VITAM - Documentation de montées de version

Version 5.rc.1

VITAM

nov. 23, 2021

Table des matières

CHAPITRE 1

Introduction

1.1 Objectif de ce document

Ce document décrit les procédures et informations utiles à une équipe d'exploitants de *VITAM* afin de réaliser les montées de version de la solution logicielle.

Il s'adresse aux personnes suivantes :

- Les architectes techniques des projets désirant intégrer la solution logicielle *VITAM* ;
- Les exploitants devant installer et/ou exploiter la solution logicielle *VITAM*.

Note : Ce document ne décrit que les chemins de montées de versions vers les versions *VITAM* maintenues. Se référer au chapitre *Généralités sur les versions* (page 6) pour plus d'informations.

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.

[5] 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

Généralités sur les versions

La numérotation des versions logicielles *VITAM* respecte le schéma suivant : X.Y.Z(-P).

- **X** : version majeure (V1, V2, V3)
- **Y** : version mineure (de type *release*, intitulées « R.Y . », contenant les nouvelles fonctionnalités)
- **Z** : version *bugfix*
- **P** : patch suite à bug critique (ne porte que sur les composants impactés)

Tableau 1: Tableau récapitulatif des versions de la solution logicielle VITAM

Code release	Version générale	Version associée	Version LTS	Version dépréciée
R6	1	1.0.x		X
R7	1	1.4.x		X
R8	1	1.10.x		X
R9	2	2.1.x	X	
R10	2	2.6.x		X
R11	2	2.11.x		X
R12	2	2.15.x		
R13	3	3.0.x	X	
R16	4	4.0.x	X	
R17	4	4.5.x		X
5rc	5.rc	5.rc.x	X	

La *release* R9 est la version *LTS* de la version majeure V2 *VITAM*.

Pour plus d'informations, se reporter au chapitre « Maintenance » du document de Présentation de la solution logicielle VITAM [11].

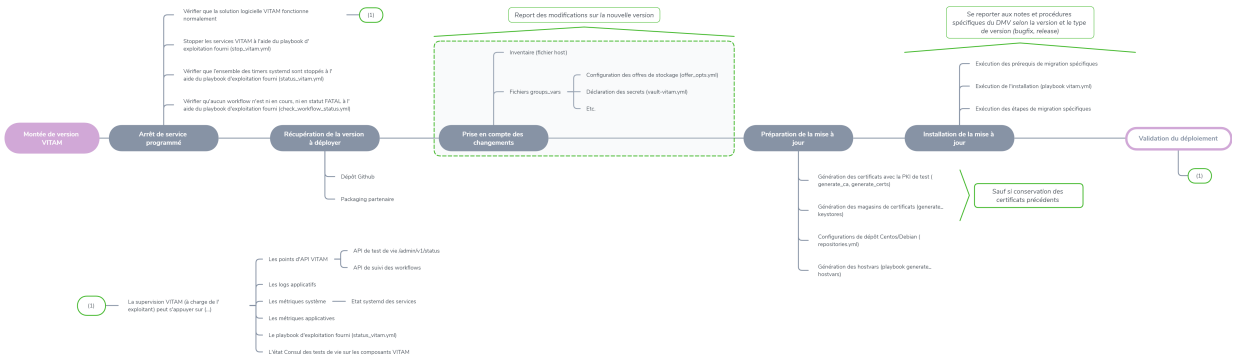
http://www.programmevitam.fr/ressources/DocCourante/autres/fonctionnel/VITAM_Presentation_solution_logicielle.pdf

CHAPITRE 4

Montées de version

4.1 Principes généraux

Le schéma ci-dessous décrit le principe général d'une montée de version de la solution logicielle *VITAM*.



4.1.1 Etats attendus

4.1.1.1 Pré-migration

Avant toute migration, il est attendu de la part des exploitants de vérifier :

- Que la solution logicielle *VITAM* fonctionne normalement
- Que l'ensemble des *timers* systemd sont stoppés
- Qu'aucun *workflow* n'est ni en cours, ni en statut **FATAL**

Voir aussi :

Se référer au chapitre « Suivi de l'état du système » du *DEX* pour plus d'informations.

Voir aussi :

Se référer au chapitre « Suivi des Workflows » du *DEX*, pour plus d'informations sur la façon de vérifier l'état des statuts des *workflows*.

4.1.1.2 Post-migration

A l'issue de toute migration, il est attendu de la part des exploitants de vérifier :

- Que la solution logicielle *VITAM* fonctionne normalement
- Que l'ensemble des *timers* systemd sont bien redémarrés (les redémarrer, le cas échéant)
- Qu'aucun *workflow* n'est en statut **FATAL**

Se référer au chapitre « Suivi de l'état du système » du *DEX* pour plus d'informations.

4.1.2 Montées de version *bugfix*

Au sein d'une même *release*, la montée de version depuis une version *bugfix* vers une version *bugfix* supérieure est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux playbooks ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Les montées de version *bugfix* ne contiennent à priori pas d'opérations de migration ou de reprises de données particulières. Toutefois, des spécificités propres aux différentes versions *bugfixes* peuvent s'appliquer ; elles sont explicitées dans le chapitre *Montées de version bugfix* (page 9).

Prudence : Parmi les versions *bugfixes* publiées au sein d'une même *release*, seuls les chemins de montées de version d'une version *bugfix* à la version *bugfix* suivante sont qualifiés par *VITAM*.

4.1.3 Montées de version mineure

La montée de version depuis une version mineure (de type *release*) vers une version mineure supérieure est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux playbooks ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Ce document décrit les chemins de montées de version depuis une version mineure, vers la version mineure maintenue supérieure.

Les montées de version mineure doivent être réalisées en s'appuyant sur les dernières versions *bugfixes* publiées.

Les opérations de migration ou de reprises de données propres aux différentes versions *releases* sont explicitées dans le chapitre *Montées de version mineure* (page 12).

Prudence : Parmi les versions mineures publiées au sein d'une même version majeure, seuls les chemins de montées de version depuis une version mineure maintenue, vers la version mineure maintenue suivante sont qualifiés par *VITAM*.

4.1.4 Montées de version majeure

La montée de version depuis une version majeure vers une version majeure supérieure s'appuie sur les chemins de montées de version mineure décrits dans le chapitre *Montées de version mineure* (page 12).

4.2 Montées de version *bugfix*

4.2.1 Notes et procédures spécifiques R6

Note : Cette version n'est à ce jour plus officiellement supportée.

4.2.2 Notes et procédures spécifiques R7

Note : Cette version n'est à ce jour plus officiellement supportée.

4.2.3 Notes et procédures spécifiques R8

Note : Cette version n'est à ce jour plus officiellement supportée.

4.2.4 Notes et procédures spécifiques R9

Sans objet.

4.2.5 Notes et procédures spécifiques R10

Note : Cette version n'est à ce jour plus officiellement supportée.

4.2.6 Notes et procédures spécifiques R11

Note : Cette version n'est à ce jour plus officiellement supportée.

4.2.7 Notes et procédures spécifiques R12

Sans objet.

4.2.8 Notes et procédures spécifiques R13

4.2.8.1 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+)

Si vous disposez d'une instance R13.16 ou inférieur (3.0.16-), vers une version R13.17+ (3.0.17+), et que vous utilisez des offres Swift V2/V3 (providers openstack-swift-v2 et/ou openstack-swift-v3), il est nécessaire de procéder à une migration des données :

```
$ ansible-playbook ansible-vitam-exploitation/migration_swift_v2_and_v3.yml -i␣
↪environments/hosts.{env} --ask-vault-pass

# Confirm playbook execution
# Enter swift offer id (ex offer-swift-1)
# Select migration mode
# > Enter '0' for analysis only mode : This mode will only log anomalies (in offer␣
↪technical logs), no update will be proceeded
# > Enter '1' to fix inconsistencies : This mode will update swift objects to fix␣
↪inconsistencies. However, this does not prune objects (delete partially written or␣
↪eliminated objects segments to free space).
# > Enter '2' to fix inconsistencies and purge all deleted objects segments to free␣
↪storage space.
# Reconfirm playbook execution
```

Il est recommandé de lancé la procédure en mode 0 (analyse seule) et de vérifier les erreurs de cohérence dans les logs Seules les offres Swift V2/V3 avec des objets volumineux (>= 4Go) nécessitent migration. Un exemple d'incohérences journalisées dans les logs (/vitam/log/offers) est donnée ici :

```
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪old segment names [aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2,␣
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1]. Run migration script with fix␣
↪inconsistencies mode to prune container.
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪missing metadata. Run migration script with fix inconsistencies mode enabled to set␣
↪object metadata.
```

Si la détection des anomalies est terminée en succès, et que des anomalies sont trouvées, il est recommandé de lancer le mode 1 (correction des anomalies). Les migrations de données sont également journalisées dans les logs (/vitam/log/offers) :

```
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000002
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000001
Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq migrated successfully.␣
↪Digest:␣
↪8959ea1290aa064a3c64d332f31e049bd4f9d4e95bebe0b46d38613bb079761d52c865dce64c88fd7e02313d340f9a2f8c0
```

Si des problèmes de cohérence de type « Orphan large object segments » persistent

```
INCONSISTENCY FOUND : Orphan large object segments [...] without parent object␣
↪manifest: env_2_object/aeaaaaaaaagbcaacaamboal2tk7dzmiaaaaq. Eliminated object?␣
↪Incomplete write? Run migration script with delete mode to prune container.
```

Dans ce cas, il est recommandé de vérifier préalablement que les objets concernés n'existent pas sur les autres offres (mêmes container & objectName). Si les objets n'existent pas dans les autres offres, il s'agit alors de reliquats d'objets non complètement éliminés. Le lancement du mode 2 (correction des anomalies + purge des objets) est à réaliser. Dans le cas contraire (cas où l'objet existe dans les autres offres), il faudra envisager la « Procédure de resynchronisation ciblée d'une offre » décrite dans la Documentation d'EXploitation (DEX) de Vitam pour synchroniser l'offre Swift pour les éléments concernés.

Note : Cette procédure doit être lancée une seule fois, et pour chaque offre Swift V2/V3, APRES upgrade Vitam.

4.2.9 Notes et procédures spécifiques R16

4.2.9.1 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+)

Si vous disposez d'une instance R16.3 ou inférieur (4.0.3-), vers une version R16.4+ (4.0.4+), et que vous utilisez des offres Swift V2/V3 (providers openstack-swift-v2 et/ou openstack-swift-v3), il est nécessaire de procéder à une migration des données :

```
$ ansible-playbook ansible-vitam-exploitation/migration_swift_v2_and_v3.yml -i␣
↪environments/hosts.{env} --ask-vault-pass

# Confirm playbook execution
# Enter swift offer id (ex offer-swift-1)
# Select migration mode
# > Enter '0' for analysis only mode : This mode will only log anomalies (in offer␣
↪technical logs), no update will be proceeded
# > Enter '1' to fix inconsistencies : This mode will update swift objects to fix␣
↪inconsistencies. However, this does not prune objects (delete partially written or␣
↪eliminated objects segments to free space).
# > Enter '2' to fix inconsistencies and purge all deleted objects segments to free␣
↪storage space.
# Reconfirm playbook execution
```

Il est recommandé de lancé la procédure en mode 0 (analyse seule) et de vérifier les erreurs de cohérence dans les logs Seules les offres Swift V2/V3 avec des objets volumineux (>= 4Go) nécessitent migration. Un exemple d'incohérences journalisées dans les logs (/vitam/log/offers) est donnée ici :

```
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪old segment names [aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2,␣
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1]. Run migration script with fix␣
↪inconsistencies mode to prune container.
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪missing metadata. Run migration script with fix inconsistencies mode enabled to set␣
↪object metadata.
```

Si la détection des anomalies est terminée en succès, et que des anomalies sont trouvées, il est recommandé de lancer le mode 1 (correction des anomalies). Les migrations de données sont également journalisées dans les logs (/vitam/log/offers) :

```
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000002
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000001
Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq migrated successfully.␣
↪Digest:␣
↪8959ea1290aa064a3c64d332f31e049bd4f9d4e95bebe0b46d38613bb079761d52c865dce64c88fd7e02313d340f9a2f8c0
```

Si des problèmes de cohérence de type « Orphan large object segments » persistent

```
INCONSISTENCY FOUND : Orphan large object segments [...] without parent object␣
↪manifest: env_2_object/aeaaaaaaaagbcaacaamboal2tk7dzmiaaaaq. Eliminated object?␣
↪Incomplete write? Run migration script with delete mode to prune container.
```

Dans ce cas, il est recommandé de vérifier préalablement que les objets concernés n'existent pas sur les autres offres (mêmes container & objectName). Si les objets n'existent pas dans les autres offres, il s'agit alors de reliquats d'objets non complètement éliminés. Le lancement du mode 2 (correction des anomalies + purge des objets) est à réaliser. Dans le cas contraire (cas où l'objet existe dans les autres offres), il faudra envisager la « Procédure de resynchronisation

ciblée d'une offre » décrite dans la Documentation d'EXploitation (DEX) de Vitam pour synchroniser l'offre Swift pour les éléments concernés.

Note : Cette procédure doit être lancée une seule fois, et pour chaque offre Swift V2/V3, APRES upgrade Vitam.

4.3 Montées de version mineure

4.3.1 Notes et procédures spécifiques R7

Note : Cette version n'est à ce jour plus officiellement supportée.

4.3.2 Notes et procédures spécifiques R8

Note : Cette version n'est à ce jour plus officiellement supportée.

4.3.3 Notes et procédures spécifiques R9

Prudence : Rappel : la montée de version vers la *release* R9 s'effectue depuis la *release* R7 (V1, *deprecated*) ou la *release* R8 (V1, *deprecated*) et doit être réalisée en s'appuyant sur les dernières versions *bugfixes* publiées.

4.3.3.1 Étapes préalables à la montée de version

4.3.3.1.1 Gestion du référentiel des formats

Prudence : Si un référentiel des formats personnalisé est utilisé avec la solution logicielle *VITAM*, il faut impérativement, lors d'une montée de version, modifier manuellement le fichier des formats livré par défaut avant toute réinstallation afin d'y réintégrer les modifications. A défaut, le référentiel des formats sera réinitialisé.

Il faut pour cela éditer le fichier situé à l'emplacement `environments/DROID_SignatureFile_<version>.xml` afin d'y réintégrer les éléments du référentiel des formats personnalisés.

4.3.3.1.2 Gestion de la rétro-compatibilité des données des offres

En préalable à l'installation, et uniquement dans le cas d'une montée de version (ne concerne pas le cas d'une installation R9 *from scratch*), il est nécessaire d'éditer le fichier d'inventaire ansible sur le modèle du fichier `deployment/environments/hosts.example` afin de décommenter la ligne ci-dessous :

```
# On offer, value is the prefix for all containers' names. If upgrading from R8, you␣
↪MUST UNCOMMENT this parameter AS IS !!!
vitam_prefix_offer=""
```

Cela est dû à la mise en place, à partir de la version R9, d'un prefixe au niveau des noms de conteneurs de *tenants* logiques *VITAM* sur les offres de stockage. Dans le cas d'une montée de version, cette étape préalable à l'installation permettra de garantir la rétro-compatibilité des données entre les versions précédentes et la version R9 (et supérieures).

4.3.3.1.3 Arrêt des *timers* systemd

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les *timers* systemd ont été arrêtés, afin de ne pas perturber la migration.

Il est également recommandé de ne lancer la procédure de migration qu'après s'être assuré que plus aucun *workflow* n'est ni en cours, ni en statut **FATAL**.

4.3.3.1.4 Arrêt des composants *externals*

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les composants *externals* ont été arrêtés, afin de ne pas perturber la migration.

4.3.3.1.5 Reprise des données de certificats

La version R9 apporte une nouvelle fonctionnalité permettant la révocation des certificats *SIA* et *Personae*, afin d'empecher des accès non autorisés aux *API VITAM* (vérification dans la couche https des *CRL*). Cette fonctionnalité impose d'effectuer une reprise des données des certificats (base MongoDB identity, collections `Certificate` et `PersonalCertificate`).

Les commandes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook ansible-vitam-exploitation/migration_r7_certificates.yml
--vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook ansible-vitam-exploitation/migration_r7_certificates.yml
--ask-vault-pass
```

4.3.3.1.6 Montée de version MongoDB 3.4 vers 4.0

La montée de version R7 vers R9 comprend une montée de version de la bases de données MongoDB de la version 3.4 à la version 4.0.

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

- Arrêt de *VITAM* (*playbook* `ansible-vitam-exploitation/stop_vitam.yml`)

Avertissement : A partir de là, la solution logicielle *VITAM* est arrêtée ; elle ne sera redémarrée qu'au déploiement de la nouvelle version.

- Démarrage des différents cluster mongodb (playbook `ansible-vitam-exploitation/start_mongodb.yml`)
- Upgrade de mongodb en version 3.6 (*playbook* `ansible-vitam-exploitation/migration_mongodb_36.yml`)
- Upgrade de mongodb en version 4.0 (*playbook* `ansible-vitam-exploitation/migration_mongodb_40.yml`)

4.3.3.2 Montée de version

La montée de version vers la *release* R9 est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux *playbooks* ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Note : Rappel : avant de procéder à la montée de version, on veillera tout particulièrement à la bonne mise en place des *repositories VITAM* associés à la nouvelle version. Se reporter à la section du *DIN* sur la mise en place des *repositories VITAM*.

Prudence : À l'issue de l'exécution du déploiement de Vitam, les composants *externals* ainsi que les *timers* systemd seront redémarrés. Il est donc recommandé de jouer les étapes de migration suivantes dans la foulée.

4.3.3.3 Etapes de migration

Dans le cadre d'une montée de version R7 vers R9, il est nécessaire d'appliquer un *playbook* de migration de données à l'issue de réinstallation de la solution logicielle *VITAM*.

Prudence : Dans le cas particulier d'une installation multi-sites, il sera nécessaire de d'abord lancer la migration des données sur le site secondaire afin de purger les registres des fonds, puis de lancer la migration sur le site primaire, et enfin de lancer la reconstruction des registres des fonds sur le site secondaire.

4.3.3.3.1 Procédure de migration des données

Lancer les commandes ci-après dans l'ordre suivant :

1. D'abord sur le site secondaire pour purger les registres des fonds
2. Ensuite sur le site primaire pour la migration des registres des fonds.

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r7_r8.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r7_r8.yml --ask-vault-pass
```

Avertissement : Selon la volumétrie des données précédement chargées, le *playbook* peut durer jusqu'à plusieurs heures.

En complément, en lien avec la correction du bug #5911, une migration du modèle de données des contrats d'entrées est également requise. Cette migration s'effectue à l'aide de la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r7_r9_ingestcontracts.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r7_r9_ingestcontracts.yml --ask-vault-pass
```

Note : Durant la migration, il est fortement recommandé de ne pas procéder à des versements de données. En effet, le *playbook* se charge d'arrêter les composants « ingest-external » et « access-external » avant de réaliser les opérations de migration de données, puis de redémarrer les composants « ingest-external » et « access-external ».

Les opérations de migration réalisées portent, entre autres, sur les éléments suivants :

- **Les registres des fonds (Accession Registers)**
  - **Diff AccessionRegisterDetail :**
    - Suppression du champs `Identifier`, remplacé par `Opc` (Opération courante)
    - Suppression du champs `OperationGroup`, remplacé par `Opi` (Opération d'ingest)
    - Suppression du champs `Symbolic`
    - Suppression des champs `attached`, `detached`, `symbolicRemained` des sous objets (`TotalUnits`, `TotalObjectGroups`, `TotalObjects`, `ObjectSize`)
    - Ajout d'un sous objet `Events`
  - **Diff AccessionRegisterSummary :**
    - Suppression des champs `attached`, `detached`, `symbolicRemained` des sous objets (`TotalUnits`, `TotalObjectGroups`, `TotalObjects`, `ObjectSize`)
- **Le journal des opérations**
  - Seules seront disponibles les données du registre des fonds selon le nouveau modèle dans le `evDetData` du journal de l'opération d'"*ingest*.
- **Les contrats d'entrées**
  - Ajout d'un mécanisme de contrôle pour la vérification du format de fichier DataObject (ajout des champs FormatUnidentifiedAuthorized, EveryFormatType et FormatType)

Note : Se reporter à la documentation du nouveau modèle de données de la release R9.

4.3.3.3.2 Procédure de réindexation des registres de fonds

Sous `deployment`, exécuter la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
reindex_es_data.yml --vault-password-file vault_pass.txt --tags
accessionregisterdetail
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
reindex_es_data.yml --ask-vault-pass --tags accessionregisterdetail
```

Les changement apportés touchent le mapping Elasticsearch de la collection `AccessionRegisterDetail`.

Note : Ce *playbook* ne supprime pas les anciens indexes pour laisser à l'exploitant le soin de verifier que la procedure de migration s'est correctement déroulée. A l'issue, la suppression des index devenus inutiles devra être realisée manuellement.

4.3.3.3.3 Procédure de réindexation des ObjectGroup

Sous `deployment`, exécuter la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r7_r9.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r7_r9.yml --ask-vault-pass
```

Les changement apportés touchent le mapping Elasticsearch sur l'attribut `qualifier.version` de la collection `ObjectGroup` (passé en nested)

Note : Ce *playbook* ne supprime pas les anciens indexes pour laisser à l'exploitant le soin de verifier que la procedure de migration s'est correctement déroulée. A l'issue, la suppression des index devenus inutiles devra être realisée manuellement.

4.3.3.3.4 Après la migration

Exécuter la commande suivante afin de réactiver les timers systemd sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
start_vitam_timers.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
start_vitam_timers.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les timers systemd ont été redémarrés.

4.3.3.3.5 Une fois le site secondaire *up*

Sur le site secondaire, vérifier sur les machines hébergeant le composant `functional-administration` que le processus de reconstruction des registres des fonds a bien démarré.

La commande à exécuter (en tant que root) est la suivante :

```
systemctl status vitam-functional-administration-accession-register-reconstruction.
service
```

4.3.3.3.6 Vérification de la bonne migration des données

A l'issue de la migration, il est fortement conseillé de lancer un « Audit de cohérence » sur les différents tenants. Pour rappel du *DEX*, pour lancer un audit de cohérence, il faut lancer le *playbook* comme suit :

> ansible-playbook -i <inventaire> ansible-playbok-exploitation/audit_coherence.yml –ask-vault-pass -e « access_contract=<contrat multitenant> »

Ou, si un fichier vault-password-file existe

```
ansible-playbook -i <inventaire> ansible-playbok-exploitation/audit_coherence.yml --
↪vault-password-file vault_pass.txt -e "access_contract=<contrat multitenant>"
```

Note : L'audit est lancé sur tous les *tenants* ; cependant, il est nécessaire de donner le contrat d'accès adapté. Se rapprocher du métier pour cet *id* de contrat. Pour limiter la liste des *tenants*, il faut rajouter un *extra var* à la ligne de commande ansible. Exemple

```
-e vitam_tenant_ids=[0,1]
```

pour limiter aux *tenants* 0 et 1.

4.3.4 Notes et procédures spécifiques R10

Note : Cette version n'est à ce jour plus officiellement supportée.

4.3.5 Notes et procédures spécifiques R11

Note : Cette version n'est à ce jour plus officiellement supportée.

4.3.6 Notes et procédures spécifiques R12

Prudence : Rappel : la montée de version vers la *release* R12 s'effectue depuis la *release* R9 (LTS V2), la *release* R10 (V2, *deprecated*) ou la *release* R11 (V2, *deprecated*) et doit être réalisée en s'appuyant sur les dernières versions *bugfixes* publiées.

4.3.6.1 Étapes préalables à la montée de version

4.3.6.1.1 Gestion du référentiel ontologique

Prudence : En lien avec la *User Story* #5928 (livrée avec la *release* R11) et les changements de comportement de l'API d'import des ontologies associés, si un référentiel ontologique personnalisé est utilisé avec la solution logicielle *VITAM*, il faut impérativement, lors d'une montée de version vers la *release* R11 ou supérieure, modifier manuellement le fichier d'ontologie livré par défaut avant toute réinstallation afin d'y réintégrer les modifications. A défaut, l'ontologie sera remplacée en mode forcé (sans contrôle de cohérence).

Il faut pour cela éditer le fichier situé à l'emplacement `deployment/ansible-vitam/roles/init_contexts_and_security_profiles/files/VitamOntology.json` afin d'y réintégrer les éléments du référentiel ontologique personnalisés.

Note : Lors de la montée de version, une sauvegarde du référentiel ontologique courant est réalisée à l'emplacement `environments/backups/ontology_backup_<date>.json`

4.3.6.1.2 Gestion du référentiel des formats

Prudence : Si un référentiel des formats personnalisé est utilisé avec la solution logicielle *VITAM*, il faut impérativement, lors d'une montée de version, modifier manuellement le fichier des formats livré par défaut avant toute réinstallation afin d'y réintégrer les modifications. A défaut, le référentiel des formats sera réinitialisé.

Il faut pour cela éditer le fichier situé à l'emplacement `environments/DROID_SignatureFile_<version>.xml` afin d'y réintégrer les éléments du référentiel des formats personnalisés.

4.3.6.1.3 Mise à jour de l'inventaire

Les versions récentes de ansible préconisent de ne plus utiliser le caractère « - » dans les noms de groupes ansible.

Pour effectuer cette modification, un script de migration est mis à disposition pour mettre en conformité votre « ancien » inventaire dans une forme compatible avec les outils de déploiement de la *release* R12.

La commande à lancer est

```
cd deployment
./upgrade_inventory.sh ${fichier_d_inventaire}
```

4.3.6.1.4 Arrêt des *timers* systemd

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les *timers* systemd ont été arrêtés, afin de ne pas perturber la migration.

Il est également recommandé de ne lancer la procédure de migration qu'après s'être assuré que plus aucun *workflow* n'est ni en cours, ni en statut **FATAL**.

4.3.6.1.5 Arrêt des composants *externals*

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les composants *externals* ont été arrêtés, afin de ne pas perturber la migration.

4.3.6.1.6 Montée de version MongoDB 4.0 vers 4.2

La montée de version vers la *release* R12 comprend une montée de version de la bases de données MongoDB de la version 4.0 à la version 4.2.

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

- Arrêt de *VITAM* (*playbook* `ansible-vitam-exploitation/stop_vitam.yml`)

Avertissement : A partir de là, la solution logicielle *VITAM* est arrêtée ; elle ne sera redémarrée qu'au déploiement de la nouvelle version.

- Démarrage des différents cluster mongodb (playbook `ansible-vitam-exploitation/start_mongodb.yml`)
- Upgrade de mongodb en version 4.2 (*playbook* `ansible-vitam-exploitation/migration_mongodb_42.yml`)

4.3.6.2 Montée de version

La montée de version vers la *release* R12 est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux *playbooks* ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Note : Rappel : avant de procéder à la montée de version, on veillera tout particulièrement à la bonne mise en place des *repositories VITAM* associés à la nouvelle version. Se reporter à la section du *DIN* sur la mise en place des *repositories VITAM*.

Prudence : À l'issue de l'exécution du déploiement de Vitam, les composants *externals* ainsi que les *timers* systemd seront redémarrés. Il est donc recommandé de jouer les étapes de migration suivantes dans la foulée.

4.3.6.3 Etapes de migration

4.3.6.3.1 Migration des données de certificats

La *release* R11 apporte une modification quant à la déclaration des certificats. En effet, un bug empêchait l'intégration dans la solution *VITAM* de certificats possédant un serial number long.

La commande suivante est à exécuter depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
R10_upgrade_serial_number.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
R10_upgrade_serial_number.yml --ask-vault-pass
```

4.3.6.3.2 Migration des contrats d'entrée

La montée de version vers la *release* R11 requiert une migration de données (contrats d'entrée) suite à une modification sur les droits relatifs aux rattachements. Cette migration s'effectue à l'aide du playbook :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r9_r10_ingestcontracts.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r9_r10_ingestcontracts.yml --ask-vault-pass
```

Le template `upgrade_contracts.js` contient :

```
// Switch to masterdata database
db = db.getSiblingDB('masterdata');

// Update IngestContract
db.IngestContract.find({}).forEach(function(item) {
    if (item.CheckParentLink == "ACTIVE") {
        item.checkParentId = new Array(item.LinkParentId);
    }
    item.CheckParentLink = "AUTHORIZED";
    db.IngestContract.update({_id: item._id}, item);
});
```

4.3.6.3.3 Nettoyage des DIPs depuis les offres

Dans le cadre d'une montée de version vers la *release* R12, il est nécessaire d'appliquer un *playbook* de migration de données à l'issue de réinstallation de la solution logicielle *VITAM*.

La migration s'effectue, uniquement sur le site principal, à l'aide de la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r11_r12_dip_cleanup.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r11_r12_dip_cleanup.yml --ask-vault-pass
```

Avertissement : Selon la volumétrie des données précédement chargées, le *playbook* peut durer quelques minutes.

4.3.6.3.4 Réindexation ES Data

La montée de version vers la *release* R11 requiert une réindexation totale d'ElasticSearch. Cette réindexation s'effectue à l'aide du playbook :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
reindex_es_data.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

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

Note : Ce *playbook* ne supprime pas les anciens indexes pour laisser à l'exploitant le soin de vérifier que la procédure de migration s'est correctement déroulée. A l'issue, la suppression des index devenus inutiles devra être réalisée manuellement.

4.3.6.3.5 Vérification de la bonne migration des données

A l'issue de la migration, il est fortement conseillé de lancer un « Audit de cohérence » sur les différents tenants. Pour rappel du *DEX*, pour lancer un audit de cohérence, il faut lancer le *playbook* comme suit :

> ansible-playbook -i <inventaire> ansible-playbok-exploitation/audit_coherence.yml –ask-vault-pass -e « access_contract=<contrat multitenant> »

Ou, si un fichier vault-password-file existe

```
ansible-playbook -i <inventaire> ansible-playbok-exploitation/audit_coherence.yml --
↪vault-password-file vault_pass.txt -e "access_contract=<contrat multitenant>"
```

Note : L'audit est lancé sur tous les *tenants* ; cependant, il est nécessaire de donner le contrat d'accès adapté. Se rapprocher du métier pour cet *id* de contrat. Pour limiter la liste des *tenants*, il faut rajouter un *extra var* à la ligne de commande ansible. Exemple

```
-e vitam_tenant_ids=[0,1]
```

pour limiter aux *tenants* 0 et 1.

4.3.7 Notes et procédures spécifiques R13

Prudence : Rappel : la montée de version vers la *release* R13 s'effectue depuis la *release* R9 (LTS V2), la *release* R10 (V2, *deprecated*), la *release* R11 (V2, *deprecated*) ou la *release* R12 (V2) et doit être réalisée en s'appuyant sur les dernières versions *bugfixes* publiées.

Note : Cette *release* de la solution logicielle *VITAM* s'appuie sur la version 11 de Java ainsi que la version 7 d'Elasticsearch. Ces mises à jour sont prises en charge par le processus de montée de version.

Prudence : La montée de version vers la *release* R13 a été validée par *VITAM* dans le cadre d'une installation de type Centos 7. L'installation dite *from scracth* a quant à elle été validée pour les installations de type Centos 7 et Debian 10 (l'utilisation de la version 11 de Java impose en effet une installation de type Debian 10). La migration d'OS vers la version Debian 10 n'est pas supportée par *VITAM* dans le cadre de la montée de version vers la *release* R13.

4.3.7.1 Étapes préalables à la montée de version

4.3.7.1.1 Gestion du référentiel de l'ontologie

Prudence : en lien avec la User Story* #6213 (livré en version *3.4.x* de *VITAM*), les ontologies externes en cours d'exploitation par *VITAM* ne sont pas touchées, et seront mergées avec les ontologies internes situés : `deployment/environments/ontology/VitamOntology.json` .

La procédure de merge manuelle du référentiel de l'ontologie avant chaque montée de version n'est plus nécessaire.

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'echec 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 role vérifie que le composant d'administration fonctionnelle `vitam-functional-administration` est bien installé et démarré, ensuite la tâche `Check Import Ontologies` réalise un import à blanc en mode `Dry Run` du référentiel de l'ontologie et remonte des éventuelles erreurs d'import.

Danger : En cas d'echec de vérification, autrement dit, en cas de présence de conflits entre les deux vocabulaires (le vocabulaire interne à mettre à jour et le vocabulaire externe en cours d'exploitation), c'est à l'exploitant d'adapter son vocabulaire externe et de veiller à ce qu'il n'ya pas de 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'
```

Dès résolution des conflicts, l'exploitant lancera la mise à jour sans toucher l'ontologie interne.

Note : Lors de la montée de version, une sauvegarde du référentiel de l'ontologie courant est réalisée à l'emplacement `environments/backups/ontology_backup_<date>.json`

A ce jour l'import de l'ontologie externe seulement n'est pas possible, ce comportement changera dans le futur.

4.3.7.1.2 Gestion du référentiel des formats

Prudence : Si un référentiel des formats personnalisé est utilisé avec la solution logicielle *VITAM*, il faut impérativement, lors d'une montée de version, modifier manuellement le fichier des formats livré par défaut avant toute réinstallation afin d'y réintégrer les modifications. A défaut, le référentiel des formats sera réinitialisé.

Il faut pour cela éditer le fichier situé à l'emplacement `environments/DROID_SignatureFile_<version>.xml` afin d'y réintégrer les éléments du référentiel des formats personnalisés.

4.3.7.1.3 Mise à jour de l'inventaire

Les versions récentes de ansible préconisent de ne plus utiliser le caractère « - » dans les noms de groupes ansible.

Pour effectuer cette modification, un script de migration est mis à disposition pour mettre en conformité votre « ancien » inventaire dans une forme compatible avec les outils de déploiement de la *release* R12.

La commande à lancer est

```
cd deployment
./upgrade_inventory.sh ${fichier_d_inventaire}
```

4.3.7.1.4 Arrêt des *timers* systemd

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les *timers* systemd ont été arrêtés, afin de ne pas perturber la migration.

Il est également recommandé de ne lancer la procédure de migration qu'après s'être assuré que plus aucun *workflow* n'est ni en cours, ni en statut **FATAL**.

4.3.7.1.5 Arrêt des composants *externals*

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les composants *externals* ont été arrêtés, afin de ne pas perturber la migration.

4.3.7.1.6 Montée de version MongoDB 4.0 vers 4.2

La montée de version vers la *release* R12 comprend une montée de version de la bases de données MongoDB de la version 4.0 à la version 4.2.

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

- Arrêt de *VITAM* (*playbook* `ansible-vitam-exploitation/stop_vitam.yml`)

Avertissement : A partir de là, la solution logicielle *VITAM* est arrêtée ; elle ne sera redémarrée qu'au déploiement de la nouvelle version.

- Démarrage des différents cluster mongodb (playbook `ansible-vitam-exploitation/start_mongodb.yml`)
- Upgrade de mongodb en version 4.2 (*playbook* `ansible-vitam-exploitation/migration_mongodb_42.yml`)

4.3.7.1.7 Arrêt de l'ensemble des composants VITAM

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les composants *VITAM* ont été arrêtés, afin de ne pas perturber la migration.

4.3.7.2 Montée de version

La montée de version vers la *release* R13 est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux *playbooks* ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Note : Rappel : avant de procéder à la montée de version, on veillera tout particulièrement à la bonne mise en place des *repositories VITAM* associés à la nouvelle version. Se reporter à la section du *DIN* sur la mise en place des *repositories VITAM*.

Prudence : À l'issue de l'exécution du déploiement de Vitam, les composants *externals* ainsi que les *timers* systemd seront redémarrés. Il est donc recommandé de jouer les étapes de migration suivantes dans la foulée.

4.3.7.3 Etapes de migration

4.3.7.3.1 Migration des données de certificats

La *release* R11 apporte une modification quant à la déclaration des certificats. En effet, un bug empêchait l'intégration dans la solution *VITAM* de certificats possédant un serial number long.

La commande suivante est à exécuter depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
R10_upgrade_serial_number.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
R10_upgrade_serial_number.yml --ask-vault-pass
```

4.3.7.3.2 Migration des contrats d'entrée

La montée de version vers la *release* R11 requiert une migration de données (contrats d'entrée) suite à une modification sur les droits relatifs aux rattachements. Cette migration s'effectue à l'aide du playbook :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r9_r10_ingestcontracts.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r9_r10_ingestcontracts.yml --ask-vault-pass
```

Le template `upgrade_contracts.js` contient :

```
// Switch to masterdata database
db = db.getSiblingDB('masterdata');

// Update IngestContract
db.IngestContract.find({}).forEach(function(item) {
    if (item.CheckParentLink == "ACTIVE") {
        item.checkParentId = new Array(item.LinkParentId);
    }
    item.CheckParentLink = "AUTHORIZED";
    db.IngestContract.update({_id: item._id}, item);
});
```

4.3.7.3.3 Nettoyage des DIPs depuis les offres

Dans le cadre d'une montée de version vers la *release* R12, il est nécessaire d'appliquer un *playbook* de migration de données à l'issue de réinstallation de la solution logicielle *VITAM*.

La migration s'effectue, uniquement sur le site principal, à l'aide de la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r11_r12_dip_cleanup.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r11_r12_dip_cleanup.yml --ask-vault-pass
```

Avertissement : Selon la volumétrie des données précédement chargées, le *playbook* peut durer quelques minutes.

4.3.7.3.4 Réindexation ES Data

La montée de version vers la *release* R11 requiert une réindexation totale d'ElasticSearch. Cette réindexation s'effectue à l'aide du playbook :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
reindex_es_data.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

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

Note : Ce *playbook* ne supprime pas les anciens indexes pour laisser à l'exploitant le soin de vérifier que la procédure de migration s'est correctement déroulée. A l'issue, la suppression des index devenus inutiles devra être réalisée manuellement.

4.3.7.3.5 Mise à jour des métadonnées de reconstruction (cas d'un site secondaire)

Dans le cadre d'une montée de version vers R13 sur un site secondaire, il est nécessaire d'appliquer un *playbook* de migration de données à l'issue de réinstallation de la solution logicielle *VITAM*.

Le *playbook* ajoute dans les données des collections *Offset* des bases *masterdata*, *logbook* et *metadata* du site secondaire la valeur `"strategy" : "default"`.

La migration s'effectue, uniquement sur le site secondaire, à l'aide de la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r12_r13_upgrade_offset_strategy.yml --vault-password-file
vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r12_r13_upgrade_offset_strategy.yml --ask-vault-pass
```

4.3.7.3.6 Vérification de la bonne migration des données

Il est recommandé de procéder à un audit de cohérence aléatoire suite à une procédure de montée de version VITAM ou de migration de données. Pour ce faire, se référer au dossier d'exploitation (DEX) de la solution VITAM, section `Audit de cohérence`.

4.3.8 Notes et procédures spécifiques R15

4.3.8.1 Étapes préalables à la montée de version

4.3.8.1.1 Déplacement des paramètres relatif à la gestion des tenants

Dans le cadre de la nouvelle fonctionnalité permettant de regrouper les tenants, les paramètres suivants ont étés déplacés du fich

- vitam_tenant_ids
- vitam_tenant_admin

Il faut supprimer les valeurs précédentes de votre fichier d'inventaire et les reporter dans le fichier tenants_vars.yml en respectant la syntaxe yml adéquate.

Voir aussi :

Se référer à la documentation d'installation pour plus d'informations concernant le fichier `environments / group_vars/all/tenants_vars.yml`

4.3.8.2 Vérification de la bonne migration des données

4.3.8.2.1 Audit coherence

Il est recommandé de procéder à un audit de cohérence aléatoire suite à une procédure de montée de version VITAM ou de migration de données. Pour ce faire, se référer au dossier d'exploitation (DEX) de la solution VITAM, section `Audit de cohérence`.

4.3.9 Notes et procédures spécifiques R16

4.3.9.1 Étapes préalables à la montée de version

4.3.9.1.1 Déplacement des paramètres relatif à la gestion des tenants

Dans le cadre de la fonctionnalité introduite en R15 permettant de regrouper les tenants, les paramètres suivants ont étés déplacés du fichier d'inventaire au fichier `group_vars/all/tenants_vars.yml`

- `vitam_tenant_ids`
- `vitam_tenant_admin`

Si la montée de version s'effectue à partir d'une R13, il faut supprimer les valeurs précédentes de votre fichier d'inventaire et les reporter dans le fichier `tenants_vars.yml` en respectant la syntaxe YML adéquate.

Voir aussi :

Se référer à la documentation d'installation pour plus d'informations concernant le fichier `environments/ group_vars/all/tenants_vars.yml`

4.3.9.1.2 Gestion des régles de gestion

Dans le cadre d'une correction de bug permettant la validation stricte des types de règles lors de l'import du référentiel des règles de gestion, il faut impérativement, lors d'une montée de version, vérifier tous les types de règles de gestion existants sur Mongo et ES et les modifier manuellement en cas d'incohérence. il faut que les types de règles de gestion respecte la casse (les majuscules et les minuscules) Exemple :

`APPRAISALRULE` devrait être `AppraisalRule`

4.3.9.2 Vérification de la bonne migration des données

4.3.9.2.1 Audit coherence

Il est recommandé de procéder à un audit de cohérence aléatoire suite à une procédure de montée de version VITAM ou de migration de données. Pour ce faire, se référer au dossier d'exploitation (DEX) de la solution VITAM, section `Audit de cohérence`.

4.3.10 Notes et procédures spécifiques V5

4.3.10.1 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+)

Si vous disposez d'une instance R16.3 ou inférieur (4.0.3-), vers une version V5.rc ou supérieur, et que vous utilisez des offres Swift V2/V3 (providers openstack-swift-v2 et/ou openstack-swift-v3), il est nécessaire de procéder à une migration des données :

```
$ ansible-playbook ansible-vitam-exploitation/migration_swift_v2_and_v3.yml -i␣
↪environments/hosts.{env} --ask-vault-pass

# Confirm playbook execution
# Enter swift offer id (ex offer-swift-1)
# Select migration mode
# > Enter '0' for analysis only mode : This mode will only log anomalies (in offer␣
↪technical logs), no update will be proceeded
# > Enter '1' to fix inconsistencies : This mode will update swift objects to fix␣
↪inconsistencies. However, this does not prune objects (delete partially written or␣
↪eliminated objects segments to free space).
# > Enter '2' to fix inconsistencies and purge all deleted objects segments to free␣
↪storage space.
# Reconfirm playbook execution
```

Il est recommandé de lancé la procédure en mode 0 (analyse seule) et de vérifier les erreurs de cohérence dans les logs Seules les offres Swift V2/V3 avec des objets volumineux (>= 4Go) nécessitent migration. Un exemple d'incohérences journalisées dans les logs (/vitam/log/offers) est donnée ici :

```
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪old segment names [aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2,␣
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1]. Run migration script with fix␣
↪inconsistencies mode to prune container.
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪missing metadata. Run migration script with fix inconsistencies mode enabled to set␣
↪object metadata.
```

Si la détection des anomalies est terminée en succès, et que des anomalies sont trouvées, il est recommandé de lancer le mode 1 (correction des anomalies). Les migrations de données sont également journalisées dans les logs (/vitam/log/offers) :

```
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000002
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000001
Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq migrated successfully.␣
↪Digest:␣
↪8959ea1290aa064a3c64d332f31e049bd4f9d4e95bebe0b46d38613bb079761d52c865dce64c88fd7e02313d340f9a2f8c0
```

Si des problèmes de cohérence de type « Orphan large object segments » persistent

```
INCONSISTENCY FOUND : Orphan large object segments [...] without parent object␣
↪manifest: env_2_object/aeaaaaaaaagbcaacaamboal2tk7dzmiaaaaq. Eliminated object?␣
↪Incomplete write? Run migration script with delete mode to prune container.
```

Dans ce cas, il est recommandé de vérifier préalablement que les objets concernés n'existent pas sur les autres offres (mêmes container & objectName). Si les objets n'existent pas dans les autres offres, il s'agit alors de reliquats d'objets non complètement éliminés. Le lancement du mode 2 (correction des anomalies + purge des objets) est à réaliser. Dans le cas contraire (cas où l'objet existe dans les autres offres), il faudra envisager la « Procédure de resynchronisation ciblée d'une offre » décrite dans la Documentation d'EXploitation (DEX) de Vitam pour synchroniser l'offre Swift pour les éléments concernés.

Note : Cette procédure doit être lancée une seule fois, et pour chaque offre Swift V2/V3, APRES upgrade Vitam.

4.3.10.2 Migrations des unités achivestiques

La migration des données est réalisée en exécutant la commande suivante (sur le site primaire uniquement, dans le cas d'une installation multi-sites) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_v5.yml --vault-password-file vault_pass.txt
```

ou, si vault_pass.txt n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_v5.yml --ask-vault-pass
```

L'indexation des champs dynamiques, créés au niveau des régles de gestion héritées, et précisément au niveau de la propriété `endDates` est rendue inactive. Il faudrait ainsi réindexer toutes les unités archivitiques.

Note : Durant la migration, il est fortement recommandé de ne pas procéder à des versements de données.

Table des figures

Liste des tableaux

Index

S

T

U

V

W