



VITAM - Documentation d'installation

Version 4.0.1

VITAM

avr. 01, 2021

Table des matières

1	Introduction	1
1.1	Objectif de ce document	1
2	Rappels	2
2.1	Information concernant les licences	2
2.2	Documents de référence	2
2.2.1	Documents internes	2
2.2.2	Référentiels externes	3
2.3	Glossaire	3
3	Prérequis à l'installation	6
3.1	Expertises requises	6
3.2	Pré-requis plate-forme	8
3.2.1	Base commune	8
3.2.2	PKI	9
3.2.3	Systèmes d'exploitation	9
3.2.3.1	Déploiement sur environnement CentOS	10
3.2.3.2	Déploiement sur environnement Debian	10
3.2.3.3	Présence d'un agent antivirus	10
3.2.4	Matériel	11
3.2.5	Librairie de cartouches pour offre froide	11
3.3	Questions préparatoires	11
3.4	Récupération de la version	12
3.4.1	Utilisation des dépôts <i>open-source</i>	12
3.4.1.1	<i>Repository</i> pour environnement CentOS	12
3.4.1.1.1	Cas de <i>griffins</i>	12
3.4.1.2	<i>Repository</i> pour environnement Debian	13
3.4.1.2.1	Cas de <i>griffins</i>	13
3.4.2	Utilisation du package global d'installation	13
4	Procédures d'installation / mise à jour	14
4.1	Vérifications préalables	14
4.2	Procédures	14
4.2.1	Cinématique de déploiement	14
4.2.2	Cas particulier d'une installation multi-sites	15
4.2.2.1	Procédure d'installation	15
4.2.2.1.1	vitam_site_name	15

4.2.2.1.2	primary_site	15
4.2.2.1.3	consul_remote_sites	16
4.2.2.1.4	vitam_offers	16
4.2.2.1.5	vitam_strategy	17
4.2.2.1.6	other_strategies	18
4.2.2.1.7	plateforme_secret	19
4.2.2.1.8	consul_encrypt	19
4.2.2.2	Procédure de réinstallation	19
4.2.2.3	Flux entre Storage et Offer	20
4.2.2.3.1	Avant la génération des keystores	20
4.2.2.3.2	Après la génération des keystores	21
4.2.3	Configuration du déploiement	21
4.2.3.1	Fichiers de déploiement	21
4.2.3.2	Informations <i>plate-forme</i>	21
4.2.3.2.1	Inventaire	21
4.2.3.2.2	Fichier vitam_security.yml	30
4.2.3.2.3	Fichier offers_opts.yml	31
4.2.3.2.4	Fichier cots_vars.yml	36
4.2.3.2.5	Fichier tenants_vars.yml	42
4.2.3.3	Déclaration des secrets	46
4.2.3.3.1	vitam	46
4.2.3.3.2	Cas des extras	50
4.2.3.3.3	Commande ansible-vault	51
4.2.3.3.3.1	Générer des fichiers <i>vaultés</i> depuis des fichier en clair	51
4.2.3.3.3.2	Ré-encoder un fichier <i>vaulté</i>	51
4.2.3.4	Le mapping Elasticsearch pour Unit et ObjectGroup	51
4.2.4	Gestion des certificats	57
4.2.4.1	Cas 1 : Configuration développement / tests	57
4.2.4.1.1	Procédure générale	57
4.2.4.1.2	Génération des CA par les scripts Vitam	58
4.2.4.1.3	Génération des certificats par les scripts Vitam	58
4.2.4.2	Cas 2 : Configuration production	58
4.2.4.2.1	Procédure générale	58
4.2.4.2.2	Génération des certificats	59
4.2.4.2.2.1	Certificats serveurs	59
4.2.4.2.2.2	Certificat clients	59
4.2.4.2.2.3	Certificats d'horodatage	59
4.2.4.2.3	Intégration de certificats existants	60
4.2.4.2.4	Intégration de certificats clients de VITAM	61
4.2.4.2.4.1	Intégration d'une application externe (cliente)	61
4.2.4.2.4.2	Intégration d'un certificat personnel (<i>personae</i>)	61
4.2.4.2.5	Cas des offres objet	61
4.2.4.2.6	Absence d'usage d'un <i>reverse</i>	61
4.2.4.3	Intégration de CA pour une offre <i>Swift</i> ou <i>s3</i>	62
4.2.4.4	Génération des magasins de certificats	62
4.2.5	Paramétrages supplémentaires	62
4.2.5.1	<i>Tuning</i> JVM	62
4.2.5.2	Installation des <i>griffins</i> (greffons de préservation)	62
4.2.5.3	Rétention liée aux logback	63
4.2.5.3.1	Cas des accesslog	63
4.2.5.4	Paramétrage de l'antivirus (ingest-external)	64
4.2.5.5	Paramétrage des certificats externes (*-externe)	64
4.2.5.6	Placer « hors Vitam » le composant ihm-demo	64
4.2.5.7	Paramétrer le secure_cookie pour ihm-demo	65

4.2.5.8	Paramétrage de la centralisation des logs VITAM	65
4.2.5.8.1	Gestion par VITAM	65
4.2.5.8.2	Redirection des logs sur un SIEM tiers	65
4.2.5.9	Passage des identifiants des référentiels en mode <i>esclave</i>	65
4.2.5.10	Paramétrage du batch de calcul pour l'indexation des règles héritées	66
4.2.5.11	Durées minimales permettant de contrôler les valeurs saisies	66
4.2.5.12	Fichiers complémentaires	67
4.2.5.13	Paramétrage de l'Offre Froide (librairies de cartouches)	85
4.2.5.14	Sécurisation SELinux	87
4.2.5.15	Installation de la stack prometheus	88
4.2.5.15.1	Playbooks ansible	89
4.2.5.16	Installation de grafana	89
4.2.5.16.1	Playbook ansible	90
4.2.5.16.2	Configuration	90
4.2.6	Procédure de première installation	90
4.2.6.1	Déploiement	90
4.2.6.1.1	Cas particulier : utilisation de ClamAv en environnement Debian	90
4.2.6.1.2	Fichier de mot de passe des vaults ansible	90
4.2.6.1.3	Mise en place des repositories VITAM (optionnel)	91
4.2.6.1.4	Génération des <i>hostvars</i>	92
4.2.6.1.4.1	Cas 1 : Machines avec une seule interface réseau	92
4.2.6.1.4.2	Cas 2 : Machines avec plusieurs interfaces réseau	92
4.2.6.1.4.3	Vérification de la génération des <i>hostvars</i>	92
4.2.6.1.5	Déploiement	92
4.2.7	Éléments <i>extras</i> de l'installation	93
4.2.7.1	Configuration des <i>extras</i>	93
4.2.7.2	Déploiement des <i>extras</i>	95
4.2.7.2.1	ihm-recette	95
4.2.7.2.2	<i>Extras</i> complet	95
5	Procédures de mise à jour de la configuration	96
5.1	Cas d'une modification du nombre de tenants	96
5.2	Cas d'une modification des paramètres JVM	96
5.3	Cas de la mise à jour des <i>griffins</i>	97
6	Post installation	98
6.1	Validation du déploiement	98
6.1.1	Sécurisation du fichier <i>vault_pass.txt</i>	98
6.1.2	Validation manuelle	98
6.1.3	Validation via Consul	98
6.1.4	Post-installation : administration fonctionnelle	99
6.2	Sauvegarde des éléments d'installation	99
6.3	Troubleshooting	99
6.3.1	Erreur au chargement des <i>index template</i> kibana	99
6.3.2	Erreur au chargement des tableaux de bord Kibana	100
6.4	Retour d'expérience / cas rencontrés	100
6.4.1	Crash rsyslog, code killed, signal : BUS	100
6.4.2	Mongo-express ne se connecte pas à la base de données associée	100
6.4.3	Elasticsearch possède des shard non alloués (état « UNASSIGNED »)	100
6.4.4	Elasticsearch possède des shards non initialisés (état « INITIALIZING »)	101
6.4.5	Elasticsearch est dans l'état « <i>read-only</i> »	101
6.4.6	MongoDB semble lent	102
6.4.7	Les shards de MongoDB semblent mal équilibrés	102
6.4.8	L'importation initiale (profil de sécurité, certificats) retourne une erreur	103

6.4.9	Problème d'ingest et/ou d'access	103
7	Montée de version	104
8	Annexes	105
8.1	Vue d'ensemble de la gestion des certificats	105
8.1.1	Liste des suites cryptographiques & protocoles supportés par VITAM	105
8.1.2	Vue d'ensemble de la gestion des certificats	106
8.1.3	Description de l'arborescence de la PKI	106
8.1.4	Description de l'arborescence du répertoire deployment/environments/certs	108
8.1.5	Description de l'arborescence du répertoire deployment/environments/keystores	109
8.1.6	Fonctionnement des scripts de la PKI	109
8.2	Spécificités des certificats	109
8.2.1	Cas des certificats serveur	110
8.2.1.1	Généralités	110
8.2.1.2	Noms DNS des serveurs https VITAM	110
8.2.2	Cas des certificats client	111
8.2.3	Cas des certificats d'horodatage	111
8.2.4	Cas des certificats des services de stockage objets	111
8.3	Cycle de vie des certificats	111
8.4	Ansible & SSH	113
8.4.1	Authentification du compte utilisateur utilisé pour la connexion SSH	113
8.4.1.1	Par clé SSH avec passphrase	113
8.4.1.2	Par login/mot de passe	113
8.4.1.3	Par clé SSH sans passphrase	113
8.4.2	Authentification des hôtes	113
8.4.3	Elévation de privilèges	113
8.4.3.1	Par sudo avec mot de passe	114
8.4.3.2	Par su	114
8.4.3.3	Par sudo sans mot de passe	114
8.4.3.4	Déjà Root	114
	Index	117

1.1 Objectif de ce document

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

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
<i>DAT</i>	http://www.programmevitam.fr/ressources/DocCourante/html/archi
<i>DIN</i>	http://www.programmevitam.fr/ressources/DocCourante/html/installation
<i>DEX</i>	http://www.programmevitam.fr/ressources/DocCourante/html/exploitation
<i>DMV</i>	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 appelé *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_publicques

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](https://www.nationalarchives.gov.uk/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/>

Prérequis à l'installation

3.1 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	Outils	Description outils	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 exécuter 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 : intermédiaire	1/4 : Mineur	Savoir exécuter 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 : intermédiaire	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 : intermédiaire	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
Appliquatif	Applicatifs Java	Composants logiciels Vitam	2/4 : intermédiaire	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'indisponibilité de VITAM (fermeture du service external, arrêt des timers)
3.1. Expertises requises					

- 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.

3.2 Pré-requis plate-forme

Les pré-requis suivants sont nécessaires :

3.2.1 Base commune

- Tous les serveurs hébergeant la solution logicielle *VITAM* doivent être synchronisés sur un serveur de temps (protocole *NTP*, pas de *stratum 10*)
- Disposer de la solution de déploiement basée sur ansible

Le déploiement est orchestré depuis un poste ou serveur d'administration ; les pré-requis suivants doivent y être présents :

- packages nécessaires :
 - **ansible** (version **2.9** minimale et conseillée ; se référer à la [documentation ansible](#)¹¹ pour la procédure d'installation)
 - **openssh-client** (client SSH utilisé par ansible)
 - **JRE OpenJDK 11** et **openssl** (du fait de la génération de certificats / *stores*, l'utilitaire *keytool* est nécessaire)
- un accès ssh vers un utilisateur d'administration avec élévation de privilèges vers les droits `root`, `vitam`, `vitamdb` (les comptes `vitam` et `vitamdb` sont créés durant le déploiement) sur les serveurs cibles.
- Le compte utilisé sur le serveur d'administration doit avoir confiance dans les serveurs sur lesquels la solution logicielle *VITAM* doit être installée (fichier `~/.ssh/known_hosts` correctement renseigné)

Note : Se référer à la [documentation d'usage](#)¹² pour les procédures de connexion aux machines-cibles depuis le serveur ansible.

Prudence : Les adresses *IP* des machines sur lesquelles la solution logicielle *VITAM* sera installée ne doivent pas changer d'adresse IP au cours du temps. En cas de changement d'adresse IP, la plateforme ne pourra plus fonctionner.

Prudence : Aucune version pré-installée de la JRE OpenJDK ne doit être présente sur les machines cibles où sera installé *VITAM*.

Prudence : La solution *VITAM* ne tolère qu'une très courte désynchronisation de temps entre les machines (par défaut, 10 secondes). La configuration NTP doit être finement monitorée. Idéalement une synchronisation doit être planifiée chaque 5/10 minutes.

http://docs.ansible.com/ansible/latest/intro_installation.html
http://docs.ansible.com/ansible/latest/intro_getting_started.html

Prudence : Dans le cadre de l'installation des packages « extra », il est nécessaire, pour les partitions hébergeant des conteneurs docker (mongo-express, head), qu'elles aient un accès internet (installation du paquet officiel docker, récupération des images).

Prudence : Dans le cadre de l'installation des packages « extra », il est nécessaire, pour les partitions hébergeant le composant ihm-recette, qu'elles aient un accès internet (installation du *repository* et installation du *package* git-lfs ; récupération des *TNR* depuis un dépôt git).

Avertissement : Dans le cas d'une installation du composant vitam-offer en filesystem-hash, il est fortement recommandé d'employer un système de fichiers xfs pour le stockage des données. Se référer au *DAT* pour connaître la structuration des *filesystems* dans la solution logicielle *VITAM*. En cas d'utilisation d'un autre type, s'assurer que le filesystem possède/gère bien l'option user_xattr.

Avertissement : Dans le cas d'une installation du composant vitam-offer en tape-library, il est fortement recommandé d'installer au préalable sur les machines cible associées les paquets pour les commandes mt, mt-x et dd. Ces composants doivent également apporter le groupe système tape. Se reporter également à *Librairie de cartouches pour offre froide* (page 11).

3.2.2 PKI

La solution logicielle *VITAM* nécessite des certificats pour son bon fonctionnement (cf. *DAT* pour la liste des secrets et *Vue d'ensemble de la gestion des certificats* (page 105) pour une vue d'ensemble de leur usage.) La gestion de ces certificats, par le biais d'une ou plusieurs *PKI*, est à charge de l'équipe d'exploitation. La mise à disposition des certificats et des chaînes de validation *CA*, placés dans les répertoires de déploiement adéquats, est un pré-requis à tout déploiement en production de la solution logicielle *VITAM*.

Voir aussi :

Veuillez vous référer à la section *Vue d'ensemble de la gestion des certificats* (page 105) pour la liste des certificats nécessaires au déploiement de la solution *VITAM*, ainsi que pour leurs répertoires de déploiement.

3.2.3 Systèmes d'exploitation

Seules deux distributions Linux suivantes sont supportées à ce jour :

- CentOS 7
- Debian 10 (buster)

SELinux doit être configuré en mode permissive ou disabled. Toutefois depuis la release R13, la solution logicielle *VITAM* prend désormais en charge l'activation de SELinux sur le périmètre du composant worker et des processus associés aux *griffins* (greffons de préservation).

Note : En cas de changement de mode SELinux, redémarrer les machines pour la bonne prise en compte de la modification avant de lancer le déploiement.

Prudence : En cas d'installation initiale, les utilisateurs et groupes systèmes (noms et *UID*) utilisés par VITAM (et listés dans le *DAT*) ne doivent pas être présents sur les serveurs cible. Ces comptes sont créés lors de l'installation de VITAM et gérés par VITAM.

3.2.3.1 Déploiement sur environnement CentOS

- Disposer d'une plate-forme Linux CentOS 7 installée selon la répartition des services souhaités. En particulier, ces serveurs doivent avoir :
 - une configuration de temps synchronisée (ex : en récupérant le temps à un serveur centralisé)
 - Des autorisations de flux conformément aux besoins décrits dans le *DAT*
 - une configuration des serveurs de noms correcte (cette configuration sera surchargée lors de l'installation)
 - un accès à un dépôt (ou son miroir) CentOS 7 (base et extras) et EPEL 7
- Disposer des binaires VITAM : paquets *RPM* de VITAM (vitam-product) ainsi que les paquets d'éditeurs tiers livrés avec VITAM (vitam-external)
- Disposer, si besoin, des binaires pour l'installation des *griffins*

3.2.3.2 Déploiement sur environnement Debian

- Disposer d'une plate-forme Linux Debian « buster » installée selon la répartition des services souhaitée. En particulier, ces serveurs doivent avoir :
 - une configuration de temps synchronisée (ex : en récupérant le temps à un serveur centralisé)
 - Des autorisations de flux conformément aux besoins décrits dans le *DAT*
 - une configuration des serveurs de noms correcte (cette configuration sera surchargée lors de l'installation)
 - un accès à un dépôt (ou son miroir) Debian (base et extras) et buster-backports
 - un accès internet, car le dépôt docker sera ajouté
- Disposer des binaires VITAM : paquets deb de VITAM (vitam-product) ainsi que les paquets d'éditeurs tiers livrés avec VITAM (vitam-external)
- Disposer, si besoin, des binaires pour l'installation des *griffins*

Avertissement : Pour l'installation des *packages* mongoDB, il est nécessaire de mettre à disposition le *package* `libcurl3` présent en *stretch* uniquement (le *package* `libcurl4` sera désinstallé).

Avertissement : Le *package* `curl` est installé depuis les dépôts *stretch*.

3.2.3.3 Présence d'un agent antivirus

Dans le cas de partitions sur lesquelles un agent antivirus est déjà configuré (typiquement, *golden image*), il est recommandé de positionner une exception sur l'arborescence `/vitam` et les sous-arborescences, hormis la partition hébergeant le composant `ingest-external` (emploi d'un agent antivirus en prérequis des *ingest* ; se reporter à *Rétention liée aux logback* (page 63)).

3.2.4 Matériel

Les prérequis matériel sont définis dans le *DAT* ; à l'heure actuelle, le minimum recommandé pour la solution Vitam est 2 CPUs. Il également est recommandé de prévoir (paramétrage par défaut à l'installation) 512Mo de RAM disponible par composant applicatif *VITAM* installé sur chaque machine (hors elasticsearch et mongo).

Concernant l'espace disque, à l'heure actuelle, aucun pré-requis n'a été défini ; cependant, sont à prévoir par la suite des espaces de stockage conséquents pour les composants suivants :

- offer
- solution de centralisation des logs (*cluster* elasticsearch de log)
- workspace
- worker (temporairement, lors du traitement de chaque fichier à traiter)
- *cluster* elasticsearch des données *VITAM*

L'arborescence associée sur les partitions associées est : `/vitam/data/<composant>`

3.2.5 Librairie de cartouches pour offre froide

Des prérequis sont à réunir pour utiliser l'offre froide de stockage « tape-library » définie dans le *DAT*.

- La librairie de cartouches doit être opérationnelle et chargée en cartouches.
- La librairie et les lecteurs doivent déjà être disponibles sur la machine devant supporter une instance de ce composant. La commande `ls SCSI -g` peut permettre de vérifier si des périphériques sont détectés.

3.3 Questions préparatoires

La solution logicielle *VITAM* permet de répondre à différents besoins.

Afin d'y répondre de la façon la plus adéquate et afin de configurer correctement le déploiement *VITAM*, il est nécessaire de se poser en amont les questions suivantes :

- **Questions techniques :**
 - Topologie de déploiement et dimensionnement de l'environnement ?
 - Espace de stockage (volumétrie métier cible, technologies d'offres de stockage, nombre d'offres, etc.) ?
 - Sécurisation des flux http (récupération des clés publiques des services versants, sécurisation des flux d'accès aux offres, etc.) ?
- **Questions liées au métier :**
 - Nombre de tenants souhaités (hormis les tenant 0 et 1 qui font respectivement office de tenant « blanc » et de tenant d'administration) ?
 - Niveau de classification (la plate-forme est-elle « Secret Défense » ?)
 - Modalités d'indexation des règles de gestion des unités archivistiques (autrement dit, sur quels tenant le recalcul des *inheritedRules* doit-il être fait complètement / partiellement) ?
 - Greffons de préservations (*griffins*) nécessaires ?
 - Fréquence de calcul de l'état des fonds symboliques souhaitée ?
 - Définition des habilitations (profil de sécurité, contextes applicatifs, ...) ?
 - Modalités de gestion des données de référence (maître/esclave) pour chaque tenant ?

Par la suite, les réponses apportées vous permettront de configurer le déploiement par la définition des paramètres ansible.

3.4 Récupération de la version

3.4.1 Utilisation des dépôts *open-source*

Les scripts de déploiement de la solution logicielle *VITAM* sont disponibles dans le dépôt github *VITAM*¹³, dans le répertoire `deployment`.

Les binaires de la solution logicielle *VITAM* sont disponibles sur des dépôts *VITAM* publics indiqués ci-dessous par type de *package* ; ces dépôts doivent être correctement configurés sur la plate-forme cible avant toute installation.

3.4.1.1 *Repository* pour environnement CentOS

Sur les partitions cibles, configurer le fichier `/etc/yum.repos.d/vitam-repositories.repo` (remplacer `<branche_vitam>` par le nom de la branche de support à installer) comme suit

```
[programmevitam-vitam-rpm-release-product]
name=programmevitam-vitam-rpm-release-product
baseurl=http://download.programmevitam.fr/vitam_repository/<vitam_version>/rpm/vitam-
↪product/
gpgcheck=0
repo_gpgcheck=0
enabled=1

[programmevitam-vitam-rpm-release-external]
name=programmevitam-vitam-rpm-release-external
baseurl=http://download.programmevitam.fr/vitam_repository/<vitam_version>/rpm/vitam-
↪external/
gpgcheck=0
repo_gpgcheck=0
enabled=1
```

Note : remplacer `<vitam_version>` par la version à déployer.

3.4.1.1.1 Cas de *griffins*

Un dépôt supplémentaire est à paramétrer pour pouvoir dérouler l'installation des *griffins*

```
[programmevitam-vitam-griffins]
name=programmevitam-vitam-griffins
baseurl=http://download.programmevitam.fr/vitam_griffins/<version_griffins>/rpm/
gpgcheck=0
repo_gpgcheck=0
enabled=1
```

Note : remplacer `<version_griffins>` par la version à déployer.

<https://github.com/ProgrammeVitam/vitam>

3.4.1.2 *Repository* pour environnement Debian

Sur les partitions cibles, configurer le fichier `/etc/apt/sources.list.d/vitam-repositories.list` comme suit

```
deb [trusted=yes] http://download.programmevitam.fr/vitam_repository/<vitam_version>/  
↳deb/vitam-product/ ./  
deb [trusted=yes] http://download.programmevitam.fr/vitam_repository/<vitam_version>/  
↳deb/vitam-external/ ./
```

Note : remplacer `<vitam_version>` par la version à déployer.

3.4.1.2.1 Cas de *griffins*

Un dépôt supplémentaire est à paramétrer pour pouvoir dérouler l'installation des *griffins*

```
deb [trusted=yes] http://download.programmevitam.fr/vitam_griffins/<version_griffins>/  
↳deb/ ./
```

Note : remplacer `<version_griffins>` par la version à déployer.

3.4.2 Utilisation du package global d'installation

Note : Le *package* global d'installation n'est pas présent dans les dépôts publics.

Le *package* global d'installation contient les livrables binaires (dépôts CentOS, Debian, Maven)

Sur la machine « *ansible* » dédiée au déploiement de la solution logicielle *VITAM*, décompresser le package (au format `tar.gz`).

Pour l'installation des *griffins*, il convient de récupérer, puis décompresser, le package associé (au format `zip`).

Sur le *repository* « *VITAM* », récupérer également depuis le fichier d'extension `tar.gz` les binaires d'installation (`rpm` pour CentOS ; `deb` pour Debian) et les faire prendre en compte par le *repository*.

Sur le *repository* « *griffins* », récupérer également depuis le fichier d'extension `zip` les binaires d'installation (`rpm` pour CentOS ; `deb` pour Debian) et les faire prendre en compte par le *repository*.

Procédures d'installation / mise à jour

4.1 Vérifications préalables

Tous les serveurs cibles doivent avoir accès aux dépôts de binaires contenant les paquets de la solution logicielle *VITAM* et des composants externes requis pour l'installation. Les autres éléments d'installation (playbook ansible, ...) doivent être disponibles sur la machine ansible orchestrant le déploiement de la solution.

4.2 Procédures

4.2.1 Cinématique de déploiement

La cinématique de déploiement d'un site *VITAM* est représentée dans le schéma suivant :

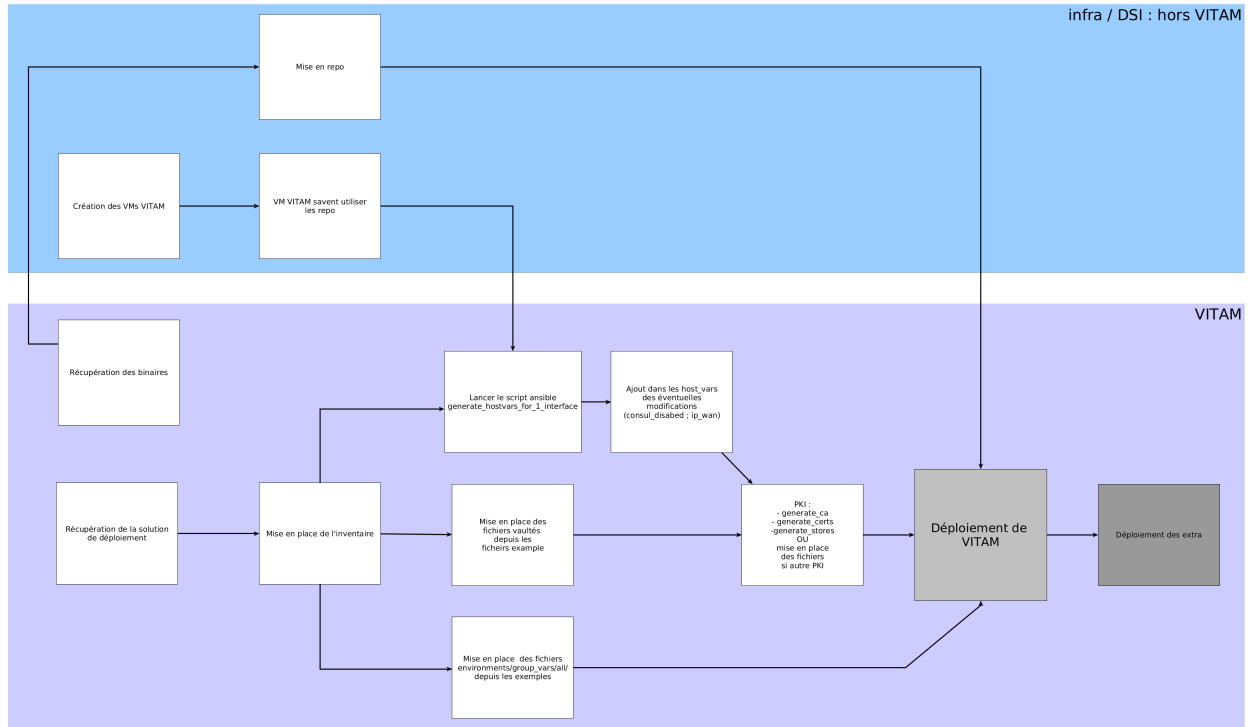


Fig. 1 – Cinématique de déploiement

4.2.2 Cas particulier d'une installation multi-sites

4.2.2.1 Procédure d'installation

Dans le cadre d'une installation multi-sites, il est nécessaire de déployer la solution logicielle **VITAM** sur le site secondaire dans un premier temps, puis déployer le site *production*.

Il faut paramétrer correctement un certain nombre de variables ansible pour chaque site :

4.2.2.1.1 vitam_site_name

Fichier : deployment/environments/hosts.<environnement>

Cette variable sert à définir le nom du site. Elle doit être différente sur chaque site.

4.2.2.1.2 primary_site

Fichier : deployment/environments/hosts.<environnement>

Cette variable sert à définir si le site est primaire ou non. Sur VITAM installé en mode multi site, un seul des sites doit avoir la valeur *primary_site* à true. Sur les sites secondaires (*primary_site* : false), certains composants ne seront pas démarrés et apparaîtront donc en orange sur l'*IHM* de consul. Certains timers systemd seront en revanche démarrés pour mettre en place la reconstruction au fil de l'eau, par exemple.

4.2.2.1.3 consul_remote_sites

Fichier : `deployment/environments/group_vars/all/cots_vars.yml`

Cette variable sert à référencer la liste des *Consul Server* des sites distants, à celui que l'on configure.

Exemple de configuration pour une installation avec 3 sites.

Site 1 :

```
consul_remote_sites:
  - dc2:
    wan: ["dc2-host-1", "dc2-host-2", "dc2-host-3"]
  - dc3:
    wan: ["dc3-host-1", "dc3-host-2", "dc3-host-3"]
```

Site 2 :

```
consul_remote_sites:
  - dc1:
    wan: ["dc1-host-1", "dc1-host-2", "dc1-host-3"]
  - dc3:
    wan: ["dc3-host-1", "dc3-host-2", "dc3-host-3"]
```

Site 3 :

```
consul_remote_sites:
  - dc1:
    wan: ["dc1-host-1", "dc1-host-2", "dc1-host-3"]
  - dc2:
    wan: ["dc2-host-1", "dc2-host-2", "dc2-host-3"]
```

Il faut également prévoir de déclarer, lors de l'installation de chaque site distant, la variable `ip_wan` pour les partitions hébergeant les serveurs Consul (groupe ansible `hosts_consul_server`) et les offres de stockage (groupe ansible `hosts_storage_offer_default`, considérées distantes par le site primaire). Ces ajouts sont à faire dans `environments/host_vars/<nom partition>`.

Exemple :

`ip_service : 172.17.0.10 ip_admin : 172.19.0.10 ip_wan : 10.2.64.3`

Ainsi, à l'usage, le composant `storage` va appeler les services `offer`. Si le service est « hors domaine » (déclaration explicite `<service>.<datacenterdistant>.service.<domaineconsul>`), un échange d'information entre « datacenters » Consul est réalisé et la valeur de `ip_wan` est fournie pour l'appel au service distant.

4.2.2.1.4 vitam_offers

Fichier : `deployment/environments/group_vars/all/offer_opts.yml`

Cette variable référence toutes les offres disponibles sur la totalité des sites VITAM. Sur les sites secondaires, il suffit de référencer les offres disponible localement.

Exemple :

```
vitam_offers:
  offer-fs-1:
    provider: filesystem-hash
  offer-fs-2:
    provider: filesystem-hash
```

(suite sur la page suivante)

(suite de la page précédente)

```
offer-fs-3:
  provider: filesystem-hash
```

4.2.2.1.5 vitam_strategy

Fichier : `deployment/environments/group_vars/all/offer_opts.yml`

Cette variable référence la stratégie de stockage de plateforme *default* sur le site courant.

Si l'offre se situe sur un site distant, il est nécessaire de préciser le nom du site, via la variable *vitam_site_name*, sur lequel elle se trouve comme dans l'exemple ci-dessous.

Il est fortement conseillé de prendre comme offre référente une des offres locale au site. Les sites secondaires doivent uniquement écrire sur leur(s) offre(s) locale(s).

Exemple pour le site 1 (site primaire) :

```
vitam_strategy:
- name: offer-fs-1
  referent: true
- name: offer-fs-2
  referent: false
  distant: true
  vitam_site_name: site2
- name: offer-fs-3
  referent: false
  distant: true
  vitam_site_name: site3
# Optional params for each offers in vitam_strategy. If not set, the default values
↳ are applied.
#   referent: false                # true / false (default), only one per site must be
↳ referent
#   status: ACTIVE                # ACTIVE (default) / INACTIVE
#   vitam_site_name: distant-dc2 # default is the value of vitam_site_name defined
↳ in your local inventory file, should be specified with the vitam_site_name defined
↳ for the distant offer
#   distant: false                # true / false (default). If set to true, it will
↳ not check if the provider for this offer is correctly set
#   id: idoffre                  # OPTIONAL, but IF ACTIVATED, MUST BE UNIQUE & SAME
↳ if on another site
#   asyncRead: false             # true / false (default). Should be set to true for
↳ tape offer only
```

Exemple pour le site 2 (site secondaire) :

```
vitam_strategy:
- name: offer-fs-2
  referent: true
```

Exemple pour le site 3 (site secondaire) :

```
vitam_strategy:
- name: offer-fs-3
  referent: true
```

4.2.2.1.6 other_strategies

Fichier : `deployment/environments/group_vars/all/offer_opts.yml`

Cette variable référence les stratégies de stockage additionnelles sur le site courant. **Elles ne sont déclarées et utilisées que dans le cas du multi-stratégies.** Si l'offre se situe sur un site distant, il est nécessaire de préciser le nom du site sur lequel elle se trouve comme dans l'exemple ci-dessous. Les sites secondaires doivent uniquement écrire sur leur(s) offre(s) locale(s).

Les offres correspondant à l'exemple `other_strategies` sont les suivantes :

```
vitam_offers:
  offer-fs-1:
    provider: filesystem-hash
  offer-fs-2:
    provider: filesystem-hash
  offer-fs-3:
    provider: filesystem-hash
  offer-s3-1:
    provider: amazon-s3-v1
  offer-s3-2:
    provider: amazon-s3-v1
  offer-s3-3:
    provider: amazon-s3-v1
```

Exemple pour le site 1 (site primaire) :

```
other_strategies:
  metadata:
    - name: offer-fs-1
      referent: true
    - name: offer-fs-2
      referent: false
      distant: true
      vitam_site_name: site2
    - name: offer-fs-3
      referent: false
      distant: true
      vitam_site_name: site3
    - name: offer-s3-1
      referent: false
    - name: offer-s3-2
      referent: false
      distant: true
      vitam_site_name: site2
    - name: offer-s3-3
      referent: false
      distant: true
      vitam_site_name: site3
  binary:
    - name: offer-s3-1
      referent: false
    - name: offer-s3-2
      referent: false
      distant: true
      vitam_site_name: site2
    - name: offer-s3-3
      referent: false
```

(suite sur la page suivante)

(suite de la page précédente)

```
distant: true
vitam_site_name: site3
```

Exemple pour le site 2 (site secondaire) :

```
other_strategies:
  metadata:
    - name: offer-fs-2
      referent: true
    - name: offer-s3-2
      referent: false
  binary:
    - name: offer-s3-2
      referent: false
```

Exemple pour le site 3 (site secondaire) :

```
other_strategies:
  metadata:
    - name: offer-fs-3
      referent: true
    - name: offer-s3-3
      referent: false
  binary:
    - name: offer-s3-3
      referent: false
```

4.2.2.1.7 plateforme_secret

Fichier : deployment/environments/group_vars/all/vault-vitam.yml

Cette variable stocke le *secret de plateforme* qui doit être commun à tous les composants de la solution logicielle *VITAM* de tous les sites. La valeur doit donc être identique pour chaque site.

4.2.2.1.8 consul_encrypt

Fichier : deployment/environments/group_vars/all/vault-vitam.yml

Cette variable stocke le *secret de plateforme* qui doit être commun à tous les *Consul* de tous les sites. La valeur doit donc être identique pour chaque site.

4.2.2.2 Procédure de réinstallation

En prérequis, il est nécessaire d'attendre que tous les *workflows* et reconstructions (sites secondaires) en cours soient terminés.

Ensuite :

- Arrêter vitam sur le site primaire.
- Arrêter les sites secondaires.
- Redéployer vitam sur les sites secondaires.
- Redéployer vitam sur le site primaire

4.2.2.3 Flux entre Storage et Offer

Dans le cas d'appel en https entre les composants Storage et Offer, il faut modifier `deployment/environments/group_vars/all/vitam_vars.yml` et indiquer `https_enabled: true` dans `storageofferdefault`.

Il convient également d'ajouter :

- **Sur le site primaire**
 - Dans le truststore de Storage : la CA ayant signé le certificat de l'Offer du site secondaire
- **Sur le site secondaire**
 - Dans le truststore de Offer : la CA ayant signé le certificat du Storage du site primaire
 - Dans le grantedstore de Offer : le certificat du storage du site primaire

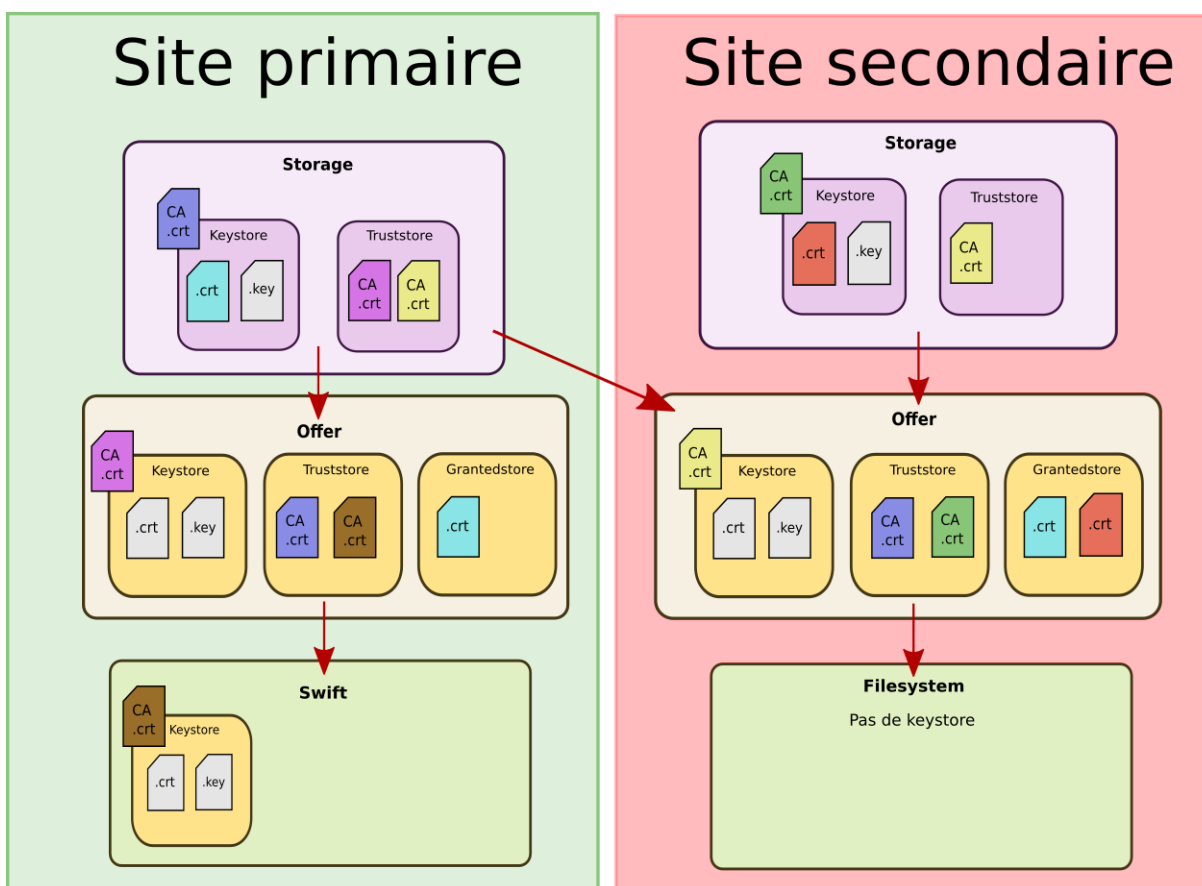


Fig. 2 – Vue détaillée des certificats entre le storage et l'offre en multi-site

Il est possible de procéder de 2 manières différentes :

4.2.2.3.1 Avant la génération des keystores

Avertissement : Pour toutes les copies de certificats indiquées ci-dessous, il est important de ne jamais les écraser, il faut donc renommer les fichiers si nécessaire.

Déposer les [CA](#) du client storage du site 1 `environments/certs/client-storage/ca/*` dans le client storage du site 2 `environments/certs/client-storage/ca/`.

Déposer le certificat du client storage du site 1 `environments/certs/client-storage/clients/storage/*.crt` dans le client storage du site 2 `environments/certs/client-storage/clients/storage/`.

Déposer les [CA](#) du serveur offer du site 2 `environments/certs/server/ca/*` dans le répertoire des [CA](#) serveur du site 1 `environments/certs/server/ca/*`

4.2.2.3.2 Après la génération des keystores

Via le script `deployment/generate_stores.sh`, il convient donc d'ajouter les [CA](#) et certificats indiqués sur le schéma ci-dessus.

Ajout d'un certificat : `keytool -import -keystore -file <certificat.crt> -alias <alias_certificat>`

Ajout d'une [CA](#) : `keytool -import -trustcacerts -keystore -file <ca.crt> -alias <alias_certificat>`

4.2.3 Configuration du déploiement

Voir aussi :

L'architecture de la solution logicielle, les éléments de dimensionnement ainsi que les principes de déploiement sont définis dans le [DAT](#).

4.2.3.1 Fichiers de déploiement

Les fichiers de déploiement sont disponibles dans la version [VITAM](#) livrée, dans le sous-répertoire `deployment/`. Concernant l'installation, ils se déclinent en 2 parties :

- les playbooks ansible de déploiement, présents dans le sous-répertoire `ansible-vitam/`, qui est indépendant de l'environnement à déployer ; ces fichiers ne sont normalement pas à modifier pour réaliser une installation.
- l'arborescence d'inventaire ; des fichiers d'exemples sont disponibles dans le sous-répertoire `environments/`. Cette arborescence est valable pour le déploiement d'un environnement, et doit être dupliquée lors de l'installation d'environnements ultérieurs. Les fichiers contenus dans cette arborescence doivent être adaptés avant le déploiement, comme expliqué dans les paragraphes suivants.

4.2.3.2 Informations *plate-forme*

4.2.3.2.1 Inventaire

Pour configurer le déploiement, il est nécessaire de créer, dans le répertoire `environments/`, un nouveau fichier d'inventaire (par la suite, ce fichier sera communément appelé `hosts.<environnement>`). Ce fichier devra se conformer à la structure présente dans le fichier `hosts.example` (et notamment respecter scrupuleusement l'arborescence des groupes *ansible*). Les commentaires dans ce fichier fournissent les explications permettant l'adaptation à l'environnement cible :

```

1  # Group definition ; DO NOT MODIFY
2  [hosts]
3
4  # Group definition ; DO NOT MODIFY
5  [hosts:children]
6  vitam
7  prometheus
8  grafana
9  reverse
10 hosts_dev_tools
11 ldap
12
13
14 ##### Tests environments specifics #####
15
16 # EXTRA : Front reverse-proxy (test environments ONLY) ; add machine name after
17 [reverse]
18 # optional : after machine, if this machine is different from VITAM machines, you can
19 ↪ specify another become user
20 # Example
21 # vitam-centos-01.vitam ansible_ssh_user=centos
22
23 ##### Extra VITAM applications #####
24 [prometheus:children]
25 hosts_prometheus
26 hosts_alertmanager
27
28 [hosts_prometheus]
29 # TODO: Put here server where this service will be deployed : prometheus server
30
31 [hosts_alertmanager]
32 # TODO: Put here servers where this service will be deployed : alertmanager
33
34 [grafana]
35 # TODO: Put here servers where this service will be deployed : grafana
36
37 [ldap] # Extra : OpenLDAP server
38 # LDAP server !!! NOT FOR PRODUCTION !!! Test only
39
40 [library]
41 # TODO: Put here servers where this service will be deployed : library
42
43 [hosts_dev_tools]
44 # TODO: Put here servers where this service will be deployed : mongo-express,
45 ↪ elasticsearch-head
46
47 [elasticsearch:children] # EXTRA : elasticsearch
48 hosts_elasticsearch_data
49 hosts_elasticsearch_log
50
51 ##### VITAM services #####
52
53 # Group definition ; DO NOT MODIFY
54 [vitam:children]
55 zone_external
56 zone_access
57 zone_applicative

```

(suite sur la page suivante)

(suite de la page précédente)

```

56 zone_storage
57 zone_data
58 zone_admin
59 library
60
61
62 ##### Zone externe
63
64
65 [zone_external:children]
66 hosts_ihm_demo
67 hosts_ihm_recette
68
69
70 [hosts_ihm_demo]
71 # TODO: Put here servers where this service will be deployed : ihm-demo. If you own_
72   ↳ another frontend, it is recommended to leave this group blank
73 # If you don't need consul for ihm-demo, you can set this var after each hostname :
74 # consul_disabled=true
75
76 [hosts_ihm_recette]
77 # TODO: Put here servers where this service will be deployed : ihm-recette (extra_
78   ↳ feature)
79
80 ##### Zone access
81
82 # Group definition ; DO NOT MODIFY
83 [zone_access:children]
84 hosts_ingest_external
85 hosts_access_external
86
87 [hosts_ingest_external]
88 # TODO: Put here servers where this service will be deployed : ingest-external
89
90 [hosts_access_external]
91 # TODO: Put here servers where this service will be deployed : access-external
92
93
94 ##### Zone applicative
95
96 # Group definition ; DO NOT MODIFY
97 [zone_applicative:children]
98 hosts_ingest_internal
99 hosts_processing
100 hosts_batch_report
101 hosts_worker
102 hosts_access_internal
103 hosts_metadata
104 hosts_functional_administration
105 hosts_logbook
106 hosts_workspace
107 hosts_storage_engine
108 hosts_security_internal
109
110 [hosts_security_internal]

```

(suite sur la page suivante)

(suite de la page précédente)

```

111 # TODO: Put here servers where this service will be deployed : security-internal
112
113
114 [hosts_logbook]
115 # TODO: Put here servers where this service will be deployed : logbook
116
117
118 [hosts_workspace]
119 # TODO: Put the server where this service will be deployed : workspace
120 # WARNING: put only one server for this service, not more !
121
122
123 [hosts_ingest_internal]
124 # TODO: Put here servers where this service will be deployed : ingest-internal
125
126
127 [hosts_access_internal]
128 # TODO: Put here servers where this service will be deployed : access-internal
129
130
131 [hosts_metadata]
132 # TODO: Put here servers where this service will be deployed : metadata
133
134
135 [hosts_functional_administration]
136 # TODO: Put here servers where this service will be deployed : functional-
137 ↪administration
138
139
140 [hosts_processing]
141 # TODO: Put the server where this service will be deployed : processing
142 # WARNING: put only one server for this service, not more !
143
144
145 [hosts_storage_engine]
146 # TODO: Put here servers where this service will be deployed : storage-engine
147
148
149 [hosts_batch_report]
150 # TODO: Put here servers where this service will be deployed : batch-report
151
152
153 [hosts_worker]
154 # TODO: Put here servers where this service will be deployed : worker
155 # Optional parameter after each host : vitam_worker_capacity=<integer> ; please refer_
156 ↪to your infrastructure for defining this number ; default is ansible_processor_
157 ↪vcpus value (cpu number in /proc/cpuinfo file)
158
159
160 ##### Zone storage
161
162 [zone_storage:children] # DO NOT MODIFY
163 hosts_storage_offer_default
164 hosts_mongodb_offer
165
166 [hosts_storage_offer_default]
167 # TODO: Put here servers where this service will be deployed : storage-offer-default
168 # LIMIT : only 1 offer per machine
169 # LIMIT and 1 machine per offer when filesystem or filesystem-hash provider

```

(suite sur la page suivante)

(suite de la page précédente)

```

165 # Possibility to declare multiple machines with same provider only when provider is_
    ↳ s3 or swift.
166 # Mandatory param for each offer is offer_conf and points to offer_opts.yml & vault-
    ↳ vitam.yml (with same tree)
167 # for swift
168 # hostname-offre-1.vitam offer_conf=offer-swift-1
169 # hostname-offre-2.vitam offer_conf=offer-swift-1
170 # for filesystem
171 # hostname-offre-2.vitam offer_conf=offer-fs-1
172 # for s3
173 # hostname-offre-3.vitam offer_conf=offer-s3-1
174 # hostname-offre-4.vitam offer_conf=offer-s3-1
175
176 [hosts_mongodb_offer:children]
177 hosts_mongos_offer
178 hosts_mongoc_offer
179 hosts_mongod_offer
180
181 [hosts_mongos_offer]
182 # WARNING : DO NOT COLLOCATE WITH [hosts_mongos_data]
183 # TODO: put here servers where this service will be deployed : mongos cluster for_
    ↳ storage offers
184 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the_
    ↳ offer_conf configuration)
185 # The recommended practice is to install the mongos instance on the same servers as_
    ↳ the mongoc instances
186 # Example (for a more complete one, see the one in the group hosts_mongos_data) :
187 # vitam-mongo-swift-offer-01    mongo_cluster_name=offer-swift-1
188 # vitam-mongo-swift-offer-02    mongo_cluster_name=offer-swift-1
189 # vitam-mongo-fs-offer-01       mongo_cluster_name=offer-fs-1
190 # vitam-mongo-fs-offer-02       mongo_cluster_name=offer-fs-1
191 # vitam-mongo-s3-offer-01       mongo_cluster_name=offer-s3-1
192 # vitam-mongo-s3-offer-02       mongo_cluster_name=offer-s3-1
193
194 [hosts_mongoc_offer]
195 # WARNING : DO NOT COLLOCATE WITH [hosts_mongoc_data]
196 # TODO: put here servers where this service will be deployed : mongoc cluster for_
    ↳ storage offers
197 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the_
    ↳ offer_conf configuration)
198 # Optional param : mandatory for 1 node of the shard, some init commands will be_
    ↳ executed on it
199 # Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add_
    ↳ this paramter on a mongo_rs_bootstrap node
200 # Recommended practice in production: use 3 instances
201 # Example :
202 # vitam-mongo-swift-offer-01    mongo_cluster_name=offer-swift-1
    ↳ mongo_rs_bootstrap=true
203 # vitam-mongo-swift-offer-02    mongo_cluster_name=offer-swift-1
204 # vitam-swift-offer             mongo_cluster_name=offer-swift-1
    ↳ mongo_arbiter=true
205 # vitam-mongo-fs-offer-01       mongo_cluster_name=offer-fs-1
    ↳ mongo_rs_bootstrap=true
206 # vitam-mongo-fs-offer-02       mongo_cluster_name=offer-fs-1
207 # vitam-fs-offer                mongo_cluster_name=offer-fs-1
    ↳ mongo_arbiter=true
208 # vitam-mongo-s3-offer-01       mongo_cluster_name=offer-s3-1
    ↳ mongo_rs_bootstrap=true

```

(suite sur la page suivante)

(suite de la page précédente)

```

209 # vitam-mongo-s3-offer-02      mongo_cluster_name=offer-s3-1
210 # vitam-s3-offer              mongo_cluster_name=offer-s3-1
    ↳mongo_arbiter=true
211
212 [hosts_mongod_offer]
213 # WARNING : DO NOT COLLOCATE WITH [hosts_mongod_data]
214 # TODO: put here servers where this service will be deployed : mongod cluster for
    ↳storage offers
215 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the
    ↳offer_conf configuration)
216 # Mandatory param : id of the current shard, increment by 1 from 0 to n
217 # Optional param : mandatory for 1 node of the shard, some init commands will be
    ↳executed on it
218 # Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add
    ↳this paramter on a mongo_rs_bootstrap node
219 # Optional param : mongod_memory=x ; this will force the wiredtiger cache size to x
    ↳(unit is GB) ; can be usefull when colocalization with elasticsearch
220 # Optional param : is_small=true ; this will force the priority for this server to be
    ↳lower when electing master ; hardware can be downgraded for this machine
221 # Recommended practice in production: use 3 instances per shard
222 # Example :
223 # vitam-mongo-swift-offer-01    mongo_cluster_name=offer-swift-1    mongo_shard_id=0
    ↳                                mongo_rs_bootstrap=true
224 # vitam-mongo-swift-offer-02    mongo_cluster_name=offer-swift-1    mongo_shard_id=0
225 # vitam-swift-offer            mongo_cluster_name=offer-swift-1    mongo_shard_id=0
    ↳                                mongo_arbiter=true
226 # vitam-mongo-fs-offer-01       mongo_cluster_name=offer-fs-1       mongo_shard_id=0
    ↳                                mongo_rs_bootstrap=true
227 # vitam-mongo-fs-offer-02       mongo_cluster_name=offer-fs-1       mongo_shard_id=0
228 # vitam-fs-offer               mongo_cluster_name=offer-fs-1       mongo_shard_id=0
    ↳                                mongo_arbiter=true
229 # vitam-mongo-s3-offer-01       mongo_cluster_name=offer-s3-1       mongo_shard_id=0
    ↳                                mongo_rs_bootstrap=true
230 # vitam-mongo-s3-offer-02       mongo_cluster_name=offer-s3-1       mongo_shard_id=0
    ↳                                is_small=true # PSSmin, this machine needs less hardware
231 # vitam-s3-offer               mongo_cluster_name=offer-s3-1       mongo_shard_id=0
    ↳                                mongo_arbiter=true
232
233 ##### Zone data
234
235 # Group definition ; DO NOT MODIFY
236 [zone_data:children]
237 hosts_elasticsearch_data
238 hosts_mongodb_data
239
240 [hosts_elasticsearch_data]
241 # TODO: Put here servers where this service will be deployed : elasticsearch-data
    ↳cluster
242 # 2 params available for huge environments (parameter to be declared after each
    ↳server) :
243 #   is_data=true/false
244 #   is_master=true/false
245 #   for site/room balancing : is_balancing=<whatever> so replica can be applied on
    ↳all sites/rooms ; default is vitam_site_name
246 #   other options are not handled yet
247 # defaults are set to true, if undefined. If defined, at least one server MUST be is
    ↳data=true

```

(suite sur la page suivante)

(suite de la page précédente)

```

248 # Examples :
249 # server1 is_master=true is_data=false
250 # server2 is_master=false is_data=true
251 # More explanation here : https://www.elastic.co/guide/en/elasticsearch/reference/5.6/
    ↳ modules-node.html
252
253
254 # Group definition ; DO NOT MODIFY
255 [hosts_mongodb_data:children]
256 hosts_mongos_data
257 hosts_mongoc_data
258 hosts_mongod_data
259
260 [hosts_mongos_data]
261 # WARNING : DO NOT COLLOCATE WITH [hosts_mongos_offer]
262 # TODO: Put here servers where this service will be deployed : mongos cluster
263 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
264 # The recommended practice is to install the mongos instance on the same servers as
    ↳ the mongoc instances
265 # Example :
266 # vitam-mdbs-01    mongo_cluster_name=mongo-data
267 # vitam-mdbs-02    mongo_cluster_name=mongo-data
268 # vitam-mdbs-03    mongo_cluster_name=mongo-data
269
270 [hosts_mongoc_data]
271 # WARNING : DO NOT COLLOCATE WITH [hosts_mongoc_offer]
272 # TODO: Put here servers where this service will be deployed : mongoc cluster
273 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
274 # Optional param : mandatory for 1 node of the shard, some init commands will be
    ↳ executed on it
275 # Recommended practice in production: use 3 instances
276 # Example :
277 # vitam-mdbc-01    mongo_cluster_name=mongo-data                mongo_rs_
    ↳ bootstrap=true
278 # vitam-mdbc-02    mongo_cluster_name=mongo-data
279 # vitam-mdbc-03    mongo_cluster_name=mongo-data
280
281 [hosts_mongod_data]
282 # WARNING : DO NOT COLLOCATE WITH [hosts_mongod_offer]
283 # TODO: Put here servers where this service will be deployed : mongod cluster
284 # Each replica_set should have an odd number of members (2n + 1)
285 # Reminder: For Vitam, one mongodb shard is using one replica_set
286 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
287 # Mandatory param : id of the current shard, increment by 1 from 0 to n
288 # Optional param : mandatory for 1 node of the shard, some init commands will be
    ↳ executed on it
289 # Optional param : mongod_memory=x ; this will force the wiredtiger cache size to x
    ↳ (unit is GB) ; can be usefull when colocalization with elasticsearch
290 # Recommended practice in production: use 3 instances per shard
291 # Example:
292 # vitam-mdbd-01    mongo_cluster_name=mongo-data    mongo_shard_id=0    mongo_rs_
    ↳ bootstrap=true
293 # vitam-mdbd-02    mongo_cluster_name=mongo-data    mongo_shard_id=0
294 # vitam-mdbd-03    mongo_cluster_name=mongo-data    mongo_shard_id=0
295 # vitam-mdbd-04    mongo_cluster_name=mongo-data    mongo_shard_id=1    mongo_rs_
    ↳ bootstrap=true
296 # vitam-mdbd-05    mongo_cluster_name=mongo-data    mongo_shard_id=1

```

(suite sur la page suivante)

(suite de la page précédente)

```

297 # vitam-mdbd-06 mongo_cluster_name=mongo-data mongo_shard_id=1
298
299 ##### Zone admin
300
301 # Group definition ; DO NOT MODIFY
302 [zone_admin:children]
303 hosts_cerebro
304 hosts_consul_server
305 hosts_kibana_data
306 log_servers
307 hosts_elasticsearch_log
308
309 [hosts_cerebro]
310 # TODO: Put here servers where this service will be deployed : vitam-elasticsearch-
    ↳ cerebro
311
312 [hosts_consul_server]
313 # TODO: Put here servers where this service will be deployed : consul
314
315 [hosts_kibana_data]
316 # TODO: Put here servers where this service will be deployed : kibana (for data_
    ↳ cluster)
317
318 [log_servers:children]
319 hosts_kibana_log
320 hosts_logstash
321
322
323 [hosts_kibana_log]
324 # TODO: Put here servers where this service will be deployed : kibana (for log_
    ↳ cluster)
325
326 [hosts_logstash]
327 # TODO: Put here servers where this service will be deployed : logstash
328 # IF you connect VITAM to external SIEM, DO NOT FILL THE SECTION
329
330
331 [hosts_elasticsearch_log]
332 # TODO: Put here servers where this service will be deployed : elasticsearch-log_
    ↳ cluster
333 # IF you connect VITAM to external SIEM, DO NOT FILL THE SECTION
334
335 ##### Global vars #####
336
337 [hosts:vars]
338
339 # =====
340 # VITAM
341 # =====
342
343 # Declare user for ansible on target machines
344 ansible_ssh_user=
345 # Can target user become as root ? ; true is required by VITAM (usage of a sudoer is_
    ↳ mandatory)
346 ansible_become=true
347 # How can ansible switch to root ?
348 # See https://docs.ansible.com/ansible/latest/user\_guide/become.html

```

(suite sur la page suivante)

(suite de la page précédente)

```

349 # Related to Consul ; apply in a table your DNS server(s)
350 # Example : dns_servers=["8.8.8.8","8.8.4.4"]
351 # If no dns recursors are available, leave this value empty.
352 dns_servers=
353
354 # Define local Consul datacenter name
355 # CAUTION !!! Only alphanumeric characters when using s3 as offer backend !!!
356 vitam_site_name=prod-dcl
357
358 # On offer, value is the prefix for all container's names. If upgrading from R8, you
359 ↪ MUST UNCOMMENT this parameter AS IS !!!
360 #vitam_prefix_offer=""
361
362 # check whether on primary site (true) or secondary (false)
363 primary_site=true
364
365 # =====
366 # EXTRA
367 # =====
368
369 ### vitam-itest repository ###
370 vitam_tests_branch=master
371 vitam_tests_gitrepo_protocol=
372 vitam_tests_gitrepo_baseurl=
373 vitam_tests_gitrepo_url=
374
375 # Used when VITAM is behind a reverse proxy (provides configuration for reverse proxy
376 ↪ && displayed in header page)
377 vitam_reverse_external_dns=
378 # For reverse proxy use
379 reverse_proxy_port=443
380 vitam_reverse_external_protocol=https
381 # http_proxy env var to use ; has to be declared even if empty
382 http_proxy_environnement=

```

Pour chaque type de *host*, indiquer le(s) serveur(s) défini(s), pour chaque fonction. Une colocalisation de composants est possible (Cf. le paragraphe idoine du *DAT*)

Note : Concernant le groupe *hosts_consul_server*, il est nécessaire de déclarer au minimum 3 machines.

Avertissement : Il n'est pas possible de colocaliser les clusters MongoDB *data* et *offer*.

Avertissement : Il n'est pas possible de colocaliser *kibana-data* et *kibana-log*.

Note : Pour les composants considérés par l'exploitant comme étant « hors *VITAM* » (typiquement, le composant *ihm-demo*), il est possible de désactiver la création du service Consul associé. Pour cela, après chaque hostname impliqué, il faut rajouter la directive suivante : `consul_disabled=true`.

Prudence : Concernant la valeur de `vitam_site_name`, seuls les caractères alphanumériques et le tiret (« - ») sont autorisés.

Note : Il est possible de multi-instancier le composant « storage-offer-default » dans le cas d'un *provider* de type objet (s3, swift). Il faut ajouter `offer_conf=<le nom>`.

4.2.3.2.2 Fichier `vitam_security.yml`

La configuration des droits d'accès à VITAM est réalisée dans le fichier `reper-toire_inventory/group_vars/all/vitam_security.yml`, comme suit :

```
1 ---
2
3 hide_passwords_during_deploy: true
4
5 ### Admin context name and tenants ###
6 admin_context_name: "admin-context"
7 admin_context_tenants: "{{ vitam_tenant_ids }}"
8 # Indicate context certificates relative paths under {{ inventory_dir }}/certs/client-
9 ↪external/clients
10 # vitam-admin-int is mandatory for internal use (PRONOM upload)
11 admin_context_certs: [ "ihm-demo/ihm-demo.crt", "ihm-recette/ihm-recette.crt",
12 ↪"reverse/reverse.crt", "vitam-admin-int/vitam-admin-int.crt" ]
13 # Indicate here all the personal certificates relative paths under {{ inventory_dir }}
14 ↪/certs/client-vitam-users/clients
15 admin_personal_certs: [ "userOK.crt" ]
16
17 # Admin security profile name
18 admin_security_profile: "admin-security-profile"
19
20 admin_basic_auth_user: "adminUser"
21
22 # SELinux state, can be: enforcing, permissive, disabled
23 selinux_state: "disabled"
24 # SELinux Policy, can be: targeted, minimum, mls
25 selinux_policy: "targeted"
26 # If needed, reboot the VM to enable SELinux
27 selinux_reboot: True
28 # Relabel the entire filesystem ?
29 selinux_relabel: False
```

Note : Pour la directive `admin_context_certs` concernant l'intégration de certificats *SIA* au déploiement, se reporter à la section *Intégration d'une application externe (cliente)* (page 61).

Note : Pour la directive `admin_personal_certs` concernant l'intégration de certificats personnels (*personae*) au déploiement, se reporter à la section *Intégration d'un certificat personnel (personae)* (page 61).

4.2.3.2.3 Fichier offers_opts.yml

Indication : Fichier à créer depuis offers_opts.yml.example et à paramétrer selon le besoin.

La déclaration de configuration des offres de stockage associées se fait dans le fichier `reper-
toire_inventory|'group_vars/all/offers_opts.yml'` :

```

1  # This is the default vitam strategy ('default'). It is mandatory and must_
   ↳define a referent offer.
2  # This list of offers is ordered. It has to be completed if more offers are_
   ↳necessary
3  # Strategy order (1st has to be the preferred one)
4  vitam_strategy:
5    - name: offer-fs-1
6      referent: true
7
8  # Optional params for each offers in vitam_strategy. If not set, the default_
   ↳values are applied.
9  #   referent: false                # true / false (default), only one per_
   ↳site must be referent
10 #   status: ACTIVE                 # ACTIVE (default) / INACTIVE
11 #   vitam_site_name: distant-dc2 # default is the value of vitam_site_name_
   ↳defined in your local inventory file, should be specified with the vitam_
   ↳site_name defined for the distant offer
12 #   distant: false                 # true / false (default). If set to true,_
   ↳it will not check if the provider for this offer is correctly set
13 #   id: idoffre                    # OPTIONAL, but IF ACTIVATED, MUST BE_
   ↳UNIQUE & SAME if on another site
14 #   asyncRead: false               # true / false (default). Should be set to_
   ↳true for tape offer only
15
16 # Example for tape offer:
17 # Tape offer mustn't be referent (referent: false) and should be configured_
   ↳as asynchrone read (asyncRead: true)
18 #   - name: offer-tape-1
19 #     referent: false
20 #     asyncRead: true
21
22 # Example distant offer:
23 #   - name: distant
24 #     referent: false
25 #     vitam_site_name: distant-dc2
26 #     distant: true # Only add this parameter when distant offer (not on same_
   ↳platform)
27
28 # WARNING : multi-strategy is a BETA functionality
29 # More strategies can be added but are optional
30 # Strategy name must only use [a-z][a-z0-9-]* pattern
31 # Any strategy must contain at least one offer
32 # This list of offers is ordered. It can and has to be completed if more_
   ↳offers are necessary
33 # Every strategy can define at most one referent offer.
34 # other_strategies:
35 #   metadata:
36 #     - name: offer-fs-1

```

(suite sur la page suivante)

(suite de la page précédente)

```

37 #     referent: true
38 #     - name: offer-fs-2
39 #     referent: false
40 # binary:
41 #     - name: offer-fs-2
42 #     referent: false
43 #     - name: offer-s3-1
44 #     referent: false
45
46 # DON'T forget to add associated passwords in vault-vitam.yml with same tree
47 # ATTENTION !!! Each offer has to have a distinct name, except for clusters
48 # WARNING : for offer names, please only use [a-z][a-z0-9-]* pattern
49 vitam_offers:
50   offer-fs-1:
51     # param can be filesystem-hash (recomended) or filesystem (not
52     # provider: filesystem-hash
53     # Offer log compaction
54     offer_log_compaction:
55       ## Expiration, here offer logs 21 days old will be compacted
56       expiration_value: 21
57       ## Choose one of "MILLENNIA", "HALF_DAYS", "MILLIS", "FOREVER", "MICROS
58       expiration_unit: "DAYS"
59       ## Compaction bulk size here 10 000 offers logs (at most) will be
60       compaction_size: 10000
61       # Batch processing thread pool size
62       maxBatchThreadPoolSize: 32
63       # Batch metadata computation timeout in seconds
64       batchMetadataComputationTimeout: 600
65 #####
66   offer-swift-1:
67     # provider : openstack-swift for v1 or openstack-swift-v3 for v3
68     provider: openstack-swift-v3
69     # swiftKeystoneAuthUrl : URL de connexion à keystone
70     swiftKeystoneAuthUrl: https://openstack-hostname:port/auth/1.0
71     # swiftDomain : domaine OpenStack dans lequel l'utilisateur est
72     swiftDomain: domaine
73     # swiftUser : identifiant de l'utilisateur
74     swiftUser: utilisateur
75     # swiftPassword: has to be set in vault-vitam.yml (encrypted) with same
76     swiftProjectName: monTenant
77     swiftProjectName: monTenant
78     ### Optional parameters
79     # swiftUrl: optional variable to force the swift URL
80     # swiftUrl: https://swift-hostname:port/swift/v1
81     #SSL TrustStore
82     swiftTrustStore: /chemin_vers_mon_fichier/monSwiftTrustStore.jks
83     #Max connection (concurrent connections), per route, to keep in pool (if
84     #a pooling ConnectionManager is used) (optional, 200 by default)

```

(suite sur la page suivante)

(suite de la page précédente)

```

84  swiftMaxConnectionsPerRoute: 200
85  #Max total connection (concurrent connections) to keep in pool (if a
↳pooling ConnectionManager is used) (optional, 1000 by default)
86  swiftMaxConnections: 1000
87  #Max time (in milliseconds) for waiting to establish connection
↳(optional, 200000 by default)
88  swiftConnectionTimeout: 200000
89  #Max time (in milliseconds) waiting for a data from the server (socket)
↳(optional, 60000 by default)
90  swiftReadTimeout: 60000
91  #Time (in seconds) to renew a token before expiration occurs (blocking)
↳(optional, 60 by default)
92  swiftHardRenewTokenDelayBeforeExpireTime: 60
93  #Time (in seconds) to renew a token before expiration occurs (optional,
↳300 by default)
94  swiftSoftRenewTokenDelayBeforeExpireTime: 300
95  # Offer log compaction
96  offer_log_compaction:
97  ## Expiration, here offer logs 21 days old will be compacted
98  expiration_value: 21
99  ## Choose one of "MILLENNIA", "HALF_DAYS", "MILLIS", "FOREVER", "MICROS
↳", "CENTURIES", "DECADES", "YEARS", "DAYS", "SECONDS", "HOURS", "MONTHS",
↳"WEEKS", "NANOS", "MINUTES", "ERAS"
100 expiration_unit: "DAYS"
101 ## Compaction bulk size here 10 000 offers logs (at most) will be
↳compacted (Expected value between 1 000 and 200 000)
102 compaction_size: 10000
103 # Batch processing thread pool size
104 maxBatchThreadPoolSize: 32
105 # Batch metadata computation timeout in seconds
106 batchMetadataComputationTimeout: 600
107 #####
↳###
108 offer-s3-1:
109 # provider : can only be amazon-s3-v1 for Amazon SDK S3 V1
110 provider: 'amazon-s3-v1'
111 # s3Endpoint : URL of connection to S3
112 s3Endpoint: https://s3.domain/
113 ### Optional parameters
114 # s3RegionName (optional): Region name (default value us-east-1)
115 s3RegionName: us-east-1
116 # s3SignerType (optional): Signing algorithm.
117 #   - signature V4 : 'AWSS3V4SignerType' (default value)
118 #   - signature V2 : 'S3SignerType'
119 s3SignerType: AWSS3V4SignerType
120 # s3PathStyleAccessEnabled (optional): 'true' to access bucket in "path-
↳style", else "virtual-hosted-style" (true by default)
121 s3PathStyleAccessEnabled: true
122 # s3MaxConnections (optional): Max total connection (concurrent
↳connections) (50 by default)
123 s3MaxConnections: 50
124 # s3ConnectionTimeout (optional): Max time (in milliseconds) for waiting
↳to establish connection (10000 by default)
125 s3ConnectionTimeout: 10000
126 # s3SocketTimeout (optional): Max time (in milliseconds) for reading
↳from a connected socket (50000 by default)
127 s3SocketTimeout: 50000

```

(suite sur la page suivante)

(suite de la page précédente)

```

128 # s3RequestTimeout (optional): Max time (in milliseconds) for a request
129 ↪ (0 by default, disabled)
129 s3RequestTimeout: 0
130 # s3ClientExecutionTimeout (optional): Max time (in milliseconds) for a
131 ↪ request by java client (0 by default, disabled)
131 s3ClientExecutionTimeout: 0
132 # Offer log compaction
133 offer_log_compaction:
134     ## Expiration, here offer logs 21 days old will be compacted
135     expiration_value: 21
136     ## Choose one of "MILLENNIA", "HALF_DAYS", "MILLIS", "FOREVER", "MICROS
137 ↪ ", "CENTURIES", "DECADES", "YEARS", "DAYS", "SECONDS", "HOURS", "MONTHS",
138 ↪ "WEEKS", "NANOS", "MINUTES", "ERAS"
137     expiration_unit: "DAYS"
138     ## Compaction bulk size here 10 000 offers logs (at most) will be
139 ↪ compacted (Expected value between 1 000 and 200 000)
139     compaction_size: 10000
140 # Batch processing thread pool size
141 maxBatchThreadPoolSize: 32
142 # Batch metadata computation timeout in seconds
143 batchMetadataComputationTimeout: 600
144 #####
145 ↪ ###
145 offer-tape-1:
146     provider: tape-library
147     tapeLibraryConfiguration:
148         maxTarEntrySize: 100000
149         maxTarFileSize: 1000000
150         # Enable overriding non empty cartridges
151         # WARNING : FOR DEV/TEST ONLY. DO NOT ENABLE IN PRODUCTION.
152         forceOverrideNonEmptyCartridges: false
153         # Archive (Tar) file expire time for retention in local FS
154         archiveRetentionCacheTimeoutInMinutes: 30
155         useSudo: false
156     topology:
157         buckets:
158             -
159                 name: test
160                 tenants: [0]
161                 tarBufferingTimeoutInMinutes: 60
162             -
163                 name: admin
164                 tenants: [1]
165                 tarBufferingTimeoutInMinutes: 60
166             -
167                 name: prod
168                 tenants: [2,3,4,5,6,7,8,9]
169                 tarBufferingTimeoutInMinutes: 60
170     tapeLibraries:
171         -
172             name: TAPE_LIB_1
173             robots:
174                 -
175                     device: /dev/tape/by-id/scsi-l1QUANTUM_10F73224E6664C84A1D00000
176                     mtXPath: "/usr/sbin/mtx"
177                     timeoutInMilliseconds: 3600000
178             drives:

```

(suite sur la page suivante)

(suite de la page précédente)

```

179         -
180         index: 0
181         device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_1235308739-nst
182         mtPath: "/bin/mt"
183         ddPath: "/bin/dd"
184         tarPath: "/bin/tar"
185         timeoutInMilliseconds: 3600000
186         readWritePriority: BACKUP
187         -
188         index: 1
189         device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0951859786-nst
190         mtPath: "/bin/mt"
191         ddPath: "/bin/dd"
192         tarPath: "/bin/tar"
193         timeoutInMilliseconds: 3600000
194         readWritePriority: READ
195         -
196         index: 2
197         device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0269493808-nst
198         mtPath: "/bin/mt"
199         ddPath: "/bin/dd"
200         tarPath: "/bin/tar"
201         timeoutInMilliseconds: 3600000
202         -
203         index: 3
204         device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0566471858-nst
205         mtPath: "/bin/mt"
206         ddPath: "/bin/dd"
207         tarPath: "/bin/tar"
208         readWritePriority: READ
209         timeoutInMilliseconds: 3600000
210     offer_log_compaction:
211         ## Expiration, here offer logs 21 days old will be compacted
212         expiration_value: 21
213         ## Choose one of "MILLENNIA", "HALF_DAYS", "MILLIS", "FOREVER", "MICROS
214         ↪ ", "CENTURIES", "DECADES", "YEARS", "DAYS", "SECONDS", "HOURS", "MONTHS",
215         ↪ "WEEKS", "NANOS", "MINUTES", "ERAS"
216         expiration_unit: "DAYS"
217         ## Compaction bulk size here 10 000 offers logs (at most) will be
218         ↪ compacted (Expected value between 1 000 and 200 000)
219         compaction_size: 10000
220         # Batch processing thread pool size
221         maxBatchThreadPoolSize: 32
222         # Batch metadata computation timeout in seconds
223         batchMetadataComputationTimeout: 600
224     #####
225     ↪ ###
226     # example_swift_v1:
227     #     provider: openstack-swift
228     #     swiftKeystoneAuthUrl: https://keystone/auth/1.0
229     #     swiftDomain: domain
230     #     swiftUser: user
231     #     swiftPassword: has to be set in vault-vitam.yml (encrypted) with same
232     ↪ structure => DO NOT COMMENT OUT
233     # THIS PART IS ONLY FOR CLEANING (and mandatory for this use case)
234     #     swiftProjectId: related to OS_PROJECT_ID
235     #     swiftRegionName: related to OS_REGION_NAME

```

(suite sur la page suivante)

(suite de la page précédente)

```

231 # swiftInterface: related to OS_INTERFACE
232 # example_swift_v3:
233 # provider: openstack-swift-v3
234 # swiftKeystoneAuthUrl: https://keystone/v3
235 # swiftDomain: domaine
236 # swiftUser: user
237 # swiftPassword: has to be set in vault-vitam.yml (encrypted) with same_
↳ structure => DO NOT COMMENT OUT
238 # swiftProjectName: monTenant
239 # projectName: monTenant
240 # THIS PART IS ONLY FOR CLEANING (and mandatory for this use case)
241 # swiftProjectId: related to OS_PROJECT_ID
242 # swiftRegionName: related to OS_REGION_NAME
243 # swiftInterface: related to OS_INTERFACE
244 #
245 # swiftTrustStore: /chemin_vers_mon_fichier/monSwiftTrustStore.jks
246 # swiftMaxConnectionsPerRoute: 200
247 # swiftMaxConnections: 1000
248 # swiftConnectionTimeout: 200000
249 # swiftReadTimeout: 60000
250 # Time (in seconds) to renew a token before expiration occurs
251 # swiftHardRenewTokenDelayBeforeExpireTime: 60
252 # swiftSoftRenewTokenDelayBeforeExpireTime: 300

```

Se référer aux commentaires dans le fichier pour le renseigner correctement.

Note : Dans le cas d'un déploiement multi-sites, dans la section `vitam_strategy`, la directive `vitam_site_name` définit pour l'offre associée le nom du datacenter Consul. Par défaut, si non définie, c'est la valeur de la variable `vitam_site_name` définie dans l'inventaire qui est prise en compte.

Avertissement : La cohérence entre l'inventaire et la section `vitam_strategy` (et `other_strategies` si multi-stratégies) est critique pour le bon déploiement et fonctionnement de la solution logicielle VITAM. En particulier, la liste d'offres de `vitam_strategy` doit correspondre *exactement* aux noms d'offres déclarés dans l'inventaire (ou les inventaires de chaque datacenter, en cas de fonctionnement multi-site).

Avertissement : Ne pas oublier, en cas de connexion à un keystone en https, de répercuter dans la *PKI* la clé publique de la *CA* du keystone.

4.2.3.2.4 Fichier `cots_vars.yml`

La configuration s'effectue dans le fichier `!repertoire_inventoir!group_vars/all/cots_vars.yml` :

```

1 ---
2
3 consul:
4   retry_interval: 10 # in seconds
5   check_interval: 10 # in seconds
6   check_timeout: 5 # in seconds
7   log_level: WARN # Available log_level are: TRACE, DEBUG, INFO, WARN or_
↳ ERR

```

(suite sur la page suivante)

(suite de la page précédente)

```

8   network: "ip_admin" # Which network to use for consul communications ?_
   ↳ip_admin or ip_service ?
9
10  consul_remote_sites:
11    # wan contains the wan addresses of the consul server instances of the_
   ↳external vitam sites
12    # Exemple, if our local dc is dc1, we will need to set dc2 & dc3 wan_
   ↳conf:
13    # - dc2:
14    #   wan: ["10.10.10.10", "1.1.1.1"]
15    # - dc3:
16    #   wan: ["10.10.10.11", "1.1.1.1"]
17  # Please uncomment and fill values if you want to connect VITAM to external_
   ↳SIEM
18  # external_siem:
19  #   host:
20  #   port:
21
22  elasticsearch:
23    log:
24      host: "elasticsearch-log.service.{{ consul_domain }}"
25      port_http: "9201"
26      groupe: "log"
27      baseuri: "elasticsearch-log"
28      cluster_name: "elasticsearch-log"
29      consul_check_http: 10 # in seconds
30      consul_check_tcp: 10 # in seconds
31      action_log_level: error
32      https_enabled: false
33      indices fielddata_cache_size: '30%' # related to https://www.elastic.
   ↳co/guide/en/elasticsearch/reference/7.6/modules-fielddata.html
34      indices_breaker_fielddata_limit: '40%' # related to https://www.
   ↳elastic.co/guide/en/elasticsearch/reference/7.6/circuit-breaker.html
   ↳#fielddata-circuit-breaker
35      dynamic_timeout: 30s
36      # default index template
37      index_templates:
38        default:
39          shards: 1
40          replica: 1
41        packetbeat:
42          shards: 5
43      log_appenders:
44        root:
45          log_level: "info"
46        rolling:
47          max_log_file_size: "100MB"
48          max_total_log_size: "5GB"
49          max_files: "50"
50        deprecation_rolling:
51          max_log_file_size: "100MB"
52          max_total_log_size: "1GB"
53          max_files: "10"
54          log_level: "warn"
55        index_search_slowlog_rolling:
56          max_log_file_size: "100MB"
57          max_total_log_size: "1GB"

```

(suite sur la page suivante)

(suite de la page précédente)

```

58         max_files: "10"
59         log_level: "warn"
60     index_indexing_slowlog_rolling:
61         max_log_file_size: "100MB"
62         max_total_log_size: "1GB"
63         max_files: "10"
64         log_level: "warn"
65     # By default, is commented. Should be uncommented if ansible
66     ↪ computes badly vCPUs number ; values are associated vCPUs numbers ;
67     ↪ please adapt to your configuration
68     # thread_pool:
69     #     index:
70     #         size: 2
71     #     get:
72     #         size: 2
73     #     search:
74     #         size: 2
75     #     write:
76     #         size: 2
77     #     warmer:
78     #         max: 2
79     data:
80         host: "elasticsearch-data.service.{{ consul_domain }}"
81         # default is 0.1 (10%) and should be quite enough in most cases
82         #index_buffer_size_ratio: "0.15"
83         port_http: "9200"
84         groupe: "data"
85         baseuri: "elasticsearch-data"
86         cluster_name: "elasticsearch-data"
87         consul_check_http: 10 # in seconds
88         consul_check_tcp: 10 # in seconds
89         action_log_level: debug
90         https_enabled: false
91         indices_fielddata_cache_size: '30%' # related to https://www.elastic.
92         ↪ co/guide/en/elasticsearch/reference/6.5/modules-fielddata.html
93         indices_breaker_fielddata_limit: '40%' # related to https://www.
94         ↪ elastic.co/guide/en/elasticsearch/reference/6.5/circuit-breaker.html
95         ↪ #fielddata-circuit-breaker
96         dynamic_timeout: 30s
97         # default index template
98         index_templates:
99         default:
100             shards: 1
101             replica: 2
102         log_appenders:
103         root:
104             log_level: "info"
105         rolling:
106             max_log_file_size: "100MB"
107             max_total_log_size: "5GB"
108             max_files: "50"
109         deprecation_rolling:
110             max_log_file_size: "100MB"
111             max_total_log_size: "5GB"
112             max_files: "50"
113             log_level: "warn"
114         index_search_slowlog_rolling:

```

(suite sur la page suivante)

(suite de la page précédente)

```

110         max_log_file_size: "100MB"
111         max_total_log_size: "5GB"
112         max_files: "50"
113         log_level: "warn"
114     index_indexing_slowlog_rolling:
115         max_log_file_size: "100MB"
116         max_total_log_size: "5GB"
117         max_files: "50"
118         log_level: "warn"
119     # By default, is commented. Should be uncommented if ansible_
    ↪ computes badly vCPUs number ; values are associated vCPUs numbers ;
    ↪ please adapt to your configuration
120     # thread_pool:
121     #     index:
122     #         size: 2
123     #     get:
124     #         size: 2
125     #     search:
126     #         size: 2
127     #     write:
128     #         size: 2
129     #     warmer:
130     #         max: 2
131
132 mongodb:
133     mongos_port: 27017
134     mongoc_port: 27018
135     mongod_port: 27019
136     mongo_authentication: "true"
137     host: "mongos.service.{{ consul_domain }}"
138     check_consul: 10 # in seconds
139     drop_info_log: false # Drop mongo (I)nformational log, for Verbosity_
    ↪ Level of 0
140     # logs configuration
141     logrotate: enabled # or disabled
142     history_days: 30 # How many days to store logs if logrotate is set to
    ↪ 'enabled'
143
144 logstash:
145     host: "logstash.service.{{ consul_domain }}"
146     user: logstash
147     port: 10514
148     rest_port: 20514
149     check_consul: 10 # in seconds
150     # logstash xms & xmx in Megabytes
151     # jvm_xms: 2048
152     # jvm_xmx: 2048
153     # workers_number: 4
154     log_appenders:
155         rolling:
156             max_log_file_size: "100MB"
157             max_total_log_size: "5GB"
158         json_rolling:
159             max_log_file_size: "100MB"
160             max_total_log_size: "5GB"
161
162 # Prometheus params

```

(suite sur la page suivante)

(suite de la page précédente)

```

163 prometheus:
164     metrics_path: /admin/v1/metrics
165     check_consul: 10 # in seconds
166     prometheus_config_file_target_directory: # Set path where "prometheus.yml
    ↪ " file will be generated. Example: /tmp/
167     server:
168         enabled: false
169         port: 19090
170     node_exporter:
171         enabled: true
172         port: 19100
173         metrics_path: /metrics
174     alertmanager:
175         enabled: false
176         api_port: 19093
177         cluster_port: 19094
178 grafana:
179     enabled: false
180     check_consul: 10 # in seconds
181     http_port: 13000
182
183 # Curator units: days
184 curator:
185     log:
186         metrics:
187             close: 7
188             delete: 30
189         logstash:
190             close: 7
191             delete: 30
192         metricbeat:
193             close: 5
194             delete: 10
195         packetbeat:
196             close: 5
197             delete: 10
198
199 kibana:
200     header_value: "reporting"
201     import_delay: 10
202     import_retries: 10
203     # logs configuration
204     logrotate: enabled # or disabled
205     history_days: 30 # How many days to store logs if logrotate is set to
    ↪ 'enabled'
206     log:
207         baseuri: "kibana_log"
208         api_call_timeout: 120
209         groupe: "log"
210         port: 5601
211         default_index_pattern: "logstash-vitam*"
212         check_consul: 10 # in seconds
213         # default shards & replica
214         shards: 1
215         replica: 1
216         # pour index logstash-*
217         metrics:

```

(suite sur la page suivante)

(suite de la page précédente)

```

218         shards: 1
219         replica: 1
220         # pour index metrics-vitam-*
221         logs:
222             shards: 1
223             replica: 1
224         # pour index metricbeat-*
225         metricbeat:
226             shards: 3 # must be a factor of 30
227             replica: 1
228         data:
229             baseuri: "kibana_data"
230             # OMA : bugdette : api_call_timeout is used for retries ; should_
231             ↪ ceate a separate variable rather than this one
232             api_call_timeout: 120
233             groupe: "data"
234             port: 5601
235             default_index_pattern: "logbookoperation_*"
236             check_consul: 10 # in seconds
237             # index template for .kibana
238             shards: 1
239             replica: 1
240     syslog:
241         # value can be syslog-ng or rsyslog
242         name: "rsyslog"
243
244     cerebro:
245         baseuri: "cerebro"
246         port: 9000
247         check_consul: 10 # in seconds
248
249     siegfried:
250         port: 19000
251         consul_check: 10 # in seconds
252
253     clamav:
254         port: 3310
255         # frequency freshclam for database update per day (from 0 to 24 - 24_
256         ↪ meaning hourly check)
257         db_update_periodicity: 1
258         # logs configuration
259         logrotate: enabled # or disabled
260         history_days: 30 # How many days to store logs if logrotate is set to
261         ↪ 'enabled'
262
263     mongo_express:
264         baseuri: "mongo-express"
265
266     ldap_authentication:
267         ldap_protocol: "ldap"
268         ldap_server: "{% if groups['ldap']|length > 0 %}{{ groups['ldap']|first }
269         ↪ }{% endif %}"
270         ldap_port: "389"
271         ldap_base: "dc=programmevitam,dc=fr"
272         ldap_login: "cn=Manager,dc=programmevitam,dc=fr"
273         uid_field: "uid"

```

(suite sur la page suivante)

(suite de la page précédente)

```

271 ldap_userDn_Template: "uid={0},ou=people,dc=programmevitam,dc=fr"
272 ldap_group_request: "(&(objectClass=groupOfNames)(member={0}))"
273 ldap_admin_group: "cn=admin,ou=groups,dc=programmevitam,dc=fr"
274 ldap_user_group: "cn=user,ou=groups,dc=programmevitam,dc=fr"
275 ldap_guest_group: "cn=guest,ou=groups,dc=programmevitam,dc=fr"
276
277 java_prerequisites:
278     debian: "openjdk-11-jre-headless"
279     redhat: "java-11-openjdk-headless"

```

Dans le cas du choix du *COTS* d'envoi des messages syslog dans logastsh, il est possible de choisir entre syslog-ng et rsyslog. Il faut alors modifier la valeur de la directive `syslog.name` ; la valeur par défaut est `rsyslog`.

Note : si vous décommentez et renseignez les valeurs dans le bloc `external_siem`, les messages seront envoyés (par syslog ou syslog-ng, selon votre choix de déploiement) dans un *SIEM* externe à la solution logicielle *VITAM*, aux valeurs indiquées dans le bloc ; il n'est alors pas nécessaire de renseigner de partitions pour les groupes `ansible [hosts_logstash]` et `[hosts_elasticsearch_log]`.

4.2.3.2.5 Fichier `tenants_vars.yml`

Indication : Fichier à créer depuis `tenants_vars.yml.example` et à paramétrer selon le besoin.

Le fichier `repertoire_inventory|group_vars/all/tenants_vars.yml` permet de gérer les configurations spécifiques associés aux tenants de la plateforme (liste des tenants, regroupement de tenants, configuration du nombre de shards et replicas, etc...).

```

1  ### tenants ###
2  # List of active tenants
3  vitam_tenant_ids: [0,1,2,3,4,5,6,7,8,9]
4  # List of dead / removed tenants that should never be reused / present in_
   ↳ vitam_tenant_ids
5  vitam_removed_tenants: []
6  # Administration tenant
7  vitam_tenant_admin: 1
8
9  ###
10 # Elasticsearch tenant indexation
11 # =====
12 #
13 # Elastic search index configuration settings :
14 # - 'number_of_shards' : number of shards per index. Every ES shard is_
   ↳ stored as a lucene index.
15 # - 'number_of_replicas': number of additional copies of primary shards
16 # The total number of shards : number_of_shards * (1 primary + M number_of_
   ↳ replicas)
17 #
18 # CAUTION : The total number of shards should be lower than or equal to the_
   ↳ number of elasticsearch-data instances in the cluster
19 #
20 # Default settings should be okay for most use cases.
21 # For more data-intensive workloads or deployments with high number of_
   ↳ tenants, custom tenant and/or collection configuration might be specified.

```

(suite sur la page suivante)

(suite de la page précédente)

```

22 #
23 # Tenant list may be specified as :
24 # - A specific tenant                               : eg.
   ↪ '1'
25 # - A tenant range                                   : eg.
   ↪ '10-19'
26 # - A comma-separated combination of specific tenants & tenant ranges : eg.
   ↪ '1, 5, 10-19, 50-59'
27 #
28 # Masterdata collections (accesscontract, filerules...) are indexed as
   ↪ single elasticsearch indexes :
29 # - Index name format : {collection}_{date_time_of_creation}. e.g.
   ↪ accesscontract_20200415_042011
30 # - Index alias name : {collection}. e.g. accesscontract
31 #
32 # Metadata collections (unit & objectgroup), and logbook operation
   ↪ collections are stored on a per-tenant index basis :
33 # - Index name      : {collection}_{tenant}_{date_time_of_creation}. e.g.
   ↪ unit_1_20200517_025041
34 # - Index alias name : {collection}_{tenant}. e.g. unit_1
35 #
36 # Very small tenants (1-100K entries) may be grouped in a "tenant group",
   ↪ and hence, stored in a single elasticsearch index.
37 # This allows reducing the number of indexes & shards that the elasticsearch
   ↪ cluster need to manage :
38 # - Index name      : {collection}_{tenant_group_name}_{date_time_of_
   ↪ creation}. e.g. logbookoperation_grp5_20200517_025041
39 # - Index alias name : {collection}_{tenant_group_name}. e.g.
   ↪ logbookoperation_grp5
40 #
41 # Tenant list can be wide ranges (eg: 100-199), and may contain non-existing
   ↪ (yet) tenants. i.e. tenant lists might be wider that 'vitam_tenant_ids'
   ↪ section
42 # This allows specifying predefined tenant families (whether normal tenants
   ↪ ranges, or tenant groups) to which tenants can be added in the future.
43 # However, tenant lists may not intersect (i.e. a single tenant cannot
   ↪ belong to 2 configuration sections).
44 #
45 # Sizing recommendations :
46 # - 1 shard per 5-10M records for small documents (eg. masterdata
   ↪ collections)
47 # - 1 shard per 1-2M records for larger documents (eg. metadata & logbook
   ↪ collections)
48 # - As a general rule, shard size should not exceed 30GB per shard
49 # - A single ES node should not handle > 200 shards (be it a primary or a
   ↪ replica)
50 # - It is recommended to start small and add more shards when needed (re-
   ↪ sharding requires a re-indexation operation)
51 #
52 # /\ IMPORTANT :
53 # Changing the configuration of an existing tenant requires re-indexation of
   ↪ the tenants and/or tenant groups
54 #
55 # Please refer to documentation for more details.
56 #
57 ###
58 vitam_elasticsearch_tenant_indexation:

```

(suite sur la page suivante)

(suite de la page précédente)

```

59
60 default_config:
61     # Default settings for masterdata collections (1 index per collection)
62     masterdata:
63         number_of_shards: 1
64         number_of_replicas: 2
65     # Default settings for unit indexes (1 index per tenant)
66     unit:
67         number_of_shards: 1
68         number_of_replicas: 2
69     # Default settings for object group indexes (1 index per tenant)
70     objectgroup:
71         number_of_shards: 1
72         number_of_replicas: 2
73     # Default settings for logbook operation indexes (1 index per tenant)
74     logbookoperation:
75         number_of_shards: 1
76         number_of_replicas: 2
77
78     ###
79     # Default masterdata collection indexation settings (default_config_
80     ↪section) apply for all master data collections
81     # Custom settings can be defined for the following masterdata collections:
82     # - accesscontract
83     # - accessionregisterdetail
84     # - accessionregistersummary
85     # - accessionregistersymbolic
86     # - agencies
87     # - archiveunitprofile
88     # - context
89     # - fileformat
90     # - filerules
91     # - griffin
92     # - ingestcontract
93     # - managementcontract
94     # - ontology
95     # - preservationscenario
96     # - profile
97     # - securityprofile
98     ###
99     masterdata:
100     # {collection}:
101     #     number_of_shards: 1
102     #     number_of_replicas: 2
103     # ...
104
105     ###
106     # Custom index settings for regular tenants.
107     ###
108     dedicated_tenants:
109     # - tenants: '1, 3, 11-20'
110     #     unit:
111     #         number_of_shards: 4
112     #         number_of_replicas: 0
113     #     objectgroup:
114     #         number_of_shards: 5

```

(suite sur la page suivante)

(suite de la page précédente)

```

115 #     number_of_replicas: 0
116 #     logbookoperation:
117 #     number_of_shards: 3
118 #     number_of_replicas: 0
119 # ...
120
121
122
123
124 ###
125 # Custom index settings for grouped tenants.
126 # Group name must meet the following criteria:
127 # - alphanumeric characters
128 # - lowercase only
129 # - not start with a number
130 # - be less than 64 characters long.
131 # - NO special characters - / _ | ...
132 ###
133 grouped_tenants:
134 # - name: 'grp1'
135 #   tenants: '5-10'
136 #   unit:
137 #     number_of_shards: 5
138 #     number_of_replicas: 0
139 #   objectgroup:
140 #     number_of_shards: 6
141 #     number_of_replicas: 0
142 #   logbookoperation:
143 #     number_of_shards: 7
144 #     number_of_replicas: 0
145 # ...

```

Se référer aux commentaires dans le fichier pour le renseigner correctement.

Une attention particulière doit être portée à la configuration du nombre de shards et de replicas dans le paramètre `vitam_elasticsearch_tenant_indexation.default_config` (le fichier `tenants_vars.yml`. exemple représente les valeurs recommandées par Vitam dans le cadre d'un déploiement en production). Ce paramètre est obligatoire.

Voir aussi :

Se référer au chapitre « Gestion des indexes Elasticsearch dans un contexte massivement multi-tenants » du [DEX](#) pour plus d'informations sur cette fonctionnalité.

Avertissement : 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.

Voir aussi :

Se référer au chapitre « Réindexation » du [DEX](#) pour plus d'informations.

4.2.3.3 Déclaration des secrets

Avertissement : L'ensemble des mots de passe fournis ci-après le sont par défaut et doivent être changés !

4.2.3.3.1 vitam

Avertissement : Cette section décrit des fichiers contenant des données sensibles. Il est important d'implémenter une politique de mot de passe robuste conforme à ce que l'ANSSI préconise. Par exemple : ne pas utiliser le même mot de passe pour chaque service, renouveler régulièrement son mot de passe, utiliser des majuscules, minuscules, chiffres et caractères spéciaux (Se référer à la documentation ANSSI <https://www.ssi.gouv.fr/guide/mot-de-passe>). En cas d'usage d'un fichier de mot de passe (*vault-password-file*), il faut renseigner ce mot de passe comme contenu du fichier et ne pas oublier de sécuriser ou supprimer ce fichier à l'issue de l'installation.

Les secrets utilisés par la solution logicielle (en-dehors des certificats qui sont abordés dans une section ultérieure) sont définis dans des fichiers chiffrés par `ansible-vault`.

Important : Tous les vault présents dans l'arborescence d'inventaire doivent être tous protégés par le même mot de passe !

La première étape consiste à changer les mots de passe de tous les vaults présents dans l'arborescence de déploiement (le mot de passe par défaut est contenu dans le fichier `vault_pass.txt`) à l'aide de la commande `ansible-vault rekey <fichier vault>`.

Voici la liste des vaults pour lesquels il est nécessaire de modifier le mot de passe :

- `environments/group_vars/all/vault-vitam.yml`
- `environments/group_vars/all/vault-keystores.yml`
- `environments/group_vars/all/vault-extra.yml`
- `environments/certs/vault-certs.yml`

2 vaults sont principalement utilisés dans le déploiement d'une version :

Avertissement : Leur contenu est donc à modifier avant tout déploiement.

- Le fichier `repertoire_inventori/group_vars/all/vault-vitam.yml` contient les secrets généraux :

```

1  ---
2  # Vitam platform secret key
3  plateforme_secret: vitamsecret
4
5  # The consul key must be 16-bytes, Base64 encoded: https://www.consul.io/docs/
6  # ↪ agent/encryption.html
7  # You can generate it with the "consul keygen" command
8  # Or you can use this script: deployment/pki/scripts/generate_consul_key.sh
9  consul_encrypt: Biz14ohqN4HtvZmrXp3N4A==
10
11 mongodb:
12   mongo-data:
13     passphrase: changeitkM4L6zBgK527tWBb

```

(suite sur la page suivante)

(suite de la page précédente)

```

13  admin:
14      user: vitamdb-admin
15      password: change_it_1MpG22m2MywvKW5E
16  localadmin:
17      user: vitamdb-localadmin
18      password: change_it_HycFEVD74g397iRe
19  system:
20      user: vitamdb-system
21      password: change_it_HycFEVD74g397iRe
22  metadata:
23      user: metadata
24      password: change_it_37b97KVADV8YbCwt
25  logbook:
26      user: logbook
27      password: change_it_jVi6q8eX4H1Ce8UC
28  report:
29      user: report
30      password: change_it_jb7TASZbU6n85t8L
31  functionalAdmin:
32      user: functional-admin
33      password: change_it_9eA2zMCL6tm6KF1e
34  securityInternal:
35      user: security-internal
36      password: change_it_m39XvRQWixyDX566
37  offer-fs-1:
38      passphrase: changeitmB5rnk1M5TY61PqZ
39      admin:
40          user: vitamdb-admin
41          password: change_it_FLkM5emt63N73EcN
42      localadmin:
43          user: vitamdb-localadmin
44          password: change_it_QeH8q4e16ah4QKXS
45      system:
46          user: vitamdb-system
47          password: change_it_HycFEVD74g397iRe
48      offer:
49          user: offer
50          password: change_it_pQi1T1yT9LAF8au8
51  offer-fs-2:
52      passphrase: changeiteSY1By57qZr4MX2s
53      admin:
54          user: vitamdb-admin
55          password: change_it_84aTMFZ7h8e2NgMe
56      localadmin:
57          user: vitamdb-localadmin
58          password: change_it_Am1B37tGY1w5VfvX
59      system:
60          user: vitamdb-system
61          password: change_it_HycFEVD74g397iRe
62      offer:
63          user: offer
64          password: change_it_mLDYds957sNQ53mA
65  offer-tape-1:
66      passphrase: changeitmB5rnk1M5TY61PqZ
67      admin:
68          user: vitamdb-admin
69          password: change_it_FLkM5emt63N73EcN

```

(suite sur la page suivante)

(suite de la page précédente)

```

70     localadmin:
71         user: vitamdb-localadmin
72         password: change_it_QeH8q4e16ah4QKXS
73     system:
74         user: vitamdb-system
75         password: change_it_HycFEVD74g397iRe
76     offer:
77         user: offer
78         password: change_it_pQilT1yT9LAF8au8
79 offer-swift-1:
80     passphrase: changeitgYvt42M2pKL6Zx3T
81     admin:
82         user: vitamdb-admin
83         password: change_it_e21hLp51WNa4sJFS
84     localadmin:
85         user: vitamdb-localadmin
86         password: change_it_QB8857SJrGrQh2yu
87     system:
88         user: vitamdb-system
89         password: change_it_HycFEVD74g397iRe
90     offer:
91         user: offer
92         password: change_it_AWJg2Bp3s69P6nMe
93 offer-s3-1:
94     passphrase: changeituF1jVdR9NqdTG625
95     admin:
96         user: vitamdb-admin
97         password: change_it_5b7cSWcS5M1NF4kv
98     localadmin:
99         user: vitamdb-localadmin
100        password: change_it_S9jE24rxHwUZP6y5
101     system:
102         user: vitamdb-system
103         password: change_it_HycFEVD74g397iRe
104     offer:
105         user: offer
106         password: change_it_TuTB1i2k7iQW3zL2
107 offer-tape-1:
108     passphrase: changeituF1jghT9NqdTG625
109     admin:
110         user: vitamdb-admin
111         password: change_it_5b7cSWcab91NF4kv
112     localadmin:
113         user: vitamdb-localadmin
114         password: change_it_S9jE24rxHwUZP5a6
115     system:
116         user: vitamdb-system
117         password: change_it_HycFEVD74g397iRe
118     offer:
119         user: offer
120         password: change_it_TuTB1i2k7iQW3c2a
121
122 vitam_users:
123     - vitam_aadmin:
124         login: aadmin
125         password: change_it_z5MP7GC4qnR8nL9t
126         role: admin

```

(suite sur la page suivante)

(suite de la page précédente)

```

127 - vitam_uuser:
128   login: uuser
129   password: change_it_w94Q3jPAT2aJYm8b
130   role: user
131 - vitam_gguest:
132   login: gguest
133   password: change_it_E5v7Tr4h6tYaQG2W
134   role: guest
135 - techadmin:
136   login: techadmin
137   password: change_it_K29EluHcPZ8zXji8
138   role: admin
139
140 ldap_authentication:
141   ldap_pwd: "change_it_t69Rn5NdUv39EYkC"
142
143 admin_basic_auth_password: change_it_5Yn74JgXwbQ9KdP8
144
145 vitam_offers:
146   offer-swift-1:
147     swiftPassword: change_it_m44j57aYeRpnPXQ2
148   offer-s3-1:
149     s3AccessKey: accessKey_change_grLS8372Uga5EJSx
150     s3SecretKey: secretKey_change_p97es2m2CHXPJA1m

```

Prudence : Seuls les caractères alphanumériques sont valides pour les directives passphrase.

Avertissement : Le paramétrage du mode d'authentications des utilisateurs à l'*IHM* démo est géré au niveau du fichier `deployment/environments/group_vars/all/vitam_vars.yml`. Plusieurs modes d'authentications sont proposés au niveau de la section `authentication_realms`. Dans le cas d'une authentification se basant sur le mécanisme `iniRealm` (configuration `shiro` par défaut), les mots de passe déclarés dans la section `vitam_users` devront s'appuyer sur une politique de mot de passe robuste, comme indiqué en début de chapitre. Il est par ailleurs possible de choisir un mode d'authentification s'appuyant sur un annuaire LDAP externe (`ldapRealm` dans la section `authentication_realms`).

Note : Dans le cadre d'une installation avec au moins une offre *swift*, il faut déclarer, dans la section `vitam_offers`, le nom de chaque offre et le mot de passe de connexion *swift* associé, défini dans le fichier `offers_opts.yml`. L'exemple ci-dessus présente la déclaration du mot de passe pour l'offre *swift offer-swift-1*.

Note : Dans le cadre d'une installation avec au moins une offre *s3*, il faut déclarer, dans la section `vitam_offers`, le nom de chaque offre et l'accès key secret *s3* associé, défini dans le fichier `offers_opts.yml`. L'exemple ci-dessus présente la déclaration du mot de passe pour l'offre *s3 offer-s3-1*.

- Le fichier `repertoire_inventory|group_vars/all/vault-keystores.yml` contient les mots de passe des magasins de certificats utilisés dans VITAM :

```
1  # NO UNDERSCORE ALLOWED IN VALUES
2  keystores:
3    server:
4      offer: changeit817NR75vWsZtgAgJ
5      access_external: changeitMZFD2YM4279miitu
6      ingest_external: changeita2C74cQhy84BLWCr
7      ihm_recette: changeit4FWYVK1347mxjGfe
8      ihm_demo: changeit6kQl6eyDY7QPS9fy
9    client_external:
10     ihm_demo: changeitGT38hhTiA32x1PLy
11     gatling: changeit2sBC5ac7NfGF9Qj7
12     ihm_recette: changeitdAZ9Eq65UhDZd9p4
13     reverse: changeite5XTzb5yVPcEX464
14     vitam_admin_int: changeitz6xZe5gDu7nhDZd9
15    client_storage:
16     storage: changeit647D7LWiyM6qYMnm
17    timestamping:
18     secure_logbook: changeitMn9Skuyx87VYU62U
19     secure_storage: changeite5gDu9Skuy84BLW9
20  truststores:
21    server: changeitxNe4JLfn528PVHj7
22    client_external: changeitJ2eS93DcPH1v4jAp
23    client_storage: changeitHpSCa31aG8ttB87S
24  grantedstores:
25    client_external: changeitLL22HkmDCA2e2vj7
26    client_storage: changeitR3wvp5C8KQS76Vcu
```

Avvertissement : Il convient de sécuriser votre environnement en définissant des mots de passe forts.

4.2.3.3.2 Cas des extras

- Le fichier `repertoire_inventory/group_vars/all/vault-extra.yml` contient les mots de passe des magasins de certificats utilisés dans VITAM :

```
1  # Example for git lfs ; uncomment & use if needed
2  #vitam_gitlab_itest_login: "account"
3  #vitam_gitlab_itest_password: "change_it_4DU42JVf2x2xmPBs"
```

Note : Il est possible, depuis le fichier `cots_var.yml` d'activer ou désactiver l'installation de la stack prometheus et grafana. La co-localisation de la stack prometheus et grafana est fortement recommandable.

Note : Le playbook `vitam.yml` comprend des étapes avec la mention `no_log` afin de ne pas afficher en clair des étapes comme les mots de passe des certificats. En cas d'erreur, il est possible de retirer la ligne dans le fichier pour une analyse plus fine d'un éventuel problème sur une de ces étapes.

4.2.3.3.3 Commande ansible-vault

Certains fichiers présents sous `repertoire_inventory/group_vars/all` commençant par **vault-** doivent être protégés (encryptés) avec l'utilitaire `ansible-vault`.

Note : Ne pas oublier de mettre en conformité le fichier `vault_pass.txt`

4.2.3.3.3.1 Générer des fichiers *vaultés* depuis des fichiers en clair

Exemple du fichier `vault-cots.example`

```
cp vault-cots.example vault-cots.yml
ansible-vault encrypt vault-cots.yml
```

4.2.3.3.3.2 Ré-encoder un fichier *vaulté*

Exemple du fichier `vault-cots.yml`

```
ansible-vault rekey vault-cots.yml
```

4.2.3.4 Le mapping Elasticsearch pour Unit et ObjectGroup

Les mappings des indexes elasticsearch pour les collections masterdata Unit et ObjectGroup sont configurables de l'extérieur, plus spécifiquement dans le dossier `repertoire_inventory/deployment/ansible-vitam/roles/elasticsearch-mapping/files/`, ce dossier contient :

- `deployment/ansible-vitam/roles/elasticsearch-mapping/files/unit-es-mapping.json`
- `deployment/ansible-vitam/roles/elasticsearch-mapping/files/og-es-mapping.json`

Exemple du fichier mapping de la collection ObjectGroup :

```
1  {
2    "dynamic_templates": [
3      {
4        "object": {
5          "match_mapping_type": "object",
6          "mapping": {
7            "type": "object"
8          }
9        }
10     ],
11     {
12       "all_string": {
13         "match": "*",
14         "mapping": {
15           "type": "text"
16         }
17       }
18     }
19   }
```

(suite sur la page suivante)

(suite de la page précédente)

```

19   },
20   "properties": {
21     "FileInfo": {
22       "properties": {
23         "CreatingApplicationName": {
24           "type": "text"
25         },
26         "CreatingApplicationVersion": {
27           "type": "text"
28         },
29         "CreatingOs": {
30           "type": "text"
31         },
32         "CreatingOsVersion": {
33           "type": "text"
34         },
35         "DateCreatedByApplication": {
36           "type": "date",
37           "format": "strict_date_optional_time"
38         },
39         "Filename": {
40           "type": "text"
41         },
42         "LastModified": {
43           "type": "date",
44           "format": "strict_date_optional_time"
45         }
46       }
47     },
48     "Metadata": {
49       "properties": {
50         "Text": {
51           "type": "object"
52         },
53         "Document": {
54           "type": "object"
55         },
56         "Image": {
57           "type": "object"
58         },
59         "Audio": {
60           "type": "object"
61         },
62         "Video": {
63           "type": "object"
64         }
65       }
66     },
67     "OtherMetadata": {
68       "type": "object",
69       "properties": {
70         "RawMetadata": {
71           "type": "object"
72         }
73       }
74     },
75     "_profil": {

```

(suite sur la page suivante)

(suite de la page précédente)

```

76     "type": "keyword"
77   },
78   "_qualifiers": {
79     "properties": {
80       "_nbc": {
81         "type": "long"
82       },
83       "qualifier": {
84         "type": "keyword"
85       },
86       "versions": {
87         "type": "nested",
88         "properties": {
89           "Compressed": {
90             "type": "text"
91           },
92           "DataObjectGroupId": {
93             "type": "keyword"
94           },
95           "DataObjectVersion": {
96             "type": "keyword"
97           },
98           "DataObjectSystemId": {
99             "type": "keyword"
100          },
101          "DataObjectGroupSystemId": {
102            "type": "keyword"
103          },
104          "_opi": {
105            "type": "keyword"
106          },
107          "FileInfo": {
108            "properties": {
109              "CreatingApplicationName": {
110                "type": "text"
111              },
112              "CreatingApplicationVersion": {
113                "type": "text"
114              },
115              "CreatingOs": {
116                "type": "text"
117              },
118              "CreatingOsVersion": {
119                "type": "text"
120              },
121              "DateCreatedByApplication": {
122                "type": "date",
123                "format": "strict_date_optional_time"
124              },
125              "Filename": {
126                "type": "text"
127              },
128              "LastModified": {
129                "type": "date",
130                "format": "strict_date_optional_time"
131              }
132            }
133          }
134        }
135      }
136    }
137  }

```

(suite sur la page suivante)

(suite de la page précédente)

```

133     },
134     "FormatIdentification": {
135         "properties": {
136             "FormatId": {
137                 "type": "keyword"
138             },
139             "FormatLitteral": {
140                 "type": "keyword"
141             },
142             "MimeType": {
143                 "type": "keyword"
144             },
145             "Encoding": {
146                 "type": "keyword"
147             }
148         }
149     },
150     "MessageDigest": {
151         "type": "keyword"
152     },
153     "Algorithm": {
154         "type": "keyword"
155     },
156     "PhysicalDimensions": {
157         "properties": {
158             "Diameter": {
159                 "properties": {
160                     "unit": {
161                         "type": "keyword"
162                     },
163                     "dValue": {
164                         "type": "double"
165                     }
166                 }
167             },
168             "Height": {
169                 "properties": {
170                     "unit": {
171                         "type": "keyword"
172                     },
173                     "dValue": {
174                         "type": "double"
175                     }
176                 }
177             },
178             "Depth": {
179                 "properties": {
180                     "unit": {
181                         "type": "keyword"
182                     },
183                     "dValue": {
184                         "type": "double"
185                     }
186                 }
187             },
188             "Shape": {
189                 "type": "keyword"

```

(suite sur la page suivante)

(suite de la page précédente)

```

190         },
191         "Thickness": {
192             "properties": {
193                 "unit": {
194                     "type": "keyword"
195                 },
196                 "dValue": {
197                     "type": "double"
198                 }
199             }
200         },
201         "Length": {
202             "properties": {
203                 "unit": {
204                     "type": "keyword"
205                 },
206                 "dValue": {
207                     "type": "double"
208                 }
209             }
210         },
211         "NumberOfPage": {
212             "type": "long"
213         },
214         "Weight": {
215             "properties": {
216                 "unit": {
217                     "type": "keyword"
218                 },
219                 "dValue": {
220                     "type": "double"
221                 }
222             }
223         },
224         "Width": {
225             "properties": {
226                 "unit": {
227                     "type": "keyword"
228                 },
229                 "dValue": {
230                     "type": "double"
231                 }
232             }
233         }
234     },
235     "PhysicalId": {
236         "type": "keyword"
237     },
238     "Size": {
239         "type": "long"
240     },
241     "Uri": {
242         "type": "keyword"
243     },
244     "_id": {
245         "type": "keyword"
246     }

```

(suite sur la page suivante)

(suite de la page précédente)

```

247         },
248         "_storage": {
249             "properties": {
250                 "_nbc": {
251                     "type": "long"
252                 },
253                 "offerIds": {
254                     "type": "keyword"
255                 },
256                 "strategyId": {
257                     "type": "keyword"
258                 }
259             }
260         }
261     }
262 }
263
264 },
265 "_v": {
266     "type": "long"
267 },
268 "_av": {
269     "type": "long"
270 },
271 "_nbc": {
272     "type": "long"
273 },
274 "_ops": {
275     "type": "keyword"
276 },
277 "_opi": {
278     "type": "keyword"
279 },
280 "_sp": {
281     "type": "keyword"
282 },
283 "_sps": {
284     "type": "keyword"
285 },
286 "_tenant": {
287     "type": "long"
288 },
289 "_up": {
290     "type": "keyword"
291 },
292 "_uds": {
293     "type": "object",
294     "enabled": false
295 },
296 "_us": {
297     "type": "keyword"
298 },
299 "_storage": {
300     "properties": {
301         "_nbc": {
302             "type": "long"
303         },

```

(suite sur la page suivante)

(suite de la page précédente)

```

304     "offerIds": {
305         "type": "keyword"
306     },
307     "strategyId": {
308         "type": "keyword"
309     }
310 },
311 "_glpd": {
312     "enabled": false
313 }
314 }
315 }
316 }

```

Note : Le paramétrage de ce mapping se fait sur les deux composants `metadata` et le composant extra `'ihm-recette'`.

Prudence : En cas de changement du mapping, il faut veiller à ce que cette mise à jour soit en accord avec l'Ontologie de *VITAM*.

Le mapping est pris en compte lors de la première création des indexes. Pour une nouvelle installation de *VITAM*, les mapping seront automatiquement pris en compte. Cependant, la modification des mappings nécessite une réindexation via l'API dédiée si VITAM est déjà installé.

4.2.4 Gestion des certificats

Une vue d'ensemble de la gestion des certificats est présentée *dans l'annexe dédiée* (page 105).

4.2.4.1 Cas 1 : Configuration développement / tests

Pour des usages de développement ou de tests hors production, il est possible d'utiliser la *PKI* fournie avec la solution logicielle *VITAM*.

4.2.4.1.1 Procédure générale

Danger : La *PKI* fournie avec la solution logicielle *VITAM* doit être utilisée UNIQUEMENT pour faire des tests, et ne doit par conséquent surtout pas être utilisée en environnement de production ! De plus il n'est pas possible de l'utiliser pour générer les certificats d'une autre application qui serait cliente de VITAM.

La *PKI* de la solution logicielle *VITAM* est une suite de scripts qui vont générer dans l'ordre ci-dessous :

- Les autorités de certification (*CA*)
- Les certificats (clients, serveurs, de *timestamping*) à partir des *CA*
- Les *keystores*, en important les certificats et *CA* nécessaires pour chacun des *keystores*

4.2.4.1.2 Génération des CA par les scripts Vitam

Il faut faire la génération des autorités de certification (CA) par le script décrit ci-dessous.

Dans le répertoire de déploiement, lancer le script :

```
pki/scripts/generate_ca.sh
```

Ce script génère sous `pki/ca` les autorités de certification *root* et intermédiaires pour générer des certificats clients, serveurs, et de timestamping. Les mots de passe des clés privées des autorités de certification sont stockés dans le vault ansible `environments/certs/vault-ca.yml`

Avvertissement : Il est impératif de noter les dates de création et de fin de validité des CA. En cas d'utilisation de la PKI fournie, la CA root a une durée de validité de 10 ans ; la CA intermédiaire a une durée de 3 ans.

4.2.4.1.3 Génération des certificats par les scripts Vitam

Le fichier d'inventaire de déploiement `environments/<fichier d'inventaire>` (cf. *Informations plateforme* (page 21)) doit être correctement renseigné pour indiquer les serveurs associés à chaque service. En prérequis les CA doivent être présentes.

Puis, dans le répertoire de déploiement, lancer le script :

```
pki/scripts/generate_certs.sh <fichier d'inventaire>
```

Ce script génère sous `environments/certs` les certificats (format `crt` & `key`) nécessaires pour un bon fonctionnement dans VITAM. Les mots de passe des clés privées des certificats sont stockés dans le vault ansible `environments/certs/vault-certs.yml`.

Prudence : Les certificats générés à l'issue ont une durée de validité de 3 ans.

4.2.4.2 Cas 2 : Configuration production

4.2.4.2.1 Procédure générale

La procédure suivante s'applique lorsqu'une PKI est déjà disponible pour fournir les certificats nécessaires.

Les étapes d'intégration des certificats à la solution Vitam sont les suivantes :

- Générer les certificats avec les bons *key usage* par type de certificat
- Déposer les certificats et les autorités de certifications correspondantes dans les bons répertoires.
- Renseigner les mots de passe des clés privées des certificats dans le vault ansible `environments/certs/vault-certs.yml`
- Utiliser le script VITAM permettant de générer les différents *keystores*.

Note : Rappel pré-requis : vous devez disposer d'une ou plusieurs PKI pour tout déploiement en production de la solution logicielle VITAM.

4.2.4.2.2 Génération des certificats

En conformité avec le document RGSV2 de l'ANSSI, il est recommandé de générer des certificats avec les caractéristiques suivantes :

4.2.4.2.2.1 Certificats serveurs

- **Key Usage**
 - digitalSignature, keyEncipherment
- **Extended Key Usage**
 - TLS Web Server Authentication

Les certificats serveurs générés doivent prendre en compte des alias « web » (`subjectAltName`).

Le `subjectAltName` des certificats serveurs (`deployment/environments/certs/server/hosts/*`) doit contenir le nom DNS du service sur consul associé.

Exemple avec un cas standard : `<composant_vitam>.service.<consul_domain>`. Ce qui donne pour le certificat serveur de `access-external` par exemple :

```
X509v3 Subject Alternative Name:
    DNS:access-external.service.consul, DNS:localhost
```

Il faudra alors mettre le même nom de domaine pour la configuration de Consul (fichier `deployment/environments/group_vars/all/vitam_vars.yml`, variable `consul_domain`)

Cas particulier pour `ihm-demo` et `ihm-recette` : il faut ajouter le nom *DNS* qui sera utilisé pour requêter ces deux applications, si celles-ci sont appelées directement en frontal https.

4.2.4.2.2.2 Certificat clients

- **Key Usage**
 - digitalSignature
- **Extended Key Usage**
 - TLS Web Client Authentication

4.2.4.2.2.3 Certificats d'horodatage

Ces certificats sont à générer pour les composants `logbook` et `storage`.

- **Key Usage**
 - digitalSignature, nonRepudiation
- **Extended Key Usage**
 - Time Stamping

4.2.4.2.3 Intégration de certificats existants

Une fois les certificats et *CA* mis à disposition par votre *PKI*, il convient de les positionner sous `environments/certs/...` en respectant la structure indiquée ci-dessous.

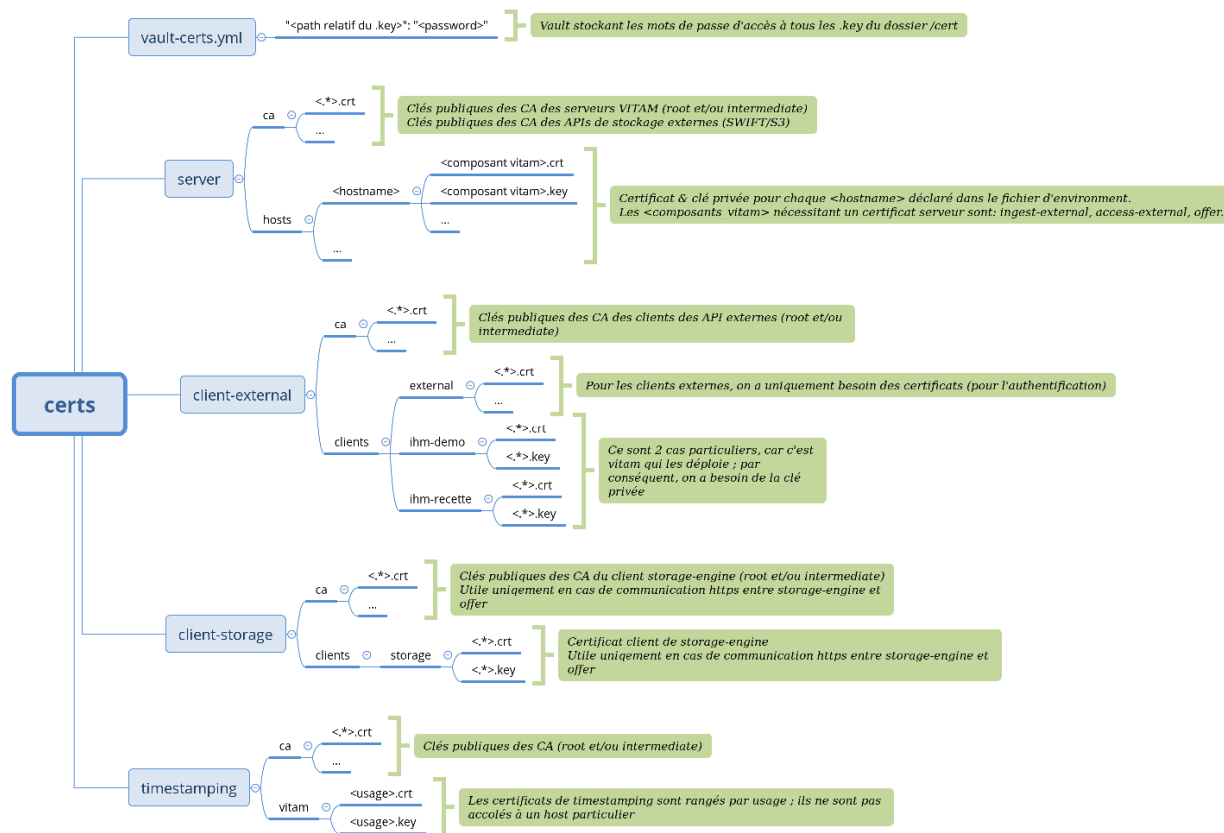


Fig. 3 – Vue détaillée de l'arborescence des certificats

Astuce : Dans le doute, n'hésitez pas à utiliser la *PKI* de test (étapes de génération de *CA* et de certificats) pour générer les fichiers requis au bon endroit et ainsi observer la structure exacte attendue ; il vous suffira ensuite de remplacer ces certificats « placeholders » par les certificats définitifs avant de lancer le déploiement.

Ne pas oublier de renseigner le vault contenant les *passphrases* des clés des certificats : `environments/certs/vault-cert.yml`

Pour modifier/créer un vault ansible, se référer à la documentation Ansible sur [cette url](http://docs.ansible.com/ansible/playbooks_vault.html) ¹⁴.

Prudence : Durant l'installation de VITAM, il est nécessaire de créer un certificat « vitam-admin-int » (à placer sous `deployment/environments/certs/client-external/clients/vitam-admin-int`).

http://docs.ansible.com/ansible/playbooks_vault.html

Prudence : Durant l'installation des extra de VITAM, il est nécessaire de créer un certificat « gatling » (à placer sous `deployment/environments/certs/client-external/clients/gatling`).

4.2.4.2.4 Intégration de certificats clients de VITAM

4.2.4.2.4.1 Intégration d'une application externe (cliente)

Dans le cas d'ajout de certificats *SIA* externes au déploiement de la solution logicielle *VITAM* :

- Déposer le certificat (.crt) de l'application client dans `environments/certs/client-external/clients/external/`
- Déposer les *CA* du certificat de l'application (.crt) dans `environments/certs/client-external/ca/`
- Editer le fichier `environments/group_vars/all/vitam_security.yml` et ajouter le(s) entrée(s) supplémentaire(s) (sous forme répertoire/fichier.crt, exemple : `external/mon_sia.crt`) dans la directive `admin_context_certs` pour que celles-ci soient associés aux contextes de sécurité durant le déploiement de la solution logicielle *VITAM*.

Note : Les certificats *SIA* externes ajoutés par le mécanisme de déploiement sont, par défaut, rattachés au contexte applicatif d'administration `admin_context_name` lui même associé au profil de sécurité `admin_security_profile` et à la liste de tenants `vitam_tenant_ids` (voir le fichier `environments/group_vars/all/vitam_security.yml`). Pour l'ajout de certificats applicatifs associés à des contextes applicatifs autres, se référer à la procédure du document d'exploitation (*DEX*) décrivant l'intégration d'une application externe dans Vitam.

4.2.4.2.4.2 Intégration d'un certificat personnel (*personae*)

Dans le cas d'ajout de certificats personnels au déploiement de la solution logicielle *VITAM* :

- Déposer le certificat personnel (.crt) dans `environments/certs/client-external/clients/external/`
- Editer le fichier `environments/group_vars/all/vitam_security.yml` et ajouter le(s) entrée(s) supplémentaire(s) (sous forme répertoire/fichier.crt, exemple : `external/mon_personae.crt`) dans la directive `admin_personal_certs` pour que ceux-ci soient ajoutés à la base de données du composant *security-internal* durant le déploiement de la solution logicielle *VITAM*.

4.2.4.2.5 Cas des offres objet

Placer le .crt de la *CA* dans `deployment/environments/certs/server/ca`.

4.2.4.2.6 Absence d'usage d'un *reverse*

Dans ce cas, il convient de :

- supprimer le répertoire `deployment/environments/certs/client-external/clients/reverse`
- supprimer les entrées **reverse** dans le fichier `vault_keystore.yml`

4.2.4.3 Intégration de CA pour une offre *Swift* ou *s3*

En cas d'utilisation d'une offre *Swift* ou *s3* en https, il est nécessaire d'ajouter les *CA* du certificat de l'*API Swift* ou *s3*.

Il faut les déposer dans `environments/certs/server/ca/` avant de jouer le script `./generate_keystores.sh`

4.2.4.4 Génération des magasins de certificats

En prérequis, les certificats et les autorités de certification (*CA*) doivent être présents dans les répertoires attendus.

Prudence : Avant de lancer le script de génération des *stores*, il est nécessaire de modifier le vault contenant les mots de passe des *stores* : `environments/group_vars/all/vault-keystores.yml`, décrit dans la section *Déclaration des secrets* (page 46).

Lancer le script : `./generate_stores.sh`

Ce script génère sous `environments/keystores` les *stores* (aux formats `jks` / `p12`) associés pour un bon fonctionnement dans la solution logicielle *VITAM*.

Il est aussi possible de déposer directement les *keystores* au bon format en remplaçant ceux fournis par défaut et en indiquant les mots de passe d'accès dans le vault : `environments/group_vars/all/vault-keystores.yml`

Note : Le mot de passe du fichier `vault-keystores.yml` est identique à celui des autres *vaults* ansible.

4.2.5 Paramétrages supplémentaires

4.2.5.1 *Tuning JVM*

Prudence : En cas de colocalisation, bien prendre en compte la taille *JVM* de chaque composant (*VITAM* : `-Xmx512m` par défaut) pour éviter de *swapper*.

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

Pour chaque composant, il est possible de modifier ces 3 variables :

- `memory` : paramètres `Xms` et `Xmx`
- `gc` : paramètres `gc`
- `java` : autres paramètres `java`

4.2.5.2 Installation des *griffins* (greffons de préservation)

Note : Fonctionnalité disponible partir de la R9 (2.1.1) .

Prudence : Cette version de *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.

Il est possible de choisir les *griffins* installables sur la plate-forme. Pour cela, il faut éditer le contenu du fichier `deployment/environments/group_vars/all/vitam_vars.yml` au niveau de la directive `vitam_griffins`. Cette action est à rapprocher de l'incorporation des binaires d'installation : les binaires d'installation des greffons doivent être accessibles par les machines hébergeant le composant **worker**.

Exemple :

```
vitam_griffins: ["vitam-imagemagick-griffin", "vitam-jhove-griffin"]
```

Voici la liste des greffons disponibles au moment de la présente publication :

```
vitam-imagemagick-griffin
vitam-jhove-griffin
vitam-libreoffice-griffin
vitam-odfvalidator-griffin
vitam-siegfried-griffin
vitam-tesseract-griffin
vitam-verapdf-griffin
vitam-ffmpeg-griffin
```

Avertissement : Ne pas oublier d'avoir déclaré au préalable sur les machines cibles le dépôt de binaires associé aux *griffins*.

4.2.5.3 Rétention liée aux logback

La solution logicielle *VITAM* utilise logback pour la rotation des log, ainsi que leur rétention.

Il est possible d'appliquer un paramétrage spécifique pour chaque composant VITAM.

Éditer le fichier `deployment/environments/group_vars/all/vitam_vars.yml` (et `extra_vars.yml`, dans le cas des extra) et appliquer le paramétrage dans le bloc `logback_total_size_cap` de chaque composant sur lequel appliquer la modification de paramétrage. Pour chaque **APPENDER**, la valeur associée doit être exprimée en taille et unité (exemple : 14GB ; représente 14 gigabytes).

Note : des *appenders* supplémentaires existent pour le composant storage-engine (`appender offersync`) et `offer` (`offer_tape` et `offer_tape_backup`).

4.2.5.3.1 Cas des accesslog

Il est également possible d'appliquer un paramétrage différent par composant VITAM sur le logback *access*.

Éditer le fichier `deployment/environments/group_vars/all/vitam_vars.yml` (et `extra_vars.yml`, dans le cas des extra) et appliquer le paramétrage dans les directives `access_retention_days` et `access_total_size_GB` de chaque composant sur lequel appliquer la modification de paramétrage.

4.2.5.4 Paramétrage de l'antivirus (ingest-external)

L'antivirus utilisé par ingest-external est modifiable (par défaut, ClamAV) ; pour cela :

- Éditer la variable `vitam.ingestexternal.antivirus` dans le fichier `deployment/environments/group_vars/all/vitam_vars.yml` pour indiquer le nom de l'antivirus à utiliser.
- Créer un script shell (dont l'extension doit être `.sh`) sous `environments/antivirus/` (norme : `scan-<vitam.ingestexternal.antivirus>.sh`) ; prendre comme modèle le fichier `scan-clamav.sh`. Ce script shell doit respecter le contrat suivant :
 - Argument : chemin absolu du fichier à analyser
 - **Sémantique des codes de retour**
 - 0 : Analyse OK - pas de virus
 - 1 : Analyse OK - virus trouvé et corrigé
 - 2 : Analyse OK - virus trouvé mais non corrigé
 - 3 : Analyse NOK
 - **Contenu à écrire dans stdout / stderr**
 - stdout : Nom des virus trouvés, un par ligne ; Si échec (code 3) : raison de l'échec
 - stderr : Log « brut » de l'antivirus

Prudence : En cas de remplacement de clamAV par un autre antivirus, l'installation de celui-ci devient dès lors un prérequis de l'installation et le script doit être testé.

Avertissement : Sur plate-forme Debian, ClamAV est installé sans base de données. Pour que l'antivirus soit fonctionnel, il est nécessaire, durant l'installation, de le télécharger ; il est donc nécessaire de renseigner dans l'inventaire la directive `http_proxy_environnement`.

4.2.5.5 Paramétrage des certificats externes (*-externe)

Se reporter au chapitre dédié à la gestion des certificats : *Gestion des certificats* (page 57)

4.2.5.6 Placer « hors Vitam » le composant ihm-demo

Sous `deployment/environments/host_vars`, créer ou éditer un fichier nommé par le nom de machine qui héberge le composant ihm-demo et ajouter le contenu ci-dessous :

```
consul_disabled : true
```

Il faut également modifier le fichier `deployment/environments/group_vars/all/vitam_vars.yml` en remplaçant :

- dans le bloc `accessexternal`, la directive `host: "access-external.service.{{ consul_domain }}"` par `host: "<adresse IP de access-external>"` (l'adresse IP peut être une *FIP*)
- dans le bloc `ingestexternal`, la directive `host: "ingest-external.service.{{ consul_domain }}"` par `host: "<adresse IP de ingest-external>"` (l'adresse IP peut être une *FIP*)

A l'issue, le déploiement n'installera pas l'agent Consul. Le composant ihm-demo appellera, alors, par l'adresse *IP* de service les composants « access-external » et « ingest-external ».

Il est également fortement recommandé de positionner la valeur de la directive `vitam.ihm_demo.metrics_enabled` à `false` dans le fichier `deployment/environments/group_vars/all/vitam_vars.yml`, afin que ce composant ne tente pas d'envoyer des données sur « elasticsearch-log ».

4.2.5.7 Paramétrer le `secure_cookie` pour ihm-demo

Le composant ihm-demo (ainsi qu'ihm-recette) dispose d'une option supplémentaire, par rapport aux autres composants VITAM, dans le fichier `deployment/environments/group_vars/all/vitam_vars.yml` : le `secure_cookie` qui permet de renforcer ces deux *IHM* contre certaines attaques assez répandues comme les CSRF (Cross-Site Request Forgery).

Il faut savoir que si cette variable est à *true* (valeur par défaut), le client doit obligatoirement se connecter en https sur l'*IHM*, et ce même si un reverse proxy se trouve entre le serveur web et le client.

Cela peut donc obliger le reverse proxy frontal de la chaîne d'accès à écouter en https.

4.2.5.8 Paramétrage de la centralisation des logs VITAM

2 cas sont possibles :

- Utiliser le sous-système de gestion des logs fourni par la solution logicielle *VITAM* ;
- Utiliser un *SIEM* tiers.

4.2.5.8.1 Gestion par VITAM

Pour une gestion des logs par *VITAM*, il est nécessaire de déclarer les serveurs ad-hoc dans le fichier d'inventaire pour les 3 gro

- `hosts_logstash`
- `hosts_kibana_log`
- `hosts_elasticsearch_log`

4.2.5.8.2 Redirection des logs sur un SIEM tiers

En configuration par défaut, les logs VITAM sont tout d'abord routés vers un serveur rsyslog installé sur chaque machine. Il est possible d'en modifier le routage, qui par défaut redirige vers le serveur logstash, via le protocole syslog en TCP.

Pour cela, il est nécessaire de placer un fichier de configuration dédié dans le dossier `/etc/rsyslog.d/` ; ce fichier sera automatiquement pris en compte par rsyslog. Pour la syntaxe de ce fichier de configuration rsyslog, se référer à la [documentation rsyslog](#)¹⁵.

Astuce : Pour cela, il peut être utile de s'inspirer du fichier de référence *VITAM* `deployment/ansible-vitam/roles/rsyslog/templates/vitam_transport.conf.j2` (attention, il s'agit d'un fichier template ansible, non directement convertible en fichier de configuration sans en ôter les directives jinja2).

4.2.5.9 Passage des identifiants des référentiels en mode *esclave*

La génération des identifiants des référentiels est géré par *VITAM* lorsqu'il fonctionne en mode maître.

Par exemple :

- Préfixé par `PR-` pour les profils
- Préfixé par `IC-` pour les contrats d'entrée

<http://www.rsyslog.com/doc/v7-stable/>

- Préfixé par AC- pour les contrats d'accès

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 :

Tableau 1: Description des identifiants de référentiels

Nom du référentiel	Description
INGEST_CONTRACT	contrats d'entrée
ACCESS_CONTRACT	contrats d'accès
PROFILE	profils <i>SEDA</i>
SECURITY_PROFILE	profils de sécurité (utile seulement sur le tenant d'administration)
CONTEXT	contextes applicatifs (utile seulement sur le tenant d'administration)
ARCHIVEUNITPROFILE	profils d'unités archivistiques

Si vous souhaitez gérer vous-même les identifiants sur un service référentiel, il faut qu'il soit en mode esclave.

Par défaut tous les services référentiels de Vitam fonctionnent en mode maître. Pour désactiver le mode maître de *VITAM*, il faut modifier le fichier `ansible/deployment/environments/group_vars/all/vitam_vars.yml` dans les sections `vitam_tenants_usage_external` (pour gérer, par tenant, les collections en mode esclave).

4.2.5.10 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 `deployment/environments/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 lesquels 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]
```

4.2.5.11 Durées minimales permettant de contrôler les valeurs saisies

Afin de se prémunir contre une alimentation du référentiel des règles de gestion avec des durées trop courtes susceptibles de déclencher des actions indésirables sur la plate-forme (ex. éliminations) – que cette tentative soit intentionnelle ou non –, la solution logicielle *VITAM* vérifie que l'association de la durée et de l'unité de mesure saisies pour chaque champ est supérieure ou égale à une durée minimale définie lors du paramétrage de la plate-forme, dans un fichier de configuration.

Pour mettre en place le comportement attendu par le métier, il faut modifier le contenu de la directive `vitam_tenant_rule_duration` dans le fichier ansible `deployment/environments/group_vars/all/vitam_vars.yml`.

Exemple :

```
vitam_tenant_rule_duration:
- name: 2 # applied tenant
  rules:
    - AppraisalRule : "1 year" # rule name : rule value
- name: 3
  rules:
    AppraisalRule : "5 year"
    StorageRule : "5 year"
    ReuseRule : "2 year"
```

Par *tenant*, les directives possibles sont :

Tableau 2: Description des règles

Règle	Valeur par défaut
AppraisalRule	
DisseminationRule	
StorageRule	
ReuseRule	
AccessRule	0 year
ClassificationRule	

Les valeurs associées sont une durée au format <nombre> <unité en anglais, au singulier>

Exemples :

6 month 1 year 5 year

Voir aussi :

Pour plus de détails, se rapporter à la documentation métier « Règles de gestion ».

4.2.5.12 Fichiers complémentaires

A titre informatif, le positionnement des variables ainsi que des dérivations des déclarations de variables sont effectuées dans les fichiers suivants :

- `deployment/environments/group_vars/all/vitam_vars.yml`, comme suit :

```
1 ---
2 ### global ###
3
4 # Disable epel or Debian backports repositories install
5 disable_internet_repositories_install: false
6
7 # TODO MAYBE : permettre la surcharge avec une syntaxe du genre vitamopts.folder_
8 ↪root | default(vitam_default.folder_root) dans les templates ?
9 droid_filename: "DROID_SignatureFile_V97.xml"
10 droid_container_filename: "container-signature-20201001.xml"
11
12 # The global defaults parameters for vitam & vitam components
13 vitam_defaults:
```

(suite sur la page suivante)

(suite de la page précédente)

```

13  folder:
14      root_path: /vitam
15      folder_permission: "0750"
16      conf_permission: "0640"
17      folder_upload_permission: "0770"
18      script_permission: "0750"
19  users:
20      vitam: "vitam"
21      vitamdb: "vitamdb"
22      group: "vitam"
23  services:
24      # Default log level for vitam components: logback values (TRACE, DEBUG,
↪INFO, WARN, ERROR, OFF)
25      log_level: WARN
26      start_timeout: 300
27      stop_timeout: 3600
28      port_service_timeout: 86400
29      api_call_timeout: 120
30      api_long_call_timeout: 300
31      status_retries_number: 60
32      status_retries_delay: 5
33  ### Trust X-SSL-CLIENT-CERT header for external api auth ? (true | false)
34      vitam_ssl_user_header: true
35  ### Force chunk mode : set true if chunk header should be checked
36      vitam_force_chunk_mode: false
37      # syslog_facility
38      syslog_facility: local0
39
40  #####
41  ### Default Components parameters
42  ### Uncomment them if you want to update the default value applied on all
↪components
43
44  ### Ontology cache settings (max entries in cache & retention timeout in seconds)
45      # ontologyCacheMaxEntries: 100
46      # ontologyCacheTimeoutInSeconds: 300
47  ### Elasticsearch scroll timeout in milliseconds settings
48      # elasticSearchScrollTimeoutInMilliseconds: 300000
49
50  ### The following values can be overwritten for each components in vitam:
↪parameters.
51      jvm_log: false
52      performance_logger: false
53
54      # consul_business_check: 10 # value in seconds
55      # consul_admin_check: 10 # value in seconds
56
57      # metricslevel: DEBUG
58      # metricsinterval: 3
59      # metricsunit: MINUTES
60
61      # access_retention_days: 30 # Number of days for file retention
62      # access_total_size_cap: "10GB" # total acceptable size
63      # logback_max_file_size: "10MB"
64      # logback_total_size_cap:
65      #     file:
66      #         history_days: 30

```

(suite sur la page suivante)

(suite de la page précédente)

```

67      #      totalsize: "5GB"
68      #      security:
69      #      history_days: 30
70      #      totalsize: "5GB"
71
72      ### Logs configuration for reconstruction services (INFO or DEBUG for active_
73      ↪logs).
74      ### Logs will be present only on secondary site.
75      ### Available for the following components: logbook, metadata & functional-
76      ↪administration.
77      reconstruction:
78          log_level: INFO
79
80      # Used in ingest, unitary update, mass-update
81      classificationList: ["Non protégé", "Secret Défense", "Confidentiel Défense"]
82      # Used in ingest, unitary update, mass-update
83      classificationLevelOptional: true
84      # Packages install retries
85      packages_install_retries_number: 1
86      packages_install_retries_delay: 10
87
88      # Request time check settings. Do NOT update except if required by Vitam support
89      # Max acceptable time desynchronization between machines (in seconds).
90      acceptableRequestTime: 10
91      # Critical time desynchronization between machines (in seconds).
92      criticalRequestTime: 60
93      # Request time alert throttling Delay (in seconds)
94      requestTimeAlertThrottlingDelay: 60
95
96      vitam_timers:
97      # /\ IMPORTANT :
98      # Please ensure timer execution is spread so that not all timers run concurrently_
99      ↪(eg. *:05:00, *:35:00, *:50:00..),
100      # Special care for heavy-load timers that run on same machines or use same_
101      ↪resources (eg. vitam-traceability-*).
102      #
103      # systemd nomenclature
104      #   minutely → *-*- *:*:00
105      #   hourly → *-*- *:00:00
106      #   daily → *-*- 00:00:00
107      #   monthly → *-*-01 00:00:00
108      #   weekly → Mon *-*- 00:00:00
109      #   yearly → *-01-01 00:00:00
110      #   quarterly → *-01,04,07,10-01 00:00:00
111      #   semiannually → *-01,07-01 00:00:00
112      logbook: # all have to run on only one machine
113      # Sécurisation des journaux des opérations
114      - name: vitam-traceability-operations
115      frequency: "*-*-* *:05:00" # every hour
116      # Sécurisation des journaux du cycle de vie des groupes d'objets
117      - name: vitam-traceability-lfc-objectgroup
118      frequency: "*-*-* *:15:00" # every hour
119      # Sécurisation des journaux du cycle de vie des unités archivistes
120      - name: vitam-traceability-lfc-unit
121      frequency: "*-*-* *:35:00" # every hour
122      # Audit de traçabilité
123      - name: vitam-traceability-audit

```

(suite sur la page suivante)

(suite de la page précédente)

```

120     frequency: "*--* 00:55:00"
121     # Reconstruction (uniquement sur site secondaire)
122     - name: vitam-logbook-reconstruction
123       frequency: "*--* *:0/5:00"
124   storage:
125     # Sauvegarde des journaux d'accès
126     - name: vitam-storage-accesslog-backup
127       frequency: "*--* 0/4:10:00" # every 4 hours
128     # Sauvegarde des journaux des écritures
129     - name: vitam-storage-log-backup
130       frequency: "*--* 0/4:15:00" # every 4 hours
131     # Sécurisation du journal des écritures
132     - name: vitam-storage-log-traceability
133       frequency: "*--* 0/4:40:00" # every 4 hours
134   functional_administration:
135     - name: vitam-create-accession-register-symbolic
136       frequency: "*--* 00:50:00"
137     - name: vitam-functional-administration-accession-register-reconstruction
138       frequency: "*--* *:0/5:00"
139     - name: vitam-rule-management-audit
140       frequency: "*--* *:40:00"
141     - name: vitam-functional-administration-reconstruction
142       frequency: "*--* *:0/5:00"
143   metadata:
144     - name: vitam-metadata-store-graph
145       frequency: "*--* *:10/30:00"
146     - name: vitam-metadata-reconstruction
147       frequency: "*--* *:0/5:00"
148     - name: vitam-metadata-computed-inherited-rules
149       frequency: "*--* 02:30:00"
150     - name: vitam-metadata-purge-dip
151       frequency: "*--* 02:20:00"
152     - name: vitam-metadata-purge-transfers-SIP
153       frequency: "*--* 02:25:00"
154     - name: vitam-metadata-audit-mongodb-es
155       frequency: "2020-01-01 00:00:00"
156   offer:
157     # Compaction offer logs
158     - name: vitam-offer-log-compaction
159       frequency: "*--* *:40:00" # every hour
160
161   ### consul ###
162   # WARNING: consul_domain should be a supported domain name for your organization
163   # You will have to generate server certificates with the same domain,
164   # ↳ name and the service subdomain name
165   # Example: consul_domain=vitam means you will have to generate some
166   # ↳ certificates with .service.vitam domain
167   # access-external.service.vitam, ingest-external.service.vitam,
168   # ↳ ...
169   consul_domain: consul
170   consul_folder_conf: "{{ vitam_defaults.folder.root_path }}/conf/consul"
171
172   # Workspace should be useless but storage have a dependency to it...
173   # elastic-kibana-interceptor is present as kibana is present, if kibana-data &
174   # ↳ interceptor are not needed in the secondary site, just do not add them in the
175   # ↳ hosts file
176   vitam_secondary_site_components: [ "logbook" , "metadata" , "functional-
177     ↳ administration" , "storage" , "storageofferdefault" , "offer" , "
178     ↳ log" , "elasticsearch-data" , "logstash" , "kibana" , "mongoc" , "mongod" ,
179     ↳ "mongos" , "elastic-kibana-interceptor" , "consul" ]

```

(suite de la page précédente)

```

172
173 # containers list
174 containers_list: ['units', 'objects', 'objectgroups', 'logbooks', 'reports',
↳ 'manifests', 'profiles', 'storagelog', 'storageaccesslog', 'storagetraceability
↳ ', 'rules', 'dip', 'agencies', 'backup', 'backupoperations', 'unitgraph',
↳ 'objectgroupgraph', 'distributionreports', 'accessionregistersdetail',
↳ 'accessionregisterssymbolic', 'tmp', 'archivaltransferreply']
175
176 # Vitams griffins required to launch preservation scenario
177 # Example:
178 # vitam_griffins: ["vitam-imagemagick-griffin", "vitam-libreoffice-griffin",
↳ "vitam-jhove-griffin", "vitam-odfvalidator-griffin", "vitam-siegfried-griffin",
↳ "vitam-tesseract-griffin", "vitam-verapdf-griffin", "vitam-ffmpeg-griffin"]
179 vitam_griffins: []
180
181 ### Composants Vitam ###
182 vitam:
183 ### All available parameters for each components are described in the vitam_
↳ defaults variable
184
185 ### Example
186 # component:
187 #     logback_rolling_policy: true
188 ## Force the log level for this component. Available logback values are (TRACE,
↳ DEBUG, INFO, WARN, ERROR, OFF)
189 ## If this var is not set, the default one will be used (vitam_defaults.
↳ services.log_level)
190 #     log_level: "DEBUG"
191
192 accessexternal:
193 # Component name: do not modify
194 vitam_component: access-external
195 # DNS record for the service:
196 # Modify if ihm-demo is not using consul (typical production deployment)
197 host: "access-external.service.{{ consul_domain }}"
198 port_admin: 28102
199 port_service: 8444
200 baseuri: "access-external"
201 https_enabled: true
202 # Use platform secret for this component ? : do not modify
203 secret_platform: "false"
204 accessinternal:
205 vitam_component: access-internal
206 host: "access-internal.service.{{ consul_domain }}"
207 port_service: 8101
208 port_admin: 28101
209 baseuri: "access-internal"
210 https_enabled: false
211 secret_platform: "true"
212 functional_administration:
213 vitam_component: functional-administration
214 host: "functional-administration.service.{{ consul_domain }}"
215 port_service: 8004
216 port_admin: 18004
217 baseuri: "adminmanagement"
218 https_enabled: false
219 secret_platform: "true"

```

(suite sur la page suivante)

(suite de la page précédente)

```

220     cluster_name: "{{ elasticsearch.data.cluster_name }}"
221     # Number of AccessionRegisterSymbolic creation threads that can be run in_
↳parallel.
222     accessionRegisterSymbolicThreadPoolSize: 16
223     # Number of rule audit threads that can be run in parallel.
224     ruleAuditThreadPoolSize: 16
225     elasticsearchinterceptor:
226     vitam_component: elastic-kibana-interceptor
227     host: "elastic-kibana-interceptor.service.{{ consul_domain }}"
228     port_service: 8014
229     port_admin: 18014
230     baseuri: ""
231     https_enabled: false
232     secret_platform: "false"
233     cluster_name: "{{ elasticsearch.data.cluster_name }}"
234     batchreport:
235     vitam_component: batch-report
236     host: "batch-report.service.{{ consul_domain }}"
237     port_service: 8015
238     port_admin: 18015
239     baseuri: "batchreport"
240     https_enabled: false
241     secret_platform: "false"
242     ingestexternal:
243     vitam_component: ingest-external
244     # DNS record for the service:
245     # Modify if ihm-demo is not using consul (typical production deployment)
246     host: "ingest-external.service.{{ consul_domain }}"
247     port_admin: 28001
248     port_service: 8443
249     baseuri: "ingest-external"
250     https_enabled: true
251     secret_platform: "false"
252     antivirus: "clamav"
253     # uncomment if huge files need to be analyzed in more than 60s (default_
↳behavior)
254     #scantimeout: 60000 # value in milliseconds
255     # Directory where files should be placed for local ingest
256     upload_dir: "/vitam/data/ingest-external/upload"
257     # Directory where successful ingested files will be moved to
258     success_dir: "/vitam/data/ingest-external/upload/success"
259     # Directory where failed ingested files will be moved to
260     fail_dir: "/vitam/data/ingest-external/upload/failure"
261     # Action done to file after local ingest (see below for further_
↳information)
262     upload_final_action: "MOVE"
263     # upload_final_action can be set to three different values (lower or_
↳upper case does not matter)
264     # MOVE : After upload, the local file will be moved to either success_
↳dir or fail_dir depending on the status of the ingest towards ingest-internal
265     # DELETE : After upload, the local file will be deleted if the upload_
↳succeeded
266     # NONE : After upload, nothing will be done to the local file (default_
↳option set if the value entered for upload_final_action does not exist)
267     ingestinternal:
268     vitam_component: ingest-internal
269     host: "ingest-internal.service.{{ consul_domain }}"

```

(suite sur la page suivante)

(suite de la page précédente)

```

270     port_service: 8100
271     port_admin: 28100
272     baseuri: "ingest"
273     https_enabled: false
274     secret_platform: "true"
275     ihm_demo:
276       vitam_component: ihm-demo
277       host: "ihm-demo.service.{{ consul_domain }}"
278       port_service: 8446
279       port_admin: 28002
280       baseurl: "/ihm-demo"
281       static_content: "{{ vitam_defaults.folder.root_path }}/app/ihm-demo/v2"
282       baseuri: "ihm-demo"
283       https_enabled: true
284       # metrics_enabled: false # Set to false if ihm_demo component is outside
↪the Vitam area (default: true)
285       secret_platform: "false"
286       # User session timeout in milliseconds (for shiro)
287       session_timeout: 1800000
288       secure_cookie: true
289       # Specify here the realms you want to use for authentication in ihm-demo
290       # You can set multiple realms, one per line
291       # With multiple realms, the user will be able to choose between the
↪allowed realms
292       # Example: authentication_realms:
293       #           - x509Realm
294       #           - ldapRealm
295       # Authorized values:
296       # x509Realm: certificate
297       # iniRealm: ini file
298       # ldapRealm: ldap
299       authentication_realms:
300         # - x509Realm
301         - iniRealm
302         # - ldapRealm
303       allowedMediaTypes:
304         - type: "application"
305           subtype: "pdf"
306         - type: "text"
307           subtype: "plain"
308         - type: "image"
309           subtype: "jpeg"
310         - type: "image"
311           subtype: "tiff"
312     logbook:
313       vitam_component: logbook
314       host: "logbook.service.{{ consul_domain }}"
315       port_service: 9002
316       port_admin: 29002
317       baseuri: "logbook"
318       https_enabled: false
319       secret_platform: "true"
320       cluster_name: "{{ elasticsearch.data.cluster_name }}"
321       # Temporization delay (in seconds) for recent logbook operation events.
322       # Set it to a reasonable delay to cover max clock difference across
↪servers + VM/GC pauses
323       operationTraceabilityTemporizationDelay: 300

```

(suite sur la page suivante)

(suite de la page précédente)

```

324     # Max delay between 2 logbook operation traceability operations.
325     # A new logbook operation traceability is generated after this delay,
↪even if tenant has no
326     # new logbook operations to secure
327     # Unit can be in DAYS, HOURS, MINUTES, SECONDS
328     # Hint: Set it to 690 MINUTES (11 hours and 30 minutes) to force new
↪traceability after +/- 12 hours (supposing
329     # logbook operation traceability timer run every hour +/- some clock
↪delays)
330     operationTraceabilityMaxRenewalDelay: 690
331     operationTraceabilityMaxRenewalDelayUnit: MINUTES
332     # Number of logbook operations that can be run in parallel.
333     operationTraceabilityThreadPoolSize: 16
334     # Temporization delay (in seconds) for recent logbook lifecycle events.
335     # Set it to a reasonable delay to cover max clock difference across
↪servers + VM/GC pauses
336     lifecycleTraceabilityTemporizationDelay: 300
337     # Max delay between 2 lifecycle traceability operations.
338     # A new unit/objectgroup lifecycle traceability is generated after this
↪delay, even if tenant has no
339     # new unit/objectgroups to secure
340     # Unit can be in DAYS, HOURS, MINUTES, SECONDS
341     # Hint: Set it to 690 MINUTES (11 hours and 30 minutes) to force new
↪traceability after +/- 12 hours (supposing
342     # LFC traceability timers run every hour +/- some clock delays)
343     lifecycleTraceabilityMaxRenewalDelay: 690
344     lifecycleTraceabilityMaxRenewalDelayUnit: MINUTES
345     # Max entries selected per (Unit or Object Group) LFC traceability
↪operation
346     lifecycleTraceabilityMaxEntries: 100000
347     metadata:
348         vitam_component: metadata
349         host: "metadata.service.{{ consul_domain }}"
350         port_service: 8200
351         port_admin: 28200
352         baseuri: "metadata"
353         https_enabled: false
354         secret_platform: "true"
355         cluster_name: "{{ elasticsearch.data.cluster_name }}"
356         # Archive Unit Profile cache settings (max entries in cache & retention
↪timeout in seconds)
357         archiveUnitProfileCacheMaxEntries: 100
358         archiveUnitProfileCacheTimeoutInSeconds: 300
359         # Schema validator cache settings (max entries in cache & retention
↪timeout in seconds)
360         schemaValidatorCacheMaxEntries: 100
361         schemaValidatorCacheTimeoutInSeconds: 300
362         # DIP cleanup delay (in minutes)
363         dipTimeToLiveInMinutes: 10080 # 7 days
364         transfersSIPTimeToLiveInMinutes: 10080 # 7 days
365         elasticsearch_mapping_dir: "{{ vitam_defaults.folder.root_path }}/conf/
↪metadata/mapping" # Directory of elasticsearch metadata mapping
366         #### Audit data consistency MongoDB-ES ####
367         isDataConsistencyAuditRunnable: false
368         dataConsistencyAuditOplogMaxSize: 100
369     processing:
370         vitam_component: processing

```

(suite sur la page suivante)

(suite de la page précédente)

```

371     host: "processing.service.{{ consul_domain }}"
372     port_service: 8203
373     port_admin: 28203
374     baseuri: "processing"
375     https_enabled: false
376     secret_platform: "true"
377     maxDistributionInMemoryBufferSize: 100000
378     maxDistributionOnDiskBufferSize: 100000000
379     security_internal:
380       vitam_component: security-internal
381       host: "security-internal.service.{{ consul_domain }}"
382       port_service: 8005
383       port_admin: 28005
384       baseuri: "security-internal"
385       https_enabled: false
386       secret_platform: "true"
387     storageengine:
388       vitam_component: storage
389       host: "storage.service.{{ consul_domain }}"
390       port_service: 9102
391       port_admin: 29102
392       baseuri: "storage"
393       https_enabled: false
394       secret_platform: "true"
395       storageTraceabilityOverlapDelay: 300
396       restoreBulkSize: 1000
397       # Storage write/access log backup max thread pool size
398       storageLogBackupThreadPoolSize: 16
399       # Storage write log traceability thread pool size
400       storageLogTraceabilityThreadPoolSize: 16
401       # Offer synchronization batch size & thread pool size
402       offerSynchronizationBulkSize: 1000
403       # Retries attempts
404       offerSyncNumberOfRetries: 3
405       offerSyncFirstAttemptWaitingTime: 15
406       offerSyncWaitingTime: 30
407       offerSyncThreadPoolSize: 32
408       logback_total_size_cap:
409         offersync:
410           history_days: 30
411           totalsize: "5GB"
412         offerdiff:
413           history_days: 30
414           totalsize: "5GB"
415       # unit time per kB (in ms) used while calculating the timeout of an http_
↪request between storage and offer (if the calculated result is less than 60s,
↪this time is used)
416       timeoutMsPerKB: 100
417     storageofferdefault:
418       vitam_component: "offer"
419       port_service: 9900
420       port_admin: 29900
421       baseuri: "offer"
422       https_enabled: false
423       secret_platform: "true"
424       logback_total_size_cap:
425         offer_tape:

```

(suite sur la page suivante)

(suite de la page précédente)

```

426     history_days: 30
427     totalsize: "5GB"
428     offer_tape_backup:
429         history_days: 30
430         totalsize: "5GB"
431     worker:
432         vitam_component: worker
433         host: "worker.service.{{ consul_domain }}"
434         port_service: 9104
435         port_admin: 29104
436         baseuri: "worker"
437         https_enabled: false
438         secret_platform: "true"
439         api_output_index_tenants: [0,1,2,3,4,5,6,7,8,9]
440         rules_index_tenants: [0,1,2,3,4,5,6,7,8,9]
441         # Archive Unit Profile cache settings (max entries in cache & retention
↪timeout in seconds)
442         archiveUnitProfileCacheMaxEntries: 100
443         archiveUnitProfileCacheTimeoutInSeconds: 300
444         # Schema validator cache settings (max entries in cache & retention
↪timeout in seconds)
445         schemaValidatorCacheMaxEntries: 100
446         schemaValidatorCacheTimeoutInSeconds: 300
447         # Batch size for bulk atomic update
448         queriesThreshold: 100000
449         # Bulk atomic update batch size
450         bulkAtomicUpdateBatchSize: 100
451         # Max threads that can be run in concurrently is thread pool for bulk
↪atomic update
452         bulkAtomicUpdateThreadPoolSize: 8
453         # Number of jobs that can be queued in memory before blocking for bulk
↪atomic update
454         bulkAtomicUpdateThreadPoolQueueSize: 16
455     workspace:
456         vitam_component: workspace
457         host: "workspace.service.{{ consul_domain }}"
458         port_service: 8201
459         port_admin: 28201
460         baseuri: "workspace"
461         https_enabled: false
462         secret_platform: "true"
463
464     # for functional-administration, manage master/slave tenant configuration
465     vitam_tenants_usage_external:
466     - name: 0
467       identifiers:
468       - INGEST_CONTRACT
469       - ACCESS_CONTRACT
470       - MANAGEMENT_CONTRACT
471       - ARCHIVE_UNIT_PROFILE
472     - name: 1
473       identifiers:
474       - INGEST_CONTRACT
475       - ACCESS_CONTRACT
476       - MANAGEMENT_CONTRACT
477       - PROFILE
478       - SECURITY_PROFILE

```

(suite sur la page suivante)

(suite de la page précédente)

```

479 - CONTEXT
480
481 vitam_tenant_rule_duration:
482 - name: 2 # applied tenant
483   rules:
484     - AppraisalRule : "1 year" # rule name : rule value
485
486 # If you want to deploy vitam in a single VM, add the vm name in this array
487 single_vm_hostnames: ['localhost']

```

Note : Cas du composant ingest-external. Les directives `upload_dir`, `success_dir`, `fail_dir` et `upload_final_action` permettent de prendre en charge (ingest) des fichiers déposés dans `upload_dir` et appliquer une règle `upload_final_action` à l'issue du traitement (NONE, DELETE ou MOVE dans `success_dir` ou `fail_dir` selon le cas). Se référer au *DEX* pour de plus amples détails. Se référer au manuel de développement pour plus de détails sur l'appel à ce cas.

Avertissement : Selon les informations apportées par le métier, redéfinir les valeurs associées dans les directives `classificationList` et `classificationLevelOptional`. Cela permet de définir quels niveaux de protection du secret de la défense nationale, supporte l'instance. Attention : une instance de niveau supérieur doit toujours supporter les niveaux inférieurs.

- `deployment/environments/group_vars/all/cots_vars.yml`, comme suit :

```

1 ---
2
3 consul:
4   retry_interval: 10 # in seconds
5   check_interval: 10 # in seconds
6   check_timeout: 5 # in seconds
7   log_level: WARN # Available log_level are: TRACE, DEBUG, INFO, WARN or ERR
8   network: "ip_admin" # Which network to use for consul communications ? ip_
  ↳ admin or ip_service ?
9
10 consul_remote_sites:
11   # wan contains the wan addresses of the consul server instances of the_
  ↳ external vitam sites
12   # Exemple, if our local dc is dc1, we will need to set dc2 & dc3 wan conf:
13   # - dc2:
14   #   wan: ["10.10.10.10", "1.1.1.1"]
15   # - dc3:
16   #   wan: ["10.10.10.11", "1.1.1.1"]
17 # Please uncomment and fill values if you want to connect VITAM to external SIEM
18 # external_siem:
19 #   host:
20 #   port:
21
22 elasticsearch:
23   log:
24     host: "elasticsearch-log.service.{{ consul_domain }}"
25     port_http: "9201"
26     groupe: "log"
27     baseuri: "elasticsearch-log"

```

(suite sur la page suivante)

(suite de la page précédente)

```

28     cluster_name: "elasticsearch-log"
29     consul_check_http: 10 # in seconds
30     consul_check_tcp: 10 # in seconds
31     action_log_level: error
32     https_enabled: false
33     indices fielddata_cache_size: '30%' # related to https://www.elastic.co/
    ↳guide/en/elasticsearch/reference/7.6/modules-fielddata.html
34     indices_breaker_fielddata_limit: '40%' # related to https://www.elastic.
    ↳co/guide/en/elasticsearch/reference/7.6/circuit-breaker.html#fielddata-circuit-
    ↳breaker
35     dynamic_timeout: 30s
36     # default index template
37     index_templates:
38         default:
39             shards: 1
40             replica: 1
41         packetbeat:
42             shards: 5
43     log_appenders:
44         root:
45             log_level: "info"
46         rolling:
47             max_log_file_size: "100MB"
48             max_total_log_size: "5GB"
49             max_files: "50"
50         deprecation_rolling:
51             max_log_file_size: "100MB"
52             max_total_log_size: "1GB"
53             max_files: "10"
54             log_level: "warn"
55         index_search_slowlog_rolling:
56             max_log_file_size: "100MB"
57             max_total_log_size: "1GB"
58             max_files: "10"
59             log_level: "warn"
60         index_indexing_slowlog_rolling:
61             max_log_file_size: "100MB"
62             max_total_log_size: "1GB"
63             max_files: "10"
64             log_level: "warn"
65     # By default, is commented. Should be uncommented if ansible computes
    ↳badly vCPUs number ; values are associated vCPUs numbers ; please adapt to
    ↳your configuration
66     # thread_pool:
67     #     index:
68     #         size: 2
69     #     get:
70     #         size: 2
71     #     search:
72     #         size: 2
73     #     write:
74     #         size: 2
75     #     warmer:
76     #         max: 2
77     data:
78         host: "elasticsearch-data.service.{{ consul_domain }}"
79         # default is 0.1 (10%) and should be quite enough in most cases

```

(suite sur la page suivante)

(suite de la page précédente)

```

80     #index_buffer_size_ratio: "0.15"
81     port_http: "9200"
82     groupe: "data"
83     baseuri: "elasticsearch-data"
84     cluster_name: "elasticsearch-data"
85     consul_check_http: 10 # in seconds
86     consul_check_tcp: 10 # in seconds
87     action_log_level: debug
88     https_enabled: false
89     indices fielddata_cache_size: '30%' # related to https://www.elastic.co/
    ↪guide/en/elasticsearch/reference/6.5/modules-fielddata.html
90     indices_breaker_fielddata_limit: '40%' # related to https://www.elastic.
    ↪co/guide/en/elasticsearch/reference/6.5/circuit-breaker.html#fielddata-circuit-
    ↪breaker
91     dynamic_timeout: 30s
92     # default index template
93     index_templates:
94         default:
95             shards: 1
96             replica: 2
97     log_appenders:
98         root:
99             log_level: "info"
100     rolling:
101         max_log_file_size: "100MB"
102         max_total_log_size: "5GB"
103         max_files: "50"
104     deprecation_rolling:
105         max_log_file_size: "100MB"
106         max_total_log_size: "5GB"
107         max_files: "50"
108         log_level: "warn"
109     index_search_slowlog_rolling:
110         max_log_file_size: "100MB"
111         max_total_log_size: "5GB"
112         max_files: "50"
113         log_level: "warn"
114     index_indexing_slowlog_rolling:
115         max_log_file_size: "100MB"
116         max_total_log_size: "5GB"
117         max_files: "50"
118         log_level: "warn"
119     # By default, is commented. Should be uncommented if ansible computes
    ↪badly vCPUs number ; values are associated vCPUs numbers ; please adapt to
    ↪your configuration
120     # thread_pool:
121     #     index:
122     #         size: 2
123     #     get:
124     #         size: 2
125     #     search:
126     #         size: 2
127     #     write:
128     #         size: 2
129     #     warmer:
130     #         max: 2
131

```

(suite sur la page suivante)

(suite de la page précédente)

```

132 mongodb:
133     mongos_port: 27017
134     mongoc_port: 27018
135     mongod_port: 27019
136     mongo_authentication: "true"
137     host: "mongos.service.{{ consul_domain }}"
138     check_consul: 10 # in seconds
139     drop_info_log: false # Drop mongo (I)nformational log, for Verbosity Level of
    ↪ 0
140     # logs configuration
141     logrotate: enabled # or disabled
142     history_days: 30 # How many days to store logs if logrotate is set to 'enabled
    ↪ '
143
144 logstash:
145     host: "logstash.service.{{ consul_domain }}"
146     user: logstash
147     port: 10514
148     rest_port: 20514
149     check_consul: 10 # in seconds
150     # logstash xms & xmx in Megabytes
151     # jvm_xms: 2048
152     # jvm_xmx: 2048
153     # workers_number: 4
154     log_appenders:
155         rolling:
156             max_log_file_size: "100MB"
157             max_total_log_size: "5GB"
158         json_rolling:
159             max_log_file_size: "100MB"
160             max_total_log_size: "5GB"
161
162     # Prometheus params
163     prometheus:
164         metrics_path: /admin/v1/metrics
165         check_consul: 10 # in seconds
166         prometheus_config_file_target_directory: # Set path where "prometheus.yml"
    ↪ file will be generated. Example: /tmp/
167         server:
168             enabled: false
169             port: 19090
170         node_exporter:
171             enabled: true
172             port: 19100
173             metrics_path: /metrics
174         alertmanager:
175             enabled: false
176             api_port: 19093
177             cluster_port: 19094
178     grafana:
179         enabled: false
180         check_consul: 10 # in seconds
181         http_port: 13000
182
183     # Curator units: days
184     curator:
185         log:

```

(suite sur la page suivante)

(suite de la page précédente)

```

186     metrics:
187         close: 7
188         delete: 30
189     logstash:
190         close: 7
191         delete: 30
192     metricbeat:
193         close: 5
194         delete: 10
195     packetbeat:
196         close: 5
197         delete: 10
198
199 kibana:
200     header_value: "reporting"
201     import_delay: 10
202     import_retries: 10
203     # logs configuration
204     logrotate: enabled # or disabled
205     history_days: 30 # How many days to store logs if logrotate is set to 'enabled'
206     log:
207         baseuri: "kibana_log"
208         api_call_timeout: 120
209         groupe: "log"
210         port: 5601
211         default_index_pattern: "logstash-vitam*"
212         check_consul: 10 # in seconds
213         # default shards & replica
214         shards: 1
215         replica: 1
216         # pour index logstash-*
217         metrics:
218             shards: 1
219             replica: 1
220         # pour index metrics-vitam-*
221         logs:
222             shards: 1
223             replica: 1
224         # pour index metricbeat-*
225         metricbeat:
226             shards: 3 # must be a factor of 30
227             replica: 1
228     data:
229         baseuri: "kibana_data"
230         # OMA : bugdette : api_call_timeout is used for retries ; should ceate a
231         ↪ separate variable rather than this one
232         api_call_timeout: 120
233         groupe: "data"
234         port: 5601
235         default_index_pattern: "logbookoperation_*"
236         check_consul: 10 # in seconds
237         # index template for .kibana
238         shards: 1
239         replica: 1
240     syslog:

```

(suite sur la page suivante)

(suite de la page précédente)

```

241     # value can be syslog-ng or rsyslog
242     name: "rsyslog"
243
244 cerebro:
245     baseuri: "cerebro"
246     port: 9000
247     check_consul: 10 # in seconds
248
249 siegfried:
250     port: 19000
251     consul_check: 10 # in seconds
252
253 clamav:
254     port: 3310
255     # frequency freshclam for database update per day (from 0 to 24 - 24 meaning_
↪ hourly check)
256     db_update_periodicity: 1
257     # logs configuration
258     logrotate: enabled # or disabled
259     history_days: 30 # How many days to store logs if logrotate is set to 'enabled
↪ '
260
261 mongo_express:
262     baseuri: "mongo-express"
263
264 ldap_authentication:
265     ldap_protocol: "ldap"
266     ldap_server: "% if groups['ldap']|length > 0 %>{{ groups['ldap']|first }}{%_
↪ endif %}"
267     ldap_port: "389"
268     ldap_base: "dc=programmevitam,dc=fr"
269     ldap_login: "cn=Manager,dc=programmevitam,dc=fr"
270     uid_field: "uid"
271     ldap_userDn_Template: "uid={0},ou=people,dc=programmevitam,dc=fr"
272     ldap_group_request: "(&(objectClass=groupOfNames)(member={0}))"
273     ldap_admin_group: "cn=admin,ou=groups,dc=programmevitam,dc=fr"
274     ldap_user_group: "cn=user,ou=groups,dc=programmevitam,dc=fr"
275     ldap_guest_group: "cn=guest,ou=groups,dc=programmevitam,dc=fr"
276
277 java_prerequisites:
278     debian: "openjdk-11-jre-headless"
279     redhat: "java-11-openjdk-headless"

```

Note : Installation multi-sites. Déclarer dans `consul_remote_sites` les datacenters Consul des autres site ; se référer à l'exemple fourni pour renseigner les informations.

Note : Concernant Curator, en environnement de production, il est recommandé de procéder à la fermeture des index au bout d'une semaine pour les index de type « logstash » (3 jours pour les index « metrics »), qui sont le reflet des traces applicatives des composants de la solution logicielle *VITAM*. Il est alors recommandé de lancer le *delete* de ces index au bout de la durée minimale de rétention : 1 an (il n'y a pas de durée de rétention minimale légale sur les index « metrics », qui ont plus une vocation technique et, éventuellement, d'investigations).

- `deployment/environments/group_vars/all/jvm_vars.yml`, comme suit :

```

1 ---
2
3 vitam:
4   accessinternal:
5     jvm_opts:
6       # memory: "-Xms512m -Xmx512m"
7       # gc: ""
8       # java: ""
9   accessexternal:
10    jvm_opts:
11      # memory: "-Xms512m -Xmx512m"
12      # gc: ""
13      # java: ""
14   elasticsearch:
15     jvm_opts:
16       # memory: "-Xms512m -Xmx512m"
17       # gc: ""
18       # java: ""
19   batchreport:
20     jvm_opts:
21       # memory: "-Xms512m -Xmx512m"
22       # gc: ""
23       # java: ""
24   ingestinternal:
25     jvm_opts:
26       # memory: "-Xms512m -Xmx512m"
27       # gc: ""
28       # java: ""
29   ingestexternal:
30     jvm_opts:
31       # memory: "-Xms512m -Xmx512m"
32       # gc: ""
33       # java: ""
34   metadata:
35     jvm_opts:
36       # memory: "-Xms512m -Xmx512m"
37       # gc: ""
38       # java: ""
39   ihm_demo:
40     jvm_opts:
41       # memory: "-Xms512m -Xmx512m"
42       # gc: ""
43       # java: ""
44   ihm_recette:
45     jvm_opts:
46       # memory: "-Xms512m -Xmx512m"
47       # gc: ""
48       # java: ""
49   logbook:
50     jvm_opts:
51       # memory: "-Xms512m -Xmx512m"
52       # gc: ""
53       # java: ""
54     timer_jvm_opts:
55       # memory: "-Xms32m -Xmx128m"
56       # gc: ""
57       # java: ""

```

(suite sur la page suivante)


```

58 workspace:
59     jvm_opts:
60         # memory: "-Xms512m -Xmx512m"
61         # gc: ""
62         # java: ""
63 processing:
64     jvm_opts:
65         # memory: "-Xms512m -Xmx512m"
66         # gc: ""
67         # java: ""
68 worker:
69     jvm_opts:
70         # memory: "-Xms512m -Xmx512m"
71         # gc: ""
72         # java: ""
73 storageengine:
74     jvm_opts:
75         # memory: "-Xms512m -Xmx512m"
76         # gc: ""
77         # java: ""
78     timer_jvm_opts:
79         # memory: "-Xms32m -Xmx128m"
80         # gc: ""
81         # java: ""
82 storageofferdefault:
83     jvm_opts:
84         # memory: "-Xms512m -Xmx512m"
85         # gc: ""
86         # java: ""
87 functional_administration:
88     jvm_opts:
89         # memory: "-Xms512m -Xmx512m"
90         # gc: ""
91         # java: ""
92     timer_jvm_opts:
93         # memory: "-Xms32m -Xmx128m"
94         # gc: ""
95         # java: ""
96 security_internal:
97     jvm_opts:
98         # memory: "-Xms512m -Xmx512m"
99         # gc: ""
100         # java: ""
101 library:
102     jvm_opts:
103         memory: "-Xms32m -Xmx128m"
104         # gc: ""
105         # java: ""

```

Note : Cette configuration est appliquée à la solution logicielle *VITAM* ; il est possible de créer un tuning par « groupe » défini dans ansible.

4.2.5.13 Paramétrage de l'Offre Froide (librairies de cartouches)

Suite à l'introduction des offres bandes, plusieurs notions supplémentaires sont prises en compte dans ce fichier. De nouvelles entrées ont été ajoutées pour décrire d'une part le matériel robotique assigné à l'offre froide, et les répertoires d'échanges temporaires d'autre part. Les éléments de configuration doivent être renseignés par l'exploitant.

- Lecture asynchrone

Un paramètre a été ajouté aux définitions de stratégie. *AsyncRead* permet de déterminer si l'offre associée fonctionne en lecture asynchrone, et désactive toute possibilité de lecture directe sur l'offre. Une offre froide « offer-tape » doit être configurée en lecture asynchrone. La valeur par défaut pour *asyncRead* est False.

Exemple :

```
vitam_strategy:
- name: offer-tape-1
  referent: false
  asyncRead: **true**
- name: offer-fs-2
  referent: true
  asyncRead: false
```

- Périphériques liés à l'usage des bandes magnétiques

Terminologie :

- **tapeLibrary** une librairie de bande dans son ensemble. Une *tapeLibrary* est constituée de 1 à n « robot » et de 1 à n « drives ». Une offre froide nécessite la déclaration d'au moins une librairie pour fonctionner. L'exploitant doit déclarer un identifiant pour chaque librairie. Ex : TAPE_LIB_1
- **drive** un drive est un lecteur de cartouches. Il doit être identifié par un *path* scsi unique. Une offre froide nécessite la déclaration d'au moins un lecteur pour fonctionner.

Note : il existe plusieurs fichiers périphériques sur Linux pour un même lecteur

Les plus classiques sont par exemple `/dev/st0` et `/dev/nst0` pour le premier drive détecté par le système. L'usage de `/dev/st0` indique au système que la bande utilisée dans le lecteur associé devra être rembobinée après l'exécution de la commande appelante. A contrario, `/dev/nst0` indique au système que la bande utilisée dans le lecteur associé devra rester positionnée après le dernier marqueur de fichier utilisé par l'exécution de la commande appelante.

Important : Pour que l'offre froide fonctionne correctement, il convient de configurer une version `/dev/nstxx`

Note : Il peut arriver sur certains systèmes que l'ordre des lecteurs de bandes varient après un reboot de la machine. Pour s'assurer la persistance de l'ordre des lecteurs dans la configuration VITAM, il est conseillé d'utiliser les fichiers périphériques présents dans `/dev/tape/by-id/` qui s'appuient sur des références au hardware pour définir les drives.

- **robot** un robot est le composant chargé de procéder au déplacement des cartouches dans une *tapeLibrary*, et de procéder à l'inventaire de ses ressources. Une offre froide nécessite la déclaration d'au moins un robot pour fonctionner. L'exploitant doit déclarer un fichier de périphérique scsi générique (ex : `/dev/sg4`) associé à la robotique sur son système. A l'instar de la configuration des drives, il est recommandé d'utiliser le device présent dans `/dev/tape/by-id/` pour déclarer les robots.

Définition d'une offre froide :

Une offre froide (OF) doit être définie dans la rubrique « vitam_offers » avec un provider de type *tape-library*

Exemple :

```
vitam_offers:
  offer-tape-1:
    provider: tape-library
    tapeLibraryConfiguration:
```

La description « tapeLibraryConfiguration » débute par la définition des répertoires de stockage ainsi que le paramétrage des *tars*. * **inputFileStorageFolder** Répertoire où seront stockés les objets à intégrer à l'OF * **inputTarStorageFolder** Répertoire où seront générés et stockés les *tars* avant transfert sur bandes * **outputTarStorageFolder** Répertoire où seront rapatriés les *tars* depuis les bandes. * **MaxTarEntrySize** Taille maximale au-delà de laquelle les fichiers entrant seront découpés en segment, en octets * **maxTarFileSize** Taille maximale des *tars* à constituer, en octets. * **forceOverrideNonEmptyCartridge** Permet de passer outre le contrôle vérifiant que les bandes nouvellement introduites sont vides. Par défaut à *false* * **useSudo** Réservé à un usage futur – laisser à *false*.

Note : MaxTarEntrySize doit être strictement inférieur à maxTarFileSize

Exemple :

```
inputFileStorageFolder: "/vitam/data/offer/offer/inputFiles"
inputTarStorageFolder: "/vitam/data/offer/offer/inputTars"
outputTarStorageFolder: "/vitam/data/offer/offer/outputTars"
maxTarEntrySize: 10000000
maxTarFileSize: 10000000000
ForceOverrideNonEmptyCartridge: False
useSudo: false
```

Par la suite, un paragraphe « topology » décrivant la topologie de l'offre doit être renseigné. L'objectif de cet élément est de pouvoir définir une segmentation de l'usage des bandes pour répondre à un besoin fonctionnel. Il convient ainsi de définir des *buckets*, qu'on peut voir comme un ensemble logique de bandes, et de les associer à un ou plusieurs tenants.

- **tenants** tableau de 1 à n identifiants de tenants au format [1,...,n]
- **tarBufferingTimeoutInMinutes** Valeur en minutes durant laquelle un tar peut rester ouvert

Exemple :

```
topology:
  buckets:
    test:
      tenants: [0]
      tarBufferingTimeoutInMinutes: 60
    admin:
      tenants: [1]
      tarBufferingTimeoutInMinutes: 60
    prod:
      tenants: [2,3,4,5,6,7,8,9]
      tarBufferingTimeoutInMinutes: 60
```

Enfin, la définition des équipements robotiques proprement dite doit être réalisée dans le paragraphe « tapeLibraries ».

- **robots** : Définition du bras robotique de la librairie.
- **device** : Chemin du fichier de périphérique scsi générique associé au bras.
- **mtxPath** : Chemin vers la commande Linux de manipulation du bras.

- **timeoutInMilliseconds** : timeout en millisecondes à appliquer aux ordres du bras.
- **drives** : Définition du/ou des lecteurs de cartouches de la librairie.
- **index** : Numéro de lecteur, valeur débutant à 0
- **device** : Chemin du fichier de périphérique scsi SANS REMBOBINAGE associé au lecteur.
- **mtPath** : Chemin vers la commande Linux de manipulation des lecteurs.
- **ddPath** : Chemin vers la commande Linux de copie de bloc de données.
- **tarPath** : Chemin vers la commande Linux de création d'archives tar.
- **timeoutInMilliseconds** : timeout en millisecondes à appliquer aux ordres du lecteur.

Exemple :

```
tapeLibraries:
  TAPE_LIB_1:
    robots:
      -
        device: /dev/tape/by-id/scsiQUANTUM_10F73224E6664C84A1D00000
        mtPath: "/usr/sbin/mtx"
        timeoutInMilliseconds: 3600000
    drives:
      -
        index: 0
        device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_1235308739-nst
        mtPath: "/bin/mt"
        ddPath: "/bin/dd"
        tarPath: "/bin/tar"
        timeoutInMilliseconds: 3600000
      -
        index: 1
        device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0951859786-nst
        mtPath: "/bin/mt"
        ddPath: "/bin/dd"
        tarPath: "/bin/tar"
        timeoutInMilliseconds: 3600000
      -
        index: 2
        device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0269493808-nst
        mtPath: "/bin/mt"
        ddPath: "/bin/dd"
        tarPath: "/bin/tar"
        timeoutInMilliseconds: 3600000
      -
        index: 3
        device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0566471858-nst
        mtPath: "/bin/mt"
        ddPath: "/bin/dd"
        tarPath: "/bin/tar"
        timeoutInMilliseconds: 3600000
```

4.2.5.14 Sécurisation SELinux

Depuis la release R13, la solution logicielle *VITAM* prend désormais en charge l'activation de SELinux sur le périmètre du composant worker et des processus associés aux *griffins* (greffons de préservation).

SELinux (Security-Enhanced Linux) permet de définir des politiques de contrôle d'accès à différents éléments du système d'exploitation en répondant essentiellement à la question « May <subject> do <action> to <object> », par exemple « May a web server access files in user's home directories ».

Chaque processus est ainsi confiné à un (voire plusieurs) domaine(s), et les fichiers sont étiquetés en conséquence. Un processus ne peut ainsi accéder qu'aux fichiers étiquetés pour le domaine auquel il est confiné.

Note : La solution logicielle *VITAM* ne gère actuellement que le mode *targeted* (« only *targeted* processes are protected »)

Les enjeux de la sécurisation SELinux dans le cadre de la solution logicielle *VITAM* sont de garantir que les processus associés aux *griffins* (greffons de préservation) n'auront accès qu'aux ressources système strictement requises pour leur fonctionnement et leurs échanges avec les composants *worker*.

Note : La solution logicielle *VITAM* ne gère actuellement SELinux que pour le système d'exploitation Centos

Avertissement : SELinux n'a pas vocation à remplacer quelque système de sécurité existant, mais vise plutôt à les compléter. Aussi, la mise en place de politiques de sécurité reste de mise et à la charge de l'exploitant. Par ailleurs, l'implémentation SELinux proposée avec la solution logicielle *VITAM* est minimale et limitée au greffon de préservation Siegfried. Cette implémentation pourra si nécessaire être complétée ou améliorée par le projet d'implémentation.

SELinux propose trois modes différents :

- *Enforcing* : dans ce mode, les accès sont restreints en fonction des règles SELinux en vigueur sur la machine ;
- *Permissive* : ce mode est généralement à considérer comme un mode de débogage. En mode permissif, les règles SELinux seront interrogées, les erreurs d'accès logguées, mais l'accès ne sera pas bloqué.
- *Disabled* : SELinux est désactivé. Rien ne sera restreint, rien ne sera loggué.

La mise en oeuvre de SELinux est prise en charge par le processus de déploiement et s'effectue de la sorte :

- Isoler dans l'inventaire de déploiement les composants worker sur des hosts dédiés (ne contenant aucun autre composant *VITAM*)
- Positionner pour ces hosts un fichier *hostvars* sous *environments/host_vars/* contenant la déclaration suivante

```
selinux_state: "enforcing"
```

- Procéder à l'installation de la solution logicielle *VITAM* grâce aux playbooks ansible fournis, et selon la procédure d'installation classique décrite dans le DIN

4.2.5.15 Installation de la stack prometheus

Note : Si vous disposez d'un prometheus server et alertmanager déjà existant, vous pouvez juste installer *node_exporter*

Prometheus server et alertmanager sont des add-ons dans la solution *VITAM*. Il est possible de les installer ou désinstaller via la configuration dans le fichier *cots_var.yml*. Voici à quoi correspond une configuration qui permettra d'installer toute la stack prometheus.

```
prometheus:
  metrics_path: /admin/v1/metrics
  check_consul: 10 # in seconds
```

(suite sur la page suivante)

(suite de la page précédente)

```

prometheus_config_file_target_directory: # Set path where "prometheus.yml" file
↳ will be generated. Example: /tmp/
server:
  enabled: true
  port: 19090
node_exporter:
  enabled: true
  port: 19100
  metrics_path: /metrics
alertmanager:
  enabled: true
  api_port: 19093
  cluster_port: 19094

```

- L'adresse d'écoute de ces composants est celle de la patte d'administration.
- Vous pouvez surcharger la valeur de certaines de ces variables (Par exemple le port d'écoute, le path de l'API).
- Pour désinstaller ou désactiver un composant de la stack prometheus il suffit de mettre la valeur de `enabled` de ce composant à `false`
- Pour générer uniquement le fichier de configuration `prometheus.yml` à partir du fichier d'inventaire de l'environnement en question, il suffit de spécifier le répertoire destination dans la variable `prometheus_config_file_target_directory`

4.2.5.15.1 Playbooks ansible

Veuillez vous référer à la documentation d'exploitation pour plus d'information.

- Installer prometheus et alertmanager

```

ansible-playbook ansible-vitam-extra/prometheus.yml -i environments/hosts.
↳ <environnement> --ask-vault-pass

```

- Générer le fichier de conf `prometheus.yml` dans le dossier `prometheus_config_file_target_directory`

```

ansible-playbook ansible-vitam-extra/prometheus.yml -i environments/hosts.
↳ <environnement> --ask-vault-pass

```

```

--tags gen_prometheus_config ..

```

4.2.5.16 Installation de grafana

Note : Si vous disposez déjà d'un Grafana server, vous pouvez l'utiliser pour l'interconnecter au serveur prometheus.

Grafana server est un add-on dans la solution *VITAM*. Il est possible de l'installer/désinstaller via la configuration dans le fichier `cots_var.yml`. Voici à quoi correspond une configuration qui permettra d'installer ce serveur.

```

grafana:
  enabled: true
  check_consul: 10 # in seconds
  http_port: 13000

```

- L'adresse d'écoute de ces composants est celle de la patte d'administration.
- Vous pouvez surcharger le numéro de port d'écoute.
- Pour désinstaller ou désactiver un composant il suffit de mettre la valeur de `enabled` à `false`

4.2.5.16.1 Playbook ansible

Veillez vous référer à la documentation d'exploitation pour plus d'informations.

- Installer Grafana

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

4.2.5.16.2 Configuration

Dans le cas ou le serveur grafana est derrière un serveur proxy, vous devez apporter des modification au fichier de configuration `grafana.conf.j2`

Voici les variables modifiées par la solution *VITAM* pour que ça marche derrière le proxy apache.

```
[server]  
root_url = http://{ ip_admin }:{ grafana.http_port }/grafana  
serve_from_sub_path = true  
  
[auth.basic]  
enabled = false
```

Avertissement : Lors de la première connexion, vous devrez changer le mot de passe par défaut (login : admin ; password : admin) et configurer le datasource et créer/importer les dashboards manuellement.

4.2.6 Procédure de première installation

4.2.6.1 Déploiement

4.2.6.1.1 Cas particulier : utilisation de ClamAv en environnement Debian

Dans le cas de l'installation en environnement Debian, la base de données n'est pas intégrée avec l'installation de ClamAv, C'est la commande `freshclam` qui en assure la charge. Si vous n'êtes pas connecté à internet, la base de données doit être installée manuellement. Les liens suivants indiquent la procédure à suivre : [Installation ClamAv](#)¹⁶ et [Section Virus Database](#)¹⁷

4.2.6.1.2 Fichier de mot de passe des vaults ansible

Par défaut, le mot de passe des *vault* sera demandé à chaque exécution d'ansible avec l'utilisation de l'option `--ask-vault-pass` de la commande `ansible-playbook`.

<https://www.clamav.net/documents/installing-clamav>
<https://www.clamav.net/downloads>

Pour simplifier l'exécution des commandes `ansible-playbook`, vous pouvez utiliser un fichier `repertoire_deploiement/vault_pass.txt` contenant le mot de passe des fichiers vault. Ainsi, vous pouvez utiliser l'option `--vault-password-file=vault_pass.txt` à la place de l'option `--ask-vault-pass` dans les différentes commandes de cette documentation.

Avertissement : Il est déconseillé de conserver le fichier `vault_pass.txt` sur la machine de déploiement `ansible` car ce fichier permet d'avoir accès à l'ensemble des secrets de *VITAM*.

4.2.6.1.3 Mise en place des repositories VITAM (optionnel)

VITAM fournit un playbook permettant de définir sur les partitions cible la configuration d'appel aux repositories spécifiques à *VITAM* :

Editer le fichier `repertoire_inventory/group_vars/all/repositories.yml` à partir des modèles suivants (décommenter également les lignes) :

Pour une cible de déploiement CentOS :

```

1 #vitam_repositories:
2 #- key: repo 1
3 #   value: "file:///code"
4 #   proxy: http://proxy
5 #- key: repo 2
6 #   value: "http://www.programmevitam.fr"
7 #   proxy: _none_
8 #- key: repo 3
9 #   value: "ftp://centos.org"
10 #   proxy:
```

Pour une cible de déploiement Debian :

```

1 #vitam_repositories:
2 #- key: repo 1
3 #   value: "file:///code"
4 #   subtree: "/"
5 #   trusted: "[trusted=yes]"
6 #- key: repo 2
7 #   value: "http://www.programmevitam.fr"
8 #   subtree: "/"
9 #   trusted: "[trusted=yes]"
10 #- key: repo 3
11 #   value: "ftp://centos.org"
12 #   subtree: "binary"
13 #   trusted: "[trusted=yes]"
```

Ce fichier permet de définir une liste de repositories. Décommenter et adapter à votre cas.

Pour mettre en place ces repositories sur les machines cibles, lancer la commande :

```

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

Note : En environnement CentOS, il est recommandé de créer des noms de *repository* commençant par *vitam-*.

4.2.6.1.4 Génération des *hostvars*

Une fois l'étape de *PKI* effectuée avec succès, il convient de procéder à la génération des *hostvars*, qui permettent de définir quelles interfaces réseau utiliser. Actuellement la solution logicielle *VITAM* est capable de gérer 2 interfaces réseau :

- Une d'administration
- Une de service

4.2.6.1.4.1 Cas 1 : Machines avec une seule interface réseau

Si les machines sur lesquelles *VITAM* sera déployé ne disposent que d'une interface réseau, ou si vous ne souhaitez en utiliser qu'une seule, il convient d'utiliser le playbook `repertoire_playbook ansible-generate_hostvars_for_1_network_interface.yml`

Cette définition des *host_vars* se base sur la directive ansible `ansible_default_ipv4.address`, qui se base sur l'adresse *IP* associée à la route réseau définie par défaut.

Avertissement : Les communications d'administration et de service transiteront donc toutes les deux via l'unique interface réseau disponible.

4.2.6.1.4.2 Cas 2 : Machines avec plusieurs interfaces réseau

Si les machines sur lesquelles *VITAM* sera déployé disposent de plusieurs interfaces et si celles-ci respectent cette règle :

- Interface nommée `eth0` = `ip_service`
- Interface nommée `eth1` = `ip_admin`

Alors il est possible d'utiliser le playbook `ansible-vitam-extra/generate_hostvars_for_2_network_interfaces.yml`

Note : Pour les autres cas de figure, il sera nécessaire de générer ces *hostvars* à la main ou de créer un script pour automatiser cela.

4.2.6.1.4.3 Vérification de la génération des *hostvars*

A l'issue, vérifier le contenu des fichiers générés sous `repertoire_inventory/host_vars/` et les adapter au besoin.

Prudence : Cas d'une installation multi-sites. Sur site secondaire, s'assurer que, pour les machines hébergeant les offres, la directive `ip_wan` a bien été déclarée (l'ajouter manuellement, le cas échéant), pour que le site *primaire* sache les contacter via une IP particulière. Par défaut, c'est l'IP de service qui sera utilisée.

4.2.6.1.5 Déploiement

Une fois les étapes précédentes correctement effectuées (en particulier, la section *Génération des magasins de certificats* (page 62)), le déploiement s'effectue depuis la machine *ansible* et va distribuer la solution *VITAM* selon l'inventaire correctement renseigné.

Une fois l'étape de la génération des hosts effectuée avec succès, le déploiement est à réaliser avec la commande suivante :

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

Note : Une confirmation est demandée pour lancer ce script. Il est possible de rajouter le paramètre `-e confirmation=yes` pour bypasser cette demande de confirmation (cas d'un déploiement automatisé).

Prudence : Dans le cas où l'installateur souhaite utiliser un *repository* de binaires qu'il gère par lui-même, il est fortement recommandé de rajouter `--skip-tags "enable_vitam_repo"` à la commande `ansible-playbook` ; dans ce cas, le comportement de `yum` n'est pas impacté par la solution de déploiement.

4.2.7 Éléments *extras* de l'installation

Prudence : Les éléments décrits dans cette section sont des éléments « extras » ; il ne sont pas officiellement supportés, et ne sont par conséquent pas inclus dans l'installation de base. Cependant, ils peuvent s'avérer utile, notamment pour les installations sur des environnements hors production.

Prudence : Dans le cas où l'installateur souhaite utiliser un *repository* de binaires qu'il gère par lui-même, il est fortement recommandé de rajouter `--skip-tags "enable_vitam_repo"` à la commande `ansible-playbook` ; dans ce cas, le comportement de `yum` n'est pas impacté par la solution de déploiement.

4.2.7.1 Configuration des *extras*

Le fichier `repertoire_inventory|group_vars/all/extra_vars.yml` contient la configuration des *extras* :

```
1 ---  
2  
3 vitam:  
4   ihm_recette:  
5     vitam_component: ihm-recette  
6     host: "ihm-recette.service.{{ consul_domain }}"  
7     port_service: 8445  
8     port_admin: 28204  
9     baseurl: /ihm-recette  
10    static_content: "{{ vitam_defaults.folder.root_path }}/app/ihm-recette"  
11    baseuri: "ihm-recette"  
12    secure_mode:  
13      - authc  
14    https_enabled: true  
15    secret_platform: "false"  
16    cluster_name: "{{ elasticsearch.data.cluster_name }}"  
17    session_timeout: 1800000  
18    secure_cookie: true  
19    use_proxy_to_clone_tests: "yes"  
20    elasticsearch_mapping_dir: "{{ vitam_defaults.folder.root_path }}/conf/ihm-  
↪ recette/mapping"
```

(suite sur la page suivante)

(suite de la page précédente)

```

21  library:
22      vitam_component: library
23      host: "library.service.{{ consul_domain }}"
24      port_service: 8090
25      port_admin: 28090
26      baseuri: "doc"
27      https_enabled: false
28      secret_platform: "false"
29      metrics_enabled: false
30      consul_business_check: 30 # value in seconds
31      consul_admin_check: 30 # value in seconds
32
33  tenant_to_clean_before_tnr: ["0", "1"]
34
35  # Period units in seconds
36  metricbeat:
37      system:
38          period: 10
39      mongodb:
40          period: 10
41      elasticsearch:
42          period: 10
43
44  docker_opts:
45      registry_httponly: yes
46      vitam_docker_tag: latest
47
48  gatling_install: false
49  docker_install: true # whether or not install docker & docker images

```

Avvertissement : À modifier selon le besoin avant de lancer le playbook ! Les composants ihm-recette et ihm-demo ont la variable `secure_cookie` paramétrée à `true` par défaut, ce qui impose de pouvoir se connecter dessus uniquement en https (même derrière un reverse proxy). Le paramétrage de cette variable se fait dans le fichier `environments/group_vars/all/vitam_vars.yml`

Note : La section `metricbeat` permet de configurer la périodicité d'envoi des informations collectées. Selon l'espace disponible sur le *cluster* Elasticsearch de log et la taille de l'environnement *VITAM* (en particulier, le nombre de machines), il peut être nécessaire d'allonger cette périodicité (en secondes).

Le fichier `repertoire_inventory/group_vars/all/all/vault-extra.yml` contient les secrets supplémentaires des *extras* ; ce fichier est encrypté par `ansible-vault` et doit être paramétré avant le lancement de l'orchestration du déploiement, si le composant ihm-recette est déployé avec récupération des *TNR*.

```

1  # Example for git lfs ; uncomment & use if needed
2  #vitam_gitlab_itest_login: "account"
3  #vitam_gitlab_itest_password: "change_it_4DU42JVf2x2xmPBs"

```

Note : Pour ce fichier, l'encrypter avec le même mot de passe que `vault-vitam.yml`.

4.2.7.2 Déploiement des *extras*

Plusieurs playbooks d'*extras* sont fournis pour usage « tel quel ».

4.2.7.2.1 ihm-recette

Ce *playbook* permet d'installer également le composant *VITAM* ihm-recette.

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

Prudence : Avant de jouer le *playbook*, ne pas oublier, selon le contexte d'usage, de positionner correctement la variable `secure_cookie` décrite plus haut.

4.2.7.2.2 Extras complet

Ce *playbook* permet d'installer :

- des éléments de monitoring système
- un serveur Apache pour naviguer sur le `/vitam` des différentes machines hébergeant *VITAM*
- mongo-express (en docker ; une connexion internet est alors nécessaire)
- le composant *VITAM* library, hébergeant la documentation du projet
- le composant *VITAM* ihm-recette (utilise si configuré des dépôts de jeux de tests)
- un reverse proxy, afin de fournir une page d'accueil pour les environnements de test
- l'outillage de tests de performance

Avertissement : Pour se connecter aux *IHM*, il faut désormais configurer `reverse_proxy_port=443` dans l'inventaire.

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

Procédures de mise à jour de la configuration

Cette section décrit globalement les processus de reconfiguration d'une solution logicielle *VITAM* déjà en place et ne peut se substituer aux recommandations effectuées dans la « release-notes » associée à la fourniture des composants mis à niveau.

Se référer également aux *DEX* pour plus de procédures.

5.1 Cas d'une modification du nombre de tenants

Modifier dans le fichier d'inventaire la directive `vitam_tenant_ids`

Exemple :

```
vitam_tenant_ids=[0,1,2]
```

A l'issue, il faut lancer le playbook de déploiement de *VITAM* (et, si déployé, les extras) avec l'option supplémentaire `--tags update_vitam_configuration`.

Exemple :

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

5.2 Cas d'une modification des paramètres JVM

Se référer à *Tuning JVM* (page 62)

Pour les partitions sur lesquelles une modification des paramètres *JVM* est nécessaire, il faut modifier les « hostvars » associées.

A l'issue, il faut lancer le playbook de déploiement de *VITAM* (et, si déployé, les *extras*) avec l'option supplémentaire `--tags update_jvmoptions_vitam`.

Exemple :

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

Prudence : Limitation technique à ce jour ; il n'est pas possible de définir des variables *JVM* différentes pour des composants colocalisés sur une même partition.

5.3 Cas de la mise à jour des *griffins*

Modifier la directive `vitam_griffins` contenue dans le fichier `environments/group_vars/all/vitam_vars.yml`.

Note : Dans le cas d'une montée de version des composant *griffins*, ne pas oublier de mettre à jour l'URL du dépôt de binaire associé.

Relancer le script de déploiement en ajoutant en fin de ligne `--tags griffins` pour ne procéder qu'à l'installation/mise à jour des *griffins*.

6.1 Validation du déploiement

La procédure de validation est commune aux différentes méthodes d'installation.

6.1.1 Sécurisation du fichier `vault_pass.txt`

Le fichier `vault_pass.txt` est très sensible; il contient le mot de passe du fichier `reper-toire_inventory|group_vars/all/vault.yml` qui contient les divers mots de passe de la plate-forme. A l'issue de l'installation, il est primordial de le sécuriser (suppression du fichier ou application d'un `chmod 400`).

6.1.2 Validation manuelle

Chaque service *VITAM* (en dehors de bases de données) expose des URL de statut à l'adresse suivante : `<protocole web http ou https>://<host>:<port>/admin/v1/status` Cette URL doit retourner une réponse HTTP 204 sur une requête HTTP GET, si OK.

Un playbook d'appel de l'intégralité des autotests est également inclus (`deployment/ansible-vitam-exploitation/status_vitam.yml`). Il est à lancer de la même manière que pour l'installation de *VITAM* (en renommant le playbook à exécuter).

Il est également possible de vérifier la version installée de chaque composant par l'URL :

`<protocole web http ou https>://<host>:<port>/admin/v1/version`

6.1.3 Validation via Consul

Consul possède une *IHM* pour afficher l'état des services *VITAM* et supervise le « `/admin/v1/status` » de chaque composant *VITAM*, ainsi que des check TCP sur les bases de données.

Pour se connecter à Consul : `http//<Nom du 1er host dans le groupe ansible hosts_consul_server>:8500/ui`

Pour chaque service, la couleur à gauche du composant doit être verte (correspondant à un statut OK).

Si une autre couleur apparaît, cliquer sur le service « KO » et vérifier le test qui ne fonctionne pas.

6.1.4 Post-installation : administration fonctionnelle

A l'issue de l'installation, puis la validation, un **administrateur fonctionnel** doit s'assurer que :

- le référentiel PRONOM ([lien vers pronom¹⁸](#)) est correctement importé depuis « Import du référentiel des formats » et correspond à celui employé dans Siegfried
- le fichier « rules » a été correctement importé via le menu « Import du référentiel des règles de gestion »
- à terme, le registre des fonds a été correctement importé

Les chargements sont effectués depuis l'*IHM* demo.

6.2 Sauvegarde des éléments d'installation

Après installation, il est fortement recommandé de sauvegarder les éléments de configuration de l'installation (i.e. le contenu du répertoire `déploiement/environnements`); ces éléments seront à réutiliser pour les mises à jour futures.

Astuce : Une bonne pratique consiste à gérer ces fichiers dans un gestionnaire de version (ex : git)

Prudence : Si vous avez modifié des fichiers internes aux rôles, ils devront également être sauvegardés.

6.3 Troubleshooting

Cette section a pour but de recenser les problèmes déjà rencontrés et y apporter une solution associée.

6.3.1 Erreur au chargement des *index template* kibana

Cette erreur ne se produit qu'en cas de *filesystem* plein sur les partitions hébergeant un cluster elasticsearch. Par sécurité, kibana passe alors ses *index* en `READ ONLY`.

Pour fixer cela, il est d'abord nécessaire de déterminer la cause du *filesystem* plein, puis libérer ou agrandir l'espace disque.

Ensuite, comme indiqué sur [ce fil de discussion¹⁹](#), vous devez désactiver le mode `READ ONLY` dans les *settings* de l'*index .kibana* du cluster elasticsearch.

Exemple :

```
PUT .kibana/_settings
{
  "index": {
    "blocks": {
```

(suite sur la page suivante)

<http://www.nationalarchives.gov.uk/aboutapps/pronom/droid-signature-files.htm>
<https://discuss.elastic.co/t/forbidden-12-index-read-only-allow-delete-api/110282/2>

(suite de la page précédente)

```
        "read_only_allow_delete": "false"
    }
}
}
```

Indication : Il est également possible de lancer cet appel via l'*IHM* du kibana associé, dans l'onglet `Dev Tools`.

À l'issue, vous pouvez relancer l'installation de la solution logicielle *VITAM*.

6.3.2 Erreur au chargement des tableaux de bord Kibana

Dans le cas de machines petitement taillées, il peut arriver que, durant le déploiement, la tâche `Wait for the kibana port to be opened` prenne plus de temps que le *timeout* défini (`vitam_defaults.services.start_timeout`). Pour fixer cela, il suffit de relancer le déploiement.

6.4 Retour d'expérience / cas rencontrés

6.4.1 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 :

- Centos²⁰
- Debian²¹

6.4.2 Mongo-express ne se connecte pas à la base de données associée

Si mongoDB a été redémarré, il faut également redémarrer mongo-express.

6.4.3 Elasticsearch possède des shard non alloués (état « UNASSIGNED »)

Lors de la perte d'un noeud d'un cluster elasticsearch, puis du retour de ce noeud, certains shards d'elasticsearch 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 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`) :

<https://www.rsyslog.com/rhelcentos-rpms/>
<https://www.rsyslog.com/debian-repository/>

```
{
  "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](#)²².

6.4.4 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](#)²³). 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.

6.4.5 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 :

```
curl -XPUT -H "Content-Type: application/json" http://<es-host>:<es-port>/_all/_
→settings -d '{"index.blocks.read_only_allow_delete": null}'
```

<https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-reroute.html>
<https://www.elastic.co/guide/en/elasticsearch/reference/1.4/states.html>

6.4.6 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](#)²⁴ et [mongotop](#)²⁵.

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 troubleshoot [les problèmes dûs à la réplication](#)²⁶ :

```
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](#)²⁷) :

```
echo "{nThreads:16,fileSizeMB:10000,r:true,w:true}" | mongoperf
```

6.4.7 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 :

<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/>

- `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.

6.4.8 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 fonctionnal-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.

6.4.9 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.

CHAPITRE 7

Montée de version

Pour toute montée de version applicative de la solution logicielle *VITAM*, se référer au *DMV*.

8.1 Vue d'ensemble de la gestion des certificats

8.1.1 Liste des suites cryptographiques & protocoles supportés par VITAM

Il est possible de consulter les *ciphers* supportés par la solution logicielle *VITAM* dans deux fichiers disponibles sur ce chemin : *ansible-vitam/roles/vitam/templates/*

- **Le fichier `jetty-config.xml.j2`**
 - La balise contenant l'attribut `name= »IncludeCipherSuites »` référence les ciphers supportés
 - La balise contenant l'attribut `name= »ExcludeCipherSuites »` référence les ciphers non supportés
- **Le fichier `java.security.j2`**
 - La ligne `jdk.tls.disabledAlgorithms` renseigne les *ciphers* désactivés au niveau java

Avertissement : Les 2 balises concernant les *ciphers* sur le fichier `jetty-config.xml.j2` sont complémentaires car elles comportent des *wildcards* (*) ; en cas de conflit, l'exclusion est prioritaire.

Voir aussi :

Ces fichiers correspondent à la configuration recommandée ; celle-ci est décrite plus en détail dans le *DAT* (chapitre sécurité).

8.1.2 Vue d'ensemble de la gestion des certificats

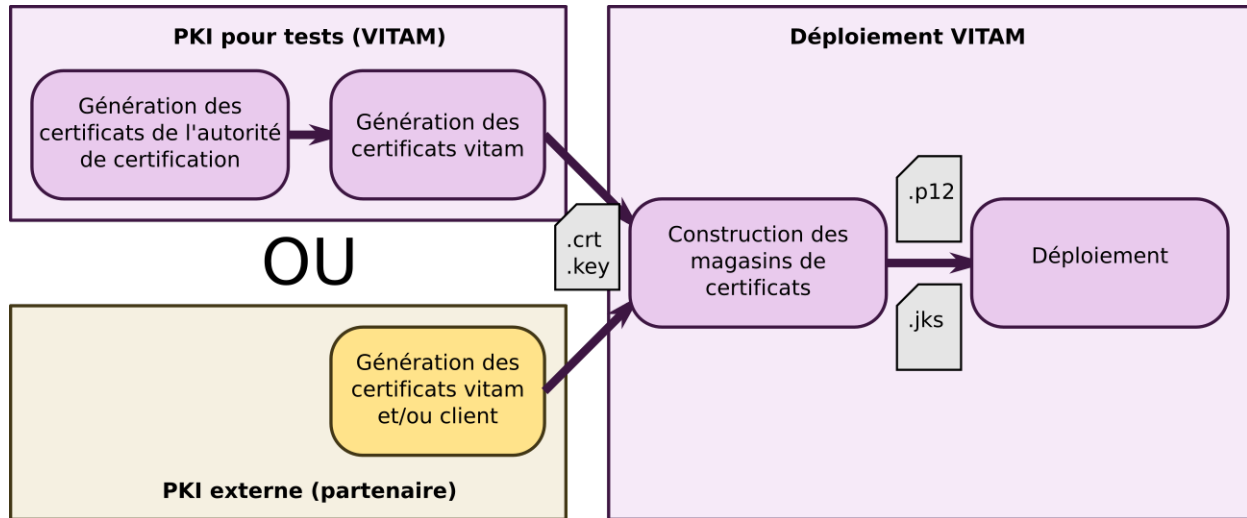
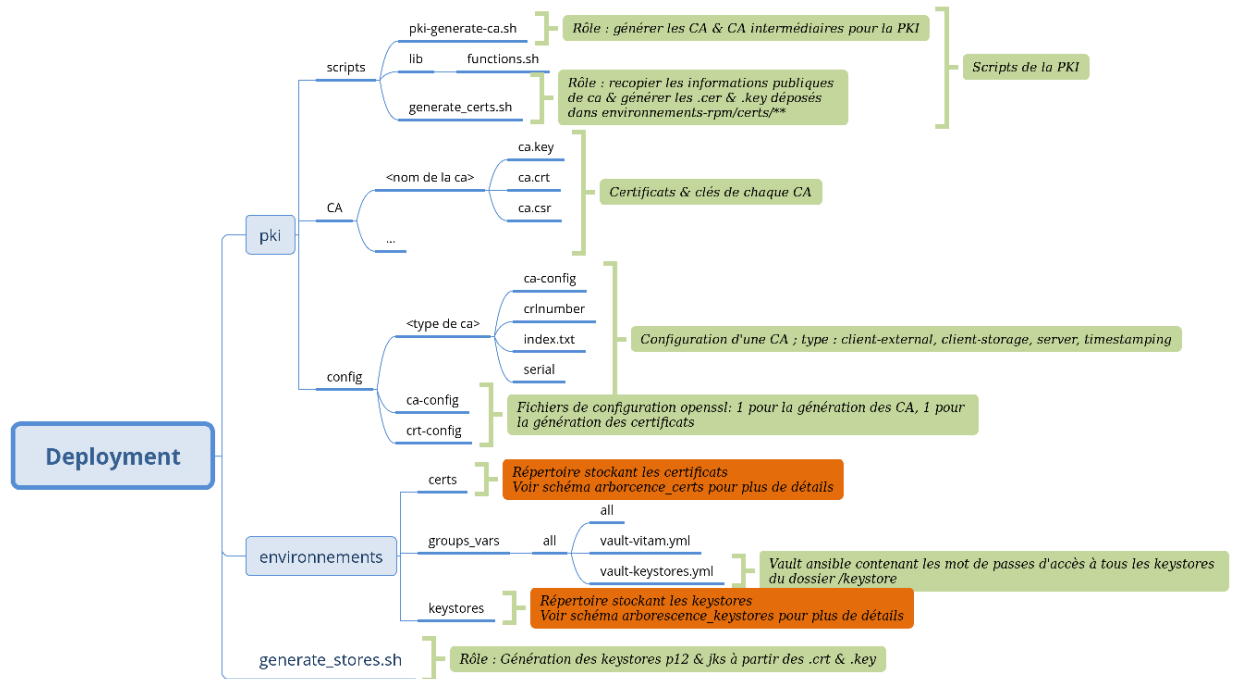


Fig. 1 – Vue d'ensemble de la gestion des certificats au déploiement

8.1.3 Description de l'arborescence de la PKI

Tous les fichiers de gestion de la *PKI* se trouvent dans le répertoire `deployment` de l'arborescence *VITAM* :

- Le sous répertoire `pki` contient les scripts de génération des *CA* & des certificats, les *CA* générées par les scripts, et les fichiers de configuration d'`openssl`
- Le sous répertoire `environments` contient tous les certificats nécessaires au bon déploiement de *VITAM* :
 - certificats publics des *CA*
 - certificats clients, serveurs, de timestamping, et coffre fort contenant les mots de passe des clés privées des certificats (sous-répertoire `certs`)
 - magasins de certificats (`keystores` / `truststores` / `grantedstores`), et coffre fort contenant les mots de passe des magasins de certificats (sous-répertoire `keystores`)
- Le script `generate_stores.sh` génère les magasins de certificats (`keystores`), cf la section *Fonctionnement des scripts de la PKI* (page 109)

Fig. 2 – Vue l'arborescence de la *PKI* Vitam

8.1.4 Description de l'arborescence du répertoire deployment/environments/certs

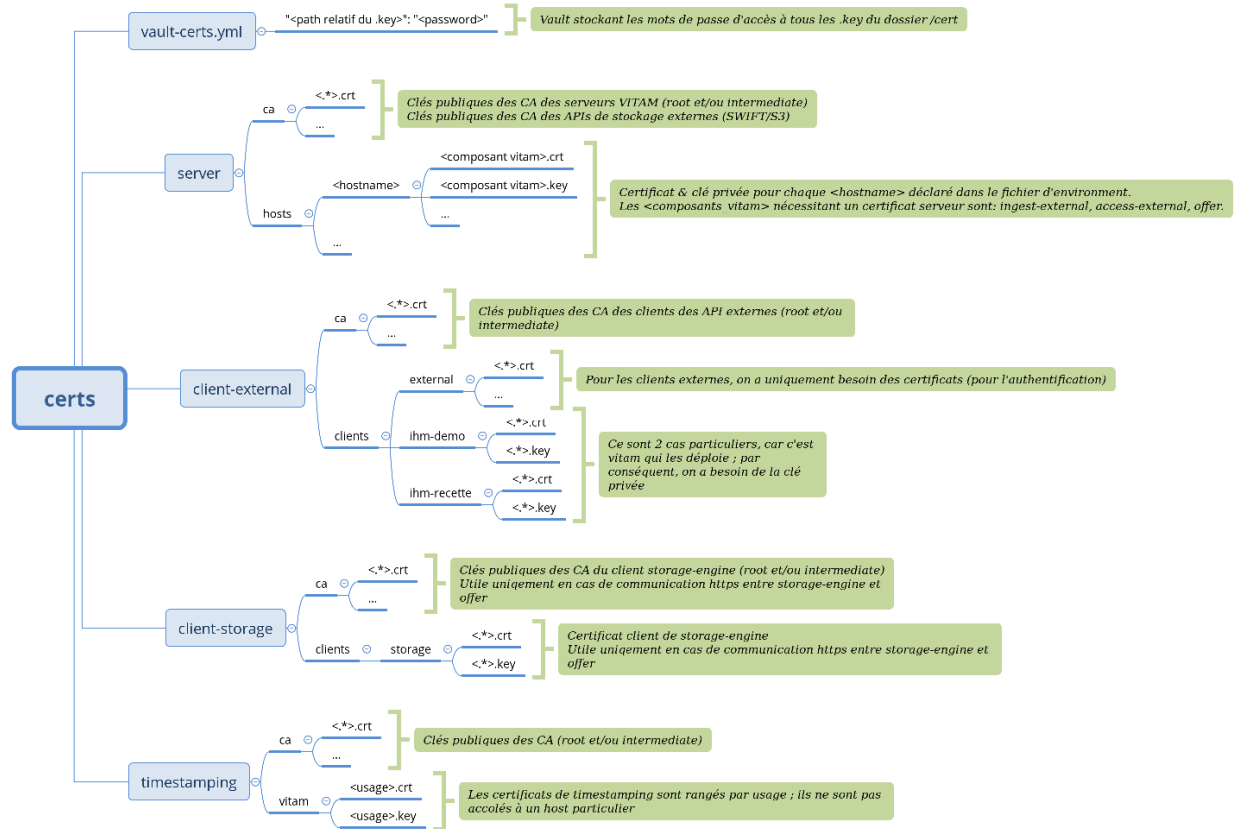


Fig. 3 – Vue détaillée de l'arborescence des certificats

8.1.5 Description de l'arborescence du répertoire deployment/environments/keystores

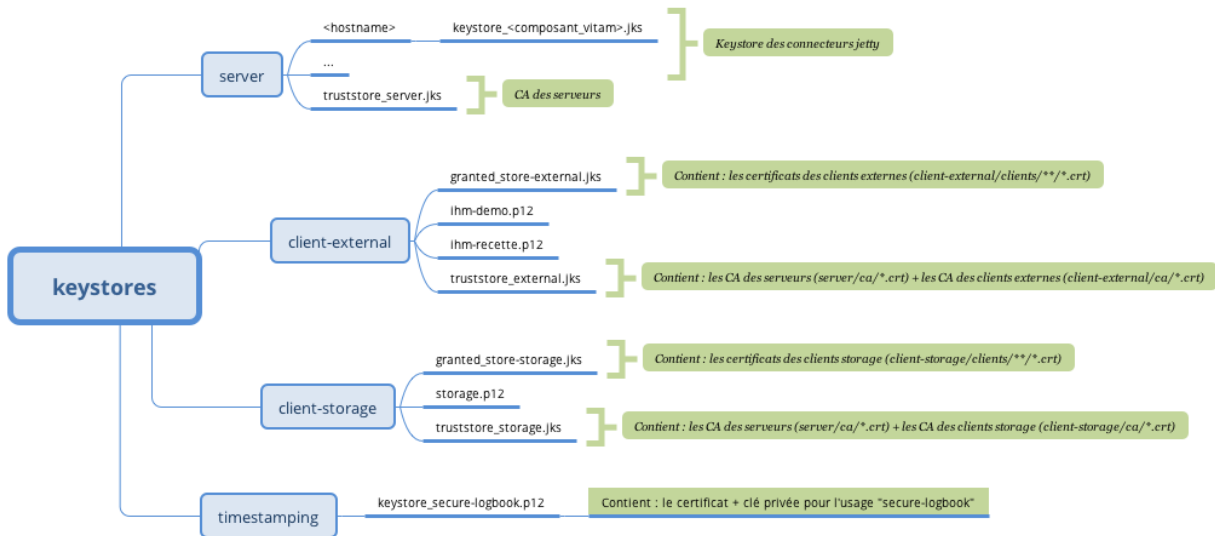


Fig. 4 – Vue détaillée de l'arborescence des keystores

8.1.6 Fonctionnement des scripts de la PKI

La gestion de la *PKI* se fait avec 3 scripts situés dans le répertoire deployment de l'arborescence *VITAM* :

- `pki/scripts/generate_ca.sh` : génère des autorités de certifications (si besoin)
- `pki/scripts/generate_certs.sh` : génère des certificats à partir des autorités de certifications présentes (si besoin)
 - Récupère le mot de passe des clés privées à générer dans le vault `environments/certs/vault-certs.yml`
 - Génère les certificats & les clés privées
- `generate_stores.sh` : génère les magasins de certificats nécessaires au bon fonctionnement de *VITAM*
 - Récupère le mot de passe du magasin indiqué dans `environments/group_vars/all/vault-keystore.yml`
 - Insère les bons certificats dans les magasins qui en ont besoin

Si les certificats sont créés par la *PKI* externe, il faut les positionner dans l'arborescence attendue avec le nom attendu pour certains (cf *l'image ci-dessus* (page 108)).

8.2 Spécificités des certificats

Trois différents types de certificats sont nécessaires et utilisés dans *VITAM* :

- Certificats serveur
- Certificats client
- Certificats d'horodatage

Pour générer des certificats, il est possible de s'inspirer du fichier `pki/config/crt-config`. Il s'agit du fichier de configuration openssl utilisé par la *PKI* de test de *VITAM*. Ce fichier dispose des 3 modes de configurations nécessaires pour générer les certificats de *VITAM* :

- `extension_server` : pour générer les certificats serveur
- `extension_client` : pour générer les certificats client
- `extension_timestamping` : pour générer les certificats d'horodatage

8.2.1 Cas des certificats serveur

8.2.1.1 Généralités

Les services *VITAM* qui peuvent utiliser des certificats serveur sont : `ingest-external`, `access-external`, `offer` (les seuls pouvant écouter en https). Par défaut, `offer` n'écoute pas en https par soucis de performances.

Pour les certificats serveur, il est nécessaire de bien réfléchir au *CN* et *subjectAltName* qui vont être spécifiés. Si par exemple le composant `offer` est paramétré pour fonctionner en https uniquement, il faudra que le *CN* ou un des *subjectAltName* de son certificat corresponde à son nom de service sur consul.

8.2.1.2 Noms DNS des serveurs https VITAM

Les noms *DNS* résolus par *Consul* seront ceux ci :

- `<nom_service>.service.<domaine_consul>` sur le datacenter local
- `<nom_service>.service.<dc_consul>.<domaine_consul>` sur n'importe quel datacenter

Rajouter le nom « Consul » avec le nom du datacenter dedans peut par exemple servir si une installation multi-site de *VITAM* est faite (appels storage -> `offer` inter *DC*)

Les variables pouvant impacter les noms d'hosts *DNS* sur *Consul* sont :

- `consul_domain` dans le fichier `environments/group_vars/all/vitam_vars.yml` -> `<domain_consul>`
- `vitam_site_name` dans le fichier d'inventaire `environments/hosts` (variable globale) -> `<dc_consul>`
- Service `offer` seulement : `offer_conf` dans le fichier d'inventaire `environments/hosts` (différente pour chaque instance du composant `offer`) -> `<nom_service>`

Exemples :

Avec `consul_domain: consul`, `vitam_site_name: dc2`, l'offre `offer-fs-1` sera résolue par

- `offer-fs-1.service.consul` depuis le `dc2`
- `offer-fs-1.service.dc2.consul` depuis n'importe quel *DC*

Avec `consul_domain: preprod.vitam`, `vitam_site_name: dc1`, les composants `ingest-external` et `access-external` seront résolu par

- `ingest-external.service.preprod.vitam` et `access-external.service.preprod.vitam` depuis le *DC* local
- `ingest-external.service.dc1.preprod.vitam` et `access-external.service.dc1.preprod.vitam` depuis n'importe quel *DC*

Avertissement : Si les composants `ingest-external` et `access-external` sont appelés via leur *IP* ou des records *DNS* autres que ceux de *Consul*, il faut également ne pas oublier de les rajouter dans les *subjectAltName*.

8.2.2 Cas des certificats client

Les services qui peuvent utiliser des certificats client sont :

- N'importe quelle application utilisant les !term :API *VITAM* exposées sur ingest-external et access-external
- Le service storage si le service offer est configuré en https
- **Un certificat client nommé vitam-admin-int est obligatoire**
 - Pour déployer *VITAM* (nécessaire pour initialisation du fichier pronom)
 - Pour lancer certains actes d'exploitation

8.2.3 Cas des certificats d'horodatage

Les services logbook et storage utilisent des certificats d'horodatage.

8.2.4 Cas des certificats des services de stockage objets

En cas d'utilisation d'offres de stockage objet avec *VITAM*, si une connexion https est utilisée, il est nécessaire de déposer les *CA* (root et/ou intermédiaire) des serveurs de ces offres de stockage dans le répertoire deployment/environments/certs/server/ca. Cela permettra d'ajouter ces *CA* dans le **truststore** du serveur offer lorsque les **keystores** seront générés.

8.3 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 : régénération 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.

8.4 Ansible & SSH

En fonction de la méthode d'authentification sur les serveurs et d'élévation de privilège, il faut rajouter des options aux lignes de commande ansible. Ces options seront à rajouter pour toutes les commandes ansible du document .

Pour chacune des 3 sections suivantes, vous devez être dans l'un des cas décrits

8.4.1 Authentification du compte utilisateur utilisé pour la connexion SSH

Pour le login du compte utilisateur, voir la section *Informations plate-forme* (page 21).

8.4.1.1 Par clé SSH avec passphrase

Dans le cas d'une authentification par clé avec passphrase, il est nécessaire d'utiliser ssh-agent pour mémoriser la clé privée. Pour ce faire, il faut :

- exécuter la commande `ssh-agent <shell utilisé>` (exemple `ssh-agent /bin/bash`) pour lancer un shell avec un agent de mémorisation de la clé privée associé à ce shell
- exécuter la commande `ssh-add` et renseigner la passphrase de la clé privée

Vous pouvez maintenant lancer les commandes ansible comme décrites dans ce document.

A noter : ssh-agent est un démon qui va stocker les clés privées (déchiffrées) en mémoire et que le client *SSH* va interroger pour récupérer les informations privées pour initier la connexion. La liaison se fait par un socket UNIX présent dans /tmp (avec les droits 600 pour l'utilisateur qui a lancé le ssh-agent). Cet agent disparaît avec le shell qui l'a lancé.

8.4.1.2 Par login/mot de passe

Dans le cas d'une authentification par login/mot de passe, il est nécessaire de spécifier l'option `--ask-pass` (ou `-k` en raccourci) aux commandes ansible ou ansible-playbook de ce document .

Au lancement de la commande `ansible` (ou `ansible-playbook`), il sera demandé le mot de passe

8.4.1.3 Par clé SSH sans passphrase

Dans ce cas, il n'y a pas de paramétrage particulier à effectuer.

8.4.2 Authentification des hôtes

Pour éviter les attaques de type *MitM*, le client *SSH* cherche à authentifier le serveur sur lequel il se connecte. Ceci se base généralement sur le stockage des clés publiques des serveurs auxquels il faut faire confiance (`~/.ssh/known_hosts`).

Il existe différentes méthodes pour remplir ce fichier (vérification humaine à la première connexion, gestion centralisée, *DNSSEC*). La gestion de fichier est hors périmètre *VITAM* mais c'est un pré-requis pour le lancement d'ansible.

8.4.3 Elévation de privilèges

Une fois que l'on est connecté sur le serveur cible, il faut définir la méthode pour accéder aux droits `root`

8.4.3.1 Par sudo avec mot de passe

Dans ce cas, il faut rajouter les options `--ask-sudo-pass`

Au lancement de la commande `ansible` (ou `ansible-playbook`), il sera demandé le mot de passe demandé par `sudo`

8.4.3.2 Par su

Dans ce cas, il faut rajouter les options `--become-method=su --ask-su-pass`

Au lancement de la commande `ansible` (ou `ansible-playbook`), il sera demandé le mot de passe `root`

8.4.3.3 Par sudo sans mot de passe

Il n'y a pas d'option à rajouter (l'élévation par `sudo` est la configuration par défaut)

8.4.3.4 Déjà Root

Dans ce cas, il n'y a pas de paramétrages supplémentaires à effectuer.

Table des figures

1	Cinématique de déploiement	15
2	Vue détaillée des certificats entre le storage et l'offre en multi-site	20
3	Vue détaillée de l'arborescence des certificats	60
1	Vue d'ensemble de la gestion des certificats au déploiement	106
2	Vue l'arborescence de la <i>PKI</i> Vitam	107
3	Vue détaillée de l'arborescence des certificats	108
4	Vue détaillée de l'arborescence des keystores	109

Liste des tableaux

1	Documents de référence VITAM	2
1	Matrice de compétences	7
1	Description des identifiants de référentiels	66
2	Description des règles	67

A

API, 3
AU, 3

B

BDD, 3
BDO, 3

C

CA, 3
CAS, 3
CCFN, 3
CN, 3
COTS, 3
CRL, 3
CRUD, 3

D

DAT, 3
DC, 3
DEX, 3
DIN, 3
DIP, 3
DMV, 3
DNS, 3
DNSSEC, 3
DSL, 3
DUA, 3

E

EAD, 3
EBIOS, 3
ELK, 3

F

FIP, 3

G

GOT, 3

I

IHM, 3
IP, 3
IsaDG, 3

J

JRE, 3
JVM, 4

L

LAN, 4
LFC, 4
LTS, 4

M

M2M, 4
MitM, 4
MoReq, 4

N

NoSQL, 4
NTP, 4

O

OAIS, 4
OOM, 4
OS, 4
OWASP, 4

P

PCA, 4
PDMA, 4
PKI, 4
PRA, 4

R

REST, 4
RGAA, 4
RGI, 4

RPM, [4](#)

S

SAE, [4](#)

SEDA, [4](#)

SGBD, [5](#)

SGBDR, [5](#)

SIA, [5](#)

SIEM, [5](#)

SIP, [5](#)

SSH, [5](#)

Swift, [5](#)

T

TLS, [5](#)

TNA, [5](#)

TNR, [5](#)

TTL, [5](#)

U

UDP, [5](#)

UID, [5](#)

V

VITAM, [5](#)

VM, [5](#)

W

WAF, [5](#)

WAN, [5](#)