DataScientest

Projet:
Suivi du Traffic Aérien

Réalisé par:

- Rim Touhami
- Wilson Hounsinou
- Soukaina Sabri
- Soumaya Khila

Plan

Contexte du projet

Les spécifications fonctionnelles /solutions techniques

Demo

Conclusions et perspectives

Plan

Contexte du projet

Besoin métier:

- **Suivi en temps réel du traffic aérien**
- Analyse des données de l'historique du vol

Défis techniques:

Back-end

- De quelles données (types et sources) avons nous besoin ?
- Quelle architecture technique considérer pour les acheminer, les traiter, les stocker et les consommer?
- Comment assurer l'automatisation des tâches précédentes?

Front-end

- Comment procurer un visuel informatif du traffic aérien (temps réel)?
- Comment peut-on récolter et analyser les informations de l'historique du vol ?

Plan

Contexte du projet

Les spécifications fonctionnelles /solutions techniques

1. Collecte
2. Stockage
3. Consommation
4. Automatisation/ Déploiement

Demo

Conclusions et perspectives

Spécification fonctionnelle ⟶ solution technique

ÉTAPES	TÂCHES
1-Récolte des données	Web scrapping: extraction des données statiques API requests: récupération des données dynamiques du traffic sur OpenSky
2-Stockage des données	MySQL: stockage des données statiques (aéroports, compagnies aériennes) MongoDB: stockage des positions géospatiales en temps réel (OpenSky) Elasticsearch: stockage des données historiques (vols passés)
3-Consommation des données	Dash: visualisation du trafic aérien temps réel Kibana: Récapitulatif de l'historique de vol • Logstash: pipeline de transfert de données statiques vers Elasticsearch • Elasticsearch: importation et jointure des données
4-Automatisation/Déploiement des tâches	Airflow: • Automatisation des pipelines • Gestion des dépendances • Schedule des processus des pipelines Docker: • Déploiement des différentes composantes grâce à la containeurisation

1- Collecte des données

Données dynamiques

- Données dynamiques: Les données qui sont mis à jour fréquemment.

a. API Requests de OpenSky

Lien: https://openskynetwork.github.io/opensky-api/.

Avantages:

- Gratuité
- Offrir un nombre très important d'appels (2000 appels par jour).

Deux types d'informations sont collectées:

- Les positions en temps réels des avions (toutes les 45 secondes)
- Les informations sur les vols effectués un jour avant.

Extrait des données des vols


```
[{
    icao24: "39ceb5",
    firstSeen: 1672944223,
    estDepartureAirport: null,
    lastSeen: 1672951583,
    estArrivalAirport: "LFPO",
    callsign: "TVF14AR ",
    estDepartureAirportHorizDistance: 0,
    estDepartureAirportVertDistance: 0,
    estArrivalAirportHorizDistance: 203,
    estArrivalAirportVertDistance: 103,
    departureAirportCandidatesCount: 0,
    arrivalAirportCandidatesCount: 88
},
{
    icao24: "710138",
    firstSeen: 1672951102,
    estDepartureAirport: "OERY",
    lastSeen: 1672951711,
    estArrivalAirport: "OERY",
    callsign: "SVA1784 ",
    estDepartureAirportHorizDistance: 7643,
    estDepartureAirportVertDistance: 5324,
    estArrivalAirportHorizDistance: 17661,
    estArrivalAirportVertDistance: 2946,
    departureAirportCandidatesCount: 0,
    arrivalAirportCandidatesCount: 0
}
```

1- Collecte des données

Données statiques

Données statiques: Données qui ne se mettent à jour que très rarement (nouvelle compagnies aérienne/nouveau aéroport/ changement de position d'un aéroport).

b. Web scrapping

Besoin: Manque des données sur les aéroports et les compagnies aériennes.
Solution: WebScrapping

1. Pour récupérer les données sur les aéroports:
   Aeroports (Nom aéroport, ville, pays, taille, IATA, ICAO)

- Source: https://www.world-airport-codes.com
- Type: Sauvegarde sous format csv
- Volumétrie: 9400 lignes

iCAO	IATA	nom	taille	pays	ville
SWRA	AAI	Arraias	Smallairport	Brazil	Arraias
SMCA	AAJ	Cayana Airstrip	Smallairport	Suriname	Awaradam
NGUK	AAK	Buariki	Smallairport	Kiribati	Buariki
EKYT	AAL	Aalborg	Largeairport	Denmark	Aalborg
FAMD	AAM	Malamala	Mediumairport	South Africa	Malamala
OMAL	AAN	Al Ain International	Mediumairport	United Arab Emirates	Al Ain
SVAN	AAO	Anaco	Mediumairport	Venezuela	Anaco
KAAP	AAP	Andrau Airpark	Closed	United States	Houston
URKA	AAQ	Anapa Vityazevo	Mediumairport	Russia	Anapa
EKAH	AAR	Aarhus	Mediumairport	Denmark	Aarhus

1- Collecte des données

Données statiques

2. Pour récupérer les données sur les compagnies:

Compagnies (icao24, registration, manufacturericao, manufacturername, model, serialnumber, ownername)

- Source: https://opensky-network.org/datasets/metadata/
- Type: Sauvegarde sous format csv
- Volumétrie: 418562 lignes

icao24	registration	manufacturericao	manufacturername	model	serialnumber	ownername
0082d7	ZS-BBI	HARBIN	Embraer	ERJ-145 LR	145223	Fly Blue Crane
0082d8	ZS-BBJ	HARBIN	Embraer	ERJ-145 LR	145277	Fly Blue Crane
0082d9	ZS-BBK	EMBRAER	Embraer	ERJ-135 LR	145396	Solenta Aviation
0082db	ZS-BBM	HARBIN	Embraer	ERJ-145 LR	145597	Solenta Aviation
0082f1	ZS-BCI	SOCATA	Eads Socata	TBM 850	397	National Airways C
008305	ZS-BDC	DASSAULT	Dassault	Falcon 10	148	Plennegy P/l
008309	ZS-BDG	RAYTHEON	Raytheon Aircraft Compan	Premier I	RB-68	Bancore Trading P/l
00831f	ZS-BEC	ROBINSON	Robinson	R44 Raven II	11937	Escape Airtours Ch
008321	ZS-BEE	ROBINSON	Robinson	R44 Raven II	11612	Private
008325	ZS-BEI	PIPER	Piper	PA-46-500TP	4697523	B And E Internation

1- Collecte des données

Données statiques

3. Pour récupérer les données sur les positions des aéroports:

Airports_locations (icao, lat, lon)

- Source: https://www.partow.net/miscellaneous/airportdatabase/
- Type: Sauvegarde sous format csv
- Volumétrie: 9300 lignes

ICAO	Latitude	Longitude
AYGA	-6.082	145.392
AYLA	0.000	0.000
AYMD	-5.207	145.789
AYMH	-5.826	144.296
AYNZ	-6.570	146.726
AYPY	-9.443	147.220
AYRB	0.000	0.000
AYWK	-3.584	143.669
BGAM	0.000	0.000
BGAS	0.000	0.000
BGAT	0.000	0.000
BGBW	61.161	-45.427
BGCH	0.000	0.000

Spécification fonctionnelle ⟶ solution technique

ÉTAPES	TÂCHES
1-Récolte des données	Web scrapping: extraction des données statiques API requests: récupération des données dynamiques du traffic sur OpenSky
2-Stockage des données	**MySQL**: stockage des données statiques (aéroports, compagnies aériennes) **MongoDB**: stockage des positions géospatiales en temps réel (OpenSky) **Elasticsearch**: stockage des données historiques (vols passés)
3-Consommation des données	Dash: visualisation du trafic aérien temps réel Kibana: Récapitulatif de l'historique de vol • Logstash: pipeline de transfert de données statiques vers Elasticsearch • Elasticsearch: importation et jointure des données
4-Automatisation/Déploiement des tâches	Airflow: • Automatisation des pipelines • Gestion des dépendances • Schedule des processus des pipelines Docker: • Déploiement des différentes composantes grâce à la containeurisation

2- Stockage des données

Choix des bases

a. Stockage des données statiques: pourquoi MySQL ?

- BD Relationnelle optimisée pour le stockage durable de données à faible évolution.
- Gratuité, facilité d'usage, cohérence des données et adaptabilité pour des projets de petite envergure.
- La plus rapide en mode lecture seulement (vs PostgreSQL)

b. Stockage des données dynamiques: pourquoi MongoDB ?

- BD orientée document optimisée pour les opérations CRUD à haut débit (OLTP)
- Performance, scalabilité et compatibilité avec les technologies Cloud

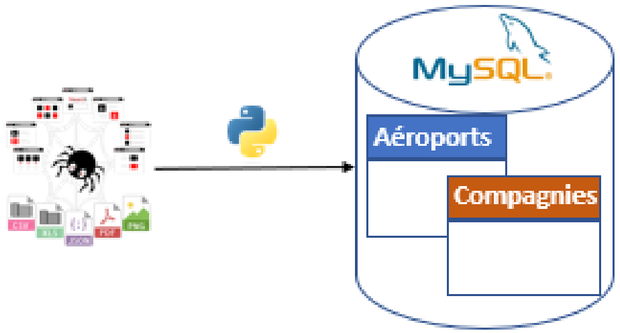
c. Stockage des données historiques: pourquoi Elasticsearch ?

- BD orientée document optimisée pour la recherche
- Équipé de Kibana facilitant la conception des dashboards

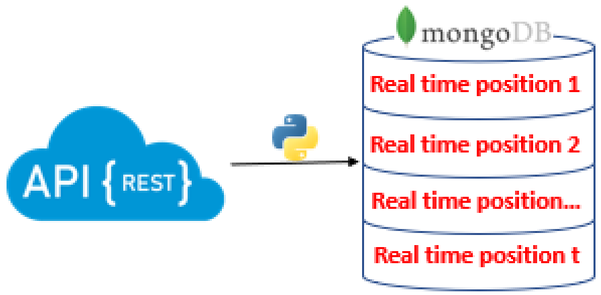
2- Stockage des données

Alimentation des bases

a. Pour alimenter MySQL --> utiliser des scripts Python qui récupèrent les données scrappées --> les charger dans des tables spécifiques: Compagnies et Aéroports (1 fois/ mois).

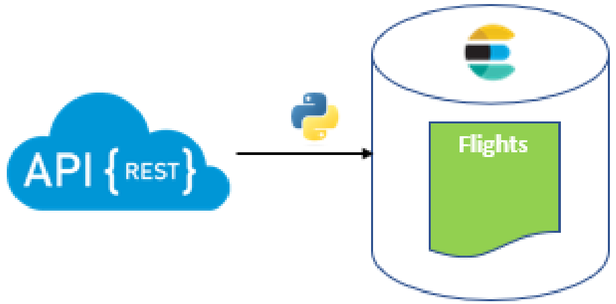


b. Pour alimenter MongoDB --> utiliser des scripts Python qui récupèrent les positions en temps réel retournées par l'API d'OpenSky --> les importer dans une collection appelée Positions (1 fois/ 45 s).

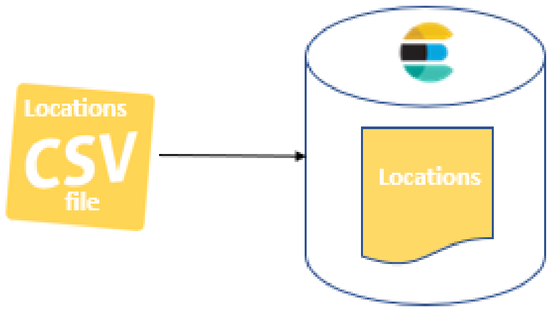


2- Stockage des données
Alimentation des bases

c. Pour alimenter Elasticsearch,
--> utiliser des scripts Python qui récupèrent les données de la veille retournées par l'API d'OpenSky
--> les charger dans un index nommé Flights. (1 fois / jr)



--> importer le fichier scrappé "airports_locations.*csv*" dans un index "locations"



Spécification fonctionnelle ⟶ solution technique

ÉTAPES	TÂCHES
1-Récolte des données	Web scrapping: extraction des données statiques API requests: récupération des données dynamiques du traffic sur OpenSky
2-Stockage des données	MySQL: stockage des données statiques (aéroports, compagnies aériennes) MongoDB: stockage des positions géospatiales en temps réel (OpenSky) Elasticsearch: stockage des données historiques (vols passés)
3-Consommation des données	Dash: visualisation du trafic aérien temps réel (MongoDB) Kibana: Récapitulatif de l'historique de vol • Logstash: pipeline de transfert de données statiques vers Elasticsearch • Elasticsearch: importation et jointure des données
4-Automatisation/Déploiement des tâches	Airflow: • Automatisation des pipelines • Gestion des dépendances • Schedule des processus des pipelines Docker: • Déploiement des différentes composantes grâce à la containeurisation

3- Consommation des données
Logstash

Problématique:

- Les données importées dans Elasticsearch ne sont pas parlantes:
  - Code ICAO identifiant d'aéroport au lieu des détails sur l'aéroport!
  - Code icao24 de chaque avion au lieu du nom de la compagnie qui l'exploite!

Besoin:

importer des données du MySQL:

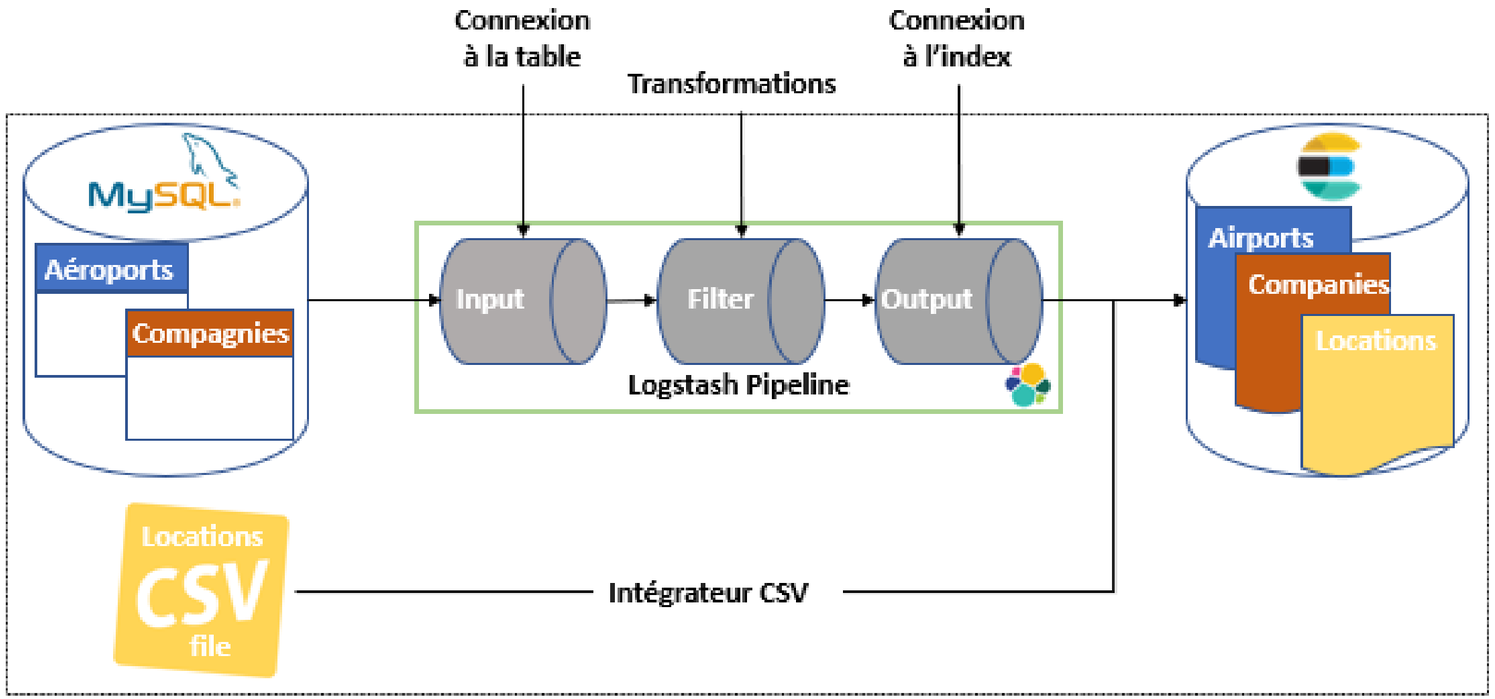
- Importer les détails des aéroports dans l'index Airports
- Importer les détails des compagnies dans l'index Companies

Importer les positions des aéroports vers l'index Locations

3- Consommation des données
Logstash

Solution:

- Utilisation de Logstash (ETL d'Elasticsearch) pour charger les tables Airports et Companies.
  - Mise en place de 2 pipelines de données avec Logstash en trois étapes: Input -> Filter -> Output
- Utilisation d'un intégrateur CSV d'Elasticsearch pour charger l'index Locations

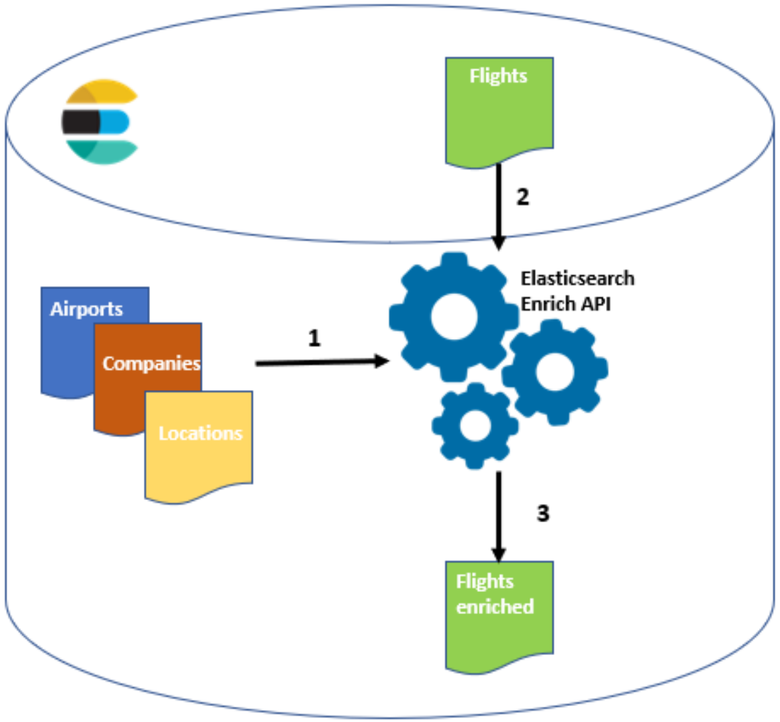


3- Consommation des données Elasticsearch

Besoin: "Joindre" l'index Flights aux indexes Airports, Companies et Locations

Solution: Utilisation de l'API Enrich d'Elasticsearch pour créer un nouvel index Flights_enriched

- Configuration et exécution des pipelines de jointures
- Réindexation des données au bon format



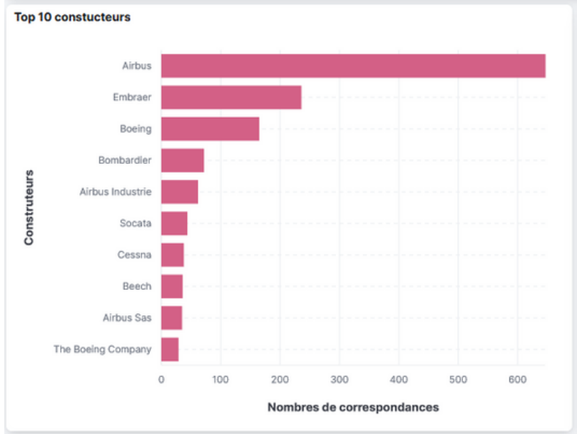
3- Consommation des données

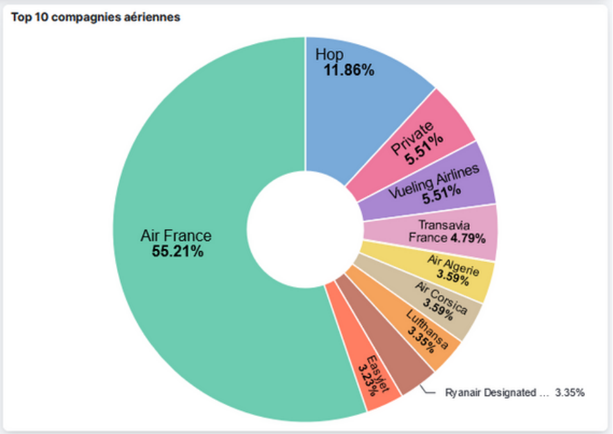
Kibana

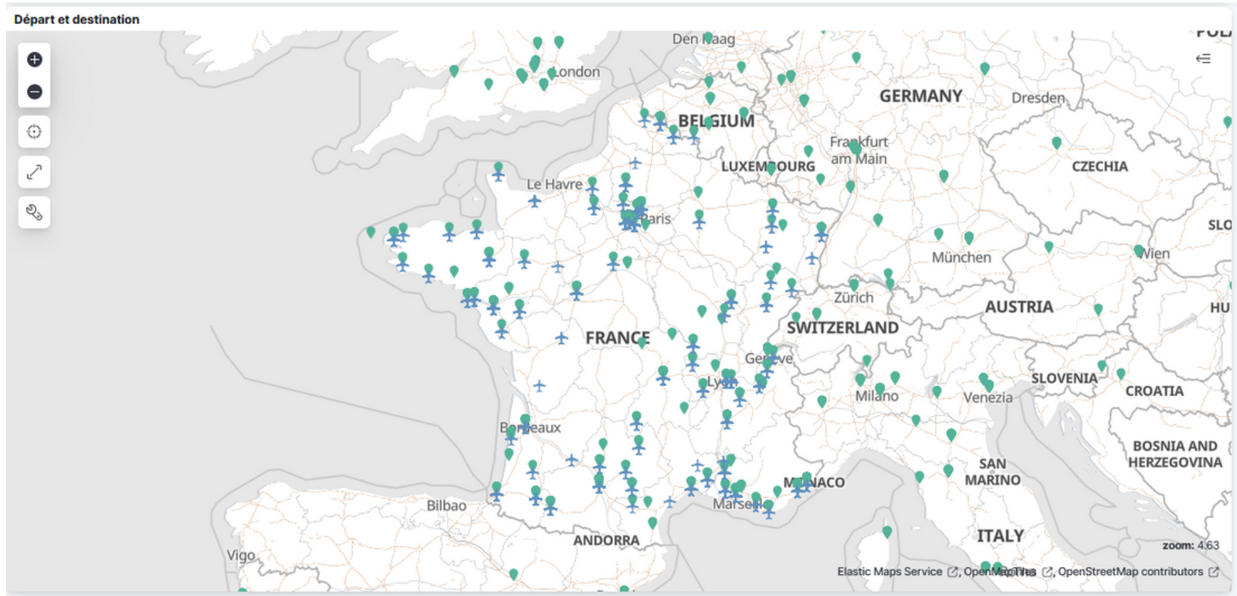
Besoin: Visualisation de l'historique de vol de chaque avion

Solution: Kibana

Utilisation: Mise en place d'un dashboard qui récapitule le trafic aérien des jours passés





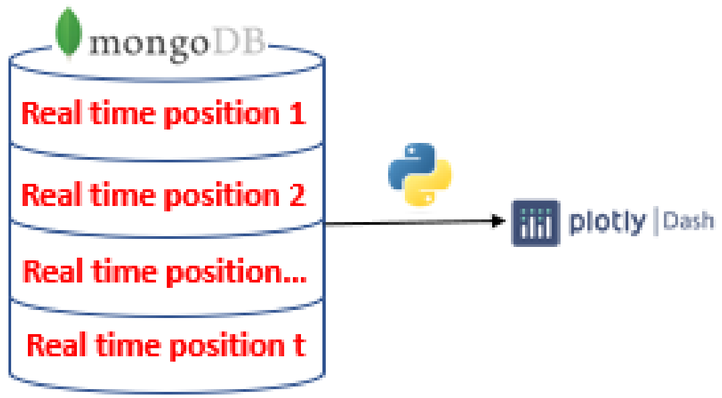


3- Consommation des données
Plotly Dash

Besoin: Visualiser les postions en temps réel des avions

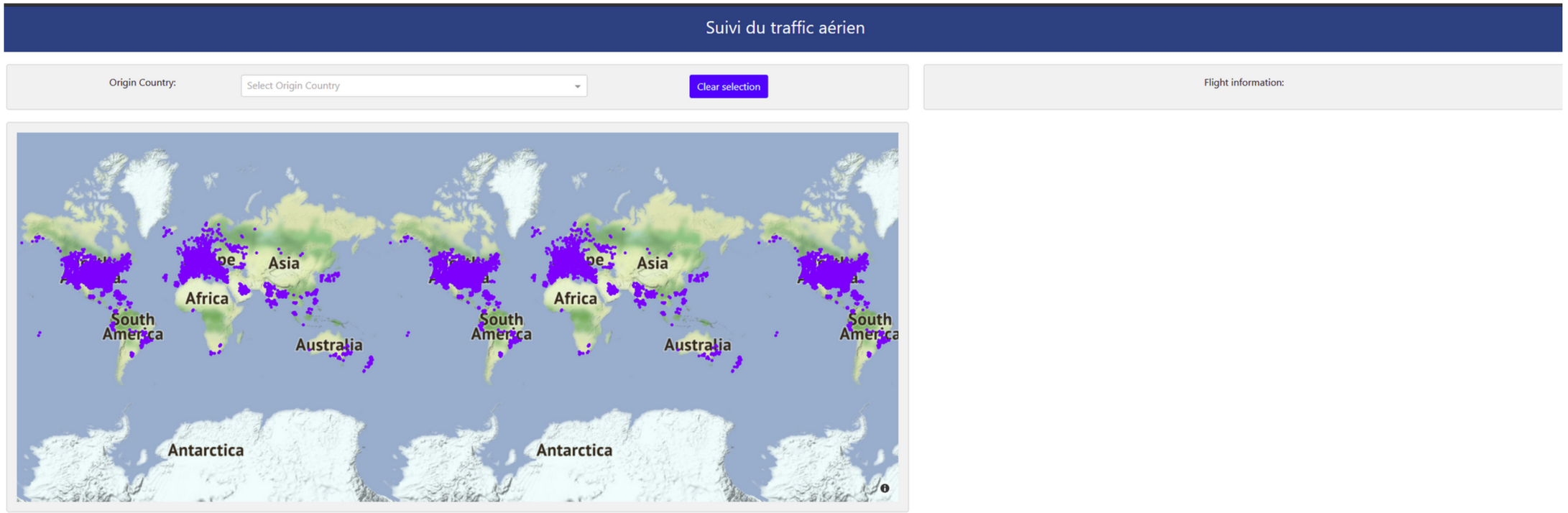
Solution: Utilisation de Dash
Framework en langage Python qui facilite la mise en place des visualisations de tout genre

Utilisation:
Visualisation des positions des avions rafraîchies toutes les 45 secondes



3- Consommation des données
Plotly Dash

Imprime-écran du dashboard:



Spécification fonctionnelle ⟶ solution technique

ÉTAPES	TÂCHES
1-Récolte des données	Web scrapping: extraction des données statiques API requests: récupération des données dynamiques du traffic sur OpenSky
2-Stockage des données	MySQL: stockage des données statiques (aéroports, compagnies aériennes) MongoDB: stockage des positions géospatiales en temps réel (OpenSky) Elasticsearch: stockage des données historiques (vols passés)
3-Consommation des données	Dash: visualisation du trafic aérien temps réel Kibana: Récapitulatif de l'historique de vol • Logstash: pipeline de transfert de données statiques vers Elasticsearch • Elasticsearch: importation et jointure des données
4-Automatisation/Déploiement des tâches	**Airflow:** • **Automatisation des pipelines** • **Gestion des dépendances** • **Schedule des processus des pipelines** **Docker:** • **Déploiement des différentes composantes grâce à la containeurisation**

4-Automatisation/Déploiement

Airflow

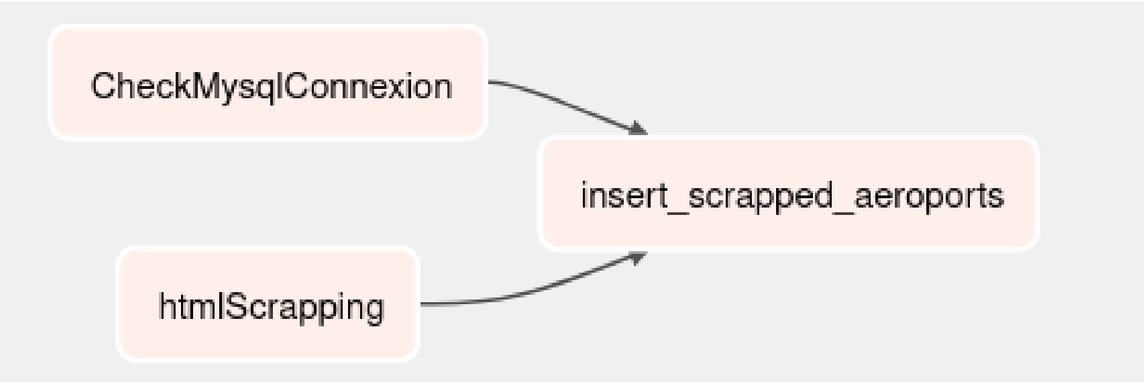
Besoin: Orchestrer le fonctionnement des différentes composantes de l'architecture

Solution proposée: Airflow (Plateforme Apache Open Source):

- Créer des pipelines (DAGs)
- Planifier le lancement des tâches (schedule);
- Surveiller des workflows (interface utilisateur).

Utilisation: création de trois DAG

1. Automatisation de la mise à jour de la table "*Aeroports*" dans MySQL:



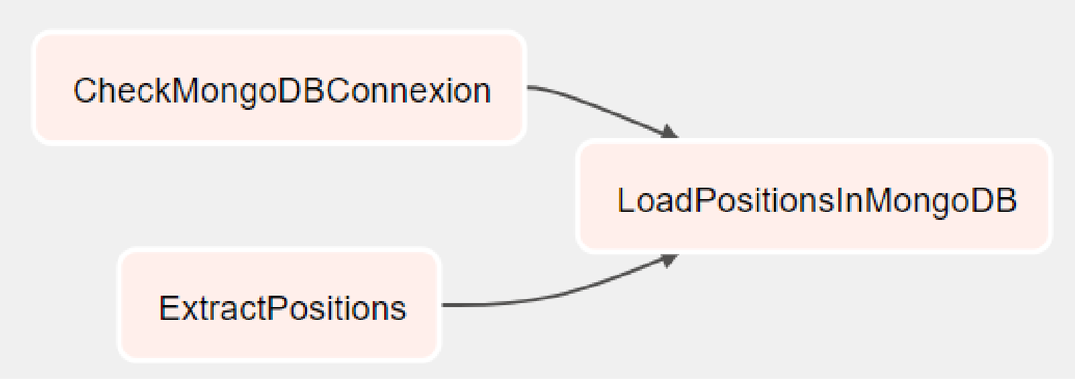
Schedule: 1 fois / mois

M.A.J uniquement des données pré-existantes

4-Automatisation/Déploiement

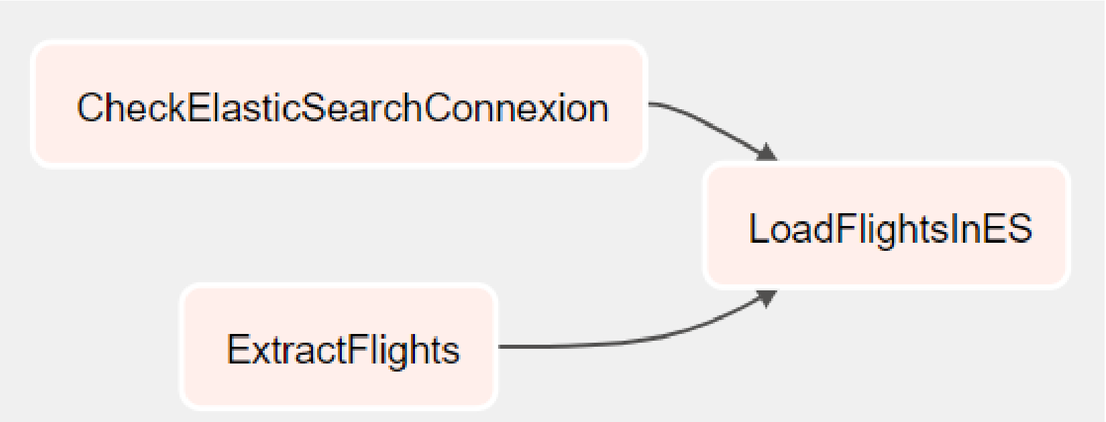
Airflow

2. Automatisation du remplissage de MongoDB:



Schedule: 1 fois / 45 s

3. Automatisation du remplissage d'Elasticsearch:



Schedule: 1 fois / jour

4- Automatisation/Déploiement

Docker

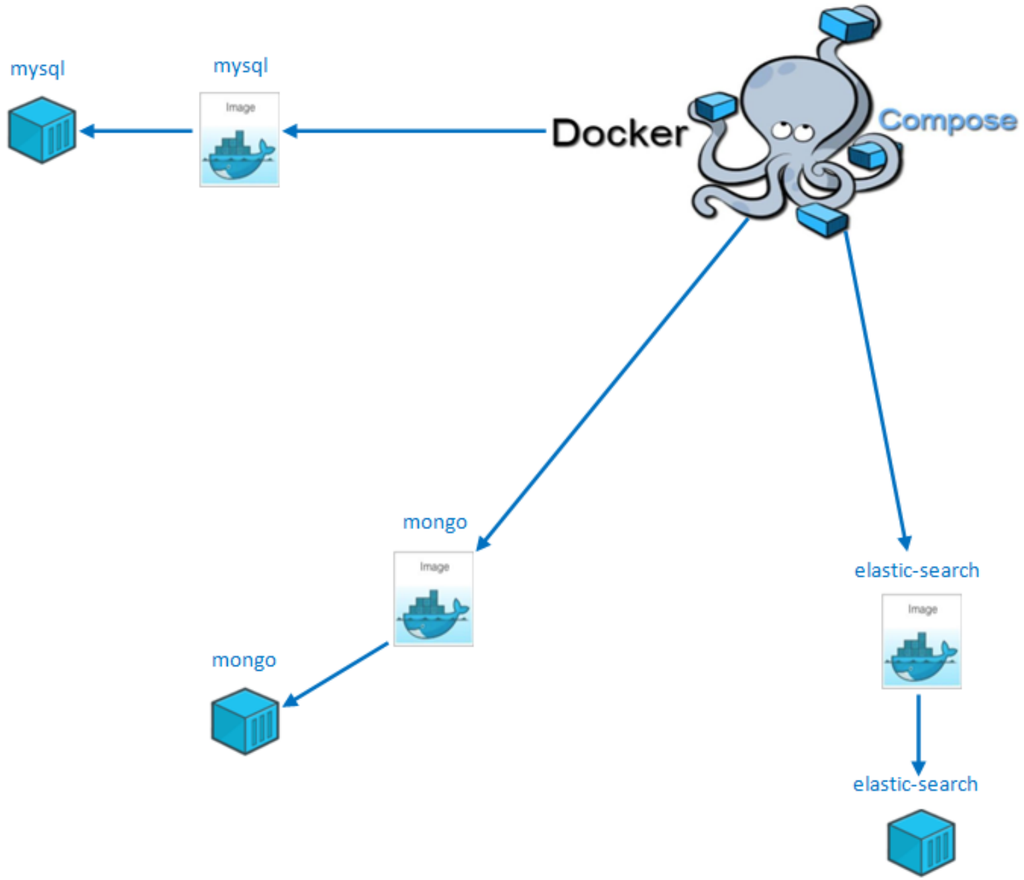
Besoin: Déployer à moindre coût l'architecture de données du projet
Garantir l'installation de notre application dans n'importe quel environnement

Solution: Docker
Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur.

Utilisation: Déploiement d'images nécessaires et de leurs dépendances

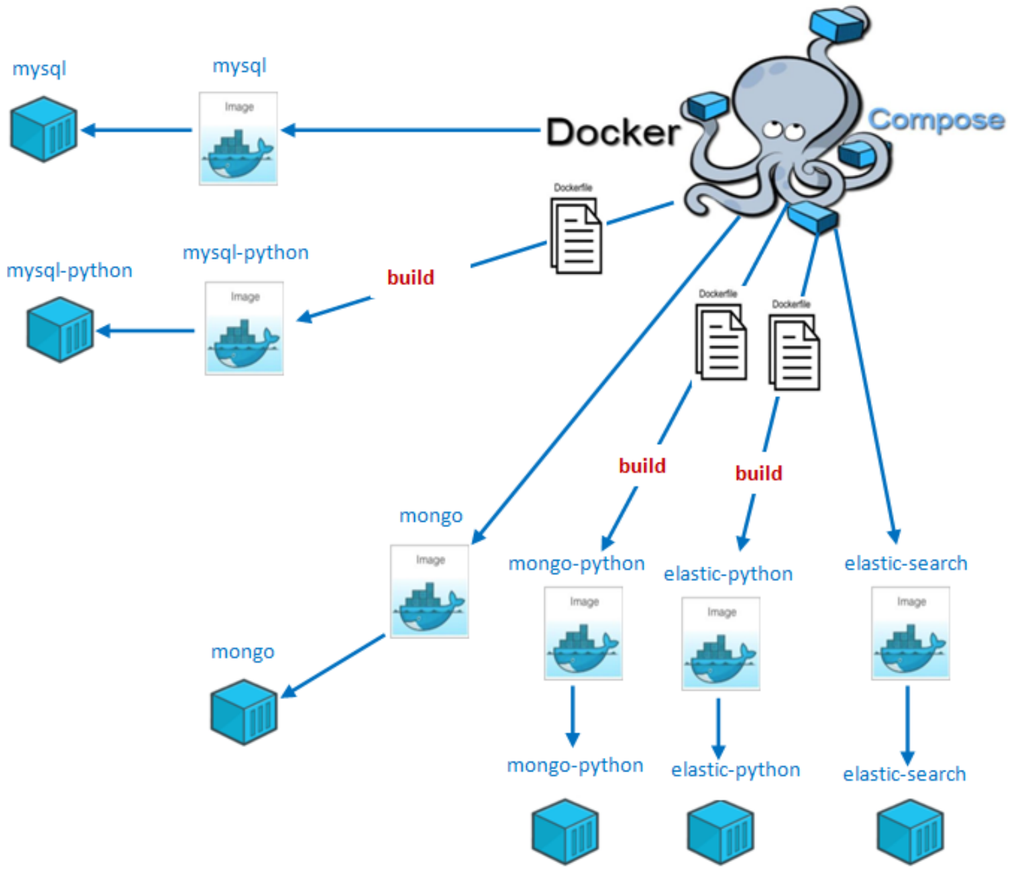
4- Automatisation/Déploiement

Docker (architecture générale)



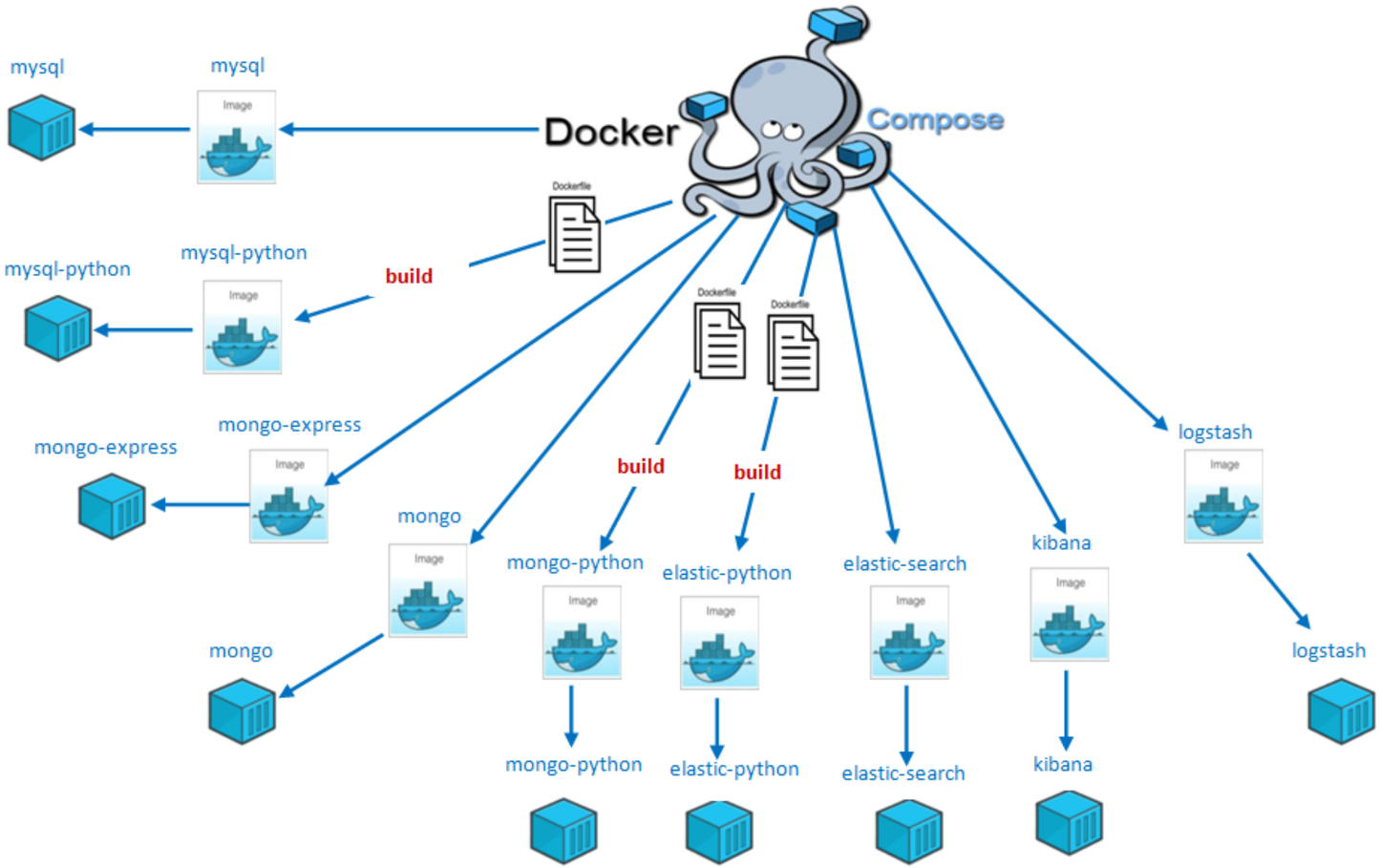
4- Automatisation/Déploiement

Docker (architecture générale)



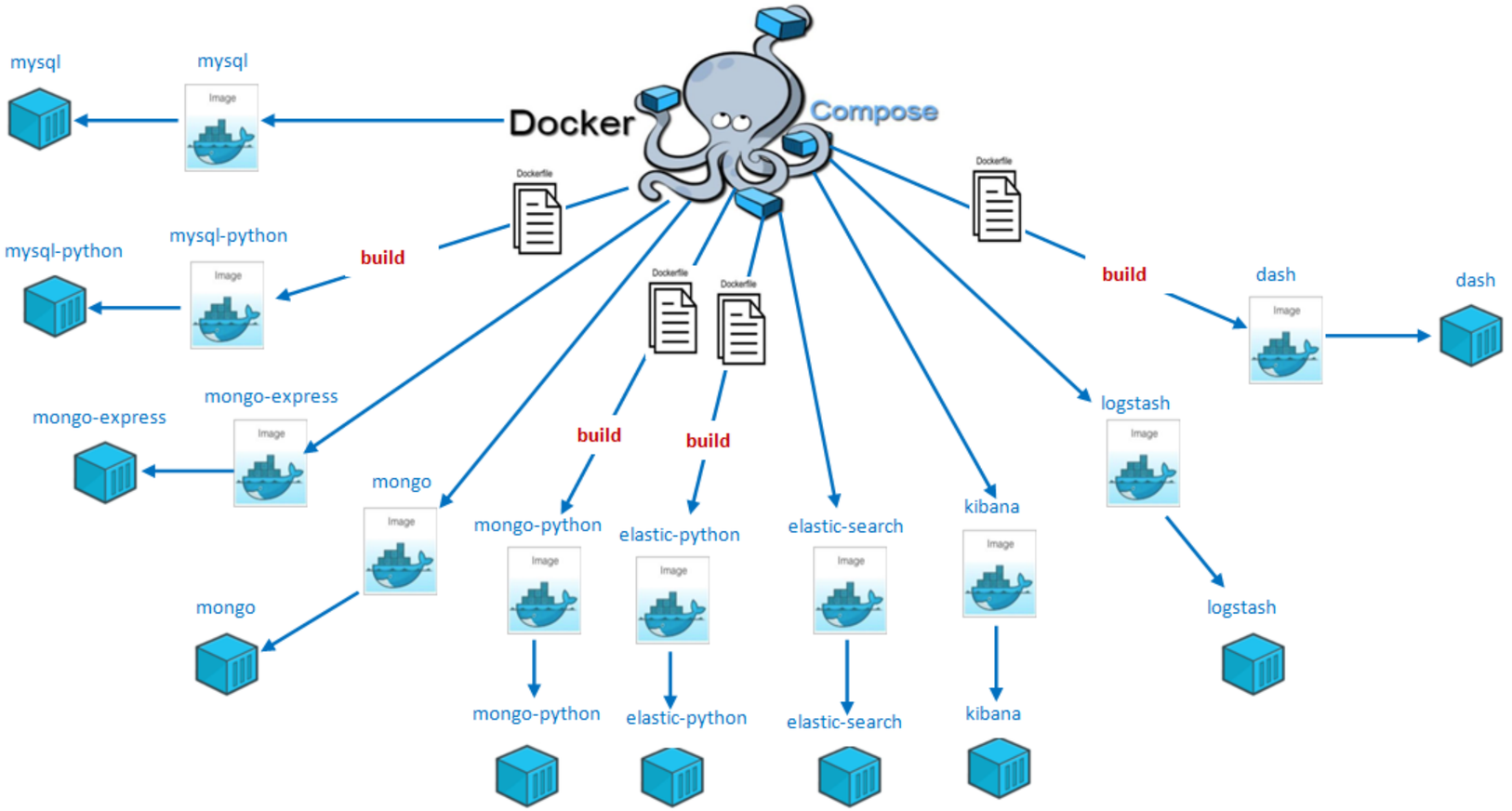
4- Automatisation/Déploiement

Docker (architecture générale)



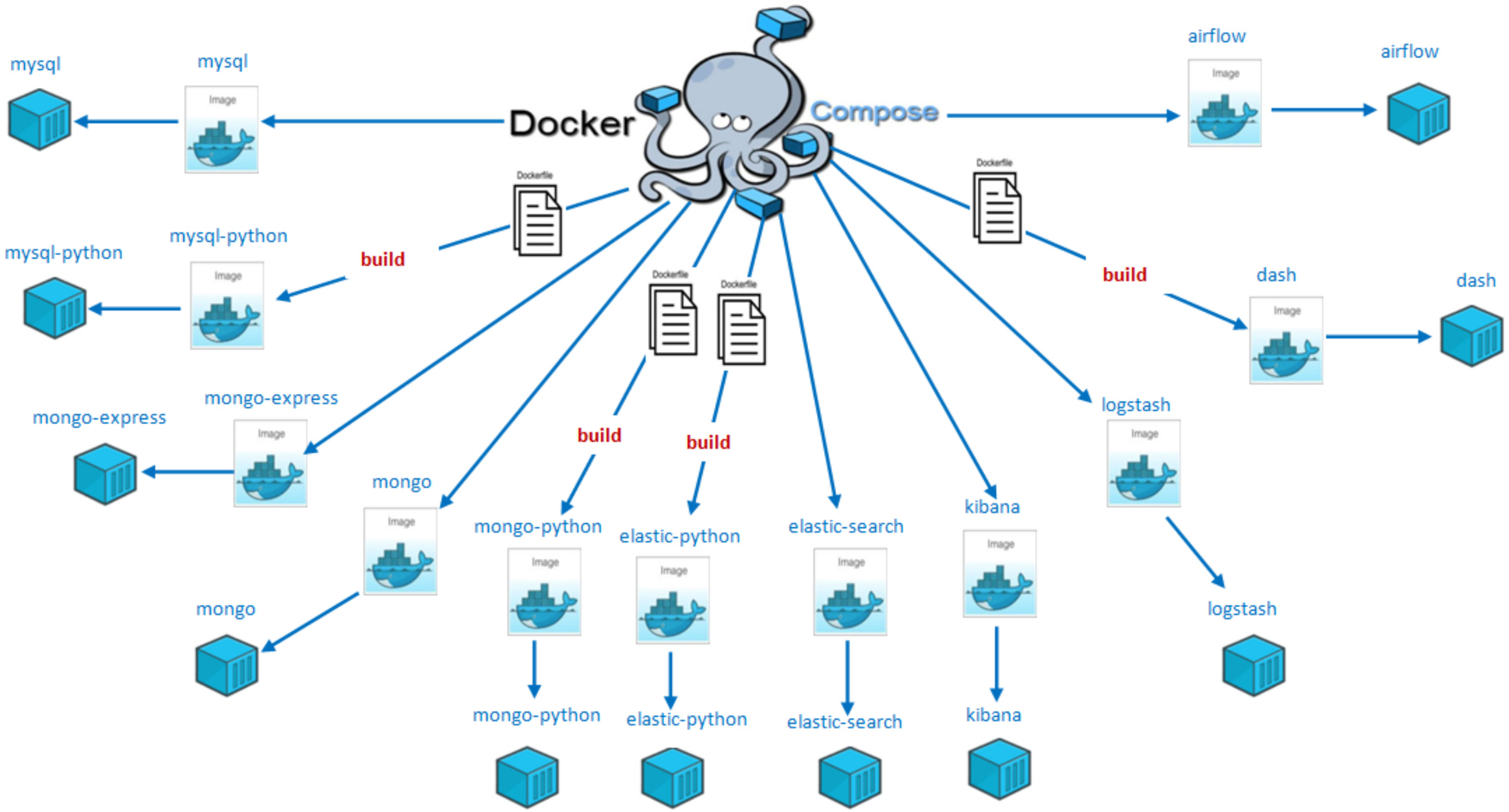
4- Automatisation/Déploiement

Docker (architecture générale)



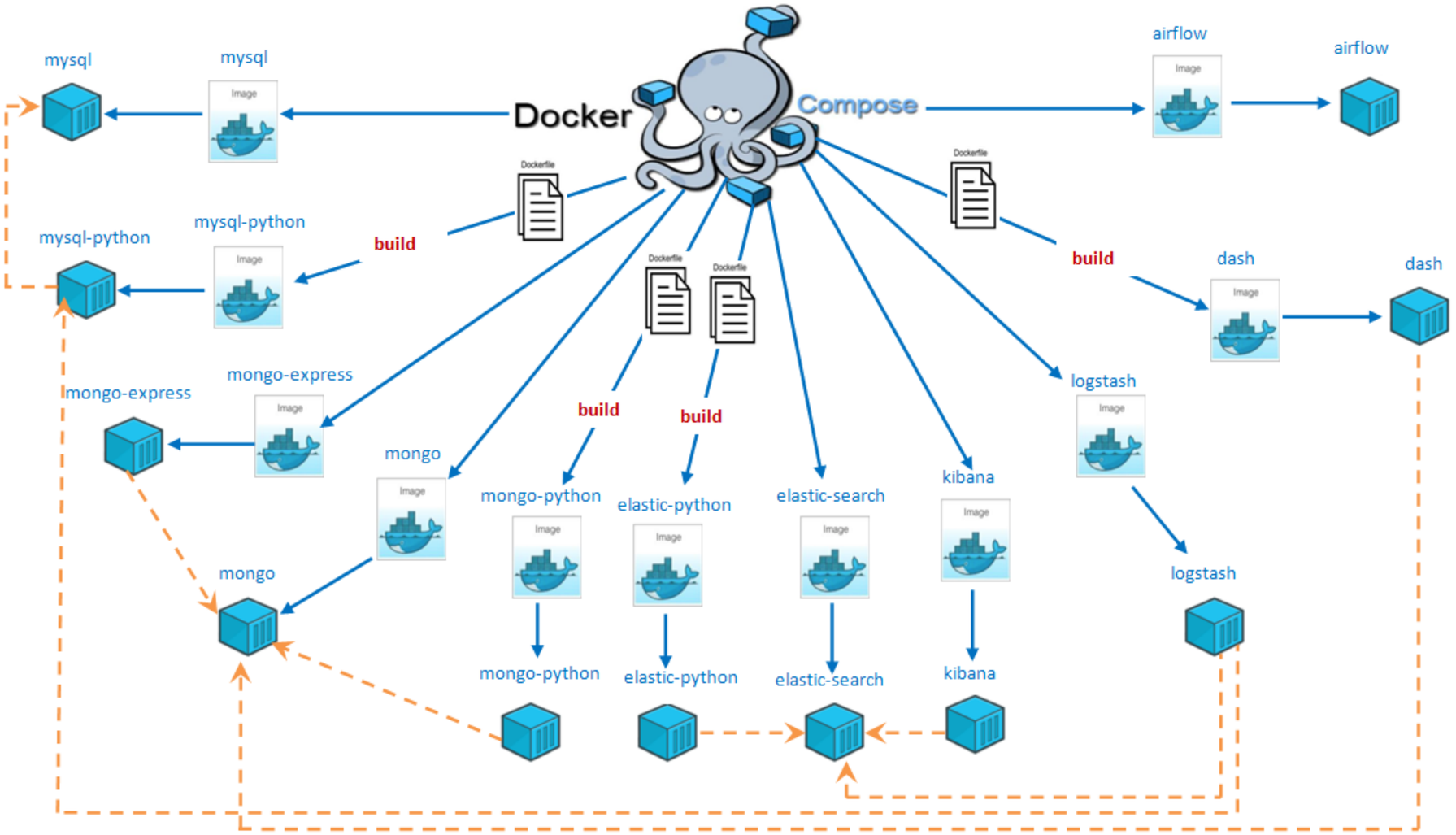
4- Automatisation/Déploiement

Docker (architecture générale)



4- Automatisation/Déploiement

Docker (architecture générale)



4- Automatisation/Déploiement

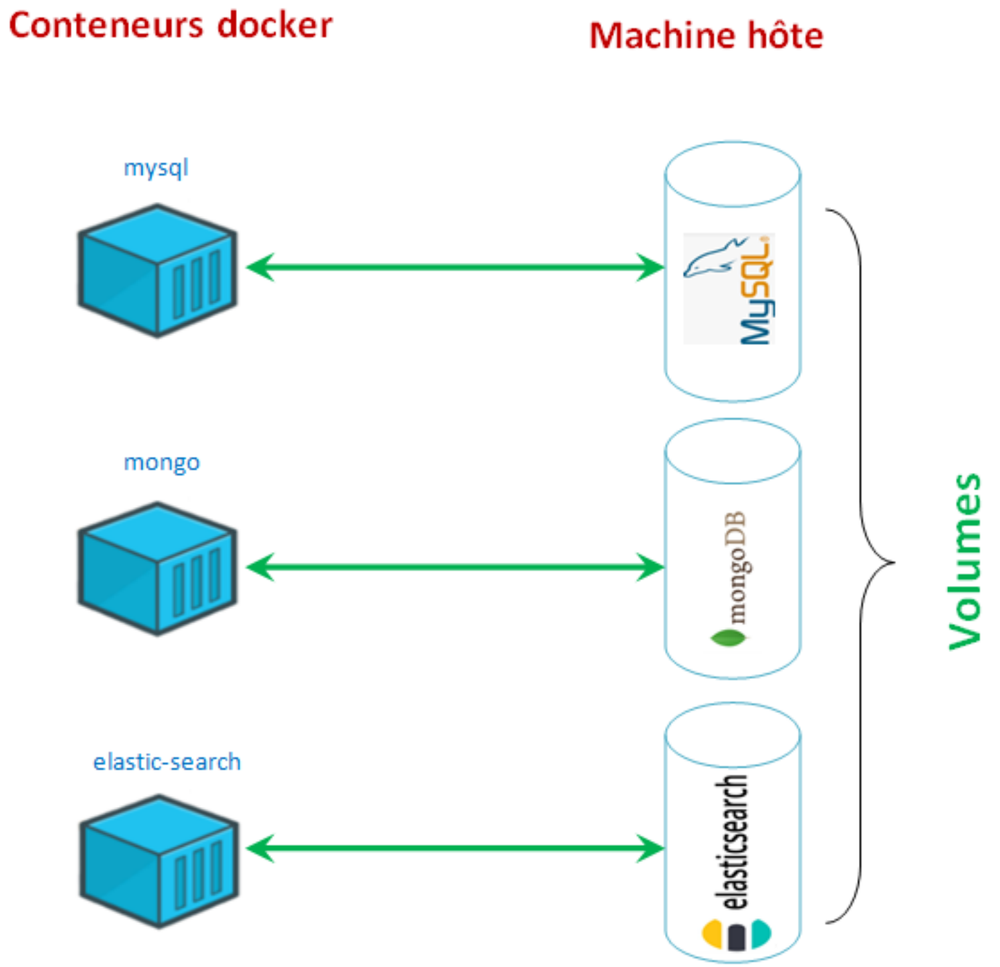
Docker (architecture générale)

Besoin:

- Stocker les données d'une manière persistante.
- Partager les données entre conteneurs.

Solution:

- Utilisation des volumes Docker (mécanisme préféré pour la persistance des données):
  - Un volume pour mysql
  - Un volume pour mongoDB
  - Un volume pour elastic-search
- Créer un network pour connecter les conteneurs



4- Automatisation/Déploiement

Docker (architecture générale)

Besoin:

Utiliser les scripts python stockés dans la machine hôte dans le conteneur airflow.

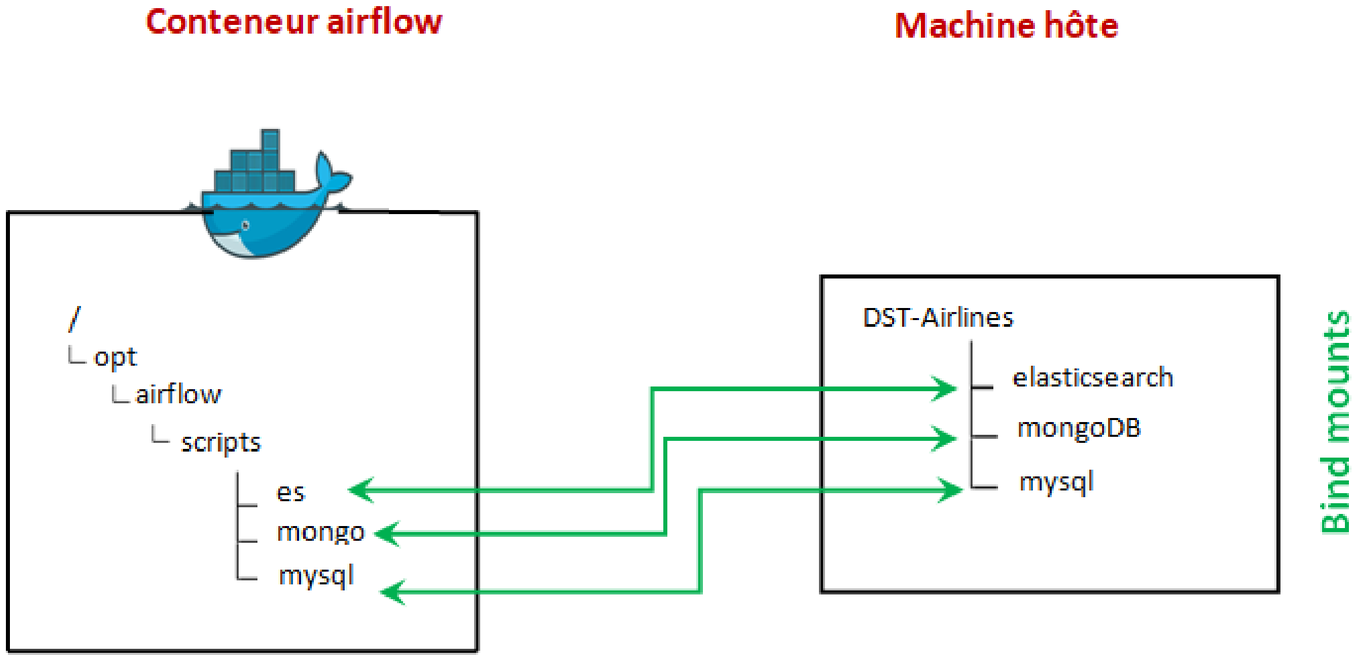
Solutions:

- Utilisation des bind mounts depuis la machine hôte
  - Un bind mount pour mysql
  - Un bind mount pour mongoDB
  - Un bind mount pour elastic-search

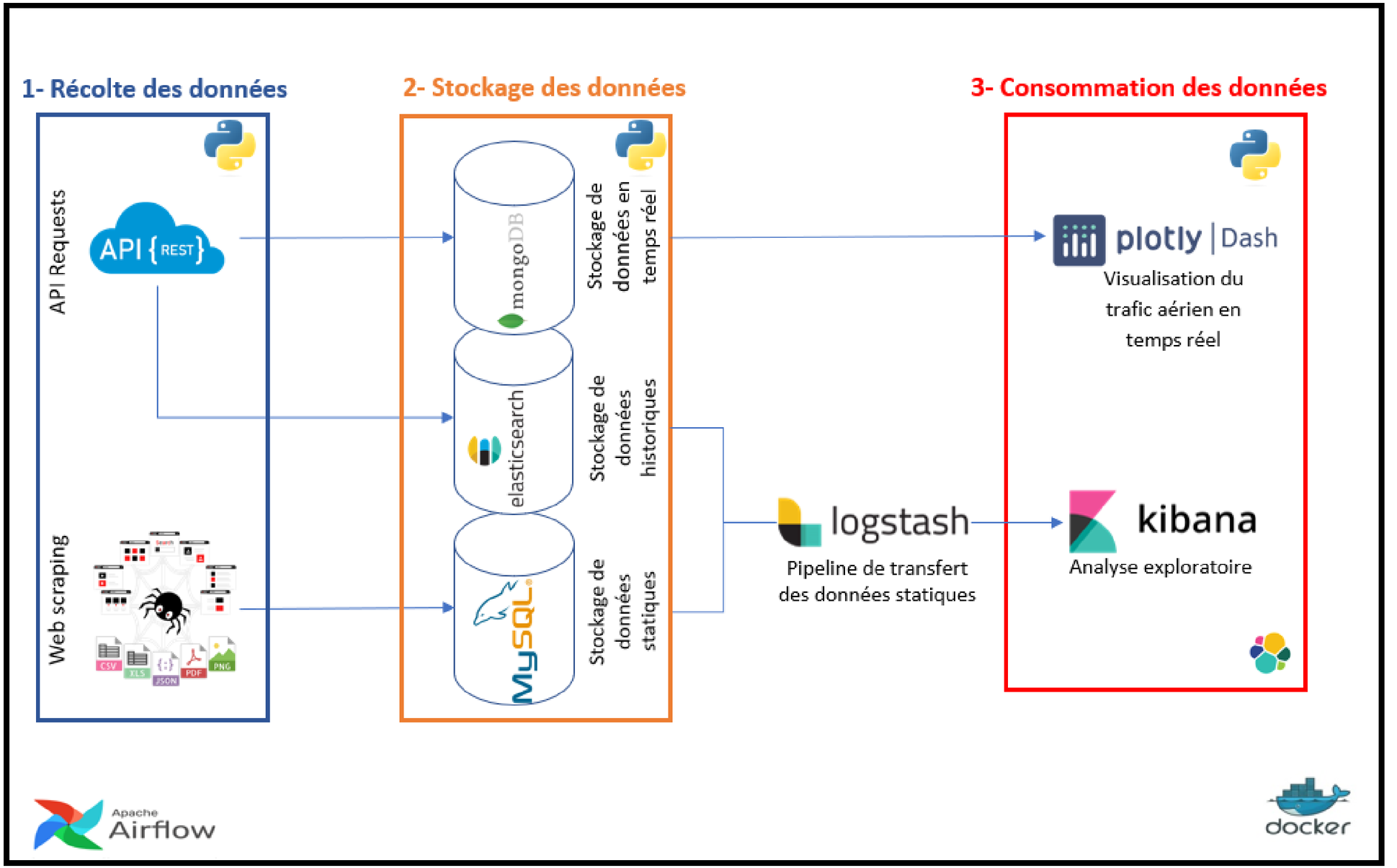
sys.path.append('/opt/airflow/scripts/...')

- Installations des modules supplémentaires dans airflow

_PIP_ADDITIONAL_REQUIREMENTS:${_PIP_ADDITIONAL_REQUIREMENTS:-pymongo beautifulsoup4}



4-Automatisation / Déploiement



Plan

Contexte du projet

Les spécifications fonctionnelles /solutions techniques

Demo

Conclusions et perspectives

DEMO

Plan

Conclusions et perspectives

Conclusions

-Proposition d'une solution pour le suivi du traffic aérien:

Back-end:

- Stockage des données dans trois bases de données (*MySQL*, *Elasticsearch* et *MongoDB*)
- Création des pipelines de transfert des données (MySQL -> Elasticsearch) avec *Logstach*
- Conteneurisation et déploiement des différents images avec *Docker*
- Orchestration des pipelines avec *Airflow*

Front-end:

- Création des dashbords statistiques avec *Kibana*
- Affichage du traffic temps réel avec *Dash*

Conclusions et perspectives

Perspectives

Back-end:

- Rajouter les données "*airports_locations*" dans MySQL.
- Automatiser le scrapping des données "*compagnies*" et "*positions*"
- Supprimer les données historiques des vols de MongoDB (selon ressources mémoires)
- Utiliser Elasticsearch (Filebeat) pour analyser les logs des différentes composantes de l'architecture
- Etablir des tests unitaires avec githubAction
- Déployer l'architecture projet dans le cloud (AWS, Google Cloud, Azure)

Front-end:

- Raffiner le filtre du traffic aérien en temps réel tout en filtrant les avions qui ont déjà atterri (On_ground: False -> True).
- Améliorer le visuel de dash en rajoutant plus de choix de filtres (liste des aéroports, date, etc)
- Choix d'une autre API qui scrappe les vols de tout les pays du monde (API payante)

Merci

pour votre attention

Des Questions ?