VITAM - Documentation d'exploitation

Version 5.0

VITAM

avr. 01, 2022

Table des matières

CHAPITRE 1

Introduction

1.1 But de cette documentation

Ce document a pour but de permettre de fournir à une équipe d'exploitants de la solution logicielle *VITAM* les procédures et informations utiles et nécessaires au bon fonctionnement de la solution logicielle.

1.2 Destinataires de ce document

Ce document s'adresse à des exploitants du secteur informatique ayant de bonnes connaissances en environnement Linux.

CHAPITRE 2

Rappels

2.1 Information concernant les licences

La solution logicielle *VITAM* est publiée sous la licence CeCILL 2.1 [1] ; la documentation associée (comprenant le présent document) est publiée sous Licence Ouverte V2.0 [2].

Les clients externes java de solution *VITAM* sont publiés sous la licence CeCILL-C [3] ; la documentation associée (comprenant le présent document) est publiée sous Licence Ouverte V2.0 [4].

2.2 Documents de référence

2.2.1 Documents internes

Tableau 1 – Documents de référence VITAM

Nom	Lien
DAT	http://www.programmevitam.fr/ressources/DocCourante/html/archi
DIN	http://www.programmevitam.fr/ressources/DocCourante/html/installation
DEX	http://www.programmevitam.fr/ressources/DocCourante/html/exploitation
DMV	http://www.programmevitam.fr/ressources/DocCourante/html/migration
Release notes	https://github.com/ProgrammeVitam/vitam/releases/latest

https://cecill.info/licences/Licence_CeCILL_V2.1-fr.html
https://www.etalab.gouv.fr/wp-content/uploads/2017/04/ETALAB-Licence-Ouverte-v2.0.pdf
https://cecill.info/licences/Licence_CeCILL-C_V1-fr.html
https://www.etalab.gouv.fr/wp-content/uploads/2017/04/ETALAB-Licence-Ouverte-v2.0.pdf

2.2.2 Référentiels externes

2.3 Glossaire

API *Application Programming Interface*

AU *Archive Unit*, unité archivistique

BDD Base De Données

BDO *Binary DataObject*

CA *Certificate Authority*, autorité de certification

CAS Content Adressable Storage

CCFN Composant Coffre Fort Numérique

CN Common Name

COTS Component Off The shelf ; il s'agit d'un composant « sur étagère », non développé par le projet *VITAM*, mais intégré à partir d'un binaire externe. Par exemple : MongoDB, ElasticSearch.

CRL *Certificate Revocation List* ; liste des identifiants des certificats qui ont été révoqués ou invalidés et qui ne sont donc plus dignes de confiance. Cette norme est spécifiée dans les RFC 5280 et RFC 6818.

CRUD *create, read, update, and delete*, s'applique aux opérations dans une base de données MongoDB

DAT Dossier d'Architecture Technique

DC Data Center

DEX Dossier d'EXploitation

DIN Dossier d'INstallation

DIP *Dissemination Information Package*

DMV Documentation de Montées de Version

DNS *Domain Name System*

DNSSEC *Domain Name System Security Extensions* est un protocole standardisé par l'IETF permettant de résoudre certains problèmes de sécurité liés au protocole DNS. Les spécifications sont publiées dans la RFC 4033 et les suivantes (une version antérieure de DNSSEC n'a eu aucun succès). Définition DNSSEC [5]

DSL *Domain Specific Language*, langage dédié pour le requêtage de VITAM

DUA Durée d'Utilité Administrative

EBIOS Méthode d'évaluation des risques en informatique, permettant d'apprécier les risques Sécurité des systèmes d'information (entités et vulnérabilités, méthodes d'attaques et éléments menaçants, éléments essentiels et besoins de sécurité...), de contribuer à leur traitement en spécifiant les exigences de sécurité à mettre en place, de préparer l'ensemble du dossier de sécurité nécessaire à l'acceptation des risques et de fournir les éléments utiles à la communication relative aux risques. Elle est compatible avec les normes ISO 13335 (GMITS), ISO 15408 (critères communs) et ISO 17799

EAD Description archivistique encodée

ELK Suite logicielle *Elasticsearch Logstash Kibana*

FIP *Floating IP*

GOT Groupe d'Objet Technique

IHM Interface Homme Machine

IP *Internet Protocol*

IsaDG Norme générale et internationale de description archivistique

JRE *Java Runtime Environment* ; il s'agit de la machine virtuelle Java permettant d'y exécuter les programmes compilés pour.

https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

JVM *Java Virtual Machine* ; Cf. *JRE*

LAN *Local Area Network*, réseau informatique local, qui relie des ordinateurs dans une zone limitée

LFC *LiFe Cycle*, cycle de vie

LTS *Long-term support*, support à long terme : version spécifique d'un logiciel dont le support est assuré pour une période de temps plus longue que la normale.

M2M *Machine To Machine*

MitM L'attaque de l'homme du milieu (HDM) ou *man-in-the-middle attack* (MITM) est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis. Le canal le plus courant est une connexion à Internet de l'internaute lambda. L'attaquant doit d'abord être capable d'observer et d'intercepter les messages d'une victime à l'autre. L'attaque « homme du milieu » est particulièrement applicable dans la méthode d'échange de clés Diffie-Hellman, quand cet échange est utilisé sans authentification. Avec authentification, Diffie-Hellman est en revanche invulnérable aux écoutes du canal, et est d'ailleurs conçu pour cela. Explication [6]

MoReq *Modular Requirements for Records System*, recueil d'exigences pour l'organisation de l'archivage, élaboré dans le cadre de l'Union européenne.

NoSQL Base de données non-basée sur un paradigme classique des bases relationnelles. Définition NoSQL [7]

NTP *Network Time Protocol*

OAIS *Open Archival Information System*, acronyme anglais pour Systèmes de transfert des informations et données spatiales – Système ouvert d'archivage d'information (SOAI) - Modèle de référence.

OOM Aussi apelé *Out-Of-Memory Killer* ; mécanisme de la dernière chance incorporé au noyau Linux, en cas de dépassement de la capacité mémoire

OS *Operating System*, système d'exploitation

OWASP *Open Web Application Security Project*, communauté en ligne de façon libre et ouverte à tous publiant des recommandations de sécurisation Web et de proposant aux internautes, administrateurs et entreprises des méthodes et outils de référence permettant de contrôler le niveau de sécurisation de ses applications Web

PDMA Perte de Données Maximale Admissible ; il s'agit du pourcentage de données stockées dans le système qu'il est acceptable de perdre lors d'un incident de production.

PKI Une infrastructure à clés publiques (ICP) ou infrastructure de gestion de clés (IGC) ou encore Public Key Infrastructure (PKI), est un ensemble de composants physiques (des ordinateurs, des équipements cryptographiques logiciels ou matériel type HSM ou encore des cartes à puces), de procédures humaines (vérifications, validation) et de logiciels (système et application) en vue de gérer le cycle de vie des certificats numériques ou certificats électroniques. Définition PKI [8]

PCA Plan de Continuité d'Activité

PRA Plan de Reprise d'Activité

REST *REpresentational State Transfer* : type d'architecture d'échanges. Appliqué aux services web, en se basant sur les appels http standard, il permet de fournir des API dites « RESTful » qui présentent un certain nombre d'avantages en termes d'indépendance, d'universalité, de maintenabilité et de gestion de charge. Définition REST [9]

RGAA Référentiel Général d'Accessibilité pour les Administrations

RGI Référentiel Général d'Interopérabilité

RPM *Red Hat Package Manager* ; il s'agit du format de paquets logiciels nativement utilisé par les distributions Linux RedHat/CentOS (entre autres)

SAE Système d'Archivage Électronique

SEDA Standard d'Échange de Données pour l'Archivage

https://fr.wikipedia.org/wiki/Attaque_de_l'homme_du_milieu
https://fr.wikipedia.org/wiki/NoSQL
https://fr.wikipedia.org/wiki/Infrastructure_%C3%A0_cl%C3%A9s_publiques
https://fr.wikipedia.org/wiki/Representational_state_transfer

SGBD Système de Gestion de Base de Données

SGBDR Système de Gestion de Base de Données Relationnelle

SIA Système d'Informations Archivistique

SIEM *Security Information and Event Management*

SIP *Submission Information Package*

SSH *Secure SHell*

Swift *OpenStack Object Store project*

TLS *Transport Layer Security*

TNA *The National Archives*, Pronom [10]

TNR Tests de Non-Régression

TTL *Time To Live*, indique le temps pendant lequel une information doit être conservée, ou le temps pendant lequel une information doit être gardée en cache

UDP *User Datagram Protocol*, protocole de datagramme utilisateur, un des principaux protocoles de télécommunication utilisés par Internet. Il fait partie de la couche transport du modèle OSI

UID *User IDentification*

VITAM Valeurs Immatérielles Transférées aux Archives pour Mémoire

VM *Virtual Machine*

WAF *Web Application Firewall*

WAN *Wide Area Network*, réseau informatique couvrant une grande zone géographique, typiquement à l'échelle d'un pays, d'un continent, ou de la planète entière

https://www.nationalarchives.gov.uk/PRONOM/

CHAPITRE 3

Expertises requises

Les équipes en charge du déploiement et de l'exploitation de la solution logicielle *VITAM* devront disposer en interne des compétences suivantes :

Tableau 1 – Matrice de compétences

Thème	Outil	Description de l'outil	Niveau requis	Niveau de criticité	Exemples de compétences requises
Système	Linux (Centos 7 ou Debian 10)	Système d'exploitation	3/4 : maitrise	3/4 : Majeur	Etre à l'aise avec l'arborescence linux / Configurer une interface réseau / Analyse avancée des logs systèmes et réseaux
Configuration	Git	Suivi des modifications quotidiennes des sources de déploiement VITAM	1/4 : débutant	1/4 : Mineur	Savoir éxécuter les commandes de bases (commit, pull, push, etc. . .)
Configuration	Git	Adaptation des sources de déploiement VITAM dans le cadre d'une montée de version	2/4 : intermediaire	1/4 : Mineur	Savoir éxécuter les commandes intermédiaires (branche, merge, etc. . .)
Configuration	Ansible	Gestion de configuration et déploiement automatisé	3/4 : maitrise	3/4 : Majeur	Adapter les paramètres pour permettre une installation spécifique / Comprendre l'arborescence des rôles et des playbooks
Exploitation	Consul	Outil d'enregistrement des services VITAM	1/4 : débutant	4/4 : critique	Contrôler l'état des services via l'interface consul Eteindre et redémarrer un Consul Agent sur une machine virtuelle
Supervision	Kibana	Interface de visualisation du contenu des bases Elasticsearch	1/4 : débutant	2/4 : significatif	Créer un nouveau dashboard avec des indicateurs spécifiques / Lire et relever les données pertinentes dans un dashboard donné
Supervision	Cerebro	Interface de contrôle des clusters Elasticsearch	1/4 : débutant	2/4 : significatif	Contrôler l'état des clusters elasticsearch via l'interface cerebro
Base de données	MongoDB	Base de données NoSQL	2/4 : intermediaire	4/4 : critique	Effectuer une recherche au sein d'une base mongoDB / Sauvegarder et restaurer une base mongoDB (data ou offer) / Augmenter la capacité de stockage d'une base mongoDB
Base de données	Elasticsearch	Moteur de recherche et d'indexation de données distribué	2/4 : intermediaire	4/4 : critique	Sauvegarder et restaurer une base elasticsearch (data ou log) / Augmenter la capacité de stockage d'une base elasticsearch / Effectuer une procédure de maintenance d'un nœud au sein d'un cluster elasticsearch
Applicatif	Applicatifs Java	Composants logiciels Vitam	2/4 : intermediaire	4/4 : critique	Appeler le point "v1/status" manuellement sur tous les composants VITAM / Arrêter et relancer selectivement les composants VITAM à l'aide d'Ansible (ordre important) / Lancer une procédure d'indisponiblité de VITAM (fermeture des services external, arrêt des timers)
Stockage	Solution de stockage objet déployée	Administration du service de stockage objet Swift ou S3 (si utilisé)	2/4 : intermediaire	4/4 : critique	pouvoir lister les containers/buckets et objets, vérifier la capacité de stockage disponible. . .

- Niveau requis : Qualifie le niveau de compétence attendue par l'exploitant de la solution logicielle Vitam.
- Niveau de criticité : Qualifie le degré d'importance pour le bon fonctionnement de la plateforme.

CHAPITRE 4

Architecture de la solution logicielle VITAM

Le schéma ci-dessous représente une solution logicielle *VITAM* :

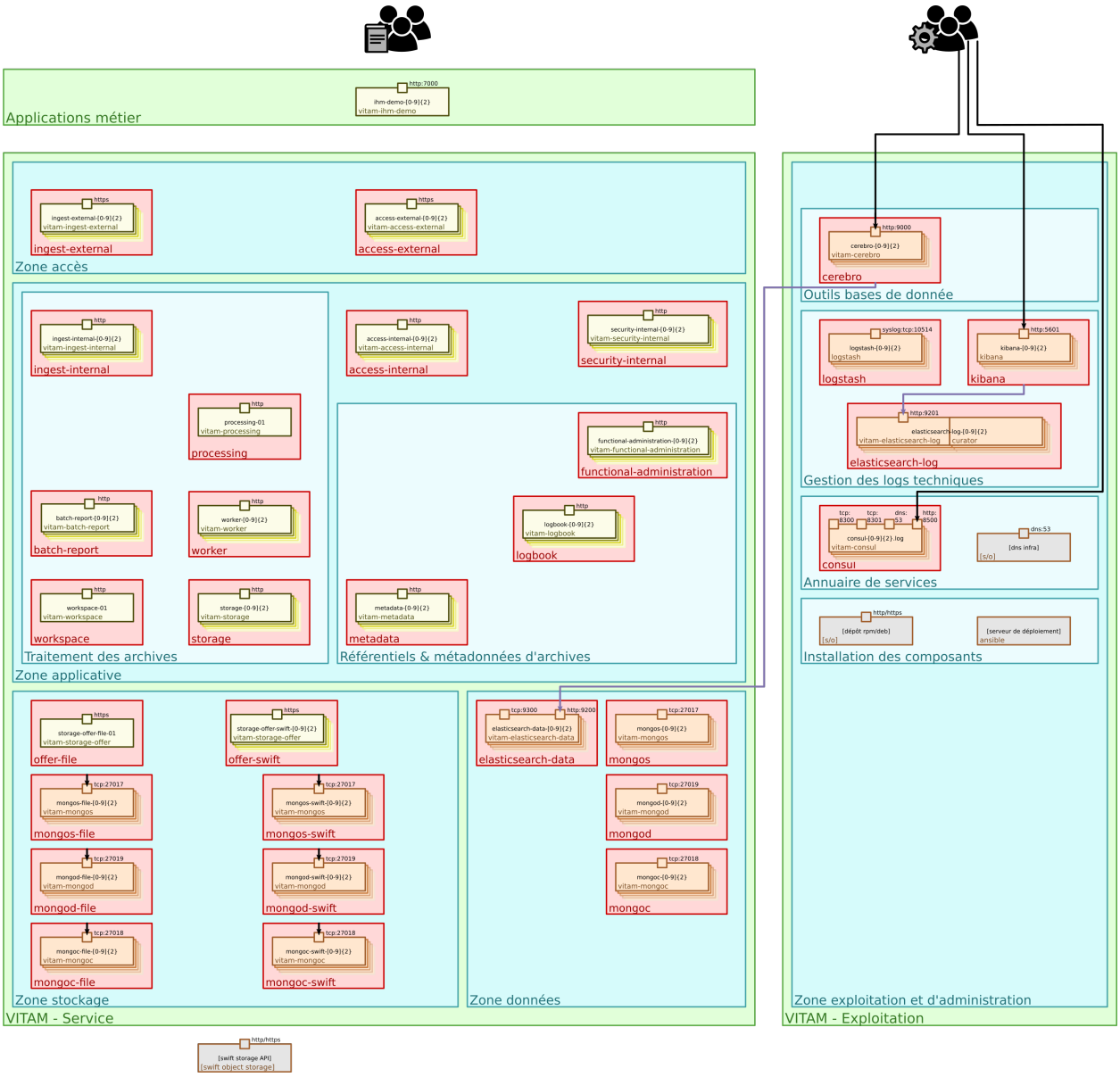


Fig. 1 – Vue d'ensemble d'un déploiement *VITAM* : zones, composants

Voir aussi :

Se référer au *DAT* (et notamment le chapitre dédié à l'architecture technique) pour plus de détails, en particulier concernant les flux entre les composants.

CHAPITRE 5

Exploitation globale

5.1 Gestion des accès

5.1.1 API

La gestion des accès aux *API* externes se fait via les *granted stores* (cf. *Fichiers communs* (page 160) pour les fichiers communs, cf. *Configuration / fichiers utiles* (page 175) pour access-external et *Configuration / fichiers utiles* (page 179) pour ingest-external).

5.1.2 IHM de démonstration

Dans cette version, la gestion et l'authentification des utilisateurs peut se faire par :

- un fichier plat qui contient logins et mots de passe (peu sécurisé)
- certificat x509
- un ldap

5.1.3 IHM de recette

Dans cette version, la gestion des utilisateurs se fait par :

- un fichier plat qui contient logins et mots de passe (peu sécurisé)
- certificat x509
- un ldap

5.2 Audit de cohérence de données entre MongoDb et Elasticsearch

Cette partie décrit le process de l'audit de cohérence de données dans Vitam.

5.2.1 Principe de fonctionnement

5.2.1.1 Configuration

Il s'agit en premier temps de pouvoir se connecter sur les différents shards du mongod (sachant qu'un seul peut shard contient 1 ou plusieurs noeuds : lieu de stockage de data) NB : A ce jour, Vitam utilise un seul shard mongod. Cette configuration existe dans le fichier metadata.conf :

```
mongodShardsConf:
 dbUserName: vitamdb-localadmin
 dbPassword: qwerty
 mongoDbShards:
   - shardName: shard0
     mongoDbNodes:
       - dbHost: 172.17.0.2
         dbPort: 27019
```

On ajoute à cette configuration d'autres paramètres qui permettent de gérer l'audit :

- isDataConsistencyAuditRunnable : Permet de lancer ou pas l'audit
- dataConsistencyAuditOplogMaxSize : Spécifier la taille maximale de données à lire depuis les oplog par noeud !

5.2.1.2 Exécution

Une fois l'audit est accessible, un contrôle est fait sur l'existence d'un audit qui pourrait déjà être en cours. S'il n'y a pas d'audit en cours, Vitam crée un logbook pour la fonctionnalité, puis les clients de connexions MongoDB. Vitam procède ensuite à la lecture des différents oplog des nœuds.

5.2.1.3 Lecture de l'oplog

L'oplogReader permet de parcourir l'ensemble des opérations de l'oplog et d'extraire les opérations les plus récentes de type (insert, update et delete) sur les collections « Unit » et « ObjectGroup » et qui ont une date d'opération postérieure à la dernière date de lancement d'audit (fichier de configuration nommé « mongoDbShardsTimestamp-Configuration.json » placé dans /workspace/dataConsistencyAuditContainer).

5.2.1.4 Traitement des opérations de l'oplog MongoDB et comparaison avec ES

Une fois les opérations d'oplog récupérées et traitées, une comparaison sera faite avec les opérations associées et récupérées depuis d'Elasticsearch afin de vérifier la cohérence et générer les différences.

5.2.1.5 Comportement de l'audit suite à la comparaison de données

Une fois la comparaison terminée, on distingue 2 cas de figure :

1. 0 différences : le logbook passera en mode success et on quitte l'audit.
2. Existence de différences : dans ce cas une correction sur ES devrait être faite, et une nouvelle comparaison faite par rapport à ce correctif. A ce stade, on distingue à nouveau 2 cas de figure :
   - 0 différences : (Données corrigées sur ES) le logbook passera en mode warning
   - Existence de différences : (Données non corrigées sur ES) le logbook passera en mode ko

5.2.2 Lancement de l'audit

L'audit est un lancé par un timer configuré dans vitam_vars.yml :

```
metadata:
    - ....
    - ....
    - name: vitam-metadata-audit-mongodb-es
      frequency: "2020-01-01 00:00:00"
```

Pour pouvoir lancer l'audit, il faut :

- Modifier la fréquence de lancement du timer selon le besoin.
- Setter isDataConsistencyAuditRunnable à true au niveau de metadata.conf

Il s'agit d'une fonction de recette, a ne pas utiliser en production.

5.3 Configuration des champs ObjectGroup devant être exclus de la recherche

Cette partie décrit le process d'exclure des propriétes des groupes d'objets lors de leurs recherche dans Vitam en tenant compte des différentes API qui remontent des données de groupes d'objets.

5.3.1 Configuration

Il faudrait tout d'abord configurer la propriété `objectGroupBlackListedFieldsForVisualizationByTenant`, qui définit une Map associant un tenant à une black liste de propriétes de groupe d'objets. Cette liste de propriétes serait exclue d'affichage lors des recherchers faites sur la collection d'objet group pour le tenant concerné.

Ci-dessous un exemple de cette configuration, qui existe dans le fichier access-external.conf :

```
authentication: true
jettyConfig: jetty-config.xml
tenantFilter : true
authorizeTrackTotalHits : false
objectGroupBlackListedFieldsForVisualizationByTenant:
 0: ['Filename', '#operations', 'Size']
 1: ['Filename', '#opi']
 2: ['Filename']
 3: ['Filename']
 4: ['Filename']
 5: ['Filename']
 6: ['Filename']
 7: ['Filename']
 8: ['Filename']
 9: ['Filename']
```

Par défaut, tout les tenants du système (9 dans cet exemple) déclarent le champ `Filename` dans leur black liste. Ensuite, libre à l'exploitant de gérer cette liste en rajoutant, modifiant, voir même supprimant le champ par défaut.

5.3.2 Fonctionnement

Une fois la configuration de la black liste est faite, et le(s) service(s) access-external démarré(s), les requêtes qui permettent de lancer des recherches sur les objets groupes, et qui prennent en considération un tenant d'utilisation, ne remontent plus au niveau des réponses, les propriétes déja mis dans la liste.

Ci-dessous les points d'API concernés :

```
/objects:
  displayName: Objects
  description: |
   API qui permet d'accéder aux objets groupes en se basant sur une requête qui␣
↪utilisele langage de requête (DSL)
   de Vitam en entrée et retourne l'objet d'archives selon le DSL Vitam en cas de␣
↪succès.
  get:
    description: |
      Requête qui retourne le résultat contenant un Object d'archives : ses␣
↪métadonnées ou un de ses objets binaires.
      Dans le cas des métadonnées, la requête utilise le langage de requête DSL de␣
↪type **recherche unitaire (GET BY ID)** de Vitam en entrée.
      'Accept' header est 'application/octet-stream' (objet binaire) ou 'application/
↪json' (métadonnées)

      Permissions requises:
        - objects:read
    is: [AccessTraits.AccessUniqueObjectQualifierResponse, AccessTraits.
↪AccessUniqueObjectResponse]
    headers:
      Accept:
        required: true
        enum: [ "application/octet-stream", "application/json" ]
```

```
/units/{{unit-id}}/objects:
  displayName: Objects of one ArchiveUnit
  description: |
   API qui définit les requêtes pour accéder à l'Objet d'archives associé à l'Unité d
↪'archives s'il existe.
   La requête utilise le langage de requête (DSL) de Vitam en entrée et retourne l
↪'objet d'archives selon le DSL Vitam en cas de succès.
  get:
    description: |
      Requête qui retourne le résultat contenant un Object d'archives : ses␣
↪métadonnées ou un de ses objets binaires, rattachés
      à l'identifiant de l'unité archivistique en paramétre.

      Permissions requises:
        - units:id:objects:read:json

        ou
        - units:id:objects:read:binary
    is: [AccessTraits.AccessUniqueObjectQualifierResponse, AccessTraits.
↪AccessUniqueObjectResponse]
    headers:
      Accept:
        required: true
        enum: [ "application/octet-stream", "application/json" ]
```

5.4 Portails d'administration

5.4.1 Technique

Aucun portail d'administration technique n'est prévu dans cette version de la solution logicielle *VITAM*.

5.4.2 Fonctionnel

Le portail d'administration fonctionnel est intégré au composant **ihm-demo** dans cette version de la solution logicielle *VITAM* (cf. *Présentation* (page 190)).

5.5 Paramétrage & configuration

L'étape de paramétrage et la configuration sont essentiellement liées à la mise en place ou la mise à niveau de la solution logicielle *VITAM* (ansible / inventaire).

Voir aussi :

Plus d'informations, et notamment les paramètres d'installation, sont disponibles dans le *DIN*.

5.5.1 Mise à niveau de la configuration de l'environnement

5.5.1.1 Mise à jour du nombre de tenants

Note : se référer au *DIN* pour plus d'informations à ce sujet.

5.5.1.2 Mise à jour des paramètres JVM

Un tuning fin des paramètres JVM de chaque composant VITAM est possible. Pour cela, il faut modifier le contenu du fichier `environments/group_vars/all/jvm_opts.yml`

Note : se référer au *DIN* pour plus d'informations à ce sujet.

5.6 Déploiement / mises à jour

5.6.1 Mise à jour des certificats

Pour mettre à jour les certificats (avant expiration par exemple), il suffit de les mettre à jour dans les répertoires de déploiement, puis de regénérer les stores (dans `environments/keystores`) et lancer leur redéploiement via cette commande ansible :

> ansible-playbook ansible-vitam/vitam.yml -i environments/hosts.<environnement> –ask-vault-pass –tags update_vitam_certificates ansible-playbook ansible-vitam-extra/extra.yml -i environments/hosts.<environnement> –ask-vault-pass –tags update_vitam_certificates

Voir aussi :

Le cycle de vie des certificats est rappelé dans les annexes. Une vue d'ensemble est également présentée dans le *DIN*.

5.6.2 Mise à jour de la solution logicielle VITAM

Pour la mise à jour de la solution logicielle *VITAM* (tout comme pour sa première installation), se référer au *DIN*, au *DMV*, ainsi qu'à la *release note* associée à toute version.

Ces documents détaillent les pré-requis, la configuration des fichiers et les procédures éventuelles de migration de données pour effectuer une mise à jour applicative. Le *DMV* explique également comment valider une montée de version applicative de la solution logicielle *VITAM*.

Voir aussi :

Plus d'informations, et notamment les paramètres d'installation, sont disponibles dans le *DIN*.

Voir aussi :

Dans le cadre d'une montée de version, se référer également au *DMV*.

5.6.3 Ajouter un/des instances de composants VITAM

Dans le cas où le dimensionnement initial ne donne pas pleinement satisfaction, il est possible de rajouter à une solution logicielle *VITAM* existante une/des instances supplémentaires de composants.

Pour le moment, il n'est pas possible de déplacer un composant automatiquement via ansible d'un serveur à un autre (implique une suppression du composant sur l'ancien serveur non gérée pour le moment)

Prudence : Dans le cas d'ajout d'une offre, il est nécessaire de suivre la procédure de resynchronisation des offres.

Avertissement : Les composants « vitam-processing », « vitam-workspace » ne sont pas multi-instantiables.

Avertissement : Le composant « vitam-offer » n'est PAS mono-instantiable lorsqu'il est déployé en mode système de fichiers, ou archivage sur bandes magnétiques. Le composant « vitam-offer » est multi-instantiable lorsqu'il est déployé en mode S3 ou Swift.

1. Modifier l'inventaire avec la/les VM supplémentaire(s)
2. Lancer un déploiement comme indiqué dans le *DIN* en rajoutant la directive `-l <liste de/des VM(s) supplémentaire(s)>`

5.6.4 Modifier la fréquence de lancement de certains *timers* systemD

Par défaut, la solution logicielle *VITAM* déploie et active, selon l'usage (site primaire / site secondaire), des *timers* systemD. Le playbook ansible d'installation de vitam `ansible-vitam/vitam.yml`, permet d'uniquement modifier la fréquence des *timers* en rajoutant le tag `update_timers_frequency`.

Pour cela, il faut éditer la section `vitam_timers` dans le fichier `environments/group_vars/all/vitam_vars.yml`.

A l'issue, lancer le playbook avec la commande

```
ansible-playbook ansible-vitam/vitam.yml -i environments/hosts.<environnement> --ask-
↪vault-pass --tags update_timers_frequency
```

5.7 Interruption / maintenance

5.7.1 Procédure d'arrêt complet

Un playbook ansible d'arrêt complet de la solution logicielle *VITAM* est fourni, sous `deployment/ansible-vitam-exploitation/stop_vitam.yml`, pour réaliser de façon automatisée les actions nécessaires. Ce playbook arrête aussi les timers systemD associés aux composants *VITAM*.

Avertissement : Ce script, en l'état, permet un *EMERGENCY BREAK*, autrement dit un arrêt brutal des composants, ne permettant pas de garantir, à l'issue, une cohérence des données. Il est donc fortement recommandé de positionner les traitements courants en pause avant de lancer la procédure d'arrêt.

Note : Une confirmation est demandée pour lancer ce script d'arrêt de la solution logicielle VITAM. Cette confirmation peut être automatisée en utilisant les extra-vars d'ansible `-e "confirmation=yes"`

5.7.2 Procédure de démarrage complet

Les machines hébergeant la solution logicielle *VITAM* doivent être allumées et en état de fonctionnement pour exécuter cette procédure.

Un playbook ansible de démarrage complet de la solution logicielle *VITAM* est fourni, sous `deployment/ansible-vitam-exploitation/start_vitam.yml`, pour réaliser de façon automatisée les actions nécessaires. Ce playbook démarre aussi les timers systemD associés aux composants *VITAM*.

5.7.3 Procédure de statut

Un playbook ansible est fourni, sous `deployment/ansible-vitam-exploitation/status_vitam.yml`, pour réaliser de façon automatisée les « autotest » intégrés dans la solution logicielle *VITAM*.

5.7.4 Autres cas

5.7.4.1 Procédure de maintenance / indisponibilité de VITAM

Deux playbooks ansible sont fournis dans `deployment/ansible-vitam-exploitation` :

- `stop_external.yml` : permettant d'arrêter uniquement les composants *VITAM* **ingest-external** et **access-external**
- `start_external.yml` : permettant de démarrer uniquement les composants *VITAM* **ingest-external** et **access-external**

Ces playbooks permettent d'empêcher l'accès à la solution logicielle *VITAM* par les services versants, tout en laissant opérationnel le reste de la solution logicielle. Ils peuvent être utiles, voire nécessaires, dans le cadre d'une migration de données ou de maintenance de la solution logicielle *VITAM*.

Ils ne stoppent donc pas :

- Les versements qui sont encore en cours de traitement (il est toutefois possible de les mettre en pause via ihm-demo par exemple)
- Les timers qui lancent divers traitements comme des sécurisations, pour cela, se référer au chapitre suivant

5.7.4.2 Procédure de maintenance liée aux *timers systemD*

Deux playbooks ansible sont fournis dans `deployment/ansible-vitam-exploitation` :

- `stop_vitam_timers.yml` : permettant d'arrêter uniquement les *timers systemD*
- `start_vitam_timers.yml` : permettant de démarrer uniquement les *timers systemD*. Ce playbook est à lancer une fois le démarrage des services correctement réalisé.

5.7.4.3 Procédure de maintenance sur les composants d'administration

Deux playbooks sont fournis dans `deployment/ansible-vitam-exploitation` :

- `stop_vitam_admin.yml` : permettant d'arrêter sélectivement les composants Consul, la chaine de log (logstash / cluster elasticsearch-log / kibana-log), cerebro et les docker mongo-express et elasticsearch-head
- `start_vitam_admin.yml` : permettant de démarrer sélectivement les composants Consul, la chaine de log (logstash / cluster elasticsearch-log / kibana-log), cerebro et les docker mongo-express et elasticsearch-head

Avertissement : En passant le playbook d'arrêt, l'ensemble de la solution logicielle *VITAM* devient inutilisable.

5.7.4.4 Procédure de maintenance des *IHM*

Deux playbooks sont fournis dans `deployment/ansible-vitam-exploitation` :

- `stop_vitam_ihm.yml` : permettant d'arrêter sélectivement les composants *VITAM IHM* ihm-demo et ihm-recette
- `start_vitam_ihm.yml` : permettant de démarrer sélectivement les composants *VITAM IHM* ihm-demo et ihm-recette

5.7.4.5 Procédure de maintenance des *Bases de données métier*

Quatre playbooks sont fournis dans `deployment/ansible-vitam-exploitation` :

- `start_elasticsearch_data.yml` : pemettant de démarrer le cluster elasticsearch-data
- `start_mongodb.yml` : permettant de démarrer les clusters mongodb (mongo-data & mongo-offer)
- `stop_elasticsearch_data.yml` : permettant d'arrêter le cluster elasticsearch-data
- `stop_mongodb.yml` : permettant d'arrêter les clusters mongodb (mongo-data & mongo-offer)

5.8 Sauvegarde / restauration

Note : La sauvegarde des bases de métadonnées MongoDB et Elasticsearch, ainsi que la restauration de leur contenu en cohérence avec les offres sous-jacentes, est déjà gérée par les mécanismes intrinsèques à la solution *VITAM*, cf. le *DAT*, chapitre **reconstruction**. Il est toutefois recommandé de réaliser des sauvegardes lors d'événements d'exploitation, tel que l'upgrade de la plateforme, ou de manière régulière, lorsque le volume de données en base est important, et ce, afin d'éviter un processus de reconstruction trop long.

Avertissement : Cette méthode s'applique uniquement pour des déploiements de petite taille et n'est pas recommandée pour un usage en production dont le volume de données géré est important (plusieurs centaines de millions d'AU).

Les procédures sont issues des documentations officielles :

- mongoDB : https://docs.mongodb.com/manual/tutorial/backup-and-restore-tools/
- elasticsearch : https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html

5.8.1 Sauvegarde

Note : Pour que cette sauvegarde soit fonctionnellement correcte, il faut que la solution logicielle *VITAM* soit dans un état stable et cohérent, sans possibilité de réaliser des versements et sans travail de fond (jobs de sécurisation, ...). Il est recommandé de l'exécuter lorsque les services client des bases sont éteints. Pour une sauvegarde à chaud, se référer à la documentation officielle afin de disposer de l'ensemble de la procédure sécurisée.

5.8.1.1 mongoDB

La commande suivante est à lancer depuis une machine hébergeant le composant `vitam-mongod` dans le cas d'un serveur standalone ou `vitam-mongos` dans le cas d'un cluster shardé. Il est aussi possible de l'exécuter pour chaque shard, en considérant le serveur primary comme un serveur standalone, autrement dit depuis un composant `vitam-mongod` de chaque shard. La commande est à lancer pour le mongo data ET pour chaque mongo offer de *VITAM* :

> mongodump –host mongodb.example.net –port 27017 –out /data/backup/ –username vitamdb-admin –password « pass » –gzip

Note : Vérifier que l'espace disponible dans le répertoire de sauvegarde (défini par `--out`) est cohérent au volume de données à sauvegarder (compressé)

Note : Se reporter au fichier `deployment/environments/group_vars/all/vault-vitam.yml` de l'ansiblerie de déploiement pour le mot de passe `vitamdb-admin`

Rapatrier sur un serveur approprié le produit généré dans la valeur de `--out`.

Pour rappel, il y a un cluster mongo de « data », ainsi qu'un cluster mongo « offer » associé à chaque offre. Pour un système cohérent, il faut effectuer la sauvegarde de chacun de ces *clusters*.

5.8.1.2 Elasticsearch

La commande suivante est à lancer depuis une machine elasticsearch de *VITAM* ayant un espace suffisant dans le répertoire de sauvegarde :

> curl -X PUT http://elasticsearch-data.service.${consul_domain} :9200/_snapshot/vitam_backup -d "{ « type » : « fs », « settings » : { « location » : « ${output_dir} » } }"

Cette étape va créer le *repository* **vitam_backup** de sauvegarde, l'arborescence étant définie par `${output_dir}`.

Pour vérifier l'état du *repository* **vitam_backup** sur les noeuds du cluster

```
curl -X POST http://elasticsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/_verify
```

Pour lancer un *snapshot* (dans l'exemple, appelé snapshot_1)

```
curl -X PUT http://elasticsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/snapshot_1?wait_for_completion=true
```

Note : la commande ne rendra la main qu'à la fin de la procédure de *snapshot*.

A l'issue de la sauvegarde, procéder à une recopie de `${output_dir}` sur un serveur à part.

5.8.2 Restauration

Note : Comme pour la sauvegarde, la restauration ne peut s'effectuer que sur un environnement *VITAM* stable et cohérent, sans possibilité de réaliser des versements et sans travail de fond (jobs de sécurisation, ...). De plus, le contenu restauré doit être cohérent avec le contenu des offres de stockage sous-jacentes.

5.8.2.1 mongoDB

Il faut d'abord procéder au rapatriement dans `${output_dir}` de la sauvegarde à appliquer.

Avertissement : une sauvegarde ne peut se restaurer que sur un environnement dans la même version.

La commande suivante est à lancer depuis une machine mongo de *VITAM* possédant le répertoire de sauvegarde à restaurer vers le serveur mongod ou mongos (selon le cas sélectionné à l'import et en rapport à la présence d'un serveur standalone ou d'un cluster shardé) :

> mongorestore –host mongodb1.example.net –port 27017 –username vitamdb-admin –password "pass" ${output_dir}/${fichier} –gzip

Note : Se reporter au fichier `deployment/environments/group_vars/all/vault-vitam.yml` de l'ansiblerie de déploiement pour le mot de passe `vitamdb-admin`

5.8.2.2 Elasticsearch

Il faut d'abord procéder au rapatriement dans `${output_dir}` de la sauvegarde à appliquer.

Commande pour lister les *snapshots* de **vitam_backup** (repository)

```
curl -X GET http://elasticsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/
```

Pour lancer une restauration, placer le nom du *snapshot* à la place de *snapshot* dans l'URL suivante

```
curl -X POST http://elasticsearch-data.service.${consul_domain}:9200/_snapshot/vitam_
↪backup/*snapshot*/_restore
```

5.8.3 Cas de la base mongo certificates

La solution logicielle *VITAM* fournit un playbook de sauvegarde de la base de données `identity` ; le *backup* réalisé est stocké sur la machine de déploiement.

Pour lancer le playbook de sauvegarde

```
ansible-playbook ansible-vitam-exploitation/backup_database_certificates.yml -i␣
↪environments/hosts.<environnement> --ask-vault-pass
```

Note : Il est recommandé de procéder à une sauvegarde régulière de la collection **identity**, ou suite à des modifications sur les certificats (ajout / mise à jour / révocation).

5.9 Sauvegarde et restauration de mongodb gros volumes

5.9.1 Préconisation

La documentation officielle de MongoDB présente différentes techniques de sauvegarde et restauration :

- utilisation des outils mongodump/mongorestore (Cf. la *section dédiée* (page 18)))
- utilisation de Filesystem snapshots
- réalisation de copies de disque.

La technique sélectionnée par *VITAM* est la troisième pour les raisons suivantes :

- Le volume de donnée en production est important. Les outils mongodump et mongorestore ne sont pas conseillés dans ce cas de figure car ils necessitent un temps d'exécution trop important (sauvegarde / restauration des données et création d'indexes).
- La technique des snapshots des disques dépend fortement des outils disponibles de l'environnement matériel.

Voir aussi :

Pour plus d'information, veuillez-vous référer à la documentation officielle : mongodump [11] et Filesystem Snapshots [12].

La sauvegarde et la restauration d'une base de forte volumétrie doit être anticipée lors du déploiement :

- en limitant le volume de donnée, géré par un shard, à une taille raisonnable ;
- en prévoyant un espace disque de taille identique au volume géré ou au moyen d'ajouter un disque supplémentaire.

5.9.2 Sauvegarde d'un cluster Mongo shardé

1. Identifier dans chaque replicaSet les instances mongo Primary (le replicaset configserver, géré par les composants vitam-mongoc, et le replicaset de chaque shard, géré par le composant vitam-mongod)
2. Arrêter proprement les services mongo
3. Réaliser la copie compressée (tar.gz, zip, ...) du dossier /vitam/data/mongoc/db (pour l'instance Primary du configserver) ou du dossier /vitam/data/mongod/db (pour l'instance Primary d'un shard), pour chaque replicaset identifié précédemment, en veillant à respecter le nommage des sauvegardes : configsrv, shard0, shard1, ...
4. Stocker les copies sur un disque séparé et sécurisé

https://docs.mongodb.com/manual/tutorial/backup-and-restore-tools/
https://docs.mongodb.com/manual/tutorial/backup-with-filesystem-snapshots/

5.9.3 Restauration d'un cluster Mongo shardé

Note : La documentation officielle est consultable ici : Restore sharded cluster [13].

1. Si le cluster existant n'est pas utilisé pour restaurer la sauvegarde, déployer un nouveau cluster mongo vide, avec les mêmes caractéristiques que l'existant (configuration et nombre de shards). Il est conseillé d'utiliser l'ansiblerie *VITAM*, en modifiant l'inventaire, et en exécutant le playbook d'initialisation du cluster Mongo (`ansible-vitam/mongodb_data.yml`)
2. Arrêter le cluster mongodb
3. Modifier la configuration des services vitam-mongoc (chaque membre du replicaset configserver) pour désactiver la replication et le sharding, en commentant dans le fichier /vitam/conf/mongoc/mongoc.conf les lignes suivantes

```
replication:
  replSetName: configsvr # name of the replica set
  enableMajorityReadConcern: true

sharding:
  clusterRole: configsvr # role du shard
```

4. Modifier la configuration des services vitam-mongod (chaque membre du replicaset de chaque shard), en commentant dans le fichier /vitam/conf/mongod/mongod.conf les lignes suivantes

```
replication:
  replSetName: shardX # name of the replica set
  enableMajorityReadConcern: true

sharding:
  clusterRole: shardsvr # role du shard
```

5. Si les services vitam-mongoc et vitam-mongod sont dans une zone réseau privée et sécurisée, il est plus simple de désactiver l'authentification en commentant la partie sécurité dans les fichiers de configuration modifiés précédemment. Sinon, dans la suite de la procédure, un chapitre permet de tenir compte de cette contrainte. Pour désactiver la sécurité, commenter les lignes suivantes

```
security:
  authorization: enabled
  clusterAuthMode: keyFile
  keyFile: "/vitam/conf/mongoc/keyfile"
```

6. Copier et décompresser les sauvegardes, en respectant chaque nom de fichier vers la machine destinataire (configsrv, shard0, shard1, ...), dans le dossier /vitam/data/mongoX de chaque instance mongo, c'est à dire de l'ensemble des membres de chaque replicaset
7. Démarrer tous les services mongo en commençant par les services vitam-mongoc puis les services vitam-mongod, ou réutiliser le playbook ansible (start_mongo.yml) en limitant aux machines des groupes hosts_mongoc_data et hosts_mongod_data (paramètre –limit)
8. Pour chacune des instances mongoc et mongod, se connecter au serveur avec le client mongo, et exécuter les opérations suivantes :
   (a) Si l'authentification est activée, il faut créer un `systemUser` (pré-requis : il faut un utilisateur ayant un role « root ») de manière à disposer des droits pour exécuter les prochaines opérations. Pour cela exécuter les commandes suivantes :

[13] https://docs.mongodb.com/manual/tutorial/restore-sharded-cluster/

```
use admin
// Authenticate as root user
db.auth("rootUser", "rootUserPassword")
// Create system user
db.createUser({user: "systmUser", pwd: "systemUserPassword", roles: [
↪"__system" ]})
// Authenticate as system user
db.auth("systmUser", "systemUserPassword")
```

(b) Supprimer la base de données `local`

```
// Drop local database
use local
db.dropDatabase()
```

(c) Pour les machines mongoc uniquement, et si la restauration est réalisée sur des nouvelles machines hébergeant les services vitam-mongod, modifier la configuration des instances mongoc (configserver) : mettre à jour la collection `shards` en spécifiant les nouvelles ips des machines :

```
use config
// spécifier les shards pour chaque mongoc
// Example
db.shards.updateOne({ "_id" : "shard0"},  { $set : { "host" : "shard0/
↪ip_member0-1:27019,ip-member0-2:27019,ip-member0-3:27019"}})
db.shards.updateOne({ "_id" : "shard1"},  { $set : { "host" : "shard1/
↪ip_member1-1:27019,ip-member1-2:27019,ip-member1-3:27019"}})
db.shards.updateOne({ "_id" : "shard2"},  { $set : { "host" : "shard2/
↪ip_member2-1:27019,ip_member2-2:27019,ip_member2-3:27019"}})
```

(d) Pour les machines mongod uniquement, et si la restauration est réalisée sur des nouvelles machines hébergeant les services vitam-mongoc, modifier la configuration des instances mongod (les shards) : mettre à jour la collection `system.version` en spécifiant les nouvelles ips des machines :

```
use admin
db.system.version.deleteOne( { "_id": "minOpTimeRecovery" } )
// spécifier les mongoc pour chaque shard
// Example
db.system.version.updateOne({ "_id" : "shardIdentity" },{ $set :{
↪"configsvrConnectionString" : "configserver/ip_member_1:27018,ip_
↪member_2:27018,ip_member_3:27018"}})
```

(e) Si un utilisateur ayant un role `__system` a été créé à l'étape (6.1), il faut le supprimer

```
// Remove system user
use admin
// Authenticate as root user
db.auth("rootUser", "rootUserPassword")
db.removeUser("systmeUser")
```

9. Arrêter l'ensemble des services mongo et réactiver la replication et le sharding (et l'authentification si désactivée) dans les fichiers de configuration de chacune des instances
10. Démarrer l'ensemble des services mongoc et mongod (en respectant l'ordre déjà spécifié précédemment)
11. Activer les `replicaSet` pour chacun des mongoc et mongod (shards) en exécutant, avec le client mongo, le script init-replica-config.js disponible sur chacune des machines dont le paramètre mongo_rs_bootstrap est spécifié dans l'inventaire ansible. Aussi depuis chacune de ces machines, il faut exécuter le script en modifiant le paramètre host de manière à l'exécuter sur chaque membre du replicaSet

```
// Sur un des mongoc
> mongo --host {{ ip_service }} --port {{ mongodb.mongoc_port }} {{ vitam_
↪defaults.folder.root_path }}/app/mongoc/init-replica-config.js
// Pour chaque shards et sur un des shards d'un replicaset
> mongo --host {{ ip_service }} --port {{ mongodb.mongod_port }} {{ vitam_
↪defaults.folder.root_path }}/app/mongod/init-replica-config.js
```

Avertissement : Chaque membre Secondary activé effectue une synchronisation initiale pour reprendre l'ensemble des commandes opérées sur le membre Primary. En fonction du volume de données géré par shard, ainsi que des performances des machines et du réseau, cette opération peut s'exécuter en un temps important, durant lequel les performances du cluster seront affaiblies.

1. Démarrer les services vitam-mongos
2. Test de la restauration
   - Un document accessible depuis un shards devrait être accessible depuis `mongos` (faire la requête de test sur chaque shard)
   - Tester aussi les collections non shardées
   - Il est conseillé d'exécuter une requête `count` sur chacune des collections avant la sauvegarde pour vérifier lors de la restauration le bon compte.

Note : L'ansiblerie *VITAM* déploie dans chacune des instances mongoc et mongod des scripts préparés restore-mongoc.js et restore-mongod.js respectivement

- {{ vitam_defaults.folder.root_path }}/app/mongoc/restaure-mongoc.js
- {{ vitam_defaults.folder.root_path }}/app/mongod/restaure-mongod.js

---

Toutes les informations sur les adresses ip et numéros de ports de toutes les instances du cluster mongodb sont automatiquement renseignés dans ces scripts

Pour exécuter ces deux scripts, il faut lancer la commande suivante que vous pouvez automatiser dans un playbook :

```
 // Sur mongoc
> mongo {{ ip_service }}:{{ mongodb.mongos_port }}/admin {{ mongo_credentials }} {{␣
↪vitam_defaults.folder.root_path }}/app/mongoc/restore-mongoc.js
 // Sur mongod
> mongo {{ ip_service }}:{{ mongodb.mongos_port }}/admin {{ mongo_credentials }} {{␣
↪vitam_defaults.folder.root_path }}/app/mongod/restore-mongod.js
```

5.9.4 Cas particulier de l'offre froide

Dans le cas particulier d'une offre de stockage froide, les fichiers backup zip sont stockés dans des bandes magnétiques.

La procédure de backup du mongo de l'offre froide est très importante, car, la base de donnée est l'unique référentiel de l'ensemble des fichiers écris dans les bandes magnétiques.

Avertissement : Si les données du cluster mongodb de l'offre froide sont perdues, toutes les informations enregistrées sur les bandes magnétiques sont inutilisables. Pour cette raison, il est impératif de stocker les sauvegardes du cluster mongo de l'offre froide dans une bande magnétique.

5.9.4.1 Sauvegarde

5.9.4.1.1 Script de sauvegarde du cluster mongodb

Un playbook, ayant les tâches ci-dessous, a été mis en place pour faire un backup du mongodb de l'offre froide :

1. Détection des noeuds mongodb `Primary` (confiserver et shards)
2. Arrêt de *VITAM*
3. Copie et ajout d'un fichier de description des instances en cours
4. Compression du dossier db de chaque instance `Primary` (configserver et shards)
5. Démarrage de *VITAM*
6. Envoi des fichiers zip (via CURL) vers l'offre froide (composant offer sur url d'admin spécifique au traitement du backup) qui seront sauvegardés sur une bande magnétique

Pour exécuter le playbook :

Prudence : Le playbook ci-dessous est à exécuter uniquement sur un *VITAM* ayant une offre froide `**tapeLibrary**`

```
ansible-playbook ansible-vitam-exploitation/backup_mongodb_tape_offer.yml -i␣
↪environments/hosts.<environnement> --ask-vault-pass
```

5.9.4.1.2 Sauvegarde des fichiers backup dans l'offre froide

Lors de l'envoi des fichiers vers l'offre froide, cette dernière va procéder au traitement suivant :

- Réception du fichier zip dans une zone temporaire
- Copie du fichier dans une zone d'écriture sur bande magnétique
- Création d'un ordre spécifique pour écrire le fichier backup zip sur une bande magnétique ayant un tag `backup`
- Le worker « TapeDriveWorker » (Thread executé dans la jvm offer) qui va exécuter la tâche consigne son ordre d'écriture dans le fichier log `offer_tape_backup_DATE.log`, en détaillant les informations : `code de la bande magnétique`, `mongoc` ou `mongod (shard(i)`, `date`.

Note : Lors de la lecture depuis une bande magnétique, on accède aux fichiers sans connaître leur nom et leur type. Si on perd le cluster mongodb, le fichier de log `offer_tape_backup_DATE.log` sera l'unique moyen d'accéder rapidement au nom du fichier sauvegardé associé au code de la bande magnétique où il a été enregistré. Le nom `DATE-disk-mongod-shard01_.zip` que l'on récupère depuis le fichier log `offer_tape_backup_DATE.log` nous renseigne sur la date et le fait que ce soit un backup du `shard01`.

Avertissement : Après chaque sauvegarde, le fichier `offer_tape_backup_DATE.log` doit être copié dans un lieu sûr, pour le besoin de restauration en cas de perte du site. Dans le cas de la perte du site, si ce fichier n'est pas disponible, la lecture de toutes les bandes magnétiques sera l'unique moyen pour récupérer les fichiers de backup.

5.9.4.2 Restauration

5.9.4.2.1 Accès aux fichiers de l'offre froide

Sur l'offre froide, toutes les écritures des fichiers backup du mongodb de l'offre, sont tracées dans le fichier log `offer_tape_backup_DATE.log`

Pour récupérer une sauvegarde, il convient donc de consulter les lignes de log ayant comme information :

- Le code de la bande magnétique sur laquelle est écrit le fichier
- Le nom du fichier de la forme `DATE-disk-mongod-shard01_.zip`

Pour restaurer une date donnée

```
- Repérer dans le fichier log ``offer_tape_backup_DATE.log`` tous les fichiers backup␣
↪``(mongoc et mongod)`` zip correspondant à cette date ainsi que les bandes␣
↪magnétiques sur lesquelles les fichiers sont stockés
- Manuellement, charger les bandes magnétiques sur une ``tape-library`` pour lire les␣
↪fichiers
- La lecture des fichiers doit être réalisée en spécifiant les noms avec la␣
↪nomenclature adéquate (le nom se retrouve aussi à l'intérieur du fichier zip dans␣
↪un fichier descriptif)
- Copier et décompresser chacun de ces fichiers dans l'instance mongo correspondante.␣
↪Par exemple le fichier ayant pour nom ``DATE-disk-mongod-shard01_.zip`` est à␣
↪copier et à décompresser dans tous les membres mongo du shard ``shard01``
```

5.9.4.2.2 Restaurer le cluster mongodb

Une fois tous les fichiers copiés et décompressés dans les instances mongo correspondantes, il faut suivre la procédure de restauration décrite ci-dessus paragraphe **Restauration d'un cluster Mongo shardé**.

5.10 Gestion des profils de sécurité

La solution logicielle *VITAM* permet de gérer des profils de sécurité.

Le profil se base sur un contexte, lui-même basé sur une/des certificat(s).

Le processus d'installation met en place le profil de sécurité d'administration, qu'il est fortement recommandé de laisser « tel quel », car ce dernier est utilisé pour des actes d'exploitation.

Il n'existe actuellement pas de point d'API permettant de mettre à jour des profils de sécurité directement dans leur globalité, il sera donc nécessaire de les supprimer puis de l'ajouter à nouveau selon la procédure ci-dessous.

5.10.1 Hiérarchie : profils de sécurité, contextes et certificats

Afin d'avoir une vue ensembliste sur la relation entre les profils de sécurité, les contextes et les certificats déjà importés dans *VITAM*, il est possible d'avoir une liste complète des données de ces collections, en tenant compte de leur interdépendance.

Cette action est faite en lançant le playbook comme suit :

> ansible-playbook ansible-vitam-exploitation/listing_securityProfiles_contexts_certificates_hierarchy.yml -i environments/hosts.<environnement> –ask-vault-pass

Note : Les certificats personnels ne sont pas pris en compte dans ce listage, vu qu'ils dépendent plutôt du client connecté et non pas du contexte.

5.10.2 Ajout/Suppression de profils de sécurité

Avertissement : Cette version est encore en cours de mise en place et est susceptible d'évoluer.

5.10.2.1 Configuration

Un playbook d'exploitation permet de rajouter des profils de sécurité.

Sur la machine de déploiement, il est nécessaire de configurer le fichier `deployment/environments/group_vars/all/postinstall_param.yml`, dans la section `vitam_additional_securityprofiles`.

Exemple :

```
---

vitam_additional_securityprofiles:
  - name: applicative01
    identifier: spidentifier01 # manadatory, cannot be naither null nor blank
    hasFullAccess: false # true/false
    permissions: "null" # possible keypairs are stored in "modèle de données
↪"  ; for example : "contexts:read"
    contexts:
      - name: contextappli01
        identifier: myContextIdentifier # please leave blank if associated␣
↪tenants are "master" (if over release 9,related to existence of CONTEXT in␣
↪vitam_tenants_usage_external directive for tenant)
        status: ACTIVE # ACTIVE or INACTIVE
        enable_control: true # can be true or false , must be true if fine␣
↪permissions tuning (see above) ; if false, no permissions
        permissions: "[ { \"tenant\": 2, \"AccessContracts\": [], \
↪"IngestContracts\": [] }]" # you can specify different permissions for all␣
↪tenants that have to be associated to "[{perms1},{perms2}, ...]" where␣
↪permsX is  something like { \"tenant\": 2, \"AccessContracts\": [], \
↪"IngestContracts\": [] }
        certificates: ['cert1.crt'] # list of public certificates files␣
↪stored in {{inventory_dir}}/certs/client-external/clients/
```

Note : les certificats devraient être de type `external/${fichier crt}`.

5.10.2.2 Ajout des fichiers crt

Placer les certificats précédemment renseignés (fichiers crt) dans {{inventory_dir}}/certs/client-external/clients/external/.

5.10.2.3 Lancement du playbook

- L'ajout de tous les profils de sécurité renseignés dans postinstall_param.yml se fait en lançant le playbook comme suit :

  ansible-playbook ansible-vitam-exploitation/add_contexts.yml -i environments/hosts.<environnement> –ask-vault-pass

Prudence : Ce playbook ne sait gérer que le cas d'ajout de profils/contextes/... . Il convient de s'assurer au préalable que les champs *name* et *identifier* à ajouter n'existent pas déjà dans la solution logicielle *VITAM*.

- L'ajout d'un seul profil de sécurité issu du fichier postinstall_param.yml se fait en lançant le playbook comme suit :

  ansible-playbook ansible-vitam-exploitation/add_contexts.yml -i environments/hosts.<environnement> -e security_profile_id= »<security_profile_id> » –ask-vault-pass

- La suppression de tous les profils de sécurité se fait en lançant le playbook comme suit :

  ansible-playbook ansible-vitam-exploitation/remove_contexts.yml -i environments/hosts.<environnement> –ask-vault-pass

Prudence : Ce playbook supprime l'ensemble des profils/contextes définis dans le fichier `environments/group_vars/all/postinstall_param.yml`.

- La suppression d'un seul profil de sécurité se fait en lançant le playbook comme suit :

  ansible-playbook ansible-vitam-exploitation/remove_contexts.yml -i environments/hosts.<environnement> -e security_profile_id= »<security_profile_id> » –ask-vault-pass

Prudence : Les opérations sur le contexte de sécurité admin-context sont interdites car réservé à Vitam.

- L'ajout d'un certificat pour un profil de sécurité existant se fait de la manière suivante :

  ansible-playbook ansible-vitam-exploitation/update_context.yml -i environments/hosts.<environnement> -e security_profile_id= »<security_profile_id> » –ask-vault-pass

Prudence : Le profil de sécurité doit exister et il ne faut pas que plusieurs certificats identiques soient insérés !

5.10.2.4 Reconfiguration de VITAM

À l'issue de la bonne exécution du playbook, il faut relancer un déploiement partiel de *VITAM* pour les groupes ansible `[hosts_ingest_external]` et `[hosts_access_external]`

5.10.2.4.1 Si utilisation de la PKI de tests

La procédure décrite ci-dessous est à appliquer dans le cas où la *PKI* de tests a été employée.

Ajouter les informations relatives au(x) certificat(s) supplémentaire(s) via la commande

```
ansible-vault edit environments/certs/vault-certs.yml --ask-vault-pass
```

Ajouter un couple clef/valeur pour chaque certificat supplémentaire selon le modèle suivant

```
client_client-external_<nom complet du fichier crt avec extension>_key: <la valeur du
↪mot de passe>
```

Exemple :

client_client-external_appliexterne.crt_key : Motd3P@sse !

Note : appliexterne ne doit pas contenir de caractère « -«

Avertissement : Si le certificat à ajouter a été généré avec une *CA* non-connue de VITAM, il faut ajouter au bon endroit la clé publique (se référer au *DIN* pour plus d'informations).

Prudence : Un fichier `crt` ne doit contenir qu'une clef publique

Ensuite, regénérer les *stores* Java avec les certificats supplémentaires (script `generate_stores.sh` ; se référer au *DIN* pour plus d'informations)

5.10.2.4.2 Cas d'une autre PKI

Mettre à jour les *stores* java avec les certificats supplémentaires à *truster*.

5.10.2.4.3 Application des *stores* mis à jour

Rejeu du déploiement en limitant aux groupes ansible `[hosts_ingest_external]` et `[hosts_access_external]` et avec le tag ansible `update_vitam_certificates`.

Exemple :

ansible-playbook ansible-vitam/vitam.yml -i environments/hosts.<environnement> –ask-vault-pass –limit hosts_ingest_external,hosts_access_external –tags update_vitam_certificates

5.11 Certificats personae

5.11.1 Configuration des permissions des certificats personae

il est possible de configurer les endpoints nécessitant une authentification personae via le fichier *deployment/ansible-vitam/roles/vitam/templates/security-internal/personal-certificate-permissions.conf.j2* Les endpoints qui requièrent une authentification personnelle doivent être déplacés de **permissionsWithoutPersonalCertificate** vers **permissionsRequiringPersonalCertificate**

Exemple :

```
permissionsRequiringPersonalCertificate:
  - 'elimination:action'

permissionsWithoutPersonalCertificate:
  - 'dipexport:create'
  - 'dipexportv2:create'
  - 'dipexport:id:dip:read'
```

5.11.2 Déploiement des certificats personae

5.11.2.1 Vitam n'est pas encore déployé

Déployer Vitam en suivant la procédure indiquée dans le *DIN*.

Note : le certificat personnel doit être pré-provisionné dans Vitam, Ceci est réalise au moment de l'installation via le paramètre admin_personal_certs du fichier de configuration {{ inventory_dir }}/group_vars/all/vitam_security.yml, et en renseignant les certificats (clés publiques) dans le bon dossier {{ inventory_dir }}/certs/client-vitam-users/clients.

5.11.2.2 Vitam est déjà déployé

Suivre la procédure de la section *Intégration d'une application externe dans Vitam* (page 243).

5.12 Gestion des indexes Elasticseach dans un contexte massivement multi-tenants

5.12.1 Présentation

Prudence : Attention, cette fonctionnalité est compliquée à utiliser et n'est nécessaire qu'en cas de contexte massivement multi-tenants. Il n'est pas utile de regrouper des tenants en dessous d'un usage inférieur à 10 tenants. Au delà de 50 tenants, cette fonctionnalité peut être utile mais uniquement dans le cas où une majorité de ces 50 tenants contiennent peu de données.

L'objectif de cette fonctionnalité est de pouvoir gérer plusieurs milliers de tenants en limitant les impacts sur les performances du cluster elasticsearch-data. Initialement, Vitam était prévu pour une gestion d'une dizaine de tenants avec l'utilisation de la topologie suivante pour la gestion des indexes dans la base elasticsearch-data :

- Masterdata (accesscontract, ingestcontract, ontology, formats…) : 1 index pour tous les tenants
- unit, logbookoperation, objectgroup : 1 index par tenant

Ainsi, cette nouvelle fonctionnalité offre la possibilité de regrouper certains tenants qui contiendraient peu de données afin d'éviter d'avoir un index dédié pour chaque tenant.

Prudence : Attention, en cas de modification de la distribution des tenants, une procédure de réindexation de la base elasticsearch-data est nécessaire. Cette procédure est à la charge de l'exploitation et nécessite un arrêt de service sur la plateforme. La durée d'exécution de cette réindexation dépend de la quantité de données à traiter.

5.12.2 Recommandations d'implémentation

En fonction de la quantité de données à traiter, il est conseillé de prévoir des plages de tenants en fonction de leur futur usage. Il est tout à fait possible de prévoir à l'avance des plages de tenants même si ils ne sont pas encore utilisés. Cela permet d'anticiper leur gestion future.

Par exemple, une recommandation serait de définir la convention suivante pour les plages de tenants en fonction de leur usage :

- [1-99] : Plage pour les gros tenants qui nécessitent un index dédié.
- [100-999] : Plage pour les tenants de taille moyenne qui pourraient être regroupés par dizaines.
- [1000-9999] : Plage pour les petits tenants qui pourraient être regroupés par centaines.

En fonction de l'usage qui sera fait de certains tenants, il peuvent être amenés à être regroupés différemment. Un suivi de l'évolution de la quantité de données dans chacun des tenants et groupe de tenants est nécessaire afin d'anticiper les modifications nécessaires à effectuer en fonction des regroupements.

5.13 Batchs et traitements

5.13.1 Curator

Il existe des jobs Curator de :

- fermeture d'index
- suppression d'index fermés

Ces jobs sont lancés via `crontab` toutes les nuits.

5.13.2 *Timers* systemD

Tous les *timers* systemD décrits ci-dessous sont paramétrables ; un comportement par défaut est appliqué. Se reporter à la procédure *Modifier la fréquence de lancement de certains timers systemD* (page 16) pour la bonne prise en compte du paramétrage attendu ou souhaité.

5.13.2.1 Sécurisation des journaux d'opérations

Un *timer* systemd a été mis au point pour réaliser ces actions :

- *vitam-traceability-operations* (page 72)

Ce *timer* est installé avec le composant `logbook`.

5.13.2.2 Sécurisation des cycles de vie

Des *timers* systemd ont été mis au point pour réaliser ces actions :

- *vitam-traceability-lfc-unit* (page 72)
- *vitam-traceability-lfc-objectgroup* (page 73)

Ces *timers* sont installés avec le composant `logbook`.

5.13.2.3 Sécurisation des offres de stockages

Des *timers* systemd ont été mis au point pour réaliser ces actions :

- *vitam-storage-log-backup* (page 71)
- *vitam-storage-log-traceability* (page 72)

Ces *timers* sont installés avec le composant `storage`.

5.13.2.4 Autres *timers*

Les *timers* suivants sont apportés par le composant `functional_administration`

```
vitam-create-accession-register-symbolic
vitam-functional-administration-accession-register-reconstruction
vitam-rule-management-audit
vitam-functional-administration-reconstruction
```

Les *timers* suivants sont apportés par le composant `metadata`

```
vitam-metadata-store-graph
vitam-metadata-reconstruction
vitam-metadata-computed-inherited-rules
vitam-metadata-purge-dip
vitam-metadata-purge-transfers-SIP
vitam-metadata-audit-mongodb-es
```

5.14 Sauvegarde des données graphe (Log shipping)

La sauvegarde des données graphe des métadonnées (UNIT/*GOT*) consiste à récupérer au fil de l'eau depuis la base de données (MongoDB) les données graphe par (UNIT/GOT) pour les stocker dans les offres de stockage.

Prudence : En cas de problème de sauvegarde des données *graphe*, on écrit dans le fichier log une `[Consistency Error]` qu'il conviendra de surveiller.

Avertissement : Si l'instance qui démarre le service de sauvegarde s'arrête, il faut lancer ce service de sauvegarde dans une autre instance.

5.14.1 Déclenchement de la sauvegarde

La sauvegarde des données *graphe* est lancée via un *timer* systemd (*vitam-metadata-store-graph* (page 75)), qui démarre le service systemd associé.

- Le timer se lance chaque 30 minutes (par défaut, modifiable selon le besoin - se reporter à *Modifier la fréquence de lancement de certains timers systemD* (page 16) -)
- Le sauvegarde des données *graphe* se fait sur un intervalle de temps (depuis la dernière sauvegarde jusqu'au temps présent)
- Le fichier généré est un fichier au format `zip`, qui contient un ou plusieurs fichiers `JSON`. Ces fichiers `JSON` contiennent un tableau de données *graphe*.

- Le nom du fichier de la dernière sauvegarde contient les dates début et fin de sauvegarde. Ce nom est utilisé pour déterminer la dernière date de sauvegarde.
- La sauvegarde des UNIT est séparée de celle des *GOT* (Deux *containers* distincts dans chaque offre de stockage)

5.14.2 Reconstruction des données *graphe*

La reconstruction des données *graphe* se fait avec le même principe que la reconstruction des méta-données (UNIT/*GOT*) :

- Gérer l'offset de reconstruction des fichiers de sauvegarde des données graphe.
- Mettre à jour uniquement les données graphe. Si un document n'est pas trouvé, une création de ce document est faite et ne contiendra que les données du graphe.
- De même, la reconstruction des métadonnées ne modifie pas les données graphe potentiellement déjà existantes.
- Une purge est faite de tous les documents ayant uniquement les données graphe et qui sont vieux de (1 mois Configurable)
- Les documents ayant uniquement les données graphe ne sont pas indexés dans ElasticSearch.
- La reconstruction est séquentielle (D'abord les métadonnées UNIT/*GOT* ensuite leur graphe)

Voir aussi :

Se reporter à la procédure de *Reconstruction* (page 36) des métadonnées pour plus d'informations.

5.15 Recalcul des données graphe

Il est possible de recalculer les données du graphe en utilisant une requête *DSL*. En effet, dans le cadre de la procédure de *PRA*, il est nécessaire de pouvoir détecter les unités archivistiques ayant un graphe incohérent (construire le DSL requis) selon la procédure de déclenchement décrite ci-dessous.

Le recalcul de graphe permet de rétablir la cohérence des données *VITAM*.

Prudence : Cette procédure s'applique à partir de la version *VITAM* R8 (1.10.0).

Prudence : En cas de données de graphe incohérentes, le résultat des requêtes *DSL* sur les unités archivistiques pourra être incorrect et l'application des filtres de sécurité définis dans les contrats d'accès pourront ne pas être pris en compte.

5.15.1 Déclenchement

Le recalcul de *graphe* est déclenché par l'appel au point d'API porté par l'URL suivante sur le composant `metadata`

```
http://{{ ip_admin }}:{{ vitam.metadata.port_admin }}/metadata/v1/computegraph
```

Exemple d'appel à l'aide de curl :

```
curl -s -X POST -H "X-Tenant-Id: <tenant>" -H "Content-Type: application/json" --user
↪"${VITAM_ADMIN_AUTH}" --data @${CURRENT_DIR}/dslQuery.json ${URL}
```

Exemple de query DSL (dslQuery.json) :

```
{
  "$roots": [
    "aeaqaaaaaqhdytymabdeialenehzphiaaaeq",
    "aeaqaaaaaqhdytymabdeialenehzpbyaaajq"

... , "guid_n"
  ],
  "$query": [],
  "$projection": {}
}
```

La valeur utilisée pour la basic authentication prend la forme suivante

```
VITAM_ADMIN_AUTH={{ admin_basic_auth_user }}:{{ admin_basic_auth_password }}
```

- Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `deployment/environments/group_vars/all/vitam_security.yml`
- Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `deployment/environments/group_vars/all/vault-vitam.yml`

Prudence : Si le *DSL* ne contient pas uniquement `$root`, alors la valeur du tenant positionnée dans le header `X-Tenant-Id` est essentielle car les index elasticsarch sont organisés par tenant.

5.16 Montée de version du fichier de signature de Siegfried

La solution logicielle *VITAM* utilise l'outil Siegfried pour la détection des formats des fichiers contenus dans les *SIP*. La nomenclature des formats est basée sur la norme « DROID » fournie par les archives nationales anglaises [14].

Ce fichier est régulièrement mis à jour et il est recommandé de procéder à sa mise en place dans *VITAM*.

Pour ce faire, il faut récupérer le fichier PRONOM [15] depuis les archives nationales anglaises et l'ajouter dans le répertoire `deployment/environments/`.

Il faut ensuite modifier la valeur de la directive `droid_filename` dans le fichier `deployment/environments/group_vars/all/vitam_vars.yml` (avec le nom du fichier récupéré précédemment).

Enfin, il faut lancer le playbook ansible suivant depuis le répertoire `deployment/`

```
ansible-playbook ansible-vitam/roy_build_signature.yml -i environments/hosts.
↪<environnement> --ask-vault-pass
```

5.17 Griffins

Note : Nouveauté introduite en R9.

Afin de prendre en compte des considérations de réidentification et/ou préservation, la solution logicielle *VITAM* intègre désormais des *griffins* - greffons - pour réaliser les actions d'analyse, (ré)identification et préservation.

http://www.nationalarchives.gov.uk/
http://www.nationalarchives.gov.uk/information-management/manage-information/preserving-digital-records/droid/

Comme décrit dans le *DIN*, le choix des *griffins* installés est défini dans le fichier `environments/group_vars/all/vitam_vars.yml` au niveau de la directive `vitam_griffins`.

Prudence : Cette version de la solution logicielle *VITAM* ne mettant pas encore en oeuvre de mesure d'isolation particulière des *griffins*, il est recommandé de veiller à ce que l'usage de chaque *griffin* soit en conformité avec la politique de sécurité de l'entité. Il est en particulier déconseillé d'utiliser un *griffin* qui utiliserait un outil externe qui n'est plus maintenu.

5.17.1 Ajout de nouveaux / mise à jour de *griffins*

Il est possible d'ajouter ou mettre à jour des *griffons* à une installation de la solution logicielle *VITAM*.

5.17.1.1 Ajout de *griffins*

Pour cela, il faut modifier le fichier `environments/group_vars/all/vitam_vars.yml` au niveau de la directive `vitam_griffins` par ajout du/des *griffon(s)* dans la liste.

Exemple d'ajout du greffon **vitam-un-nouveau-greffon-qui-est-nécessaire**

```
vitam_griffins: ["vitam-imagemagick-griffin", "vitam-jhove-griffin", "vitam-un-
↪nouveau-greffon-qui-est-nécessaire"]
```

5.17.1.2 Mise à jour des *griffins*

Dans le cadre d'une montée de version des composants *griffins*, le playbook se charge de déployer les composants les plus à jour ; il n'est pas nécessaire de modifier la directive `vitam_griffins`.

Note : Ne pas oublier, sur les partitions associées, de mettre à jour, si nécessaire, l'adresse du dépôt de binaires.

5.17.1.3 Préparation du système

Il faut également prévoir de mettre à disposition sur le(s) dépôt(s) de binaires les packages d'installation correspondants. Le nom indiqué dans la liste doit correspondre au nom du package (format rpm ou deb selon la plateforme). Les packages d'installation associés doivent se situer dans un dépôt accessible et connu par la machine sur laquelle ils vont être installés.

5.17.1.4 Prise en compte technique par VITAM

Enfin, il suffit de relancer le playbook d'installation de *VITAM* avec, en fin de ligne, cette directive

```
--tags griffins
```

5.18 Reconstruction

La reconstruction consiste à recréer le contenu des bases de données (MongoDB-data, Elasticsearch-data) en cas de perte de l'une ou l'autre à partir des informations présentes dans les offres de stockage. Elle part du principe que le contenu des offres n'a pas été altéré.

Prudence : Dans cette version de la solution logicielle *VITAM*, la reconstruction nécessite de couper le service aux utilisateurs.

Prudence : Une reconstruction complète à partir des offres de stockage peut être extrêmement longue, et ne doit être envisagée qu'en dernier recours.

5.18.1 Procédure multi-sites

5.18.1.1 Cas du site primaire

La reconstruction se réalise de la manière suivante :

1. Arrêt de *VITAM* sur le site à reconstruire
   - Utiliser le playbook `ansible-vitam-exploitation/stop_vitam.yml`

Il est indispensable de valider que tous les services *VITAM* (y compris les *timers* systemd) sont bien arrêtés

2. Purge des données (le cas échéant) stockées dans MongoDB-data, excepté les bases **identity**, **config** et **admin** :
   - Utiliser le playbook `ansible-vitam-exploitation/start_mongodb.yml` pour démarrer les bases mongodb
   - **Procéder à la purge des données en utilisant l'outil mongo shell ou un outil équivalent :**
     - Se connecter avec l'utilisateur `vitamdb-admin`
     - Lister les bases via la commande `show dbs`
     - Pour chacune des bases, excepté les bases **identity**, **config** et **admin**, les vider via la commande `db.getCollectionNames().forEach(function(x) {db[x].remove({})});`
   - Utiliser le playbook `ansible-vitam-exploitation/stop_mongodb.yml` pour stopper les bases mongodb
3. Purge des données (le cas échéant) stockées dans Elasticsearch-data :
   - Utiliser le playbook `ansible-vitam-exploitation/start_elasticsearch_data.yml` pour démarrer elasticsearch-data
   - Dans le cas où Cerebro ou un outil équivalent est disponible, lister les indexes et les purger via l'IHM
   - **Sinon :**
     - Se connectant en ssh sur un des nœuds elasticsearch-data
     - Lister les indexes ES via `curl 'http://localhost:9200/_cat/indices?v'`
     - Pour chacun des indexes, vider l'index via : `curl -XDELETE 'http://localhost:9200/{index_name}'`

     (Pour cette action de purge d'elasticsearch-data un playbook équivalent est disponible : `ansible-vitam-exploitation/clean_indexes_es_data.yml`)
   - Utiliser le playbook `ansible-vitam-exploitation/stop_elasticsearch_data.yml` pour stopper elasticsearch-data
4. Reconfiguration et démarrage en tant que site secondaire :

- Paramétrer la variable `primary_site` à `false` dans le fichier `environmenrs/group_vars/all/vitam_vars.yml` puis utiliser le playbook `ansible-vitam/vitam.yml`

5. Dès lors, l'accès utilisateur reste coupé, et l'intégralité des données est reconstruite progressivement
   - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB-data
   - Pour la release 8, la procédure est décrite dans la section « Recalcul des données graphe »
6. La collection `Offset` de la base de données `metadata` est créée et permet de suivre l'avancement de la reconstruction.
7. Une fois la reconstruction terminée (plus de modification dans la collection `Offset`), il convient de reconfigurer en tant que site primaire, puis redémarrer :
   - Paramétrer la directive `primary_site` à `true` puis utiliser le playbook `ansible-vitam/vitam.yml`

5.18.1.2 Cas du site secondaire

La reconstruction se réalise de la manière suivante :

1. Arrêt de *VITAM* sur le site à reconstruire
   - Utiliser le playbook `ansible-vitam-exploitation/stop_vitam.yml`

Il est indispensable de valider que tous les services *VITAM* (y compris les *timers* systemd) sont bien arrêtés.

2. Purge des données (le cas échéant) stockées dans MongoDB-data, excepté les bases **identity**, **config** et **admin** (procédure identique au cas du site primaire)
3. Purge des données (le cas échéant) stockées dans Elasticsearch-data (procédure identique au cas du site primaire)
4. Redémarrage du site secondaire Vitam
   - Utiliser le playbook `ansible-vitam-exploitation/start_vitam.yml`
   - La prochaine itération de reconstruction au fil de l'eau redémarrera la reconstruction à partir du début
   - Attendre la fin de la reconstruction au fil de l'eau sur le site secondaire
     - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB-data
     - Pour la release 7 (version 1.4.x) il faut lancer le service dédié `vitam-metadata-graph-builder.service` sur le composant metadata pour recalculer le graphe des unités archivistiques et des groupes d'objets techniques n'ayant pas encore reconstruit leurs données graphe

5.18.2 Procédure mono-site

La procédure à appliquer est la même que la procédure du site primaire pour une installation multi-sites.

5.18.2.1 Contrôle des données reconstruites

La reconstruction des objets en base de données que ce soit sur MongoDB-data ou Elasticsearch-data est un processus long. Afin de contrôler si tous les objets ont été reconstruits ou si la reconstruction est toujours en cours il est nécessaire de compter les objets des collections Units et ObjectGroups de la base Metadata.

Un playbook a été réalisé afin de réaliser ce comptage à la fois sur Elasticsearch-data et sur MongoDB-data.

Il s'execute pour le site où la reconstruction se fait, via :

- ansible-playbook -i environments/<host site en cours de reconstruction> ansible-vitam-exploitation/reconstruction_doc_count.yml -e site= »reconstruction » –vault-password-file vault_pass.txt

Ou pour le site servant de référence :

- ansible-playbook -i environments/<host site de référence> ansible-vitam-exploitation/reconstruction_doc_count.yml -e site= »reference » –vault-password-file vault_pass.txt

Pour chaque execution un fichier résultat est produit : - "doc_count_site_ref.report" pour le site de référence - "doc_count_site_reconst.report" pour le site qui se reconstruit

5.19 Plan de Reprise d'Activité (PRA)

Le *PRA* consiste à passer un site *VITAM* secondaire en site primaire après incident majeur survenu sur le site primaire (cas de l'indisponibilité complète du site primaire).

Note : Les actions en cours sur le site primaire sont perdues (versements non terminés, batchs etc.). L'incohérence des données sera traitée dans une version ultérieure de *VITAM*..

Cette section décrit des actions qui ne peuvent donc s'effectuer que si une installation multi-sites a été effectuée au préalable.

Cette section s'appuie sur les procédures décrites dans les chapitres suivants :

- Resynchronisation d'une offre à partir d'une autre offre (*Resynchronisation d'une offre* (page 40))
- Reconstruction des bases de données (MongoDB-data, Elasticsearch-data) en cas de perte de l'une ou l'autre, à partir des informations présentes dans les offres (*Reconstruction* (page 36))

5.19.1 Déclenchement

Avant le déclenchement de la procédure de *PRA*, le système fonctionne en mode multi-sites (primaire/secondaire). Le service est indisponible à la suite de la perte du site primaire.

Le déclenchement du *PRA* s'effectue selon la procédure suivante :

1. Vérifier que le site primaire est bien complètement arrêté
   - Il est indispensable de valider que tous les services VITAM (y compris les *timers* systemd) sont bien arrêtés
2. Attendre la fin de la reconstruction au fil de l'eau sur le site secondaire
   - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB-data.
3. Reconfigurer le site secondaire en site primaire :
   - Attention à adapter la stratégie de stockage en fonction du mode d'utilisation choisi pour le site de secours :
     - Mode « lecture/écriture » : la stratégie de stockage doit être modifiée afin de limiter les écritures aux seules offres encore disponibles sur le site de secours (*Activation/désactivation d'une offre* (page 55))
     - Mode « lecture seule » (recherche et consultation avec profil de droits dédié) : la stratégie de stockage ne change pas. Seule une reconfiguration du site primaire initial en mode secondaire permettra le retour à un fonctionnement nominal (cf. ci-dessous)
   - Paramétrer la variable `primary_site` à `true` dans le fichier d'inventaire puis jouer le playbook `ansible-vitam/vitam.yml`
   - Si le site secondaire était partiellement déployé, ne pas oublier de rajouter tous les composants requis pour un fonctionnement en site primaire.
4. En lien avec le processus de reconstruction (cf. *Reconstruction* (page 36)), en cas de bascule sur le site secondaire, il sera préférable de purger les documents des Unit et ObjectGroup reconstruits mais ne contenant potentiellement que des données de graphe (cas de l'éliminitaion par exemple). Cette opération s'effectue avec la commande suivante :

```
curl -s -X DELETE -H "X-Tenant-Id: {{ vitam_tenant_admin }}" -H "Accept: application/
↪json" -H "Content-Type: application/json" --user "{{ admin_basic_auth_user }}:{{␣
↪admin_basic_auth_password }}" http://{{ ip_admin }}:{{ vitam.metadata.port_admin }}/
↪metadata/v1/purgeGraphOnlyDocuments/[UNIT | OBJECTGROUP | UNIT_OBJECTGROUP]
```

Après modification des accès pour les applications versantes (action infra. de type modification DNS, routage, conf etc.), le site secondaire peut alors être ouvert au service en tant que site primaire.

Le système fonctionne désormais en mode mono-site (primaire). Le service est de nouveau disponible sur le site de secours.

5.19.2 Retour en situation nominale

Le retour à la solution nominale s'effectue en deux étapes :

- Rétablissement du contenu du site primaire initial par reconfiguration temporaire en tant que site secondaire
- Retour à la configuration multi-sites initiale

Avertissement : Dans cette version, la resynchronisation partielle d'une offre de stockage n'étant pas supportée, le retour à la configuration multi-sites initiale nécessite de repartir d'offres vierges de toutes données sur le site à resynchroniser (on parle ici d'offre de remplacement)

5.19.2.1 Déclenchement

Avant déclenchement de la procédure de *PRA* inverse (retour en situation nominale), le système fonctionne en mode mono-site (primaire). Le service est disponible sur le site de secours.

Le déclenchement du *PRA* inverse s'effectue selon la procédure suivante :

- Vérifier que le site primaire initial est bien complètement arrêté
  - Il est indispensable de valider que tous les services VITAM (y compris les *timers* systemd) sont bien arrêtés
- Purger les données (le cas échéant) stockées dans MongoDB-data, excepté les bases **identity**, **config** et **admin**
- Purger les données (le cas échéant) stockées dans Elasticsearch-data
- Reconfigurer et démarrer le site primaire initial en tant que site secondaire :
  - Paramétrer la variable `primary_site` à `false` dans le fichier d'inventaire puis jouer le playbook `ansible-vitam/vitam.yml`
  - Le mécanisme de reconstruction du contenu des bases de données (MongoDB-data, Elasticsearch-data) à partir des informations présentes dans les offres de stockage est actif (aucune donnée à resynchroniser à cette étape)
- Resynchroniser les offres de stockage à partir des offres du site de secours en se référant à la procédure suivante *Resynchronisation d'une offre* (page 40)
  - En fonction du mode d'utilisation choisi pour le site de secours :
    - Mode lecture/écriture : la stratégie de stockage du site de secours doit auparavant être modifiée afin de référencer de nouveau les offres du site primaire initial
    - Mode lecture seule : la stratégie de stockage ne change pas. Les offres du site primaire initial sont toujours connues du site de secours
- Le mécanisme de reconstruction au fil de l'eau reconstruit progressivement le contenu des bases de données
  - Le suivi de la reconstruction se fait en observant l'évolution de l'offset de reconstruction stocké dans MongoDB data

    - Pour la release 7 (version 1.4.x) il faut lancer le service dédié `vitam-metadata-graph-builder.service` sur le composant metadata pour recalculer les données graphe des unités archivistiques et des groupes d'objets techniques n'ayant pas encore reconstruit leurs données graphe
- Une fois la reconstruction terminée, reconfiguration en tant que site primaire et démarrage :
    - Paramétrer la variable `primary_site` à `true` dans le fichier d'inventaire puis jouer le playbook `ansible-vitam/vitam.yml`
- Reconfiguration et démarrage en tant que site secondaire du site de secours :

> **Avertissement :** Cette opération provoque une indisponibilité temporaire des principaux services *VITAM* (versement, gestion, recherche et consultation)

    - Paramétrer la variable `primary_site` à `false` dans le fichier d'inventaire puis jouer le playbook `ansible-vitam/vitam.yml`

Après modification des accès pour les applications versantes (action infra. de type modification DNS, routage, conf etc.), le site primaire initial peut alors être de nouveau ouvert au service en tant que site primaire.

Le système fonctionne désormais de nouveau en mode multi-sites (primaire/secondaire). Le service est de nouveau disponible sur le site primaire initial.

5.20 Resynchronisation d'une offre

Une offre de stockage peut être désynchronisée par rapport à une autre à la suite d'une indisponibilité plus ou moins longue voire totale de l'offre (*crash* majeur du système, panne matérielle etc.) ou bien encore à la suite d'une mise en maintenance programmée.

Le mécanisme de resynchronisation d'une offre par rapport à une autre nécessite une intervention d'exploitation manuelle permettant de remédier à la perte de données dans le système.

Note : En cas d'indisponibilité d'une offre, le processus d'entrée d'un *SIP* n'étant réussi que si et seulement si toutes les offres de stockage définies dans la stratégie sont accessibles, et que tous les fichiers sont bien copiés sur la totalité de ces offres, il sera nécessaire de désactiver l'offre (cf. chapitre *Activation/désactivation d'une offre* (page 55)) afin de permettre à nouveau les entrées de *SIP* (ingest/versement).

> **Prudence :** Il est recommandé de procéder à un audit d'intégrité dans le cadre d'opérations techniques ciblées, telles que l'évolution de la stratégie de stockage, un changement de stockage.

5.20.1 Cas de l'ajout d'une nouvelle offre

> **Avertissement :** Lors de l'ajout d'une nouvelle offre (portant un nouvel identifiant d'offre), les métadonnées des AU / GOT existants ne seront pas mis à jour avec l'information sur la nouvelle stratégie de stockage utilisée. L'ajout d'un mécanisme de mise à jour / propagation des métadonnées est prévu dans une version ultérieure. Lors de l'ajout d'une offre en remplacement d'une précédente offre, l'intégrité des métadonnées sera garantie en conservant le même identifiant d'offre.

L'ajout d'une nouvelle offre de stockage requiert le déploiement des applicatifs *VITAM* associés selon la procédure suivante :

- Éditer le fichier de configuration de l'inventaire de déploiement (généralement, fichier `hosts`) afin d'ajouter les nouveaux serveurs portant les composants à déployer (fonction de la topologie de déploiement retenue) :

```
[hosts_storage_offer_default]
hostname-new-storage-offer      offer_conf=<offer-z>

[hosts_mongos_offer]
hostname-new-mongos-offer       offer_conf=<offer-z>

[hosts_mongoc_offer]
hostname-new-mongoc-offer       offer_conf=<offer-z>

[hosts_mongod_offer]
hostname-new-mongod-offer       offer_conf=<offer-z>
```

- Éditer le fichier de configuration de la stratégie de stockage `deployment/environments/group_vars/all/offer_opts.yml` afin d'ajouter la nouvelle offre :

```
vitam_strategy:
- name: <offer-x>
    referent: true
    rank: 0
# <offer-z> is the new offer
- name: <offer-z>
    referent: false
    rank: 10

vitam_offers:
    <offer-x>:
        provider: filesystem
    # <offer-z> is the new offer
    <offer-z>:
        provider: filesystem
```

Si la nouvelle offre est utilisée dans une stratégie additionnelle (`other_strategies`), la modification sera la suivante :

```
other_strategies:
    metadata:
    - name: <offer-x>
        referent: false
        rank: 0
    # <offer-z> is the new offer
    - name: <offer-z>
        referent: false
        rank: 10
vitam_offers:
    <offer-x>:
        provider: filesystem
    # <offer-z> is the new offer
    <offer-z>:
        provider: filesystem
```

- Éditer le fichier de déclaration des secrets généraux `deployment/environments/group_vars/all/vault-vitam.yml` afin d'y ajouter les secrets associés à la nouvelle offre :

```
mongodb:
    <offer-z>:
```

(suite sur la page suivante)

(suite de la page précédente)

```
        passphrase: <passphrase>
        admin:
          user: <admin-user>
          password: <admin-password>
        localadmin:
          user: <localadmin-user>
          password: <localadmin-password>
        offer:
          user: <offer-user>
          password: <offer-password>
```

- Exécuter la commande suivante afin de déployer les nouveaux composants storage-offer, mongos-offer, mongoc-offer, mongod-offer :

Note : On considère que les étapes de génération des *hostvars*, de génération des magasins de certificats et de mise en place des repositories *VITAM* ont été réalisées au préalable pour les serveurs concernées (se référer aux sections du *DIN* correspondantes).

```
ansible-playbook ansible-vitam/vitam.yml -i environments/hosts.<environnement> --ask-
↪vault-pass --limit "hostname-new-storage-offer,hostname-new-mongos-offer,hostname-
↪new-mongoc-offer,hostname-new-mongod-offer"
```

La nouvelle offre doit ensuite être déclarée dans la stratégie de stockage par reconfiguration du moteur de stockage selon la procédure suivante :

Avertissement : Cette opération provoque une indisponibilité temporaire des principaux services *VITAM* (versement, gestion, recherche et consultation)

- Exécuter la commande suivante afin de reconfigurer le composant storage-engine :

```
ansible-playbook ansible-vitam/vitam.yml -i environments/hosts.
↪<environnement> --ask-vault-pass --limit hosts_storage_engine --tags␣
↪update_vitam_configuration
```

5.20.2 Procédure de resynchronisation d'une offre

La resynchronisation d'une offre à partir du contenu d'une autre offre s'effectue en suivant la procédure suivante :

Note : Cette procédure n'impacte pas les services *VITAM*. Le mécanisme de reconstruction du contenu des bases de données (MongoDB-data, Elasticsearch-data) à partir des informations présentes dans les offres de stockage fonctionne de manière concurrente au mécanisme de resynchronisation.

- Exécuter la commande suivante afin de resynchroniser la nouvelle offre vis-à-vis de l'offre (des offres) source(s) :

```
curl -v -X POST -u adminUser:adminPassword --header 'content-type:␣
↪application/json' --header 'accept: application/json' http://
↪<storageengine>:29102/storage/v1/offerSync --data '
{
```

(suite sur la page suivante)

(suite de la page précédente)

```
    "sourceOffer": "<offer-x>.service.consul",
    "targetOffer": "<offer-z>.service.consul",
    "strategyId": <strategyId>,
    "container": <datatype>,
    "tenantId": <tenantId>
}'
```

- Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `deployment/environments/group_vars/all/vitam_security.yml`
- Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `deployment/environments/group_vars/all/vault-vitam.yml`
- Le paramètre `sourceOffer` correspond à l'**id** de l'offre source utilisée pour la resynchronisation de la nouvelle offre
- Le paramètre `targetOffer` correspond à l'**id** de l'offre à resynchroniser
- Le paramètre `strategyId` correspond à la stratégie des offres source et cible
- le paramètre `tenantId` correspond au tenant sur lequel appliquer la synchronisation
- Le paramètre `container` correspond à un élément datatype de la liste suivante :

```
"units"
"objects"
"objectgroups"
"logbooks"
"reports"
"manifests"
"profiles"
"storagelog"
"storageaccesslog"
"storagetraceability"
"rules"
"dip"
"agencies"
"backup"
"backupoperations"
"unitgraph"
"objectgroupgraph"
"distributionreports"
"accessionregistersdetail"
"accessionregisterssymbolic"
"tmp"
"archivaltransferreply"
```

- Suivre les journaux de la resynchronisation dans les logs du composant storage offer avec la commande suivante :

```
tail -F /vitam/log/storage/storage_offer_sync.\*.log
```

- Vérifier l'état d'exécution de la synchronisation via la commande (peut être scriptée) :

```
curl -v -X HEAD -i -u adminUser:adminPassword http://<storageengine>
↪:29102/storage/v1/offerSync
```

  L'entête `Running` indique l'état d'exécution de processus de synchronisation.

- Vérifier le détail d'exécution de la synchronisation via la commande :

```
curl -v -X GET -u adminUser:adminPassword http://<storageengine>:29102/
↪storage/v1/offerSync
```

- Exemple de script shell permettant de faire une resynchronisation complète de l'ensemble des containers d'une offre à une autre. Ce script est à exécuter à partir d'une vm du groupe *hosts_storage_engine*. Il est recommandé de l'exécuter à l'intérieur d'un *screen* car l'exécution peut-être longue en fonction de la volumétrie à transférer.

```
#!/bin/bash
# Script permettant de lancer la synchronisation de l'ensemble des␣
↪containers d'une offre de stockage à une autre.
# /!\ Ce script est à exécuter à partir d'une vm du groupe hosts_storage_
↪engine

#########################################################################
↪######
## TODO: Paramètres à personnaliser

# cat environments/group_vars/all/vitam_security.yml | grep admin_basic_
↪auth_user
admin_basic_auth_user='adminUser'
# ansible-vault view environments/group_vars/all/vault-vitam.yml --vault-
↪password-file vault_pass.txt | grep admin_basic_auth_password
admin_basic_auth_password='adminPassword'

# cat environments/group_vars/all/vitam_vars.yml | grep consul_domain:
## consul_domain: consul
# cat environments/group_vars/all/offers_opts.yml
## vitam_strategy.{name,vitam_site_name}
# cat environments/hosts.env | grep ^vitam_site_name
## vitam_site_name=dc1

## {{ vitam_strategy.name }}.service.{{ vitam_strategy.vitam_site_name }}.
↪{{ consul_domain }}
# ou l'id de l'offre à synchroniser tel que définit dans environments/
↪group_vars/all/offers_opts.yml.
sourceOffer="offer-fs-1.service.dc1.consul"
targetOffer="offer-fs-2.service.dc1.consul"

# cat environments/group_vars/all/tenants_vars.yml | grep vitam_tenant_ids
declare -a tenants="0 1 2"

#########################################################################
↪######
## Récupération de l'ip:port du processus storage-engine local
service_storage=`netstat -ln | awk -F" " '{print $4}' | grep 29102`

if [ -z "$service_storage" ]; then
    echo "ERROR: Could not find service vitam-storage running on port␣
↪29102."
    exit 1
fi

## Liste des containers
declare -a containers="units objects objectgroups logbooks reports␣
↪manifests profiles storagelog storageaccesslog storagetraceability␣
↪rules dip agencies backup backupoperations unitgraph objectgroupgraph␣
↪distributionreports accessionregistersdetail accessionregisterssymbolic␣
↪tmp archivaltransferreply"
```

(suite sur la page suivante)

(suite de la page précédente)

```
## Confirmation du lancement de la synchro
echo "Synchronisation de $sourceOffer => $targetOffer"
read -p "Êtes-vous sûr de vouloir lancer la synchronisation ? YES/[NO]" GO
if [[ "$GO" != "YES" ]]
then
echo "Arrêt de la procédure de synchronisation."
exit 0
fi

echo "# Lancement de la procédure de synchronisation de la nouvelle offre"
for tenant in $tenants; do
    echo
↪"********************************************************************************
↪"
    echo "# Synchronisation du tenant $tenant"
    for container in $containers; do
        echo "## Synchronisation ${tenant}_${container}"
        curl -X POST -u $admin_basic_auth_user:$admin_basic_auth_password␣
↪--header 'accept: application/json' --header 'content-type: application/
↪json' http://$service_storage/storage/v1/offerSync \
        --data "{ \"sourceOffer\": \"${sourceOffer}\", \"targetOffer\": \"
↪${targetOffer}\", \"strategyId\": \"default\", \"container\": \"$
↪{container}\", \"tenantId\": ${tenant} }'"

        while curl --silent -X HEAD -i -u $admin_basic_auth_user:$admin_
↪basic_auth_password http://$service_storage/storage/v1/offerSync | grep
↪'Running: true'; do
            curl -X GET -u $admin_basic_auth_user:$admin_basic_auth_
↪password http://$service_storage/storage/v1/offerSync
            echo ""
            sleep 5
        done
        echo "## Fin de la synchronisation du tenant ${tenant}_$
↪{container}"
        curl -X GET -u $admin_basic_auth_user:$admin_basic_auth_password␣
↪http://$service_storage/storage/v1/offerSync
        echo ""
    done
done
```

5.20.3 Procédure de resynchronisation partielle d'une offre

- En cas de resynchronisation partielle d'une offre, il est possible d'exécuter le processus de resynchronisation à partir d'un `offset` :

```
curl -v -X POST -u adminUser:adminPassword --header 'content-type:␣
↪application/json' --header 'accept: application/json' http://
↪<storageengine>:29102/storage/v1/offerSync --data '
{
    "sourceOffer": "<offer-x>.<consul_domain>",
    "targetOffer": "<offer-z>.<consul_domain>",
    "strategyId": <strategyId>,
    "offset": <offset>,
    "container": <datatype>,
```

(suite sur la page suivante)

(suite de la page précédente)

```
    "tenantId": <tenantId>
}'
```

Où <datatype> doit prendre les valeurs suivantes :

```
"units"
"objects"
"objectgroups"
"logbooks"
"reports"
"manifests"
"profiles"
"storagelog"
"storageaccesslog"
"storagetraceability"
"rules"
"dip"
"agencies"
"backup"
"backupoperations"
"unitgraph"
"objectgroupgraph"
"distributionreports"
"accessionregistersdetail"
"accessionregisterssymbolic"
"tmp"
"archivaltransferreply"
```

- Le paramètre `offset` correspond à la valeur du dernier `offset` observé dans les logs du composant storage offer (cas d'une reprise suite à interruption ou échec de la procédure de resynchronisation). Le paramètre `offset` peut également être déterminé via les enregistrements de la collection `OfferLog` (database `offer`) depuis la base MongoDB associée à l'offre à resynchroniser (cas d'une panne ou d'une mise en maintenance programmée à une date précise).

5.20.4 Procédure de resynchronisation ciblée d'une offre

La release R13 permet l'exécution d'une resynchronisation ciblée d'une offre de stockage, à partir d'une liste d'items à resynchroniser.

```
curl -v -X POST -u adminUser:adminPassword --header 'content-type: application/json' -
↪-header 'accept: application/json' http://<storageengine>:29102/storage/v1/
↪offerPartialSync --data '
{
    "strategyId" : <strategyId>,
    "sourceOffer" : "<offer-x>.service.consul",
    "targetOffer" : "<offer-z>.service.consul",
    "itemsToSynchronize" : [ {
        "container" : "objects",
        "tenantId" : 0,
        "filenames" : [ "ObjectId0", "ObjectId1", "ObjectId2", "ObjectId3", "ObjectId4
↪" ]
    },{
        "container" : "units",
        "tenantId" : 0,
        "filenames" : [ "UnitId0", "UnitId1"]
```

(suite sur la page suivante)

(suite de la page précédente)

```
    } ]
}'
```

- Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `deployment/environments/group_vars/all/vitam_security.yml`
- Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `deployment/environments/group_vars/all/vault-vitam.yml`
- Le paramètre `sourceOffer` correspond à l'**id** de l'offre source utilisée pour la resynchronisation de la nouvelle offre
- Le paramètre `targetOffer` correspond à l'**id** de l'offre à resynchroniser
- Le paramètre `strategyId` correspond à la stratégie des offres source et cible
- le paramètre `tenantId` correspond au tenant sur lequel appliquer la synchronisation
- Le paramètre `container` correspond à un élément datatype de la liste suivante :

```
"units"
"objects"
"objectgroups"
"logbooks"
"reports"
"manifests"
"profiles"
"storagelog"
"storageaccesslog"
"storagetraceability"
"rules"
"dip"
"agencies"
"backup"
"backupoperations"
"unitgraph"
"objectgroupgraph"
"distributionreports"
"accessionregistersdetail"
"accessionregisterssymbolic"
"tmp"
"archivaltransferreply"
```

Depuis l'offre `sourceOffer`, la procédure récupère les `filenames` pour le `tenantId` et le `container` concerné. * Si les fichiers sont trouvés, ils sont alors recopiés sur l'offre `targetOffer` * Si les fichiers ne sont pas trouvés, ils seront supprimés de l'offre `targetOffer`

5.21 Audit comparatif entre 2 offres de stockage miroirs

Une offre de stockage peut être désynchronisée par rapport à une autre à la suite d'une indisponibilité plus ou moins longue voire totale de l'offre (*crash* majeur du système, panne matérielle etc.) ou bien encore à la suite d'une mise en maintenance programmée.

Le mécanisme d'audit comparatif entre 2 offres est un audit technique à disposition de l'exploitant. Il permet d'identifier l'ensemble des fichiers désynchronisés entre les 2 offres (existence et size).

5.21.1 Procédure de lancement et de suivi de l'audit comparatif d'offres

Le déclenchement se fait de la manière suivante :

```
ansible-playbook ansible-vitam-exploitation/diff_offers.yml -i environments/hosts.
↪<environnement> --ask-vault-pass -e "offer1=offer-fs1.service.dc1.consul␣
↪offer2=offer-fs-2.service.dc2.consul container=units tenantId=0"
```

- Le paramètre `offer1` spécifie l'identifiant complet de la première offre à comparer (<nom_offre>.service.<vitam_site_name>.consul)
- Le paramètre `offer2` spécifie l'identifiant complet de la seconde offre à comparer (<nom_offre>.service.<vitam_site_name>.consul)
- le paramètre `tenantId` correspond au tenant sur lequel appliquer la synchronisation
- Le paramètre `container` correspond à un élément datatype de la liste suivante :

```
"units"
"objects"
"objectgroups"
"logbooks"
"reports"
"manifests"
"profiles"
"storagelog"
"storageaccesslog"
"storagetraceability"
"rules"
"dip"
"agencies"
"backup"
"backupoperations"
"unitgraph"
"objectgroupgraph"
"distributionreports"
"accessionregistersdetail"
"accessionregisterssymbolic"
"tmp"
"archivaltransferreply"
```

  Si l'audit comparatif des offres remonte des anomalies, un rapport détaillé est mis à disposition.
- Les journaux de l'audit comparatif se trouvent dans les logs du composant **storage**. Ils peuvent être suivis via la commande suivante :

```
tail -F /vitam/log/storage/storage_offer_diff.\*.log
```

5.22 Procédure d'exploitation suite à la création ou la modification d'une ontologie

Au préalable à la création ou à la modification d'une ontologie, les index Elasticsearch correspondant aux ontologies doivent être créés ou mis à jour.

5.22.1 Création d'une ontologie

Suite à la création d'une nouvelle ontologie, les index Elasticsearch doivent être mis à jour selon la procédure suivante :

- Dans le cas d'une création, il suffit de créer un nouveau mapping dans les index concernés.

Ex : Ajout d'une propriété Licence dans tous les index unit (unit* signifiant tous les index unit unit_0, unit_1 etc . . .)

Commandes à lancer sur une des partitions hébergeant le cluster elasticsearch « data » :

```
curl -XPUT "http://localhost:9200/unit*/_mapping/typeunique?update_all_types" -d'
{
      "properties": {
    "Licence": {
      "type": "text"
    }
      }
}'
```

Pour verifier sur un ou tous les index `unit` :

```
curl -XGET "http://localhost:9200/unit_0/_mapping/?pretty=true"
```

```
curl -XGET "http://localhost:9200/unit*/_mapping/?pretty=true"
```

5.22.2 Changement de type d'une ontologie existante

Dans ce cas, le changement de type dans elasticsearch n'est pas possible. Il faut donc créer un nouvel index ElasticSearch avec un nouveau mapping, puis reindexer l'ancien index dans ce dernier.

On récupère d'abord l'ancien index

```
curl -XGET 'localhost:9200/unit_1/_mapping?pretty=true'
```

On créé un fichier json et on y copie les données obtenues (ne conserver que la balise « mappings » : { . . . } et son contenu). On modifie le mapping en changeant le type des propriétés choisies. On créé un nouvel index on lui passant en paramètre le fichier du nouveau mapping .

```
curl -XPUT "http://localhost:9200/new_unit_1" -H 'Content-Type: application/json' -d␣
↪@newmapping.json
```

Verifier l'index :

```
curl -XGET 'localhost:9200/new_unit_1/_mapping/'
```

On reindexe unit_1 vers le nouvel index new_unit_1

```
curl -XPOST 'localhost:9200/_reindex' -H 'Content-Type:application/json' -d '{
    "source" : {
        "index" : "unit_1"
    },
    "dest" : {
        "index" : "new_unit_1",
        "version_type": "external"
    }
}'
```

On efface l'alias de l'ancien index unit_1

```
curl -XDELETE 'localhost:9200/unit_1/_alias/unit_1'
```

et on l'affecte au nouvel index new_unit_1

```
curl -XPUT 'localhost:9200/new_unit_1/_alias/unit_1'
```

Avertissement : les index elasticsearch de *VITAM* sont créés par tenant. Il faudra refaire l'opération ci-dessus pour chaque tenant.

Note : En cas du changement des mappings elasticsearch, il faudra veiller à ce qu'ils soient en cohérence avec l'ontologie.

5.23 L'ontologie externe suite à la montée de version de VITAM

Lors de la montée de version, les ontologies externes en cours d'exploitation par *VITAM* ne sont pas touchées, et seront mergées avec les ontologies internes de *VITAM*.

Le fichier du référentiel de l'ontologie se trouve désormais dans `deployment/environments/ontology/VitamOntology.json`

La procédure de merge manuelle du référentiel de l'ontologie avant chaque montée de version n'est plus nécessaire. Depuis la version 3.4.0 de *VITAM*, le vocabulaire externe de l'ontologie est géré automatiquement avec le vocabulaire interne.

Lors du lancement du procédure de mise à jour de *VITAM*, une phase préliminaire de vérification et validation sera faite pour détecter des éventuelles conflits entre les vocabulaires internes et externes.

Afin d'assurer que la montée de version de *VITAM* passera sans affecter le système , cette vérification s'exécute dans les phases préliminaires de l'ansiblerie, avant la phase de l'installation des composants *VITAM*, (en cas d'échec à cette étape, la solution logicielle déjà installé ne sera pas affectée).

Le script ansible qui fait le check est situé dans : `deployment/ansible-vitam/roles/check_ontologies/tasks/main.yml`, le rôle vérifie que le composant d'administration fonctionnelle `vitam-functional-administration` est bien installé et démarré, ensuite la tâche ansible `Check Import Ontologies` réalise un import à blanc en mode `Dry Run` du référentiel de l'ontologie et remonte des éventuelles erreurs d'import.

Prudence : En cas d'echec de vérification, autrement dit, en cas de présence de conflits entre les deux vocabulaires, l'exploitant adaptera son vocabulaire et veillera à éviter des moindres conflits.

L'exploitant pour vérifier ses corrections en cas d'erreurs, pourra toutefois lancer la commande depuis le dossier *deployment*, depuis une instance hébergeant le composant `vitam-functional-administration` :

```
curl -XPOST -H "Content-type: application/json" -H "X-Tenant-Id: 1" --data-binary␣
↪@environments/ontology/VitamOntology.json 'http://{{ hostvars[groups['hosts_
↪functional_administration'][0]]['ip_admin'] }}:{{ vitam.functional_administration.
↪port_admin }}/v1/admin/ontologies/check'
```

Prudence : Dans le cadre d'une montée de version, se référer également au *DMV*.

5.24 Procédure d'exploitation pour la mise en pause forcée d'une opération

Pour permettre le traitement non-concurrent de certaines opérations (ingest et reclassement en particulier), il est possible de pouvoir forcer la mise en pause à la réception d'opérations (toutes ou seulement d'un type donné, sur tous les tenants ou un tenant donné en particulier).

Concrètement, elle permet de forcer le mode « pas à pas » pour toutes ou un type donné seulement d'opérations, sur l'ensemble des tenants ou sur un tenant donné seulement.

5.24.1 Mise en pause forcée

La mise en pause forcée est déclenchée par l'appel au point d'API porté par le composant `access-external` à l'URL suivante : `http://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/forcepause`

Exemple d'appel à l'aide de curl :

```
curl -X POST -k
--key vitam-vitam_1.key
--cert vitam-vitam_1.pem
``https://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/
↪forcepause``
-H 'X-Tenant-Id: 0'
-H 'X-Access-Contract-Id: ContratTNR'
-H 'Content-Type: application/json;charset=UTF-8'
-H 'Accept: application/json'
--data-binary '{"type" : "INGEST", "tenant" : "0"}'
--compressed
```

Exemple de Json pour mettre une pause sur le processus d'ingest pour le tenant 0 :

```
'{"type" : "INGEST", "tenant" : "0"}'
```

Exemple de Json pour mettre une pause sur tous les processus pour le tenant 0 :

```
'{"tenant" : "0"}'
```

Exemple de Json pour mettre une pause sur tous les processus pour tous les tenants :

```
'{"pauseAll":true}'
```

5.24.2 Sortie de la mise en pause forcée

La sortie de mise en pause forcée est déclenchée par l'appel au point d'API porté par le composant `access-external` à l'URL suivante : `http://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/removeforcepause`

Exemple d'appel à l'aide de curl :

```
curl -X POST -k
--key vitam-vitam_1.key
--cert vitam-vitam_1.pem
``https://{{ ip_service }}:{{ vitam.accessexternal.port_service }}/admin-external/v1/
↪removeforcepause``
```

(suite sur la page suivante)

(suite de la page précédente)

```
-H 'X-Tenant-Id: 0'
-H 'X-Access-Contract-Id: ContratTNR'
-H 'Content-Type: application/json;charset=UTF-8'
-H 'Accept: application/json'
--data-binary '{"type" : "INGEST", "tenant" : "0"}'
--compressed
```

Exemple de Json pour sortir de la mise en pause sur le processus d'ingest pour le tenant 0 :

```
'{"type" : "INGEST", "tenant" : "0"}'
```

Exemple de Json pour sortir de la mise en pause sur tous les processus pour le tenant 0 :

```
'{"tenant" : "0"}'
```

Exemple de Json pour sortir de la mise en pause sur tous les processus pour tous les tenants :

```
'{"pauseAll":false}'
```

Avertissement : Les états de mise en pause ne sont pas sauvegardés. En cas de redémarrage des applications (en particulier le composant access-external), ces états sont perdus.

5.25 Réindexation

Cette procédure consiste à réindexer le contenu des bases de données Elasticsearch-data (cluster d'indexation dédié aux données métier) en cas de perte ou d'inconsistance de données, à partir des informations présentes dans les bases de données MongoDB-data (replicaset MongoDB stockant les données métier de Vitam). Elle part du principe que le contenu des collections MongoDB-data n'a pas été altéré et que les différents indexes Elasticsearch-data sont toujours existants.

5.25.1 Déclenchement

La réindexation se déclenche de la manière suivante :

```
ansible-playbook ansible-vitam-exploitation/reindex_es_data.yml -i environments/hosts.
↪<environnement> --ask-vault-pass
```

Ce playbook s'assure que le composant `vitam-functional-administration` est démarré, puis procède à la réindexation et au *re-aliasing* (bascule sur le nouvel index) des collections suivantes :

- unit
- objectgroup
- logbookoperation
- securityprofile
- context
- ontology
- ingestcontract
- agencies
- accessionregisterdetail

- archiveunitprofile
- accessionregistersummary
- accesscontract
- fileformat
- filerules
- profile
- griffin
- preservationscenario
- managementcontract

Note : La réindexation peut s'opérer au besoin sur uniquement l'une des collections ci-dessus en spécifiant l'option –tags <collection> à l'exécution de la commande ansible.

Prudence : La réindexation de la collection griffin n'est pas utilisable dans cette version (bug 5762).

Prudence : La purge des anciens index n'est pas réalisée par cette procédure scriptée et est laissée à la charge de l'exploitant.

Note : Les fichiers mapping ES des collections metadata Unit et ObjectGroup sont externalisés (configurables de l'extérieur).

5.26 Nettoyage des ingests incomplets

Cette procédure permet de nettoyer les données suite à un ingest incomplet / corrompu. Elle permet de purger toutes les unités archivistiques, groupes d'objets et objets binaires liés à l'ingest.

5.26.1 Conditions d'éligibilité des ingests à nettoyer

L'ingest à nettoyer doit satisfaire les conditions d'éligibilité suivantes :

- L'ingest n'est plus en cours d'exécution (RUNNING ou PAUSE)
- L'ingest s'est terminé avec une erreur (KO ou FATAL)
- Aucune unité d'un autre ingest n'a été rattachée en dessous d'une des unités de l'ingest à nettoyer
- L'ingest à nettoyer n'a pas rajouté d'objets binaires à un groupe d'objets existant
- Aucun autre ingest n'a rajouté d'objets binaires à l'un des groupes d'objets de l'ingest à nettoyer
- L'ingest à nettoyer n'a pas rattaché une unité à un groupe d'objets existant
- Aucun autre ingest n'a rattaché une autre unité à un groupe d'objets de l'ingest à nettoyer

5.26.2 Déclenchement

Le déclenchement se fait de la manière suivante :

```
ansible-playbook ansible-vitam-exploitation/ingest_cleanup.yml -i environments/hosts.
↪<environnement> --ask-vault-pass -e "ingestOperationId=${guid_ingest_a_nettoyer}" -
↪e "tenantId=${tenant}"
```

Ce playbook s'assure que le composant `vitam-functional-administration` est démarré, puis procède au lancement d'un workflow de nettoyage.

Note : Cette procédure ne doit être exécutée que pour nettoyer les ingests incomplets / corrompus qui sont éligibles aux conditions d'éligibilité.

5.27 Suppression des DIP et des fichiers de transfert

Les DIP générés dans le cadre d'une demande de communication et les SIP générés dans le cadre d'une demande de transfert sont stockés dans des dossiers spécifiques.

Ils sont purgés automatiquement à l'expiration d'un délai paramétrable par l'administrateur technique.

Par défaut, la solution logicielle Vitam retient les DIP durant 7 jours si l'espace libre du workspace représente plus de 25% de la taille totale du workspace et durant 1 jour si ce dernier représente moins de 25% de la taille totale du workspace, quel que soit leur type, et les efface au moyen d'un batch qui est lancé chaque heure à 0 minute 0 seconde.

Cette configuration par défaut peut être modifiée lors du paramétrage initial de la plate-forme par les administrateurs – fonctionnel pour la définition du besoin et technique pour la saisie réelle des informations – de chaque implémentation de la solution logicielle Vitam et définit, pour tous les tenants et pour chaque type de DIP – DIP générés dans le cadre d'une demande de communication et SIP générés dans le cadre d'une demande de transfert – la durée de rétention dans l'espace de stockage et la fréquence du batch permettant de les purger du système.

Le fichier de configuration (`deployment/environments/group_vars/all/vitam_vars.yml`) se présente comme suit (paramétrage par défaut) :

```
### global ###
[...]
vitam_timers:
[...]
    metadata:
        [...]
        - name: vitam-metadata-purge-dip
          frequency: "*-*-* *:00:00"
        - name: vitam-metadata-purge-transfers-SIP
          frequency: "*-*-* 02:25:00"
### Composants Vitam ###
vitam:
    [...]
    metadata :
     [...]
     # DIP cleanup delay (in minutes)
        dipTimeToLiveInMinutes: 10080 # 7 days
        criticalDipTimeToLiveInMinutes: 1440 # 1 day
        transfersSIPTimeToLiveInMinutes: 10080 # 7 days
        workspaceFreespaceThreshold: 25 # when below use critical time to live when␣
↪above use normal time to live
```

Note : Veuillez vous référer aux documents d'architecture (chapitre 5.13.2 « Stockage ») et d'installation (chapitre 4.2.5.12 « Fichiers complémentaires ») pour obtenir plus d'informations sur le stockage dans la solution *Vitam*

5.28 Procédure d'exploitation pour la révocation des certificats SIA et Personae

Cette section fait référence au chapitre *Intégration d'une application externe dans Vitam* (page 243).

La version 1.10.0 (« R8 ») introduit une nouvelle fonctionnalité permettant la révocation des certificats *SIA* et *Personae* afin d'empêcher des accès non autorisés aux *API* de la solution logicielle *VITAM* (vérification dans la couche https des *CRL*).

Le fonctionnement de la validation des certificats de la solution logicielle *VITAM SIA* et *Personae* par *CRL* est le suivant :

- L'administrateur transmet à la solution logicielle *VITAM* le *CRL* d'un *CA* qui a émis le certificat présent dans la solution logicielle *VITAM*, via le point d'API suivant

```
http://{{ hosts_security_internal }}:{{ vitam.security_internal.port_admin }}/v1/
↪api/crl
```

Prudence : La CRL fournie doit être obligatoirement au format DER (cf. http://www.ietf.org/rfc/rfc3280.txt »>RFC 3280 : *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*)

Exemple :

```
curl -v -X POST -u {{ admin_basic_auth_user }}:{{ admin_basic_auth_password }}  http:/
↪/{{ hosts_security_internal }}:{{vitam.security_internal.port_admin}}/v1/api/crl -H
↪'Content-Type: application/octet-stream' --data-binary @/path/to/crl/my.crl
```

- Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `deployment/environments/group_vars/all/vitam_security.yml`
- Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `deployment/environments/group_vars/all/vault-vitam.yml`
- Le système va contrôler tous les certificats (collections `identity.Certificate` et `identity.PersonalCertificate`) émis par le *IssuerDN* correspondant à la *CRL*, en vérifiant si ces derniers sont révoqués ou non. Si c'est le cas, alors la solution logicielle *VITAM* positionne le statut du certificat révoqué à **REVOKED**. Cela a pour conséquence le rejet de tout accès aux *API VITAM* avec utilisation du certificat révoqué (les filtres de sécurité émettront des exceptions dans les journaux de *log*).
- Une alerte de sécurité est émise dans les journaux en cas de révocation.

5.29 Activation/désactivation d'une offre

Dans le cadre de la maintenance ou de la perte ponctuelle d'une offre de stockage, il peut être nécessaire de la rendre `INACTIVE` afin de permettre de nouveaux versements.

Prudence : Attention, durant toute la durée d'indisponibilité d'une offre, aucune donnée ne sera écrite dessus. *VITAM* ne peux pas assurer la pérénité des données si le nombre de copies attendues selon l'homologation de sécurité n'est pas atteint (>= 2). De plus, lors de la remise à `status: ACTIVE`, il sera nécessaire de faire une resynchronisation de cette offre pour récupérer les données versées durant la durée d'indisponibilité (se référer au chapitre *Resynchronisation d'une offre* (page 40)).

Pour chacune des offres de la `vitam_strategy` dans le fichier `environments/group_vars/all/offers_opts.yml`, rajoutez le paramètre `status: INACTIVE` pour désactiver une offre (par défaut, la valeur est `status: ACTIVE`).

Exemple pour rendre l'offre *offer-fs-2* inactive :

```
vitam_strategy:
    - name: offer-fs-1
      referent: true
      rank: 0
    - name: offer-fs-2
      referent: false
      rank: 1
      status: INACTIVE
```

Prudence : En cas de désactivation d'une offre considérée référente par *VITAM*, ne pas oublier de déclarer une autre offre (contenant les données) comme nouvelle référente (modifications à apporter dans `deployment/environments/group_vars/all/offers_opts.yml` par la directive `referent: true`).

- Afin d'appliquer la nouvelle stratégie de stockage, il va être nécessaire de reconfigurer le composant *storage-engine* :

```
ansible-playbook ansible-vitam/vitam.yml -i environments/hosts.
↪<environnement> --ask-vault-pass --limit hosts_storage_engine --tags␣
↪update_vitam_configuration
```

Avertissement : Cette opération provoque une indisponibilité temporaire des principaux services *VITAM* (versement, gestion, recherche et consultation).

5.30 Nettoyage d'un environnement

Avertissement : La procédure suivante ne doit être appliquée QUE sur un environnement de recette et NE DOIT PAS être utilisée sur un environnement de production.

Un playbook ansible de nettoyage d'un environnement est fourni et permet de purger un environnement afin de le réinitialiser à son état presque initial.

Le nettoyage d'un environnement s'effectue selon la procédure suivante :

- Exécuter la commande suivante afin de nettoyer l'environnement (offres de stockage, workspace et bases de données) :

```
ansible-playbook ansible-vitam-exploitation/cleaning.yml -i environments/
↪hosts.<environnement> --ask-vault-pass
```

En détails, le nettoyage d'un environnement va exécuter la liste des actions suivantes :

1. Arrêt des modules externes, afin que Vitam ne puisse plus accepter de demandes provenant de l'extérieur.
2. Purge des collections MongoDB :

- `LogbookLifeCycleObjectGroup`
- `LogbookLifeCycleObjectGroupInProcess`
- `LogbookLifeCycleUnit`
- `LogbookLifeCycleUnitInProcess`
- `LogbookOperation`
- `AccessionRegisterDetail`
- `AccessionRegisterSummary`
- `AccessionRegisterSymbolic`
- `ObjectGroup`
- `Unit`
- `EliminationActionObjectGroup`
- `EliminationActionUnit`
- `PreservationReport`

3. Réindexation des collections à indexer et purgées en phase 2 : le but étant d'obtenir en fin de traitement des indexes vides (ne contenant aucun document).

- `LogbookOperation`, `ObjectGroup` & `Unit` : une réindexation sera effectuée pour chaque tenant configuré (ex : unit_0_20190130_104530, unit_1_20190130_104540, etc...).
- `AccessionRegisterDetail`, `AccessionRegisterSummary`, `AccessionRegisterSymbolic` : 3 nouveaux indexes seront créés au total pour les 3 collections (ex : `accessionregistersymbolic_20190121_133507`, `accessionregistersummary_20190121_133503` & `accessionregisterdetail_20190121_133505`)

4. Nettoyage des offres de stockage. Selon la configuration de l'environnement, le script est en charge de nettoyer chaque offre de stockage configurée :

- Pour chaque tenant configuré dans Vitam, le script va supprimer tous les sous-containers ainsi que leurs contenus exceptés : « backup » (contenant les référentiels intégrés dans Vitam) et « rules ».
- Pour l'offre File-System, chaque sous-containers supprimé et vidé de son contenu, sera recréé à vide (ex : `int_0_accessionregisterdetail`)

5. Nettoyage du *workspace* : le contenu des objets contenus dans le workspace sera purgé également.

Avertissement : Pour le moment, seules les offres de stockage S3 et File-System sont prises en compte dans la purge des offres de stockage.

Prudence : Les nettoyage des offres Swift ne fonctionne que dans le cas d'une installation de la solution logicielle *VITAM* en environnement CentOS.

Avertissement : Cette opération provoque une indisponibilité complète de *VITAM*

5.30.1 Etat des lieux après purge

Après le passage du script, l'environnement est purgé :

- Les référentiels sont toujours présents (Contrats d'accès, contrats d'entrée, rêgles de gestion, etc...).
- L'environnement est disponible et utilisable : les modules externes sont accessibles (IngestExternal et AccessExternal).
- Les offres de stockage sont vidées, à l'exception des backups des référentiels.
- La cohérence entre MongoDB et ElasticSearch est assurée. La plupart des collections sont vidées, et les indexes E/S associés ne contiennent aucun document.
- Le workspace est purgé, aucune opération n'est en cours et ne peut être relancée.

5.30.2 Limitations

Le fait de purger les journaux et non les référentiels provoquera une incohérence de la plate-forme vis-à-vis de la norme **NFZ-42020** (suppression des logs d'imports de référentiels, mais présence de ceux-ci).

CHAPITRE 6

Suivi de l'état du système

6.1 Veille et patchs sécurité

Les éléments d'infrastructure suivants sont particulièrement sensibles pour la sécurité de la solution logicielle *VITAM* et nécessitent d'être intégrés à la veille sécurité du système :

- Runtime Java (OpenJDK 11)

6.2 API de de supervision

Chaque composant *VITAM* peut dialoguer, selon le paramétrage, via 2 réseaux :

- patte d'administration
- patte de service

Si les partitions ne possèdent qu'une seule interface, les deux « pattes » passent par cette unique interface.

6.2.1 Patte d'administration

La solution logicielle *VITAM* expose en interne de la plate-forme les *API REST* suivantes sur ses composants :

- `/admin/v1/status` : statut simple, renvoyant un statut de fonctionnement incluant des informations techniques sur l'état actuel du composant. Un exemple d'utilisation typique est l'intégration à un outil de supervision ou à un élément actif tiers (ex : load-balancer, ...) . L'appel doit être peu coûteux.
- `/admin/v1/version` : informations de version, build, commit git ayant servi à builder les différents jar.
- `/admin/v1/autotest` : autotest du composant, lançant un test de présence des différentes ressources requises par le composant et renvoyant un statut d'état de ces ressources.

6.2.1.1 /admin/v1/status

L'API de status renvoie un fichier JSON contenant les informations suivantes :


```
{
    "serverIdentity": {
        "Name":"vitam-iaas-app-01",
        "Role":"logbook",
    "PlatformId":425367
    },
    "status":true,
    "detail": {
    },
        "componentsVersions": {
        "e2eb99d93a74409b3ebc5224e596953e9b8a178f":18
    }
}
```


Signification des champs :

- **serverIdentity**
    - Name : hostname du serveur hébergeant le composant (type : texte)
    - Role : Nom du composant (type : texte)
    - PlatformId : ID de l'environnement (type : entier)
- status : Statut du composant (OK/KO) (type : booléen)
- detail : vide dans cette version, sera défini ultérieurement
- **componentsVersions**
    - hash de commit git : nombre de jars avec buildés depuis ce hash

6.2.1.2 /admin/v1/version

L'API de version renvoie les informations suivantes :


```
[
    {
        "Scm-tags":"",
        "Scm-commit-id":"e2eb99d93a74409b3ebc5224e596953e9b8a178f",
        "Scm-commit-id-abbrev":"e2eb99d",
        "Maven-version":"0.13.0-SNAPSHOT",
        "Scm-dirty":"false",
        "Scm-commit-time":"2017-01-11T16:38:14+01",
        "Maven-build-timestamp":"2017-01-11T16:06:09Z",
        "Scm-branch":"origin/master_iteration_13",
        "Build-Jdk":"1.8.0_111",
        "Maven-artefactId":"logbook-rest",
        "Maven-groupId":"fr.gouv.vitam"
    },
    {
        "Scm-tags":"",
        "Scm-commit-id":"e2eb99d93a74409b3ebc5224e596953e9b8a178f",
        "Scm-commit-id-abbrev":"e2eb99d",
        "Maven-version":"0.13.0-SNAPSHOT",
        "Scm-dirty":"false",
        "Scm-commit-time":"2017-01-11T16:38:14+01",
        "Maven-build-timestamp":"2017-01-11T16:06:09Z",
```


(suite sur la page suivante)

(suite de la page précédente)


```
        "Scm-branch":"origin/master_iteration_13",
        "Build-Jdk":"1.8.0_111",
        "Maven-artefactId":"logbook-administration",
        "Maven-groupId":"fr.gouv.vitam"
    },
    ...
    ...
    ...
]
```


Signification des champs :

- Scm-tags : en cours de définition
- Scm-commit-id : hash de commit git à partir duquel le composant à été buildé
- Scm-commit-id-abbrev : hash de commit abrégé
- Maven-version : Version indiquée à maven dans le fichier pom.xml
- Scm-dirty : Etat du repo git au moment du build (si présence de fichiers unstaged => dirty)
- Scm-commit-time : Date du commit git
- Maven-build-timestamp : Date du build par maven
- Scm-branch : Nom de la branche git à partir de laquelle le composant a été buildé
- Build-Jdk : Version de la jdk ayant servit à builder le composant
- Maven-artefactId : Nom du composant
- Maven-groupId : namespace du composant

6.2.1.3 /admin/v1/autotest

L'API d'autotest renvoie les informations suivantes :


```
{
    "httpCode":200,
    "code":"000000",
    "context":"logbook",
    "state":"OK",
    "message":"All services are available",
    "description":"All services are available",
    "errors": [
        {
            "httpCode":200,
            "code":"1",
            "context":"LogbookMongoDbAccessImpl",
            "state":"OK",
            "message":"Sub service is available",
            "description":"LogbookMongoDbAccessImpl service is available"
        },
        {
            "httpCode":200,
            "code":"2",
            "context":"logbook",
            "state":"OK",
            "message":"Internal service is available",
            "description":"vitam-iaas-app-01 service is available"
        }
    ]
}
```

Signification des champs :

- **httpCode** : code de retour http
- **code** : en cours de définition ; futur code retour interne VITAM
- **context** : Nom du composant
- **state** : Etat du composant (OK/KO)
- **message** : Message de statut
- **description** : Message de description
- **errors :**
    - httpCode : code de retour http
    - code : code de retour
    - context : nom du composant
    - state : État du composant
    - message : Message sur l'état du composant
    - description : Description sur l'état du composant

6.2.2 Patte de service

- `/<composant>/v1/status` : statut simple, renvoyant un statut de fonctionnement incluant des informations techniques sur l'état actuel du composant. Un exemple d'utilisation typique est l'intégration à un outil de supervision ou à un élément actif tiers (ex : load-balancer, ...) . L'appel doit être peu coûteux. Le statut normal HTTP renvoyé est 204.

Avertissement : Les composants **vitam-elastic-kibana-interceptor**, **security-internal**, **library** et les *IHM* ne possèdent pas ce statut.

Note : Pour le composant **security-internal**, le point d'*API* est `/v1/status`.

6.3 Logs

La solution logicielle *VITAM* propose une solution ouverte, au choix de l'exploitant. Ce dernier peut, à l'installation, comme à la mise à jour de la solution logicielle *VITAM*, choisir d'utiliser sa propre solution de « regroupement » des logs ou la solution embarquée dans la solution logicielle *VITAM*.

Dans le cas de la solution embarquée, celle-ci se décompose en :

- `rsyslog` ou `syslog-ng` (choix à l'installation, se référer au *DIN*) déployé sur les machines « applicatives » *VITAM* et les envois applicatifs syslog vers un serveur de centralisation de logs (via `facility local0`)
- un serveur de centralisation de logs, comprenant :
    - un mono-noeud (au minimum, ou multi-noeuds) Elasticsearch
    - un moteur logstash, parsant les messages VITAM
    - un afficheur de rendu/aggrégation de données Kibana dédié

Voir aussi :

Les principes et implémentations du système de gestion de logs inclus dans VITAM sont décrits plus en détail dans le DAT.

6.3.1 Paramétrage des règles de log

Il est conseillé de modifier le paramétrage du rolling des fichiers de log afin d'éviter une saturation des volumes disque. En fonction de la topologie de déploiement des composants vitam, il faut s'assurer que les quantités maximales de fichiers de log (en nombre et taille) ne dépassent pas la taille du disque, en décomptant la volumétrie évaluée aux données ainsi qu'aux fichiers temporaires.

Avertissement : Dans le cas de la colocalisation des composants Vitam, il faut prendre en compte les quantités maximales par composant, afin de paramétrer des quantités cohérentes pour ne pas dépasser la taille du disque.

Pour cela, il faut éditer le fichier `deployment/environments/group_vars/all/vitam_vars.yml` et définir, pour chaque composant, les paramètres suivants :

- logback_rolling_policy : { true | false } active le rolling des fichiers de logs (applicatif et accès), avec la politique logback TimeBasedRollingPolicy (voir doc `<http://logback.qos.ch/manual/appenders.html#RollingFileAppender>`) dont les paramètres sont définis ci-après. Si ce paramètre est désactivé, les logs ne tournent pas et leur taille augmente à l'infini (cas ou l'implémentation de rolling est géré avec un composant externe à Vitam). La valeur par défaut est true.

- Pour les logs applicatif :
  - logback_max_file_size : { \d*[KB|MB|GB] } spécifie le seuil pour lequel une fois atteint, le fichier tourne. Le nom du fichier de log est suffixé par un index. La valeur par défaut est « 10MB ».
  - **logback_total_size_cap : {struct yml} :**

    <type> : { file | security | offer_tape | offer_tape_backup | offersync } : spécifie le type de fichier de log (correspond à un appender logback). Tous les composants Vitam dispose au moins des types file et security. Les 3 autres types sont disponibles pour le composant offer.

    Pour chacun des types spécifiés dans la structure yml, les paramètres suivants sont éditables :
    - history_days : { \d* } spécifie le nombre total de fichiers conservés. Ce nombre total est équivalent en nombre de jour conservé lorsque le fichier ne tourne pas dans une journée (lorsqu'il n'a pas atteint la limite logback_max_file_size). Cette équivalence est induite par le mécanisme interne au framework logback qui utilise le pattern date spécifié dans le nom du fichier (cf. documentation officielle pour plus de détail). Lorsqu'un fichier tourne et que le nombre de fichier présent est atteint, le premier fichier qui a été tourné est supprimé. La valeur par défaut est 10.
    - totalsize : { \d*[KB|MB|GB] } spécifie la taille totale des fichiers de logs. Lorsque la taille totale des fichiers présents dépasse la taille paramétrée, le premier fichier qui a été tourné est supprimé. La valeur par défaut est 5GB.

  L'ensemble de ces paramètres sont utilisés lors de la génération du fichier `/vitam/conf/<composant>/logback.xml` qui ne doit pas être édité manuellement, au risque de perdre les modifications au prochain déploiement.
- Pour les logs d'accès http :
  - access_retention_days : { \d* } spécifie le nombre total de fichiers conservés. Ce nombre total est équivalent en nombre de jour conservé lorsque le fichier ne tourne pas dans une journée (lorsqu'il n'a pas atteint la limite logback_max_file_size). La valeur par défaut est 30.
  - access_total_size_cap : { \d*[KB|MB|GB] } spécifie la taille totale des fichiers de logs. Lorsque la taille totale des fichiers présents dépasse la taille paramétrée, le premier fichier qui a été tourné est supprimé. La valeur par défaut est 10GB.

  L'ensemble de ces paramètres sont utilisés lors de la génération du fichier `/vitam/conf/<service_id>/logback-access.xml` qui ne doit pas être édité manuellement, au risque de perdre les modifications au prochain déploiement.

Prudence : La configuration de la durée de rétention des logs d'accès et/ou leur externalisation devra être ajustée pour respecter les contraintes légales en vigueur pour le système déployé.

D'autres paramètres sont disponibles :

- Pour les logs de la jvm :
    - jvm_log : { true | false } : active les logs jvm en spécifiant les paramètres -XX :+UnlockDiagnosticVMOptions -XX :+LogVMOutput -XX :LogFile={vitam_folder_log}/jvm.log à la commande java. La valeur par défaut est false.

  Ce paramètre est utilisé lors de la génération du fichier `/vitam/conf/<service_id>/java_opts` qui ne doit pas être édité manuellement, au risque de perdre les modifications au prochain déploiement.

  L'activation de ces logs peut être nécessaire pour analyser un problème en rapport avec la jvm. Si besoin, d'autres paramètres sont disponibles, mais ne sont pas prévus dans les paramètres du fichier de déploiement. Il est possible toutefois de les ajouter temporairement au fichier généré. Par exemple, les paramètres des logs du GC sont :
    - Niveau de détail : activation des détails et des timestamps (paramètres JVM `-XX:+PrintGCDetails -XX:+PrintGCApplicationStoppedTime`)
    - Roulement : le roulement des fichiers dépend de la taille des fichiers, avec un nombre de fichiers maximal ; il est défini comme suit :
        - Activation du roulement : paramètre JVM `-XX:+UseGCLogFileRotation`
        - Nombre total de fichiers conservés : paramètre JVM `-XX:NumberOfGCLogFiles=10`
        - Taille unitaire maximale d'un fichier de logs : paramètre JVM `-XX:GCLogFileSize=10M`
        - Pattern des fichiers : dans le répertoire de logs de l'application (paramètre `-Xloggc:$LOG_FOLDER/gc.log`) pour le fichier courant ; après roulement, les fichiers sont nommés `gc.log.<n>` (avec `<n>` le numéro du fichier, sur base 0).
- Pour les logs applicatif :
    - performance_logger : { true | false } : active les traces qui consignent le temps d'exécution passé dans un composant ou traitement vitam. Ces métriques sont utilisées dans le dashboard Kibana `Metrics workflow vitam`. La valeur par défaut est false.

6.3.2 Rétention des index sous elasticsearch-log

Curator est l'outil défini dans *VITAM* pour nettoyer les index du *cluster* elasticsearch de log. Curator a été paramétré avec les informations contenues, durant l'installation, dans le fichier `cots_vars.yml`.

Pour les différents index dans le *cluster* Elasticsearch de log, deux paramètres sont définis pour Curator :

- close : { \d* } nombre de jours avant clôture de l'index. La valeur par défaut est 7 (5 pour metricbeat/packetbeat).
- delete : { \d* } nombre de jours avant suppression de l'index. La valeur par défaut est 30 (10 pour metricbeat/packetbeat).

Note : concernant les index « logstash-* », il est recommandé de laisser une durée de rétention de 1 an dans un contexte de production.

Il est possible de modifier le comportement de curator. Pour ce faire, il faut :

1. modifier le fichier `environments/group_vars/all/cots_vars.yml`
2. rejouer le playbook de déploiement, en ajoutant en fin de commande `--tags curator_logs`.

6.4 Audit

Divers audits sont mis à disposition des utilisateurs et administrateurs par le biais de l'*IHM* de démonstration sont décrits dans le Manuel Utilisateurs.

6.4.1 Audit de cohérence

Note : Il est recommandé de procéder à un audit de cohérence aléatoire dans le cadre d'opérations techniques ciblées, telles qu'une migration de plate-forme et de données.

Pour lancer un audit de cohérence, il faut lancer le playbook comme suit :

```
ansible-playbook ansible-vitam-exploitation/audit_coherence.yml -i environments/hosts.
↪<environnement> --ask-vault-pass  -e "tenantId=<tenant id> access_contract=<access␣
↪contrat> operationIds=<operationid1,operationid2...>"
```

Les paramètres à spécifier sont :

- *tenantId* : Le tenant à auditer
- *access_contract* : Le contrat d'accès
- *operationIds* : La liste des opérations d'entrée (ingest) à auditer au format *guid1,guid2,... guidN* (séparées par des virgules, sans espaces). L'audit portera sur les unités archivistiques versées au cours des opérations d'entrée spécifiées.

Les identifiants des opérations d'entrée (ingest) à cibler peuvent être récupérés depuis l'IHM via le journal des opérations, en filtrant sur les opérations de type « Entrée » (INGEST). Il peut s'agir d'opérations choisies aléatoirement ou ciblées sur une période donnée.

Prudence : L'audit de cohérence est un workflow lourd qui doit être lancé sur un volume modéré d'unités archivistiques (100K maximum). Les opérations d'entrée (ingest) à auditer doivent être renseignées en conséquent (ex. Si les versement contiennent 10K unités archivistiques en moyenne, ne pas renseigner plus de 10 opérations).

6.4.2 Audit sur les collections d'administration

Note : Cet audit permet de contrôler la présence/absence d'un référentiel (une collection d'administration fonctionnelle ou technique) sur les offres de stockage.

Pour lancer un audit des référentiels, il faut lancer le playbook comme suit :

```
ansible-playbook ansible-vitam-exploitation/audit_referential.yml -i environments/
↪hosts.<environnement> --ask-vault-pass -e "tenant_id=<tenant> collectionName=<nom_
↪collection>"
```

Liste des *collectionName* disponible pour l'audit :

- Access_Contract
- Accession_Register_Summary
- Agencies
- Archive_Unit_Profile

- Formats
- Ingest_Contract
- Profile
- Rules

Indication : L'audit est lancé sur un seul tenant et une seule collection. Le tenant doit être le tenant d'administration dans le cas où la collection n'est accessible que par le tenant d'administration.

6.5 Gestion de la capacité

La gestion de la scalabilité du système dépend de ses usages métier ; le lien entre les usages et les composants *VITAM* sollicités est indiqué dans le *DAT*, avec des dimensionnements de plateforme standard pour différents usages.

Le suivi de la charge sur chaque serveur se fait par les outils standard de l'exploitant.

6.6 Suivi de l'état de sécurité

Une étude est actuellement en cours pour réaliser ce type de suivi.

6.7 Alerting

6.7.1 Système

Le suivi des alertes système est à charge de l'exploitant.

6.7.2 Applicatif

Les logs applicatifs de la solution *VITAM* permettent à l'exploitant de mettre en place un *alerting* adapté à l'usage de son équipe métier et technique. Par défaut, et en guise d'exemple, des dashboards Kibana sont disponibles avec un rassemblement des événements courants de sécurité / erreur (ex. : incohérence règles de gestion, désynchronisation MongoDB / ElasticSearch. . .).

6.8 Suivi des Workflows

La solution logicielle *VITAM* intègre une solution de suivi et de gestion des *Workflows*. Elle permet entre autres de :

- Relancer un Workflow arrêté
- Mettre en pause un Workflow démarré
- Rejouer une étape d'un Workflow
- Annuler un workflow

6.8.1 Suivi

Le suivi peut être réalisé via *IHM*, par des appels *REST* ou par un playbook ansible.

6.8.1.1 IHM

Il existe une page dans l'*IHM* de démonstration, permettant d'influer sur les processus en cours. Tous les processus mis en pause, automatiquement (lors d'un FATAL) ou bien manuellement (Mode pas à pas) apparaissent sur cette *IHM*. Il est également possible, à partir de cette *IHM*, de relancer le processus ou bien de rejouer une étape, après action d'exploitation.

6.8.1.2 Appels REST

Il est possible d'exécuter ces différentes actions sur l'*API* en direct, via des appels `curl` par exemple sur le composant access-external :

- PUT sur le endpoint /operations/GUID avec comme header X-Action :RESUME par exemple.

Pour plus d'information, consulter la documentation des *API* externes.

6.8.1.3 Playbook ansible

Lancer le script suivant

```
ansible-playbook ansible-vitam-exploitation/check_workflow_status.yml -i environments/
↪hosts.<environnement> --ask-vault-pass
```

Avertissement : Le playbook ansible ne peut être exécuté que dans le cas où une installation a déjà été effectuée, et que la *PKI* n'a pas été rejouée (les certificats présents dans `environments/certs` doivent être ceux mis en place dans *VITAM*).

6.8.2 Cas des *worklows* en FATAL

Un *workflow* se met en pause dès qu'il se retrouve en statut **FATAL**. Plusieurs causes peuvent expliquer un tel état.

6.8.2.1 Plugins et Handlers

Plusieurs problèmes peuvent expliquer qu'un *Handler* ou un *plugin* retourne une erreur « FATAL » et donc provoque la mise en pause du *Worfklow*.

Si le composant workspace est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour tous les *Handlers* et *plugins*.

Si le composant logbook est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les *handlers* suivants :

- CommitLifeCycleActionHandler
- CommitLifeCycleObjectGroupActionHandler
- CommitLifeCycleUnitActionHandler
- ListLifecycleTraceabilityActionHandler
- FinalizeLifecycleTraceabilityActionHandler
- RollBackActionHandler

Si le composant functional-administration est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- CheckArchiveProfileRelationActionHandler

- CheckArchiveProfileActionHandler
- GenerateAuditReportActionHandler
- PrepareAuditActionHandler

Si le composant metadata est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- AccessionRegisterActionHandler
- ListArchiveUnitsActionHandler
- PrepareAuditActionHandler
- ArchiveUnitRulesUpdateActionPlugin
- AuditCheckObjectPlugin
- IndexObjectGroupActionPlugin
- IndexUnitActionPlugin
- RunningIngestsUpdateActionPlugin

Si le composant storage est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- CheckStorageAvailabilityActionHandler
- FinalizeLifecycleTraceabilityActionHandler
- GenerateAuditReportActionHandler
- PrepareTraceabilityCheckProcessActionHandler
- PutBinaryOnWorkspace
- CheckIntegrityObjectPlugin
- CheckExistenceObjectPlugin
- StoreMetaDataObjectGroupActionPlugin
- StoreMetaDataUnitActionPlugin
- StoreObjectActionHandler
- StoreObjectGroupActionPlugin

Si le composant processing est défectueux ou ne répond plus, alors un FATAL pourra être obtenu pour les Handlers suivants :

- ListRunningIngestsActionHandler

Si le composant FormatIdentifier est défectueux et ne répond plus, alors un FATAL pourra être obtenu pour le Handler suivant :

- FormatIdentificationActionPlugin

6.8.2.2 Distributor

Plusieurs cas peuvent provoquer un FATAL au niveau du processing :

- si metadata ou workspace est injoignable
- si un *handler* (ou plugin) inexistant est appelé.
- si le distributeur tente d'appeler une famille de worker inexistante

6.8.2.3 Processing - State Machine

Dans le cas ou le Processing ne parvient pas à enregistrer l'état du workflow sur le workspace, un FATAL est provoqué. Il en va de même si le composant logbook est défectueux.

6.8.3 Redémarrer un processus en cas de pause

6.8.3.1 Trouver la cause

De manière générale, il convient d'identifier le composant (ou les composants) posant problème. Il s'agira majoritairement de metadata, de logbook, du storage ou encore du workspace.

A partir du Guid de l'opération mise en pause, il est facilement possible de voir, dans les logs du processing ou des workers quels sont les composants incriminés.

6.8.3.2 Relancer le Workflow

A partir du Guid de l'opération mise en pause et une fois le composant redémarré, il est possible de relancer le workflow.

6.8.3.2.1 Vérifier les inputs

S'assurer à partir du GUID de l'opération que l'on nommera X la présence :

- d'un fichier X.json dans /vitam/data/workspace/process/distributorIndex/
- d'un répertoire X dans `/vitam/data/workspace/` contenant à minima une liste de sous-répertoires (et notamment le *SIP* décompressé dans le sous répertoire `SIP`).

6.8.3.2.2 Rejouer une étape

Depuis l'*IHM*, relancer l'étape précédente en cliquant sur l'icône « Replay ». Via les *API*, il suffit de lancer un appel `curl` sur le composant access external : PUT sur le endpoint /operations/GUID avec comme header X-Action :REPLAY.

Cette action aura pour résultat d'exécuter une deuxième fois l'étape qui a échoué. En sortie de ce replay, le statut du workflow doit passer à OK et l'état à PAUSE.

6.8.3.2.3 Prochaine étape

Depuis l'*IHM*, exécuter l'étape suivante en cliquant sur l'icône « Next ». Via les *API*, il suffit de lancer un appel curl sur le composant « access-external » : PUT sur le endpoint /operations/GUID avec comme header X-Action :NEXT.

Cette action aura pour résultat d'exécuter l'étape suivante. En sortie de ce replay, le statut du workflow doit passer à OK et l'état à PAUSE.

6.8.3.2.4 Finaliser le workflow

Il est possible de poursuivre le workflow jusqu'à son terme.

Depuis l'*IHM*, finaliser le workflow en cliquant sur l'icône « Fast Forward ».

Via les *API*, il suffit de lancer un appel curl sur le composant `access-external` : PUT sur le endpoint /operations/GUID avec comme header X-Action :RESUME.

6.9 Cohérence des journaux

Il existe un outil d'administration utilisable par l'exploitant afin de réaliser un test de cohérence des journaux. Cet outil permet de vérifier que les données enregistrées dans la collection LogbookOperations sont bien en cohérence avec les informations sauvegardées dans les collections LFC.

Actuellement, seuls les *TNR* utilisent le point d'API.

A l'avenir, il sera possible de préciser les modalités dans un fichier json associé, et il sera possible d'utiliser le contrôle de cohérence indépendamment.

6.9.1 Lancement

Pour lancer l'outil de cohérence, il suffit de lancer une requête (`curl`, par exemple) sur le serveur logbook interne (sur la « patte » d'administration) :

- POST sur le endpoint /checklogbook

6.9.2 Résultat

L'outil de cohérence renvoie un code OK, si l'opération s'est bien déroulée. En cas d'erreur interne, un code HTTP 500 sera renvoyé.

Dans le cadre d'un OK, un rapport au format Json sera généré, et sera enregistré sur les offres de stockage.

Le rapport contiendra les informations suivantes :

- checkedEvents : la liste des événements vérifiés.
- checkErrors : la liste des erreurs constatées.

6.10 Liste des *timers* systemd

Note : Dans les sections suivantes, les éléments de type `<curator.log.metrics.close>` correspondent à des variables de l'inventaire ansible utilisé.

Voir aussi :

La fréquence de la plupart *timers* est modifiable (avec un comportement par défaut) ; se reporter au *DIN* et à *Modifier la fréquence de lancement de certains timers systemD* (page 16) pour plus d'informations.

6.10.1 *Timers* de maintenance des index elasticsearch-log

Ces *timers* gèrent la maintenance des index elasticsearch du cluster elasticsearch-log.

Ces *timers* sont activés sur tous les sites d'un déploiement multi-sites.

6.10.1.1 vitam-curator-metrics-indexes

Maintenance des indexes `metrics-vitam-*` (sur elasticsearch-log) (qui contiennent les métriques remontées par les composants VITAM) :

- Ferme les indexes de plus de `<curator.log.metrics.close>` jours ;

- Supprime les indexes de plus de `<curator.log.metrics.delete>` jours.

Units systemd :

- vitam-curator-metrics-indexes.service
- vitam-curator-metrics-indexes.timer

Exécution :

- Localisation : groupe ansible `[hosts_elasticsearch_log]` (sur toutes les instances du groupe)
- Périodicité : Lancé chaque jour à 00 :30.

6.10.1.2 vitam-curator-close-old-indexes

Fermeture des anciens indexes `logstash-*` (sur elasticsearch-log) de plus de `<curator.log.logstash.close>` jours (ces indexes contiennent les logs remontés par les composants et COTS VITAM).

Units systemd :

- vitam-curator-close-old-indexes.service
- vitam-curator-close-old-indexes.timer

Exécution :

- Localisation : groupe ansible `[hosts_elasticsearch_log]` (sur toutes les instances du groupe)
- Périodicité : Lancé chaque jour à 00 :10.

6.10.1.3 vitam-curator-delete-old-indexes

Suppression des indexes `logstash-*` (sur elasticsearch-log) de plus de `<curator.log.logstash.delete>` jours (ces indexes contiennent les logs remontés par les composants et COTS VITAM).

Units systemd :

- vitam-curator-delete-old-indexes.service
- vitam-curator-delete-old-indexes.timer

Exécution :

- Localisation : groupe ansible `[hosts_elasticsearch_log]` (sur toutes les instances du groupe)
- Périodicité : Lancé chaque jour à 00 :20.

6.10.2 *Timers* de gestion des journaux (preuve systémique)

Ces *timers* gèrent la sécurisation des journaux métier *VITAM*.

Ces *timers* sont activés uniquement sur le site primaire d'un déploiement multi-sites.

6.10.2.1 vitam-storage-log-backup

Backup des journaux d'écriture de storage dans les offres de stockage.

Units systemd :

- vitam-storage-log-backup.service
- vitam-storage-log-backup.timer

Exécution :

- Localisation : groupe ansible `[hosts_storage_engine]` (sur toutes les instances du groupe)
- Périodicité : Lancé toutes les 4 heures à 15 minutes 0 secondes, par défaut.

6.10.2.2 vitam-storage-accesslog-backup

Backup des journaux d'accès de storage dans les offres de stockage.

Units systemd :

- vitam-storage-accesslog-backup.service
- vitam-storage-accesslog-backup.timer

Exécution :

- Localisation : groupe ansible `[hosts_storage_engine]` (sur toutes les instances du groupe)
- Périodicité : Lancé toutes les 4 heures à 10 minutes 0 secondes, par défaut.

6.10.2.3 vitam-storage-log-traceability

Sécurisation des journaux d'écriture de storage.

Units systemd :

- vitam-storage-log-traceability.service
- vitam-storage-log-traceability.timer

Exécution :

- Localisation : groupe ansible `[hosts_storage_engine]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé toutes les 4 heures à 40 minutes 0 secondes, par défaut.

6.10.2.4 vitam-traceability-operations

Sécurisation du journal des opérations.

Units systemd :

- vitam-traceability-operations.service
- vitam-traceability-operations.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé chaque heure à 05 minutes 0 secondes, par défaut.

6.10.2.5 vitam-traceability-lfc-unit

Sécurisation du journal du cycle de vie des unités archivistiques.

Units systemd :

- vitam-traceability-lfc-unit.service
- vitam-traceability-lfc-unit.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé chaque heure à 35 minutes 0 secondes, par défaut.

6.10.2.6 vitam-traceability-lfc-objectgroup

Sécurisation du journal du cycle de vie des groupes d'objets.

Units systemd :

- vitam-traceability-lfc-objectgroup.service
- vitam-traceability-lfc-objectgroup.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé chaque heure à 15 minutes 0 secondes, par défaut.

6.10.3 *Timers* d'audit interne VITAM

Ces *timers* gèrent le déclenchement périodique des tâches d'audit interne *VITAM*.

Ces *timers* sont activés uniquement sur le site primaire d'un déploiement multi-sites.

6.10.3.1 vitam-traceability-audit

Contrôle de la validité de la sécurisation des journaux.

Units systemd :

- vitam-traceability-audit.service
- vitam-traceability-audit.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé chaque jour à 00 :55, par défaut.

6.10.3.2 vitam-rule-management-audit

Validation de la cohérence des règles de gestion entre les offres de stockage et les bases de données.

Units systemd :

- vitam-rule-management-audit.service
- vitam-rule-management-audit.timer

Exécution :

- Localisation : groupe ansible `[hosts_functional_administration]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé chaque heure à 40 minutes 0 secondes, par défaut.

6.10.4 *Timer* relatif aux liens symboliques de *accession register*

6.10.4.1 vitam-create-accession-register-symbolic

Déclenche une commande qui va calculer le registre des fonds symbolique et les ajoute dans les bases de données.

Units systemd :

- vitam-create-accession-register-symbolic.service (activé sur site primaire uniquement)

- vitam-create-accession-register-symbolic.timer (activé sur site primaire uniquement)

Exécution :

- Localisation : groupe ansible `[hosts_functional_administration]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé chaque jour à 0 :50, par défaut.

6.10.5 *Timers* de reconstruction VITAM

Ces timers gèrent la reconstruction des bases de données VITAM à partir des informations persistées dans les offres de stockage.

Ces timers sont activés uniquement sur le site secondaire d'un déploiement multi-sites.

6.10.5.1 vitam-functional-administration-reconstruction

Reconstruction des données portées par le composant functional-administration.

Units systemd :

- vitam-functional-administration-reconstruction.service
- vitam-functional-administration-reconstruction.timer
- vitam-functional-administration-accession-register-reconstruction.service (activé sur site secondaire seulement)
- vitam-functional-administration-accession-register-reconstruction.timer (activé sur site secondaire seulement)

Exécution :

- Localisation : groupe ansible `[hosts_functional_administration]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé tous les cinq minutes, par défaut.

6.10.5.2 vitam-logbook-reconstruction

Reconstruction des données portées par le composant logbook.

Units systemd :

- vitam-logbook-reconstruction.service
- vitam-logbook-reconstruction.timer

Exécution :

- Localisation : groupe ansible `[hosts_logbook]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé tous les 5 minutes, par défaut.

6.10.5.3 vitam-metadata-reconstruction

Reconstruction des données portées par le composant metadata.

Units systemd :

- vitam-metadata-reconstruction.timer
- vitam-metadata-reconstruction.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les 5 minutes, par défaut.

6.10.5.4 vitam-metadata-store-graph

Log shipping des données graphes portées par le composant metadata.

Units systemd :

- vitam-metadata-store-graph.timer
- vitam-metadata-store-graph.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les 30 minutes (00 :10, 00 :40, 01 :10. . .), par défaut.

6.10.5.5 vitam-metadata-computed-inherited-rules

Recalcul des *computedInheritedRules* pour les *units* dont les *computedInheritedRules* sont marquées comme obsolètes.

Units systemd :

- vitam-metadata-computed-inherited-rules.timer
- vitam-metadata-computed-inherited-rules.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les nuits, à 2h30, par défaut.

6.10.6 *Timers* techniques VITAM

6.10.6.1 vitam-metadata-purge-dip

Nettoyage des exports DIPs expirés.

Units systemd :

- vitam-metadata-purge-dip.timer
- vitam-metadata-purge-dip.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé chaque heure à 0 minute 0 seconde, par défaut.

6.10.6.2 vitam-metadata-purge-transfers-SIP

Nettoyage des exports transfers expirés.

Units systemd :

- vitam-metadata-purge-transfers-SIP.timer
- vitam-metadata-purge-transfers-SIP.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les nuits, à 2h25, par défaut.

6.10.6.3 vitam-offer-log-compaction

Compaction technique des journaux des offres de stockage.

Units systemd :

- vitam-offer-log-compaction.timer
- vitam-offer-log-compaction.service

Exécution :

- Localisation : groupe ansible `[hosts_storage_offer_default]` (sur la dernière instance du groupe uniquement)
- Périodicité : Lancé chaque heure à 40 minutes 0 secondes, par défaut.

6.10.6.4 vitam-metadata-audit-mongodb-es

Audit sur la cohérance de donées MongoDB et Elasticsearch

Units systemd :

- vitam-metadata-audit-mongodb-es.timer
- vitam-metadata-audit-mongodb-es.service

Exécution :

- Localisation : groupe ansible `[hosts_metadata]` (sur la dernière instance du groupe uniquement)
- Périodicité : lancé toutes les nuits, à 2h00, par défaut.

CHAPITRE 7

Exploitation des COTS de la solution logicielle VITAM

7.1 Généralités

Les composants de la solution logicielle *VITAM* sont déployés par un playbook ansible qui :

1. déploie, selon l'inventaire employé, les *packages* nécessaires
2. applique la configuration de chaque composant selon son contexte défini dans l'inventaire

Les composants *VITAM* sont décrits ci-après.

Avertissement : En cas de modification de la configuration, redémarrer le service associé.

7.2 COTS

7.2.1 Cerebro

7.2.1.1 Présentation

Cerebro est un utilitaire de supervision de l'état d'un cluster ElasticSearch.

7.2.1.2 Configuration / fichiers utiles

7.2.1.2.1 Fichier `/vitam/conf/cerebro/application.conf`

```
http.port = {{ cerebro.port }}
http.address = {{ ip_admin }}
# Secret will be used to sign session cookies, CSRF tokens and for other encryption␣
↪utilities.
```

(suite sur la page suivante)

(suite de la page précédente)

```
# It is highly recommended to change this value before running cerebro in production.
secret = "{{ cerebro.secret_key }}"

# Application base path
basePath = "/{{ cerebro.baseuri }}/"

# Defaults to RUNNING_PID at the root directory of the app.
# To avoid creating a PID file set this value to /dev/null
pidfile.path = "/dev/null"

# Rest request history max size per user
rest.history.size = 50 // defaults to 50 if not specified

# Path of local database file
data.path = "{{ vitam_defaults.folder.root_path }}/data/cerebro/cerebro.db"

# Authentication
auth = {
  # Example of LDAP authentication
  #type: ldap
    #settings: {
      #url = "ldap://host:port"
      #base-dn = "ou=active,ou=Employee"
      #method  = "simple"
      #user-domain = "domain.com"
    #}
{% if cerebro.basicauth is defined %}
  # Simple username/password authentication
  type: basic
    settings: {
      username = "{{ cerebro.basicauth.username }}"
      password = "{{ cerebro.basicauth.password }}"
    }
{% else %}
  # Example of simple username/password authentication
  #type: basic
    #settings: {
      #username = "admin"
      #password = "1234"
    #}
{% endif %}
}

# A list of known hosts
hosts = [
{% if groups['hosts_elasticsearch_log']|length > 0 %}
  {
    host = "http://{{ elasticsearch.log.host }}:{{ elasticsearch.log.port_http }}"
    name = "{{ elasticsearch.log.cluster_name }}"
  },
{% endif %}
{% if groups['hosts_elasticsearch_data']|length > 0 %}
  {
    host = "http://{{ elasticsearch.data.host }}:{{ elasticsearch.data.port_http }}"
    name = "{{ elasticsearch.data.cluster_name }}"
  },
{% endif %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
  #{
  #  host = "http://localhost:9200"
  #  name = "Some Cluster"
  #},
  # Example of host with authentication
  #{
  #  host = "http://some-authenticated-host:9200"
  #  name = "Secured Cluster"
  #  auth = {
  #    username = "username"
  #    password = "secret-password"
  #  }
  #}
]
```

7.2.1.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines hébergeant le composant vitam-elasticsearch-cerebro.

En tant qu'utilisateur root : `systemctl start vitam-elasticsearch-cerebro`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le composant vitam-elasticsearch-cerebro.

En tant qu'utilisateur root : `systemctl stop vitam-elasticsearch-cerebro`

- Sauvegarde du service

N/A

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:9000/cerebro`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

N/A

- cas des batches

N/A

7.2.2 Consul

7.2.2.1 Présentation

Consul est un DNS applicatif.

7.2.2.1.1 Cas serveur

Le serveur Consul fédère les agents dans leurs requètes « DNS-like » et permet de rebondir sur un DNS externe, s'il ne permet pas de lui-même, de faire la résolution.

7.2.2.1.2 Cas agent

L'agent Consul annonce aux serveurs les services qu'il permet de porter et *checke* régulièrement l'état de ces services.

7.2.2.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

7.2.2.2.1 Cas des applicatifs monitorés par Consul

Pour chaque composant *VITAM* nécessitant une supervision de la part de Consul, un fichier est installé sur l'agent de la machine sous `vitam/conf/consul` et est basé sur ce squelette :

7.2.2.2.1.1 Fichier `/vitam/conf/consul/service-<composant>.json`

```
{
  "service": {
{% if vitam_struct.vitam_component == vitam.storageofferdefault.vitam_component %}
    "name": "{{ offer_conf }}",
    "address": "{{ ip_wan | default(ip_service) }}",
{% else %}
    "name": "{{ vitam_struct.vitam_component }}",
    "address": "{{ ip_service }}",
{% endif %}
    "port": {{ vitam_struct.port_service }},
    "enable_tag_override": false,
    "tags": ["vitam","{{ vitam_struct.vitam_component }}"],

    "checks": [
      {
        "name": "{{ vitam_struct.vitam_component }} : business service check",
{% if vitam_struct.https_enabled==true %}
        "notes": "HTTPS port opened",
{% else %}
        "notes": "HTTP port opened",
{% endif %}
        "tcp": "{{ ip_service }}:{{ vitam_struct.port_service }}",
        "interval": "{{ vitam_struct.consul_business_check | default(consul_business_
↪check) }}s"
      },
      {
        "name": "{{ vitam_struct.vitam_component }} : admin service check",
        "notes": "Status admin : /admin/v1/status",
        "http": "http://{{ ip_admin }}:{{ vitam_struct.port_admin }}/admin/v1/status",
        "interval": "{{ vitam_struct.consul_admin_check | default(consul_admin_check)␣
↪}}s"
```

(suite sur la page suivante)

(suite de la page précédente)

```
      }
{% if (vitam_struct.https_enabled != true) and (vitam_struct.vitam_component != vitam.elastickibanainterceptor.vitam_component) and (vitam_struct.vitam_component != vitam.security_internal.vitam_component) and (vitam_struct.vitam_component != vitam.ihm_demo.vitam_component) and (vitam_struct.vitam_component != vitam.ihm_recette.vitam_component) and (vitam_struct.vitam_component != vitam.library.vitam_component) %}
      ,{
        "name": "{{ vitam_struct.vitam_component }} : http business service check",
        "notes": "Status business : /{{ vitam_struct.baseuri }}/v1/status",
        "http": "http://{{ ip_service }}:{{ vitam_struct.port_service }}/{{ vitam_struct.baseuri }}/v1/status",
        "interval": "{{ vitam_struct.consul_admin_check | default(consul_admin_check) }}s"
      }
{% endif %}
{% if (vitam_struct.vitam_component == vitam.security_internal.vitam_component) %}
      ,{
        "name": "{{ vitam_struct.vitam_component }} : http business service check",
        "notes": "Status business : /status",
        "http": "http://{{ ip_service }}:{{ vitam_struct.port_service }}/status",
        "interval": "{{ vitam_struct.consul_business_check | default(consul_business_check) }}s"
      }
{% endif %}
{% if (vitam_struct.vitam_component == vitam.worker.vitam_component) or (vitam_struct.vitam_component == vitam.ingestexternal.vitam_component) %}
      ,{
        "name": "Siegfried check",
        "notes": "Is siegfried running ?",
        "tcp": "localhost:{{ siegfried.port }}",
        "interval": "{{ siegfried.consul_check }}s"
      }
{% endif %}
{% if vitam_struct.antivirus is defined %}
      ,{
        "name": "Antivirus check",
        "notes": "Is {{ vitam_struct.antivirus }} running ?",
        "args": ["{{ vitam_folder_conf }}/scan-{{ vitam_struct.antivirus}}.sh","{{ vitam_folder_conf }}/scan-{{ vitam_struct.antivirus}}.sh"],
        "interval": "30s",
        "timeout": "5s"
      }
{% endif %}
    ]

  }
}
```

7.2.2.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-consul`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-consul`

Avertissement : en cas de redémarrage du cluster serveur consul, il faut procéder à un arret/relance par serveur avant de passer au suivant.

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Logs

Les logs applicatifs sont envoyés par rsyslog à la solution de centralisation des logs ; il est néanmoins possible d'en virsionner une représentation par la commande :

```
journalctl --unit vitam-consul
```

- Supervision du service

Consul possède une IHM permettant de superviser l'ensemble des services qu'il couvre.

http(s) ://<adresse> :<port>/ui

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

7.2.3 Kibana interceptor

7.2.3.1 Présentation

Le composant est une interface d'accès entre kibana « métier » et le *cluster* Elasticsearch de données métier.

Prudence : Ce composant **N'EST PAS** à installer en environnement de production.

7.2.3.2 Configuration / fichiers utiles

Les fichiers de configuration sont définis sous `/vitam/conf/elastic-kibana-interceptor`.

7.2.3.2.1 Fichier `elastic-kibana-interceptor.conf`

```
jettyConfig: jetty-config.xml

clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  httpPort: {{ elasticsearch.data.port_http }}
{% endfor %}
whitelist : ["tenant", "all", "mgt", "min", "max", "nbc", "og", "ops", "opi", "sp",
↪"sps", "uds", "up", "us", "storage", "unitType", "v", "qualifiers"]
```

7.2.3.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines constituant le *cluster* Elasticsearch de données.

En tant qu'utilisateur root : `systemctl start vitam-elastic-kibana-interceptor`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le *cluster* Elasticsearch de données.

En tant qu'utilisateur root : `systemctl stop vitam-elastic-kibana-interceptor`

- Sauvegarde du service

N/A

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

- Modification de la liste blanche

Modifier dans le fichier `/vitam/conf/elastic-kibana-interceptor/elastic-kibana-interceptor.conf` le contenu de la directive `whitelist`.

A l'issue, redémarrer le composant.

7.2.4 Elasticsearch chaîne de log

7.2.4.1 Présentation

Le composant `vitam-elasticsearch-log` est une instance de la base d'indexation elasticsearch stockant les informations suivantes :

- les logs des applications VITAM ;
- les logs des applications du sous-système de centralisation des logs ;
- les métriques applicatives.

7.2.4.2 Configuration / fichiers utiles

Se reporter au *DIN*, qui configure le *cluster* ElasticSearch de la chaîne de log.

Les fichiers de configuration sont définis sous `/vitam/conf/elasticsearch-log`.

7.2.4.2.1 Fichier `/vitam/conf/elasticsearch-log/log4j2.properties`

```
status = error

# log action execution errors for easier debugging
logger.action.name = org.elasticsearch.action
logger.action.level = {{ composant.action_log_level }}

appender.console.type = Console
appender.console.name = console
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_name]%marker
↪%m%n

appender.syslog.type = Syslog
appender.syslog.name = syslog
appender.syslog.appName = {{ composant.cluster_name }}
appender.syslog.facility = {{ vitam_defaults.syslog_facility }}
appender.syslog.host = {{ inventory_hostname }}
appender.syslog.protocol = UDP
appender.syslog.port = 514
appender.syslog.layout.type = PatternLayout
# Note: rsyslog only parse RFC3195-formatted syslog messages by default ; AND, to␣
↪make it work with log4j2, we need to start the layout by the app-name.
# IF we were in 5424, we wouldn't have to do this.
appender.syslog.layout.pattern = {{ composant.cluster_name }}: [%d{ISO8601}][%-5p][%-
↪25c{1.}] [%node_name]%marker%m%n
# appender.syslog.format = RFC5424
# appender.syslog.mdcId = esdata

appender.rolling.type = RollingFile
appender.rolling.name = rolling
appender.rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}.log
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_name]%marker
↪%.-10000m%n
appender.rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}-%d{yyyy-MM-dd}-%i.log.gz
appender.rolling.policies.type = Policies
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval = 1
appender.rolling.policies.time.modulate = true
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size = {{ composant.log_appenders.rolling.max_log_file_
↪size }}
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.fileIndex = nomax
appender.rolling.strategy.action.type = Delete
appender.rolling.strategy.action.basepath = ${sys:es.logs.base_path}
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.rolling.strategy.action.condition.type = IfFileName
appender.rolling.strategy.action.condition.glob = ${sys:es.logs.cluster_name}-*
appender.rolling.strategy.action.condition.nested_condition.type =␣
↪IfAccumulatedFileSize
appender.rolling.strategy.action.condition.nested_condition.exceeds = {{ composant.
↪log_appenders.rolling.max_total_log_size }}

rootLogger.level = {{ composant.log_appenders.root.log_level }}
rootLogger.appenderRef.console.ref = console
rootLogger.appenderRef.rolling.ref = rolling
rootLogger.appenderRef.syslog.ref = syslog

appender.deprecation_rolling.type = RollingFile
appender.deprecation_rolling.name = deprecation_rolling
appender.deprecation_rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_deprecation.log
appender.deprecation_rolling.layout.type = PatternLayout
appender.deprecation_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_
↪name]%marker %.-10000m%n
appender.deprecation_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_deprecation-%i.log.gz
appender.deprecation_rolling.policies.type = Policies
appender.deprecation_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.deprecation_rolling.policies.size.size = {{ composant.log_appenders.
↪deprecation_rolling.max_log_file_size }}
appender.deprecation_rolling.strategy.type = DefaultRolloverStrategy
appender.deprecation_rolling.strategy.max = {{ composant.log_appenders.deprecation_
↪rolling.max_files }}


logger.deprecation.name = org.elasticsearch.deprecation
logger.deprecation.level = {{ composant.log_appenders.deprecation_rolling.log_level }}
logger.deprecation.appenderRef.deprecation_rolling.ref = deprecation_rolling
logger.deprecation.additivity = false

appender.index_search_slowlog_rolling.type = RollingFile
appender.index_search_slowlog_rolling.name = index_search_slowlog_rolling
appender.index_search_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_search_slowlog.log
appender.index_search_slowlog_rolling.layout.type = PatternLayout
appender.index_search_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c] [
↪%node_name]%marker %.-10000m%n
appender.index_search_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_search_slowlog-%d{yyyy-MM-dd}.
↪log
appender.index_search_slowlog_rolling.policies.type = Policies
appender.index_search_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_search_slowlog_rolling.policies.time.interval = 1
appender.index_search_slowlog_rolling.policies.time.modulate = true

logger.index_search_slowlog_rolling.name = index.search.slowlog
logger.index_search_slowlog_rolling.level = {{ composant.log_appenders.index_search_
↪slowlog_rolling.log_level }}
logger.index_search_slowlog_rolling.appenderRef.index_search_slowlog_rolling.ref =␣
↪index_search_slowlog_rolling
logger.index_search_slowlog_rolling.additivity = false
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.index_indexing_slowlog_rolling.type = RollingFile
appender.index_indexing_slowlog_rolling.name = index_indexing_slowlog_rolling
appender.index_indexing_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_indexing_slowlog.log
appender.index_indexing_slowlog_rolling.layout.type = PatternLayout
appender.index_indexing_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c] [
↪%node_name]%marker %.-10000m%n
appender.index_indexing_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_indexing_slowlog-%d{yyyy-MM-
↪dd}.log
appender.index_indexing_slowlog_rolling.policies.type = Policies
appender.index_indexing_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_indexing_slowlog_rolling.policies.time.interval = 1
appender.index_indexing_slowlog_rolling.policies.time.modulate = true

logger.index_indexing_slowlog.name = index.indexing.slowlog.index
logger.index_indexing_slowlog.level = {{ composant.log_appenders.index_indexing_
↪slowlog_rolling.log_level }}
logger.index_indexing_slowlog.appenderRef.index_indexing_slowlog_rolling.ref = index_
↪indexing_slowlog_rolling
logger.index_indexing_slowlog.additivity = false

appender.audit_rolling.type = RollingFile
appender.audit_rolling.name = audit_rolling
appender.audit_rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_audit.log
appender.audit_rolling.layout.type = PatternLayout
appender.audit_rolling.layout.pattern = {\
                "@timestamp":"%d{ISO8601}"\
                %varsNotEmpty{, "node.name":"%enc{\%map{node.name}}{JSON}"}\
                %varsNotEmpty{, "node.id":"%enc{\%map{node.id}}{JSON}"}\
                %varsNotEmpty{, "host.name":"%enc{\%map{host.name}}{JSON}"}\
                %varsNotEmpty{, "host.ip":"%enc{\%map{host.ip}}{JSON}"}\
                %varsNotEmpty{, "event.type":"%enc{\%map{event.type}}{JSON}"}\
                %varsNotEmpty{, "event.action":"%enc{\%map{event.action}}{JSON}"}\
                %varsNotEmpty{, "user.name":"%enc{\%map{user.name}}{JSON}"}\
                %varsNotEmpty{, "user.run_by.name":"%enc{\%map{user.run_by.name}}
↪{JSON}"}\
                %varsNotEmpty{, "user.run_as.name":"%enc{\%map{user.run_as.name}}
↪{JSON}"}\
                %varsNotEmpty{, "user.realm":"%enc{\%map{user.realm}}{JSON}"}\
                %varsNotEmpty{, "user.run_by.realm":"%enc{\%map{user.run_by.realm}}
↪{JSON}"}\
                %varsNotEmpty{, "user.run_as.realm":"%enc{\%map{user.run_as.realm}}
↪{JSON}"}\
                %varsNotEmpty{, "user.roles":%map{user.roles}}\
                %varsNotEmpty{, "origin.type":"%enc{\%map{origin.type}}{JSON}"}\
                %varsNotEmpty{, "origin.address":"%enc{\%map{origin.address}}{JSON}"}\
                %varsNotEmpty{, "realm":"%enc{\%map{realm}}{JSON}"}\
                %varsNotEmpty{, "url.path":"%enc{\%map{url.path}}{JSON}"}\
                %varsNotEmpty{, "url.query":"%enc{\%map{url.query}}{JSON}"}\
                %varsNotEmpty{, "request.method":"%enc{\%map{request.method}}{JSON}"}\
                %varsNotEmpty{, "request.body":"%enc{\%map{request.body}}{JSON}"}\
                %varsNotEmpty{, "request.id":"%enc{\%map{request.id}}{JSON}"}\
                %varsNotEmpty{, "action":"%enc{\%map{action}}{JSON}"}\
                %varsNotEmpty{, "request.name":"%enc{\%map{request.name}}{JSON}"}\
                %varsNotEmpty{, "indices":%map{indices}}\
```

(suite sur la page suivante)

(suite de la page précédente)

```
                %varsNotEmpty{, "opaque_id":"%enc{\%map{opaque_id}}{JSON}"}\
                %varsNotEmpty{, "x_forwarded_for":"%enc{\%map{x_forwarded_for}}{JSON}
↪"}\
                %varsNotEmpty{, "transport.profile":"%enc{\%map{transport.profile}}
↪{JSON}"}\
                %varsNotEmpty{, "rule":"%enc{\%map{rule}}{JSON}"}\
                %varsNotEmpty{, "event.category":"%enc{\%map{event.category}}{JSON}"}\
                }%n
# "node.name" node name from the `elasticsearch.yml` settings
# "node.id" node id which should not change between cluster restarts
# "host.name" unresolved hostname of the local node
# "host.ip" the local bound ip (i.e. the ip listening for connections)
# "event.type" a received REST request is translated into one or more transport␣
↪requests. This indicates which processing layer generated the event "rest" or
↪"transport" (internal)
# "event.action" the name of the audited event, eg. "authentication_failed", "access_
↪granted", "run_as_granted", etc.
# "user.name" the subject name as authenticated by a realm
# "user.run_by.name" the original authenticated subject name that is impersonating␣
↪another one.
# "user.run_as.name" if this "event.action" is of a run_as type, this is the subject␣
↪name to be impersonated as.
# "user.realm" the name of the realm that authenticated "user.name"
# "user.run_by.realm" the realm name of the impersonating subject ("user.run_by.name")
# "user.run_as.realm" if this "event.action" is of a run_as type, this is the realm␣
↪name the impersonated user is looked up from
# "user.roles" the roles array of the user; these are the roles that are granting␣
↪privileges
# "origin.type" it is "rest" if the event is originating (is in relation to) a REST␣
↪request; possible other values are "transport" and "ip_filter"
# "origin.address" the remote address and port of the first network hop, i.e. a REST␣
↪proxy or another cluster node
# "realm" name of a realm that has generated an "authentication_failed" or an
↪"authentication_successful"; the subject is not yet authenticated
# "url.path" the URI component between the port and the query string; it is percent␣
↪(URL) encoded
# "url.query" the URI component after the path and before the fragment; it is percent␣
↪(URL) encoded
# "request.method" the method of the HTTP request, i.e. one of GET, POST, PUT, DELETE,
↪ OPTIONS, HEAD, PATCH, TRACE, CONNECT
# "request.body" the content of the request body entity, JSON escaped
# "request.id" a synthentic identifier for the incoming request, this is unique per␣
↪incoming request, and consistent across all audit events generated by that request
# "action" an action is the most granular operation that is authorized and this␣
↪identifies it in a namespaced way (internal)
# "request.name" if the event is in connection to a transport message this is the␣
↪name of the request class, similar to how rest requests are identified by the url␣
↪path (internal)
# "indices" the array of indices that the "action" is acting upon
# "opaque_id" opaque value conveyed by the "X-Opaque-Id" request header
# "x_forwarded_for" the addresses from the "X-Forwarded-For" request header, as a␣
↪verbatim string value (not an array)
# "transport.profile" name of the transport profile in case this is a "connection_
↪granted" or "connection_denied" event
# "rule" name of the applied rulee if the "origin.type" is "ip_filter"
# "event.category" fixed value "elasticsearch-audit"
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.audit_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_audit-%d{yyyy-MM-dd}.log
appender.audit_rolling.policies.type = Policies
appender.audit_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.audit_rolling.policies.time.interval = 1
appender.audit_rolling.policies.time.modulate = true

appender.deprecated_audit_rolling.type = RollingFile
appender.deprecated_audit_rolling.name = deprecated_audit_rolling
appender.deprecated_audit_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_access.log
appender.deprecated_audit_rolling.layout.type = PatternLayout
appender.deprecated_audit_rolling.layout.pattern = [%d{ISO8601}] %m%n
appender.deprecated_audit_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_access-%d{yyyy-MM-dd}.log
appender.deprecated_audit_rolling.policies.type = Policies
appender.deprecated_audit_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.deprecated_audit_rolling.policies.time.interval = 1
appender.deprecated_audit_rolling.policies.time.modulate = true

logger.xpack_security_audit_logfile.name = org.elasticsearch.xpack.security.audit.
↪logfile.LoggingAuditTrail
logger.xpack_security_audit_logfile.level = info
logger.xpack_security_audit_logfile.appenderRef.audit_rolling.ref = audit_rolling
logger.xpack_security_audit_logfile.additivity = false

logger.xpack_security_audit_deprecated_logfile.name = org.elasticsearch.xpack.
↪security.audit.logfile.DeprecatedLoggingAuditTrail
# set this to "off" instead of "info" to disable the deprecated appender
# in the 6.x releases both the new and the previous appenders are enabled
# for the logfile auditing
logger.xpack_security_audit_deprecated_logfile.level = info
logger.xpack_security_audit_deprecated_logfile.appenderRef.deprecated_audit_rolling.
↪ref = deprecated_audit_rolling
logger.xpack_security_audit_deprecated_logfile.additivity = false

logger.xmlsig.name = org.apache.xml.security.signature.XMLSignature
logger.xmlsig.level = error
logger.samlxml_decrypt.name = org.opensaml.xmlsec.encryption.support.Decrypter
logger.samlxml_decrypt.level = fatal
logger.saml2_decrypt.name = org.opensaml.saml.saml2.encryption.Decrypter
logger.saml2_decrypt.level = fatal
```

7.2.4.2.2 Fichier `/vitam/conf/elasticsearch-log/jvm.options`

```
## JVM configuration

################################################################
## IMPORTANT: JVM heap size
################################################################
##
## You should always set the min and max JVM heap
## size to the same value. For example, to set
## the heap to 4 GB, set:
```

(suite sur la page suivante)

(suite de la page précédente)

```
##
## -Xms4g
## -Xmx4g
##
## See https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html
## for more information
##
################################################################

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms{{ elasticsearch_memory }}
-Xmx{{ elasticsearch_memory }}

################################################################
## Expert settings
################################################################
##
## All settings below this section are considered
## expert settings. Don't tamper with them unless
## you understand what you are doing
##
################################################################

## GC configuration
8-13:-XX:+UseConcMarkSweepGC
8-13:-XX:CMSInitiatingOccupancyFraction=75
8-13:-XX:+UseCMSInitiatingOccupancyOnly

## G1GC Configuration
# NOTE: G1 GC is only supported on JDK version 10 or later
# to use G1GC, uncomment the next two lines and update the version on the
# following three lines to your version of the JDK
# 10-13:-XX:-UseConcMarkSweepGC
# 10-13:-XX:-UseCMSInitiatingOccupancyOnly
14-:-XX:+UseG1GC
14-:-XX:G1ReservePercent=25
14-:-XX:InitiatingHeapOccupancyPercent=30

## DNS cache policy
# cache ttl in seconds for positive DNS lookups noting that this overrides the
# JDK security property networkaddress.cache.ttl; set to -1 to cache forever
-Des.networkaddress.cache.ttl=60
# cache ttl in seconds for negative DNS lookups noting that this overrides the
# JDK security property networkaddress.cache.negative ttl; set to -1 to cache
# forever
-Des.networkaddress.cache.negative.ttl=10

## optimizations

# pre-touch memory pages used by the JVM during initialization
-XX:+AlwaysPreTouch

## basic
# force the server VM (remove on 32-bit client JVMs)
-server
```

(suite sur la page suivante)

(suite de la page précédente)

```
# explicitly set the stack size
-Xss1m

# set to headless, just in case
-Djava.awt.headless=true

# ensure UTF-8 encoding by default (e.g. filenames)
-Dfile.encoding=UTF-8

# use our provided JNA always versus the system one
-Djna.nosys=true

# turn off a JDK optimization that throws away stack traces for common
# exceptions because stack traces are important for debugging
-XX:-OmitStackTraceInFastThrow

# flags to configure Netty
-Dio.netty.noUnsafe=true
-Dio.netty.noKeySetOptimization=true
-Dio.netty.recycler.maxCapacityPerThread=0

# log4j 2
-Dlog4j.shutdownHookEnabled=false
-Dlog4j2.disable.jmx=true
# Prevent from exploit in old log4j2 versions <2.17.1
-Dlog4j2.formatMsgNoLookups=true

-Dlog4j.skipJansi=true
-Djava.io.tmpdir=${ES_TMPDIR}

## heap dumps

# generate a heap dump when an allocation from the Java heap fails
# heap dumps are created in the working directory of the JVM
-XX:+HeapDumpOnOutOfMemoryError

# specify an alternative path for heap dumps; ensure the directory exists and
# has sufficient space
-XX:HeapDumpPath={{ elasticsearch_log_dir }}

# specify an alternative path for JVM fatal error logs
-XX:ErrorFile={{ elasticsearch_log_dir }}/hs_err_pid%p.log

## JDK 8 GC logging

8:-XX:+PrintGCDetails
8:-XX:+PrintGCDateStamps
8:-XX:+PrintTenuringDistribution
8:-XX:+PrintGCApplicationStoppedTime
8:-Xloggc:/var/log/elasticsearch/gc.log
8:-XX:+UseGCLogFileRotation
8:-XX:NumberOfGCLogFiles=32
8:-XX:GCLogFileSize=64m

# JDK 9+ GC logging
9-:-Xlog:gc*,gc+age=trace,safepoint:file={{ elasticsearch_log_dir }}/gc.log:utctime,
↪pid,tags:filecount=32,filesize=64m
```

(suite sur la page suivante)

(suite de la page précédente)

```
-Djna.tmpdir={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}
```

7.2.4.2.3 Fichier `/vitam/conf/elasticsearch-log/elasticsearch.yml`

```
# ======================== Elasticsearch Configuration =========================
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please consult the documentation for further information on configuration options:
# https://www.elastic.co/guide/en/elasticsearch/reference/index.html
#
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
cluster.name: {{ composant.cluster_name }}
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: {{ inventory_hostname }}
# TODO: Better handling of this as we have to modify wich nodes are requested by
↪logstash / kibana
node.master: {{ is_master|default('true') }}
node.data: {{ is_data|default('true') }}
node.ingest: false
node.ml: false
xpack.ml.enabled: false
#
# Add custom attributes to the node:
#
# node.rack: r1
#
# ----------------------------------- Paths ------------------------------------
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: {{ elasticsearch_data_dir }}
#
# Path to log files:
#
path.logs: {{ elasticsearch_log_dir }}
#
# ----------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
```

(suite sur la page suivante)

(suite de la page précédente)

```
# = Disable swapping
bootstrap.memory_lock: true
#
# Make sure that the `ES_HEAP_SIZE` environment variable is set to about half the␣
↪memory
# available on the system and that the owner of the process is allowed to use this␣
↪limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
# Note : if installing to localhost, notably a docker container, we need to bind␣
↪larger than localhost
{% if inventory_hostname in single_vm_hostnames %}
network.host: 0.0.0.0
http.cors.enabled: true
http.cors.allow-origin: "*"
{% else %}
# KWA TODO: Check it again (ansible_hostname VS inventory_hostname VS ip_service)
network.host: {{ ip_admin }}
{% endif %}
# Set a custom port for HTTP:
#
http.port: {{ composant.port_http }}
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.seed_hosts: [ {% for host in groups['hosts_elasticsearch_log'] %}"{{␣
↪hostvars[host]['ip_admin'] }}"{% if not loop.last %},{% endif %}{% endfor %} ]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
# TODO OMA : faire mieux, plus propre et prenant bien en compte is_master de chaque␣
↪membre
cluster.initial_master_nodes: [ {% for host in groups['hosts_elasticsearch_log'] %}"{
↪{ hostvars[host]['ip_admin'] }}"{% if not loop.last %},{% endif %}{% endfor %} ]
#
# For more information, consult the discovery and cluster formation module␣
↪documentation.
#
# ---------------------------------- Gateway -----------------------------------
#
# Block initial recovery after a full cluster restart until N nodes are started:
#
#gateway.recover_after_nodes: 3
#
# For more information, consult the gateway module documentation.
#
# --------------------------------- Various -----------------------------------
```

(suite sur la page suivante)

(suite de la page précédente)

```
#
# Require explicit names when deleting indices:
#
action.destructive_requires_name: true

# related to https://www.elastic.co/guide/en/elasticsearch/reference/7.3/modules-
↪fielddata.html
indices.fielddata.cache.size: {{ composant.indices_fielddata_cache_size }}

# related to https://www.elastic.co/guide/en/elasticsearch/reference/7.3/circuit-
↪breaker.html#fielddata-circuit-breaker
indices.breaker.fielddata.limit: {{ composant.indices_breaker_fielddata_limit }}

indices.mapping.dynamic_timeout: {{ composant.dynamic_timeout|default('30s') }}

# thread_pool configuration
thread_pool:
    analyze:
        size: {{ (ansible_processor_cores * ansible_processor_threads_per_core) |␣
↪round (0, 'floor') | int }}
        queue_size: 5000
    get:
        size: {{ elasticsearch.log.thread_pool.get.size |default((ansible_processor_
↪cores * ansible_processor_threads_per_core)| round (0, 'floor') | int) }}
        queue_size: 1000
    search:
        size: {{ elasticsearch.log.thread_pool.search.size |default(((ansible_
↪processor_cores * ansible_processor_threads_per_core * 3 / 2) + 1) | round (0,
↪'floor') | int) }}
        queue_size: 1000
    write:
        size: {{ elasticsearch.log.thread_pool.write.size |default((ansible_processor_
↪cores * ansible_processor_threads_per_core + 1)| round (0, 'floor') | int) }}
        queue_size: 5000
    warmer:
        core: 1
        max: {{ elasticsearch.log.thread_pool.warmer.max |default(((ansible_processor_
↪cores * ansible_processor_threads_per_core / 2) + 0.5) | round (0, 'floor') | int) }
↪}
        keep_alive: 2m

{% if groups['hosts_elasticsearch_log']|length > 1 %}
# related to affinity and balancing between racks / rooms https://www.elastic.co/
↪guide/en/elasticsearch/reference/current/allocation-awareness.html
cluster.routing.allocation.awareness.attributes: rack_id
node.attr.rack_id: {{ is_balancing|default(vitam_site_name) }}
{% endif %}

# Related to https://www.elastic.co/guide/en/elasticsearch/reference/current/ilm-
↪settings.html
xpack.ilm.enabled: false
indices.lifecycle.history_index_enabled: true

indices.breaker.total.use_real_memory: false

# More tuning
xpack.security.enabled: false
```

(suite sur la page suivante)

(suite de la page précédente)

```
xpack.watcher.enabled: false
```

7.2.4.2.4 Fichier `/vitam/conf/elasticsearch-log/sysconfig/elasticsearch`

```
################################
# Elasticsearch
################################

# Elasticsearch home directory
#ES_HOME=/usr/share/elasticsearch

# Elasticsearch configuration directory
ES_PATH_CONF={{ vitam_defaults.folder.root_path }}/conf/{{ composant.cluster_name }}

# Elasticsearch data directory
#DATA_DIR={{ vitam_defaults.folder.root_path }}/data/{{ composant.cluster_name }}

# Elasticsearch logs directory
#LOG_DIR={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}

# Elasticsearch PID directory
#PID_DIR=/var/run/{{ composant.cluster_name }}

# Heap size defaults to 256m min, 1g max
# Set ES_HEAP_SIZE to 50% of available RAM, but no more than 31g
#ES_JAVA_OPTS=

################################
# Elasticsearch service
################################

# SysV init.d
#
# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5


# Heap new generation
#ES_HEAP_NEWSIZE=

# Maximum direct memory
#ES_DIRECT_SIZE=

# Additional Java OPTS
ES_JAVA_OPTS=""

# Configure restart on package upgrade (true, every other setting will lead to not␣
↪restarting)
#RESTART_ON_UPGRADE=true

# Path to the GC log file
#ES_GC_LOG_FILE={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}
↪/gc.log
```

(suite sur la page suivante)

(suite de la page précédente)

```
ES_TMPDIR={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}

################################
# Elasticsearch service
################################

# SysV init.d
#
# When executing the init script, this user will be used to run the elasticsearch␣
↪service.
# The default value is 'elasticsearch' and is declared in the init.d file.
# Note that this setting is only used by the init script. If changed, make sure that
# the configured user can read and write into the data, work, plugins and log␣
↪directories.
# For systemd service, the user is usually configured in file /usr/lib/systemd/system/
↪elasticsearch.service

# Note: useless for VITAM, as the startup is managed by systemd
ES_USER={{ vitam_defaults.users.vitamdb }}
ES_GROUP={{ vitam_defaults.users.group }}

# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5

################################
# System properties
################################

# Specifies the maximum file descriptor number that can be opened by this process
# When using Systemd, this setting is ignored and the LimitNOFILE defined in
# /usr/lib/systemd/system/elasticsearch.service takes precedence
#MAX_OPEN_FILES=65536

# The maximum number of bytes of memory that may be locked into RAM
# Set to "unlimited" if you use the 'bootstrap.memory_lock: true' option
# in elasticsearch.yml (ES_HEAP_SIZE  must also be set).
# When using Systemd, the LimitMEMLOCK property must be set
# in /usr/lib/systemd/system/elasticsearch.service
#MAX_LOCKED_MEMORY=unlimited

# Maximum number of VMA (Virtual Memory Areas) a process can own
# When using Systemd, this setting is ignored and the 'vm.max_map_count'
# property is set at boot time in /usr/lib/sysctl.d/elasticsearch.conf
#MAX_MAP_COUNT=262144
```

7.2.4.2.5 Fichier `/usr/lib/tmpfiles.d/elasticsearch-log.conf`

```
d    /var/run/{{ composant.cluster_name }}   0755 {{ vitam_defaults.users.vitamdb }} {
↪{ vitam_defaults.users.group }} - -
```

7.2.4.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl start vitam-elasticsearch-log`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl stop vitam-elasticsearch-log`

- Sauvegarde du service

Dans cette version du système, seule une sauvegarde à froid du service est supportée (par la sauvegarde des fichiers de données présents dans `/vitam/data`)

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/`

- Exports

N/A

- gestion de la capacité

N/A

- Réouverture d'un index fermé

Les index sont fermés par action récurrente de Curator ; il est néanmoins possible de rouvrir un index fermé par la commande suivante :

```
curl -XPOST '<adresseIP>:<port>/<index_fermé>/_open'
```

Référence [16]

- actions récurrentes

- cas des batches

N/A

7.2.5 Elasticsearch Data

7.2.5.1 Présentation

Le composant `vitam-elasticsearch-data` est une instance de la base d'indexation elasticsearch stockant les informations relatives aux archives hébergées dans *VITAM*. Elle participe dans ce sens à l'indexation et la recherche des données contenues dans MongoDB.

7.2.5.2 Configuration / fichiers utiles

Se reporter au *DIN*, qui configure le *cluster* ElasticSearch de données.

Les fichiers de configuration sont définis sous `/vitam/conf/elasticsearch-data`.

7.2.5.2.1 Fichier `log4j2.properties`

[16] https://www.elastic.co/guide/en/elasticsearch/reference/2.4/indices-open-close.html

```
status = error

# log action execution errors for easier debugging
logger.action.name = org.elasticsearch.action
logger.action.level = {{ composant.action_log_level }}

appender.console.type = Console
appender.console.name = console
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_name]%marker
↪%m%n

appender.syslog.type = Syslog
appender.syslog.name = syslog
appender.syslog.appName = {{ composant.cluster_name }}
appender.syslog.facility = {{ vitam_defaults.syslog_facility }}
appender.syslog.host = {{ inventory_hostname }}
appender.syslog.protocol = UDP
appender.syslog.port = 514
appender.syslog.layout.type = PatternLayout
# Note: rsyslog only parse RFC3195-formatted syslog messages by default ; AND, to␣
↪make it work with log4j2, we need to start the layout by the app-name.
# IF we were in 5424, we wouldn't have to do this.
appender.syslog.layout.pattern = {{ composant.cluster_name }}: [%d{ISO8601}][%-5p][%-
↪25c{1.}] [%node_name]%marker%m%n
# appender.syslog.format = RFC5424
# appender.syslog.mdcId = esdata

appender.rolling.type = RollingFile
appender.rolling.name = rolling
appender.rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}.log
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_name]%marker
↪%.-10000m%n
appender.rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}${sys:es.
↪logs.cluster_name}-%d{yyyy-MM-dd}-%i.log.gz
appender.rolling.policies.type = Policies
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval = 1
appender.rolling.policies.time.modulate = true
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size = {{ composant.log_appenders.rolling.max_log_file_
↪size }}
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.fileIndex = nomax
appender.rolling.strategy.action.type = Delete
appender.rolling.strategy.action.basepath = ${sys:es.logs.base_path}
appender.rolling.strategy.action.condition.type = IfFileName
appender.rolling.strategy.action.condition.glob = ${sys:es.logs.cluster_name}-*
appender.rolling.strategy.action.condition.nested_condition.type =␣
↪IfAccumulatedFileSize
appender.rolling.strategy.action.condition.nested_condition.exceeds = {{ composant.
↪log_appenders.rolling.max_total_log_size }}

rootLogger.level = {{ composant.log_appenders.root.log_level }}
rootLogger.appenderRef.console.ref = console
```

(suite sur la page suivante)

(suite de la page précédente)

```
rootLogger.appenderRef.rolling.ref = rolling
rootLogger.appenderRef.syslog.ref = syslog

appender.deprecation_rolling.type = RollingFile
appender.deprecation_rolling.name = deprecation_rolling
appender.deprecation_rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_deprecation.log
appender.deprecation_rolling.layout.type = PatternLayout
appender.deprecation_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c{1.}] [%node_
↪name]%marker %.-10000m%n
appender.deprecation_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_deprecation-%i.log.gz
appender.deprecation_rolling.policies.type = Policies
appender.deprecation_rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.deprecation_rolling.policies.size.size = {{ composant.log_appenders.
↪deprecation_rolling.max_log_file_size }}
appender.deprecation_rolling.strategy.type = DefaultRolloverStrategy
appender.deprecation_rolling.strategy.max = {{ composant.log_appenders.deprecation_
↪rolling.max_files }}


logger.deprecation.name = org.elasticsearch.deprecation
logger.deprecation.level = {{ composant.log_appenders.deprecation_rolling.log_level }}
logger.deprecation.appenderRef.deprecation_rolling.ref = deprecation_rolling
logger.deprecation.additivity = false

appender.index_search_slowlog_rolling.type = RollingFile
appender.index_search_slowlog_rolling.name = index_search_slowlog_rolling
appender.index_search_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_search_slowlog.log
appender.index_search_slowlog_rolling.layout.type = PatternLayout
appender.index_search_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c] [
↪%node_name]%marker %.-10000m%n
appender.index_search_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_search_slowlog-%d{yyyy-MM-dd}.
↪log
appender.index_search_slowlog_rolling.policies.type = Policies
appender.index_search_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_search_slowlog_rolling.policies.time.interval = 1
appender.index_search_slowlog_rolling.policies.time.modulate = true

logger.index_search_slowlog_rolling.name = index.search.slowlog
logger.index_search_slowlog_rolling.level = {{ composant.log_appenders.index_search_
↪slowlog_rolling.log_level }}
logger.index_search_slowlog_rolling.appenderRef.index_search_slowlog_rolling.ref =␣
↪index_search_slowlog_rolling
logger.index_search_slowlog_rolling.additivity = false

appender.index_indexing_slowlog_rolling.type = RollingFile
appender.index_indexing_slowlog_rolling.name = index_indexing_slowlog_rolling
appender.index_indexing_slowlog_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_index_indexing_slowlog.log
appender.index_indexing_slowlog_rolling.layout.type = PatternLayout
appender.index_indexing_slowlog_rolling.layout.pattern = [%d{ISO8601}][%-5p][%-25c] [
↪%node_name]%marker %.-10000m%n
appender.index_indexing_slowlog_rolling.filePattern = ${sys:es.logs.base_path}$
↪{sys:file.separator}${sys:es.logs.cluster_name}_index_indexing_slowlog-%d{yyyy-MM-
↪dd}.log
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.index_indexing_slowlog_rolling.policies.type = Policies
appender.index_indexing_slowlog_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.index_indexing_slowlog_rolling.policies.time.interval = 1
appender.index_indexing_slowlog_rolling.policies.time.modulate = true

logger.index_indexing_slowlog.name = index.indexing.slowlog.index
logger.index_indexing_slowlog.level = {{ composant.log_appenders.index_indexing_
↪slowlog_rolling.log_level }}
logger.index_indexing_slowlog.appenderRef.index_indexing_slowlog_rolling.ref = index_
↪indexing_slowlog_rolling
logger.index_indexing_slowlog.additivity = false

appender.audit_rolling.type = RollingFile
appender.audit_rolling.name = audit_rolling
appender.audit_rolling.fileName = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_audit.log
appender.audit_rolling.layout.type = PatternLayout
appender.audit_rolling.layout.pattern = {\
                "@timestamp":"%d{ISO8601}"\
                %varsNotEmpty{, "node.name":"%enc{\%map{node.name}}{JSON}"}\
                %varsNotEmpty{, "node.id":"%enc{\%map{node.id}}{JSON}"}\
                %varsNotEmpty{, "host.name":"%enc{\%map{host.name}}{JSON}"}\
                %varsNotEmpty{, "host.ip":"%enc{\%map{host.ip}}{JSON}"}\
                %varsNotEmpty{, "event.type":"%enc{\%map{event.type}}{JSON}"}\
                %varsNotEmpty{, "event.action":"%enc{\%map{event.action}}{JSON}"}\
                %varsNotEmpty{, "user.name":"%enc{\%map{user.name}}{JSON}"}\
                %varsNotEmpty{, "user.run_by.name":"%enc{\%map{user.run_by.name}}
↪{JSON}"}\
                %varsNotEmpty{, "user.run_as.name":"%enc{\%map{user.run_as.name}}
↪{JSON}"}\
                %varsNotEmpty{, "user.realm":"%enc{\%map{user.realm}}{JSON}"}\
                %varsNotEmpty{, "user.run_by.realm":"%enc{\%map{user.run_by.realm}}
↪{JSON}"}\
                %varsNotEmpty{, "user.run_as.realm":"%enc{\%map{user.run_as.realm}}
↪{JSON}"}\
                %varsNotEmpty{, "user.roles":%map{user.roles}}\
                %varsNotEmpty{, "origin.type":"%enc{\%map{origin.type}}{JSON}"}\
                %varsNotEmpty{, "origin.address":"%enc{\%map{origin.address}}{JSON}"}\
                %varsNotEmpty{, "realm":"%enc{\%map{realm}}{JSON}"}\
                %varsNotEmpty{, "url.path":"%enc{\%map{url.path}}{JSON}"}\
                %varsNotEmpty{, "url.query":"%enc{\%map{url.query}}{JSON}"}\
                %varsNotEmpty{, "request.method":"%enc{\%map{request.method}}{JSON}"}\
                %varsNotEmpty{, "request.body":"%enc{\%map{request.body}}{JSON}"}\
                %varsNotEmpty{, "request.id":"%enc{\%map{request.id}}{JSON}"}\
                %varsNotEmpty{, "action":"%enc{\%map{action}}{JSON}"}\
                %varsNotEmpty{, "request.name":"%enc{\%map{request.name}}{JSON}"}\
                %varsNotEmpty{, "indices":%map{indices}}\
                %varsNotEmpty{, "opaque_id":"%enc{\%map{opaque_id}}{JSON}"}\
                %varsNotEmpty{, "x_forwarded_for":"%enc{\%map{x_forwarded_for}}{JSON}
↪"}\
                %varsNotEmpty{, "transport.profile":"%enc{\%map{transport.profile}}
↪{JSON}"}\
                %varsNotEmpty{, "rule":"%enc{\%map{rule}}{JSON}"}\
                %varsNotEmpty{, "event.category":"%enc{\%map{event.category}}{JSON}"}\
                }%n
# "node.name" node name from the `elasticsearch.yml` settings
# "node.id" node id which should not change between cluster restarts
```

(suite sur la page suivante)

(suite de la page précédente)

```
# "host.name" unresolved hostname of the local node
# "host.ip" the local bound ip (i.e. the ip listening for connections)
# "event.type" a received REST request is translated into one or more transport␣
↪requests. This indicates which processing layer generated the event "rest" or
↪"transport" (internal)
# "event.action" the name of the audited event, eg. "authentication_failed", "access_
↪granted", "run_as_granted", etc.
# "user.name" the subject name as authenticated by a realm
# "user.run_by.name" the original authenticated subject name that is impersonating␣
↪another one.
# "user.run_as.name" if this "event.action" is of a run_as type, this is the subject␣
↪name to be impersonated as.
# "user.realm" the name of the realm that authenticated "user.name"
# "user.run_by.realm" the realm name of the impersonating subject ("user.run_by.name")
# "user.run_as.realm" if this "event.action" is of a run_as type, this is the realm␣
↪name the impersonated user is looked up from
# "user.roles" the roles array of the user; these are the roles that are granting␣
↪privileges
# "origin.type" it is "rest" if the event is originating (is in relation to) a REST␣
↪request; possible other values are "transport" and "ip_filter"
# "origin.address" the remote address and port of the first network hop, i.e. a REST␣
↪proxy or another cluster node
# "realm" name of a realm that has generated an "authentication_failed" or an
↪"authentication_successful"; the subject is not yet authenticated
# "url.path" the URI component between the port and the query string; it is percent␣
↪(URL) encoded
# "url.query" the URI component after the path and before the fragment; it is percent␣
↪(URL) encoded
# "request.method" the method of the HTTP request, i.e. one of GET, POST, PUT, DELETE,
↪ OPTIONS, HEAD, PATCH, TRACE, CONNECT
# "request.body" the content of the request body entity, JSON escaped
# "request.id" a synthentic identifier for the incoming request, this is unique per␣
↪incoming request, and consistent across all audit events generated by that request
# "action" an action is the most granular operation that is authorized and this␣
↪identifies it in a namespaced way (internal)
# "request.name" if the event is in connection to a transport message this is the␣
↪name of the request class, similar to how rest requests are identified by the url␣
↪path (internal)
# "indices" the array of indices that the "action" is acting upon
# "opaque_id" opaque value conveyed by the "X-Opaque-Id" request header
# "x_forwarded_for" the addresses from the "X-Forwarded-For" request header, as a␣
↪verbatim string value (not an array)
# "transport.profile" name of the transport profile in case this is a "connection_
↪granted" or "connection_denied" event
# "rule" name of the applied rulee if the "origin.type" is "ip_filter"
# "event.category" fixed value "elasticsearch-audit"

appender.audit_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.separator}$
↪{sys:es.logs.cluster_name}_audit-%d{yyyy-MM-dd}.log
appender.audit_rolling.policies.type = Policies
appender.audit_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.audit_rolling.policies.time.interval = 1
appender.audit_rolling.policies.time.modulate = true

appender.deprecated_audit_rolling.type = RollingFile
appender.deprecated_audit_rolling.name = deprecated_audit_rolling
appender.deprecated_audit_rolling.fileName = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_access.log
```

(suite sur la page suivante)

(suite de la page précédente)

```
appender.deprecated_audit_rolling.layout.type = PatternLayout
appender.deprecated_audit_rolling.layout.pattern = [%d{ISO8601}] %m%n
appender.deprecated_audit_rolling.filePattern = ${sys:es.logs.base_path}${sys:file.
↪separator}${sys:es.logs.cluster_name}_access-%d{yyyy-MM-dd}.log
appender.deprecated_audit_rolling.policies.type = Policies
appender.deprecated_audit_rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.deprecated_audit_rolling.policies.time.interval = 1
appender.deprecated_audit_rolling.policies.time.modulate = true

logger.xpack_security_audit_logfile.name = org.elasticsearch.xpack.security.audit.
↪logfile.LoggingAuditTrail
logger.xpack_security_audit_logfile.level = info
logger.xpack_security_audit_logfile.appenderRef.audit_rolling.ref = audit_rolling
logger.xpack_security_audit_logfile.additivity = false

logger.xpack_security_audit_deprecated_logfile.name = org.elasticsearch.xpack.
↪security.audit.logfile.DeprecatedLoggingAuditTrail
# set this to "off" instead of "info" to disable the deprecated appender
# in the 6.x releases both the new and the previous appenders are enabled
# for the logfile auditing
logger.xpack_security_audit_deprecated_logfile.level = info
logger.xpack_security_audit_deprecated_logfile.appenderRef.deprecated_audit_rolling.
↪ref = deprecated_audit_rolling
logger.xpack_security_audit_deprecated_logfile.additivity = false

logger.xmlsig.name = org.apache.xml.security.signature.XMLSignature
logger.xmlsig.level = error
logger.samlxml_decrypt.name = org.opensaml.xmlsec.encryption.support.Decrypter
logger.samlxml_decrypt.level = fatal
logger.saml2_decrypt.name = org.opensaml.saml.saml2.encryption.Decrypter
logger.saml2_decrypt.level = fatal
```

7.2.5.2.2 Fichier `jvm.options`

```
## JVM configuration

################################################################
## IMPORTANT: JVM heap size
################################################################
##
## You should always set the min and max JVM heap
## size to the same value. For example, to set
## the heap to 4 GB, set:
##
## -Xms4g
## -Xmx4g
##
## See https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html
## for more information
##
################################################################

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space
```

(suite sur la page suivante)

(suite de la page précédente)

```
-Xms{{ elasticsearch_memory }}
-Xmx{{ elasticsearch_memory }}

################################################################
## Expert settings
################################################################
##
## All settings below this section are considered
## expert settings. Don't tamper with them unless
## you understand what you are doing
##
################################################################

## GC configuration
8-13:-XX:+UseConcMarkSweepGC
8-13:-XX:CMSInitiatingOccupancyFraction=75
8-13:-XX:+UseCMSInitiatingOccupancyOnly

## G1GC Configuration
# NOTE: G1 GC is only supported on JDK version 10 or later
# to use G1GC, uncomment the next two lines and update the version on the
# following three lines to your version of the JDK
# 10-13:-XX:-UseConcMarkSweepGC
# 10-13:-XX:-UseCMSInitiatingOccupancyOnly
14-:-XX:+UseG1GC
14-:-XX:G1ReservePercent=25
14-:-XX:InitiatingHeapOccupancyPercent=30

## DNS cache policy
# cache ttl in seconds for positive DNS lookups noting that this overrides the
# JDK security property networkaddress.cache.ttl; set to -1 to cache forever
-Des.networkaddress.cache.ttl=60
# cache ttl in seconds for negative DNS lookups noting that this overrides the
# JDK security property networkaddress.cache.negative ttl; set to -1 to cache
# forever
-Des.networkaddress.cache.negative.ttl=10

## optimizations

# pre-touch memory pages used by the JVM during initialization
-XX:+AlwaysPreTouch

## basic
# force the server VM (remove on 32-bit client JVMs)
-server

# explicitly set the stack size
-Xss1m

# set to headless, just in case
-Djava.awt.headless=true

# ensure UTF-8 encoding by default (e.g. filenames)
-Dfile.encoding=UTF-8

# use our provided JNA always versus the system one
```

(suite sur la page suivante)

(suite de la page précédente)

```
-Djna.nosys=true

# turn off a JDK optimization that throws away stack traces for common
# exceptions because stack traces are important for debugging
-XX:-OmitStackTraceInFastThrow

# flags to configure Netty
-Dio.netty.noUnsafe=true
-Dio.netty.noKeySetOptimization=true
-Dio.netty.recycler.maxCapacityPerThread=0

# log4j 2
-Dlog4j.shutdownHookEnabled=false
-Dlog4j2.disable.jmx=true
# Prevent from exploit in old log4j2 versions <2.17.1
-Dlog4j2.formatMsgNoLookups=true

-Dlog4j.skipJansi=true
-Djava.io.tmpdir=${ES_TMPDIR}

## heap dumps

# generate a heap dump when an allocation from the Java heap fails
# heap dumps are created in the working directory of the JVM
-XX:+HeapDumpOnOutOfMemoryError

# specify an alternative path for heap dumps; ensure the directory exists and
# has sufficient space
-XX:HeapDumpPath={{ elasticsearch_log_dir }}

# specify an alternative path for JVM fatal error logs
-XX:ErrorFile={{ elasticsearch_log_dir }}/hs_err_pid%p.log

## JDK 8 GC logging

8:-XX:+PrintGCDetails
8:-XX:+PrintGCDateStamps
8:-XX:+PrintTenuringDistribution
8:-XX:+PrintGCApplicationStoppedTime
8:-Xloggc:/var/log/elasticsearch/gc.log
8:-XX:+UseGCLogFileRotation
8:-XX:NumberOfGCLogFiles=32
8:-XX:GCLogFileSize=64m

# JDK 9+ GC logging
9-:-Xlog:gc*,gc+age=trace,safepoint:file={{ elasticsearch_log_dir }}/gc.log:utctime,
↪pid,tags:filecount=32,filesize=64m


-Djna.tmpdir={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}
```

7.2.5.2.3 Fichier `elasticsearch.yml`

```
# ======================== Elasticsearch Configuration =========================
```

(suite sur la page suivante)

(suite de la page précédente)

```
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please consult the documentation for further information on configuration options:
# https://www.elastic.co/guide/en/elasticsearch/reference/index.html
#
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
cluster.name: {{ composant.cluster_name }}
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: {{ inventory_hostname }}
# TODO: Better handling of this as we have to modify wich nodes are requested by VITAM
node.master: {{ is_master|default('true') }}
node.data: {{ is_data|default('true') }}
node.ingest: false
node.ml: false
xpack.ml.enabled: false
#
# Add custom attributes to the node:
#
# node.rack: r1
#
# ----------------------------------- Paths ------------------------------------
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: {{ elasticsearch_data_dir }}
#
# Path to log files:
#
path.logs: {{ elasticsearch_log_dir }}
#
# ----------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
# = Disable swapping
bootstrap.memory_lock: true
#
# Make sure that the `ES_HEAP_SIZE` environment variable is set to about half the␣
↪memory
# available on the system and that the owner of the process is allowed to use this␣
↪limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
```

(suite sur la page suivante)

(suite de la page précédente)

```
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
# Note : if installing to localhost, notably a docker container, we need to bind␣
↪larger than localhost
{% if inventory_hostname in single_vm_hostnames %}
network.host: 0.0.0.0
http.cors.enabled: true
http.cors.allow-origin: "*"
{% else %}
network.host: {{ ip_service }}
{% endif %}
#
# Set a custom port for HTTP:
#
http.port: {{ composant.port_http }}
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.seed_hosts: [ {% for host in groups['hosts_elasticsearch_data'] %}"{{␣
↪hostvars[host]['ip_service'] }}"{% if not loop.last %},{% endif %}{% endfor %} ]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
cluster.initial_master_nodes: [ {% for host in groups['hosts_elasticsearch_data'] %}"{
↪{ host }}"{% if not loop.last %},{% endif %}{% endfor %} ]

cluster.no_master_block: all
#
# For more information, consult the discovery and cluster formation module␣
↪documentation.
#
# ---------------------------------- Gateway -----------------------------------
#
# Block initial recovery after a full cluster restart until N nodes are started:
#
gateway.expected_nodes: {{ (groups['hosts_elasticsearch_data'] | length) }}
gateway.recover_after_nodes: {{ ((groups['hosts_elasticsearch_data']|length / 2)+1)|␣
↪round (0, 'floor')| int }}
#
# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-gateway.
↪html>
#
# --------------------------------- Various -----------------------------------
#
# Disable starting multiple nodes on a single system:
#
# node.max_local_storage_nodes: 1
#
# Require explicit names when deleting indices:
#
```

(suite sur la page suivante)

(suite de la page précédente)

```
action.destructive_requires_name: true

# For Vitam multiquery
indices.query.bool.max_clause_count: 10000

{% if composant.index_buffer_size_ratio is defined %}
# some perforamnce tuning ; see https://www.elastic.co/guide/en/elasticsearch/
↪reference/6.4/tune-for-indexing-speed.html
# 0.1 may be enough, cots_vars declares {{ composant.index_buffer_size_ratio }} as␣
↪ratio on total memory {{ elasticsearch_memory }}
indices.memory.index_buffer_size: {{ ((elasticsearch_memory_value|int)*(composant.
↪index_buffer_size_ratio|float))|round (0, 'floor')| int }}{{ elasticsearch_memory_
↪unit }}
{% endif %}

indices.mapping.dynamic_timeout: {{ composant.dynamic_timeout |default('30s') }}

# thread_pool configuration
thread_pool:
    analyze:
        size: {{ (ansible_processor_cores * ansible_processor_threads_per_core) |␣
↪round (0, 'floor') | int }}
        queue_size: 5000
    get:
        size: {{ elasticsearch.data.thread_pool.get.size |default((ansible_processor_
↪cores * ansible_processor_threads_per_core)| round (0, 'floor') | int) }}
        queue_size: 5000
    search:
        size: {{ elasticsearch.data.thread_pool.search.size |default(((ansible_
↪processor_cores * ansible_processor_threads_per_core * 3 / 2) + 1) | round (0,
↪'floor') | int) }}
        min_queue_size: 1000
        queue_size: 5000
    write:
        size: {{ elasticsearch.data.thread_pool.write.size |default((ansible_
↪processor_cores * ansible_processor_threads_per_core + 1)| round (0, 'floor') |␣
↪int) }}
        queue_size: 5000
    warmer:
        core: 1
        max: {{ elasticsearch.data.thread_pool.warmer.max |default(((ansible_
↪processor_cores * ansible_processor_threads_per_core / 2) + 0.5) | round (0, 'floor
↪') | int) }}
        keep_alive: 2m

# Note : the 0.5 in the previous expression is for there is only 1 CPU (else the␣
↪thread pool size would be zero) ! ; Note bis : max 10 threads #
# Note : in ES5 and further : the thread pool "refresh" is of type scaling with a␣
↪keep-alive of 5m and a max of min(10, (# of available processors)/2)

# related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/modules-
↪fielddata.html
indices.fielddata.cache.size: {{ composant.indices_fielddata_cache_size }}

# related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/circuit-
↪breaker.html#fielddata-circuit-breaker
indices.breaker.fielddata.limit: {{ composant.indices_breaker_fielddata_limit }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% if groups['hosts_elasticsearch_data']|length > 1 %}
# related to affinity and balancing between racks / rooms https://www.elastic.co/
↪guide/en/elasticsearch/reference/current/allocation-awareness.html
cluster.routing.allocation.awareness.attributes: rack_id
node.attr.rack_id: {{ is_balancing|default(vitam_site_name) }}
{% endif %}

indices.breaker.total.use_real_memory: false


# Related to https://www.elastic.co/guide/en/elasticsearch/reference/current/ilm-
↪settings.html
xpack.ilm.enabled: false
indices.lifecycle.history_index_enabled: false

# More tuning
xpack.security.enabled: false
xpack.watcher.enabled: false
```

7.2.5.2.4 Fichier `sysconfig/elasticsearch`

```
################################
# Elasticsearch
################################

# Elasticsearch home directory
#ES_HOME=/usr/share/elasticsearch

# Elasticsearch configuration directory
ES_PATH_CONF={{ vitam_defaults.folder.root_path }}/conf/{{ composant.cluster_name }}

# Elasticsearch data directory
#DATA_DIR={{ vitam_defaults.folder.root_path }}/data/{{ composant.cluster_name }}

# Elasticsearch logs directory
#LOG_DIR={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}

# Elasticsearch PID directory
#PID_DIR=/var/run/{{ composant.cluster_name }}

# Heap size defaults to 256m min, 1g max
# Set ES_HEAP_SIZE to 50% of available RAM, but no more than 31g
#ES_JAVA_OPTS=

################################
# Elasticsearch service
################################

# SysV init.d
#
# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5
```

(suite sur la page suivante)

(suite de la page précédente)

```
# Heap new generation
#ES_HEAP_NEWSIZE=

# Maximum direct memory
#ES_DIRECT_SIZE=

# Additional Java OPTS
ES_JAVA_OPTS=""

# Configure restart on package upgrade (true, every other setting will lead to not␣
↪restarting)
#RESTART_ON_UPGRADE=true

# Path to the GC log file
#ES_GC_LOG_FILE={{ vitam_defaults.folder.root_path }}/log/{{ composant.cluster_name }}
↪/gc.log

ES_TMPDIR={{ vitam_defaults.folder.root_path }}/tmp/{{ composant.cluster_name }}

################################
# Elasticsearch service
################################

# SysV init.d
#
# When executing the init script, this user will be used to run the elasticsearch␣
↪service.
# The default value is 'elasticsearch' and is declared in the init.d file.
# Note that this setting is only used by the init script. If changed, make sure that
# the configured user can read and write into the data, work, plugins and log␣
↪directories.
# For systemd service, the user is usually configured in file /usr/lib/systemd/system/
↪elasticsearch.service

# Note: useless for VITAM, as the startup is managed by systemd
ES_USER={{ vitam_defaults.users.vitamdb }}
ES_GROUP={{ vitam_defaults.users.group }}

# The number of seconds to wait before checking if Elasticsearch started successfully␣
↪as a daemon process
ES_STARTUP_SLEEP_TIME=5

################################
# System properties
################################

# Specifies the maximum file descriptor number that can be opened by this process
# When using Systemd, this setting is ignored and the LimitNOFILE defined in
# /usr/lib/systemd/system/elasticsearch.service takes precedence
#MAX_OPEN_FILES=65536

# The maximum number of bytes of memory that may be locked into RAM
# Set to "unlimited" if you use the 'bootstrap.memory_lock: true' option
# in elasticsearch.yml (ES_HEAP_SIZE  must also be set).
# When using Systemd, the LimitMEMLOCK property must be set
```

(suite sur la page suivante)

(suite de la page précédente)

```
# in /usr/lib/systemd/system/elasticsearch.service
#MAX_LOCKED_MEMORY=unlimited

# Maximum number of VMA (Virtual Memory Areas) a process can own
# When using Systemd, this setting is ignored and the 'vm.max_map_count'
# property is set at boot time in /usr/lib/sysctl.d/elasticsearch.conf
#MAX_MAP_COUNT=262144
```

7.2.5.2.5 Fichier `/usr/lib/tmpfiles.d/elasticsearch-data.conf`

```
d    /var/run/{{ composant.cluster_name }}   0755 {{ vitam_defaults.users.vitamdb }} {
↪{ vitam_defaults.users.group }} - -
```

7.2.5.3 Opérations

- Démarrage du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl start vitam-elasticsearch-data`

- Arrêt du service

Les commandes suivantes sont à passer sur les différentes machines constituant le cluster Elasticsearch.

En tant qu'utilisateur root : `systemctl stop vitam-elasticsearch-data`

- Sauvegarde du service

Dans cette version du système, seule une sauvegarde à froid du service est supportée (par la sauvegarde des fichiers de données présents dans `/vitam/data`)

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

7.2.6 Grafana

7.2.6.1 Présentation

Grafana est l'outil permettant de visualiser et de représenter de façon agrégée des données stockées dans Prometheus.

Afin de forcer la bonne version de grafana, *VITAM* déploie un installeur-chapeau `vitam-grafana`.

Prudence : Lors de la première installation de grafana, il convient de le configurer manuellement.

7.2.6.2 Configuration / fichiers utiles

Cet outil est fourni en tant que composant extra dans la suite logicielle *VITAM*.

Grafana sera déployé sur l'ensemble des machines renseignées dans le groupe `[hosts_grafana]` de votre fichier d'inventaire.

Grafana est déployé avec un dashboard général pour le suivi des indicateurs clés de la solution Vitam.

Note : Dans le cas d'utilisation d'une offre froide, un dashboard dédié est disponible sur Grafana pour le suivi du bon fonctionnement de l'offre.

7.2.6.2.1 Fichier `/etc/grafana/grafana.ini`

Ce fichier est à récupérer depuis la version à installer. Ensuite il faut apporter les modifications nécessaires.

Avertissement : Lors des montées de version de Grafana, il faut récupérer le fichier original `grafana.ini` de la nouvelle version et reporter les modifications apportées par Vitam.

```
######################### Grafana Configuration ##############################
#
# Everything has defaults so you only need to uncomment things you want to
# change

# possible values : production, development
;app_mode = production

# instance name, defaults to HOSTNAME environment variable value or hostname if␣
↪HOSTNAME var is empty
;instance_name = ${HOSTNAME}

################################### Paths ####################################
[paths]
# Path to where grafana can store temp files, sessions, and the sqlite3 db (if that␣
↪is used)
;data = /var/lib/grafana

# Temporary files in `data` directory older than given duration will be removed
;temp_data_lifetime = 24h

# Directory where grafana can store logs
logs = {{ vitam_defaults.folder.root_path | default('/vitam') }}/log/grafana

# Directory where grafana will automatically scan and look for plugins
;plugins = /var/lib/grafana/plugins

# folder that contains provisioning config files that grafana will apply on startup␣
↪and while running.
```

(suite sur la page suivante)

(suite de la page précédente)

```
;provisioning = conf/provisioning

#################################### Server ####################################
[server]
# Protocol (http, https, h2, socket)
;protocol = http

# The ip address to bind to, empty will bind to all interfaces
;http_addr =

# The http port to use
http_port = {{ grafana.http_port | default(3000) }}

# The public facing domain name used to access grafana from a browser
;domain = localhost

# Redirect to correct domain if host header does not match domain
# Prevents DNS rebinding attacks
;enforce_domain = false

# The full public facing url you use in browser, used for redirects and emails
# If you use reverse proxy and sub path specify full url (with sub path)
;root_url = %(protocol)s://%(domain)s:%(http_port)s/

# Serve Grafana from subpath specified in `root_url` setting. By default it is set to␣
↪`false` for compatibility reasons.
;serve_from_sub_path = false
#################################################################################
# VITAM added this, if you dont use reverse proxy, remove following two lines  #
root_url = http://{{ ip_admin }}:{{ grafana.http_port | default(3000) }}/grafana
serve_from_sub_path = true
################################## VITAM END ###################################
# Log web requests
;router_logging = false

# the path relative working path
;static_root_path = public

# enable gzip
;enable_gzip = false

# https certs & key file
;cert_file =
;cert_key =

# Unix socket path
;socket =

#################################### Database ####################################
[database]
# You can configure the database connection by specifying type, host, name, user and␣
↪password
# as separate properties or as on string using the url properties.

# Either "mysql", "postgres" or "sqlite3", it's your choice
;type = sqlite3
;host = 127.0.0.1:3306
```

(suite sur la page suivante)

(suite de la page précédente)

```
;name = grafana
;user = root
# If the password contains # or ; you have to wrap it with triple quotes. Ex """
↪#password;"""
;password =

# Use either URL or the previous fields to configure the database
# Example: mysql://user:secret@host:port/database
;url =

# For "postgres" only, either "disable", "require" or "verify-full"
;ssl_mode = disable

;ca_cert_path =
;client_key_path =
;client_cert_path =
;server_cert_name =

# For "sqlite3" only, path relative to data_path setting
;path = grafana.db

# Max idle conn setting default is 2
;max_idle_conn = 2

# Max conn setting default is 0 (mean not set)
;max_open_conn =

# Connection Max Lifetime default is 14400 (means 14400 seconds or 4 hours)
;conn_max_lifetime = 14400

# Set to true to log the sql calls and execution times.
;log_queries =

# For "sqlite3" only. cache mode setting used for connecting to the database.␣
↪(private, shared)
;cache_mode = private

################################## Cache server #################################
[remote_cache]
# Either "redis", "memcached" or "database" default is "database"
;type = database

# cache connectionstring options
# database: will use Grafana primary database.
# redis: config like redis server e.g. `addr=127.0.0.1:6379,pool_size=100,db=0,
↪ssl=false`. Only addr is required. ssl may be 'true', 'false', or 'insecure'.
# memcache: 127.0.0.1:11211
;connstr =

################################### Data proxy ###################################
[dataproxy]

# This enables data proxy logging, default is false
;logging = false

# How long the data proxy should wait before timing out default is 30 (seconds)
;timeout = 30
```

(suite sur la page suivante)

(suite de la page précédente)

```
# If enabled and user is not anonymous, data proxy will add X-Grafana-User header␣
↪with username into the request, default is false.
;send_user_header = false

#################################### Analytics ####################################
[analytics]
# Server reporting, sends usage counters to stats.grafana.org every 24 hours.
# No ip addresses are being tracked, only simple counters to track
# running instances, dashboard and error counts. It is very helpful to us.
# Change this option to false to disable reporting.
reporting_enabled = false

# Set to false to disable all checks to https://grafana.net
# for new vesions (grafana itself and plugins), check is used
# in some UI views to notify that grafana or plugin update exists
# This option does not cause any auto updates, nor send any information
# only a GET request to http://grafana.com to get latest versions
check_for_updates = false

# Google Analytics universal tracking code, only enabled if you specify an id here
;google_analytics_ua_id =

# Google Tag Manager ID, only enabled if you specify an id here
;google_tag_manager_id =

#################################### Security ####################################
[security]
# disable creation of admin user on first start of grafana
;disable_initial_admin_creation = false

# default admin user, created on startup
admin_user = {{ grafana.admin_user }}

# default admin password, can be changed before first start of grafana, or in profile␣
↪settings
admin_password = {{ grafana.admin_password }}

# used for signing
;secret_key = SW2YcwTIb9zpOOhoPsMm

# disable gravatar profile images
;disable_gravatar = false

# data source proxy whitelist (ip_or_domain:port separated by spaces)
;data_source_proxy_whitelist =

# disable protection against brute force login attempts
;disable_brute_force_login_protection = false

# set to true if you host Grafana behind HTTPS. default is false.
;cookie_secure = false

# set cookie SameSite attribute. defaults to `lax`. can be set to "lax", "strict",
↪"none" and "disabled"
;cookie_samesite = lax
```

(suite sur la page suivante)

(suite de la page précédente)

```
# set to true if you want to allow browsers to render Grafana in a <frame>, <iframe>,
↪<embed> or <object>. default is false.
;allow_embedding = false

# Set to true if you want to enable http strict transport security (HSTS) response␣
↪header.
# This is only sent when HTTPS is enabled in this configuration.
# HSTS tells browsers that the site should only be accessed using HTTPS.
# The default version will change to true in the next minor release, 6.3.
;strict_transport_security = false

# Sets how long a browser should cache HSTS. Only applied if strict_transport_
↪security is enabled.
;strict_transport_security_max_age_seconds = 86400

# Set to true if to enable HSTS preloading option. Only applied if strict_transport_
↪security is enabled.
;strict_transport_security_preload = false

# Set to true if to enable the HSTS includeSubDomains option. Only applied if strict_
↪transport_security is enabled.
;strict_transport_security_subdomains = false

# Set to true to enable the X-Content-Type-Options response header.
# The X-Content-Type-Options response HTTP header is a marker used by the server to␣
↪indicate that the MIME types advertised
# in the Content-Type headers should not be changed and be followed. The default will␣
↪change to true in the next minor release, 6.3.
;x_content_type_options = false

# Set to true to enable the X-XSS-Protection header, which tells browsers to stop␣
↪pages from loading
# when they detect reflected cross-site scripting (XSS) attacks. The default will␣
↪change to true in the next minor release, 6.3.
;x_xss_protection = false

################################### Snapshots #####################################
[snapshots]
# snapshot sharing options
;external_enabled = true
;external_snapshot_url = https://snapshots-origin.raintank.io
;external_snapshot_name = Publish to snapshot.raintank.io

# Set to true to enable this Grafana instance act as an external snapshot server and␣
↪allow unauthenticated requests for
# creating and deleting snapshots.
;public_mode = false

# remove expired snapshot
;snapshot_remove_expired = true

############################### Dashboards History ################################
[dashboards]
# Number dashboard versions to keep (per dashboard). Default: 20, Minimum: 1
;versions_to_keep = 20

# Minimum dashboard refresh interval. When set, this will restrict users to set the␣
↪refresh interval of a dashboard lower than given interval. Per default this is 5␣
↪seconds.
```

(suite sur la page suivante)

(suite de la page précédente)

```
# The interval string is a possibly signed sequence of decimal numbers, followed by a␣
↪unit suffix (ms, s, m, h, d), e.g. 30s or 1m.
;min_refresh_interval = 5s

#################################### Users ###############################################
[users]
# disable user signup / registration
;allow_sign_up = true

# Allow non admin users to create organizations
;allow_org_create = true

# Set to true to automatically assign new users to the default organization (id 1)
;auto_assign_org = true

# Set this value to automatically add new users to the provided organization (if auto_
↪assign_org above is set to true)
;auto_assign_org_id = 1

# Default role new users will be automatically assigned (if disabled above is set to␣
↪true)
;auto_assign_org_role = Viewer

# Require email validation before sign up completes
;verify_email_enabled = false

# Background text for the user field on the login page
;login_hint = email or username
;password_hint = password

# Default UI theme ("dark" or "light")
;default_theme = dark

# External user management, these options affect the organization users view
;external_manage_link_url =
;external_manage_link_name =
;external_manage_info =

# Viewers can edit/inspect dashboard settings in the browser. But not save the␣
↪dashboard.
;viewers_can_edit = false

# Editors can administrate dashboard, folders and teams they create
;editors_can_admin = false

[auth]
# Login cookie name
;login_cookie_name = grafana_session

# The lifetime (days) an authenticated user can be inactive before being required to␣
↪login at next visit. Default is 7 days,
;login_maximum_inactive_lifetime_days = 7

# The maximum lifetime (days) an authenticated user can be logged in since login time␣
↪before being required to login. Default is 30 days.
;login_maximum_lifetime_days = 30
```

(suite sur la page suivante)

(suite de la page précédente)

```
# How often should auth tokens be rotated for authenticated users when being active.␣
↪The default is each 10 minutes.
;token_rotation_interval_minutes = 10

# Set to true to disable (hide) the login form, useful if you use OAuth, defaults to␣
↪false
;disable_login_form = false

# Set to true to disable the signout link in the side menu. useful if you use auth.
↪proxy, defaults to false
;disable_signout_menu = false

# URL to redirect the user to after sign out
;signout_redirect_url =

# Set to true to attempt login with OAuth automatically, skipping the login screen.
# This setting is ignored if multiple OAuth providers are configured.
;oauth_auto_login = false

# OAuth state max age cookie duration. Defaults to 60 seconds.
;oauth_state_cookie_max_age = 60

# limit of api_key seconds to live before expiration
;api_key_max_seconds_to_live = -1

################################### Anonymous Auth ##############################
[auth.anonymous]
# enable anonymous access
;enabled = false

# specify organization name that should be used for unauthenticated users
;org_name = Main Org.

# specify role for unauthenticated users
;org_role = Viewer

#################################### Github Auth ################################
[auth.github]
;enabled = false
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = user:email,read:org
;auth_url = https://github.com/login/oauth/authorize
;token_url = https://github.com/login/oauth/access_token
;api_url = https://api.github.com/user
;allowed_domains =
;team_ids =
;allowed_organizations =

#################################### GitLab Auth ################################
[auth.gitlab]
;enabled = false
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = api
```

(suite sur la page suivante)

(suite de la page précédente)

```
;auth_url = https://gitlab.com/oauth/authorize
;token_url = https://gitlab.com/oauth/token
;api_url = https://gitlab.com/api/v4
;allowed_domains =
;allowed_groups =

#################################### Google Auth ##########################
[auth.google]
;enabled = false
;allow_sign_up = true
;client_id = some_client_id
;client_secret = some_client_secret
;scopes = https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/
↪auth/userinfo.email
;auth_url = https://accounts.google.com/o/oauth2/auth
;token_url = https://accounts.google.com/o/oauth2/token
;api_url = https://www.googleapis.com/oauth2/v1/userinfo
;allowed_domains =
;hosted_domain =

#################################### Grafana.com Auth ####################
[auth.grafana_com]
;enabled = false
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = user:email
;allowed_organizations =

#################################### Azure AD OAuth #######################
[auth.azuread]
;name = Azure AD
;enabled = false
;allow_sign_up = true
;client_id = some_client_id
;client_secret = some_client_secret
;scopes = openid email profile
;auth_url = https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/authorize
;token_url = https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token
;allowed_domains =
;allowed_groups =

#################################### Okta OAuth ##########################
[auth.okta]
;name = Okta
;enabled = false
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = openid profile email groups
;auth_url = https://<tenant-id>.okta.com/oauth2/v1/authorize
;token_url = https://<tenant-id>.okta.com/oauth2/v1/token
;api_url = https://<tenant-id>.okta.com/oauth2/v1/userinfo
;allowed_domains =
;allowed_groups =
;role_attribute_path =
```

(suite sur la page suivante)

(suite de la page précédente)

```
#################################### Generic OAuth ##########################
[auth.generic_oauth]
;enabled = false
;name = OAuth
;allow_sign_up = true
;client_id = some_id
;client_secret = some_secret
;scopes = user:email,read:org
;email_attribute_name = email:primary
;email_attribute_path =
;auth_url = https://foo.bar/login/oauth/authorize
;token_url = https://foo.bar/login/oauth/access_token
;api_url = https://foo.bar/user
;allowed_domains =
;team_ids =
;allowed_organizations =
;role_attribute_path =
;tls_skip_verify_insecure = false
;tls_client_cert =
;tls_client_key =
;tls_client_ca =

################################### Basic Auth ##############################
[auth.basic]
enabled = false

################################### Auth Proxy ##############################
[auth.proxy]
;enabled = false
;header_name = X-WEBAUTH-USER
;header_property = username
;auto_sign_up = true
;sync_ttl = 60
;whitelist = 192.168.1.1, 192.168.2.1
;headers = Email:X-User-Email, Name:X-User-Name
# Read the auth proxy docs for details on what the setting below enables
;enable_login_token = false

################################### Auth LDAP ###############################
[auth.ldap]
;enabled = false
;config_file = /etc/grafana/ldap.toml
;allow_sign_up = true

# LDAP backround sync (Enterprise only)
# At 1 am every day
;sync_cron = "0 0 1 * * *"
;active_sync_enabled = true

################################### SMTP / Emailing #########################
[smtp]
;enabled = false
;host = localhost:25
;user =
# If the password contains # or ; you have to wrap it with triple quotes. Ex """
↪#password;"""
;password =
```

(suite sur la page suivante)

(suite de la page précédente)

```
;cert_file =
;key_file =
;skip_verify = false
;from_address = admin@grafana.localhost
;from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com

[emails]
;welcome_email_on_sign_up = false
;templates_pattern = emails/*.html

#################################### Logging ##########################################
[log]
# Either "console", "file", "syslog". Default is console and file
# Use space to separate multiple modes, e.g. "console file"
;mode = console file

# Either "debug", "info", "warn", "error", "critical", default is "info"
;level = info

# optional settings to set different levels for specific loggers. Ex filters =␣
↪sqlstore:debug
;filters =

# For "console" mode only
[log.console]
;level =

# log line format, valid options are text, console and json
;format = console

# For "file" mode only
[log.file]
;level =

# log line format, valid options are text, console and json
;format = text

# This enables automated log rotate(switch of following options), default is true
;log_rotate = true

# Max line number of single file, default is 1000000
;max_lines = 1000000

# Max size shift of single file, default is 28 means 1 << 28, 256MB
;max_size_shift = 28

# Segment log daily, default is true
;daily_rotate = true

# Expired days of log file(delete after max days), default is 7
;max_days = 7

[log.syslog]
;level =
```

(suite sur la page suivante)

(suite de la page précédente)

```
# log line format, valid options are text, console and json
;format = text

# Syslog network type and address. This can be udp, tcp, or unix. If left blank, the␣
↪default unix endpoints will be used.
;network =
;address =

# Syslog facility. user, daemon and local0 through local7 are valid.
;facility =

# Syslog tag. By default, the process' argv[0] is used.
;tag =

#################################### Usage Quotas ############################
[quota]
; enabled = false

#### set quotas to -1 to make unlimited. ####
# limit number of users per Org.
; org_user = 10

# limit number of dashboards per Org.
; org_dashboard = 100

# limit number of data_sources per Org.
; org_data_source = 10

# limit number of api_keys per Org.
; org_api_key = 10

# limit number of orgs a user can create.
; user_org = 10

# Global limit of users.
; global_user = -1

# global limit of orgs.
; global_org = -1

# global limit of dashboards
; global_dashboard = -1

# global limit of api_keys
; global_api_key = -1

# global limit on number of logged in users.
; global_session = -1

#################################### Alerting ################################
[alerting]
# Disable alerting engine & UI features
;enabled = true
# Makes it possible to turn off alert rule execution but alerting UI is visible
;execute_alerts = true

# Default setting for new alert rules. Defaults to categorize error and timeouts as␣
↪alerting. (alerting, keep_state)
```

(suite sur la page suivante)

(suite de la page précédente)

```
;error_or_timeout = alerting

# Default setting for how Grafana handles nodata or null values in alerting.␣
↪(alerting, no_data, keep_state, ok)
;nodata_or_nullvalues = no_data

# Alert notifications can include images, but rendering many images at the same time␣
↪can overload the server
# This limit will protect the server from render overloading and make sure␣
↪notifications are sent out quickly
;concurrent_render_limit = 5


# Default setting for alert calculation timeout. Default value is 30
;evaluation_timeout_seconds = 30

# Default setting for alert notification timeout. Default value is 30
;notification_timeout_seconds = 30

# Default setting for max attempts to sending alert notifications. Default value is 3
;max_attempts = 3

# Makes it possible to enforce a minimal interval between evaluations, to reduce load␣
↪on the backend
;min_interval_seconds = 1

#################################### Explore #####################################
[explore]
# Enable the Explore section
;enabled = true

############################## Internal Grafana Metrics ##########################
# Metrics available at HTTP API Url /metrics
[metrics]
# Disable / Enable internal metrics
;enabled           = true
# Graphite Publish interval
;interval_seconds  = 10
# Disable total stats (stat_totals_*) metrics to be generated
;disable_total_stats = false

#If both are set, basic auth will be required for the metrics endpoint.
; basic_auth_username =
; basic_auth_password =

# Send internal metrics to Graphite
[metrics.graphite]
# Enable by setting the address setting (ex localhost:2003)
;address =
;prefix = prod.grafana.%(instance_name)s.

############################### Grafana.com integration ##########################
# Url used to import dashboards directly from Grafana.com
[grafana_com]
;url = https://grafana.com

############################### Distributed tracing ##############################
```

(suite sur la page suivante)

(suite de la page précédente)

```
[tracing.jaeger]
# Enable by setting the address sending traces to jaeger (ex localhost:6831)
;address = localhost:6831
# Tag that will always be included in when creating new spans. ex (tag1:value1,
↪tag2:value2)
;always_included_tag = tag1:value1
# Type specifies the type of the sampler: const, probabilistic, rateLimiting, or␣
↪remote
;sampler_type = const
# jaeger samplerconfig param
# for "const" sampler, 0 or 1 for always false/true respectively
# for "probabilistic" sampler, a probability between 0 and 1
# for "rateLimiting" sampler, the number of spans per second
# for "remote" sampler, param is the same as for "probabilistic"
# and indicates the initial sampling rate before the actual one
# is received from the mothership
;sampler_param = 1
# Whether or not to use Zipkin propagation (x-b3- HTTP headers).
;zipkin_propagation = false
# Setting this to true disables shared RPC spans.
# Not disabling is the most common setting when using Zipkin elsewhere in your␣
↪infrastructure.
;disable_shared_zipkin_spans = false

############################## External image storage ##########################
[external_image_storage]
# Used for uploading images to public servers so they can be included in slack/email␣
↪messages.
# you can choose between (s3, webdav, gcs, azure_blob, local)
;provider =

[external_image_storage.s3]
;endpoint =
;path_style_access =
;bucket =
;region =
;path =
;access_key =
;secret_key =

[external_image_storage.webdav]
;url =
;public_url =
;username =
;password =

[external_image_storage.gcs]
;key_file =
;bucket =
;path =

[external_image_storage.azure_blob]
;account_name =
;account_key =
;container_name =

[external_image_storage.local]
```

(suite sur la page suivante)

(suite de la page précédente)

```
# does not require any configuration

[rendering]
# Options to configure a remote HTTP image rendering service, e.g. using https://
↪github.com/grafana/grafana-image-renderer.
# URL to a remote HTTP image renderer service, e.g. http://localhost:8081/render,␣
↪will enable Grafana to render panels and dashboards to PNG-images using HTTP␣
↪requests to an external service.
;server_url =
# If the remote HTTP image renderer service runs on a different server than the␣
↪Grafana server you may have to configure this to a URL where Grafana is reachable,␣
↪e.g. http://grafana.domain/.
;callback_url =
# Concurrent render request limit affects when the /render HTTP endpoint is used.␣
↪Rendering many images at the same time can overload the server,
# which this setting can help protect against by only allowing a certain amount of␣
↪concurrent requests.
;concurrent_render_request_limit = 30

[panels]
# If set to true Grafana will allow script tags in text panels. Not recommended as it␣
↪enable XSS vulnerabilities.
;disable_sanitize_html = false

[plugins]
;enable_alpha = false
;app_tls_skip_verify_insecure = false
# Enter a comma-separated list of plugin identifiers to identify plugins that are␣
↪allowed to be loaded even if they lack a valid signature.
;allow_loading_unsigned_plugins =

######################### Grafana Image Renderer Plugin ##########################
[plugin.grafana-image-renderer]
# Instruct headless browser instance to use a default timezone when not provided by␣
↪Grafana, e.g. when rendering panel image of alert.
# See ICU's metaZones.txt (https://cs.chromium.org/chromium/src/third_party/icu/
↪source/data/misc/metaZones.txt) for a list of supported
# timezone IDs. Fallbacks to TZ environment variable if not set.
;rendering_timezone =

# Instruct headless browser instance to use a default language when not provided by␣
↪Grafana, e.g. when rendering panel image of alert.
# Please refer to the HTTP header Accept-Language to understand how to format this␣
↪value, e.g. 'fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5'.
;rendering_language =

# Instruct headless browser instance to use a default device scale factor when not␣
↪provided by Grafana, e.g. when rendering panel image of alert.
# Default is 1. Using a higher value will produce more detailed images (higher DPI),␣
↪but will require more disk space to store an image.
;rendering_viewport_device_scale_factor =

# Instruct headless browser instance whether to ignore HTTPS errors during navigation.
↪ Per default HTTPS errors are not ignored. Due to
# the security risk it's not recommended to ignore HTTPS errors.
;rendering_ignore_https_errors =
```

(suite sur la page suivante)

(suite de la page précédente)

```
# Instruct headless browser instance whether to capture and log verbose information␣
↪when rendering an image. Default is false and will
# only capture and log error messages. When enabled, debug messages are captured and␣
↪logged as well.
# For the verbose information to be included in the Grafana server log you have to␣
↪adjust the rendering log level to debug, configure
# [log].filter = rendering:debug.
;rendering_verbose_logging =

# Instruct headless browser instance whether to output its debug and error messages␣
↪into running process of remote rendering service.
# Default is false. This can be useful to enable (true) when troubleshooting.
;rendering_dumpio =

# Additional arguments to pass to the headless browser instance. Default is --no-
↪sandbox. The list of Chromium flags can be found
# here (https://peter.sh/experiments/chromium-command-line-switches/). Multiple␣
↪arguments is separated with comma-character.
;rendering_args =

# You can configure the plugin to use a different browser binary instead of the pre-
↪packaged version of Chromium.
# Please note that this is not recommended, since you may encounter problems if the␣
↪installed version of Chrome/Chromium is not
# compatible with the plugin.
;rendering_chrome_bin =

# Instruct how headless browser instances are created. Default is 'default' and will␣
↪create a new browser instance on each request.
# Mode 'clustered' will make sure that only a maximum of browsers/incognito pages can␣
↪execute concurrently.
# Mode 'reusable' will have one browser instance and will create a new incognito page␣
↪on each request.
;rendering_mode =

# When rendering_mode = clustered you can instruct how many browsers or incognito␣
↪pages can execute concurrently. Default is 'browser'
# and will cluster using browser instances.
# Mode 'context' will cluster using incognito pages.
;rendering_clustering_mode =
# When rendering_mode = clustered you can define maximum number of browser instances/
↪incognito pages that can execute concurrently..
;rendering_clustering_max_concurrency =

# Limit the maxiumum viewport width, height and device scale factor that can be␣
↪requested.
;rendering_viewport_max_width =
;rendering_viewport_max_height =
;rendering_viewport_max_device_scale_factor =

# Change the listening host and port of the gRPC server. Default host is 127.0.0.1␣
↪and default port is 0 and will automatically assign
# a port not in use.
;grpc_host =
;grpc_port =

[enterprise]
```

(suite sur la page suivante)

(suite de la page précédente)

```
# Path to a valid Grafana Enterprise license.jwt file
;license_path =

[feature_toggles]
# enable features, separated by spaces
;enable =
```

7.2.6.3 Opérations

- Démarrage du service :

  En tant qu'utilisateur root :

```
systemctl start grafana-server
```

- Arrêt du service :

  En tant qu'utilisateur root :

```
systemctl stop grafana-server
```

- Consultation des logs :

  En tant qu'utilisateur root :

```
journalctl -u grafana-server
```

- Supervision du service

Grafana possède une IHM accessible via la « patte » d'administration.

Le port d'écoute (default : 3000) est modifiable via la variable `grafana.http_port` dans le fichier `environments/group_vars/all/cots_var.yml`.

```
# Vérifier l'API
http(s)://<adresse>:<grafana.http_port>/
# Vérifier le port d'écoute. Ce commande devrait afficher le port en écoute: <grafana.
↪http_port>
sudo ss -anp | grep grafana | grep LISTEN
```

- Exports

Il est possible d'exporter les dashboards créés.

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

7.2.7 Kibana

7.2.7.1 Présentation

Kibana est l'outil permettant de représenter de façon agrégée des données stockées dans ElasticSearch.

Afin de forcer la bonne version de Kibana, *VITAM* déploie un installeur-chapeau `vitam-kibana`.

Prudence : le composant kibana ne peut se connecter qu'à un cluster ElasticSearch ; pour superviser les clusters Elasticsearch de données et de log, il convient de définir des machines différentes (pour chaque kibana) durant l'installation.

VITAM injecte des *dashboards* durant l'installation.

7.2.7.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Le *playbook* d'installation effectue des actions de modification du fichier de configuration standard `/etc/kibana/kibana.yml`.

7.2.7.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start kibana`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop kibana`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Logs

Les logs applicatifs sont envoyés par rsyslog à la solution de centralisation des logs ; il est néanmoins possible d'en versionner une représentation par la commande :

```
journalctl --unit kibana
```

- Supervision du service

Kibana possède une IHM accessible via la « patte » d'administration :

http(s) ://<adresse> :5601/

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

7.2.8 Log server

7.2.8.1 Présentation

Ce composant représente en réalité l'ensemble des 3 composants suivants :

- Kibana, pour la présentation des dashboards de logs et de métriques ;
- Logstash, pour l'analyse et la centralisation des logs ;
- Curator, pour la maintenance des index elasticsearch de log.

Le présent chapitre ne s'intéressera qu'à logstash.

7.2.8.2 Configuration / fichiers utiles

L'ansiblerie se charge du paramétrage de ces composants.

7.2.8.3 Opérations

- Démarrage du service

En tant qu'utilisateur root :

Pré-requis : le cluster elasticsearch associé est déjà démarré.

```
systemctl start logstash
```

- Arrêt du service

En tant qu'utilisateur root :

```
systemctl stop logstash
```

Post-requis : le cluster elasticsearch-log associé est arrêté.

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

N/A

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

batch Curator, actuellement purgeant les données de plus de XX jours (selon ce qui a été défini dans l'inventaire de ansible) dans Elasticsearch de logs.

- cas des batches

Curator

7.2.9 MongoDB

Les composants `vitam-mongos`, `vitam-mongoc` et `vitam-mongod` sont des instances de la base de données MongoDB et constituent les briques distribuées d'un cluster MongoDB. La base de données est utilisée pour stocker les informations relatives aux archives hébergées dans Vitam.

Sous forme de cluster, elle est déployée en différentes instances :

- un cluster, nommé `mongodb-data`, stocke les métadonnées archivistiques (Unit, GOT) et les logbooks (LFC Unit, LFC GOT, Opérations), ainsi que les données de sécurité, de référence et les rapports (identity, masterdata et report).
- un cluster, par offre de stockage, nommé `mongodb-offer`, stocke les ordres d'écritures opérées sur les offres. Pour l'offre froide, cette base contient aussi les données d'emplacement de stockage dans l'offre (bandes magnétiques).

Les requêtes émises par les composants Vitam sont réceptionnées par le composant `mongos`, qui communique avec le composant `mongoc` afin de faire exécuter ces requêtes sur les composants `mongod`. La configuration et l'exploitation de ces 3 composants est détaillée ci-après.

Un paragraphe détaille les différentes topologies de déploiement ainsi que les recommendations pour augmenter la tolérance aux pannes du système. Un paragraphe détaille l'exploitation du cluster et notamment l'ajout de nouveaux shards.

7.2.9.1 Service vitam-mongos

7.2.9.1.1 Présentation

Le composant `vitam-mongos` est le point d'accès frontal à une base de données MongoDB de Vitam. Il exécute les requêtes envoyées par les composants Vitam, en communiquant avec le `config server`, représenté par l'ensemble des services `vitam-mongoc`, afin de déterminer les `Shards` (concept détaillé ci-après) sur lesquels exécuter la requête.

Pour assurer une haute disponibilité du service et répartir la charge des traitements du service `mongos` (majoritairement induite par une étape de regroupement des données (SHARD_MERGE)), il est recommandé de déployer ce service sur plusieurs machines. Les requêtes émises par les composants Vitam n'etant pas complexe et ne faisant pas intervenir une forte volumétrie, la charge estimée sur ce composant est faible. Aussi, les ressources machines allouées à ce service peuvent être réduites (par exemple 2 vCPU / 4Go RAM).

Note : les tests de performance du système ont montrés que la colocalisation des services vitam-mongos et vitam-mongoc sur la même machine n'a pas d'impact.

7.2.9.1.2 Configuration / fichiers utiles

Les fichiers de configuration du composant `vitam-mongos` sont accessibles dans le répertoire `/vitam/conf/mongos`.

7.2.9.1.2.1 Fichier `mongos.conf`

```
# mongos.conf

# for documentation of all options, see:
```

(suite sur la page suivante)

(suite de la page précédente)

```
#   http://docs.mongodb.org/manual/reference/configuration-options/


# where to write logging data.
systemLog:
  destination: file
  syslogFacility: local0
  logAppend: true
  logRotate: reopen
  path: {{ mongo_folder_log }}/mongos.log

# network interfaces
net:
  port: {{ mongodb.mongos_port }}
  bindIp: {{ ip_service }}{% if groups['hosts_dev_tools'] | length > 0 and ip_service␣
↪!= ip_admin %},{{ ip_admin }}{% endif %}

  unixDomainSocket:
    enabled: true
    pathPrefix: {{ mongo_tmp_path }}
    filePermissions: 0700

sharding:
  configDB: configsvr/{% for item in mongoc_list %}{{ hostvars[item]['ip_service'] }}:
↪{{ mongodb.mongoc_port }}{% if not loop.last %},{% endif %}{% endfor %}


# ansible managed security conf
```

7.2.9.1.2.2 Fichier `keyfile`

```
{{ mongodb[mongo_cluster_name].passphrase }}
```

7.2.9.1.2.3 Fichier de données

Ce composant n'utilise pas de fichiers de données.

7.2.9.1.3 Opérations

- Démarrage du service :

  En tant qu'utilisateur root :

```
systemctl start vitam-mongos
```

- Arrêt du service :

  En tant qu'utilisateur root :

```
systemctl stop vitam-mongos
```

- Consultation des logs du service :

En tant qu'utilisateur root :

```
journalctl -u vitam-mongos
```

- Accès au service pour réaliser un acte d'exploitation :

  Depuis une machine ou l'utilitaire mongo est installé et pour laquelle le flux réseau vers le service est ouvert. Le cas échéant, se connecter en ssh sur la machine pour utiliser l'utilitaire mongo en spécifiant le hostname de la machine (pas localhost) :

```
mongo --host <hostname> --port 27017 --username vitamdb-admin --password <password> --
↪authenticationDatabase admin
```

7.2.9.2 Service vitam-mongoc

7.2.9.2.1 Présentation

Le composant `vitam-mongoc` représente le service de configuration du cluster MongoDB déployé. Techniquement, il s'agit d'une instance du serveur de la base de données MongoDB (processus mongod) qui détient des bases et collections de configuration.

Prudence : Il est fortement recommandé de ne pas modifier ces collections, excepté lorsqu'un mode opératoire Vitam le mentionne ou à la demande explicite de l'éditeur de la base MongoDB.

Ce composant est déployé sous forme de `ReplicaSet`, c'est à dire un groupe de serveurs qui détiennent les mêmes données, afin de permettre une haute disponibilité du service ainsi qu'une réplication des données gérées.

A un instant donné, un serveur est identifié comme `Master` tandis que les autres serveurs sont identifiés comme `Slave`. Un serveur est appellé `membre` (d'un `ReplicaSet`). La réplication des données est réalisée, au travers d'une communication `Master/Slave`, sur les serveurs `Slave`, en reprenant les opérations (`OpLog`) réalisées sur le serveur `Master`. Le serveur `Master` est nommé membre `Primary` et les serveurs `Slave` sont nommés membres `Secondary`. Un mécanisme de vérification du statut d'un membre permet de gérer la haute disponibilité : si le membre `Primary` devient indisponible, un membre `Secondary` sera élu pour devenir `Primary`. Ces concepts sont utilisés dans la configuration du cluster.

Toutes les requêtes d'écriture sont réalisées uniquement sur le membre `Primary`, alors que les requêtes de lecture peuvent aussi être réalisées sur les membres `Secondary`. Pour gérer la consistance de la donnée entre les membres (concept `eventual consistency`), un mécanisme permet d'acquitter une écriture lorsque le membre `Primary` a pris en compte l'écriture (écris dans les logs d'opérations), et éventuellement lorsque l'écriture a été répliquée sur un, des ou l'ensemble des membres `Secondary`. Les requêtes Vitam sont executées avec ce mécanisme, configuré pour qu'une majorité de membre retourne un acquittement (voir `Write Concern` valué à `Majority`). Le même mécanisme existe pour les lectures, à savoir, qu'une lecture peut éventuellement être vérifiée sur plusieurs membres afin d'assurer la consistance de la lecture. La encore, les requêtes Vitam sont executées avec ce mécasnime, configuré pour qu'une majorité de membres confirme disposer de la même donnée (voir `Read Concern` valué à `Majority`). Les paramètres `Write Concern` et `Read Concern` positionnés à `Majority` permettent de configurer le niveau de consistance d'un `ReplicaSet` en jouant uniquement sur le nombre de membres déployés dans un `ReplicaSet`. Ces paramètres ne sont donc pas modifiables vis à vis des fichiers de paramétrage Vitam.

Le déploiement standard d'un `ReplicaSet` en production, est composé de 3 membres. Cet exemple de déploiement est nommé `PSS`, pour illustrer la composition du `ReplicaSet` avec des membres `Primary Secondary Secondary`. Avec cette topologie, les mécanismes pour garantir la consistance de la donnée (configurés avec la valeur `Majority`) permettent de s'assurer que la donnée est pris en compte par 2 membres (sur les 3).

Note : cette configuration permet une tolérance aux pannes satisfaisante. Toutefois, il est possible d'améliorer cette tolérance en ajoutant des membres. Consulter la documentation officielle pour plus de détail : `https://docs.mongodb.com/manual/core/replica-set-architectures/`.

La charge estimée du service `mongoc` est faible, aussi les ressources machines allouées peuvent être réduites (par exemple, 2 vCPU / 4 Go RAM).

Note : les tests de performance du système ont montrés que la colocalisation des services vitam-mongos et vitam-mongoc sur la même machine n'a pas d'impact.

7.2.9.2.2 Configuration / fichiers utiles

Les fichiers de configuration du composant `vitam-mongoc` sont accessibles dans le répertoire `/vitam/conf/mongoc`.

7.2.9.2.2.1 Fichier `mongoc.conf`

```
# mongoc.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/


# where to write logging data.
systemLog:
  destination: file
  syslogFacility: local0
  logAppend: true
  logRotate: reopen
  path: {{ mongo_folder_log }}/mongoc.log


# Where and how to store data.
storage:
  dbPath: {{ mongo_db_path }}
  directoryPerDB: true


# network interfaces
net:
  port: {{ mongodb.mongoc_port }}
  bindIp: {{ ip_service }}{% if groups['hosts_dev_tools'] | length > 0 and ip_service␣
↪!= ip_admin %},{{ ip_admin }}{% endif %}

  unixDomainSocket:
    enabled: true
    pathPrefix: {{ mongo_tmp_path }}
    filePermissions: 0700


# operationProfiling:
```

(suite sur la page suivante)

(suite de la page précédente)

```
replication:
  replSetName: configsvr # name of the replica set
  enableMajorityReadConcern: true

sharding:
  clusterRole: configsvr # role du shard


# ansible managed security conf
```

7.2.9.2.2.2 Fichier `keyfile`

```
{{ mongodb[mongo_cluster_name].passphrase }}
```

7.2.9.2.2.3 Fichier de données

Ce service utilise des fichiers de données localisés dans le répertoire `/vitam/data/mongoc/db`

7.2.9.2.3 Opérations

- Démarrage du service :

  En tant qu'utilisateur root :

```
systemctl start vitam-mongoc
```

- Arrêt du service :

  En tant qu'utilisateur root :

```
systemctl stop vitam-mongoc
```

- Consultation des logs :

  En tant qu'utilisateur root :

```
journalctl -u vitam-mongoc
```

- Accès au service pour réaliser un acte d'exploitation :

  Depuis une machine ou l'utilitaire mongo est installé et pour laquelle le flux réseau vers le service est ouvert. Le cas échéant, se connecter en ssh sur la machine pour utiliser l'utilitaire mongo en spécifiant le hostname de la machine (pas localhost) :

```
mongo --host <hostname> --port 27018 --username vitamdb-localadmin --password
↪<password> --authenticationDatabase admin
```

7.2.9.3 Service vitam-mongod

7.2.9.3.1 Présentation

Le composant `vitam-mongod` représente le service de données du cluster MongoDB déployé. Techniquement, il s'agit d'une instance du serveur de la base de données MongoDB (processus mongod) qui détient les bases et collections Vitam. Des bases et collections de configuration sont aussi présentes pour le fonctionnement de la solution MongoDB.

Prudence : Il est fortement recommandé de ne pas modifier ces collections, excepté lorsqu'un mode opératoire Vitam le mentionne ou à la demande explicite de l'éditeur de la base MongoDB.

Ce service est déployé, comme le service `vitam-mongoc`, sous forme de `ReplicaSet` (voir le paragraphe précédent pour plus d'explication).

Pour gérer de grosses volumétries, les services `mongod` sont déployés de manière à pouvoir gérer un sous ensemble de données, afin de répartir la charge d'utilisation. Ce concept est nommé `Sharding` et le `ReplicaSet` représente alors un sous-ensemble des données gérées par le cluster (pour les collections dites `shardées`). Ce sous-ensemble est nommé `Shard` et ce concept est utilisé dans la configuration du cluster. Une collection non `shardée` sera disponible sur un unique `Shard`.

Dans Vitam, les collections des bases metadata et logbook de mongodb-data (exceptée la collection Operation) sont `shardées`, car la dimension de leur volume est estimée importante (plusieurs millions ou milliards). Les collections des bases identity, masterdata et report ne sont pas `shardées`.

Dans le cadre d'une offre « froide », la collection TapeAccessRequestReferential de la base offer de mongodb-offer est également `shardée`. Les autres collections ne sont pas `shardées`.

Dans le cas de mongodb-data, les ressources machines allouées à ce service doivent être relativement importantes en fonction de :

- la volumétrie et du débit des versements d'archives dans le système :

  Plus le système est sollicité, notamment dans le cas du versement, plus les clusters MongoDB, et donc les services `vitam-mongod`, sont sollicités. Les composants Vitam qui éméttent les requêtes vers le cluster `mongodb-data` sont `metadata`, `logbook` et `functional-administration`. Vers le cluster `mongodb-offer`, seul le composant `offer` émet les requêtes.

  Le nombre de requêtes émises vers un cluster est dépendant du nombre de composant `vitam-worker` déployé dans le système ainsi que du paramètre qui spécifie le nombre de worker exécutant une tâche Vitam au sein d'un composant (Thread dans la jvm).

  En fonction de la distribution des tâches, opérée dans le composant `vitam-processing`, du nombre de versement en parallèle et du nombre d'unité archivistique par SIP, un certain nombre de worker vont émettre des requêtes en parallèle.

  Différentes optimisations ont été réalisées dans les tâches Vitam, pour notamment, diminuer le nombre de requêtes vers les clusters MongoDB, en privilégiant une requête en mode `bulk`. Aussi, avec ces optimisations, les services `vitam-mongod` sont bien plus sollicités. Toutefois, si le nombre d'unité archivistique positionnées dans les SIP est trop faible, le nombre de requête par bulk est réduit. Les recommendations des paramètres et de l'utilisation du système de manière efficiente sont détaillées dans un autre document.

  Le nombre de vCPU disponible pour le service `vitam-mongod` influe sur les performances d'exécution en parallèle des requêtes MongoDB. Dans le cas des versements, la sollicitation des clusters MongoDB est essentiellement liée à des requêtes d'écriture. Dans ce mode, l'utilisation des index MongoDB ainsi que la lecture des documents en base sont limitées. Et donc dans ce contexte, la quantité de RAM disponible pour les index et pour les caches de document (`workingSet` MongoDB) ne doit pas être nécessairement importante.

- des fonctionnalités Vitam utilisées :

Le versement d'archive est une fonctionnalité importante du système pour laquelle la sollicitation des clusters MongoDB est majoritairement liée à des requêtes d'écriture. D'autres fonctionnalités Vitam, comme la consultation d'archives ou la réalisation de `DIP`, la sécurisation des opérations et données ingérées par le système, ou tout autre traitement en masse qui nécessite un accès aux documents, sollicitent le cluster `mongodb-data` avec des requêtes de lecture.

Dans Vitam, un paradigme d'architecture est posé et est respecté par l'ensemble des composants (sauf exceptions cités ci-après) : toute requête sur les données qui nécessite un filtre sur les métadonnées, est executée dans le moteur de recherche ElasticSearch, afin de récuperer une liste d'identifiants de documents à requêter dans MongoDB. Ce pattern permet en outre de limiter le nombre d'index à créer dans MongoDB.

Aussi, dans le cas des accès aux documents, les performances seront réduites lorsque les identifiants des documents requêtés ne seront plus indexés en RAM (cas ou la quantité de RAM nécessaire pour contenir l'ensemble des index en mémoire est insuffisante) et surtout lorsque le document ne sera plus disponible dans le cache. Si les données ne sont plus disponibles en mémoire, le moteur MongoDB doit procéder à des lectures sur disque en remplaçant les données les plus anciennes par les nouvellement lues. Ce mécasnime (`evictions page`) est utilisé pour les index et les documents.

Le cas particulier des sécurisations, sollicitent les clusters MongoDB en lecture, sans passer par le moteur de recherche (pour des raisons de sécurité et d'intégrité de la donnée). Elles utilisent un index supplémentaire directement dans MongoDB. Aussi, dans ce contexte, pour assurer de bonnes performances à ces opérations, il faut veiller à dimensionner le cache des documents suffisament important pour conserver les documents, et index liés aux documents, qui ont été ingérés par le système et qui n'ont pas encore été sécurisés.

Par défaut, les sécurisations des cycles de vie des métadonnées archivistiques et des opérations Vitam sont executées toutes les 2 ou 3 heures. Durant cette période, un versement d'un million d'AU nécessitera l'utilisation d'un million de clé (identifiant Mongo et identifiant de sharding) dans les index, soit une centaine de Mo, et nécessitera l'utilisation d'un million de documents dans le cache, soit 5,7 Go.

La quantité de RAM disponible pour le service `vitam-mongod` influe sur les performances d'exécution des requêtes MongoDB. Dans le contexte des fonctionnalités citées, la quantité de RAM doit être importante. La recommendation Vitam est d'allouer la quantité de RAM nécessaire pour traiter au minimum les sécurisations des documents versées dans les dernières heures, afin de ne pas ralentir le système avec ces opérations. Par exemple, pour des versements à 450.000 AU/h, les sécurisations LFC AU et GOT, et Opérations, nécessiteront environ 12 Go de RAM (quantité répartie sur l'ensemble des `Shards`).

Dans le cas de mongodb-offer, les ressources machines allouées à ce service sont modérément élevés, excepté le cas des offres froides où l'utilisation des ressources peut être plus conséquente, en fonction des traitements en cours.

Note : Par défaut, 50% de la RAM disponible est allouée au processus mongod pour gérer les index et caches de document, de manière à laisser disponible en cache OS les données lues sur disque (blocs des fichiers de données MongoDB).

Avertissement : Les performances du cluster MongoDB, en écriture ou lecture, restent dépendantes de la performance des disques. Il est recommandé de privilégier une infrastructure avec des disques physiques attachés aux machines qui déploient les services `mongod`. A défaut d'une telle configuration, il est recommandé d'octroyer la meilleure qualité de service de l'offre disque utilisée (débit ou priorité d'accès).

7.2.9.3.2 Configuration / fichiers utiles

Les fichiers de configuration du composant `vitam-mongod` sont accessibles dans le répertoire `/vitam/conf/mongod`.

7.2.9.3.2.1 Fichier `mongod.conf`

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/


# where to write logging data.
systemLog:
  destination: file
  syslogFacility: local0
  logAppend: true
  logRotate: reopen
  path: {{ mongo_folder_log }}/mongod.log


# Where and how to store data.
storage:
  dbPath: {{ mongo_db_path }}
  directoryPerDB: true

{% if mongod_memory is defined and mongod_memory != '' %}
  wiredTiger:
    engineConfig:
        cacheSizeGB: {{ mongod_memory }}
{% endif %}


# network interfaces
net:
  port: {{ mongodb.mongod_port }}
  bindIp: {{ ip_service }}{% if groups['hosts_dev_tools'] | length > 0 and ip_service␣
↪!= ip_admin %},{{ ip_admin }}{% endif %}

  unixDomainSocket:
    enabled: true
    pathPrefix: {{ mongo_tmp_path }}
    filePermissions: 0700

# operationProfiling:
replication:
  replSetName: shard{{ mongo_shard_id }} # name of the replica set
  enableMajorityReadConcern: true

sharding:
  clusterRole: shardsvr # role du shard


# ansible managed security conf
```

7.2.9.3.2.2 Fichier `keyfile`

```
{{ mongodb[mongo_cluster_name].passphrase }}
```

7.2.9.3.2.3 Fichier de données

Ce service utilise des fichiers de données localisés dans le répertoire `/vitam/data/mongod/db`

7.2.9.3.3 Opérations

- Démarrage du service :

  En tant qu'utilisateur root :

```
systemctl start vitam-mongod
```

- Arrêt du service :

  En tant qu'utilisateur root :

```
systemctl stop vitam-mongod
```

- Consultation des logs :

  En tant qu'utilisateur root :

```
journalctl -u vitam-mongod
```

- Accès au service pour réaliser un acte d'exploitation :

  Depuis une machine ou l'utilitaire mongo est installé et pour laquelle le flux réseau vers le service est ouvert. Le cas échéant, se connecter en ssh sur la machine pour utiliser l'utilitaire mongo en spécifiant le hostname de la machine (pas localhost) :

```
mongo --host <hostname> --port 27019 --username vitamdb-localadmin --password
↪<password> --authenticationDatabase admin
```

7.2.9.4 Topologies de déploiement et tolérance aux pannes

7.2.9.4.1 Présentation

Les clusters MongoDB déployés dans la solution Vitam sont par défaut, et obligatoirement, configurés pour supporter le sharding des bases de données metadata, logbook et offer. Aussi, les composants `vitam-mongos`, `vitam-mongoc` et `vitam-mongod` sont configurés pour fonctionner ensemble. Le déploiement automatique d'une mono-instance MongoDB (un seul et unique serveur mongod) n'est pas supporté.

Les topologies de déploiement des services `mongos`, ainsi que des `ReplicaSet` des services `mongoc` et `mongod`, restent à la main de l'administrateur technique Vitam. Les choix sont réalisés lors de l'installation de la solution Vitam, mais peuvent être modifiés à posteriori (avec potentiellement des temps de migration de données à prévoir).

Le nombre de composants `vitam-mongos` peut varier au cours de l'utilisation du système sans contrainte particulière. Le nombre de `Shards` déployés, lui par contre, s'il varie, impliquera des migrations de données, réalisées en tâche de fond. Les choix opérés par les concepteurs de la base de données MongoDB ne peuvent pas être modifiés mais permettent une migration sans altérer les performances du système. Un paragraphe entier détaille ce scénario ainsi que les contraintes et temps nécessaires à la réalisation complète de l'opération.

Le nombre de composants par `ReplicaSet` des services `mongoc` et `mongod` reste aussi un choix à la main de l'administrateur technique et peut aussi varier dans le temps. Il est aussi assujetti à une opération plus ou moins longue

de synchronisation des données. Il est important de respecter la contrainte MongoDB concernant le choix du nombre de membres d'un `ReplicaSet`. Pour des raisons liées à l'algorithme de l'élection d'un membre primaire, parmis les membres d'un `ReplicaSet` (cf. documentation officielle [17]), ce nombre doit être impair.

La tolérance aux pannes est directement liée au choix du nombre de membres déployés par `ReplicaSet` et particulièrement à la redondance physique des composants et équipements réseaux qui assurent la communication entre eux. Aussi, le coût de déploiement, en ressources matérielles, est extrêmement lié au niveau de tolérance selectionné. Étant donné, la consommation importante de ressources matérielles par les services MongoDB (notamment en RAM), une option est laissée à la main de l'administrateur technique, pour configurer un `ReplicaSet` dans un mode particulier, nommé `PSSmin`, sans modifier le niveau de tolérance aux pannes. Ce point est détaillé ci-après.

Dans Vitam, la gestion du risque de la perte de données est centralisée au niveau des offres. Si un incident intervient sur le cluster `mongodb-data` provoquant une perte totale du service, un processus de reconstruction est prévu. Toutefois, il est important de noter que ce processus nécessite un temps relativement important, fonction de la volumétrie des données dans le système. Un autre processus de reconstruction du cluster `mongodb-offer` est prévu mais uniquement pour les offres chaudes. Dans le cas d'une offre froide, la reconstruction du cluster est réalisée par l'opération de restoration de la dernière sauvegarde réalisée sur le cluster.

7.2.9.4.2 Déploiement d'un cluster de développement

En environnement de développement, la tolérance aux pannes, ainsi que la performance du système testé, ne sont pas attendues. Il est donc possible de déployer un cluster minimaliste, constitué d'un composant par type et associé à des ressources machines minimales. La configuration minimum pour un service mongoDB est de 1 vCPU et 512 Mo de RAM. Il est possible de colocaliser les trois services `vitam-mongos`, `vitam-mongoc` et `vitam-mongod` sur la même machine en prévoyant 1 vCPU et 2 Go de RAM.

Pour réaliser le déploiement du cluster `mongodb-data` sur une seule machine, l'inventaire ansible doit référencer la même machine pour chacun des groupes `hosts-mongos-data`, `hosts-mongoc-data` et `hosts-mongod-data`. Bien qu'il n'y ait qu'un seul `Shard` et qu'un membre par `ReplicaSet`, le paramètres `mongo_rs_bootstrap` et `mongo_shard_id` sont attendus. Ces paramètres sont détaillés dans le paragraphe qui suit.

Exemple

```
[hosts-mongodb-data:children]
hosts-mongos-data
hosts-mongoc-data
hosts-mongod-data

[hosts-mongos-data]
host1.vm    mongo_cluster_name=mongodb-data

[hosts-mongoc-data]
host1.vm    mongo_cluster_name=mongodb-data                    mongo_rs_bootstrap=true

[hosts-mongod-data]
host1.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=0  mongo_rs_bootstrap=true
```

Le déploiement d'un cluster `mongodb-offer` suit les mêmes règles que l'exemple illustré ci-dessus (les groupes ansible ne sont pas les mêmes).

https://docs.mongodb.com/manual/core/replica-set-elections/

7.2.9.4.3 Déploiement d'un cluster de production

En environnement de production, la tolérance aux pannes doit être prévue. Avec trois membres par `ReplicaSet`, le niveau de tolérance aux pannes est satisfaisant : si le noeud primaire n'est plus disponible, le système provoque une élection au sein du `ReplicaSet`, permettant alors à un noeud secondaire de devenir noeud primaire. Dans cette situation, si le nouveau noeud primaire devient indisponible, il ne restera plus qu'un seul noeud disponible pour disposer d'un noeud primaire et la situation devient alors critique (par rapport à la disponibilité de la donnée). Pour autant, dans le cas ou un noeud primaire et un seul noeud secondaire sont disponibles, la consistance de la donnée n'est potentiellement plus assurée, puisque les mécanismes configurés avec la valeur `Majority` ne permettent plus une vérification de la réplication : la majorité correspond à un membre, et donc uniquement le membre `Primary` acquitera l'écriture. Aussi, dans ce cas, si le membre primaire devient indisponible après une écriture et avant la réplication, lorsque le dernier membre disponible devient primaire, l'écriture n'a pas été répliquée. Dans le cas nominal ou les deux membres secondaires étaient disponibles, c'est le membre qui a réalisé l'acquitement de la replication qui devient primaire (dans les faits, le membre secondaire le plus à jour vis à vis de la réplication) et alors l'écriture a bien été prise en compte.

Une tolérance aux pannes plus importante peut être mise en place en déployant un quatrième membre. Et afin de respecter un nombre impair de membres déployés, il est possible de déployer un cinquième membre qui ne consomme pas autant de ressources matérielles qu'un membre actif et dont la responsabilité n'est d'intervenir que dans l'élection d'un noeud primaire. Ce type de membre est nommé membre `Arbiter`. La topologie de déploiement du `ReplicaSet` est alors nommée `PSSSA`, pour illustrer le déploiement d'un membre `Primary`, de 3 membres `Secondary` et d'un membre `Arbiter`.

L'indisponibilité d'un noeud peut être liée à différentes causes : un problème matériel provoquant un `crash` du processus, mais pour lequel le service sera redémarré automatiquement suite à l'incident (faute de mémoire, faute d'accès disque durant l'écriture, ...) ; ou un problème matériel important provoquant la perte de la machine physique et pour lequel le service ne pourra pas redémarrer (provisionning virtuel, coupure éléctrique, ...). Aussi, il est important de déployer les membres d'un `replicaSet` dans des zones matérielles différentes. Dans le cas, d'un environnement virtuel opéré physiquement par différentes machines, il est opportun de spécifier un provisionning physique afin d'assurer une répartition physique des machines.

Pour augmenter véritablement la tolérance aux pannes, en plus d'augmenter le nombre de membre d'un `ReplicaSet` correctement réparti physiquement, il est recommandé de déployer un environnement de production constitué de deux salles physiquement indépendantes (alimentation éléctrique, droits d'accès, ...) et pour lesquelles les communications réseaux sont autorisées et resteront performantes. Il s'agit bien la, de discerner le cas du site secondaire prévu dans le système Vitam, notamment pour gérer la criticité de la perte d'une offre de stockage. Dans ce scénario, il est recommandé que ce second site soit géographiquement distant du premier, de plusieurs dizaines de kilomètres. Aussi, les performances réseaux entre les deux sites pourraient être insuffisante au regard des performances attendues dans le cluster MongoDB.

Pour réaliser le déploiement du cluster `mongodb-data`, en mode `PSS`, composé de 3 machines hebergeant les services `vitam-mongos` et `vitam-mongoc` et de 6 machines hebergeant le service `vitam-mongod` (pour disposer de 2 `Shards`), l'inventaire ansible doit référencer les 3 premières machines pour les groupes `hosts-mongos-data` et `hosts-mongoc-data` et les 6 autres machines pour le groupe `hosts-mongod-data`. Le paramètre `mongo_shard_id` spécifie le regroupement des machines dans le même `ReplicaSet` et identifie un numéro de `Shard`. Le paramètre `mongo_rs_bootstrap` spécifie la machine sur laquelle l'initialisation du `ReplicaSet` sera réalisée (voir command rs.initiate()). Les autres machines du `ReplicaSet` y seront alors ajoutées (voir commande rs.add()).

Exemple

```
[hosts-mongodb-data:children]
hosts-mongos-data
hosts-mongoc-data
hosts-mongod-data
```

(suite sur la page suivante)

(suite de la page précédente)

```
[hosts-mongos-data]
host1.vm    mongo_cluster_name=mongodb-data
host2.vm    mongo_cluster_name=mongodb-data
host3.vm    mongo_cluster_name=mongodb-data                    mongo_rs_bootstrap=true

[hosts-mongoc-data]
host1.vm    mongo_cluster_name=mongodb-data
host2.vm    mongo_cluster_name=mongodb-data
host3.vm    mongo_cluster_name=mongodb-data                    mongo_rs_bootstrap=true

[hosts-mongod-data]
host4.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=0
host5.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=0
host6.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=0  mongo_rs_bootstrap=true
host7.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=1
host8.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=1
host9.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=1  mongo_rs_bootstrap=true
```

Le déploiement d'un cluster `mongodb-offer` en mode `PSS` suit les mêmes règles que l'exemple illustré ci-dessus (les groupes ansible ne sont pas les mêmes).

7.2.9.4.4 Déploiement d'un cluster de production avec réduction de la RAM

Le déploiement d'un cluster mongoDB, pour gérer de forte volumétrie avec les performances initiales estimées et une tolérance aux pannes satisfaisante, va s'avérer gourmand en ressource matérielle.

Par exemple, pour gérer un milliard d'AU dans le système, avec une performance de versement estimée à 450.000 AU par heure, et une tolérance aux pannes fondée sur l'utilisation de 3 membres par `ReplicaSet`, il est estimé le besoin de 960 Go de RAM (donnée extrapolée à partir d'un environnement chargé à 200 millions d'AU avec 4 Shards constitués de machines 4 vCPU / 16 Go RAM).

Si le coût d'une telle infrastructure ne peut pas être supportée, il est possible de diminuer l'allocation RAM sur un membre par `ReplicaSet`. Le principe étant de conserver une haute disponibilité (et une consistance des données), à performance constante, avec deux membres dont les ressources physiques le permettent. Le troisième membre, dont les ressources ont été abaissées au minimum, est configuré pour ne pas être prioritaire à l'élection d'un noeud primaire. Dans le cas d'une indisponibilité des deux premiers membres, le service reste assuré (avec niveau de tolérance aux pannes toujours équivalent à 3 membres) mais dont les performances initiales ne sont plus assurées.

Pour réaliser le déploiement du cluster `mongodb-data`, en mode `PSSmin`, composé de 3 machines hebergeant les services `vitam-mongos` et `vitam-mongoc` et de 6 machines hebergeant le service `vitam-mongod` (pour disposer de 2 `Shards`), l'inventaire ansible doit référencer les 3 premières machines pour les groupes `hosts-mongos-data` et `hosts-mongoc-data` et les 6 autres machines pour le groupe `hosts-mongod-data`. Parmis les 3 machines de chaque `replicaSet`, la machine qui est déployée avec les ressources matérielles minimum, doit être suivi du paramètre `is_small=true`.

Exemple

```
[hosts-mongodb-data:children]
hosts-mongos-data
hosts-mongoc-data
hosts-mongod-data

[hosts-mongos-data]
host1.vm    mongo_cluster_name=mongodb-data
host2.vm    mongo_cluster_name=mongodb-data
```

(suite sur la page suivante)

(suite de la page précédente)

```
host3.vm    mongo_cluster_name=mongodb-data                                    mongo_
↪rs_bootstrap=true

[hosts-mongoc-data]
host1.vm    mongo_cluster_name=mongodb-data
host2.vm    mongo_cluster_name=mongodb-data
host3.vm    mongo_cluster_name=mongodb-data                                    mongo_
↪rs_bootstrap=true

[hosts-mongod-data]
host4.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=0
host5.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=0  is_small=true
host6.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=0                  mongo_
↪rs_bootstrap=true
host7.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=1
host8.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=1  is_small=true
host9.vm    mongo_cluster_name=mongodb-data  mongo_shard_id=1                  mongo_
↪rs_bootstrap=true
```

Le déploiement d'un cluster `mongodb-offer` en mode `PSSmin` suit les mêmes règles que l'exemple illustré ci-dessus (les groupes ansible ne sont pas les mêmes).

7.2.9.5 Exploitation d'un cluster MongoDB

7.2.9.5.1 Extension du cluster : ajouter un ou n `Shards`

Lorsque la volumétrie des données gérées par un seul `Shard` devient trop importante, il faut ajouter de nouveaux `Shards` dans le cluster. Des lors que les nouveaux `Shards` sont opérationnels, une opération de migration des données (voir `balancing`) est réalisée en tâche de fond afin d'équilibrer l'ensemble des `Shards`.

Les éléments déplacés entre `Shard` sont des regroupements de données par collection. Ce regroupement est nommé `Chunk` et le composant MongoDB qui réalise ce déplacement est nommé `balancer`.

Le mécanisme interne du `balancer` n'est pas paramétrable. Une opération de transfert de `Chunk` est réalisée exclusivement entre un `Shard` et un autre `Shard`. Si le cluster contient 4 `Shards`, 2 opérations, au maximum, pourront être réalisées en parallèle à un instant donné.

Note : il est recommandé de favoriser un grand nombre de `Shards` (avec un nombre pair), déployés sur des "petites" machines, afin de favoriser le re-équilibrages des `Shards`.

Note : Dans Vitam, les clés de `Sharding` implémentées permettent une répartition uniforme des données. En régime de croisière, les `Shards` n'ont donc pas besoin d'être équilibrés et l'opération de `balancing` ne devrait pas logué une activité.

Pour ajouter un nouveau `Shard` au cluster, il faut exécuter les opérations suivantes :

- Créer les machines qui seront utilisées comme membre des nouveaux shards.
- Ajouter ces machines dans le fichier d'inventaire ansible.
- se connecter à un service `vitam-mongos` :

  Depuis une machine ou l'utilitaire mongo est installé et pour laquelle le flux réseau vers le service est ouvert. Le cas échéant, se connecter en ssh sur la machine pour utiliser l'utilitaire mongo en spécifiant le hostname de la machine (pas localhost) :

```
mongo --host <hostname> --port 27017 --username vitamdb-admin --password <password> --
↪authenticationDatabase admin
```

- Vérifier l'état du sharding dans le cluster en tappant la commande suivante :

```
sh.status()
```

La commande retourne un ensemble d'informations dont 2 sont importantes pour cette opération d'exploitation. La première détaille la composition des shards opérationnels, dont voici un exemple :

```
shards:
    {  "_id" : "shard0",  "host" : "shard0/<ipMember1>:27019,<ipMember2>:27019,
↪<ipMember3>:27019",  "state" : 1 }
    {  "_id" : "shard1",  "host" : "shard1/<ipMember1>:27019,<ipMember2>:27019,
↪<ipMember3>:27019",  "state" : 1 }
```

La seconde détaille les activités de balancing des `Chunks`, dont voici un exemple :

```
balancer:
      Currently enabled:  yes
      Currently running:  no
      Failed balancer rounds in last 5 attempts:  0
      Last reported error:
      Time of Reported error:  Tue Jul 30 2019 09:05:45 GMT+0000 (UTC)
      Migration Results for the last 24 hours:
              No recent migrations
```

Dans ces informations, on constate que le service de `balancing` est actif et qu'aucune migration n'est en cours. De plus, aucune migration n'a été réalisée au cours des dernières 24 heures.

- Ajouter les shards en récupérant les adresses Ip de chacun des membres, en conservant le regroupement qui a été configuré dans l'ansiblerie Vitam, et en executant la commande suivante pour chaque shard :

```
sh.addShard("shard2/<ipMember1>:27019,<ipMember2>:27019,<ipMember3>:27019")
```

- Vérifier que le balancing a démarré en récupérant l'état du sharding via la commande :

```
sh.status()
```

Voici un exemple d'information que l'on peut récupérer :

```
balancer:
      Currently enabled:  yes
      Currently running:  yes
      Collections with active migrations:
              logbook.LogbookLifeCycleObjectGroup started at Wed Jul 31 2019 12:43:19␣
↪GMT+0000 (UTC)
      Failed balancer rounds in last 5 attempts:  0
      Migration Results for the last 24 hours:
              22 : Success
```

Note : cette opération a été testée dans l'environnement de performance Vitam, qui contenait environ 200 millions d'AU. Le temps de migration des `Chunks` depuis les 2 `Shards` existants vers les 2 nouveaux `Shards` a été d'une trentaine de jours.

7.2.10 Prometheus

Les composants ci-dessous sont des services de la solution de supervision Prometheus.

- Le composant `vitam-prometheus` dispose d'une base de donnée dans laquelle les métriques à collecter seront stockées.
- Le compodant `vitam-alertmanager` gère tout ce qui est alerting.
- Le composant `vitam-node-exporter` permet d'exposer des métriques liées au matériel et au noyau du système.

Les composants `vitam-prometheus` et `vitam-alertmanager` sont optionnels. Une entitée utilisant la solution VITAM et disposant déjà de sa propre solution de supervision peut récupérer les métriques en utilisant les API exposées par VITAM.

Pour se faire, un playbook est proposé pour générer la configuration Prometheus depuis l'environnement de la solution VITAM.

Un paragraphe dans la documentation du composant Prometheus détaille comment générer cette configuration, l'enrichir des règles d'alerte et l'intégrer dans un serveur Prometheus déjà existant.

7.2.10.1 Service vitam-prometheus

7.2.10.1.1 Présentation

Le composant `vitam-prometheus` permet de stocker et visualiser les métriques techniques et métier collectées depuis les différents composants de la solution VITAM. Il permet aussi d'explorer les données en appliquant différentes fonctions statistiques.

La solution VITAM, par défaut, déploie une seule instance de ce service. Veuillez vous référer à la documentation officielle prometheus pour pouvoir scaller le déploiement de cet outil.

7.2.10.1.1.1 Générer le fichier de configuration prometheus.yml

Dans le cas ou vous disposez d'un serveur prometheus, vous n'avez qu'à générer la configuration `'prometheus.yml` depuis l'inventaire de l'environnement de la solution VITAM.

Pour générer uniquement la configuration il faut exécuter la ligne de commande suivante :

Depuis le serveur ansible, aller au dossier `path_to/vitam/deploiement/`

```
# Spécifier le répertoire de sortie dans le fichier cots_var.yml {{ prometheus.
↪prometheus_config_file_target_directory: path_dir_output }}
ansible-playbook ansible-vitam-extra/prometheus.yml -i environments/hosts.
↪<environnement> --ask-vault-pass --tags gen_prometheus_config
```

Le fichier de configuration sera généré dans le répertoire de sortie avec le nom `prometheus.yml`. Il suffit de récupérer les parties nécessaires, comme par exemple `scrape_configs` et les intégrer dans la configuration du serveur prometheus déjà existant.

Avertissement : Les flux réseaux entre le serveur prometheus existant et les différents machines hébergeant le solution VITAM doivent être ouverts sur la patte d'administration.

7.2.10.1.1.2 Intégrer de nouvelle règles d'alertes

Déposez les fichiers des règles dans le dossier : `../../../../../../deployment/ansible-vitam-extra/roles/prometheus-server/rules/` Ensuite lancez la commande suivante :

```
ansible-playbook ansible-vitam-extra/prometheus.yml -i environments/hosts.
↪<environnement> --ask-vault-pass
```

7.2.10.1.1.3 Exemple de fichiers de règles

- Règle sur le disque

```
groups:
  - name: system_disk
    rules:
      - alert: OutOfDiskSpace
        expr: (node_filesystem_avail_bytes{mountpoint="/rootfs"} * 100) / node_
↪filesystem_size_bytes{mountpoint="/rootfs"} < 15
        for: 10m
        labels:
          severity: warning
        annotations:
          description: |-
            Disk is almost full (< 10% left)
              VALUE = {{ $value }}
              LABELS: {{ $labels }}
          summary: Out of disk space (instance {{ $labels.instance }})

      - alert: OutOfDiskSpace
        expr: (node_filesystem_avail_bytes{mountpoint="/rootfs"} * 100) / node_
↪filesystem_size_bytes{mountpoint="/rootfs"} < 5
        for: 5m
        labels:
          severity: critical
        annotations:
          description: |-
            Disk is almost full (< 10% left)
              VALUE = {{ $value }}
              LABELS: {{ $labels }}
          summary: Out of disk space (instance {{ $labels.instance }})

  - name: vitam_disk
    rules:
      - alert: OutOfDiskSpace
        expr: (node_filesystem_avail_bytes{mountpoint="/vitam"} * 100) / node_
↪filesystem_size_bytes{mountpoint="/vitam"} < 20
        for: 10m
        labels:
          severity: warning
        annotations:
```

(suite sur la page suivante)

(suite de la page précédente)

```
          description: |-
            Disk is almost full (< 20% left)
              VALUE = {{ $value }}
              LABELS: {{ $labels }}
          summary: Out of disk space (instance {{ $labels.instance }})

      - alert: OutOfDiskSpace
        expr: (node_filesystem_avail_bytes{mountpoint="/vitam"} * 100) / node_
↪filesystem_size_bytes{mountpoint="/vitam"} < 5
        for: 5m
        labels:
          severity: critical
        annotations:
          description: |-
            Disk is almost full (< 5% left)
              VALUE = {{ $value }}
              LABELS: {{ $labels }}
          summary: Out of disk space (instance {{ $labels.instance }})
```

- Règle sur le host

```
groups:
- name: host
  rules:
  - alert: high_cpu_load
    expr: node_load1 > 1.5
    for: 30s
    labels:
      severity: warning
    annotations:
      summary: "Server under high load"
      description: "Host is under high load, the avg load 1m is at {{ $value}}.␣
↪Reported by instance {{ $labels.instance }} of job {{ $labels.job }}."

  - alert: high_memory_load
    expr: (sum(node_memory_MemTotal) - sum(node_memory_MemFree + node_memory_Buffers␣
↪+ node_memory_Cached) ) / sum(node_memory_MemTotal) * 100 > 85
    for: 30s
    labels:
      severity: warning
    annotations:
      summary: "Server memory is almost full"
      description: "Host memory usage is {{ humanize $value}}%. Reported by instance {
↪{ $labels.instance }} of job {{ $labels.job }}."

  - alert: high_storage_load
    expr: (node_filesystem_size{fstype="aufs"} - node_filesystem_free{fstype="aufs"})␣
↪/ node_filesystem_size{fstype="aufs"}  * 100 > 85
    for: 30s
    labels:
      severity: warning
    annotations:
      summary: "Server storage is almost full"
      description: "Host storage usage is {{ humanize $value}}%. Reported by instance
↪{{ $labels.instance }} of job {{ $labels.job }}."
```

- Règle sur l'utilisation de la mémoire

```
groups:
  - name: system_disk
    rules:
      - alert: MemoryUsage
        expr: (100 - ((node_memory_MemAvailable_bytes * 100) / node_memory_MemTotal_
↪bytes)) > 85
        for: 10m
        labels:
          severity: warning
        annotations:
          description: RAM of {{$labels.instance}} has been too used for more than 10␣
↪minutes
          summary: Instance {{$labels.instance}} start to use too many memory

      - alert: MemoryUsage
        expr: (100 - ((node_memory_MemAvailable_bytes * 100) / node_memory_MemTotal_
↪bytes)) > 90
        for: 10m
        labels:
          severity: critical
        annotations:
          description: RAM of {{$labels.instance}} has been too used for more than 10␣
↪minutes
          summary: Instance {{$labels.instance}} is in danger
```

- Règle sur erreur bloquante dans une offre froide

```
groups:
  - name: tape_offer
    rules:
      - alert: DriveWorkerKO
        expr: vitam_offer_tape_workers_interrupted > 0
        for: 1s
        labels:
          severity: critical
        annotations:
          description: |-
            At least one tape offer drive worker is KO
          summary: Drive worker KO (offerId {{$labels.offerId}})
```

7.2.10.1.2 Configuration / fichiers utiles

Prometheus server est optionnel.

Important : L'utilisation d'une solution de monitoring (Prometheus ou autre) est critique pour le suivi du bon fonctionnement de Vitam dans un environnement de production.

Note : Dans le cas d'utilisation d'une offre froide, le déploiement de Prometheus (+ Grafana) est fortement recommandé pour le monitoring des métriques de l'offre froide. Un dashboard dédié est disponible sur Grafana.

7.2.10.1.2.1 Fichier `prometheus.yml`

```
# my global config
global:
  scrape_interval:     {{ prometheus.server.scrape_interval | default(15) }}s # Set␣
↪the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: {{ prometheus.server.evaluation_interval | default(15) }}s #␣
↪Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
    # - alertmanager:9093
{% for host in groups['hosts_alertmanager'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ prometheus.alertmanager.api_port |␣
↪default(9093) }}
{% endfor %}

# Load rules once and periodically evaluate them according to the global 'evaluation_
↪interval'.
rule_files:
  # - rule.yml
{% for item in rules_files.files %}
  - {{ item.path }}
{% endfor %}

scrape_configs:
{% if prometheus.node_exporter.enabled | default(true) | bool == true %}
  - job_name: vitam-node-exporter
    metrics_path: {{ prometheus.node_exporter.metrics_path | default('/metrics') }}
    static_configs:
    - targets:
{% for host in groups['vitam'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ prometheus.node_exporter.port |␣
↪default(9101) }}
{% endfor %}
{% endif %}

{% if prometheus.consul_exporter.enabled | default(true) | bool == true %}
  - job_name: vitam-consul_exporter-exporter
    metrics_path: {{ prometheus.consul_exporter.metrics_path | default('/metrics') }}
    static_configs:
    - targets:
{% for host in groups['vitam'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ prometheus.consul_exporter.port |␣
↪default(9107) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_access_internal']|length >0) %}
  - job_name: vitam-access-internal
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% for host in groups['hosts_access_internal'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.accessinternal.port_admin |␣
↪default(28101) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_access_external']|length >0) %}
  - job_name: vitam-access-external
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_access_external'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.accessexternal.port_admin |␣
↪default(28102) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_ingest_internal']|length >0) %}
  - job_name: vitam-ingest-internal
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_ingest_internal'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ingestinternal.port_admin |␣
↪default(28100) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_ingest_external']|length >0) %}
  - job_name: vitam-ingest-external
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_ingest_external'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ingestexternal.port_admin |␣
↪default(28001) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_metadata']|length >0) %}
  - job_name: vitam-metadata
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_metadata'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.metadata.port_admin |␣
↪default(28200) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_ihm_demo']|length >0) %}
  - job_name: vitam-ihm-demo
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_ihm_demo'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ihm_demo.port_admin |␣
↪default(28002) }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% endfor %}
{% endif %}

{% if (groups['hosts_ihm_recette']|length >0) %}
  - job_name: vitam-ihm-recette
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_ihm_recette'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ihm_recette.port_admin |␣
↪default(28204) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_logbook']|length >0) %}
  - job_name: vitam-logbook
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_logbook'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.logbook.port_admin | default(29002)␣
↪}}
{% endfor %}
{% endif %}

{% if (groups['hosts_workspace']|length >0) %}
  - job_name: vitam-workspace
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_workspace'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.workspace.port_admin |␣
↪default(28201) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_processing']|length >0) %}
  - job_name: vitam-processing
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_processing'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.processing.port_admin |␣
↪default(28203) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_worker']|length >0) %}
  - job_name: vitam-worker
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_worker'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.worker.port_admin | default(29104) }
↪}
{% endfor %}
{% endif %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% if (groups['hosts_storage_engine']|length >0) %}
  - job_name: vitam-storage-engine
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_storage_engine'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.storageengine.port_admin |␣
↪default(29102) }}
{% endfor %}
{% endif %}

{% if (groups['hosts_storage_offer_default']|length >0) %}
{% set offerInstances = [] %}
{% for host in groups['hosts_storage_offer_default'] %}
{{ offerInstances.append({"offerId": hostvars[host]['offer_conf'], "host": host }) }}
{% endfor %}
  - job_name: vitam-storage-offer-default
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
{% for offerId, hosts in offerInstances | groupby('offerId') %}
    - targets:
{% for host in hosts %}
      - {{ hostvars[host.host]['ip_admin'] }}:{{ vitam.storageofferdefault.port_admin␣
↪| default(29900) }}
{% endfor %}
      labels:
        offerId: {{ offerId }}
{% endfor %}
{% endif %}

{% if (groups['hosts_functional_administration']|length >0) %}
  - job_name: vitam-functional-administration
    metrics_path: {{ prometheus.metrics_path | default('/admin/v1/metrics') }}
    static_configs:
    - targets:
{% for host in groups['hosts_functional_administration'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.functional_administration.port_
↪admin | default(18004) }}
{% endfor %}
{% endif %}
```

7.2.10.1.2.2 Fichier de variable d'environnement

```
PROMETHEUS_OPTS='--web.listen-address={{ ip_admin }}:{{ prometheus.server.port |␣
↪default(9090) }} --web.external-url=http://{{ ip_admin }}:{{ prometheus.server.port␣
↪| default(9090) }}/prometheus --config.file=/vitam/conf/prometheus/prometheus.yml --
↪storage.tsdb.path=/vitam/data/prometheus'
# Following params can be added
# --web.enable-admin-api
# --web.page-title
# --web.cors.origin
# --web.route-prefix
```

7.2.10.1.2.3 Fichier de données

Ce service utilise des fichiers de données localisés dans le répertoire `/vitam/data/prometheus/`

7.2.10.1.3 Opérations

- Démarrage du service :

  En tant qu'utilisateur root :

```
systemctl start vitam-prometheus
```

- Arrêt du service :

  En tant qu'utilisateur root :

```
systemctl stop vitam-prometheus
```

- Consultation des logs du service :

  En tant qu'utilisateur root :

```
journalctl -u vitam-prometheus
```

- Superviser le service :

  Ce composant expose un ports en écoute :

```
# La commande ci-dessous doit afficher les numéros du port en écoute : <prometheus.
↪server.port>
sudo ss -anp | grep prometheus | grep LISTEN
```

7.2.10.2 Service vitam-alertmanager

7.2.10.2.1 Présentation

Le composant `vitam-alertmanager` permet de configurer les channels vers lesquels les alertes seront envoyées (ex. SLACK, Email, . . .).

Ce service est optionnel, et est déployé avec le nom `vitam-alertmanager`. La solution VITAM, par defaut, déploie une seule instance de ce service. Veuillez vous référer à la documentation officielle prometheus alertmanager pour pouvoir déployer un cluster alertmanager.

7.2.10.2.2 Configuration / fichiers utiles

Prometheus alertmanager est optionnel.

7.2.10.2.2.1 Fichier `alertmanager.yml`

```
global:
{% if http_proxy_environnement is defined and http_proxy_environnement|length > 0 %}
  http_config:
    proxy_url: '{{ http_proxy_environnement }}'
{% endif %}
  resolve_timeout: 5m

route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h
{% if prometheus.alertmanager.receivers is defined and prometheus.alertmanager.
↪receivers|length > 0 %}
  receiver: {{ prometheus.alertmanager.receivers.0.name }}
receivers:
{{ prometheus.alertmanager.receivers | to_nice_yaml(width=80, indent=2) }}
{% else %}
  receiver: 'null'
receivers:
- name: 'null'
{% endif %}
inhibit_rules:
  - source_match:
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['alertname', 'dev', 'instance']
```

7.2.10.2.2.2 Fichier de variable d'environnement

```
ALERTMANAGER_OPTS='--web.listen-address={{ ip_admin }}:{{ prometheus.alertmanager.api_
↪port | default(9093) }} --cluster.listen-address={{ ip_admin }}:{{ prometheus.
↪alertmanager.cluster_port | default(9094) }} --config.file=/vitam/conf/alertmanager/
↪alertmanager.yml --storage.path=/vitam/data/alertmanager'
# data.retention default 120h
# alerts.gc-interval default 30m
# web.external-url default
# web.external-url
# web.route-prefix default to path of --web.external-url
# more here https://github.com/prometheus/alertmanager/blob/master/cmd/alertmanager/
↪main.go
```

7.2.10.2.2.3 Fichier de données

Ce service utilise des fichiers de données localisés dans le répertoire `/vitam/data/alertmanager/`

7.2.10.2.3 Opérations

- Démarrage du service :

  En tant qu'utilisateur root :

```
systemctl start vitam-alertmanager
```

- Arrêt du service :

  En tant qu'utilisateur root :

```
systemctl stop vitam-alertmanager
```

- Consultation des logs :

  En tant qu'utilisateur root :

```
journalctl -u vitam-alertmanager
```

- Superviser le service :

  Ce composant expose deux ports en écoute, un port d'API et un autre pour le cluster :

```
# La commande ci-dessous doit afficher les numéros de ports en écoute : <prometheus.
↪alertmanager.api_port> et <prometheus.alertmanager.cluster_port>
sudo ss -anp | grep alertmanager | grep LISTEN
```

7.2.10.3 Service vitam-node-exporter

7.2.10.3.1 Présentation

Le composant `vitam-node-exporter` permet d'exposer via une API un ensemble de métriques liées au matériel et au noyau du système.

Ce service est déployé avec le nom `vitam-node-exporter`. Il doit être installé sur toutes les machines à superviser.

7.2.10.3.2 Configuration / fichiers utiles

Prometheus node exporter est activé par défaut. Cependant il est possible de désactiver son installation depuis la conf en éditant la variable `prometheus.node_exporter.enabled: false` dans le fichier `environments/group_vars/all/cots_var.yml`.

7.2.10.3.2.1 Fichier de variable d'environnement

```
NODE_EXPORTER_OPTS='--web.listen-address={{ ip_admin }}:{{ prometheus.node_exporter.
↪port | default(9101) }} --web.telemetry-path={{ prometheus.node_exporter.metrics_
↪path | default('/metrics') }} --collector.textfile.directory /vitam/data/node_
↪exporter/textfile_collector'
```

7.2.10.3.2.2 Fichier de données

Ce service n'utilise pas de fichier de données.

7.2.10.3.3 Opérations

- Démarrage du service :

  En tant qu'utilisateur root :

```
systemctl start vitam-node-exporter
```

- Arrêt du service :

  En tant qu'utilisateur root :

```
systemctl stop vitam-node-exporter
```

- Consultation des logs :

  En tant qu'utilisateur root :

```
journalctl -u vitam-node-exporter
```

- Accès au service pour réaliser un acte d'exploitation :

  Depuis une machine pour laquelle le flux réseau vers le service est ouvert, il suffit de faire un CURL sur l'API exposée par ce service :

```
curl <adresse>:<prometheus.node_exporter.port>/metrics
```

7.2.10.4 Exploitation du prometheus server

7.2.10.4.1 Ajouter alertmanager à la configuration prometheus

Ansible permet de générer automatiquement la configuration alertmanager dans le fichier `prometheus.yml`

Pour se faire, il suffit de rajouter les machines dédiées à alertmanager dans le groupe `[hosts_alertmanager]` de votre fichier d'inventaire.

```
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
    # - alertmanager:9093
{% for host in groups['hosts_alertmanager'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ prometheus.alertmanager.api_port }}
{% endfor %}
```

7.2.10.4.2 Ajouter des rules à la configuration prometheus

Ansible permet d'ajouter automatiquement les règles d'alertes dans le fichier `prometheus.yml`. Il suffit d'ajouter les fichiers de règles d'alerte dans le dossier `../../../../../../deployment/ansible-vitam-extra/roles/prometheus-server/rules/`

```
# Load rules once and periodically evaluate them according to the global 'evaluation_
↪interval'.
rule_files:
{% for item in rules_files.files %}
  - {{ item.path }}
{% endfor %}
```

7.2.10.4.3 Ajouter la configuration des hosts dans prometheus

Ansible permet d'ajouter automatiquement la `scrape_configs` chargée par des `job_name` depuis l'inventaire.

```
scrape_configs:
  - job_name: vitam-node-exporter
    metrics_path: {{ prometheus.node_exporter.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['vitam'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{prometheus.node_exporter.port}}
{% endfor %}

{% if (groups['hosts_access_internal']|length >0) %}
  - job_name: vitam-access-internal
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_access_internal'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.accessinternal.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_access_external']|length >0) %}
  - job_name: vitam-access-external
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_access_external'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.accessexternal.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_ingest_internal']|length >0) %}
  - job_name: vitam-ingest-internal
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_ingest_internal'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ingestinternal.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_ingest_external']|length >0) %}
  - job_name: vitam-ingest-external
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_ingest_external'] %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ingestexternal.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_metadata']|length >0) %}
  - job_name: vitam-metadata
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_metadata'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.metadata.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_ihm_demo']|length >0) %}
  - job_name: vitam-ihm-demo
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_ihm_demo'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ihm_demo.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_ihm_recette']|length >0) %}
  - job_name: vitam-ihm-recette
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_ihm_recette'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.ihm_recette.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_logbook']|length >0) %}
  - job_name: vitam-logbook
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_logbook'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.logbook.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_workspace']|length >0) %}
  - job_name: vitam-workspace
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_workspace'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.workspace.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_processing']|length >0) %}
  - job_name: vitam-processing
    metrics_path: {{ prometheus.metrics_path }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
    static_configs:
    - targets:
{% for host in groups['hosts_processing'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.processing.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_worker']|length >0) %}
  - job_name: vitam-worker
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_worker'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.worker.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_storage_engine']|length >0) %}
  - job_name: vitam-storage-engine
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_storage_engine'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.storageengine.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_storage_offer_default']|length >0) %}
  - job_name: vitam-storage-offer-default
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_storage_offer_default'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.storageofferdefault.port_admin }}
{% endfor %}
{% endif %}

{% if (groups['hosts_functional_administration']|length >0) %}
  - job_name: vitam-functional-administration
    metrics_path: {{ prometheus.metrics_path }}
    static_configs:
    - targets:
{% for host in groups['hosts_functional_administration'] %}
      - {{ hostvars[host]['ip_admin'] }}:{{ vitam.functional_administration.port_
↪admin }}
{% endfor %}
{% endif %}
```

7.2.11 Restic

7.2.11.1 Présentation

Restic est un outil opensource de sauvegarde fourni en tant qu'extra (beta) dans la suite logicielle Vitam. Son installation est optionnelle.

Il a pour but de simplifier la mise en oeuvre des sauvegardes des bases mongo sur les offres de stockages.

Ces sauvegardes sont utiles dans la reprise d'activité en cas de perte des bases de données (à minima les bases mongo-offer par défaut).

restic est déployé sur les machines du groupe [hosts_storage_offer_default] qui ont le paramètre *restic_enabled=true* de défini.

Note : Seuls les providers d'offres suivants sont supportés : *filesystem*, *filesystem-hash*, *openstack-swift-v3* et *amazon-s3-v1*.

7.2.11.1.1 Comment fonctionne Restic ?

L'outil Restic prend en charge les objets présents (fichiers et/ou sous arborescence) dans le répertoire qui lui est indiqué en ligne de commande. Dans notre cas, les dumps sont effectués sous `/vitam/tmp/restic/backup/`.

7.2.11.1.1.1 La notion d'«Incremental For Ever»

Techniquement, les sauvegardes Restic fonctionnent dans le mode suivant :

- Lors de la première sauvegarde, l'intégralité des données contenues dans le dossier à sécuriser sera recopiée et stockée dans ce repository.
- lors des sauvegardes suivantes, seuls les objets ajoutés ou modifiés seront sauvegardés.

La notion de sauvegarde totale ne s'applique donc que pour la toute première sauvegarde, les suivantes étant exclusivement incrémentales ; c'est le mode communément appelé « Incremental for ever ». Ce mécanisme permet de limiter drastiquement la volumétrie du flux de sauvegarde.

Les sauvegardes réalisées sont stockées sous forme de « snapshots » dans un repository hébergé sur l'offre de stockage.

7.2.11.1.1.2 La notion de snapshot

Dans le jargon Restic chaque sauvegarde est stockée sous la forme d'un snapshot et se voit attribuer un ID unique.

Le déclenchement des sauvegardes est réalisé via une crontab qui lance l'exécution du script `/vitam/script/restic/restic_backup`.

La sauvegarde peut être effectuée manuellement en ligne de commande par l'exécution de ce même script.

Par défaut, la purge des anciens snapshots est automatisé lors de l'exécution de ce même script. La configuration du nombre de snapshots à conserver est personnalisable.

7.2.11.2 Configuration / fichiers utiles

La configuration de restic est centralisée sous */vitam/conf/restic/*

7.2.11.2.1 Fichier restic.conf

Contient les paramètres de configuration de restic tel que le mot de passe des containers de sauvegarde ainsi que les paramètres d'accès au stockage associée.

7.2.11.2.2 Fichier conf.d/{{ restic.backup.name }}.conf

Contient les paramètres de configuration et d'accès de la base à sauvegarder.

7.2.11.3 Opérations

7.2.11.3.1 restic_backup

Le script est lancé automatiquement à partir d'une crontab associée à l'utilisateur vitam.

Pour éditer cette crontab manuellement, il est possible d'utiliser la commande suivante en tant qu'utilisateur root :

```
crontab -e -u vitam
```

Pour effectuer manuellement un backup ; exécutez la commande suivante en tant qu'utilisateur root :

```
/vitam/script/restic/restic_backup
```

Le répertoire temporaire pour le backup est */vitam/tmp/restic/backup/*.

7.2.11.3.2 restic_restore

Pour restaurer un snapshot, il est possible d'exécuter la commande suivante en tant qu'utilisateur root :

```
/vitam/script/restic/restic_restore
```

Il sera demandé de sélectionner le {{ restic.backup.name }} à restaurer, puis le snapshot_ID souhaité.

La restauration est faite sous */vitam/tmp/restic/restore/*.

7.2.11.3.3 Consultation des logs

En tant qu'utilisateur root :

```
vim /vitam/log/restic/restic-{{ restic.backup.name }}.log
```

7.2.12 Siegfried

7.2.12.1 Présentation

Siegfried est un outil permettant la détection de format d'un fichier.

7.2.12.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

7.2.12.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-siegfried`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-siegfried`

Avertissement : ne pas oublier que cela peut perturber le comportement de certains composants Vitam (ingest-external et worker).

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Logs

Les logs applicatifs sont envoyés par rsyslog à la solution de centralisation des logs ; il est néanmoins possible d'en visionner une représentation par la commande :

```
journalctl --unit vitam-siegfried
```

- Supervision du service

N/A

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

- Montée de version du fichier de signatures

Se reporter à *Montée de version du fichier de signature de Siegfried* (page 34)

CHAPITRE 8

Exploitation des composants de la solution logicielle VITAM

Les sections qui suivent donnent une description plus fine pour l'exploitation des services VITAM.

8.1 Généralités

Les composants de la solution logicielle *VITAM* sont déployés par un *playbook* ansible qui :

1. déploie, selon l'inventaire employé, les *packages* nécessaires
2. applique la configuration de chaque composant selon son contexte défini dans l'inventaire

Les composants *VITAM* sont décrits ci-après.

Avertissement : En cas de modification de la configuration, redémarrer le service associé.

8.2 Composants

8.2.1 Fichiers communs

Les composants de la solution logicielle *VITAM* utilisent un socle de fichiers communs.

8.2.1.1 Fichier `/vitam/conf/<composant>/sysconfig/java_opts`

Ce fichier définit les JVMARGS.

```
#*****************************************************************************
# Copyright French Prime minister Office/SGMAP/DINSIC/Vitam Program (2015-2022)
#
# contact.vitam@culture.gouv.fr
```

(suite sur la page suivante)

(suite de la page précédente)

```
#
# This software is a computer program whose purpose is to implement a digital archiving back-office system managing
# high volumetry securely and efficiently.
#
# This software is governed by the CeCILL 2.1 license under French law and abiding by the rules of distribution of free
# software. You can use, modify and/ or redistribute the software under the terms of the CeCILL 2.1 license as
# circulated by CEA, CNRS and INRIA at the following URL "https://cecill.info".
#
# As a counterpart to the access to the source code and rights to copy, modify and redistribute granted by the license,
# users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the
# successive licensors have only limited liability.
#
# In this respect, the user's attention is drawn to the risks associated with loading, using, modifying and/or
# developing or reproducing the software by the user in light of its specific status of free software, that may mean
# that it is complicated to manipulate, and that also therefore means that it is reserved for developers and
# experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the
# software's suitability as regards their requirements in conditions enabling the security of their systems and/or data
# to be ensured and, more generally, to use and operate it in the same conditions as regards security.
#
# The fact that you are presently reading this means that you have had knowledge of the CeCILL 2.1 license and that you
# accept its terms.
#*******************************************************************************
JAVA_OPTS="{{ vitam_struct.jvm_opts.gc | default(gc_opts) }} {{ vitam_struct.jvm_opts.memory | default(memory_opts) }} {{ vitam_struct.jvm_opts.java | default(java_opts) }} -Dorg.owasp.esapi.resources={{ vitam_folder_conf }} -Dlogback.configurationFile={{ vitam_folder_conf }}/logback.xml -Dvitam.config.folder={{ vitam_folder_conf }} -Dvitam.data.folder={{ vitam_folder_data }} -Dvitam.tmp.folder={{ vitam_folder_tmp }} -Dvitam.log.folder={{ vitam_folder_log }} -Djava.security.properties={{ vitam_folder_conf }}/java.security -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath={{ vitam_folder_log }}{% if jvm_log %} -XX:+UnlockDiagnosticVMOptions -XX:+LogVMOutput -XX:LogFile={{ vitam_folder_log }}/jvm.log{% endif %} -XX:+UseG1GC {{ vitam_struct.jmx_exporter | default(jmx_exporter_opts) }}"
JAVA_ARGS="{{ vitam_folder_conf }}/{{ vitam_struct.vitam_component }}.conf"
```

8.2.1.2 Fichier `/vitam/conf/<composant>/logback-access.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
{% if logback_rolling_policy|lower == "true" %}

  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <fileNamePattern>{{ vitam_folder_log }}/accesslog-{{ vitam_struct.vitam_component }}.%d{yyyy-MM-dd}.log</fileNamePattern>
```

(suite sur la page suivante)

(suite de la page précédente)

```
      <maxHistory>{{ vitam_struct.access_retention_days | default(access_retention_days) }}</maxHistory>
          <totalSizeCap>{{ vitam_struct.access_total_size_cap | default(access_total_size_cap) }}</totalSizeCap>
    </rollingPolicy>
{% else %}

  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>{{ vitam_folder_log }}/accesslog-{{ vitam_struct.vitam_component }}.log</file>
    <append>true</append>
{% endif %}
    <encoder>
      <pattern>%h %l %u %t "%r" %s %b "%i{Referer}" "%i{User-agent}" %D %i{X-Request-Id} %i{X-Tenant-Id} %i{X-Application-Id}</pattern>
    </encoder>
  </appender>
  <appender-ref ref="FILE" />
</configuration>
```

8.2.1.3 Fichier `/vitam/conf/<composant>/logback.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

  <!-- Send debug messages to System.out -->
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <!-- By default, encoders are assigned the type ch.qos.logback.classic.encoder.PatternLayoutEncoder -->
    <encoder>
      <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level %logger - %replace(%caller{1..2}){'Caller\+1   at |\n',''} : %msg %rootException%n</pattern>
    </encoder>
  </appender>

{% if logback_rolling_policy|lower == "true" %}
  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
      <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
      <maxFileSize>{{ vitam_struct.logback_max_file_size | default(logback_max_file_size) }}</maxFileSize>
      <maxHistory>{{ vitam_struct.logback_total_size_cap.file.history_days | default(logback_total_size_cap.file.history_days) }}</maxHistory>
          <totalSizeCap>{{ vitam_struct.logback_total_size_cap.file.totalsize | default(logback_total_size_cap.file.totalsize) }}</totalSizeCap>
    </rollingPolicy>
{% else %}
  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}.log</file>
    <append>true</append>
{% endif %}
    <encoder>
      <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level %logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n</pattern>
    </encoder>
  </appender>
</configuration>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    </encoder>
  </appender>

{% if logback_rolling_policy|lower == "true" %}
  <appender name="SECURITY" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
      <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_security.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
      <maxFileSize>{{ vitam_struct.logback_max_file_size | default(logback_max_file_size) }}</maxFileSize>
      <maxHistory>{{ vitam_struct.logback_total_size_cap.security.history_days | default(logback_total_size_cap.security.history_days) }}</maxHistory>
          <totalSizeCap>{{ vitam_struct.logback_total_size_cap.security.totalsize | default(logback_total_size_cap.security.totalsize) }}</totalSizeCap>
    </rollingPolicy>
{% else %}
  <appender name="SECURITY" class="ch.qos.logback.core.FileAppender">
    <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_security.log</file>
    <append>true</append>
{% endif %}
    <encoder>
      <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level %logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n</pattern>
    </encoder>
  </appender>

{% if vitam_struct.vitam_component == 'storage' %}
 {% if logback_rolling_policy|lower == "true" %}
  <appender name="OFFERSYNC" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
      <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_sync.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
      <maxFileSize>{{ vitam_struct.logback_max_file_size | default(logback_max_file_size) }}</maxFileSize>
      <maxHistory>{{ vitam_struct.logback_total_size_cap.offersync.history_days }}</maxHistory>
      <totalSizeCap>{{ vitam_struct.logback_total_size_cap.offersync.totalsize }}</totalSizeCap>
    </rollingPolicy>
  {% else %}
  <appender name="OFFERSYNC" class="ch.qos.logback.core.FileAppender">
    <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_sync.log</file>
    <append>true</append>
  {% endif %}
    <encoder>
      <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level %logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n</pattern>
    </encoder>
  </appender>
{% endif %}

{% if vitam_struct.vitam_component == 'storage' %}
 {% if logback_rolling_policy|lower == "true" %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
  <appender name="OFFERDIFF" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
      <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_diff.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
      <maxFileSize>{{ vitam_struct.logback_max_file_size | default(logback_max_file_size) }}</maxFileSize>
      <maxHistory>{{ vitam_struct.logback_total_size_cap.offerdiff.history_days }}</maxHistory>
      <totalSizeCap>{{ vitam_struct.logback_total_size_cap.offerdiff.totalsize }}</totalSizeCap>
    </rollingPolicy>
  {% else %}
  <appender name="OFFERDIFF" class="ch.qos.logback.core.FileAppender">
    <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_diff.log</file>
    <append>true</append>
  {% endif %}
    <encoder>
      <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level %logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n</pattern>
    </encoder>
  </appender>
{% endif %}

{% if vitam_struct.vitam_component == 'offer' %}
  {% if vitam_offers[offer_conf]["provider"] == 'tape-library' %}
    {% if logback_rolling_policy|lower == "true" %}
  <appender name="OFFER_TAPE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
      <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_tape.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
      <maxFileSize>{{ vitam_struct.logback_max_file_size | default(logback_max_file_size) }}</maxFileSize>
      <maxHistory>{{ vitam_struct.logback_total_size_cap.offer_tape.history_days }}</maxHistory>
      <totalSizeCap>{{ vitam_struct.logback_total_size_cap.offer_tape.totalsize }}</totalSizeCap>
    </rollingPolicy>
    {% else %}
  <appender name="OFFER_TAPE" class="ch.qos.logback.core.FileAppender">
    <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_offer_tape.log</file>
    <append>true</append>
    {% endif %}
    <encoder>
      <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level %logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n</pattern>
    </encoder>
  </appender>
  {% endif %}
{% endif %}

{% if vitam_struct.vitam_component == 'offer' %}
  {% if vitam_offers[offer_conf]["provider"] == 'tape-library' %}
    {% if logback_rolling_policy|lower == "true" %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
117   <appender name="OFFER_TAPE_BACKUP" class="ch.qos.logback.core.rolling.
↪RollingFileAppender">
118     <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
119       <fileNamePattern>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_tape_
↪backup.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
120       <maxFileSize>{{ vitam_struct.logback_max_file_size | default(logback_max_file_
↪size) }}</maxFileSize>
121       <maxHistory>{{ vitam_struct.logback_total_size_cap.offer_tape_backup.history_
↪days }}</maxHistory>
122       <totalSizeCap>{{ vitam_struct.logback_total_size_cap.offer_tape_backup.
↪totalsize }}</totalSizeCap>
123     </rollingPolicy>
124     {% else %}
125   <appender name="OFFER_TAPE_BACKUP" class="ch.qos.logback.core.FileAppender">
126     <file>{{ vitam_folder_log }}/{{ vitam_struct.vitam_component }}_tape_backup.log</
↪file>
127     <append>true</append>
128     {% endif %}
129     <encoder>
130       <pattern>%d{ISO8601} [[%thread]] [%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level
↪%logger - %replace(%caller{1..2}){'Caller\+1     at |\n',''} : %msg %rootException%n
131       </pattern>
132     </encoder>
133   </appender>
134   {% endif %}
135 {% endif %}
136
137   <appender name="SYSLOG" class="ch.qos.logback.classic.net.SyslogAppender">
138     <syslogHost>localhost</syslogHost>
139     <facility>{{ vitam_defaults.syslog_facility }}</facility>
140     <suffixPattern>vitam-{{ vitam_struct.vitam_component }}: %d{ISO8601} [[%thread]] [
↪%X{X-Request-Id}] [%X{X-Tenant-Id}] %-5level %logger - %replace(%caller{1..2}){
↪'Caller\+1     at |\n',''} : %msg %rootException%n</suffixPattern>
141   </appender>
142   <!-- By default, the level of the root level is set to TRACE -->
143   <root level="{{ vitam_struct.log_level | default(vitam_defaults.services.log_level)␣
↪}}">
144     <!-- <appender-ref ref="STDOUT" /> -->
145     <appender-ref ref="FILE" />
146     <appender-ref ref="SYSLOG" />
147   </root>
148
149   <logger name="org.eclipse.jetty" level="WARN"/>
150   <logger name="fr.gouv.vitam.storage.engine.server.logbook.StorageLogbookMock" level=
↪"INFO"/>
151   <logger name="fr.gouv.vitam.metadata.core.graph.StoreGraphService" level="INFO"/>
152   <logger name="fr.gouv.vitam.metadata.core.graph.GraphComputeServiceImpl" level="INFO
↪"/>
153   <logger name="fr.gouv.vitam.common" level="WARN" />
154 {% if vitam_defaults.reconstruction.log_level is defined or reconstruction.log_level␣
↪is defined %}
155   <logger name="fr.gouv.vitam.metadata.core.reconstruction.ReconstructionService"␣
↪level="{{ vitam_struct.reconstruction.log_level | default(reconstruction.log_level)␣
↪}}"/>
156   <logger name="fr.gouv.vitam.metadata.core.reconstruction.RestoreBackupService"␣
↪level="{{ vitam_struct.reconstruction.log_level |default(reconstruction.log_level) }
↪}"/>
```

(suite sur la page suivante)

(suite de la page précédente)

```
157   <logger name="fr.gouv.vitam.logbook.common.server.reconstruction.
      ↪ReconstructionService" level="{{ vitam_struct.reconstruction.log_level |␣
      ↪default(reconstruction.log_level) }}"/>
158   <logger name="fr.gouv.vitam.logbook.common.server.reconstruction.
      ↪RestoreBackupService" level="{{ vitam_struct.reconstruction.log_level |␣
      ↪default(reconstruction.log_level) }}"/>
159   <logger name="fr.gouv.vitam.functional.administration.common.impl.
      ↪ReconstructionServiceImpl" level="{{ vitam_struct.reconstruction.log_level |␣
      ↪default(reconstruction.log_level) }}"/>
160   <logger name="fr.gouv.vitam.functional.administration.common.impl.
      ↪RestoreBackupServiceImpl" level="{{ vitam_struct.reconstruction.log_level |␣
      ↪default(reconstruction.log_level) }}"/>
161 {% endif %}
162 {% if performance_logger|lower == "true" %}}
163   <logger name="fr.gouv.vitam.common.performance.PerformanceLogger" level="DEBUG"␣
      ↪additivity="false" >
164       <appender-ref ref="SYSLOG" />
165   </logger>
166 {% endif %}
167   <logger name="fr.gouv.vitam.common.alert.AlertServiceImpl" level="INFO">
168     <appender-ref ref="SECURITY" />
169   </logger>
170
171 {% if vitam_struct.vitam_component == 'storage' %}
172   <logger name="fr.gouv.vitam.storage.engine.server.offersynchronization" level="INFO
      ↪">
173         <appender-ref ref="OFFERSYNC" />
174   </logger>
175   <logger name="fr.gouv.vitam.storage.engine.server.offerdiff" level="INFO">
176         <appender-ref ref="OFFERDIFF" />
177   </logger>
178 {% endif %}
179
180 {% if vitam_struct.vitam_component == 'offer' %}
181   <logger name="fr.gouv.vitam.storage.offers.tape.process.ProcessExecutor" level="INFO
      ↪" additivity="false" >
182         <appender-ref ref="OFFER_TAPE" />
183   </logger>
184
185   <logger name="fr.gouv.vitam.storage.offers.tape.utils.BackupLogInformation" level=
      ↪"INFO" additivity="false" >
186           <appender-ref ref="OFFER_TAPE_BACKUP" />
187   </logger>
188
189 {% endif %}
190
191
192 {% if vitam_struct.vitam_component == 'metadata' %}
193   <logger name="fr.gouv.vitam.metadata.core.migration" level="INFO"/>
194 {% endif %}
195
196 </configuration>
```

8.2.1.4 Fichier `/vitam/conf/<composant>/jetty-config.xml`

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/
↪configure_9_0.dtd">

<!-- ============================================================= -->
<!-- Documentation of this file format can be found at:            -->
<!-- http://wiki.eclipse.org/Jetty/Reference/jetty.xml_syntax      -->
<!--                                                               -->
<!-- Additional configuration files are available in $JETTY_HOME/etc -->
<!-- and can be mixed in. See start.ini file for the default       -->
<!-- configuration files.                                          -->
<!--                                                               -->
<!-- For a description of the configuration mechanism, see the     -->
<!-- output of:                                                    -->
<!--   java -jar start.jar -?                                      -->
<!-- ============================================================= -->

<!-- ============================================================= -->
<!-- Configure a Jetty Server instance with an ID "Server"         -->
<!-- Other configuration files may also configure the "Server"     -->
<!-- ID, in which case they are adding configuration to the same   -->
<!-- instance.  If other configuration have a different ID, they   -->
<!-- will create and configure another instance of Jetty.          -->
<!-- Consult the javadoc of o.e.j.server.Server for all            -->
<!-- configuration that may be set here.                           -->
<!-- ============================================================= -->
<Configure id="Server" class="org.eclipse.jetty.server.Server">


    <!-- ============================================================= -->
    <!-- Add shared Scheduler instance                                 -->
    <!-- ============================================================= -->
    <Call name="addBean">
        <Arg>
            <New class="org.eclipse.jetty.util.thread.ScheduledExecutorScheduler"/>
        </Arg>
    </Call>

    <!-- ============================================================= -->
    <!-- Http Configuration.                                           -->
    <!-- This is a common configuration instance used by all           -->
    <!-- connectors that can carry HTTP semantics (HTTP, HTTPS, SPDY)-->
    <!-- It configures the non wire protocol aspects of the HTTP       -->
    <!-- semantic.                                                     -->
    <!--                                                               -->
    <!-- This configuration is only defined here and is used by        -->
    <!-- reference from the jetty-http.xml, jetty-https.xml and        -->
    <!-- jetty-spdy.xml configuration files which instantiate the      -->
    <!-- connectors.                                                   -->
    <!--                                                               -->
    <!-- Consult the javadoc of o.e.j.server.HttpConfiguration         -->
    <!-- for all configuration that may be set here.                   -->
    <!-- ============================================================= -->
    <New id="httpConfig" class="org.eclipse.jetty.server.HttpConfiguration">
        <Set name="secureScheme">http</Set>
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <Set name="securePort">8443</Set>
        <Set name="outputBufferSize">32768</Set>
        <Set name="requestHeaderSize">8192</Set>
        <Set name="responseHeaderSize">8192</Set>
        <Set name="sendServerVersion">false</Set>
        <Set name="sendDateHeader">false</Set>
        <Set name="headerCacheSize">512</Set>

        <!-- Uncomment to enable handling of X-Forwarded- style headers -->
        <!-- <Call name="addCustomizer">
          <Arg><New class="org.eclipse.jetty.server.ForwardedRequestCustomizer"/></Arg>
        </Call> -->

    </New>

    <!-- ======================= Original Connector ======================= -->
    <!-- <Call name="addConnector">
        <Arg>
            <New class="org.eclipse.jetty.server.ServerConnector">
                <Arg name="server"><Ref refid="Server" /></Arg>
                <Arg name="factories">
                    <Array type="org.eclipse.jetty.server.ConnectionFactory">
                        <Item>
                            <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                                <Arg name="config"><Ref refid="httpConfig" /></Arg>
                            </New>
                        </Item>
                    </Array>
                </Arg>
                <Set name="port">{{ vitam_struct.port_service }}</Set>
                <Set name="idleTimeout">
                    <Property name="http.timeout" default="{{ vitam_defaults.services.port_service_timeout }}"/>
                </Set>
            </New>
        </Arg>
    </Call> -->

    <!-- ============================================================ -->
    <!-- Set the default handler structure for the Server             -->
    <!-- A handler collection is used to pass received requests to    -->
    <!-- both the ContextHandlerCollection, which selects the next    -->
    <!-- handler by context path and virtual host, and the            -->
    <!-- DefaultHandler, which handles any requests not handled by    -->
    <!-- the context handlers.                                        -->
    <!-- Other handlers may be added to the "Handlers" collection,    -->
    <!-- for example the jetty-requestlog.xml file adds the           -->
    <!-- RequestLogHandler after the default handler                  -->
    <!-- ============================================================ -->
    <Set name="handler">
        <New id="Handlers" class="org.eclipse.jetty.server.handler.HandlerCollection">
            <Set name="handlers">
                <Array type="org.eclipse.jetty.server.Handler">
                    <Item>
                        <New id="Contexts" class="org.eclipse.jetty.server.handler.ContextHandlerCollection"/>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                    </Item>
                    <Item>
                        <New id="DefaultHandler" class="org.eclipse.jetty.server.handler.DefaultHandler"/>
                    </Item>
                </Array>
            </Set>
        </New>
    </Set>

    <Set name="RequestLog">
            <New id="RequestLogImpl" class="ch.qos.logback.access.jetty.VitamRequestLogImpl">
                <Set name="fileName">{{ vitam_folder_conf }}/logback-access.xml</Set>
            </New>
    </Set>
    <Ref refid="RequestLogImpl">
             <Call name="start"/>
    </Ref>

    <!-- =========================================================== -->
    <!-- extra server options                                        -->
    <!-- =========================================================== -->
    <Set name="stopAtShutdown">true</Set>
    <Set name="stopTimeout">5000</Set>
    <Set name="dumpAfterStart">false</Set>
    <Set name="dumpBeforeStop">false</Set>

{% if vitam_struct.https_enabled==true %}
   <New id="httpsConfig" class="org.eclipse.jetty.server.HttpConfiguration">
        <Set name="sendServerVersion">false</Set>
        <Set name="sendDateHeader">false</Set>
        <Call name="addCustomizer">
               <Arg>
                   <New class="org.eclipse.jetty.server.SecureRequestCustomizer">
                       <Arg name="sniHostCheck" type="boolean">
                           <Property name="jetty.ssl.sniHostCheck" default="false" />
                       </Arg>
                   </New>
               </Arg>
        </Call>
    </New>
        <New id="sslContextFactory" class="org.eclipse.jetty.util.ssl.SslContextFactory$Server">
        <Set name="KeyStorePath">{{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.jks</Set>
        <Set name="KeyStorePassword">{{ password_keystore }}</Set>
        <Set name="KeyManagerPassword">{{ password_manager_keystore }}</Set>
        <Set name="TrustStorePath">{{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks</Set>
        <Set name="TrustStorePassword">{{ password_truststore }}</Set>
        <Set name="TrustStoreType">JKS</Set>
        <Set name="NeedClientAuth">false</Set>
        <Set name="WantClientAuth">true</Set>
      <Set name="IncludeCipherSuites">
        <Array type="String">
          <Item>TLS_ECDHE.*</Item>
```

(suite sur la page suivante)

(suite de la page précédente)

```
          <Item>TLS_DHE_RSA.*</Item>
        </Array>
      </Set>
       <Set name="IncludeProtocols">
        <Array type="String">
          <Item>TLSv1.2</Item>
        </Array>
      </Set>
      <Set name="ExcludeCipherSuites">
        <Array type="String">
          <Item>.*NULL.*</Item>
          <Item>.*RC4.*</Item>
          <Item>.*MD5.*</Item>
          <Item>.*DES.*</Item>
          <Item>.*DSS.</Item>
          <Item>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA</Item>
          <Item>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA</Item>
          <Item>TLS_DHE_RSA_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_DHE_RSA_WITH_AES_128_CBC_SHA</Item>
        </Array>
      </Set>
        <Set name="UseCipherSuitesOrder">true</Set>
        <Set name="RenegotiationAllowed">true</Set>
    </New>
    <New id="sslConnectionFactory" class="org.eclipse.jetty.server.
↪SslConnectionFactory">
        <Arg name="sslContextFactory">
            <Ref refid="sslContextFactory" />
        </Arg>
        <Arg name="next">http/1.1</Arg>
    </New>
    <New id="businessConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server">
            <Ref refid="Server" />
        </Arg>
        <Arg name="factories">
            <Array type="org.eclipse.jetty.server.ConnectionFactory">
                <Item>
                    <Ref refid="sslConnectionFactory" />
                </Item>
                <Item>
                    <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                        <Arg name="config">
                            <Ref refid="httpsConfig" />
                        </Arg>
                    </New>
                </Item>
            </Array>
        </Arg>
        <Set name="host">{{ ip_service }}</Set>
        <Set name="port">
            <SystemProperty name="jetty.port" default="{{ vitam_struct.port_service }}
↪"/>
        </Set>
        <Set name="name">business</Set>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    </New>

{% else %}

    <!-- ============================================================= -->
    <!-- Connector for API business                                    -->
    <!-- Attach all ContextHanlder except Admin                        -->
    <!-- ============================================================= -->

    <New id="businessConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server"><Ref refid="Server" /></Arg>
        <Arg name="factories">
            <Array type="org.eclipse.jetty.server.ConnectionFactory">
                <Item>
                    <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                        <Arg name="config"><Ref refid="httpConfig" /></Arg>
                    </New>
                </Item>
            </Array>
        </Arg>
        <Set name="host">{{ ip_service }}</Set>
        <Set name="port">{{ vitam_struct.port_service }}</Set>
        <Set name="name">business</Set>
        <Set name="idleTimeout">
            <Property name="http.timeout" default="{{ vitam_defaults.services.port_
↪service_timeout }}"/>
        </Set>
    </New>

{% endif %}

    <!-- ============================================================= -->
    <!-- Connector for API Admin                                       -->
    <!-- Attach all ContextHanlder                                     -->
    <!-- ============================================================= -->

    <New id="adminConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server"><Ref refid="Server" /></Arg>
        <Arg name="factories">
            <Array type="org.eclipse.jetty.server.ConnectionFactory">
                <Item>
                    <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                        <Arg name="config"><Ref refid="httpConfig" /></Arg>
                    </New>
                </Item>
            </Array>
        </Arg>
        <Set name="host">{{ ip_admin }}</Set>
        <Set name="port">{{ vitam_struct.port_admin }}</Set>
        <Set name="name">admin</Set>
        <Set name="idleTimeout">
            <Property name="http.timeout" default="{{ vitam_defaults.services.port_
↪service_timeout }}"/>
        </Set>
    </New>

    <Call name="setConnectors">
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <Arg>
             <Array type="org.eclipse.jetty.server.ServerConnector">
                <Item>
                    <Ref refid="businessConnector" />
                </Item>
                <Item>
                    <Ref refid="adminConnector" />
                </Item>
            </Array>
        </Arg>
    </Call>

</Configure>
```

8.2.1.5 Fichier /vitam/conf/<composant>/logbook-client.conf

Ce fichier permet de configurer l'appel au composant logbook.

```
serverHost: {{ vitam.logbook.host }}
serverPort: {{ vitam.logbook.port_service }}
```

8.2.1.6 Fichier /vitam/conf/<composant>/server-identity.conf

```
identityName: {{ ansible_nodename }}
identityRole: {{ vitam_struct.vitam_component }}
identitySiteId: {{ vitam_site_id }}
```

8.2.1.7 Fichier /vitam/conf/<composant>/antisamy-esapi.xml

8.2.1.8 Fichier /vitam/conf/<composant>/vitam.conf

```
secret : {{ plateforme_secret }}
filterActivation : {{ vitam_struct.secret_platform }}
{% if vitam_struct.vitam_component == vitam.processing.vitam_component %}
distributeurBatchSize: 800
workerBulkSize: 16
{% endif %}
{% if vitam_struct.vitam_component == vitam.metadata.vitam_component %}
storeGraphElementsPerFile: 10000
storeGraphOverlapDelay: 300
expireCacheEntriesDelay: 300
deleteIncompleteReconstructedUnitDelay: 2592000
migrationBulkSize: 10000
workspaceFreespaceThreshold: {{ vitam.metadata.workspaceFreespaceThreshold |␣
↪default(25) }}
{% endif %}
distributionThreshold : 100000
eliminationAnalysisThreshold : 100000
eliminationActionThreshold : 10000
computedInheritedRulesThreshold : 100000000
intervalDelayCheckIdle : 5000
```

(suite sur la page suivante)

(suite de la page précédente)

```
maxDelayUnusedConnection : 5000
delayValidationAfterInactivity : 2500
tenants: [ "{{ vitam_tenant_ids | join('", "') }}" ]
adminTenant : {{ vitam_tenant_admin | default(1) }}
forceChunkModeInputStream : {{ vitam_defaults.vitam_force_chunk_mode | default(false) }}

{% if vitam_struct.vitam_component == vitam.worker.vitam_component %}
reclassificationMaxBulkThreshold: 1000
reclassificationMaxUnitsThreshold: 10000
reclassificationMaxGuildListSizeInLogbookOperation: 1000
queriesThreshold: {{ vitam.worker.queriesThreshold | default(100000) }}
bulkAtomicUpdateBatchSize: {{ vitam.worker.bulkAtomicUpdateBatchSize | default(100) }}
bulkAtomicUpdateThreadPoolSize: {{ vitam.worker.bulkAtomicUpdateThreadPoolSize | default(8) }}
bulkAtomicUpdateThreadPoolQueueSize: {{ vitam.worker.bulkAtomicUpdateThreadPoolQueueSize | default(16) }}

binarySizePlatformThreshold: # 1 Go
 limit: {{ vitam.worker.binarySizePlatformThreshold | default(1) }}
 sizeUnit: {{ vitam.worker.binarySizePlatformThresholdSizeUnit | default('GIGABYTE') }}

binarySizeTenantThreshold: # exemple for tenant 0 max dip/transfer size is 20 Mo, true means can exceed tenant threshold.
#  - tenant: 0
#    limit: 20
#    sizeUnit: MEGABYTE
#    authorize: true

{% endif %}

keywordMaxLength: 32766
textMaxLength: 32766

classificationLevel :
  allowList : [{% for classification in classificationList %}{{ classification }}{% if not loop.last %},{% endif %}{% endfor %}]
  authorizeNotDefined: {{ classificationLevelOptional }}

indexInheritedRulesWithAPIV2OutputByTenant: [ "{{ vitam.worker.api_output_index_tenants | join('", "') }}" ]
indexInheritedRulesWithRulesIdByTenant: [ "{{ vitam.worker.rules_index_tenants | join('", "') }}" ]

environmentName: {{ vitam_prefix_offer|default(vitam_site_name) }}

acceptableRequestTime: {{ acceptableRequestTime|default(10) }}
criticalRequestTime: {{ criticalRequestTime|default(60) }}
requestTimeAlertThrottlingDelay: {{ requestTimeAlertThrottlingDelay|default(60) }}

# Ontology cache settings (max entries in cache & retention timeout in seconds)
ontologyCacheMaxEntries: {{ vitam_defaults.ontologyCacheMaxEntries | default(100) }}
ontologyCacheTimeoutInSeconds: {{ vitam_defaults.ontologyCacheTimeoutInSeconds | default(300) }}

# Elasticsearch scroll timeout settings
```

(suite sur la page suivante)

(suite de la page précédente)

```
elasticSearchScrollTimeoutInMilliseconds: {{ vitam_defaults.elasticSearchScrollTimeoutInMilliseconds | default(300000) }}

{% if vitam_struct.vitam_component == vitam.processing.vitam_component %}
processEngineWaitForStepTimeout: 172800
{% endif %}
```

Ce fichier permet de définir les variables d'environnement vitam.

- ***binarySizePlatformThreshold* est le seuil de plate-forme du poids binaire max autorisé pour un DIP.**

  elle comporte deux clés : limit : le seuil sizeUnit : l'unité de taille (GIGABYTE / MEGABYTE / KILOBYTE / BYTE)

  par défaut le seuil est 1 Go. exemple :

```
binarySizePlatformThreshold:
    limit: 1
    sizeUnit: GIGABYTE / MEGABYTE / KILOBYTE / BYTE
```

- ***binarySizeTenantThreshold* est une liste qui constitue l'ensemble des seuils du poids binaire max autorisé pour un DIP par**

  Cette liste comporte 4 clés : tenant : le tenant limit : le seuil sizeUnit : l'unité de taille (GIGABYTE / MEGABYTE / KILOBYTE / BYTE) authorise : true si l'utilisateur peut excéder le seuil prédéfini.

  exemple :

```
binarySizeTenantThreshold:
   - tenant: 0
     limit: 20
     sizeUnit: MEGABYTE
     authorize: false
   - tenant: 1
     limit: 100
     sizeUnit: MEGABYTE
     authorize: true
```

8.2.1.9 Fichier `/vitam/conf/<composant>/java.security`

```
# Use Bouncy Castle Provider when it is available
security.provider.9=org.bouncycastle.jce.provider.BouncyCastleProvider

# Override the default list of Centos 7 that disable Elliptic Curved Based Algorithms
jdk.tls.disabledAlgorithms="SSLv3, TLSv1, TLSv1.1, RC4, MD5withRSA, DH keySize < 768, RSA keySize < 2048"
```

8.2.2 Access

8.2.2.1 access external

8.2.2.1.1 Présentation

Access-external est le composant d'interface entre *VITAM* et un *SIA* client, permettant de réaliser des recherches sur les objets archivés et les journaux. Il permet également quelques fonctions d'administration, en particulier les chargements des référentiels.

Rôle :

- Exposer les API publiques du système
- Sécuriser l'accès aux API de VITAM

8.2.2.1.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/access-external`.

8.2.2.1.2.1 Fichier `access-external.conf`

```
authentication: false
jettyConfig: jetty-config.xml
tenantFilter : true
authorizeTrackTotalHits : {{ vitam.accessexternal.authorizeTrackTotalHits |␣
↪default(false) }}
objectGroupBlackListedFieldsForVisualizationByTenant:
{% for tenant in vitam_tenant_ids %}
   {{ tenant }} : ['Filename']
{% endfor %}
```

8.2.2.1.2.2 Fichier `access-internal-client.conf`

```
serverHost: {{ vitam.accessinternal.host }}
serverPort: {{ vitam.accessinternal.port_service }}
```

8.2.2.1.2.3 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.2.1.2.4 Fichier `ingest-internal-client.conf`

```
serverHost: {{ vitam.ingestinternal.host }}
serverPort: {{ vitam.ingestinternal.port_service }}
```

8.2.2.1.2.5 Fichier `internal-security-client.conf`

```
serverHost: {{ vitam.security_internal.host }}
serverPort: {{ vitam.security_internal.port_service }}
secure: false
```

8.2.2.1.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-access-external`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-access-external`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/access-external/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.2.2 access-internal

8.2.2.2.1 Présentation du composant

Access-internal est le composant *VITAM*, permettant de réaliser des recherches et consultations sur les objets archivés et les journaux. Il permet également de modifier les informations d'un *ArchiveUnit*.

Rôle :

- Permettre l'accès aux données du système VITAM

Fonction :

- Exposition des fonctions de recherche d'archives offertes par metadata ;
- Exposition des fonctions de parcours de journaux offertes par logbook.

8.2.2.2.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/access`.

8.2.2.2.2.1 Fichier `access-internal.conf`

Ce fichier permet de définir l'URL d'accès au metadata server.

```
urlMetaData: {{vitam.metadata | client_url}}
urlWorkspace: {{vitam.workspace | client_url}}
urlProcessing: {{vitam.processing | client_url}}
jettyConfig: jetty-config.xml
```

8.2.2.2.2.2 Fichier `storage-client.conf`

Ce fichier permet de définir l'accès au storage-engine.

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.2.2.2.3 Fichier `metadata-client.conf`

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

8.2.2.2.2.4 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.2.2.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-access-internal`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-access-internal`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/access/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.3 Batch-Report

8.2.3.1 Présentation

Le composant batch-report permet de stocker des données de traitements de masse (en particulier, élimination) pour les aggréger sous forme de rapports.

Ce module utilise une base de données MongoDB « report », dans laquelle sont stockées, sous différentes collections (entre autres, EliminationActionObjectGroup et EliminationActionUnit), les données.

Ce module est appelé par le composant « worker » pour collecter les données durant les *workflows* d'élimination, entre autres.

Ce module est ensuite appelé par le composant « worker » pour restituer les données aggrégées sous forme de rapports.

8.2.3.2 Configuration

Ce document spécifie la configuration (fichiers de config) pour lancer le services de batch-report.

Tous ces fichiers de configuration sont placés dans le répertoire `/vitam/conf/batch-report`.

8.2.3.2.1 Fichier `batch-report.conf`

```
# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: report
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].report.user }}
dbPassword: {{ mongodb['mongo-data'].report.password }}
jettyConfig: jetty-config.xml

workspaceUrl: {{ vitam.workspace | client_url }}
```

8.2.3.3 Client batch-report

Pour la création d'un client batch-report, nous avons besoin aussi du fichier de configuration `batch-report-client.conf` qui précise le serveur host et la porte du serveur où le client se connecte pour les requêtes.

8.2.3.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-batch-report`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-batch-report`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.4 Logbook

8.2.4.1 Présentation

Rôle :

- Permettre de gérer l'archivage sur VITAM (A faire)

Fonctions :

- Intialiser des transacrions.
- Ajouter des Unités d'archives à la transaction.
- Ajouter un object group a une unité d'archive.
- Ajouter un binaire a un object group.
- Fermer la transaction.
- Produire un SIP et l'envoyer a Vitam (Ingest).

8.2.4.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/collect`.

8.2.4.2.1 Fichier `collect.conf`

Ce fichier précise les URLs pour le service « Workspace », configuration base de données, et la configuration du serveur jetty.

8.2.4.2.2 Fichier `functionnal-administration-client.conf`

8.2.4.2.3 Fichier `internal-security-client.conf`

8.2.4.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-collect`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-collect`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/collect/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

8.2.5 common-plugin

8.2.5.1 Présentation du composant

common-plugin est le composant permettant de réaliser des plugins sans appel à des package privé.

Rôle :

- l'objet de ce common-plugin n'est pas que de fournir des interfaces à implémenter mais aussi les classes d'implémentations imposées par Vitam pour réaliser des plugins.

Fonction :

- Exposition des interfaces à implémenter et les classes d'implémentations pour réaliser des plugins .

8.2.5.2 Classes utiles

L'Objectif de Plugin Common est d'inclure tous les classes utiles afin de créer un plugin à partir de ce package .

Les classes de model sont définis sous `/vitam/common/model`.

8.2.5.2.1 Classe `Item Status`

Ce classe permet de retourner le statut d'un Item.

8.2.5.2.2 Classe `VitamAutoCloseable`

Le mot clé try-with-resources garantit que chaque ressource sera fermée lorsqu'elle n'est plus utilisée. Une ressource et un objet qui implémente l'interface VitamAutoCloseable. Il est donc possible d'utiliser une instance de ces interfaces avec le mot clé try-with-resources.

Les classes de common parameter sont définis sous `/vitam/common/parameter`.

8.2.5.2.3 Classe `ParameterHelper`

Ce classe permet de faire un check sur les paramètres et avoir le tenant parameter de session vitam .

8.2.5.2.4 Classe `VitamParameter`

Cet interface permet d'aider à créer des nouveaux paramètres liés au classes .

Les classes de common exception sont définis sous `/vitam/processing/common/exception`.

8.2.5.2.5 Classe `ProcessingException`

Ce classe est le classe père de tous les Vitam Processing Exception .

Les classes de model common processing sont définis sous `/vitam/processing/common/model`.

8.2.5.2.6 Classe `IOParameter`

Ce class permet de définir les paramètres Input et Output pour une action et une step .

8.2.5.2.7 Classe `ProcessingUri`

Ce classe permet de formatter le processing URI .

8.2.5.2.8 Classe `UriPrefix`

C'est le Handler IO

Les classes des paramètres common sont définis sous `/vitam/processing/common/parameter`.

8.2.5.2.9 Classe `AbstractWorkerParameters`

C'est une implémentation abstraite de tous les paramètres de workers .

8.2.5.2.10 Classe `DefaultWorkerParameters`

Ce classe permet de définir les paramètres par défaut d'un worker.

8.2.5.2.11 Classe `WorkerParameterName`

Ce classe inclut une énumération avec tous les noms des paramètres d'un worker .

8.2.5.2.12 Classe `WorkerParameters`

Ce classe permet de définir les paramètres de worker.

8.2.5.2.13 Classe `WorkerParametersDeserializer`

Ce classe permet de définir les paramètres d'un worker deserializer.

8.2.5.2.14 Classe `WorkerParametersFactory`

Ce classe permet de définir les paramètres d'un worker Factory.

8.2.5.2.15 Classe `WorkerParametersSerializer`

Ce classe permet de définir les paramètres de Worker Serializer.

Les classes de model sont définis sous `/vitam/worker/common`.

8.2.5.2.16 Interface `HandlerIO`

Cet interface permet de définir les paramètres in et out de tous les Handlers.

Les classes de l'api sont définis sous `/vitam/worker/core/api`.

8.2.5.2.17 Classe `WorkerAction`

C'est l'interface contrat de tous les actions Handler event. Un action Handler doit implémenter cette interface .

Les classes de l'implémentation sont définis sous `/vitam/worker/core/impl`.

8.2.5.2.18 Classe `HandlerIOImpl`

Ce classe définit les paramètres in et out d'un Handler

How to use : Pour créer un Plugin :

- extends Abstract Class Action Handler
- implementer l'interface VitamAutoCloseable pour garantir qu'une ressource sera fermée lorsqu'elle n'est plus utilisée.
- Un constructeur par défaut
- **redéfinir la méthode execute de l'Action Handler :**
    - Paramètre WorkerParameters et Handler IO
    - type de retour Item Status
    - throws Processing Exception
- **faire l'override de méthode CheckMandatoryIOParameter**
    - Paramètre Handler IO
    - throws Processing Exception

8.2.6 Common

8.2.6.1 Présentation

8.2.6.2 Format Identifiers

Les services d'identification de formats peuvent être déployés sur tous les serveurs applicatifs vitam.

8.2.6.2.1 Configuration des services d'identification des formats

Dans **/vitam/conf** du serveur applicatif où sont déployés les services d'identification de formats, il faut un fichier **format-identifiers.conf**. C'est un fichier YAML de configuration des services d'identification de format. Il possède les configurations des services que l'on souhaite déployer sur le serveur.

Le code suivant contient un exemple de toutes les configurations possibles :

```
  siegfried-local:
type: SIEGFRIED
client: http
host: localhost
port: 55800
rootPath: /root/path
versionPath: /root/path/version/folder
createVersionPath: false
  mock:
type: MOCK
```

- **Le service Mock :**
  - identifié par *mock*
  - *type* : le type de service déployé : *MOCK*
- **Le service Siegfried :**
  - identifié par *siegfried-local*
  - *type* : le type de service déployé : *SIEGFRIED*
  - *client* : type de client (pour le moment seul *http* existe).
  - *host* : le host du serveur siegfried déployé (devrait être le host du serveur courant)
  - *port* : le port du serveur siegfried déployé
  - *rootPath* : la racine sur laquelle le service Siegfried doit résoudre les fichiers à tester (ex : « /data »)
  - *versionPath* : le chemin vers un dossier vide pour renvoyer la version (Doit posséder des droits en lecture)
  - *createVersionPath* : Si *false* le dossier doit pré-existant sur le server sur lequel tourne Siegfried. Sinon, le client siegfried tente de créer automatiquement le dossier en local.

NOTE : Chaque serveur est en charge de décrire la configuration nécessaire

8.2.7 Functional administration

8.2.7.1 Présentation

8.2.7.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/functional-administration`.

8.2.7.2.1 Fichier `functional-administration.conf`

ce fichier permet de définir l'URL d'accès au access server.

```
# Configuration MongoDB
mongoDbNodes:
{% for host in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[host]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: masterdata
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].functionalAdmin.user }}
dbPassword: {{ mongodb['mongo-data'].functionalAdmin.password }}

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}

jettyConfig: jetty-config.xml
workspaceUrl: {{vitam.workspace | client_url}}
processingUrl: {{vitam.processing | client_url}}

# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for host in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[host]['ip_service'] }}
  httpPort: {{ elasticsearch.data.port_http }}
{% endfor %}

# ElasticSearch tenant indexation
elasticsearchTenantIndexation:
  default_config:
    number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪masterdata.number_of_shards }}
    number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪masterdata.number_of_replicas }}

{% for collection in ["accesscontract", "accessionregisterdetail",
↪"accessionregistersummary", "accessionregistersymbolic", "agencies",
↪"archiveunitprofile", "context", "fileformat", "filerules", "griffin",
↪"ingestcontract", "managementcontract", "ontology", "preservationscenario", "profile
↪", "securityprofile"] %}
{% if vitam_elasticsearch_tenant_indexation.masterdata[collection] is defined %}
  {{collection}}:
{% if vitam_elasticsearch_tenant_indexation.masterdata[collection].number_of_shards␣
↪is defined %}
    number_of_shards: {{ vitam_elasticsearch_tenant_indexation.masterdata[collection].
↪number_of_shards }}
{% endif %}
{% if vitam_elasticsearch_tenant_indexation.masterdata[collection].number_of_replicas␣
↪is defined %}
    number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.
↪masterdata[collection].number_of_replicas }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% endif %}
{% endif %}
{% endfor %}

# ExternalId configuration
listEnableExternalIdentifiers:
{% if vitam_tenants_usage_external is iterable %}
{% for tenant in vitam_tenants_usage_external %}
{% if tenant.identifiers is defined %}
  {{ tenant.name }}:
{% for external in tenant.identifiers %}
    - {{ external }}
{% endfor %}
{% endif %}
{% endfor %}
{% endif %}

listMinimumRuleDuration:
{% if vitam_tenant_rule_duration is iterable %}
{% for tenant in vitam_tenant_rule_duration %}
  {{ tenant.name }}:
{% for rule in tenant.rules %}
{% for key, value in rule.items() %}
    {{ key }}: {{ value }}
{% endfor %}
{% endfor %}
{% endfor %}
{% endif %}

accessionRegisterSymbolicThreadPoolSize: {{ vitam.functional_administration.
↪accessionRegisterSymbolicThreadPoolSize}}
ruleAuditThreadPoolSize: {{ vitam.functional_administration.ruleAuditThreadPoolSize}}
```

8.2.7.2.2 Passage des identifiants des référentiels en mode esclave

La génération des identifiants des référentiels est gérée par Vitam quand il fonctionne en mode maître.

Par exemple :

- Préfixé par `PR-` our les profils
- Préfixé par `IC-` pour les contrats d'entrée
- Préfixé par `AC-` pour les contrats d'accès

Si vous souhaitez gérer vous-même les identifiants sur un service référentiel, il faut qu'il soit en mode esclave.

Note : Cette modification de comportement est réalisable post-installation. Une interruption temporaire de service est à prévoir (redémarrage du service `vitam-functional-administration`.

Par défaut, tous les services référentiels de *VITAM* fonctionnent en mode maître. Pour désactiver le mode maître de Vitam, il faut modifier le fichier de configuration `/vitam/conf/functional-administration/functional-administration.conf`.

```
# ExternalId configuration

```

(suite sur la page suivante)

(suite de la page précédente)

```
listEnableExternalIdentifiers:
 0:
   - INGEST_CONTRACT
   - ACCESS_CONTRACT
 1:
   - INGEST_CONTRACT
   - ACCESS_CONTRACT
   - PROFILE
   - SECURITY_PROFILE
   - CONTEXT
```

Depuis la version 1.0.4, la configuration par défaut de Vitam autorise des identifiants externes (ceux qui sont dans le fichier json importé).

- pour le tenant 0 pour les référentiels : contrat d'entrée et contrat d'accès.
- pour le tenant 1 pour les référentiels : contrat d'entrée, contrat d'accès, profil, profil de sécurité et contexte.

La liste des choix possibles, pour chaque tenant, est :

- INGEST_CONTRACT : contrats d'entrée
- ACCESS_CONTRACT : contrats d'accès
- PROFILE : profils SEDA
- SECURITY_PROFILE : profils de sécurité (utile seulement sur le tenant d'administration)
- CONTEXT : contextes applicatifs (utile seulement sur le tenant d'administration)
- ARCHIVE_UNIT_PROFILE : profils d'unités archivistiques

Note : se référer au métier pour ces choix.

Avertissement : Cette modification implique le redémarrage du/des composants (si mono-instance ou multi-instances du composant).

Prudence : En mode « esclave », il est fortement recommandé de faire débuter les référentiels avec d'autres chaînes de caractères que celle définies en mode « maître ».

Prudence : Ne pas oublier de répercuter cette modification sur le site secondaire

8.2.7.2.3 Paramétrage du batch de calcul pour l'indexation des règles héritées

La paramétrage du batch de calcul pour l'indexation des règles héritées peut être réalisé dans le fichier `/group_vars/all/vitam_vars.yml`.

La section suivante du fichier `vitam_vars.yml` permet de paramétrer la fréquence de passage du batch :

```
vitam_timers:
    metadata:
        - name: vitam-metadata-computed-inherited-rules
          frequency: "*-*-* 02:30:00"
```

La section suivante du fichier `vitam_vars.yml` permet de paramétrer la liste des tenants sur lequels s'exécute le batch :

```
vitam:
  worker:
        # api_output_index_tenants : permet d'indexer les règles de gestion, les␣
↪chemins des règles et les services producteurs
        api_output_index_tenants: [0,1,2,3,4,5,6,7,8,9]
        # rules_index_tenants : permet d'indexer les règles de gestion
        rules_index_tenants: [0,1,2,3,4,5,6,7,8,9]
```

8.2.7.2.4 Configuration du Functional administration

functional-administration.conf : Fichier Yaml de configuration du server *worker*. Il possède une propriété :

- **listMinimumRuleDuration** : la durée minimum de chaque type de règle par tenant

```
listMinimumRuleDuration:
  2:
    AppraisalRule : 1 year
```

8.2.7.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-functional-administration`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-functional-administration`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/functional-administration/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.8 Hello World Plugin

8.2.8.1 Présentation

Le composant hello-world-plugin est un exemple qui montre comment développer un plugin custom. Il suffit donc de prendre ce projet maven comme exemple et d'adapter le plugin à souhait.

Note : Sur le **pom** de ce module, seule la dépendance vers `common-plugin` est nécessaire.

```
<dependency>
    <groupId>fr.gouv.vitam</groupId>
    <artifactId>common-plugin</artifactId>
    <version>${vitam.version}</version>
</dependency>
```

Vous pouvez bien sûr ajouter d'autres dépendances qui servent à votre *plugin custom*.

8.2.8.1.1 Comment intégrer votre plugins dans vitam ?

Après avoir développé votre *plugin* en suivant les consignes ci-dessus, il faut faire ce qui suit :

- Générer votre jar
- Copier votre jar manuellement dans le dossier `/vitam/conf/worker/plugins-workspace/`, ou bien copier le dans le dossier de déploiement ansible `~/vitam/deployment/ansible-vitam/roles/vitam/files/worker/plugins-workspace/`
- Modifier le fichier plugins.json qui se trouve soit dans le dossier `/vitam/conf/worker/plugins.json` en production, ou bien dans le dossier de déploiement ansible qui se trouve :`vitam/deployment/ansible-vitam/roles/vitam/files/worker/plugins.json`, comme suit :

```
"HELLO_WORLD_PLUGIN": {
  "className": "fr.vitam.plugin.custom.HelloWorldPlugin",
  "propertiesFile": "hello_world_plugin.properties",
  "jarName": "hello-world-plugin-1.14.0-SNAPSHOT.jar"
}
```

Avertissement : jarName doit contenir uniquement le nom du jar avec extension ".jar"

A présent sur n'importe quel workflow, vous pouvez ajouter une action ayant comme « actionKey » la clé de votre plugin. Dans cet exemple : actionKey=HELLO_WORLD_PLUGIN

8.2.8.1.2 Créer un nouveau *workflow*

Tout d'abord création d'un nouveau workflow :

- Créer un nouveau *workflow* peut se résumer juste à copier un *workflow* existant et modifier son *identifier* et son *workerGroupId* pour, par exemple, utiliser des machines plus puissantes pour ce *workflow*
- L'identifier d'un *workflow* doit être UNIQUE, sinon un *workflow* existant portant le même *identifier* sera remplacé par le nouveau.
- La valeur de `typeProc` n'est pas actuellement dynamique (veuillez vous référer à l'enum **LogbookTypeProcess** pour voir les différentes valeurs possibles)

- La valeur `actionKey` doit être soit le *handler name* d'un *plugin* ou *handler* existant, soit celui d'un nouveau *plugin*.

Exemple d'un *workflow* :

```
{
  "id": "SampleIngestWorkflow",
  "name": "Sample Ingest Workflow",
  "identifier": "SAMPLE_PROCESS_SIP_UNITARY",
  "typeProc": "INGEST",
  "comment": "Sample Ingest Workflow V6",
  "steps": [
    {
      "workerGroupId": "DefaultWorker",
      "stepName": "STP_INGEST_CONTROL_SIP",
      "behavior": "BLOCKING",
      "distribution": {
        "kind": "REF"
      },
      "actions": [
        {
          "action": {
            "actionKey": "HELLO_WORLD_PLUGIN",
            "behavior": "BLOCKING",
            "in": [
              {
                "name": "var_name",
                "uri": "VALUE:Hello World"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Avertissement : Le fichier *workflow* doit être un fichier json avec comme extension (`.json`) sinon le fichier ne sera pas pris en compte.

8.2.8.1.2.1 Comment ajouter un nouveau workflow dans vitam ?

Il tout d'abord créer un fichier json avec un nom de votre choix et ayant la forme de l'exemple ci-dessus. Veuillez vous référer au différents workflow existants pour avoir plus d'information.

Il faut ensuite copier ce fichier (`CustomWorkflow.json`) dans :

- En production : Manuellement dans le dossier `/vitam/conf/processing/workflows/`
- Via ansible : Dans le dossier `~/vitam/deployment/ansible-vitam/roles/vitam/files/processing/workflows/`

8.2.8.1.2.2 Comment ajouter la traduction de clés des Plugins ?

On peut dans n'importe quel service vitam ajouter dans le dossier conf les deux fichiers suivants : - `vitam-logbook-messages_fr.properties` - `vitam-error-messages.properties`

Ces deux fichiers garantissent la traduction des clés.

Pour le nouveau plugins ajouté, le fichier properties qui est à l'intérieur du jar n'est visible que par le service (worker). Pour qu'on puisse avoir ces clés traduites,le fichier `vitam-logbook-messages_fr.properties` doit contenir les traductions des clés de ce nouveau *plugin*. Il faut copier ce fichier dans le dossier de conf de processing s'il n'existe pas.

```
HELLO_WORLD_PLUGIN=Test d''un plugin Hello World
HELLO_WORLD_PLUGIN.OK=Test d''un plugin Hello World réalisé avec succès
HELLO_WORLD_PLUGIN.KO=Échec lors du test d''un plugin Hello World
HELLO_WORLD_PLUGIN.FATAL=Erreur fatale lors du test d''un plugin Hello World
HELLO_WORLD_PLUGIN.WARNING=Avertissement lors du test d''un plugin Hello␣
↪World
```

8.2.8.1.2.3 Comment appeler le nouveau workflow ?

En utilisant l'API d'*ingest* et en passant les paramètres suivants :

- `X_CONTEXT_ID` : l'identifier de votre workflow (dans l'exemple ci-dessus SAMPLE_PROCESS_SIP_UNITARY)
- `X_ACTION` : votre action (RESUME, NEXT)

Le reste se fait automatiquement par le *back office*.

8.2.8.1.2.4 Remarques

- L'ajout d'un *workflow* dans processing en production ne nécessite pas de redémarrage. Un thread passe chaque heure configurale pour recharger les derniers workflow (ajoutés ou modifiés)
- L'ajout d'un jar dans les workers et les fichiers properties nécessitent, cependant, le redémarrage des workers et des services concernés.

8.2.8.1.2.5 Securité

Les plugins externes sont exécutés au même niveau de sécurité que ceux interne à *VITAM*. L'isolation de l'exécution des plugins externes n'est pas assurée par *VITAM*. C'est donc à l'exploitant de garantir la sécurité des plugins intégrés.

8.2.9 ihm-demo

8.2.9.1 Présentation

Cette IHM a été développée pour des fins de tests de VITAM.

Rôle :

- Permettre une utilisation basique de VITAM, notamment sans SIA

Fonctions :

- Représentation des arborescences et des graphes
- Formulaires dynamiques
- Suivi des opérations
- Gestion des référentiels

8.2.9.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ihm-demo`.

8.2.9.2.1 Fichier `access-external-client.conf`

Ce fichier permet de définir l'URL d'accès au service access-external.

```
serverHost: {{ vitam.accessexternal.host }}
serverPort: {{ vitam.accessexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_demo }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: true
```

8.2.9.2.2 Fichier `ihm-demo.conf`

```
serverHost: {{ ip_service }}
port: {{ vitam_struct.port_service }}

baseUrl: "{{ vitam_struct.baseurl }}"
staticContent: {{ vitam_struct.static_content }}
baseUri: /{{ vitam_struct.baseuri }}
secureMode:
{% for realm in vitam_struct.authentication_realms %}
- {{ realm }}
{% endfor %}

jettyConfig: jetty-config.xml
authentication: true
enableXsrFilter: true
enableSession: true

allowedMediaTypes:
{% for mediaType in vitam_struct.allowedMediaTypes %}
    - type: {{ mediaType.type }}
      subtype: {{ mediaType.subtype }}
{% endfor %}
```

tenants : liste des tenants disponibles sur l'ihm-demo.

8.2.9.2.3 Fichier `ingest-external-client.conf`

```
serverHost: {{ vitam.ingestexternal.host }}
serverPort: {{ vitam.ingestexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_demo }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: true
```

8.2.9.2.4 Fichier `shiro.ini`

```
# =======================
# Shiro INI configuration
# =======================

[main]
# Objects and their properties are defined here,
# Such as the securityManager, Realms and anything
# else needed to build the SecurityManager


# Cache Manager
builtInCacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager

# Security Manager
securityManager.cacheManager = $builtInCacheManager

sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager = $sessionManager
securityManager.sessionMode = native
securityManager.sessionManager.globalSessionTimeout = {{ vitam_struct.session_timeout␣
↪}}
securityManager.sessionManager.sessionIdUrlRewritingEnabled = false
securityManager.sessionManager.sessionIdCookie.secure = {{ vitam_struct.secure_cookie␣
↪}}
securityManager.rememberMeManager.cookie.secure = {{ vitam_struct.secure_cookie }}
securityManager.rememberMeManager.cookie.httpOnly = true

# Notice how we didn't define the class for the FormAuthenticationFilter ('authc') -␣
↪it is instantiated and available already:
authc.loginUrl = /#!/login


# credentialsMatcher
sha256Matcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher

{% if "iniRealm" in vitam_struct.authentication_realms %}
iniRealm.credentialsMatcher = $sha256Matcher
{% endif %}

{% if "ldapRealm" in vitam_struct.authentication_realms %}
contextFactory = org.apache.shiro.realm.ldap.JndiLdapContextFactory
```

(suite sur la page suivante)

(suite de la page précédente)

```
contextFactory.url = {{ ldap_authentification.ldap_protocol }}://{{ ldap_
↪authentification.ldap_server }}:{{ ldap_authentification.ldap_port }}
{% if ldap_authentification.ldap_login is defined and ldap_authentification.ldap_pwd␣
↪is defined %}
{% if ldap_authentification.ldap_login != "" and ldap_authentification.ldap_pwd != ""
↪%}
contextFactory.systemUsername = {{ ldap_authentification.ldap_login }}
contextFactory.systemPassword = {{ ldap_authentification.ldap_pwd }}
{% endif %}
{% endif %}

ldapRealm = fr.gouv.vitam.common.auth.core.realm.LdapRealm
ldapRealm.ldapContextFactory = $contextFactory
ldapRealm.searchBase = "{{ ldap_authentification.ldap_base }}"
ldapRealm.groupRequestFilter = {{ ldap_authentification.ldap_group_request }}
ldapRealm.userDnTemplate = {{ ldap_authentification.ldap_userDn_Template }}
ldapRealm.groupRolesMap = "{{ ldap_authentification.ldap_admin_group }}":"admin", "{{␣
↪ldap_authentification.ldap_user_group }}":"user", "{{ ldap_authentification.ldap_
↪guest_group }}":"guest"
{% endif %}

x509 = fr.gouv.vitam.common.auth.web.filter.X509AuthenticationFilter

x509.useHeader = False

x509credentialsMatcher = fr.gouv.vitam.common.auth.core.authc.
↪X509CredentialsSha256Matcher

{% if "x509Realm" in vitam_struct.authentication_realms %}
x509Realm = fr.gouv.vitam.common.auth.core.realm.X509KeystoreFileWithRoleRealm
x509Realm.grantedKeyStoreName = {{ vitam_folder_conf }}/grantedstore_ihm-demo.jks
x509Realm.grantedKeyStorePassphrase = {{ password_grantedstore }}
x509Realm.trustedKeyStoreName = {{ vitam_folder_conf }}/truststore_ihm-demo.jks
x509Realm.trustedKeyStorePassphrase = {{ password_truststore }}
x509Realm.credentialsMatcher = $x509credentialsMatcher
x509Realm.certificateDnRoleMapping = "CN=userAdmin,O=Vitam,L=Paris":"admin",
↪"CN=userUser,O=Vitam,L=Paris,C=FR":"user"
{% endif %}

securityManager.realms = {% for realm in vitam_struct.authentication_realms %}{% if␣
↪not loop.first %},{% endif %}${{ realm }}{% endfor %}

{% if "iniRealm" in vitam_struct.authentication_realms %}

[users]
# The 'users' section is for simple deployments
# # when you only need a small number of statically-defined
# # set of User accounts.
# #username = password
{% for item in vitam_users %}
 {{ item.login }}={{ item.password|hash('sha256') }}, {{ item.role }}
{% endfor %}

{% endif %}


[roles]
```

(suite sur la page suivante)

(suite de la page précédente)

```
admin = *
user = messages:*, archivesearch:*, logbook:*, ingest:*, archiveupdate:*,␣
↪archiveunit:*, ingests:read, admin:formats:read, admin:rules:read, admin:accession-
↪register:read, logbookunitlifecycles:*, logbookobjectslifecycles:*, clear:delete,␣
↪check:read, traceability:content:read, accesscontracts:read, profiles:read,␣
↪contracts:read, contexts:read, archiveunitprofiles:read, ontologies:read,␣
↪accessionregisterssymbolic:read
guest = archivesearch:*, archiveunit:*, units:*, unit:*, admin:accession-
↪register:read, accesscontracts:read


[urls]
# make sure the end-user is authenticated.  If not, redirect to the 'authc.loginUrl'␣
↪above,
# and after successful authentication, redirect them back to the original account␣
↪page they
# were trying to view:
/v1/api/login = anon
/v1/api/logout = logout
/v1/api/messages/logbook = anon
/v1/api/tenants = anon
/v1/api/securemode = anon
/v1/api/admintenant = anon
/v1/api/permissions = x509
/v1/api/** = authc, x509
/#/** = authc
```

8.2.9.3 Configuration de apache shiro

8.2.9.3.1 Présentation authentification via LDAP et via certificat

Afin de pouvoir authentifier des clients via une base de données LDAP il suffit de bien configurer shiro. Pour ce faire, Vitam utilise le fichier shiro.ini qui a la forme suivante.

```
[main]
contextFactory = org.apache.shiro.realm.ldap.JndiLdapContextFactory
contextFactory.url = ldap://localhost:389
contextFactory.systemUsername = cn=admin,dc=example,dc=org
contextFactory.systemPassword = password
realm = fr.gouv.vitam.common.security.rest.LdapRealm
realm.ldapContextFactory = $contextFactory
realm.searchBase = "dc=example,dc=org"
realm.groupRequestFilter = (&(objectClass=groupOfNames)(member={0}))
realm.userDnTemplate = uid={0},dc=example,dc=org
realm.groupRolesMap = "cn=gadmins,dc=example,dc=org":"admin", "cn=gusers,dc=example,
↪dc=org":"user", "cn=gadmins,dc=example,dc=org":"guest"
securityManager.realms = $realm
```

```
x509 = fr.gouv.vitam.common.auth.web.filter.X509AuthenticationFilter
x509.useHeader = false
x509credentialsMatcher = fr.gouv.vitam.common.auth.core.authc.
↪X509CredentialsSha256Matcher
x509Realm = fr.gouv.vitam.common.auth.core.realm.X509KeystoreFileWithRoleRealm
x509Realm.grantedKeyStoreName = /vitam/conf/ihm-demo/grantedstore_ihm-demo.jks
```

(suite sur la page suivante)

(suite de la page précédente)

```
x509Realm.grantedKeyStorePassphrase = azerty12
x509Realm.trustedKeyStoreName = /vitam/conf/ihm-demo/truststore_ihm-demo.jks
x509Realm.trustedKeyStorePassphrase = azerty10
x509Realm.credentialsMatcher = $x509credentialsMatcher
x509Realm.certificateDnRoleMapping = "CN=userAdmin,O=Vitam,L=Paris":"admin",
↪"CN=userUser,O=Vitam,L=Paris,C=FR":"user"
securityManager.realms = $x509Realm
```

8.2.9.3.2 Décryptage de `shiro.ini`

`[main]` Contient la déclaration des options et mappings dans l'authentication ldap :

- contextFactory.url : url du serveur ldap ;
- contextFactory.systemUsername : identifiant de l'utilisateur ;
- contextFactory.systemPassword : mot de passe ;
- realm.searchBase : le domaine de recherche dans LDAP ;
- realm.groupRequestFilter : chaque utilisateur est déclaré dans un groupe, cette requête sert à chercher les groupes de l'utilisateur ;
- realm.userDnTemplate : le modèle pour traduire un identifiant de l'utilisateur en DN (distinguished name) dans ldap ;
- realm.groupRolesMap : le mapping entre le DN des group de l'utilisateur et les rôles dans ihm ;
- x509Realm.grantedKeyStoreName : le fichier grantedstore ;
- x509Realm.trustedKeyStoreName : le fichier trustedstore ;
- x509Realm.certificateDnRoleMapping : le mapping entre le DisplayName de certificat et les rôles dans ihm.

Note : on peut déclarer plusieurs groups qui ont la même rôle admin avec cette syntaxe :

```
"groupeA" : "admin", "groupeB" : "admin", "groupeC" : "admin"
```

8.2.9.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ihm-demo`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ihm-demo`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/ihm-demo/v1/status`

- Gestion des utilisateurs

Les utilisateurs sont actuellement gérés via le fichier `shiro.ini`, dans la section `[users]`.

- Créer un utilisateur

  Lancer la commande shell suivante pour générer le mot de passe :

```
echo -n <motdepasse> | sha256sum
```

Copier le résultat.
Ensuite, éditer le fichier `/vitam/conf/ihm-demo/shiro.ini` et ajouter, dans la section `[users]`, la ligne suivante :

```
<login de l'utilisateur>=<résultat de la commande de génération de mot de␣
↪passe précédente>
```

Pour terminer, relancer le service `vitam-ihm-demo` par la commande :

```
systemctl restart vitam-ihm-demo
```

- Supprimer un utilisateur

Dans la section `[users]`, enlever la ligne correspondant à l'utilisateur à supprimer. Pour terminer, relancer le service `vitam-ihm-demo` par la commande :

```
systemctl restart vitam-ihm-demo
```

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

8.2.10 ihm-recette

8.2.10.1 Présentation

Cette IHM a été développée pour des fins de validation de VITAM. Elle permet de réaliser des tests de non-régression, mais également des actions sur le contenu des bases de données.

Danger : Cette IHM ne doit PAS être déployée dans un environnement de production !

8.2.10.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ihm-recette`.

8.2.10.2.1 Fichier `access-external-client.conf`

Ce fichier permet de définir l'URL d'accès au service access-external.

```
serverHost: {{ vitam.accessexternal.host }}
serverPort: {{ vitam.accessexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_recette }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: false
```

8.2.10.2.2 Fichier `driver-location.conf`

```
driverLocation: {{ vitam_folder_lib }}
```

8.2.10.2.3 Fichier `driver-mapping.conf`

```
driverMappingPath: {{ vitam_folder_data }}/
delimiter: ;
```

8.2.10.2.4 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.10.2.5 Fichier `ihm-recette-client.conf`

```
serverHost: {{ vitam_struct.host }}
serverPort: {{ vitam_struct.port_service }}
```

8.2.10.2.6 Fichier `ihm-recette.conf`

```
serverHost: {{ ip_service }}
port: {{ vitam_struct.port_service }}

baseUrl: "/{{ vitam_struct.baseuri }}"
baseUri: "/{{ vitam_struct.baseuri }}"
staticContent: "{{ vitam_struct.static_content }}"

jettyConfig: jetty-config.xml
authentication: true
enableXsrFilter: true
enableSession: true

secureMode:
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% for securemode in vitam_struct.secure_mode %}
- {{ securemode }}
{% endfor %}
sipDirectory: {{ vitam_folder_data }}/test-data
performanceReportDirectory: {{ vitam_folder_data }}/report/performance

testSystemSipDirectory: {{ vitam_folder_data }}/test-data/system
testSystemReportDirectory: {{ vitam_folder_data }}/report/system
ingestMaxThread: {{ ansible_processor_cores * ansible_processor_threads_per_core + 1 }
↪}

#
workspaceUrl: {{vitam.workspace | client_url}}

# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
# Actually need this field for compatibility
dbName: admin
# @integ: parametrize it !
masterdataDbName: masterdata
logbookDbName: logbook
metadataDbName: metadata
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data']['admin']['user'] }}
dbPassword: {{ mongodb['mongo-data']['admin']['password'] }}

# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  httpPort: {{ elasticsearch.data.port_http }}
{% endfor %}

# ElasticSearch External Metadata Mapping
elasticsearchExternalMetadataMappings:
- collection: Unit
  mappingFile: {{ vitam.ihm_recette.elasticsearch_mapping_dir }}/unit-es-mapping.json
- collection: ObjectGroup
  mappingFile: {{ vitam.ihm_recette.elasticsearch_mapping_dir }}/og-es-mapping.json

# Functional Admin Configuration
functionalAdminAdmin:
   functionalAdminServerHost: {{ vitam.functional_administration.host }}
   functionalAdminServerPort: {{ vitam.functional_administration.port_admin }}
   adminBasicAuth:
      userName: {{ admin_basic_auth_user }}
      password: {{ admin_basic_auth_password }}

# ES index configuration
functionalAdminIndexationSettings:
  default_config:
    number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪masterdata.number_of_shards }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
    number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪masterdata.number_of_replicas }}

{% for collection in ["accesscontract", "accessionregisterdetail",
↪"accessionregistersummary", "accessionregistersymbolic", "agencies",
↪"archiveunitprofile", "context", "fileformat", "filerules", "griffin",
↪"ingestcontract", "managementcontract", "ontology", "preservationscenario", "profile
↪", "securityprofile"] %}
{% if vitam_elasticsearch_tenant_indexation.masterdata[collection] is defined %}
  {{collection}}:
{% if vitam_elasticsearch_tenant_indexation.masterdata[collection].number_of_shards␣
↪is defined %}
    number_of_shards: {{ vitam_elasticsearch_tenant_indexation.masterdata[collection].
↪number_of_shards }}
{% endif %}
{% if vitam_elasticsearch_tenant_indexation.masterdata[collection].number_of_replicas␣
↪is defined %}
    number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.
↪masterdata[collection].number_of_replicas }}
{% endif %}
{% endif %}
{% endfor %}

metadataIndexationSettings:

  default_config:
    unit:
      number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.unit.
↪number_of_shards }}
      number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪unit.number_of_replicas }}
    objectgroup:
      number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪objectgroup.number_of_shards }}
      number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪objectgroup.number_of_replicas }}

{% if vitam_elasticsearch_tenant_indexation.dedicated_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.dedicated_tenants is not none %}
  dedicated_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.dedicated_tenants %}
{% if (entry.unit is defined and (entry.unit.number_of_shards is defined or entry.
↪unit.number_of_replicas is defined)) or
      (entry.objectgroup is defined and (entry.objectgroup.number_of_shards is␣
↪defined or entry.objectgroup.number_of_replicas is defined)) %}
  - tenants: '{{ entry.tenants }}'
{% if entry.unit is defined %}
    unit:
{% if entry.unit.number_of_shards is defined %}
      number_of_shards: {{ entry.unit.number_of_shards }}
{% endif %}
{% if entry.unit.number_of_replicas is defined %}
      number_of_replicas: {{ entry.unit.number_of_replicas }}
{% endif %}
{% endif %}
{% if entry.objectgroup is defined %}
    objectgroup:
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% if entry.objectgroup.number_of_shards is defined %}
      number_of_shards: {{ entry.objectgroup.number_of_shards }}
{% endif %}
{% if entry.objectgroup.number_of_replicas is defined %}
      number_of_replicas: {{ entry.objectgroup.number_of_replicas }}
{% endif %}
{% endif %}
{% endif %}
{% endfor %}
{% endif %}

{% if vitam_elasticsearch_tenant_indexation.grouped_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.grouped_tenants is not none %}
  grouped_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.grouped_tenants %}
{% if (entry.unit is defined and (entry.unit.number_of_shards is defined or entry.
↪unit.number_of_replicas is defined)) or
      (entry.objectgroup is defined and (entry.objectgroup.number_of_shards is␣
↪defined or entry.objectgroup.number_of_replicas is defined)) %}
  - name: '{{ entry.name }}'
    tenants: '{{ entry.tenants }}'
{% if entry.unit is defined %}
    unit:
{% if entry.unit.number_of_shards is defined %}
      number_of_shards: {{ entry.unit.number_of_shards }}
{% endif %}
{% if entry.unit.number_of_replicas is defined %}
      number_of_replicas: {{ entry.unit.number_of_replicas }}
{% endif %}
{% endif %}
{% if entry.objectgroup is defined %}
    objectgroup:
{% if entry.objectgroup.number_of_shards is defined %}
      number_of_shards: {{ entry.objectgroup.number_of_shards }}
{% endif %}
{% if entry.objectgroup.number_of_replicas is defined %}
      number_of_replicas: {{ entry.objectgroup.number_of_replicas }}
{% endif %}
{% endif %}
{% endif %}
{% endfor %}
{% endif %}

logbookIndexationSettings:
  default_config:
    logbookoperation:
      number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪logbookoperation.number_of_shards }}
      number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪logbookoperation.number_of_replicas }}

{% if vitam_elasticsearch_tenant_indexation.dedicated_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.dedicated_tenants is not none %}
  dedicated_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.dedicated_tenants %}
{% if (entry.logbookoperation is defined and (entry.logbookoperation.number_of_shards␣
↪is defined or entry.logbookoperation.number_of_replicas is defined)) %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
  - tenants: '{{ entry.tenants }}'
    logbookoperation:
{% if entry.logbookoperation.number_of_shards is defined %}
      number_of_shards: {{ entry.logbookoperation.number_of_shards }}
{% endif %}
{% if entry.logbookoperation.number_of_replicas is defined %}
      number_of_replicas: {{ entry.logbookoperation.number_of_replicas }}
{% endif %}
{% endif %}
{% endfor %}
{% endif %}

{% if vitam_elasticsearch_tenant_indexation.grouped_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.grouped_tenants is not none %}
  grouped_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.grouped_tenants %}
{% if (entry.logbookoperation is defined and (entry.logbookoperation.number_of_shards␣
↪is defined or entry.logbookoperation.number_of_replicas is defined)) %}
  - name: '{{ entry.name }}'
    tenants: '{{ entry.tenants }}'
    logbookoperation:
{% if entry.logbookoperation.number_of_shards is defined %}
      number_of_shards: {{ entry.logbookoperation.number_of_shards }}
{% endif %}
{% if entry.logbookoperation.number_of_replicas is defined %}
      number_of_replicas: {{ entry.logbookoperation.number_of_replicas }}
{% endif %}
{% endif %}
{% endfor %}
{% endif %}
```

8.2.10.2.7 Fichier `ingest-external-client.conf`

```
serverHost: {{ vitam.ingestexternal.host }}
serverPort: {{ vitam.ingestexternal.port_service }}
secure: true
sslConfiguration :
 keystore :
  - keyPath: {{ vitam_folder_conf }}/keystore_{{ vitam_struct.vitam_component }}.p12
    keyPassword: {{ keystores.client_external.ihm_recette }}
 truststore :
  - keyPath: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.jks
    keyPassword: {{ truststores.client_external }}
hostnameVerification: false
```

8.2.10.2.8 Fichier `shiro.ini`

```
[main]

{% if vitam_struct.secure_mode == 'x509' %}
x509 = fr.gouv.vitam.common.auth.web.filter.X509AuthenticationFilter
```

(suite sur la page suivante)

(suite de la page précédente)

```
x509.useHeader = {{ vitam_ssl_user_header }}

x509credentialsMatcher = fr.gouv.vitam.common.auth.core.authc.
↪X509CredentialsSha256Matcher

x509Realm = fr.gouv.vitam.common.auth.core.realm.X509KeystoreFileRealm
x509Realm.grantedKeyStoreName = {{ vitam_folder_conf }}/grantedstore_ihm-recette.jks
x509Realm.grantedKeyStorePassphrase = {{ password_grantedstore }}
x509Realm.trustedKeyStoreName = {{ vitam_folder_conf }}/truststore_ihm-recette.jks
x509Realm.trustedKeyStorePassphrase = {{ password_truststore }}
x509Realm.credentialsMatcher = $x509credentialsMatcher
securityManager.realm = $x509Realm
securityManager.subjectDAO.sessionStorageEvaluator.sessionStorageEnabled = false
[urls]
/v1/api/** = x509

{% else %}
# Objects and their properties are defined here,
# Such as the securityManager, Realms and anything
# else needed to build the SecurityManager
# credentialsMatcher
sha256Matcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher
iniRealm.credentialsMatcher = $sha256Matcher
# Cache Manager
builtInCacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
# Security Manager
securityManager.cacheManager = $builtInCacheManager
sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager = $sessionManager
securityManager.sessionMode=native
securityManager.sessionManager.globalSessionTimeout = {{ vitam_struct.session_timeout␣
↪}}
securityManager.sessionManager.sessionIdUrlRewritingEnabled = false
securityManager.sessionManager.sessionIdCookie.secure = {{ vitam_struct.secure_cookie␣
↪}}
securityManager.rememberMeManager.cookie.secure = {{ vitam_struct.secure_cookie }}
securityManager.rememberMeManager.cookie.httpOnly = true
# Notice how we didn't define the class for the FormAuthenticationFilter ('authc') -␣
↪it is instantiated and available already:
authc.loginUrl = /#!/login
[users]
# The 'users' section is for simple deployments
# when you only need a small number of statically-defined
# set of User accounts.
#username = password
{% for item in vitam_users %}
{% if item.role == "admin" %}
{{ item.login }}={{ item.password|hash('sha256') }}
{% endif %}
{% endfor %}
[roles]
# The 'roles' section is for simple deployments
# when you only need a small number of statically-defined
# roles.
[urls]
# make sure the end-user is authenticated.  If not, redirect to the 'authc.loginUrl'␣
↪above,
```

(suite sur la page suivante)

(suite de la page précédente)

```
# and after successful authentication, redirect them back to the original account␣
↪page they
# were trying to view:
/v1/api/login = anon
/v1/api/logout = logout
/v1/api/securemode = anon
/** = authc

{% endif %}
```

8.2.10.2.9 Fichier `static-offer.json`

```
{% if vitam.storageofferdefault.https_enabled==true %}
    {% set protocol = 'https' %}
{% else %}
    {% set protocol = 'http' %}
{% endif %}
[
{% for item in all_used_offers %}
{
{% if item.id is defined %}
    "id" : "{{ item.id }}",
{% else %}
    "id" : "{{ item.name }}.service.{{ item.vitam_site_name |default(vitam_site_name)␣
↪}}.{{ consul_domain }}",
{% endif %}
    "baseUrl" : "{{ protocol }}://{{ item.name }}.service.{{ item.vitam_site_name␣
↪|default(vitam_site_name) }}.{{ consul_domain }}:{{ vitam.storageofferdefault.port_
↪service }}",
    {% if item.asyncRead is defined %} "asyncRead": {{ item.asyncRead|lower }}, {%␣
↪endif %}
    "parameters" : {
        {% if vitam.storageofferdefault.https_enabled==true %}
        "keyStore-keyPath": "{{ vitam_folder_conf }}/keystore_storage.p12",
        "keyStore-keyPassword": "{{ keystores.client_storage.storage }}",
        "trustStore-keyPath": "{{ vitam_folder_conf }}/truststore_storage.jks",
        "trustStore-keyPassword": "{{ truststores.client_storage }}"
        {% endif %}
    }
}
{% if not loop.last %},
{% endif %}
{% endfor %}
]
```

8.2.10.2.10 Fichier `static-strategy.json`

```
[
    {
        "id" : "default",
        "offers" : [
{% for item in vitam_strategy %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% if item.id is defined %}
            {
                "id" : "{{ item.id }}"{% if item.referent | default(false) | bool ==␣
↪true %}, "referent" : true{% endif %}{% if item.status is defined %}, "status" : "{
↪{ item.status | upper }}" {% endif %}{% if item.rank is defined %}, "rank" : "{{␣
↪item.rank }}"{% endif %}
            }{% if not loop.last %},{% endif %}
{% else %}
            {
                "id" : "{{ item.name }}.service.{{ item.vitam_site_name |␣
↪default(vitam_site_name) }}.{{ consul_domain }}"{% if item.referent |␣
↪default(false) | bool == true %}, "referent" : true{% endif %}{% if item.status is␣
↪defined %}, "status" : "{{ item.status | upper }}" {% endif %}{% if item.rank is␣
↪defined %}, "rank" : "{{ item.rank }}"{% endif %}
            }{% if not loop.last %},{% endif %}
{% endif %}
{% endfor %}
        ]
    }
{% if other_strategies is defined %}
{% for strategy_name, strategy_offers in other_strategies.items() %}
    ,
    {
        "id" : "{{ strategy_name }}",
        "offers" : [
{% for strategy_offer in strategy_offers %}
            {
              "id" : "{{ strategy_offer.name }}.service.{{ strategy_offer.vitam_
↪site_name | default(vitam_site_name) }}.{{ consul_domain }}"{% if strategy_offer.
↪referent | default(false) | bool == true %}, "referent" : true{% endif %}{% if␣
↪strategy_offer.status is defined %}, "status" : "{{ strategy_offer.status | upper }}
↪" {% endif %}{% if strategy_offer.rank is defined %}, "rank" : "{{ strategy_offer.
↪rank }}"{% endif %}
            }{% if not loop.last %},{% endif %}
{% endfor %}
        ]
    }
{% endfor %}
{% endif %}

]
```

8.2.10.2.11 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.10.2.12 Fichier `storage.conf`

```
urlWorkspace: {{ vitam.workspace | client_url }}
timeoutMsPerKB: 100
jettyConfig: jetty-config.xml
```

(suite sur la page suivante)

(suite de la page précédente)

```
zippingDirecorty: {{ vitam_folder_data }}/storage_archives
loggingDirectory: {{ vitam_folder_log }}
```

8.2.10.2.13 Fichier `storage-offer.conf`

```
strategy_name=[{% for item in vitam_strategy %}"{{ item.name }}.service.{{ consul_
↪domain }}"{% if not loop.last %},{% endif %}{% endfor %}]
```

8.2.10.2.14 Fichier `tnr.conf`

```
urlWorkspace: {{vitam.workspace | client_url}}
tenantsTest: [ "0" ]
vitamSecret: {{ plateforme_secret }}
tenants: [ "{{ vitam_tenant_ids | join('", "') }}" ]
adminTenant: {{ vitam_tenant_admin }}
```

8.2.10.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ihm-recette`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ihm-recette`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Gestion des utilisateurs

Les utilisateurs sont actuellement gérés via le fichier `shiro.ini`, dans la section `[users]`.

- Créer un utilisateur

Lancer la commande shell suivante pour générer le mot de passe :

```
echo -n <motdepasse> | sha256sum
```

Copier le résultat.

Ensuite, éditer le fichier `/vitam/conf/ihm-recette/shiro.ini` et ajouter, dans la section `[users]`, la ligne suivante :

```
<login de l'utilisateur>=<résultat de la commande de génération de mot de␣
↪passe précédente>
```

Pour terminer, relancer le service `vitam-ihm-recette` par la commande :

```
systemctl restart vitam-ihm-recette
```

- Supprimer un utilisateur

Dans la section `[users]`, enlever la ligne correspondant à l'utilisateur à supprimer. Pour terminer, relancer le service `vitam-ihm-recette` par la commande :

```
systemctl restart vitam-ihm-recette
```

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

8.2.11 Ingest

8.2.11.1 Introduction

Ce document présente les configurations pour utiliser les différents modules de *ingest*.

8.2.11.2 ingest-external

8.2.11.2.1 Présentation

Ingest-external est le composant d'interface entre *VITAM* et un *SIA* client, permettant de réaliser des entrées d'archives dans *VITAM*.

Rôle :

- Exposer les API publiques du système
- Sécuriser l'accès aux API de VITAM

8.2.11.2.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ingest-external`.

8.2.11.2.2.1 Fichier `ingest-external.conf`

```
path: {{ vitam_folder_data }}
jettyConfig: jetty-config.xml
authentication: false
tenantFilter : true
antiVirusScriptName: scan-{{ vitam_struct.antivirus }}.sh
timeoutScanDelay: {{ vitam_struct.scantimeout | default(1200000) }}
baseUploadPath: {{ vitam_struct.upload_dir | default('/vitam/data/ingest-external/
↪upload') }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
successfulUploadDir: {{ vitam_struct.success_dir | default('/vitam/data/ingest-
↪external/upload/success') }}
failedUploadDir: {{ vitam_struct.fail_dir | default('/vitam/data/ingest-external/
↪upload/failure') }}
fileActionAfterUpload: {{ vitam_struct.upload_final_action | default('MOVE') }}
```

Ce fichier contient un appel au shell d'antivirus (par défaut, ClamAV) ; se reporter au *DIN*.

Il est possible, dans le cas de fichiers SIP volumineux, d'héberger des fichiers directement dans ingest-external (valeur de la directive `baseUploadPath`). Ces fichiers doivent être accessibles et utilisables par le *user* système vitam.

Les options associées à cette fonctionnalité peuvent être paramétrées dans le fichier deployment/environment/group_vars/all/vitam_vars.yml avant installation de Vitam.

La directive `fileActionAfterUpload` accepte les valeurs :

- `NONE` : le fichier reste
- `MOVE` : déplace le fichiers vers les valeurs des directives `successfulUploadDir` (en cas de succès de l'ingest) et `failedUploadDir` (en cas de non-succès de l'ingest)
- `DELETE` : le fichier est supprimé en cas de succès de l'ingest uniquement

A charge à l'exploitant de bien gérer l'espace disque de ces répertoires (il faut penser aux ingests en échecs par exemple).

Se reporter au manuel de développement pour l'appel d'API associé.

8.2.11.2.2.2 Fichier `ingest-internal-client.conf`

```
serverHost: {{ vitam.ingestinternal.host }}
serverPort: {{ vitam.ingestinternal.port_service }}
```

8.2.11.2.2.3 Fichier `internal-security-client.conf`

```
serverHost: {{ vitam.security_internal.host }}
serverPort: {{ vitam.security_internal.port_service }}
secure: {{ vitam.security_internal.https_enabled }}
```

8.2.11.2.2.4 Fichier `format-identifiers.conf`

```
siegfried-local:
  type: SIEGFRIED
  client: http
  host: localhost
  port: {{ siegfried.port }}
  rootPath: {{ vitam_folder_data }}/
  versionPath: {{ vitam_folder_data }}/version/folder
```

8.2.11.2.2.5 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.11.2.2.6 Fichier `scan-clamav.sh`

Ce script de *scan* appelle l'antivirus (par défaut, clamAV ; ce paramètre est surchargeable à l'installation ; se référer au :term'DIN' pour plus de précisions) pour détecter les virus.

```
#!/bin/sh

#################################################################################
# Role:                                                                         #
# Scan a single file using clamav anti-virus                                    #
#################################################################################
# Args:                                                                         #
# - file to scan                                                                #
#################################################################################
# Return:                                                                       #
# - 0: scan OK - no virus                                                       #
RET_NOTVIRUS=0
# - 1: virus found and corrected                                                #
RET_VIRUS_FOUND_FIXED=1
# - 2: virus found but not corrected                                            #
RET_VIRUS_FOUND_NOTFIXED=2
# - 3: Fatal scan not performed                                                 #
RET_FAILURE=3
# stdout : names of virus found (1 per line) if virus found ;                   #
#          failure description if failure                                       #
# stderr : full ouput of clamav                                                 #
#################################################################################

# Default return code : scan NOK
RET=3
OUTPUT_DIR=$(mktemp -d)
if [ $# -ne 1 ]; then # Argument number must be one
  echo "ERROR : $# parameter(s) provided, only one parameter is needed"
else # one argument, let's go
  if [ ! -f "$1" ];then # if the file wich will be scan is existing, keep going
    echo "ERROR : \"$1\" doesn't exit"
  else
    clamdscan -z --stream "$1" 1> ${OUTPUT_DIR}/stdout 2> ${OUTPUT_DIR}/stderr #␣
↪scanning the file and store the output OUTPUT
    RET=$? # return code of clamscan

    # Always output clamscan outputs to our own stderr
    (>&2 cat ${OUTPUT_DIR}/stdout  ${OUTPUT_DIR}/stderr)

    if [ ${RET} -eq ${RET_VIRUS_FOUND_FIXED} ] ; then
      RET=2 # if virus found clamscan return 1; the script must return 2
      (>&1 cat ${OUTPUT_DIR}/stdout  | grep `basename ${1}` | cut -d ' ' -f 2) #␣
↪sending the list of virus to our own stdout
    elif [ ${RET} -eq 2 ] ; then
      RET=3 # if scan not performed clamscan return 2; the script must return 3
      (>&1 cat ${OUTPUT_DIR}/stdout  | grep `basename ${1}` | cut -d ' ' -f 2-) #␣
↪sending the failure reason to our own stdout
```

(suite sur la page suivante)

(suite de la page précédente)

```
    fi

    if [ -f "${OUTPUT_DIR}/stdout" ]
    then
      rm ${OUTPUT_DIR}/stdout
    fi
    if [ -f "${OUTPUT_DIR}/stderr" ]
    then
      rm ${OUTPUT_DIR}/stderr
    fi
  fi
fi
rmdir ${OUTPUT_DIR}
exit ${RET}
```

8.2.11.2.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ingest-external`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ingest-external`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/ingest-external/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.11.3 ingest-internal

8.2.11.3.1 Présentation

Rôle :

- Permettre l'entrée d'une archive SEDA dans le SAE

Fonctions :

- Upload HTTP de fichiers au format SEDA

- Sas de validation antivirus des fichiers entrants
- Persistance du SEDA dans workspace
- Lancement des workflows de traitements liés à l'entrée dans processing

8.2.11.3.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/ingest-internal`.

8.2.11.3.2.1 Fichier `ingest-internal.conf`

```
workspaceUrl: {{vitam.workspace | client_url}}
processingUrl: {{vitam.processing | client_url}}
jettyConfig: jetty-config.xml
```

Ce fichier précise les URLs pour les services « Processing » et « Workspace », et la configuration du serveur jetty.

8.2.11.3.2.2 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.11.3.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-ingest-internal`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-ingest-internal`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/ingest-internal/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.12 Security-Internal

8.2.12.1 Introduction

Ce document présente la configuration pour le module security-internal.

8.2.12.2 security-internal-exploitation

Ce document spécifie la configuration (fichiers de config) pour lancer le services de `security-internal`.

8.2.12.2.1 Fichier `security-internal.conf`

Ce fichier permet de définir la configuration du serveur MongoDB, du serveur jetty, les tenants, ainsi que la configuration de l'authentification *personae* pour les permissions des *endpoints* externes de *VITAM*.

```
# Configuration MongoDB
mongoDbNodes:
{% for host in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[host]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: identity
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].securityInternal.user }}
dbPassword: {{ mongodb['mongo-data'].securityInternal.password }}

jettyConfig: jetty-config.xml

personalCertificatePermissionConfig: personal-certificate-permissions.conf

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}
```

8.2.12.2.2 Fichier `personal-certificate-permissions.conf`

Configuration des permissions nécessitant une authentification personae ou ne nécessitant pas d'authentification personae.

```
# Personal certification configuration for endpoint permissions

permissionsRequiringPersonalCertificate:



permissionsWithoutPersonalCertificate:
  - 'dipexport:create'
  - 'dipexportv2:create'
  - 'dipexport:id:dip:read'
  - 'transfers:create'
  - 'transfers:reply'
  - 'transfers:id:sip:read'
```

(suite sur la page suivante)

(suite de la page précédente)

```
  - 'logbookobjectslifecycles:id:read'
  - 'logbookoperations:read'
  - 'logbookoperations:id:read'
  - 'logbookunitlifecycles:id:read'
  - 'units:read'
  - 'units:stream'
  - 'units:id:read:json'
  - 'units:id:update'
  - 'units:id:objects:read:json'
  - 'units:id:objects:read:binary'
  - 'units:id:objects:accessrequests:create'
  - 'accessrequests:check'
  - 'accessrequests:remove'
  - 'units:update'
  - 'units:update:revert'
  - 'unitsWithInheritedRules:read'
  - 'units:rules:update'
  - 'units:bulk:update'
  - 'accesscontracts:create:json'
  - 'accesscontracts:read'
  - 'accesscontracts:id:read'
  - 'accesscontracts:id:update'
  - 'accessionregisters:read'
  - 'accessionregisters:id:accessionregisterdetails:read'
  - 'agencies:create'
  - 'agencies:read'
  - 'agencies:id:read'
  - 'agenciesfile:check'
  - 'agenciesreferential:id:read'
  - 'audits:create'
  - 'contexts:create:json'
  - 'contexts:read'
  - 'contexts:id:read'
  - 'contexts:id:update'
  - 'distributionreport:id:read'
  - 'formats:read'
  - 'formats:create'
  - 'formats:id:read'
  - 'formatsfile:check'
  - 'ingestcontracts:create:json'
  - 'ingestcontracts:read'
  - 'ingestcontracts:id:read'
  - 'ingestcontracts:id:update'
  - 'operations:read'
  - 'operations:id:read:status'
  - 'operations:id:read'
  - 'operations:id:update'
  - 'operations:id:delete'
  - 'profiles:create:binary'
  - 'profiles:create:json'
  - 'profiles:read'
  - 'profiles:id:read:json'
  - 'profiles:id:update:binaire'
  - 'profiles:id:read:binary'
  - 'profiles:id:update:json'
  - 'rules:read'
  - 'rules:create'
```

(suite sur la page suivante)

(suite de la page précédente)

```
  - 'rules:id:read'
  - 'rulesfile:check'
  - 'rulesreport:id:read'
  - 'rulesreferential:id:read'
  - 'securityprofiles:create:json'
  - 'securityprofiles:read'
  - 'securityprofiles:id:read'
  - 'securityprofiles:id:update'
  - 'traceability:id:read'
  - 'traceabilitychecks:create'
  - 'traceabilitylinkedchecks:create'
  - 'workflows:read'
  - 'ingests:create'
  - 'ingests:local:create'
  - 'ingests:id:archivetransfertreply:read'
  - 'ingests:id:manifests:read'
  - 'switchindex:create'
  - 'reindex:create'
  - 'evidenceaudit:check'
  - 'referentialaudit:check'
  - 'archiveunitprofiles:create:binary'
  - 'archiveunitprofiles:create:json'
  - 'archiveunitprofiles:read'
  - 'archiveunitprofiles:id:read:json'
  - 'archiveunitprofiles:id:update:json'
  - 'ontologies:create:binary'
  - 'ontologies:create:json'
  - 'ontologies:read'
  - 'ontologies:id:read:json'
  - 'ontologies:id:read:binary'
  - 'ontologies:id:update:json'
  - 'reclassification:update'
  - 'rectificationaudit:check'
  - 'storageaccesslog:read:binary'
  - 'objects:read'
  - 'elimination:analysis'
  - 'elimination:action'
  - 'forcepause:check'
  - 'removeforcepause:check'
  - 'probativevalue:check'
  - 'probativevalue:create'
  - 'accessionregisterssymbolic:read'
  - 'griffins:create'
  - 'preservationScenarios:create'
  - 'griffins:read'
  - 'griffin:read'
  - 'preservationScenarios:read'
  - 'preservationScenario:read'
  - 'preservation:update'
  - 'batchreport:id:read'
  - 'preservationreport:id:read'
  - 'logbookoperations:create'
  - 'computeInheritedRules:action'
  - 'computeInheritedRules:delete'
  - 'managementcontracts:create:json'
  - 'managementcontracts:read'
  - 'managementcontracts:id:read'
```

(suite sur la page suivante)

(suite de la page précédente)

```
  - 'managementcontracts:id:update'
  - 'audit:data:consistency'
  - 'objects:deleteGotVersions'
  - 'accessionregisterdetails:read'
  - 'transaction:create'
  - 'transaction:close'
  - 'transaction:send'
  - 'transaction:unit:create'
  - 'transaction:object:upsert'
  - 'transaction:binary:upsert'
```

8.2.12.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-security-internal`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-security-internal`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/vitam-security-internal/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.13 Logbook

8.2.13.1 Présentation

8.2.13.2 Logbook Exploitation

8.2.13.2.1 Configuration du Logbook

logbook.conf : fichier Yaml de configuration du serveur *logbook*. Celle-ci possède une propriété :

- **alertEvents** : configuration des alertes de sécurité

une alerte est déclenchée soit sur l'analyse du couple {evType,outCome} soit sur celle du {outDetail}

1. Dans le cas du déclenchement sur l'analyse du couple {evType, outCome}

```
- evType: 'CHECK_HEADER.CHECK_CONTRACT_INGEST'
  outcome: 'KO'
```

2. Dans le cas du déclenchement sur l'analyse du {outComeDetail}

```
- outDetail: 'CHECK_HEADER.CHECK_CONTRACT_INGEST.KO'
```

3. La liste des détections de l'alerte

- non conformité de la base des règles de gestion au référentiel enregistré (CHECK_RULES)
- refus d'entrée d'un SIP pour des raisons d'inadéquation de contrats (CHECK_HEADER.CHECK_CONTRACT_INGEST)
- soumission d'un SIP avec une classification incompatible avec la plateforme (CHECK_CLASSIFICATION_LEVEL)
- valeur de durée dans les régle de gestion inférieure à la durée minimum (CHECK_RULES.MAX_DURATION_EXCEEDS)
- refus d'un accès avec les droits personae (STP_PERSONAL_CERTIFICATE_CHECK)
- absence de sécurisation des journaux sur 12h (TODO)

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/logbook`.

8.2.13.2.2 Fichier `logbook.conf`

```
# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: logbook
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].logbook.user }}
dbPassword: {{ mongodb['mongo-data'].logbook.password }}
jettyConfig: jetty-config.xml
p12LogbookPassword: {{ keystores.timestamping.secure_logbook }}
p12LogbookFile: keystore_secure-logbook.p12
workspaceUrl: {{ vitam.workspace | client_url }}
processingUrl: {{ vitam.processing | client_url }}

# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  httpPort: {{ elasticsearch.data.port_http }}
{% endfor %}

# ElasticSearch tenant indexation
elasticsearchTenantIndexation:
```

(suite sur la page suivante)

(suite de la page précédente)

```
  default_config:
    logbookoperation:
      number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪logbookoperation.number_of_shards }}
      number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪logbookoperation.number_of_replicas }}

{% if vitam_elasticsearch_tenant_indexation.dedicated_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.dedicated_tenants is not none %}
  dedicated_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.dedicated_tenants %}
{% if (entry.logbookoperation is defined and (entry.logbookoperation.number_of_shards␣
↪is defined or entry.logbookoperation.number_of_replicas is defined)) %}
  - tenants: '{{ entry.tenants }}'
    logbookoperation:
{% if entry.logbookoperation.number_of_shards is defined %}
      number_of_shards: {{ entry.logbookoperation.number_of_shards }}
{% endif %}
{% if entry.logbookoperation.number_of_replicas is defined %}
      number_of_replicas: {{ entry.logbookoperation.number_of_replicas }}
{% endif %}
{% endif %}
{% endfor %}
{% endif %}

{% if vitam_elasticsearch_tenant_indexation.grouped_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.grouped_tenants is not none %}
  grouped_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.grouped_tenants %}
{% if (entry.logbookoperation is defined and (entry.logbookoperation.number_of_shards␣
↪is defined or entry.logbookoperation.number_of_replicas is defined)) %}
  - name: '{{ entry.name }}'
    tenants: '{{ entry.tenants }}'
    logbookoperation:
{% if entry.logbookoperation.number_of_shards is defined %}
      number_of_shards: {{ entry.logbookoperation.number_of_shards }}
{% endif %}
{% if entry.logbookoperation.number_of_replicas is defined %}
      number_of_replicas: {{ entry.logbookoperation.number_of_replicas }}
{% endif %}
{% endif %}
{% endfor %}
{% endif %}

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}

## Configuration for logbook coherence check
# list of operations that generate LFC
opWithLFC: [
  "PROCESS_SIP_UNITARY",
  "FILINGSCHEME",
  "HOLDINGSCHEME",
  "UPDATE_RULES_ARCHIVE_UNITS",
  "PROCESS_AUDIT",
```

(suite sur la page suivante)

(suite de la page précédente)

```
  "STP_UPDATE_UNIT"]
# list of events not declared in wf
opEventsNotInWf: [
  "STP_SANITY_CHECK_SIP",
  "SANITY_CHECK_SIP",
  "CHECK_CONTAINER",
  "STP_UPLOAD_SIP"
]
# list of events to skip for OP-LFC check
opLfcEventsToSkip: [
  "STP_SANITY_CHECK_SIP", "SANITY_CHECK_SIP", "CHECK_CONTAINER", "STP_UPLOAD_SIP",
↪"ATR_NOTIFICATION", "ROLL_BACK",
  "STORAGE_AVAILABILITY_CHECK", "ACCESSION_REGISTRATION",
  "ROLL_BACK", "ATR_NOTIFICATION", "COMMIT_LIFE_CYCLE_OBJECT_GROUP", "COMMIT_LIFE_
↪CYCLE_UNIT",
  "LIST_OBJECTGROUP_ID", "REPORT_AUDIT",
  "LIST_ARCHIVE_UNITS", "LIST_RUNNING_INGESTS"]

# Configuration des alertes de securite
alertEvents:
- evType: 'CHECK_HEADER.CHECK_CONTRACT_INGEST'
  outcome: 'KO'
- evType: 'CHECK_RULES.MAX_DURATION_EXCEEDS'
  outcome: 'KO'
- evType: 'CHECK_RULES'
  outcome: 'KO'
- outDetail: 'CHECK_CLASSIFICATION_LEVEL.KO'
- outDetail: 'STP_PERSONAL_CERTIFICATE_CHECK.KO'

# Traceability params
operationTraceabilityTemporizationDelay: {{ vitam.logbook.
↪operationTraceabilityTemporizationDelay }}
operationTraceabilityMaxRenewalDelay: {{ vitam.logbook.
↪operationTraceabilityMaxRenewalDelay }}
operationTraceabilityMaxRenewalDelayUnit: {{ vitam.logbook.
↪operationTraceabilityMaxRenewalDelayUnit }}
operationTraceabilityThreadPoolSize: {{ vitam.logbook.
↪operationTraceabilityThreadPoolSize }}
lifecycleTraceabilityTemporizationDelay: {{ vitam.logbook.
↪lifecycleTraceabilityTemporizationDelay }}
lifecycleTraceabilityMaxRenewalDelay: {{ vitam.logbook.
↪lifecycleTraceabilityMaxRenewalDelay }}
lifecycleTraceabilityMaxRenewalDelayUnit: {{ vitam.logbook.
↪lifecycleTraceabilityMaxRenewalDelayUnit }}
lifecycleTraceabilityMaxEntries: {{ vitam.logbook.lifecycleTraceabilityMaxEntries }}
```

8.2.13.2.3 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.13.2.4 Fichier `logbook-client.conf`

```
serverHost: {{ vitam.logbook.host }}
serverPort: {{ vitam.logbook.port_service }}
```

8.2.13.2.5 Fichier `securisationDaemon.conf`

```
tenants: [ "{{ vitam_tenant_ids | join('", "') }}" ]
adminTenant : {{ vitam_tenant_admin }}
```

8.2.13.2.6 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.13.2.7 Fichier `traceabilityAudit.conf`

```
tenants: [ "{{ vitam_tenant_ids | join('", "') }}" ]
operationTraceabilityMaxRenewalDelay: {{ vitam.logbook.
↪operationTraceabilityMaxRenewalDelay }}
operationTraceabilityMaxRenewalDelayUnit: {{ vitam.logbook.
↪operationTraceabilityMaxRenewalDelayUnit }}
lifecycleTraceabilityMaxRenewalDelay: {{ vitam.logbook.
↪lifecycleTraceabilityMaxRenewalDelay }}
lifecycleTraceabilityMaxRenewalDelayUnit: {{ vitam.logbook.
↪lifecycleTraceabilityMaxRenewalDelayUnit }}
```

8.2.13.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-logbook`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-logbook`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/logbook/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

8.2.14 Metadata

8.2.14.1 Présentation

8.2.14.2 Configuration / fichiers utiles

8.2.14.2.1 Fichier `metadata.conf`

```
workspaceUrl: {{vitam.workspace | client_url}}
urlProcessing: {{vitam.processing | client_url}}
contextPath: {{ vitam_struct.context_path }}

# Archive Unit Profile cache settings (max entries in cache & retention timeout in␣
↪seconds)
archiveUnitProfileCacheMaxEntries: {{ vitam.metadata.
↪archiveUnitProfileCacheMaxEntries }}
archiveUnitProfileCacheTimeoutInSeconds: {{ vitam.metadata.
↪archiveUnitProfileCacheTimeoutInSeconds }}

# Schema validator cache settings (max entries in cache & retention timeout in␣
↪seconds)
schemaValidatorCacheMaxEntries: {{ vitam.metadata.schemaValidatorCacheMaxEntries }}
schemaValidatorCacheTimeoutInSeconds: {{ vitam.metadata.
↪schemaValidatorCacheTimeoutInSeconds }}

# DIP purge service (in minutes)
dipTimeToLiveInMinutes: {{ vitam.metadata.dipTimeToLiveInMinutes }}
criticalDipTimeToLiveInMinutes: {{ vitam.metadata.criticalDipTimeToLiveInMinutes }}

# TRANSFER purge service (in minutes)
transfersSIPTimeToLiveInMinutes: {{ vitam.metadata.transfersSIPTimeToLiveInMinutes }}

# Units Stream Threshold
unitsStreamThreshold: {{ vitam.metadata.unitsStreamThreshold | default(1000000) }}
streamExecutionLimit: {{ vitam.metadata.streamExecutionLimit | default(3) }}

# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_data'] %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endfor %}
dbName: metadata
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb['mongo-data'].metadata.user }}
dbPassword: {{ mongodb['mongo-data'].metadata.password }}

jettyConfig: jetty-config.xml
```

(suite sur la page suivante)

(suite de la page précédente)

```
# ElasticSearch
clusterName: {{ vitam_struct.cluster_name }}
elasticsearchNodes:
{% for server in groups['hosts_elasticsearch_data'] %}
- hostName: {{ hostvars[server]['ip_service'] }}
  httpPort: {{ elasticsearch.data.port_http }}
{% endfor %}

# ElasticSearch External Metadata Mapping
elasticsearchExternalMetadataMappings:
- collection: Unit
  mappingFile: {{ vitam.metadata.elasticsearch_mapping_dir }}/unit-es-mapping.json
- collection: ObjectGroup
  mappingFile: {{ vitam.metadata.elasticsearch_mapping_dir }}/og-es-mapping.json

# ElasticSearch tenant indexation
elasticsearchTenantIndexation:

  default_config:
    unit:
      number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.unit.
↪number_of_shards }}
      number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪unit.number_of_replicas }}
    objectgroup:
      number_of_shards: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪objectgroup.number_of_shards }}
      number_of_replicas: {{ vitam_elasticsearch_tenant_indexation.default_config.
↪objectgroup.number_of_replicas }}

{% if vitam_elasticsearch_tenant_indexation.dedicated_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.dedicated_tenants is not none %}
  dedicated_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.dedicated_tenants %}
{% if (entry.unit is defined and (entry.unit.number_of_shards is defined or entry.
↪unit.number_of_replicas is defined)) or
      (entry.objectgroup is defined and (entry.objectgroup.number_of_shards is␣
↪defined or entry.objectgroup.number_of_replicas is defined)) %}
  - tenants: '{{ entry.tenants }}'
{% if entry.unit is defined %}
    unit:
{% if entry.unit.number_of_shards is defined %}
      number_of_shards: {{ entry.unit.number_of_shards }}
{% endif %}
{% if entry.unit.number_of_replicas is defined %}
      number_of_replicas: {{ entry.unit.number_of_replicas }}
{% endif %}
{% endif %}
{% if entry.objectgroup is defined %}
    objectgroup:
{% if entry.objectgroup.number_of_shards is defined %}
      number_of_shards: {{ entry.objectgroup.number_of_shards }}
{% endif %}
{% if entry.objectgroup.number_of_replicas is defined %}
      number_of_replicas: {{ entry.objectgroup.number_of_replicas }}
{% endif %}
{% endif %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% endif %}
{% endfor %}
{% endif %}

{% if vitam_elasticsearch_tenant_indexation.grouped_tenants is defined and vitam_
↪elasticsearch_tenant_indexation.grouped_tenants is not none %}
  grouped_tenants:
{% for entry in vitam_elasticsearch_tenant_indexation.grouped_tenants %}
{% if (entry.unit is defined and (entry.unit.number_of_shards is defined or entry.
↪unit.number_of_replicas is defined)) or
      (entry.objectgroup is defined and (entry.objectgroup.number_of_shards is␣
↪defined or entry.objectgroup.number_of_replicas is defined)) %}
  - name: '{{ entry.name }}'
    tenants: '{{ entry.tenants }}'
{% if entry.unit is defined %}
    unit:
{% if entry.unit.number_of_shards is defined %}
      number_of_shards: {{ entry.unit.number_of_shards }}
{% endif %}
{% if entry.unit.number_of_replicas is defined %}
      number_of_replicas: {{ entry.unit.number_of_replicas }}
{% endif %}
{% endif %}
{% if entry.objectgroup is defined %}
    objectgroup:
{% if entry.objectgroup.number_of_shards is defined %}
      number_of_shards: {{ entry.objectgroup.number_of_shards }}
{% endif %}
{% if entry.objectgroup.number_of_replicas is defined %}
      number_of_replicas: {{ entry.objectgroup.number_of_replicas }}
{% endif %}
{% endif %}
{% endif %}
{% endfor %}
{% endif %}

#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}

isDataConsistencyAuditRunnable: {{ vitam.metadata.isDataConsistencyAuditRunnable }}
dataConsistencyAuditOplogMaxSize: {{ vitam.metadata.dataConsistencyAuditOplogMaxSize }
↪}
mongodShardsConf:
 dbUserName: {{ mongodb['mongo-data']['localadmin']['user'] }}
 dbPassword: {{ mongodb['mongo-data']['localadmin']['password'] }}
 mongoDbShards:
{% for shard in groups['hosts_mongod_data'] %}
   - shardName: shard{{ hostvars[shard]['mongo_shard_id']}}
     mongoDbNodes:
{% for server in groups['hosts_mongod_data'] %}
      - dbHost: {{ hostvars[server]['ip_service'] }}
        dbPort: {{ mongodb.mongod_port }}
{% endfor %}
{% endfor %}
```

8.2.14.2.1.1 Paramétrage des caches

Metadata maintient en mémoire un ensemble de caches pour la gestion des données peu modifiées et qui interviennent lors des modifications de métadonnées (référentiels d'ontologie, schéma de donnée).

Cache du référentiel de l'ontologie :

- `ontologyCacheMaxEntries` : Nombre maximum d'objets à maintenir dans le cache (par défaut 100). Ce paramètre dépend du nombre de traitements actifs.
- `ontologyCacheTimeoutInSeconds` : Durée en secondes de rétention des objets en cache (par défaut 300, soit 5 minutes)

Cache du référentiel des profiles d'unités archivistiques :

- `archiveUnitProfileCacheMaxEntries` : Nombre maximum d'objets à maintenir dans le cache (par défaut 100). Ce paramètre dépend du nombre de traitements actifs.
- `archiveUnitProfileCacheTimeoutInSeconds` : Durée en secondes de rétention des objets en cache (par défaut 300, soit 5 minutes)

Cache des validateurs de schémas chargés en mémoire :

- `schemaValidatorCacheMaxEntries` : Nombre maximum d'objets à maintenir dans le cache (par défaut 100). Ce paramètre dépend du nombre de traitements actifs.
- `schemaValidatorCacheTimeoutInSeconds` : Durée en secondes de rétention des objets en cache (par défaut 300, soit 5 minutes)

8.2.14.2.1.2 Paramétrage des mappings externes elasticsearch

- `elasticsearchExternalMetadataMappings` : La liste des collections et le chemin vers le fichier de mapping elasticsearch associé dans le dossier de configuration `/vitam/conf/metadata/mapping`

la Liste `elasticsearchExternalMetadataMappings` est composée comme suit :

- `collection` : La collection Unit en premier
- `mappingFile` : Le chemin vers le fichier mapping de la collection, généralement dans le fichier de configuration du composant metadata.
- `collection` : La collection ObjectGroup
- `mappingFile` : Le chemin vers le fichier mapping de la collection, généralement dans le fichier de configuration du composant metadata.

Avertissement : ces mapping devront être en cohérence avec l'ontologie.

8.2.14.2.1.3 Paramétrage de la limite du flux des unités archivestiques

- `unitsStreamThreshold` : c'est le seuil en nombre d'unités archivestiques de plateforme, si le nombre des résultats dépassent ce seuil.

aucun résultat ne sera fourni.

- `streamExecutionLimit` : la limite d'exécution d'une recherche par jour. Si cette valeur est égale à zéro, le nombre de recherche est illimité.

8.2.14.2.2 Fichier `functional-administration-client.conf`

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.14.2.3 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.14.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-metadata`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-metadata`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/metadata/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.15 Processing

8.2.15.1 Introduction

8.2.15.1.1 But de cette documentation

Le but de cette documentation est d'expliquer la configuration et l'exploitation de ce module.

8.2.15.2 Processing

Nom de l'image docker : **processing**

Dans cette image est déployé le module processing

8.2.15.2.1 Configuration du worker

Dans **/vitam/conf** :

1. **processing.conf** : Fichier Yaml de configuration du server *processing*. Il possède une propriété :

- **jettyConfig** : emplacement du ficher de configuration XML *jetty* (exemple jetty-config.xml)
- **urlWorkspace** : URL d'accès au service distant *workspace* (exemple http://localhost:8088)
- **urlMetadata** : URL d'accès au service distant *metadata* (exemple http://localhost:8088)

2. **logbook-client.conf** : Fichier de configuration du client qui communique avec le **logbook**. Il contient les propriétés suivantes :

- **serverHost** : host distant du service logbook
- **serverPort** : port distant du service logbook

3. **server-identity.conf** : identification du serveur
4. **logback.xml** : configuration des logs

8.2.15.2.2 Supervision du service

Contrôler le retour HTTP 200 et identité du serveur (cf *server-identity.conf*) sur l'URL `<protocole web https ou https>://<host>:<port>/processing/v1/status`

8.2.15.3 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/processing`.

8.2.15.3.1 Fichier `processing.conf`

```
urlMetadata: {{ vitam_struct | client_url }}
urlWorkspace: {{ vitam.workspace | client_url }}
jettyConfig: jetty-config.xml
workflowRefreshPeriod: 1
processingCleanerPeriod: 1
maxDistributionInMemoryBufferSize: {{vitam.processing.
↪maxDistributionInMemoryBufferSize | default(100000) }}
maxDistributionOnDiskBufferSize: {{vitam.processing.maxDistributionOnDiskBufferSize |␣
↪default(100000000) }}

# Async resource monitoring config (for unavailable async resources when using tape␣
↪storage offer)
delayAsyncResourceMonitor: {{vitam.processing.delayAsyncResourceMonitor |␣
↪default(300) }}
delayAsyncResourceCleaner: {{vitam.processing.delayAsyncResourceCleaner |␣
↪default(300) }}
```

8.2.15.3.2 Fichier `version.conf`

```
binaryDataObjectVersions:
- BinaryMaster
- Dissemination
- Thumbnail
- TextContent
physicalDataObjectVersions:
- PhysicalMaster
- Dissemination
```

8.2.15.3.3 Fichier `storage-client.conf`

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.15.3.4 Fichier `metadata-client.conf`

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

8.2.15.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-processing`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-processing`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/processing/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.15.5 Parallélisation des workflows et des opérations

- Plusieurs workflows et opérations fonctionnelles peuvent être bloqués par certains processus qui les empêchent de terminer en succès. Cet

interblocage est dû à une utilisation mutuelles de certaines ressources qui ne peuvent pas être traitées par plusieurs processus différents. Ci-dessous un tableau récapitulatif de la liste des workflows que VITAM exposent, leurs dépendances par rapport à d'autres workflows qui les bloquent, ainsi que d'autres opérations qui présentent une dépendance à d'autres opérations.

Tableau 1: Parallélisation des workflows et des opérations

Fonctionnalité	Workflow / Endpoint	Bloqué(e) par
Ingest	BigIngestWorkflow.json	
	DefaultFilingSchemeWorkflow.json	
	DefaultHoldingSchemeWorkflow.json	
	DefaultIngestBlankTestWorkflow.json	
	DefaultIngestWorkflow.json	/
Clean up Ingest	IngestCleanupWorkflow.json	Reclassification Unit,Elimination(action),Transfer reply, Clean up Ingest
MAJ units	BulkAtomicUpdateUnitDescWorkflow.json	/
MAJ RG units	MassUpdateUnitRuleWorkflow.json	/
MAJ units	MassUpdateUnitDescWorkflow.json	/
Calcul échéances	ComputeInheritedRulesDeleteWorkflow.json	
	ComputeInheritedRulesWorkflow.json	/
Revert Unit data	RevertEssentialMetadataWorkflow.json	/
Reclassification Unit	ReclassificationWorkflow.json	/
Calcul valeur probante	ExportProbativeValueWorkflowV2.json	/
Export Units	ExportUnitWorkflow.json	/
MAJ RG dans les units	DefaultRulesUpdateWorkflow.json	/
Migration de données	DataMigrationWorkflow.json	/
Audit	DefaultAuditWorkflow.json	/
Audit d'évidence de données	EvidenceAuditWorkflow.json	/
Audit correctif de données	RectificationAuditWorkflow.json	/
Preservation	PreservationWorkflow.json	/
Suppression des Got	DeleteGotVersionsWorkflow.json	/
Sécurisation LFC Got	DefaultObjectGroupLifecycleTraceability.json	Sécurisation LFC Got en cours, Gots à sécuriser
Sécurisation LFC Unit	DefaultUnitLifecycleTraceability.json	Sécurisation LFC Unit en cours, Units à sécuriser
Securisation logbook	LinkedCheckTraceability.json	/
Elimination (analyse)	EliminationAnalysisWorkflow.json	/
Elimination (action)	EliminationActionWorkflow.json	Reclassification Unit,Elimination(action)
Transfer reply	TransferReplyWorkflow.json	/
Transfer units	TransferUnitWorkflow.json	/
Import des régles de gestion	`/adminmanagement/v1/rules/import`	Import des régles de gestion

Suite sur la page suivante

Tableau 1 – suite de la page précédente

Fonctionnalité	Workflow / Endpoint	Bloqué(e) par
Import des services agents	`/adminmanagement/v1/agencies/import`	Import des services agents

8.2.16 Storage

8.2.16.1 Introduction

8.2.16.1.1 But de cette documentation

Le but de cette documentation est d'expliquer la configuration et l'exploitation des modules :

- **storage-engine**
- **storage-offer-default**

8.2.16.2 storage-engine

8.2.16.2.1 Présentation

Rôle :

- Stockage des données (Méta Données, Objets Numériques et journaux SAE et de l'archive)

Fonctions :

- Utilisation de stratégie de stockage (abstraction par rapport aux offres de stockage sous-jacentes)
- Gestion des différentes offres de stockage

8.2.16.2.2 Storage Engine

Nom de l'image docker : **storage-engine**

Dans cette image sont déployés :

- le moteur de stockage (storage-engine)
- l'implémentation du driver correspondant à l'offre de stockage par défaut (storage-offer-default)

8.2.16.2.2.1 Configuration du moteur de stockage

Dans **/vitam/conf** :

1. **storage-engine.conf** : Fichier Yaml de configuration du server *storage-engine*. Il possède une propriété :

- **urlWorkspace** : URL d'accès au service distant *workspace* (exemple http://localhost:8088)

2. **driver-location.conf** : Fichier Yaml de configuration du DriverManager, Il permet de définir l'emplacement où sont stockés les fichiers JAR contenant les implémentations des différents drivers pour les différentes offres. Il possède une seule propriété :

- **driverLocation** : emplacement des jars (chemin absolu de préférence)

3. **driver-mapping.conf** : Fichier Yaml de configuration du DriverMapper (persistance de l'association driver / offre). Pour le moment, ce fichier de configuration contient le chemin d'accès aux fichiers qui définissent le mapping driver<->offre, plus tard il évoluera sans doute pour prendre en compte des données en base et donc contenir la configuration d'accès à la base. Il contient deux propriétés :

- **driverMappingPath** : Définit l'emplacement des fichiers de persistance (au jourd'hui on a 1 seul driver/offre, donc 1 seul fichier de persistence sera présent). La propriété doit finir par « / ».
- **delimiter** : Définit le « délimiteur » (CSV style) des fichiers.

4. **static-offer.json** : Contient la description de l'offre "default" au format JSON (un jour sera sans doute dans une base de données). En PJ un exemple de ce fichier. La propriété baseUrl et parameters nécessitent d'être templaté. Et la propriété parameters doit contenir keystore, trustore et leur mot de passe que le storage driver va utiliser pour la vérification de l'authentication. Il s'agit de l'URL d'accès à l'offre de stockage "default". Exemple :

```
{
  "id" : "default",
  "baseUrl" : "https://localhost:8088",
  "parameters" : {
    "user" : "bob"
    "keyStore-keyPath": "src/test/resources/storage-test/tls/client/client.p12",
    "keyStore-keyPassword": "vitam2016",
    "trustStore-keyPath": "src/test/resources/storage-test/tls/server/truststore.jks",
    "trustStore-keyPassword": "tazerty",
    "referent": "true"
  }
}
```

To remove TLS support :

- change « https » to « http » in **baseUrl**

```
{
  "id" : "default",
  "baseUrl" : "http://localhost:8088",
  "parameters" : {
    "user" : "bob"
  }
}
```

To define « referent » offer :

- choose **exactly one** offer by adding parameter **referent**

```
[
  {
    "id" : "default",
    "baseUrl" : "http://localhost:8088",
    "parameters" : {
      "user" : "bob",
      "referent": "true"
    }
  },
  {
    "id" : "offer2",
    "baseUrl" : "http://localhost:8089",
    "parameters" : {
      "user" : "bob"
```

(suite sur la page suivante)

(suite de la page précédente)

```
    }
  }
]
```

- change storage-default-offer.json to disable authentication

```
jettyConfig: jetty-config-nossl.xml
authentication : false
```

- change the jetty-config-nossl.xml of the offer (CAS Manager) to not include any TLS configuration

5. **static-strategy.json** : Contient les informations de la stratégie de stockage (1 seule pour le moment). Ce fichier n'est pas à modifier.

```
{
 "id" : "default",
 "hot" : {
   "copy" : 1,
   "offers" : [
     {
      "id" : "default",
      "rank": 0
     }
   ]
 }
}
```

6. **server-identity.conf** : identification du serveur
7. **logback.xml** : configuration des logs

8.2.16.2.2.2 Configuration du driver de l'offre de stockage par défaut

Dans **/vitam/data** :

1. **fr.gouv.vitam.storage.offers.workspace.driver.DriverImpl** : Il s'agit du fichier de persistence. Il contient l'identifiant de l'offre associée au driver (plus tard potentiellement DES offres associées) : *« default »*. Il DOIT être placé dans le répertoire défini dans le fichier *driver-mapping.conf*.

Dans **/vitam/lib** :

1. **storage-driver-default.jar** : Il s'agit d'un jar contenant l'implémentation du Driver vitam pour l'offre *« storage-offer-default »*. Ce jar DOIT être placé dans le dossier défini dans la propriété *driverLocation* du fichier *driver-location.conf*. Par défaut il est chargé en tant que dépendance du projet.

8.2.16.2.2.3 Supervision du service

Contrôler le retour HTTP 200 et identité du serveur (cf *server-identity.conf*) sur l'URL `<protocole web https ou https>://<host>:<port>/storage/v1/status`

8.2.16.2.3 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/storage-engine`.

8.2.16.2.3.1 Fichier `driver-location.conf`

```
driverLocation: {{ vitam_folder_lib }}
```

8.2.16.2.3.2 Fichier `driver-mapping.conf`

```
driverMappingPath: {{ vitam_folder_data }}/
delimiter: ;
```

8.2.16.2.3.3 Fichier `static-offer.json`

```
{% if vitam.storageofferdefault.https_enabled==true %}
    {% set protocol = 'https' %}
{% else %}
    {% set protocol = 'http' %}
{% endif %}
[
{% for item in all_used_offers %}
{
{% if item.id is defined %}
    "id" : "{{ item.id }}",
{% else %}
    "id" : "{{ item.name }}.service.{{ item.vitam_site_name | default(vitam_site_
↪name) }}.{{ consul_domain }}",
{% endif %}
    "baseUrl" : "{{ protocol }}://{{ item.name }}.service.{{ item.vitam_site_name |␣
↪default(vitam_site_name) }}.{{ consul_domain }}:{{ vitam.storageofferdefault.port_
↪service }}",
    {% if item.asyncRead is defined %} "asyncRead": {{item.asyncRead|lower }}, {%␣
↪endif %}
    "parameters" : {
        {% if vitam.storageofferdefault.https_enabled==true %}
        "keyStore-keyPath": "{{ vitam_folder_conf }}/keystore_storage.p12",
        "keyStore-keyPassword": "{{ keystores.client_storage.storage }}",
        "trustStore-keyPath": "{{ vitam_folder_conf }}/truststore_storage.jks",
        "trustStore-keyPassword": "{{ truststores.client_storage }}"
        {% endif %}
    }
}
{% if not loop.last %},
{% endif %}
{% endfor %}
]
```

8.2.16.2.3.4 Fichier `static-strategy.json`

Ce fichier décrit les stratégies de stockage définies, ainsi que les offres en associées. Chaque offre de stockage définit un ordre de lecture `rank`. Cet ordre sera pris en considération pour la lecture des objets dans une stratégie. Cependant, pour le cas de la reconstruction, la lecture des objets se fait à travers l'offre référente de la startégie (celle qui a la propriété `referent` en `true`).

```
[
    {
        "id" : "default",
        "offers" : [
{% for item in vitam_strategy %}
{% if item.id is defined %}
            {"id" : "{{ item.id }}"{% if item.referent | default(false) | bool ==␣
↪true %}, "referent" : true{% endif %}{% if item.status is defined %}, "status" : "{
↪{ item.status | upper }}" {% endif %}{% if item.rank is defined %}, "rank" : {{␣
↪item.rank }} {% endif %}}{% if not loop.last %},{% endif %}
{% else %}
            {"id" : "{{ item.name }}.service.{{ item.vitam_site_name |default(vitam_
↪site_name) }}.{{ consul_domain }}"{% if item.referent | default(false) | bool ==␣
↪true %}, "referent" : true{% endif %}{% if item.status is defined %}, "status" : "{
↪{ item.status| upper }}" {% endif %}{% if item.rank is defined %}, "rank" : {{ item.
↪rank }} {% endif %}}{% if not loop.last %},{% endif %}
{% endif %}
{% endfor %}
        ]
    }
{% if other_strategies is defined %}
{% for strategy_name, strategy_offers in other_strategies.items() %}
    ,
    {
        "id" : "{{ strategy_name }}",
        "offers" : [
{% for strategy_offer in strategy_offers %}
            {"id" : "{{ strategy_offer.name }}.service.{{ strategy_offer.vitam_site_
↪name |default(vitam_site_name) }}.{{ consul_domain }}"{% if strategy_offer.referent␣
↪| default(false) | bool == true %}, "referent" : true{% endif %}{% if strategy_
↪offer.status is defined %}, "status" : "{{ strategy_offer.status | upper }}" {%␣
↪endif %}{% if strategy_offer.rank is defined %}, "rank" : {{ strategy_offer.rank }}
↪{% endif %}}{% if not loop.last %},{% endif %}
{% endfor %}
        ]
    }
{% endfor %}
{% endif %}

]
```

8.2.16.2.3.5 Fichier `storage-engine.conf`

```
urlWorkspace: {{ vitam.workspace | client_url }}
timeoutMsPerKB: {{ vitam.storageengine.timeoutMsPerKB }}
minWriteTimeoutMs: {{ vitam.storageengine.minWriteTimeoutMs }}
minBulkWriteTimeoutMsPerObject: {{ vitam.storageengine.minBulkWriteTimeoutMsPerObject␣
↪}}
jettyConfig: jetty-config.xml
zippingDirecorty: {{ vitam_folder_data }}/storage_archives
loggingDirectory: {{ vitam_folder_log }}
p12LogbookPassword: {{ keystores.timestamping.secure_storage }}
p12LogbookFile: keystore_{{ vitam_timestamp_usage }}.p12
storageTraceabilityOverlapDelay: {{ vitam.storageengine.
↪storageTraceabilityOverlapDelay }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
restoreBulkSize: {{ vitam.storageengine.restoreBulkSize }}
offerSynchronizationBulkSize: {{ vitam.storageengine.offerSynchronizationBulkSize }}
offerSyncThreadPoolSize: {{ vitam.storageengine.offerSyncThreadPoolSize }}
offerSyncNumberOfRetries: {{ vitam.storageengine.offerSyncNumberOfRetries }}
offerSyncFirstAttemptWaitingTime: {{ vitam.storageengine.
↪offerSyncFirstAttemptWaitingTime }}
offerSyncWaitingTime: {{ vitam.storageengine.offerSyncWaitingTime }}
offerSyncAccessRequestCheckWaitingTime: {{ vitam.storageengine.
↪offerSyncAccessRequestCheckWaitingTime }}
storageLogBackupThreadPoolSize: {{ vitam.storageengine.storageLogBackupThreadPoolSize␣
↪}}
storageLogTraceabilityThreadPoolSize: {{ vitam.storageengine.
↪storageLogTraceabilityThreadPoolSize }}
#Basic Authentication
adminBasicAuth:
- userName: {{ admin_basic_auth_user }}
  password: {{ admin_basic_auth_password }}
```

8.2.16.2.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-storage`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-storage`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/storage/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.16.2.4.1 access-log

Le log des accès est généré lors d'un accès à l'objet (fichier numérique), que ce soit par téléchargement de l'objet ou export d'un DIP. Les accès à l'unité archivistique ne sont pas concernés.

Exemple de log généré lors de l'export d'un DIP d'une unité archivistique ayant un GOT contenant un objet

```
{"eventDateTime":"2019-01-11T12:50:53.344","xRequestId":
↪"aeeaaaaaachfmo4dabyw6aliht3q74aaaaaq","applicationId":"MyApplicationId-ChangeIt",
↪"objectIdentifier":"aeaaaaaaaahk2vrsabz26alhywthyoaaaaba","size":"11","qualifier":
↪"BinaryMaster","version":"1","contextId":"CT-000001","contractId":"ContratTNR",
↪"archivesId":"aeaqaaaaaahk2vrsabz26alhywthzbaaaaea"}
```

Structure des logs :

- « eventDateTime » : date et heure de l'accès au format AAAA-MM-JJTHH :MM :SS.[digits de millisecondes]
- « xRequestId » : identifiant de l'opération d'export du DIP
- « applicationId » : identifiant de l'application ayant demandé l'export du DIP
- « objectIdentifier » : identifiant de l'objet auquel on a accédé
- « size » : taille en octets de l'objet
- « qualifier » : usage de l'objet
- « version » : version de l'usage de l'objet
- « contextId » : identifiant du contexte utilisé pour l'accès
- « contractId » : identifiant du contrat utilisé pour l'accès
- « archivesId » : identifiant de l'unité archivistique dont dépend le groupe d'objets contenant l'objet auquel on a accédé

Selon le paramétrage du contrat d'accès (AccessLog ACTIVE/INACTIVE), l'accès à un objet sera journalisé ou non. Par défaut, l'accès n'est pas journalisé.

Pour l'heure système en cours, ces fichiers sont présents sur les machines hébergeant le composant **storage** sous l'arborescence `/vitam/log/storage/access-log/`. Chaque fichier est nommé tel que :

`<tenant>_<date>_<id opération>.log`

Le *timer* systemD `vitam-storage-accesslog-backup` effectue la pérénisation sur offre de ces fichiers chaque heure. Dès lors, les *accesslog* sont accessibles dans des *containers* nommés `<environnement>_<tenant>_storageaccesslog`.

Exemple en stockage filesystem pour un environnement nommé int : `/vitam/data/offer/container/int_<tenant>_storageaccesslog/`

8.2.16.3 offer

8.2.16.3.1 Présentation

Ce composant est une déclinaison des offres de stockage pérenne des données.

Rôle :

- Fournir une offre de stockage par défaut permettant la persistance des objets.

Fonctions :

- Offre de stockage fournie par défaut
- Stockage simple des objets numériques sur plusieurs providers :
  - stockage sur système de fichiers local
  - stockage sur object store compatible S3
  - stockage sur object store compatible Swift
  - stockage sur bibliothèques de bandes magnétiques (offre froide).

Par tenant VITAM déclaré, jusqu'à 17 *containers* sont créés :

- accessionregisterdetail
- accessionregistersymbolic
- backup
- backup_operation
- check_logbookreports
- dip
- distribution_reports
- logbook
- manifest
- object
- objectGroup
- report
- rules
- storageaccesslog
- storagelog
- storagetraceability
- unit

selon la norme `<vitam_site_name>_<tenant>_<container>` (R9 et plus) ou `<tenant>_<container>` (R7 et migrations depuis R7).

8.2.16.3.2 Storage Offer Default

Nom de l'image docker : **storage-offer-default**

Dans cette image est déployée l'offre de stockage par défaut utilisant le workspace.

8.2.16.3.2.1 Configuration de l'offre de stockage

1. **default-storage.conf** : Fichier Yaml de configuration du service. Contient les propriétés suivantes :

- **contextPath** : context path du server (mettre / par défaut)
- **storagePath** : chemin sur le filesystem sur lequel sont stockés les objects (/vitam/data).

2. **server-identity.conf** : identification du serveur
3. **logback.xml** : configuration des logs

8.2.16.3.2.2 Supervision du service

Contrôler le retour HTTP 200 et identité du serveur (cf *server-identity.conf*) sur l'URL `<protocole web https ou https>://<host>:<port>/offer/v1/status`

8.2.16.3.3 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/offer`.

8.2.16.3.3.1 Fichier `default-offer.conf`

```
contextPath: /
# Smile : TODO : remove storagePath from this file
storagePath: {{ vitam_folder_data }}
jettyConfig: jetty-config.xml
authentication : {{ vitam_struct.https_enabled }}
# Configuration MongoDB
mongoDbNodes:
{% for server in groups['hosts_mongos_offer'] %}
{% if hostvars[server]['mongo_cluster_name'] == offer_conf or inventory_hostname in␣
↪single_vm_hostnames %}
- dbHost: {{ hostvars[server]['ip_service'] }}
  dbPort: {{ mongodb.mongos_port }}
{% endif %}
{% endfor %}
dbName: offer
dbAuthentication: {{ mongodb.mongo_authentication }}
dbUserName: {{ mongodb[offer_conf].offer.user }}
dbPassword: {{ mongodb[offer_conf].offer.password }}

offerLogCompaction:
  expirationValue: {{ vitam_offers[offer_conf].offer_log_compaction.expiration_value␣
↪| default(21) }}
  expirationUnit: {{ vitam_offers[offer_conf].offer_log_compaction.expiration_unit |␣
↪default("DAYS") }}
  compactionSize: {{ vitam_offers[offer_conf].offer_log_compaction.compaction_size |␣
↪default(10000) }}

maxBatchThreadPoolSize: {{ vitam_offers[offer_conf]["maxBatchThreadPoolSize"] |␣
↪default(32) }}
batchMetadataComputationTimeout: {{ vitam_offers[offer_conf][
↪"batchMetadataComputationTimeout"] | default(600) }}
```

8.2.16.3.3.2 Fichier `default-storage.conf`

```
provider: {{ vitam_offers[offer_conf]["provider"] }}

{% if vitam_offers[offer_conf]["provider"] in ["filesystem","filesystem-hash"] %}
storagePath: {{ vitam_folder_data }}
{% endif %}

{% if vitam_offers[offer_conf]["provider"] in ["openstack-swift","openstack-swift-v2",
↪"openstack-swift-v3"] %}
swiftKeystoneAuthUrl: {{ vitam_offers[offer_conf]["swiftKeystoneAuthUrl"] | default("
↪") }}
swiftDomain: {{ vitam_offers[offer_conf]["swiftDomain"] | default("") }}
swiftProjectName: {{ vitam_offers[offer_conf]["swiftProjectName"] | default("") }}
swiftUser: {{ vitam_offers[offer_conf]["swiftUser"] | default("") }}
swiftPassword: {{ vitam_offers[offer_conf]["swiftPassword"] | default("") }}
swiftUrl: {{ vitam_offers[offer_conf]["swiftUrl"] | default("") }}
swiftTrustStore: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}
↪.jks
swiftTrustStorePassword: {{ password_truststore }}
swiftMaxConnectionsPerRoute: {{ vitam_offers[offer_conf]["swiftMaxConnectionsPerRoute
↪"] | default(200) }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
swiftMaxConnections: {{ vitam_offers[offer_conf]["swiftMaxConnections"] |␣
↪default(1000) }}
swiftConnectionTimeout: {{ vitam_offers[offer_conf]["swiftConnectionTimeout"] |␣
↪default(200000) }}
swiftReadTimeout: {{ vitam_offers[offer_conf]["swiftReadTimeout"] | default(60000) }}
swiftHardRenewTokenDelayBeforeExpireTime: {{ vitam_offers[offer_conf][
↪"swiftHardRenewTokenDelayBeforeExpireTime"] | default(60) }}
swiftSoftRenewTokenDelayBeforeExpireTime: {{ vitam_offers[offer_conf][
↪"swiftSoftRenewTokenDelayBeforeExpireTime"] | default(300) }}
{% if vitam_offers[offer_conf]["enableCustomHeaders"] is defined and vitam_
↪offers[offer_conf]["enableCustomHeaders"] is sameas true %}
enableCustomHeaders: true
customHeaders:
{% for header in vitam_offers[offer_conf]["customHeaders"] %}
  - key: {{ header.key }}
    value: {{ header.value }}
{% endfor %}
{% endif %}

{% endif %}

{% if vitam_offers[offer_conf]["provider"] == "amazon-s3-v1" %}
s3RegionName: {{ vitam_offers[offer_conf]["s3RegionName"] }}
s3Endpoint: {{ vitam_offers[offer_conf]["s3Endpoint"] }}
s3AccessKey: {{ vitam_offers[offer_conf]["s3AccessKey"] }}
s3SecretKey: {{ vitam_offers[offer_conf]["s3SecretKey"] }}
s3PathStyleAccessEnabled: {{ vitam_offers[offer_conf]["s3PathStyleAccessEnabled"] |␣
↪default(true) }}
s3SignerType: {{ vitam_offers[offer_conf]["s3SignerType"] | default("AWSS3V4SignerType
↪") }}
s3MaxConnections: {{ vitam_offers[offer_conf]["s3MaxConnections"] | default(50) }}
s3ConnectionTimeout: {{ vitam_offers[offer_conf]["s3ConnectionTimeout"] |␣
↪default(10000) }}
s3SocketTimeout: {{ vitam_offers[offer_conf]["s3SocketTimeout"] | default(50000) }}
s3RequestTimeout: {{ vitam_offers[offer_conf]["s3RequestTimeout"] | default(0) }}
s3ClientExecutionTimeout: {{ vitam_offers[offer_conf]["s3ClientExecutionTimeout"] |␣
↪default(0) }}
s3TrustStore: {{ vitam_folder_conf }}/truststore_{{ vitam_struct.vitam_component }}.
↪jks
s3TrustStorePassword: {{ password_truststore }}
{% endif %}

{% if vitam_offers[offer_conf]["provider"] in ["tape-library"] %}
tapeLibraryConfiguration:
  inputFileStorageFolder: "{{ vitam_folder_data }}/inputFiles"
  inputTarStorageFolder: "{{ vitam_folder_data }}/inputTars"
  tmpTarOutputStorageFolder: "{{ vitam_folder_data }}/tmpTarOutput"
  cachedTarStorageFolder: "{{ vitam_folder_data }}/cachedTars"
  maxTarEntrySize: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration"][
↪"maxTarEntrySize"] | default(100000) }}
  maxTarFileSize: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration"][
↪"maxTarFileSize"] | default(1000000) }}
  forceOverrideNonEmptyCartridges: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["forceOverrideNonEmptyCartridges"] | default('false') }}
  cachedTarMaxStorageSpaceInMB: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration
↪"]["cachedTarMaxStorageSpaceInMB"] }}
  cachedTarEvictionStorageSpaceThresholdInMB: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["cachedTarEvictionStorageSpaceThresholdInMB"] }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
  cachedTarSafeStorageSpaceThresholdInMB: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["cachedTarSafeStorageSpaceThresholdInMB"] }}
  maxAccessRequestSize: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration"][
↪"maxAccessRequestSize"] }}
  readyAccessRequestExpirationDelay: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["readyAccessRequestExpirationDelay"] }}
  readyAccessRequestExpirationUnit: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["readyAccessRequestExpirationUnit"] }}
  readyAccessRequestPurgeDelay: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration
↪"]["readyAccessRequestPurgeDelay"] }}
  readyAccessRequestPurgeUnit: {{ vitam_offers[offer_conf]["tapeLibraryConfiguration
↪"]["readyAccessRequestPurgeUnit"] }}
  accessRequestCleanupTaskIntervalDelay: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["accessRequestCleanupTaskIntervalDelay"] }}
  accessRequestCleanupTaskIntervalUnit: {{ vitam_offers[offer_conf][
↪"tapeLibraryConfiguration"]["accessRequestCleanupTaskIntervalUnit"] }}

  topology:
    buckets:
 {% for bucket in vitam_offers[offer_conf]["topology"]["buckets"] %}
        {{ bucket.name }}:
          tenants: {{ bucket.tenants }}
          tarBufferingTimeoutInMinutes: {{ bucket.tarBufferingTimeoutInMinutes }}
 {% endfor %}

  tapeLibraries:
{% for library in vitam_offers[offer_conf]["tapeLibraries"] %}
    {{ library.name }}:
      robots:
{% for robot in library.robots %}
        -
          device: {{ robot.device }}
          mtxPath: "{{  robot.mtxPath }}"
          timeoutInMilliseconds: {{ robot.timeoutInMilliseconds }}
{% endfor %}
      drives:
{% for drive in library.drives %}
        -
          index: {{ drive.index }}
          device: {{ drive.device }}
          mtPath: "{{ drive.mtPath }}"
          ddPath: "{{ drive.ddPath }}"
          timeoutInMilliseconds: {{ drive.timeoutInMilliseconds }}
          readWritePriority: {{ drive.readWritePriority | default('WRITE') }}
{% endfor %}
      fullCartridgeDetectionThresholdInMB: {{ library.
↪fullCartridgeDetectionThresholdInMB }}

{% endfor %}
{% endif %}
```

L'arborescence de stockage des fichiers dans l'offre est décrite dans le *DAT*.

8.2.16.3.4 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-offer`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-offer`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/offer/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes
- cas des batches

N/A

8.2.17 Technical administration

8.2.17.1 Présentation

8.2.18 Worker

8.2.18.1 Introduction

Le but de cette documentation est d'expliquer la configuration et l'exploitation du module **worker**.

8.2.18.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/worker`.

8.2.18.2.1 Fichier `batch-report-client.conf`

```
serverHost: {{ vitam.batchreport.host }}
serverPort: {{ vitam.batchreport.port_service }}
secure: {{ vitam.batchreport.https_enabled }}
```

8.2.18.2.2 Fichier `format-identifiers.conf`

Ce fichier permet de définir l'URL d'accès à Siegfried.

```
siegfried-local:
  type: SIEGFRIED
  client: http
  host: localhost
  port: {{ siegfried.port }}
  rootPath: {{ vitam_folder_tmp }}/
  versionPath: {{ vitam_folder_data }}/version/folder
```

8.2.18.2.3 Fichier `functional-administration-client.conf.j2`

Ce fichier permet de définir l'accès à functional-administration.

```
serverHost: {{ vitam.functional_administration.host }}
serverPort: {{ vitam.functional_administration.port_service }}
```

8.2.18.2.4 Fichier `metadata-client.conf`

Ce fichier permet de définir l'accès au metadata.

```
serverHost: {{ vitam.metadata.host }}
serverPort: {{ vitam.metadata.port_service }}
```

8.2.18.2.5 Fichier `storage-client.conf`

Ce fichier permet de définir l'accès au storage.

```
serverHost: {{ vitam.storageengine.host }}
serverPort: {{ vitam.storageengine.port_service }}
```

8.2.18.2.6 Fichier `verify-timestamp.conf`

```
# Configuration - verify timestamp
p12LogbookPassword: {{ keystores.timestamping.secure_logbook }}
p12LogbookFile: keystore_secure-logbook.p12
```

8.2.18.2.7 Fichier `version.conf`

```
binaryDataObjectVersions:
- BinaryMaster
- Dissemination
- Thumbnail
- TextContent
physicalDataObjectVersions:
- PhysicalMaster
- Dissemination
```

8.2.18.2.8 Fichier `worker.conf`

Ce fichier permet de définir le paramétrage du composant worker.

```
# Configuration processing
# HERE MUST BE MY (WORKER) current configuration
registerServerHost: {{ ip_service }}
registerServerPort: {{ vitam_struct.port_service }}
# Configuration handler
processingUrl: {{vitam.processing | client_url}}
urlMetadata: {{vitam.metadata | client_url}}
urlWorkspace: {{vitam.workspace | client_url}}
# Configuration jetty
jettyConfig: jetty-config.xml
#Configuration parallele
capacity: {{ vitam_worker_capacity }}
{% if vitam_worker_workerFamily is defined %}
workerFamily: {{ vitam_worker_workerFamily }}
{% endif %}

indexInheritedRulesWithAPIV2OutputByTenant: [ "{{  vitam.worker.api_output_index_
↪tenants | join('", "') }}" ]
indexInheritedRulesWithRulesIdByTenant: [ "{{  vitam.worker.rules_index_tenants |␣
↪join('", "') }}" ]

# Archive Unit Profile cache settings (max entries in cache & retention timeout in␣
↪seconds)
archiveUnitProfileCacheMaxEntries: {{ vitam.metadata.
↪archiveUnitProfileCacheMaxEntries }}
archiveUnitProfileCacheTimeoutInSeconds: {{ vitam.worker.
↪archiveUnitProfileCacheTimeoutInSeconds }}

# Schema validator cache settings (max entries in cache & retention timeout in␣
↪seconds)
schemaValidatorCacheMaxEntries: {{ vitam.metadata.schemaValidatorCacheMaxEntries }}
schemaValidatorCacheTimeoutInSeconds: {{ vitam.worker.
↪schemaValidatorCacheTimeoutInSeconds }}
```

Paramètres obligatoires :

- **processingUrl** : URL de connexion au compsoant Vitam processing
- **urlMetadata** : URL de connexion au composant VITAM metadata
- **urlWorkspace** : URL de connexion au composant VITAM workspace
- **registerServerHost** : host ou le worker déployé
- **registerServerPort** : port ou le worker déployé
- **jettyConfig** : le fichier config jetty associé au service du worker

Paramètres optionnels :

- **workerFamily** : la famille dont le worker appartant en fonction de tache exécutée
- **capacity** : capacité du worker en mode parallèle de tache (par défaut à 1 dans l'ansiblerie, si non définie)

8.2.18.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-workspace`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-workspace`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/workspace/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

8.2.19 Workspace

8.2.19.1 Présentation

Rôle :

- Fourniture d'un espace pour l'échange de fichiers (et faire un appel par pointeur lors des appels entre composants) entre les différents composants de Vitam
- Fourniture d'un espace pour le stockage temporaire de données (exports DIP disponibles en téléchargement)

Fonctions :

- Utilisation du moteur de stockage dans un mode minimal (Opérations CREATE, READ, DELETE sur 1 seule offre de stockage)

8.2.19.2 Configuration / fichiers utiles

Les fichiers de configuration sont gérés par les procédures d'installation ou de mise à niveau de l'environnement *VITAM*. Se référer au *DIN*.

Les fichiers de configuration sont définis sous `/vitam/conf/workspace`.

8.2.19.2.1 Fichier `workspace.conf`

```
storagePath: {{ vitam_folder_data }}
jettyConfig: jetty-config.xml
provider: filesystem
contextPath: {{ vitam_struct.context_path }}
```

8.2.19.3 Opérations

- Démarrage du service

En tant qu'utilisateur root : `systemctl start vitam-workspace`

- Arrêt du service

En tant qu'utilisateur root : `systemctl stop vitam-workspace`

- Sauvegarde du service

Ce service ne nécessite pas de sauvegarde particulière.

- Supervision du service

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port>/workspace/v1/status`

Contrôler le retour HTTP 200 sur l'URL `<protocole web https ou https>://<host>:<port admin>/admin/v1/status`

- Exports

N/A

- gestion de la capacité

N/A

- actions récurrentes

- cas des batches

N/A

CHAPITRE 9

Intégration d'une application externe dans Vitam

9.1 Prérequis

L'application externe devra être en mesure de requêter les composants *VITAM* `ingest-external` et `access-external` sur leurs ports de service respectifs par le protocole HTTPS.

Il faut donc prévoir une ouverture de flux réseau pour le protocole TCP (selon l'infrastructure en place) sur les ports de service des composants *VITAM* `ingest-external` et `access-external`.

La sécurisation des connexions HTTP avec les applications externes est déléguée aux composants *VITAM* `ingest-external` et `access-external` (ou bien éventuellement à un reverse-proxy, selon l'infrastructure de déploiement *VITAM* retenue).

La création d'un certificat TLS client pour l'application externe est requise afin de permettre l'habilitation et l'authentification applicative des requêtes émises depuis l'application externe vers les composants *VITAM* ingest-external et access-external.

9.2 Intégration de certificats clients de VITAM

Note : Cette section décrit l'ajout de certificats *SIA* ou personnels (*Personae*) dans le cadre des procédures d'exploitation de la solution logicielle *VITAM*. L'intégration de certificats clients de *VITAM* au déploiement est décrite dans le document d'installation (*DIN*).

9.2.1 Authentification applicative SIA

Le certificat client de l'application externe doit être ajouté dans la base de données du composant `security-internal`, afin de permettre la gestion des habilitations et l'authentification applicative de l'application externe.

L'exemple suivant permet d'ajouter un certificat présent sous `/path/to/certificate` et associé au contexte applicatif portant l'identifiant `contexte_identifier`.

9.2.1.1 Ajout d'un certificat pour l'authentification applicative SIA

```
curl -XPOST -H "Content-Type:application/json" 'http://<ip admin security-internal>:
↪<port admin security-internal>/v1/api/identity' -d "
{
    \"contextId\":\"contexte_identifier\",
    \"certificate\":\"$(base64 /path/to/certificate)\"
}"
```

Note : Se repporter à la documentation sur la gestion des habilitations pour ce qui concerne l'ajout de contextes applicatifs.

Avertissement : Lorsque l'authentification du client, par le protocole TLS, est réalisée, la *CA* du certificat client doit être déployée dans le *truststore* des composants *VITAM* `ingest-external` et `access-external`. Dans ce cas de figure, suivre la procédure de la section *Mise à jour des certificats* (page 15), en ayant pris soin au préalable de déposer les certificats de la chaîne de certification client dans le répertoire *environments/certs/client-external/ca*. Si le certificat client est passé dans le *header* http (cas de l'authentification applicative par *header* http X-SSL-CLIENT-CERT), le certificat client n'est alors pas utilisé dans la négociation TLS et il n'est donc pas nécéssaire d'inclure la *CA* associée dans le *truststore* des composants *VITAM* `ingest-external` et `access-external`.

9.2.2 Authentification *Personae*

Le certificat personnel (*Personae*) doit être ajouté dans la base de données du composant `security-internal`, afin de permettre l'authentification renforcée de l'utilisateur.

Les exemples suivants permettent d'ajouter ou supprimer un certificat présent sous `/path/to/certificate`.

9.2.2.1 Ajout d'un certificat pour l'authentification *Personae*

```
curl -XPOST -H "Content-type: application/octet-stream" --data-binary @/path/to/
↪certificate 'http://<ip admin security-internal>:<port admin security-internal>/v1/
↪api/personalCertificate'
```

9.2.2.2 Suppression d'un certificat pour l'authentification *Personae*

```
curl -XDELETE -H "Content-type: application/octet-stream" --data-binary @/path/to/
↪certificate 'http://<ip admin security-internal>:<port admin security-internal>/v1/
↪api/personalCertificate'
```

9.3 Révocation de certificats clients de VITAM

La release « R8 » introduit une nouvelle fonctionnalité permettant la révocation des certificats *SIA* et *Personae* afin d'empecher des accès non autorisés aux *API* de la solution logicielle *VITAM* (vérification dans la couche https des *CRL*).

Le fonctionnement de la validation des certifcats de la solution logicielle *VITAM* *SIA* et *Personae* par *CRL* est le suivant :

- L'administrateur transmet à la solution logicielle *VITAM* le *CRL* d'un *CA* qui a émis le certificat présent dans la solution logicielle *VITAM*, via le point d'API suivant

```
http://{{ hosts_security_internal }}:{{vitam.security_internal.port_admin}}/v1/
↪api/crl
```

Prudence : La CRL fournie doit être obligatoirement au format DER (cf. http://www.ietf.org/rfc/rfc3280.txt »>RFC 3280 : *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*)

Exemple :

```
curl -v -X POST -u {{ admin_basic_auth_user }}:{{ admin_basic_auth_password }}  http:/
↪/{{ hosts_security_internal }}:{{vitam.security_internal.port_admin}}/v1/api/crl -H
↪'Content-Type: application/octet-stream' --data-binary @/path/to/crl/my.crl
```

Le paramètre `adminUser` correspond à la valeur `admin_basic_auth_user` déclarée dans le fichier `vitam_security.yml`

Le paramètre `adminPassword` correspond à la valeur `admin_basic_auth_password` déclarée dans le fichier `vault-vitam.yml`

- Le système va contrôler tous les certificats (collections `identity.Certificate` et `identity.PersonalCertificate`) émis par le *IssuerDN* correspondant à la *CRL*, en vérifiant si ces derniers sont révoqués ou non. Si c'est le cas, alors la solution logicielle *VITAM* positionne le statut du certificat révoqué à **REVOKED**. Cela a pour conséquence le rejet de tout accès aux *API* *VITAM* avec utilisation du certificat révoqué (les filtres de sécurité émettront des exceptions dans les journaux de *log*).
- Une alerte de sécurité est émise dans les journaux en cas de révocation.

9.4 Déploiement des keystores

9.4.1 Vitam n'est pas encore déployé

Déployer Vitam en suivant la procédure indiquée dans le *DIN*.

9.4.2 Vitam est déjà déployé

Suivre la procédure de la section *Mise à jour des certificats* (page 15).

CHAPITRE 10

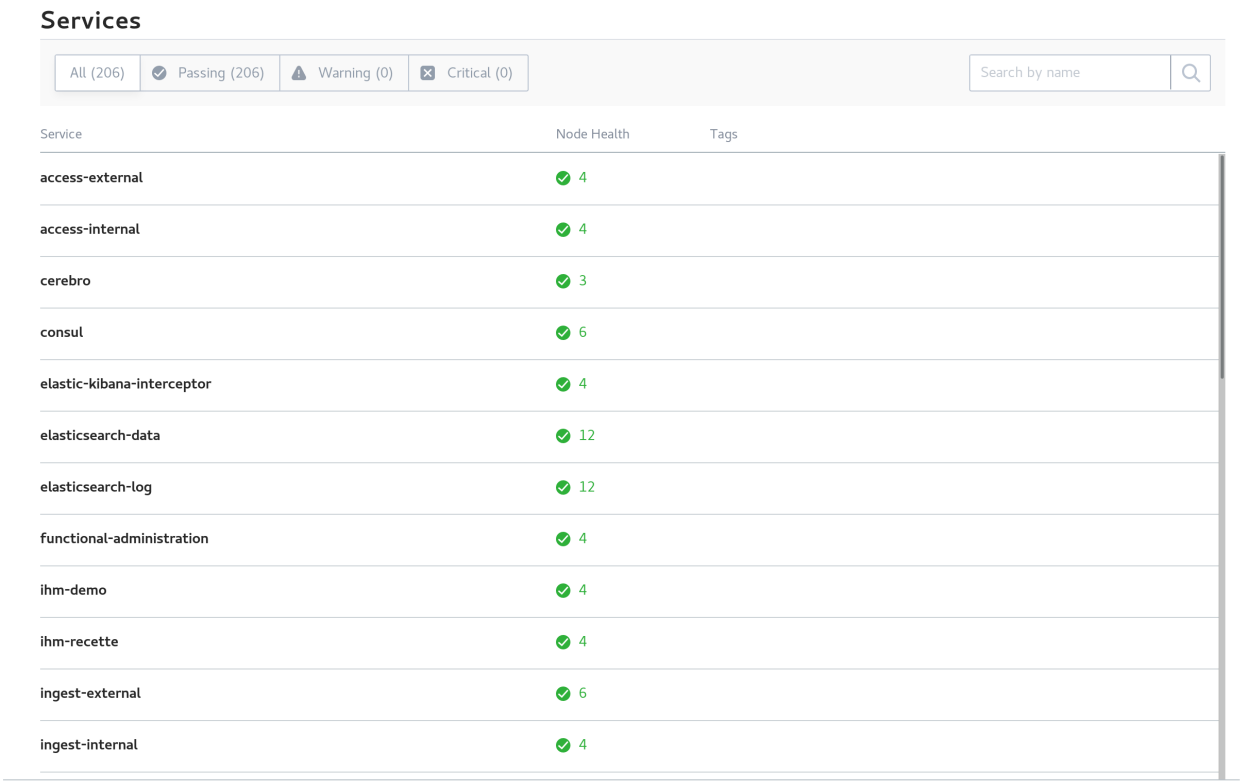
Aide à l'exploitation

10.1 Analyse de premier niveau

Cette section a pour but de présenter les premiers outils à utiliser pour réaliser une analyse de premier niveau, en cas de problème avec la solution logicielle *VITAM*.

10.1.1 Etat par Consul

Se connecter à l'IHM de Consul et recenser les états des composants de la solution logicielle *VITAM*.



A l'heure actuelle, tous les composants doivent avoir un statut de couleur verte. Si ce n'est pas le cas :

1. seul un composant est KO, alors redémarrer le composant incriminé
2. si plusieurs services sont KO, suivre la procédure de redémarrage de VITAM
3. si tous les « check-DNS » (visible dans le détail des checks de chaque service) sont KO, s'assurer que, sur les machines hébergeant VITAM, le fichier `/etc/resolv.conf` contient, en début de fichier, la ligne : `nameserver 127.0.0.1`.

10.1.2 Etat par Kibana

Se connecter à Kibana, aller dans « Dashboards ». Cliquer sur le bouton « Load Saved Dashboard » et sélectionner « Composants VITAM ». Eventuellement, changer la résolution (en haut à droite, par défaut, réglé sur les 15 dernières minutes).

Sur « pie-logback-error-level », cliquer sur la section de camembert d'intérêt (ERROR) et regarder, en bas de page, les éventuelles erreurs remontées dans Kibana.

10.2 Playbook ansible pour échanger avec le support

Afin de simplifier les échanges avec le support *VITAM*, un playbook d'exploitation a été développé pour collecter automatiquement les informations suivantes :

- la récupération des informations machines de la solution logicielle *VITAM*
- l'état Consul des composants
- la récupération des traces applicatives (fichiers log)
- l'état des clusters Elasticsearch

- la possibilité, au choix de l'exploitant, de fournir également les clés publiques des certificats

À l'issue de l'exécution de ce playbook, les données collectées sont compactées et accessibles à l'emplacement suivant selon la règle de nommage : `environments/troubleshoot_<date>-<log_files_age>_<vitam_site_name>.zip`

Ce fichier pourra ainsi être communiqué à *VITAM* dans le cadre d'une demande de Support.

La commande pour générer le fichier est à lancer depuis le répertoire `deployment`

```
ansible-playbook ansible-vitam-exploitation/troubleshoot.yml -i environments/hosts.
↪<environnement> --ask-vault-pass
```

Certaines questions seront demandées durant l'exécution du playbook. Si vous souhaitez automatiser l'exécution de cette commande, voici les paramètres que vous pouvez passer en extra-vars :

`--extra-vars "confirmation=YES log_files_age=1"`

- confirmation [YES, default :NO] : Pour confirmer le lancement du playbook.
- log_files_age [default :2] : Pour définir le nombre de jours de logs à récupérer (doit être >0).

Paramètres optionnels :

`--extra-vars "sync_type=rsync get_crt=YES excludes=gc*"`

- sync_type [default :fetch, rsync] : Pour définir la méthode de récupération des fichiers de logs. La méthode rsync va installer le package rsync sur les machines et nécessitera l'ouverture des ports réseaux associés au niveau des Firewall pour fonctionner.
- get_crt [YES, default :NO] : Pour confirmer la récupération des certificats dans le troubleshoot.
- excludes [xxx*,yyy*, default :""] : Pour exclure certains fichiers à récupérer dans le troubleshoot.

10.3 Identification des AU non conformes

Le schéma JSON des AU a été corrigé et rationalisé. Toute donnée issue d'un Ingest d'une release précédente est conforme à cette expression corrigée du schéma. Dans les versions précédentes, le DSL autorisait des modifications non conformes au SEDA. Ce n'est plus possible dans la présente version.

Si des modifications non conformes ont eu lieu, alors des AU non conformes peuvent donc se retrouver en base. Lorsque cela arrive : * L'export DIP de l'AU peut échouer. * La modification d'un champ A rapportera une erreur sur un champs B non-conforme. La correction de ce problème se fera avec une requête DSL qui modifie le champ A et le champs B (en lui donnant une valeur conforme). * Maintenant les champs SEDA sont soit des tableaux, soit des valeurs simples (string ou objet). Certains champs qui pouvaient s'écrire sous forme de valeurs non tabulaires, doivent maintenant être écrits sous forme de tableau, même s'ils n'ont qu'un seul élément. Ex : Coverage.Spatial.

Cette procédure scriptée permet de détecter l'ensemble des AU non conformes. Les retours associés pourront être transmis au support VITAM (assistance@programmevitam.fr).

La commande pour générer le fichier est à lancer depuis le répertoire `deployment`

```
ansible-playbook ansible-vitam-exploitation/check_unit_compatibility.yml -i␣
↪environments/hosts.<environnement> --ask-vault-pass
```

A l'issue, si des *AU* sont considérées en erreur, se rapprocher du métier pour réaliser une première analyse / tentative de correction, par correction des données incorrectes.

Si l'erreur persiste, contacter le support.

CHAPITRE 11

Questions Fréquemment Posées

11.1 Présentation

Cette section a vocation à répertorier les différents problèmes rencontrés et apporter la solution la plus appropriée ; elle est amenée à être régulièrement mise à jour pour répertorier les problèmes rencontrés.

11.2 Retour d'expérience / cas rencontrés

11.2.1 Crash rsyslog, code killed, signal : BUS

Il a été remarqué chez un partenaire du projet Vitam, que rsyslog se faisait *killer* peu après son démarrage par le signal SIGBUS. Il s'agit très probablement d'un bug rsyslog <= 8.24 https://github.com/rsyslog/rsyslog/issues/1404

Pour fixer ce problème, il est possible d'upgrader rsyslog sur une version plus à jour en suivant cette documentation :

- Centos [18]
- Debian [19]

11.2.2 Mongo-express ne se connecte pas à la base de données associée

Si mongoDB a été redémarré, il faut également redémarrer mongo-express.

11.2.3 Elasticsearch possède des shard non alloués (état « UNASSIGNED »)

Lors de la perte d'un noeud d'un cluster elasticseach, puis du retour de ce noeud, certains shards d'elasticseach peuvent rester dans l'état `UNASSIGNED` ; dans ce cas, cerebro affiche les shards correspondant en gris (au-dessus des noeuds) dans la vue « cluster », et l'état du cluster passe en « yellow ». Il est possible d'avoir plus d'informations sur la

[18] https://www.rsyslog.com/rhelcentos-rpms/
[19] https://www.rsyslog.com/debian-repository/

cause du problème via une requête `POST` sur l'API elasticsearch `_cluster/reroute?explain`. Si la cause de l'échec de l'assignation automatique a été résolue, il est possible de relancer les assignations automatiques en échec via une requête `POST` sur l'API `_cluster/reroute?retry_failed`. Dans le cas où l'assignation automatique ne fonctionne pas, il est nécessaire de faire l'assignation à la main pour chaque shard incriminé (requête `POST` sur `_cluster/reroute`) :

```
{
    "commands": [
        {
            "allocate": {
                "index": "topbeat-2016.11.22",
                "shard": 3,
                "node": "vitam-iaas-dblog-01.int"
            }
        }
    ]
}
```

Cependant, un shard primaire ne peut être réalloué de cette manière (il y a risque de perte de données). Si le défaut d'allocation provient effectivement de la perte puis de la récupération d'un noeud, et que TOUS les noeuds du cluster sont de nouveaux opérationnels et dans le cluster, alors il est possible de forcer la réallocation sans perte.

```
{
    "commands": [
        {
            "allocate": {
                "index": "topbeat-2016.11.22",
                "shard": 3,
                "node": "vitam-iaas-dblog-01.int",
                "allow_primary": "true"
            }
        }
    ]
}
```

Sur tous ces sujets, Cf. la documentation officielle [20].

11.2.4 Elasticsearch possède des shards non initialisés (état « INITIALIZING »)

Tout d'abord, il peut être difficile d'identifier les shards en questions dans cerebro ; une requête HTTP `GET` sur l'API `_cat/shards` permet d'avoir une liste plus compréhensible. Un shard non initialisé correspond à un shard en cours de démarrage (Cf. une ancienne page de documentation [21]. Si les shards non initialisés sont présents sur un seul noeud, il peut être utile de redémarrer le noeud en cause. Sinon, une investigation plus poussée doit être menée.

11.2.5 Elasticsearch est dans l'état « *read-only* »

Lorsque Elasticsearch répond par une erreur 403 et que le message suivant est observé dans les logs `ClusterBlockException[blocked by: [FORBIDDEN/xx/index read-only / allow delete (api)];`, cela est probablement consécutif à un remplissage à 100% de l'espace de stockage associé aux index Elasticsearch. Elasticsearch passe alors en lecture seule s'il ne peut plus indexer de documents et garantit ainsi la disponibilité des requêtes en lecture seule uniquement.

Afin de rétablir Elasticsearch dans un état de fonctionnement nominal, il vous faudra alors exécuter la requête suivante :

https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-reroute.html
https://www.elastic.co/guide/en/elasticsearch/reference/1.4/states.html

```
curl -XPUT -H "Content-Type: application/json" http://<es-host>:<es-port>/_all/_
↪settings -d '{"index.blocks.read_only_allow_delete": null}'
```

11.2.6 MongoDB semble lent

Pour analyser la performance d'un cluster MongoDB, ce dernier fournit quelques outils permettant de faire une première analyse du comportement : mongostat [22] et mongotop [23] .

Dans le cas de VITAM, le cluster MongoDB comporte plusieurs shards. Dans ce cas, l'usage de ces deux commandes peut se faire :

- soit sur le cluster au global (en pointant sur les noeuds mongos) : cela permet d'analyser le comportement global du cluster au niveau de ses points d'entrées ;

```
mongostat --host <ip_service> --port 27017 --username vitamdb-admin --
↪password <password ; défaut : azerty> --authenticationDatabase admin
mongotop --host <ip_service> --port 27017 --username vitamdb-admin --
↪password <password ; défaut : azerty> --authenticationDatabase admin
```

- soit directement sur les noeuds de stockage (mongod) : cela donne des résultats plus fins, et permet notamment de séparer l'analyse sur les noeuds primaires & secondaires d'un même replicaset.

```
mongotop --host <ip_service> --port 27019 --username vitamdb-localadmin --
↪password <password ; défaut : qwerty> --authenticationDatabase admin
mongostat --host <ip_service> --port 27019 --username vitamdb-localadmin -
↪-password <password ; défaut : qwerty> --authenticationDatabase admin
```

D'autres outils sont disponibles directement dans le client mongo, notamment pour troubleshooter les problèmes dûs à la réplication [24] :

```
mongo --host <ip_service> --port 27019 --username vitamdb-localadmin --password
↪<password ; défaut : qwerty> --authenticationDatabase admin
> rs.printSlaveReplicationInfo()
> rs.printReplicationInfo()
> db.runCommand( { serverStatus: 1 } )
```

D'autres commandes plus complètes existent et permettent d'avoir plus d'informations, mais leur analyse est plus complexe :

```
# returns a variety of storage statistics for a given collection
> use metadata
> db.stats()
> db.runCommand( { collStats: "Unit" } )
```

Enfin, un outil est disponible en standard afin de mesurer des performances des lecture/écritures avec des patterns proches de ceux utilisés par la base de données (mongoperf [25]) :

```
echo "{nThreads:16,fileSizeMB:10000,r:true,w:true}" | mongoperf
```

https://docs.mongodb.com/manual/reference/program/mongostat/
https://docs.mongodb.com/manual/reference/program/mongotop/
https://docs.mongodb.com/manual/tutorial/troubleshoot-replica-sets
https://docs.mongodb.com/manual/reference/program/mongoperf/

11.2.7 Les shards de MongoDB semblent mal équilibrés

Normalement, un processus interne à MongoDB (le `balancer`) s'occupe de déplacer les données entre les shards (par `chunk`) pour équilibrer la taille de ces derniers. Les commandes suivantes (à exécuter dans un shell mongo sur une instance mongos - attention, ces commandes ne fonctionnent pas directement sur les instances mongod) permettent de s'assurer du bon fonctionnement de ce processus :

- `sh.status()` : donne le status du sharding pour le cluster complet ; c'est un bon premier point d'entrée pour connaître l'état du balancer.
- `use <dbname>`, puis `db.<collection>.getShardDistribution()`, en indiquant le bon nom de base de données (ex : `metadata`) et de collection (ex : `Unit`) : donne les informations de répartition des chunks dans les différents shards pour cette collection.

11.2.8 L'importation initiale (profil de sécurité, certificats) retourne une erreur

Les playbooks d'initialisation importent des éléments d'administration du système (profils de sécurité, certificats) à travers des APIs de la solution VITAM. Cette importation peut être en échec, par exemple à l'étape `TASK [init_contexts_and_security_profiles : Import admin security profile to functionnal-admin]`, avec une erreur de type 400. Ce type d'erreur peut avoir plusieurs causes, et survient notamment lors de redéploiements après une première tentative non réussie de déploiement ; même si la cause de l'échec initial est résolue, le système peut se trouver dans un état instable. Dans ce cas, un déploiement complet sur environnement vide est nécessaire pour revenir à un état propre.

Une autre cause possible ici est une incohérence entre l'inventaire, qui décrit notamment les offres de stockage liées aux composants offer, et le paramétrage `vitam_strategy` porté par le fichier `offers_opts.yml`. Si une offre indiquée dans la stratégie n'existe nulle part dans l'inventaire, le déploiement sera en erreur. Dans ce cas, il faut remettre en cohérence ces paramètres et refaire un déploiement complet sur environnement vide.

11.2.9 Problème d'ingest et/ou d'access

Si vous repérez un message de ce type dans les log *VITAM* :

```
fr.gouv.vitam.common.security.filter.RequestAuthorizationValidator.
↪checkTimestamp(AuthorizationWrapper.java:102) : [vitam-env-int8-app-04.vitam-
↪env:storage:239079175] Timestamp check failed. 16s
fr.gouv.vitam.common.security.filter.RequestAuthorizationValidator.
↪checkTimestamp(AuthorizationWrapper.java:107) : [vitam-env-int8-app-04.vitam-
↪env:storage:239079175] Critical timestamp check failure. 61s
```

Il faut vérifier / corriger l'heure des machines hébergeant la solution logicielle *VITAM*.

Prudence : Si un *delta* de temps important (10s par défaut) a été détecté entre les machines, des erreurs sont tracées dans les logs et une alerte est remontée dans le dashboard Kibana des Alertes de sécurité.

Au delà d'un seuil critique (60s par défaut) d'écart de temps entre les machines, les requêtes sont systématiquement rejetées, ce qui peut causer des dysfonctionnements majeurs de la solution.

11.3 Erreur d'inconsistance des données MongoDB / ES

En cas de détection d'un problème de synchronisation des données entre les bases de données Elasticsearch-data (cluster d'indexation dédié aux données métier) et les bases de données MongoDB-data (replicaset MongoDB stockant les

données métier de Vitam) avec un message d'erreur du type : « An internal data consistency error has been detected », la procédure suivante pourra être appliquée : *Réindexation* (page 52).

CHAPITRE 12

Annexes

12.1 Cycle de vie des certificats

Le tableau ci-dessous indique le mode de fonctionnement actuel pour les différents certificats et *CA*. Précisions :

- Les « procédures par défaut » liées au cycle de vie des certificats dans la présente version de la solution *VITAM* peuvent être résumées ainsi :
  - Création : génération par *PKI* partenaire + copie dans répertoires de déploiement + script `generate_stores.sh` + déploiement ansible
  - Suppression : suppression dans répertoires de déploiement + script `generate_stores.sh` + déploiement ansible
  - Renouvellement : regénération par *PKI* partenaire + suppression / remplacement dans répertoires de déploiement + script `generate_stores.sh` + redéploiement ansible
- Il n'y a pas de contrainte au niveau des *CA* utilisées (une *CA* unique pour tous les usages *VITAM* ou plusieurs *CA* séparées – cf. *DAT*). On appelle ici :
  - « *PKI* partenaire » : *PKI* / *CA* utilisées pour le déploiement et l'exploitation de la solution *VITAM* par le partenaire.
  - « *PKI* distante » : *PKI* / *CA* utilisées pour l'usage des frontaux en communication avec le back office *VITAM*.

Classe	Type	Usages	Origine	Création	Suppression	Renouvellement
Interne	*CA*	ingest & access	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	*CA*	offer	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Horodatage	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Storage (*Swift*)	Offre de stockage	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Storage (s3)	Offre de stockage	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	ingest	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	access	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	offer	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Timestamp	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
IHM demo	*CA*	ihm-demo	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
IHM demo	Certif	ihm-demo	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
SIA	*CA*	Appel *API*	*PKI* distante	*proc. par défaut* (*PKI* distante)	*proc. par défaut*	*proc. par défaut* (*PKI* distante)+recharger Certifs
SIA	Certif	Appel *API*	*PKI* distante	Génération + copie répertoire + deploy(par la suite appel *API* d'insertion)	Suppression Mongo	Suppression Mongo + *API* d'insertion
Personae	Certif	Appel *API*	*PKI* distante	*API* ajout	*API* suppression	*API* suppression + *API* ajout

Remarques :

- Lors d'un renouvellement de *CA SIA*, il faut s'assurer que les certificats qui y correspondaient soient retirés de MongoDB et que les nouveaux certificats soient ajoutés par le biais de l'" *API* dédiée.
- Lors de toute suppression ou remplacement de certificats *SIA*, s'assurer que la suppression ou remplacement des contextes associés soit également réalisé.
- L'expiration des certificats n'est pas automatiquement prise en charge par la solution *VITAM* (pas de notification en fin de vie, pas de renouvellement automatique). Pour la plupart des usages, un certificat expiré est proprement rejeté et la connexion ne se fera pas ; les seules exceptions sont les certificats *Personae*, pour lesquels la validation de l'arborescence *CA* et des dates est à charge du front office en interface avec *VITAM*.

12.2 Gestion des anomalies en production

Les anomalies empêchant le bon fonctionnement de la solution *VITAM* déjà déployée dans un système en production sont gérées par le programme VITAM selon un processus dédié. Il reprend la terminologie du « Contrat de Service VITAM ».

12.2.1 Numérotation des versions

A partir de la version 1 de la solution VITAM, la numérotation des versions du logiciel est du type *X.Y.Z(-P)* selon les principes suivants :

- X : version majeure de la solution VITAM. Elle suit le calendrier des versions majeures, construit de concert avec les partenaires.
- Y : version mineure de la solution VITAM. Elle suit le calendrier des itérations, typiquement une itération dure trois semaines.
- Z : version *bugfix* de la solution VITAM. Elle suit le calendrier des itérations, typiquement une itération à chaque trois semaines.
- Seules les versions maintenues continuent de bénéficier de nouvelles versions *bugfix*.
- P : patch de la solution VITAM. Un patch correspond à la mise à disposition, entre deux releases, de binaires et/ou fichiers de configuration et de déploiement, pour corriger des bugs bloquants.
- Seules les versions maintenues continuent de bénéficier de patchs.

12.2.2 Mise à disposition du logiciel

La solution VITAM est mise à disposition des partenaires selon le calendrier suivant :

- Des *releases* sont mises à disposition des partenaires et du grand public régulièrement, typiquement une release pour cinq itérations de développement. Il s'agit alors de la version mineure courante. Pour rappel, la version 1.0.0 correspond à la release 6 (*R6*).
- Les versions *bugfix* de chaque version maintenue sont mises à disposition des partenaires et du grand public régulièrement, à chaque itération (s'il y a eu des anomalies corrigées dans la période).
- Les patchs de chaque version maintenue sont mis à disposition des partenaires à chaque fois qu'une anomalie de production critique est identifiée et corrigée. Les correctifs correspondant aux patchs sont ensuite inclus dans une version *bugfix* ultérieure.

12.2.3 Gestion des patchs

L'objectif d'un patch est de rétablir au plus vite le fonctionnement en production des systèmes partenaires. La livraison se limite ainsi aux packages (RPM / DEB) concernés par la correction, avec les fichiers de déploiement et de configuration nécessaires. Les instructions pour « patcher » l'applicatif sont également mises à disposition, en fonction du périmètre impacté (simple arrêt / relance ; purges ; scripts de déploiement. . .).

Les patchs sont mis à disposition des partenaires sur un dépôt en ligne. L'objectif est d'offrir la possibilité pour les partenaires d'automatiser la récupération des packages mis à jour, et éventuellement de pouvoir reconstituer un packaging complet de Vitam.

Note : Ce choix de gestion de patchs implique des numéros de version qui pourront être différents entre chaque paquet. Le réalignement se fait au niveau des versions *bugfix* ou mineures.

La mise à disposition du code source du patch est considérée comme moins critique et se réalise dans un second temps, sur Github.

Table des figures

Liste des tableaux

Index

S

T

U

V

W