



VITAM - Documentation d'installation

Version 1.10.3

VITAM

janv. 04, 2019

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	2
2.3	Glossaire	2
3	Prérequis à l’installation	4
3.1	Expertises requises	4
3.2	Pré-requis plate-forme	4
3.2.1	Base commune	4
3.2.2	PKI	5
3.2.3	Systèmes d’exploitation	5
3.2.3.1	Déploiement sur environnement CentOS	6
3.2.3.2	Déploiement sur environnement Debian	6
3.2.4	Matériel	6
3.3	Récupération de la version	7
3.3.1	Utilisation des dépôts open-source	7
3.3.1.1	Repository pour environnement CentOS	7
3.3.1.2	Repository pour environnement Debian	7
3.3.2	Utilisation du package global d’installation	7
4	Procédures d’installation / mise à jour	9
4.1	Vérifications préalables	9
4.2	Procédures	9
4.2.1	Multi-sites	9
4.2.1.1	Procédure d’installation	9
4.2.1.1.1	vitam_site_name	9
4.2.1.1.2	primary_site	9
4.2.1.1.3	consul_remote_sites	10
4.2.1.1.4	vitam_offers	10
4.2.1.1.5	vitam_strategy	10
4.2.1.1.6	plateforme_secret	11
4.2.1.1.7	consul_encrypt	11

4.2.1.2	Procédure de réinstallation	11
4.2.1.3	Flux entre Storage et offer	12
4.2.1.3.1	Avant la génération des keystores	12
4.2.1.3.2	Après la génération des keystores	13
4.2.2	Configuration du déploiement	13
4.2.2.1	Fichiers de déploiement	13
4.2.2.2	Informations « plate-forme »	13
4.2.2.2.1	Inventaire	13
4.2.2.2.2	Fichier vitam_security.yml	20
4.2.2.2.3	Fichier offers_opts.yml	21
4.2.2.2.4	Fichier cots_vars.yml	22
4.2.2.3	Déclaration des secrets	25
4.2.2.3.1	Cas des extra	28
4.2.3	Gestion des certificats	28
4.2.3.1	Cas 1 : Configuration développement / tests	28
4.2.3.1.1	Procédure générale	29
4.2.3.1.2	Génération des CA par les scripts Vitam	29
4.2.3.1.3	Génération des certificats par les scripts Vitam	29
4.2.3.2	Cas 2 : Configuration production	29
4.2.3.2.1	Procédure générale	29
4.2.3.2.2	Génération des certificats	30
4.2.3.2.2.1	Certificats serveurs	30
4.2.3.2.2.2	Certificat clients	30
4.2.3.2.2.3	Certificats d'horodatage	31
4.2.3.2.3	Intégration de certificats existants	31
4.2.3.2.4	Intégration d'une application externe (cliente)	32
4.2.3.3	Intégration de CA pour une offre swift	32
4.2.3.4	Génération des magasins de certificats	32
4.2.4	Paramétrages supplémentaires	32
4.2.4.1	Tuning JVM	32
4.2.4.2	Paramétrage de l'antivirus (ingest-externe)	33
4.2.4.3	Paramétrage des certificats externes (*-externe)	33
4.2.4.4	Placer « hors Vitam » le composant ihm-demo	34
4.2.4.5	Paramétrage de la centralisation des logs Vitam	34
4.2.4.5.1	Gestion par Vitam	34
4.2.4.5.2	Redirection des logs sur un SIEM tiers	34
4.2.4.6	Fichiers complémentaires	34
4.2.5	Procédure de première installation	46
4.2.5.1	Déploiement	46
4.2.5.1.1	Cas particulier : utilisation de ClamAv en environnement Debian	46
4.2.5.1.2	Fichier de mot de passe	46
4.2.5.1.3	Mise en place des repositories VITAM (optionnel)	46
4.2.5.1.4	Génération des hostvars	47
4.2.5.1.4.1	Cas 1 : Machines avec une seule interface réseau	47
4.2.5.1.4.2	Cas 2 : Machines avec plusieurs interfaces réseau	47
4.2.5.1.4.3	Vérification de la génération des hostvars	48
4.2.5.1.5	Passage des identifiants des référentiels en mode esclave	48
4.2.5.1.6	Durées minimales permettant de contrôler les valeurs saisies	49
4.2.5.1.7	Déploiement	49
4.2.6	Elements extras de l'installation	50
4.2.6.1	Configuration des extra	50
4.2.6.2	Déploiement des extra	52
4.2.6.2.1	ihm-recette	52
4.2.6.2.2	extra complet	52

5	Procédures de mise à jour	53
5.1	Reconfiguration	53
5.1.1	Cas d'une modification du nombre de tenants	53
5.1.2	Cas d'une modification des paramètres JVM	54
6	Post installation	55
6.1	Validation du déploiement	55
6.1.1	Sécurisation du fichier <code>vault_pass.txt</code>	55
6.1.2	Validation manuelle	55
6.1.3	Validation via Consul	56
6.1.4	Post-installation : administration fonctionnelle	56
6.2	Sauvegarde des éléments d'installation	56
6.3	Troubleshooting	56
6.3.1	Erreur au chargement des tableaux de bord Kibana	56
6.4	Retour d'expérience / cas rencontrés	57
6.4.1	Crash <code>rsyslog</code> , code killed, signal : BUS	57
6.4.2	Mongo-express ne se connecte pas à la base de données associée	57
6.4.3	Elasticsearch possède des shard non alloués (état « UNASSIGNED »)	57
6.4.4	Elasticsearch possède des shards non initialisés (état « INITIALIZING »)	58
6.4.5	MongoDB semble lent	58
6.4.6	Les shards de MongoDB semblent mal équilibrés	59
6.4.7	L'importation initiale (profil de sécurité, certificats) retourne une erreur	59
6.4.8	Problème d'ingest et/ou d'accès	59
7	Annexes	60
7.1	Vue d'ensemble de la gestion des certificats	60
7.1.1	Liste des suites cryptographiques & protocoles supportés par Vitam	60
7.1.2	Vue d'ensemble de la gestion des certificats	61
7.1.3	Description de l'arborescence de la PKI	61
7.1.4	Description de l'arborescence du répertoire <code>deployment/environments/certs</code>	63
7.1.5	Description de l'arborescence du répertoire <code>deployment/environments/keystores</code>	64
7.1.6	Fonctionnement des scripts de la PKI	64
7.2	Cycle de vie des certificats	64
7.3	Ansible & ssh	66
7.3.1	Authentification du compte utilisateur utilisé pour la connexion SSH	66
7.3.1.1	Par clé SSH avec passphrase	66
7.3.1.2	Par login/mot de passe	66
7.3.1.3	Par clé SSH sans passphrase	66
7.3.2	Authentification des hôtes	67
7.3.3	Elevation de privilèges	67
7.3.3.1	Par <code>sudo</code> avec mot de passe	67
7.3.3.2	Par <code>su</code>	67
7.3.3.3	Par <code>sudo</code> sans mot de passe	67
7.3.3.4	Déjà Root	67
	Index	70

1.1 Objectif de ce document

Ce document a pour but de permettre de fournir à une équipe d'exploitants de *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 license [CeCILL 2.1](http://www.cecill.info/licences/Licence_CeCILL_V2.1-fr.html)¹ ; la documentation associée (comprenant le présent document) est publiée sous [Licence Ouverte V2.0](https://www.etalab.gouv.fr/wp-content/uploads/2017/04/ETALAB-Licence-Ouverte-v2.0.pdf)².

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
Release notes	

2.2.2 Référentiels externes

2.3 Glossaire

API Application Programming Interface

BDD Base De Données

CA Certificate Authority

¹http://www.cecill.info/licences/Licence_CeCILL_V2.1-fr.html

²<https://www.etalab.gouv.fr/wp-content/uploads/2017/04/ETALAB-Licence-Ouverte-v2.0.pdf>

- COTS** Component Off The Shelves ; 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.
- DAT** Dossier d'Architecture Technique
- DEX** Dossier d'EXploitation
- DIN** Dossier d'Installation
- 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](#)³
- DUA** Durée d'Utilité Administrative
- IHM** Interface Homme Machine
- JRE** Java Runtime Environment ; il s'agit de la machine virtuelle Java permettant d'y exécuter les programmes compilés pour.
- JVM** Java Virtual Machine ; Cf. *JRE*
- 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](#)⁴
- NoSQL** Base de données non-basée sur un paradigme classique des bases relationnelles. [Définition NoSQL](#)⁵
- 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.
- 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](#)⁶
- 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](#)⁷
- RPM** Red Hat Package Manager ; il s'agit du format de packets logiciels nativement utilisé par les distributions CentOS (entre autres)
- SAE** Système d'Archivage Électronique
- SEDA** Standard d'Échange de Données pour l'Archivage
- SIA** Système d'Informations Archivistique
- SIP** Submission Information Package
- TNR** Tests de Non-Régression
- VITAM** Valeurs Immatérielles Transférées aux Archives pour Mémoire

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

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

Prérequis à l'installation

3.1 Expertises requises

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

- connaissance d'ansible en tant qu'outil de déploiement automatisé ;
- connaissance de Consul en tant qu'outil de découverte de services ;
- maîtrise de MongoDB et ElasticSearch par les administrateurs de bases de données.

3.2 Pré-requis plate-forme

Les pré-requis suivants sont nécessaires :

3.2.1 Base commune

- Tous les serveurs hébergeant la solution *VITAM* doivent être synchronisés sur un serveur de temps (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 **2.7** minimale et conseillée ; se référer à la [documentation ansible](http://docs.ansible.com/ansible/latest/intro_installation.html)⁸ pour la procédure d'installation)
 - openssh-clients (client SSH utilisé par ansible)

⁸http://docs.ansible.com/ansible/latest/intro_installation.html

- java-1.8.0-openjdk & 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 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](#)⁹ pour les procédures de connexion aux machines-cibles depuis le serveur ansible.

Prudence : Les IP des machines sur lesquelles la solution Vitam sera installée ne doivent pas changer d'IP au cours du temps, en cas de changement d'IP, la plateforme ne pourra plus fonctionner.

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.

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 *VITAM*. En cas d'utilisation d'un autre type, s'assurer que le filesystem possède/gère bien l'option `user_xattr`.

3.2.2 PKI

La solution 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 60) 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 VITAM.

Voir aussi :

Veuillez vous référer à la section *Vue d'ensemble de la gestion des certificats* (page 60) 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 8 (jessie)

SELinux doit être configuré en mode `permissive` ou `disabled`.

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

⁹http://docs.ansible.com/ansible/latest/intro_getting_started.html

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 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) 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)

3.2.3.2 Déploiement sur environnement Debian

- Disposer d'une plate-forme Linux Debian « jessie » 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 jessie-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)

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 :

- storage-offer-default
- solution de centralisation des logs (elasticsearch)
- workspace
- worker (temporairement, lors du traitement de chaque fichier à traiter)
- elasticsearch des données Vitam

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

3.3 Récupération de la version

3.3.1 Utilisation des dépôts open-source

Les scripts de déploiement de VITAM sont disponibles dans le dépôt github [VITAM¹⁰](#) , dans le répertoire deployment.

Les binaires de VITAM sont disponibles sur un dépôt vitam public indiqué ci-dessous par type de package ; ces dépôts doivent être correctement configurés sur la plate-forme cible avant toute installation.

3.3.1.1 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.

3.3.1.2 Repository pour environnement Debian

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

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

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

3.3.2 Utilisation du package global d'installation

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

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

Le package global d'installation contient :

- le package proprement dit
- la release notes
- les empreintes de contrôle

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

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.

Procédures d'installation / mise à jour

4.1 Vérifications préalables

Tous les serveurs cibles doivent avoir accès aux dépôts de binaires contenant les paquets des logiciels 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 Multi-sites

4.2.1.1 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 est nécessaire de paramétrer correctement un certain de variables ansible pour chaque site :

4.2.1.1.1 vitam_site_name

Fichier : `deployment/environments/hosts.my_env`

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

4.2.1.1.2 primary_site

Fichier : `deployment/environments/hosts.my_env`

Cette variable sert à définir si le site est primaire ou non. Sur un vitam installé en mode multi site, un seul des sites doit avoir la valeur à true. Sur les sites secondaires (`primary_site : false`), certains composants ne seront pas démarrés

et apparaîtront donc en orange sur consul. Certains timers systemd seront en revanche démarrés pour mettre en place la reconstruction au fil de l'eau par exemple.

4.2.1.1.3 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 qu'on est en train de configurer.

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"]
```

4.2.1.1.4 vitam_offers

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

Cette variable référence toutes les offres disponibles sur la totalité des sites vitam.

Exemple :

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

4.2.1.1.5 vitam_strategy

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

Cette variable référence la stratégie de stockage sur le site courant. 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. 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
    vitam_site_name: site2
  - name: offer-fs-3
    referent: false
    vitam_site_name: site3
```

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
```

4.2.1.1.6 plateforme_secret

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

Cette variable stocke le secret de plateforme qui doit être commun entre tous les composants vitam de tous les sites. La valeur doit donc être la même entre chaque site.

4.2.1.1.7 consul_encrypt

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

Cette variable stocke le secret de plateforme qui doit être commun entre tous les consul de tous les sites. La valeur doit donc être la même entre chaque site.

4.2.1.2 Procédure de réinstallation

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

Ensuite :

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

4.2.1.3 Flux entre Storage et offer

Dans le cas d'appel en https entre les composants Storage et Offer, il convient également de rajouter :

- 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

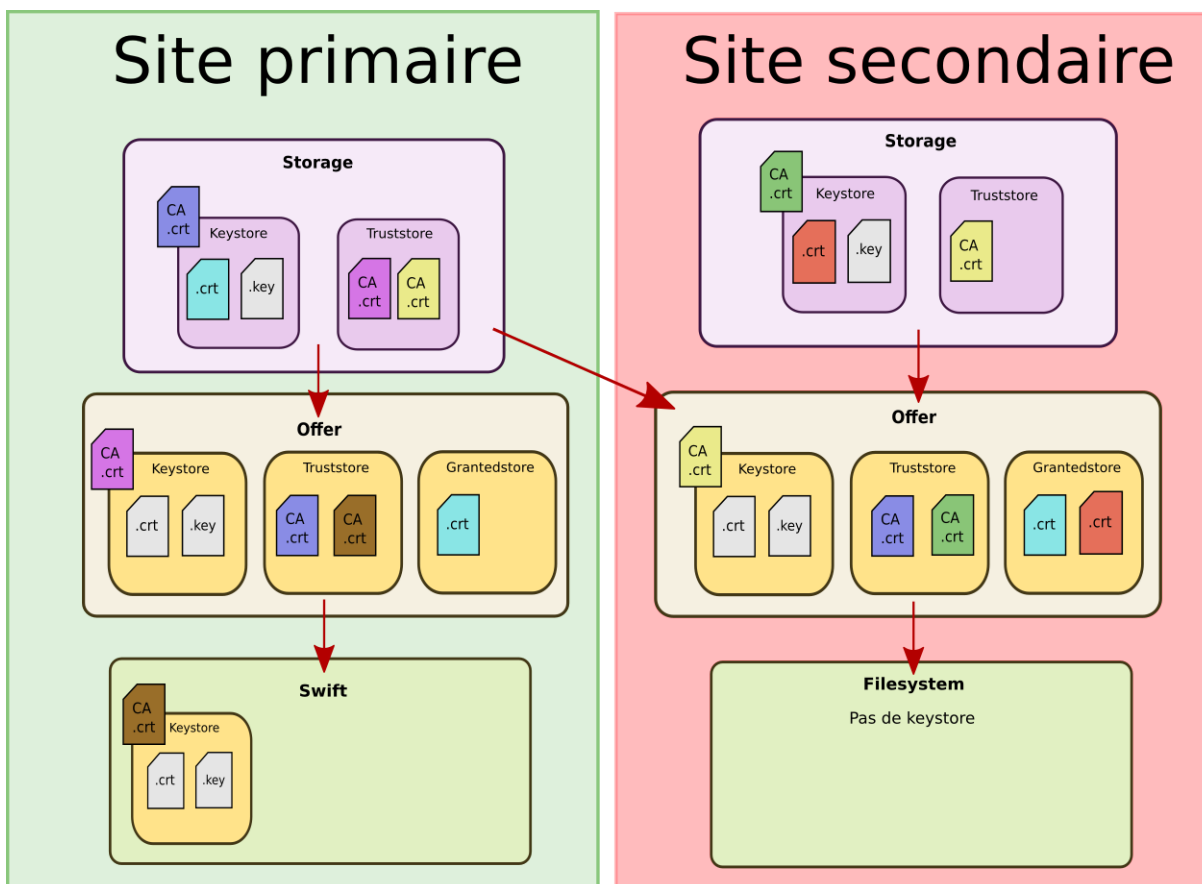


Fig. 1 – Vue détaillée des certificats entre le storage et l'offre en multi-site

Il est possible de procéder de 2 manières différentes :

4.2.1.3.1 Avant la génération des keystores

Avertissement : Pour toutes les copies de certificats indiquées ci-dessous, il est important de ne jamais en 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/*` 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/*`

4.2.1.3.2 Après la génération des keystores

Via le script `deployment/generate_stores.sh`, il convient donc de rajouter 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>`

4.2.2 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*.

4.2.2.1 Fichiers de déploiement

Les fichiers de déploiement sont disponibles dans la version VITAM livrée dans le sous-répertoire `deployment`. Concernant l'installation, ils consistent en 2 parties :

- les playbook 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'exemple sont disponibles dans le sous-répertoire `environments`. Cette arborescence est valable pour le déploiement d'un environnement, et est à dupliquer lors de l'installation d'environnements ultérieurs. Les fichiers qui y sont contenus doivent être adaptés avant le déploiement, comme il est expliqué dans les paragraphes suivants.

4.2.2.2 Informations « plate-forme »

4.2.2.2.1 Inventaire

Pour configurer le déploiement, il est nécessaire de créer dans le répertoire `environments` un nouveau fichier d'inventaire (dans la suite, ce fichier sera communément appelé `hosts.<environnement>`). Ce fichier doit être basé sur la structure présente dans le fichier `hosts.example` (et notamment respecter scrupuleusement l'arborescence des groupes ansible) ; les commentaires dans ce fichier donnent 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

```

(suite sur la page suivante)

(suite de la page précédente)

```

7 reverse
8 library
9 hosts-dev-tools
10 ldap
11
12
13 ##### Tests environments specifics #####
14
15 # EXTRA : Front reverse-proxy (test environments ONLY) ; add machine name after
16 [reverse]
17 # optional : after machine, if this machine is different from VITAM machines, you can
18 ↪ specify another become user
19 # Example
20 # vitam-centos-01.vitam ansible_ssh_user=centos
21
22 ##### Extra VITAM applications #####
23
24 [ldap] # Extra : OpenLDAP server
25 # LDAP server !!! NOT FOR PRODUCTION !!! Test only
26
27 [library]
28 # TODO: Put here servers where this service will be deployed : library
29
30 [hosts-dev-tools]
31 # TODO: Put here servers where this service will be deployed : mongo-express,
32 ↪ elasticsearch-head
33
34 [elasticsearch:children] # EXTRA : elasticsearch
35 hosts-elasticsearch-data
36 hosts-elasticsearch-log
37
38 ##### VITAM services #####
39
40 # Group definition ; DO NOT MODIFY
41 [vitam:children]
42 zone-external
43 zone-access
44 zone-applicative
45 zone-storage
46 zone-data
47 zone-admin
48
49 ##### Zone externe
50
51 [zone-external:children]
52 hosts-ihm-demo
53 hosts-cerebro
54 hosts-ihm-recette
55
56
57 [hosts-ihm-demo]
58 # TODO: Put here servers where this service will be deployed : ihm-demo
59 # If you don't need consul for ihm-demo, you can set this var after each hostname :
60 # consul_disabled=true
61

```

(suite sur la page suivante)

(suite de la page précédente)

```

62 [hosts-ihm-recette]
63 # TODO: Put here servers where this service will be deployed : ihm-recette (extra_
    ↳ feature)
64
65 [hosts-cerebro]
66 # TODO: Put here servers where this service will be deployed : vitam-elasticsearch-
    ↳ cerebro
67
68
69 ##### Zone access
70
71 # Group definition ; DO NOT MODIFY
72 [zone-access:children]
73 hosts-ingest-external
74 hosts-access-external
75
76 [hosts-ingest-external]
77 # TODO: Put here servers where this service will be deployed : ingest-external
78
79
80 [hosts-access-external]
81 # TODO: Put here servers where this service will be deployed : access-external
82
83
84 ##### Zone applicative
85
86 # Group definition ; DO NOT MODIFY
87 [zone-applicative:children]
88 hosts-ingest-internal
89 hosts-processing
90 hosts-batch-report
91 hosts-worker
92 hosts-access-internal
93 hosts-metadata
94 hosts-functional-administration
95 hosts-logbook
96 hosts-workspace
97 hosts-storage-engine
98 hosts-security-internal
99
100 [hosts-security-internal]
101 # TODO: Put here servers where this service will be deployed : security-internal
102
103
104 [hosts-logbook]
105 # TODO: Put here servers where this service will be deployed : logbook
106
107
108 [hosts-workspace]
109 # TODO: Put here servers where this service will be deployed : workspace
110
111
112 [hosts-ingest-internal]
113 # TODO: Put here servers where this service will be deployed : ingest-internal
114
115
116 [hosts-access-internal]

```

(suite sur la page suivante)

(suite de la page précédente)

```

117 # TODO: Put here servers where this service will be deployed : access-internal
118
119
120 [hosts-metadata]
121 # TODO: Put here servers where this service will be deployed : metadata
122
123
124 [hosts-functional-administration]
125 # TODO: Put here servers where this service will be deployed : functional-
    ↳ administration
126
127
128 [hosts-processing]
129 # TODO: Put here servers where this service will be deployed : processing
130
131
132 [hosts-storage-engine]
133 # TODO: Put here servers where this service will be deployed : storage-engine
134
135 [hosts-batch-report]
136 # TODO: Put here servers where this service will be deployed : batch-report
137
138 [hosts-worker]
139 # TODO: Put here servers where this service will be deployed : worker
140 # Optional parameter after each host : vitam_worker_capacity=<integer> ; please refer_
    ↳ to your infrastructure for defining this number ; default is ansible_processor_
    ↳ vcpus value (cpu number in /proc/cpuinfo file)
141
142
143 ##### Zone storage
144
145 [zone-storage:children] # DO NOT MODIFY
146 hosts-storage-offer-default
147 hosts-mongodb-offer
148
149 [hosts-storage-offer-default]
150 # TODO: Put here servers where this service will be deployed : storage-offer-default
151 # LIMIT : only 1 offer per machine and 1 machine per offer
152 # Mandatory param for each offer is offer_conf and points to offer_opts.yml & vault-
    ↳ vitam.yml (with same tree)
153 # hostname-offre-1.vitam offer_conf=offer-swift-1
154 # for filesystem
155 # hostname-offre-2.vitam offer_conf=offer-fs-1
156
157 [hosts-mongodb-offer:children]
158 hosts-mongos-offer
159 hosts-mongoc-offer
160 hosts-mongod-offer
161
162 [hosts-mongos-offer]
163 # WARNING : DO NOT COLLOCATE WITH [hosts-mongos-data]
164 # TODO: put here servers where this service will be deployed : mongos cluster for_
    ↳ storage offers
165 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the_
    ↳ offer_conf configuration)
166 # Example (for a more complete one, see the one in the group hosts-mongos-data) :
167 # vitam-mongo-swift-offer-01 mongo_cluster_name=offer-swift-1

```

(suite sur la page suivante)

(suite de la page précédente)

```

168 # vitam-mongo-swift-offer-02    mongo_cluster_name=offer-swift-1
169 # vitam-mongo-fs-offer-01       mongo_cluster_name=offer-fs-1
170 # vitam-mongo-fs-offer-02       mongo_cluster_name=offer-fs-1
171
172 [hosts-mongoc-offer]
173 # WARNING : DO NOT COLLOCATE WITH [hosts-mongoc-data]
174 # TODO: put here servers where this service will be deployed : mongoc cluster for
↳ storage offers
175 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the
↳ offer_conf configuration)
176 # Optional param : mandatory for 1 node of the shard, some init commands will be
↳ executed on it
177 # Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add
↳ this paramter on a mongo_rs_bootstrap node
178 # Example :
179 # vitam-mongo-swift-offer-01    mongo_cluster_name=offer-swift-1
↳ mongo_rs_bootstrap=true
180 # vitam-mongo-swift-offer-02    mongo_cluster_name=offer-swift-1
181 # vitam-swift-offer             mongo_cluster_name=offer-swift-1
↳ mongo_arbiter=true
182 # vitam-mongo-fs-offer-01       mongo_cluster_name=offer-fs-1
↳ mongo_rs_bootstrap=true
183 # vitam-mongo-fs-offer-02       mongo_cluster_name=offer-fs-1
184 # vitam-fs-offer               mongo_cluster_name=offer-fs-1
↳ mongo_arbiter=true
185
186 [hosts-mongod-offer]
187 # WARNING : DO NOT COLLOCATE WITH [hosts-mongod-data]
188 # TODO: put here servers where this service will be deployed : mongod cluster for
↳ storage offers
189 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the
↳ offer_conf configuration)
190 # Mandatory param : id of the current shard, increment by 1 from 0 to n
191 # Optional param : mandatory for 1 node of the shard, some init commands will be
↳ executed on it
192 # Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add
↳ this paramter on a mongo_rs_bootstrap node
193 # Example :
194 # vitam-mongo-swift-offer-01    mongo_cluster_name=offer-swift-1    mongo_shard_id=0
↳
↳ mongo_rs_bootstrap=true
195 # vitam-mongo-swift-offer-02    mongo_cluster_name=offer-swift-1    mongo_shard_id=0
196 # vitam-swift-offer            mongo_cluster_name=offer-swift-1    mongo_shard_id=0
↳
↳ mongo_arbiter=true
197 # vitam-mongo-fs-offer-01       mongo_cluster_name=offer-fs-1       mongo_shard_id=0
↳
↳ mongo_rs_bootstrap=true
198 # vitam-mongo-fs-offer-02       mongo_cluster_name=offer-fs-1       mongo_shard_id=0
199 # vitam-fs-offer               mongo_cluster_name=offer-fs-1       mongo_shard_id=0
↳
↳ mongo_arbiter=true
200
201 ##### Zone data
202
203 # Group definition ; DO NOT MODIFY
204 [zone-data:children]
205 hosts-elasticsearch-data
206 hosts-mongodb-data
207
208 [hosts-elasticsearch-data]

```

(suite sur la page suivante)

(suite de la page précédente)

```

209 # TODO: Put here servers where this service will be deployed : elasticsearch-data_
    ↳ cluster
210 # 2 params available for huge environments (parameter to be declared after each_
    ↳ server) :
211 #     is_data=true/false
212 #     is_master=true/false
213 #     other options are not handled yet
214 # defaults are set to true, if undefined. If defined, at least one server MUST be is_
    ↳ data=true
215 # Examples :
216 # server1 is_master=true is_data=false
217 # server2 is_master=false is_data=true
218 # More explanation here : https://www.elastic.co/guide/en/elasticsearch/reference/5.6/
    ↳ modules-node.html
219
220
221 # Group definition ; DO NOT MODIFY
222 [hosts-mongodb-data:children]
223 hosts-mongos-data
224 hosts-mongoc-data
225 hosts-mongod-data
226
227 [hosts-mongos-data]
228 # WARNING : DO NOT COLLOCATE WITH [hosts-mongos-offer]
229 # TODO: Put here servers where this service will be deployed : mongos cluster
230 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
231 # Example :
232 # vitam-mdbs-01     mongo_cluster_name=mongo-data
233 # vitam-mdbs-01     mongo_cluster_name=mongo-data
234 # vitam-mdbs-01     mongo_cluster_name=mongo-data
235
236 [hosts-mongoc-data]
237 # WARNING : DO NOT COLLOCATE WITH [hosts-mongoc-offer]
238 # TODO: Put here servers where this service will be deployed : mongoc cluster
239 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
240 # Optional param : mandatory for 1 node of the shard, some init commands will be_
    ↳ executed on it
241 # Example :
242 # vitam-mdbc-01     mongo_cluster_name=mongo-data                mongo_rs_
    ↳ bootstrap=true
243 # vitam-mdbc-01     mongo_cluster_name=mongo-data
244 # vitam-mdbc-01     mongo_cluster_name=mongo-data
245
246 [hosts-mongod-data]
247 # WARNING : DO NOT COLLOCATE WITH [hosts-mongod-offer]
248 # TODO: Put here servers where this service will be deployed : mongod cluster
249 # Each replica_set should have an odd number of members (2n + 1)
250 # Reminder: For Vitam, one mongodb shard is using one replica_set
251 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
252 # Mandatory param : id of the current shard, increment by 1 from 0 to n
253 # Optional param : mandatory for 1 node of the shard, some init commands will be_
    ↳ executed on it
254 # Example:
255 # vitam-mdbd-01     mongo_cluster_name=mongo-data     mongo_shard_id=0  mongo_rs_
    ↳ bootstrap=true
256 # vitam-mdbd-02     mongo_cluster_name=mongo-data     mongo_shard_id=0
257 # vitam-mdbd-03     mongo_cluster_name=mongo-data     mongo_shard_id=0

```

(suite sur la page suivante)

(suite de la page précédente)

```

258 # vitam-mdbd-04  mongo_cluster_name=mongo-data  mongo_shard_id=1  mongo_rs_
    ↳bootstrap=true
259 # vitam-mdbd-05  mongo_cluster_name=mongo-data  mongo_shard_id=1
260 # vitam-mdbd-06  mongo_cluster_name=mongo-data  mongo_shard_id=1
261
262 ##### Zone admin
263
264 # Group definition ; DO NOT MODIFY
265 [zone-admin:children]
266 hosts-consul-server
267 hosts-kibana-data
268 log-servers
269 hosts-elasticsearch-log
270
271 [hosts-consul-server]
272 # TODO: Put here servers where this service will be deployed : consul
273
274 [hosts-kibana-data]
275 # TODO: Put here servers where this service will be deployed : kibana (for data_
    ↳cluster)
276
277 [log-servers:children]
278 hosts-kibana-log
279 hosts-logstash
280
281
282 [hosts-kibana-log]
283 # TODO: Put here servers where this service will be deployed : kibana (for log_
    ↳cluster)
284
285 [hosts-logstash]
286 # TODO: Put here servers where this service will be deployed : logstash
287
288
289 [hosts-elasticsearch-log]
290 # TODO: Put here servers where this service will be deployed : elasticsearch-log_
    ↳cluster
291
292 ##### Global vars #####
293
294 [hosts:vars]
295
296 # =====
297 # VITAM
298 # =====
299
300 # Declare user for ansible on target machines
301 ansible_ssh_user=
302 # Can target user become as root ? ; true is required by VITAM (usage of a sudoer is_
    ↳mandatory)
303 ansible_become=true
304
305 # Related to Consul ; apply in a table your DNS server(s)
306 # Example : dns_servers=["8.8.8.8","8.8.4.4"]
307 dns_servers=
308
309 # Vitam tenants to create

```

(suite sur la page suivante)

(suite de la page précédente)

```

310 vitam_tenant_ids=[0,1,2]
311 vitam_tenant_admin=1
312
313 ### Logback configuration ###
314 # Days before deleting logback log files (java & access logs for vitam components)
315 days_to_delete_logback_logfiles=
316
317 # Define local Consul datacenter name
318 vitam_site_name=prod-dcl
319 # EXAMPLE : vitam_site_name = prod-dcl
320 # check whether on primary site (true) or secondary (false)
321 primary_site=true
322
323
324 # =====
325 # EXTRA
326 # =====
327 # Environment (defines title in extra on reverse homepage). Variable is DEPRECATED !
328 #environnement=
329
330 ### vitam-itest repository ###
331 vitam_tests_branch=master
332 vitam_tests_gitrepo_protocol=
333 vitam_tests_gitrepo_baseurl=
334 vitam_tests_gitrepo_url=
335
336 # Used when VITAM is behind a reverse proxy (provides configuration for reverse proxy,
337 ↪ && displayed in header page)
338 vitam_reverse_external_dns=
339 # For reverse proxy use
340 reverse_proxy_port=80
341 # http_proxy env var to use ; has to be declared even if empty
342 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 un minimum de 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 ».

4.2.2.2 Fichier vitam_security.yml

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


```

1  ---
2
3  # Business vars
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-
   ↳external/clients
9  # vitam-admin-int is mandatory for internal use (PRONOM upload)
10 admin_context_certs: [ "ihm-demo/ihm-demo.crt", "ihm-recette/ihm-recette.crt",
   ↳"reverse/reverse.crt", "vitam-admin-int/vitam-admin-int.crt" ]
11 # Indicate here all the personal certificates relative paths under {{inventory_dir}}/
   ↳certs/client-vitam-users/clients
12 admin_personal_certs: [ "userOK.crt" ]
13
14 # Admin security profile name
15 admin_security_profile: "admin-security-profile"
16
17 admin_basic_auth_user: "adminUser"

```

4.2.2.2.3 Fichier offers_opts.yml

Enfin, la déclaration des configuration des offres de stockage est réalisée dans le fichier environments / group_vars/all/offers_opts.yml :

```

1  # This list is ordered. It can and has to be completed if more offers are
   ↳necessary
2  # Strategy order (1st has to be the preferred one)
3  vitam_strategy:
4    - name: offer-fs-1
5      referent: true
6    # vitam_site_name: prod-dc2
7    # - name: offer-swift-1
8    # Example :
9    # - name: distant
10   # referent: true
11   # vitam_site_name: distant-dc2
12
13  # DON'T forget to add associated passwords in vault-vitam.yml with same tree
   ↳when using provider openstack-swift*
14  # ATTENTION !!! Each offer has to have a distinct name, except for clusters
   ↳binding a same physical storage
15  # WARNING : for offer names, please only use [a-z][a-z0-9-]* pattern
16  vitam_offers:
17    offer-fs-1:
18      # param can be filesystem-hash (recomended) or filesystem (not
   ↳recomended)
19      provider: filesystem-hash
20    offer-swift-1:
21      # provider : openstack-swift for v1 or openstack-swift-v3 for v3
22      provider: openstack-swift-v3
23      # swiftKeystoneAuthUrl : URL de connexion à keystone
24      swiftKeystoneAuthUrl: https://openstack-hostname:port/auth/1.0
25      # swiftDomain : domaine OpenStack dans lequel l'utilisateur est
   ↳enregistré

```

(suite sur la page suivante)

(suite de la page précédente)

```

26  swiftDomain: domaine
27  # swiftUser : identifiant de l'utilisateur
28  swiftUser: utilisateur
29  # swiftPassword: has to be set in vault-vitam.yml (encrypted) with same_
↪structure => DO NOT COMMENT OUT
30  # swiftProjectName : nom du projet openstack
31  swiftProjectName: monTenant
32  # swiftUrl: optional variable to force the swift URL
33  # swiftUrl: https://swift-hostname:port/swift/v1
34
35  # example_swift_v1:
36  #   provider: openstack-swift
37  #   swiftKeystoneAuthUrl: https://keystone/auth/1.0
38  #   swiftDomain: domaine
39  #   swiftUser: user
40  #   swiftPassword: has to be set in vault-vitam.yml (encrypted) with same_
↪structure => DO NOT COMMENT OUT
41  # example_swift_v3:
42  #   provider: openstack-swift-v3
43  #   swiftKeystoneAuthUrl: https://keystone/v3
44  #   swiftDomain: domaine
45  #   swiftUser: user
46  #   swiftPassword: has to be set in vault-vitam.yml (encrypted) with same_
↪structure => DO NOT COMMENT OUT
47  #   swiftProjectName: monTenant
48  #   projectName: monTenant

```

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.

Avertissement : La cohérence entre l'inventaire et la section `vitam_strategy` 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'offre 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.

4.2.2.2.4 Fichier `cots_vars.yml`

Dans le cas du choix du *COTS* d'envoi des messages syslog dans logastsh, il est possible de choisir entre `syslog-ng` et `rsyslog` dans le fichier `environments/group_vars/all/cots_vars.yml` :

```

1  ---
2
3  consul:

```

(suite sur la page suivante)

(suite de la page précédente)

```

4     dns_port: 53
5
6     consul_remote_sites:
7         # wan contains the wan addresses of the consul server instances of the
8         ↪ external vitam sites
9         # Exemple, if our local dc is dc1, we will need to set dc2 & dc3 wan
10        ↪ conf:
11        # - dc2:
12        #   wan: ["10.10.10.10", "1.1.1.1"]
13        # - dc3:
14        #   wan: ["10.10.10.11", "1.1.1.1"]
15
16     elasticsearch:
17         log:
18             host: "elasticsearch-log.service.{{ consul_domain }}"
19             port_http: "9201"
20             port_tcp: "9301"
21             groupe: "log"
22             baseuri: "elasticsearch-log"
23             cluster_name: "elasticsearch-log"
24             https_enabled: false
25             # default index template
26             index_templates:
27                 default:
28                     shards: 1
29                     replica: 1
30
31         data:
32             host: "elasticsearch-data.service.{{ consul_domain }}"
33             # default is 0.1 (10%) and should be quite enough in most cases
34             #index_buffer_size_ratio: "0.15"
35             port_http: "9200"
36             port_tcp: "9300"
37             groupe: "data"
38             baseuri: "elasticsearch-data"
39             cluster_name: "elasticsearch-data"
40             https_enabled: false
41             # default index template
42             index_templates:
43                 default:
44                     shards: 10
45                     replica: 2
46
47     mongodb:
48         mongos_port: 27017
49         mongoc_port: 27018
50         mongod_port: 27019
51         mongo_authentication: "true"
52         host: "mongos.service.{{ consul_domain }}"
53
54     logstash:
55         host: "logstash.service.{{ consul_domain }}"
56         user: logstash
57         port: 10514
58         rest_port: 20514
59
60     # Curator units: days
61     curator:

```

(suite sur la page suivante)

(suite de la page précédente)

```

59 log:
60     metrics:
61         close: 5
62         delete: 30
63     logstash:
64         close: 5
65         delete: 30
66     metricbeat:
67         close: 5
68         delete: 30
69     packetbeat:
70         close: 5
71         delete: 30
72
73 kibana:
74     header_value: "reporting"
75     log:
76         baseuri: "kibana_log"
77         api_call_timeout: 120
78         groupe: "log"
79         port: 5601
80         default_index_pattern: "logstash-vitam*"
81         # default shards & replica
82         shards: 5
83         replica: 1
84         # pour index logstash-*
85     metrics:
86         shards: 5
87         replica: 1
88         # pour index metrics-vitam-*
89     logs:
90         shards: 5
91         replica: 1
92         # pour index metricbeat-*
93     metricbeat:
94         shards: 5 # must be a factor of 30
95         replica: 1
96     data:
97         baseuri: "kibana_data"
98         # OMA : bugdette : api_call_timeout is used for retries ; should_
99         ↪ceate a separate variable rather than this one
100         api_call_timeout: 120
101         groupe: "data"
102         port: 5601
103         default_index_pattern: "logbookoperation_*"
104         # index template for .kibana
105         shards: 1
106         replica: 1
107
108 syslog:
109     # value can be syslog-ng or rsyslog
110     name: "rsyslog"
111
112 cerebro:
113     baseuri: "cerebro"
114     port: 9000

```

(suite sur la page suivante)

(suite de la page précédente)

```

115 siegfried:
116     port: 19000
117
118 clamav:
119     port: 3310
120     # frequency freshclam for database update per day (from 0 to 24 - 24_
↪ meaning hourly check)
121     db_update_periodicity: 1
122
123 mongo_express:
124     baseuri: "mongo-express"
125
126 ldap_authentication:
127     ldap_protocol: "ldap"
128     ldap_server: "{% if groups['ldap']|length > 0 %}{{ groups['ldap']|first }
↪ }{% endif %}"
129     ldap_port: "389"
130     ldap_base: "dc=programmevitam,dc=fr"
131     ldap_login: "cn=Manager,dc=programmevitam,dc=fr"
132     uid_field: "uid"
133     ldap_userDn_Template: "uid={0},ou=people,dc=programmevitam,dc=fr"
134     ldap_group_request: "(&(objectClass=groupOfNames)(member={0}))"
135     ldap_admin_group: "cn=admin,ou=groups,dc=programmevitam,dc=fr"
136     ldap_user_group: "cn=user,ou=groups,dc=programmevitam,dc=fr"
137     ldap_guest_group: "cn=guest,ou=groups,dc=programmevitam,dc=fr"

```

Il faut alors modifier la valeur de la directive `syslog.name` ; la valeur par défaut est `rsyslog`.

4.2.2.3 Déclaration des secrets

Avertissement : Cette section décrit des fichiers contenant des données sensibles ; il convient de sécuriser ces fichiers avec un mot de passe « fort ». 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 vault 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 ; leur contenu est donc à modifier avant tout déploiement :

- Le fichier `environments /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: mongogo
14     admin:
15       user: vitamdb-admin
16       password: azerty
17     localadmin:
18       user: vitamdb-localadmin
19       password: qwerty
20   metadata:
21     user: metadata
22     password: azerty1
23   logbook:
24     user: logbook
25     password: azerty2
26   report:
27     user: report
28     password: azerty5
29   functionalAdmin:
30     user: functional-admin
31     password: azerty3
32   securityInternal:
33     user: security-internal
34     password: azerty4
35   offer-fs-1:
36     passphrase: mongogo
37     admin:
38       user: vitamdb-admin
39       password: azerty
40     localadmin:
41       user: vitamdb-localadmin
42       password: qwerty
43     offer:
44       user: offer
45       password: azerty5
46   offer-fs-2:
47     passphrase: mongogo
48     admin:
49       user: vitamdb-admin
50       password: azerty
51     localadmin:
52       user: vitamdb-localadmin
53       password: qwerty
54     offer:

```

(suite sur la page suivante)

(suite de la page précédente)

```

54     user: offer
55     password: azerty5
56 offer-swift-1:
57     passphrase: mongogo
58     admin:
59         user: vitamdb-admin
60         password: azerty
61     localadmin:
62         user: vitamdb-localadmin
63         password: qwerty
64     offer:
65         user: offer
66         password: azerty5
67
68 vitam_users:
69     - vitam_aadmin:
70         login: aadmin
71         password: aadmin1234
72         role: admin
73     - vitam_uuser:
74         login: uuser
75         password: uuser1234
76         role: user
77     - vitam_gguest:
78         login: gguest
79         password: gguest1234
80         role: guest
81     - techadmin:
82         login: techadmin
83         password: techadmin1234
84         role: admin
85
86 ldap_authentification:
87     ldap_pwd: "admin"
88
89 admin_basic_auth_password: adminPassword
90
91 vitam_offers:
92     offer-swift-1:
93         swiftPassword: password

```

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 ».

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

```

1 keystores:
2   server:
3     offer: azerty1
4     access_external: azerty2
5     ingest_external: azerty3

```

(suite sur la page suivante)

(suite de la page précédente)

```
6     ihm_recette: azerty16
7     ihm_demo: azerty17
8 client_external:
9     ihm_demo: azerty4
10    gatling: azerty4
11    ihm_recette: azerty5
12    reverse: azerty6
13 client_storage:
14    storage: azerty7
15    timestamping:
16    secure_logbook: azerty8
17 truststores:
18    server: azerty9
19    client_external: azerty10
20    client_storage: azerty11
21 grantedstores:
22    client_external: azerty12
23    client_storage: azerty13
```

Avvertissement : il convient de sécuriser votre environnement en définissant des mots de passe « forts ».

4.2.2.3.1 Cas des extra

- Le fichier `environments/group_vars/all/vault-extra.yml` contient le mot 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: "password"
```

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 Gestion des certificats

Une vue d'ensemble de la gestion des certificats est présentée *dans l'annexe dédiée* (page 60).

4.2.3.1 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.

4.2.3.1.1 Procédure générale

Danger : La *PKI* fournie avec la solution logicielle Vitam ne doit être utilisée que pour faire des tests, et ne doit par conséquent surtout pas être utilisée en environnement de production ! De plus il n'est pas prévu 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

4.2.3.1.2 Génération des CA par les scripts Vitam

Il faut faire générer les autorités de certification 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`

Avertissement : Bien 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.

4.2.3.1.3 Génération des certificats par les scripts Vitam

Le fichier d'inventaire de déploiement `environments/<fichier d'inventaire>` (cf. *Informations « plate-forme »* (page 13)) 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.

4.2.3.2 Cas 2 : Configuration production

4.2.3.2.1 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 VITAM.

4.2.3.2.2 Génération des certificats

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

4.2.3.2.2.1 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 rajouter le nom dns qui sera utilisé pour requêter ces deux applications si celles-ci sont appelées directement en frontal en https.

4.2.3.2.2.2 Certificat clients

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

4.2.3.2.3 Certificats d'horodatage

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

4.2.3.2.3 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.

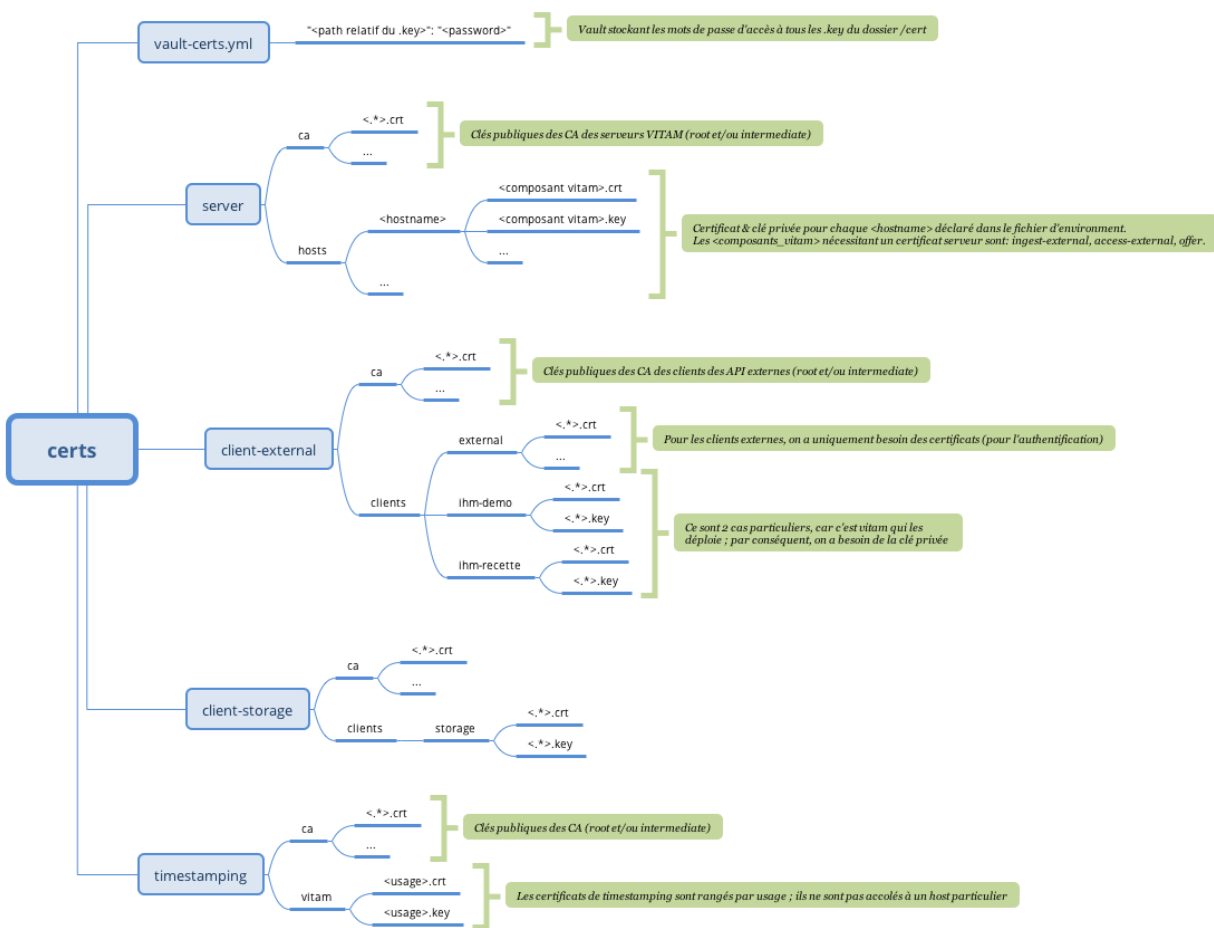


Fig. 2 – Vue détaillée de l'arborescence des certificats

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 voir 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-certs.yml`

Pour modifier/créer un vault ansible, se référer à la documentation sur [cette url](#) ¹¹.

4.2.3.2.4 Intégration d'une application externe (cliente)

Dans le cas d'ajout de certificats *SIA* externes :

- 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 ceux-ci soient ajoutés aux profils de sécurité durant le déploiement de la solution logicielle *VITAM*.

4.2.3.3 Intégration de CA pour une offre swift

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

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

4.2.3.4 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 25).

Lancer le script : `./generate_stores.sh`

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

Il est aussi possible de déposer directement les *keystores* au bon format en remplaçant ceux fournis par défaut, 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.

4.2.4 Paramétrages supplémentaires

4.2.4.1 Tuning JVM

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

Note : Cette section est en cours de développement.

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 fichier `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

4.2.4.2 Paramétrage de l'antivirus (ingest-externe)

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

- Modifier le fichier `environments/group_vars/all/vitam_vars.yml` pour indiquer le nom de l'antivirus qui sera utilisé (norme : `scan-<nom indiqué dans vitam-vars.yml>.sh`)
- Créer un shell (dont l'extension doit être `.sh`) sous `environments/antivirus/` (norme : `scan-<nom indiqué dans vitam-vars.yml>.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 la télécharger ; il est donc nécessaire de renseigner dans l'inventaire la directive `http_proxy_environnement`.

4.2.4.3 Paramétrage des certificats externes (*-externe)

Se reporter au chapitre dédié à la gestion des certificats : *Gestion des certificats* (page 28)

4.2.4.4 Placer « hors Vitam » le composant ihm-demo

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

```
consul_disabled: true
```

A l'issue, le déploiement n'installera pas l'agent Consul. Le composant ihm-demo appellera, alors, par l'adresse IP de services 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 de données sur « elasticsearch-log ».

4.2.4.5 Paramétrage de la centralisation des logs Vitam

2 cas sont possibles :

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

4.2.4.5.1 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 group

- `hosts-logstash`
- `hosts-kibana-log`
- `hosts-elasticsearch-log`

4.2.4.5.2 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](#)¹².

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).

4.2.4.6 Fichiers complémentaires

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

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

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

```

1  ---
2
3  ### global ###
4
5  # TODO MAYBE : permettre la surcharge avec une syntax du genre vitamopts.folder_
6  ↳root | default(vitam_default.folder_root) dans les templates ?
7
8  vitam_defaults:
9      folder:
10         root_path: /vitam
11         folder_permission: "0750"
12         conf_permission: "0640"
13         folder_upload_permission: "0770"
14         script_permission: "0750"
15
16     users:
17         vitam: "vitam"
18         vitamdb: "vitamdb"
19         group: "vitam"
20
21     services:
22         # Default log level for vitam components: logback values (TRACE, DEBUG,
23         ↳INFO, WARN, ERROR, OFF)
24         log_level: WARN
25         start_timeout: 300
26         stop_timeout: 3600
27         port_service_timeout: 86400
28         api_call_timeout: 120
29
30         # Filter for the vitam package version to install
31         # FIXME : commented as has to be removed becuse doesn't work under Debain
32         #package_version: "*"
33
34         ### Trust X-SSL-CLIENT-CERT header for external api auth ? (true | false) ###
35         vitam_ssl_user_header: true
36         ### Force chunk mode : set true if chunk header should be checked
37         vitam_force_chunk_mode: false
38         # syslog_facility
39         syslog_facility: local0
40
41     # Used in ingest, unitary update, mass-update
42     classificationList: ["Secret Défense", "Confidentiel Défense"]
43     # Used in ingest, unitary update, mass-update
44     classificationLevelOptional: true
45
46
47     ### consul ###
48     # FIXME: Consul à la racine pour le moment à cause de problèmes de récursivité
49     ↳dans le parsing yaml
50     # WARNING: consul_domain should be a supported domain name for your organization
51     # You will have to generate server certificates with the same domain
52     ↳name and the service subdomain name
53     # Example: consul_domain=vitam means you will have to generate some
54     ↳certificates with .service.vitam domain
55     # access-external.service.vitam, ingest-external.service.vitam,
56     ↳...
57     consul_domain: consul
58     consul_component: consul
59     consul_folder_conf: "{{ vitam_defaults.folder.root_path }}/conf/{{ consul_
60     ↳component }}"

```

(suite sur la page suivante)

(suite de la page précédente)

```

51 # Workspace should be useless but storage have a dependency to it...
52 # elastic-kibana-interceptor is present as kibana is present, if kibana-data &
↪ interceptor are not needed in the secondary site, just do not add them in the
↪ hosts file
53 vitam_secondary_site_components: [ "logbook" , "metadata" , "functional-
↪ administration" , "storage" , "storageofferdefault" , "offer" , "elasticsearch-
↪ log" , "elasticsearch-data" , "logstash" , "kibana" , "mongoc" , "mongod" ,
↪ "mongos" , "elastic-kibana-interceptor" ]
54
55
56 ### Composants Vitam ###
57
58 vitam:
59     accessexternal:
60         # Component name: do not modify
61         vitam_component: access-external
62         # DNS record for the service:
63         # Modify if ihm-demo is not using consul (typical production deployment)
64         host: "access-external.service.{{ consul_domain }}"
65         port_admin: 28102
66         port_service: 8444
67         baseuri: "access-external"
68         https_enabled: true
69         # Use platform secret for this component ? : do not modify
70         secret_platform: "false"
71         # Force the log level for this component: this are logback values (TRACE,
↪ DEBUG, INFO, WARN, ERROR, OFF)
72         # If this var is not set, the default one will be used (vitam_defaults.
↪ services.log_level)
73         # log_level: "DEBUG"
74         metrics_enabled: true
75         logback_rolling_policy: true
76         logback_max_file_size: "10MB"
77         logback_total_size_cap: "5GB"
78         jvm_log: false
79     accessinternal:
80         vitam_component: access-internal
81         host: "access-internal.service.{{ consul_domain }}"
82         port_service: 8101
83         port_admin: 28101
84         baseuri: "access-internal"
85         https_enabled: false
86         secret_platform: "true"
87         # log_level: "DEBUG"
88         metrics_enabled: true
89         logback_rolling_policy: true
90         logback_max_file_size: "10MB"
91         logback_total_size_cap: "5GB"
92         jvm_log: false
93     functional_administration:
94         vitam_component: functional-administration
95         host: "functional-administration.service.{{ consul_domain }}"
96         port_service: 8004
97         port_admin: 18004
98         baseuri: "functional-administration"
99         https_enabled: false
100        secret_platform: "true"

```

(suite sur la page suivante)

(suite de la page précédente)

```

101     cluster_name: "{{ elasticsearch.data.cluster_name }}"
102     # log_level: "DEBUG"
103     metrics_enabled: true
104     logback_rolling_policy: true
105     logback_max_file_size: "10MB"
106     logback_total_size_cap: "5GB"
107     jvm_log: false
108     elasticsearchinterceptor:
109         vitam_component: elastic-kibana-interceptor
110         host: "elastic-kibana-interceptor.service.{{ consul_domain }}"
111         port_service: 8014
112         port_admin: 18014
113         baseuri: ""
114         https_enabled: false
115         secret_platform: "false"
116         cluster_name: "{{ elasticsearch.data.cluster_name }}"
117         # log_level: "DEBUG"
118         metrics_enabled: true
119         logback_rolling_policy: true
120         logback_max_file_size: "10MB"
121         logback_total_size_cap: "5GB"
122         jvm_log: false
123     batchreport:
124         vitam_component: batch-report
125         host: "batch-report.service.{{ consul_domain }}"
126         port_service: 8015
127         port_admin: 18015
128         baseuri: "batch-report"
129         https_enabled: false
130         secret_platform: "false"
131         # log_level: "DEBUG"
132         metrics_enabled: true
133         logback_rolling_policy: true
134         logback_max_file_size: "10MB"
135         logback_total_size_cap: "5GB"
136         jvm_log: false
137     ingestexternal:
138         vitam_component: ingest-external
139         # DNS record for the service:
140         # Modify if ihm-demo is not using consul (typical production deployment)
141         host: "ingest-external.service.{{ consul_domain }}"
142         port_admin: 28001
143         port_service: 8443
144         baseuri: "ingest-external"
145         https_enabled: true
146         secret_platform: "false"
147         antivirus: "clamav"
148         # Directory where files should be placed for local ingest
149         upload_dir: "/vitam/data/ingest-external/upload"
150         # Directory where successful ingested files will be moved to
151         success_dir: "/vitam/data/ingest-external/upload/success"
152         # Directory where failed ingested files will be moved to
153         fail_dir: "/vitam/data/ingest-external/upload/failure"
154         # Action done to file after local ingest (see below for further
155         ↪ information)
156         upload_final_action: "MOVE"
157         # log_level: "DEBUG"

```

(suite sur la page suivante)

(suite de la page précédente)

```

157     # upload_final_action can be set to three different values (lower or
↳upper case does not matter)
158     # 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
159     # DELETE : After upload, the local file will be deleted if the upload
↳succeeded
160     # 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)
161     metrics_enabled: true
162     logback_rolling_policy: true
163     logback_max_file_size: "10MB"
164     logback_total_size_cap: "5GB"
165     jvm_log: false
166     ingestinternal:
167       vitam_component: ingest-internal
168       host: "ingest-internal.service.{{ consul_domain }}"
169       port_service: 8100
170       port_admin: 28100
171       baseuri: "ingest-internal"
172       https_enabled: false
173       secret_platform: "true"
174       # log_level: "DEBUG"
175       metrics_enabled: true
176       logback_rolling_policy: true
177       logback_max_file_size: "10MB"
178       logback_total_size_cap: "5GB"
179       jvm_log: false
180     ihm_demo:
181       vitam_component: "ihm-demo"
182       host: "ihm-demo.service.{{ consul_domain }}"
183       port_service: 8002
184       port_admin: 28002
185       baseurl: "/ihm-demo"
186       static_content: "{{ vitam_defaults.folder.root_path }}/app/ihm-demo/v2"
187       baseuri: "ihm-demo"
188       https_enabled: false
189       secret_platform: "false"
190       # User session timeout in milliseconds (for shiro)
191       session_timeout: 1800000
192       secure_cookie: false
193       # Specify here the realms you want to use for authentication in ihm-demo
194       # You can set multiple realms, one per line
195       # With multiple realms, the user will be able to choose between the
↳allowed realms
196       # Example: authentication_realms:
197       #           - x509Realm
198       #           - ldapRealm
199       # Authorized values:
200       # x509Realm: certificate
201       # iniRealm: ini file
202       # ldapRealm: ldap
203       authentication_realms:
204         # - x509Realm
205         - iniRealm
206         # - ldapRealm
207       # log_level: "DEBUG"
208       metrics_enabled: true

```

(suite sur la page suivante)

(suite de la page précédente)

```

209     logback_rolling_policy: true
210     logback_max_file_size: "10MB"
211     logback_total_size_cap: "5GB"
212     jvm_log: false
213   logbook:
214     vitam_component: logbook
215     host: "logbook.service.{{ consul_domain }}"
216     port_service: 9002
217     port_admin: 29002
218     baseuri: "logbook"
219     https_enabled: false
220     secret_platform: "true"
221     cluster_name: "{{ elasticsearch.data.cluster_name }}"
222     # Temporization delay (in seconds) for recent logbook operation events.
223     # Set it to a reasonable delay to cover max clock difference across_
↪servers + VM/GC pauses
224     operationTraceabilityTemporizationDelay: 300
225     # Temporization delay (in seconds) for recent logbook lifecycle events.
226     # Set it to a reasonable delay to cover max clock difference across_
↪servers + VM/GC pauses
227     lifecycleTraceabilityTemporizationDelay: 300
228     # Max entries selected per (Unit or Object Group) LFC traceability_
↪operation
229     lifecycleTraceabilityMaxEntries: 100000
230     # log_level: "DEBUG"
231     metrics_enabled: true
232     logback_rolling_policy: true
233     logback_max_file_size: "10MB"
234     logback_total_size_cap: "5GB"
235     jvm_log: false
236   metadata:
237     vitam_component: metadata
238     host: "metadata.service.{{ consul_domain }}"
239     port_service: 8200
240     port_admin: 28200
241     baseuri: "metadata"
242     https_enabled: false
243     secret_platform: "true"
244     cluster_name: "{{ elasticsearch.data.cluster_name }}"
245     # log_level: "DEBUG"
246     metrics_enabled: true
247     logback_rolling_policy: true
248     logback_max_file_size: "10MB"
249     logback_total_size_cap: "5GB"
250     jvm_log: false
251   processing:
252     vitam_component: processing
253     host: "processing.service.{{ consul_domain }}"
254     port_service: 8203
255     port_admin: 28203
256     baseuri: "processing"
257     https_enabled: false
258     secret_platform: "true"
259     # log_level: "DEBUG"
260     metrics_enabled: true
261     logback_rolling_policy: true
262     logback_max_file_size: "10MB"

```

(suite sur la page suivante)

(suite de la page précédente)

```

263     logback_total_size_cap: "5GB"
264     jvm_log: false
265 security_internal:
266     vitam_component: security-internal
267     host: "security-internal.service.{{ consul_domain }}"
268     port_service: 8005
269     port_admin: 28005
270     baseuri: "security-internal"
271     https_enabled: false
272     secret_platform: "true"
273     # log_level: "DEBUG"
274     metrics_enabled: true
275     logback_rolling_policy: true
276     logback_max_file_size: "10MB"
277     logback_total_size_cap: "5GB"
278     jvm_log: false
279 storageengine:
280     vitam_component: storage
281     host: "storage.service.{{ consul_domain }}"
282     port_service: 9102
283     port_admin: 29102
284     baseuri: "storage-engine"
285     https_enabled: false
286     secret_platform: "true"
287     storageTraceabilityOverlapDelay: 300
288     restoreBulkSize: 1000
289     # log_level: "DEBUG"
290     metrics_enabled: true
291     logback_rolling_policy: true
292     logback_max_file_size: "10MB"
293     logback_total_size_cap: "5GB"
294     jvm_log: false
295 storageofferdefault:
296     vitam_component: "offer"
297     port_service: 9900
298     port_admin: 29900
299     baseuri: "storage-offer-default"
300     https_enabled: false
301     secret_platform: "true"
302     # log_level: "DEBUG"
303     metrics_enabled: true
304     logback_rolling_policy: true
305     logback_max_file_size: "10MB"
306     logback_total_size_cap: "5GB"
307     jvm_log: false
308 worker:
309     vitam_component: worker
310     port_service: 9104
311     port_admin: 29104
312     baseuri: "worker"
313     https_enabled: false
314     secret_platform: "true"
315     # log_level: "DEBUG"
316     metrics_enabled: true
317     logback_rolling_policy: true
318     logback_max_file_size: "10MB"
319     logback_total_size_cap: "5GB"

```

(suite sur la page suivante)

(suite de la page précédente)

```

320     jvm_log: false
321   workspace:
322     vitam_component: workspace
323     host: "workspace.service.{{ consul_domain }}"
324     port_service: 8201
325     port_admin: 28201
326     baseuri: "workspace"
327     https_enabled: false
328     secret_platform: "true"
329     # log_level: "DEBUG"
330     metrics_enabled: true
331     logback_rolling_policy: true
332     logback_max_file_size: "10MB"
333     logback_total_size_cap: "5GB"
334     jvm_log: false

```

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

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

```

1  ---
2
3  consul:
4    dns_port: 53
5
6  consul_remote_sites:
7    # wan contains the wan addresses of the consul server instances of the_
8    ↪external vitam sites
9    # Exemple, if our local dc is dc1, we will need to set dc2 & dc3 wan conf:
10   # - dc2:
11   #   wan: ["10.10.10.10", "1.1.1.1"]
12   # - dc3:
13   #   wan: ["10.10.10.11", "1.1.1.1"]
14
15  elasticsearch:
16    log:
17      host: "elasticsearch-log.service.{{ consul_domain }}"
18      port_http: "9201"
19      port_tcp: "9301"
20      groupe: "log"
21      baseuri: "elasticsearch-log"
22      cluster_name: "elasticsearch-log"
23      https_enabled: false

```

(suite sur la page suivante)

(suite de la page précédente)

```

23     # default index template
24     index_templates:
25         default:
26             shards: 1
27             replica: 1
28     data:
29         host: "elasticsearch-data.service.{{ consul_domain }}"
30         # default is 0.1 (10%) and should be quite enough in most cases
31         #index_buffer_size_ratio: "0.15"
32         port_http: "9200"
33         port_tcp: "9300"
34         groupe: "data"
35         baseuri: "elasticsearch-data"
36         cluster_name: "elasticsearch-data"
37         https_enabled: false
38         # default index template
39         index_templates:
40             default:
41                 shards: 10
42                 replica: 2
43
44     mongodb:
45         mongos_port: 27017
46         mongoc_port: 27018
47         mongod_port: 27019
48         mongo_authentication: "true"
49         host: "mongos.service.{{ consul_domain }}"
50
51     logstash:
52         host: "logstash.service.{{ consul_domain }}"
53         user: logstash
54         port: 10514
55         rest_port: 20514
56
57     # Curator units: days
58     curator:
59         log:
60             metrics:
61                 close: 5
62                 delete: 30
63             logstash:
64                 close: 5
65                 delete: 30
66             metricbeat:
67                 close: 5
68                 delete: 30
69             packetbeat:
70                 close: 5
71                 delete: 30
72
73     kibana:
74         header_value: "reporting"
75         log:
76             baseuri: "kibana_log"
77             api_call_timeout: 120
78             groupe: "log"
79             port: 5601

```

(suite sur la page suivante)

(suite de la page précédente)

```

80     default_index_pattern: "logstash-vitam*"
81     # default shards & replica
82     shards: 5
83     replica: 1
84     # pour index logstash-*
85     metrics:
86         shards: 5
87         replica: 1
88     # pour index metrics-vitam-*
89     logs:
90         shards: 5
91         replica: 1
92     # pour index metricbeat-*
93     metricbeat:
94         shards: 5 # must be a factor of 30
95         replica: 1
96     data:
97         baseuri: "kibana_data"
98         # OMA : bugdette : api_call_timeout is used for retries ; should ceate a
↳separate variable rather than this one
99         api_call_timeout: 120
100        groupe: "data"
101        port: 5601
102        default_index_pattern: "logbookoperation*"
103        # index template for .kibana
104        shards: 1
105        replica: 1
106
107    syslog:
108        # value can be syslog-ng or rsyslog
109        name: "rsyslog"
110
111    cerebro:
112        baseuri: "cerebro"
113        port: 9000
114
115    siegfried:
116        port: 19000
117
118    clamav:
119        port: 3310
120        # frequency freshclam for database update per day (from 0 to 24 - 24 meaning
↳hourly check)
121        db_update_periodicity: 1
122
123    mongo_express:
124        baseuri: "mongo-express"
125
126    ldap_authentication:
127        ldap_protocol: "ldap"
128        ldap_server: "%{ if groups['ldap']|length > 0 %}{ groups['ldap']|first }}{%
↳endif %}"
129        ldap_port: "389"
130        ldap_base: "dc=programmevitam,dc=fr"
131        ldap_login: "cn=Manager,dc=programmevitam,dc=fr"
132        uid_field: "uid"
133        ldap_userDn_Template: "uid={0},ou=people,dc=programmevitam,dc=fr"

```

(suite sur la page suivante)

(suite de la page précédente)

```

134 ldap_group_request: "(&(objectClass=groupOfNames)(member={0}))"
135 ldap_admin_group: "cn=admin,ou=groups,dc=programmevitam,dc=fr"
136 ldap_user_group: "cn=user,ou=groups,dc=programmevitam,dc=fr"
137 ldap_guest_group: "cn=guest,ou=groups,dc=programmevitam,dc=fr"

```

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).

- `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    elasticsearchinterceptor:
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"

```

(suite sur la page suivante)

(suite de la page précédente)

```

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     workspace:
55         jvm_opts:
56             # memory: "-Xms512m -Xmx512m"
57             # gc: ""
58             # java: ""
59     processing:
60         jvm_opts:
61             # memory: "-Xms512m -Xmx512m"
62             # gc: ""
63             # java: ""
64     worker:
65         jvm_opts:
66             # memory: "-Xms512m -Xmx512m"
67             # gc: ""
68             # java: ""
69     storageengine:
70         jvm_opts:
71             # memory: "-Xms512m -Xmx512m"
72             # gc: ""
73             # java: ""
74     storageofferdefault:
75         jvm_opts:
76             # memory: "-Xms512m -Xmx512m"
77             # gc: ""
78             # java: ""
79     functional_administration:
80         jvm_opts:
81             # memory: "-Xms512m -Xmx512m"
82             # gc: ""
83             # java: ""
84     security_internal:
85         jvm_opts:
86             # memory: "-Xms512m -Xmx512m"
87             # gc: ""
88             # java: ""
89     library:
90         jvm_opts:
91             memory: "-Xms32m -Xmx128m"
92             # gc: ""
93             # 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 Procédure de première installation

4.2.5.1 Déploiement

4.2.5.1.1 Cas particulier : utilisation de ClamAv en environnement Debian

Dans le cas de l'installation en environnement Debian, la base de donnée 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 s'installer manuellement. Les liens suivants indiquent la procédure à suivre : [Installation ClamAv](#)¹³ et [Section Virus Database](#)¹⁴

4.2.5.1.2 Fichier de mot de passe

Par défaut, le mot de passe des « vault » sera demandé à chaque exécution d'ansible. Si le fichier `deployment/vault_pass.txt` est renseigné avec le mot de passe du fichier `environments/group_vars/all/vault-vitam.yml`, le mot de passe ne sera pas demandé (dans ce cas, changez l'option `--ask-vault-pass` des invocations ansible par l'option `--vault-password-file=VAULT_PASSWORD_FILES`).

4.2.5.1.3 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 `environments/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: "/"
```

(suite sur la page suivante)

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

(suite de la page précédente)

```

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/<fichier d
↳ 'inventaire' --ask-vault-pass

```

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

4.2.5.1.4 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

4.2.5.1.4.1 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 `ansible-vitam/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.

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

4.2.5.1.4.2 Cas 2 : Machines avec plusieurs interfaces réseau

Si les machines sur lesquelles Vitam sera déployé disposent de plusieurs interfaces, 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.

4.2.5.1.4.3 Vérification de la génération des hostvars

A l'issue, vérifier le contenu des fichiers générés sous `environments/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.

4.2.5.1.5 Passage des identifiants des référentiels en mode esclave

La génération des identifiants des référentiels est gérée par Vitam quand il fonctionne en mode maître.

Par exemple :

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

Si vous souhaitez gérer vous-même les identifiants sur un service référentiel, il faut qu'il soit en mode esclave. 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/ansible-vitam/roles/vitam/templates/functional-administration/functional-administration.conf.j2`. Un exemple de ce fichier se trouve dans la Documentation d'exploitation au chapitre « Exploitation des composants de la solution logicielle VITAM ».

```
# ExternalId configuration

listEnableExternalIdentifiers:
  0:
    - INGEST_CONTRACT
    - ACCESS_CONTRACT
  1:
    - INGEST_CONTRACT
    - ACCESS_CONTRACT
    - PROFILE
    - SECURITY_PROFILE
    - CONTEXT
```

Depuis la version 1.0.4, la configuration par défaut de Vitam autorise des identifiants externes (ceux qui sont dans le fichier json importé).

- pour le tenant 0 pour les référentiels : contrat d'entrée et contrat d'accès.
- pour le tenant 1 pour les référentiels : contrat d'entrée, contrat d'accès, profil, profil de sécurité et contexte.

La liste des choix possibles, pour chaque tenant, est :

- `INGEST_CONTRACT` : contrats d'entrée
- `ACCESS_CONTRACT` : contrats d'accès
- `PROFILE` : profils SEDA

- SECURITY_PROFILE : profils de sécurité (utile seulement sur le tenant d'administration)
- CONTEXT : contextes applicatifs (utile seulement sur le tenant d'administration)
- ARCHIVEUNITPROFILE : profils d'unités archivistiques

4.2.5.1.6 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 `listMinimumRuleDuration` dans le fichier ansible `deployment/ansible-vitam/roles/vitam/templates/functional-administration/functional-administration.conf.j2`.

Exemple :

```
listMinimumRuleDuration:
  2:
    AppraisalRule : 1 year
    DisseminationRule : 10 year

  3:
    AppraisalRule : 5 year
    StorageRule : 5 year
    ReuseRule : 2 year
```

Par tenant, les directives possibles sont :

- AppraisalRule
- DisseminationRule
- StorageRule
- ReuseRule
- AccessRule (valeur par défaut : 0 year)
- ClassificationRule

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

Exemples :

```
6 month
1 year
5 year
```

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

4.2.5.1.7 Déploiement

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 a été effectuée avec succès, le déploiement est à réaliser avec la commande suivante :

```
ansible-playbook ansible-vitam/vitam.yml -i environments/<fichier d'inventaire> --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.

4.2.6 Elements 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 installation 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.

4.2.6.1 Configuration des extra

Le fichier `environments /group_vars/all/extra_vars.yml` contient la configuration des extra :

```
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    metrics_enabled: true
21    logback_rolling_policy: true
22    logback_max_file_size: "10MB"
23    logback_total_size_cap: "5GB"
```

(suite sur la page suivante)

(suite de la page précédente)

```

24     jvm_log: false
25   library:
26     vitam_component: library
27     host: "library.service.{{consul_domain}}"
28     port_service: 8090
29     port_admin: 28090
30     baseuri: "doc"
31     https_enabled: false
32     secret_platform: "false"
33     metrics_enabled: false
34     logback_rolling_policy: true
35     logback_max_file_size: "10MB"
36     logback_total_size_cap: "5GB"
37     jvm_log: false
38
39
40   # Period units in seconds
41   metricbeat:
42     system:
43       period: 10
44     mongodb:
45       period: 10
46     elasticsearch:
47       period: 10
48
49   docker_opts:
50     registry_httponly: yes
51     vitam_docker_tag: latest

```

Avertissement : A modifier selon le besoin avant de lancer le playbook ! Le composant ihm-recette, s'il est déployé en « https », doit avoir un paramétrage différent dans `environments/group_vars/all/extra_vars.yml` sur le paramètre `secure_cookie` selon qu'il est attaqué en direct ou derrière un proxy https (`secure_cookie: true`) ou un proxy http (`secure_cookie: false`). Par défaut, la variable est à `true`. Le symptôme observé, en cas de problème, est une bonne authentification, mais l'impossibilité de sélectionner un tenant.

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 `environments /group_vars/all/all/vault-extra.yml` contient les secrets supplémentaires des extra ; ce fichier est encrypté par `ansible-vault` et doit être paramétré avant le lancement de l'orchestration de 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: "password"

```

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

4.2.6.2 Déploiement des extra

Plusieurs playbook d'extra sont fournis pour usage « tel quel ».

4.2.6.2.1 ihm-recette

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

```
ansible-playbook ansible-vitam-extra/ihm-recette.yml -i environments/<fichier d'inventaire> --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.

4.2.6.2.2 extra 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

```
ansible-playbook ansible-vitam-extra/extra.yml -i environments/<fichier d'inventaire> --ask-vault-pass
```

Procédures de mise à jour

Cette section décrit globalement le processus de mise à niveau d'une solution VITAM déjà en place et ne peut se substituer aux recommandations effectuées dans la « release note » associée à la fourniture des composants mis à niveau.

Prudence : Seule la mise à jour depuis la version « 1.0.3 » est supportée dans cette version de la solution logicielle VITAM. Se référer à `upgrade_r6_r7` pour plus de détails.

5.1 Reconfiguration

5.1.1 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 extra) avec l'option supplémentaire `--tags update_vitam_configuration`.

Exemple :

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

5.1.2 Cas d'une modification des paramètres JVM

Se référer à *Tuning JVM* (page 32)

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 extra) avec l'option supplémentaire `--tags update_jvmoptions_vitam`.

Exemple :

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

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 `environments/group_vars/all/vault.yml` qui contient les divers mots de passe de la plate-forme. Il est fortement déconseillé de ne pas l'utiliser en production. A l'issue de l'installation, il est nécessaire 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 présente 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 changeant juste le nom du playbook à exécuter).

Avertissement : les composants VITAM « ihm » n'intègrent pas `/admin/v1/status` ».

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>: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.

Avertissement : les composants *VITAM* « ihm » (ihm-demo, ihm-recette) n'intègrent pas /admin/v1/status » et donc sont indiqués « KO » sous Consul ; il ne faut pas en tenir compte, sachant que si l'IHM s'affiche en appel « classique », le composant fonctionne.

6.1.4 Post-installation : administration fonctionnelle

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

- le référentiel PRONOM ([lien vers pronom¹⁵](#)) 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 apporter une solution associée.

6.3.1 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 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.

<http://www.nationalarchives.gov.uk/aboutapps/pronom/droid-signature-files.htm>

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 :

- Centos ¹⁶
- Debian ¹⁷

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 elasticseach, puis du retour de ce noeud, certains shards d'elasticseach peuvent rester dans l'état UNASSIGNED ; dans ce cas, cerebro affiche les shards correspondant en gris (au-dessus des noeuds) dans la vue « cluster », et l'état du cluster passe en « yellow ». Il est possible d'avoir plus d'informations sur la cause du problème via une requête POST sur l'API elasticsearch `_cluster/reroute?explain`. Si la cause de l'échec de l'assignation automatique a été résolue, il est possible de relancer les assignations automatiques en échec via une requête POST sur l'API `_cluster/reroute?retry_failed`. Dans le cas où l'assignation automatique ne fonctionne pas, il est nécessaire de faire l'assignation à la main pour chaque shard incriminé (requête POST sur `_cluster/reroute`):

```
{
  "commands": [
    {
      "allocate": {
        "index": "topbeat-2016.11.22",
        "shard": 3,
        "node": "vitam-iaas-dblog-01.int"
      }
    }
  ]
}
```

Cependant, un shard primaire ne peut être réalloué de cette manière (il y a risque de perte de données). Si le défaut d'allocation provient effectivement de la perte puis de la récupération d'un noeud, et que TOUS les noeuds du cluster sont de nouveaux opérationnels et dans le cluster, alors il est possible de forcer la réallocation sans perte.

```
{
  "commands": [
    {
      "allocate": {
        "index": "topbeat-2016.11.22",
        "shard": 3,
        "node": "vitam-iaas-dblog-01.int",
        "allow_primary": "true"
      }
    }
  ]
}
```

(suite sur la page suivante)

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

(suite de la page précédente)

```

    }
  }
}

```

Sur tous ces sujets, Cf. la [documentation officielle](#) ¹⁸.

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](#) ¹⁹. 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 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` ²⁰ et `mongotop` ²¹.

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 dus à la réplication](#) ²² :

```

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 :

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

```
# 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](#)²³) :

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

6.4.6 Les shards de MongoDB semblent mal équilibrés

Normalement, un processus interne à MongoDB (le balancer) s'occupe de déplacer les données entre les shards (par chunk) pour équilibrer la taille de ces derniers. Les commandes suivantes (à exécuter dans un shell mongo sur une instance mongos - attention, ces commandes ne fonctionnent pas directement sur les instances mongod) permettent de s'assurer du bon fonctionnement de ce processus :

- `sh.status()` : donne le status du sharding pour le cluster complet ; c'est un bon premier point d'entrée pour connaître l'état du balancer.
- `use <dbname>`, puis `db.<collection>.getShardDistribution()`, en indiquant le bon nom de base de données (ex : metadata) et de collection (ex : Unit) : donne les informations de répartition des chunks dans les différents shards pour cette collection.

6.4.7 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.

6.4.8 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.AuthorizationWrapper.
↪checkTimestamp(AuthorizationWrapper.java:117) : [vitam-env-int8-app-04.vitam-
↪env:storage:239079175] Timestamp check failed
```

Il faut vérifier / corriger l'heure des machines hébergeant la solution logicielle [VITAM](#) ; un *delta* de temps supérieur à 10s a été détecté entre les machines.

<https://docs.mongodb.com/manual/reference/program/mongoperf/>

7.1 Vue d'ensemble de la gestion des certificats

7.1.1 Liste des suites cryptographiques & protocoles supportés par Vitam

Il est possible de consulter les ciphers supportés par 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).

7.1.2 Vue d'ensemble de la gestion des certificats

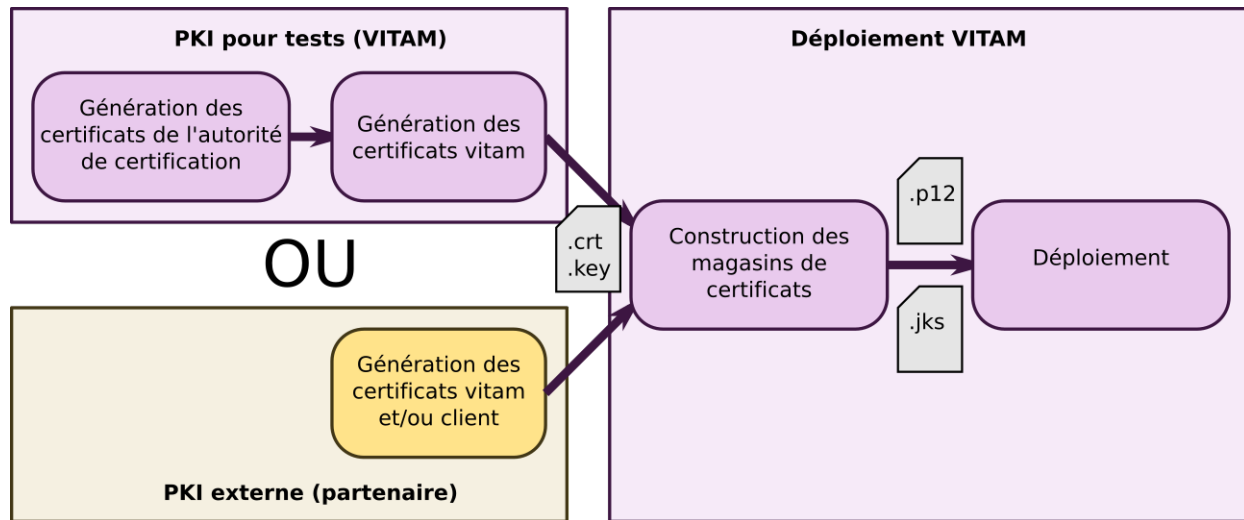


Fig. 1 – Vue d'ensemble de la gestion des certificats au déploiement

7.1.3 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 64)

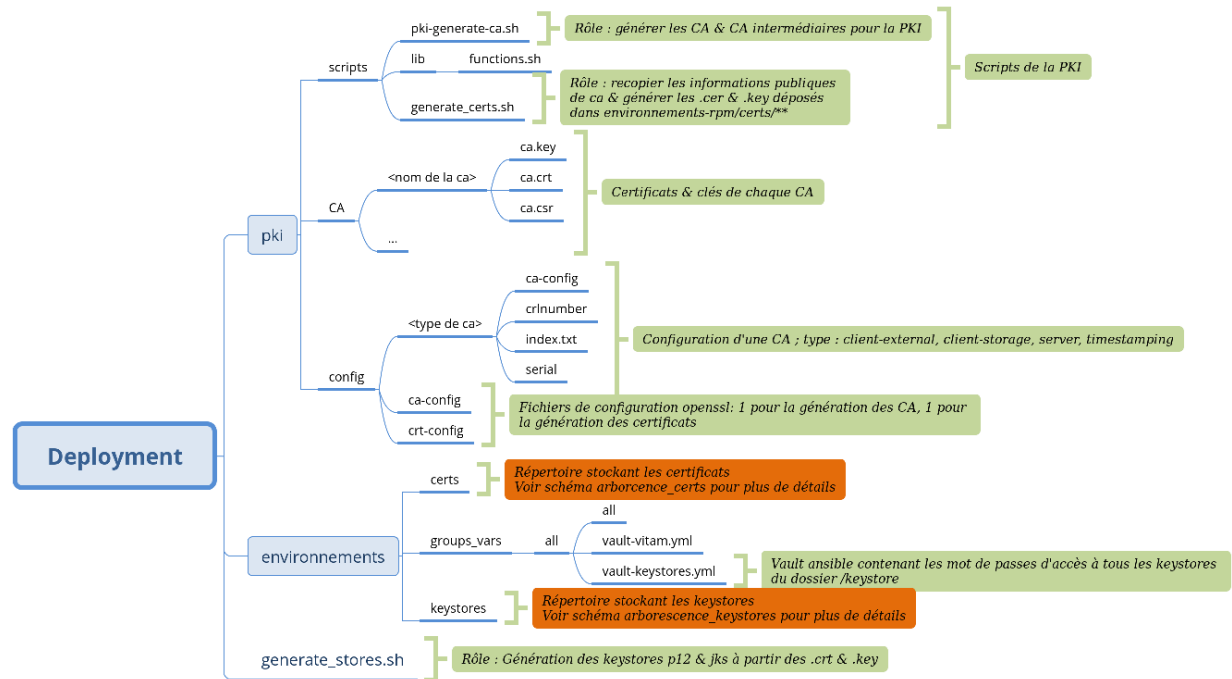


Fig. 2 – Vue l'arborescence de la PKI Vitam

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

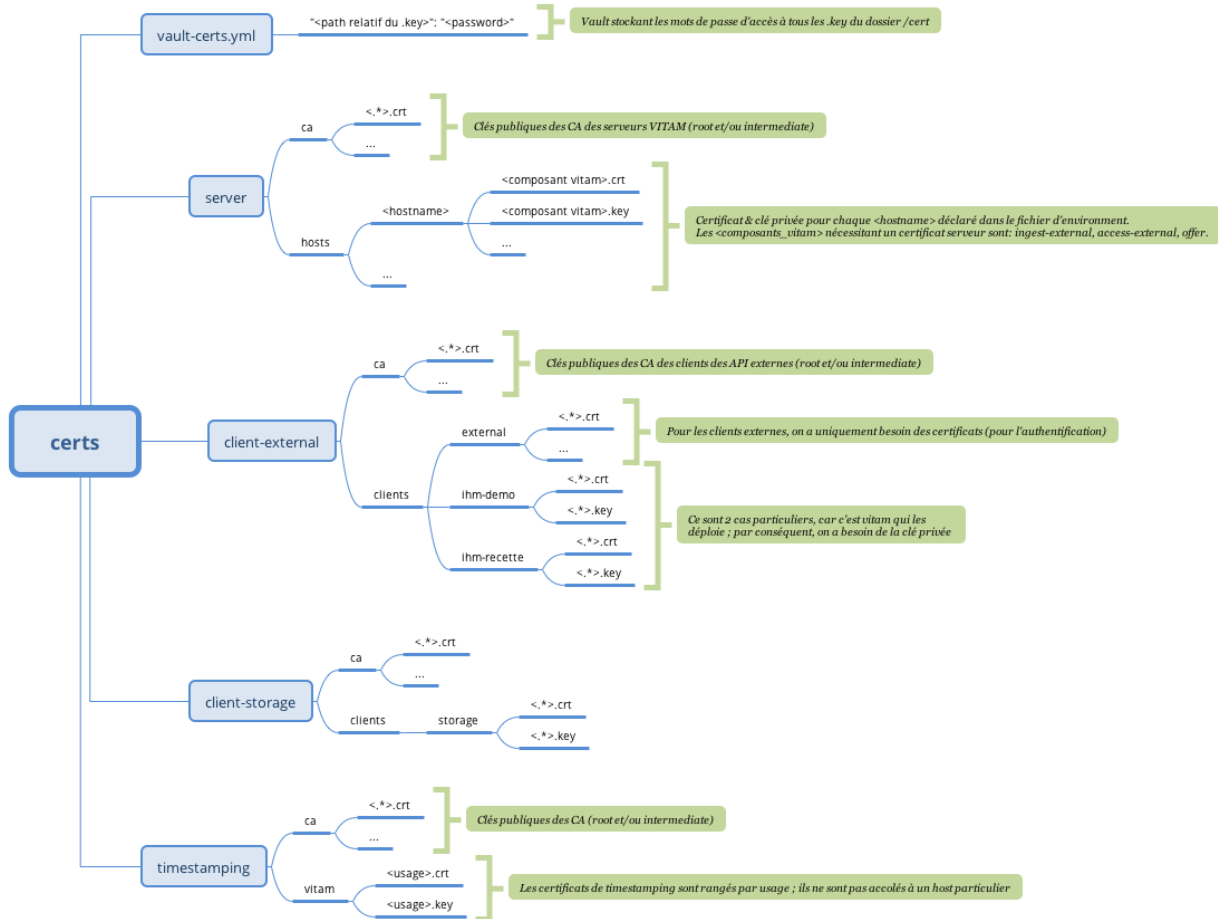


Fig. 3 – Vue détaillée de l'arborescence des certificats

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

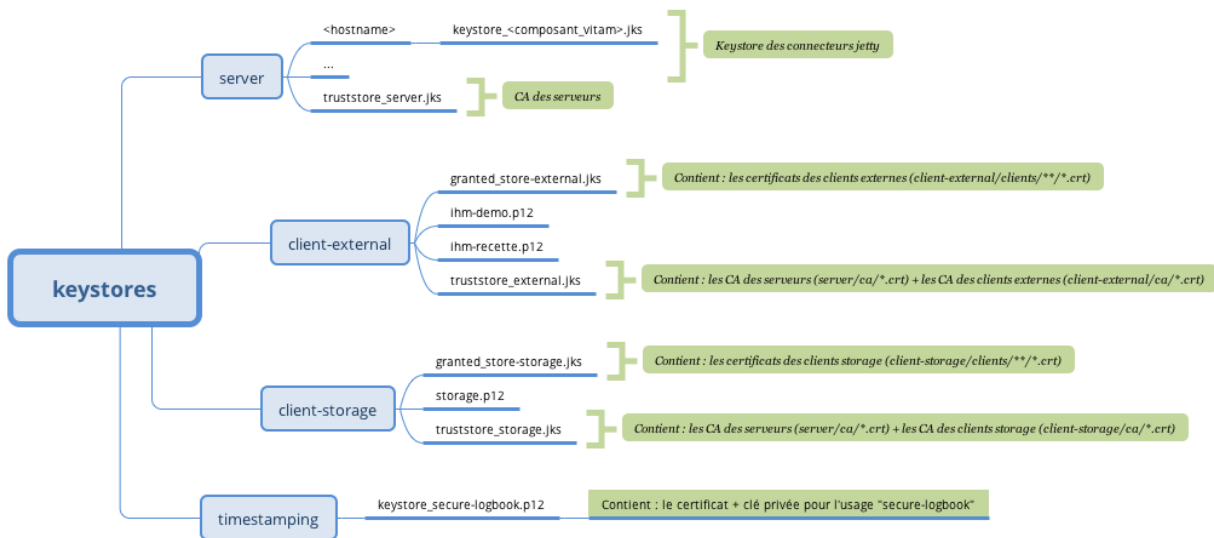


Fig. 4 – Vue détaillée de l'arborescence des keystores

7.1.6 Fonctionnement des scripts de la PKI

La gestion de la PKI se fait avec 3 scripts 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 donc les positionner dans l'arborescence attendue avec le nom attendu pour certains (cf *l'image ci-dessus* (page 63)).

7.2 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	Us-ages	Origine	Création	Sup-pression	Renouvellement
Interne	CA	ingest & access	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		offer	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
	Certif	Horo-datage	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		Stor-age (swift)	Offre de stock-age	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		ingest	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		access	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		offer	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		Times-tamp	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
IHM demo	CA	ihm-demo	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
	Cer-tif	ihm-demo	PKI parte-naire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
SIA	CA	Appel API	PKI dis-tante	proc. par défaut (PKI distante)	<i>proc. par défaut</i>	proc. par défaut (PKI distante) + recharger Certifs
	Cer-tif	Appel API	PKI dis-tante	Génération + copie répertoire + deploy (par la suite, appel API d'insertion)	Sup-pression Mongo	Suppression Mongo + API d'insertion
Per-sonae	Cer-tif	Appel API	PKI dis-tante	API ajout	API sup-pression	API suppression + API ajout

Remarques :

- Lors d'un renouvellement de CA SIA, il faut s'assurer que les certificats qui y correspondaient sont retirés de MongoDB et que les nouveaux certificats sont ajoutés par le biais de l'API dédiée.
- Lors de toute suppression ou remplacement de certificats SIA, s'assurer que la suppression / remplacement des contextes associés soit également réalisée.
- 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.

7.3 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

7.3.1 Authentification du compte utilisateur utilisé pour la connexion SSH

Pour le login du compte utilisateur, voir la section *Informations « plate-forme »* (page 13).

7.3.1.1 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.

7.3.1.2 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

7.3.1.3 Par clé SSH sans passphrase

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

7.3.2 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.

7.3.3 Elevation de privilèges

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

7.3.3.1 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`

7.3.3.2 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

7.3.3.3 Par sudo sans mot de passe

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

7.3.3.4 Déjà Root

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

Table des figures

1	Vue détaillée des certificats entre le storage et l'offre en multi-site	12
2	Vue détaillée de l'arborescence des certificats	31
1	Vue d'ensemble de la gestion des certificats au déploiement	61
2	Vue l'arborescence de la PKI Vitam	62
3	Vue détaillée de l'arborescence des certificats	63
4	Vue détaillée de l'arborescence des keystores	64

Liste des tableaux

1	Documents de référence VITAM	2
---	--	---

A

API, [2](#)

B

BDD, [2](#)

C

CA, [2](#)

COTS, [3](#)

D

DAT, [3](#)

DEX, [3](#)

DIN, [3](#)

DNSSEC, [3](#)

DUA, [3](#)

I

IHM, [3](#)

J

JRE, [3](#)

JVM, [3](#)

M

MitM, [3](#)

N

NoSQL, [3](#)

O

OAIS, [3](#)

P

PCA, [3](#)

PDMA, [3](#)

PKI, [3](#)

PRA, [3](#)

R

REST, [3](#)

RPM, [3](#)

S

SAE, [3](#)

SEDA, [3](#)

SIA, [3](#)

SIP, [3](#)

T

TNR, [3](#)

V

VITAM, [3](#)