VITAM - Documentation d'installation

Version 2.15.1-1

VITAM

déc. 11, 2019

Table des matières

CHAPITRE 1

Introduction

1.1 Objectif de ce document

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

Il s'adresse aux personnes suivantes :

- Les architectes techniques des projets désirant intégrer la solution logicielle *VITAM* ;
- Les exploitants devant installer la solution logicielle *VITAM*.

CHAPITRE 2

Rappels

2.1 Information concernant les licences

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

2.2 Documents de référence

2.2.1 Documents internes

Tableau 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
DMV	http://www.programmevitam.fr/ressources/DocCourante/html/migration
Release notes	https://github.com/ProgrammeVitam/vitam/releases/latest

2.2.2 Référentiels externes

2.3 Glossaire

API *Application Programming Interface*

AU *Archive Unit*, unité archivistique

[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

BDD Base De Données

BDO *Binary DataObject*

CA *Certificate Authority*, autorité de certification

CAS Content Adressable Storage

CCFN Composant Coffre Fort Numérique

CN Common Name

COTS Component Off The shelf ; il s'agit d'un composant « sur étagère », non développé par le projet *VITAM*, mais intégré à partir d'un binaire externe. Par exemple : MongoDB, ElasticSearch.

CRL *Certificate Revocation List* ; liste des identifiants des certificats qui ont été révoqués ou invalidés et qui ne sont donc plus dignes de confiance. Cette norme est spécifiée dans les RFC 5280 et RFC 6818.

CRUD *create, read, update, and delete*, s'applique aux opérations dans une base de données MongoDB

DAT Dossier d'Architecture Technique

DC Data Center

DEX Dossier d'EXploitation

DIN Dossier d'INstallation

DIP *Dissemination Information Package*

DMV Documentation de Montées de Version

DNS *Domain Name System*

DNSSEC *Domain Name System Security Extensions* est un protocole standardisé par l'IETF permettant de résoudre certains problèmes de sécurité liés au protocole DNS. Les spécifications sont publiées dans la RFC 4033 et les suivantes (une version antérieure de DNSSEC n'a eu aucun succès). Définition DNSSEC [3]

DSL *Domain Specific Language*, language dédié pour le requêtage de VITAM

DUA Durée d'Utilité Administrative

EBIOS Méthode d'évaluation des risques en informatique, permettant d'apprécier les risques Sécurité des systèmes d'information (entités et vulnérabilités, méthodes d'attaques et éléments menaçants, éléments essentiels et besoins de sécurité...), de contribuer à leur traitement en spécifiant les exigences de sécurité à mettre en place, de préparer l'ensemble du dossier de sécurité nécessaire à l'acceptation des risques et de fournir les éléments utiles à la communication relative aux risques. Elle est compatible avec les normes ISO 13335 (GMITS), ISO 15408 (critères communs) et ISO 17799

EAD Description archivistique encodée

ELK *Elasticsearch Logstash Kibana*

FIP *Floating IP*

GOT Groupe d'Objet Technique

IHM Interface Homme Machine

IP *Internet Protocol*

IsaDG Norme générale et internationale de description archivistique

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

JVM *Java Virtual Machine* ; Cf. *JRE*

LAN *Local Area Network*, réseau informatique local, qui relie des ordinateurs dans une zone limitée

LFC *LiFe Cycle*, cycle de vie

LTS *Long-term support*, support à long terme : version spécifique d'un logiciel dont le support est assuré pour une période de temps plus longue que la normale.

M2M *Machine To Machine*

[3] https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

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]

MoReq *Modular Requirements for Records System*, recueil d'exigences pour l'organisation de l'archivage, élaboré dans le cadre de l'Union européenne.

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

NTP *Network Time Protocol*

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

OOM Aussi apelé *Out-Of-Memory Killer* ; mécanisme de la dernière chance incorporé au noyau Linux, en cas de dépassement de la capacité mémoire

OS *Operating System*, système d'exploitation

OWASP *Open Web Application Security Project*, communauté en ligne de façon libre et ouverte à tous publiant des recommandations de sécurisation Web et de proposant aux internautes, administrateurs et entreprises des méthodes et outils de référence permettant de contrôler le niveau de sécurisation de ses applications Web

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

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

PCA Plan de Continuité d'Activité

PRA Plan de Reprise d'Activité

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

RGAA Référentiel Général d'Accessibilité pour les Administrations

RGI Référentiel Général d'Interopérabilité

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

SGBD Système de Gestion de Base de Données

SGBDR Système de Gestion de Base de Données Relationnelle

SIA Système d'Informations Archivistique

SIEM *Security Information and Event Management*

SIP *Submission Information Package*

SSH *Secure SHell*

[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

Swift *OpenStack Object Store project*

TLS *Transport Layer Security*

TNR Tests de Non-Régression

TTL *Time To Live*, indique le temps pendant lequel une information doit être conservée, ou le temps pendant lequel une information doit être gardée en cache

UDP *User Datagram Protocol*, protocole de datagramme utilisateur, un des principaux protocoles de télécommunication utilisés par Internet. Il fait partie de la couche transport du modèle OSI

UID *User IDentification*

VITAM Valeurs Immatérielles Transférées aux Archives pour Mémoire

VM *Virtual Machine*

WAF *Web Application Firewall*

WAN *Wide Area Network*, réseau informatique couvrant une grande zone géographique, typiquement à l'échelle d'un pays, d'un continent, ou de la planète entière

CHAPITRE 3

Prérequis à l'installation

3.1 Expertises requises

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

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

3.2 Pré-requis plate-forme

Les pré-requis suivants sont nécessaires :

3.2.1 Base commune

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

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

- packages nécessaires :
  - **ansible** (version **2.7** minimale et conseillée ; se référer à la documentation ansible [8] pour la procédure d'installation)
  - **openssh-client** (client SSH utilisé par ansible)

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

  - **java-1.8.0-openjdk** et **openssl** (du fait de la génération de certificats / *stores*, l'utilitaire `keytool` est nécessaire)
- un accès ssh vers un utilisateur d'administration avec élévation de privilèges vers les droits `root`, `vitam`, `vitamdb` (les comptes `vitam` et `vitamdb` sont créés durant le déploiement) sur les serveurs cibles.
- Le compte utilisé sur le serveur d'administration doit avoir confiance dans les serveurs sur lesquels la solution logicielle *VITAM* doit être installée (fichier `~/.ssh/known_hosts` correctement renseigné)

Note : Se référer à la documentation d'usage [9] pour les procédures de connexion aux machines-cibles depuis le serveur ansible.

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

Prudence : 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 (installation du paquet officiel `docker`, récupération des images).

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

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

Avertissement : Dans le cas d'une installation du composant `vitam-offer` en `tape-library`, il est fortement recommandé d'installer au préalable sur les machines cible associées les paquets pour les commandes `mt`, `mtx` et `dd`. Ces composants doivent également apporter le groupe système `tape`. Se reporter également à *Librairie de cartouches pour offre froide* (page 9).

3.2.2 PKI

La solution logicielle *VITAM* nécessite des certificats pour son bon fonctionnement (cf. *DAT* pour la liste des secrets et *Vue d'ensemble de la gestion des certificats* (page 92) pour une vue d'ensemble de leur usage.) La gestion de ces certificats, par le biais d'une ou plusieurs *PKI*, est à charge de l'équipe d'exploitation. La mise à disposition des certificats et des chaînes de validation *CA*, placés dans les répertoires de déploiement adéquats, est un pré-requis à tout déploiement en production de la solution logicielle *VITAM*.

Voir aussi :

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

Veuillez vous référer à la section *Vue d'ensemble de la gestion des certificats* (page 92) pour la liste des certificats nécessaires au déploiement de la solution VITAM, ainsi que pour leurs répertoires de déploiement.

3.2.3 Systèmes d'exploitation

Seules deux distributions Linux suivantes sont supportées à ce jour :

- CentOS 7
- Debian 9 (stretch)

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

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

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

3.2.3.1 Déploiement sur environnement CentOS

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

3.2.3.2 Déploiement sur environnement Debian

- Disposer d'une plate-forme Linux Debian « stretch » 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 stretch-backports
  - un accès internet, car le dépôt docker sera ajouté
- Disposer des binaires VITAM : paquets deb de VITAM (vitam-product) ainsi que les paquets d'éditeurs tiers livrés avec VITAM (vitam-external)
- Disposer, si besoin, des binaires pour l'installation des *griffins*

3.2.3.3 Présence d'un agent antiviral

Dans le cas de partitions sur lesquelles un agent antiviral est déjà configuré (typiquement, *golden image*), il est recommandé de positionner une exception sur l'arborescence `/vitam` et les sous-arborescences, hormis la partition hébergeant le composant `ingest-exteral` (emploi d'un agent antiviral en prérequis des *ingest* ; se reporter à *Rétention liée aux logback* (page 49)).

3.2.4 Matériel

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

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

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

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

3.2.5 Librairie de cartouches pour offre froide

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

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

3.3 Questions préparatoires

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

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

- **Questions techniques :**
    - Topologie de déploiement et dimensionnement de l'environnement ?
    - Espace de stockage (volumétrie métier cible, technologies d'offres de stockage, nombre d'offres, etc.) ?
    - Sécurisation des flux http (récupération des clés publiques des servcies versants, sécurisation des flux d'accès aux offres, etc.) ?
- **Questions liées au métier :**
    - Nombre de tenants souhaités (hormis les tenant 0 et 1 qui font respectivement office de tenant « blanc » et de tenant d'administration) ?
    - Niveau de classification (la plate-forme est-elle « Secret Défense » ?)
    - Modalités d'indexation des règles de gestion des unités archivistiques (autrement dit, sur quels tenant le recalcul des inheritedRules doit-il être fait complètement / partiellement) ?

- Greffons de préservations (*griffins*) nécessaires ?
- Fréquence de calcul de l'état des fonds symboliques souhaitée ?
- Définition des habilitations (profil de sécurité, contextes applicatifs, . . .) ?
- Modalités de gestion des données de référence (maître/esclave) pour chaque tenant ?

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

3.4 Récupération de la version

3.4.1 Utilisation des dépôts *open-source*

Les scripts de déploiement de la solution logicielle *VITAM* sont disponibles dans le dépôt github VITAM [10] , dans le répertoire `deployment`.

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

3.4.1.1 *Repository* pour environnement CentOS

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

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

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

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

3.4.1.1.1 Cas de *griffins*

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

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

(suite sur la page suivante)

https://github.com/ProgrammeVitam/vitam

(suite de la page précédente)

```
gpgcheck=0
repo_gpgcheck=0
enabled=1
```

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

3.4.1.2 *Repository* pour environnement Debian

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

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

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

3.4.1.2.1 Cas de *griffins*

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

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

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

3.4.2 Utilisation du package global d'installation

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

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

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

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

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

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

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 de la solution logicielle *VITAM* et des composants externes requis pour l'installation. Les autres éléments d'installation (*playbook* ansible, ...) doivent être disponibles sur la machine ansible orchestrant le déploiement de la solution.

4.2 Procédures

4.2.1 Cinématique de déploiement

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

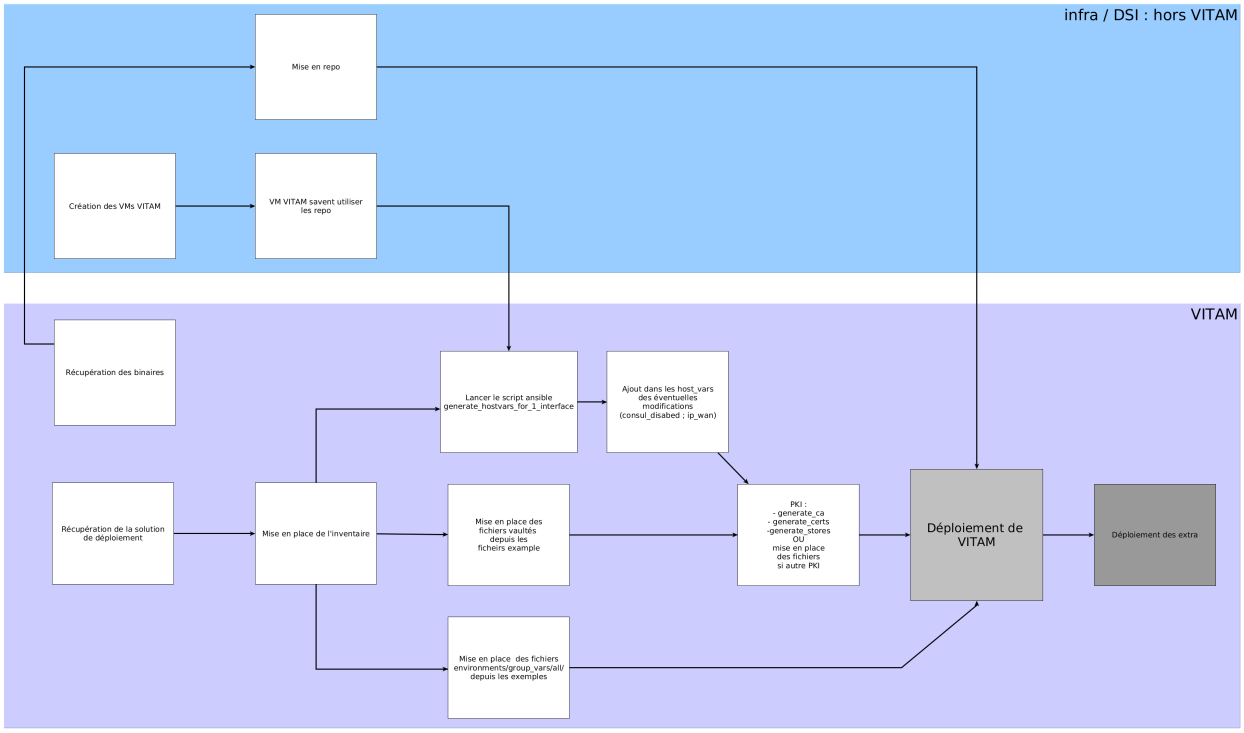

Fig. 1 – Cinématique de déploiement

4.2.2 Cas particulier d'une installation multi-sites

4.2.2.1 Procédure d'installation

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

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

4.2.2.1.1 vitam_site_name

Fichier : `deployment/environments/hosts.<my_env>`

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

4.2.2.1.2 primary_site

Fichier : `deployment/environments/hosts.<my_env>`

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

4.2.2.1.3 consul_remote_sites

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

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

Exemple de configuration pour une installation avec 3 sites.

Site 1 :

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

Site 2 :

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

Site 3 :

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

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

Exemple

```
ip_service: 172.17.0.10
ip_admin: 172.19.0.10
ip_wan: 10.2.64.3
```

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

4.2.2.1.4 vitam_offers

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

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

Exemple :

```
vitam_offers:
    offer-fs-1:
```

(suite sur la page suivante)

(suite de la page précédente)

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

4.2.2.1.5 vitam_strategy

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

Cette variable référence la stratégie de stockage de plateforme *default* sur le site courant. Si l'offre se situe sur un site distant, il est nécessaire de préciser le nom du site sur lequel elle se trouve comme dans l'exemple ci-dessous. Il est fortement conseillé de prendre comme offre référente une des offres locale au site. Les sites secondaires doivent uniquement écrire sur leur(s) offre(s) locale(s).

Exemple pour le site 1 (site primaire) :

```
vitam_strategy:
    - name: offer-fs-1
      referent: true
      # status: INACTIVE (valeur par défaut: ACTIVE)
      # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
      # asyncRead: true # ONLY ENABLE WHEN tape-library
      # vitam_site_name: prod-dc2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
    - name: offer-fs-2
      referent: false
      vitam_site_name: site2
      # status: INACTIVE (valeur par défaut: ACTIVE)
      # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
      # asyncRead: true # ONLY ENABLE WHEN tape-library
      # vitam_site_name: prod-dc2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
    - name: offer-fs-3
      referent: false
      vitam_site_name: site3
      # status: INACTIVE (valeur par défaut: ACTIVE)
      # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
      # asyncRead: true # ONLY ENABLE WHEN tape-library
      # vitam_site_name: prod-dc2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
```

Exemple pour le site 2 (site secondaire) :

```
vitam_strategy:
    - name: offer-fs-2
      referent: true
      # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
      # status: INACTIVE (valeur par défaut: ACTIVE)
      # asyncRead: true # ONLY ENABLE WHEN tape-library
      # vitam_site_name: prod-dc2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
```

Exemple pour le site 3 (site secondaire) :

```
vitam_strategy:
    - name: offer-fs-3
      referent: true
      # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
      # status: INACTIVE (valeur par défaut: ACTIVE)
      # asyncRead: true # ONLY ENABLE WHEN tape-library
      # vitam_site_name: prod-dc2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
```

4.2.2.1.6 other_strategies

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

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

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

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

Exemple pour le site 1 (site primaire) :

```
other_strategoies:
    metadata:
        - name: offer-fs-1
        referent: false
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site1 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
        referent: false
        - name: offer-fs-2
        vitam_site_name: site2
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        - name: offer-fs-3
        referent: false
        vitam_site_name: site3
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
```

(suite sur la page suivante)

(suite de la page précédente)

```
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        - name: offer-s3-1
        referent: false
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site1 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
        - name: offer-s3-2
        referent: false
        vitam_site_name: site2
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        - name: offer-s3-3
        referent: false
        vitam_site_name: site3
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
    binary:
        - name: offer-s3-1
        referent: false
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site1 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
        - name: offer-s3-2
        referent: false
        vitam_site_name: site2
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        - name: offer-s3-3
        referent: false
        vitam_site_name: site3
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
```

Exemple pour le site 2 (site secondaire) :

```
other_strategies:
    metadata:
        - name: offer-fs-2
        referent: false
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
        - name: offer-s3-2
        referent: false
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
```

(suite sur la page suivante)

(suite de la page précédente)

```
        # vitam_site_name: site2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
    binary:
        - name: offer-s3-2
        referent: false
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site2 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
```

Exemple pour le site 3 (site secondaire) :

```
other_strategies:
    metadata:
        - name: offer-fs-3
        referent: false
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site3 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
        - name: offer-s3-3
        referent: false
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site3 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
    binary:
        - name: offer-s3-3
        referent: false
        # status: INACTIVE (valeur par défaut: ACTIVE)
        # id: idoffre # OPTIONAL, if used, MUST BE UNIQUE & same on each site
        # asyncRead: true # ONLY ENABLE WHEN tape-library
        # vitam_site_name: site3 # OPtional, needed only when call to distant site␣
↪(indicate distant vitam_site_name)
```

4.2.2.1.7 plateforme_secret

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

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

4.2.2.1.8 consul_encrypt

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

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

4.2.2.2 Procédure de réinstallation

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

Ensuite :

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

4.2.2.3 Flux entre Storage et Offer

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

Il convient également également d'ajouter :

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

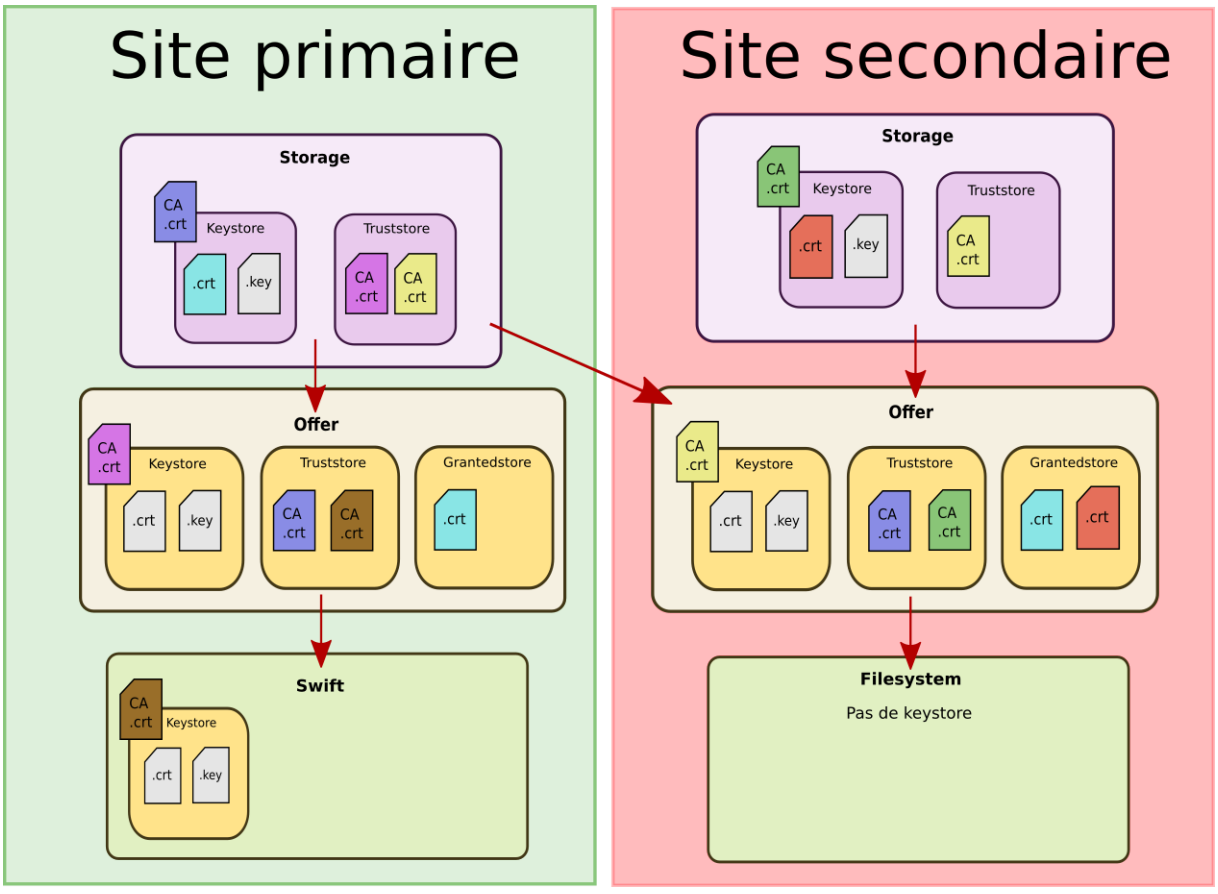

Fig. 2 – Vue détaillée des certificats entre le storage et l'offre en multi-site

Il est possible de procéder de 2 manières différentes :

4.2.2.3.1 Avant la génération des keystores

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

Déposer les *CA* du client storage du site 1 `environments/certs/client-storage/ca/*` dans le client storage du site 2 `environments/certs/client-storage/ca/`.

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

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

4.2.2.3.2 Après la génération des keystores

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

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

Ajout d'une *CA* : `keytool -import -trustcacerts -keystore -file <ca.crt> -alias <alias_certificat>`

4.2.3 Configuration du déploiement

Voir aussi :

L'architecture de la solution logicielle, les éléments de dimensionnement ainsi que les principes de déploiement sont définis dans le *DAT*.

4.2.3.1 Fichiers de déploiement

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

- les *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'exemples sont disponibles dans le sous-répertoire `environments`. Cette arborescence est valable pour le déploiement d'un environnement, et doit être dupliquée lors de l'installation d'environnements ultérieurs. Les fichiers contenus dans cette arborescence doivent être adaptés avant le déploiement, comme expliqué dans les paragraphes suivants.

4.2.3.2 Informations *plate-forme*

4.2.3.2.1 Inventaire

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

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

# Group definition ; DO NOT MODIFY
[hosts:children]
vitam
reverse
hosts_dev_tools
ldap


########### 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 ###########

[ldap] # Extra : OpenLDAP server
# LDAP server !!! NOT FOR PRODUCTION !!! Test only

[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
library
```

(suite sur la page suivante)

(suite de la page précédente)

```
46
47
48  ##### Zone externe
49
50
51  [zone_external:children]
52  hosts_ihm_demo
53  hosts_ihm_recette
54
55
56  [hosts_ihm_demo]
57  # TODO: Put here servers where this service will be deployed : ihm-demo. If you own␣
    ↪another frontend, it is recommended to leave this group blank
58  # If you don't need consul for ihm-demo, you can set this var after each hostname :
59  # consul_disabled=true
60
61  [hosts_ihm_recette]
62  # TODO: Put here servers where this service will be deployed : ihm-recette (extra␣
    ↪feature)
63
64
65  ##### Zone access
66
67  # Group definition ; DO NOT MODIFY
68  [zone_access:children]
69  hosts_ingest_external
70  hosts_access_external
71
72  [hosts_ingest_external]
73  # TODO: Put here servers where this service will be deployed : ingest-external
74
75
76  [hosts_access_external]
77  # TODO: Put here servers where this service will be deployed : access-external
78
79
80  ##### Zone applicative
81
82  # Group definition ; DO NOT MODIFY
83  [zone_applicative:children]
84  hosts_ingest_internal
85  hosts_processing
86  hosts_batch_report
87  hosts_worker
88  hosts_access_internal
89  hosts_metadata
90  hosts_functional_administration
91  hosts_logbook
92  hosts_workspace
93  hosts_storage_engine
94  hosts_security_internal
95
96  [hosts_security_internal]
97  # TODO: Put here servers where this service will be deployed : security-internal
98
99
100 [hosts_logbook]
```

(suite sur la page suivante)

(suite de la page précédente)

```
# TODO: Put here servers where this service will be deployed : logbook


[hosts_workspace]
# TODO: Put the server where this service will be deployed : workspace
# WARNING: put only one server for this service, not more !


[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 the server where this service will be deployed : processing
# WARNING: put only one server for this service, not more !


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

[hosts_batch_report]
# TODO: Put here servers where this service will be deployed : batch-report

[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 ansible_processor_
↪vcpus value (cpu number in /proc/cpuinfo file)


##### Zone storage

[zone_storage:children] # DO NOT MODIFY
hosts_storage_offer_default
hosts_mongodb_offer

[hosts_storage_offer_default]
# TODO: Put here servers where this service will be deployed : storage-offer-default
# LIMIT : only 1 offer per machine
# LIMIT and 1 machine per offer when filesystem or filesystem-hash provider
# Possibility to declare multiple machines with same provider only when provider is␣
↪s3 or swift.
# Mandatory param for each offer is offer_conf and points to offer_opts.yml & vault-
↪vitam.yml (with same tree)
```

(suite sur la page suivante)

(suite de la page précédente)

```
# for swift
# hostname-offre-1.vitam offer_conf=offer-swift-1
# hostname-offre-2.vitam offer_conf=offer-swift-1
# for filesystem
# hostname-offre-2.vitam offer_conf=offer-fs-1
# for s3
# hostname-offre-3.vitam offer_conf=offer-s3-1
# hostname-offre-4.vitam offer_conf=offer-s3-1

[hosts_mongodb_offer:children]
hosts_mongos_offer
hosts_mongoc_offer
hosts_mongod_offer

[hosts_mongos_offer]
# WARNING : DO NOT COLLOCATE WITH [hosts_mongos_data]
# TODO: put here servers where this service will be deployed : mongos cluster for storage offers
# Mandatory param : mongo_cluster_name : name of the cluster (should exist in the offer_conf configuration)
# The recommended practice is to install the mongos instance on the same servers as the mongoc instances
# Example (for a more complete one, see the one in the group hosts_mongos_data) :
# vitam-mongo-swift-offer-01   mongo_cluster_name=offer-swift-1
# vitam-mongo-swift-offer-02   mongo_cluster_name=offer-swift-1
# vitam-mongo-fs-offer-01      mongo_cluster_name=offer-fs-1
# vitam-mongo-fs-offer-02      mongo_cluster_name=offer-fs-1
# vitam-mongo-s3-offer-01      mongo_cluster_name=offer-s3-1
# vitam-mongo-s3-offer-02      mongo_cluster_name=offer-s3-1

[hosts_mongoc_offer]
# WARNING : DO NOT COLLOCATE WITH [hosts_mongoc_data]
# TODO: put here servers where this service will be deployed : mongoc cluster for storage offers
# Mandatory param : mongo_cluster_name : name of the cluster (should exist in the offer_conf configuration)
# Optional param : mandatory for 1 node of the shard, some init commands will be executed on it
# Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add this paramter on a mongo_rs_bootstrap node
# Recommended practice in production: use 3 instances
# Example :
# vitam-mongo-swift-offer-01   mongo_cluster_name=offer-swift-1                    mongo_rs_bootstrap=true
# vitam-mongo-swift-offer-02   mongo_cluster_name=offer-swift-1
# vitam-swift-offer            mongo_cluster_name=offer-swift-1                    mongo_arbiter=true
# vitam-mongo-fs-offer-01      mongo_cluster_name=offer-fs-1                       mongo_rs_bootstrap=true
# vitam-mongo-fs-offer-02      mongo_cluster_name=offer-fs-1
# vitam-fs-offer               mongo_cluster_name=offer-fs-1                       mongo_arbiter=true
# vitam-mongo-s3-offer-01      mongo_cluster_name=offer-s3-1                       mongo_rs_bootstrap=true
# vitam-mongo-s3-offer-02      mongo_cluster_name=offer-s3-1
# vitam-s3-offer               mongo_cluster_name=offer-s3-1                       mongo_arbiter=true
```

(suite sur la page suivante)

(suite de la page précédente)

```
197
198 [hosts_mongod_offer]
199 # WARNING : DO NOT COLLOCATE WITH [hosts_mongod_data]
200 # TODO: put here servers where this service will be deployed : mongod cluster for␣
    ↪storage offers
201 # Mandatory param : mongo_cluster_name : name of the cluster (should exist in the␣
    ↪offer_conf configuration)
202 # Mandatory param : id of the current shard, increment by 1 from 0 to n
203 # Optional param : mandatory for 1 node of the shard, some init commands will be␣
    ↪executed on it
204 # Optional param : mongo_arbiter=true : the node will be only an arbiter ; do not add␣
    ↪this paramter on a mongo_rs_bootstrap node
205 # Optional param : mongod_memory=x ; this will force the wiredtiger cache size to x␣
    ↪(unit is GB) ; can be usefull when colocalization with elasticsearch
206 # Recommended practice in production: use 3 instances per shard
207 # Example :
208 # vitam-mongo-swift-offer-01   mongo_cluster_name=offer-swift-1    mongo_shard_id=0  ␣
    ↪                mongo_rs_bootstrap=true
209 # vitam-mongo-swift-offer-02   mongo_cluster_name=offer-swift-1    mongo_shard_id=0
210 # vitam-swift-offer            mongo_cluster_name=offer-swift-1    mongo_shard_id=0  ␣
    ↪                mongo_arbiter=true
211 # vitam-mongo-fs-offer-01      mongo_cluster_name=offer-fs-1       mongo_shard_id=0  ␣
    ↪                mongo_rs_bootstrap=true
212 # vitam-mongo-fs-offer-02      mongo_cluster_name=offer-fs-1       mongo_shard_id=0
213 # vitam-fs-offer               mongo_cluster_name=offer-fs-1       mongo_shard_id=0  ␣
    ↪                mongo_arbiter=true
214 # vitam-mongo-s3-offer-01      mongo_cluster_name=offer-s3-1       mongo_shard_id=0  ␣
    ↪                mongo_rs_bootstrap=true
215 # vitam-mongo-s3-offer-02      mongo_cluster_name=offer-s3-1       mongo_shard_id=0
216 # vitam-s3-offer               mongo_cluster_name=offer-s3-1       mongo_shard_id=0  ␣
    ↪                mongo_arbiter=true
217
218 ##### Zone data
219
220 # Group definition ; DO NOT MODIFY
221 [zone_data:children]
222 hosts_elasticsearch_data
223 hosts_mongodb_data
224
225 [hosts_elasticsearch_data]
226 # TODO: Put here servers where this service will be deployed : elasticsearch-data␣
    ↪cluster
227 # 2 params available for huge environments (parameter to be declared after each␣
    ↪server) :
228 #    is_data=true/false
229 #    is_master=true/false
230 #    other options are not handled yet
231 # defaults are set to true, if undefined. If defined, at least one server MUST be is_
    ↪data=true
232 # Examples :
233 # server1 is_master=true is_data=false
234 # server2 is_master=false is_data=true
235 # More explanation here : https://www.elastic.co/guide/en/elasticsearch/reference/5.6/
    ↪modules-node.html
236
237
238 # Group definition ; DO NOT MODIFY
```

(suite sur la page suivante)

(suite de la page précédente)

```
[hosts_mongodb_data:children]
hosts_mongos_data
hosts_mongoc_data
hosts_mongod_data

[hosts_mongos_data]
# WARNING : DO NOT COLLOCATE WITH [hosts_mongos_offer]
# TODO: Put here servers where this service will be deployed : mongos cluster
# Mandatory param : mongo_cluster_name=mongo-data  ("mongo-data" is mandatory)
# The recommended practice is to install the mongos instance on the same servers as the mongoc instances
# Example :
# vitam-mdbs-01   mongo_cluster_name=mongo-data
# vitam-mdbs-02   mongo_cluster_name=mongo-data
# vitam-mdbs-03   mongo_cluster_name=mongo-data

[hosts_mongoc_data]
# WARNING : DO NOT COLLOCATE WITH [hosts_mongoc_offer]
# TODO: Put here servers where this service will be deployed : mongoc cluster
# Mandatory param : mongo_cluster_name=mongo-data  ("mongo-data" is mandatory)
# Optional param : mandatory for 1 node of the shard, some init commands will be executed on it
# Recommended practice in production: use 3 instances
# Example :
# vitam-mdbc-01   mongo_cluster_name=mongo-data                    mongo_rs_bootstrap=true
# vitam-mdbc-02   mongo_cluster_name=mongo-data
# vitam-mdbc-03   mongo_cluster_name=mongo-data

[hosts_mongod_data]
# WARNING : DO NOT COLLOCATE WITH [hosts_mongod_offer]
# 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
# Mandatory param : mongo_cluster_name=mongo-data ("mongo-data" is mandatory)
# Mandatory param : id of the current shard, increment by 1 from 0 to n
# Optional param : mandatory for 1 node of the shard, some init commands will be executed on it
# Optional param : mongod_memory=x ; this will force the wiredtiger cache size to x (unit is GB) ; can be usefull when colocalization with elasticsearch
# Recommended practice in production: use 3 instances per shard
# Example:
# vitam-mdbd-01  mongo_cluster_name=mongo-data   mongo_shard_id=0  mongo_rs_bootstrap=true
# vitam-mdbd-02  mongo_cluster_name=mongo-data   mongo_shard_id=0
# vitam-mdbd-03  mongo_cluster_name=mongo-data   mongo_shard_id=0
# vitam-mdbd-04  mongo_cluster_name=mongo-data   mongo_shard_id=1  mongo_rs_bootstrap=true
# vitam-mdbd-05  mongo_cluster_name=mongo-data   mongo_shard_id=1
# vitam-mdbd-06  mongo_cluster_name=mongo-data   mongo_shard_id=1

###### Zone admin

# Group definition ; DO NOT MODIFY
[zone_admin:children]
hosts_cerebro
hosts_consul_server
```

(suite sur la page suivante)

(suite de la page précédente)

```
hosts_kibana_data
log_servers
hosts_elasticsearch_log

[hosts_cerebro]
# TODO: Put here servers where this service will be deployed : vitam-elasticsearch-cerebro

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

[hosts_logstash]
# TODO: Put here servers where this service will be deployed : logstash
# IF you connect VITAM to external SIEM, DO NOT FILL THE SECTION


[hosts_elasticsearch_log]
# TODO: Put here servers where this service will be deployed : elasticsearch-log cluster
# IF you connect VITAM to external SIEM, DO NOT FILL THE SECTION

########### 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
# How can ansible switch to root ?
# See https://docs.ansible.com/ansible/latest/user_guide/become.html

# Related to Consul ; apply in a table your DNS server(s)
# Example : dns_servers=["8.8.8.8","8.8.4.4"]
# If no recursors, use : dns_servers=
dns_servers=

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

(suite sur la page suivante)

(suite de la page précédente)

```
vitam_tenant_admin=1

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

# Define local Consul datacenter name
# CAUTION !!! Only alphanumeric characters when using s3 as offer backend !!!
vitam_site_name=prod-dc1
# On offer, value is the prefix for all containers' names. If upgrading from R8, you MUST UNCOMMENT this parameter AS IS !!!
#vitam_prefix_offer=""
# EXAMPLE : vitam_site_name = prod-dc1
# check whether on primary site (true) or secondary (false)
primary_site=true


# ================================
# EXTRA
# ================================
# Environment (defines title in extra on reverse homepage). Variable is DEPRECATED !
#environnement=

### 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=443
vitam_reverse_external_protocol=https
# http_proxy env var to use ; has to be declared even if empty
http_proxy_environnement=
```

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

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

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

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

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

impliqué, il faut rajouter la directive suivante : `consul_disabled=true`.

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

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

4.2.3.2.2 Fichier `vitam_security.yml`

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

```
---

hide_passwords_during_deploy: true

### Admin context name and tenants ###
admin_context_name: "admin-context"
admin_context_tenants: "{{ vitam_tenant_ids }}"
# Indicate context certificates relative paths under {{ inventory_dir }}/certs/client-external/clients
# vitam-admin-int is mandatory for internal use (PRONOM upload)
admin_context_certs: [ "ihm-demo/ihm-demo.crt", "ihm-recette/ihm-recette.crt", "reverse/reverse.crt", "vitam-admin-int/vitam-admin-int.crt" ]
# Indicate here all the personal certificates relative paths under {{ inventory_dir }}/certs/client-vitam-users/clients
admin_personal_certs: [ "userOK.crt" ]

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

admin_basic_auth_user: "adminUser"
```

Note : Pour la directive `admin_context_certs` concernant l'intégration de certificats *SIA* au déploiement, se reporter à la section *Intégration d'une application externe (cliente)* (page 47).

Note : Pour la directive `admin_personal_certs` concernant l'intégration de certificats personnels (*personae*) au déploiement, se reporter à la section *Intégration d'un certificat personnel (personae)* (page 47).

4.2.3.2.3 Fichier `offers_opts.yml`

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

La déclaration de configuration des offres de stockage associées se fait dans le fichier `environments / group_vars/all/offers_opts.yml` :

```
# This is the default vitam strategy ('default'). It is mandatory and must define a referent offer.
# This list of offers is ordered. It can and has to be completed if more offers are necessary
# Strategy order (1st has to be the prefered one)
vitam_strategy:
  - name: offer-fs-1
    referent: true
#    status: ACTIVE # status : enable (value=ACTIVE, default value) or disable (value=INACTIVE) this offer
#    vitam_site_name: prod-dc2 # optional, should be related to vitam_site_name if local ; remote vitam_site_name if distant
#  - name: offer-swift-1
# Example distant:
#  - name: distant
#    referent: true
#    status: INACTIVE
#    vitam_site_name: distant-dc2
#    distant: true # Only add this parameter when distant offer (not on same platform)

# WARNING : multi-strategy is a BETA functionality
# More strategies can be added but are optional
# Strategy name must only use [a-z][a-z0-9-]* pattern
# Any strategy must contain at least one offer
# This list of offers is ordered. It can and has to be completed if more offers are necessary
# Offers can't be defined as referent other_strategies
# other_strategies:
#  metadata:
#    - name: offer-fs-1
#      referent: false
#    - name: offer-fs-2
#      referent: false
#  binary:
#    - name: offer-fs-2
#      referent: false
#    - name: offer-s3-1
#      referent: false

# DON'T forget to add associated passwords in vault-vitam.yml with same tree when using provider openstack-swift*
# ATTENTION !!! Each offer has to have a distinct name, except for clusters binding a same physical storage
# WARNING : for offer names, please only use [a-z][a-z0-9-]* pattern
vitam_offers:
  offer-tape-1:
    provider: tape-library
    tapeLibraryConfiguration:
      maxTarEntrySize: 100000
      maxTarFileSize: 1000000
      # Enable overriding non empty cartridges
      # WARNING : FOR DEV/TEST ONLY. DO NOT ENABLE IN PRODUCTION.
      forceOverrideNonEmptyCartridges: false
      # Archive (Tar) file expire time for retention in local FS
```

(suite sur la page suivante)

(suite de la page précédente)

```
      archiveRetentionCacheTimeoutInMinutes: 30

      useSudo: false
    topology:
      buckets:
        -
          name: test
          tenants: [0]
          tarBufferingTimeoutInMinutes: 60
        -
          name: admin
          tenants: [1]
          tarBufferingTimeoutInMinutes: 60
        -
          name: prod
          tenants: [2,3,4,5,6,7,8,9]
          tarBufferingTimeoutInMinutes: 60
    tapeLibraries:
      -
        name: TAPE_LIB_1
        robots:
          -
            device: /dev/tape/by-id/scsi-1QUANTUM_10F73224E6664C84A1D00000
            mtxPath: "/usr/sbin/mtx"
            timeoutInMilliseconds: 3600000
        drives:
          -
            index: 0
            device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_1235308739-nst
            mtPath: "/bin/mt"
            ddPath: "/bin/dd"
            tarPath: "/bin/tar"
            timeoutInMilliseconds: 3600000
            readWritePriority: BACKUP
          -
            index: 1
            device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0951859786-nst
            mtPath: "/bin/mt"
            ddPath: "/bin/dd"
            tarPath: "/bin/tar"
            timeoutInMilliseconds: 3600000
            readWritePriority: READ
          -
            index: 2
            device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0269493808-nst
            mtPath: "/bin/mt"
            ddPath: "/bin/dd"
            tarPath: "/bin/tar"
            timeoutInMilliseconds: 3600000
          -
            index: 3
            device: /dev/tape/by-id/scsi-1IBM_ULT3580-TD6_0566471858-nst
            mtPath: "/bin/mt"
            ddPath: "/bin/dd"
            tarPath: "/bin/tar"
            readWritePriority: READ
            timeoutInMilliseconds: 3600000
```

(suite sur la page suivante)

(suite de la page précédente)

```
  offer-fs-1:
    # param can be filesystem-hash (recomended) or filesystem (not recomended)
    provider: filesystem-hash
  offer-swift-1:
    # provider : openstack-swift for v1 or openstack-swift-v3 for v3
    provider: openstack-swift-v3
    # swiftKeystoneAuthUrl : URL de connexion à keystone
    swiftKeystoneAuthUrl: https://openstack-hostname:port/auth/1.0
    # swiftDomain : domaine OpenStack dans lequel l'utilisateur est enregistré
    swiftDomain: domaine
    # swiftUser : identifiant de l'utilisateur
    swiftUser: utilisateur
    # swiftPassword: has to be set in vault-vitam.yml (encrypted) with same structure => DO NOT COMMENT OUT
    # swiftProjectName : nom du projet openstack
    swiftProjectName: monTenant
    # swiftUrl: optional variable to force the swift URL
    # swiftUrl: https://swift-hostname:port/swift/v1
    #SSL TrustStore
    swiftTrustStore: /chemin_vers_mon_fichier/monSwiftTrustStore.jks
    #Max connection (concurrent connections), per route, to keep in pool (if a pooling ConnectionManager is used) (by default 2 for Apache HttpClient)
    swiftMaxConnectionsPerRoute: 200
    #Max total connection (concurrent connections) to keep in pool (if a pooling ConnectionManager is used) (by default 20 for Apache HttpClient)
    swiftMaxConnections: 1000
    #Max time (in milliseconds) for waiting to establish connection
    swiftConnectionTimeout: 200000
    #Max time (in milliseconds) waiting for a data from the server (socket)
    swiftReadTimeout: 60000
    #Time (in seconds) to renew a token before expiration occurs (blocking)
    swiftHardRenewTokenDelayBeforeExpireTime: 60
  offer-s3-1:
    # provider : can only be amazon-s3-v1 for Amazon SDK S3 V1
    provider: 'amazon-s3-v1'
    # s3Endpoint :  : URL of connection to S3
    s3Endpoint: https://s3.domain/
    # s3RegionName (optional):  Region name (default value us-east-1)
    s3RegionName: us-east-1
    # s3SignerType (optional):  Signing algorithm.
    #     - signature V4 : 'AWSS3V4SignerType' (default value)
    #     - signature V2 : 'S3SignerType'
    s3SignerType: AWSS3V4SignerType
    # s3PathStyleAccessEnabled (optional):  'true' to access bucket in "path-style", else "virtual-hosted-style" (false by default in java client, true by default in ansible scripts)
    s3PathStyleAccessEnabled: true
    # s3MaxConnections (optional): Max total connection (concurrent connections) (50 by default)
    s3MaxConnections: 50
    # s3ConnectionTimeout (optional): Max time (in milliseconds) for waiting to establish connection (10000 by default)
    s3ConnectionTimeout: 10000
    # s3SocketTimeout (optional): Max time (in milliseconds) for reading from a connected socket (50000 by default)
```

(suite sur la page suivante)

(suite de la page précédente)

```
    s3SocketTimeout: 50000
    # s3RequestTimeout (optional): Max time (in milliseconds) for a request (0 by default, disabled)
    s3RequestTimeout: 0
    # s3ClientExecutionTimeout (optional): Max time (in milliseconds) for a request by java client (0 by default, disabled)
    s3ClientExecutionTimeout: 0

    #Time (in seconds) to renew a token before expiration occurs
    swiftSoftRenewTokenDelayBeforeExpireTime: 300
  # example_swift_v1:
  #    provider: openstack-swift
  #    swiftKeystoneAuthUrl: https://keystone/auth/1.0
  #    swiftDomain: domain
  #    swiftUser: user
  #    swiftPassword: has to be set in vault-vitam.yml (encrypted) with same structure => DO NOT COMMENT OUT
  # example_swift_v3:
  #    provider: openstack-swift-v3
  #    swiftKeystoneAuthUrl: https://keystone/v3
  #    swiftDomain: domaine
  #    swiftUser: user
  #    swiftPassword: has to be set in vault-vitam.yml (encrypted) with same structure => DO NOT COMMENT OUT
  #    swiftProjectName: monTenant
  #    projectName: monTenant
  # swiftTrustStore: /chemin_vers_mon_fichier/monSwiftTrustStore.jks
  # swiftMaxConnectionsPerRoute: 200
  # swiftMaxConnections: 1000
  # swiftConnectionTimeout: 200000
  # swiftReadTimeout: 60000
  # Time (in seconds) to renew a token before expiration occurs
  # swiftHardRenewTokenDelayBeforeExpireTime: 60
  # swiftSoftRenewTokenDelayBeforeExpireTime: 300
```

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

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

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

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

4.2.3.2.4 Fichier `cots_vars.yml`

Fichier le fichier `environments/group_vars/all/cots_vars.yml` :

```
---

consul:
    dns_port: 53
    retry_interval: 10 # in seconds
    check_internal: 10 # in seconds
    check_timeout: 5 # in seconds
    network: "ip_admin" # Which network to use for consul communications ? ip_admin or ip_service ?

consul_remote_sites:
    # wan contains the wan addresses of the consul server instances of the external vitam sites
    # Exemple, if our local dc is dc1, we will need to set dc2 & dc3 wan conf:
    # - dc2:
    #   wan: ["10.10.10.10","1.1.1.1"]
    # - dc3:
    #   wan: ["10.10.10.11","1.1.1.1"]
# Please uncomment and fill values if you want to connect VITAM to external SIEM
# external_siem:
#     host:
#     port:

elasticsearch:
    log:
        host: "elasticsearch-log.service.{{ consul_domain }}"
        port_http: "9201"
        port_tcp: "9301"
        groupe: "log"
        baseuri: "elasticsearch-log"
        cluster_name: "elasticsearch-log"
        consul_check_http: 10 # in seconds
        consul_check_tcp: 10 # in seconds
        action_log_level: error
        https_enabled: false
        indices_fielddata_cache_size: 0.3 # related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/modules-fielddata.html
        indices_breaker_fielddata_limit: 0.4 # related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/circuit-breaker.html#fielddata-circuit-breaker
        # default index template
        index_templates:
            default:
                shards: 1
                replica: 1
            packetbeat:
                shards: 5
        log_appenders:
            root:
                log_level: "info"
            rolling:
```

(suite sur la page suivante)

(suite de la page précédente)

```
                max_log_file_size: "100MB"
                max_total_log_size: "5GB"
            deprecation_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_search_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_indexing_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
    data:
        host: "elasticsearch-data.service.{{ consul_domain }}"
        # default is 0.1 (10%) and should be quite enough in most cases
        #index_buffer_size_ratio: "0.15"
        port_http: "9200"
        port_tcp: "9300"
        groupe: "data"
        baseuri: "elasticsearch-data"
        cluster_name: "elasticsearch-data"
        consul_check_http: 10 # in seconds
        consul_check_tcp: 10 # in seconds
        action_log_level: debug
        https_enabled: false
        indices_fielddata_cache_size: 0.3 # related to https://www.elastic.
↪co/guide/en/elasticsearch/reference/6.8/modules-fielddata.html
        indices_breaker_fielddata_limit: 0.4 # related to https://www.
↪elastic.co/guide/en/elasticsearch/reference/6.8/circuit-breaker.html
↪#fielddata-circuit-breaker
        # default index template
        index_templates:
            default:
                shards: 10
                replica: 2
        log_appenders:
            root:
                log_level: "info"
            rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "5GB"
            deprecation_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_search_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_indexing_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"

mongodb:
```

(suite sur la page suivante)

(suite de la page précédente)

```
    mongos_port: 27017
    mongoc_port: 27018
    mongod_port: 27019
    mongo_authentication: "true"
    host: "mongos.service.{{ consul_domain }}"
    check_consul: 10 # in seconds
    drop_info_log: false # Drop mongo (I)nformational log, for Verbosity␣
↪Level of 0

logstash:
    host: "logstash.service.{{ consul_domain }}"
    user: logstash
    port: 10514
    rest_port: 20514
    check_consul: 10 # in seconds
    # logstash xms & xmx in Megabytes
    # jvm_xms: 2048
    # jvm_xmx: 2048
    # workers_number: 4
    log_appenders:
        rolling:
            max_log_file_size: "100MB"
            max_total_log_size: "5GB"
        json_rolling:
            max_log_file_size: "100MB"
            max_total_log_size: "5GB"

# Curator units: days
curator:
    log:
        metrics:
            close: 5
            delete: 30
        logstash:
            close: 5
            delete: 30
        metricbeat:
            close: 5
            delete: 30
        packetbeat:
            close: 5
            delete: 30

kibana:
    header_value: "reporting"
    import_delay: 10
    import_retries: 10
    log:
        baseuri: "kibana_log"
        api_call_timeout: 120
        groupe: "log"
        port: 5601
        default_index_pattern: "logstash-vitam*"
        check_consul: 10 # in seconds
        # default shards & replica
        shards: 5
        replica: 1
```

(suite sur la page suivante)

(suite de la page précédente)

```
        # pour index logstash-*
        metrics:
            shards: 5
            replica: 1
        # pour index metrics-vitam-*
        logs:
            shards: 5
            replica: 1
        # pour index metricbeat-*
        metricbeat:
            shards: 5 # must be a factor of 30
            replica: 1
    data:
        baseuri: "kibana_data"
        # OMA : bugdette : api_call_timeout is used for retries ; should create a separate variable rather than this one
        api_call_timeout: 120
        groupe: "data"
        port: 5601
        default_index_pattern: "logbookoperation_*"
        check_consul: 10 # in seconds
        # index template for .kibana
        shards: 1
        replica: 1

syslog:
    # value can be syslog-ng or rsyslog
    name: "rsyslog"

cerebro:
    baseuri: "cerebro"
    port: 9000
    check_consul: 10 # in seconds

siegfried:
    port: 19000
    consul_check: 10 # in seconds

clamav:
    port: 3310
    # frequency freshclam for database update per day (from 0 to 24 - 24 meaning hourly check)
    db_update_periodicity: 1

mongo_express:
    baseuri: "mongo-express"

ldap_authentification:
    ldap_protocol: "ldap"
    ldap_server: "{% if groups['ldap']|length > 0 %}{{ groups['ldap']|first }}{% endif %}"
    ldap_port: "389"
    ldap_base: "dc=programmevitam,dc=fr"
    ldap_login: "cn=Manager,dc=programmevitam,dc=fr"
    uid_field: "uid"
    ldap_userDn_Template: "uid={0},ou=people,dc=programmevitam,dc=fr"
    ldap_group_request: "(&(objectClass=groupOfNames)(member={0}))"
```

(suite sur la page suivante)

(suite de la page précédente)

```
211     ldap_admin_group: "cn=admin,ou=groups,dc=programmevitam, dc=fr"
212     ldap_user_group: "cn=user,ou=groups,dc=programmevitam, dc=fr"
213     ldap_guest_group: "cn=guest,ou=groups,dc=programmevitam, dc=fr"
```

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

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

4.2.3.3 Déclaration des secrets

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

4.2.3.3.1 vitam

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

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

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

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

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

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

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

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

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

```
---
# Vitam platform secret key
plateforme_secret: vitamsecret

# The consul key must be 16-bytes, Base64 encoded: https://www.consul.io/docs/agent/encryption.html
# You can generate it with the "consul keygen" command
# Or you can use this script: deployment/pki/scripts/generate_consul_key.sh
consul_encrypt: Biz14ohqN4HtvZmrXp3N4A==

mongodb:
  mongo-data:
    passphrase: changeitkM4L6zBgK527tWBb
    admin:
      user: vitamdb-admin
      password: change_it_1MpG22m2MywvKW5E
    localadmin:
      user: vitamdb-localadmin
      password: change_it_HycFEVD74g397iRe
    system:
      user: vitamdb-system
      password: change_it_HycFEVD74g397iRe
    metadata:
      user: metadata
      password: change_it_37b97KVaDV8YbCwt
    logbook:
      user: logbook
      password: change_it_jVi6q8eX4H1Ce8UC
    report:
      user: report
      password: change_it_jb7TASZbU6n85t8L
    functionalAdmin:
      user: functional-admin
      password: change_it_9eA2zMCL6tm6KF1e
    securityInternal:
      user: security-internal
      password: change_it_m39XvRQWixyDX566
  offer-fs-1:
    passphrase: changeitmB5rnk1M5TY61PqZ
    admin:
      user: vitamdb-admin
      password: change_it_FLkM5emt63N73EcN
    localadmin:
      user: vitamdb-localadmin
      password: change_it_QeH8q4e16ah4QKXS
    system:
      user: vitamdb-system
      password: change_it_HycFEVD74g397iRe
    offer:
      user: offer
      password: change_it_pQi1T1yT9LAF8au8
  offer-fs-2:
    passphrase: changeiteSY1By57qZr4MX2s
    admin:
      user: vitamdb-admin
      password: change_it_84aTMFZ7h8e2NgMe
```

(suite sur la page suivante)

(suite de la page précédente)

```
    localadmin:
      user: vitamdb-localadmin
      password: change_it_Am1B37tGY1w5VfvX
    system:
      user: vitamdb-system
      password: change_it_HycFEVD74g397iRe
    offer:
      user: offer
      password: change_it_mLDYds957sNQ53mA
  offer-tape-1:
    passphrase: changeitmB5rnk1M5TY61PqZ
    admin:
      user: vitamdb-admin
      password: change_it_FLkM5emt63N73EcN
    localadmin:
      user: vitamdb-localadmin
      password: change_it_QeH8q4e16ah4QKXS
    system:
      user: vitamdb-system
      password: change_it_HycFEVD74g397iRe
    offer:
      user: offer
      password: change_it_pQi1T1yT9LAF8au8
  offer-swift-1:
    passphrase: changeitgYvt42M2pKL6Zx3T
    admin:
      user: vitamdb-admin
      password: change_it_e21hLp51WNa4sJFS
    localadmin:
      user: vitamdb-localadmin
      password: change_it_QB8857SJrGrQh2yu
    system:
      user: vitamdb-system
      password: change_it_HycFEVD74g397iRe
    offer:
      user: offer
      password: change_it_AWJg2Bp3s69P6nMe
  offer-s3-1:
    passphrase: changeituF1jVdR9NqdTG625
    admin:
      user: vitamdb-admin
      password: change_it_5b7cSWcS5M1NF4kv
    localadmin:
      user: vitamdb-localadmin
      password: change_it_S9jE24rxHwUZP6y5
    system:
      user: vitamdb-system
      password: change_it_HycFEVD74g397iRe
    offer:
      user: offer
      password: change_it_TuTB1i2k7iQW3zL2
  offer-tape-1:
    passphrase: changeituF1jghT9NqdTG625
    admin:
      user: vitamdb-admin
      password: change_it_5b7cSWcab91NF4kv
    localadmin:
```

(suite sur la page suivante)

(suite de la page précédente)

```
      user: vitamdb-localadmin
      password: change_it_S9jE24rxHwUZP5a6
    system:
      user: vitamdb-system
      password: change_it_HycFEVD74g397iRe
    offer:
      user: offer
      password: change_it_TuTB1i2k7iQW3c2a

vitam_users:
  - vitam_aadmin:
    login: aadmin
    password: change_it_z5MP7GC4qnR8nL9t
    role: admin
  - vitam_uuser:
    login: uuser
    password: change_it_w94Q3jPAT2aJYm8b
    role: user
  - vitam_gguest:
    login: gguest
    password: change_it_E5v7Tr4h6tYaQG2W
    role: guest
  - techadmin:
    login: techadmin
    password: change_it_K29E1uHcPZ8zXji8
    role: admin

ldap_authentification:
    ldap_pwd: "change_it_t69Rn5NdUv39EYkC"

admin_basic_auth_password: change_it_5Yn74JgXwbQ9KdP8

vitam_offers:
    offer-swift-1:
        swiftPassword: change_it_m44j57aYeRPnPXQ2
    offer-s3-1:
        s3AccessKey: accessKey_change_grLS8372Uga5EJSx
        s3SecretKey: secretKey_change_p97es2m2CHXPJA1m
```

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

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

Note : Dans le cadre d'une installation avec au moins une offre *swift*, il faut déclarer, dans la section `vitam_offers`, le nom de chaque offre et le mot de passe de connexion *swift* associé, défini dans le fichier `offers_opts.yml`.

L'exemple ci-dessus présente la déclaration du mot de passe pour l'offre swift *offer-swift-1*.

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

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

```
# NO UNDERSCORE ALLOWED IN VALUES
keystores:
  server:
    offer: changeit817NR75vWsZtgAgJ
    access_external: changeitMZFD2YM4279miitu
    ingest_external: changeita2C74cQhy84BLWCr
    ihm_recette: changeit4FWYVK1347mxjGfe
    ihm_demo: changeit6kQ16eyDY7QPS9fy
  client_external:
    ihm_demo: changeitGT38hhTiA32x1PLy
    gatling: changeit2sBC5ac7NfGF9Qj7
    ihm_recette: changeitdAZ9Eq65UhDZd9p4
    reverse: changeite5XTzb5yVPcEX464
    vitam_admin_int: changeitz6xZe5gDu7nhDZd9
  client_storage:
    storage: changeit647D7LWiyM6qYMnm
  timestamping:
    secure_logbook: changeitMn9Skuyx87VYU62U
    secure_storage: changeite5gDu9Skuy84BLW9
truststores:
  server: changeitxNe4JLfn528PVHj7
  client_external: changeitJ2eS93DcPH1v4jAp
  client_storage: changeitHpSCa31aG8ttB87S
grantedstores:
  client_external: changeitLL22HkmDCA2e2vj7
  client_storage: changeitR3wwp5C8KQS76Vcu
```

Avertissement : il convient de sécuriser votre environnement en définissant des mots de passe *forts*.

4.2.3.3.2 Cas des extras

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

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

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

une analyse plus fine d'un éventuel problème sur une de ces étapes.

4.2.3.3.3 Commande ansible-vault

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

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

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

Exemple du fichier `vault-cots.example`

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

4.2.3.3.3.2 Ré-encoder un fichier *vaulté*

Exemple du fichier `vault-cots.yml`

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

4.2.4 Gestion des certificats

Une vue d'ensemble de la gestion des certificats est présentée *dans l'annexe dédiée* (page 92).

4.2.4.1 Cas 1 : Configuration développement / tests

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

4.2.4.1.1 Procédure générale

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

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

- Les autorités de 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*

4.2.4.1.2 Génération des CA par les scripts Vitam

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

Dans le répertoire de déploiement, lancer le script :

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

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

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

4.2.4.1.3 Génération des certificats par les scripts Vitam

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

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

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

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

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

4.2.4.2 Cas 2 : Configuration production

4.2.4.2.1 Procédure générale

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

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

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

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

4.2.4.2.2 Génération des certificats

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

4.2.4.2.2.1 Certificats serveurs

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

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

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

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

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

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

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

4.2.4.2.2.2 Certificat clients

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

4.2.4.2.2.3 Certificats d'horodatage

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

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

4.2.4.2.3 Intégration de certificats existants

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

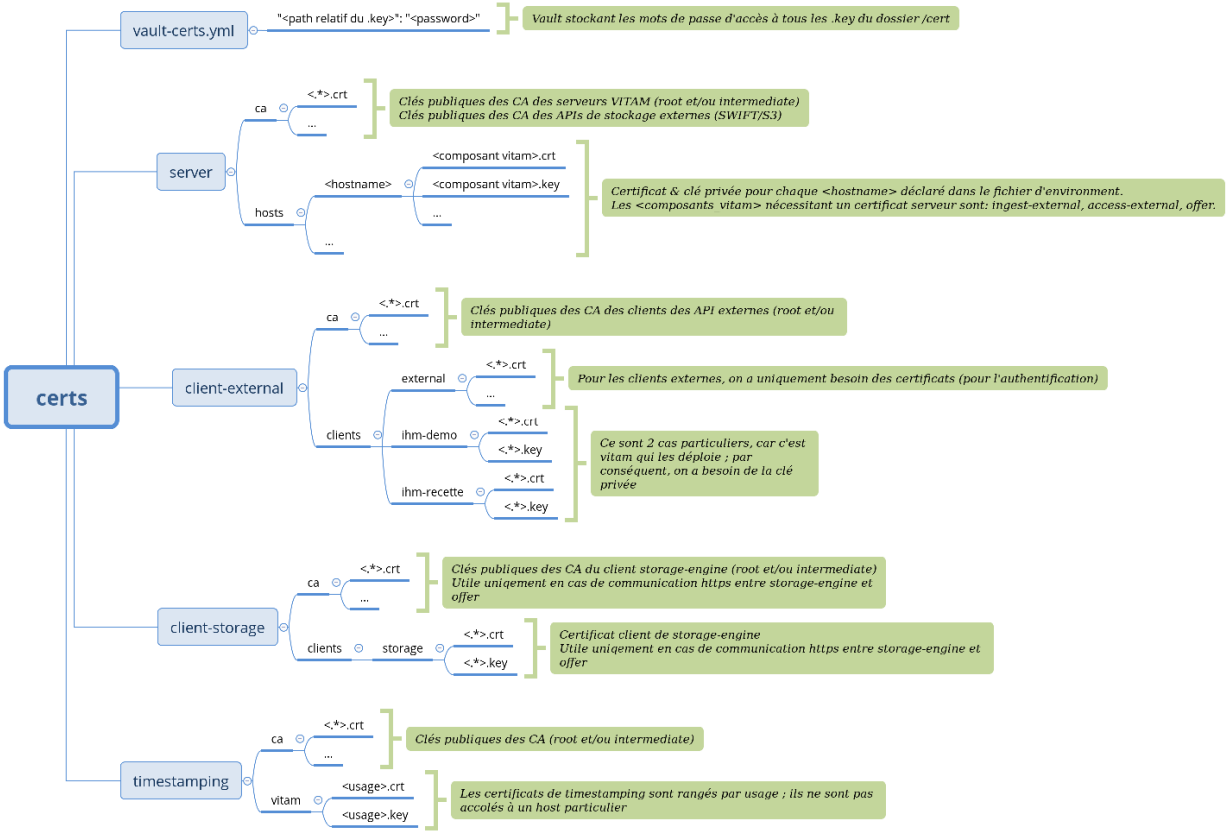

Fig. 3 – Vue détaillée de l'arborescence des certificats

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

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

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

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

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

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

4.2.4.2.4 Intégration de certificats clients de VITAM

4.2.4.2.4.1 Intégration d'une application externe (cliente)

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

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

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

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

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

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

4.2.4.2.5 Cas des offres objet

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

4.2.4.2.6 Absence d'usage d'un *reverse*

Dans ce cas, il convient de :

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

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

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

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

4.2.4.4 Génération des magasins de certificats

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

Prudence : Avant de lancer le script de génération des *stores*, il est nécessaire de modifier le vault contenant les mots de passe des *stores* : `environments/group_vars/all/vault-keystores.yml`, décrit dans la section *Déclaration des secrets* (page 38).

Lancer le script : `./generate_stores.sh`

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

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

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

4.2.5 Paramétrages supplémentaires

4.2.5.1 *Tuning* JVM

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

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

4.2.5.2 Installation des *griffins* (greffons de préservation)

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

Prudence : Cette version de *VITAM* ne mettant pas encore en oeuvre de mesure d'isolation particulière des *griffins*, il est recommandé de veiller à ce que l'usage de chaque *griffin* soit en conformité avec la politique de sécurité de l'entité. Il est en particulier déconseillé d'utiliser un griffon qui utiliserait un outil externe qui n'est plus maintenu.

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

Exemple :

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

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

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

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

4.2.5.3 Rétention liée aux logback

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

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

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

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

4.2.5.3.1 Cas des access_log

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

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

4.2.5.4 Paramétrage de l'antivirus (ingest-externe)

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

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

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

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

4.2.5.5 Paramétrage des certificats externes (*-externe)

Se reporter au chapitre dédié à la gestion des certificats : *Gestion des certificats* (page 43)

4.2.5.6 Placer « hors Vitam » le composant ihm-demo

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

```
consul_disabled: true
```

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

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

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

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

4.2.5.7 Paramétrer le `secure_cookie` pour ihm-demo

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

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

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

4.2.5.8 Paramétrage de la centralisation des logs VITAM

2 cas sont possibles :

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

4.2.5.8.1 Gestion par VITAM

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

- hosts_logstash
- hosts_kibana_log
- hosts_elasticsearch_log

4.2.5.8.2 Redirection des logs sur un SIEM tiers

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

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

---

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

4.2.5.9 Passage des identifiants des référentiels en mode *esclave*

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

Par exemple :

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

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

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

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

Tableau 1: Description des identifiants de référentiels

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

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

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

4.2.5.10 Durées minimales permettant de contrôler les valeurs saisies

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

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

Exemple :

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

Par *tenant*, les directives possibles sont :

Tableau 2: Description des règles

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

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

Exemples :

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

Voir aussi :

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

4.2.5.11 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 :

```
---
### global ###

# Disable epel or Debian backports repositories install
disable_internet_repositories_install: false

# TODO MAYBE : permettre la surcharge avec une syntax du genre vitamopts.folder_root | default(vitam_default.folder_root) dans les templates ?
droid_filename: "DROID_SignatureFile_V95.xml"
droid_container_filename: "container-signature-20180920.xml"

vitam_defaults:
    folder:
        root_path: /vitam
        folder_permission: "0750"
        conf_permission: "0640"
        folder_upload_permission: "0770"
        script_permission: "0750"
    users:
        vitam: "vitam"
        vitamdb: "vitamdb"
        group: "vitam"
    services:
        # Default log level for vitam components: logback values (TRACE, DEBUG, INFO, WARN, ERROR, OFF)
        log_level: WARN
```

(suite sur la page suivante)

(suite de la page précédente)

```
        start_timeout: 300
        stop_timeout: 3600
        port_service_timeout: 86400
        api_call_timeout: 120
        api_long_call_timeout: 300
        status_retries_number: 60
        status_retries_delay: 5
    # 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
    ### Force chunk mode : set true if chunk header should be checked
    vitam_force_chunk_mode: false
    # syslog_facility
    syslog_facility: local0
    # Configuration of log for reconstruction services (INFO or DEBUG for active logs). Logs will be present only on secondary site.
    reconstruction:
        log_level: INFO

# Used in ingest, unitary update, mass-update
classificationList: ["Non protégé","Secret Défense", "Confidentiel Défense"]
# Used in ingest, unitary update, mass-update
classificationLevelOptional: true
# Packages install retries
packages_install_retries_number: 1
packages_install_retries_delay: 10


vitam_timers:
# systemd nomenclature
#    minutely → *-*-* *:*:00
#    hourly → *-*-* *:00:00
#    daily → *-*-* 00:00:00
#    monthly → *-*-01 00:00:00
#    weekly → Mon *-*-* 00:00:00
#    yearly → *-01-01 00:00:00
#    quarterly → *-01,04,07,10-01 00:00:00
#    semiannually → *-01,07-01 00:00:00
    logbook: # all have to run on only one machine
        # Sécurisation des journaux des opérations
        - name: vitam-traceability-operations
          frequency: "*-*-* 0/2:00:00" # each 2 hours
        # Sécurisation des journaux du cycle de vie des groupes d'objets
        - name: vitam-traceability-lfc-objectgroup
          frequency: "*-*-* 0/4:00:00" # each 4 hours
        # Sécurisation des journaux du cycle de vie des unités archivistiques
        - name: vitam-traceability-lfc-unit
          frequency: "*-*-* 0/3:00:00" # each 3 hours
        # Audit de traçabilité
        - name: vitam-traceability-audit
          frequency: "*-*-* 00:00:00"
        # Reconstruction
        - name: vitam-logbook-reconstruction
          frequency: "*-*-* *:0/5:00"
    storage:
```

(suite sur la page suivante)

(suite de la page précédente)

```
        # Sauvegarde des journaux des écritures
        - name: vitam-storage-accesslog-backup
          frequency: "*-*-* 0/4:00:00" # each 4 hours
        # Sécurisation du journal des écritures
        - name: vitam-storage-log-backup
          frequency: "*-*-* 0/2:00:00" # each 2 hours
        # Log traceability
        - name: vitam-storage-log-traceability
          frequency: "*-*-* 0/2:10:00" # each 2 hours (10 minutes)
    functional_administration:
        - name: vitam-create-accession-register-symbolic
          frequency: "*-*-* 00:00:00"
        - name: vitam-functional-administration-accession-register-reconstruction
          frequency: "*-*-* *:0/5:00"
        - name: vitam-rule-management-audit
          frequency: "*-*-* *:00:00"
        - name: vitam-functional-administration-reconstruction
          frequency: "*-*-* *:0/5:00"
    metadata:
        - name: vitam-metadata-store-graph
          frequency: "*-*-* *:0/30:00"
        - name: vitam-metadata-reconstruction
          frequency: "*-*-* *:0/5:00"
        - name: vitam-metadata-computed-inherited-rules
          frequency: "*-*-* 02:30:00"
        - name: vitam-metadata-purge-dip
          frequency: "*-*-* 02:20:00"
        - name: vitam-metadata-purge-transfers-SIP
          frequency: "*-*-* 02:20:00"

### consul ###
# FIXME: Consul à la racine pour le moment à cause de problèmes de récursivité␣
↪dans le parsing yaml
# WARNING: consul_domain should be a supported domain name for your organization
#          You will have to generate server certificates with the same domain␣
↪name and the service subdomain name
#          Example: consul_domain=vitam means you will have to generate some␣
↪certificates with .service.vitam domain
#                   access-external.service.vitam, ingest-external.service.vitam,␣
↪...
consul_domain: consul
consul_component: consul
consul_folder_conf: "{{ vitam_defaults.folder.root_path }}/conf/{{ consul_
↪component }}"

# Workspace should be useless but storage have a dependency to it...
# elastic-kibana-interceptor is present as kibana is present, if kibana-data &␣
↪interceptor are not needed in the secondary site, just do not add them in the␣
↪hosts file
vitam_secondary_site_components: [ "logbook" , "metadata" , "functional-
↪administration" , "storage" , "storageofferdefault" , "offer" , "elasticsearch-
↪log" , "elasticsearch-data" , "logstash" , "kibana" , "mongoc" , "mongod" ,
↪"mongos", "elastic-kibana-interceptor" , "consul" ]

# Vitams griffins required to launch preservation scenario
vitam_griffins: []

```

(suite sur la page suivante)

(suite de la page précédente)

```
### Composants Vitam ###

vitam:
    # Ontology cache settings (max entries in cache & retention timeout in seconds)
    ontologyCacheMaxEntries: 100
    ontologyCacheTimeoutInSeconds: 300
    # Elasticsearch scroll timeout in milliseconds settings
    elasticSearchScrollTimeoutInMilliseconds: 300000
    accessexternal:
        # Component name: do not modify
        vitam_component: access-external
        # DNS record for the service:
        # Modify if ihm-demo is not using consul (typical production deployment)
        host: "access-external.service.{{ consul_domain }}"
        port_admin: 28102
        port_service: 8444
        baseuri: "access-external"
        https_enabled: true
        # Use platform secret for this component ? : do not modify
        secret_platform: "false"
        # Force the log level for this component: this are logback values (TRACE, DEBUG, INFO, WARN, ERROR, OFF)
        # If this var is not set, the default one will be used (vitam_defaults.services.log_level)
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    accessinternal:
        vitam_component: access-internal
        host: "access-internal.service.{{ consul_domain }}"
        port_service: 8101
        port_admin: 28101
        baseuri: "access-internal"
        https_enabled: false
        secret_platform: "true"
        # log_level: "DEBUG"
        metrics_enabled: true
```

(suite sur la page suivante)

(suite de la page précédente)

```
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    functional_administration:
        vitam_component: functional-administration
        host: "functional-administration.service.{{ consul_domain }}"
        port_service: 8004
        port_admin: 18004
        baseuri: "adminmanagement"
        https_enabled: false
        secret_platform: "true"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    elastickibanainterceptor:
        vitam_component: elastic-kibana-interceptor
        host: "elastic-kibana-interceptor.service.{{ consul_domain }}"
        port_service: 8014
        port_admin: 18014
        baseuri: ""
```

(suite sur la page suivante)

(suite de la page précédente)

```
        https_enabled: false
        secret_platform: "false"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    batchreport:
        vitam_component: batch-report
        host: "batch-report.service.{{ consul_domain }}"
        port_service: 8015
        port_admin: 18015
        baseuri: "batchreport"
        https_enabled: false
        secret_platform: "false"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    ingestexternal:
        vitam_component: ingest-external
```

(suite sur la page suivante)

(suite de la page précédente)

```
        # DNS record for the service:
        # Modify if ihm-demo is not using consul (typical production deployment)
        host: "ingest-external.service.{{ consul_domain }}"
        port_admin: 28001
        port_service: 8443
        baseuri: "ingest-external"
        https_enabled: true
        secret_platform: "false"
        antivirus: "clamav"
        # uncomment if huge files need to be analyzed in more than 60s (default behavior)
        #scantimeout: 60000 # value in milliseconds
        # Directory where files should be placed for local ingest
        upload_dir: "/vitam/data/ingest-external/upload"
        # Directory where successful ingested files will be moved to
        success_dir: "/vitam/data/ingest-external/upload/success"
        # Directory where failed ingested files will be moved to
        fail_dir: "/vitam/data/ingest-external/upload/failure"
        # Action done to file after local ingest (see below for further information)
        upload_final_action: "MOVE"
        # log_level: "DEBUG"
        # upload_final_action can be set to three different values (lower or upper case does not matter)
        #   MOVE : After upload, the local file will be moved to either success_dir or fail_dir depending on the status of the ingest towards ingest-internal
        #   DELETE : After upload, the local file will be deleted if the upload succeeded
        #   NONE : After upload, nothing will be done to the local file (default option set if the value entered for upload_final_action does not exist)
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    ingestinternal:
        vitam_component: ingest-internal
        host: "ingest-internal.service.{{ consul_domain }}"
        port_service: 8100
        port_admin: 28100
        baseuri: "ingest"
```

(suite sur la page suivante)

(suite de la page précédente)

```
        https_enabled: false
        secret_platform: "true"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    ihm_demo:
        vitam_component: ihm-demo
        host: "ihm-demo.service.{{ consul_domain }}"
        port_service: 8446
        port_admin: 28002
        baseurl: "/ihm-demo"
        static_content: "{{ vitam_defaults.folder.root_path }}/app/ihm-demo/v2"
        baseuri: "ihm-demo"
        https_enabled: true
        secret_platform: "false"
        # User session timeout in milliseconds (for shiro)
        session_timeout: 1800000
        secure_cookie: true
        # Specify here the realms you want to use for authentication in ihm-demo
        # You can set multiple realms, one per line
        # With multiple realms, the user will be able to choose between the␣
↪allowed realms
        # Example: authentication_realms:
        #               - x509Realm
        #               - ldapRealm
        # Authorized values:
        # x509Realm: certificate
        # iniRealm: ini file
        # ldapRealm: ldap
        authentication_realms:
            # - x509Realm
            - iniRealm
            # - ldapRealm
        # log_level: "DEBUG"
        allowedMediaTypes:
            - type: "application"
              subtype: "pdf"
            - type: "text"
```

(suite sur la page suivante)

(suite de la page précédente)

```
              subtype: "plain"
            - type: "image"
              subtype: "jpeg"
            - type: "image"
              subtype: "tiff"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    logbook:
        vitam_component: logbook
        host: "logbook.service.{{ consul_domain }}"
        port_service: 9002
        port_admin: 29002
        baseuri: "logbook"
        https_enabled: false
        secret_platform: "true"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
        # Temporization delay (in seconds) for recent logbook operation events.
        # Set it to a reasonable delay to cover max clock difference across␣
↪servers + VM/GC pauses
        operationTraceabilityTemporizationDelay: 300
        # Temporization delay (in seconds) for recent logbook lifecycle events.
        # Set it to a reasonable delay to cover max clock difference across␣
↪servers + VM/GC pauses
        lifecycleTraceabilityTemporizationDelay: 300
        # Max entries selected per (Unit or Object Group) LFC traceability␣
↪operation
        lifecycleTraceabilityMaxEntries: 100000
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
```

(suite sur la page suivante)

(suite de la page précédente)

```
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    metadata:
        vitam_component: metadata
        host: "metadata.service.{{ consul_domain }}"
        port_service: 8200
        port_admin: 28200
        baseuri: "metadata"
        https_enabled: false
        secret_platform: "true"
        cluster_name: "{{ elasticsearch.data.cluster_name }}"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        # Archive Unit Profile cache settings (max entries in cache & retention␣
↪timeout in seconds)
        archiveUnitProfileCacheMaxEntries: 100
        archiveUnitProfileCacheTimeoutInSeconds: 300
        # Schema validator cache settings (max entries in cache & retention␣
↪timeout in seconds)
        schemaValidatorCacheMaxEntries: 100
        schemaValidatorCacheTimeoutInSeconds: 300
        acceptableRequestTime: 10 # value in seconds
        # DIP cleanup delay (in minutes)
        dipTimeToLiveInMinutes: 10080 # 7 days
        transfersSIPTimeToLiveInMinutes: 10080 # 7 days
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    processing:
        vitam_component: processing
        host: "processing.service.{{ consul_domain }}"
        port_service: 8203
```

(suite sur la page suivante)

(suite de la page précédente)

```
        port_admin: 28203
        baseuri: "processing"
        https_enabled: false
        secret_platform: "true"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        maxDistributionInMemoryBufferSize: 100000
        maxDistributionOnDiskBufferSize: 100000000
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    security_internal:
        vitam_component: security-internal
        host: "security-internal.service.{{ consul_domain }}"
        port_service: 8005
        port_admin: 28005
        baseuri: "security-internal"
        https_enabled: false
        secret_platform: "true"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
```

(suite sur la page suivante)

(suite de la page précédente)

```
        access_total_size_cap: "14GB" # total acceptable size
    storageengine:
        vitam_component: storage
        host: "storage.service.{{ consul_domain }}"
        port_service: 9102
        port_admin: 29102
        baseuri: "storage"
        https_enabled: false
        secret_platform: "true"
        storageTraceabilityOverlapDelay: 300
        restoreBulkSize: 1000
        # batch thread pool size
        minBatchThreadPoolSize: 4
        maxBatchThreadPoolSize: 32
        # Digest computation timeout in seconds
        batchDigestComputationTimeout: 300
        # Offer synchronization batch size & thread pool size
        offerSynchronizationBulkSize: 1000
        # Retries attempts
        offerSyncNumberOfRetries: 3
        offerSyncFirstAttemptWaitingTime: 15
        offerSyncWaitingTime: 30
        offerSyncThreadPoolSize: 32
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
          offersync:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        # unit time per kB (in ms) used while calculating the timeout of an http request between storage and offer (if the calculated result is less than 60s, this time is used)
        timeoutMsPerKB: 100
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 60 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    storageofferdefault:
        vitam_component: "offer"
        port_service: 9900
        port_admin: 29900
        baseuri: "offer"
```

(suite sur la page suivante)

(suite de la page précédente)

```
        https_enabled: false
        secret_platform: "true"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
          offer_tape:
            history_days: 10
            totalsize: "5GB"
          offer_tape_backup:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 60 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    worker:
        vitam_component: worker
        host: "worker.service.{{ consul_domain }}"
        port_service: 9104
        port_admin: 29104
        baseuri: "worker"
        https_enabled: false
        secret_platform: "true"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 60 # value in seconds
        api_output_index_tenants: [0,1,2,3,4,5,6,7,8,9]
        rules_index_tenants: [0,1,2,3,4,5,6,7,8,9]
```

(suite sur la page suivante)

(suite de la page précédente)

```
        # Archive Unit Profile cache settings (max entries in cache & retention␣
↪timeout in seconds)
        archiveUnitProfileCacheMaxEntries: 100
        archiveUnitProfileCacheTimeoutInSeconds: 300
        # Schema validator cache settings (max entries in cache & retention␣
↪timeout in seconds)
        schemaValidatorCacheMaxEntries: 100
        schemaValidatorCacheTimeoutInSeconds: 300
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    workspace:
        vitam_component: workspace
        host: "workspace.service.{{ consul_domain }}"
        port_service: 8201
        port_admin: 28201
        baseuri: "workspace"
        https_enabled: false
        secret_platform: "true"
        # log_level: "DEBUG"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size

# for functional-administration, manage master/slave tenant configuration
vitam_tenants_usage_external:
  - name: 0
    identifiers:
        - INGEST_CONTRACT
        - ACCESS_CONTRACT
        - MANAGEMENT_CONTRACT
        - ARCHIVE_UNIT_PROFILE
  - name: 1
    identifiers:
      - INGEST_CONTRACT
      - ACCESS_CONTRACT
      - MANAGEMENT_CONTRACT
```

(suite sur la page suivante)

(suite de la page précédente)

```
      - PROFILE
      - SECURITY_PROFILE
      - CONTEXT

vitam_tenant_rule_duration:
  - name: 2 # applied tenant
    rules:
      - AppraisalRule : "1 year" # rule name : rule value

# If you want to deploy vitam in a single VM, add the vm name in this array
single_vm_hostnames: ['localhost']
```

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

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

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

```
---

consul:
    dns_port: 53
    retry_interval: 10 # in seconds
    check_internal: 10 # in seconds
    check_timeout: 5 # in seconds
    network: "ip_admin" # Which network to use for consul communications ? ip_
↪admin or ip_service ?

consul_remote_sites:
    # wan contains the wan addresses of the consul server instances of the␣
↪external vitam sites
    # Exemple, if our local dc is dc1, we will need to set dc2 & dc3 wan conf:
    # - dc2:
    #   wan: ["10.10.10.10","1.1.1.1"]
    # - dc3:
    #   wan: ["10.10.10.11","1.1.1.1"]
# Please uncomment and fill values if you want to connect VITAM to external SIEM
# external_siem:
#     host:
#     port:

elasticsearch:
    log:
        host: "elasticsearch-log.service.{{ consul_domain }}"
        port_http: "9201"
```

(suite sur la page suivante)

(suite de la page précédente)

```
        port_tcp: "9301"
        groupe: "log"
        baseuri: "elasticsearch-log"
        cluster_name: "elasticsearch-log"
        consul_check_http: 10 # in seconds
        consul_check_tcp: 10 # in seconds
        action_log_level: error
        https_enabled: false
        indices_fielddata_cache_size: 0.3 # related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/modules-fielddata.html
        indices_breaker_fielddata_limit: 0.4 # related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/circuit-breaker.html#fielddata-circuit-breaker
        # default index template
        index_templates:
            default:
                shards: 1
                replica: 1
            packetbeat:
                shards: 5
        log_appenders:
            root:
                log_level: "info"
            rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "5GB"
            deprecation_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_search_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_indexing_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
    data:
        host: "elasticsearch-data.service.{{ consul_domain }}"
        # default is 0.1 (10%) and should be quite enough in most cases
        #index_buffer_size_ratio: "0.15"
        port_http: "9200"
        port_tcp: "9300"
        groupe: "data"
        baseuri: "elasticsearch-data"
        cluster_name: "elasticsearch-data"
        consul_check_http: 10 # in seconds
        consul_check_tcp: 10 # in seconds
        action_log_level: debug
        https_enabled: false
        indices_fielddata_cache_size: 0.3 # related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/modules-fielddata.html
        indices_breaker_fielddata_limit: 0.4 # related to https://www.elastic.co/guide/en/elasticsearch/reference/6.8/circuit-breaker.html#fielddata-circuit-breaker
        # default index template
```

(suite sur la page suivante)

(suite de la page précédente)

```
        index_templates:
            default:
                shards: 10
                replica: 2
        log_appenders:
            root:
                log_level: "info"
            rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "5GB"
            deprecation_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_search_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"
            index_indexing_slowlog_rolling:
                max_log_file_size: "100MB"
                max_total_log_size: "1GB"
                log_level: "warn"

mongodb:
    mongos_port: 27017
    mongoc_port: 27018
    mongod_port: 27019
    mongo_authentication: "true"
    host: "mongos.service.{{ consul_domain }}"
    check_consul: 10 # in seconds
    drop_info_log: false # Drop mongo (I)nformational log, for Verbosity Level of 0

logstash:
    host: "logstash.service.{{ consul_domain }}"
    user: logstash
    port: 10514
    rest_port: 20514
    check_consul: 10 # in seconds
    # logstash xms & xmx in Megabytes
    # jvm_xms: 2048
    # jvm_xmx: 2048
    # workers_number: 4
    log_appenders:
        rolling:
            max_log_file_size: "100MB"
            max_total_log_size: "5GB"
        json_rolling:
            max_log_file_size: "100MB"
            max_total_log_size: "5GB"

# Curator units: days
curator:
    log:
        metrics:
            close: 5
            delete: 30
```

(suite sur la page suivante)

(suite de la page précédente)

```
        logstash:
            close: 5
            delete: 30
        metricbeat:
            close: 5
            delete: 30
        packetbeat:
            close: 5
            delete: 30

kibana:
    header_value: "reporting"
    import_delay: 10
    import_retries: 10
    log:
        baseuri: "kibana_log"
        api_call_timeout: 120
        groupe: "log"
        port: 5601
        default_index_pattern: "logstash-vitam*"
        check_consul: 10 # in seconds
        # default shards & replica
        shards: 5
        replica: 1
        # pour index logstash-*
        metrics:
            shards: 5
            replica: 1
        # pour index metrics-vitam-*
        logs:
            shards: 5
            replica: 1
        # pour index metricbeat-*
        metricbeat:
            shards: 5 # must be a factor of 30
            replica: 1
    data:
        baseuri: "kibana_data"
        # OMA : bugdette : api_call_timeout is used for retries ; should ceate a separate variable rather than this one
        api_call_timeout: 120
        groupe: "data"
        port: 5601
        default_index_pattern: "logbookoperation_*"
        check_consul: 10 # in seconds
        # index template for .kibana
        shards: 1
        replica: 1

syslog:
    # value can be syslog-ng or rsyslog
    name: "rsyslog"

cerebro:
    baseuri: "cerebro"
    port: 9000
    check_consul: 10 # in seconds
```

(suite sur la page suivante)

(suite de la page précédente)

```
189
190 siegfried:
191     port: 19000
192     consul_check: 10 # in seconds
193
194 clamav:
195     port: 3310
196     # frequency freshclam for database update per day (from 0 to 24 - 24 meaning␣
    ↪hourly check)
197     db_update_periodicity: 1
198
199 mongo_express:
200     baseuri: "mongo-express"
201
202 ldap_authentification:
203     ldap_protocol: "ldap"
204     ldap_server: "{% if groups['ldap']|length > 0 %}{{ groups['ldap']|first }}{%␣
    ↪endif %}"
205     ldap_port: "389"
206     ldap_base: "dc=programmevitam,dc=fr"
207     ldap_login: "cn=Manager,dc=programmevitam,dc=fr"
208     uid_field: "uid"
209     ldap_userDn_Template: "uid={0},ou=people,dc=programmevitam,dc=fr"
210     ldap_group_request: "(&(objectClass=groupOfNames)(member={0}))"
211     ldap_admin_group: "cn=admin,ou=groups,dc=programmevitam, dc=fr"
212     ldap_user_group: "cn=user,ou=groups,dc=programmevitam, dc=fr"
213     ldap_guest_group: "cn=guest,ou=groups,dc=programmevitam, dc=fr"
```

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

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

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

```
1  ---
2
3  vitam:
4      accessinternal:
5          jvm_opts:
6              # memory: "-Xms512m -Xmx512m"
7              # gc: ""
8              # java: ""
9      accessexternal:
10         jvm_opts:
11             # memory: "-Xms512m -Xmx512m"
12             # gc: ""
13             # java: ""
```

(suite sur la page suivante)

(suite de la page précédente)

```
    elastickibanainterceptor:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    batchreport:
          jvm_opts:
              # memory: "-Xms512m -Xmx512m"
              # gc: ""
              # java: ""
    ingestinternal:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    ingestexternal:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    metadata:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    ihm_demo:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    ihm_recette:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    logbook:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    workspace:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    processing:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    worker:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    storageengine:
        jvm_opts:
```

(suite sur la page suivante)

(suite de la page précédente)

```
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    storageofferdefault:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    functional_administration:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    security_internal:
        jvm_opts:
            # memory: "-Xms512m -Xmx512m"
            # gc: ""
            # java: ""
    library:
        jvm_opts:
            memory: "-Xms32m -Xmx128m"
            # gc: ""
            # java: ""
```

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

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

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

- Lecture asynchrone

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

Exemple :

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

- Périphériques liés à l'usage des bandes magnétiques

Terminologie :

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

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

---

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

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

---

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

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

Définition d'une offre froide :

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

Exemple :

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

.

La description « tapeLibraryConfiguration » débute par la définition des répertoires de sockage ainsi que le paramétrage des *tar*.

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

Note : N.B. : MaxTarEntrySize doit être strictement inférieur à maxTarFileSize

---

Exemple :

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

.

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

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

Exemple :

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

.

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

robots : Définition du bras robotique de la librairie.

device : Chemin du fichier de périphérique scsi générique associé au bras.

mtxPath : Chemin vers la commande Linux de manipulation du bras.

timeoutInMilliseconds : timeout en millisecondes à appliquer aux ordres du bras.

drives : Définition du/ou des lecteurs de cartouches de la librairie.

index : Numéro de lecteur, valeur débutant à 0

device : Chemin du fichier de périphérique scsi SANS REMBOBINAGE associé au lecteur.

mtPath : Chemin vers la commande Linux de manipulation des lecteurs.

ddPath : Chemin vers la commande Linux de copie de bloc de données.

tarPath : Chemin vers la commande Linux de création d'archives tar.

timeoutInMilliseconds : timeout en millisecondes à appliquer aux ordres du lecteur.

Exemple :

```
tapeLibraries:
  TAPE_LIB_1:
    robots:
      -
        device: /dev/tape/by-id/scsiQUANTUM_10F73224E6664C84A1D00000
```

(suite sur la page suivante)

(suite de la page précédente)

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

.

4.2.6 Procédure de première installation

4.2.6.1 Déploiement

4.2.6.1.1 Cas particulier : utilisation de ClamAv en environnement Debian

Dans le cas de l'installation en environnement Debian, la base de données n'est pas intégrée avec l'installation de ClamAv, C'est la commande `freshclam` qui en assure la charge. Si vous n'êtes pas connecté à internet, la base de données doit être installée manuellement. Les liens suivants indiquent la procédure à suivre : Installation ClamAv [13] et Section Virus Database [14]

4.2.6.1.2 Fichier de mot de passe

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

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

4.2.6.1.3 Mise en place des repositories VITAM (optionnel)

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

Editer le fichier `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-* .

4.2.6.1.4 Génération des *hostvars*

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

- Une d'administration
- Une de service

4.2.6.1.4.1 Cas 1 : Machines avec une seule interface réseau

Si les machines sur lesquelles *VITAM* sera déployé ne disposent que d'une interface réseau, ou si vous ne souhaitez en utiliser qu'une seule, il convient d'utiliser le playbook `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 communications d'administration et de service transiteront donc toutes les deux via l'unique interface réseau disponible.

4.2.6.1.4.2 Cas 2 : Machines avec plusieurs interfaces réseau

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

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

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

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

4.2.6.1.4.3 Vérification de la génération des hostvars

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

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

4.2.6.1.5 Déploiement

Une fois les étapes précédentes correctement effectuées (en particulier, la section *Génération des magasins de certificats* (page 48)), le déploiement s'effectue depuis la machine *ansible* et va distribuer la solution *VITAM* selon l'inventaire correctement renseigné.

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

```
ansible-playbook ansible-vitam/vitam.yml -i environments/<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).

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

4.2.7 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 installations sur des environnements hors production.

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

4.2.7.1 Configuration des *extras*

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

```
---

vitam:
    ihm_recette:
        vitam_component: ihm-recette
        host: "ihm-recette.service.{{ consul_domain }}"
        port_service: 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 }}"
        session_timeout: 1800000
        secure_cookie: true
        use_proxy_to_clone_tests: "yes"
        metrics_enabled: true
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
```

(suite sur la page suivante)

(suite de la page précédente)

```
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 10 # value in seconds
        consul_admin_check: 10 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size
    library:
        vitam_component: library
        host: "library.service.{{ consul_domain }}"
        port_service: 8090
        port_admin: 28090
        baseuri: "doc"
        https_enabled: false
        secret_platform: "false"
        metrics_enabled: false
        logback_rolling_policy: true
        logback_max_file_size: "10MB"
        logback_total_size_cap:
          file:
            history_days: 10
            totalsize: "5GB"
          security:
            history_days: 10
            totalsize: "5GB"
        jvm_log: false
        performance_logger: "false"
        reconstruction:
        consul_check_business: 30 # value in seconds
        consul_admin_check: 30 # value in seconds
        acceptableRequestTime: 10 # value in seconds
        # metricslevel: DEBUG
        # metricsinterval: 3
        # metricsunit: MINUTES
        access_retention_days: 15 # Number of days for file retention
        access_total_size_cap: "14GB" # total acceptable size

tenant_to_clean_before_tnr: ["0","1"]

# Period units in seconds
metricbeat:
    system:
        period: 10
    mongodb:
        period: 10
    elasticsearch:
        period: 10

docker_opts:
    registry_httponly: yes
```

(suite sur la page suivante)

(suite de la page précédente)

```
84      vitam_docker_tag: latest
```

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

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

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

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

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

4.2.7.2 Déploiement des *extras*

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

4.2.7.2.1 ihm-recette

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

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

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

4.2.7.2.2 *Extras* complet

Ce *playbook* permet d'installer :

- des éléments de monitoring système
- un serveur Apache pour naviguer sur le `/vitam` des différentes machines hébergeant *VITAM*
- mongo-express (en docker ; une connexion internet est alors nécessaire)
- le composant *VITAM* library, hébergeant la documentation du projet

- le composant *VITAM* ihm-recette (utilise si configuré des dépôts de jeux de tests)
- un reverse proxy, afin de fournir une page d'accueil pour les environnements de test
- l'outillage de tests de performance

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

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

CHAPITRE 5

Procédures de mise à jour de la configuration

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

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

5.1 Cas d'une modification du nombre de tenants

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

Exemple :

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

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

Exemple :

```
ansible-playbook -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
```

5.2 Cas d'une modification des paramètres JVM

Se référer à *Tuning JVM* (page 48)

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

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

Exemple :

```
ansible-playbook -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.

5.3 Cas de la mise à jour des *griffins*

Modifier la directive `vitam_griffins` contenue dans le fichier `environments/group_vars/all/vitam-vars.yml`.

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

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

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. A l'issue de l'installation, il est primordial de le sécuriser (suppression du fichier ou application d'un `chmod 400`).

6.1.2 Validation manuelle

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

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

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

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

6.1.3 Validation via Consul

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

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

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

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

6.1.4 Post-installation : administration fonctionnelle

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

- le référentiel PRONOM (lien vers pronom [15]) est correctement importé depuis « Import du référentiel des formats » et correspond à celui employé dans Siegfried
- le fichier « rules » a été correctement importé via le menu « Import du référentiel des règles de gestion »
- à terme, le registre des fonds a été correctement importé

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

6.2 Sauvegarde des éléments d'installation

Après installation, il est fortement recommandé de sauvegarder les 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 y apporter une solution associée.

6.3.1 Erreur au chargement des *index template* kibana

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

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

Ensuite, comme indiqué sur ce fil de discussion [16], vous devez désactiver le mode `READ ONLY` dans les *settings* de l'index `.kibana` du cluster elasticsearch.

Exemple :

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

(suite sur la page suivante)

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

(suite de la page précédente)

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

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

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

6.3.2 Erreur au chargement des tableaux de bord Kibana

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

6.4 Retour d'expérience / cas rencontrés

6.4.1 Crash rsyslog, code killed, signal : BUS

Il a été remarqué chez un partenaire du projet Vitam, que rsyslog se faisait *killer* peu après son démarrage par le signal SIGBUS. Il s'agit très probablement d'un bug rsyslog <= 8.24 https://github.com/rsyslog/rsyslog/issues/1404

Pour fixer ce problème, il est possible d'upgrader rsyslog sur une version plus à jour en suivant cette documentation :

- Centos [17]
- Debian [18]

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

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

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

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

[17] https://www.rsyslog.com/rhelcentos-rpms/
[18] https://www.rsyslog.com/debian-repository/

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

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

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

Sur tous ces sujets, Cf. la documentation officielle [19].

6.4.4 Elasticsearch possède des shards non initialisés (état « INITIALIZING »)

Tout d'abord, il peut être difficile d'identifier les shards en questions dans cerebro ; une requête HTTP `GET` sur l'API `_cat/shards` permet d'avoir une liste plus compréhensible. Un shard non initialisé correspond à un shard en cours de démarrage (Cf. une ancienne page de documentation [20]. Si les shards non initialisés sont présents sur un seul noeud, il peut être utile de redémarrer le noeud en cause. Sinon, une investigation plus poussée doit être menée.

6.4.5 MongoDB semble lent

Pour analyser la performance d'un cluster MongoDB, ce dernier fournit quelques outils permettant de faire une première analyse du comportement : mongostat [21] et mongotop [22] .

Dans le cas de VITAM, le cluster MongoDB comporte plusieurs shards. Dans ce cas, l'usage de ces deux commandes peut se faire :

- soit sur le cluster au global (en pointant sur les noeuds mongos) : cela permet d'analyser le comportement global du cluster au niveau de ses points d'entrées ;

https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-reroute.html
https://www.elastic.co/guide/en/elasticsearch/reference/1.4/states.html
https://docs.mongodb.com/manual/reference/program/mongostat/
https://docs.mongodb.com/manual/reference/program/mongotop/

```
mongostat --host <ip_service> --port 27017 --username vitamdb-admin --
↪password <password ; défaut : azerty> --authenticationDatabase admin
mongotop --host <ip_service> --port 27017 --username vitamdb-admin --
↪password <password ; défaut : azerty> --authenticationDatabase admin
```

- soit directement sur les noeuds de stockage (mongod) : cela donne des résultats plus fins, et permet notamment de séparer l'analyse sur les noeuds primaires & secondaires d'un même replicaset.

```
mongotop --host <ip_service> --port 27019 --username vitamdb-localadmin --
↪password <password ; défaut : qwerty> --authenticationDatabase admin
mongostat --host <ip_service> --port 27019 --username vitamdb-localadmin -
↪-password <password ; défaut : qwerty> --authenticationDatabase admin
```

D'autres outils sont disponibles directement dans le client mongo, notamment pour troubleshooter les problèmes dûs à la réplication [23] :

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

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

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

Enfin, un outil est disponible en standard afin de mesurer des performances des lecture/écritures avec des patterns proches de ceux utilisés par la base de données (mongoperf [24]) :

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

6.4.6 Les shards de MongoDB semblent mal équilibrés

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

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

6.4.7 L'importation initiale (profil de sécurité, certificats) retourne une erreur

Les playbooks d'initialisation importent des éléments d'administration du système (profils de sécurité, certificats) à travers des APIs de la solution VITAM. Cette importation peut être en échec, par exemple à

https://docs.mongodb.com/manual/tutorial/troubleshoot-replica-sets
https://docs.mongodb.com/manual/reference/program/mongoperf/

l'étape `TASK [init_contexts_and_security_profiles : Import admin security profile to functionnal-admin]`, avec une erreur de type 400. Ce type d'erreur peut avoir plusieurs causes, et survient notamment lors de redéploiements après une première tentative non réussie de déploiement ; même si la cause de l'échec initial est résolue, le système peut se trouver dans un état instable. Dans ce cas, un déploiement complet sur environnement vide est nécessaire pour revenir à un état propre.

Une autre cause possible ici est une incohérence entre l'inventaire, qui décrit notamment les offres de stockage liées aux composants offer, et le paramétrage `vitam_strategy` porté par le fichier `offers_opts.yml`. Si une offre indiquée dans la stratégie n'existe nulle part dans l'inventaire, le déploiement sera en erreur. Dans ce cas, il faut remettre en cohérence ces paramètres et refaire un déploiement complet sur environnement vide.

6.4.8 Problème d'ingest et/ou d'access

Si vous repérez un message de ce type dans les log *VITAM* :

```
fr.gouv.vitam.common.security.filter.AuthorizationWrapper.
↪checkTimestamp(AuthorizationWrapper.java:117) : [vitam-env-int8-app-04.vitam-
↪env:storage:239079175] Timestamp check failed
```

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

CHAPITRE 7

Montée de version

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

CHAPITRE 8

Annexes

8.1 Vue d'ensemble de la gestion des certificats

8.1.1 Liste des suites cryptographiques & protocoles supportés par VITAM

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

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

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

Voir aussi :

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

8.1.2 Vue d'ensemble de la gestion des certificats

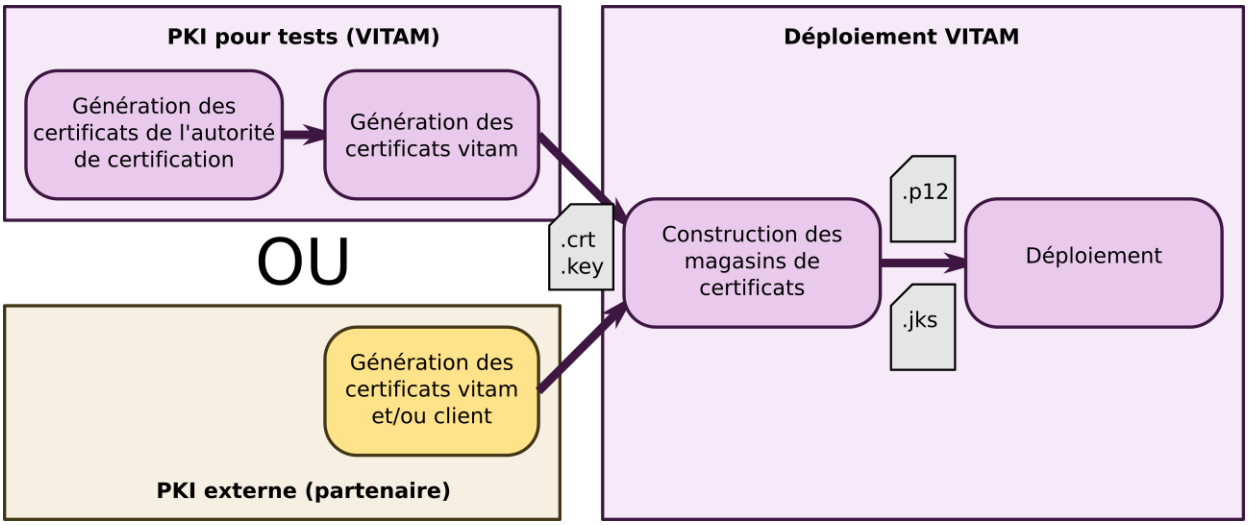

Fig. 1 – Vue d'ensemble de la gestion des certificats au déploiement

8.1.3 Description de l'arborescence de la PKI

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

- Le sous répertoire `pki` contient les scripts de génération des *CA* & des certificats, les *CA* générées par les scripts, et les fichiers de configuration d'openssl
- Le sous répertoire `environments` contient tous les certificats nécessaires au bon déploiement de *VITAM* :
  - certificats publics des *CA*
  - certificats clients, serveurs, de timestamping, et coffre fort contenant les mots de passe des clés privées des certificats (sous-répertoire `certs`)
  - magasins de certificats (keystores / truststores / grantedstores), et coffre fort contenant les mots de passe des magasins de certificats (sous-répertoire `keystores`)
- Le script `generate_stores.sh` génère les magasins de certificats (keystores), cf la section *Fonctionnement des scripts de la PKI* (page 96)

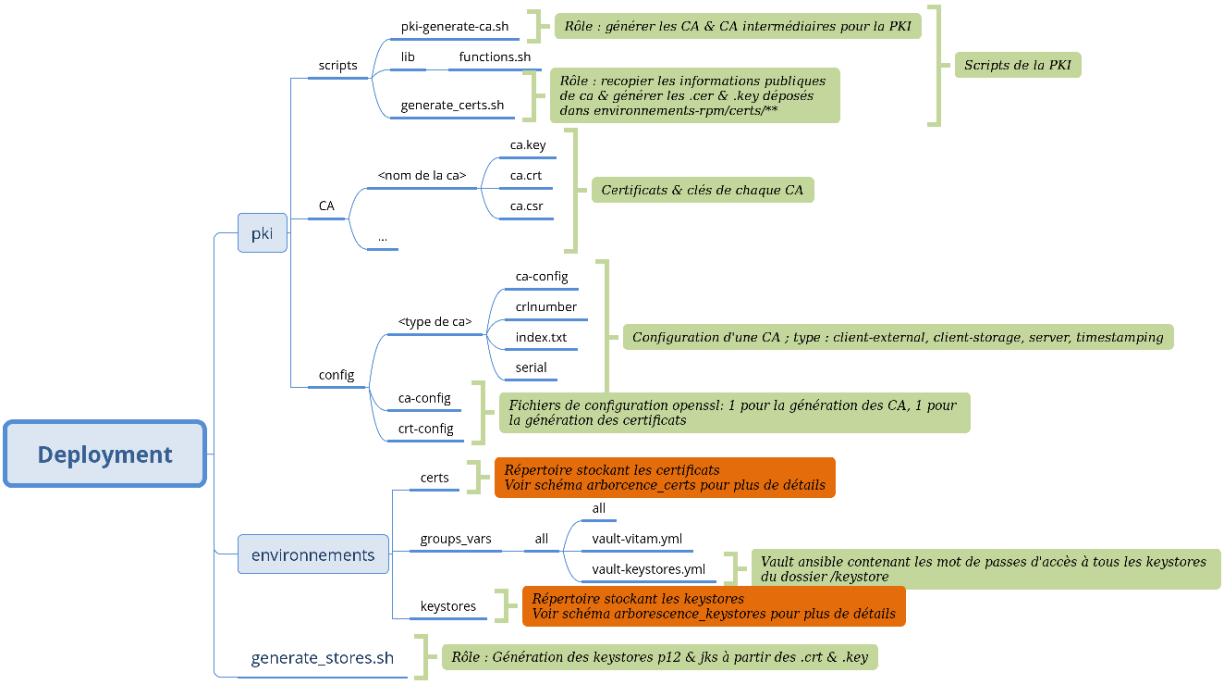

Fig. 2 – Vue l'arborescence de la *PKI* Vitam

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

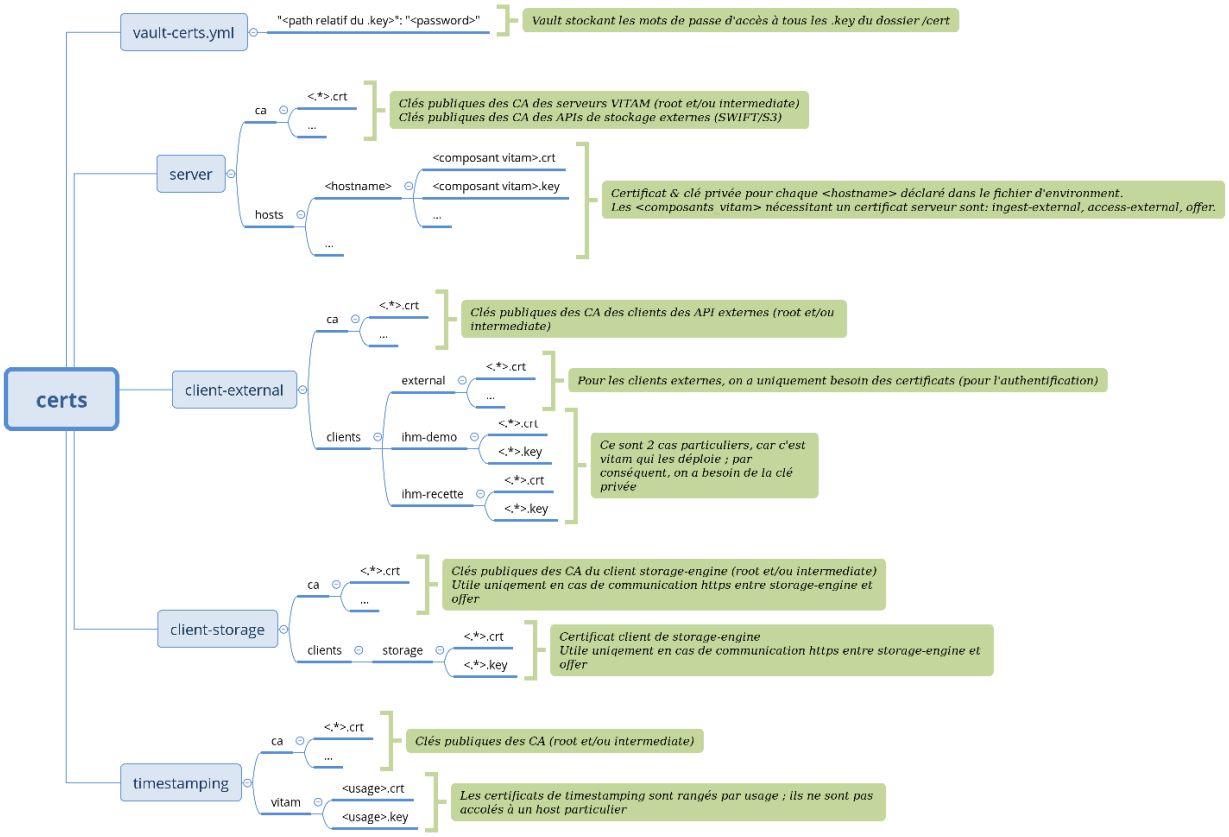

Fig. 3 – Vue détaillée de l'arborescence des certificats

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

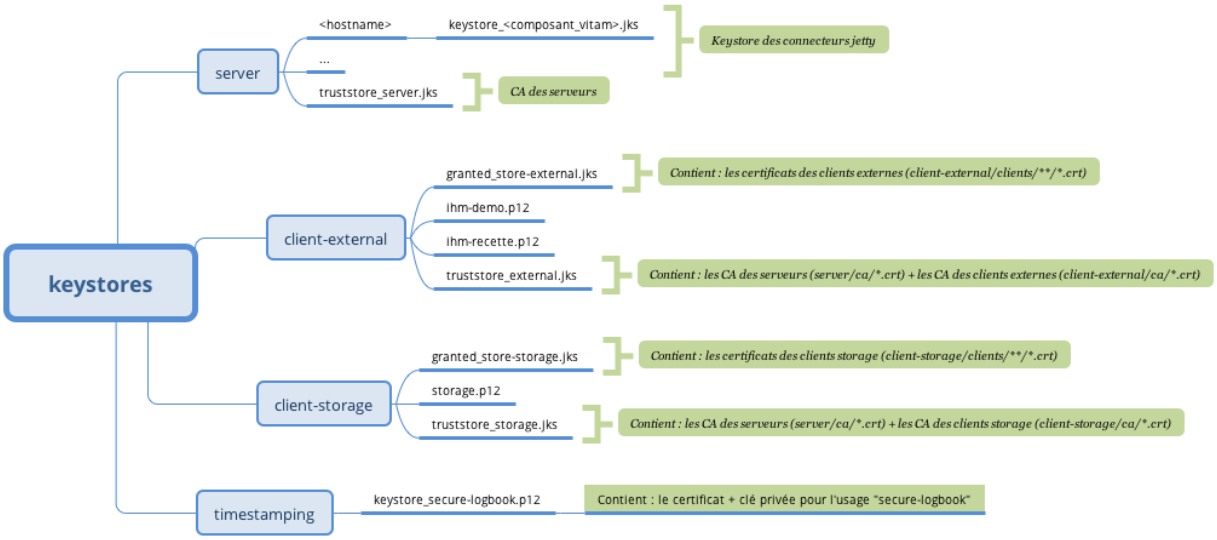

Fig. 4 – Vue détaillée de l'arborescence des keystores

8.1.6 Fonctionnement des scripts de la PKI

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

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

Si les certificats sont créés par la *PKI* externe, il faut les positionner dans l'arborescence attendue avec le nom attendu pour certains (cf *l'image ci-dessus* (page 95)).

8.2 Spécificités des certificats

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

- Certificats serveur
- Certificats client
- Certificats d'horodatage

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

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

8.2.1 Cas des certificats serveur

8.2.1.1 Généralités

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

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

8.2.1.2 Noms DNS des serveurs https VITAM

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

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

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

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

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

Exemples :

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

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

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

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

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

8.2.2 Cas des certificats client

Les services qui peuvent utiliser des certificats client sont :

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

8.2.3 Cas des certificats d'horodatage

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

8.2.4 Cas des certificats des services de stockage objets

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

8.3 Cycle de vie des certificats

Le tableau ci-dessous indique le mode de fonctionnement actuel pour les différents certificats et *CA*. Précisions :

- Les « procédures par défaut » liées au cycle de vie des certificats dans la présente version de la solution *VITAM* peuvent être résumées ainsi :
    - Création : génération par *PKI* partenaire + copie dans répertoires de déploiement + script `generate_stores.sh` + déploiement ansible
    - Suppression : suppression dans répertoires de déploiement + script `generate_stores.sh` + déploiement ansible
    - Renouvellement : regénération par *PKI* partenaire + suppression / remplacement dans répertoires de déploiement + script `generate_stores.sh` + redéploiement ansible
- Il n'y a pas de contrainte au niveau des *CA* utilisées (une *CA* unique pour tous les usages *VITAM* ou plusieurs *CA* séparées – cf. *DAT*). On appelle ici :
    - « *PKI* partenaire » : *PKI* / *CA* utilisées pour le déploiement et l'exploitation de la solution *VITAM* par le partenaire.
    - « *PKI* distante » : *PKI* / *CA* utilisées pour l'usage des frontaux en communication avec le back office *VITAM*.

Classe	Type	Usages	Origine	Création	Suppression	Renouvellement
Interne	*CA*	ingest & access	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	*CA*	offer	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Horodatage	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Storage (*Swift*)	Offre de stockage	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Storage (s3)	Offre de stockage	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	ingest	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	access	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	offer	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
Interne	Certif	Timestamp	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
IHM demo	*CA*	ihm-demo	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
IHM demo	Certif	ihm-demo	*PKI* partenaire	*proc. par défaut*	*proc. par défaut*	*proc. par défaut*
SIA	*CA*	Appel *API*	*PKI* distante	*proc. par défaut* (*PKI* distante)	*proc. par défaut*	*proc. par défaut* (*PKI* distante)+recharger Certifs
SIA	Certif	Appel *API*	*PKI* distante	Génération + copie répertoire + deploy(par la suite appel *API* d'insertion)	Suppression Mongo	Suppression Mongo + *API* d'insertion
Personae	Certif	Appel *API*	*PKI* distante	*API* ajout	*API* suppression	*API* suppression + *API* ajout

Remarques :

- Lors d'un renouvellement de *CA SIA*, il faut s'assurer que les certificats qui y correspondaient soient retirés de MongoDB et que les nouveaux certificats soient ajoutés par le biais de l'" *API* dédiée.
- Lors de toute suppression ou remplacement de certificats *SIA*, s'assurer que la suppression ou remplacement des contextes associés soit également réalisé.
- L'expiration des certificats n'est pas automatiquement prise en charge par la solution *VITAM* (pas de notification en fin de vie, pas de renouvellement automatique). Pour la plupart des usages, un certificat expiré est proprement rejeté et la connexion ne se fera pas ; les seules exceptions sont les certificats *Personae*, pour lesquels la validation de l'arborescence *CA* et des dates est à charge du front office en interface avec *VITAM*.

8.4 Ansible & SSH

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

8.4.1 Authentification du compte utilisateur utilisé pour la connexion SSH

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

8.4.1.1 Par clé SSH avec passphrase

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

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

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

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

8.4.1.2 Par login/mot de passe

Dans le cas d'une 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

8.4.1.3 Par clé SSH sans passphrase

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

8.4.2 Authentification des hôtes

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

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

8.4.3 Elévation de privilèges

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

8.4.3.1 Par sudo avec mot de passe

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

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

8.4.3.2 Par su

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

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

8.4.3.3 Par sudo sans mot de passe

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

8.4.3.4 Déjà Root

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

Table des figures

Liste des tableaux

Index