



# VITAM - Documentation d'installation

*Version 1.0.0*

**VITAM**

**mars 22, 2018**



<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Objectif de ce document . . . . .	1
<b>2</b>	<b>Rappels</b>	<b>3</b>
2.1	Information concernant les licences . . . . .	3
2.2	Documents de référence . . . . .	3
2.2.1	Documents internes . . . . .	3
2.2.2	Référentiels externes . . . . .	3
2.3	Glossaire . . . . .	3
<b>3</b>	<b>Prérequis à l’installation</b>	<b>5</b>
3.1	Expertises requises . . . . .	5
3.2	Pré-requis plate-forme . . . . .	5
3.2.1	Base commune . . . . .	5
3.2.2	PKI . . . . .	6
3.2.3	Systèmes d’exploitation . . . . .	6
3.2.3.1	Déploiement sur environnement CentOS . . . . .	7
3.2.3.2	Déploiement sur environnement Debian . . . . .	7
3.2.4	Matériel . . . . .	7
3.3	Récupération de la version . . . . .	7
3.3.1	Utilisation des dépôts open-source . . . . .	7
3.3.1.1	Repository pour environnement CentOS . . . . .	8
3.3.1.2	Repository pour environnement Debian . . . . .	8
3.3.2	Utilisation du package global d’installation . . . . .	8
<b>4</b>	<b>Procédures d’installation / mise à jour</b>	<b>9</b>
4.1	Vérifications préalables . . . . .	9
4.2	Procédures . . . . .	9
4.2.1	Multi-sites . . . . .	9
4.2.1.1	Procédure . . . . .	9
4.2.2	Configuration du déploiement . . . . .	9
4.2.2.1	Fichiers de déploiement . . . . .	10
4.2.2.2	Informations “plate-forme” . . . . .	10
4.2.2.3	Déclaration des secrets . . . . .	18
4.2.2.3.1	Cas des extra . . . . .	21
4.2.3	Gestion des certificats . . . . .	21
4.2.3.1	Cas 1 : Configuration développement / tests . . . . .	21
4.2.3.1.1	Procédure générale . . . . .	21

4.2.3.1.2	Génération des CA par les scripts Vitam . . . . .	22
4.2.3.1.3	Génération des certificats par les scripts Vitam . . . . .	22
4.2.3.1.4	Intégration de CA pour une offre swift . . . . .	22
4.2.3.1.5	Génération des magasins de certificats . . . . .	22
4.2.3.2	Cas 2 : Configuration production . . . . .	23
4.2.3.2.1	Procédure générale . . . . .	23
4.2.3.2.2	Intégration de certificats existants . . . . .	23
4.2.3.2.3	Intégration de CA pour une offre swift . . . . .	24
4.2.3.2.4	Génération des magasins de certificats . . . . .	25
4.2.4	Paramétrages supplémentaires . . . . .	25
4.2.4.1	Tuning JVM . . . . .	25
4.2.4.2	Paramétrage de l'antivirus (ingest-externe) . . . . .	25
4.2.4.3	Paramétrage des certificats externes (*-externe) . . . . .	26
4.2.4.4	Paramétrage de la centralisation des logs Vitam . . . . .	26
4.2.4.4.1	Gestion par Vitam . . . . .	26
4.2.4.4.2	Redirection des logs sur un SIEM tiers . . . . .	26
4.2.4.5	Fichiers complémentaires . . . . .	27
4.2.5	Procédure de première installation . . . . .	34
4.2.5.1	Déploiement . . . . .	34
4.2.5.1.1	Fichier de mot de passe . . . . .	34
4.2.5.1.2	Mise en place des repositories VITAM (optionnel) . . . . .	34
4.2.5.1.3	Génération des hostvars . . . . .	35
4.2.5.1.3.1	Cas 1 : Machines avec une seule interface réseau . . . . .	35
4.2.5.1.3.2	Cas 2 : Machines avec plusieurs interfaces réseau . . . . .	35
4.2.5.1.3.3	Vérification de la génération des hostvars . . . . .	36
4.2.5.1.4	Déploiement . . . . .	36
4.2.5.1.5	Extra . . . . .	36
4.2.5.1.5.1	ihm-recette . . . . .	36
4.2.5.1.5.2	extra complet . . . . .	36
4.2.6	Elements extras de l'installation . . . . .	37
4.2.6.1	Configuration des extra . . . . .	37
4.2.6.2	Déploiement des extra . . . . .	38
4.2.6.2.1	ihm-recette . . . . .	38
4.2.6.2.2	extra complet . . . . .	38
<b>5</b>	<b>Procédures de mise à jour</b> . . . . .	<b>39</b>
5.1	Reconfiguration . . . . .	39
5.1.1	Cas d'une modification du nombre de tenants . . . . .	39
5.1.2	Cas d'une modification des paramètres JVM . . . . .	39
<b>6</b>	<b>Post installation</b> . . . . .	<b>41</b>
6.1	Validation du déploiement . . . . .	41
6.1.1	Sécurisation du fichier vault_pass.txt . . . . .	41
6.1.2	Validation manuelle . . . . .	41
6.1.3	Validation via Consul . . . . .	41
6.1.4	Post-installation : administration fonctionnelle . . . . .	42
6.1.4.1	Cas du référentiel PRONOM . . . . .	42
6.2	Sauvegarde des éléments d'installation . . . . .	42
6.3	Troubleshooting . . . . .	42
6.3.1	Erreur au chargement des tableaux de bord Kibana . . . . .	43
6.4	Retour d'expérience / cas rencontrés . . . . .	43
6.4.1	Mongo-express ne se connecte pas à la base de données associée . . . . .	43
6.4.2	Elasticsearch possède des shard non alloués (état "UNASSIGNED") . . . . .	43
6.4.3	Elasticsearch possède des shards non initialisés (état "INITIALIZING") . . . . .	44

6.4.4	MongoDB semble lent . . . . .	44
6.4.5	Les shards de MongoDB semblent mal équilibrés . . . . .	45
<b>7</b>	<b>Annexes</b>	<b>47</b>
7.1	Vue d'ensemble de la gestion des certificats . . . . .	47
7.1.1	Liste des suites cryptographiques & protocoles supportés par Vitam . . . . .	47
7.1.2	Vue d'ensemble de la gestion des certificats . . . . .	48
7.1.3	Description de l'arborescence de la PKI . . . . .	48
7.1.4	Description de l'arborescence du répertoire deployment/environments/certs . . . . .	50
7.1.5	Description de l'arborescence du répertoire deployment/environments/keystores . . . . .	51
7.1.6	Fonctionnement des scripts de la PKI . . . . .	51
7.2	Cycle de vie des certificats . . . . .	51
7.3	Ansible & ssh . . . . .	53
7.3.1	Authentification du compte utilisateur utilisé pour la connexion SSH . . . . .	53
7.3.1.1	Par clé SSH avec passphrase . . . . .	53
7.3.1.2	Par login/mot de passe . . . . .	53
7.3.1.3	Par clé SSH sans passphrase . . . . .	53
7.3.2	Authentification des hôtes . . . . .	54
7.3.3	Elevation de privilèges . . . . .	54
7.3.3.1	Par sudo avec mot de passe . . . . .	54
7.3.3.2	Par su . . . . .	54
7.3.3.3	Par sudo sans mot de passe . . . . .	54
7.3.3.4	Déjà Root . . . . .	54
<b>8</b>	<b>Annexes</b>	<b>55</b>
	<b>Index</b>	<b>61</b>



---

## Introduction

---

### 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.





## 2.1 Information concernant les licences

La solution logicielle *VITAM* est publiée sous la licence [CeCILL 2.1](http://www.cecill.info/licences/Licence_CeCILL_V2.1-fr.html)<sup>1</sup> ; 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)<sup>2</sup>.

## 2.2 Documents de référence

### 2.2.1 Documents internes

Tableau 2.1 – Documents de référence VITAM

Nom	Lien
<i>DAT</i>	<a href="http://www.programmevitam.fr/ressources/DocCourante/html/archi">http://www.programmevitam.fr/ressources/DocCourante/html/archi</a>
<i>DIN</i>	<a href="http://www.programmevitam.fr/ressources/DocCourante/html/installation">http://www.programmevitam.fr/ressources/DocCourante/html/installation</a>
<i>DEX</i>	<a href="http://www.programmevitam.fr/ressources/DocCourante/html/exploitation">http://www.programmevitam.fr/ressources/DocCourante/html/exploitation</a>
Release notes	

### 2.2.2 Référentiels externes

## 2.3 Glossaire

**API** Application Programming Interface

**BDD** Base De Données

**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

1. [http://www.cecill.info/licences/Licence\\_CeCILL\\_V2.1-fr.html](http://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>

**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](#)<sup>3</sup>

**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](#)<sup>4</sup>

**NoSQL** Base de données non-basée sur un paradigme classique des bases relationnelles. [Définition NoSQL](#)<sup>5</sup>

**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](#)<sup>6</sup>

**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](#)<sup>7</sup>

**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

**TNR** Tests de Non-Régression

**VITAM** Valeurs Immatérielles Transférées aux Archives pour Mémoire

---

3. [https://fr.wikipedia.org/wiki/Domain\\_Name\\_System\\_Security\\_Extensions](https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions)

4. [https://fr.wikipedia.org/wiki/Attaque\\_de\\_l'\\_homme\\_du\\_milieu](https://fr.wikipedia.org/wiki/Attaque_de_l'_homme_du_milieu)

5. <https://fr.wikipedia.org/wiki/NoSQL>

6. [https://fr.wikipedia.org/wiki/Infrastructure\\_%C3%A0\\_cl%C3%A9s\\_publicues](https://fr.wikipedia.org/wiki/Infrastructure_%C3%A0_cl%C3%A9s_publicues)

7. [https://fr.wikipedia.org/wiki/Representational\\_state\\_transfer](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 **2.4** minimale et conseillée ; se référer à la [documentation ansible](http://docs.ansible.com/ansible/latest/intro_installation.html)<sup>8</sup> pour la procédure d'installation)
  - openssh-clients (client SSH utilisé par ansible)
  - 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)

---

8. [http://docs.ansible.com/ansible/latest/intro\\_installation.html](http://docs.ansible.com/ansible/latest/intro_installation.html)

**Note :** Se référer à la [documentation d'usage](#)<sup>9</sup> 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 47) 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 :

Veillez vous référer à la section [Vue d'ensemble de la gestion des certificats](#) (page 47) 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.

---

**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.

---

9. [http://docs.ansible.com/ansible/latest/intro\\_getting\\_started.html](http://docs.ansible.com/ansible/latest/intro_getting_started.html)

### 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 [VITAM<sup>10</sup>](https://github.com/ProgrammeVitam/vitam) , dans le répertoire `deployment`.

Les binaires de VITAM sont disponibles sur les dépôts [bintray<sup>11</sup>](https://bintray.com/programmevitam) ; ces dépôts doivent être correctement configurés sur la plate-forme cible avant toute installation.

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

11. <https://bintray.com/programmevitam>

### 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

```
[vitam-bintray--programmevitam-vitam-rpm-release-product]
name=vitam-bintray--programmevitam-vitam-rpm-release-product
baseurl=https://dl.bintray.com/programmevitam/vitam-rpm-release/centos/7/vitam-
↳product/<branche_vitam>/
gpgcheck=0
repo_gpgcheck=0
enabled=1

[vitam-bintray--programmevitam-vitam-rpm-release-external]
name=vitam-bintray--programmevitam-vitam-rpm-release-external
baseurl=https://dl.bintray.com/programmevitam/vitam-rpm-release/centos/7/vitam-
↳external/<branche_vitam>/
gpgcheck=0
repo_gpgcheck=0
enabled=1
```

---

**Note :** remplacer `<branche_vitam>` 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] https://dl.bintray.com/programmevitam/vitam-deb-release jessie_
↳vitam-product-<branche_vitam> vitam-external-<branche_vitam>
```

---

**Note :** remplacer `<branche_vitam>` 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.

---

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*, décompresser 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

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".

<p><b>Prudence :</b> noter de bien prendre en compte, selon le site, les variables ansible <code>vitam_site_name</code>. La variable <code>secret_plateforme</code> doit être commune sur les différents sites.</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

**Note :** Cas d'appel https au composant offer sur site secondaire. Dans ce cas, il convient également de rajouter, sur le site "primaire", les certificats relatifs ( CA du site secondaire ) à l'offre secondaire. Il faut également rapatrier sur site secondaire la CA et la certificat client du site primaire.

---

#### 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"

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
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

```



```
36 ##### VITAM services #####
37
38 # Group definition ; DO NOT MODIFY
39 [vitam:children]
40 zone-external
41 zone-access
42 zone-applicative
43 zone-storage
44 zone-data
45 zone-admin
46
47
48 ##### Zone externe
49
50
51 [zone-external:children]
52 hosts-ihm-demo
53 hosts-cerebro
54 hosts-ihm-recette
55
56 [hosts-ihm-demo]
57 # TODO: Put here servers where this service will be deployed : ihm-demo
58
59 [hosts-ihm-recette]
60 # TODO: Put here servers where this service will be deployed : ihm-recette (extra_
61 ↪feature)
62
63 [hosts-cerebro]
64 # TODO: Put here servers where this service will be deployed : vitam-elasticsearch-
65 ↪cerebro
66
67 ##### Zone access
68
69 # Group definition ; DO NOT MODIFY
70 [zone-access:children]
71 hosts-ingest-external
72 hosts-access-external
73
74 [hosts-ingest-external]
75 # TODO: Put here servers where this service will be deployed : ingest-external
76
77 [hosts-access-external]
78 # TODO: Put here servers where this service will be deployed : access-external
79
80
81 ##### Zone applicative
82
83 # Group definition ; DO NOT MODIFY
84 [zone-applicative:children]
85 hosts-ingest-internal
86 hosts-processing
87 hosts-worker
88 hosts-access-internal
89 hosts-metadata
90 hosts-functional-administration
91 hosts-logbook
```

```
92 hosts-workspace
93 hosts-storage-engine
94 hosts-security-internal
95
96 [hosts-security-internal]
97 # TODO: Put here servers where this service will be deployed : security-internal
98
99
100 [hosts-logbook]
101 # TODO: Put here servers where this service will be deployed : logbook
102
103
104 [hosts-workspace]
105 # TODO: Put here servers where this service will be deployed : workspace
106
107
108 [hosts-ingest-internal]
109 # TODO: Put here servers where this service will be deployed : ingest-internal
110
111
112 [hosts-access-internal]
113 # TODO: Put here servers where this service will be deployed : access-internal
114
115
116 [hosts-metadata]
117 # TODO: Put here servers where this service will be deployed : metadata
118
119
120 [hosts-functional-administration]
121 # TODO: Put here servers where this service will be deployed : functional-
↪administration
122
123
124 [hosts-processing]
125 # TODO: Put here servers where this service will be deployed : processing
126
127
128 [hosts-storage-engine]
129 # TODO: Put here servers where this service will be deployed : storage-engine
130
131
132 [hosts-worker]
133 # TODO: Put here servers where this service will be deployed : worker
134 # Optional parameter after each host : vitam_worker_capacity=<integer> ; please refer ↪
↪to your infrastructure for defining this number ; default is 1
135
136
137 ##### Zone storage
138
139 [zone-storage:children] # DO NOT MODIFY
140 hosts-storage-offer-default
141
142
143 [hosts-storage-offer-default]
144 # TODO: Put here servers where this service will be deployed : storage-offer-default
145 # LIMIT : only 1 offer per machine and 1 machine per offer
146 # Mandatory param for each offer is offer_conf and points to offer_opts.yml & vault-
↪vitam.yml (with same tree)
```

```

147 # hostname-offre-1.vitam offer_conf=offer-swift-1
148 # for filesystem
149 # hostname-offre-2.vitam offer_conf=offer-fs-1
150
151 [hosts-mongodb-offer:children]
152 hosts-mongos-data
153 hosts-mongoc-data
154 hosts-mongod-data
155
156 [hosts-mongos-offer]
157 # TODO: put here servers where this service will be deployed : mongos cluster for
↳storage offers
158 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the
↳offer_conf configuration)
159 # Example (for a more complete one, see the one in the group hosts-mongos-data) :
160 # vitam-mongo-swift-offer-01 mongo_cluster_name=offer-swift-1
161 # vitam-mongo-swift-offer-02 mongo_cluster_name=offer-swift-1
162 # vitam-mongo-fs-offer-01 mongo_cluster_name=offer-fs-1
163 # vitam-mongo-fs-offer-02 mongo_cluster_name=offer-fs-1
164
165 [hosts-mongoc-offer]
166 # TODO: put here servers where this service will be deployed : mongoc cluster for
↳storage offers
167 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the
↳offer_conf configuration)
168 # Optional param : mandatory for 1 node of the shard, some init commands will be
↳executed on it
169 # Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add
↳this paramter on a mongo_rs_bootstrap node
170 # Example :
171 # vitam-mongo-swift-offer-01 mongo_cluster_name=offer-swift-1
↳mongo_rs_bootstrap=true
172 # vitam-mongo-swift-offer-02 mongo_cluster_name=offer-swift-1
173 # vitam-swift-offer mongo_cluster_name=offer-swift-1
↳mongo_arbiter=true
174 # vitam-mongo-fs-offer-01 mongo_cluster_name=offer-fs-1
↳mongo_rs_bootstrap=true
175 # vitam-mongo-fs-offer-02 mongo_cluster_name=offer-fs-1
176 # vitam-fs-offer mongo_cluster_name=offer-fs-1
↳mongo_arbiter=true
177
178 [hosts-mongod-offer]
179 # TODO: put here servers where this service will be deployed : mongod cluster for
↳storage offers
180 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the
↳offer_conf configuration)
181 # Mandatory param : id of the current shard, increment by 1 from 0 to n
182 # Optional param : mandatory for 1 node of the shard, some init commands will be
↳executed on it
183 # Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add
↳this paramter on a mongo_rs_bootstrap node
184 # Example :
185 # vitam-mongo-swift-offer-01 mongo_cluster_name=offer-swift-1 mongo_shard_id=0
↳
↳ mongo_rs_bootstrap=true
186 # vitam-mongo-swift-offer-02 mongo_cluster_name=offer-swift-1 mongo_shard_id=0
187 # vitam-swift-offer mongo_cluster_name=offer-swift-1 mongo_shard_id=0
↳
↳ mongo_arbiter=true
188 # vitam-mongo-fs-offer-01 mongo_cluster_name=offer-fs-1 mongo_shard_id=0
↳
↳ mongo_rs_bootstrap=true

```

```

189 # vitam-mongo-fs-offer-02      mongo_cluster_name=offer-fs-1      mongo_shard_id=0
190 # vitam-fs-offer              mongo_cluster_name=offer-fs-1      mongo_shard_id=0
191 ↪                               mongo_arbiter=true
192 ##### Zone data
193
194 # Group definition ; DO NOT MODIFY
195 [zone-data:children]
196 hosts-elasticsearch-data
197 hosts-mongodb-data
198
199 [hosts-elasticsearch-data]
200 # TODO: Put here servers where this service will be deployed : elasticsearch-data
201 ↪cluster
202 # 2 params available for huge environments (parameter to be declared after each
203 ↪server) :
204 #   is_data=true/false
205 #   is_master=true/false
206 #   other options are not handled yet
207 # defaults are set to true
208 # Examples :
209 # server1 is_master=true is_data=false
210 # server2 is_master=false is_data=true
211 # More explanation here : https://www.elastic.co/guide/en/elasticsearch/reference/5.6/
212 ↪modules-node.html
213
214 # Group definition ; DO NOT MODIFY
215 [hosts-mongodb-data:children]
216 hosts-mongos-data
217 hosts-mongoc-data
218 hosts-mongod-data
219
220 [hosts-mongos-data]
221 # TODO: Put here servers where this service will be deployed : mongos cluster
222 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
223 # Example :
224 # vitam-mdbs-01      mongo_cluster_name=mongo-data
225 # vitam-mdbs-01      mongo_cluster_name=mongo-data
226 # vitam-mdbs-01      mongo_cluster_name=mongo-data
227
228 [hosts-mongoc-data]
229 # TODO: Put here servers where this service will be deployed : mongoc cluster
230 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
231 # Optional param : mandatory for 1 node of the shard, some init commands will be
232 ↪executed on it
233 # Example :
234 # vitam-mdbc-01      mongo_cluster_name=mongo-data      mongo_rs_
235 ↪bootstrap=true
236 # vitam-mdbc-01      mongo_cluster_name=mongo-data
237 # vitam-mdbc-01      mongo_cluster_name=mongo-data
238
239 [hosts-mongod-data]
240 # TODO: Put here servers where this service will be deployed : mongod cluster
241 # Each replica_set should have an odd number of members (2n + 1)
242 # Reminder: For Vitam, one mongodb shard is using one replica_set
243 # Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
244 # Mandatory param : id of the current shard, increment by 1 from 0 to n

```

```

241 # Optional param : mandatory for 1 node of the shard, some init commands will be_
    ↳executed on it
242 # Example:
243 # vitam-mdbd-01 mongo_cluster_name=mongo-data mongo_shard_id=0 mongo_rs_
    ↳bootstrap=true
244 # vitam-mdbd-02 mongo_cluster_name=mongo-data mongo_shard_id=0
245 # vitam-mdbd-03 mongo_cluster_name=mongo-data mongo_shard_id=0
246 # vitam-mdbd-04 mongo_cluster_name=mongo-data mongo_shard_id=1 mongo_rs_
    ↳bootstrap=true
247 # vitam-mdbd-05 mongo_cluster_name=mongo-data mongo_shard_id=1
248 # vitam-mdbd-06 mongo_cluster_name=mongo-data mongo_shard_id=1
249
250 ##### Zone admin
251
252 # Group definition ; DO NOT MODIFY
253 [zone-admin:children]
254 hosts-consul-server
255 hosts-kibana-data
256 log-servers
257 hosts-elasticsearch-log
258
259 [hosts-consul-server]
260 # TODO: Put here servers where this service will be deployed : consul
261
262 [hosts-kibana-data]
263 # TODO: Put here servers where this service will be deployed : kibana (for data_
    ↳cluster)
264
265 [log-servers:children]
266 hosts-kibana-log
267 hosts-logstash
268
269
270 [hosts-kibana-log]
271 # TODO: Put here servers where this service will be deployed : kibana (for log_
    ↳cluster)
272
273 [hosts-logstash]
274 # TODO: Put here servers where this service will be deployed : logstash
275
276
277 [hosts-elasticsearch-log]
278 # TODO: Put here servers where this service will be deployed : elasticsearch-log_
    ↳cluster
279
280 ##### Global vars #####
281
282 [hosts:vars]
283
284 # =====
285 # VITAM
286 # =====
287
288 # Declare user for ansible on target machines
289 ansible_ssh_user=
290 # Can target user become as root ? ; true is required by VITAM (usage of a sudoer is_
    ↳mandatory)
291 ansible_become=true

```

```
292
293 # Related to Consul ; apply in a table your DNS server(s)
294 # Example : dns_servers=["8.8.8.8","8.8.4.4"]
295 dns_servers=
296
297 # Vitam tenants to create
298 vitam_tenant_ids=[0,1,2]
299 vitam_tenant_admin=1
300
301 ### Logback configuration ###
302 # Days before deleting logback log files (java & access logs for vitam components)
303 days_to_delete_logback_logfiles=
304
305 # Configuration for Curator
306 #         Days before deletion on log management cluster; 365 for production_
    ↪environment
307 days_to_delete_logstash_indexes=
308 #         Days before closing "old" indexes on log management cluster; 30 for_
    ↪production environment
309 days_to_close_logstash_indexes=
310
311 # Define local Consul datacenter name
312 vitam_site_name=prod-dc1
313 # EXAMPLE : vitam_site_name = prod-dc1
314 # check whether on primary site (true) or secondary (false)
315 primary_site=true
316
317
318 # =====
319 # EXTRA
320 # =====
321 # Environment (defines title in extra on reverse homepage). Variable is DEPRECATED !
322 #environnement=
323
324 ### vitam-itest repository ###
325 vitam_tests_branch=master
326 vitam_tests_gitrepo_protocol=
327 vitam_tests_gitrepo_baseurl=
328 vitam_tests_gitrepo_url=
329
330 # Curator configuration
331 #         Days before deletion for packetbeat index only on log management cluster
332 days_to_delete_packetbeat_indexes=5
333 #         Days before deletion for metricbeat index only on log management cluster; 30_
    ↪for production environment
334 days_to_delete_metricbeat_indexes=30
335 # Days before closing metrics elasticsearch indexes
336 days_to_close_metrics_indexes=7
337 # Days before deleting metrics elasticsearch indexes
338 days_to_delete_metrics_indexes=30
339 days_to_delete_packetbeat_indexes=20
340 days_to_delete_metricbeat_indexes=20
341
342
343
344 # Used when VITAM is behind a reverse proxy (provides configuration for reverse proxy_
    ↪&& displayed in header page)
345 vitam_reverse_external_dns=
```

```

346 # For reverse proxy use
347 reverse_proxy_port=80
348 # http_proxy env var to use ; has to be declared even if empty
349 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.

---

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-
9 ↪external/clients
10 # vitam-admin-int is mandatory for internal use (PRONOM upload)
11 admin_context_certs: [ "ihm-demo/ihm-demo.crt", "ihm-recette/ihm-recette.crt",
12 ↪"reverse/reverse.crt", "vitam-admin-int/vitam-admin-int.crt" ]
13 # Indicate here all the personal certificates relative paths under {{inventory_dir}}/
14 ↪certs/client-vitam-users/clients
15 admin_personal_certs: [ "userOK.crt" ]
16
17 # Admin security profile name
18 admin_security_profile: "admin-security-profile"
19
20 admin_basic_auth_user: "adminUser"

```

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_
2 ↪necessary
3 # Strategy order (1st has to be the preferred one)
4 vitam_strategy:
5   - name: offer-fs-1
6     referent: true
7     vitam_site_name: prod-dc2
8   # - name: offer_swift_1
9   # Example :
10  # - name: distant
11  #   referent: true
12  #   vitam_site_name: distant-dc2
13
14 # DON'T forget to add associated passwords in vault-vitam.yml with same tree_
15 ↪when using provider openstack-swift*
16 # ATTENTION !!! Each offer has to have a distinct name, except for clusters_
17 ↪binding a same physical storage
18 # WARNING : for offer names, please only use [a-z][a-z0-9-]* pattern
19 vitam_offers:

```

```
17 offer-fs-1:
18   # param can be filesystem or filesystem-hash
19   provider: filesystem
20 offer-swift-1:
21   # provider : openstack-swift for v1 or openstack-swift-v3 for v3
22   provider: openstack-swift
23   # keystoneEndPoint : URL de connexion à keystone
24   keystoneEndPoint: http://hostname-rados-gw:port/auth/1.0
25   # deprecated
26   keystone_auth_url: http://hostname-rados-gw:port/auth/1.0
27   # swiftUid : domaine OpenStack dans lequel l'utilisateur est enregistré
28   swift_uid: domaine
29   # swiftSubUser : identifiant de l'utilisateur
30   swift_subuser: utilisateur
31   # cephMode : doit être à false si offre v3 ; true si offre v1
32   cephMode: false
33   # projectName : tenant openstack
34   projectName: monTenant
35
36
37
38 # example_swift_v1:
39 #   provider: openstack-swift
40 #   keystoneEndPoint: https://keystone/auth/1.0
41 #   swift_uid: tenant$user # <tenant>${user}
42 #   swift_subuser: subuser
43 #   cephMode: true
44 # example_swift_v3:
45 #   provider: openstack-swift-v3
46 #   keystoneEndPoint: https://keystone/v3
47 #   swift_uid: domaine
48 #   swift_subuser: user
49 #   cephMode: false
50 #   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 défini, c'est la valeur de la variable `vitam_site_name` définie dans l'inventaire.

---

**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.

### 4.2.2.3 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>`.

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 # Cerebro key
6 cerebro_secret_key: tGz28hJkiW[p@a34G
7
8 # The consul key must be 16-bytes, Base64 encoded: https://www.consul.io/docs/
  ↳agent/encryption.html
9 # You can generate it with the "consul keygen" command
10 # Or you can use this script: deployment/pki/scripts/generate_consul_key.sh
11 consul_encrypt: Biz14ohqN4HtvZmrXp3N4A==
12
13 mongodb:
14   mongo-data:
15     passphrase: mongogo
16     admin:
17       user: vitamdb-admin
18       password: azerty
19     localadmin:
20       user: vitamdb-localadmin
21       password: qwerty
22     metadata:
23       user: metadata
24       password: azerty1
25     logbook:
26       user: logbook
27       password: azerty2
28     functionalAdmin:
29       user: functional-admin
30       password: azerty3
31     securityInternal:
32       user: security-internal
33       password: azerty4
34   offer-fs-1:
35     passphrase: mongogo
36     admin:
37       user: vitamdb-admin
38       password: azerty
39     localadmin:
40       user: vitamdb-localadmin
41       password: qwerty
42     offer:

```

```
43     user: offer
44     password: azerty5
45 offer-fs-2:
46     passphrase: mongogo
47     admin:
48         user: vitamdb-admin
49         password: azerty
50     localadmin:
51         user: vitamdb-localadmin
52         password: qwerty
53     offer:
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_admin:
70         login: aadmin
71         password: aadmin1234
72         role: admin
73     - vitam_user:
74         login: user
75         password: user1234
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 ldap_authentication:
86     ldap_pwd: "admin"
87
88 admin_basic_auth_password: adminPassword
```

- 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
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

```

**Avertissement :** 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 les 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 47).

### 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 !

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.

**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 `environnements/<fichier d'inventaire>` (cf. *Informations "plate-forme"* (page 10)) 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 `environnements/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 `environnements/certs/vault-certs.yml`

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

### 4.2.3.1.4 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 `environnements/certs/server/ca/`

### 4.2.3.1.5 Génération des magasins de certificats

En prérequis, les certificats et les autorités de certification 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 : `environnements/group_vars/all/vault-keystores.yml`, décrit dans la section *Déclaration des secrets* (page 18).

Lancer le script :

```
./generate_stores.sh
```

Ce script génère sous `environnements/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 : `environnements/group_vars/all/vault-keystores.yml`

### 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 :

- 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 `environnements/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.

---

---

**Note :** Ne pas oublier, dans les certificats, de prendre en compte des alias "web" (`subjectAltName`).

---

#### 4.2.3.2.2 Intégration de certificats existants

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

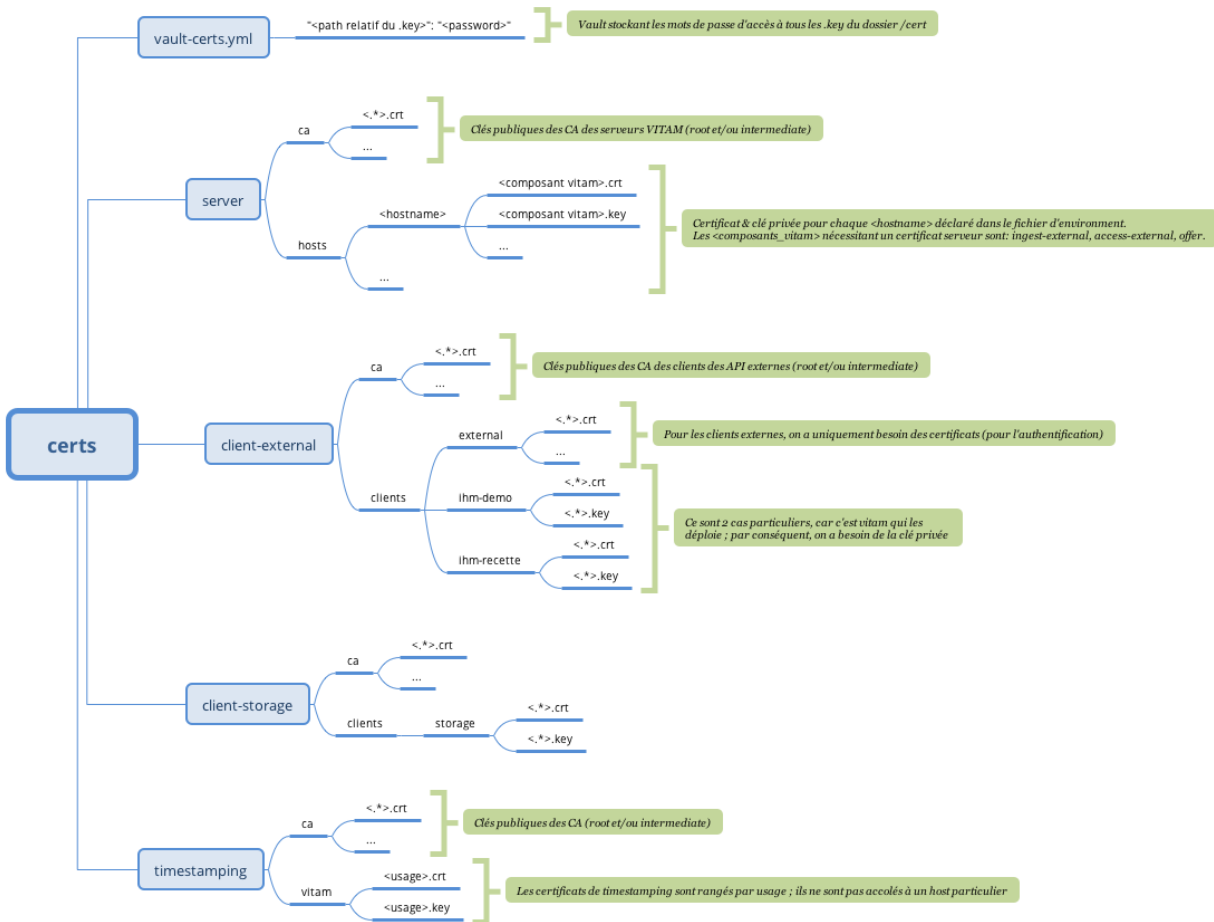


Fig. 4.1 – Vue détaillée de l'arborescence des certificats

**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 : `environnements/certs/vault-certs.yml`

Dans le cas d'ajout de certificats *SIA* externes, éditer le fichier `environnements/group_vars/all/vitam_security.yml` et ajouter le(s) entrée(s) supplémentaire(s) (sous forme répertoire/fichier.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*.

Pour modifier/créer un vault ansible, se référer à la documentation sur [cette url](http://docs.ansible.com/ansible/playbooks_vault.html)<sup>12</sup>.

#### 4.2.3.2.3 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 `environnements/certs/server/ca/`

12. [http://docs.ansible.com/ansible/playbooks\\_vault.html](http://docs.ansible.com/ansible/playbooks_vault.html)

#### 4.2.3.2.4 Génération des magasins de certificats

En prérequis, les certificats et les autorités de certification 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 : `environnements/group_vars/all/vault-keystores.yml`, décrit dans la section *Déclaration des secrets* (page 18).

Lancer le script :

```
./generate_stores.sh
```

Ce script génère sous `environnements/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 : `environnements/group_vars/all/vault-keystores.yml`

## 4.2.4 Paramétrages supplémentaires

### 4.2.4.1 Tuning JVM

---

**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 `environnements/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 `environnements/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_environment`.

### 4.2.4.3 Paramétrage des certificats externes (\*-externe)

Se reporter au chapitre dédié à la gestion des certificats : *Gestion des certificats* (page 21)

### 4.2.4.4 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.

#### 4.2.4.4.1 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

#### 4.2.4.4.2 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](#)<sup>13</sup>.

---

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



**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).

#### 4.2.4.5 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 :

```

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     users:
16         vitam: "vitam"
17         vitamdb: "vitamdb"
18         group: "vitam"
19     services:
20         log_level: WARN
21         start_timeout: 150
22         stop_timeout: 3600
23         port_service_timeout: 86400
24     # Filter for the vitam package version to install
25     # FIXME : commented as has to be removed because doesn't work under Debain
26     #package_version: "*"
27     ### Trust X-SSL-CLIENT-CERT header for external api auth ? (true | false) ###
28     vitam_ssl_user_header: true
29     # syslog_facility
30     syslog_facility: local0
31
32  ### consul ###
33  # FIXME: Consul à la racine pour le moment à cause de problèmes de récursivité_
34  ↪dans le parsing yaml
35  # TODO : consul_domain should be in inventory as choosable by customer
36  consul_domain: consul
37  consul_component: consul
38  consul_folder_conf: "{{ vitam_defaults.folder.root_path }}/conf/{{ consul_
39  ↪component }}"
40
41  # Workspace should be useless but storage have a dependency to it...
42  vitam_secondary_site_components: [ "consul" , "logbook" , "metadata" ,
43  ↪"functional-administration" , "storage" , "storageofferdefault" , "offer" ,
44  ↪"elasticsearch-log" , "elasticsearch-data" , "logstash" , "kibana" , "mongoc" ,
45  ↪"mongod" , "mongos" , "vitam-metadata-reconstruction.timer" , "vitam-logbook-
46  ↪reconstruction.timer" , "vitam-functional-administration-reconstruction.timer"
47  ↪"library" , "cerebro" ]

```

```
41
42
43 ### Composants Vitam ###
44
45 vitam:
46   accessexternal:
47     vitam_component: access-external
48     host: "access-external.service.{{ consul_domain }}"
49     port_admin: 28102
50     port_service: 8444
51     baseuri: "access-external"
52     https_enabled: true
53     secret_platform: "false"
54   accessinternal:
55     vitam_component: access-internal
56     host: "access-internal.service.{{ consul_domain }}"
57     port_service: 8101
58     port_admin: 28101
59     baseuri: "access-internal"
60     https_enabled: false
61     secret_platform: "true"
62   functional_administration:
63     vitam_component: functional-administration
64     host: "functional-administration.service.{{ consul_domain }}"
65     port_service: 8004
66     port_admin: 18004
67     baseuri: "functional-administration"
68     https_enabled: false
69     secret_platform: "true"
70     cluster_name: "{{ elasticsearch.data.cluster_name }}"
71   ingestexternal:
72     vitam_component: ingest-external
73     host: "ingest-external.service.{{ consul_domain }}"
74     port_admin: 28001
75     port_service: 8443
76     baseuri: "ingest-external"
77     https_enabled: true
78     secret_platform: "false"
79     antivirus: "clamav"
80     # Directory where files should be placed for local ingest
81     upload_dir: "/vitam/data/ingest-external/upload"
82     # Directory where successful ingested files will be moved to
83     success_dir: "/vitam/data/ingest-external/upload/success"
84     # Directory where failed ingested files will be moved to
85     fail_dir: "/vitam/data/ingest-external/upload/failure"
86     # Action done to file after local ingest (see below for further
87     →information)
88     upload_final_action: "MOVE"
89   ingestinternal:
90     vitam_component: ingest-internal
91     host: "ingest-internal.service.{{ consul_domain }}"
92     port_service: 8100
93     port_admin: 28100
94     baseuri: "ingest-internal"
95     https_enabled: false
96     secret_platform: "true"
97   ihm_demo:
98     vitam_component: "ihm-demo"
```

```

98     host: "ihm-demo.service.{{ consul_domain }}"
99     port_service: 8002
100    port_admin: 28002
101    baseurl: "/ihm-demo"
102    static_content: "{{ vitam_defaults.folder.root_path }}/app/ihm-demo/v2"
103    baseuri: "ihm-demo"
104    https_enabled: false
105    secret_platform: "false"
106    # User session timeout in milliseconds (for shiro)
107    session_timeout: 1800000
108    secure_cookie: false
109    # Specify here the realms you want to use for authentication in ihm-demo
110    # You can set multiple realms, one per line
111    # With multiple realms, the user will be able to choose between the
↪allowed realms
112    # Example: authentication_realms:
113    #             - x509Realm
114    #             - ldapRealm
115    # Authorized values:
116    # x509Realm: certificate
117    # iniRealm: ini file
118    # ldapRealm: ldap
119    authentication_realms:
120        # - x509Realm
121        - iniRealm
122        # - ldapRealm
123    logbook:
124        vitam_component: logbook
125        host: "logbook.service.{{ consul_domain }}"
126        port_service: 9002
127        port_admin: 29002
128        baseuri: "logbook"
129        https_enabled: false
130        secret_platform: "true"
131        cluster_name: "{{ elasticsearch.data.cluster_name }}"
132        # Overlap delay (in seconds) for logbook operation & lifecycle
↪traceability events.
133        # Set it to a reasonable delay to cover max clock difference across
↪servers
134        operationTraceabilityOverlapDelay: 300
135        lifecycleTraceabilityOverlapDelay: 300
136        disablePurgeForTempLFC: false
137    metadata:
138        vitam_component: metadata
139        host: "metadata.service.{{ consul_domain }}"
140        port_service: 8200
141        port_admin: 28200
142        baseuri: "metadata"
143        https_enabled: false
144        secret_platform: "true"
145        cluster_name: "{{ elasticsearch.data.cluster_name }}"
146    processing:
147        vitam_component: processing
148        host: "processing.service.{{ consul_domain }}"
149        port_service: 8203
150        port_admin: 28203
151        baseuri: "processing"
152        https_enabled: false

```

```

153     secret_platform: "true"
154 security_internal:
155     vitam_component: security-internal
156     host: "security-internal.service.{{ consul_domain }}"
157     port_service: 8005
158     port_admin: 28005
159     baseuri: "security-internal"
160     https_enabled: false
161     secret_platform: "true"
162 storageengine:
163     vitam_component: storage
164     host: "storage.service.{{ consul_domain }}"
165     port_service: 9102
166     port_admin: 29102
167     baseuri: "storage-engine"
168     https_enabled: false
169     secret_platform: "true"
170     storageTraceabilityOverlapDelay: 300
171 storageofferdefault:
172     vitam_component: "offer"
173     port_service: 9900
174     port_admin: 29900
175     baseuri: "storage-offer-default"
176     https_enabled: false
177     secret_platform: "true"
178 worker:
179     vitam_component: worker
180     port_service: 9104
181     port_admin: 29104
182     baseuri: "worker"
183     https_enabled: false
184     secret_platform: "true"
185 workspace:
186     vitam_component: workspace
187     host: "workspace.service.{{ consul_domain }}"
188     port_service: 8201
189     port_admin: 28201
190     baseuri: "workspace"
191     https_enabled: false
192     secret_platform: "true"
193
194 # ingestexternal:
195 # upload_final_action can be set to three different values (lower or upper case,
196 # →does not matter)
197 # MOVE : After upload, the local file will be moved to either success_dir or,
198 # →fail_dir depending on the status of the ingest towards ingest-internal
199 # DELETE : After upload, the local file will be deleted if the upload succeeded
200 # NONE : After upload, nothing will be done to the local file (default option,
201 # →set if the value entered for upload_final_action does not exist)

```

- environments /group\_vars/all/cots\_vars.yml, comme suit :

```

1 ---
2 consul_remote_sites:
3     # - dc2:
4     #   domain_name: "dc2"
5     #   wan: ["10.10.10.10", "1.1.1.1"]
6     # - dc3:

```

```

7   #   domain_name: "site3"
8   #   wan: ["10.10.10.11", "1.1.1.1"]
9
10  elasticsearch:
11    log:
12      host: "elasticsearch-log.service.{{ consul_domain }}"
13      port_http: "9201"
14      port_tcp: "9301"
15      groupe: "log"
16      baseuri: "elasticsearch-log"
17      cluster_name: "elasticsearch-log"
18      https_enabled: false
19    data:
20      host: "elasticsearch-data.service.{{ consul_domain }}"
21      port_http: "9200"
22      port_tcp: "9300"
23      groupe: "data"
24      baseuri: "elasticsearch-data"
25      cluster_name: "elasticsearch-data"
26      https_enabled: false
27
28  mongodb:
29    mongos_port: 27017
30    mongoc_port: 27018
31    mongod_port: 27019
32    mongo_authentication: "true"
33    host: "mongos.service.{{ consul_domain }}"
34
35  logstash:
36    host: "logstash.service.{{ consul_domain }}"
37    user: logstash
38    port: 10514
39    rest_port: 20514
40
41  curator:
42    log:
43      metrics:
44        close: 5
45        delete: 30
46      logstash:
47        close: 5
48        delete: 30
49      metricbeat:
50        close: 5
51        delete: 30
52      packetbeat:
53        close: 5
54        delete: 30
55
56  kibana:
57    log:
58      baseuri: "kibana_log"
59      groupe: "log"
60      port: 5601
61      # pour index logstash-*
62      metrics:
63        shards: 5
64        replica: 1

```

```

65     # pour index metrics-vitam-*
66     logs:
67         shards: 5
68         replica: 1
69     # KWA FIXME : changing port doesn't work, yet (not taken into account in
↳kibana configuration)
70     data:
71         baseuri: "kibana_data"
72         groupe: "data"
73         port: 5601
74         shards: 10
75         replica: 2
76
77 cerebro:
78     baseuri: "cerebro"
79     port: 9000
80
81 siegfried:
82     port: 19000
83
84 clamav:
85     port: 3310
86     # frequency freshclam for database update per day (from 0 to 24 - 24 meaning
↳hourly check)
87     db_update_periodicity: 1
88
89 mongo_express:
90     baseuri: "mongo-express"
91
92 ldap_authentication:
93     ldap_server: "% if groups['ldap']|length > 0 %}{ groups['ldap']|first }}{%
↳endif %}"
94     ldap_port: "389"
95     ldap_base: "dc=programmevitam,dc=fr"
96     ldap_login: "cn=Manager,dc=programmevitam,dc=fr"
97     uid_field: "uid"
98     ldap_userDn_Template: "uid={0},ou=people,dc=programmevitam,dc=fr"
99     ldap_group_request: "(&(objectClass=groupOfNames)(member={0}))"
100    ldap_admin_group: "cn=admin,ou=groups,dc=programmevitam, dc=fr"
101    ldap_user_group: "cn=user,ou=groups,dc=programmevitam, dc=fr"
102    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.

- `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 ingestinternal:
15     jvm_opts:
16         # memory: "-Xms512m -Xmx512m"
17         # gc: ""
18         # java: ""
19 ingestexternal:
20     jvm_opts:
21         # memory: "-Xms512m -Xmx512m"
22         # gc: ""
23         # java: ""
24 metadata:
25     jvm_opts:
26         # memory: "-Xms512m -Xmx512m"
27         # gc: ""
28         # java: ""
29 ihm_demo:
30     jvm_opts:
31         # memory: "-Xms512m -Xmx512m"
32         # gc: ""
33         # java: ""
34 ihm_recette:
35     jvm_opts:
36         # memory: "-Xms512m -Xmx512m"
37         # gc: ""
38         # java: ""
39 logbook:
40     jvm_opts:
41         # memory: "-Xms512m -Xmx512m"
42         # gc: ""
43         # java: ""
44 workspace:
45     jvm_opts:
46         # memory: "-Xms512m -Xmx512m"
47         # gc: ""
48         # java: ""
49 processing:
50     jvm_opts:
51         # memory: "-Xms512m -Xmx512m"
52         # gc: ""
53         # java: ""
54 worker:
55     jvm_opts:
56         # memory: "-Xms512m -Xmx512m"
57         # gc: ""
58         # java: ""
59 storageengine:
60     jvm_opts:
61         # memory: "-Xms512m -Xmx512m"
62         # gc: ""
63         # java: ""
64 storageofferdefault:
65     jvm_opts:
66         # memory: "-Xms512m -Xmx512m"
67         # gc: ""

```

```
68     # java: ""
69     functional_administration:
70       jvm_opts:
71         # memory: "-Xms512m -Xmx512m"
72         # gc: ""
73         # java: ""
74     security_internal:
75       jvm_opts:
76         # memory: "-Xms512m -Xmx512m"
77         # gc: ""
78         # java: ""
79     library:
80       jvm_opts:
81         memory: "-Xms32m -Xmx128m"
82         # gc: ""
83         # 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 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`).

#### 4.2.5.1.2 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: "./"
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-”.

#### 4.2.5.1.3 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

##### 4.2.5.1.3.1 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.

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

##### 4.2.5.1.3.2 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`.

---

**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.

---

### 4.2.5.1.3.3 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.

### 4.2.5.1.4 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).

---

### 4.2.5.1.5 Extra

Deux playbooks d'extra sont fournis pour usage "tel quel".

#### 4.2.5.1.5.1 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
```

#### 4.2.5.1.5.2 extra complet

Ce playbook permet d'installer :

- topbeat
- packetbeat

- 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 les documentations du projet
- le composant *VITAM* ihm-recette (nécessite un accès à un répertoire “partagé” pour récupérer les jeux de tests)
- un reverse proxy, afin de simplifier les appels aux composants

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

## 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.

### 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    library:
20      vitam_component: library
21      host: "library.service.{{consul_domain}}"
22      port_service: 8090
23      port_admin: 28090
24      baseuri: "doc"
25      https_enabled: false
26      secret_platform: "false"
27
28 docker_opts:
29   registry_httponly: yes
30   vitam_docker_tag: latest
```

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
```

#### 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 :** La mise à jour depuis une version précédente n'est pas supportée dans cette version de la solution logicielle VITAM.

### 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 25)

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.

---

## Post installation

---

### 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 hosts-consul-server>:8500/ui`

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

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

**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 pronom](#)<sup>14</sup> ) 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.

#### 6.1.4.1 Cas du référentiel PRONOM

Un playbook a été créé pour charger le référentiel PRONOM dans une version compatible avec celui intégré dans le composant Siegfried.

Ce playbook n’est à passer que si aucun référentiel PRONOM n’a été chargé, permettant d’accélérer l’utilisation de VITAM.

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

**Prudence :** le playbook termine en erreur (code HTTP 403) si un référentiel PRONOM a déjà été chargé.

## 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.

---

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



### 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.

## 6.4 Retour d'expérience / cas rencontrés

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

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

### 6.4.2 Elasticsearch possède des shard non alloués (état "UNASSIGNED")

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

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

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

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

Sur tous ces sujets, Cf. la [documentation officielle](#) <sup>15</sup>.

### 6.4.3 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](#) <sup>16</sup>. 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.

### 6.4.4 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` <sup>17</sup> et `mongotop` <sup>18</sup>.

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](#) <sup>19</sup> :

```
mongo --host <ip_service> --port 27019 --username vitamdb-localadmin --password  
↳<password ; défaut : qwerty> --authenticationDatabase admin  
> rs.printSlaveReplicationInfo()  
> rs.printReplicationInfo()  
> db.runCommand( { serverStatus: 1 } )
```

D'autres commandes plus complètes existent et permettent d'avoir plus d'informations, mais leur analyse est plus complexe :

```
# returns a variety of storage statistics for a given collection  
> use metadata  
> db.stats()  
> db.runCommand( { collStats: "Unit" } )
```

Enfin, un outil est disponible en standard afin de mesurer des performances des lecture/écritures avec des patterns proches de ceux utilisés par la base de données (`mongoperf` <sup>20</sup>) :

15. <https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-reroute.html>
16. <https://www.elastic.co/guide/en/elasticsearch/reference/1.4/states.html>
17. <https://docs.mongodb.com/manual/reference/program/mongostat/>
18. <https://docs.mongodb.com/manual/reference/program/mongotop/>
19. <https://docs.mongodb.com/manual/tutorial/troubleshoot-replica-sets>
20. <https://docs.mongodb.com/manual/reference/program/mongoperf/>

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

### 6.4.5 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.



---

## 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

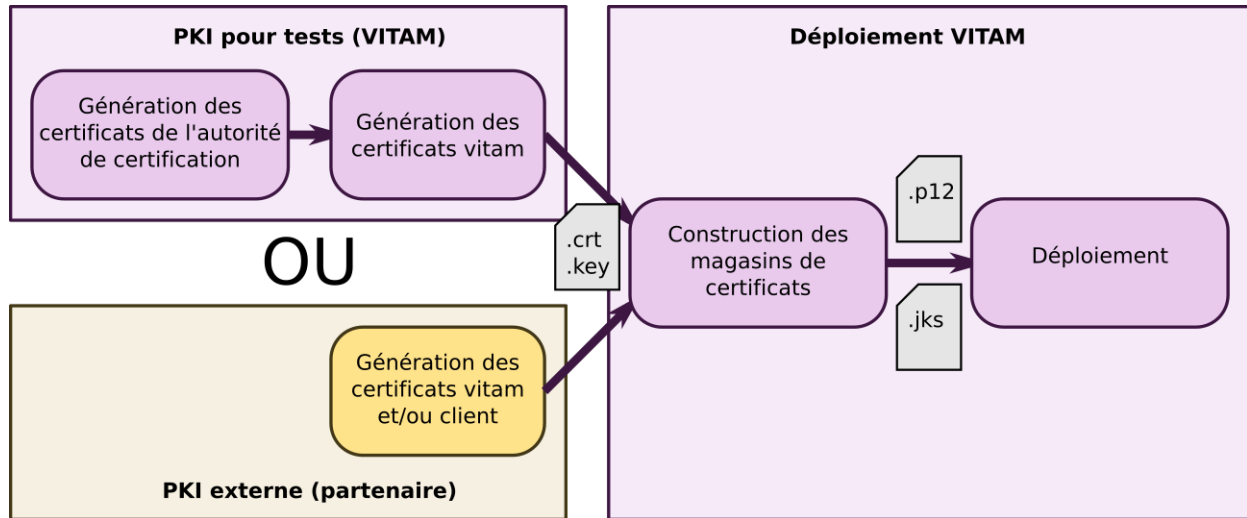


Fig. 7.1 – Vue d'ensemble de la gestion des certificats au déploiement

## 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 51)

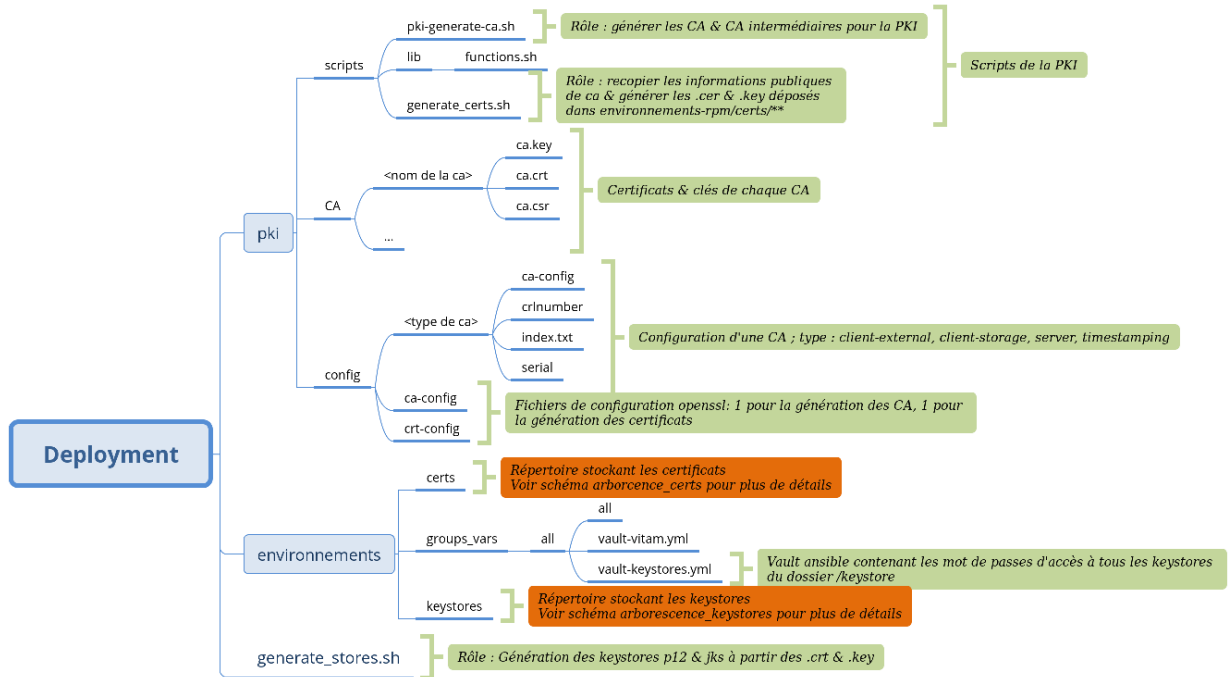


Fig. 7.2 – Vue l'arborescence de la PKI Vitam

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

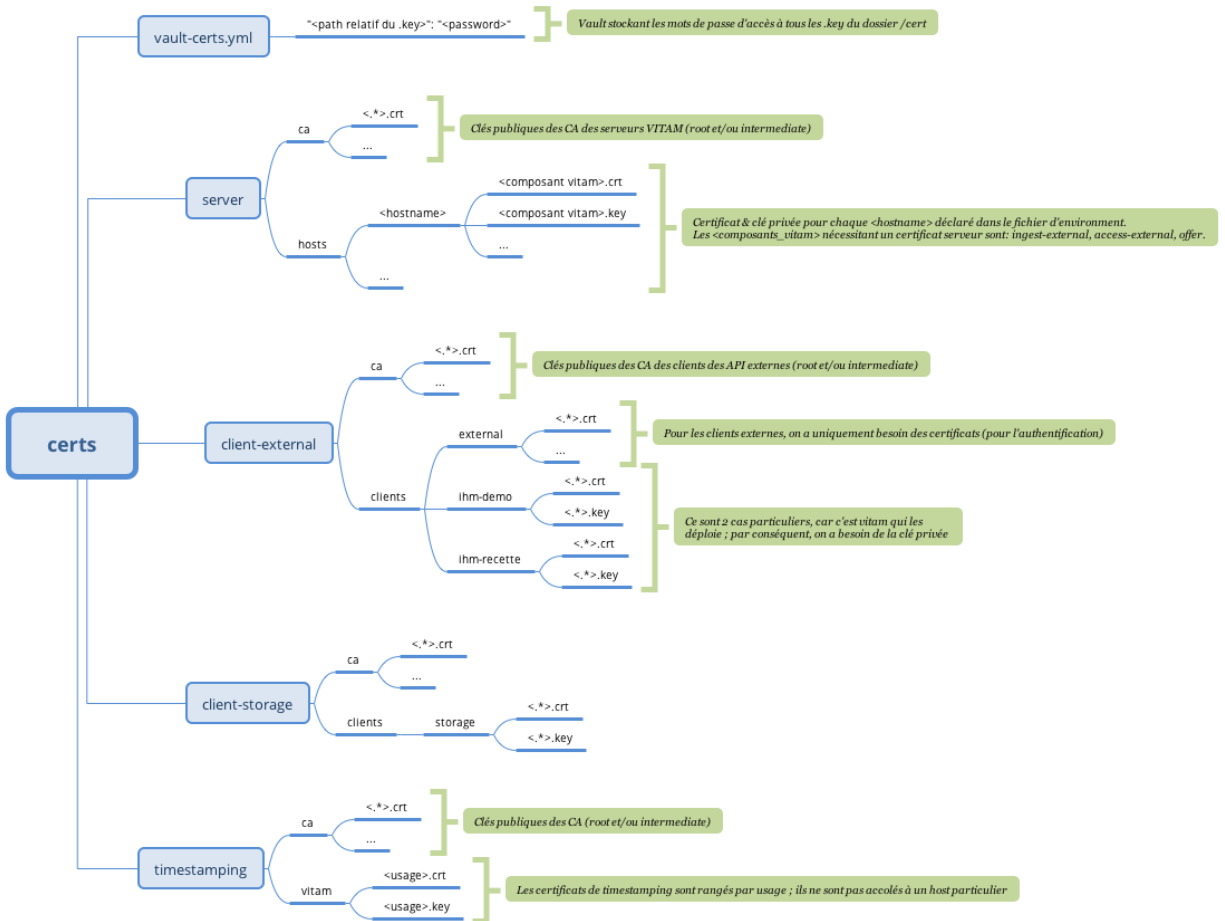


Fig. 7.3 – Vue détaillée de l'arborescence des certificats



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

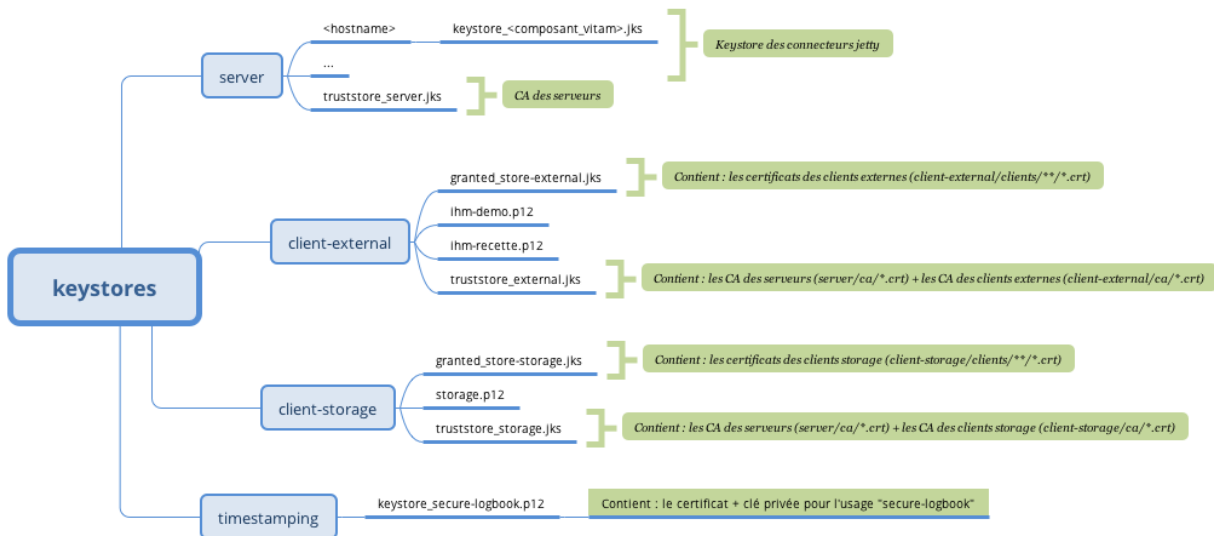


Fig. 7.4 – Vue détaillée de l'arborescence des keystores

## 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 bon 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 50)).

## 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 partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		offer	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
	Certif	Horo-datage	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		Storage (swift)	Offre de stockage	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		ingest	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		access	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		offer	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
		Times-tamp	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
IHM demo	CA	ihm-demo	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
	Certif	ihm-demo	PKI partenaire	<i>proc. par défaut</i>	<i>proc. par défaut</i>	<i>proc. par défaut</i>
SIA	CA	Appel API	PKI distante	proc. par défaut (PKI distante)	<i>proc. par défaut</i>	proc. par défaut (PKI distante) + recharger Certifs
	Certif	Appel API	PKI distante	Génération + copie répertoire + deploy (par la suite, appel API d’insertion)	Suppression Mongo	Suppression Mongo + API d’insertion
Personae	Certif	Appel API	PKI distante	API ajout	API suppression	API suppression + API ajout

**Remarques :**

- Lors d’un renouvellement de CA SIA, il faut s’assurer que les certificats qui y correspondaient 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 10).

#### 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.

---

**Annexes**

---



4.1	Vue détaillée de l'arborescence des certificats . . . . .	24
7.1	Vue d'ensemble de la gestion des certificats au déploiement . . . . .	48
7.2	Vue l'arborescence de la PKI Vitam . . . . .	49
7.3	Vue détaillée de l'arborescence des certificats . . . . .	50
7.4	Vue détaillée de l'arborescence des keystores . . . . .	51





2.1 Documents de référence VITAM . . . . . 3



## A

API, 3

## B

BDD, 3

## C

COTS, 3

## D

DAT, 3

DEX, 3

DIN, 3

DNSSEC, 4

DUA, 4

## I

IHM, 4

## J

JRE, 4

JVM, 4

## M

MitM, 4

## N

NoSQL, 4

## O

OAIS, 4

## P

PDMA, 4

PKI, 4

## R

REST, 4

RPM, 4

## S

SAE, 4

SEDA, 4

SIA, 4

## T

TNR, 4

## V

VITAM, 4