elk\_network \

-v logstash\_config:/usr/share/logstash/pipeline \

docker.elastic.co/logstash/logstash:7.15.2

**Étape 5 : Démarrage de Kibana**

Lancez Kibana en utilisant Docker, en reliant le conteneur au réseau précédemment créé.

docker run -d \

--name kibana \

--net elk\_network \

-p 5601:5601 \

docker.elastic.co/kibana/kibana:7.15.2

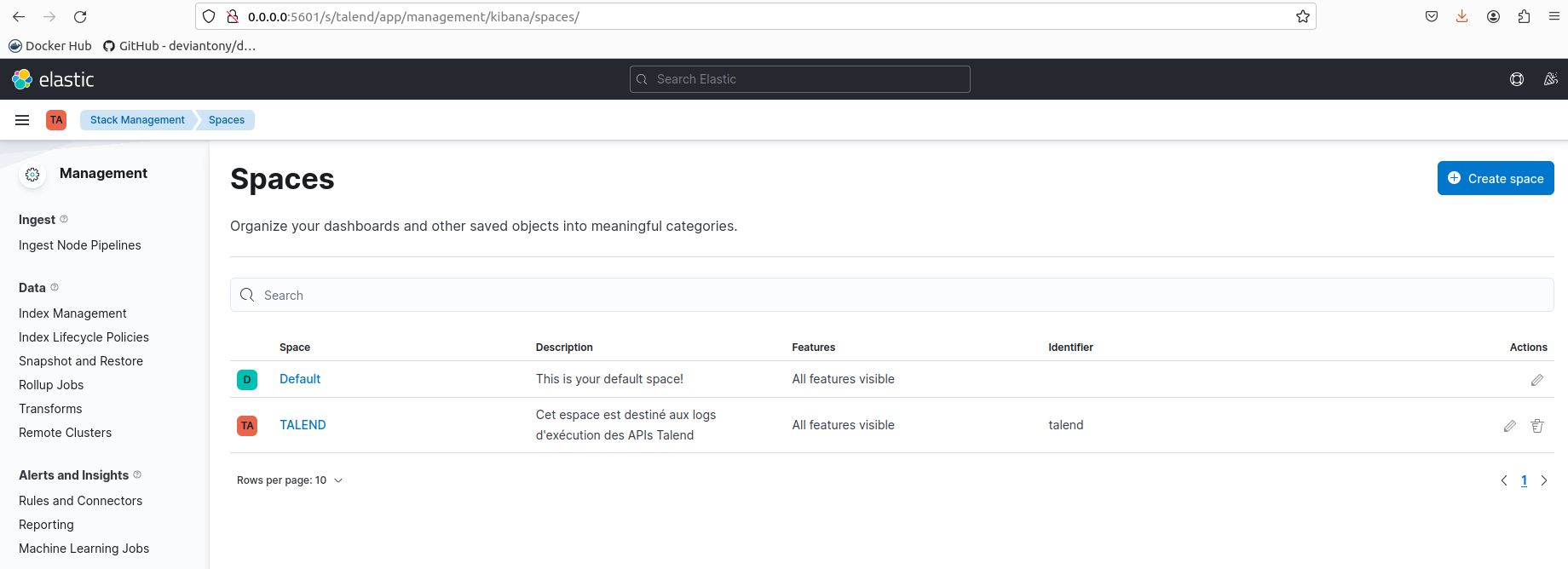
**Note:**

* Les versions des images Docker (7.x) peuvent être mises à jour en fonction de la version d'Elastic Stack que vous souhaitez utiliser.
* Les ports sont exposés selon les configurations par défaut, vous pouvez les ajuster si nécessaire.
* Les volumes (**es\_data** pour Elasticsearch, **logstash\_config** pour Logstash) sont utilisés pour persister les données et configurations respectivement.

**Configuration ELK**

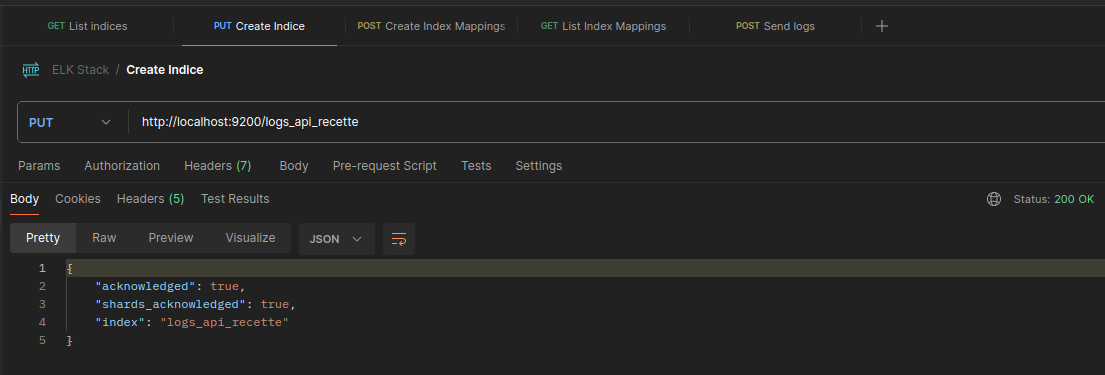
Créer un espace kibana pour le projet Talend (ça permettrai d’organiser les accès si on ajoute un nouveau projet par la suite).

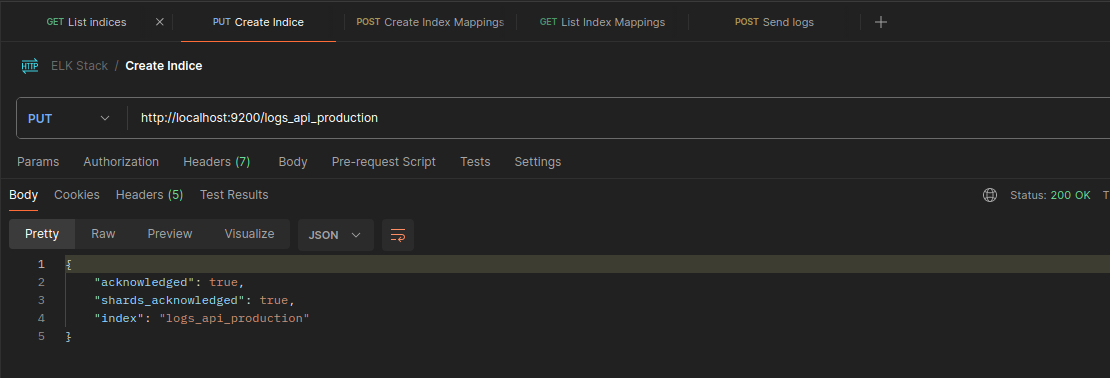
Cliquer sur Create space dans Space Management <http://0.0.0.0:5601/app/management/kibana/spaces>



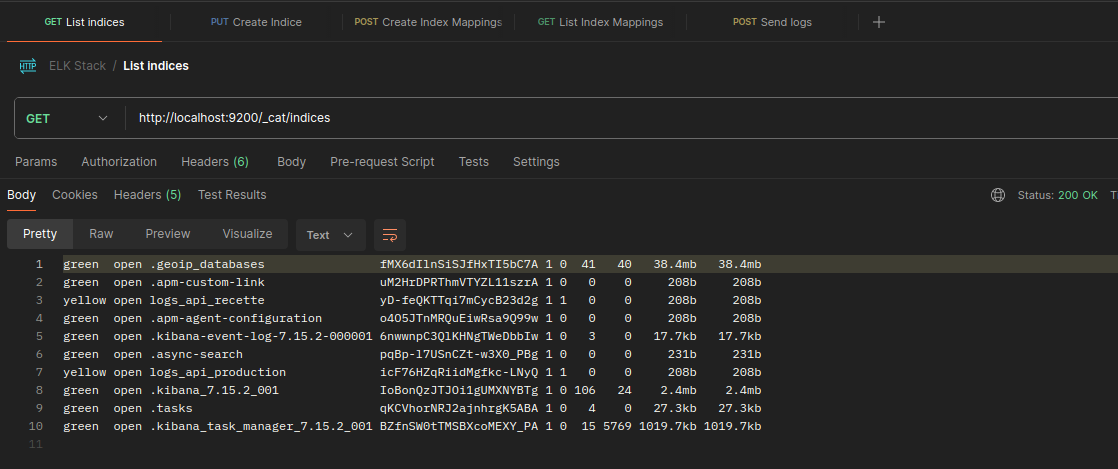
Se positionner dans l’espace Talend

Créer deux indexes elasticSearch logs\_api\_recette et logs\_api\_production





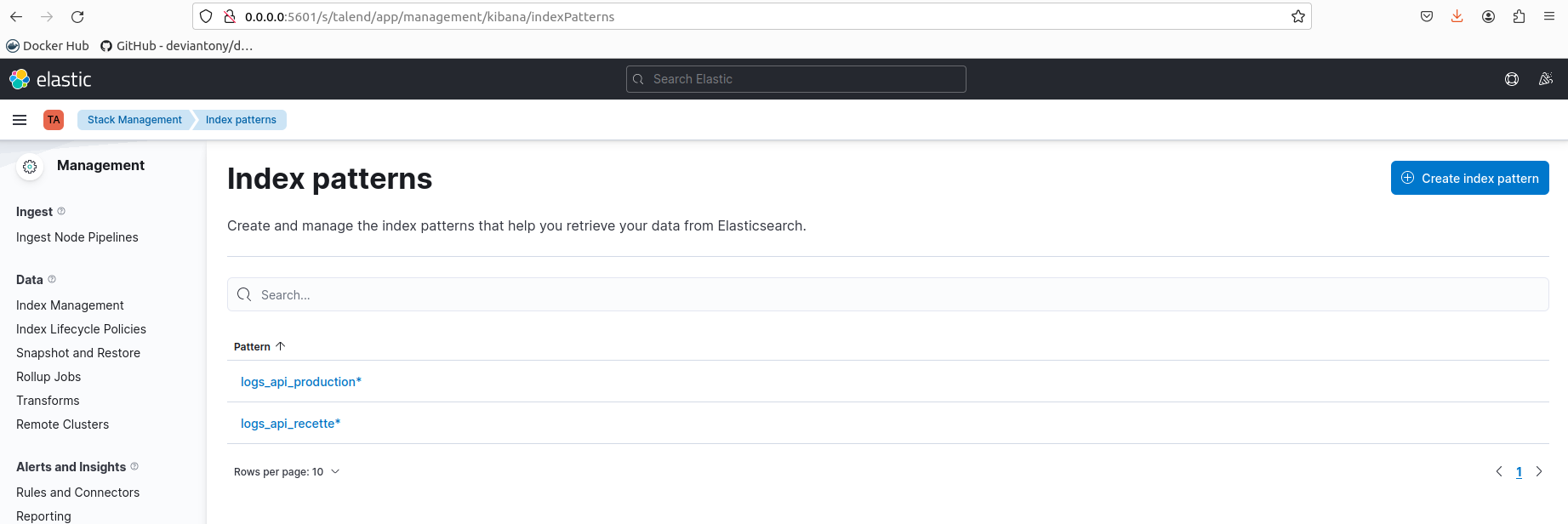
Vérifier



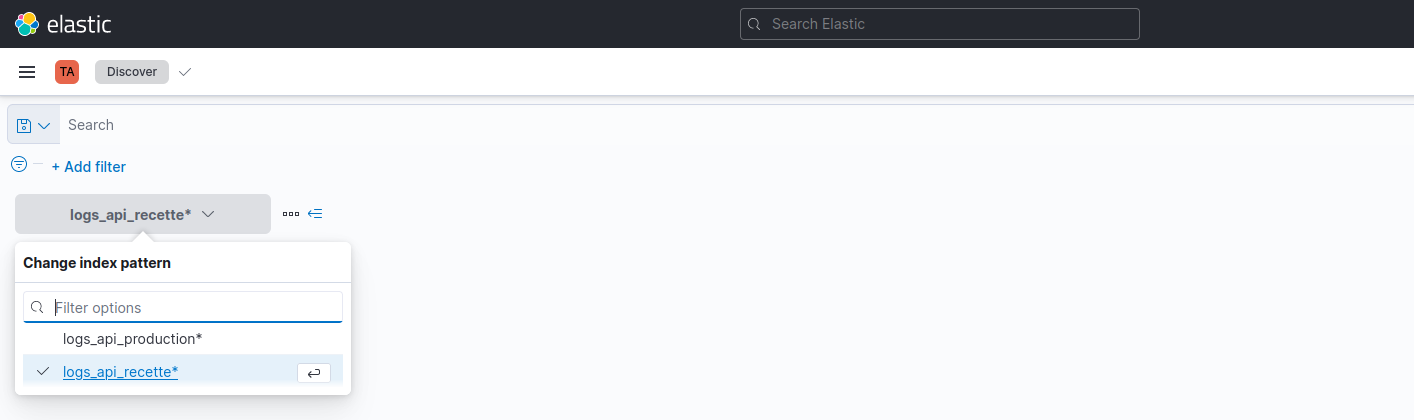
aller dans Index Patterns

<http://0.0.0.0:5601/app/management/kibana/indexPatterns>

Créer deux index patterns



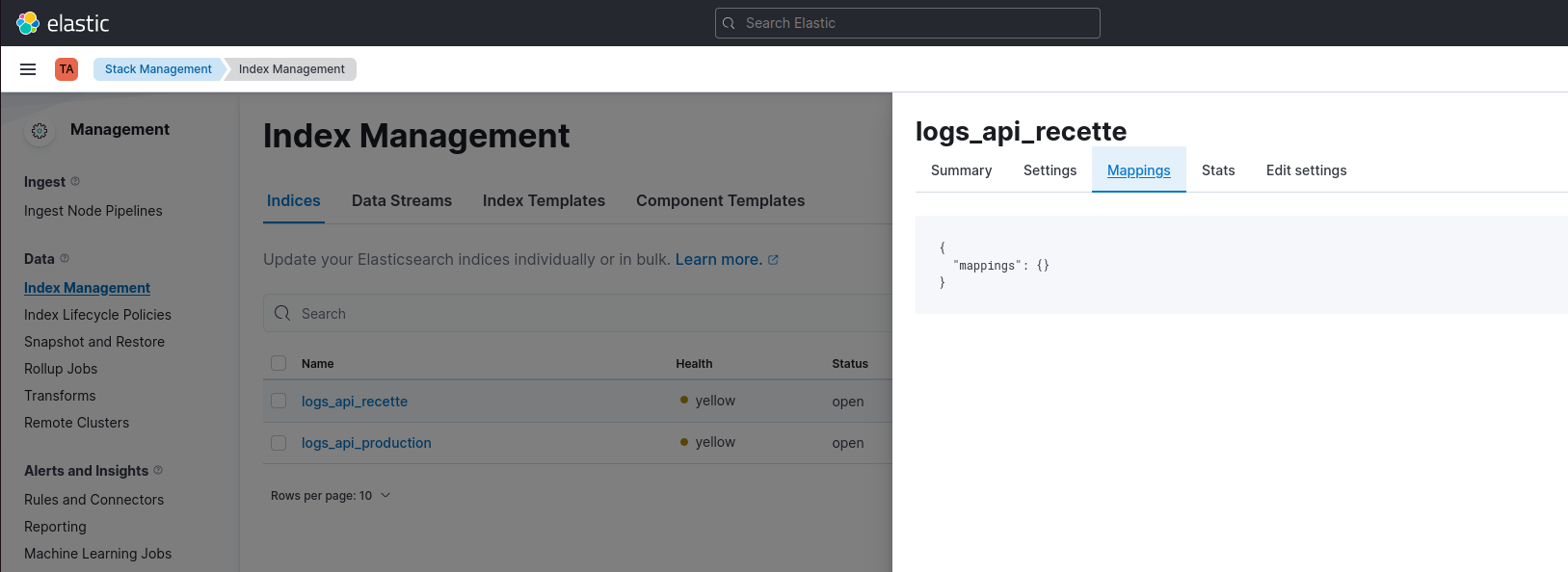
Ces deux index patterns sont visible dans la vue Discover



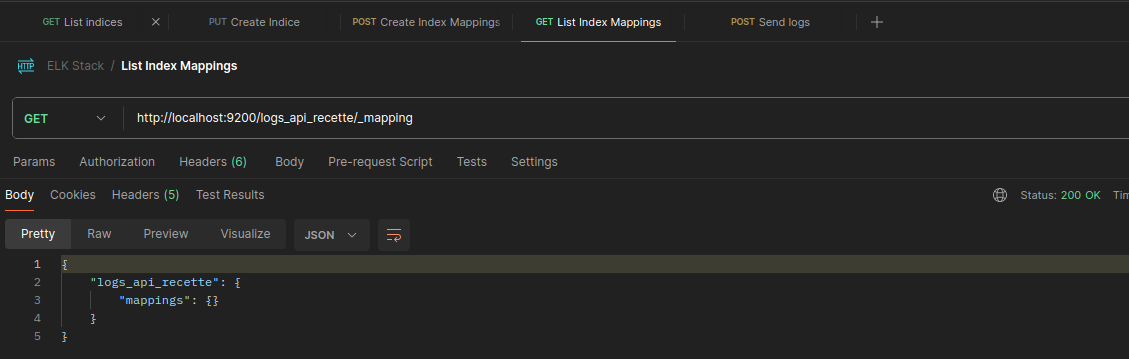
**Mappings** :

Pour capturer des informations significatives dans les logs de votre API, vous pouvez inclure plusieurs champs qui vous aideront à analyser et à comprendre le comportement de votre application.

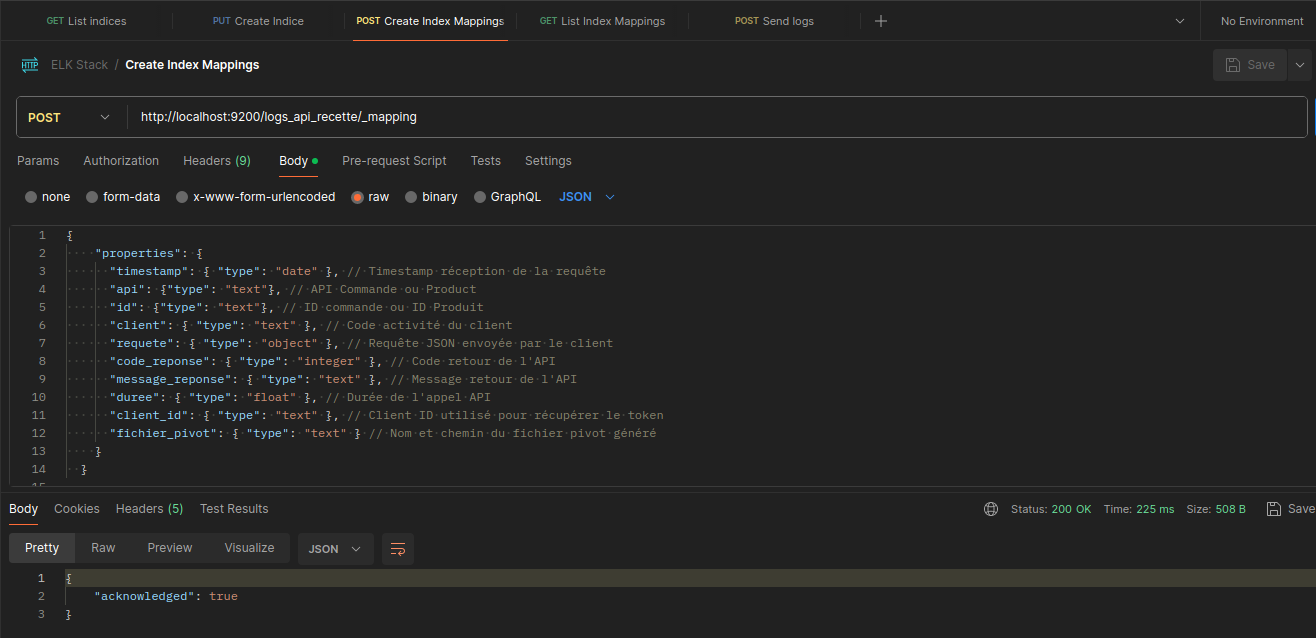
Aller dans Index Management <http://0.0.0.0:5601/app/management/data/index_management/indices>



Pas de mappings pour l’instant. On doit ajouter des mapping à nos index Management



Créer un mapping pour l’index pattern via api REST elasticSearch



Voici quelques champs pertinents que vous pouvez envisager de capturer dans vos logs :

"timestamp": { "type": "date" }, // Timestamp réception de la requête

"api": {"type": "text"}, // API Commande ou Product

"id": {"type": "text"}, // ID commande ou ID Produit

"client": { "type": "text" }, // Code activité du client

"requete": { "type": "object" }, // Requête JSON envoyée par le client

"code\_reponse": { "type": "integer" }, // Code retour de l'API

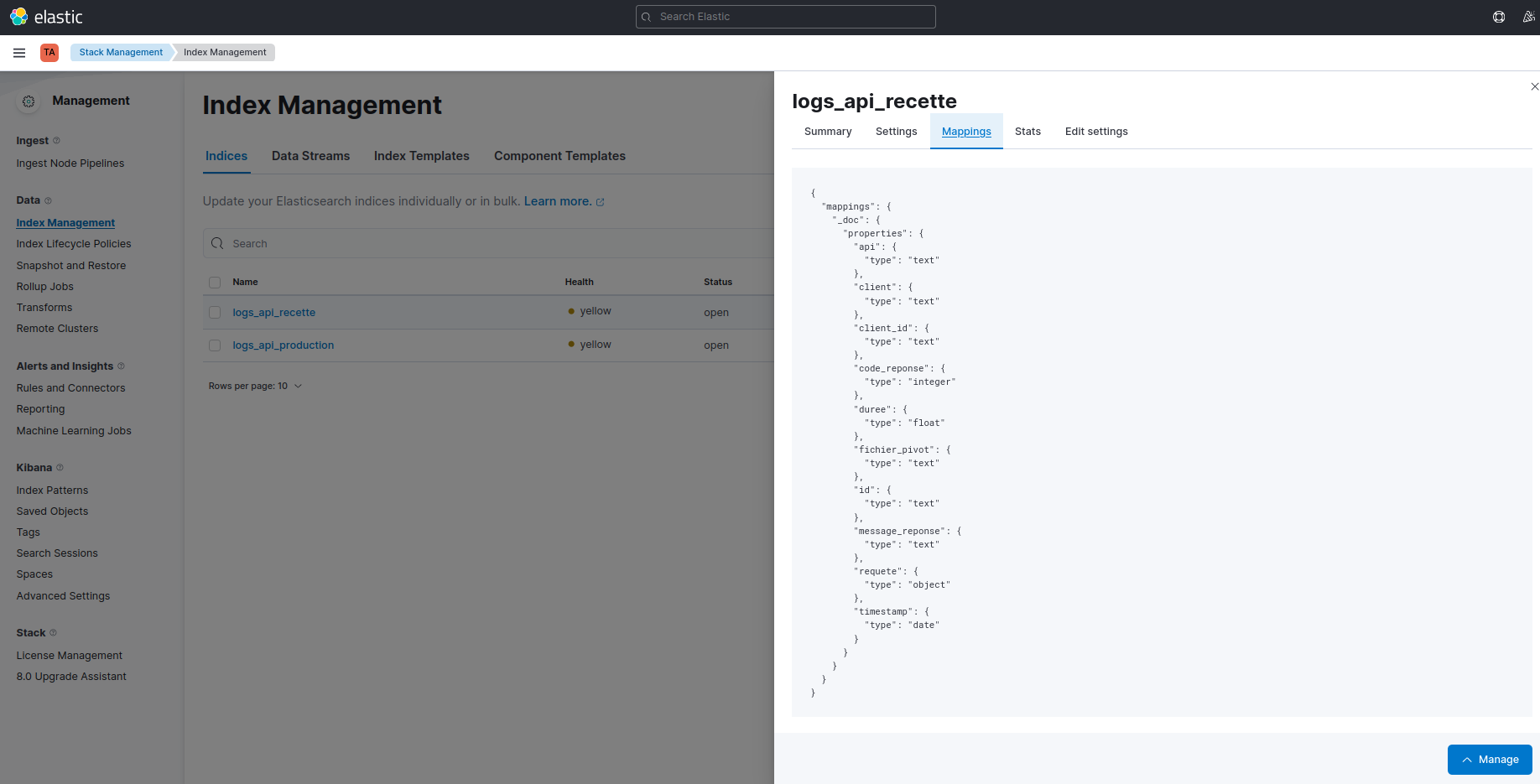
"message\_reponse": { "type": "text" }, // Message retour de l'API

"duree": { "type": "float" }, // Durée de l'appel API

"client\_id": { "type": "text" }, // Client ID utilisé pour récupérer le token

"fichier\_pivot": { "type": "text" } // Nom et chemin du fichier pivot généré

Le mapping est visible sur la vue Index Management et sur l’api rest



Envoyer des logs pour tester en utilisant l’api rest d’elasticsearch

curl --location 'http://localhost:9200/logs\_api\_recette/\_doc' \

--header 'Content-Type: application/json' \

--data '{

"timestamp": "",

"api": "",

"id": "",

"client": "",

"requete": "",

"code\_reponse": "",

"message\_reponse": "",

"duree": "",

"client\_id": "",

"fichier\_pivot": ""

}'

Test envoi log depuis Talend