

VITAM - Documentation de montées de version

Version 10.0.0

VITAM

nov. 15, 2026

Table des matières

1 Introduction **1**
1.1 Objectif de ce document . . . 1

2 Rappels **2**
2.1 Information concernant les licences . . . 2
2.2 Documents de référence . . . 2
2.2.1 Documents internes . . . 2
2.2.2 Référentiels externes . . . 3
2.3 Glossaire . . . 3

3 Généralités sur les versions **6**

4 Montées de version **7**
4.1 Principes généraux . . . 7
4.1.1 États attendus . . . 7
4.1.1.1 Pré-migration . . . 7
4.1.1.2 Post-migration . . . 8
4.1.2 Montées de version *bugfix* . . . 8
4.1.3 Montées de version mineure . . . 8
4.1.4 Montées de version majeure . . . 8
4.2 Montées de version *bugfix* . . . 9
4.2.1 Notes et procédures spécifiques R13 . . . 9
4.2.1.1 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+) 9
4.2.2 Notes et procédures spécifiques R16 . . . 10
4.2.2.1 Procédures à exécuter AVANT la montée de version . . . 10
4.2.2.1.1 Supprimer les indexes de configuration kibana . . . 10
4.2.2.1.2 Réinitialisation de la reconstruction des registres de fond des sites secondaires . . . 10
4.2.2.1.3 Contrôle et nettoyage de journaux du storage engine des sites secondaires 11
4.2.2.2 Procédures à exécuter APRÈS la montée de version . . . 11
4.2.2.2.1 Arrêt des timers et des accès externes à Vitam . . . 11
4.2.2.2.2 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+) . . . 11
4.2.2.2.3 Recalcul du graph des métadonnées des sites secondaires . . . 12
4.2.2.2.4 Redémarrage des timers et des accès externes à Vitam . . . 13
4.2.3 Notes et procédures spécifiques V5RC . . . 13
4.2.3.1 Procédures à exécuter AVANT la montée de version . . . 13

4.2.3.1.1 Réinitialisation de la reconstruction des registres de fond des sites secondaires . . . 13
4.2.3.1.2 Contrôle et nettoyage de journaux du storage engine des sites secondaires 13
4.2.3.2 Procédures à exécuter APRÈS la montée de version . . . 14
4.2.3.2.1 Arrêt des timers et des accès externes à Vitam . . . 14
4.2.3.2.2 Recalcul du graph des métadonnées des sites secondaires . . . 14
4.2.3.2.3 Redémarrage des timers et des accès externes à Vitam . . . 14
4.2.4 Notes et procédures spécifiques V5 . . . 14
4.2.4.1 Procédures à exécuter AVANT la montée de version . . . 14
4.2.4.1.1 Réinitialisation de la reconstruction des registres de fond des sites secondaires . . . 14
4.2.4.1.2 Contrôle et nettoyage de journaux du storage engine des sites secondaires 15
4.2.4.2 Procédures à exécuter APRÈS la montée de version . . . 15
4.2.4.2.1 Arrêt des timers et des accès externes à Vitam . . . 15
4.2.4.2.2 Recalcul du graph des métadonnées des sites secondaires . . . 15
4.2.4.2.3 Redémarrage des timers et des accès externes à Vitam . . . 16
4.2.5 Notes et procédures spécifiques V6RC . . . 16
4.2.5.1 Procédures à exécuter AVANT la montée de version . . . 16
4.2.5.1.1 Duplication des packages lors de la mise à jour . . . 16
4.2.5.2 Procédures à exécuter APRÈS la montée de version . . . 16
4.2.5.2.1 Migration des mappings elasticsearch . . . 16
4.2.6 Notes et procédures spécifiques V6 . . . 17
4.2.6.1 Procédures à exécuter AVANT la montée de version . . . 17
4.2.6.1.1 Duplication des packages lors de la mise à jour . . . 17
4.2.6.2 Procédures à exécuter APRÈS la montée de version . . . 18
4.2.6.2.1 Migration des mappings elasticsearch . . . 18
4.2.6.2.2 Migration unités archivistiques avec demande de transfert non complétée . 19
4.2.7 Notes et procédures spécifiques V7.0 . . . 19
4.2.7.1 Procédures à exécuter AVANT la montée de version . . . 19
4.2.7.1.1 Duplication des packages lors de la mise à jour . . . 19
4.2.7.2 Procédures à exécuter APRÈS la montée de version . . . 20
4.2.7.2.1 Migration unités archivistiques avec demande de transfert non complétée . 20
4.2.8 Notes et procédures spécifiques V7.1 . . . 20
4.2.8.1 Procédures à exécuter AVANT la montée de version . . . 20
4.2.8.1.1 Mise à jour de MongoDB vers la version 7.0.28 . . . 20
4.2.8.2 Procédures à exécuter APRÈS la montée de version . . . 21
4.2.8.2.1 Migration unités archivistiques avec demande de transfert non complétée . 21
4.2.9 Notes et procédures spécifiques V8.0 . . . 21
4.2.9.1 Procédures à exécuter AVANT la montée de version . . . 21
4.2.9.1.1 Mise à jour de MongoDB vers la version 7.0.28 . . . 21
4.2.10 Notes et procédures spécifiques V8.1 . . . 22
4.2.10.1 Procédures à exécuter AVANT la montée de version . . . 22
4.2.10.1.1 Mise à jour de MongoDB vers la version 8.0.23 . . . 22
4.2.11 Notes et procédures spécifiques V9.0 . . . 22
4.2.11.1 Procédures à exécuter AVANT la montée de version . . . 22
4.2.11.1.1 Mise à jour de MongoDB vers la version 8.0.23 . . . 22
4.3 Montées de version majeures . . . 22
4.3.1 Notes et procédures spécifiques R13 . . . 23
4.3.1.1 Étapes préalables à la montée de version . . . 23
4.3.1.1.1 Gestion du référentiel de l'ontologie . . . 23
4.3.1.1.2 Gestion du référentiel des formats . . . 24
4.3.1.1.3 Mise à jour de l'inventaire . . . 24
4.3.1.1.4 Arrêt des *timers* systemd . . . 25
4.3.1.1.5 Arrêt des composants *externals* . . . 25

4.3.1.1.6 Montée de version MongoDB 4.0 vers 4.2 . . . 25
4.3.1.1.7 Arrêt de l'ensemble des composants VITAM . . . 25
4.3.1.2 Montée de version . . . 26
4.3.1.3 Etapes de migration . . . 26
4.3.1.3.1 Migration des données de certificats . . . 26
4.3.1.3.2 Migration des contrats d'entrée . . . 26
4.3.1.3.3 Nettoyage des DIPs depuis les offres . . . 27
4.3.1.3.4 Réindexation ES Data . . . 27
4.3.1.3.5 Mise à jour des métadonnées de reconstruction (cas d'un site secondaire) . 27
4.3.1.3.6 Vérification de la bonne migration des données . . . 28
4.3.2 Notes et procédures spécifiques R16 . . . 28
4.3.2.1 Vérification préalable avant la migration . . . 28
4.3.2.1.1 Gestion des règles de gestion . . . 28
4.3.2.2 Adaptation des sources de déploiement ansible . . . 28
4.3.2.2.1 Déplacement des paramètres relatif à la gestion des tenants . . . 28
4.3.2.3 Procédures à exécuter AVANT la montée de version . . . 29
4.3.2.3.1 Supprimer les indexes de configuration kibana . . . 29
4.3.2.3.2 Réinitialisation de la reconstruction des registres de fond des sites secondaires . . . 29
4.3.2.3.3 Contrôle et nettoyage de journaux du storage engine des sites secondaires 30
4.3.2.4 Procédures à exécuter APRÈS la montée de version . . . 30
4.3.2.4.1 Arrêt des timers et des accès externes à Vitam . . . 30
4.3.2.4.2 Recalcul du graph des métadonnées des sites secondaires . . . 30
4.3.2.4.3 Redémarrage des timers et des accès externes à Vitam . . . 30
4.3.2.5 Vérification de la bonne migration des données . . . 31
4.3.2.5.1 Audit coherence . . . 31
4.3.3 Notes et procédures spécifiques V5RC . . . 31
4.3.3.1 Procédures à exécuter AVANT la montée de version . . . 31
4.3.3.1.1 Arrêt des timers et des accès externes à Vitam . . . 31
4.3.3.1.2 Supprimer les indexes de configuration kibana . . . 31
4.3.3.1.3 Réinitialisation de la reconstruction des registres de fond des sites secondaires . . . 32
4.3.3.1.4 Contrôle et nettoyage de journaux du storage engine des sites secondaires 32
4.3.3.1.5 Arrêt complet de Vitam . . . 32
4.3.3.2 Procédures à exécuter APRÈS la montée de version . . . 33
4.3.3.2.1 Arrêt des timers et des accès externes à Vitam . . . 33
4.3.3.2.2 Migration des unités archivistiques . . . 33
4.3.3.2.3 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+) . . . 33
4.3.3.2.4 Recalcul du graph des métadonnées des sites secondaires . . . 35
4.3.3.2.5 Redémarrage des timers et des accès externes à Vitam . . . 35
4.3.4 Notes et procédures spécifiques V5 . . . 35
4.3.4.1 Adaptation des sources de déploiement ansible . . . 35
4.3.4.1.1 Classement des offres dans une stratégie . . . 35
4.3.4.1.2 Ajout d'un nouveau module VITAM : Module de collecte . . . 36
4.3.4.2 Procédures à exécuter AVANT la montée de version . . . 37
4.3.4.2.1 Arrêt des timers et des accès externes à Vitam . . . 37
4.3.4.2.2 Réinitialisation de la reconstruction des registres de fond des sites secondaires . . . 38
4.3.4.2.3 Contrôle et nettoyage de journaux du storage engine des sites secondaires 38
4.3.4.2.4 Arrêt complet de Vitam . . . 38
4.3.4.3 Application de la montée de version . . . 39
4.3.4.3.1 Lancement du master playbook vitam . . . 39
4.3.4.3.2 Lancement du master playbook extra . . . 39

4.3.4.4 Procédures à exécuter APRÈS la montée de version . . . 39
4.3.4.4.1 Arrêt des timers et des accès externes à Vitam . . . 39
4.3.4.4.2 Migration des unités archivistiques . . . 39
4.3.4.4.3 Mise à jour des certificats . . . 40
4.3.4.4.4 Migration des registres de fonds en détails . . . 40
4.3.4.4.5 Recalcul du graph des métadonnées des sites secondaires . . . 40
4.3.4.4.6 Redémarrage des timers et des accès externes à Vitam . . . 41
4.3.5 Notes et procédures spécifiques V6RC . . . 41
4.3.5.1 Adaptation des sources de déploiement ansible . . . 41
4.3.5.1.1 Réorganisation des variables . . . 41
4.3.5.1.2 Ajout du nouveau composant scheduler . . . 41
4.3.5.2 Procédures à exécuter AVANT la montée de version . . . 42
4.3.5.2.1 Arrêt des timers et des accès externes à Vitam . . . 42
4.3.5.2.2 Mise à jour des dépôts (YUM/APT) . . . 42
4.3.5.2.3 Montée de version vers mongo 4.4 . . . 43
4.3.5.2.4 Montée de version vers mongo 5.0 . . . 43
4.3.5.2.5 Réinitialisation de la reconstruction des registres de fond des sites secondaires . . . 43
4.3.5.2.6 Contrôle et nettoyage de journaux du storage engine des sites secondaires 44
4.3.5.2.7 Arrêt complet de Vitam . . . 44
4.3.5.2.8 Nettoyage des fichiers timers, services et conf suite à la migration vers le scheduler . . . 44
4.3.5.3 Application de la montée de version . . . 45
4.3.5.3.1 Lancement du master playbook vitam . . . 45
4.3.5.3.2 Lancement du master playbook extra . . . 45
4.3.5.4 Procédures à exécuter APRÈS la montée de version . . . 45
4.3.5.4.1 Arrêt des jobs Vitam et des accès externes à Vitam . . . 45
4.3.5.4.2 Migration des groupes d'objets . . . 46
4.3.5.4.3 Recalcul du graph des métadonnées des sites secondaires . . . 46
4.3.5.4.4 Réindexation des référentiels sur elasticsearch . . . 46
4.3.5.4.5 Migration des mappings elasticsearch pour les métadonnées . . . 47
4.3.5.4.6 Redémarrage des Jobs Vitam et des accès externes à Vitam . . . 47
4.3.6 Notes et procédures spécifiques V6 . . . 47
4.3.6.1 Adaptation des sources de déploiement ansible . . . 47
4.3.6.1.1 Mise à jour de l'architecture du module de collecte . . . 47
4.3.6.1.2 Modification de l'indexation par défaut dans elasticsearch des indexes de collecte . . . 48
4.3.6.2 Procédures à exécuter AVANT la montée de version . . . 49
4.3.6.2.1 Arrêt des timers et des accès externes à Vitam . . . 49
4.3.6.2.2 Mise à jour des dépôts (YUM/APT) . . . 49
4.3.6.2.3 Nettoyage des anciens fichiers du module de collecte suite au changement d'architecture . . . 50
4.3.6.2.4 Montée de version mineure de mongo 5.0.13 -> 5.0.14 . . . 50
4.3.6.2.5 Arrêt complet de Vitam . . . 50
4.3.6.2.6 Duplication des packages lors de la mise à jour . . . 51
4.3.6.3 Application de la montée de version . . . 51
4.3.6.3.1 Lancement du master playbook vitam . . . 51
4.3.6.3.2 Lancement du master playbook extra . . . 52
4.3.6.4 Procédures à exécuter APRÈS la montée de version . . . 52
4.3.6.4.1 Arrêt des jobs Vitam et des accès externes à Vitam . . . 52
4.3.6.4.2 Réindexation des référentiels sur elasticsearch . . . 52
4.3.6.4.3 Migration des mappings elasticsearch pour les métadonnées . . . 52
4.3.6.4.4 Redémarrage des Jobs Vitam et des accès externes à Vitam . . . 52
4.3.7 Notes et procédures spécifiques V7 . . . 53

4.3.7.1 Procédures à exécuter AVANT la montée de version . . . 53
4.3.7.1.1 Arrêt des timers et des accès externes à Vitam . . . 53
4.3.7.1.2 Mise à jour des dépôts (YUM/APT) . . . 53
4.3.7.1.3 Workaround pour la mise à jour de MongoDB en cas de saut de versions . 54
4.3.7.1.4 Arrêt complet de Vitam . . . 54
4.3.7.1.5 Duplication des packages lors de la mise à jour . . . 54
4.3.7.2 Application de la montée de version . . . 55
4.3.7.2.1 Lancement du master playbook vitam . . . 55
4.3.7.2.2 Lancement du master playbook extra . . . 55
4.3.7.3 Procédures à exécuter APRÈS la montée de version . . . 55
4.3.7.3.1 Arrêt des jobs Vitam et des accès externes à Vitam . . . 55
4.3.7.3.2 Réindexation des référentiels sur elasticsearch . . . 55
4.3.7.3.3 Migration des mappings elasticsearch pour les métadonnées . . . 55
4.3.7.3.4 Redémarrage des Jobs Vitam et des accès externes à Vitam . . . 56
4.3.8 Notes et procédures spécifiques V7.1 . . . 56
4.3.8.1 Adaptation des sources de déploiement ansible . . . 56
4.3.8.1.1 Modification de la personnalisation des rôles des clusters Elasticsearch . . 56
4.3.8.1.2 Modifications du rôle curator . . . 56
4.3.8.1.3 Ajout de nouveaux exporters . . . 57
4.3.8.1.4 Modification de la durée de rétention des logs par défaut . . . 58
4.3.8.1.5 Modification de la méthodologie de concentration des logs . . . 58
4.3.8.1.6 Nouveau mode de déploiement en container (beta) . . . 58
4.3.8.2 Procédures à exécuter AVANT la montée de version . . . 59
4.3.8.2.1 Arrêt des timers et des accès externes à Vitam . . . 59
4.3.8.2.2 Mise à jour des dépôts (YUM/APT) . . . 59
4.3.8.2.3 Montée de version vers mongo 6.0 . . . 59
4.3.8.2.4 Montée de version vers mongo 7.0 . . . 60
4.3.8.2.5 Arrêt complet de Vitam . . . 60
4.3.8.2.6 Suppression des anciens meta-packages de COTS . . . 60
4.3.8.2.7 Duplication des packages lors de la mise à jour . . . 60
4.3.8.3 Application de la montée de version . . . 61
4.3.8.3.1 Lancement du master playbook vitam . . . 61
4.3.8.3.2 Lancement du master playbook extra . . . 61
4.3.8.4 Procédures à exécuter APRÈS la montée de version . . . 61
4.3.8.4.1 Réindexation de l'indice logstash-vitam courant . . . 61
4.3.8.4.2 Migration des mappings elasticsearch pour les métadonnées . . . 62
4.3.9 Notes et procédures spécifiques V8.0 . . . 62
4.3.9.1 Adaptation des sources de déploiement ansible . . . 62
4.3.9.1.1 Configuration des jobs d'audit . . . 62
4.3.9.1.2 Déploiement des exporters pour VitamUI . . . 63
4.3.9.1.2.1 # ZONE VITAMUI . . . 63
4.3.9.2 Procédures à exécuter AVANT la montée de version . . . 64
4.3.9.2.1 Arrêt des timers et des accès externes à Vitam . . . 64
4.3.9.2.2 Mise à jour des dépôts (YUM/APT) . . . 64
4.3.9.2.3 Mise à jour de MongoDB 7.0.28 . . . 64
4.3.9.2.4 Arrêt complet de Vitam . . . 65
4.3.9.3 Application de la montée de version . . . 65
4.3.9.3.1 Lancement du master playbook vitam . . . 65
4.3.9.3.2 Lancement du master playbook extra . . . 65
4.3.9.4 Procédures à exécuter APRÈS la montée de version . . . 65
4.3.9.4.1 Migration des mappings elasticsearch pour les métadonnées . . . 65
4.3.9.4.2 Migration des profiles . . . 65
4.3.9.4.3 Migration unités archivistiques avec demande de transfert non complétée . 66
4.3.10 Notes et procédures spécifiques V8.1 . . . 66

4.3.10.1 Adaptation des sources de déploiement ansible . . . 66
4.3.10.2 Procédures à exécuter AVANT la montée de version . . . 67
4.3.10.2.1 Arrêt des timers et des accès externes à Vitam . . . 67
4.3.10.2.2 Mise à jour des dépôts (YUM/APT) . . . 67
4.3.10.2.3 Montée de version vers mongo 8.0 . . . 67
4.3.10.2.4 Arrêt complet de Vitam . . . 68
4.3.10.3 Application de la montée de version . . . 68
4.3.10.3.1 Lancement du master playbook vitam . . . 68
4.3.10.3.2 Lancement du master playbook extra . . . 68
4.3.10.4 Procédures à exécuter APRÈS la montée de version . . . 68
4.3.10.4.1 Redémarrage des *Jobs Vitam* . . . 68
4.3.11 Notes et procédures spécifiques V9.0 . . . 69
4.3.11.1 Adaptation des sources de déploiement ansible . . . 69
4.3.11.1.1 Migration de la configuration d'offres S3 . . . 69
4.3.11.1.2 Déploiement du nouveau composant vitam-antivirus . . . 69
4.3.11.2 Procédures à exécuter AVANT la montée de version . . . 70
4.3.11.2.1 Arrêt des timers et des accès externes à Vitam . . . 70
4.3.11.2.2 Mise à jour des dépôts (YUM/APT) . . . 70
4.3.11.2.3 Mise à jour de MongoDB 8.0.23 . . . 70
4.3.11.2.4 Arrêt complet de Vitam . . . 71
4.3.11.3 Application de la montée de version . . . 71
4.3.11.3.1 Lancement du master playbook vitam . . . 71
4.3.11.3.2 Lancement du master playbook extra . . . 71
4.3.11.4 Procédures à exécuter APRÈS la montée de version . . . 71
4.3.11.4.1 Redémarrage des *Jobs Vitam* . . . 71
4.3.12 Notes et procédures spécifiques V9.1 . . . 72
4.3.12.1 Adaptation des sources de déploiement ansible . . . 72
4.3.12.1.1 Renommage de variables pour mise en cohérence des noms et groupes . . . 72
4.3.12.1.2 Configuration du nombre de shards/replicas pour chaque indices elasticsearch-log . . . 72
4.3.12.2 Procédures à exécuter AVANT la montée de version . . . 73
4.3.12.2.1 Arrêt des timers et des accès externes à Vitam . . . 73
4.3.12.2.2 Mise à jour des dépôts (YUM/APT) . . . 73
4.3.12.2.3 Mise à jour intermédiaire des clusters ElasticSearch . . . 74
4.3.12.2.3.1 Pour ElasticSearch-Log . . . 74
4.3.12.2.3.2 Pour ElasticSearch-Data . . . 74
4.3.12.2.4 Mise à jour de MongoDB 8.0.23 . . . 75
4.3.12.2.5 Arrêt complet de Vitam . . . 75
4.3.12.3 Application de la montée de version . . . 76
4.3.12.3.1 Lancement du master playbook vitam . . . 76
4.3.12.3.2 Lancement du master playbook extra . . . 76
4.3.12.4 Procédures à exécuter APRÈS la montée de version . . . 76
4.3.12.4.1 Migration des mappings elasticsearch pour les métadonnées . . . 76
4.3.12.4.2 Redémarrage des *Jobs Vitam* . . . 76
4.3.13 Notes et procédures spécifiques V10.0 . . . 77
4.3.13.1 Adaptation des sources de déploiement ansible . . . 77
4.3.13.1.1 Mise à jour de la pile Elastic vers 9.4.9 (US #17109) . . . 77
4.3.13.1.1.1 Préparation . . . 77
4.3.13.1.1.2 Déploiement et qualification . . . 77
4.3.13.1.1.3 Restauration . . . 78
4.3.13.1.2 Réorganisation du stockage workspace par tenant et seuils de saturation . . . 78
4.3.13.1.3 Validation de la chaîne de confiance des certificats TSA . . . 79
4.3.13.1.4 Gestion du démarrage des Jobs Vitam lors d'une montée de version . . . 79
4.3.14 Migration d'un vitam legacy vers un vitam conteneurisé . . . 80

4.3.14.1 Adaptation des sources de déploiement ansible 80
4.3.14.2 Procédures à exécuter AVANT la migration 80
4.3.14.2.1 Arrêt complet de Vitam 80
4.3.14.3 Application de la migration 81
4.3.14.3.1 Lancement du master playbook vitam 81
4.3.14.3.2 Lancement du master playbook extra 81
4.3.14.4 Procédures à exécuter APRÈS la migration 81

Index **84**

CHAPITRE 1

Introduction

1.1 Objectif de ce document

Ce document décrit les procédures et informations utiles à une équipe d'exploitants de *VITAM* afin de réaliser les montées de version 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 et/ou exploiter la solution logicielle *VITAM*.

Note : Ce document ne décrit que les chemins de montées de versions vers les versions *VITAM* maintenues. Se référer au chapitre *Généralités sur les versions* (page 6) pour plus d'informations.

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].

Les clients externes java de solution *VITAM* sont publiés sous la licence CeCILL-C [3] ; la documentation associée (comprenant le présent document) est publiée sous Licence Ouverte V2.0 [4].

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

1. https://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
3. https://cecill.info/licences/Licence_CeCILL-C_V1-fr.html
4. https://www.etalab.gouv.fr/wp-content/uploads/2017/04/ETALAB-Licence-Ouverte-v2.0.pdf

2.2.2 Référentiels externes

2.3 Glossaire

API *Application Programming Interface*

AU *Archive Unit*, unité archivistique

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 [5]

DSL *Domain Specific Language*, langage 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 Suite logicielle *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.

5. https://fr.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

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*

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 [6]

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 [7]

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 [8]

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 [9]

RGAA Référentiel Général d'Accessibilité pour les Administrations

RGI Référentiel Général d'Interopérabilité

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

SAE Système d'Archivage Électronique

SEDA Standard d'Échange de Données pour l'Archivage

6. https://fr.wikipedia.org/wiki/Attaque_de_l'homme_du_milieu
7. https://fr.wikipedia.org/wiki/NoSQL
8. https://fr.wikipedia.org/wiki/Infrastructure_%C3%A0_cl%C3%A9s_publiques
9. https://fr.wikipedia.org/wiki/Representational_state_transfer

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*

Swift *OpenStack Object Store project*

TLS *Transport Layer Security*

TNA *The National Archives*, Pronom [10]

TNR Tests de Non-Régression

TSA *Time Stamping Authority*, autorité d'horodatage délivrant les jetons d'horodatage utilisés dans les preuves

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

10. https://www.nationalarchives.gov.uk/PRONOM/

CHAPITRE 3

Généralités sur les versions

La numérotation des versions logicielles *VITAM* respecte le schéma suivant : X.Y.Z(-P).

- **X** : version majeure (V1, V2, V3)
- **Y** : version mineure (de type *release*, intitulées « R.Y . », contenant les nouvelles fonctionnalités)
- **Z** : version *bugfix*
- **P** : patch suite à bug critique (ne porte que sur les composants impactés)

TABLEAU 1: Tableau récapitulatif des versions de la solution logicielle VITAM

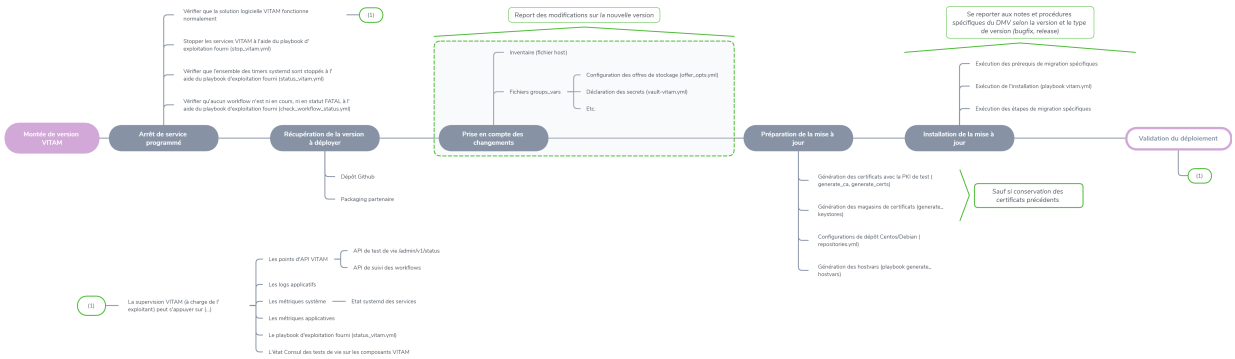
Code release	Version générale	Version associée	Version LTS	Version dépréciée
R6	1	1.0.x		X
R7	1	1.4.x		X
R8	1	1.10.x		X
R9	2	2.1.x	X	X
R10	2	2.6.x		X
R11	2	2.11.x		X
R12	2	2.15.x		X
R13	3	3.0.x	X	X
R16	4	4.0.x	X	X
R17	4	4.5.x		X
5rc	5.rc	5.rc.x		X
V5	5	5.x	X	X
6rc	6.rc	6.rc.x		X
V6	6	6.x	X	
V7.0	7.0	7.0.x		
V7.1	7.1	7.1.x	X	
V8.0	8.0	8.0.x		

CHAPITRE 4

Montées de version

4.1 Principes généraux

Le schéma ci-dessous décrit le principe général d'une montée de version de la solution logicielle *VITAM*.



4.1.1 États attendus

4.1.1.1 Pré-migration

Avant toute migration, il est attendu de la part des exploitants de vérifier :

- Que la solution logicielle *VITAM* fonctionne normalement
- Que l'ensemble des tâches planifiées soient stoppées
- Qu'aucun *workflow* ne soit ni en cours, ni en statut **FATAL**

Voir aussi :

Se référer au chapitre « Suivi de l'état du système » du *DEX* pour plus d'informations.

Voir aussi :

Se référer au chapitre « Suivi des Workflows » du *DEX*, pour plus d'informations sur la façon de vérifier l'état des statuts des *workflows*.

4.1.1.2 Post-migration

A l'issue de toute migration, il est attendu de la part des exploitants de vérifier :

- Que la solution logicielle *VITAM* fonctionne normalement
- Que l'ensemble des tâches planifiées soient bien redémarrées (les redémarrer, le cas échéant)
- Qu'aucun *workflow* ne soit en statut **FATAL**

Se référer au chapitre « Suivi de l'état du système » du *DEX* pour plus d'informations.

4.1.2 Montées de version *bugfix*

Au sein d'une même *release*, la montée de version depuis une version *bugfix* vers une version *bugfix* supérieure est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux playbooks ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Les montées de version *bugfix* ne contiennent à priori pas d'opérations de migration ou de reprises de données particulières. Toutefois, des spécificités propres aux différentes versions *bugfixes* peuvent s'appliquer ; elles sont explicitées dans le chapitre *Montées de version bugfix* (page 9).

Prudence : Parmi les versions *bugfixes* publiées au sein d'une même *release*, seuls les chemins de montées de version d'une version *bugfix* à la version *bugfix* suivante sont qualifiés par *VITAM*.

4.1.3 Montées de version mineure

La montée de version depuis une version mineure (de type *release*) vers une version mineure supérieure est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux playbooks ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Ce document décrit les chemins de montées de version depuis une version mineure, vers la version mineure maintenue supérieure.

Les montées de version mineure doivent être réalisées en s'appuyant sur les dernières versions *bugfixes* publiées.

Les opérations de migration ou de reprises de données propres aux différentes versions *releases* sont explicitées dans le chapitre *Montées de version majeures* (page 22).

Prudence : Parmi les versions mineures publiées au sein d'une même version majeure, seuls les chemins de montées de version depuis une version mineure maintenue, vers la version mineure maintenue suivante sont qualifiés par *VITAM*.

4.1.4 Montées de version majeure

La montée de version depuis une version majeure vers une version majeure supérieure s'appuie sur les chemins de montées de version mineure décrits dans le chapitre *Montées de version majeures* (page 22).

4.2 Montées de version *bugfix*

4.2.1 Notes et procédures spécifiques R13

4.2.1.1 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+)

Si vous disposez d'une instance R13.16 ou inférieur (3.0.16-), vers une version R13.17+ (3.0.17+), et que vous utilisez des offres Swift V2/V3 (providers openstack-swift-v2 et/ou openstack-swift-v3), il est nécessaire de procéder à une migration des données :

```
$ ansible-playbook ansible-vitam-exploitation/migration_swift_v2_and_v3.yml -i␣
↪environments/hosts.{env} --ask-vault-pass

# Confirm playbook execution
# Enter swift offer id (ex offer-swift-1)
# Select migration mode
# > Enter '0' for analysis only mode : This mode will only log anomalies (in offer␣
↪technical logs), no update will be proceeded
# > Enter '1' to fix inconsistencies : This mode will update swift objects to fix␣
↪inconsistencies. However, this does not prune objects (delete partially written or␣
↪eliminated objects segments to free space).
# > Enter '2' to fix inconsistencies and purge all deleted objects segments to free␣
↪storage space.
# Reconfirm playbook execution
```

Il est recommandé de lancé la procédure en mode 0 (analyse seule) et de vérifier les erreurs de cohérence dans les logs Seules les offres Swift V2/V3 avec des objets volumineux (>= 4Go) nécessitent migration. Un exemple d'incohérences journalisées dans les logs (/vitam/log/offers) est donnée ici :

```
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪old segment names [aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2,␣
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1]. Run migration script with fix␣
↪inconsistencies mode to prune container.
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪missing metadata. Run migration script with fix inconsistencies mode enabled to set␣
↪object metadata.
```

Si la détection des anomalies est terminée en succès, et que des anomalies sont trouvées, il est recommandé de lancer le mode 1 (correction des anomalies). Les migrations de données sont également journalisées dans les logs (/vitam/log/offers) :

```
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000002
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000001
Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq migrated successfully.␣
↪Digest:␣
↪8959ea1290aa064a3c64d332f31e049bd4f9d4e95bebe0b46d38613bb079761d52c865dce64c88fd7e02313d340f9a2f8c0
```

Si des problèmes de cohérence de type « Orphan large object segments » persistent

```
INCONSISTENCY FOUND : Orphan large object segments [...] without parent object␣
↪manifest: env_2_object/aeaaaaaaaagbcaacaamboal2tk7dzmiaaaaq. Eliminated object?␣
↪Incomplete write? Run migration script with delete mode to prune container.
```

Dans ce cas, il est recommandé de vérifier préalablement que les objets concernés n'existent pas sur les autres offres

(mêmes container & objectName). Si les objets n'existent pas dans les autres offres, il s'agit alors de reliquats d'objets non complètement éliminés. Le lancement du mode 2 (correction des anomalies + purge des objets) est à réaliser. Dans le cas contraire (cas où l'objet existe dans les autres offres), il faudra envisager la « Procédure de resynchronisation ciblée d'une offre » décrite dans la Documentation d'EXploitation (DEX) de Vitam pour synchroniser l'offre Swift pour les éléments concernés.

Note : Cette procédure doit être lancée une seule fois, et pour chaque offre Swift V2/V3, APRES upgrade Vitam.

4.2.2 Notes et procédures spécifiques R16

4.2.2.1 Procédures à exécuter AVANT la montée de version

4.2.2.1.1 Supprimer les indexes de configuration kibana

Prudence : Cette opération doit être effectuée avant la montée de version si vous disposez d'une instance R16.4 ou inférieur (4.0.4-), vers une version R16.5 ou supérieur (4.0.5+).

Prudence : Sans cette opération, l'installation kibana est bloquée et arrête l'installation de Vitam

Lors de la montée de version ELK, les indices de configuration kibana : .kibana et .kibana_task_manager persistent avec une version et des informations incorrectes (celles de la version d'avant). Il est nécessaire des les effacer ; autrement la montée de version est bloquée.

Exécutez le playbook suivant :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/remove_old_
↪kibana_indexes.yml --ask-vault-pass
```

Ce playbook clone les indices de configuration (.kibana et .kibana_task_manager) et efface les originaux. Les clones d'indice sont conservés.

La montée de version va recréer ces indices avec les nouvelles configurations relatives au nouvel ELK.

4.2.2.1.2 Réinitialisation de la reconstruction des registres de fond des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version R16.6- (4.0.6 ou inférieure) vers une version R16.7+ (4.0.7 ou supérieure). Elle permet la réinitialisation de la reconstruction des registre de fonds sur les sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que les timers de Vitam aient bien été préalablement arrêtés (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_accession_register_
↪reconstruction.yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.2.1.3 Contrôle et nettoyage de journaux du storage engine des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version R16.6- (4.0.6 ou inférieure) vers une version R16.7+ (4.0.7 ou supérieure). Elle permet le contrôle et la purge des journaux d'accès et des journaux d'écriture du storage engine des sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_purge_storage_logs_secondary_
↪sites.yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.2.2 Procédures à exécuter APRÈS la montée de version

4.2.2.2.1 Arrêt des timers et des accès externes à Vitam

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.2.2.2.2 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+)

Note : Cette procédure doit être lancée une seule fois, et pour chaque offre Swift V2/V3, APRES upgrade Vitam, et uniquement depuis le **site primaire**.

Si vous disposez d'une instance R16.3 ou inférieur (4.0.3-), vers une version R16.4+ (4.0.4+), et que vous utilisez des offres Swift V2/V3 (providers openstack-swift-v2 et/ou openstack-swift-v3), il est nécessaire de procéder à une migration des données :

```
$ ansible-playbook ansible-vitam-migration/migration_swift_v2_and_v3.yml -i␣
↪environments/hosts.{env} --ask-vault-pass

# Confirm playbook execution
# Enter swift offer id (ex offer-swift-1)
# Select migration mode
# > Enter '0' for analysis only mode : This mode will only log anomalies (in offer␣
↪technical logs), no update will be proceeded
# > Enter '1' to fix inconsistencies : This mode will update swift objects to fix␣
↪inconsistencies. However, this does not prune objects (delete partially written or␣
↪eliminated objects segments to free space).
```

(suite sur la page suivante)

(suite de la page précédente)

```
# > Enter '2' to fix inconsistencies and purge all deleted objects segments to free␣
↪storage space.
# Reconfirm playbook execution
```

Il est recommandé de lancé la procédure en mode 0 (analyse seule) et de vérifier les erreurs de cohérence dans les logs Seules les offres Swift V2/V3 avec des objets volumineux (>= 4Go) nécessitent migration. Un exemple d'incohérences journalisées dans les logs (/vitam/log/offers) est donnée ici :

```
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪old segment names [aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2,␣
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1]. Run migration script with fix␣
↪inconsistencies mode to prune container.
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪missing metadata. Run migration script with fix inconsistencies mode enabled to set␣
↪object metadata.
```

Si la détection des anomalies est terminée en succès, et que des anomalies sont trouvées, il est recommandé de lancer le mode 1 (correction des anomalies). Les migrations de données sont également journalisées dans les logs (/vitam/log/offers) :

```
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000002
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000001
Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq migrated successfully.␣
↪Digest:␣
↪8959ea1290aa064a3c64d332f31e049bd4f9d4e95bebe0b46d38613bb079761d52c865dce64c88fd7e02313d340f9a2f8c0
```

Si des problèmes de cohérence de type « Orphan large object segments » persistent

```
INCONSISTENCY FOUND : Orphan large object segments [...] without parent object␣
↪manifest: env_2_object/aeaaaaaaaagbcaacaamboal2tk7dzmiaaaaq. Eliminated object?␣
↪Incomplete write? Run migration script with delete mode to prune container.
```

Dans ce cas, il est recommandé de vérifier préalablement que les objets concernés n'existent pas sur les autres offres (mêmes container & objectName). Si les objets n'existent pas dans les autres offres, il s'agit alors de reliquats d'objets non complètement éliminés. Le lancement du mode 2 (correction des anomalies + purge des objets) est à réaliser. Dans le cas contraire (cas où l'objet existe dans les autres offres), il faudra envisager la « Procédure de resynchronisation ciblée d'une offre » décrite dans la Documentation d'EXploitation (DEX) de Vitam pour synchroniser l'offre Swift pour les éléments concernés.

4.2.2.2.3 Recalcul du graph des métadonnées des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version R16.6- (4.0.6 ou inférieure) vers une version R16.7+ (4.0.7 ou supérieure). Elle permet le recalcul du graphe des métadonnées sur les sites secondaires

La procédure est à réaliser sur tous les **sites secondaires** de Vitam APRÈS l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_metadata_graph_reconstruction.
↪yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.2.2.4 Redémarrage des timers et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les timers sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪timers.yml --ask-vault-pass
```

4.2.3 Notes et procédures spécifiques V5RC

4.2.3.1 Procédures à exécuter AVANT la montée de version

4.2.3.1.1 Réinitialisation de la reconstruction des registres de fond des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 5.rc.3- (v5.rc.3 ou inférieur) vers une version 5.rc.4+ (5.rc.4 ou supérieure). Elle permet la réinitialisation de la reconstruction des registre de fonds sur les sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que les timers de Vitam aient bien été préalablement arrêtés (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_accession_register_
↪reconstruction.yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.3.1.2 Contrôle et nettoyage de journaux du storage engine des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 5.rc.3- (v5.rc.3 ou inférieur) vers une version 5.rc.4+ (5.rc.4 ou supérieure). Elle permet le contrôle et la purge des journaux d'accès et des journaux d'écriture du storage engine des sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_purge_storage_logs_secondary_
↪sites.yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.3.2 Procédures à exécuter APRÈS la montée de version

4.2.3.2.1 Arrêt des timers et des accès externes à Vitam

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.2.3.2.2 Recalcul du graph des métadonnées des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 5.rc.3- (v5.rc.3 ou inférieur) vers une version 5.rc.4+ (5.rc.4 ou supérieure). Elle permet le recalcul du graphe des métadonnées sur les sites secondaires

La procédure est à réaliser sur tous les **sites secondaires** de Vitam APRÈS l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_metadata_graph_reconstruction.
↪yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.3.2.3 Redémarrage des timers et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les timers sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪timers.yml --ask-vault-pass
```

4.2.4 Notes et procédures spécifiques V5

4.2.4.1 Procédures à exécuter AVANT la montée de version

4.2.4.1.1 Réinitialisation de la reconstruction des registres de fond des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 5.0 vers une version 5.1+ (5.1 ou supérieure). Elle permet la réinitialisation de la reconstruction des registre de fonds sur les sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que les timers de Vitam aient bien été préalablement arrêtés (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_accession_register_
↪reconstruction.yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.4.1.2 Contrôle et nettoyage de journaux du storage engine des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 5.0 vers une version 5.1+ (5.1 ou supérieure). Elle permet le contrôle et la purge des journaux d'accès et des journaux d'écriture du storage engine des sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_purge_storage_logs_secondary_
↪sites.yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.4.2 Procédures à exécuter APRÈS la montée de version

4.2.4.2.1 Arrêt des timers et des accès externes à Vitam

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.2.4.2.2 Recalcul du graph des métadonnées des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 5.0 vers une version 5.1+ (5.1 ou supérieure). Elle permet le recalcul du graphe des métadonnées sur les sites secondaires

La procédure est à réaliser sur tous les **sites secondaires** de Vitam APRÈS l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_metadata_graph_reconstruction.
↪yml -i environments/hosts.{env} --ask-vault-pass
```

4.2.4.2.3 Redémarrage des timers et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les timers sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪timers.yml --ask-vault-pass
```

4.2.5 Notes et procédures spécifiques V6RC

4.2.5.1 Procédures à exécuter AVANT la montée de version

4.2.5.1.1 Duplication des packages lors de la mise à jour

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 6.rc.1- (v6.rc.1 ou inférieur) vers une version 6.rc.2+ (v6.rc.2 ou supérieure).

Prudence : Cette opération doit être effectuée uniquement pour les systèmes d'exploitation à base de RPM (CentOS 7 & AlmaLinux 9) sur un vitam éteint.

Un bug a été introduit dans la construction des packages RPM à partir de la V6RC. Si vous effectuez une montée de version depuis la V6RC ou de la V6, vous devrez appliquer la commande suivante pour supprimer les précédents packages. Sinon lors de de la mise à jour, le package précédent ne sera pas désinstallé conduisant à la présence de plusieurs versions du même package.

Exemple d'erreur :

```
/var/tmp/rpm-tmp.fGwZMk: ligne 1 : fg: pas de contrôle de tâche
erreur : %preun(vitam-offer-6.2-1.noarch) scriptlet échoué, état de sortie 1
Error in PREUN scriptlet in rpm package vitam-offer
erreur : vitam-offer-6.2-1.noarch: effacer échoué
```

Voici la commande à exécuter pour permettre la désinstallation des packages de la version précédente :

```
ansible vitam --ask-vault-pass -v -a "yum --setopt=tsflags=noscripts remove -y vitam-
↪*.noarch" -i environments/<inventaire>
```

4.2.5.2 Procédures à exécuter APRÈS la montée de version

4.2.5.2.1 Migration des mappings elasticsearch

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 6.rc.3- (v6.rc.3 ou inférieur) vers une version 6.rc.4+ (6.rc.4 ou supérieure).

Cette migration de données consiste à mettre à jour le modèle d'indexation elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

- Les jobs Vitam et les services externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

- Réindexation des référentiels sur elasticsearch :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --ask-vault-pass --tags "securityprofile, context, ontology,␣
↪ingestcontract, agencies, accessionregisterdetail, archiveunitprofile,␣
↪accessionregistersummary, accesscontract, fileformat, filerules, profile, griffin,␣
↪preservationscenario, managementcontract"
```

- Lancement de la migration du modèles d'indexation des métadonnées sur elasticsearch-data :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

- Réactivation des services externals ainsi que les timers sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduler.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.2.6 Notes et procédures spécifiques V6

4.2.6.1 Procédures à exécuter AVANT la montée de version

4.2.6.1.1 Duplication des packages lors de la mise à jour

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 6.2- (v6.2 ou inférieur) vers une version 6.3+ (v6.3 ou supérieure).

Prudence : Cette opération doit être effectuée uniquement pour les systèmes d'exploitation à base de RPM (CentOS 7 & AlmaLinux 9) sur un vitam éteint.

Un bug a été introduit dans la construction des packages RPM à partir de la V6RC. Si vous effectuez une montée de version depuis la V6RC ou de la V6, vous devrez appliquer la commande suivante pour supprimer les précédents

packages. Sinon lors de de la mise à jour, le package précédent ne sera pas désinstallé conduisant à la présence de plusieurs versions du même package.

Exemple d'erreur :

```
/var/tmp/rpm-tmp.fGwZMk: ligne 1 : fg: pas de contrôle de tâche
erreur : %preun(vitam-offer-6.2-1.noarch) scriptlet échoué, état de sortie 1
Error in PREUN scriptlet in rpm package vitam-offer
erreur : vitam-offer-6.2-1.noarch: effacer échoué
```

Voici la commande à exécuter pour permettre la désinstallation des packages de la version précédente :

```
ansible vitam --ask-vault-pass -v -a "yum --setopt=tsflags=noscripts remove -y vitam-
↪*.noarch" -i environments/<inventaire>
```

4.2.6.2 Procédures à exécuter APRÈS la montée de version

4.2.6.2.1 Migration des mappings elasticsearch

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 6.1- (v6.1 ou inférieur) vers une version 6.2+ (6.2 ou supérieure).

Cette migration de données consiste à mettre à jour le modèle d'indexation elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

- Les jobs Vitam et les services externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

- Réindexation des référentiels sur elasticsearch :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --ask-vault-pass --tags "securityprofile, context, ontology,␣
↪ingestcontract, agencies, accessionregisterdetail, archiveunitprofile,␣
↪accessionregistersummary, accesscontract, fileformat, filerules, profile, griffin,␣
↪preservationscenario, managementcontract"
```

- Lancement de la migration du modèles d'indexation des métadonnées sur elasticsearch-data :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

- Réactivation des services externals ainsi que les timers sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduler.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.2.6.2.2 Migration unités archivistiques avec demande de transfert non complétée

Cette migration permet de corriger des erreurs de sécurisation ou d'audit sur des unités archivistiques pour lesquelles une demande de transfert a été initiée, mais dont le transfert effectif n'a pas été finalisée.

Prudence : Cette procédure doit être exécutée uniquement en cas de migration mineure depuis une version 6.3-1- (v6.3-1 ou inférieure) vers une version 6.4+ (v6.4 ou supérieure).

Prudence : Cette migration est à effectuer **APRES** l'installation et uniquement sur le **site primaire**.

Prudence : Dans le cas où un tenant comporte un très grand nombre d'unités archivistiques dont la demande de transfert a été initiée pour un tenant donné (plus de 100 000 unités archivistiques), la migration ne pourra alors s'opérer. Le support Vitam devra alors être contacté.

Pour lancer la migration, le playbook suivant doit être exécuté :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪metadata_transfer_request_traceability.yml --ask-vault-pass
```

4.2.7 Notes et procédures spécifiques V7.0

4.2.7.1 Procédures à exécuter AVANT la montée de version

4.2.7.1.1 Duplication des packages lors de la mise à jour

Prudence : Cette procédure doit être exécutée uniquement en cas de migration depuis une version 7.0.1- (v7.0.1 ou inférieur) vers une version 7.0.2+ (v7.0.2 ou supérieure).

Prudence : Cette opération doit être effectuée uniquement pour les systèmes d'exploitation à base de RPM (CentOS 7 & AlmaLinux 9) sur un vitam éteint.

Un bug a été introduit dans la construction des packages RPM à partir de la V6RC. Si vous effectuez une montée de version depuis la V6RC ou de la V6, vous devrez appliquer la commande suivante pour supprimer les précédents packages. Sinon lors de de la mise à jour, le package précédent ne sera pas désinstallé conduisant à la présence de plusieurs versions du même package.

Exemple d'erreur :

```
/var/tmp/rpm-tmp.fGwZMk: ligne 1 : fg: pas de contrôle de tâche
erreur : %preun(vitam-offer-6.2-1.noarch) scriptlet échoué, état de sortie 1
Error in PREUN scriptlet in rpm package vitam-offer
erreur : vitam-offer-6.2-1.noarch: effacer échoué
```

Voici la commande à exécuter pour permettre la désinstallation des packages de la version précédente :

```
ansible vitam --ask-vault-pass -v -a "yum --setopt=tsflags=noscripts remove -y vitam-
↪*.noarch" -i environments/<inventaire>
```

4.2.7.2 Procédures à exécuter APRÈS la montée de version

4.2.7.2.1 Migration unités archivistiques avec demande de transfert non complétée

Cette migration permet de corriger des erreurs de sécurisation ou d'audit sur des unités archivistiques pour lesquelles une demande de transfert a été initiée, mais dont le transfert effectif n'a pas été finalisée.

Prudence : Cette procédure doit être exécutée uniquement en cas de migration mineure depuis une version 7.0.1- (v7.0.1 ou inférieure) vers une version 7.0.2+ (v7.0.2 ou supérieure).

Prudence : Cette migration est à effectuer **APRES** l'installation et uniquement sur le **site primaire**.

Prudence : Dans le cas où un tenant comporte un très grand nombre d'unités archivistiques dont la demande de transfert a été initiée pour un tenant donné (plus de 100 000 unités archivistiques), la migration ne pourra alors s'opérer. Le support Vitam devra alors être contacté.

Pour lancer la migration, le playbook suivant doit être exécuté :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪metadata_transfer_request_traceability.yml --ask-vault-pass
```

4.2.8 Notes et procédures spécifiques V7.1

4.2.8.1 Procédures à exécuter AVANT la montée de version

4.2.8.1.1 Mise à jour de MongoDB vers la version 7.0.28

Prudence : Attention Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V7.1. Cette opération est à effectuer si vous venez des versions de Vitam suivante : V7.1.4-. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V7.1 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_70.yml --ask-vault-pass
```

4.2.8.2 Procédures à exécuter APRÈS la montée de version

4.2.8.2.1 Migration unités archivistiques avec demande de transfert non complétée

Cette migration permet de corriger des erreurs de sécurisation ou d'audit sur des unités archivistiques pour lesquelles une demande de transfert a été initiée, mais dont le transfert effectif n'a pas été finalisée.

Prudence : Cette procédure doit être exécutée uniquement en cas de migration mineure depuis une version 7.1.1 (v7.1.1 ou inférieure) vers une version 7.1.2+ (v7.1.2 ou supérieure).

Prudence : Cette migration est à effectuer **APRES** l'installation et uniquement sur le **site primaire**.

Prudence : Dans le cas où un tenant comporte un très grand nombre d'unités archivistiques dont la demande de transfert a été initiée pour un tenant donné (plus de 100 000 unités archivistiques), la migration ne pourra alors s'opérer. Le support Vitam devra alors être contacté.

Pour lancer la migration, le playbook suivant doit être exécuté :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪metadata_transfer_request_traceability.yml --ask-vault-pass
```

4.2.9 Notes et procédures spécifiques V8.0

4.2.9.1 Procédures à exécuter AVANT la montée de version

4.2.9.1.1 Mise à jour de MongoDB vers la version 7.0.28

Prudence : **Attention** Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V8.0. Cette opération est à effectuer si vous venez des versions de Vitam suivante : V8.0.2-. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V8.0 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_70.yml --ask-vault-pass
```

4.2.10 Notes et procédures spécifiques V8.1

4.2.10.1 Procédures à exécuter AVANT la montée de version

4.2.10.1.1 Mise à jour de MongoDB vers la version 8.0.23

> **Prudence : Attention** Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V8.1. Cette opération est à effectuer si vous venez des versions de Vitam suivante : V8.1.2-. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V8.1 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_80.yml --ask-vault-pass
```

4.2.11 Notes et procédures spécifiques V9.0

4.2.11.1 Procédures à exécuter AVANT la montée de version

4.2.11.1.1 Mise à jour de MongoDB vers la version 8.0.23

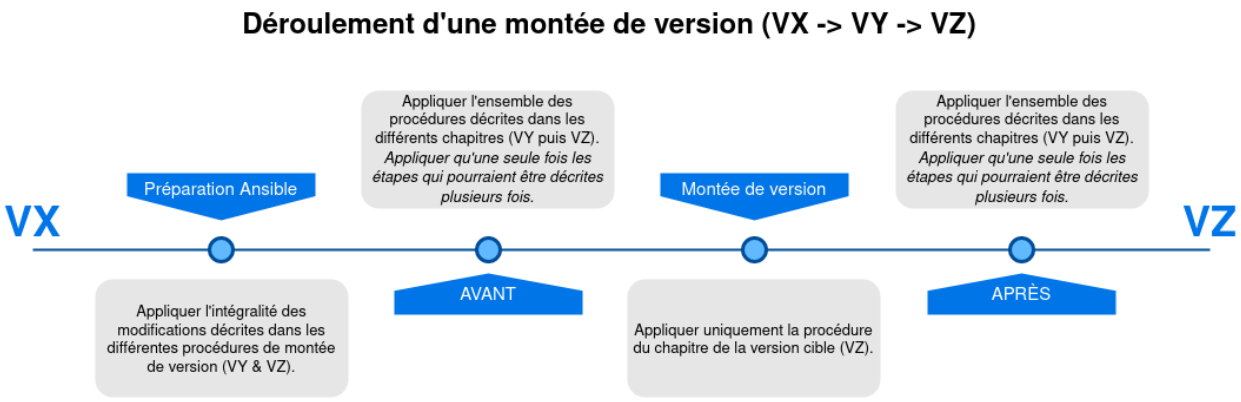
> **Prudence : Attention** Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V9.0. Cette opération est à effectuer si vous venez des versions de Vitam suivante : V9.0.0. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V9.0 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_80.yml --ask-vault-pass
```

4.3 Montées de version majeures

Le schéma ci-dessous décrit le principe général pour effectuer plusieurs montée de version majeures en une seule opération de maintenance. Tous les chemins ne sont pas forcément supportés, veuillez vous reporter au support *VITAM* afin de vous assurer que le chemin que vous prévoyez soit officiellement supporté.



4.3.1 Notes et procédures spécifiques R13

Prudence : Rappel : la montée de version vers la *release* R13 s'effectue depuis la *release* R9 (LTS V2), la *release* R10 (V2, *deprecated*), la *release* R11 (V2, *deprecated*) ou la *release* R12 (V2) et doit être réalisée en s'appuyant sur les dernières versions *bugfixes* publiées.

Note : Cette *release* de la solution logicielle *VITAM* s'appuie sur la version 11 de Java ainsi que la version 7 d'Elasticsearch. Ces mises à jour sont prises en charge par le processus de montée de version.

Prudence : La montée de version vers la *release* R13 a été validée par *VITAM* dans le cadre d'une installation de type Centos 7. L'installation dite *from scracth* a quant à elle été validée pour les installations de type Centos 7 et Debian 10 (l'utilisation de la version 11 de Java impose en effet une installation de type Debian 10). La migration d'OS vers la version Debian 10 n'est pas supportée par *VITAM* dans le cadre de la montée de version vers la *release* R13.

4.3.1.1 Étapes préalables à la montée de version

4.3.1.1.1 Gestion du référentiel de l'ontologie

Prudence : en lien avec la User Story* #6213 (livré en version *3.4.x* de *VITAM*), les ontologies externes en cours d'exploitation par *VITAM* ne sont pas touchées, et seront mergées avec les ontologies internes situés : `deployment/environments/ontology/VitamOntology.json` .

La procédure de merge manuelle du référentiel de l'ontologie avant chaque montée de version n'est plus nécessaire.

Lors du lancement du procédure de mise à jour de *VITAM*, une phase préliminaire de vérification et validation sera faite pour détecter des éventuelles conflits entre les vocabulaires internes et externes.

Afin d'assurer que la montée de version de *VITAM* passera sans affecter le système , cette vérification s'exécute dans les phases préliminaires de l'ansiblerie, avant la phase de l'installation des composants *VITAM*, (en cas d'echec à cette étape, la solution logicielle déjà installé ne sera pas affectée).

Le script ansible qui fait le check est situé dans : `deployment/ansible-vitam/roles/check_ontologies/tasks/main.yml`, le role vérifie que le composant d'administration fonctionnelle `vitam-functional-administration` est bien installé et démarré, ensuite la tâche `Check Import Ontologies` réalise un import à blanc en mode `Dry Run` du référentiel de l'ontologie et remonte des éventuelles erreurs d'import.

Danger : En cas d'echec de vérification, autrement dit, en cas de présence de conflits entre les deux vocabulaires (le vocabulaire interne à mettre à jour et le vocabulaire externe en cours d'exploitation), c'est à l'exploitant d'adapter son vocabulaire externe et de veiller à ce qu'il n'ya pas de moindres conflits.

L'exploitant pour vérifier ses corrections en cas d'erreurs, pourra toutefois lancer la commande depuis le dossier *deployment*, depuis une instance hébergeant le composant `vitam-functional-administration` :

```
curl -XPOST -H "Content-type: application/json" -H "X-Tenant-Id: 1" --data-binary␣
↪@environments/ontology/VitamOntology.json 'http://{{ hostvars[groups['hosts_
↪functional_administration'][0]]['ip_admin'] }}:{{ vitam.functional_administration.
↪port_admin }}/v1/admin/ontologies/check'
```

Dès résolution des conflicts, l'exploitant lancera la mise à jour sans toucher l'ontologie interne.

Note : Lors de la montée de version, une sauvegarde du référentiel de l'ontologie courant est réalisée à l'emplacement `environments/backups/ontology_backup_<date>.json`

A ce jour l'import de l'ontologie externe seulement n'est pas possible, ce comportement changera dans le futur.

4.3.1.1.2 Gestion du référentiel des formats

Prudence : Si un référentiel des formats personnalisé est utilisé avec la solution logicielle *VITAM*, il faut impérativement, lors d'une montée de version, modifier manuellement le fichier des formats livré par défaut avant toute réinstallation afin d'y réintégrer les modifications. A défaut, le référentiel des formats sera réinitialisé.

Il faut pour cela éditer le fichier situé à l'emplacement `environments/DROID_SignatureFile_<version>.xml` afin d'y réintégrer les éléments du référentiel des formats personnalisés.

4.3.1.1.3 Mise à jour de l'inventaire

Les versions récentes de ansible préconisent de ne plus utiliser le caractère « - » dans les noms de groupes ansible.

Pour effectuer cette modification, un script de migration est mis à disposition pour mettre en conformité votre « ancien » inventaire dans une forme compatible avec les outils de déploiement de la *release* R12.

La commande à lancer est

```
cd deployment
./upgrade_inventory.sh ${fichier_d_inventaire}
```

4.3.1.1.4 Arrêt des *timers* systemd

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam_timers.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les *timers* systemd ont été arrêtés, afin de ne pas perturber la migration.

Il est également recommandé de ne lancer la procédure de migration qu'après s'être assuré que plus aucun *workflow* n'est ni en cours, ni en statut **FATAL**.

4.3.1.1.5 Arrêt des composants *externals*

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_external.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les composants *externals* ont été arrêtés, afin de ne pas perturber la migration.

4.3.1.1.6 Montée de version MongoDB 4.0 vers 4.2

La montée de version vers la *release* R12 comprend une montée de version de la bases de données MongoDB de la version 4.0 à la version 4.2.

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

- Arrêt de *VITAM* (*playbook* `ansible-vitam-exploitation/stop_vitam.yml`)

Avertissement : A partir de là, la solution logicielle *VITAM* est arrêtée ; elle ne sera redémarrée qu'au déploiement de la nouvelle version.

- Démarrage des différents cluster mongodb (playbook `ansible-vitam-exploitation/start_mongodb.yml`)
- Upgrade de mongodb en version 4.2 (*playbook* `ansible-vitam-exploitation/migration_mongodb_42.yml`)

4.3.1.1.7 Arrêt de l'ensemble des composants VITAM

Les commandes suivantes sont à lancer depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
stop_vitam.yml --ask-vault-pass
```

A l'issue de l'exécution du *playbook*, les composants *VITAM* ont été arrêtés, afin de ne pas perturber la migration.

4.3.1.2 Montée de version

La montée de version vers la *release* R13 est réalisée par réinstallation de la solution logicielle *VITAM* grâce aux *playbooks* ansible fournis, et selon la procédure d'installation classique décrite dans le *DIN*.

Note : Rappel : avant de procéder à la montée de version, on veillera tout particulièrement à la bonne mise en place des *repositories VITAM* associés à la nouvelle version. Se reporter à la section du *DIN* sur la mise en place des *repositories VITAM*.

Prudence : À l'issue de l'exécution du déploiement de Vitam, les composants *externals* ainsi que les *timers* systemd seront redémarrés. Il est donc recommandé de jouer les étapes de migration suivantes dans la foulée.

4.3.1.3 Etapes de migration

4.3.1.3.1 Migration des données de certificats

La *release* R11 apporte une modification quant à la déclaration des certificats. En effet, un bug empêchait l'intégration dans la solution *VITAM* de certificats possédant un serial number long.

La commande suivante est à exécuter depuis le répertoire `deployment` sur les différents sites hébergeant la solution logicielle *VITAM* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
R10_upgrade_serial_number.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
R10_upgrade_serial_number.yml --ask-vault-pass
```

4.3.1.3.2 Migration des contrats d'entrée

La montée de version vers la *release* R11 requiert une migration de données (contrats d'entrée) suite à une modification sur les droits relatifs aux rattachements. Cette migration s'effectue à l'aide du playbook :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r9_r10_ingestcontracts.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r9_r10_ingestcontracts.yml --ask-vault-pass
```

Le template `upgrade_contracts.js` contient :

```
// Switch to masterdata database
db = db.getSiblingDB('masterdata');

// Update IngestContract
db.IngestContract.find({}).forEach(function(item) {
    if (item.CheckParentLink == "ACTIVE") {
        item.checkParentId = new Array(item.LinkParentId);
    }
    item.CheckParentLink = "AUTHORIZED";
    db.IngestContract.updateOne({_id: item._id}, item);
});
```

4.3.1.3.3 Nettoyage des DIPs depuis les offres

Dans le cadre d'une montée de version vers la *release* R12, il est nécessaire d'appliquer un *playbook* de migration de données à l'issue de réinstallation de la solution logicielle *VITAM*.

La migration s'effectue, uniquement sur le site principal, à l'aide de la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r11_r12_dip_cleanup.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r11_r12_dip_cleanup.yml --ask-vault-pass
```

Avertissement : Selon la volumétrie des données précédement chargées, le *playbook* peut durer quelques minutes.

4.3.1.3.4 Réindexation ES Data

La montée de version vers la *release* R11 requiert une réindexation totale d'ElasticSearch. Cette réindexation s'effectue à l'aide du playbook :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
reindex_es_data.yml --vault-password-file vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
reindex_es_data.yml --ask-vault-pass
```

Note : Ce *playbook* ne supprime pas les anciens indexes pour laisser à l'exploitant le soin de vérifier que la procédure de migration s'est correctement déroulée. A l'issue, la suppression des index devenus inutiles devra être réalisée manuellement.

4.3.1.3.5 Mise à jour des métadonnées de reconstruction (cas d'un site secondaire)

Dans le cadre d'une montée de version vers R13 sur un site secondaire, il est nécessaire d'appliquer un *playbook* de migration de données à l'issue de réinstallation de la solution logicielle *VITAM*.

Le *playbook* ajoute dans les données des collections *Offset* des bases *masterdata*, *logbook* et *metadata* du site secondaire la valeur `"strategy" : "default"`.

La migration s'effectue, uniquement sur le site secondaire, à l'aide de la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r12_r13_upgrade_offset_strategy.yml --vault-password-file
vault_pass.txt
```

ou, si `vault_pass.txt` n'a pas été renseigné :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/
migration_r12_r13_upgrade_offset_strategy.yml --ask-vault-pass
```

4.3.1.3.6 Vérification de la bonne migration des données

Il est recommandé de procéder à un audit de cohérence aléatoire suite à une procédure de montée de version VITAM ou de migration de données. Pour ce faire, se référer au dossier d'exploitation (DEX) de la solution VITAM, section `Audit de cohérence`.

4.3.2 Notes et procédures spécifiques R16

4.3.2.1 Vérification préalable avant la migration

4.3.2.1.1 Gestion des règles de gestion

Dans le cadre d'une correctif concernant la validation stricte des types de règles lors de l'import du référentiel des règles de gestion, il faut impérativement, AVANT la montée de version, vérifier tous les types de règles de gestion existants sur Mongo et ES et les modifier manuellement en cas d'incohérence.

Pour ce faire, il faut s'assurer que tous les types de règles de gestion (`RuleType`) respectent la casse (les majuscules et les minuscules).

Ci-après la liste des valeurs valides autorisées :

- `AppraisalRule`
- `AccessRule`
- `StorageRule`
- `DisseminationRule`
- `ClassificationRule`
- `ReuseRule`
- `HoldRule`

Exemple : `APPRAISALRULE` devrait être `AppraisalRule`

4.3.2.2 Adaptation des sources de déploiement ansible

4.3.2.2.1 Déplacement des paramètres relatif à la gestion des tenants

Dans le cadre de la fonctionnalité introduite en R15 permettant de regrouper les tenants, les paramètres suivants ont étés déplacés du fichier d'inventaire au fichier `group_vars/all/advanced/tenants_vars.yml`

- `vitam_tenant_ids`

- vitam_tenant_admin

Si la montée de version s'effectue à partir d'une R13, il faut supprimer les valeurs précédentes de votre fichier d'inventaire et les reporter dans le fichier tenants_vars.yml en respectant la syntaxe YML adéquate.

Voir aussi :

Se référer à la documentation d'installation pour plus d'informations concernant le fichier environments/group_vars/all/advanced/tenants_vars.yml

4.3.2.3 Procédures à exécuter AVANT la montée de version

4.3.2.3.1 Supprimer les indexes de configuration kibana

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la R16.5 ou supérieur (4.0.5+).

Prudence : Sans cette opération, l'installation kibana est bloquée et arrête l'installation de Vitam

Lors de la montée de version ELK, les indices de configuration kibana : .kibana et .kibana_task_manager persistent avec une version et des informations incorrectes (celles de la version d'avant). Il est nécessaire des les effacer ; autrement la montée de version est bloquée.

Exécutez le playbook suivant :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/remove_old_
↪kibana_indexes.yml --ask-vault-pass
```

Ce playbook clone les indices de configuration (.kibana et .kibana_task_manager) et efface les originaux. Les clones d'indice sont conservés.

La montée de version va recréer ces indices avec les nouvelles configurations relatives au nouvel ELK.

4.3.2.3.2 Réinitialisation de la reconstruction des registres de fond des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration majeure vers une version R16.7+ (4.0.7 ou supérieure). Elle permet la réinitialisation de la reconstruction des registre de fonds sur les sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que les timers de Vitam aient bien été préalablement arrêtés (via le playbook ansible-vitam-exploitation/stop_vitam_timers.yml)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_accession_register_
↪reconstruction.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.2.3.3 Contrôle et nettoyage de journaux du storage engine des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration majeure vers une version R16.7+ (4.0.7 ou supérieure). Elle permet le contrôle et la purge des journaux d'accès et des journaux d'écriture du storage engine des sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_purge_storage_logs_secondary_
↪sites.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.2.4 Procédures à exécuter APRÈS la montée de version

4.3.2.4.1 Arrêt des timers et des accès externes à Vitam

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.3.2.4.2 Recalcul du graph des métadonnées des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration majeure vers une version R16.7+ (4.0.7 ou supérieure). Elle permet le recalcul du graphe des métadonnées sur les sites secondaires

La procédure est à réaliser sur tous les **sites secondaires** de Vitam APRÈS l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_metadata_graph_reconstruction.
↪yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.2.4.3 Redémarrage des timers et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les timers sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪timers.yml --ask-vault-pass
```

(suite sur la page suivante)

(suite de la page précédente)

4.3.2.5 Vérification de la bonne migration des données

4.3.2.5.1 Audit coherence

Il est recommandé de procéder à un audit de cohérence aléatoire suite à une procédure de montée de version VITAM ou de migration de données. Pour ce faire, se référer au dossier d'exploitation (DEX) de la solution VITAM, section `Audit de cohérence`.

4.3.3 Notes et procédures spécifiques V5RC

4.3.3.1 Procédures à exécuter AVANT la montée de version

4.3.3.1.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V5RC

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.3.3.1.2 Supprimer les indexes de configuration kibana

Prudence : Cette opération doit être effectuée AVANT la montée majeure depuis une version R16.4- (R16.4 ou inférieur)

Prudence : Sans cette opération, l'installation kibana est bloquée et arrête l'installation de Vitam

Lors de la montée de version ELK, les indices de configuration kibana : .kibana et .kibana_task_manager persistent avec une version et des informations incorrectes (celles de la version d'avant). Il est nécessaire des les effacer ; autrement la montée de version est bloquée.

Exécutez le playbook suivant :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/remove_old_
↪kibana_indexes.yml --ask-vault-pass
```

Ce playbook clone les indices de configuration (.kibana et .kibana_task_manager) et efface les originaux. Les clones d'indice sont conservés.

La montée de version va recréer ces indices avec les nouvelles configurations relatives au nouvel ELK.

4.3.3.1.3 Réinitialisation de la reconstruction des registres de fond des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration majeure depuis une version R16.6-(4.0.6 ou inférieure). Elle permet la réinitialisation de la reconstruction des registre de fonds sur les sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que les timers de Vitam aient bien été préalablement arrêtés (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_accession_register_
↪reconstruction.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.3.1.4 Contrôle et nettoyage de journaux du storage engine des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration majeure depuis une version R16.6-(4.0.6 ou inférieure). Elle permet le contrôle et la purge des journaux d'accès et des journaux d'écriture du storage engine des sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_purge_storage_logs_secondary_
↪sites.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.3.1.5 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V5RC

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.3.2 Procédures à exécuter APRÈS la montée de version

4.3.3.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée IMMÉDIATEMENT APRÈS la montée de version vers la V5RC

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.3.3.2.2 Migration des unités archivistiques

Prudence : Cette migration doit être effectuée APRÈS la montée de version V5RC mais avant la réouverture du service aux utilisateurs.

Cette migration de données consiste à :

- Supprimer le champ `_us_sp`.
- Rendre inactive l'indexation des champs dynamiques créés au niveau des règles de gestion héritées au niveau de la propriété `endDates`.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

Lancez la migration via la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_v5rc.
↪yml --ask-vault-pass
```

Après le passage du script de migration, il faut procéder à la réindexation de toutes les unités archivistiques :

```
  ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --tags unit --ask-vault-pass

..
```

4.3.3.2.3 Migrations offres Swift V2 & V3 en cas de présence d'objets très volumineux (4Go+)

Prudence : Cette procédure doit être lancée une seule fois, et pour chaque offre Swift V2/V3, APRÈS upgrade Vitam.

Si vous disposez d'une instance R16.3 ou inférieur (4.0.3-), et que vous utilisez des offres Swift V2/V3 (providers openstack-swift-v2 et/ou openstack-swift-v3), il est nécessaire de procéder à une migration des données :

```
$ ansible-playbook ansible-vitam-migration/migration_swift_v2_and_v3.yml -i␣
↪environments/hosts.{env} --ask-vault-pass

# Confirm playbook execution
# Enter swift offer id (ex offer-swift-1)
# Select migration mode
# > Enter '0' for analysis only mode : This mode will only log anomalies (in offer␣
↪technical logs), no update will be proceeded
# > Enter '1' to fix inconsistencies : This mode will update swift objects to fix␣
↪inconsistencies. However, this does not prune objects (delete partially written or␣
↪eliminated objects segments to free space).
# > Enter '2' to fix inconsistencies and purge all deleted objects segments to free␣
↪storage space.
# Reconfirm playbook execution
```

Il est recommandé de lancer la procédure en mode 0 (analyse seule) et de vérifier les erreurs de cohérence dans les logs.

Seul les offres Swift V2/V3 avec des objets volumineux (>= 4Go) nécessitent une migration. Un exemple d'incohérence journalisés dans les logs (/vitam/log/offers) est donnée ici :

```
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪old segment names [aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2,␣
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1]. Run migration script with fix␣
↪inconsistencies mode to prune container.
INCONSISTENCY FOUND : Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq has␣
↪missing metadata. Run migration script with fix inconsistencies mode enabled to set␣
↪object metadata.
```

Si la détection des anomalies est terminée en succès, et que des anomalies sont trouvées, il est recommandé de lancer le mode 1 (correction des anomalies). Les migrations de données sont également journalisées dans les logs (/vitam/log/offers) :

```
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/2 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000002
Renaming segment env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/1 to env_2_object/
↪aeaaaaaaaagbcaacaamboal2tk643jqaaaaq/00000001
Object env_2_object/aeaaaaaaaagbcaacaamboal2tk643jqaaaaq migrated successfully.␣
↪Digest:␣
↪8959ea1290aa064a3c64d332f31e049bd4f9d4e95bebe0b46d38613bb079761d52c865dce64c88fd7e02313d340f9a2f8c0
```

Si des problèmes de cohérence de type « Orphan large object segments » persistent

```
INCONSISTENCY FOUND : Orphan large object segments [...] without parent object␣
↪manifest: env_2_object/aeaaaaaaaagbcaacaamboal2tk7dzmiaaaaq. Eliminated object?␣
↪Incomplete write? Run migration script with delete mode to prune container.
```

Dans ce cas, il est recommandé de vérifier préalablement que les objets concernés n'existent pas sur les autres offres (mêmes container & objectName). Si les objets n'existent pas dans les autres offres, il s'agit alors de reliquats d'objets non complètement éliminés. Le lancement du mode 2 (correction des anomalies + purge des objets) est à réaliser. Dans le cas contraire (cas où l'objet existe dans les autres offres), il faudra envisager la « Procédure de resynchronisation ciblée d'une offre » décrite dans la Documentation d'EXploitation (DEX) de Vitam pour synchroniser l'offre Swift pour les éléments concernés.

4.3.3.2.4 Recalcul du graph des métadonnées des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de migration majeure depuis une version R16.6-(4.0.6 ou inférieure). Elle permet le recalcul du graphe des métadonnées sur les sites secondaires

La procédure est à réaliser sur tous les **sites secondaires** de Vitam APRÈS l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_metadata_graph_reconstruction.
↪yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.3.2.5 Redémarrage des timers et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les timers sur **tous les sites** :

```
  ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
  ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪vitam_timers.yml --ask-vault-pass

..
```

4.3.4 Notes et procédures spécifiques V5

Prudence : Pour une montée de version depuis la R16 de Vitam, veuillez appliquer les procédures spécifiques de la V5RC en complément des procédures suivantes. Pour une montée de version depuis la V5RC, vous pouvez appliquer la procédure suivante directement.

4.3.4.1 Adaptation des sources de déploiement ansible

4.3.4.1.1 Classement des offres dans une stratégie

Dans une stratégie de stockage, chaque offre renseignée déclare un ordre de lecture. Cet ordre est manifeste à travers la propriété `rank`. Il est obligatoire de la renseigner dans chaque offre utilisée. La lecture depuis les offres se fait selon un ordre ascendant en se basant sur cette propriété. Ci-dessous un exemple de déclaration de stratégie de stockage et ses offres, dans le fichier de configuration `deployment/environments/group_vars/all/offer_opts.yml` :

```
vitam_strategy:
- name: offer-1
  referent: true
  rank: 10
- name: offer-2
  referent: false
```

(suite sur la page suivante)

(suite de la page précédente)

```
  rank: 20
- name: offer-3
  referent: false
  rank: 30

vitam_offers:
    offer-1:
        provider: filesystem
    offer-2:
        provider: filesystem
    offer-3:
        provider: filesystem
```

4.3.4.1.2 Ajout d'un nouveau module VITAM : Module de collecte

Prudence : À préparer dans les sources de déploiement AVANT le déploiement de la V5. Ce module est optionnel, si vous ne souhaitez pas l'activer, laissez les groupes de hosts vides dans le fichier d'inventaire.

Ce module a pour but de faciliter l'intégration d'archives dans Vitam via une API constructive de SIP.

Le module de *collect* nécessite la configuration et l'ajout : - d'une autre instance de metadata appelée *metadata-collect* - d'une autre instance de workspace appelée *workspace-collect*

Pour la mise en oeuvre de cette nouvelle application, veuillez éditer les paramètres suivants :

- Ajout des groupes de hosts du module de collect à votre fichier d'inventaire (cf. fichier d'inventaire d'exemple : `environments/hosts.example`).

```
[zone_applicative:children]
hosts_collect
hosts_metadata_collect
hosts_workspace_collect

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


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


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

- Ajout des bases mongo pour le module de collect dans le fichier `environments/group_vars/all/vault-vitam.yml` :

Prudence : Pensez à éditer les password avec des passwords sécurisés.

```
mongodb:
  mongo-data:
```

(suite sur la page suivante)

(suite de la page précédente)

```
    collect:
      user: collect
      password: change_it_m39XvRQWixyDX566
    metadataCollect:
      user: metadata-collect
      password: change_it_37b97KVaDV8YbCwt
```

- Ajouter des mots de passe des keystores du module de collecte le fichier `environments/group_vars/all/vault-keystores.yml` :

```
keystores:
  server:
```

  - collect : changeit6kQ16eyDYAoPS9fy

    client_external :

  - collect : changeitz6xZe5gDu7nhDZA12

- Création de certificats dédiés au module de collecte
  - Créer un certificat client et un certificat serveur dédiés au module de collecte à l'aide de votre `PKI` et le mettre dans les chemins attendus (`environments/certs/client-external/clients/collect/` et `environments/certs/server/hosts/{hosts}`).
    - Dans le cas de l'utilisation de la PKI de test de Vitam, vous pouvez simplement re-générer de nouveaux certificats à l'aide de la commande : `./pki/scripts/generate_certs.sh <fichier_inventaire>`
  - Re-générer les stores : `./generate_stores.sh`
  - Ajouter le contexte de sécurité pour le module de collecte dans le fichier `environments/group_vars/all/vitam_security.yml` :

```
admin_context_certs:
  - "collect/collect.crt"
```

4.3.4.2 Procédures à exécuter AVANT la montée de version

4.3.4.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V5

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.3.4.2.2 Réinitialisation de la reconstruction des registres de fond des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de :

- migration majeure depuis une version R16.6- (4.0.6 ou inférieure)
- migration majeure depuis une version v5.rc.3- (v5.rc.3 ou inférieure)

Cette procédure permet la réinitialisation de la reconstruction des registre de fonds sur les sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que les timers de Vitam aient bien été préalablement arrêtés (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_accession_register_
↪reconstruction.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.4.2.3 Contrôle et nettoyage de journaux du storage engine des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de :

- migration majeure depuis une version R16.6- (4.0.6 ou inférieure)
- migration majeure depuis une version v5.rc.3- (v5.rc.3 ou inférieure)

Cette procédure permet le contrôle et la purge des journaux d'accès et des journaux d'écriture du storage engine des sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_purge_storage_logs_secondary_
↪sites.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.4.2.4 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V5

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.4.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

Prudence : Sous Debian, si vous appliquez la montée de version depuis la V5.RC, vous devrez rajouter le paramètre `-e force_vitam_version=5.x` (exemple : `-e force_vitam_version=5.4`) aux commandes suivantes. Sinon les packages vitam ne seront pas correctement mis à jour. En effet, Debian considère que 5.rc.X > 5.X.

4.3.4.3.1 Lancement du master playbook vitam

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-pass
```

4.3.4.3.2 Lancement du master playbook extra

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

4.3.4.4 Procédures à exécuter APRÈS la montée de version

4.3.4.4.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée IMMÉDIATEMENT APRÈS la montée de version vers la V5

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.3.4.4.2 Migration des unités archivistiques

Prudence : Cette migration doit être effectuée APRÈS la montée de version V5 mais avant la réouverture du service aux utilisateurs.

Cette migration de données consiste à ajouter les champs `_acd` (date de création approximative) et `_aud` (date de modification approximative) dans la collection Unit.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

- Migration des unités archivistiques sur mongo-data (le playbook va stopper les externals et les timers de Vitam avant de procéder à la migration) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_v5.
↪yml --ask-vault-pass
```

Après le passage du script de migration, il faut procéder à la réindexation de toutes les unités archivistiques :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --tags unit --ask-vault-pass
```

4.3.4.4.3 Mise à jour des certificats

Cette migration de données consiste à mettre à jour le champ `ExpirationDate` pour les anciens certificats existants dans la base de donnée.

Elle est réalisée en exécutant les commandes suivantes sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_v5_
↪certificate.yml --ask-vault-pass
```

4.3.4.4.4 Migration des registres de fonds en détails

Prudence : Cette migration doit être effectuée APRÈS la montée de version V5 mais avant la réouverture du service aux utilisateurs.

Suite à l'ajout des nouvelles propriétés `Comment` (Commentaire) et `obIdIn` (Identifiant du message) au niveau de la collection `AccessionRegisterDetail`, il faut lancer une migration sur les anciennes données.

Exécutez la commande suivante uniquement sur **le site primaire** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪accession_register_details_v5.yml --vault-password-file vault_pass.txt
```

4.3.4.4.5 Recalcul du graph des métadonnées des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de :

- migration majeure depuis une version R16.6- (4.0.6 ou inférieure)
- migration majeure depuis une version v5.rc.3- (v5.rc.3 ou inférieure)

Cette procédure permet le recalcul du graphe des métadonnées sur les sites secondaires

La procédure est à réaliser sur tous les **sites secondaires** de Vitam APRÈS l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_metadata_graph_reconstruction.
↪yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.4.4.6 Redémarrage des timers et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les timers sur **tous les sites** :

```
  ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
  ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪vitam_timers.yml --ask-vault-pass

..
```

4.3.5 Notes et procédures spécifiques V6RC

Prudence : Pour une montée de version depuis la version R16 de Vitam, veuillez appliquer les procédures spécifiques de la V5RC et de la V5 en complément des procédures suivantes. Pour une montée de version depuis la version V5RC de Vitam, veuillez appliquer les procédures spécifiques de la V5 en complément des procédures suivantes. Pour une montée de version depuis la V5, vous pouvez appliquer la procédure suivante directement.

4.3.5.1 Adaptation des sources de déploiement ansible

4.3.5.1.1 Réorganisation des variables

Afin de simplifier la préparation des sources de déploiement, les fichiers ont étés répartis dans 2 sous répertoires `main` et `advanced`. Le répertoire `main` est le répertoire principal qui nécessite une attention particulière à la préparation des sources de déploiement.

Afin de vous adapter à cette nouvelle organisation, vous devez redispatcher les fichiers de configuration initialement sous `environments/group_vars/all/` dans les 2 sous répertoires `environments/group_vars/all/{main,advanced}/`.

4.3.5.1.2 Ajout du nouveau composant scheduler

Prudence : À préparer dans les sources de déploiement AVANT le déploiement de la V6RC. Ce nouveau module est obligatoire et vient en remplacement des timers systemd pour l'ordonnancement des tâches planifiées dans Vitam.

- Ajout du groupe `[hosts_scheduler]` à votre fichier d'inventaire (cf. fichier d'inventaire d'exemple : `environments/hosts.example`).

```
[zone_applicative:children]
hosts_scheduler

[hosts_scheduler]
# TODO: Put here servers where this service will be deployed : scheduler
```

- Ajout des bases mongo pour le scheduler dans le fichier `environments/group_vars/all/main/vault-vitam.yml` :

> **Prudence :** Pensez à éditer les password avec des passwords sécurisés.

```
mongodb:
  mongo-data:
    scheduler:
      user: scheduler
      password: change_it_xyz
```

- Personnaliser les paramètres jvm pour le scheduler dans le fichier de configuration `environments/group_vars/all/main/jvm_opts.yml`.

4.3.5.2 Procédures à exécuter AVANT la montée de version

4.3.5.2.1 Arrêt des timers et des accès externes à Vitam

> **Prudence :** Cette opération doit être effectuée AVANT la montée de version vers la V6RC

> **Prudence :** Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass
```

4.3.5.2.2 Mise à jour des dépôts (YUM/APT)

> **Prudence :** Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable `vitam_repositories` sous `environments/group_vars/all/main/repositories.yml` afin de renseigner les dépôts à la version cible.

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.5.2.3 Montée de version vers mongo 4.4

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V6RC

Prudence : Sans cette opération, la montée de version d'une version existante vers une V6RC sera bloquée au démarrage des instances mongod par une incompatibilité.

Exécutez le playbook suivant :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_44.yml --ask-vault-pass
```

Ce playbook effectue la montée de version de mongodb d'une version 4.2 vers une version 4.4 selon la procédure indiquée dans la documentation Mongodb. Cette procédure n'a pas été testée avec une version mongodb inférieure à 4.2.

4.3.5.2.4 Montée de version vers mongo 5.0

Prudence : Cette montée de version doit être effectuée AVANT la montée de version V6RC de vitam et après la montée de version en mongodb 4.4 ci-dessus.

Exécutez le playbook suivant :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_50.yml --ask-vault-pass
```

Ce playbook change le « Read and write Concern » des replicaset par reconfiguration, il désinstalle et réinstalle les binaires et il change également le paramètre « SetFeatureCompatibility » à 5.0.

Une fois ces montées de version de Mongodb réalisées la montée de version Vitam classique peut être réalisée.

4.3.5.2.5 Réinitialisation de la reconstruction des registres de fond des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de :

- migration majeure depuis une version R16.6- (4.0.6 ou inférieure)
- migration majeure depuis une version v5.rc.3- (v5.rc.3 ou inférieure)
- migration majeure depuis une version v5.0.

Cette procédure permet la réinitialisation de la reconstruction des registre de fonds sur les sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que les timers de Vitam aient bien été préalablement arrêtés (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_accession_register_
↪reconstruction.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.5.2.6 Contrôle et nettoyage de journaux du storage engine des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de :

- migration majeure depuis une version R16.6- (4.0.6 ou inférieure)
- migration majeure depuis une version v5.rc.3- (v5.rc.3 ou inférieure)
- migration majeure depuis une version v5.0.

Cette procédure permet le contrôle et la purge des journaux d'accès et des journaux d'écriture du storage engine des sites secondaires.

La procédure est à réaliser sur tous les **sites secondaires** de Vitam AVANT l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_purge_storage_logs_secondary_
↪sites.yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.5.2.7 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V6RC

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.5.2.8 Nettoyage des fichiers timers, services et conf suite à la migration vers le scheduler

Prudence : Cette étape doit être effectuée AVANT la montée de version V6RC et sur un Vitam éteint.

Prudence : Cette opération doit être effectuée avec les sources de déploiement de la nouvelle version.

Executez le playbook suivant :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/remove_old_
↪files_for_scheduler_migration.yml --ask-vault-pass
```

Ce playbook supprime les fichiers .service, .sh, .timers et .conf suite au passage vers le scheduler Quartz sur les hosts concernés.

4.3.5.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

4.3.5.3.1 Lancement du master playbook vitam

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-pass
```

4.3.5.3.2 Lancement du master playbook extra

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

4.3.5.4 Procédures à exécuter APRÈS la montée de version

4.3.5.4.1 Arrêt des jobs Vitam et des accès externes à Vitam

Prudence : Cette opération doit être effectuée IMMÉDIATEMENT APRÈS la montée de version vers la V6RC

Les jobs Vitam et les services externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.5.4.2 Migration des groupes d'objets

Prudence : Cette migration doit être effectuée APRÈS la montée de version V6RC mais avant la réouverture du service aux utilisateurs.

Cette migration de données consiste à ajouter les champs `_acd` (date de création approximative) et `_aud` (date de modification approximative) dans la collection ObjectGroup.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

- Migration des unités archivistiques sur mongo-data (le playbook va stopper les externals avant de procéder à la migration) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_v6rc.
↪yml --ask-vault-pass
```

Après le passage du script de migration, il faut procéder à la réindexation de toutes les groupes d'objets :

```
  ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --tags objectgroup --ask-vault-pass

..
```

4.3.5.4.3 Recalcul du graph des métadonnées des sites secondaires

Prudence : Cette procédure doit être exécutée uniquement en cas de :

- migration majeure depuis une version R16.6- (4.0.6 ou inférieure)
- migration majeure depuis une version v5.rc.3- (v5.rc.3 ou inférieure)
- migration majeure depuis une version v5.0.

Cette procédure permet le recalcul du graphe des métadonnées sur les sites secondaires

La procédure est à réaliser sur tous les **sites secondaires** de Vitam APRÈS l'installation de la nouvelle version :

- S'assurer que Vitam soit bien préalablement arrêté (via le playbook `ansible-vitam-exploitation/stop_vitam_timers.yml`)
- Exécuter le playbook :

```
ansible-playbook ansible-vitam-migration/migration_metadata_graph_reconstruction.
↪yml -i environments/hosts.{env} --ask-vault-pass
```

4.3.5.4.4 Réindexation des référentiels sur elasticsearch

Cette migration de données consiste à mettre à jour le modèle d'indexation des référentiels sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --ask-vault-pass --tags "securityprofile, context, ontology,␣
↪ingestcontract, agencies, accessionregisterdetail, archiveunitprofile,␣
↪accessionregistersummary, accesscontract, fileformat, filerules, profile, ↪
↪preservationscenario, managementcontract"
```

(suite sur la page suivante)

(suite de la page précédente)

```
```

4.3.5.4.5 Migration des mappings elasticsearch pour les métadonnées

Cette migration de données consiste à mettre à jour le modèle d'indexation des métadonnées sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

4.3.5.4.6 Redémarrage des Jobs Vitam et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les jobs Vitam sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduler.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.6 Notes et procédures spécifiques V6

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : R16 -> V5RC -> V5 -> V6RC.

4.3.6.1 Adaptation des sources de déploiement ansible

4.3.6.1.1 Mise à jour de l'architecture du module de collecte

- Ajouter les composants suivants à votre fichier d'inventaire * `[hosts_collect_external]` dans la `[zone_access:children]` * `[hosts_collect_internal]` dans la `[zone_applicative:children]`
- Modifier le fichier `environments/group_vars/all/main/vault-keystores.yml`

```
keystores:
  server:
```

  - collect : changeit6kQ16eyDYAoPS9fy
  - collect_external : changeit6kQ16eyDYAoPS9fy

    client_external :

  - collect : changeitz6xZe5gDu7nhDZA12
  - collect_external : changeitz6xZe5gDu7nhDZA12

- Création de certificats dédiés au module de collecte

- Supprimer, si ils existent, les certificats de l'ancien module de collecte `rm -rf environments/certs/client-external/clients/collect/ environments/certs/server/hosts/*/collect.{crt,key}`.
- Créer un certificat client et un certificat serveur dédiés au module de collecte à l'aide de votre `PKI` et le mettre dans les chemins attendus (`environments/certs/client-external/clients/collect-external/` et `environments/certs/server/hosts/{hosts}`).
- Dans le cas de l'utilisation de la PKI de test de Vitam, vous pouvez simplement re-générer de nouveaux certificats à l'aide de la commande : `./pki/scripts/generate_certs.sh <fichier_inventaire>`
- Re-générer les stores : `./generate_stores.sh`
- Ajouter le contexte de sécurité pour le module de collecte dans le fichier `environments/group_vars/all/advanced/vitam_security.yml` :

```
admin_context_certs:
```

  - « collect/collect.crt »
  - « collect-external/collect-external.crt »

- Ne pas oublier les paramètres de configuration associés aux jvms de ces nouveaux composants dans le fichier `environments/group_vars/all/main/jvm_opts.yml`

```
collect_internal:
    jvm_opts:
        # memory: "-Xms512m -Xmx512m"
        # gc: ""
        # java: ""
collect_external:
    jvm_opts:
        # memory: "-Xms512m -Xmx512m"
        # gc: ""
        # java: ""
```

4.3.6.1.2 Modification de l'indexation par défaut dans elasticsearch des indexes de collecte

Prudence : Attention, ce changement d'indexation vous fera perdre les données en cours dans le module de collecte. Il est conseillé de terminer et de purger les transactions en cours avant de procéder à la montée de version. Si malgré tout, vous souhaitez conserver l'indexation actuelle, il vous faudra supprimer les lignes de la variable `collect_grouped_tenants`.

Initialement, il n'était pas possible de définir une configuration spécifique lié à l'indexation des unit & objectgroup pour le module de collecte.

Ainsi, la mécanique de personnalisation des indexes Vitam a été mise en oeuvre pour les indexes du module de collecte. Par défaut, la configuration ainsi proposée regroupe l'ensemble des tenants dans un indexe unique pour chacun des index unit & objectgroup.

Le module de collecte a pour vocation de sas tampon de transfert, il n'est donc pas nécessaire d'allouer un shard par tenant.

La configuration par défaut permet de limiter l'empreinte mémoire et l'utilisation du cluster elasticsearch-data. En fonction de votre besoin, vous pouvez rajouter des shards ou bien découper sur des indexes dédiés certains tenants de Vitam.

Dans le fichier de configuration suivant : `environments/group_vars/all/main/main.yml`

```
vitam_elasticsearch_tenant_indexation:
  default_config:
    # Default settings for collect_unit indexes
    collect_unit:
      number_of_shards: 1
      number_of_replicas: 2
    # Default settings for collect_objectgroup indexes
    collect_objectgroup:
      number_of_shards: 1
      number_of_replicas: 2

  collect_grouped_tenants:
  - name: 'all'
    # Group all tenants for collect's indexes (collect_unit & collect_objectgroup)
    tenants: "{{ vitam_tenant_ids }}"
```

4.3.6.2 Procédures à exécuter AVANT la montée de version

4.3.6.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V6

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass

# Si Version < V6RC:
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪timers.yml --ask-vault-pass

# Si Version >= V6RC:
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.6.2.2 Mise à jour des dépôts (YUM/APT)

Prudence : Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable `vitam_repositories` sous `environments/group_vars/all/main/repositories.yml` afin de renseigner les dépôts à la version cible.

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.6.2.3 Nettoyage des anciens fichiers du module de collecte suite au changement d'architecture

Prudence : Cette étape doit être effectuée AVANT la montée de version V6 de vitam et seulement si la V6RC ou V5 a été déployée avec le module de collecte.

Prudence : Attention, cette procédure va supprimer l'ensemble des éléments stockés dans la partie externe du module de collecte. Veuillez vous assurer que les transactions en cours sont bien purgées avant de procéder à la montée de version.

Exécutez le playbook suivant à partir de l'ansiblerie de la V6 **sur le site primaire** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/remove_old_
↪collect.yml --ask-vault-pass
```

Ce playbook supprime les anciens éléments suite aux modifications de l'architecture du module de collecte sur les machines `[hosts_collect]`.

Après exécution de ce playbook, vous pouvez supprimer de votre inventaire le groupe `[hosts_collect]`.

4.3.6.2.4 Montée de version mineure de mongo 5.0.13 -> 5.0.14

Prudence : Cette montée de version doit être effectuée AVANT la montée de version V6 de Vitam et après l'arrêt des Timers et des externals.

Prudence : Cette opération doit être effectuée après avoir mis à jour les dépôts en V6.

Exécutez le playbook suivant à partir de l'ansiblerie de la V6 **sur tous les sites** :

```
# Mise à jour mongo
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_50.yml --ask-vault-pass
```

4.3.6.2.5 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V6

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.6.2.6 Duplication des packages lors de la mise à jour

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V6

Prudence : Cette opération doit être effectuée uniquement pour les systèmes d'exploitation à base de RPM (CentOS 7 & AlmaLinux 9) sur un vitam éteint.

Un bug a été introduit dans la construction des packages RPM à partir de la V6RC. Si vous effectuez une montée de version depuis la V6RC (v6.rc.1 ou inférieure), vous devrez appliquer la commande suivante pour supprimer les précédents packages. Sinon lors de de la mise à jour, le package précédent ne sera pas désinstallé conduisant à la présence de plusieurs versions du même package.

Exemple d'erreur :

```
/var/tmp/rpm-tmp.fGwZMk: ligne 1 : fg: pas de contrôle de tâche
erreur : %preun(vitam-offer-6.2-1.noarch) scriptlet échoué, état de sortie 1
Error in PREUN scriptlet in rpm package vitam-offer
erreur : vitam-offer-6.2-1.noarch: effacer échoué
```

Voici la commande à exécuter pour permettre la désinstallation des packages de la version précédente :

```
ansible vitam --ask-vault-pass -v -a "yum --setopt=tsflags=noscripts remove -y vitam-
↪*.noarch" -i environments/<inventaire>
```

4.3.6.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

Prudence : Sous Debian, si vous appliquez la montée de version depuis la V6.RC, vous devrez rajouter le paramètre `-e force_vitam_version=6.x` (exemple : `-e force_vitam_version=6.2`) aux commandes suivantes. Sinon les packages vitam ne seront pas correctement mis à jour. En effet, Debian considère que 6.rc.X > 6.X.

4.3.6.3.1 Lancement du master playbook vitam

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-pass
```

4.3.6.3.2 Lancement du master playbook extra

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

4.3.6.4 Procédures à exécuter APRÈS la montée de version

4.3.6.4.1 Arrêt des jobs Vitam et des accès externes à Vitam

Prudence : Cette opération doit être effectuée IMMÉDIATEMENT APRÈS la montée de version vers la V6

Les jobs Vitam et les services externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.6.4.2 Réindexation des référentiels sur elasticsearch

Cette migration de données consiste à mettre à jour le modèle d'indexation des référentiels sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --ask-vault-pass --tags "securityprofile, context, ontology,␣
↪ingestcontract, agencies, accessionregisterdetail, archiveunitprofile,␣
↪accessionregistersummary, accesscontract, fileformat, filerules, profile, griffin,␣
↪preservationscenario, managementcontract"
```

4.3.6.4.3 Migration des mappings elasticsearch pour les métadonnées

Cette migration de données consiste à mettre à jour le modèle d'indexation des métadonnées sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

4.3.6.4.4 Redémarrage des Jobs Vitam et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les jobs Vitam sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduler.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.7 Notes et procédures spécifiques V7

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : R16 -> V5RC -> V5 -> V6RC -> V6.

4.3.7.1 Procédures à exécuter AVANT la montée de version

4.3.7.1.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V7

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.7.1.2 Mise à jour des dépôts (YUM/APT)

Prudence : Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable `vitam_repositories` sous `environments/group_vars/all/main/repositories.yml` afin de renseigner les dépôts à la version cible.

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.7.1.3 Workaround pour la mise à jour de MongoDB en cas de saut de versions

Si vous effectuez une montée de version depuis une version antérieure à la V5.1 (ou inférieure) ou à la V6.rc.1 (ou inférieure) vers la V7.0.X (ou supérieure), vous devrez appliquer le playbook suivant avant de procéder à l'étape `Montée de version vers mongo 4.4` de la procédure de migration vers la V6RC.

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/pre_migration_
↪mongodb_44.yml --tags update_systemd_configuration --ask-vault-pass
```

4.3.7.1.4 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V7

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.7.1.5 Duplication des packages lors de la mise à jour

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V7.0

Prudence : Cette opération doit être effectuée uniquement pour les systèmes d'exploitation à base de RPM (CentOS 7 & AlmaLinux 9) sur un vitam éteint.

Un bug a été introduit dans la construction des packages RPM à partir de la V6RC. Si vous effectuez une montée de version depuis la V6RC (v6.rc.1 ou inférieure) ou de la V6 (v6.2 ou inférieure), vous devrez appliquer la commande suivante pour supprimer les précédents packages. Sinon lors de de la mise à jour, le package précédent ne sera pas désinstallé conduisant à la présence de plusieurs versions du même package.

Exemple d'erreur :

```
/var/tmp/rpm-tmp.fGwZMk: ligne 1 : fg: pas de contrôle de tâche
erreur : %preun(vitam-offer-6.2-1.noarch) scriptlet échoué, état de sortie 1
Error in PREUN scriptlet in rpm package vitam-offer
erreur : vitam-offer-6.2-1.noarch: effacer échoué
```

Voici la commande à exécuter pour permettre la désinstallation des packages de la version précédente :

```
ansible vitam --ask-vault-pass -v -a "yum --setopt=tsflags=noscripts remove -y vitam-
↪*.noarch" -i environments/<inventaire>
```

4.3.7.2 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

4.3.7.2.1 Lancement du master playbook vitam

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-pass
```

4.3.7.2.2 Lancement du master playbook extra

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

4.3.7.3 Procédures à exécuter APRÈS la montée de version

4.3.7.3.1 Arrêt des jobs Vitam et des accès externes à Vitam

Prudence : Cette opération doit être effectuée IMMÉDIATEMENT APRÈS la montée de version vers la V7

Les jobs Vitam et les services externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.7.3.2 Réindexation des référentiels sur elasticsearch

Cette migration de données consiste à mettre à jour le modèle d'indexation des référentiels sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --ask-vault-pass --tags "securityprofile, context, ontology,␣
↪ingestcontract, agencies, accessionregisterdetail, archiveunitprofile,␣
↪accessionregistersummary, accesscontract, fileformat, filerules, profile, griffin,␣
↪preservationscenario, managementcontract"
```

4.3.7.3.3 Migration des mappings elasticsearch pour les métadonnées

Cette migration de données consiste à mettre à jour le modèle d'indexation des métadonnées sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

4.3.7.3.4 Redémarrage des Jobs Vitam et des accès externes à Vitam

La montée de version est maintenant terminée, vous pouvez réactiver les services externals ainsi que les jobs Vitam sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduler.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.8 Notes et procédures spécifiques V7.1

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : V5 -> V6RC -> V6 -> V7.0.

4.3.8.1 Adaptation des sources de déploiement ansible

4.3.8.1.1 Modification de la personnalisation des rôles des clusters Elasticsearch

Elasticsearch ayant fait évoluer la définition des paramètres des rôles pour chacun des noeuds, si vous avez personnalisés les rôles associés à vos serveurs pour chacun de vos clusters Elasticsearch (log ou data), les paramètres de configuration au niveau de l'ansiblerie doivent être adaptés.

Les paramètres suivants permettant la personnalisation des rôles ont étés retirés :

```
is_master=true is_data=true is_ingest=false
```

Vous devrez les remplacer par la nouvelle convention sous forme de tableau pour chacun des hosts personnalisés :

```
elasticsearch_roles=[master, data]
```

Cette nouvelle convention offre plus de souplesse dans la définition de l'ensemble des rôles possible pour un cluster Elasticsearch (cf. Documentation officielle - Node Roles [11]). Attention une mauvaise configuration de ces paramètres avancés pourrait mener à une incohérence de la configuration rendant vos clusters non fonctionnels.

4.3.8.1.2 Modifications du rôle curator

Le rôle curator permet la rotation des indexes du cluster elasticsearch-log. Il a pour but de permettre de fermer et supprimer au bout d'un certain temps les anciens indexes afin de limiter l'empreinte mémoire associée.

Dans un contexte d'environnement de production, les logs applicatif vitam son stockés par défaut pour une durée de 365j et les accesslogs pour une durée de 180j.

Les paramètres `curator.log.*` ont évolués en `curator.indices.*`.

11. https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html#node-roles

Ancienne configuration par défaut :

curator :

log :

metrics : close : 7 delete : 30

logstash : close : 10 delete : 30

metricbeat : close : 5 delete : 10

packetbeat : close : 5 delete : 10

Nouvelle configuration par défaut :

curator : ## Pour personnaliser les dates d'exécution des actions close/delete # actions : # close : # calendar : "--* 00 :10 :00" # delete : # calendar : "--* 00 :20 :00" indices :

vitam : close : 30 delete : 365

access : close : 30 delete : 180

system : close : 7 delete : 30

metricbeat : close : 5 delete : 10

packetbeat : close : 5 delete : 10

Exemple d'index personnel avec préfixe personnalisé ## Sans le paramètre prefix, les actions seront exécutées sur les indexes nommés logstash-mycustom* ## Avec le paramètre prefix défini, les actions seront exécutées sur les indexes nommés myprefix* # mycustom : # prefix : myprefix # close : 15 # delete : 30

Le listing des éléments `curator.indices.<index_name>` permet de créer les fichiers de configuration adaptés à la mise en place des rotations de chacun des indexes.

Chacun des nouveaux indices sera préfixé selon la convention de nommage `logstash-<index_name>`.

Si vous aviez personnalisés le prefix des indexes à gérer par curator à l'aide de la variable `curator.log.prefix` (par défaut "logstash-*"). Vous devez maintenant la modifier à l'aide du paramètre `curator.indices.<index_name>.prefix`.

4.3.8.1.3 Ajout de nouveaux exporters

Ces nouveaux exporters permettent de collecter des métriques afin d'afficher ces éléments via Grafana dans les dashboards associés.

Blackbox Exporter permet de collecter des métriques sur le statut des URLs listées dans `prometheus.blackbox_exporter.targets`.

MongoDB Exporter permet de collecter des métriques sur les différents clusters mongo-vitam.

Il est possible de les désactiver via `prometheus.blackbox_exporter.enabled: false` ou `prometheus.mongodb_exporter.enabled: false`

Ouvrir les ports suivants de Prometheus -> Exporters pour permettre la collecte des métriques :

- Blackbox Exporter : `prometheus.blackbox_exporter.port: 9115`
- MongoC Exporter : `prometheus.mongodb_exporter.port_mongoc: 9216`
- MongoD Exporter : `prometheus.mongodb_exporter.port_mongod: 9217`

4.3.8.1.4 Modification de la durée de rétention des logs par défaut

Par défaut on conserve maintenant 365j de logs (accesslogs & applicatif) dans une limite de 5GB (par composant). De plus, nous avons réduit la quantité de logs gc de `32*64m=2048m` à `8*32m=256m`.

Il est toujours possible de personnaliser ce paramétrage par défaut via les variables suivantes :

- Pour les gc : * `vitam_defaults.jvm_opts.gc` ou par composant en utilisant la variable `vitam.<composant>.jvm_opts.gc`.
- Pour les accesslogs : * `vitam_defaults.access_retention_days: 365` ou par composant en utilisant la variable `vitam.<composant>.access_retention_days: 365`. * `vitam_defaults.access_total_size_cap: 5GB` ou par composant en utilisant la variable `vitam.<composant>.access_total_size_cap: 5GB`.
- Pour les logs applicatifs : * `vitam_defaults.logback_total_size_cap.file.history_days: 365` ou par composant en utilisant la variable `vitam.<composant>.logback_total_size_cap.file.history_days: 365`. * `vitam_defaults.logback_total_size_cap.file.totalsize: 5GB` ou par composant en utilisant la variable `vitam.<composant>.logback_total_size_cap.file.totalsize: 5GB`.

4.3.8.1.5 Modification de la méthodologie de concentration des logs

Un nouveau composant applicatif (Filebeat) permettant de collecter les logs dans le cluster elasticsearch-log a été ajouté.

La méthode de collecte via rsyslog et syslog-ng sera donc dépréciée dans les futures releases.

Vous pouvez continuer à utiliser les précédentes méthodes de concentation de logs via la configuration du paramètre `syslog.name: filebeat` (rsyslog, syslog-ng).

4.3.8.1.6 Nouveau mode de déploiement en container (beta)

Prudence : Attention, à ne pas utiliser en production.

Pour permettre le déploiement en mode conteneur de Vitam, vous devez configurer les valeurs suivantes :

Dans le fichier de configuration des repositories `environments/group_vars/all/main/repositories.yml`

```
install_mode: container # Default to legacy

container_repository:
  registry_url:
  username:
  password:
```

Actuellement l'antivirus n'est pas supporté par le déploiement en mode conteneur, vous devrez configurer la valeur suivante : `vitam.ingest_external.ignore_antivirus_check: true`.

De plus, le composant library n'est pas supporté en mode container, vous devrez laisser vide le groupe `[library]` de votre inventaire.

4.3.8.2 Procédures à exécuter AVANT la montée de version

4.3.8.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V7.1.

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.8.2.2 Mise à jour des dépôts (YUM/APT)

Prudence : Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable `vitam_repositories` sous `environments/group_vars/all/main/repositories.yml` afin de renseigner les dépôts à la version cible.

Pour le dépôt vitam-external, vous devez renseigner la version adaptée à votre système d'exploitation (par exemple pour la version 7.1.0) :

- CentOS 7 : https://download.programmevitam.fr/vitam_repository/7.1.0/rpm/vitam-external/7/
- AlmaLinux 9 : https://download.programmevitam.fr/vitam_repository/7.1.0/rpm/vitam-external/9/
- Debian 11 : https://download.programmevitam.fr/vitam_repository/7.1.0/deb/vitam-external/11/
- Debian 12 : https://download.programmevitam.fr/vitam_repository/7.1.0/deb/vitam-external/12/

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.8.2.3 Montée de version vers mongo 6.0

Prudence : Attention Cette montée de version doit être effectuée AVANT la montée de version V7.1 de Vitam et après l'arrêt des tâches planifiées et des externals. Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V7.1. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V7.1 **sur tous les sites** :

```
# Mise à jour mongo
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_60.yml --ask-vault-pass
```

4.3.8.2.4 Montée de version vers mongo 7.0

Prudence : Cette montée de version doit être effectuée AVANT la montée de version V7.1 de vitam et après la montée de version de MongoDB 6.0 ci-dessus.

Exécutez le playbook suivant à partir de l'ansiblerie de la V7.1 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_70.yml --ask-vault-pass
```

4.3.8.2.5 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V7.1

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.8.2.6 Suppression des anciens meta-packages de COTS

Exécutez le playbook suivant à partir de l'ansiblerie de la V7.1 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/remove_old_
↪metapackages.yml --ask-vault-pass
```

4.3.8.2.7 Duplication des packages lors de la mise à jour

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V7.1

Prudence : Cette opération doit être effectuée uniquement pour les systèmes d'exploitation à base de RPM (CentOS 7 & AlmaLinux 9) sur un vitam éteint.

Un bug a été introduit dans la construction des packages RPM à partir de la V6RC. Si vous effectuez une montée de version depuis la V6RC (v6.rc.1 ou inférieure), la V6 (v6.2 ou inférieure) ou la V7.0 (v7.0.1 ou inférieure), vous devrez appliquer la commande suivante pour supprimer les précédents packages. Sinon lors de de la mise à jour, le package précédent ne sera pas désinstallé conduisant à la présence de plusieurs versions du même package.

Exemple d'erreur :

```
/var/tmp/rpm-tmp.fGwZMk: ligne 1 : fg: pas de contrôle de tâche
erreur : %preun(vitam-offer-6.2-1.noarch) scriptlet échoué, état de sortie 1
Error in PREUN scriptlet in rpm package vitam-offer
erreur : vitam-offer-6.2-1.noarch: effacer échoué
```

Voici la commande à exécuter pour permettre la désinstallation des packages de la version précédente :

```
ansible vitam --ask-vault-pass -v -a "yum --setopt=tsflags=noscripts remove -y vitam-
↪*.noarch" -i environments/<inventaire>
```

4.3.8.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

4.3.8.3.1 Lancement du master playbook vitam

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-pass
```

4.3.8.3.2 Lancement du master playbook extra

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

4.3.8.4 Procédures à exécuter APRÈS la montée de version

4.3.8.4.1 Réindexation de l'indice logstash-vitam courant

Suite à la mise en place de filebeat (mode par défaut), le mapping associé aux indices a évolué. Afin de prendre en compte le nouveau mapping dans l'indice de la journée courante, vous devrez appliquer le playbook de migration suivant sur **tous les sites**.

Tant que vous n'aurez pas appliqué ce playbook, la centralisation des logs sera défectueuse et entrainera de très nombreux logs d'erreurs au niveau des serveurs logstash.

Si vous n'appliquez pas ce playbook, cette erreur se corrigera automatiquement à la création de l'indice du lendemain. Vous n'aurez juste pas accès à la centralisation des logs pour la journée courante dans les outils tel que Kibana.

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/rename_current_
↪logstash_vitam_index.yml --ask-vault-pass
```

4.3.8.4.2 Migration des mappings elasticsearch pour les métadonnées

Cette migration de données consiste à mettre à jour le modèle d'indexation des métadonnées sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

4.3.9 Notes et procédures spécifiques V8.0

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : V6 -> V7.0 -> V7.1 -> V8.0

4.3.9.1 Adaptation des sources de déploiement ansible

4.3.9.1.1 Configuration des jobs d'audit

Il est maintenant possible de configurer les jobs d'audit d'intégrité et d'existence par tenant, et par conséquent la définition des jobs d'audit a changé.

L'ancienne configuration définissait la fréquence d'exécution et les paramètres des jobs dans le fichier `deployment/environments/group_vars/all/advanced/vitam_vars.yml` :

vitam_timers :

functional_administration : frequency_integrity_audit : « 0 0 0 1 JAN ? 2020 » frequency_existence_audit : « 0 0 0 1 JAN ? 2020 »

scheduler :

job_parameters :

integrity_audit : operations_delay_in_minutes : 1440

existence_audit : operations_delay_in_minutes : 1440

La nouvelle configuration n'utilise plus les valeurs de `vitam_timers.functional_administration.frequency_integrity_audit` et `vitam_timers.functional_administration.frequency_existence_audit`, et prend la forme suivante :

scheduler :

job_parameters :

integrity_audit :

- key : SYSTEM selected_tenants : [1] operations_delay_in_minutes : 1440 frequency : « 0 0 2 ? * * * » # Every day at 2am
- key : TENANT2 selected_tenants : [2] operations_delay_in_minutes : 1440 frequency : « 0 0 4 ? * * * » # Every day at 4am

existence_audit :

- key : SYSTEM selected_tenants : [1,2] operations_delay_in_minutes : 1440 frequency : « 0 0 6 ? * * * » # Every day at 6am

Pour `integrity_audit` et `existence_audit`, il conviendra de définir une entrée par tenant ou groupe de tenant devant s'exécuter à la même fréquence. Les différents champs sont décrits ci-dessous :

Le paramètre `key` doit être unique et sera ajouté au nom du job. Il doit donc être de forme alphanumérique, sans espaces.

Le paramètre `selected_tenants` liste les identifiants de tenants auquel la fréquence s'applique. S'il est vide ou non renseigné, la fréquence s'appliquera à tous les tenants.

Le paramètre `operations_delay_in_minutes` correspond au paramètre de la version précédente.

Le paramètre `frequency` correspond aux fréquences précédemment définies dans les paramètres `vitam_timers.functional_administration.frequency_integrity_audit` et `vitam_timers.functional_administration.frequency_existence_audit`.

4.3.9.1.2 Déploiement des exporters pour VitamUI

Si vous utilisez VitamUI et que vous souhaitez déployer les exporters permettant de mettre en place la remontée de métriques dans Prometheus, vous devrez ajouter les groupes suivants au fichier d'inventaire (cf. inventaire d'exemple : `environments/hosts.example`) :

[hosts :children] hosts_vitamui

4.3.9.1.2.1 # ZONE VITAMUI

[hosts_vitamui] # optional : To deploy exporters on VitamUI

[hosts_vitamui :children] hosts_vitamui_mongod

[hosts_vitamui_mongod] # optional : To deploy mongodb-exporter on VitamUI

Pour permettre de déployer l'exporter mongodb, vous devrez ajouter au fichier `environments/group_vars/all/main/vault-vitam.yml` les paramètres associés à la BDD mongo-vitamui :

mongodb :

mongo-vitamui : mongod_port : 27017 admin :

user : vitamdb-admin password : admin1234

Avant de pouvoir exécuter le playbook de configuration des exporters pour VitamUI, il faudra au préalable récupérer les host_vars à l'aide du playbook `ansible-vitam-exploitation/generate_hostvars_for_2_network_interfaces.yml` ou `ansible-vitam-exploitation/generate_hostvars_for_1_network_interface.yml`.

Ensuite, ces exporters seront déployés à l'aide du playbook : `ansible-vitam-extra/vitamui_prometheus.yml`.

4.3.9.2 Procédures à exécuter AVANT la montée de version

4.3.9.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V8.0.

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.9.2.2 Mise à jour des dépôts (YUM/APT)

Prudence : Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable `vitam_repositories` sous `environments/group_vars/all/main/repositories.yml` afin de renseigner les dépôts à la version cible.

Pour le dépôt vitam-external, vous devez renseigner la version adaptée à votre système d'exploitation (par exemple pour la version 8.0.0) :

- AlmaLinux 9 : https://download.programmevitam.fr/vitam_repository/8.0.0/rpm/vitam-external/9/
- Debian 12 : https://download.programmevitam.fr/vitam_repository/8.0.0/deb/vitam-external/12/

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.9.2.3 Mise à jour de MongoDB 7.0.28

Prudence : Attention Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V8.0. Cette opération est à effectuer si vous venez des versions de Vitam suivante : V7.1.4-. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V8.0 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_70.yml --ask-vault-pass
```

4.3.9.2.4 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V8.0

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.9.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

4.3.9.3.1 Lancement du master playbook vitam

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-pass
```

4.3.9.3.2 Lancement du master playbook extra

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

4.3.9.4 Procédures à exécuter APRÈS la montée de version

4.3.9.4.1 Migration des mappings elasticsearch pour les métadonnées

Cette migration de données consiste à mettre à jour le modèle d'indexation des métadonnées sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

4.3.9.4.2 Migration des profiles

Prudence : Cette migration est à effectuer APRÈS la montée de version vers Vitam V8.0 et uniquement sur **site primaire**.

Ce playbook de migration a pour but de mettre à jour les notices en y intégrant la notion de **SedaVersion**.

Le script effectue les actions suivantes : - Définit par défaut la version **SEDA 2.3** pour les AUP (*Archive Unit Profile*). - Si un fichier est présent pour les AP (*Archive Profile*), il tente de déterminer la version à partir de celui-ci. Dans le cas contraire, la version par défaut sera appliquée.

Pour lancer ce playbook, utilisez la commande suivante :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migrate_
↪profiles_v7.1_to_v8.0.yml --ask-vault-pass
```

4.3.9.4.3 Migration unités archivistiques avec demande de transfert non complétée

Cette migration permet de corriger des erreurs de sécurisation ou d'audit sur des unités archivistiques pour lesquelles une demande de transfert a été initiée, mais dont le transfert effectif n'a pas été finalisée.

Prudence : Cette procédure doit être exécutée uniquement en cas de migration majeure vers Vitam V8.0+ depuis : - Une version 7.1.1 ou inférieure - Une version 7.0.1 ou inférieure - Une version 6.3-1 ou inférieure

Prudence : Cette migration est à effectuer **APRES** l'installation et uniquement sur le **site primaire**.

Prudence : Dans le cas où un tenant comporte un très grand nombre d'unités archivistiques dont la demande de transfert a été initiée pour un tenant donné (plus de 100 000 unités archivistiques), la migration ne pourra alors s'opérer. Le support Vitam devra alors être contacté.

Pour lancer la migration, le playbook suivant doit être exécuté :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪metadata_transfer_request_traceability.yml --ask-vault-pass
```

4.3.10 Notes et procédures spécifiques V8.1

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : V7.1 -> V8.0 -> V8.1

4.3.10.1 Adaptation des sources de déploiement ansible

N/A

4.3.10.2 Procédures à exécuter AVANT la montée de version

4.3.10.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V8.1.

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.10.2.2 Mise à jour des dépôts (YUM/APT)

Prudence : Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable `vitam_repositories` sous `environments/group_vars/all/main/repositories.yml` afin de renseigner les dépôts à la version cible.

Pour le dépôt vitam-external, vous devez renseigner la version adaptée à votre système d'exploitation (par exemple pour la version 8.1.0) :

- AlmaLinux 9 : https://download.programmevitam.fr/vitam_repository/8.1.0/rpm/vitam-external/9/
- Debian 12 : https://download.programmevitam.fr/vitam_repository/8.1.0/deb/vitam-external/12/

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.10.2.3 Montée de version vers mongo 8.0

Prudence : Attention Cette montée de version doit être effectuée AVANT la montée de version V8.1 de vitam. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V8.1 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_80.yml --ask-vault-pass
```

4.3.10.2.4 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V8.1

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.10.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

4.3.10.3.1 Lancement du master playbook vitam

Prudence : La variable `prevent_scheduling_start` empêche le redémarrage automatique des *Jobs Vitam* en fin de playbook. Ceux-ci doivent rester à l'arrêt tant que les procédures de post-migration et les contrôles ne sont pas terminés ; ils seront redémarrés explicitement à la dernière étape de cette procédure.

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-
↪pass -e prevent_scheduling_start=true
```

4.3.10.3.2 Lancement du master playbook extra

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

4.3.10.4 Procédures à exécuter APRÈS la montée de version

4.3.10.4.1 Redémarrage des *Jobs Vitam*

Une fois l'ensemble des procédures de post-migration exécutées et les contrôles effectués, redémarrer les *Jobs Vitam* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.11 Notes et procédures spécifiques V9.0

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : V7.1 -> V8.0 -> V8.1 -> V9.0

4.3.11.1 Adaptation des sources de déploiement ansible

4.3.11.1.1 Migration de la configuration d'offres S3

Dans le cas de l'utilisation d'une offre de stockage S3, le provider `amazon-s3-v1` n'est plus disponible. Il est remplacé par le provider `amazon-s3-v2`.

La clé de configuration `vitam_offers.<offre>.s3SignerType` n'est plus paramétrable à présent et doit être supprimée. Seul le mode de signature `AWSS3V4SignerType` (par défaut dans les versions précédentes de Vitam) est supportée à présent. Le mode de signature `S3SignerType` (dangereux / déprécié par Amazon AWS) n'est plus supporté.

De même, la validation des hostnames des certificats HTTPS est dorénavant activée par défault dans le nouveau provider `amazon-s3-v2`. La validation peut être désactivée via la clé `vitam_offers.<offre>.s3IgnoreCertificateHostnameValidation` (uniquement pour environnements de test).

Il convient de mettre à jour la configuration des offres de stockage en conséquent (`vitam_offers.<offre>.provider`) dans les sources de déploiement.

4.3.11.1.2 Déploiement du nouveau composant vitam-antivirus

L'appel a l'antivirus été externalisé dans un nouveau composant permettant son déploiement autonome. Actuellement, ce nouveau composant est colocalisé avec les groupes `hosts_ingest_external` & `hosts_worker`. Cependant, il n'est utile sur les workers que lors de l'utilisation du module de collecte.

Ainsi, vous pouvez désactiver son déploiement sur les workers si vous n'utilisez pas le module de collecte en appliquant la procédure suivante.

Dans le playbook `ansible-vitam/services/vitam/antivirus.yml`, supprimez la chaine de caractère `:hosts_worker`.

Résultat attendu après modification :

```
- hosts: hosts_ingest_external
  any_errors_fatal: true
  roles:
    - antivirus
    - vitam
  vars:
    vitam_struct: "{{ vitam.antivirus }}"
  tags: antivirus
```

4.3.11.2 Procédures à exécuter AVANT la montée de version

4.3.11.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V9.0

Prudence : Cette opération doit être effectuée avec les sources de déploiement de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduler.yml --ask-vault-pass
```

4.3.11.2.2 Mise à jour des dépôts (YUM/APT)

Prudence : Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable `vitam_repositories` sous `environments/group_vars/all/main/repositories.yml` afin de renseigner les dépôts à la version cible.

Pour le dépôt vitam-external, vous devez renseigner la version adaptée à votre système d'exploitation (par exemple pour la version 9.0.0) :

- AlmaLinux 9 : https://download.programmevitam.fr/vitam_repository/9.0.0/rpm/vitam-external/9/
- Debian 12 : https://download.programmevitam.fr/vitam_repository/9.0.0/deb/vitam-external/12/

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.11.2.3 Mise à jour de MongoDB 8.0.23

Prudence : Attention Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V9.1. Cette opération est à effectuer si vous venez des versions de Vitam suivante : V8.1.2-. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V9.0 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_80.yml --ask-vault-pass
```

4.3.11.2.4 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V9.0

Prudence : Cette opération doit être effectuée avec les sources de déploiements de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.11.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

4.3.11.3.1 Lancement du master playbook vitam

Prudence : La variable `prevent_scheduling_start` empêche le redémarrage automatique des *Jobs Vitam* en fin de playbook. Ceux-ci doivent rester à l'arrêt tant que les procédures de post-migration et les contrôles ne sont pas terminés ; ils seront redémarrés explicitement à la dernière étape de cette procédure.

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-
↪pass -e prevent_scheduling_start=true
```

4.3.11.3.2 Lancement du master playbook extra

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

4.3.11.4 Procédures à exécuter APRÈS la montée de version

4.3.11.4.1 Redémarrage des *Jobs Vitam*

Une fois l'ensemble des procédures de post-migration exécutées et les contrôles effectués, redémarrer les *Jobs Vitam* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.12 Notes et procédures spécifiques V9.1

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : V7.1 -> V8.0 -> V8.1 -> V9.0 -> V9.1

4.3.12.1 Adaptation des sources de déploiement ansible

4.3.12.1.1 Renommage de variables pour mise en cohérence des noms et groupes

Afin de faciliter la maintenance et les évolutions autour de l'ansiblerie, certaines variables et groupes d'hôtes ont étés renommés pour correspondre à la convention de nommage établie.

Les variables de configuration suivantes ont étés renommées ainsi :

```
vitam:
  accessexternal: -> access_external
  accessinternal: -> access_internal
  batchreport: -> batch_report
  ingestexternal: -> ingest_external
  ingestinternal: -> ingest_internal
  elastickibanainterceptor: -> elastic_kibana_interceptor
  storageengine: -> storage
  storageofferdefault: -> offer
```

Les groupes d'hôtes suivants du fichier d'inventaire ont étés renommés ainsi :

```
[hosts_storage_engine] -> [hosts_storage]
[hosts_storage_offer_default] -> [hosts_offer]
```

Ces modifications sont à reporter dans vos configurations, vous pouvez effectuer cette opération en exécutant les commandes sed suivantes :

```
sed -i 's/accessexternal/access_external/g' environments/group_vars/all/*/*.yml
sed -i 's/accessinternal/access_internal/g' environments/group_vars/all/*/*.yml
sed -i 's/batchreport/batch_report/g' environments/group_vars/all/*/*.yml
sed -i 's/ingestexternal/ingest_external/g' environments/group_vars/all/*/*.yml
sed -i 's/ingestinternal/ingest_internal/g' environments/group_vars/all/*/*.yml
sed -i 's/elastickibanainterceptor/elastic_kibana_interceptor/g' environments/group_
↪vars/all/*/*.yml
sed -i 's/storageengine/storage/g' environments/group_vars/all/*/*.yml
sed -i 's/storageofferdefault/offer/g' environments/group_vars/all/*/*.yml

sed -i 's/hosts_storage_engine/hosts_storage/g' environments/<inventaire>
sed -i 's/hosts_storage_offer_default/hosts_offer/g' environments/<inventaire>
```

Pour information, le playbook `ansible-vitam/services/vitam/storage-engine.yml` a été renommé en `ansible-vitam/services/vitam/storage.yml`.

4.3.12.1.2 Configuration du nombre de shards/replicas pour chaque indices elasticsearch-log

Ces modifications impliquent des changements de la configuration Ansible associée aux indices pour plus de cohérence avec les autres variables de configuration.

Ainsi, les clés de configurations suivantes ont étés retirées :

- kibana.log.metrics.replica
- kibana.log.metrics.shards

Elles ont étés remplacées par ces variables spécifiques pour chacun de vos indices :

```
elasticsearch:
  log:
    indices:
      default_config: # Default configuration if others indices are undefined
        number_of_shards: 1
        number_of_replicas: 1
      vitam: # Configuration for indexes logstash-vitam-*
        number_of_shards: 1
        number_of_replicas: 1
      access: # Configuration for indexes logstash-access-*
        number_of_shards: 1
        number_of_replicas: 1
```

- Si aucune valeur pour les indices vitam ou access n'est spécifiée, c'est la configuration default_config qui sera appliquée.
- Si la configuration default_config n'est pas spécifiée, les valeurs par défaut sont shards: 1 et replicas: 1.

4.3.12.2 Procédures à exécuter AVANT la montée de version

4.3.12.2.1 Arrêt des timers et des accès externes à Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V9.1

Prudence : Cette opération doit être effectuée avec les sources de déploiement de l'ancienne version.

Les timers et les externals de Vitam doivent être arrêtés sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_
↪external.yml --ask-vault-pass
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.12.2.2 Mise à jour des dépôts (YUM/APT)

Prudence : Cette opération doit être effectuée AVANT la montée de version

Afin de pouvoir déployer la nouvelle version, vous devez mettre à jour la variable vitam_repositories sous environments/group_vars/all/main/repositories.yml afin de renseigner les dépôts à la version cible.

Pour le dépôt vitam-external, vous devez renseigner la version adaptée à votre système d'exploitation (par exemple pour la version 9.1.0) :

- AlmaLinux 9 : https://download.programmevitam.fr/vitam_repository/9.1.0/rpm/vitam-external/9/
- Debian 12 : https://download.programmevitam.fr/vitam_repository/9.1.0/deb/vitam-external/12/

Puis exécutez le playbook suivant **sur tous les sites** :

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

4.3.12.2.3 Mise à jour intermédiaire des clusters ElasticSearch

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V9.1

Cette procédure permet de faire le saut intermédiaire entre les versions 7 et 8 des clusters ElasticSearch (data & log).

Le saut final vers la version 9 d'ElasticSearch sera effectué lors de la mise à jour de Vitam via l'exécution du master playbook `ansible-vitam/vitam.yml`.

Exécutez les playbooks suivants sur **tous les sites** pour mettre à jour vos clusters ElasticSearch vers la version intermédiaire 8.19.8 :

```
ansible-playbook -i environments/<inventaire> ansible-vitam/services/cots/
↪elasticsearch_log.yml --ask-vault-pass -e "elasticsearch_version=8.19.8"
ansible-playbook -i environments/<inventaire> ansible-vitam/services/cots/
↪elasticsearch_data.yml --ask-vault-pass -e "elasticsearch_version=8.19.8"
```

Attention, à partir de cette étape, il va falloir s'assurer que l'ensemble des indexes présents sur les clusters ElasticSearch (data & log) sont compatibles avec la version 9 d'ElasticSearch avant de poursuivre la procédure de montée de version. Les indexes incompatibles sont ceux créés avec une version antérieure à la version 8.x d'ElasticSearch.

4.3.12.2.3.1 Pour ElasticSearch-Log

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/check_
↪incompatible_elasticsearch_v9_indices.yml --ask-vault-pass -t elasticsearch_log
```

Vous pouvez aussi vérifier la compatibilité via Kibana-Log :

- Accédez à la section `Management > Stack Management > Stack > Upgrade Assistant`
- Vérifiez que les indexes ne sont pas marqués comme `Incompatible with the current version of Elasticsearch`

Si des indexes incompatibles sont détectés, soit vous procédez à la migration des données à l'aide de la procédure dédiée, sinon, si les données ne sont pas critiques, vous pouvez les supprimer en ajoutant les paramètres `-e delete_incompatible_indices=true --tags elasticsearch_log` au playbook de détection.

Prudence : Cette opération supprimera définitivement les indexes concernés. Assurez-vous d'avoir sauvegardé ou réindexé l'ensemble des données nécessaires avant de poursuivre.

4.3.12.2.3.2 Pour ElasticSearch-Data

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/check_
↪incompatible_elasticsearch_v9_indices.yml --ask-vault-pass -t elasticsearch_data
```

Si des indexes incompatibles sont détectés, vous devez procéder à la réindexation de ces indexes. Pour cela, exécutez le playbook de réindexation :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/reindex_es_
↪data.yml --ask-vault-pass
```

Note : Si seulement une sous-partie des indexes est incompatible avec la nouvelle version d'ElasticSearch, il ne sera pas nécessaire de réindexer l'ensemble des données. Dans ce cas, référez-vous à la documentation d'exploitation sur la réindexation reindexation_es pour cibler uniquement les indexes concernés.

Prudence : Durant la réindexation, les tâches planifiées et les accès externes à Vitam doivent être arrêtés.

Cette opération peut prendre plusieurs heures en fonction de la quantité de données à réindexer. Si la durée de réindexation est trop longue pour votre production, veuillez vous référer à la procédure de migration de données Vitam vers un nouveau cluster ElasticSearch tampon.

À l'issue de la réindexation, vous pouvez réexécuter le playbook de détection avec les paramètres `-e delete_incompatible_indices=true --tags elasticsearch_data` afin de supprimer les indexes incompatibles restants.

Prudence : Cette opération supprimera définitivement les indexes concernés. Assurez-vous d'avoir sauvegardé ou réindexé l'ensemble des données nécessaires avant de poursuivre.

4.3.12.2.4 Mise à jour de MongoDB 8.0.23

Prudence : Attention Cette opération doit être effectuée après avoir mis à jour les dépôts Vitam en V9.1. Cette opération est à effectuer si vous venez des versions de Vitam suivante : V8.1.2- ou V9.0.0. Il est recommandé d'effectuer un backup des bases de données à l'aide de mongodump avant de poursuivre.

Exécutez le playbook suivant à partir de l'ansiblerie de la V9.1 **sur tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪mongodb_80.yml --ask-vault-pass
```

4.3.12.2.5 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la montée de version vers la V9.1

Prudence : Cette opération doit être effectuée avec les sources de déploiement de l'ancienne version.

Vitam doit être arrêté sur **tous les sites** :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.12.3 Application de la montée de version

Prudence : L'application de la montée de version s'effectue d'abord sur les sites secondaires puis sur le site primaire.

4.3.12.3.1 Lancement du master playbook vitam

Prudence : La variable `prevent_scheduling_start` empêche le redémarrage automatique des *Jobs Vitam* en fin de playbook. Ceux-ci doivent rester à l'arrêt tant que les procédures de post-migration et les contrôles ne sont pas terminés ; ils seront redémarrés explicitement à la dernière étape de cette procédure.

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-
↪pass -e prevent_scheduling_start=true
```

4.3.12.3.2 Lancement du master playbook extra

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

4.3.12.4 Procédures à exécuter APRÈS la montée de version

4.3.12.4.1 Migration des mappings elasticsearch pour les métadonnées

Cette migration de données consiste à mettre à jour le modèle d'indexation des métadonnées sur elasticsearch-data.

Elle est réalisée en exécutant la procédure suivante sur **tous les sites** (primaire et secondaire(s)) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/migration_
↪elasticsearch_mapping.yml --ask-vault-pass
```

4.3.12.4.2 Redémarrage des *Jobs Vitam*

Une fois l'ensemble des procédures de post-migration exécutées et les contrôles effectués, redémarrer les *Jobs Vitam* :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.13 Notes et procédures spécifiques V10.0

Prudence : Veuillez appliquer les procédures spécifiques à chacune des versions précédentes en fonction de la version de départ selon la suite suivante : V8.1 -> V9.0 -> V9.1 -> V10.0

4.3.13.1 Adaptation des sources de déploiement ansible

4.3.13.1.1 Mise à jour de la pile Elastic vers 9.4.9 (US #17109)

Kibana passe de 9.3.2 à 9.4.9 pour corriger la vulnérabilité Brotli du runtime Node.js embarqué (CVE-2026-111497, ESA-2026-232). Cette distribution embarque Node.js 24.22.1 ; ne pas remplacer ce runtime séparément. Elasticsearch, son plugin `analysis-icu`, Logstash et Filebeat sont alignés sur 9.4.9. Les paquets DEB/RPM, les images conteneurs, les valeurs Ansible par défaut et les serveurs Elasticsearch de développement et de CI sont concernés. Les artefacts Elasticsearch/ICU 8.19.8 nécessaires aux migrations depuis les anciennes versions sont conservés.

Les clients Java Elasticsearch restent en 9.3.2 : ils sont compatibles avec un serveur de version mineure supérieure dans la même version majeure, conformément à la politique de compatibilité du client Java [12]. Cela ne dispense pas de qualifier les recherches et l'indexation de Vitam.

4.3.13.1.1.1 Préparation

Avant la montée de version, pour chaque environnement :

- Recenser les versions réellement exécutées, les références et empreintes des artefacts, les plugins et les éventuelles surcharges `elasticsearch.version`, `kibana.version`, `logstash.version` et `filebeat.version`. Adapter ces surcharges à la cible ; elles prennent le pas sur les valeurs par défaut.
- Distinguer les clusters logs et données et leurs instances Kibana. Kibana-data passe par `elastic-kibana-interceptor` ; ce chemin doit être testé séparément.
- Lire les changements incompatibles et les problèmes connus de chaque composant entre la version de départ et la cible. Vérifier notamment les usages ILM, TSDB, les modèles d'index et les plugins personnalisés. Prévoir au moins 2 Go de mémoire par instance Kibana, selon la recommandation Elastic depuis 9.4.
- Publier les paquets, le plugin ICU et les images 9.4.9 dans les dépôts utilisés par l'environnement avant de lancer le déploiement. Les images Elasticsearch, Kibana et Logstash sont récupérées depuis `docker.elastic.co` par la CI, puis publiées sous les noms habituels `vitam-external/<composant>:9.4.9`. Autoriser l'accès au registre Elastic depuis les machines de build et de développement ; les images de déploiement restent servies par le registre interne.
- Suspendre les écritures, les traitements et les accès selon la procédure de migration applicable. Réaliser et vérifier des snapshots cohérents des deux clusters Elasticsearch, y compris l'état fonctionnel Kibana pour ses objets sauvegardés. Conserver également les exports des objets Kibana, les configurations, les clés de chiffrement et les artefacts de la version de départ. Un export des tableaux de bord seul ne constitue pas une sauvegarde complète de Kibana.

4.3.13.1.1.2 Déploiement et qualification

Mettre à jour Elasticsearch avant Kibana : Kibana 9.4 ne doit pas démarrer sur Elasticsearch 9.3. Le master playbook `ansible-vitam/vitam.yml` respecte cet ordre pour les usages logs et données. En déploiement sélectif, terminer et contrôler la mise à jour du cluster concerné et du plugin ICU avant celle de Kibana, puis terminer l'alignement de Logstash et Filebeat. Maintenir les Jobs Vitam arrêtés jusqu'à la fin de la post-migration, comme indiqué ci-dessous.

12. https://github.com/elastic/elasticsearch-java#compatibility

En qualification, exécuter une installation neuve et une montée de version sur un environnement représentatif, pour les modes de déploiement utilisés :

- Contrôler la version de chaque nœud Elasticsearch, la santé des deux clusters et la version du plugin ICU. Vérifier l'indexation, les recherches et les analyseurs utilisés par Vitam.
- Contrôler la version de Kibana et du Node.js fourni par le paquet ou l'image réellement installé. Vérifier `/status` et l'absence d'erreurs de migration des objets sauvegardés pour Kibana-log et Kibana-data.
- Vérifier les imports des tableaux de bord, leurs objets liés et l'index par défaut. Tester la consultation des logs, les recherches sur les données via l'intercepteur, ainsi que les parcours Vitam/Vitam UI et connecteurs utilisés.
- Vérifier que Filebeat et Logstash acheminent de nouveaux événements jusqu'à Elasticsearch et que ceux-ci sont consultables dans Kibana.
- Répéter une restauration de la sauvegarde de départ en qualification. Joindre au ticket les versions observées, MR, résultats CI, preuves de ces contrôles et liste des environnements restant à traiter avant clôture.

4.3.13.1.1.3 Restauration

En cas d'échec, arrêter les composants mis à jour et les écritures. Reconstituer un environnement de la version de départ avec ses configurations, plugins et clés, puis restaurer les snapshots pris avant la migration et vérifier la cohérence avec les autres données Vitam avant de rouvrir les accès.

Prudence : Une rétrogradation des binaires ne restaure pas les index ni les objets sauvegardés migrés. Ne pas ouvrir les répertoires de données mis à jour avec une ancienne version d'Elasticsearch. Les écritures postérieures au snapshot ne sont pas récupérées par cette restauration ; prévoir leur reprise avant de rétablir le service.

Références : avis de sécurité Elastic ESA-2026-232 [13], compatibilité et ordre d'installation [14], notes Kibana [15] et notes Elasticsearch [16].

4.3.13.1.2 Réorganisation du stockage workspace par tenant et seuils de saturation

Le composant `workspace-collect` persiste les objets binaires sous un répertoire propre à chaque tenant. Cette organisation permet de renforcer des quotas de stockage et de refuser les versements du module de collecte lorsque le stockage approche de la saturation. Le composant `workspace` conserve son organisation historique.

Prudence : Avant toute migration depuis une version de Vitam V9.1- (V9.1 ou inférieure) vers une version V10.0+ (V10.0 ou supérieure), assurez-vous préalablement que le module de collecte ne contienne aucune transaction active. Seules les transactions versées avec succès (`ACK_OK`), versées avec avertissement(s) (`ACK_WARNING`) ou abandonnées (`ABORTED`), et dont la purge des données a été réalisée, peuvent être migrées sans risque. Les transactions ouvertes en édition (`OPEN`), en cours de validation (`READY`), validées (`VALIDATED`), en cours d'envoi (`SENDING`), envoyées et en cours de traitement par le SAE (`SENT`), rejetées par le SAE (`ACK_KO`) ou en erreur technique (`KO`) peuvent contenir des binaires et doivent être finalisées (versées et acquittées par Vitam, ou abandonnées) avant la migration, sans quoi ces transactions seront corrompues et ne pourront plus être ni reprises ni purgées.

Il convient de :

- finaliser toutes les transactions sur l'ensemble des tenants (verser et attendre l'acquittement en succès ou avec avertissement(s), ou abandonner) ;

13. https://discuss.elastic.co/t/kibana-8-19-24-9-4-9-9-5-6-security-update-esa-2026-232/391369
14. https://www.elastic.co/docs/deploy-manage/deploy/self-managed/install-kibana
15. https://www.elastic.co/docs/release-notes/kibana
16. https://www.elastic.co/docs/release-notes/elasticsearch

- attendre la purge des binaires (délai pouvant atteindre 2 heures) ;
- vérifier sur le serveur `workspace-collect` que le répertoire de stockage `/vitam/data/workspace-collect/` ne contient plus aucun binaire.

La configuration des seuils pour le module de collecte et les précautions d'usage des quotas de versement sont décrites dans la documentation d'exploitation, au chapitre workspace_storage_thresholds.

4.3.13.1.3 Validation de la chaîne de confiance des certificats TSA

Le composant worker peut désormais valider la chaîne de confiance des certificats TSA utilisés lors de la vérification des preuves d'horodatage. Cette validation est **désactivée par défaut** ; aucune action n'est requise lors de la montée de version si vous ne souhaitez pas l'activer.

Pour activer cette validation, positionner la variable suivante (par exemple dans le bloc `worker` du fichier `environments/group_vars/all/advanced/vitam_vars.yml`) :

```
vitam:
  worker:
    timestamp_truststore_enabled: true
```

Avant le déploiement, déposez ensuite les certificats TSA de confiance dans le répertoire `environments/certs/timestamping/trusted`, sous forme de fichiers portant l'extension `.crt` (encodés indifféremment en DER ou en PEM), puis exécutez :

```
./generate_stores.sh
```

Prudence : Le répertoire `environments/certs/timestamping/trusted` doit contenir l'ensemble des certificats TSA utilisés au fil du temps : les certificats actuellement en service, mais également les anciens certificats si des renouvellements ont eu lieu. Les preuves d'horodatage déjà produites ont été signées avec les anciens certificats ; leur vérification échouerait si ceux-ci étaient absents du truststore.

Cette commande génère le fichier `environments/keystores/timestamping/truststore_timestamping.p12` en utilisant le mot de passe `truststores.timestamping` défini dans `environments/group_vars/all/main/vault-keystores.yml`.

Lorsque la validation est activée, le déploiement du composant worker échoue en l'absence de ce truststore. Après le déploiement, une preuve d'horodatage dont le certificat TSA ne peut pas être rattaché à l'un des certificats de confiance est rejetée lors de sa vérification.

4.3.13.1.4 Gestion du démarrage des Jobs Vitam lors d'une montée de version

Les *Jobs Vitam* sont démarrés automatiquement à la fin du master playbook `ansible-vitam/vitam.yml`. Ce démarrage est désormais conditionné par l'extra var `prevent_scheduling_start`, positionnée à `false` par défaut.

Lors d'une montée de version, il **faut impérativement** passer `prevent_scheduling_start=true` au lancement du master playbook, afin que les *Jobs Vitam* ne se relancent pas alors que la post-migration n'est pas terminée :

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-
↪pass -e prevent_scheduling_start=true
```

Une fois les procédures de post-migration finalisées, les *Jobs Vitam* devront être démarrés explicitement via le playbook d'exploitation :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/start_vitam_
↪scheduling.yml --ask-vault-pass
```

4.3.14 Migration d'un vitam legacy vers un vitam conteneurisé

4.3.14.1 Adaptation des sources de déploiement ansible

Prudence : Les composants `antivirus`, `worker` (en cas d'utilisation des griffons) et `library` ne sont actuellement pas compatible avec le mode de déploiement conteneurisé. La variable `legacy_components_list` permet de déployer n'importe quel composant en mode legacy.

Il faut éditer le contenu du fichier `environments/group_vars/all/main/repositories.yml`. Pour cela il faut rajouter les paramètres présentés dans l'exemple :

```
install_mode: container

legacy_components_list: [ "antivirus", "worker", "library" ]

container_repository:
  registry_url: https://docker.programmevitam.fr/
  username: ''
  password: ''

vitam_container_version: <vitam_version>
```

Avertissement : Dans le cas d'utilisation d'une registry interne il vous faudra effectuer une synchronisation à partir de la registry docker du programme Vitam : https://docker.programmevitam.fr

4.3.14.2 Procédures à exécuter AVANT la migration

4.3.14.2.1 Arrêt complet de Vitam

Prudence : Cette opération doit être effectuée AVANT la migration vers le mode conteneurisé.

Vitam doit être arrêté sur **tous les sites** (site primaire en premier) :

```
ansible-playbook -i environments/<inventaire> ansible-vitam-exploitation/stop_vitam.
↪yml --ask-vault-pass
```

4.3.14.3 Application de la migration

Prudence : L'application de la migration s'effectue d'abord sur les sites secondaires puis sur le site primaire.

Prudence : Il faut s'assurer que la variable `install_mode: container` est bien configurée.

```
ansible-playbook -i environments/<inventaire> ansible-vitam-migration/remove_legacy_
↪packages.yml --ask-vault-pass
```

4.3.14.3.1 Lancement du master playbook vitam

```
ansible-playbook -i environments/<inventaire> ansible-vitam/vitam.yml --ask-vault-pass
```

4.3.14.3.2 Lancement du master playbook extra

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

4.3.14.4 Procédures à exécuter APRÈS la migration

N/A

Table des figures

Liste des tableaux

1 Documents de référence VITAM 2
1 Tableau récapitulatif des versions de la solution logicielle VITAM 6

Index

A

API, 3
AU, 3

B

BDD, 3
BDO, 3

C

CA, 3
CAS, 3
CCFN, 3
CN, 3
COTS, 3
CRL, 3
CRUD, 3

D

DAT, 3
DC, 3
DEX, 3
DIN, 3
DIP, 3
DMV, 3
DNS, 3
DNSSEC, 3
DSL, 3
DUA, 3

E

EAD, 3
EBIOS, 3
ELK, 3

F

FIP, 3

G

GOT, 3

I

IHM, 3
IP, 3
IsaDG, 3

J

JRE, 3
JVM, 4

L

LAN, 4
LFC, 4
LTS, 4

M

M2M, 4
MitM, 4
MoReq, 4

N

NoSQL, 4
NTP, 4

O

OAIS, 4
OOM, 4
OS, 4
OWASP, 4

P

PCA, 4
PDMA, 4
PKI, 4
PRA, 4

R

REST, 4
RGAA, 4
RGI, 4

RPM, 4

S

SAE, 4
SEDA, 4
SGBD, 5
SGBDR, 5
SIA, 5
SIEM, 5
SIP, 5
SSH, 5
Swift, 5

T

TLS, 5
TNA, 5
TNR, 5
TSA, 5
TTL, 5

U

UDP, 5
UID, 5

V

VITAM, 5
VM, 5

W

WAF, 5
WAN, 5