D

<?xml version="1.0"?><DocumentFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <GraphicCharterDefinitionId>0</GraphicCharterDefinitionId> <TemplateBaseTypeId>0</TemplateBaseTypeId> <CompanyId>1</CompanyId> <ConfidentialId>0</ConfidentialId> <ConfidentialDescription /> <CountryId>0</CountryId> <PageSizeId>1</PageSizeId> <PageOrientationId>1</PageOrientationId> <PrePrintedStationary>false</PrePrintedStationary> <Project>Digiroom - Parking connecté</Project> <Reference>20170725-080726-AM</Reference> <TemplateType>3</TemplateType> <CultureId>fr-FR</CultureId> <LanguageId>2</LanguageId> <Customer /> <DocumentDate>2017-08-04T11:39:18.6523833+02:00</DocumentDate> <Saved>false</Saved> <IsValid>true</IsValid> <FirstPageCover>false</FirstPageCover> <IsNew>false</IsNew> <CurrentVersion>1.00</CurrentVersion> <DocumentType>Dossier</DocumentType> <DocumentTypeId>1</DocumentTypeId> <Entity>127 - Montpellier</Entity> <HasDistributionList>false</HasDistributionList> <HasForeword>false</HasForeword> <Title>Rapport de stage</Title> <Status>1</Status> <StatusDescription>Travail</StatusDescription> <SetEdition>false</SetEdition> <SetVersion>false</SetVersion></DocumentFile>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  | | |  |
|  |  |  | | |  |
|  |  | Rapport de stage | | |  |
|  |  | vendredi 4 août 2017 | | |  |
|  |  | : | | |  |
|  |  |  | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Origine de la mise à jour** | **Rédigée par** | **Validée par** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Sommaire

[1. Remerciements 4](#_Toc489610089)

[2. Introduction 5](#_Toc489610090)

[3. Sopra Steria 6](#_Toc489610091)

[3.1. Le groupe 6](#_Toc489610092)

[3.2. Marché et répartition mondiale 7](#_Toc489610093)

[3.3. La division sud-est 8](#_Toc489610094)

[3.4. L’agence de Montpellier 8](#_Toc489610095)

[4. Le parking connecté 9](#_Toc489610096)

[4.1. Présentation du projet 9](#_Toc489610097)

[4.2. Attentes mission 10](#_Toc489610098)

[4.3. Présentation de l’architecture 11](#_Toc489610099)

[4.3.1. Hardware et IOT : 12](#_Toc489610100)

[4.3.2. Big Data et Machine learning : 12](#_Toc489610101)

[4.3.3. Restitution 13](#_Toc489610102)

[**5.** Présentation détaillée des technologies 15](#_Toc489610103)

[5.1. Elastic Stack 15](#_Toc489610104)

[5.1.1. Présentation de la suite 15](#_Toc489610105)

[5.1.2. Fonctionnement de l’API 18](#_Toc489610106)

[5.2. Explication détaillée du schéma et de la communication entre les outils : 21](#_Toc489610107)

[6. Déroulement du stage 26](#_Toc489610108)

# Remerciements

Je tiens à remercier toute l’équipe pédagogique de Polytech Montpellier, notamment Lysiane BUISSON qui contribue à la réussite de nos stages en créant des relations avec les entreprises, ainsi que Christophe FIORIO qui a été mon tuteur de stage.

Je remercie aussi Chantal DE WIT pour m’avoir permis de réaliser ce stage, Anthony MATINI pour m’avoir encadré ainsi que Morgan BRAULT pour m’avoir aidé à réaliser ma mission dans les meilleures conditions.

Je remercie également Pierre COQUEREL qui était en stage de fin d’études et qui m’a conseillé lors de la rédaction de ce rapport.

Enfin mes remerciements vont à tout le personnel de Sopra Steria pour l’accueil chaleureux auquel j’ai eu droit.

# Introduction

Dans le cadre du stage de 4ème année d’école d’ingénieurs en informatique et gestion, j’ai décidé d’effectuer ce dernier au sein du groupe Sopra Steria Montpellier.

C’est après l’intervention de Chantal De Wit, Anthony MATINI et Yann PUGLIESE à Polytech que j’ai eu envie d’intégrer le groupe. Le sujet du stage qu’ils ont proposé m’a beaucoup intéressé car il faisait intervenir un grand nombre de nouvelles technologies. Habituellement les grandes ESN (Entreprise au Service du Numérique), telles que Sopra Steria, travaillent avec de gros clients (banques, assurances, grands groupes) qui utilisent des technologies « anciennes » qui ont déjà fait leurs preuves. Le sujet proposé permettait à la fois de travailler dans une grande structure et à la fois de travailler sur un projet utilisant des nouvelles technologies. De plus l’idée de pouvoir éventuellement effectuer un contrat de professionnalisation en 5ème année a été une motivation supplémentaire, et cela me permettrait d’avoir une porte d’entrée dans l’entreprise et de pouvoir ensuite y débuter ma carrière. C’est pour cela que j’ai intégré la Digiroom de Sopra Steria Montpellier.

Ce rapport est une synthèse de mon stage de 4ème année. Il présente d’abord l’environnement dans lequel j’ai évolué tout au long du stage, puis les tâches auxquelles j’ai pris part et les actions que j’ai effectuées. Je terminerai par un bilan des compétences acquises au cours de cette expérience.

# Sopra Steria

## Le groupe

Figure 1 : Logo du groupe

Sopra Steria est une entreprise de services du numérique (ESN, anciennement SSII), leader européen proposant un portefeuille d’offres des plus complets : conseil, intégration de systèmes, édition de solutions métier, gestion d’infrastructures informatiques et business process services.



Figure 2 : répartition des métiers en 2015

En 2015, Sopra Steria est le fruit de la fusion de deux entreprises françaises de services numériques Sopra et Steria, créées respectivement en 1968 et 1969. Le groupe compte désormais 38 000 collaborateurs répartis dans plus de 20 pays et réalise en 2014 un chiffre d’affaires de 3,4 milliards d’euros, de 3,6 milliards en 2015 et de 913,2 millions sur le premier trimestre 2016.

## Marché et répartition mondiale

Le marché des services financiers et du secteur public constitue plus de 50% du chiffre d’affaires du groupe, mais on le retrouve également dans les transports, la défense ou l’énergie. Les offres proposées visent essentiellement les grands comptes comme le Crédit Agricole, La Poste ou EDF.



Figure 3 : verticaux stratégiques du groupe

Sopra Steria est implantée dans plus de vingt pays, principalement européens avec la France et le Royaume-Uni, mais aussi en Afrique (Maroc, Caméroun, etc.) et en Asie (Inde et Singapour).



Figure 4 : répartition mondiale

## La division sud-est

Sopra Steria est répartie en France en division. L’agence montpelliéraine fait ainsi partie de la division Sud-Est regroupant quatre sites avec plus d’un millier de collaborateurs.



Figure 5 : la division sud-est

## L’agence de Montpellier

Anciennement dépendante de l’agence d’Aix en Provence Sopra Steria Montpellier est devenue une agence à part entière le 1er janvier 2005 et appartient désormais à la division Sud Est du groupe. Elle s’occupe ainsi de l’activité du bassin languedocien.

Dans une optique de recrutement l’agence de Montpellier compte actuellement un peu plus de 80 collaborateurs et elle regroupe quatre sites différents dont le chiffre d’affaires s’élève à plus de cent millions d’euros.

Faisant partie des agences les plus représentées lors du trophée de l’innovation organisé par Sopra Steria, l’agence de Montpellier s’exerce dans plusieurs spécialités dont le pôle SharePoint\*.

Elle travaille également en collaboration avec l’AMUE\* sur l’un des projets les plus importants de l’agence : le projet Apogée\* qui compte une trentaine de membres répartis en plusieurs équipes.



Figure 6 : organigramme de l’agence

# Le parking connecté

## Présentation du projet

Tout d’abord il est nécessaire de présenter la « Digiroom ». La Digiroom est un concept très récent (moins d’un an) mis en place au sein de l’agence de Montpellier dans le but de promouvoir les nouvelles technologies auprès des collaborateurs dans un esprit ludique et convivial. Le but est de permettre à ces derniers de participer à un projet innovant et de monter en compétences sur des technologies nouvelles et de plus en plus utilisées. Les collaborateurs montent en compétence sur des technologies qui les intéressent, et ces compétences seront valorisées pour obtenir de nouveaux projets. Le lancement de la Digiroom de Montpellier est accompagné de la mise en place d’un projet innovant : le parking connecté. L’objectif est de prédire une place de parking libre selon plusieurs critères (heure, lieu de rendez-vous, tarif, …) définis par l’utilisateur.

Use Case (Cas D’utilisation) :

Pierre est commercial et à rendez-vous avec un de ces clients au « 30 rue Jean Moulin, 69000 Lyon ». Pierre ne connait absolument pas Lyon et il ne sait absolument pas où il va se garer. Le but du parking connecté et de proposer à l’utilisateur LA place de parking qui correspond à ses attentes et qui sera disponible à l’heure de son rendez-vous.

La mise en place de ce parking fait intervenir plusieurs notions nouvelles :

* L’IOT : Internet Of Things, ou objets connectés : analyse de données non disponible sur internet. Exemple : Analyse des caméras de surveillance dans les rues pour identifier les places gratuites disponibles.
* Le Big Data : acquérir, adapter et stocker de la donnée en grande quantité disponible sur internet. Exemples : Évènements culturel, sportif.
* Le machine learning : Mise en place d’un modèle mathématique qui à partir des informations collectées est capable de s’adapter pour augmenter le taux de réussite de la prédiction.

En effet, plusieurs critères (non définis pour le moment) ont vraisemblablement un impact sur l’occupation d’une place de parking :

* Les places à proximités des stades doivent être vraisemblablement plus occupées les soirs de match.
* L’ouverture des soldes
* Un concert de Beyoncé
* Etc…

## Attentes mission

La mission que l’on m’a proposée est de mettre en place l’architecture qui permettra par la suite aux collaborateurs de travailler sur le développement de l’application et d’expérimenter les nouvelles technologies afin d’acquérir des compétences. L’objectif est d’avoir un environnement de travail afin de se focaliser sur le projet en lui-même. Mon rôle était d’éprouver l’architecture proposée par Morgan Brault en mettant en place toutes ses composantes et en les faisant fonctionner sur des exemples simples. J’ai aussi eu pour mission de faire une documentation présentant chaque technologie, leur but et leur fonctionnement, pour que les nouveaux collaborateurs puissent par la suite s’intégrer rapidement dans le projet et contribuer à sa progression tout en apprenant des nouvelles technologies.

Le parking connecté est un projet qui nécessite que plusieurs personnes y travaillent dessus à temps plein pendant plusieurs mois, voire plusieurs années avec la possibilité qu’un tel modèle de prédiction n’existe pas avec les moyens actuels technologiques.

La mission qui m’a été confiée pour mes 2 mois de stage n’a donc pas été de réaliser le parking connecté mais de mettre en place toute l’architecture technique pour aider le projet à débuter. Un choix d’architecture m’a été proposé et j’ai du tout au long de mon stage, valider cette architecture et proposé des solutions alternatives en cas de blocage.

Il m’a donc été demandé de réaliser une fonctionnalité simple mais transverse pour valider l’architecture.

L’installation de chaque « module » technologique était sous ma responsabilité. Je devais écrire au fur et à mesure la documentation d’installation qui sera suivie par les collaborateurs de Sopra Steria lorsqu’ils voudront contribuer sur le projet.

## Présentation de l’architecture

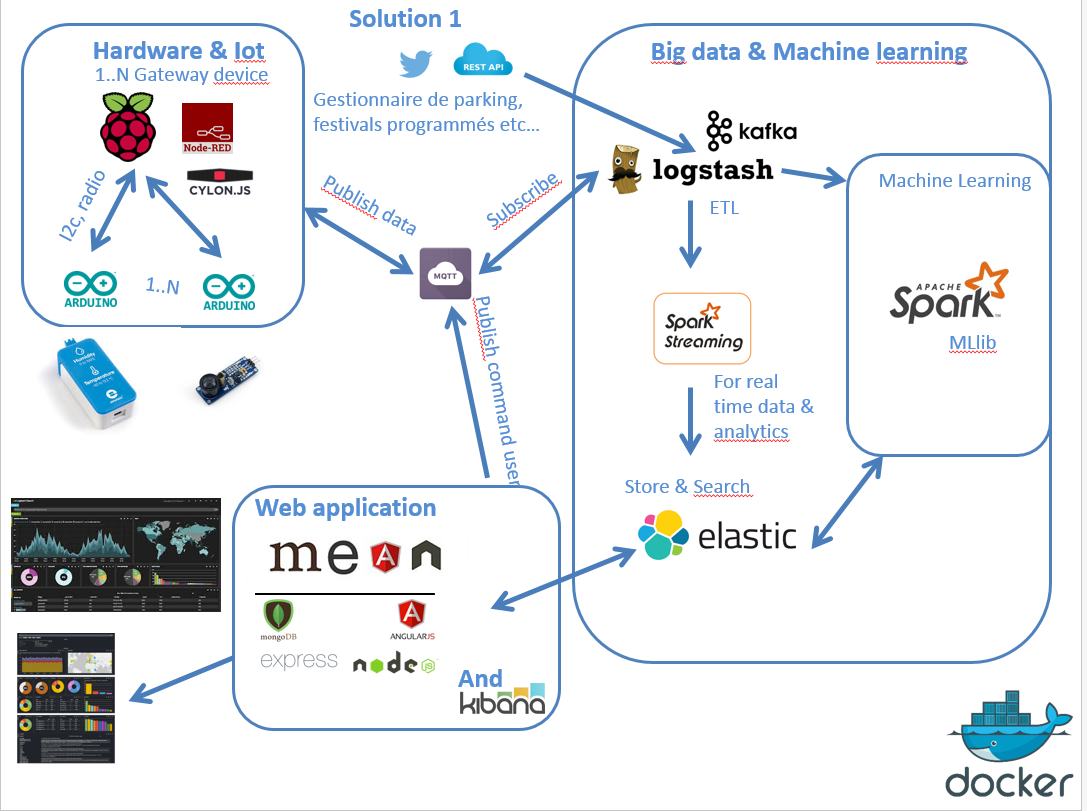
 À mon arrivée, l’architecture avait déjà été définie. Plusieurs technologies avaient été choisies et agencées tel que le montre le schéma ci-dessous qui m’a été fourni :

Figure 5: ????

Cette architecture se divise en trois grandes composantes :

### Hardware et IOT :

Elle regroupe tous les objets connectés et les technologies qui leur permettent d’émettre des données. On utilise des Raspberry Pi 3 et des Arduino avec Cylon.js ou Node RED pour communiquer avec ces derniers. Le protocole de communication avec le reste de l’architecture est MQTT qui est le standard le plus utilisé. On utilise Mosquitto pour envoyer les données.

* Raspberry Pi : c’est un nano-ordinateur qui fait la taille d’une carte de crédit et qui permet d’exécuter des variantes des systèmes d’exploitation GNU/Linux.
* Arduino : C’est un microcontrôleur qui permet de faire fonctionner des objets tels que des capteurs.
* Node RED est un outil logiciel permettant de faire communiquer les objets connectés (i.e. les Arduino et Raspberry). Il est basé sur la technologie node.js.
* Cylon.js : c’est une librairie JavaScript permettant également de faire communiquer les Arduino et les Raspberry.
* MQTT (MQ Telemetry Transport) est un protocole de messagerie basé sur la publication de messages et à la souscription à des flux via le protocole TCP/IP. On publie des messages sur un flux et les souscripteurs de ce flux reçoivent ces messages.
* Mosquitto est un « Broker » MQTT, c’est-à-dire qu’il permet d’envoyer des messages MQTT et de souscrire à des flux.

### Big Data et Machine learning :

Son rôle est de récupérer les informations depuis les objets connectés ainsi que dans les banques de données  existantes (météo, trafic routier, événements, etc.), de les adapter et de les stocker afin de les réutiliser. Kafka, logstash, et Spark Streaming sont des outils permettant de le faire. Le stockage est assuré par Elasticsearch. L’aspect « Machine learning » est assuré par Spark et la librairie MLlib.

* Kafka : cette technologie permet de manipuler des flux de données en temps réel. Par exemple il peut souscrire à un flux de données MQTT (topic) et faire des traitements de données pour les transformer et les rendre disponibles via un nouveau flux.
* Logstash : tout comme kafka logstash permet de gérer des données en temps réel. En plus de cela il est très facile d’intégrer ces données à une base Elasticsearch car ils font partie de la même suite d’outils (voir plus bas)
* Spark Streaming : c’est également un outil d’acquisition de données depuis les flux de données.
* Elastic Search : c’est une base de données no SQL qui permet de faire des recherches efficaces et qui fournit une API permettant d’interagir avec la base via des requêtes http.
* Spark & MLlib : Spark et MLlib sont des outils puissants qui permettent d’exploiter les données en grande quantité et de faire de l’apprentissage pour créer des modèles de prédiction.

### Restitution

Elle prendra la forme d’une application WEB qui utilisera les technologies node js, angular js, mongoDB, et Kibana.

* Node js : Une technologie web basée sur l’évènement et permettant d’avoir une application web robuste pouvant accepter une montée en charge (augmentation du nombre d’utilisations simultanées).
* Angular js : c’est un framework Javascript permettant de gérer dynamiquement la mise en page d’un site web.
* mongoDB : Un Système de Gestion des Bases de Données orientées document et pouvant être réparti sur plusieurs machines.
* Kibana : c’est un outil associé à elasticsearch qui permet de visualiser les données de la base et de réaliser notamment des tableaux de bord.

Chacune de ces technologies sera installée indépendamment grâce à Docker. Docker est un outil de virtualisation qui permet d’installer des logiciels dans des images qui contiennent un système d’exploitation allégé. Les containers sont une sorte de machine virtuelle qui permet d’exécuter les images. Elles démarrent en quelques secondes. On peut trouver des images déjà prêtes pour certaines technologies. Il faut ensuite les lancers pour effectuer les modifications que l’on souhaite, puis enregistrer son état dans une nouvelle image (Commit).

Voici une liste des commandes que j’ai pu utiliser :

* **docker run –it --name kafkacontainer kafkadocker bash**
* docker run : commande qui permet de lancer une image dans un container
* --it : options
* Option i : se connecter aux entrées et sorties standard du container
* Option t : se connecter au docker via un pseudo terminal pour exécuter des commandes
* Option --name kafkacontainer : Option pour donner un nom au container. En l’absence de cette option un nom aléatoire est attribué.
* Kafkadocker : nom de l’image
* Bash : commande à exécuter
* Pour résumer on démarre un container qui s’appellera kafkacontainer et qui exécutera un shell bash sur l’image nommée kafkadocker pour pouvoir par exemple se connecter à un topic MQTT et créer un topic kafka.
* **docker exec -it kafkacontainer bash**
* Docker exec : permet d’executer une commande sur un container en cours d’exécution et sur lequel on n’a pas la main. On peut avoir un aperçu des containers démarrés, de leurs noms et identifiants (hashcode) grâce à la commande **docker ps**
* –it : options
* Option i : se connecter aux entrées et sorties standard du container
* Option t : se connecter au docker via un pseudo terminal pour exécuter des commandes
* Kafkacontainer : nom du container à utiliser
* Bash : commande à exécuter
* Pour résumer on reprend la main sur le container nommé kafkacontainer et on dispose d’un invité de commandes pour pouvoir manipuler le container.
* **docker commit kafkacontainer kafkadocker**
* docker commit : commande qui permet d’enregistrer le container dans une nouvelle image.
* Ici on enregistre l’état du container kafkacontainer, dans l’image kafkadocker. Attention, si l’image existe elle sera remplacée, sinon une nouvelle image sera créée. Le container doit être en fonctionnement pour pouvoir effectuer l’opération « commit »
* **docker ps** : permet d’avoir un aperçu des containers en exécution, leur nom, identifiant temps écoulé depuis le démarrage, depuis le début d’exécution,…
* avec l’option –a on a aussi les containers qui ne sont pas en cours d’exécution, pour pouvoir les supprimer et gagner de la place sur le disque.
* **docker rmi kafkadocker** :
* Permet de supprimer une image
* **docker rm kafkacontainer** :
* Permet de supprimer un container (très utile car les containers peuvent s’accumuler et prendre beaucoup d’espace sur le disque)
* **docker images** :
* Permet de voir toutes les images
* Remarque : on peut utiliser le pipe « | » pour faire une recherche grâce à la commande « grep » : **docker images | grep kafk**
* **docker pull** docker.elastic.co/elasticsearch/elasticsearch:5.5.0
* permet de récupérer une image depuis une url : ici c’est une image contenant elasticsearch (le SGBD)

Afin de simplifier le lancement des containers et la communication entre eux, j’ai utilisé docker-compose. C’est un outil permettant de lancer une application qui repose sur plusieurs containers, grâce à la commande **docker-compose up**. Il utilise un fichier de configuration au format YAML. Ce fichier s’appelle **docker-compose.yml**. On y définit plusieurs informations pour chaque container que l’on veut démarrer :

* **image** : l’image à utiliser
* **container\_name :** le nom du container
* **command :** la commande à exécuter au démarrage
* **ports :** les ports réseaux à exposer aux autres containers
* **links :** les containers accessibles par leur nom (plutôt que par l’adresse ip, qui peut changer d’une exécution à l’autre).

# Présentation détaillée des technologies

## Elastic Stack

### Présentation de la suite

Site Web : <https://www.elastic.co> 07/06/2017

Elasticsearch : <https://www.elastic.co/fr/products/elasticsearch>

Logstash : <https://www.elastic.co/fr/products/logstash>

Kibana : <https://www.elastic.co/fr/products/kibana>

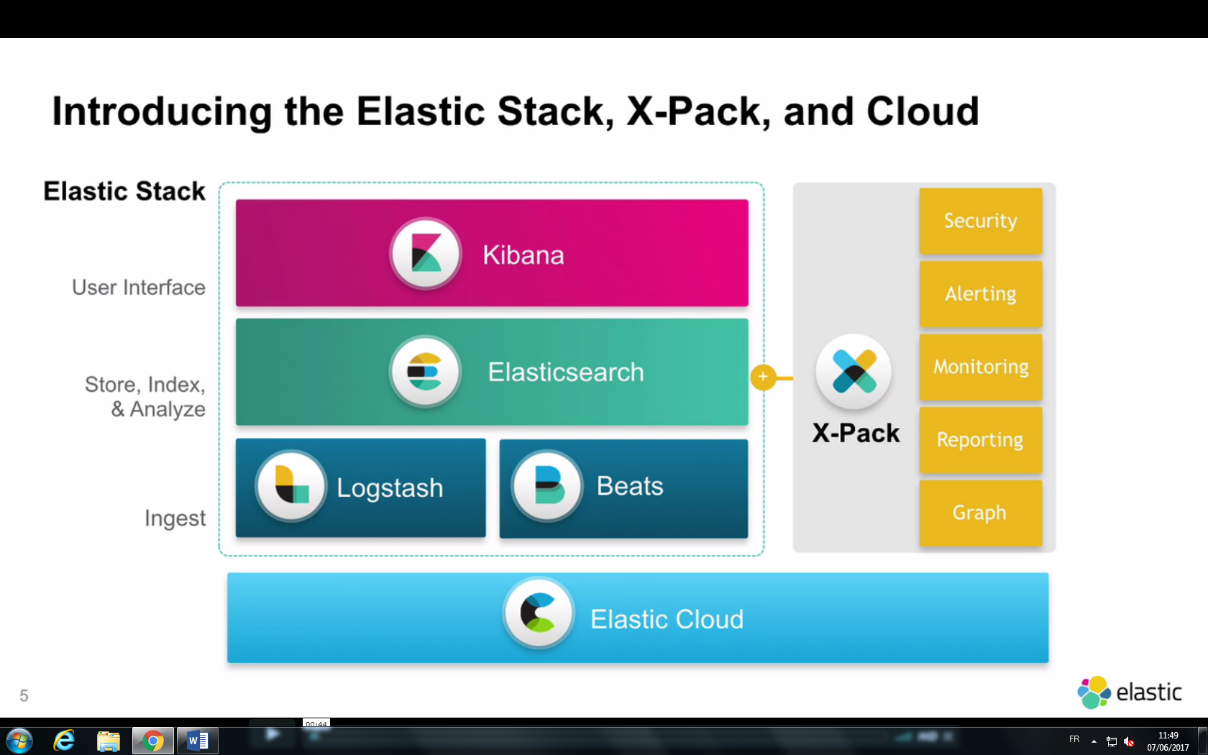
Elastic Stack est une architecture qui permet de récupérer des données (Logstash, Beats), de les stocker, indexer et analyser (Elasticsearch), et de les visualiser (Kibana). Elastic Cloud permet d’héberger le serveur sur le cloud et d’y accéder à distance (par opposition à la version locale) et X-Pack est une bibliothèque de plugins permettant par exemple de créer des graphes ou de sécuriser l’accès aux différents outils.

Figure 6: La suite Elastic (capture vidéo)

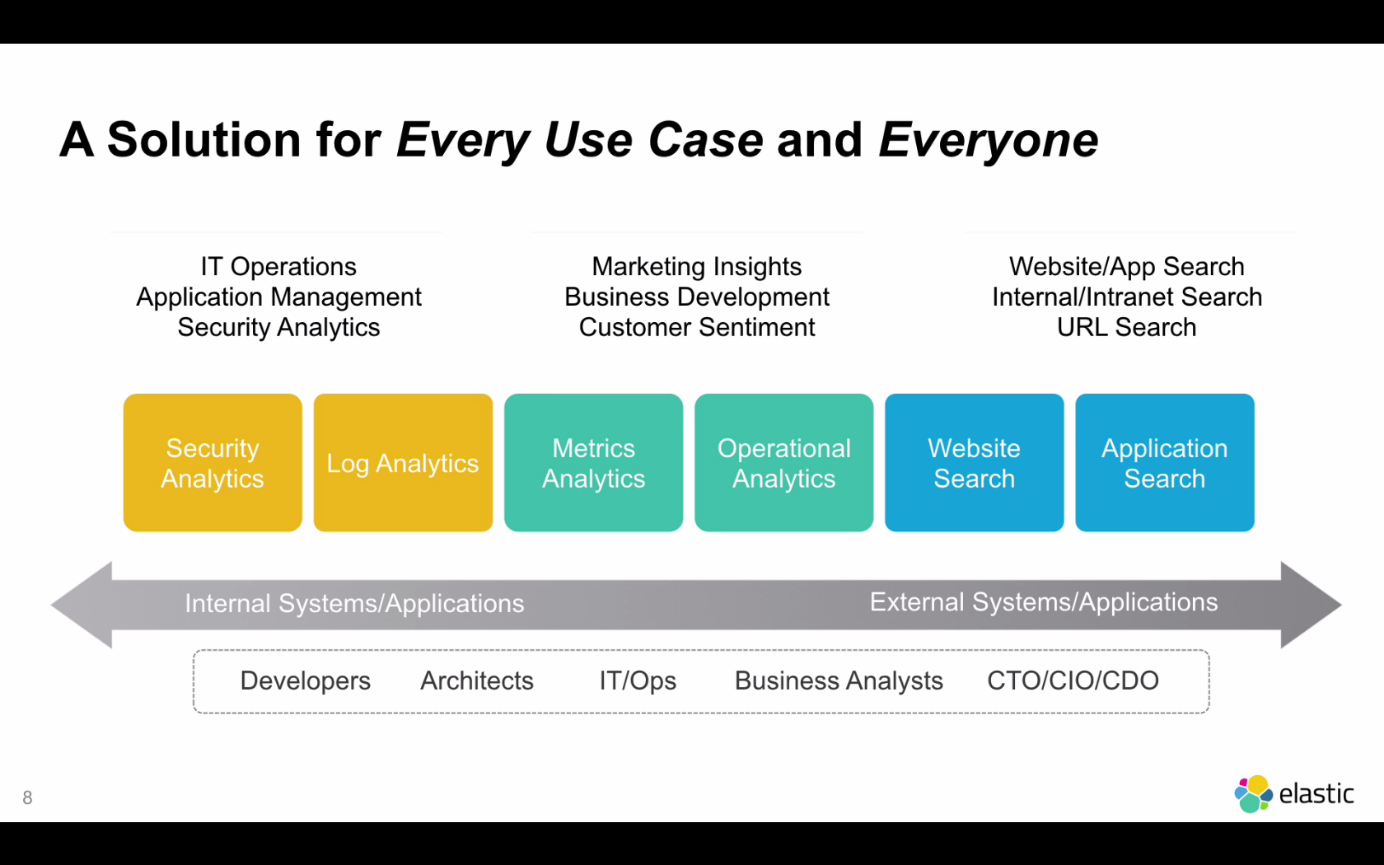


Figure 7: Champ d’utilisation de la suite Elastic

#### Elasticsearch

Au cœur de la suite Elastic, Elasticsearch est un outil de base de données sans schéma et au modèle de données flexible. Il est programmé en java. C’est un serveur d’API sur lequel on fait les requêtes via le protocole HTTP.

Clients :

* PSA Peugeot Citroën : efficience et amélioration des processus
* Orange : création d’un moteur de recherche intégré
* AXA : enrichissement et visualisation des données utilisateur
* Github : accélérer le développement de logiciels

Les avantages mis en avant sont les suivants :

* Scalabilité horizontale : si on veut traiter deux fois plus de données aussi rapidement, il faudra deux fois plus de machines
* Disponibilité en temps réel : les données envoyées au serveur sont disponibles pour les requêtes après une seconde environ
* Modèle de données flexible : ajout de champs sans réindexation
* Exécution rapide des requêtes
* Langage de requête sophistiqué : efficace avec les données structurées (mots-clés) ou numériques par exemple
* Pas de schéma : cela permet de se décharger de la définition d’un schéma qui est fastidieuse. Les champs sont détectés automatiquement ainsi que leurs caractéristiques, et on peut ensuite apporter les corrections nécessaires.
* Une fonctionnalité clé est la géolocalisation que l’on peut affecter à des documents.

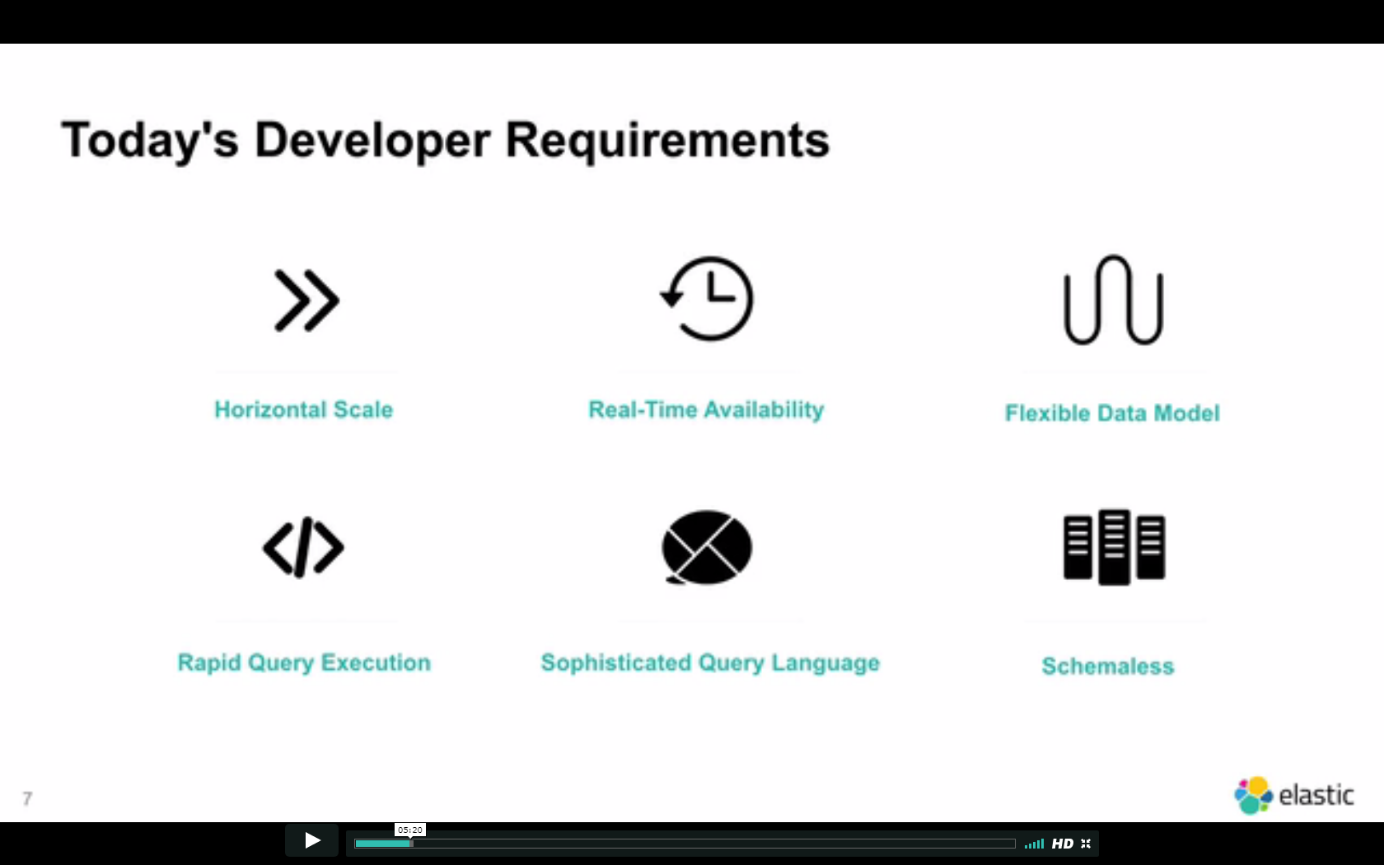


Figure 8: Points forts de Elastic search (Capture vidéo)

#### Logstash

Logstash est l’outil qui collecte les données en continu depuis différentes sources (logs Windows, twitter, Amazon, kafka), les filtre et les envoie vers une base de données (ici Elasticsearch).

#### Kibana

Kibana est un outil de visualisation des données avec des histogrammes, des graphes, des plans sur des tableaux de bord. Il permet aussi de faire les requêtes http et visualiser le résultat au format JSON.

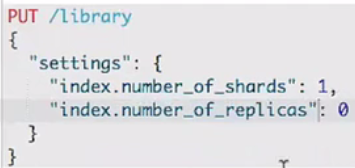


Figure 9: Tableau de bord avec Kibana

### Fonctionnement de l’API

#### Création d’un index

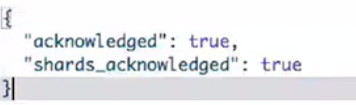
On peut comparer l’index à une table en BD relationnelle, ou à une collection sur les BD NoSQL. A un index peut être associé un mapping (schéma) et de settings (paramètres). Pour créer un index il suffit d’utiliser le verbe http PUT comme suit :



Avec :

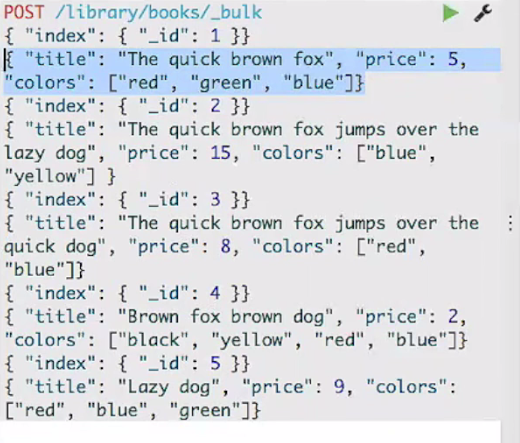
* Number of shards : Les shards (ou primary shards) sont des conteneurs qui permettent de faciliter la recherche des documents. Ces derniers sont répartis sur les shards en fonction de leur ID ou bien sur le même shard que le document parent s’il existe.
* Number of replicas : C’est le nombre de shards répliques (ou replica shards). Un document créé est ensuite créé dans les répliques.

Résultat :



#### \_bulk : Ajout de documents

L’Api \_bulk permet de créer plusieurs documents à la fois. On peut donner un type aux documents (ici « books »). Les documents, qui sont au format JSON, peuvent être précédés d’une ligne qui indique leur ID. On peut laisser le serveur attribuer l’ID en laissant le champ « index » vide.



#### \_search : recherche de documents

L’API \_search permet de faire une recherche sur les documents. On peut chercher par type ou bien sur la totalité d’un index.



Le résultat est le suivant :



En plus des infos de base des documents on trouve pour chaque document un champ « \_score ». Ce score est en fait un indice de pertinence des résultats. Ici ils sont tous égaux à 1 car on n’a pas de critère de recherche.

## Explication détaillée du schéma et de la communication entre les outils :

#### d:\Profiles\aNasraddine\Desktop\rapport\raspi_kafka.pngRaspberry 🡪 Kafka

Mosquitto a été installé sur le raspberry, ce qui lui permet d’envoyer des messages sur un topic (flux). Pour ce faire on utilise la commande **mosquitto\_pub –h localhost –t hello-mqtt –m ‘’hello world ‘’**. Analysons cette dernière :

* **-h localhost** : permet de préciser l’adresse du serveur mosquitto. Pour le moment nous n’avons qu’un raspberry sur lequel est installé le serveur. Plus tard il faudra préciser une adresse ip afin d’envoyer des messages.
* **-t hello-mqtt :** c’est le nom du topic sur lequel on publie les messages.
* **-m ‘’hello world’’ :** c’est le contenu du message.

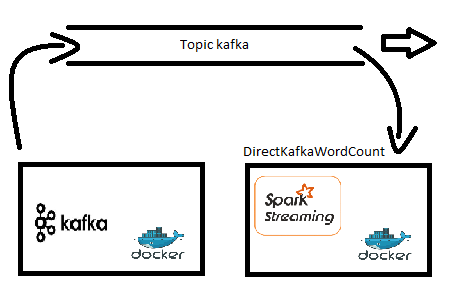
Cela permet au container Kafka de récupérer les messages, en s’abonnant au topic correspondant. Cela est faisable grâce à un plugin qui s’ajoute à Kafka et qui permet de s’abonner aux topics Mqtt. Ce plugin s’appelle **kafka-connect-mqtt.** Ensuite Kafka crée un nouveau topic Kafka dans lequel il transmet les messages. Pour configurer ce système on utilise entre autre le fichier **mqtt.properties** qui contient les informations permettant de s’abonner au topic Mqtt, et de créer un nouveau topic Kafka. Il y a deux autres fichiers de configuration : **zookeeper.properties** et **consumer.propoerties.** Ils permettent de faire fonctionner Kafka. Nous ne nous attarderons pas sur leur description car ils ne concernent pas la communication avec les autres containers.

Revenons au fichier **mqtt.properties.** Il contient plusieurs informations. Voici les plus importantes :

* **connector.class=com.evokly.kafka.connect.mqtt.MqttSourceConnector** : c’est le nom de la classe à utiliser pour se connecter au topic Mqtt.
* **kafka.topic=** hello-kafka: c’est le nom du topic que l’on va créer et qui retransmettra les messages reçus sur le topic Mqtt.
* **mqtt.server\_uris=tcp://mqtt:1883 :** c’est l’adresse du serveur Mqtt. Ici on a précisé le nom du container Mqtt (le lien est fait grâce au champ **link** du fichier **docker-compose.yml).**
* **mqtt.topic=hello-mqtt :** c’est le topic Mqtt auquel on se connecte.

#### Kafka 🡪 Spark Streaming :

Cette étape a été supprimée du schéma, mais il m’a paru judicieux de la présenter car elle est fortement susceptible d’être réintégrée.



Le topic Kafka chargé de transmettre les messages récupérés depuis la Raspberry étant en place, il faut récupérer ces messages avec Spark Streaming, effectuer des traitements et les transmettre sur elasticsearch pour qu’ils soient stockés et réutilisés. Dans notre cas j’ai réussi à trouver une application Spark Streaming qui se connecte au topic Kafka et qui compte les occurrences des mots contenus dans les messages, à intervalle régulier.

J’ai estimé que l’apprentissage du langage Scala qui est le plus recommandé pour Spark me prendrait trop de temps et m’empêcherait de voir d’autres aspects du projet, notamment la partie Hardware (raspberry, Arduino) qui m’intéressait particulièrement.

#### d:\Profiles\aNasraddine\Desktop\rapport\SparkStreaming_Es.pngSpark Streaming 🡪 elasticsearch

La partie Spark Streaming ayant été supprimée, la communication avec Elastic Search n’est plus utile. Elle utilise une fonction **SaveToEs()** qui prend en paramètre le nom de la ressource à enrichir. On l’appelle pour chaque message après avoir fait les prétraitements souhaités.

#### d:\Profiles\aNasraddine\Desktop\rapport\kafka_logstash.pngKafka 🡪logstash

La solution trouvée pour contourner l’utilisation de Spark Streaming est Logstash. On va récupérer les messages depuis le topic Kafka et les envoyer sur Elasticsearch. On peut aussi filtrer les messages. Pour configurer Logstash on crée un fichier de configuration que l’on va indiquer en argument lorsqu’on va lancer Logstash. Ce fichier contient les informations suivantes :

* **zk\_connect => "kafka:2181" :** l’adresse ip et le port pour se connecter au serveur Kafka et s’abonner au topic. Ici on indique directement le nom du container et le port car ils sont connectés.
* **topic\_id => "word-count" :** le nom du topic auquel on s’abonne

#### d:\Profiles\aNasraddine\Desktop\rapport\Logstash_Es.pngLogstash 🡪 elasticsearch

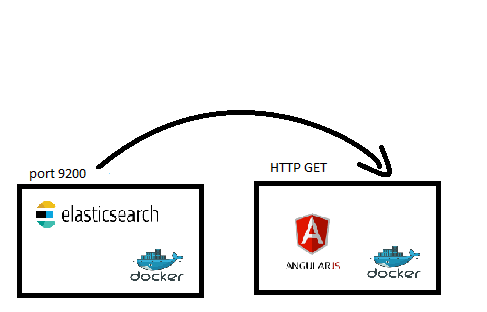
Pour que Logstash stocke les messages dans Elastic search il suffit de préciser dans le champ ‘’output’’ qu’on veut utiliser Elasticsearch. La facilité est due au fait que les deux technologies font partie de la suite Elastic. Ce sera également le cas pour la communication entre Elasticsearch et Kibana pour la visualisation.

#### Elasticsearch 🡪 kibana

#### d:\Profiles\aNasraddine\Desktop\rapport\Ex_Kibana.png

Kibana se connecte au serveur elasticsearch pour récupérer les données et permettre de créer des visualisations, puis des tableaux de bords.

#### Elasticsearch 🡪 Appli web



On pourra effectuer des requêtes HTTP sur le serveur elasticsearch afin de récupérer les données, et les restituer.

# Déroulement du stage

L’objectif étant de tester une architecture afin de la valider ou de la corriger, j’ai décidé de procéder par étapes, c’est-à-dire de mettre en place les technologies deux à deux et de les faire communiquer entre elles.

Il était important de garder une trace du travail effectué afin de le rendre disponible aux autres collaborateurs qui voudront travailler sur le projet par la suite. Pour ce faire, j’ai recensé toutes les actions effectuées lors de l’installation et la mise en marche des technologies, même si j’ai fourni toutes les images (pour Docker) déjà fonctionnelles. Cela permet aux développeurs de suivre ma progression afin de comprendre le fonctionnement de chaque container, mais d’avoir quand même un dernier recours s’ils rencontrent des soucis lors des installations.

Pour organiser le travail j’ai choisi de ne pas planifier toute la période de stage, car toutes les technologies étaient nouvelles et qu’il aurait été très difficile d’estimer les charges en tenant compte des temps d’apprentissages. C’est pour cela que je me suis fixé des objectifs journaliers. On peut considérer que c’est une méthode agile car mes objectifs du jour suivant dépendaient des résultats atteints la veille.