externalisations des logs des application Talend avec ELK Stack dans un environnement Kubernetes déployé sur Azure

Introduction

Dans le cadre accelerateur de projet talend, dans le partie développement d'APIs RESTful, le monitoring des logs applicatifs est crucial pour assurer la performance, la sécurité, et la disponibilité des services. Talend, en tant qu'outil puissant de traitement des données et de création d'APIs, permet également de gérer et surveiller les logs des systèmes pour une supervision en temps réel. Ce projet vise à intégrer des éléments de logs dans les services API développés avec Talend et à mettre en place un système de monitoring qui permet une analyse continue des logs afin d'identifier les anomalies et d'améliorer la performance.

Problématique

Le suivi des applications en production nécessite une visibilité complète sur leur fonctionnement, en particulier via les fichiers de logs qui fournissent des informations détaillées sur les événements, les erreurs, et les activités des utilisateurs. Le défi réside dans la capacité à capturer ces logs de manière systématique et à les exploiter pour identifier rapidement les problèmes avant qu'ils n'affectent les utilisateurs.

Comment intégrer efficacement la gestion des logs pour des projets Talend pour surveiller les APIs et améliorer leur fiabilité ?

Objectif

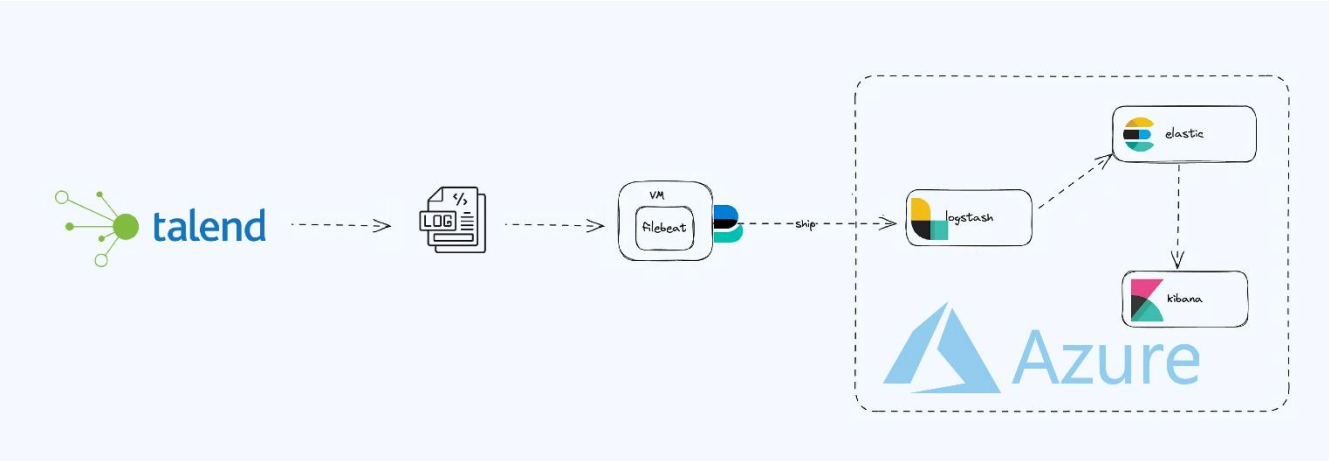
Le projet consiste à mettre en place une solution complète de monitoring pour externaliser et centraliser les logs des applications Talend. Il s'agit d'automatiser le déploiement de la stack ELK (Elasticsearch, Logstash, Kibana) dans un environnement Kubernetes sur Azure via une pipeline GitHub Actions et Terraform. Un agent Filebeat sera configuré pour extraire les logs Talend à partir de Log4j et les transmettre à la stack ELK. L'objectif est de fournir une gestion centralisée des logs et de créer des tableaux de bord interactifs dans Kibana, permettant une supervision efficace, l'analyse des performances et la détection des anomalies en temps réel.

Prérequis

1. Azure CLI
2. Terraform
3. Helm
4. Kubectl
5. Github

Architecture:

Pour visualiser comment les composants s'intègrent, voici un diagramme qui illustre le flux des données de logs depuis les applications Talend jusqu'à l'interface Kibana, en passant par les différents éléments de la stack ELK :



1. Description des Composants:

Filebeat

Filebeat est un agent léger pour l'envoi et la centralisation des données de journaux. Installé comme un agent sur vos serveurs, Filebeat surveille les fichiers journaux ou les emplacements que vous spécifiez, collecte les événements de logs, puis les envoie soit à Elasticsearch soit à Logstash pour être indexés.

Logstash (Port par défaut 5044)

Logstash est un pipeline de traitement de données côté serveur qui ingère des données provenant de plusieurs sources simultanément, les transforme, puis les envoie à elasticsearch.

Elasticsearch (Port par défaut 9200)

Elasticsearch est un moteur de recherche et d'analyse distribué, basé sur REST, capable de répondre à un nombre croissant de cas d'utilisation. Au cœur de la pile Elastic, il stocke vos données de manière centralisée, vous permettant de découvrir ce que vous attendiez et de révéler l'inattendu.

Kibana (Port par défaut 5601)

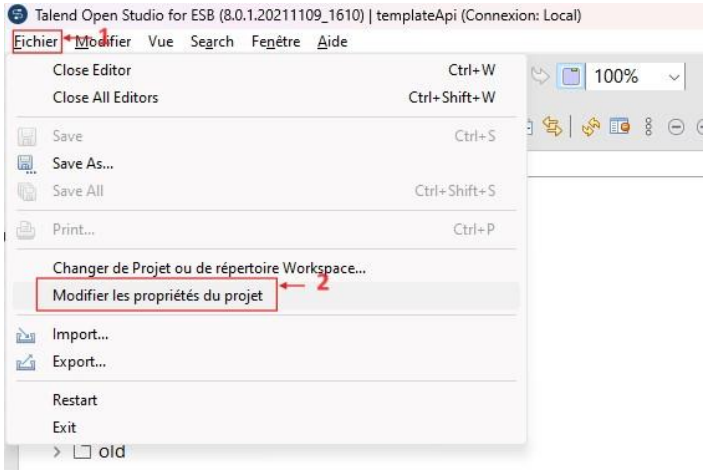
Kibana est un outil de visualisation et d'exploration de données, utilisé pour l'analyse des journaux et des séries temporelles, la surveillance des applications et l'intelligence opérationnelle. Il propose des fonctionnalités puissantes et faciles à utiliser telles que des histogrammes, des graphiques en ligne, des camemberts, des cartes de chaleur et un support géospatial intégré.

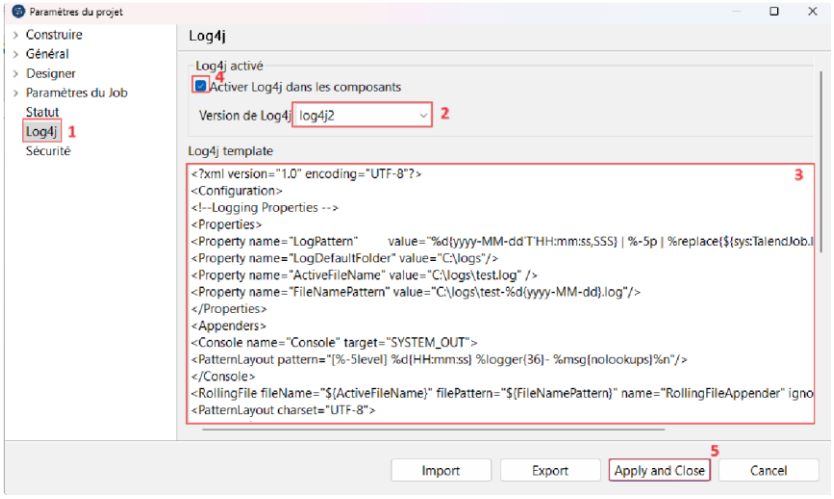
2.Ajout composant de Logs dans Talend avec Log4j

Chaque job Talend a été configuré pour générer des logs à l'aide de Log4j. Cette denier permet une gestion flexible et puissante des logs, avec une configuration qui peut être adaptée aux besoins du projet. Voici les étapes suivies :

- **Configuration de Log4j** : Talend est configuré pour utiliser Log4j comme système de logging. Le fichier `log4j.properties` est ajusté pour définir le format des logs et les destinations des logs…
- **Envoi des logs localement** : Log4j est configuré pour envoyer les logs générés localement sur votre PC. Les logs sont enregistrés dans des fichiers dans un répertoire spécifique (par exemple, `C:/logs/`).
- **Logs capturés** : Les informations suivantes sont capturées par Log4j :
  - **Requêtes HTTP entrantes** (endpoints, paramètres).
  - **Réponses HTTP** (statut, contenu).
  - **Erreurs ou exceptions** rencontrées lors du traitement des données.

Les composants Talend tels que `tWarn` et `tDie` ont été utilisés pour capturer et structurer ces logs. Les fichiers de logs sont générés et stockés localement dans des répertoires surveillés par Filebeat.





code Log4j template :

```
<?xml version="1.0" encoding="UTF-8"? >
<Configuration >
<!--Logging Properties  -->
<Properties >
<Property name="LogPattern"          value="%d{yyyy-MM-dd'T'HH:mm:ss,SSS} | %-5p | %replace{${sys:TalendJob.log    }}
{.log}{}.%c{1} | %m%n"/ >
<Property name="LogDefaultFolder" value="C:\logs"/  >
<Property name="ActiveFileName" value="C:\logs\test.log" /  >
<Property name="FileNamePattern" value="C:\logs\test-%d{yyyy-MM-dd}.log"/  >
</Properties >
<Appenders >
<Console name="Console" target="SYSTEM_OUT" >
<PatternLayout pattern="[%-5level] %d{HH:mm:ss} %logger{36}- %msg{nolookups}%n"/  >
</Console >
<RollingFile fileName="${ActiveFileName}" filePattern="${FileNamePattern}" name="RollingFileAppender"
ignoreExceptions="false">
<PatternLayout charset="UTF-8" >
<pattern>${LogPattern}</pattern >
</PatternLayout >
<Policies >
<TimeBasedTriggeringPolicy interval="1" modulate="true"/  >
<SizeBasedTriggeringPolicy size="10 MB"/ >
</Policies >
<DefaultRolloverStrategy max="20" / >
</RollingFile >
</Appenders >
<Loggers >
<Root level="INFO" >
<AppenderRef ref="Console"/ >
<AppenderRef ref="RollingFileAppender"/ >
</Root></Loggers></Configuration >
```

3.Configurer Filebeat pour collecter ces logs et l'indexation

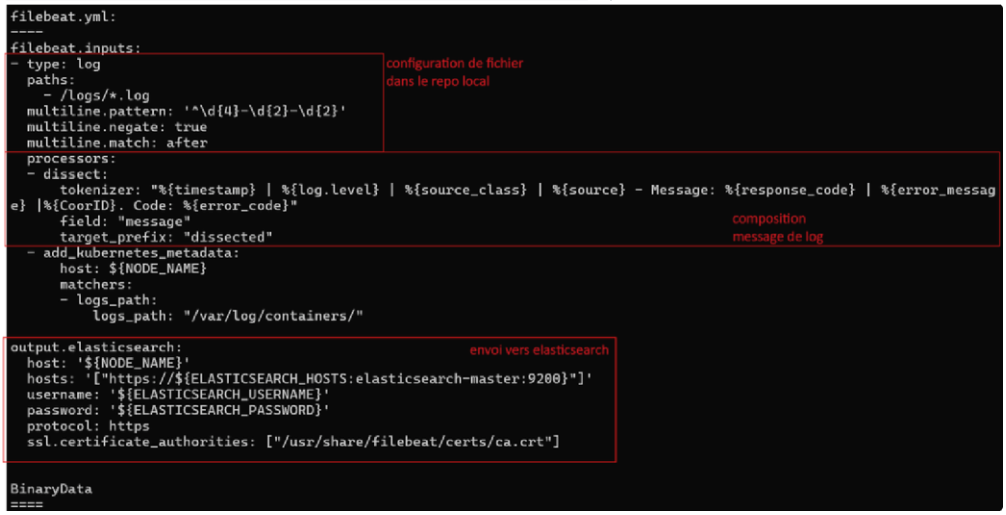
Filebeat est un agent installé sur minikube ou dans un VM . Il est configuré pour surveiller les fichiers de logs générés par Talend et les envoyer vers Elasticsearch ou logstash . Voici les étapes de la configuration :

1. **Configuration des paths de logs** : Le fichier de configuration `filebeat.yml` est ajusté pour surveiller les répertoires contenant les fichiers de logs Talend (par exemple, `C:/logs/`).
2. **Envoi vers Elasticsearch** : Filebeat est configuré pour envoyer les logs directement à elasticsearch, où ils seront indexés et analysés

étape a suivre :

`kubectl edit configmap filebeat-filebeat-daemonset-config` : pour faire l'edit

`kubectl rollout restart daemonset/filebeat-filebeat` : pour restarté filebeat



NB : il faut montée le fichier qui contient les logs avec cette commande : `minikube mount C:\logs:/logs` (C:\logs\test : dans mon pc)

il faut ajouté la configuration de deamonset de fileBeat ces deux partie : `kubectl edit daemonset filebeat-filebeat`

:

`kubectl describe configmap filebeat-filebeat-daemonset-config` : pour voir la configuration

Dans partie volumeMounts :

```
- mountPath: /logs
  name: log-directory
```

dans partie volume :

```
hostPath:
    path: /logs
type: Directory
  name: log-directory
```

restarté fileBeat pour assure les nouvelle config : `kubectl rollout restart daemonset/filebeat-filebeat`

envoi vers Logstash:

```
filebeat.yml:
----
filebeat.inputs:
  - type: log
    paths:
      - /logs/*.log
    multiline.pattern: '^\d{4}-\d{2}-\d{2}'
    multiline.negate: true
    multiline.match: after
    processors:
      - dissect:
          tokenizer: "%{timestamp} | %{log.level} | %{source_class} | %{source} - Message: %{response_code} | %{error_me
ssage} |%{CoorID}. Code: %{error_code}"
          field: "message"
          target_prefix: "dissected"
      - add_kubernetes_metadata:
          host: ${NODE_NAME}
          matchers:
            - logs_path:
                logs_path: "/var/log/containers/"

output.logstash:
  hosts: ["4.178.146.215:5044"]
  #ssl.certificate_authorities: ["/usr/share/filebeat/certs/ca.crt"]
```

3. **Composition des logs**

   Chaque log capturé contient des informations telles que :

   - **Timestamp**.
   - **Gravité de l'événement** (info, warning, error).
   - **Détails sur la requête ou l'erreur**.
   - **Error code**
   - **Error response**
   - **Source class** (GetAll, delete,..)
   - **CorrelationId**

   Elasticsearch permet de rechercher, filtrer, et analyser les logs pour détecter les anomalies et comprendre les causes des problèmes de performance.

4. Déploiement Automatisé avec GitHub Actions et Terraform

- Code Terraform :

Modifier subscription_id dans main.tf line 12:

```
11      provider "azurerm" {
12        subscription_id = "3e46735e-0bec-4d79-b3f7-06e45329de56"
13        features {
```

vm_size : minimum of 8 GB memory, 4 vCPU

VM Size	CPU	Memory (GB)
Default	4	4
Standard_A2_v2	2	4
Standard_A4_v2	4	8
Standard_D2s_v3	2	8
Standard_D4s_v3	4	16
Standard_D8s_v3	8	32
Standard_D16s_v3	16	64
Standard_D32s_v3	32	128
Standard_DS2_v2	2	7
Standard_DS3_v2	2	14
Standard_DS4_v2	8	28

Changer vm_size si neccesaire :

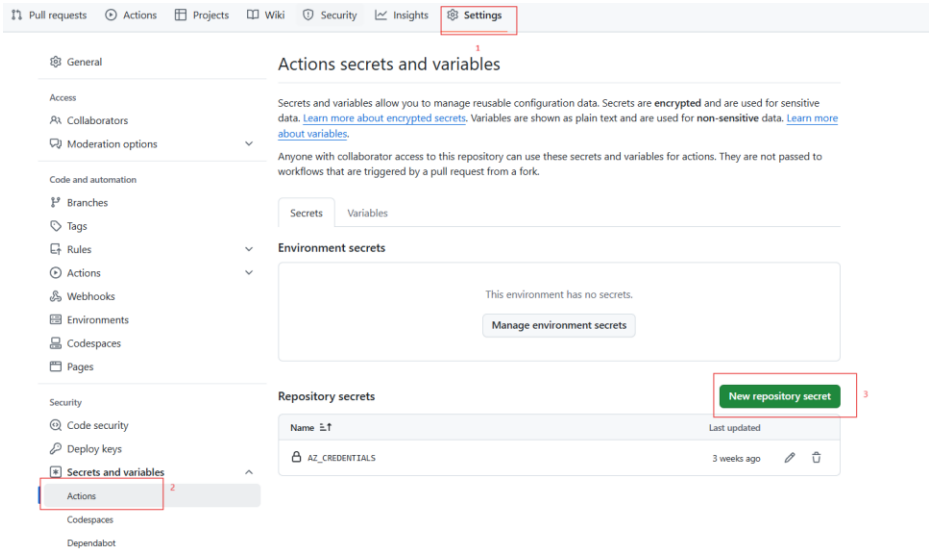
```
30
31      default_node_pool {
32        name       = "default"
33        node_count = 1
34        vm_size    = "Standard_A4_v2"
35      }
```

- **Configuration PWD elasticsearch**

changer le mot de passe dans le fichier **values.yaml** d'Elasticsearch à la ligne **61** et dans le fichier values.yaml de Logstash aux lignes **23 et 79**.

- **Configuration des secrets** :

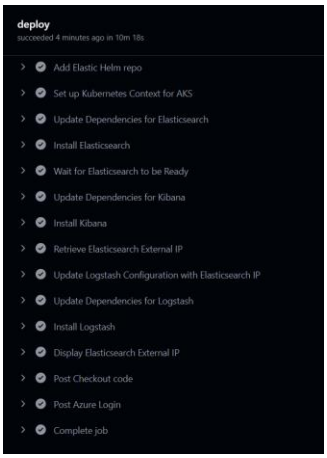
- Ajoutez les secrets Azure dans GitHub :
    - Allez dans votre dépôt GitHub → Settings → Secrets → Actions.



    - Voir secrets avec cette commande : `az ad sp create-for-rbac --name "myServicePrincipal" --role contributor --scopes /subscriptions/{yourSubscriptionId} --sdk-auth`
    - Ajoutez les identifiants sous les noms AZURE_SUBSCRIPTION_ID, AZURE_CLIENT_ID, AZURE_CLIENT_SECRET, AZURE_TENANT_ID, et AZURE_CREDENTIALS… (en JSON pour l'authentification avec azure/login).

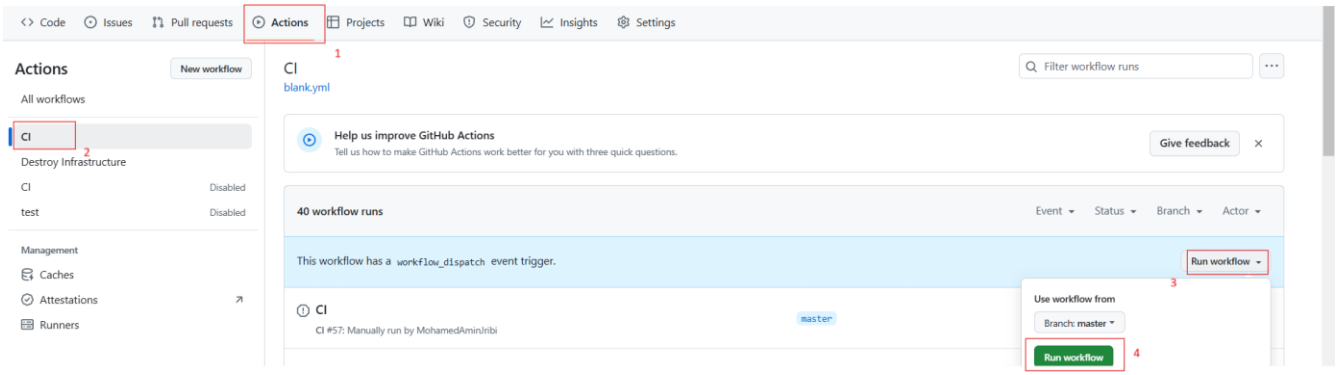
- **Étapes du workflow** :

- **Étape 1** : Récupère le code source avec actions/checkout@v2.
- **Étape 2** : Configure Terraform avec hashicorp/setup-terraform@v1.
- **Étape 3** : Authentifie sur Azure via azure/login@v1.
- **Étape 4** : Exécute Terraform pour déployer l'infrastructure.
- **Étape 5** : Installe Helm puis déploie les composants Elasticsearch, Kibana, et Logstash en utilisant Helm.

- 

-Déclenchement manuel :

Avec `workflow_dispatch`, vous pouvez déclencher le workflow manuellement depuis l'interface GitHub.



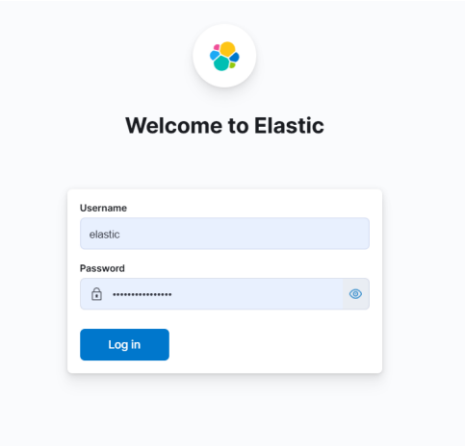
5.Changer le configuration output de Filebeat pour envoyer les logs vers la nouvelle EXTERNAL-IP de Logstash.

`kubectl edit configmap filebeat-filebeat-daemonset-config`

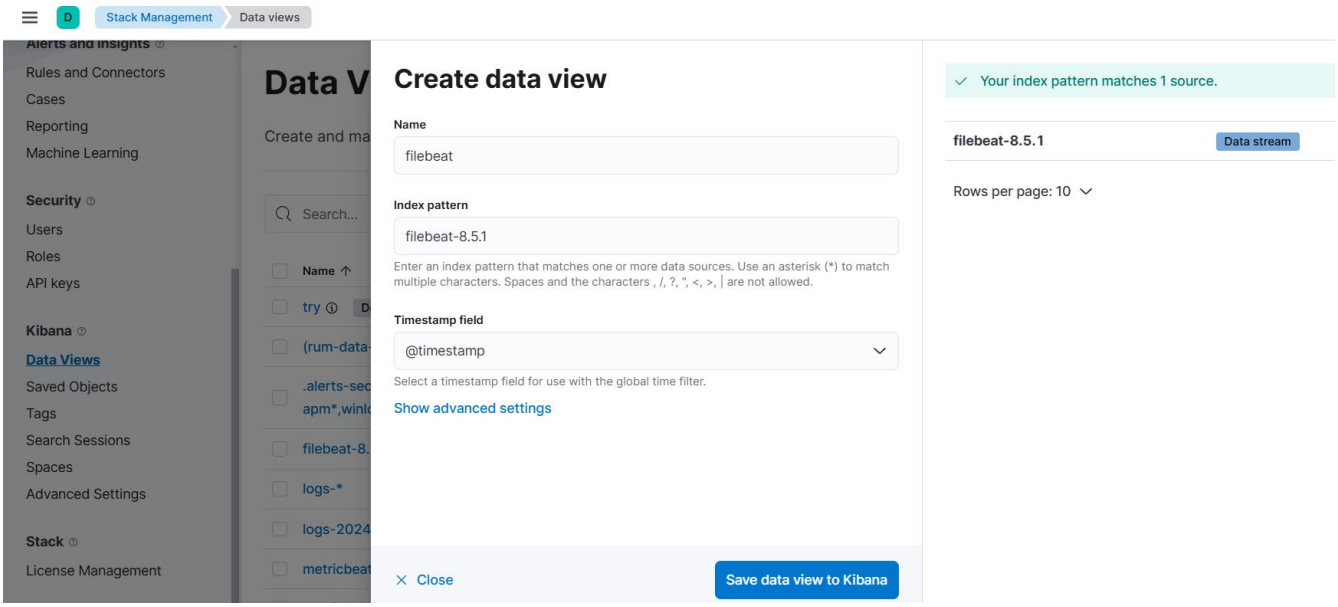
```
filebeat.yml:
----
filebeat.inputs:
  - type: log
    paths:
      - /logs/*.log
    multiline.pattern: '^\d{4}-\d{2}-\d{2}'
    multiline.negate: true
    multiline.match: after
    processors:
      - dissect:
          tokenizer: "%{timestamp} | %{log.level} | %{source_class} | %{source} - Message: %{response_code} | %{error_me
ssage} |%{CoorID}. Code: %{error_code}"
          field: "message"
          target_prefix: "dissected"
      - add_kubernetes_metadata:
          host: ${NODE_NAME}
          matchers:
            - logs_path:
                logs_path: "/var/log/containers/"

output.logstash:
  hosts: ["4.178.146.215:5044"]
  #ssl.certificate_authorities: ["/usr/share/filebeat/certs/ca.crt"]
```

6.Visualisation et Monitoring avec Kibana

Kibana, l'interface de visualisation d'Elasticsearch, est utilisée pour explorer les logs et configurer des dashboards de surveillance.



-Creation de Data View : Stack Management → Data view →Create Data View



-Tu peux maintenant voir les logs en temps reel et faire le dashboard: