VITAM - Documentation d'installation

Version 0.11.1-RC2-SNAPSHOT

VITAM

janv. 12, 2017

Table des matières

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

CHAPITRE 1

Introduction

1.1 But de cette documentation

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

1.2 Destinataires de ce document

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

CHAPITRE 2

Rappels

2.1 Information concernant les licences

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

2.2 Documents de référence

2.2.1 Documents internes

Tableau 2.1 – Documents de référence VITAM

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

2.2.2 Référentiels externes

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

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

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

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

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

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

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

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

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

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

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

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

2.3 Glossaire

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

DIN Dossier d'Installation

DEX Dossier d'EXploitation

DAT Dossier d'Architecture Technique

IHM Interface Homme Machine

VITAM Valeurs Immatérielles Transférées aux Archives pour Mémoire

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

API Application Programming Interface

BDD Base De Données

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

JVM Java Virtual Machine ; Cf. *JRE*

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

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

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

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

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

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

SIA Système d'Informations Archivistique

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

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

CHAPITRE 3

Architecture de la solution logicielle VITAM

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

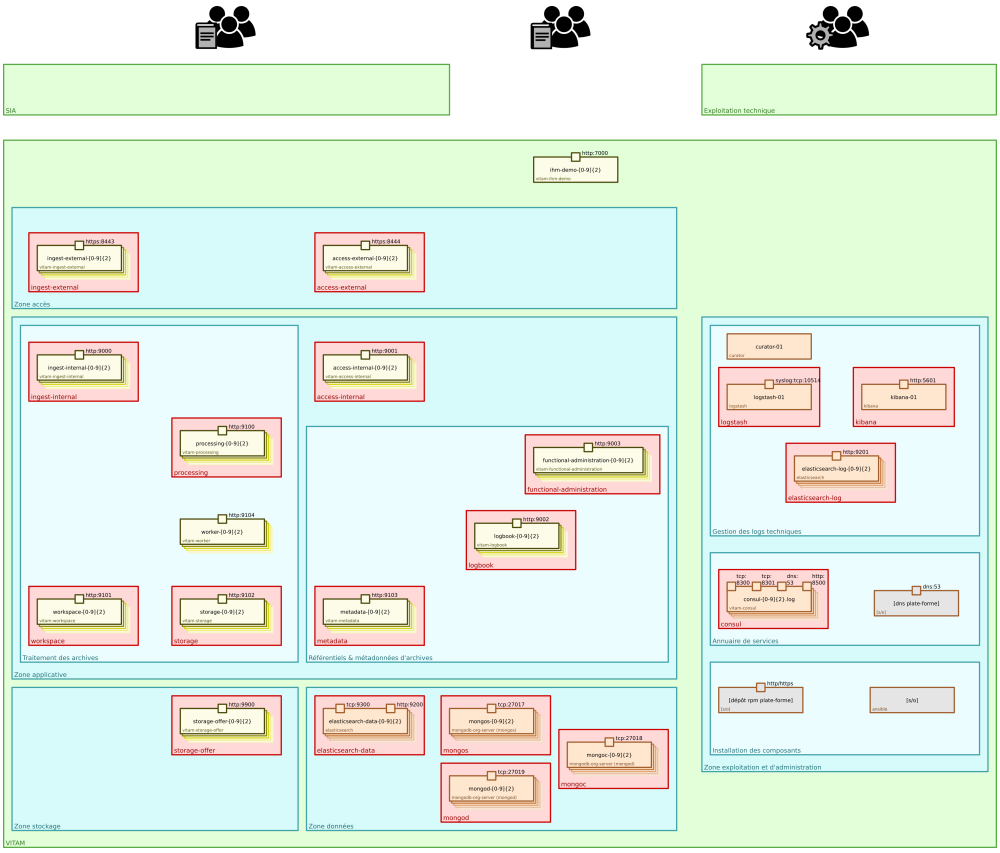

Fig. 3.1 – Vue d'ensemble d'un déploiement VITAM : zones, composants

Voir aussi :

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

CHAPITRE 4

Pré-requis

4.1 Description

Les pré-requis logiciels suivants sont nécessaires :

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

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

- packages RPM nécessaires (fournis par la distribution Centos 7) :
  - ansible (version 2.0.2 minimale et conseillée)
  - openssh-clients (client SSH utilisé par ansible)
  - java-1.8.0-openjdk & openssl (du fait de la génération de certificats / stores, l'utilitaire `keytool` est nécessaire)
- un accès ssh vers un utilisateur d'administration avec élévation de privilèges vers les droits root, vitam, vitamdb sur les serveurs cibles.
- Le compte utilisé sur le serveur d'administration doit avoir confiance dans les serveurs cibles (fichier ~/.ssh/known_hosts rempli)

4.2 Matériel

Les prérequis matériel sont définis dans le *DAT* ; à l'heure actuelle, le minimum recommandé pour la solution Vitam est 2 CPUs et 512Mo de RAM disponible par composant applicatif installé sur chaque machine.

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

- default-offer
- solution de centralisation des logs (elasticsearch)

- workspace
- elasticsearch des données Vitam

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

CHAPITRE 5

Dépendances aux services d'infrastructures

5.1 Ordonnanceurs techniques / batchs

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

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

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

5.1.1 Cas de la sauvegarde

Se référer à la section dédiée du *DAT*.

5.2 Socles d'exécution

5.2.1 OS

- CentOS 7

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

5.2.2 Middlewares

- Java : JRE 8 ; les versions suivantes ont été testées :
  - OpenJDK 1.8.0, dans la version présente dans les dépôts officiels CentOS 7 au moment de la parution de la version VITAM (actuellement : 1.8.0.101)

CHAPITRE 6

Fiche type de déploiement VITAM

6.1 Fiche-type VITAM

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

Tableau 6.1 – Tableau récapitulatif des informations à renseigner pour VITAM

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

CHAPITRE 7

Guidelines de déploiement

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

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

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

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

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

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

CHAPITRE 8

Récupération de la version

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

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

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

Sur le repository "VITAM", récupérer également depuis le tar.gz les rpm et les faire prendre en compte par le repository.

7. https://support.programmevitam.fr/releases/0.1.0/

CHAPITRE 9

Procédures d'installation / mise à jour : Package RPM

9.1 Pré-requis supplémentaire

Tous les serveurs cibles doivent avoir accès aux dépôts rpm contenant les paquets des logiciels VITAM et des composants externes requis pour l'installation. Les autres éléments d'installation (playbook ansible, ...) doivent être disponibles sur la machine ansible orchestrant le déploiement de la solution.

9.2 Procédures

9.2.1 Configuration de sécurité

En fonction de la méthode d'authentification sur les serveurs et d'élevation de privilège, il faut rajouter des options aux lignes de commande ansible. Ces options seront à rajouter pour toutes les commandes ansible du document .

Pour chacune des 3 sections suivantes, vous devez être dans l'un des cas décrits

9.2.1.1 Authentification du compte utilisateur utilisé pour la connexion SSH

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

9.2.1.1.1 Par clé SSH avec passphrase

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

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

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

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

9.2.1.1.2 Par login/mot de passe

Dans le cas d'une authentificatioon par login/mot de passe, il est nécessaire de spécifier l'option –ask-pass (ou -k en raccourci) aux commandes ansible ou ansible-playbook de ce document .

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

9.2.1.1.3 Par clé SSH sans passphrase

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

9.2.1.2 Authentification des hôtes

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

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

9.2.1.3 Elevation de privilèges

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

9.2.1.3.1 Par sudo avec mot de passe

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

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

9.2.1.3.2 Par su

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

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

9.2.1.3.3 Par sudo sans mot de passe

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

9.2.1.3.4 Déjà Root

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

9.2.2 Explications relatives à la PKI

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

9.2.2.1 Valorisation des variables propres à l'environnement

Note : Afin de réaliser l'étape ci-dessous, le mot de passe par défaut du fichier `vault.yml` se situe dans le fichier `vault_pass.txt`. Après avoir changé ce mot de passe, ne pas oublier de le mettre à jour dans le fichier `vault_pass.txt`.

Le fichier `environments-rpm/group_vars/all/vault.yml` a été généré avec un mot de passe (change_it) ; le changer par la commande :

```
ansible-vault rekey environments-rpm/group_vars/all/vault.yml
```

Pour modifier et adapter au besoin le "vault" (qui, pour rappel, contient les mots de passe sensibles de la plate-forme), éditer le fichier avec la commande :

```
ansible-vault edit environments-rpm/group_vars/all/vault.yml
```

Puis, éditer le fichier `environments-rpm/hosts.<environnement>` et le mettre en conformité de l'environnement souhaité. Ce fichier est l'inventaire associé au playbook de déploiement VITAM et il est décrit dans le paragraphe *Informations "plate-forme"* (page 31)

Note : les scripts des étapes suivantes utilisent `environments-rpm/group_vars/all/vault.yml` et, s'il existe, le fichier `vault_pass.txt` qui contient le mot de passe du fichier `vault`. Si `vault_pass.txt` n'existe pas, le mot de passe de `environments-rpm/group_vars/all/vault.yml` sera demandé.

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

9.2.2.2 Génération des autorités de certification

9.2.2.2.1 Cas d'une PKI inexistante

Dans le répertoire de déploiement, lancer le script :

```
./pki-generate-ca.sh
```

Ce script génère sous `PKI/CA` les certificats CA et intermédiaires pour client et server.

Voici ci-dessous un exemple de rendu du script :

```
Lancement de la procédure de création d'une CA
==============================================
Répertoire ./PKI/CA absent ; création...
Création du répertoire de travail temporaire newcerts sous ./PKI/newcerts...
        Création de CA root pour server...
        Create CA request...
Generating a 512 bit RSA private key
..............++++++++++++
...++++++++++++
writing new private key to './PKI/CA/server/ca.key'
-----
        Create CA certificate...
Using configuration from ./PKI/config/server/ca-config
```

```
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :T61STRING:'CA_server'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 14 15:44:32 2026 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
        CA root pour server créée sous ./PKI/CA/server !
        Création de la CA intermediate pour server...
        Generate intermediate request...
Generating a 4096 bit RSA private key
..............................................................................
 ↪..............................................................................
 ↪.................++
.............................................++
writing new private key to './PKI/CA/server_intermediate/ca.key'
-----
        Sign...
Using configuration from ./PKI/config/server/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :T61STRING:'CA_server_intermediate'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 14 15:44:33 2026 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
        CA intemédiaire server créée sous ./PKI/CA/server_intermediate !
-----------------------------------------------------------------------
        Création de CA root pour client...
        Create CA request...
Generating a 512 bit RSA private key
.....++++++++++++
.................++++++++++++
writing new private key to './PKI/CA/client/ca.key'
-----
        Create CA certificate...
Using configuration from ./PKI/config/client/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :T61STRING:'CA_client'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 14 15:44:33 2026 GMT (3650 days)

Write out database with 1 new entries
```

```
Data Base Updated
        CA root pour client créée sous ./PKI/CA/client !
        Création de la CA intermediate pour client...
        Generate intermediate request...
Generating a 4096 bit RSA private key
..................++
.....................................................................................
 ↪.....................++
writing new private key to './PKI/CA/client_intermediate/ca.key'
-----
        Sign...
Using configuration from ./PKI/config/client/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :T61STRING:'CA_client_intermediate'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 14 15:44:34 2026 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
        CA intemédiaire client créée sous ./PKI/CA/client_intermediate !
-------------------------------------------------------------------------
=========================================================================
Fin du shell
```

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

9.2.2.2.2 Cas d'une CA déjà existante

Si le client possède déjà une *PKI*, ou ne compte pas utiliser la *PKI* fournie par VITAM, il convient de positionner les fichiers `ca.crt` et `ca.key` sous `PKI/CA/<usage>`, où usage est :

- server
- server_intermediate
- client
- client_intermediate

9.2.2.3 Génération des certificats

9.2.2.3.1 Cas de certificats inexistants

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

Editer complètement le fichier `environments-rpm/<inventaire>` pour indiquer les serveurs associés à chaque service.

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

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

Ci-dessous un exemple de sortie du script :

```
Sourcer les informations nécessaires dans vault.yml
Generation du certificat client de ihm-demo
        Création du certificat  pour ihm-demo hébergé sur localhost.localdomain...
        Generation de la clé...
Generating a 4096 bit RSA private key
.....................................................................................
 ↪........++
.....................................................................................
 ↪.....................................................................................
 ↪.........................++
writing new private key to './PKI/certificats/client/ihm-demo/ihm-demo.key'
-----
        Generation du certificat signé avec client...
Using configuration from ./PKI/config/client/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :PRINTABLE:'ihm-demo'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 16 15:48:11 2019 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
        Conversion en p12...
        Fin de conversion sous ./PKI/certificats/client/ihm-demo/ !
Fin de génération du certificat client de ihm-demo
-------------------------------------------------
Generation du certificat client de ihm-recette
        Création du certificat  pour ihm-recette hébergé sur localhost.localdomain...
        Generation de la clé...
Generating a 4096 bit RSA private key
..............................++
.....................................................++
writing new private key to './PKI/certificats/client/ihm-recette/ihm-recette.key'
-----
        Generation du certificat signé avec client...
Using configuration from ./PKI/config/client/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :PRINTABLE:'ihm-recette'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 16 15:48:11 2019 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
        Conversion en p12...
```

```
        Fin de conversion sous ./PKI/certificats/client/ihm-recette/ !
Fin de génération du certificat client de ihm-recette
-----------------------------------------------
Generation du certificat server de ingest-external
        Génération pour vitam-iaas-app-01.int...
        Création du certificat server pour ingest-external hébergé sur vitam-iaas-app-
↪01.int...
        Generation de la clé...
Generating a 4096 bit RSA private key
.....................................................++
.....................................................++
writing new private key to './PKI/certificats/server/hosts/vitam-iaas-app-01.int/
↪ingest-external.key'
-----
        Generation du certificat signé avec CA server...
Using configuration from ./PKI/config/server/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :PRINTABLE:'ingest-external.service.consul'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 16 15:48:12 2019 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
        Conversion en p12...
        Fin de conversion sous ./PKI/certificats/server/hosts/vitam-iaas-app-01.int/ !
Fin de génération du certificat server de ingest-external
-----------------------------------------------------------
Generation du certificat server de access-external
        Génération pour vitam-iaas-app-01.int...
        Création du certificat server pour access-external hébergé sur vitam-iaas-app-
↪01.int...
        Generation de la clé...
Generating a 4096 bit RSA private key
.............++
.....................................................................................
↪.....................................................................................
↪.............................++
writing new private key to './PKI/certificats/server/hosts/vitam-iaas-app-01.int/
↪access-external.key'
-----
        Generation du certificat signé avec CA server...
Using configuration from ./PKI/config/server/ca-config
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :PRINTABLE:'access-external.service.consul'
organizationName      :PRINTABLE:'Vitam.'
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'idf'
localityName          :PRINTABLE:'paris'
Certificate is to be certified until Nov 16 15:48:14 2019 GMT (1095 days)

Write out database with 1 new entries
```

```
Data Base Updated
        Conversion en p12...
        Fin de conversion sous ./PKI/certificats/server/hosts/vitam-iaas-app-01.int/ !
Fin de génération du certificat server de access-external
-----------------------------------------------------
=============================================================================================
Fin de script.
```

Ce script génère sous `PKI/certificats` les certificats (format p12) nécessaires pour un bon fonctionnement dans VITAM.

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

9.2.2.3.2 Cas de certificats déjà créés par le client

Si le client possède déjà une *PKI*, ou ne compte pas utiliser la *PKI* fournie par VITAM, il convient de positionner les certificats sous `PKI/certificats/<usage>`, où usage est :

- client/ihm-recette/ihm-recette.p12
- client/ihm-demo/ihm-recette.crt
- client/ihm-demo/ihm-demo.p12
- client/ihm-demo/ihm-demo.crt
- **server/hosts/<hostname défini dans l'inventaire>/<nom composant vitam>.p12 pour**
    - ingest-external
    - access-external

9.2.2.4 Génération des stores

Lancer le script :

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

Ci-dessous un exemple de sortie du script :

```
Sourcer les informations nécessaires dans vault.yml
Génération du keystore de ihm-demo
        Génération pour vitam-iaas-ext-01.int...
Génération du truststore de ihm-demo...
        Import des CA server dans truststore de ihm-demo...
                ... import CA server root...
Certificat ajouté au fichier de clés
                ... import CA server intermediate...
Certificat ajouté au fichier de clés
                ... import CA client root...
Certificat ajouté au fichier de clés
                ... import CA client intermediate...
Certificat ajouté au fichier de clés
Fin de génération du trustore de ihm-demo
-----------------------------------------------
Génération du keystore de ihm-recette
        Génération pour vitam-iaas-ext-01.int...
Génération du truststore de ihm-recette...
```

```
        Import des CA server dans truststore de ihm-recette...
                ... import CA server root...
Certificat ajouté au fichier de clés
                ... import CA server intermediate...
Certificat ajouté au fichier de clés
                ... import CA client root...
Certificat ajouté au fichier de clés
                ... import CA client intermediate...
Certificat ajouté au fichier de clés
Fin de génération du trustore de ihm-recette
-----------------------------------------------
Génération du keystore de access-external
        Génération pour vitam-iaas-app-01.int...
        Import du p12 de ingest-external dans le keystore
L'entrée de l'alias vitam-iaas-app-01.int a été importée.
Commande d'import exécutée : 1 entrées importées, échec ou annulation de 0 entrées
Fin de génération du keystore ingest-external
-----------------------------------------------
Génération du truststore de ingest-external...
        Import des CA server dans truststore de ingest-external...
                ... import CA server root...
Certificat ajouté au fichier de clés
                ... import CA server intermediate...
Certificat ajouté au fichier de clés
                ... import CA client root...
Certificat ajouté au fichier de clés
                ... import CA client intermediate...
Certificat ajouté au fichier de clés
Fin de génération du trustore de ingest-external
-----------------------------------------------
Génération du grantedstore de ingest-external...
        Import certificat IHM-demo & ihm-recette du grantedstore de ingest-external...
Certificat ajouté au fichier de clés
Certificat ajouté au fichier de clés
-----------------------------------------------
Génération du keystore de access-external
        Génération pour vitam-iaas-app-01.int...
        Import du p12 de access-external dans le keystore
L'entrée de l'alias vitam-iaas-app-01.int a été importée.
Commande d'import exécutée : 1 entrées importées, échec ou annulation de 0 entrées
Fin de génération du keystore access-external
-----------------------------------------------
Génération du truststore de access-external...
        Import des CA server dans truststore de access-external...
                ... import CA server root...
Certificat ajouté au fichier de clés
                ... import CA server intermediate...
Certificat ajouté au fichier de clés
                ... import CA client root...
Certificat ajouté au fichier de clés
                ... import CA client intermediate...
Certificat ajouté au fichier de clés
Fin de génération du trustore de access-external
-----------------------------------------------
Génération du grantedstore de access-external...
        Import certificat IHM-demo & ihm-recette du grantedstore de access-external...
Certificat ajouté au fichier de clés
Certificat ajouté au fichier de clés
```

```
-------------------------------------------
================================================================================================
Fin de script.
```

Ce script génère sous `PKI/certificats` les stores (jks) associés pour un bon fonctionnement dans VITAM.

9.2.2.5 Recopie des bons fichiers dans l'ansiblerie

Lancer le script :

```
./copie_fichiers_vitam.sh <environnement>
```

Ci-dessous un exemple de sortie du script :

```
Recopie des stores dans VITAM
        Recopie pour access-external...
        Fichiers recopiés
------------------------
        Recopie pour ingest-external...
        Fichiers recopiés
------------------------
        Recopie pour ihm-demo...
        Fichiers recopiés
------------------------
        Recopie pour ihm-recette...
        Fichiers recopiés
------------------------
================================================================================================
Fin de procédure ; vous pouvez déployer l'ansiblerie.
```

Ce script recopie les fichiers nécessaires (certificats, stores) aux bons endroits de l'ansiblerie (sous `ansible-vitam-rpm/roles/vitam/files/<composant>`).

9.2.2.5.1 Cas des SIA

Pour le moment, la prise en charge des certificats des SIA n'est pas effective ; seuls les certificats d'ihm-demo et ihm-recette sont aujourd'hui intégrés dans l'installation.

Indice : Pour connecter un client externe à une instance de test Vitam, utiliser donc l'un des certificats cités (ihm-demo ou ihm-recette).

9.2.3 Procédure de première installation

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

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

9.2.3.1 Configuration du déploiement

9.2.3.1.1 Informations "plate-forme"

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

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

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


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

# EXTRA : Front reverse-proxy (test environments ONLY) ; add machine name after
[reverse]


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

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

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

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

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

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


##### Zone externe


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

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

```

```
51
52  ##### Zone access
53
54  # Group definition ; DO NOT MODIFY
55  [zone-access:children]
56  hosts-ingest-external
57  hosts-access-external
58
59  [hosts-ingest-external]
60  # TODO: Put here servers where this service will be deployed : ingest-external
61
62
63  [hosts-access-external]
64  # TODO: Put here servers where this service will be deployed : access-external
65
66
67  ##### Zone applicative
68
69  # Group definition ; DO NOT MODIFY
70  [zone-applicative:children]
71  hosts-ingest-internal
72  hosts-processing
73  hosts-worker
74  hosts-access-internal
75  hosts-metadata
76  hosts-functional-administration
77  hosts-logbook
78  hosts-workspace
79  hosts-storage-engine
80
81  [hosts-logbook]
82  # TODO: Put here servers where this service will be deployed : logbook
83
84
85  [hosts-workspace]
86  # TODO: Put here servers where this service will be deployed : workspace
87
88
89  [hosts-ingest-internal]
90  # TODO: Put here servers where this service will be deployed : ingest-internal
91
92
93  [hosts-access-internal]
94  # TODO: Put here servers where this service will be deployed : access-internal
95
96
97  [hosts-metadata]
98  # TODO: Put here servers where this service will be deployed : metadata
99
100
101 [hosts-functional-administration]
102 # TODO: Put here servers where this service will be deployed : functional-
    ↪administration
103
104
105 [hosts-processing]
106 # TODO: Put here servers where this service will be deployed : processing
107
```

```
108
109 [hosts-storage-engine]
110 # TODO: Put here servers where this service will be deployed : storage-engine
111
112
113 [hosts-worker]
114 # TODO: Put here servers where this service will be deployed : worker
115
116
117 ##### Zone storage
118
119 [zone-storage:children] # DO NOT MODIFY
120 hosts-storage-offer-default
121
122
123 [hosts-storage-offer-default]
124 # TODO: Put here servers where this service will be deployed : storage-offer-default
125
126
127
128 ##### Zone data
129
130 # Group definition ; DO NOT MODIFY
131 [zone-data:children]
132 hosts-elasticsearch-data
133 mongo_common
134
135
136 [hosts-elasticsearch-data]
137 # TODO: Put here servers where this service will be deployed : elasticsearch-data␣
    ↪cluster
138
139
140 # Group definition ; DO NOT MODIFY
141 [mongo_common:children]
142 mongos
143 mongoc
144 mongod
145
146 [mongos]
147 # TODO: Put here servers where this service will be deployed : mongos cluster ; add␣
    ↪after name shard_id=0
148 # Example : vitam-iaas-mongos-01.int shard_id=0
149
150 [mongoc]
151 # TODO: Put here servers where this service will be deployed : mongoc cluster
152
153
154 [mongod] # mongod declaration ; add machines name after ; add after shard_id=0 & rs_
    ↪member_id=<increasing number, starting from 0, for each line>
155 # TODO: Put here servers where this service will be deployed : mongod cluster ; add␣
    ↪after name shard_id=0
156 # Example : vitam-iaas-db-01.int rs_member_id=0 shard_id=0
157 # Example : vitam-iaas-db-02.int rs_member_id=1 shard_id=0
158 # Example : vitam-iaas-db-03.int rs_member_id=2 shard_id=0
159
160 ###### Zone admin
161
```

```
# Group definition ; DO NOT MODIFY
[zone-admin:children]
hosts-consul-server
hosts-log-server
hosts-elasticsearch-log
hosts-mongoclient

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


[hosts-log-server]
# TODO: Put here servers where this service will be deployed : log-server (kibana/
↪logstash)


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

[hosts-mongoclient]
# TODO: Put here servers where this service will be deployed : mongos cluster ; add␣
↪after name shard_id=0
# Example : vitam-iaas-mongos-01.int shard_id=0

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

[hosts:vars]
# Declare user for ansible on target machines
ansible_ssh_user=

# Can target user become as root ? ; true is required by VITAM (usage of a sudoer is␣
↪mandatory)
ansible_become=true

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

# EXTRA : FQDN of the front reverse-proxy ; used when VITAM is behind a reverse proxy␣
↪(provides configuration for reverse proxy && displayed in header page)
vitam_reverse_external_dns=

# Version that has to be deployed (defined in the release note)
# Example: rpm_version=0.9.0-RC1*
rpm_version=

# Configuration for Curator
#       Days before deletion on log management cluster; 365 for production␣
↪environment
days_to_delete=

#       Days before closing "old" indexes on log management cluster; 30 for␣
↪production environment
days_to_close=

#       Days before deletion for topbeat index only on log management cluster; 365␣
↪for production environment
days_to_delete_topbeat=
```

```
212
213 # Related to Consul ; apply in a table your DNS server(s)
214 # Example : dns_servers=["8.8.8.8","8.8.4.4"]
215 dns_servers=
216
217 #        LOG level defined in logback files ; can be a value in "ERROR","WARN","INFO",
    ↪"DEBUG","TRACE". Recommended value is "WARN"
218 log_level=
219 # FOr SoapUI files tests
220 web_dir_soapui_tests=http://vitam-prod-ldap-1.internet.agri:8083/webdav
221
222 # For reverse proxy use
223 reverse_proxy_port=80
224
225 # For metrics
226 # curator job : days before closing
227 days_to_close_metrics=7
228 # curator job : days before deleting
229 days_to_delete_metrics=30
230 # Installation ClamAV ? true/false
231 installation_clamav=true
232
233 # cas de l'appel au webDAV pour récupérer les jeux de tests
234 http_proxy_environnement=
```

Pour chaque type de "host" (lignes 2 à 176), indiquer le(s) serveur(s) défini(s) pour chaque fonction. Une colocalisation de composants est possible.

Avertissement : indiquer les contre-indications !

Ensuite, dans la section `hosts:vars` (lignes 179 à 240), renseigner les valeurs comme décrit :

Tableau 9.1 – Définition des variables

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

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

```
---

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

syslog_facility: local0

# Composants colocalisés
```

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

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

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

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

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


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

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

# Internal components communication configuration
vitam_logbook_host: "logbook.service.{{consul_domain}}"
vitam_logbook_port: 9002
vitam_logbook_baseurl: "http://{{vitam_logbook_host}}:{{vitam_logbook_port}}"

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

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

vitam_worker_port: 9104

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

vitam_storageofferdefault_host: "storage-offer-default.service.{{consul_domain}}"
```

```
vitam_storageofferdefault_port: 9900
vitam_storageofferdefault_baseurl: "http://{{vitam_storageofferdefault_host}}:{{vitam_
↪storageofferdefault_port}}"

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

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

elasticsearch_data_http_port: "9200"
elasticsearch_data_tcp_port: "9300"

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

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

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

# Normally no need for the host ?
vitam_kibana_host: "kibana.service.{{consul_domain}}"
vitam_kibana_port: 5601

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

vitam_library_port: 8090

vitam_siegfried_port: 19000

vitam_user: vitam
vitamdb_user: "vitamdb"
vitam_group: vitam

consul_domain: consul

vitam_folder_permission: 0750

vitam_conf_permission: 0640

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

soapui_component: soapui
```

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

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

service_restart_timeout: 30
```

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

```
KeyStorePassword_ingest_external: azerty
KeyManagerPassword_ingest_external: azerty
TrustStorePassword_ingest_external: tazerty
grantedKeyStorePassphrase_ingest_external: gazerty
p12_ihm_demo_password: vitam1234
TrustStore_ihm_demo_password: jksazerty
KeyStorePassword_access_external: qsdfgh
KeyManagerPassword_access_external: qsdfgh
TrustStorePassword_access_external: tqsdfgh
grantedKeyStorePassphrase_access_external: gqsdfgh
plateforme_secret: vitamsecret
mongoAdminUser: vitamdb-admin
mongoAdminPassword: azerty
mongoMetadataUser: metadata
mongoMetadataPassword: azerty
mongoLogbookUser: logbook
mongoLogbookPassword: azerty
mongoFunctionalAdminUser: functional-admin
mongoFunctionalAdminPassword: azerty
mongoClientUser: mongoclient
mongoClientPassword: azerty
mongoPassPhrase: mongogo
p12_ihm_recette_password: grokbizftej
TrustStore_ihm_recette_password: AzErTy
p12_logbook_password: p12logbook1234
```

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

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

Avertissement : le playbook `vitam.yml` comprend des étapes avec la mention `no_log` afin de ne pas afficher en clair des étapes comme les mots de passe des certificats. En cas d'erreur, il est possible de retirer la ligne dans le fichier pour une analyse plus fine d'un éventuel problème sur une de ces étapes.

9.2.3.2 Paramétrage de mongoclient (aministration mongoclient)

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

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

9.2.3.3 Première utilisation de mongoclient

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

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

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

Procédure pour configurer la connexion aux bases vitam : :

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

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

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

9.2.3.3.1 Paramétrage de l'antivirus (ingest-externe)

L'antivirus utilisé par ingest-externe est modifiable ; pour cela :

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

Ce script shell doit respecter le contrat suivant :

- Argument : chemin absolu du fichier à analyser
- Sémantique des codes de retour

  - 0 : Analyse OK - pas de virus
  - 1 : Analyse OK - virus trouvé et corrigé
  - 2 : Analyse OK - virus trouvé mais non corrigé
  - 3 : Analyse NOK
- Contenu à écrire dans stdout / stderr
  - stdout : Nom des virus trouvés, un par ligne ; Si échec (code 3) : raison de l'échec
  - stderr : Log « brut » de l'antivirus

9.2.3.3.2 Paramétrage des certificats (*-externe)

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

9.2.3.4 Déploiement

9.2.3.4.1 Fichier de mot de passe

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

9.2.3.4.2 PKI

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

```
pki-generate-ca.sh
```

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

3. Lancer le script

```
generate_certs.sh <environnement>
```

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

3. Lancer le script

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

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

4. Lancer le script

```
copie_fichiers_vitam.sh <environnement>
```

pour recopier dans les bons répertoires d'ansiblerie les certificats et stores précédemment créés.

9.2.3.4.3 Déploiement

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

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

9.2.3.4.4 Extra

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

1. ihm-recette

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

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

2. extra complet

Ce playbook permet d'installer :

- topbeat
- packetbeat
- un serveur Apache pour naviguer sur le `/vitam` des différentes machines hébergeant *VITAM*
- mongo-express (en docker ; une connexion internet est alors nécessaire)
- le composant *VITAM* library, hébergeant les documentations du projet
- le composant *VITAM* ihm-recette (nécessite un accès à un répertoire "partagé" pour récupérer les jeux de tests)
- un reverse proxy, afin de siplifier les appels aux composants

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

9.2.3.5 Import automatique d'objets dans Kibana

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

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

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

1. Faire une requête GET sur l'url suivante `http://<ip-elasticsearch-log>/.kibana/index-pattern/_search`.

2. Récupérer le contenu au format JSON et extraire le contenu de la clé **hits.hits** (qui doit être un tableau).
3. Copier ce tableau dans un fichier.
4. Copier le fichier crée à l'étape 3 dans l'emplacement `deployment\ansible-vitam-rpm\roles\log-server\files\`
5. Les index-pattern sont prêts à être importés.

9.2.4 Procédure de mise à niveau

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

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

CHAPITRE 10

Validation de la procédure

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

10.1 Sécurisation du fichier `vault_pass.txt`

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

10.2 Validation par ansible

Pour tester le déploiement de VITAM, il faut se placer dans le répertoire `deployment` et entrer la commande suivante :

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

Note : A l'issue du passage du playbook, les étapes doivent toutes passer en vert.

10.3 Validation manuelle

Chaque service VITAM (en dehors de bases de données) expose des URL de statut présente à l'adresse suivante : `<protocole web https ou https>://<host>:<port>/<composant>/v1/status` Cette URL doit retourner une réponse HTTP 200 (sans body) sur une requête HTTP GET.

`<protocole web https ou https>://<host>:<port>/admin/v1/status` => renvoie un statut HTTP 204 si OK

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

Avertissement : les composants VITAM "ihm" n'intègrent pas /admin/v1/status".

10.4 Validation via Consul

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

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

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

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

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

10.5 Validation via SoapUI

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

10.6 Post-installation : administration fonctionnelle

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

- le référentiel PRONOM (lien vers pronom [8]) est correctement importé depuis "Import du référentiel des formats" et correspond à celui employé dans Siegfried
- le fichier "rules" a été correctement importé via le menu "Import du référentiel des règles de gestion"
- à terme, le registre des fonds a été correctement importé

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

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

CHAPITRE 11

Troubleshooting

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

11.1 FAQ de la solution VITAM

11.1.1 Généralités

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

11.1.2 Retour d'expérience / cas rencontrés

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

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

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

Cependant, un shard primaire ne peut être réalloué de cette manière (il y a risque de perte de données). Si le défaut d'allocation provient effectivement de la perte puis de la récupération d'un noeud, et que TOUS les

noeuds du cluster sont de nouveaux opérationnels et dans le cluster, alors il est possible de forcer la réallocation sans perte.

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

Sur tous ces sujets, Cf. la documentation officielle [9].

Elasticsearch possède des shards non initialisés (état "INITIALIZING") Tout d'abord, il peut être difficile d'identifier les shards en questions dans le plugin Kopf ; 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 [10]. 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.

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

CHAPITRE 12

Elements extras de l'installation

Les élements décrits dans cette section sont des élements "extras" non inclus dans l'installation de base mais pouvant être utile

12.1 ClamAV en mode démon

Cette procédure permet de passer ClamAV en mode démon et non plus en mode fork à chaque SIP envoyé dans ingest. En cas d'envoi de SIP en parallèle, cela permet de limiter fortement la consommation mémoire.

Cette configuration est actuellement en extra mais devrait être réintégré dans le standard Vitam d'ici les prochaines itérations

Pour chacune des machines disposant du rôle ingest-external (entre parenthèses, les commandes sous Centos) :

- (en root) S'assurer que le package clamav-server est installé (fourni par l'EPEL) (yum install clamav-server,clamav-scanner,clamav-server-systemd)
- (en root) Editer le fichier `/etc/clamav.d/scan.conf` en modifiant les points suivants
  - Supprimer la ligne commencant par Example
  - Décommenter la ligne `TCPSocket 3310`
  - Décommenter la ligne `TCPAddr 127.0.0.1`
  - Décommenter la ligne `DetectPUA yes`
  - Décommenter la ligne `ArchiveBlockEncrypted yes`
  - Modifier la ligne `User clamscan` en `User vitam`
- (en root) Démarrer et activer le démarrage automatique du service clamd (systemctl start clamd@scan && systemctl enable clamd@scan)
- (avec l'utilisateur vitam) Editer le fichier `/vitam/conf/ingest-external/scan-clamav.sh`
  - Remplacer `clamscan -z --detect-pua=yes --block-encrypted=yes` par `clamdscan -z --config-file=/etc/clamd.d/scan.conf`

A noter qu'en cas de relance du job ansible, le fichier `/vitam/conf/ingest-external/scan-clamav.sh` sera ecrasé

CHAPITRE 13

Contacts et support

13.1 Contacts

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

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

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

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

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

CHAPITRE 14

Annexes

Table des figures

Liste des tableaux