Documentation de la Plateforme

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Mis à jour par | Révisé par |
| 1.0 | 26/06/2018 | Philippe Mondon |  |
| 1.5 | 17/07/2018 | Philippe Mondon |  |

[Description de la Plateforme 2](#_Toc519610066)

[INGESTER\_NODE 2](#_Toc519610067)

[Structure d’un environnement fournisseur de données 3](#_Toc519610068)

[Normalisation des Indexes et Pipelines 5](#_Toc519610069)

[Indexes de données - Plateforme de développement 5](#_Toc519610070)

[Indexes Templates de données - Plateforme de développement 5](#_Toc519610071)

[Ingest Node Pipelines Simulate – Plateforme de développement 5](#_Toc519610072)

[Ingest Node Pipelines – Plateforme de développement 5](#_Toc519610073)

[Indexes de monitoring – Plateforme de développement 5](#_Toc519610074)

[Inventaire des services 6](#_Toc519610075)

[Serveur d’Ingestion 6](#_Toc519610076)

[Description technique des outils de monitoring et d’administration 7](#_Toc519610077)

[Monitoring System 7](#_Toc519610078)

[Monitoring : Recherche d’un processus filebeat particulier 7](#_Toc519610079)

[Processus de mise en place du monitoring de processus 7](#_Toc519610080)

[Monitoring : mise en place d’un watcher (alerte) sur un processus 7](#_Toc519610081)

[Monitoring de Qualité de données 8](#_Toc519610082)

[Duplication d’enregistrements 8](#_Toc519610083)

[Contrôle du nombre d’enregistrements intégrés 8](#_Toc519610084)

[Service associé à la duplication d’enregistrements 9](#_Toc519610085)

[Archivage des fichiers 9](#_Toc519610086)

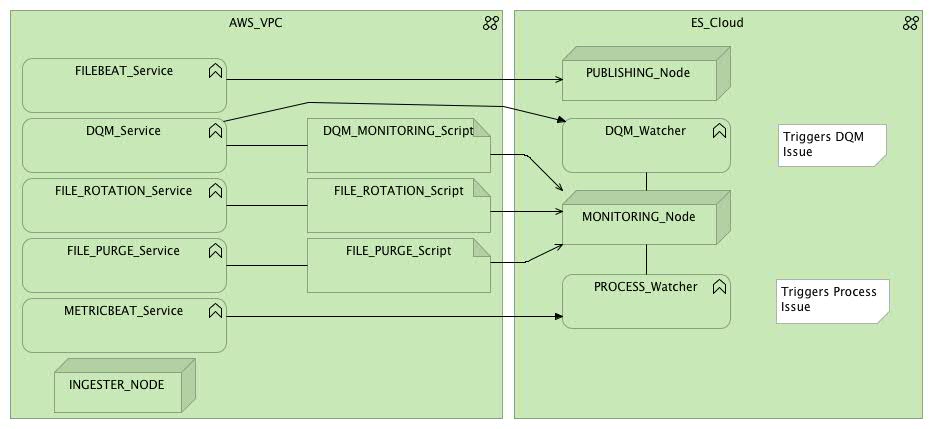
[REFERENCES 10](#_Toc519610087)

[Lignes de commandes 10](#_Toc519610088)

[Filebeat : Lancement de l’ingester 10](#_Toc519610089)

[Filebeat : contrôle du service Systemd – Exemple sur la plateforme d’intégration 10](#_Toc519610090)

# Description de la Plateforme



La plateforme est composée de 3 environnements :

* INGESTER\_node :
  + En charge de la collecte des données injectées par les fournisseurs (MNC, SCS, SDS),
  + En charge de l’injection des données de vente à travers le FILEBEAT\_Service,
  + En charge de l’hébergement de divers services de monitoring : hôte de l’agent de monitoring système METRICBEAT\_Service,
  + Hébergé à terme dans un environnement AWS.
* MONITORING\_Node :
  + Hôte des différents indexes de monitoring,
  + Hôte des watchers (alertes mail pré-configurées),
  + Hébergé à terme dans ES\_Cloud.
* PUBLISHING\_Node :
  + Hôte des indexes des données de vente,
  + Hébergée à terme dans ES\_Cloud.

## INGESTER\_NODE

Machine linux hôte de la collecte et de l’ingestion des données de vente pour les 3 fournisseurs identifiés.

Hôte de divers services de monitoring et de services de gestion de fichiers de données.

Ainsi, pour chaque fournisseur de données sont configurés :

* Un service Filebeat d’ingestion de données,
* Un service FILE\_ROTATION : pour l’archivage des fichiers ingérés,
* Un service FILE\_PURGE : pour la **purge éventuelle** des fichiers archivés,
* Un service DQM : service de qualité de donnée en charge du contrôle de cohérence du pipeline d’ingestion,

En complément, on doit installer :

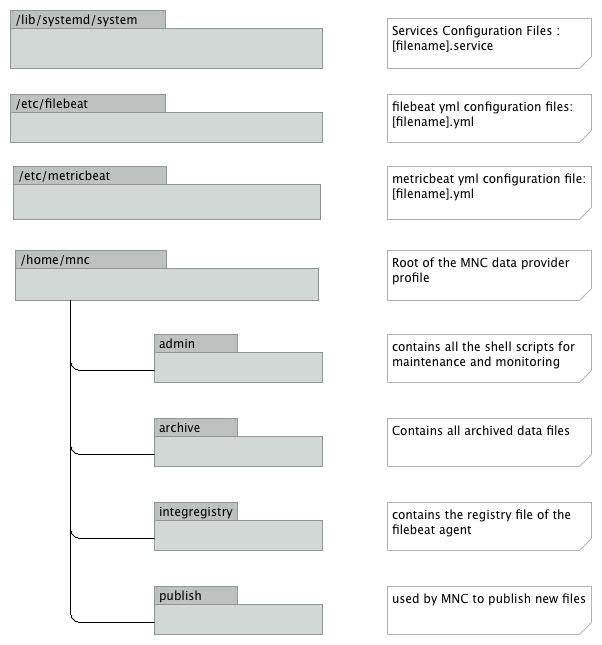
* La librairie « jq » pour parser JSon dans les services FILE\_ROTATION, FILE\_PURGE, DQM,
* Python3 pour pouvoir utiliser éventuellement Ansible.

Attention à installer si nécessaire un client NTP et à positionner la timezone dès que possible : par défaut de nombreux serveurs sont positionnés en UTC.

### Structure d’un environnement fournisseur de données

Chaque fournisseur de données se voit affecter un « file système », un user et des droits appropriés. Nous prendrons comme exemple le fournisseur MNC.

#### File System du User MNC



##### Filesystems intégrés au profil du fournisseur de données

Chaque dossier **home** de fournisseur de données disposera :

* D’un dossier **publish** pour pousser les fichiers de données à ingérer,
* D’un dossier **archive** pour pour archiver les fichiers ingérés,
* D’un dossier de **registre** qui héberge l’index filebeat qui conserve l’état d’ingestion des fichiers (peut être détruit en cas de besoin de ré ingestion de fichiers) 🡪 **integregistry** dans notre exemple,
* **Passer tous les filesystems en root sauf publish**

##### Filesystems communs

Ils hébergent les fichiers de configuration système et les dossiers des agents :

* **/etc/systemd/system** :
* **/lib/systemd/system** : héberge les fichiers **.service** des services system.
  + Tous les agents sont exécutés par des services.
  + Tous les scripts d’administration sont exécutés par des services.
  + Chaque service est **systématiquement spécialisé par profil de fournisseur de données (e.g MNC, SDS, etc.)**
* **/etc/filebeat** : héberge les fichiers de configuration filebeat au format yml.
  + Une **configuration spécialisée** devra être créée pour **chaque profil de fournisseur de données**.

#### Rôle et droits du User « fournisseur de données » : exemple MNC

* Le user MNC peut se connecter à l’INGESTER\_NODE grâce à une connexion SSH « password-less » (basée sur un certificat),
* Ce user peut poster toutes les 15 minutes ses fichiers de données

# Normalisation des Indexes et Pipelines

## Indexes de données - Plateforme de développement

*Nom des indexes stockés dans la plateforme*

Index MNC : tpgbam \_prd\_mnc \_index

Index SCS : tpgbam \_prd\_scs \_index

Index SDS : tpgbam \_prd\_sds\_index

Tous les indexes utilisent le préfix « tpgbam\_prd » : il permettra dans Kibana de créer un index pattern «  « tpgbam\_prd\* » et de ce fait de traiter ces indexes dans les dashboards de façon unifiée.

## Indexes Templates de données - Plateforme de développement

*Nom des index templates stockés dans la plateforme*

Index Template MNC : tpgbam\_mnc\_prd\_index\_template

Index Template SCS : tpgbam\_scs\_prd\_index\_template

Index Template SDS : tpgbam\_sds\_prd\_index\_template

## Ingest Node Pipelines Simulate – Plateforme de développement

*Nom du fichier stockant le pipeline simulate.*

*ATTENTION : aucun pipeline simulate n’est stocké dans la plateforme. La définition contenue dans ces fichiers est utilisée que dans l’environnement de développement*

Pipeline Simulate MNC : tpgbam\_mnc\_prd\_pipeline\_simulate.json

Pipeline Simulate SCS : tpgbam\_scs\_prd\_pipeline\_simulate.json

Pipeline Simulate SDS : tpgbam\_sds\_prd\_pipeline\_simulate.json

## Ingest Node Pipelines – Plateforme de développement

Pipeline MNC : tpgbam\_mnc\_prd\_pipeline

Pipeline SCS : tpgbam\_scs\_prd\_pipeline

Pipeline SDS : tpgbam\_sds\_prd\_pipeline

## Indexes de monitoring – Plateforme de développement

User de monitoring :

Username : tpgbam\_monitoring\_dev\_user

Password : monitoringdev$1

Role : logstash\_system

Monitoring Notes :

For Elasticsearch 5.x clusters: Monitoring data is retained for three days by default or as specified by the xpack.monitoring.history.duration user setting.

# Inventaire des services

## Serveur d’Ingestion

Ce serveur intègre :

* Les services d’ingestion de fichier 🡪 spécialisés par profil de fournisseur de données,
* Les services de qualité de données 🡪 spécialisés par profil de fournisseur de données,
* Les services de rotation de fichiers (archivage) 🡪 spécialisés par profil de fournisseur de données.

Le profil du fournisseur MNC sera utilisé comme exemple :

|  |  |  |
| --- | --- | --- |
| **Nom** | **Path** | **Description** |
| filebeat.mnc.integ | /lib/systemd/system/filebeat.mnc.integ.service | Service de gestion de l’agent d’ingestion de données **filebeat.** |
| metricbeat | /lib/systemd/system/metricbeat.service | Service de gestion de l’agent de monitoring système. Capture un grand nombre d’indicateurs dont tout ce qui est lié aux processus (consommation CPU, consommation mémoire, nbr de fichiers ouverts, etc.).  **A lancer de préférence en root** |
| monitoring.mnc.ingester.data | /etc/systemd/system/monitoring.mnc.ingester.data.service | Service de monitoring/contrôle qualité de l’ingestion de données. |
| data.ingestion.mnc.file.rotation | /etc/systemd/system/data.ingestion.mnc.file.rotation.service | Service d’archivage des fichiers. Archive dans le dossier **/home/mnc/archive** tous les fichiers de plus de 5 minutes. |

# Description technique des outils de monitoring et d’administration

## Monitoring System

### Monitoring : Recherche d’un processus filebeat particulier

Le monitoring des processus « filebeat » se fait grâce aux agents **metricbeat**.

Exemple de recherche de processus filebeat et classement par les dates pour ne garder que les plus récents depuis l’interface **Kibana>Dev Tools**:

get metricbeat-6.2.4-2018.06.12/\_search

{

"query": {

"bool": {

"must": [

{"match": {"system.process.name": "filebeat"}}

]

}

},

"\_source": ["@timestamp","system.process.name","system.process.cgroup.cpu.id"],

"sort": { "@timestamp": { "order": "desc"}},

"size": 100

}

### Processus de mise en place du monitoring de processus

* 1 : Installer MetricBeat sur la plateforme hébergeant le processus à monitorer
* 2 : Après mise en œuvre de MetricBeat et collecte de données, construire dans **Kibana>Management** un index-pattern de type **metricbeat\***,
* 3 : Créer une requête de recherche dans l’onglet **Kibana>Discover** et la sauvegarder
* 4 : Dans l’onglet **Kibana>Visualize**, utiliser la recherche sauvegardée pour créer un **Chart** destiné à visualiser la métrique du processus et le sauvegarder,
* 5 : Au besoin dans l’onglet **Kibana>Dashboard**, intégrer le nouveau chart à un Dashboard de monitoring complet.

**Exemple :** requête vers les indexes **metricbeat\*** sauvegardée dans l’onglet **Kibana>Discover** :

system.process.name : "filebeat" and system.process.cgroup.id:"filebeat.mnc.integ.service"

### Monitoring : mise en place d’un watcher (alerte) sur un processus

Cette alerte sera mise en place grace aux agents de monitoring Metricbeat.

Dans cet exemple, on monitor l’ingester mnc en préproduction.

**Index pattern :** metricbeat\*

**Critère de recherche :** system.process.cpu.total.value : 0

**Critères de filtrage :** system.process.name : "filebeat" and system.process.cgroup.id:"filebeat.mnc.integ.service"

## Monitoring de Qualité de données

### Déduplication d’enregistrements

La stratégie proposée repose sur la détection d’enregistrements dupliqués :

Dès qu’un enregistrement est dupliqué, il est détecté.

Cette technique permettra :

* De détecter la liste de données caractérisée par le As-Of / As-At intégrant la duplication,
* De détruire la liste de données caractérisée par le As-Of / As-At intégrant la duplication,
* De réinjecter la liste de données caractérisée par le As-Of / As-At incriminé.

ATTENTION :

Cette technique ne détecte pas tous les enregistrements dupliqués : c’est un indicateur de duplication sur une période.

POST tpgbam\_mnc\_dev\_test\_index/\_search

{

"aggs":{

"dedup" : {

"terms": {

"script": {

"lang": "painless",

"source":"doc['@timestamp'].value.toString() + doc['phone'].value"

},

"min\_doc\_count": 2

}

,

"aggs":{

"dedup\_docs":{

"top\_hits":{

}

}

}

}

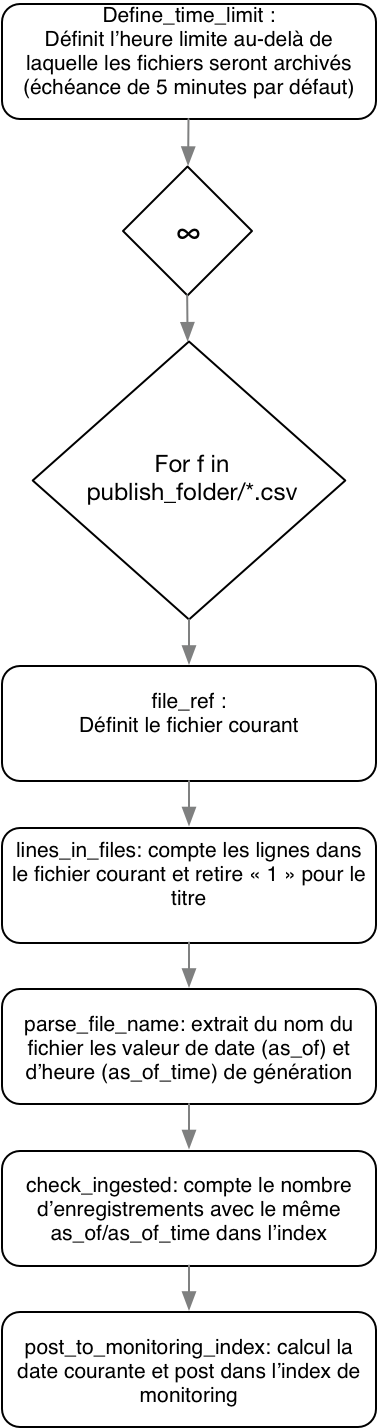
}

}

}

### Contrôle du nombre d’enregistrements intégrés

#### Algorithme



#### Cas normal : enregistrements normalement ingérés

La stratégie proposée compare le nombre d’enregistrements dans un fichier au nombre d’enregistrements intégrés dans l’index :

* On liste tous les fichiers présents dans le filesystem /home/[nom\_fournisseur]/publish
* On parse dans le nom de fichier, le As-Of et As-At correspondant aux enregistrements,
* On compte le nombre d’enregistrements dans le fichier :
  + wc -l [nom-du-fichier]
* On requête l’index pour récupérer le nombre d’enregistrements intégrés :
  + curl --user TPG\_BAM:antaes\*1 <https://416d580d881f24fb8cc3e9be04c9ce89.us-west-2.aws.found.io:9243/tpgbam_dev_mnc_index/_count?q=as_of:2018-06-14>
* Puis on insère le résultat dans un index

**Requête Elastic relative à un fichier ingéré :**

POST tpgbam\_dev\_mnc\_index/\_count

{

"query":

{

"bool": {

"should": [

{"match": { "as\_of\_time": "17:45:01"}},

{"match": { "as\_of": "2018-06-14"}}

],

"minimum\_should\_match": 2

}

}

}

**Requête http équivalente avec extraction du count en JSON :**

curl --user TPG\_BAM:antaes\*1 https://416d580d881f24fb8cc3e9be04c9ce89.us-west-2.aws.found.io:9243/tpgbam\_dev\_mnc\*/\_count -d '{ "query": { "bool": { "should": [ {"match": { "as\_of\_time": "17:45:01"}}, {"match": { "as\_of": "2018-06-14"}} ], "minimum\_should\_match": 2 } }}' -H'Content-Type: application/json' **| jq -r '.count'**

#### Cas particulier : enregistrements mal ingérés

Pour monitorer les défauts d’ingestion, le principe est de :

* Calculer (par un champ calculé en Painless Script) un **Flag** indiquant si le **nombre d’enregistrements dans l’index est le même que dans le fichier,**
* Créer un **watcher** qui contrôle qu’aucun **Flag** n’est **resté à false** au-delà d’une demie heure.

**Champ calculé – Flag de contrôle de l’ingestion des fichiers :**

(doc['num\_in\_file'].value - doc['num\_in\_index'].value)==0?true:false

### Service associé à la duplication d’enregistrements

Un service “systemd“ est créé pour lancer le script de monitoring de qualité de données.

Exemple de configuration de l’intégration :

* **Script de monitoring :** data\_ingestion\_monitoring\_integ\_v2.sh
* **Fichier de configuration du service :** /lib/systemd/system/monitoring.mnc.ingester.data.service
* **Ligne de commande associée :** service monitoring.mnc.ingester.data start

### Rapprochements de valeurs

L’objectif est de rapprocher des sommes sur des valeurs de champs dans le fichier d’origine de sommes réalisées dans l’index sur le/les même champs.

Sous Bash, une librairie est nécessaire :

**CSVKIT**

Exemple de requête pour réaliser une somme sur le champ « cost » :

cat 2018-09-14T00:00:00\_DATT\_VENTE.old2 | csvsql --query "SELECT SUM(cost) AS sum FROM stdin"

ou

csvsql --query "SELECT SUM(cost) AS sum from '2018-09-18T00:00:00\_DATT\_VENTE'" 2018-09-18T00:00:00\_DATT\_VENTE.csv

résultat :

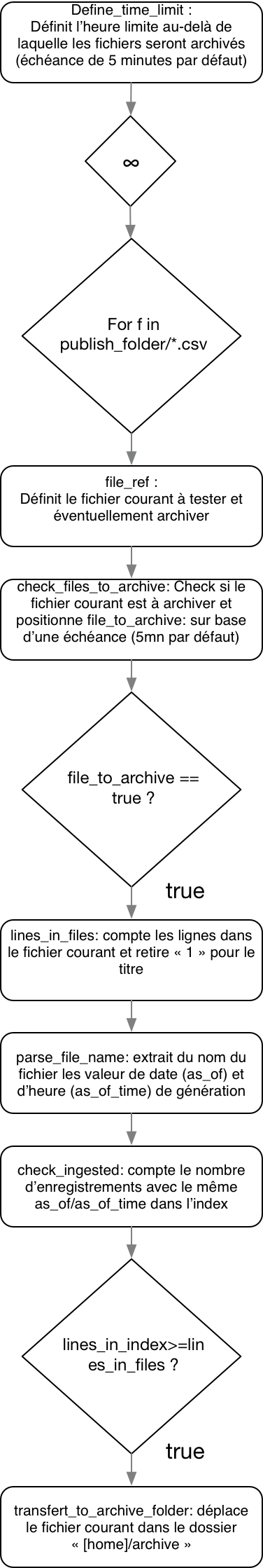
sum

14400

## Archivage des fichiers

### Algorithme - Script « data\_ingestion\_file\_rotation »

Globalement l’algorithme utilisé est le même que pour le contrôle du nombre d’enregistrements intégrés.



## Déployer avec Ansible

Ansible est un outil de scripting souvent utiliser pour le devops et permettant de déployer et configurer de façon répétable des environnements systèmes.

Dans notre cas Ansible pourra nous permettre :

* De créer les comptes et les différents dossiers des « fournisseurs de données »,
* De déployer et paramétrer les différents outils de monitoring (e.g. processus et qualité de données),
* De déployer et paramétrer les différents services nécessaires,
* Etc.

Exemple de ligne de commande de déploiement Ansible :

ansible-playbook site.yml -i integration –extra-vars “role=deploy\_filebeat data\_provider=sds”

Cette ligne se lance sur la configuration d’intégration et permet de :

* De déployer le fichier de configuration de filebeat ainsi que le fichier de service,
* De les paramétrer pour le data\_provider sds.

Le script de play site.yml permert d’exécuter dynamiquement un seul rôle.

Pour exécuter tous les rôles à la suite, il faut utiliser full\_site.yml.

Exemple de ligne de commande pour exécuter full\_tite.yml :

ansible-playbook full\_site.yml -i developpement --extra-vars "onhosts=tpgbam data\_provider=dedup depenvironment=dev enable\_service=true"

Cette ligne lance tous les rôles sur la configuration de développement et active les services « systemd ».

# REFERENCES

## Lignes de commandes

### Filebeat : Lancement de l’ingester

sudo -u mnc /usr/bin/filebeat -e -c /home/mnc/filebeat.yml -path.home /home/mnc -path.config /home/mnc -path.data /home/mnc/fbregistry -path.logs /home/mnc/log/ -d "elasticsearch"

### Filebeat : contrôle du service Systemd – Exemple sur la plateforme d’intégration

**Démarrage –** service filebeat.mnc.integ start

**Arrêt –** service filebeat.mnc.integ stop

**Status –** service filebeat.mnc.integ status

**Logs –** journalctl -u filebeat.mnc.integ.service