VITAM - Documentation d'installation

Version 0.15.1

VITAM

avr. 26, 2017

Table des matières

1 Introduction **3**
- 1.1 But de cette documentation . . . 3
- 1.2 Destinataires de ce document . . . 3

2 Rappels **5**
- 2.1 Information concernant les licences . . . 5
- 2.2 Documents de référence . . . 5
  - 2.2.1 Documents internes . . . 5
  - 2.2.2 Référentiels externes . . . 5
- 2.3 Glossaire . . . 6

3 Architecture de la solution logicielle VITAM **9**

4 Pré-requis **11**
- 4.1 Description . . . 11
  - 4.1.1 Base commune . . . 11
  - 4.1.2 Déploiement sur environnement CentOS . . . 11
  - 4.1.3 Déploiement sur environnement Debian . . . 12
- 4.2 Matériel . . . 12

5 Dépendances aux services d'infrastructures **13**
- 5.1 Ordonnanceurs techniques / batchs . . . 13
  - 5.1.1 Cas de la sauvegarde . . . 13
- 5.2 Socles d'exécution . . . 13
  - 5.2.1 OS . . . 13
  - 5.2.2 Middlewares . . . 13

6 Fiche type de déploiement VITAM **15**
- 6.1 Fiche-type VITAM . . . 15

7 Guidelines de déploiement **17**

8 Récupération de la version **19**

9 Procédures d'installation / mise à jour **21**
- 9.1 Pré-requis supplémentaire . . . 21
- 9.2 Procédures . . . 21
  - 9.2.1 Configuration de sécurité . . . 21
    - 9.2.1.1 Authentification du compte utilisateur utilisé pour la connexion SSH . . . 21

9.2.1.1.1 Par clé SSH avec passphrase . . . 21
9.2.1.1.2 Par login/mot de passe . . . 22
9.2.1.1.3 Par clé SSH sans passphrase . . . 22
9.2.1.2 Authentification des hôtes . . . 22
9.2.1.3 Elevation de privilèges . . . 22
9.2.1.3.1 Par sudo avec mot de passe . . . 22
9.2.1.3.2 Par su . . . 22
9.2.1.3.3 Par sudo sans mot de passe . . . 22
9.2.1.3.4 Déjà Root . . . 22
9.2.2 Explications relatives à la PKI . . . 23
9.2.2.1 Génération des autorités de certification . . . 23
9.2.2.1.1 Cas d'une PKI inexistante . . . 23
9.2.2.1.2 Cas d'une CA déjà existante . . . 27
9.2.2.2 Génération des certificats . . . 27
9.2.2.2.1 Cas de certificats inexistants . . . 27
9.2.2.2.2 Cas de certificats déjà créés par le client . . . 31
9.2.2.3 Génération des stores . . . 32
9.2.3 Procédure de première installation . . . 34
9.2.3.1 Configuration du déploiement . . . 34
9.2.3.1.1 Informations "plate-forme" . . . 34
9.2.3.2 Paramétrage de mongoclient (administration mongoclient) . . . 44
9.2.3.3 Première utilisation de mongoclient . . . 44
9.2.3.3.1 Paramétrage de l'antivirus (ingest-externe) . . . 45
9.2.3.3.2 Paramétrage des certificats (*-externe) . . . 45
9.2.3.4 Déploiement . . . 46
9.2.3.4.1 Fichier de mot de passe . . . 46
9.2.3.4.2 PKI . . . 46
9.2.3.4.3 Mise en place des repositories VITAM (optionnel) . . . 46
9.2.3.4.4 Déploiement . . . 47
9.2.3.4.5 Extra . . . 47
9.2.3.5 Import automatique d'objets dans Kibana . . . 48
9.2.4 Procédure de mise à niveau . . . 48

10 Validation de la procédure **49**
10.1 Sécurisation du fichier `vault_pass.txt` . . . 49
10.2 Validation manuelle . . . 49
10.3 Validation via Consul . . . 49
10.4 Validation via SoapUI . . . 50
10.5 Post-installation : administration fonctionnelle . . . 50
10.5.1 Cas du référentiel PRONOM . . . 50

11 Troubleshooting **51**

12 Présentation **53**

13 Elements extras de l'installation **55**

14 Contacts et support **57**
14.1 Contacts . . . 57

15 Annexes **59**

Index **65**

Prudence : Cette documentation est un travail en cours ; elle est susceptible de changer de manière conséquente.

CHAPITRE 1

Introduction

1.1 But de cette documentation

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.

1.2 Destinataires de ce document

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

CHAPITRE 2

Rappels

2.1 Information concernant les licences

Le logiciel *VITAM* est publié sous la license CeCILL 2.1 [1] ; la documentation associée (comprenant le présent document) est publiée sous license CC-BY-SA 3.0 [2].

2.2 Documents de référence

2.2.1 Documents internes

Tableau 2.1 – Documents de référence VITAM

Nom	Lien
DAT	(à renseigner)
DIN	(à renseigner)
DEX	(à renseigner)
Release notes	(à renseigner)

2.2.2 Référentiels externes

Référentiel Général d'Interopérabilité [RGI] V1.0 du 12 juin 2009 approuvé par arrêté du Premier ministre du 9 novembre 2009

Règles d'interopérabilité (format, protocoles, encodages, etc.) rentrant dans le champ d'application de l'ordonnance n°2005-1516 du 8 décembre 2005 relative aux échanges électroniques entre les usagers et les autorités administratives et entre les autorités administratives.

https://references.modernisation.gouv.fr/rgi-interoperabilite

Référentiel Général de Sécurité [RGS] V2.0 du 13 juin 2014 approuvé par arrêté du Premier ministre du 13 juin 2014

Le RGS précise les règles de sécurité s'imposant aux autorités administratives dans la sécurisation de leur SI et notamment sur les dispositifs de sécurité relatifs aux mécanismes cryptographiques et à l'utilisation de certificats électroniques et contremarques de temps. Le RGS propose également des

1. http://www.cecill.info/licences/Licence_CeCILL_V2.1-fr.html
2. https://creativecommons.org/licenses/by-sa/3.0/fr/legalcode

bonnes pratiques en matière de SSI. Le RGS découle de l'application de l'ordonnance n°2005-1516 du 8 décembre 2005 relative aux échanges électroniques entre les usagers et les autorités administratives et entre les autorités administratives.

https://references.modernisation.gouv.fr/rgs-securite

Norme OAIS (ISO 14721 :2012 – 1 septembre 2012) Systèmes de transfert desinformations et données spatiales – Système ouvert d'archivage d'information (SOAI) - Modèle de référence

Standard d'échange de données pour l'archivage (SEDA) Transfert, communication, élimination, restitution, modification – Version 1.0 – Septembre 2012

Cadre normatif pour les différents échanges d'informations entre les services d'archives publics et leurs partenaires : entités productrices des archives, entités gestionnaires, entités de contrôle des processus, et enfin entités qui utilisent ces archives. Il concerne également les échanges entre plusieurs services d'archives (services publics d'archives, prestataires d'archivage, archivage intermédiaire, archivage définitif).

http://www.archivesdefrance.culture.gouv.fr/seda/

2.3 Glossaire

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.

DIN Dossier d'Installation

DEX Dossier d'EXploitation

DAT Dossier d'Architecture Technique

IHM Interface Homme Machine

VITAM Valeurs Immatérielles Transférées aux Archives pour Mémoire

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

Deb Debian ; il s'agit du format de packets logiciels nativement utilisé par les distributions Debian GNU/Linux

API Application Programming Interface

BDD Base De Données

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*

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.

NoSQL Base de données non-basée sur un paradigme classique des bases relationnelles. Définition [3]

MitM L'attaque de l'homme du milieu (HDM) ou *man-in-the-middle attack* (MITM) est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis. Le canal le plus courant est une connexion à Internet de l'internaute lambda. L'attaquant doit d'abord être capable d'observer et d'intercepter les messages d'une victime à l'autre. L'attaque « homme du milieu » est particulièrement applicable dans la méthode d'échange de clés Diffie-Hellman, quand cet échange est utilisé sans authentification. Avec authentification, Diffie-Hellman est en revanche invulnérable aux écoutes du canal, et est d'ailleurs conçu pour cela. Explication [4]

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

3. https://fr.wikipedia.org/wiki/NoSQL
4. https://fr.wikipedia.org/wiki/Attaque_de_l'homme_du_milieu
5. https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

PKI Une infrastructure à clés publiques (ICP) ou infrastructure de gestion de clés (IGC) ou encore Public Key Infrastructure (PKI), est un ensemble de composants physiques (des ordinateurs, des équipements cryptographiques logiciels ou matériel type HSM ou encore des cartes à puces), de procédures humaines (vérifications, validation) et de logiciels (système et application) en vue de gérer le cycle de vie des certificats numériques ou certificats électroniques. Définition PKI [6]

SIA Système d'Informations Archivistique

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

TNR Tests de Non-Régression

6. https://fr.wikipedia.org/wiki/Infrastructure_%C3%A0_cl%C3%A9s_publiques

CHAPITRE 3

Architecture de la solution logicielle VITAM

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

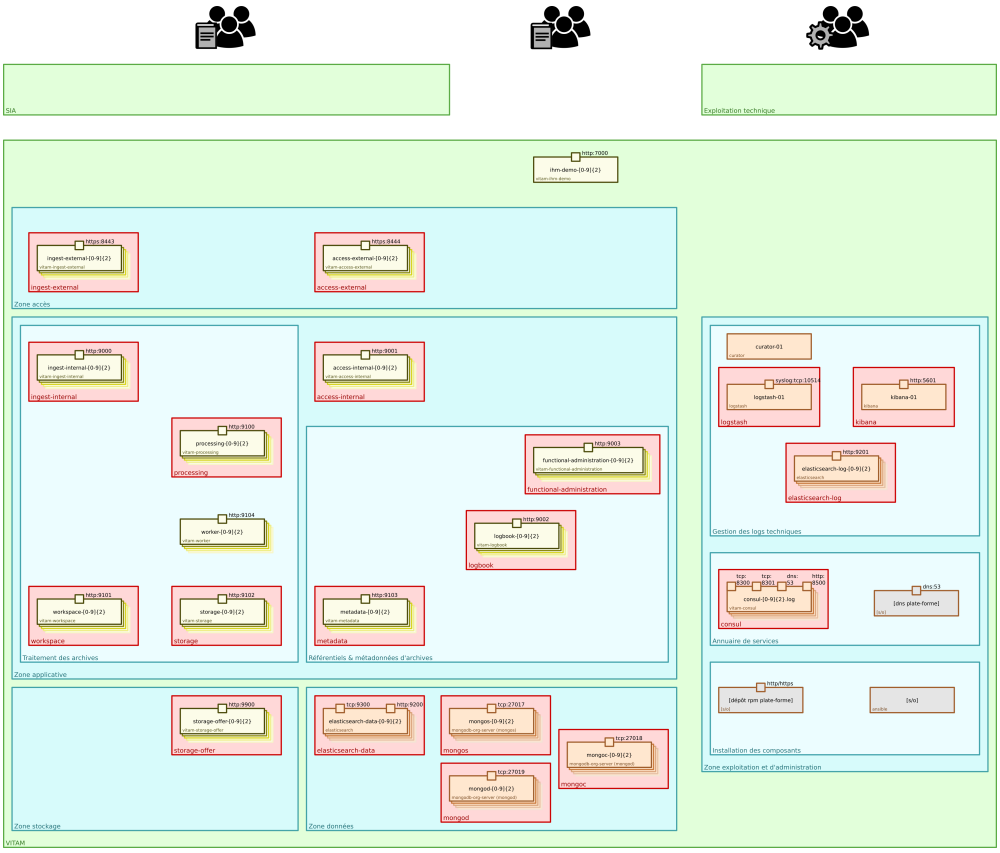

Fig. 3.1 – Vue d'ensemble d'un déploiement VITAM : zones, composants

Voir aussi :

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

CHAPITRE 4

Pré-requis

4.1 Description

Les pré-requis logiciels suivants sont nécessaires :

4.1.1 Base commune

- Disposer de la solution de déploiement basé 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 2.0.2 minimale et conseillée)
  - 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 cibles (fichier ~/.ssh/known_hosts correctement renseigné)

4.1.2 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 Vitam (vitam-product) ainsi que les paquets d'éditeurs tiers livrés avec Vitam (vitam-external)

4.1.3 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 Vitam (vitam-product) ainsi que les paquets d'éditeurs tiers livrés avec Vitam (vitam-external)

4.2 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>`

CHAPITRE 5

Dépendances aux services d'infrastructures

5.1 Ordonnanceurs techniques / batchs

Note : Curator permet d'effectuer des opérations périodiques de maintenance sur les index elasticsearch. Les jobs Curator sont initiés automatiquement au déploiement de VITAM et sont lancés via crontab.

Note : Des batchs d'exploitation seront disponibles dans les versions ultérieures de la solution VITAM (ex : validation périodique de la validité des certificats clients)

Job de sécurisation du logbook : lancé toutes les nuits peu après minuit sur une des machines (la dernière) hébergeant le composant vitam-logbook.

5.1.1 Cas de la sauvegarde

Se référer à la section dédiée du *DAT*.

5.2 Socles d'exécution

5.2.1 OS

Prudence : SELinux doit être configuré en mode `permissive` ou `disabled`

5.2.2 Middlewares

- Java : JRE 8 ; les versions suivantes ont été testées :
  - OpenJDK 1.8.0, dans la version présente dans les dépôts officiels au moment de la parution cette release de Vitam (Centos et Debian en 1.8.0_121)

CHAPITRE 6

Fiche type de déploiement VITAM

6.1 Fiche-type VITAM

Prudence : cette liste a pour but d'évoluer et s'étoffer au fur et à mesure des mises à jour des composants et du contenu des fichiers de déploiement de VITAM.

Tableau 6.1 – Tableau récapitulatif des informations à renseigner pour VITAM

Nom du composant	Descriptif	Valeur d'exemple	Valeur choisie	Si HA ?
IHM-demo machine	interface web	vitam-prod-app-1.internet.agri		
ingest-external machine	interface web	vitam-prod-app-1.internet.agri		
ingest-internal machine	interface web	vitam-prod-app-1.internet.agri		
access-external machine	interface web	vitam-prod-app-1.internet.agri		
access-internal machine	interface web	vitam-prod-app-1.internet.agri		
logbook machine	interface web	vitam-prod-app-1.internet.agri		
metadata machine	interface web	vitam-prod-app-1.internet.agri		
processing machine(s)	base de données	vitam-prod-app-1.internet.agri		
worker machine(s)	Traitement de fichiers	vitam-prod-wrk-1.internet.agri		
storage-engine machine(s)	xxxx	vitam-prod-app-1.internet.agri		
storage-offer-default machine(s)	implémentation de pilote de stockage	vitam-prod-app-1.internet.agri		
Consul servers	implémentation de Consul pour un DNS applicatif (nécessite 3 serveurs minimum ; règle (2*n+1))	vitam-prod-app-1.internet.agri, vitam-prod-app-2.internet.agri, vitam-prod-app-3.internet.agri		
elasticsearch data machine(s)	Cluster ElasticSearch de données VITAM (3 machines)	vitam-prod-ela-1.internet.agri,vitam-prod-ela-2.internet.agri,vitam-prod-ela-3.internet.agri		
elasticsearch log machine(s)	Cluster ElasticSearch de log VITAM (3 machines)	vitam-prod-log-1.internet.agri,vitam-prod-log-2.internet.agri,vitam-prod-log-3.internet.agri		
mongo-s machine(s)	Cluster MongoDB de routage de data VITAM (3 machines)	vitam-prod-ms-1.internet.agri,vitam-prod-ms-2.internet.agri,vitam-prod-ms-3.internet.agri		
mongo-c machine(s)	Cluster MongoDB de configuration des données VITAM (3 machines)	vitam-prod-mc-1.internet.agri,vitam-prod-mc-2.internet.agri,vitam-prod-mc-3.internet.agri		
mongo-d machine(s)	Cluster Mongo de données VITAM (3 machines)	vitam-prod-md-1.internet.agri,vitam-prod-md-2.internet.agri,vitam-prod-md-3.internet.agri		
log central machine(s)	Centralisation des logs	vitam-prod-log-1.internet.agri		

CHAPITRE 7

Guidelines de déploiement

Les principes de zoning associés à l'architecture du systèmes VITAM ont été présentés lors de la description des principes de déploiement ; cette section a pour but de compléter ces principes par des recommandations concernant la colocalisation des composants.

De manière générale, pour des raisons de sécurité, il est déconseillé de colocaliser des composants appartenant à des zones différentes. Il est par contre possible de colocaliser des composants appartenant à des sous-zones différentes dans la zone des services internes ; ainsi, les colocalisations des composants suivants sont relativement pertinentes :

- ingest-external, access-external et administration-external ;
- ingest-internal et access-internal ;
- elasticsearch-data et mongod ;
- mongos et mongoc ;
- logstash, elasticsearch-log, kibana (pour les déploiements de taille limitée) ; elasticsearch-log et consul (serveur) (pour des déploiements de taille moyenne)
- workspace et storage ;

Prudence : Il est recommandé de ne pas colocaliser les composants restants :

- storage-offer-default, étant dans une zone logique particulière ;
- worker, ayant une consommation de resources système potentiellement importante.

Note : Ces principes de colocation sont les préconisations initiales relatives à cette version du système VITAM ; ils seront revus suite aux campagnes de tests de performance en cours.

CHAPITRE 8

Récupération de la version

Se connecter sur l'URL support [7] et récupérer :

- le package de livraison
- la release notes
- les empreintes de contrôle

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

Sur le repository "VITAM", récupérer également depuis le tar.gz les packages de binaires (rpm pour une installation cible sous CentOS ; deb pour une installation cible sous Debian) et les faire prendre en compte par le repository.

7. https://support.programmevitam.fr/releases/

CHAPITRE 9

Procédures d'installation / mise à jour

9.1 Pré-requis supplémentaire

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.

9.2 Procédures

9.2.1 Configuration de sécurité

En fonction de la méthode d'authentification sur les serveurs et d'élevation 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

9.2.1.1 Authentification du compte utilisateur utilisé pour la connexion SSH

Pour le login du compte utilisateur, voir le paragraphe décrivant le fichier d'inventaire

9.2.1.1.1 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.

9.2.1.1.2 Par login/mot de passe

Dans le cas d'une authentificatioon 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

9.2.1.1.3 Par clé SSH sans passphrase

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

9.2.1.2 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.

9.2.1.3 Elevation de privilèges

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

9.2.1.3.1 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

9.2.1.3.2 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

9.2.1.3.3 Par sudo sans mot de passe

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

9.2.1.3.4 Déjà Root

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

9.2.2 Explications relatives à la PKI

Les commandes sont à passer dans le sous-répertoire `deployment` de la livraison.

Prudence : par la suite, le terme <environnement> correspond à l'extension du nom de fichier d'inventaire.

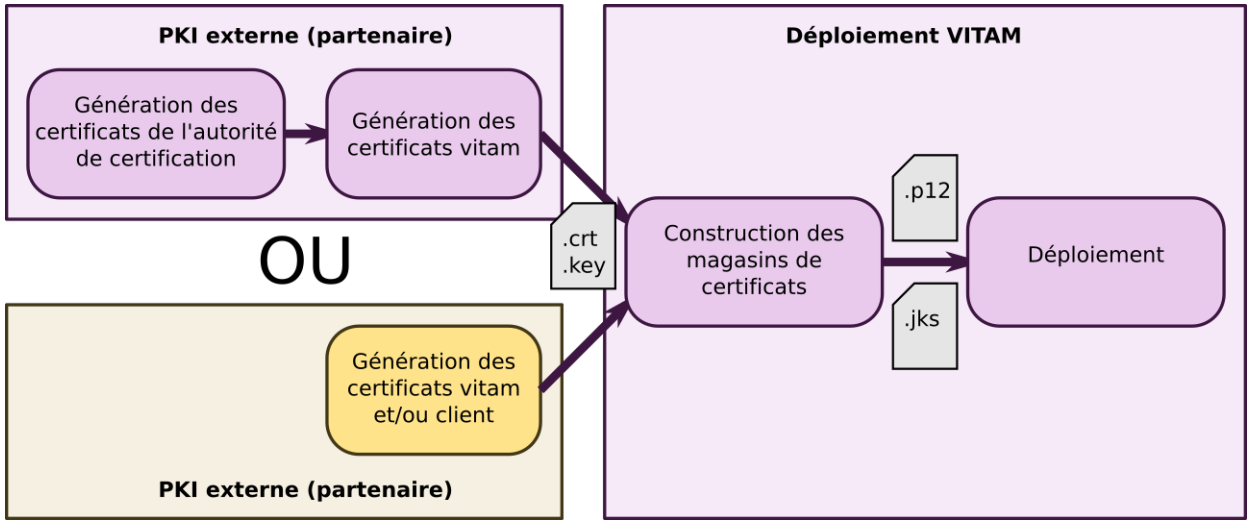

Fig. 9.1 – Vue d'ensemble de la gestion des certificats au déploiement

9.2.2.1 Génération des autorités de certification

9.2.2.1.1 Cas d'une PKI inexistante

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 clients, serveurs, et timestamping.

Voici ci-dessous un exemple de rendu du script :

```
[INFO] [generate_ca.sh] Lancement de la procédure de création des CA
[INFO] [generate_ca.sh] ================================================
[INFO] [generate_ca.sh] Répertoire /home/nico/git/vitam/deployment/pki/ca absent ;␣
↪création...
[INFO] [generate_ca.sh] Création du répertoire de travail temporaire tempcerts sous /
↪home/nico/git/vitam/deployment/pki/tempcerts...
[INFO] [generate_ca.sh] Création de CA root pour server...
[INFO] [generate_ca.sh] Create CA request...
Generating a 2048 bit RSA private key
.........................+++
.....................+++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/server/ca-root.key'
-----
[INFO] [generate_ca.sh] Create CA certificate...
Using configuration from /home/nico/git/vitam/deployment/pki/config/server/ca-config
```

```
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_server'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:14 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[INFO] [generate_ca.sh] Création de la CA intermediate pour server...
[INFO] [generate_ca.sh] Generate intermediate request...
Generating a 4096 bit RSA private key
..............................++
.....................................................................................
 ↪............++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/server/ca-
 ↪intermediate.key'
-----
[INFO] [generate_ca.sh] Sign...
Using configuration from /home/nico/git/vitam/deployment/pki/config/server/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_server'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:14 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[INFO] [generate_ca.sh] ----------------------------------------------
[INFO] [generate_ca.sh] Création de CA root pour client-external...
[INFO] [generate_ca.sh] Create CA request...
Generating a 2048 bit RSA private key
.....................................................+++
..................................+++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/client-external/ca-
 ↪root.key'
-----
[INFO] [generate_ca.sh] Create CA certificate...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-external/
 ↪ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_client-external'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:14 2027 GMT (3650 days)

Write out database with 1 new entries
```

```
Data Base Updated
[INFO] [generate_ca.sh] Création de la CA intermediate pour client-external...
[INFO] [generate_ca.sh] Generate intermediate request...
Generating a 4096 bit RSA private key
.....................++
....++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/client-external/ca-
↪intermediate.key'
-----
[INFO] [generate_ca.sh] Sign...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-external/
↪ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_client-external'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:14 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[INFO] [generate_ca.sh] ----------------------------------------------
[INFO] [generate_ca.sh] Création de CA root pour client-storage...
[INFO] [generate_ca.sh] Create CA request...
Generating a 2048 bit RSA private key
..............+++
................................+++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/client-storage/ca-
↪root.key'
-----
[INFO] [generate_ca.sh] Create CA certificate...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-storage/ca-
↪config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_client-storage'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:14 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[INFO] [generate_ca.sh] Création de la CA intermediate pour client-storage...
[INFO] [generate_ca.sh] Generate intermediate request...
Generating a 4096 bit RSA private key
..........................................................................................
↪.....................................................................++
..........................................................................................
↪.....................................................................................
↪.....................................................................................
↪.............................++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/client-storage/ca-
↪intermediate.key'
```

```
-----
[INFO] [generate_ca.sh] Sign...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-storage/ca-
↪config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_client-storage'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:16 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[INFO] [generate_ca.sh] ------------------------------------------------
[INFO] [generate_ca.sh] Création de CA root pour timestamping...
[INFO] [generate_ca.sh] Create CA request...
Generating a 2048 bit RSA private key
.........................+++
.........................................+++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/timestamping/ca-
↪root.key'
-----
[INFO] [generate_ca.sh] Create CA certificate...
Using configuration from /home/nico/git/vitam/deployment/pki/config/timestamping/ca-
↪config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_timestamping'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:16 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[INFO] [generate_ca.sh] Création de la CA intermediate pour timestamping...
[INFO] [generate_ca.sh] Generate intermediate request...
Generating a 4096 bit RSA private key
.........................++
.........++
writing new private key to '/home/nico/git/vitam/deployment/pki/ca/timestamping/ca-
↪intermediate.key'
-----
[INFO] [generate_ca.sh] Sign...
Using configuration from /home/nico/git/vitam/deployment/pki/config/timestamping/ca-
↪config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CA_timestamping'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
```

```
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 26 16:29:16 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[INFO] [generate_ca.sh] ----------------------------------------------
[INFO] [generate_ca.sh] ==============================================
[INFO] [generate_ca.sh] Fin de la procédure de création des CA
```

Note : 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.

9.2.2.1.2 Cas d'une CA déjà existante

Pas de support pour le moment en cas de CA déjà existante uniquement. Il est nécessaire de générer et déposer manuellement les certificats (voir étape ci-dessous)

9.2.2.2 Génération des certificats

9.2.2.2.1 Cas de certificats inexistants

Avertissement : cette étape n'est à effectuer que pour les clients ne possédant pas de certificats.

Editer complètement le fichier `environments-rpm/<inventaire>` 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 <environnement>
```

Ci-dessous un exemple de sortie du script :

```
[INFO] [generate_certs.sh] Suppression de l'ancien vault
[INFO] [generate_certs.sh] Recopie des clés publiques des CA
[INFO] [generate_certs.sh] Copie de la CA (root + intermediate) de client-external
[INFO] [generate_certs.sh] Copie de la CA (root + intermediate) de client-storage
[INFO] [generate_certs.sh] Copie de la CA (root + intermediate) de server
[INFO] [generate_certs.sh] Copie de la CA (root + intermediate) de timestamping
[INFO] [generate_certs.sh] Génération des certificats serveurs
[INFO] [generate_certs.sh] Création du certificat server pour ingest-external hébergé␣
↪sur localhost...
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
.....................................................................................
↪.....................................................++
.....................................++
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
↪server/hosts/localhost/ingest-external.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec CA server...
```

```
Using configuration from /home/nico/git/vitam/deployment/pki/config/server/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'ingest-external.service.consul'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 29 09:37:00 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Encryption successful
[INFO] [generate_certs.sh] Création du certificat server pour access-external hébergé␣
↪sur localhost...
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
..........++
..................++
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
↪server/hosts/localhost/access-external.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec CA server...
Using configuration from /home/nico/git/vitam/deployment/pki/config/server/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'access-external.service.consul'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 29 09:37:01 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Decryption successful
Encryption successful
[INFO] [generate_certs.sh] Création du certificat server pour storage-offer-default␣
↪hébergé sur localhost...
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
.....................++
........++
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
↪server/hosts/localhost/storage-offer-default.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec CA server...
Using configuration from /home/nico/git/vitam/deployment/pki/config/server/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'storage-offer-default.service.consul'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
```

```
Certificate is to be certified until Feb 29 09:37:02 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Decryption successful
Encryption successful
[INFO] [generate_certs.sh] Génération des certificats timestamping
[INFO] [generate_certs.sh] Création du certificat timestamping pour logbook
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
.....................................................................................
 ↪...................++
...........++
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
 ↪timestamping/vitam/logbook.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec CA timestamping...
Using configuration from /home/nico/git/vitam/deployment/pki/config/timestamping/ca-
 ↪config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'logbook.service.consul'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 29 09:37:04 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Decryption successful
Encryption successful
[INFO] [generate_certs.sh] Génération des certificats clients
[INFO] [generate_certs.sh] Création du certificat client pour ihm-demo
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
....++
........++
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
 ↪client-external/clients/ihm-demo/ihm-demo.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec client-external...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-external/
 ↪ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'ihm-demo'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 29 09:37:04 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Decryption successful
```

```
Encryption successful
[INFO] [generate_certs.sh] Création du certificat client pour ihm-recette
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
.....................................................................++
.....................................................................................
 ↪.....................................................................++
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
 ↪client-external/clients/ihm-recette/ihm-recette.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec client-external...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-external/
 ↪ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'ihm-recette'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 29 09:37:06 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Decryption successful
Encryption successful
[INFO] [generate_certs.sh] Création du certificat client pour reverse
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
..............................++
.....................................................................................
 ↪...........++
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
 ↪client-external/clients/reverse/reverse.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec client-external...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-external/
 ↪ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'reverse'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 29 09:37:07 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Decryption successful
Encryption successful
[INFO] [generate_certs.sh] Création du certificat client pour storage-engine
[INFO] [generate_certs.sh] Generation de la clé...
Generating a 4096 bit RSA private key
...........++
.....................................................................................
 ↪....++
```

```
writing new private key to '/home/nico/git/vitam/deployment/environments-rpm/certs/
↪client-storage/clients/storage-engine/storage-engine.key'
-----
[INFO] [generate_certs.sh] Generation du certificat signé avec client-storage...
Using configuration from /home/nico/git/vitam/deployment/pki/config/client-storage/ca-
↪config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'storage-engine'
organizationName      :ASN.1 12:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :ASN.1 12:'idf'
localityName          :ASN.1 12:'paris'
Certificate is to be certified until Feb 29 09:37:08 2020 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
Decryption successful
Encryption successful
[INFO] [generate_certs.sh] Fin de script
```

Ce script génère sous `environnements-rpm/certs` les certificats (format crt & key) nécessaires pour un bon fonctionnement dans VITAM.

Prudence : Les certificats générés à l'issue ont une durée de validité de (à vérifier).

9.2.2.2.2 Cas de certificats déjà créés par le client

Si le client possède déjà une *PKI*, ou ne compte pas utiliser la *PKI* fournie par VITAM, il convient de positionner les certificats et CA sous `environnements-rpm/certs/....` en respectant la structure indiquée ci-dessous

- **cert**
  - **client-external**
    - ca : CA(s) des certificats clients external
    - **clients**
      - external : Certificats des SIAs
      - ihm-demo : Certificat de ihm-demo
      - ihm-recette : Certificat de ihm-recette
      - reverse : Certificat du reverse
  - **client-storage**
    - ca : CA(s) des certificats clients storage
    - **clients**
      - storage-engine : Certificat de storage-engine
  - **server**
    - ca : CA(s) des certificats côté serveurs
    - **hosts**
      - [nom_serveur] : certificats des composants installés sur le serveur donné, [nom_serveur] doit être identique à ce qui est référencé dans le ficheir d'inventaire

- **timestamping**
  - ca : CA des certificats de timestamping
  - vitam : Certificats de timestamping

Il est aussi nécessaire de renseigner le vault contenant les passphrases des clés des certificats : `environmements/certs/vault-certs.yml`

9.2.2.3 Génération des stores

Prudence : Avant de lancer le script de génération des stores, il est nécessaire de modifier le vault contenant les mots de passe des stores

Lancer le script :

```
./generate_stores.sh <environnement>
```

Ci-dessous un exemple de sortie du script :

```
[INFO] [generate_stores.sh] ------------------------------------------
[INFO] [generate_stores.sh] Creation du keystore de access-external pour le serveur␣
↪localhost
[INFO] [generate_stores.sh] Génération du p12
[INFO] [generate_stores.sh] Génération du jks
Entry for alias access-external successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or␣
↪cancelled
[INFO] [generate_stores.sh] Suppression du p12
[INFO] [generate_stores.sh] ------------------------------------------
[INFO] [generate_stores.sh] Creation du keystore de ingest-external pour le serveur␣
↪localhost
[INFO] [generate_stores.sh] Génération du p12
[INFO] [generate_stores.sh] Génération du jks
Entry for alias ingest-external successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or␣
↪cancelled
[INFO] [generate_stores.sh] Suppression du p12
[INFO] [generate_stores.sh] ------------------------------------------
[INFO] [generate_stores.sh] Creation du keystore de storage-offer-default pour le␣
↪serveur localhost
[INFO] [generate_stores.sh] Génération du p12
[INFO] [generate_stores.sh] Génération du jks
Entry for alias storage-offer-default successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or␣
↪cancelled
[INFO] [generate_stores.sh] Suppression du p12
[INFO] [generate_stores.sh] ------------------------------------------
[INFO] [generate_stores.sh] Creation du keystore timestamp de logbook
[INFO] [generate_stores.sh] ------------------------------------------
[INFO] [generate_stores.sh] Creation du keystore client de ihm-demo
[INFO] [generate_stores.sh] Génération du p12
[INFO] [generate_stores.sh] Ajout du certificat public de ihm-demo dans le␣
↪grantedstore external
Certificate was added to keystore
[INFO] [generate_stores.sh] ------------------------------------------
[INFO] [generate_stores.sh] Creation du keystore client de ihm-recette
```

```
[INFO] [generate_stores.sh] Génération du p12
[INFO] [generate_stores.sh] Ajout du certificat public de ihm-recette dans le␣
↪grantedstore external
Certificate was added to keystore
[INFO] [generate_stores.sh] ----------------------------------------
[INFO] [generate_stores.sh] Creation du keystore client de reverse
[INFO] [generate_stores.sh] Génération du p12
[INFO] [generate_stores.sh] Ajout du certificat public de reverse dans le␣
↪grantedstore external
Certificate was added to keystore
[INFO] [generate_stores.sh] ----------------------------------------
[INFO] [generate_stores.sh] Ajout des certificat public du répertoire external dans␣
↪le grantedstore external
[INFO] [generate_stores.sh] ----------------------------------------
[INFO] [generate_stores.sh] Génération du truststore client-external
[INFO] [generate_stores.sh] Ajout des certificats client dans le truststore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/client-
↪external/ca-intermediate.crt dans le truststore external
Certificate was added to keystore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/client-
↪external/ca-root.crt dans le truststore external
Certificate was added to keystore
[INFO] [generate_stores.sh] Ajout des certificats serveur dans le truststore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/server/ca-
↪intermediate.crt dans le truststore external
Certificate was added to keystore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/server/ca-
↪root.crt dans le truststore external
Certificate was added to keystore
[INFO] [generate_stores.sh] ----------------------------------------
[INFO] [generate_stores.sh] Creation du keystore client de storage-engine
[INFO] [generate_stores.sh] Génération du p12
[INFO] [generate_stores.sh] Ajout du certificat public de storage-engine dans le␣
↪grantedstore storage
Certificate was added to keystore
[INFO] [generate_stores.sh] ----------------------------------------
[INFO] [generate_stores.sh] Ajout des certificat public du répertoire external dans␣
↪le grantedstore storage
[INFO] [generate_stores.sh] ----------------------------------------
[INFO] [generate_stores.sh] Génération du truststore client-storage
[INFO] [generate_stores.sh] Ajout des certificats client dans le truststore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/client-
↪storage/ca-intermediate.crt dans le truststore storage
Certificate was added to keystore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/client-
↪storage/ca-root.crt dans le truststore storage
Certificate was added to keystore
[INFO] [generate_stores.sh] Ajout des certificats serveur dans le truststore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/server/ca-
↪intermediate.crt dans le truststore storage
Certificate was added to keystore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/server/ca-
↪root.crt dans le truststore storage
Certificate was added to keystore
[INFO] [generate_stores.sh] ----------------------------------------
[INFO] [generate_stores.sh] Génération du truststore server
[INFO] [generate_stores.sh] Ajout des certificats client dans le truststore
[INFO] [generate_stores.sh] Ajout des certificats serveur dans le truststore
```

```
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/server/ca-
↪intermediate.crt dans le truststore server
Certificate was added to keystore
[INFO] [generate_stores.sh] Ajout de /home/nico/git/vitam/deployment/pki/ca/server/ca-
↪root.crt dans le truststore server
Certificate was added to keystore
[INFO] [generate_stores.sh] ------------------------------------------
[INFO] [generate_stores.sh] Fin de la génération des stores
```

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

9.2.3 Procédure de première installation

Les fichiers de déploiement sont disponibles dans la version VITAM livrée dans le sous-répertoire `deployment` . Ils consistent en 2 parties :

- le playbook ansible, présent dans le sous-répertoire `ansible-vitam-rpm`, qui est indépendant de l'environnement à déployer
- les fichiers d'inventaire (1 par environnement à déployer) ; des fichiers d'exemple sont disponibles dans le sous-répertoire `environments-rpm`

9.2.3.1 Configuration du déploiement

9.2.3.1.1 Informations "plate-forme"

Pour configurer le déploiement, il est nécessaire de créer dans le répertoire `environments-rpm` un nouveau fichier d'inventaire à nommer `hosts.<environnement>` (où <environnement> sera utilisé par la suite) comportant les informations suivantes :

```
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-mongo-express
10
11
12 ########### Tests environments specifics ###########
13
14 # EXTRA : Front reverse-proxy (test environments ONLY) ; add machine name after
15 [reverse]
16 # optional : after machine, if this machine is different from VITAM machines, you can␣
   ↪specify another become user
17 # Example
18 # vitam-centos-01.vitam ansible_ssh_user=centos
19
20 ########### Extra VITAM applications ###########
```

```
21
22 [library]
23 # TODO: Put here servers where this service will be deployed : library
24
25 [hosts-mongo-express]
26 # TODO: Put here servers where this service will be deployed : mongo-express
27
28 [elasticsearch:children] # EXTRA : elasticsearch
29 hosts-elasticsearch-data
30 hosts-elasticsearch-log
31
32 ########### VITAM services ###########
33
34 # Group definition ; DO NOT MODIFY
35 [vitam:children]
36 zone-external
37 zone-access
38 zone-applicative
39 zone-storage
40 zone-data
41 zone-admin
42
43
44 ##### Zone externe
45
46
47 [zone-external:children]
48 hosts-ihm-demo
49
50 [hosts-ihm-demo]
51 # TODO: Put here servers where this service will be deployed : ihm-demo
52
53
54 ##### Zone access
55
56 # Group definition ; DO NOT MODIFY
57 [zone-access:children]
58 hosts-ingest-external
59 hosts-access-external
60
61 [hosts-ingest-external]
62 # TODO: Put here servers where this service will be deployed : ingest-external
63
64
65 [hosts-access-external]
66 # TODO: Put here servers where this service will be deployed : access-external
67
68
69 ##### Zone applicative
70
71 # Group definition ; DO NOT MODIFY
72 [zone-applicative:children]
73 hosts-ingest-internal
74 hosts-processing
75 hosts-worker
76 hosts-access-internal
77 hosts-metadata
78 hosts-functional-administration
```

```
hosts-logbook
hosts-workspace
hosts-storage-engine

[hosts-logbook]
# TODO: Put here servers where this service will be deployed : logbook


[hosts-workspace]
# TODO: Put here servers where this service will be deployed : workspace


[hosts-ingest-internal]
# TODO: Put here servers where this service will be deployed : ingest-internal


[hosts-access-internal]
# TODO: Put here servers where this service will be deployed : access-internal


[hosts-metadata]
# TODO: Put here servers where this service will be deployed : metadata


[hosts-functional-administration]
# TODO: Put here servers where this service will be deployed : functional-
↪administration


[hosts-processing]
# TODO: Put here servers where this service will be deployed : processing


[hosts-storage-engine]
# TODO: Put here servers where this service will be deployed : storage-engine


[hosts-worker]
# TODO: Put here servers where this service will be deployed : worker
# Optional parameter after each host : vitam_worker_capacity=<integer> ; please refer␣
↪to your infrastructure for defining this number ; default is 1


##### Zone storage

[zone-storage:children] # DO NOT MODIFY
hosts-storage-offer-default


[hosts-storage-offer-default]
# TODO: Put here servers where this service will be deployed : storage-offer-default
# LIMIT : only 1 offer per machine and 1 machine per offer
# Additional params for openstack-swift
# hostname-offre-1.vitam vitam_keystone_auth_url=http://hostname-rados-gw:port/auth/1.
↪0 vitam_swift_subuser=subuser vitam_swift_uid=tenant$user vitam_provider_
↪offer=openstack-swift
# for filesystem
# hostname-offre-2.vitam vitam_provider_offer=filesystem
```

```
133
134
135 ##### Zone data
136
137 # Group definition ; DO NOT MODIFY
138 [zone-data:children]
139 hosts-elasticsearch-data
140 mongo_common
141
142
143 [hosts-elasticsearch-data]
144 # TODO: Put here servers where this service will be deployed : elasticsearch-data␣
    ↪cluster
145
146
147 # Group definition ; DO NOT MODIFY
148 [mongo_common:children]
149 mongos
150 mongoc
151 mongod
152
153 [mongos]
154 # TODO: Put here servers where this service will be deployed : mongos cluster ; add␣
    ↪after name shard_id=0
155 # Example : vitam-iaas-mongos-01.int shard_id=0
156
157 [mongoc]
158 # TODO: Put here servers where this service will be deployed : mongoc cluster
159
160
161 [mongod] # mongod declaration ; add machines name after ; add after shard_id=0 & rs_
    ↪member_id=<increasing number, starting from 0, for each line>
162 # TODO: Put here servers where this service will be deployed : mongod cluster ; add␣
    ↪after name shard_id=0
163 # Example : vitam-iaas-db-01.int rs_member_id=0 shard_id=0
164 # Example : vitam-iaas-db-02.int rs_member_id=1 shard_id=0
165 # Example : vitam-iaas-db-03.int rs_member_id=2 shard_id=0
166
167 ###### Zone admin
168
169 # Group definition ; DO NOT MODIFY
170 [zone-admin:children]
171 hosts-consul-server
172 hosts-log-server
173 hosts-elasticsearch-log
174 hosts-mongoclient
175
176 [hosts-consul-server]
177 # TODO: Put here servers where this service will be deployed : consul
178
179
180 [hosts-log-server]
181 # TODO: Put here servers where this service will be deployed : log-server (kibana/
    ↪logstash)
182
183
184 [hosts-elasticsearch-log]
185 # TODO: Put here servers where this service will be deployed : elasticsearch-log␣
    ↪cluster
```

```
186
187 [hosts-mongoclient]
188 # TODO: Put here servers where this service will be deployed : mongos cluster ; add␣
    ↪after name shard_id=0
189 # Example : vitam-iaas-mongos-01.int shard_id=0
190
191 ########### Global vars ###########
192
193 [hosts:vars]
194 # Declare user for ansible on target machines
195 ansible_ssh_user=
196
197 # Can target user become as root ? ; true is required by VITAM (usage of a sudoer is␣
    ↪mandatory)
198 ansible_become=true
199
200 # Environment (defines consul environment name ; in extra on homepage)
201 environnement=
202
203 # EXTRA : FQDN of the front reverse-proxy ; used when VITAM is behind a reverse proxy␣
    ↪(provides configuration for reverse proxy && displayed in header page)
204 vitam_reverse_external_dns=
205
206 # Version that has to be deployed (defined in the release note)
207 # Example: package_version=0.9.0-RC1*
208 package_version=
209
210 # Configuration for Curator
211 #       Days before deletion on log management cluster; 365 for production␣
    ↪environment
212 days_to_delete=
213
214 #       Days before closing "old" indexes on log management cluster; 30 for␣
    ↪production environment
215 days_to_close=
216
217 #       Days before deletion for topbeat index only on log management cluster; 365␣
    ↪for production environment
218 days_to_delete_topbeat=
219
220 # Related to Consul ; apply in a table your DNS server(s)
221 # Example : dns_servers=["8.8.8.8","8.8.4.4"]
222 dns_servers=
223
224 #       LOG level defined in logback files ; can be a value in "ERROR","WARN","INFO",
    ↪"DEBUG","TRACE". Recommended value is "WARN"
225 log_level=
226 # FOr SoapUI files tests
227 web_dir_soapui_tests=http://vitam-prod-ldap-1.internet.agri:8083/webdav
228
229 # For reverse proxy use
230 reverse_proxy_port=80
231
232 # For metrics
233 # curator job : days before closing
234 days_to_close_metrics=7
235 # curator job : days before deleting
236 days_to_delete_metrics=30
```

```
# Installation ClamAV ? true/false
installation_clamav=true

# cas de l'appel au webDAV pour récupérer les jeux de tests
http_proxy_environnement=

vitam_tenant_ids=[0,1,2]

# ces paramètres peuvent être soit globaux, ici, soit ajoutés après chaque partition hébergeant une offre de stockage

# useless now as it is declared in [hosts-storage-offer-default]
# vitam_provider_offer=openstack-swift
# URL d'authentification si openstack-swift
# vitam_keystone_auth_url=http://xxxxx.xxxx.xxx:8080/auth/1.0
# subUser pour swift
# vitam_swift_subuser=
# Nom du tenant associé (concaténation tenant$user ; le mot de passe est renseigné dans vault.yml sous la directive vitam_keystone_passwd )
# vitam_swift_uid=

# Pour les Tests de Non-Regression (en git LFS), si URL et branche définies, récupération des jeux de tests
vitam_swift_uid=

# Git LFS for TNR
vitam_tests_gitrepo_protocol=https
vitam_tests_gitrepo_baseurl=dev.programmevitam.fr
vitam_tests_gitrepo_url={{vitam_tests_gitrepo_protocol}}://{{vitam_tests_gitrepo_baseurl}}/gitlab/vitam/vitam-itests.git
vitam_tests_branch=master
```

Pour chaque type de "host", indiquer le(s) serveur(s) défini(s) pour chaque fonction. Une colocalisation de composants est possible.

Avertissement : en cas de colocalisation, bien prendre en compte la taille JVM de chaque composant (VITAM : Xmx512m) pour éviter de swapper.

Note : pour les "hosts-worker", il est possible d'ajouter, à la suite de chaque "host", 2 paramètres optionnels : capacity et workerFamily. Se référer au *DEX* pour plus de précisions.

Ensuite, dans la section `hosts:vars`, renseigner les valeurs comme décrit :

Tableau 9.1 – Définition des variables

Clé	Description	Valeur d'exemple
ansible_ssh_user	Utilisateurs ansible sur les machines sur lesquelles VITAM sera déployé	
ansible_become	Propriété interne à ansible pour passer root	
local_user	En cas de déploiement en local	
environnement	Suffixe	
vitam_reverse_domain	Cas de la gestion d'un reverse proxy	
consul_domain	nom de domaine consul	
vitam_ihm_demo_external_dns	Déprécié ; ne pas utiliser	
package_version	Version à installer	
days_to_delete	Période de grâce des données sous Elastricsearch avant destruction (valeur en jours)	
days_to_close	Période de grâce des données sous Elastricsearch avant fermeture des index (valeur en jours)	
days_to_delete_topbeat	Période de grâce des données sous Elastricsearch - index Topbeat - avant destruction (valeur en jours)	
days_to_delete_local	Période de grâce des log VITAM - logback (valeur en jours)	
dns_server	Serveur DNS que Consul peut appeler s'il n'arrive pas à faire de résolution	172.16.1.21
log_level	Niveau de log de logback	WARN
web_dir_soapui_tests	URL pour récupérer data.json et les tests pour SoapUI	http://vitam-prod-ldap-1.internet.agri:8083/webdav
reverse_proxy_port	port du reverse proxy pour configuration du vhost	8080
days_to_close_metrics	Période de grâce avant fermeture des index des métriques JVM	7
days_to_delete_metrics	Période de grâce avant destruction des index fermés des métriques JVM	30
installation_clamav	Choix d'installation de ClamAV (true/false)	true
http_proxy_environnement	Cas particulier de la récupération des jeux de tests ; URL de squid	
mongoclientPort	Port par lequel mongoclient est acessible	27016
mongoclientDbName	Nom de la Base de donnée stockant la configuration mongoclient	mongoclient
vitam_tenant_ids	Liste des tenants de plateforme	[0,1,2] ; [0] par défaut
vitam_tests_gitrepo_protocol	Protocole d'attaque du git lfs des TNR	
vitam_tests_gitrepo_baseurl	domaine du git lfs des TNR	
vitam_tests_gitrepo_url	Création de l'URL à partir des lignes précédentes	
vitam_tests_branch	Branche à récupérer sur le git lfs	master

A titre informatif, le positionnement des variables ainsi que des dérivations des déclarations de variables sont effectuées sous `environments-rpm /group_vars/all/all`, comme suit :

```
---

vitam_folder_root: /vitam
docker_registry_httponly: yes
vitam_docker_tag: latest
port_http_timeout: 86400

syslog_facility: local0

# Composants colocalisés

vitam_tenant_ids: [0]

vitam_accessinternal_host: "access-internal.service.{{consul_domain}}"
vitam_accessinternal_port: 8101
vitam_accessinternal_baseurl: "http://{{vitam_accessinternal_host}}:{{vitam_
↪accessinternal_port}}"
vitam_accessinternal_baseuri: "/access-internal"

vitam_accessexternal_host: "access-external.service.{{consul_domain}}"
vitam_accessexternal_port: 8102
vitam_accessexternal_port_https: 8444
vitam_accessexternal_baseurl: "http://{{vitam_accessexternal_host}}:{{vitam_
↪accessexternal_port}}"
vitam_accessexternal_baseuri: "/access-external"

vitam_ingestinternal_host: "ingest-internal.service.{{consul_domain}}"
vitam_ingestinternal_port: 8100
vitam_ingestinternal_baseurl: "http://{{vitam_ingestinternal_host}}:{{vitam_
↪ingestinternal_port}}"
vitam_ingestinternal_baseuri: "/ingest-internal"

vitam_ingestexternal_host: "ingest-external.service.{{consul_domain}}"
vitam_ingestexternal_port: 8001
vitam_ingestexternal_port_https: 8443
vitam_ingestexternal_baseurl: "http://{{vitam_ingestexternal_host}}:{{vitam_
↪ingestexternal_port}}"
vitam_ingestexternal_baseuri: "/ingest-external"

vitam_metadata_host: "metadata.service.{{consul_domain}}"
vitam_metadata_port: 8200
vitam_metadata_baseurl: "http://{{vitam_metadata_host}}:{{vitam_metadata_port}}"
vitam_metadata_baseuri: "/metadata"

vitam_ihm_demo_host: "{{groups['hosts-ihm-demo'][0]}}"
vitam_ihm_demo_port: 8002
vitam_ihm_demo_baseurl: /ihm-demo
vitam_ihm_demo_static_content: "{{vitam_folder_root}}/app/ihm-demo"
vitam_ihm_demo_baseuri: "/ihm-demo"

vitam_ihm_recette_host: "{{groups['hosts-ihm-recette'][0]}}"
vitam_ihm_recette_port: 8204
vitam_ihm_recette_baseurl: /ihm-recette
vitam_ihm_recette_static_content: "{{vitam_folder_root}}/app/ihm-recette"
vitam_ihm_recette_baseuri: "/ihm-recette"

# Internal components communication configuration
vitam_logbook_host: "logbook.service.{{consul_domain}}"
```

```
vitam_logbook_port: 9002
vitam_logbook_baseurl: "http://{{vitam_logbook_host}}:{{vitam_logbook_port}}"
vitam_logbook_baseuri: "/logbook"

vitam_workspace_host: "workspace.service.{{consul_domain}}"
vitam_workspace_port: 8201
vitam_workspace_baseurl: "http://{{vitam_workspace_host}}:{{vitam_workspace_port}}"
vitam_workspace_baseuri: "/workspace"

vitam_processing_host: "processing.service.{{consul_domain}}"
vitam_processing_port: 8203
vitam_processing_baseurl: "http://{{vitam_processing_host}}:{{vitam_processing_port}}"
vitam_processing_baseuri: "/processing"

vitam_worker_port: 9104
vitam_worker_baseuri: "/worker"

vitam_storageengine_host: "storage.service.{{consul_domain}}"
vitam_storageengine_port: 9102
vitam_storageengine_baseurl: "http://{{vitam_storageengine_host}}:{{vitam_
↪storageengine_port}}"
vitam_storageengine_baseuri: "/storage-engine"

vitam_storageofferdefault_port: 9900
vitam_storageofferdefault_port_https: 9901
vitam_storageofferdefault_baseuri: "/storage-offer-default"

vitam_functional_administration_host: "functional-administration.service.{{consul_
↪domain}}"
vitam_functional_administration_port: 8004
vitam_functional_administration_baseurl: "http://{{vitam_functional_administration_
↪host}}:{{vitam_functional_administration_port}}"
vitam_functional_administration_baseuri: "/functional-administration"

# Normally no need for the host ? Maybe use the same strategy as data ?
elasticsearch_log_host: "elasticsearch-log.service.{{consul_domain}}"
elasticsearch_log_http_port: "9201"
elasticsearch_log_tcp_port: "9301"

elasticsearch_data_http_port: "9200"
elasticsearch_data_tcp_port: "9300"

mongo_base_path: "{{vitam_folder_root}}"
mongos_port: 27017
mongoc_port: 27018
mongod_port: 27019
mongo_authentication: "true"
mongoclientDbName: "mongoclient"
mongoclientPort: 27016
mongoclientbaseUrl: "/mongoclient"

vitam_mongodb_host: "mongos.service.{{consul_domain}}"
vitam_mongodb_port: "{{mongos_port}}"

vitam_logstash_host: "{{ groups['hosts-log-server'][0] }}"
vitam_logstash_port: 10514

# Normally no need for the host ?
```

```
vitam_kibana_host: "kibana.service.{{consul_domain}}"
vitam_kibana_port: 5601

vitam_curator_host: "{{ (groups['hosts-log-server'] | length > 0) | ternary(groups[
↪'hosts-log-server'][0], '') }}"

vitam_library_port: 8090

vitam_siegfried_port: 19000

vitam_user: vitam
vitamdb_user: "vitamdb"
vitam_group: vitam

consul_domain: consul

vitam_folder_permission: 0750

vitam_conf_permission: 0640

consul_component: consul
consul_folder_conf: "{{vitam_folder_root}}/conf/{{consul_component}}"

soapui_component: soapui
soapui_folder_app: "{{vitam_folder_root}}/app/{{soapui_component}}"

mongod_folder_database: "{{vitam_folder_root}}/data/mongod/db"
mongoc_folder_database: "{{vitam_folder_root}}/data/mongoc/db"

service_restart_timeout: 30
clamav_port: 3310
```

Le fichier `vault-vitam.yml` est également présent sous `environments-rpm /group_vars/all/all` et contient les secrets ; ce fichier est encrypté par `ansible-vault` et doit être paramétré avant le lancement de l'orchestration de déploiement.

```
#plateforme_secret: vitamsecret
#mongoAdminUser: vitamdb-admin
#mongoAdminPassword: azerty
#mongoMetadataUser: metadata
#mongoMetadataPassword: azerty
#mongoLogbookUser: logbook
#mongoLogbookPassword: azerty
#mongoFunctionalAdminUser: functional-admin
#mongoFunctionalAdminPassword: azerty
#mongoClientUser: mongoclient
#mongoClientPassword: azerty
#mongoPassPhrase: mongogo
#vitam_keystone_passwd: 2kwRWUKXyjR65VtUCma1vd5TS8DFZjQnpeJ0sLbN
#vitam_users:
#  - vitam_uuser:
#    login: uuser
#    password: uuser1234
#  - vitam_aadmin:
#    login: aadmin
#    password: aadmin1234
#  - vitam_gguest:
#    login: gguest
```

```
#    password: gguest1234
#  - techadmin:
#    login: techadmin
#    password: techadmin1234
#
```

Note : Si le mot de passe du fichier `vault-vitam.yml` est changé, ne pas oublier de le répercuter dans le fichier `vault_pass.txt` (et le sécuriser à l'issue de l'installation).

Le fichier `vault-extra.yml` peut être également présent sous `environments-rpm/group_vars/all/all` et contient des secrets supplémentaires ; 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.

```
# Example for git lfs ; uncomment & use if needed
#vitam_gitlab_itest_login: "account"
#vitam_gitlab_itest_password: "password"
```

Note : pour , utiliser le même mot de passe que `vault-vitam.yml`.

Le déploiement s'effectue depuis la machine "ansible" et va distribuer la solution VITAM selon l'inventaire correctement renseigné.

Avertissement : 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.

9.2.3.2 Paramétrage de mongoclient (administration mongoclient)

Le package vitam-mongoclient nécessite une bases de données mongoDB (mongoclient) pour stocker sa configuration. Cette base de données est créée dans *VITAM* durant la première installation. La configuration est également générée en fonction des paramètres de l'inventaire.

Mongoclient permet de se connecter aux différentes bases de données mongoDB utilisées par VITAM.

9.2.3.3 Première utilisation de mongoclient

Par défault, mongoclient est accessible par l'url : http://hostname:27016/mongoclient suivant les hôtes configurés dans le groupes hosts-mongoclients de l'inventaire Vitam.

Avertissement : les versions de mongoclient inférieures à la version 1.5.0 présentent un message d'erreur "route not found" à l'apparition de l'interface. les fonctionnalités de l'application sont indisponibles dans cet état. Ce problème est aisemment contournable en cliquant sur le bouton "Go to Dashboard" pour revenir à un état normal de l'application.

Lors de la première utilisation de mongoclient, il convient de configurer les connexions aux bases de données à superviser. (Cette procédure devrait disparaître à l'issue de la phase Beta)

Procédure pour configurer la connexion aux bases vitam : :

1. Cliquer sur le bouton "Connect" situé en haut de la page (l'emplacement dépend de la taille de la fenêtre)
2. Dans la fenêtre "Connections", cliquer sur le bouton "Create New". => la fenêtre Add connection apparait contenant 4 sections : Connection, Authentication, URL, SSH
3. Dans la section "Connection", saisir un nom à donner à la connexion dans "name", le nom ou l'ip du server mongos à cibler dans "hostname", changer éventuellement le "port", définir la base de donnée sur laquelle le client doit se connecter
4. Dans la section "Authentication", saisir les paramètres d'autentification du compte à utiliser pour se connecter à la base configurée en section "connection"
5. Dans la section URL, en fonction du la configuration des services, choisir cette méthode de connexion en lieu et place des autres méthodes.
6. Dans la section "SSH", si le service mongoDB n'est accessible qu'au travers d'une connexion SSH, renseigner les paramètres de cette connexion pour accéder au serveur.
7. Sauvegarder les paramètres avec le boutton "save changes"
8. La nouvelle connexion doit apparaître avec un résumé de ses paramètres dans la fenêtre "Connections"
9. CLiquer sur la ligne de la connexion puis cliquer sur le boutton "Connect Now" pour utiliser se connecter.

Si les identifiants utilisés disposent de droit suffisants, Mongoclient vas afficher les métriques du service mongoDB.

Mongoclient ne permet de gérer qu'une seule base à la fois, il est toutefois possible de changer de base de donnée rapidement en ouvrant le menu "More" => "Switch Database" qui affichera la liste des bases de données accessibles (suivant les identifiants renseignés).

9.2.3.3.1 Paramétrage de l'antivirus (ingest-externe)

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

- Créer un autre shell (dont l'extension doit être `.sh.j2`) sous `ansible-vitam-rpm/roles/vitam/templates/ingest`
  prendre comme modèle le fichier `scan-clamav.sh.j2`. Ce fichier est un template Jinja2, et peut donc contenir des variables qui seront interprétées lors de l'installation.
- Modifier le fichier `ansible-vitam-rpm/roles/vitam/templates/ingest-external/ingest-external.co`
  en pointant sur le nouveau fichier.

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

9.2.3.3.2 Paramétrage des certificats (*-externe)

Se reporter à l'étape "PKI" du déploiement, décrite plus bas.

9.2.3.4 Déploiement

9.2.3.4.1 Fichier de mot de passe

Si le fichier `deployment/vault_pass.txt` est renseigné avec le mot de passe du fichier `environnements-rpm/group_vars/all/vault.yml`, le mot de passe ne sera pas demandé. Si le fichier est absent, le mot de passe du "vault" sera demandé.

9.2.3.4.2 PKI

Se positionner dans le répertoire `deployment`.

1. paramétrer le fichier `environnements-rpm/group_vars/all/vault.yml` et le fichier d'inventaire de la plate-forme sous `environnements-rpm` (se baser sur le fichier hosts.example)
2. En absence d'une PKI, exécuter le script

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

Note : En cas d'absence de PKI, il permet de générer une PKI, ainsi que des certificats pour les échanges https entre composants. Se reporter au chapitre PKI si le client préfère utiliser sa propre PKI.

3. Génération des certificats, si aucun n'est fourni par le client

```
pki/scripts/generate_certs.sh <environnement>
```

Note : Basé sur le contenu du fichier `vault.yml`, ce script génère des certificats nécessaires au bon fonctionnement de VITAM.

3. Génération des stores Java, s'ile ne sont pas fournis par le client

```
./generate_stores.sh <environnement>
```

Note : Basé sur le contenu du fichier `vault.yml`, ce script génère des stores nécessaires au bon fonctionnement de VITAM et les positionne au bon endroit pour le déploiement.

9.2.3.4.3 Mise en place des repositories VITAM (optionnel)

Si gestion par VITAM des repositories CentOS spécifiques à VITAM :

Editer le fichier `environments-rpm/group_vars/all/repo.yml` à partir des modèles suivants (décommenter également les lignes) :

Pour une cible de déploiement CentOS :

```
#vitam_repositories:
#- key: repo 1
#  value: "file:///code"
#  proxy: http://proxy
#- key: repo 2
#  value: "http://www.programmevitam.fr"
#  proxy: _none_
#- key: repo 3
#  value: "ftp://centos.org"
#  proxy:
```

Pour une cible de déploiement Debian :

```
#vitam_repositories:
#- key: repo 1
#  value: "file:///code"
#  subtree: "./"
#  trusted: "[trusted=yes]"
#- key: repo 2
#  value: "http://www.programmevitam.fr"
#  subtree: "./"
#  trusted: "[trusted=yes]"
#- key: repo 3
#  value: "ftp://centos.org"
#  subtree: "binary"
#  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-rpm-extra/bootstrap.yml -i
environments-rpm/<fichier d'inventaire> --ask-vault-pass
```

ou

```
ansible-playbook ansible-vitam-rpm-extra/bootstrap.yml -i
environments-rpm/<fichier d'inventaire> --vault-password-file vault_pass.txt
```

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

9.2.3.4.4 Déploiement

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

```
ansible-playbook ansible-vitam-rpm/vitam.yml -i environments-rpm/<ficher d'inventaire>
↪ --vault-password-file vault_pass.txt
```

9.2.3.4.5 Extra

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

1. ihm-recette

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

```
ansible-playbook ansible-vitam-rpm-extra/ihm-recette.yml -i environments-rpm/<ficher d
↪'inventaire> --vault-password-file vault_pass.txt
```

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-rpm-extra/extra.yml -i environments-rpm/<ficher d
↪'inventaire> --vault-password-file vault_pass.txt
```

9.2.3.5 Import automatique d'objets dans Kibana

Il peut être utile de vouloir automatiquement importer dans l'outil de visualisation Kibana des dashboards préalablement crées. Cela ce fait simplement avec le système d'import automatique mis en place. Il suffit de suivre les différentes étapes :

1. Ouvrir l'outil Kibana dans son navigateur.
2. Créer ses dashboards puis sauvegarder.
3. Aller dans l'onglets **Settings** puis **Objects**.
4. Sélectionner les composants à exporter puis cliquer sur le bouton **Export**. (ou bien cliquer sur **Export Everything** pour tout exporter).
5. Copier le/les fichier(s) *.json* téléchargés à l'emplacement `deployment\ansible-vitam-rpm\roles\log-server\fil`
6. Les composants sont prêts à être importés automatique lors du prochain déploiement.

Pour éviter d'avoir à recréer les "index-pattern" définis dans l'onglet **Settings** de Kibana, ceux-ci aussi sont pris en charge par le système de déploiement automatique. En revanche ils ne sont pas exportables, il est donc nécessaire de créer à la main le fichier *.json* correspondant. Pour ce faire :

1. Faire une requête GET sur l'url suivante `http://<ip-elasticsearch-log>/.kibana/index-pattern/_search`.
2. Récupérer le contenu au format JSON et extraire le contenu de la clé **hits.hits** (qui doit être un tableau).
3. Copier ce tableau dans un fichier.
4. Copier le fichier créé à l'étape 3 dans l'emplacement `deployment\ansible-vitam-rpm\roles\log-server\files\`
5. Les index-pattern sont prêts à être importés.

9.2.4 Procédure de mise à niveau

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.

La mise à jour peut actuellement être effectuée comme une "première installation".

CHAPITRE 10

Validation de la procédure

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

10.1 Sécurisation du fichier `vault_pass.txt`

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

10.2 Validation manuelle

Chaque service VITAM (en dehors de bases de données) expose des URL de statut présente à l'adresse suivante : `<protocole web https 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 https ou https>://<host>:<port>/admin/v1/version`

10.3 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.

10.4 Validation via SoapUI

Pour les environnements de recette, il est possible de lancer les tests de validation métier au sein de l'interface du composant IHM-recette (menu > tests SOAP-UI).

Prudence : Cette méthode de validation est dépréciée en Release3 et sera prochainement supprimée.

10.5 Post-installation : administration fonctionnelle

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

- le référentiel PRONOM (lien vers pronom [8]) 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.

10.5.1 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.

Si le fichier vault-password est renseigné :

```
ansible-playbook ansible-vitam-rpm-extra/init_pronom.yml -i
environments-rpm/<fichier d'inventaire> --vault-password-file vault_pass.txt
```

Sinon

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

Prudence : le playbook ne se termine pas correctement (code HTTP 403) si un référentiel PRONOM a déjà été chargé.

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

CHAPITRE 11

Troubleshooting

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

CHAPITRE 12

Présentation

Cette section a vocation à répertorier les différents problèmes rencontrés et apporter la solution la plus appropriée ; elle est amenée à être régulièrement mise à jour pour répertorier les problèmes rencontrés.

CHAPITRE 13

Elements extras de l’installation

Cette section est actuellement vide.

CHAPITRE 14

Contacts et support

14.1 Contacts

En cas de problème, il convient d'ouvrir un ticket à l'URL suivante : https://support.programmevitam.fr

Dans ce ticket, il est nécessaire d'expliciter le contexte (comportement observé vs comportement attendu), tant fonctionnel que technique (copies écran et code d'erreur sont utiles). Si possible et applicable, fournir également le jeu de tests, déterminer le niveau de criticité du problème, son taux de reproduction et un lien avec l'US concerné, si applicable.

Avertissement : Le support ne prend en charge l'appel qu'à la fourniture d'un numéro de ticket valide.

Suite à l'ouverture du ticket, le programme VITAM étudie et qualifie le problème rencontré :

- Bloquant, qui empêche toute action de recette
- Majeur, qui bloque une action de recette mais permet d'en continuer d'autres
- Mineur, à corriger après la recette

CHAPITRE 15

Annexes

Table des figures

3.1 Vue d'ensemble d'un déploiement VITAM : zones, composants 9

9.1 Vue d'ensemble de la gestion des certificats au déploiement 23

Liste des tableaux

2.1 Documents de référence VITAM 5
6.1 Tableau récapitulatif des informations à renseigner pour VITAM 16
9.1 Définition des variables 40

A
API, 6

B
BDD, 6

C
COTS, 6

D
DAT, 6
Deb, 6
DEX, 6
DIN, 6
DNSSEC, 6

I
IHM, 6

J
JRE, 6
JVM, 6

M
MitM, 6

N
NoSQL, 6

O
OAIS, 7

P
PDMA, 6
PKI, 7

R
RPM, 6

S
SIA, 7

T
TNR, 7

V
VITAM, 6