VITAM - Documentation d'installation

Version 0.26.1

VITAM

nov. 22, 2017

Table des matières

CHAPITRE 1

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.

CHAPITRE 2

Rappels

2.1 Information concernant les licences

La solution logicielle *VITAM* est publiée sous la license CeCILL 2.1 [1] ; la documentation associée (comprenant le présent document) est publiée sous Licence Ouverte V2.0 [2].

2.2 Documents de référence

2.2.1 Documents internes

Tableau 2.1 – Documents de référence VITAM

Nom	Lien
DAT	http://www.programmevitam.fr/ressources/DocCourante/html/archi
DIN	http://www.programmevitam.fr/ressources/DocCourante/html/installation
DEX	http://www.programmevitam.fr/ressources/DocCourante/html/exploitation
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
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 [3]

DUA Durée d'Utilité Administrative

IHM Interface Homme Machine

JRE Java Runtime Environment ; il s'agit de la machine virtuelle Java permettant d'y exécuter les programmes compilés pour.

JVM Java Virtual Machine ; Cf. *JRE*

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

NoSQL Base de données non-basée sur un paradigme classique des bases relationnelles. Définition NoSQL [5]

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

PDMA Perte de Données Maximale Admissible ; il s'agit du pourcentage de données stockées dans le système qu'il est acceptable de perdre lors d'un incident de production.

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

REST REpresentational State Transfer : type d'architecture d'échanges. Appliqué aux services web, en se basant sur les appels http standard, il permet de fournir des API dites "RESTful" qui présentent un certain nombre d'avantages en termes d'indépendance, d'universalité, de maintenabilité et de gestion de charge. Définition REST [7]

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

SAE Système d'Archivage Électronique

SEDA Standard d'Échange de Données pour l'Archivage

SIA Système d'Informations Archivistique

TNR Tests de Non-Régression

VITAM Valeurs Immatérielles Transférées aux Archives pour Mémoire

3. https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions
4. 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_publiques
7. https://fr.wikipedia.org/wiki/Representational_state_transfer

CHAPITRE 3

Prérequis à l'installation

3.1 Pré-requis plate-forme

Les pré-requis suivants sont nécessaires :

3.1.1 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 [8] 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 cibles (fichier ~/.ssh/known_hosts correctement renseigné)

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 containeurs docker (mongo-express, head), qu'elles aient un accès internet.

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

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`.

3.1.2 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`.

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

3.1.2.1 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)

3.1.2.2 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)

3.1.3 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>`

3.2 Récupération de la version

3.2.1 Utilisation des dépôts open-source

Les scripts de déploiement de VITAM sont disponibles dans le dépôt github VITAM [9] , dans le répertoire `deployment`.

Les binaires de VITAM sont disponibles sur les dépôts bintray [10] ; ces dépôts doivent être correctement configurés sur la plate-forme cible avant toute installation.

3.2.1.1 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
```

9. https://github.com/ProgrammeVitam/vitam
10. https://bintray.com/programmevitam

3.2.1.2 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>
```

3.2.2 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" dévouée au déploiement de *VITAM*, décompacter le package (au format `tar.gz`).

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

CHAPITRE 4

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 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*.

4.2.1.1 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.

4.2.1.2 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 :

```
# Group definition ; DO NOT MODIFY
[hosts]

# Group definition ; DO NOT MODIFY
[hosts:children]
vitam
reverse
library
hosts-dev-tools


########### Tests environments specifics ###########

# EXTRA : Front reverse-proxy (test environments ONLY) ; add machine name after
[reverse]
# optional : after machine, if this machine is different from VITAM machines, you can specify another become user
# Example
# vitam-centos-01.vitam ansible_ssh_user=centos

########### Extra VITAM applications ###########

[library]
# TODO: Put here servers where this service will be deployed : library

[hosts-dev-tools]
# TODO: Put here servers where this service will be deployed : mongo-express, elasticsearch-head

[elasticsearch:children] # EXTRA : elasticsearch
hosts-elasticsearch-data
hosts-elasticsearch-log

########### VITAM services ###########

# Group definition ; DO NOT MODIFY
[vitam:children]
zone-external
zone-access
zone-applicative
zone-storage
zone-data
zone-admin


##### Zone externe


[zone-external:children]
hosts-ihm-demo
hosts-cerebro
hosts-ihm-recette

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

[hosts-ihm-recette]
# TODO: Put here servers where this service will be deployed : ihm-recette (extra feature)
```

```
57
58  [hosts-cerebro]
59  # TODO: Put here servers where this service will be deployed : vitam-elasticsearch-
    ↪cerebro
60
61
62  ##### Zone access
63
64  # Group definition ; DO NOT MODIFY
65  [zone-access:children]
66  hosts-ingest-external
67  hosts-access-external
68
69  [hosts-ingest-external]
70  # TODO: Put here servers where this service will be deployed : ingest-external
71
72
73  [hosts-access-external]
74  # TODO: Put here servers where this service will be deployed : access-external
75
76
77  ##### Zone applicative
78
79  # Group definition ; DO NOT MODIFY
80  [zone-applicative:children]
81  hosts-ingest-internal
82  hosts-processing
83  hosts-worker
84  hosts-access-internal
85  hosts-metadata
86  hosts-functional-administration
87  hosts-logbook
88  hosts-workspace
89  hosts-storage-engine
90
91  [hosts-logbook]
92  # TODO: Put here servers where this service will be deployed : logbook
93
94
95  [hosts-workspace]
96  # TODO: Put here servers where this service will be deployed : workspace
97
98
99  [hosts-ingest-internal]
100 # TODO: Put here servers where this service will be deployed : ingest-internal
101
102
103 [hosts-access-internal]
104 # TODO: Put here servers where this service will be deployed : access-internal
105
106
107 [hosts-metadata]
108 # TODO: Put here servers where this service will be deployed : metadata
109
110
111 [hosts-functional-administration]
112 # TODO: Put here servers where this service will be deployed : functional-
    ↪administration
```

```
113
114
115 [hosts-processing]
116 # TODO: Put here servers where this service will be deployed : processing
117
118
119 [hosts-storage-engine]
120 # TODO: Put here servers where this service will be deployed : storage-engine
121
122
123 [hosts-worker]
124 # TODO: Put here servers where this service will be deployed : worker
125 # Optional parameter after each host : vitam_worker_capacity=<integer> ; please refer␣
    ↪to your infrastructure for defining this number ; default is 1
126
127
128 ##### Zone storage
129
130 [zone-storage:children] # DO NOT MODIFY
131 hosts-storage-offer-default
132
133
134 [hosts-storage-offer-default]
135 # TODO: Put here servers where this service will be deployed : storage-offer-default
136 # LIMIT : only 1 offer per machine and 1 machine per offer
137 # Mandatory param for each offer is offer_conf and points to offer_opts.yml & vault-
    ↪vitam.yml (with same tree)
138 # hostname-offre-1.vitam offer_conf=offer_swift_1
139 # for filesystem
140 # hostname-offre-2.vitam offer_conf=offer_fs_1
141
142
143 ##### Zone data
144
145 # Group definition ; DO NOT MODIFY
146 [zone-data:children]
147 hosts-elasticsearch-data
148 mongo_common
149
150
151 [hosts-elasticsearch-data]
152 # TODO: Put here servers where this service will be deployed : elasticsearch-data␣
    ↪cluster
153
154
155 # Group definition ; DO NOT MODIFY
156 [mongo_common:children]
157 mongos
158 mongoc
159 mongod
160
161 [mongos]
162 # TODO: Put here servers where this service will be deployed : mongos cluster
163
164 [mongoc]
165 # TODO: Put here servers where this service will be deployed : mongoc cluster
166
167 [mongod] # mongod declaration ; add host name after
```

```
# TODO: Put here servers where this service will be deployed : mongod cluster
# Each replica_set should have an odd number of members (2n + 1)
# Reminder: For Vitam, one mongodb shard is using one replica_set
# Each host need 2 vars:
#     - mongo_shard_id: id of the current shard, increment by 1 from 0 to n
#     - mongo_rs_bootstrap: mandatory for 1 node of the shard, some init commands will be executed on it
# Example:
# vitam-iaas-db-01.int  mongo_shard_id=0  mongo_rs_bootstrap=true
# vitam-iaas-db-02.int  mongo_shard_id=0
# vitam-iaas-db-03.int  mongo_shard_id=0
# vitam-iaas-db-04.int  mongo_shard_id=1  mongo_rs_bootstrap=true
# vitam-iaas-db-05.int  mongo_shard_id=1
# vitam-iaas-db-06.int  mongo_shard_id=1

###### Zone admin

# Group definition ; DO NOT MODIFY
[zone-admin:children]
hosts-consul-server
hosts-kibana-data
log-servers
hosts-elasticsearch-log

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

[hosts-kibana-data]
# TODO: Put here servers where this service will be deployed : kibana (for data cluster)

[log-servers:children]
hosts-kibana-log
hosts-logstash

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

[hosts-kibana-log]
# TODO: Put here servers where this service will be deployed : kibana (for log cluster)

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

########### Global vars ###########

[hosts:vars]

# ==============================
# VITAM
# ==============================

# Declare user for ansible on target machines
ansible_ssh_user=
# Can target user become as root ? ; true is required by VITAM (usage of a sudoer is mandatory)
```

```
ansible_become=true
# Environment (defines consul environment name ; in extra on homepage)
environnement=

# Related to Consul ; apply in a table your DNS server(s)
# Example : dns_servers=["8.8.8.8","8.8.4.4"]
dns_servers=

# Vitam tenants to create
vitam_tenant_ids=[0,1,2]

### Logback configuration ###
# Days before deleting logback log files (java & access logs for vitam components)
days_to_delete_logback_logfiles=

# Configuration for Curator
#       Days before deletion on log management cluster; 365 for production environment
days_to_delete_logstash_indexes=
#       Days before closing "old" indexes on log management cluster; 30 for production environment
days_to_close_logstash_indexes=

# ================================
# EXTRA
# ================================


# Configuration for Curator
#       Days before deletion for packetbeat index only on log management cluster
days_to_delete_packetbeat_indexes=5
#       Days before deletion for metricbeat index only on log management cluster; 30 for production environment
days_to_delete_metricbeat_indexes=30
# Days before closing metrics elasticsearch indexes
days_to_close_metrics_indexes=7
# Days before deleting metrics elasticsearch indexes
days_to_delete_metrics_indexes=30

### vitam-itest repository ###
vitam_tests_branch=master
vitam_tests_gitrepo_protocol=
vitam_tests_gitrepo_baseurl=
vitam_tests_gitrepo_url=

# Used when VITAM is behind a reverse proxy (provides configuration for reverse proxy && displayed in header page)
vitam_reverse_external_dns=
# For reverse proxy use
reverse_proxy_port=80
# http_proxy env var to use
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*)

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

```
---


# Business vars

### Admin context name ###
admin_context_name: "admin-context"
# Indicate certificates relative paths under {{inventory_dir}}/certs/client-external/clients
admin_context_certs: [ "ihm-demo/ihm-demo.crt", "ihm-recette/ihm-recette.crt", "reverse/reverse.crt" ]
admin_context_tenants: "{{vitam_tenant_ids}}"

# Admin security profile name
admin_security_profile: "admin-security-profile"
```

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

```
# This list can and has to be completed if more offers are necessary
# DON'T forget to add associated passwords in vault-vitam.yml with same tree when using provider openstack-swift

vitam_offers:
  offer_fs_1:
    # param can be filesystem or filesystem-hash
    provider: filesystem
  offer_swift_1:
    provider: openstack-swift
    # keystone URL
    keystone_auth_url: http://hostname-rados-gw:port/auth/1.0
    #
    swift_uid: tenant$user
    #
    swift_subuser: subuser
```

4.2.1.3 Déclaration des secrets

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 mot 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 :

```
plateforme_secret: vitamsecret
cerebro_secret_key: tGz28hJkiW[p@a34G
mongoAdminUser: vitamdb-admin
```

```
mongoAdminPassword: azerty
mongoLocalAdminUser: vitamdb-localadmin
mongoLocalAdminPassword: qwerty
mongoMetadataUser: metadata
mongoMetadataPassword: azerty
mongoLogbookUser: logbook
mongoLogbookPassword: azerty
mongoFunctionalAdminUser: functional-admin
mongoFunctionalAdminPassword: azerty
mongoSecurityInternalUser: security-internal
mongoSecurityInternalPassword: azerty
mongoPassPhrase: mongogo
consul_encrypt: Biz14ohqN4HtvZmrXp3N4A==
vitam_users:
  - vitam_aadmin:
    login: aadmin
    password: aadmin1234
    role: admin
  - vitam_uuser:
    login: uuser
    password: uuser1234
    role: user
  - vitam_gguest:
    login: gguest
    password: gguest1234
    role: guest
  - techadmin:
    login: techadmin
    password: techadmin1234
    role: admin
vitam_offers:
  offer_swift_1:
    swift_password: password

```

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

```
keystores:
  server:
    offer: azerty1
    access_external: azerty2
    ingest_external: azerty3
    ihm_recette: azerty16
  client_external:
    ihm_demo: azerty4
    gatling: azerty4
    ihm_recette: azerty5
    reverse: azerty6
  client_storage:
    storage: azerty7
  timestamping:
    secure_logbook: azerty8
truststores:
  server: azerty9
  client_external: azerty10
  client_storage: azerty11
grantedstores:
```

```
21  client_external: azerty12
22  client_storage: azerty13
```

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.

4.2.2 Gestion des certificats

Une vue d'ensemble de la gestion des certificats est présentée *dans l'annexe dédiée annexes* (page 37).

4.2.2.1 Cas 1 : Je ne dispose pas de PKI, je souhaite utiliser celle de Vitam

Dans ce cas, il est nécessaire d'utiliser la *PKI* fournie avec la solution logicielle VITAM.

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

4.2.2.1.2 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.

4.2.2.1.3 Génération des certificats par les scripts Vitam

Le fichier d'inventaire de déploiement `environments/<fichier d'inventaire>` (cf. *Informations "plate-forme"* (page 9)) 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 `environmements/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 environmements/certs/vault-certs.yml

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

4.2.2.1.4 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 : `environmements/group_vars/all/vault-keystores.yml`, décrit dans la section *Mise en place des repositories VITAM (optionnel)* (page 26).

Lancer le script :

```
./generate_stores.sh
```

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

4.2.2.2 Cas 2 : Je dispose d'une PKI

4.2.2.2.1 Procédure générale

Si vous disposez d'une PKI, il n'est pas nécessaire d'utiliser celle de Vitam.

Il est par contre être nécessaire de :

- déposer les certificats et les autorités de certifications correspondantes dans les bon répertoires.
- renseigner les mots de passe des clés privées des certificats dans le vault ansible environmements/certs/vault-certs.yml
- utiliser le script Vitam permettant de générer les différents keystores.

4.2.2.2.2 Intégration de certificats existants

Si vous possédez déjà une *PKI*, il convient de positionner les certificats et CA sous `environmements/certs/....` en respectant la structure indiquée ci-dessous.

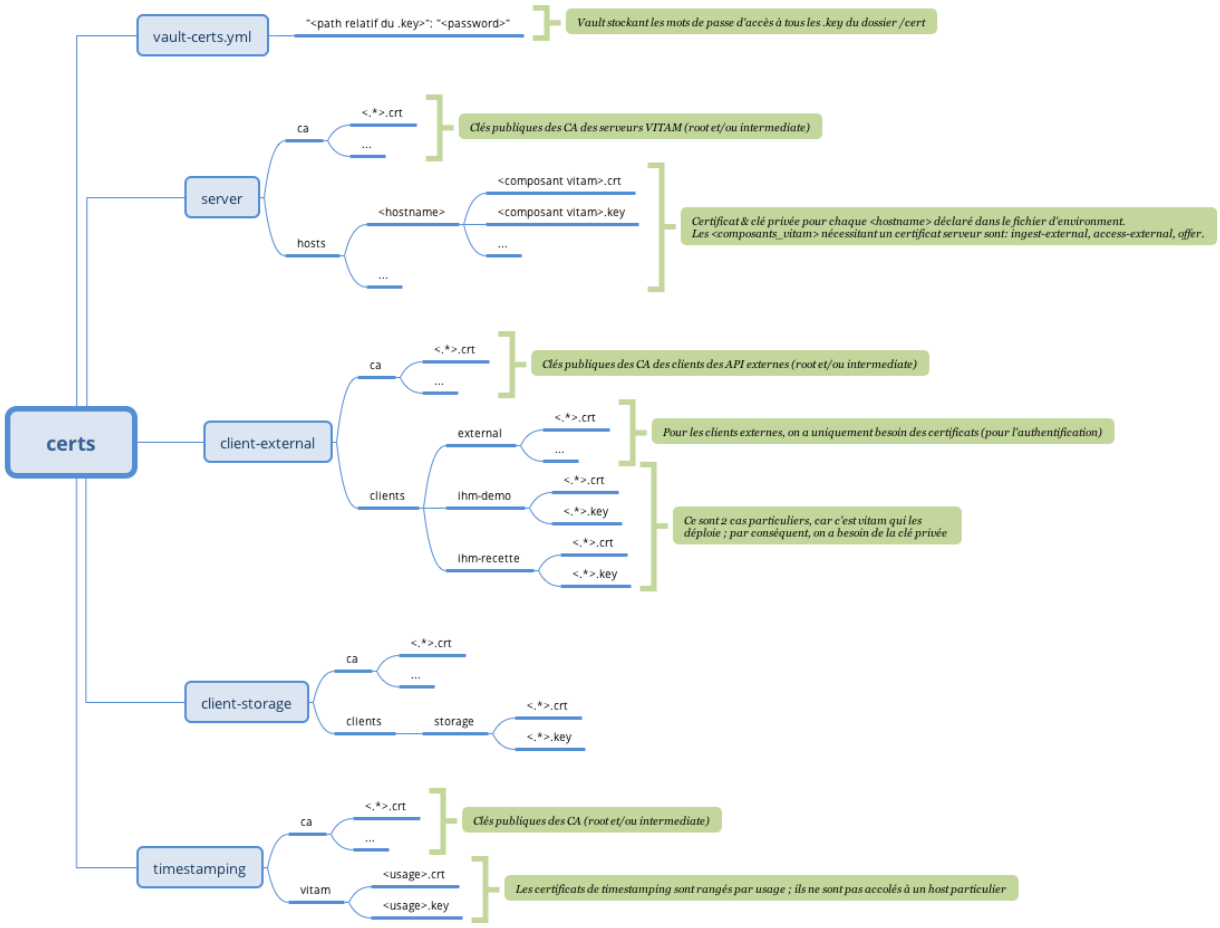

Fig. 4.1 – Vue détaillée de l'arborescence des certificats

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 lncer le déploiement.

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

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

4.2.2.2.3 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 : `environmements/group_vars/all/vault-keystores.yml`, décrit dans la section *Mise en place des repositories VITAM (optionnel)* (page 26).

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

Lancer le script :

```
./generate_stores.sh
```

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

4.2.3 Paramétrages supplémentaires

4.2.3.1 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 : parmètres gc
- java : autres paramètres java

4.2.3.2 Paramétrage de l'antivirus (ingest-externe)

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

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

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

4.2.3.3 Paramétrage des certificats externes (*-externe)

Se reporter au chapitre dédié à la gestion des certificats : *Gestion des certificats* (page 17)

4.2.3.4 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.

4.2.3.4.1 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 grou

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

4.2.3.4.2 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 [12].

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

4.2.3.5 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 :

12. http://www.rsyslog.com/doc/v7-stable/

```
---

### global ###

# TODO MAYBE : permettre la surcharge avec une syntax du genre vitamopts.folder_root | default(vitam_default.folder_root) dans les templates ?

vitam_defaults:
    folder:
        root_path: /vitam
        folder_permission: "0750"
        conf_permission: "0640"
    users:
        vitam: "vitam"
        vitamdb: "vitamdb"
        group: "vitam"
    services:
        log_level: WARN
        start_timeout: 150
        stop_timeout: 3600
        port_http_timeout: 86400
    # Filter for the vitam package version to install
    # FIXME : commented as has to be removed becuase doesn't work under Debain
    #package_version: "*"
    ### Trust X-SSL-CLIENT-CERT header for external api auth ? (true | false) ###
    vitam_ssl_user_header: true
    # syslog_facility
    syslog_facility: local0


### consul ###
# FIXME: Consul à la racine pour le moment à cause de problèmes de récursivité dans le parsing yaml
# TODO : consul_domain should be in inventory as choosable by customer
consul_domain: consul
consul_component: consul
consul_folder_conf: "{{vitam_defaults.folder.root_path}}/conf/{{consul_component}}"



### Composants Vitam ###

vitam:
    accessexternal:
        vitam_component: access-external
        host: "access-external.service.{{consul_domain}}"
        port_http: 8102
        port_admin: 28102
        port_https: 8444
        baseuri: "access-external"
        https_enabled: true
        secret_platform: "false"
    accessinternal:
        vitam_component: access-internal
        host: "access-internal.service.{{consul_domain}}"
        port_http: 8101
        port_admin: 28101
```

```
        baseuri: "access-internal"
        https_enabled: false
        secret_platform: "true"
    functional_administration:
        vitam_component: functional-administration
        host: "functional-administration.service.{{consul_domain}}"
        port_http: 8004
        port_admin: 18004
        baseuri: "functional-administration"
        https_enabled: false
        secret_platform: "true"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
    ingestexternal:
        vitam_component: ingest-external
        host: "ingest-external.service.{{consul_domain}}"
        port_http: 8001
        port_admin: 28001
        port_https: 8443
        baseuri: "ingest-external"
        https_enabled: true
        secret_platform: "false"
        antivirus: "clamav"
    ingestinternal:
        vitam_component: ingest-internal
        host: "ingest-internal.service.{{consul_domain}}"
        port_http: 8100
        port_admin: 28100
        baseuri: "ingest-internal"
        https_enabled: false
        secret_platform: "true"
    ihm_demo:
        vitam_component: "ihm-demo"
        host: "{{groups['hosts-ihm-demo'][0]}}"
        port_http: 8002
        port_admin: 28002
        baseurl: "/ihm-demo"
        static_content: "{{ vitam_defaults.folder.root_path }}/app/ihm-demo/v1"
        baseuri: "ihm-demo"
        static_content_v2: "{{ vitam_defaults.folder.root_path }}/app/ihm-demo/v2"
        baseuri_v2: "ihm-demo-v2"
        https_enabled: false
        secret_platform: "false"
    logbook:
        vitam_component: logbook
        host: "logbook.service.{{consul_domain}}"
        port_http: 9002
        port_admin: 29002
        baseuri: "logbook"
        https_enabled: false
        secret_platform: "true"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
    metadata:
        vitam_component: metadata
        host: "metadata.service.{{consul_domain}}"
        port_http: 8200
        port_admin: 28200
        baseuri: "metadata"
        https_enabled: false
```

```
        secret_platform: "true"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
    processing:
        vitam_component: processing
        host: "processing.service.{{consul_domain}}"
        port_http: 8203
        port_admin: 28203
        baseuri: "processing"
        https_enabled: false
        secret_platform: "true"
    security_internal:
        vitam_component: security-internal
        host: "security-internal.service.{{consul_domain}}"
        port_http: 8005
        port_admin: 28005
        baseuri: "security-internal"
        https_enabled: false
        secret_platform: "true"
    storageengine:
        vitam_component: storage
        host: "storage.service.{{consul_domain}}"
        port_http: 9102
        port_admin: 29102
        baseuri: "storage-engine"
        # FIXME: replace with https_enabled
        test_tls_offer_enabled: false
        https_enabled: false
        secret_platform: "true"
    storageofferdefault:
        vitam_component: "offer"
        port_http: 9900
        port_admin: 29900
        port_https: 9901
        baseuri: "storage-offer-default"
        https_enabled: false
        secret_platform: "true"
    worker:
        vitam_component: worker
        port_http: 9104
        port_admin: 29104
        baseuri: "worker"
        https_enabled: false
        secret_platform: "true"
    workspace:
        vitam_component: workspace
        host: "workspace.service.{{consul_domain}}"
        port_http: 8201
        port_admin: 28201
        baseuri: "workspace"
        https_enabled: false
        secret_platform: "true"
```

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

```
---

elasticsearch:
    log:
```

```
        host: "elasticsearch-log.service.{{ consul_domain }}"
        port_http: "9201"
        port_tcp: "9301"
        groupe: "log"
        baseuri: "elasticsearch-log"
        cluster_name: "elasticsearch-log"
        https_enabled: false
    data:
        host: "elasticsearch-data.service.{{ consul_domain }}"
        port_http: "9200"
        port_tcp: "9300"
        groupe: "data"
        baseuri: "elasticsearch-data"
        cluster_name: "elasticsearch-data"
        https_enabled: false

mongodb:
    mongos_port: 27017
    mongoc_port: 27018
    mongod_port: 27019
    mongo_authentication: "true"
    host: "mongos.service.{{ consul_domain }}"

logstash:
    user: logstash
    port: 10514
    rest_port: 20514

kibana:
    log:
        baseuri: "kibana_log"
        groupe: "log"
        port: 5601
    # KWA FIXME : changing port doesn't work, yet (not taken into account in kibana configuration)
    data:
        baseuri: "kibana_data"
        groupe: "data"
        port: 5601

cerebro:
    baseuri: "cerebro"
    port: 9000

siegfried:
    port: 19000

clamav:
    port: 3310

mongo_express:
    baseuri: "mongo-express"
```

4.2.4 Procédure de première installation

4.2.4.1 Déploiement

4.2.4.1.1 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`.

4.2.4.1.2 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 :

```
#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-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-".

4.2.4.1.3 Réseaux

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

4.2.4.1.3.1 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.

4.2.4.1.3.2 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.

4.2.4.1.3.3 Vérification de la génération des hostvars

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

4.2.4.1.4 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/<ficher 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).

4.2.4.1.5 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-extra/ihm-recette.yml -i environments/<ficher d
↪'inventaire> --ask-vault-pass
```

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/<ficher d'inventaire> -
↪-ask-vault-pass
```

4.2.5 Elements extras de l'installation

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

4.2.5.1 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
```

```
        host: "{{groups['hosts-ihm-recette'][0]}}"
        port_http: 8204
        port_https: 8445
        port_admin: 28204
        baseurl: /ihm-recette
        static_content: "{{ vitam_defaults.folder.root_path }}/app/ihm-recette"
        baseuri: "ihm-recette"
        secure_mode: authc
        https_enabled: true
        secret_platform: "false"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
    library:
        vitam_component: library
        host: "library.service.{{consul_domain}}"
        port_http: 8090
        port_admin: 28090
        baseuri: "doc"
        https_enabled: false
        secret_platform: "false"

docker_opts:
    registry_httponly: yes
    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.

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

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

4.2.5.2 Déploiement des extra

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

4.2.5.2.1 ihm-recette

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

```
ansible-playbook ansible-vitam-extra/ihm-recette.yml -i environments/<ficher d
↪'inventaire> --ask-vault-pass
```

4.2.5.2.2 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/<ficher d'inventaire> -
↪-ask-vault-pass
```

CHAPITRE 5

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 20)

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.

CHAPITRE 6

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

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

- le référentiel PRONOM (lien vers pronom [13]) 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 environments/<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 élements 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.

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

1. Le service ihm-demo est toujours dans l'état "critical" dans Consul ; cela correspond à une limitation connue. Cependant, cet état ne nuit en rien au bon fonctionnement du système.

6.4 Retour d'expérience / cas rencontrés

Mongo-express ne se connecte pas à la base de données associée Si mongoDB a été redémarré, il faut également redémarrer mongo-express.

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

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

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

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

Sur tous ces sujets, Cf. la documentation officielle [14].

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 [15]. 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.

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

CHAPITRE 7

Annexes

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

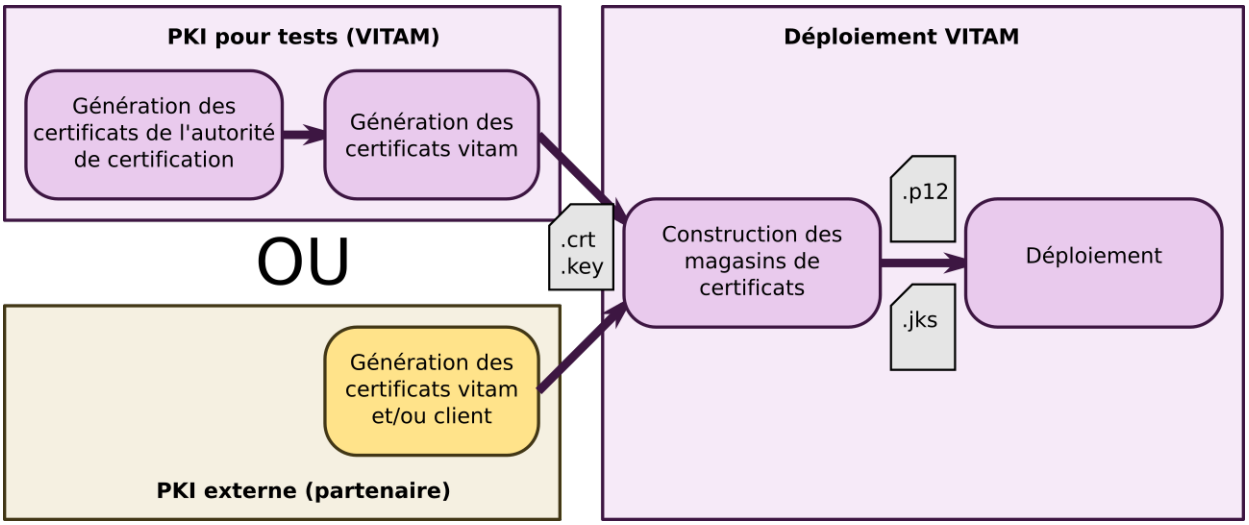

Fig. 7.1 – Vue d'ensemble de la gestion des certificats au déploiement

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 41)

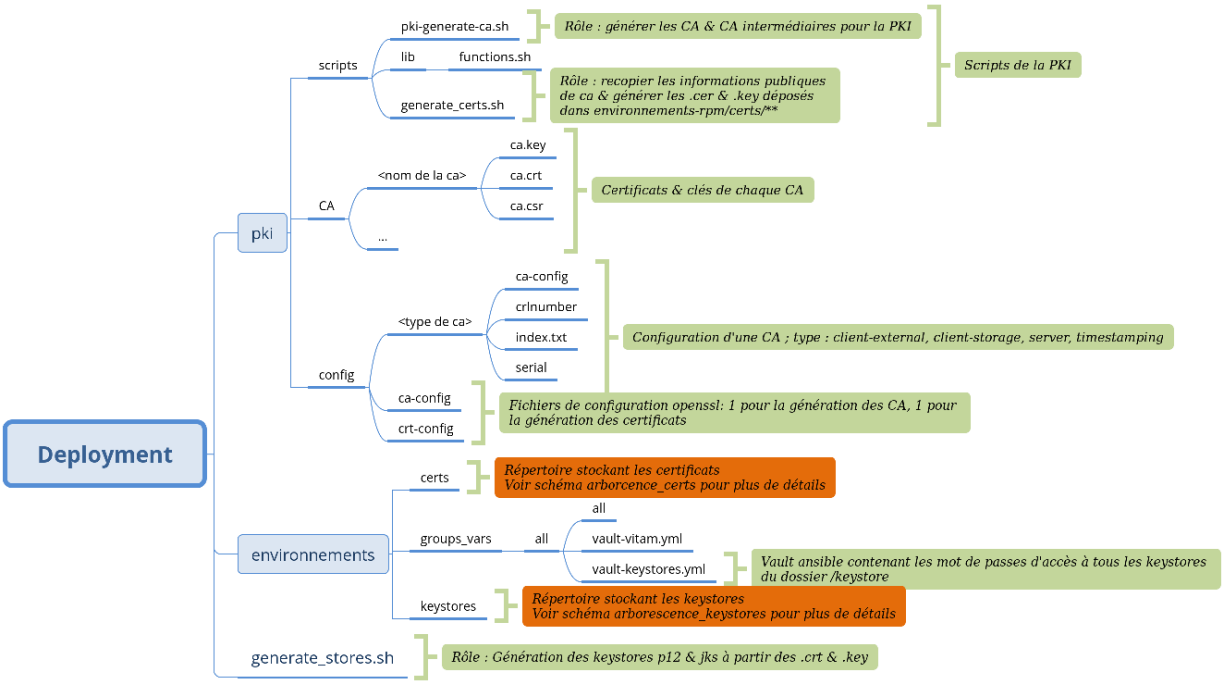

Fig. 7.2 – Vue l'arborescence de la PKI Vitam

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

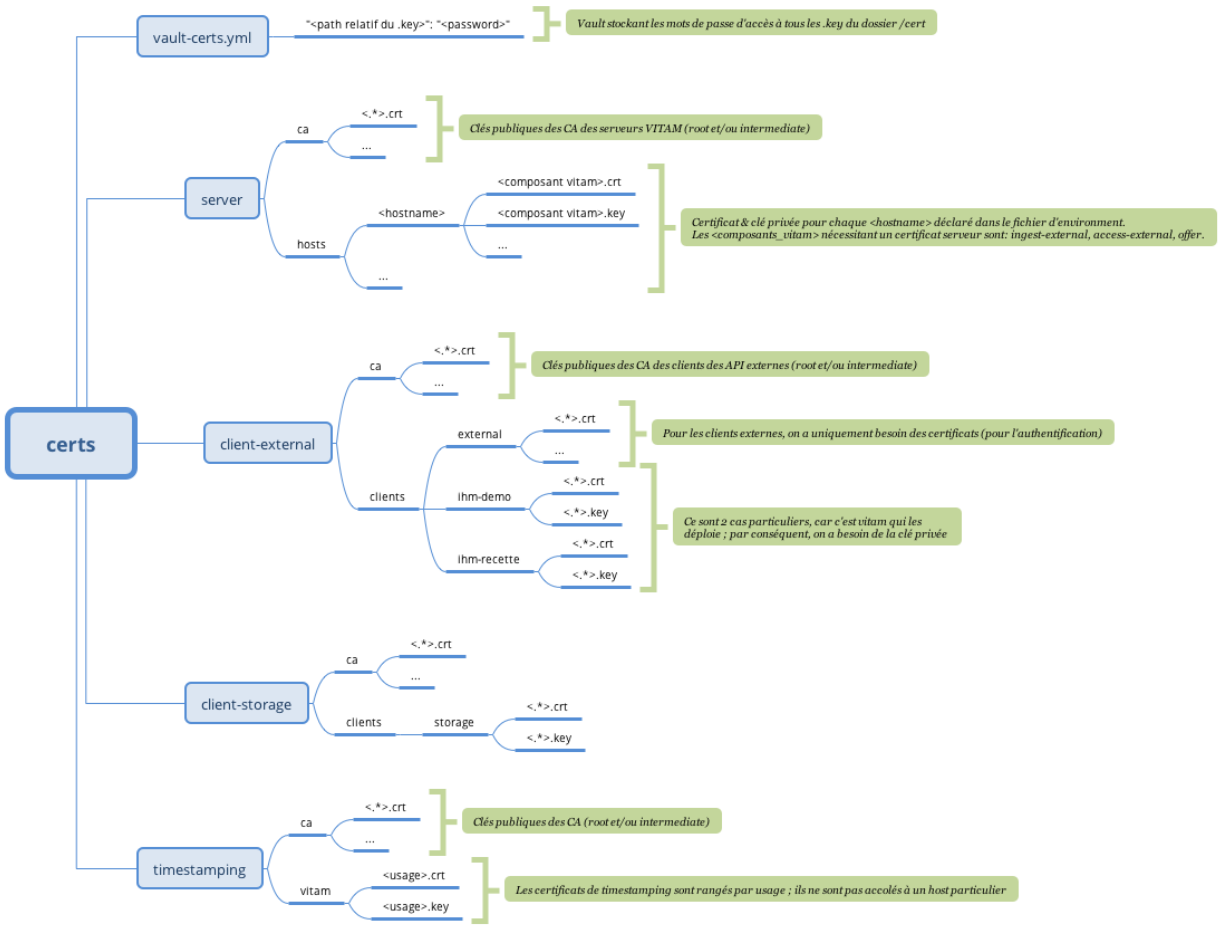

Fig. 7.3 – Vue détaillée de l'arborescence des certificats

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

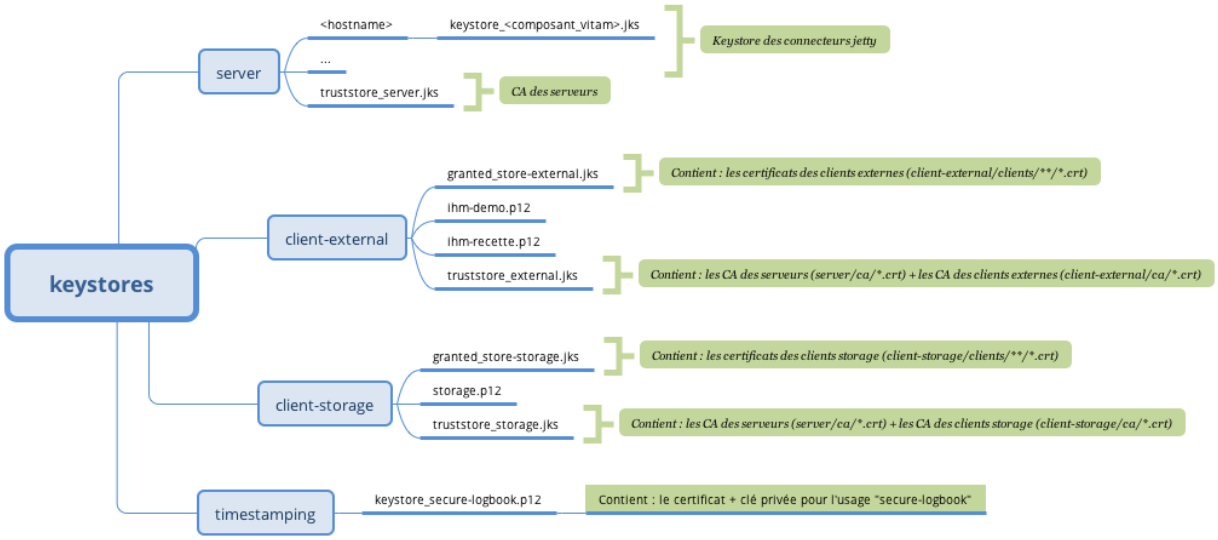

Fig. 7.4 – Vue détaillée de l'arborescence des keystores

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 *Vue détaillée de l'arborescence des certificats* (page 40))

7.2 Ansible & ssh

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

7.2.1 Authentification du compte utilisateur utilisé pour la connexion SSH

Pour le login du compte utilisateur, voir la section *Informations "plate-forme"* (page 9).

7.2.1.1 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.

7.2.1.2 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

7.2.1.3 Par clé SSH sans passphrase

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

7.2.2 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.

7.2.3 Elevation de privilèges

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

7.2.3.1 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`

7.2.3.2 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

7.2.3.3 Par sudo sans mot de passe

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

7.2.3.4 Déjà Root

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

CHAPITRE 8

Annexes

Table des figures

Liste des tableaux