- Rapport de stage –

- Erwann Laplante –

Remerciements :

Je tiens à remercier en premier lieu, mon tuteur M. Laurent GARNIER, expert IHM, de m'avoir conseillé et permis d'avancer tout au long de mon stage.

Je tiens aussi à remercier M. Cédric BERBERA, ingénieur logiciel, pour m'avoir permis et aider à mettre en place mon projet.

De plus, je tiens à remercier M. Khalid AZARINE, référent automatisation, pour avoir mis en place les outils nécessaires à la réalisation de mon projet.

J'exprime aussi ma gratitude à M. Benoit BURIN-DESROZIERS qui a accepté de m'accueillir durant ces six mois dans son entreprise.

Enfin, je remercie aussi ma famille et mes amis qui m'ont soutenu tout au long de la durée de mon stage.

Sommaire :

Tables de matières :

Acronymes :

VM : Virtual machine, en français machine virtuelle

TR : temps réels

IHM ou HMI : Interface homme machine, ou en anglais Human Machine Interface

PDG ou CEO : président-directeur général, ou en anglais Chief Executive Officer

R&D : Recherche et développement

SAV : service après-vente

Liste des annexes :

Introduction :

Ce stage de début de seconde année à Epitech, s'est déroulé à TIAMA, du 4 juillet 2022 au 30 décembre 2022.

Ce stage a permis une découverte du monde de l'entreprise au sein d'une PME mondialement connue, en particulier chez les producteurs de contenant en verre.

Ce rapport de stage contient des informations anonymisées volontairement, ainsi que des données censurées afin de respecter le sujet donné par Epitech : « Votre document écrit doit éviter de transmettre des informations sensibles : vous devez rendre vos documents génériques si nécessaire en supprimant ou en remplaçant les éléments cibles (nom d'un client, informations techniques précieuses, lieu spécifique...). »

I. Le contexte :

TIAMA est l'un des leaders mondiaux sur le marché des solutions de contrôle qualité dédiées à l'industrie des packagings en verre.

TIAMA est un acteur mondial avec 91 % de leurs ventes réalisées à l'export, dont 61 % situées hors-Europe. Ils sont situés dans plus de 80 pays.

L'innovation, le lancement continu de nouveaux produits et les solutions de pointe font pleinement partie du « Business Model » de TIAMA. Cela permet donc à TIAMA de maintenir sa place parmi les leaders mondiaux.

Figure 1 - Emplacement des principaux locaux TIAMA dans le monde

Leurs clients sont des verriers. Un matériau reconnu dans le monde entier comme étant le plus sûr des emballages pour les aliments et les boissons, le verre est à la fois le meilleur choix pour l'environnement et l'allié incontournable de notre santé.

Les machines TIAMA sont disposées tout au long de la ligne de productions afin d'aider les verriers à conserver les contenants sans défauts et à recycler ceux qui en ont. Mais aussi pour les aider au niveau du processus de création, si des bouteilles ont des défauts, cela peut provenir du moule qui s'use par exemple.

Figure 2 - Exemple d'usine équipé par les machines TIAMA (en jaunes)

TIAMA compte 300 salariés, principalement des ingénieurs : un tiers d'entre eux travaillent à la R&D et 40 % se consacrent aux métiers de la Vente et au Service Client.

Le service R&D compte 110 personnes à Saint-Genis-Laval.

A TIAMA, la chaîne hiérarchique au-dessus de moi était constituée de M. Benoit BURIN-DESROZIERS, CEO, supérieur de M. Benjamin COCQUELIN, directeur R&D, supérieur de M. Laurent COSNEAU, directeur adjoint R&D supérieur de M. Laurent Garnier, expert IHM.

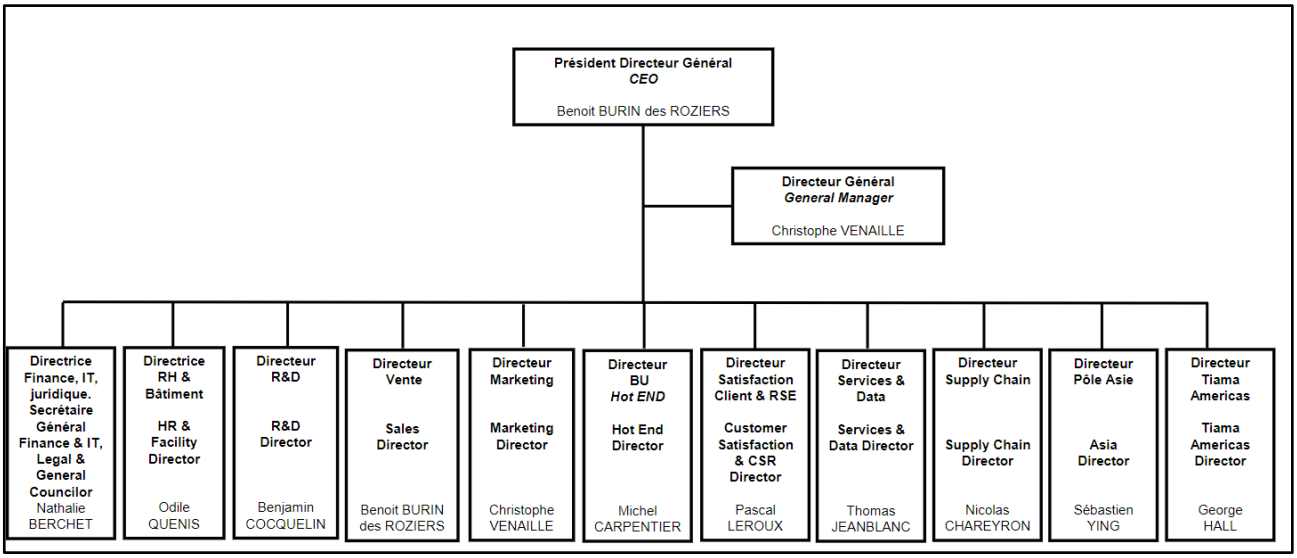

Figure 3 - Organigramme de TIAMA partie 1/3

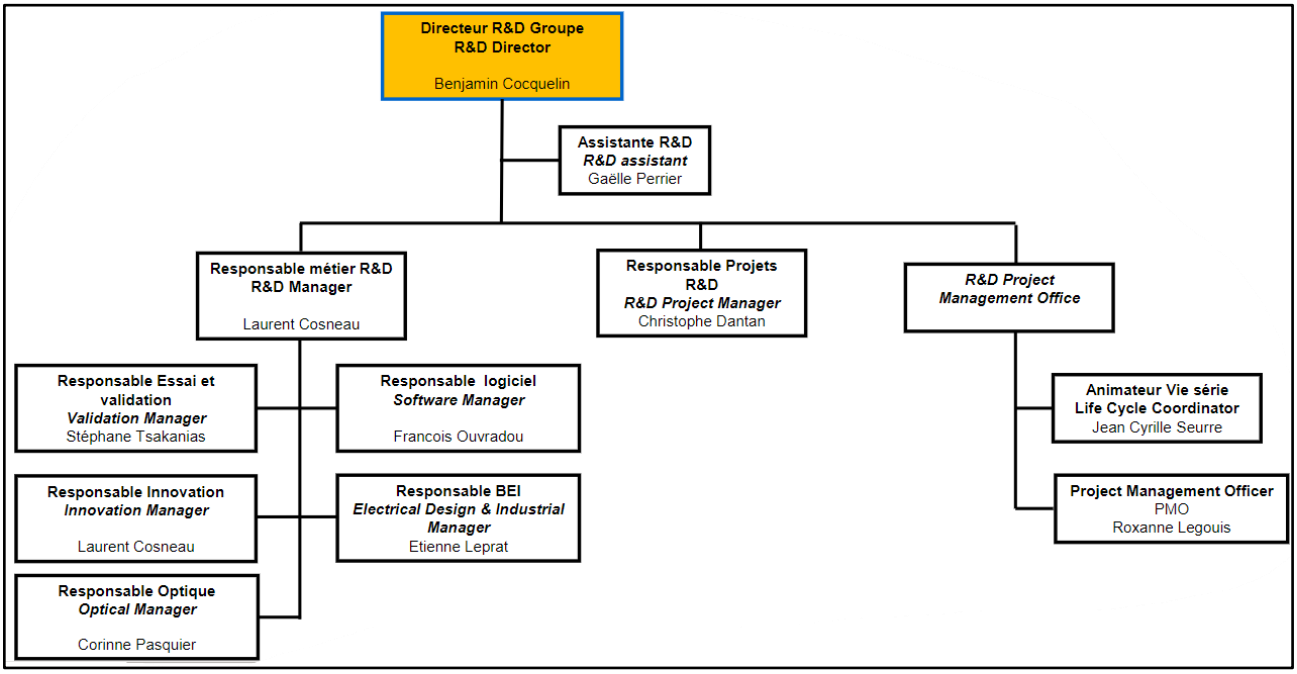

Figure 4 - Organigramme de TIAMA partie 2/3

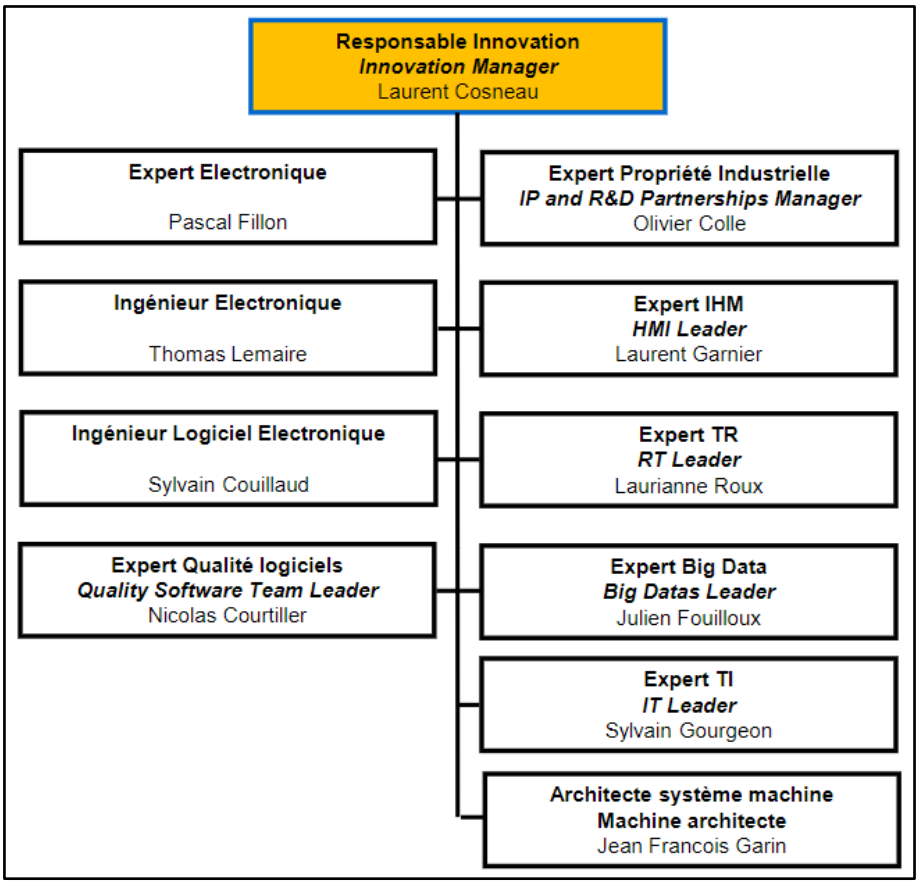

Figure 5 - Organigramme de TIAMA partie 3/3

II. Ma mission

Ma mission se définit comme étant l'étude et à la mise en place d'une plate-forme de monitoring de l'ensemble des micro-services de l'architecture logicielle afin de répondre aux attentes des différents utilisateurs.

Le monitoring consiste à prendre les données, les logs (journal présenté sous forme de fichier contenant les événements survenus dans un logiciel, une application ou un ordinateur), les stocker et à en faire des graphiques, des tableaux, des histogrammes, des diagrammes.

Machine 1	Détaille de l'erreur 1
Machine 2	Détaille de l'erreur 2
Machine 1	Détaille de l'erreur 3
Machine 4	Détaille de l'erreur 4

Figure 6 - Tableau d'exemple de donnée

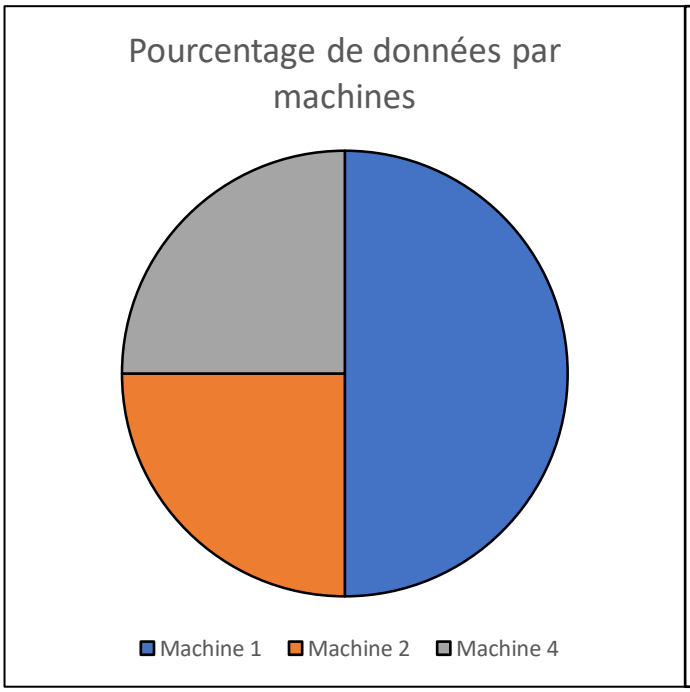

Figure 7 - Diagramme d'exemple de donnée

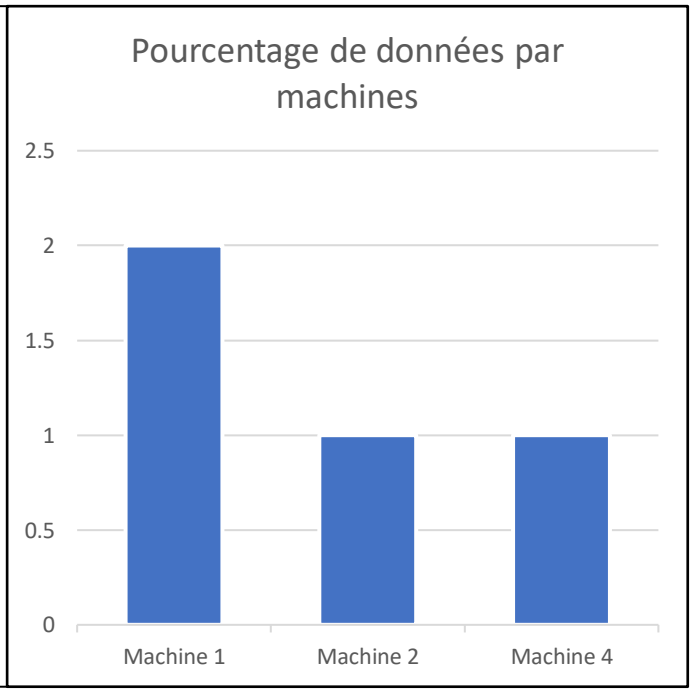

Figure 8 – Histogramme d'exemple de donnée

L'étude de la plateforme de monitoring consiste à regarder les différentes plateformes disponibles sur le marché, à les comparer et à sélectionner celle qui est la plus intéressante pour TIAMA. Selon différents critères comme les Royalties d'utilisation (redevance versée périodiquement par un

franchisé afin de rétribuer les services apportés par le franchiseur durant toute la durée du contrat de franchise liant les deux parties).

Royalties d'utilisation	Limite de stockage	Mise à jour non rétro active	Pas de graphique

Figure 9 - Exemple de critère rédhibitoire pour le choix de la plateforme

Les micro-services de l'architecture logicielle comprennent tout ce qui touche aux machines comme le contrôle, le fonctionnement, la communication et l'affichage IHM. Cela permet d'avoir de la souplesse et de couvrir le fonctionnel avec un maximum de performances et de disponibilité.

Les utilisateurs du système de monitoring sont les équipes R&D de TIAMA afin de voir le comportement des machines à chaque changement dans la programmation. Mais aussi les clients pour leur permettre de voir rapidement si une machine de l'usine a des problèmes Et enfin les ingénieurs SAV lors d'intervention pour agir plus rapidement, plus simplement et en ayant une connaissance plus profonde du problème.

Ainsi, ma mission consiste à rechercher le meilleur moyen de récolter toutes les données des machines, les stocker et en faire des graphiques en fonction des besoins des développeurs et des clients sans impact de performances sur les machines.

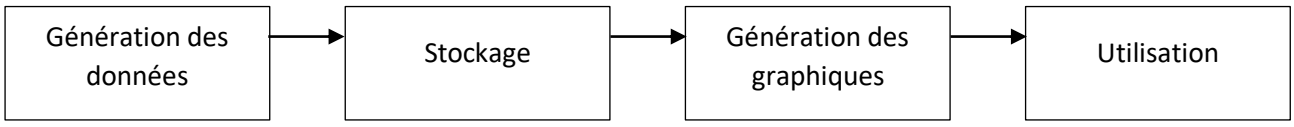

Figure 10 - Chemin simplifié de l'information

III. L’accomplissement de ma mission

1. Compréhension du système de gestion mis en place :

Le système utilisé ne récolte pas les logs, mais uniquement les métriques (mesures quantifiables utilisées en informatique pour faciliter la gestion des activités. Ils servent à comprendre et à savoir ce qu'il se passe dans une machine, une application ou un ordinateur). Les métriques de Windows des machines TIAMA sont récoltés par Windows Exporter, stockés dans Prometheus et visualisés dans Grafana.

Figure 11 - Chemin des métriques

2. Les recherches de base de données :

Mes recherches m'ont amené à utiliser la base de données elasticsearch, car cette base de données permet un stockage horodaté de données non structurées, accompagné de filebeat, qui va chercher les données et les transmet à logstash qui à son tour sépare toutes les données et les incorpore à la base de données.

Ainsi que winlogbeat qui va chercher les données liées à Windows et les stocke directement dans la base de données elasticsearch.

Mais aussi de kibana afin de visualiser, via des graphiques, toutes les données présentes dans elasticsearch.

3. Paramétrage de filebeat :

Le paramétrage de filebeat, c'est fait en 2 étapes, tout d'abord aller chercher les données dans les fichiers souhaités. Et les envoyer dans logstash. Voir la configuration de filebeat en annexe 1, présent sur la page 23.

4. Paramétrage de logstash :

Le paramétrage de logstash, c'est fait de façon qu'il permette une séparation des messages en plusieurs parties.

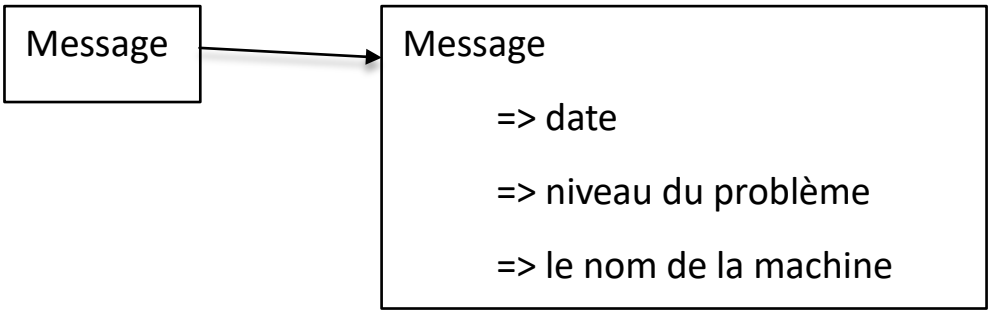

Figure 12 - Schéma de la séparation du message

Puis un envoi du message séparé en de multiples informations dans la base de données d'elasticsearch dans une case horodatée par jour, appelé indice. Voir la configuration de logstash en annexe 2, présent sur la page 24.

5. Paramétrage de winlogbeat :

Le paramétrage de winlogbeat a été semblable à celui de filebeat, car ils ont la même utilité, mais aussi plus facile, car les logs Windows se font forcément au même endroit. Ainsi le paramétrage de winlogbeat a consisté principalement à adapter le paramétrage par défaut pour notre cas et à envoyer les données dans les indices d'elasticsearch, car winlogbeat récupère les logs Windows au format séparé directement. Voir la configuration de winlogbeat en annexe 3, présent sur la page 25.

6. Paramétrage d'elasticsearch :

La récupération des données dans elasticsearch se fait dans des indices (Exemple : pour logstash : logstash-2022.11.03 ou pour winlogbeat : winlogbeat-2-routing). Chaque indice est stocké pendant 15 jours puis supprimé pour permettre un roulement des données. Ce roulement de données est présent pour permettre un affichage rapide de toutes les données stockées (80 000 000 en moyenne sur 15 jours), ce roulement est appelé cycle de vie. Le cycle de vie utilisé pour nos données va faire en sorte que de la création de l'indice à 5 jours stockés, il se trouve en zone chaude, donc charger en premier. De 5 jours à 10 jours, l'indice est en zone tempérée et enfin du 10ème jour à sa suppression l'indice est en zone froide et est donc chargé en dernier. Ainsi lors de l'affichage dans kibana, les données vont arriver en 3 vagues pour faciliter le traitement effectué dans l'affichage. Voir la configuration d'elasticsearch en annexe 4, présent sur la page 26.

7. Paramétrage de kibana :

La base de données elasticsearch va être interrogée par kibana afin de générer des diagrammes, des histogrammes, des conteurs, des tableaux et d'autres formes de graphiques en fonction des filtres appliqués et des dates de débuts et de fin sélectionnées. Voir la configuration de kibana en annexe 5, présent sur la page 27.

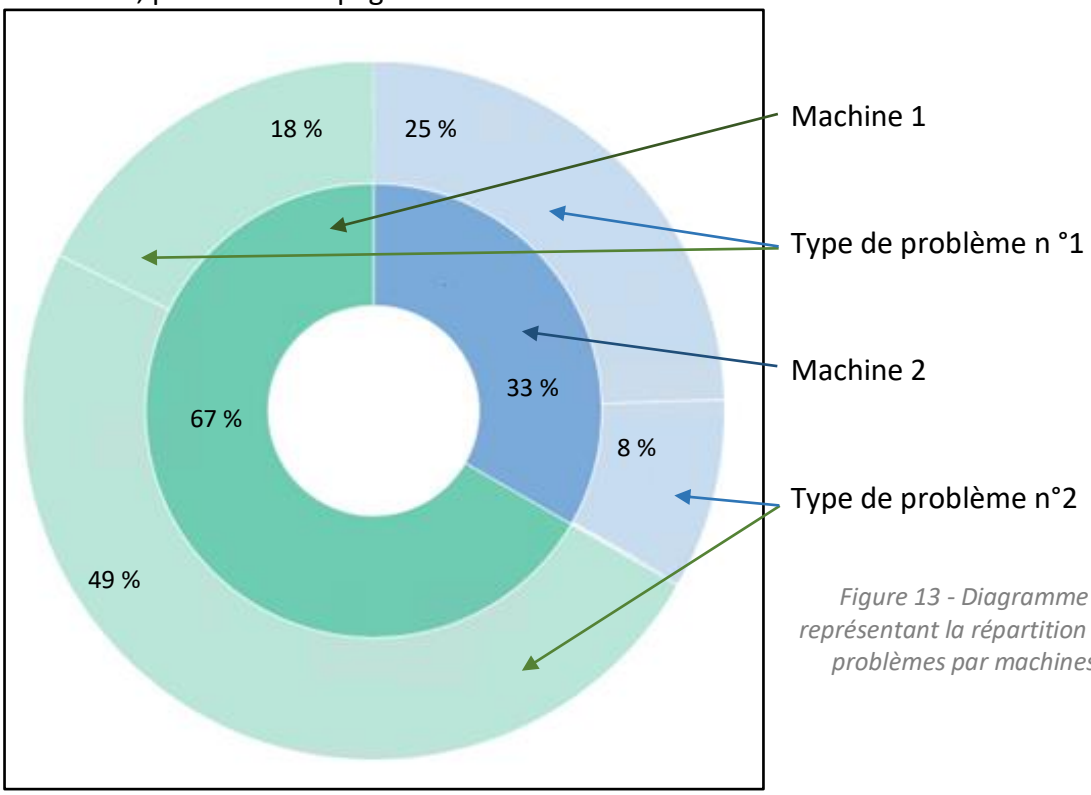

Figure 13 - Diagramme représentant la répartition des problèmes par machines

8. Utilisation de RabbitMQ :

RabbitMQ est une messagerie dans laquelle l'émetteur et le destinataire du message sont séparés par une file d'attente dans laquelle les messages sont stockés temporairement. Il possède une procédure asynchrone, ainsi l'émetteur et le destinataire n'ont pas à agir au même rythme.

Ainsi, l'écriture d'un script python permettant la récupération du nombre d'erreurs et envoyer ce nombre avec le nom et l'IP de la machine correspondant dans RabbitMQ a permis la communication entre la base de données elasticsearch et de la messagerie RabbitMQ.

Il faut savoir qu'en usine aucun serveur mail est présent, mais des serveurs RabbitMQ communiquent avec un serveur RabbitMQ central situé à TIAMA. Donc la communication via RabbitMQ est la seule solution possible pour faire sortir de l'information d'usine.

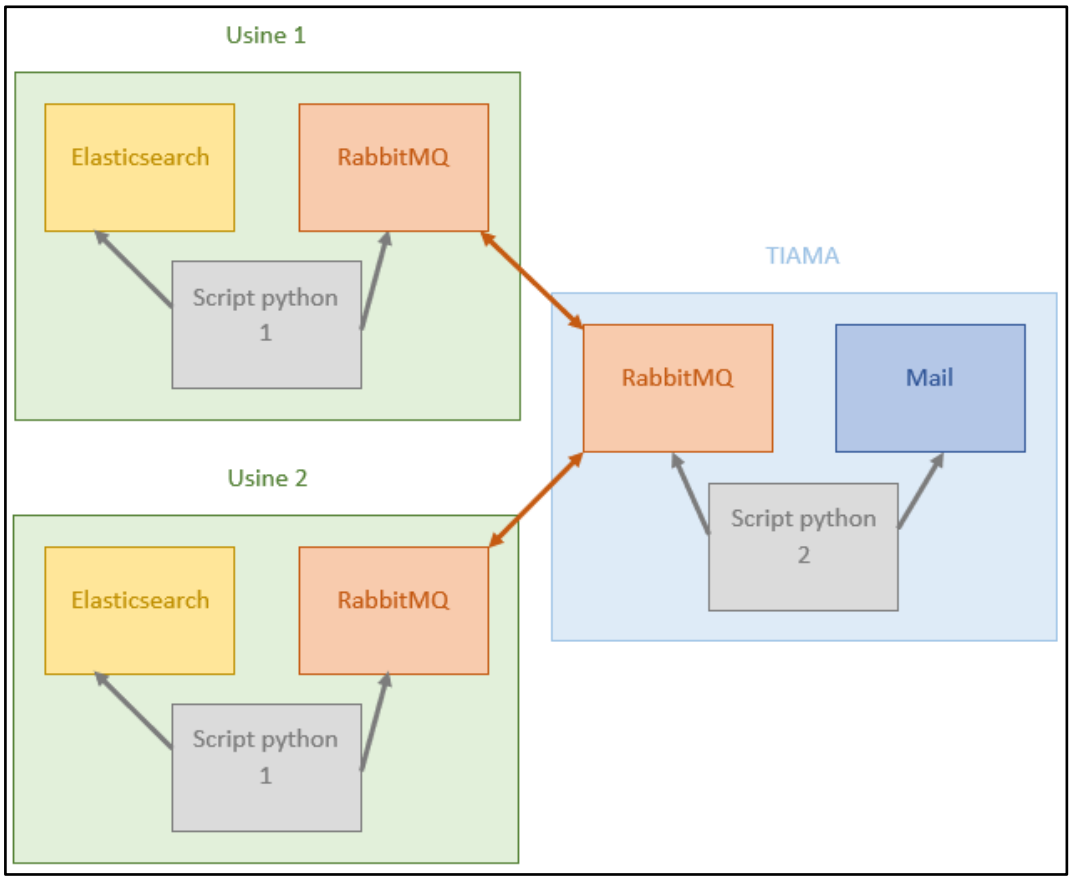

Figure 14 - Schéma du système de messagerie par mail via RabbitMQ

Voir la configuration du script python 1 en annexe 6, présent sur la page 29.

Le message étant arrivé dans le serveur RabbitMQ de TIAMA, l'écriture d'un second script python pour requêter la liste des messages de RabbitMQ envoyés depuis les usines et de les envoyer par mail aux personnes concernées en fonction du type de machines, le type de problème, le nombre de problèmes, etc.

Voir la configuration du script python 2 en annexe 7, présent sur la page 36.

9. Résumé du chemin détaillé de l'information :

Comme vu précédemment, les informations transitent dans plusieurs logiciels et scripts différents. Pour simplifier, le chemin principal de l'information consiste à être créé dans des logs, puis être récupéré par filebeat, à être traité par logstash, puis être stocké dans elasticsearch et enfin être visualisé dans kibana.

La première dérive de ce chemin, consiste à être une entrée annexe dans elasticsearch pour les logs Windows récupérés par winlogbeat.

La seconde dérive du chemin principe, est une sortie annexe des données afin qu'elles soient récupérées au sein de TIAMA, via des scripts python, des serveurs RabbitMQ et l'envoie de mail.

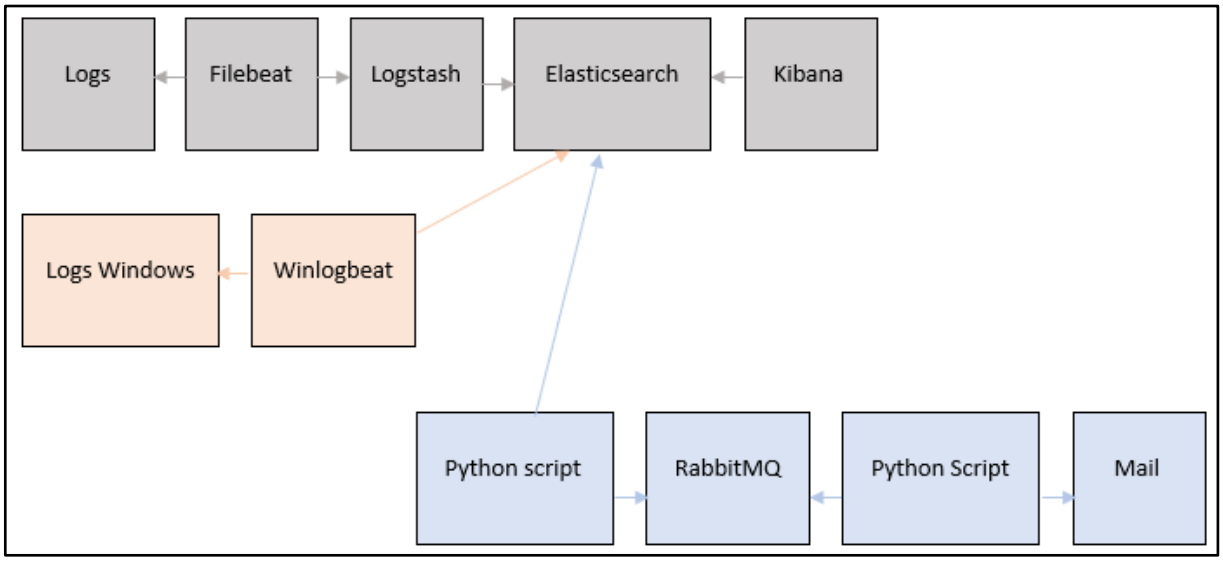

Figure 15 - Chemin détaillé de l'information

IV. Conclusion

Dans un premier temps, j'ai pris connaissance et utilisé la suite ELK (filebeat, winlogbeat, logstash, elasticsearch et kibana) ainsi que la messagerie RabbitMQ.

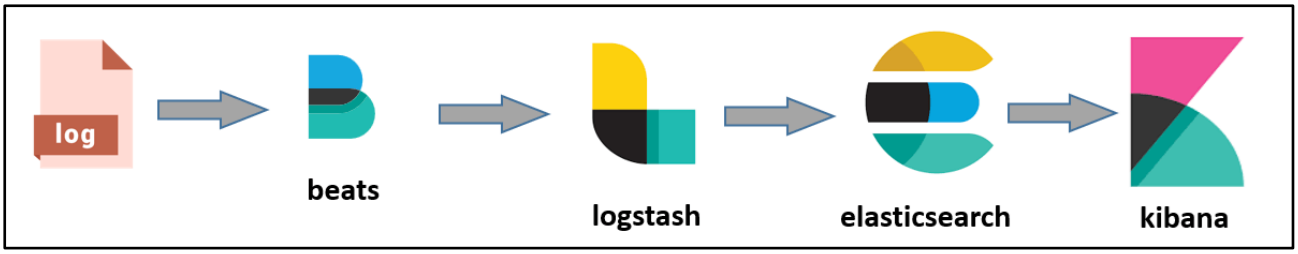

Figure 16 - Schéma simplifié de la suite ELK

Figure 17 - Logo de RabbitMQ

J'ai aussi appris à paramétrer des logiciels, ce qui est très important sachant que les machines de tests ou en usine fonctionnent 24h/24 et 7j/7 à une cadence de 300 à 600 articles par minutes. C'est pourquoi un bon paramétrage prenant tous les cas en compte dès le premier lancement est indispensable pour ne perdre aucune information.

Durant mon stage, la mission que j'ai réalisée permet à toutes les équipes de la R&D de consulter les résultats de leur test fait sur les machines, sous forme de graphiques, de les comparer aux résultats d'autres tests. A contrario, avant la mise en place de mon projet, les résultats des tests étaient stockés dans des fichiers qu'il fallait copier-coller pour ne pas les perdre (à cause des roulements des fichiers). Maintenant, tous les résultats sont au même endroit et gérés automatiquement.

Les tests d'un week-end passent d'un traitement humain d'un jour à un coup d'œil avec une analyse ciblée grâce à sa capacité de stockage et aux graphiques de la suite ELK.

Ce stage, au sein de TIAMA, m'a appris que dans une entreprise, nous ne sommes pas uniquement soumis à notre emploi du temps, mais aussi à celui des autres lorsqu'il s'agit d'attendre des informations par exemple.

- Annexes -

Annexe 1 - filebeat.yml :

```
filebeat.inputs:
- type: log
  paths:
    - "chemin d'accès des données"

setup.template.settings:
  index.number_of_shards: 1

output.logstash:
    hosts: ["IP de logstash : port de réception de logstash"]
```

Annexe 2 - logstash.conf :

```
input {
        beats {
                port => "port de réception de logstash"
        }
}

filter {
        # cherche pattern spécifique dans le message
        dissect {
                mapping => {
                        "message" => "pattern du message spécifique"
                }
        }
        #si echec enregistre dans le format brut
        if "_dissectfailure" in [tags] {
                mapping => {
                        "message" => " pattern par défaut du message"
                }
        }
}

output {
        elasticsearch {
                hosts => ["IP de elasticsearch : port de réception de elasticsearch"]
                index => "logstash-%{+YYYY.MM.dd}"
        }
}
```

Annexe 3 - winlogbeat.yml :

```
winlogbeat.event_logs:
  - name: Application
    ignore_older: 72h
  - name: System
  - name: Security
  - name: Microsoft-Windows-Sysmon/Operational
  - name: Windows PowerShell
    event_id: 400, 403, 600, 800
  - name: Microsoft-Windows-PowerShell/Operational
    event_id: 4103, 4104, 4105, 4106
  - name: ForwardedEvents
    tags: [forwarded]

setup.template.settings:
  index.number_of_shards: 1

output.elasticsearch:
  hosts: ["IP de la machine sur laquelle est elasticsearch : port de réception de elasticsearch"]
  pipeline: "winlogbeat-%{[agent.version]}-routing"
```

Annexe 4 - elasticsearch.yml :

xpack.security.enabled: **false**
xpack.security.enrollment.enabled: **false**
xpack.ml.enabled: **false**
ingest.geoip.downloader.enabled: **false**

logger.org.elastisearch.discovery: niveau de log elasticsearch (DEBUG, INFO, WARNING, ERROR)

indices.lifecycle.poll_interval : temps d'actualisation des cycles de vie des données

Annexe 5 - kibana.yml :

```
xpack.reporting.roles.enabled: false
xpack.data_enhanced.search.sessions.enabled: false
```

Annexe 6 - from_elasticsearch_to_rabbitmq.py :

```
from elasticsearch import Elasticsearch
from datetime import datetime
from dateutil import tz
import pika
import time
import sys


# renvoie uniquement la première ligne de src
def add(src):
    dest = ""
    for i in src:
        if i == '\n':
            return dest
        dest += i
    return dest


# retrouve l'IP de la machine à partir du nom dans le json
def search_ip(src, all):
    dest = ""
    test = "          \"ip\":["[::-1]
    i = 151
    src = src[::-1]
    all = all[::-1]

    # recherche de l'IP en montant ou du début du fichier
    while True:
        if src[i: i + len(test)] == test:
            break

        # si début du fichier recherche IP en descendant
        if src[i:] == "\n{\n    \"_index\":"[::-1]:
            j = 0
            while True:
                if all[j: j + len(test)] == test:
                    j += len(test)

                    # quand IP trouver l'enregistre et la renvoie
```

```
            while True:
                if all[j] == ']':
                    dest += all[j]
                    break
                if all[j] != '\n' and all[j] != ' ':
                    dest += all[j]
                j -= 1
            return dest
        j += 1
    i += 1

  # quand IP trouver l'enregistre et la renvoie
  i += len(test) - 1
  while True:
    if src[i] == ']':
      dest += src[i]
      break
    if src[i] != '\n' and src[i] != ' ':
      dest += src[i]
    i -= 1
  return dest


# trouve IP et nom de la machine
def get_name_ip(str):
  data = str
  new = "\n"
  space = ""

  # transformation du texte en json
  for i in data:
    if i == '}' or i == ']':
      new += "\n"
      space = space[4:]
      new += space
    new += i
    if i == '{' or i == "[":
      new += "\n"
      space += "    "
      new += space
    if i == ',':
      new += "\n"
      new += space
```

```
    new += "\n"
    search = new
    hostname = {}

    # enregistre le nom et l'IP
    for i in range(len(search)):
        if search[i: i + len("          \"hostname\":\"")] == "          \"hostname\":\"":
            message = add(search[i + len("          \"hostname\":\""):])
            message = message[:len(message) - 2]
            if message not in hostname.keys():
                hostname[message] = [1, search_ip(search[:i], search[i:])]
            else:
                hostname[message][0] += 1
    return hostname


# Variables globales
ip_elasticsearch = "IP elasticsearch"
port_elasticsearch = "port elasticsearch"
index_elasticsearch = "indice elasticsearch"
ip_rabbitmq = "IP RabbitMQ"
virtual_host_rabbitmq = "hôte virtuel RabbitMQ"
port_rabbitmq = "port RabbitMQ"
username_rabbitmq = "Nom d'utilisateur RabbitMQ"
password_rabbitmq = "mot de passe RabbitMQ"
exchange_rabbitmq = "transmetteur du message dans RabbitMQ"
routing_key_rabbitmq = "clef de routage dans RabbitMQ"
help_txt = "\nThis script make a request in elasticsearch and send the result to RabbitMQ.\n"
Fuseau = tz.gettz("fuseau horaire de l'utilisateur")


def main():
    global ip_elasticsearch, port_elasticsearch, index_elasticsearch, ip_rabbitmq, virtual_host_rabbitmq,\
        port_rabbitmq, username_rabbitmq, password_rabbitmq, exchange_rabbitmq, routing_key_rabbitmq

    # affiche l'aide
    for i in sys.argv:
        if i == "help" or i == "-h" or i == "--help":
            print(help_txt)
            sys.exit()

    #initialisation des variables
    total = []
```

```
connection_e = True
connection_r = True

# connection à elasticsearch
es = Elasticsearch(hosts="http://" + ip_elasticsearch + ":" + port_elasticsearch)

#tant que programme pas arrêter
while True:

    # Test connection à elasticsearch
    if not es.ping():
        if connection_e is True:
            print("Connection failed to elasticsearch")
        time.sleep(1)
        connection_e = False
    else:
        connection_e = True

        # récupère les données à elasticsearch
        resp = es.search(index=index_elasticsearch, query={"match all": {}},\
            fields=["liste de data que l'on veut requêter"], size=10, source=False, sort={"date": "desc"})

        # récupère les information
        liste = []
        for i in resp['hits']['hits']:
            liste.append(i['fields'])
        liste.reverse()

        # parcours les données
        for i in liste:

            # récupère la date
            date = ""
            for j in i['date']:
                date += j

            # format de date de YYY-MM-DDTHH:mm:ss:SSSZ à YYYY-MM-DD HH:mm:ss.SSSSSS
            annees = 0
            mois = 0
            jours = 0
            heurs = 0
            minutes = 0
            seconds = 0
```

```
        centiemes = 0
        temp = ""
        where = 0
        for k in date:
            if k.isnumeric():
                temp += k
            else:
                if where == 0:
                    annees = int(temp)
                elif where == 1:
                    mois = int(temp)
                elif where == 2:
                    jours = int(temp)
                elif where == 3:
                    heurs = int(temp)
                elif where == 4:
                    minutes = int(temp)
                elif where == 5:
                    seconds = int(temp)
                else:
                    centiemes = int(temp) * 1000
                where += 1
                temp = ""
        dt = datetime(annees, mois, jours, heurs, minutes, seconds, centiemes)
        timestamp = dt.replace(tzinfo=Fuseau).timestamp() + 3600 * 2
        now = datetime.utcfromtimestamp(timestamp)

        # écriture du message
        message = "exemple de message : à x heure et y erreurs reçu"

        # enregistre et affiche le message s'il n'est pas déjà enregistré
        if message not in total:

            # Configuration paramètre de la connexion à RabbitMQ
            params = pika.ConnectionParameters(host=ip_rabbitmq, virtual_host=virtual_host_rabbitmq,\
                port=int(port_rabbitmq), credentials=pika.credentials.PlainCredentials(username_rabbitmq,\
                password_rabbitmq))

            # test connexion à RabbitMQ
            try:

                # connexion à RabbitMQ
                connection = pika.BlockingConnection(params)
```

```
            # ouverture de la discussion
            channel = connection.channel()

            # ouverture de l'échange
            channel.exchange_declare(exchange=exchange_rabbitmq)

            # publication du message
            channel.basic_publish(exchange=exchange_rabbitmq, routing_key=routing_key_rabbitmq,\
              body=message)

            # accusé d'envoi du message
            print("Sent : \"%s\"" % message)
            total.append(message)

            # déconnection à RabbitMQ
            connection.close()
            connection_r = True

        # Message erreur connection RabbitMQ
        except Exception as e:
            if connection_r == True:
                print("failed to connect to RabbitMQ")
            connection_r = False

        # suppression des + 20 derniers messages
        while len(total) >= 20:
            total = total[1:]
    return


# lance le programme et permet de le stopper sans problème
if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        sys.exit()
```

Annexe 7 - from_rabbitmq_to_mail.py :

```
from email.message import EmailMessage
import smtplib
import pika
import sys


# envoie les informations présentent dans RabbitMQ par mail
def send_mail(ch, method, properties, body):

    # split message
    message = str(body)[2:len(str(body)) - 1]
    new = ""
    step = 0
    wait = 0

    # sépare le message pour le mail
    for i in range(len(message)):
        if message[i - len("got"):i] == "got" and step == 0:
            step = 1
            new += " on "
        if message[i - 1] == '{' and step == 1:
            step = 2
        if message[i: i + len(": [")] == ": [" and step == 2:
            step = 3
            wait = 3
            new += " "
        if message[i] == "," and step == 2:
            new += " error(s) at "
            step = 4
        if message[i:i + len("\"ip")] == "\"ip" and step == 4:
            step = 5
        if message[i] == "[" and step == 5:
            step = 6
        if message[i:i + len("']")] == "']" and step == 6:
            step = 7
        if message[i - 1] == "," and step == 7:
            new += " and"
```

```
            step = 2
        if step == 0 or step == 2 or step == 5 or step == 6:
            new += message[i]
        if wait != 0:
            wait -= 1
        if wait == 0 and step == 3:
            step = 2

    # paramètre du mail
    em = EmailMessage()
    em['From'] = from_email
    em['To'] = to_email
    em['Subject'] = title_email
    em.set_content(new)

    # envoie du mail
    with smtplib .SMTP(serveur_email, int(port_email)) as smtp:
        smtp.sendmail(from_email, to_email, em.as_string())
    print("email sent")
    return


# definition de variables
ip_rabbitmq = "IP RabbitMQ"
virtual_host_rabbitmq = "hôte virtuel RabbitMQ"
port_rabbitmq = "port RabbitMQ"
username_rabbitmq = "nom utilisateur RabbitMQ"
password_rabbitmq = "mot de passe RabbitMQ"
exchange_rabbitmq = "transmetteur de message dans RabbitMQ"
routing_key_rabbitmq = "clef de routage dans RabbitMQ"
from_email = "email émettrice"
to_email = "email receveur"
title_email = "titre/objet du mail"
serveur_email = "nom du serveur mail"
port_email = "port du serveur mail"
help_txt = "\nThis script make a request in RabbitMQ and send the result by mail.\n "


def main():
    global ip_rabbitmq, virtual_host_rabbitmq, port_rabbitmq, username_rabbitmq, password_rabbitmq,\
        exchange_rabbitmq, routing_key_rabbitmq, from_email, to_email, title_email, serveur_email,\
        port_email
```

```
# affiche l'aide
for i in sys.argv:
    if i == "help" or i == "-h" or i == "--help":
        print(help_txt)
        sys.exit()

#tant que programme pas arrêter
while True:

    # configuration paramètre de la connection
    params = pika.ConnectionParameters(host=ip_rabbitmq, virtual_host=virtual_host_rabbitmq,\
        port=int(port_rabbitmq), credentials=pika.credentials.PlainCredentials(username_rabbitmq,\
        password_rabbitmq))

    # test connexion à RabbitMQ
    try:

        # connection à RabbitMQ
        connection = pika.BlockingConnection(params)

        # ouverture de la discussion
        channel = connection.channel()

        # ouverture de l'échange
        channel.exchange_declare(exchange=exchange_rabbitmq)

        # connexion à la file
        queue_name = channel.queue_declare(queue='', exclusive=True).method.queue

        # liaison entre échange et files des message
        channel.queue_bind(exchange=exchange_rabbitmq, queue=queue_name,\
            routing_key=routing_key_rabbitmq)

        # accusé de connexion à l'échange
        print('Waiting for logs. To exit press CTRL+C')

        # lecture d'un message avec accusé de réception
        channel.basic_consume(queue=queue_name, on_message_callback=send_mail, auto_ack=True)

        # défilement des messages
        channel.start_consuming()

    # message d'erreur connection RabbitMQ
```

```
        except Exception:
            print("failed to connect to RabbitMQ")
    return


# lance le programme et permet de le stopper sans problème
if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        sys.exit()
```