|  |  |
| --- | --- |
|  | Direction des Systèmes d’information  Domaine Ingénierie |
|  |
| 95 rue de MAUBEUGE  75010 PARIS |
|  |
|  |
|  | |
| DOSSIER D’ARCHITECTURE Ansible Tower | |
| barre_jaune | |

Dossier d’architecture Ansible Tower

* date : 30/09/2021
* version du document : 0.9
* auteurs : Erwin Poul & philippe nakache

Historique des versions

| Version | Date | Description | Paragraphes concernés |
| --- | --- | --- | --- |
| 0.1 | 06/05/2021 | Version initiale | Tous |
| 0.2 | 17/07/2021 | Version avancée | Tous |
| 0.3 | 31/08/2021 | Ajout sur la supervision, sur la localisation réseau et le PCI | 2.3, 5.1 et 5.5 |
| 0.4 | 06/09/2021 | Suppression de l’externalisation des sauvegardes logicielles sur NAS (Commvault permet de restaurer des fichiers <1Go sans agent) | 2.3.4, 5.3 |
| 0.5 | 16/09/2021 | Ajout norme de nommage des objets Tower. Correction sur le nombre de clés SSH pour l’utilisateur tower | 3.3, 5.2 |
| 0.6 | 21/09/2021 | Ajout des comptes de services | 5.2 |
| 0.7 | 23/09/2021 | Mise à jour des matrices de flux | 2.2.4, 2.3.4 |
| 0.8 | 26/09/2021 | Changement de la présentation de Ansible Tower | 2.1 |
| 0.9 | 30/09/2021 | Suite retours équipe sécurité | 2.2.4, 2.3.4, 5.4, 5.5 |

Documents de Référence

| Titre | Référence |
| --- | --- |
| Ansible Tower Quick Install Guide v3.8.3 | https://docs.ansible.com/ansible-tower/3.8.3/html/quickinstall/prepare.html |
| Top Support Policies for Red Hat Ansible Automation | https://access.redhat.com/ansible-top-support-policies |
| Red Hat Ansible Tower Life Cycle | https://access.redhat.com/support/policy/updates/ansible-tower |
| A collection of roles to manage Ansible Tower | https://github.com/redhat-cop/tower\_configuration/tree/devel/roles |
| Red Hat Ansible Automation Platform Database Scope of Coverage | https://access.redhat.com/articles/4010491 |
| How do I Download and Install Red Hat Ansible Engine? | https://access.redhat.com/articles/3174981 |
| Tower API Reference Guide | https://docs.ansible.com/ansible-tower/latest/html/towerapi/api\_ref.html |
| Tower : Setting up LDAP Authentication | https://docs.ansible.com/ansible-tower/latest/html/administration/ldap\_auth.html |
| Django Auth-LDAP : Working With Groups | <https://django-auth-ldap.readthedocs.io/en/latest/groups.html#types-of-groups> |
| Django Auth-LDAP : AUTH\_LDAP\_USER\_SEARCH | https://django-auth-ldap.readthedocs.io/en/latest/reference.html#auth-ldap-user-search |
| ISP |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table des matières

* [1 Contexte 6](#_Toc83932818)
* [2 Design de la solution 7](#_Toc83932819)

[2.1 Présentation de Ansible Tower 7](#_Toc83932820)

[2.2 Architecture générale 9](#_Toc83932821)

[2.2.1 Ansible Tower en Innovation 11](#_Toc83932822)

[2.2.2 Ansible Tower en Production 11](#_Toc83932823)

[2.2.3 Ansible Tower en Homologation 11](#_Toc83932824)

[2.2.4 Interactions avec Ansible Tower 11](#_Toc83932825)

[2.3 Architecture détaillée 12](#_Toc83932826)

[2.3.1 Version et mise à jour 12](#_Toc83932827)

[2.3.2 Liste des serveurs Tower par environnement 13](#_Toc83932828)

[2.3.3 Localisation réseau 14](#_Toc83932829)

[2.3.4 Matrice de flux 15](#_Toc83932830)

* [3 Installation de la solution 19](#_Toc83932831)

[3.1 Installation de base 19](#_Toc83932832)

[3.2 Installation spécifique GRDF 19](#_Toc83932833)

[3.3 Nommage des objets Tower 20](#_Toc83932834)

* [4 Cas d’utilisation 21](#_Toc83932835)

[4.1 Post-provisioning d’une VM 21](#_Toc83932836)

[4.2 Décommissionnement d’une VM 23](#_Toc83932837)

[4.3 Déploiement de middleware 25](#_Toc83932838)

[4.3.1 Déploiement de middleware lors du Post-provisioning 25](#_Toc83932839)

[4.3.2 Déploiement de middleware sur une VM existante 27](#_Toc83932840)

[4.4 Conformité 28](#_Toc83932841)

[4.5 Livraison applicative 28](#_Toc83932842)

* [5 Maintenance, Sécurité et Exploitation 29](#_Toc83932843)

[5.1 Supervision 29](#_Toc83932844)

[5.2 Accès 29](#_Toc83932845)

[5.3 Sauvegarde et restauration 31](#_Toc83932846)

[5.3.1 Via Hyperviseur 31](#_Toc83932847)

[5.3.2 Logicielle 31](#_Toc83932848)

[5.4 Maintenance et arrêt 31](#_Toc83932849)

[5.5 Reprise sur incident et PCI 32](#_Toc83932850)

[5.5.1 Sinistre local (SL) 32](#_Toc83932851)

[5.5.2 Sinistre majeur (SM) 32](#_Toc83932852)

|  |  |  |
| --- | --- | --- |
|  |  |  |
| GRDF – Société Anonyme au capital de 1 800 745 000 euros – Siège social : 6 rue Condorcet – 75009 Paris – RCS : PARIS 444 786 511 |
|  |
| pieddepage_grdf_cahier.png | | |

# Contexte

Courant 2021, GRDF se dote d’une solution de CMS (Configuration Management System) afin de centraliser les opérations de provisioning, déploiement middleware, gestion des configuration et mise en conformité système. Cette solution couvre l’ensemble du périmètre système : VMs Linux et Windows, virtuels et physiques et pourra même à terme être utilisée pour réaliser les livraisons applicatives.

Ansible Tower est le produit retenu. L’outil d’automatisation Ansible, très utilisé dans de nombreuses entreprises, permet de définir facilement les opérations que l’on souhaite automatiser de manière déclarative. Le langage YAML, utilisé dans les playbooks, offre une prise en main immédiate et garantie une lecture simple, orientée IaC (Infrastructure as Code), pour l’ensemble des développeurs, contributeurs ou responsables IT. Quant à Tower, son interface Web et son REST API associé va permettre à GRDF à la fois :

* D’exécuter à la demande du code Ansible de manière simple selon les permissions accordées à un utilisateur
* De déléguer ces exécutions à un orchestrateur tiers

Actuellement utilisé chez GRDF, l’outil de gestion de configuration Chef sera remplacé par cette solution, lorsque le périmètre de ce dernier sera entièrement repris par Ansible Tower. L’outil de provisioning de GRDF « Refinfra » sera également décommissionné à ce terme.

Cette nouvelle brique technique s’inscrit dans le programme d’Evolution du Datacenter GRDF pour un Avenir Responsable (EDGAR) et pourra être pilotée, entre autres, par la solution de CMP (Cloud Management Platform) également prévue (vRO : VMware vRealize Orchestrator)

Tower a été qualifié pour répondre aux différents cas d’usage listé dans le tableau ci-après.

| Cas d’utilisation identifiés | Description |
| --- | --- |
| Post-provisioning d’une VM | Tower est piloté par vRO (VMware vRealize Orchestrator) pour exécuter les actions de Post-provisioning sur les VMs Linux et Windows.  Le rollback (retour arrière) sur incident est également géré par Tower |
| Décommissionnement d’une VM | Tower est piloté par vRO pour exécuter les actions de décommissionnement des VMs Linux et Windows. |
| Déploiement de middleware | Sont pilotés par vRO dans le cadre du Post-provisioning avec middleware et peuvent être exécutés directement depuis Tower. |
| Conformité | Permet de vérifier que les VMs Linux et Windows sont conformes aux règles d’ingénierie GRDF. Certains écarts de configurations sont rectifiés automatiquement, d’autres génèrent des alertes uniquement |
| Livraison applicative | Tower sera piloté par les outils d’intégration continue pour automatiser le déploiement des applications de bout en bout |

# Design de la solution

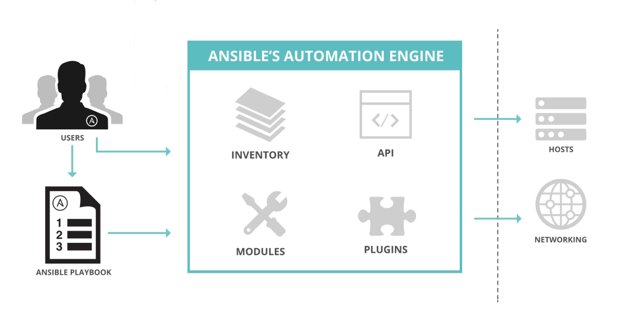
La société Red Hat, éditrice du logiciel Ansible Tower a accompagné l’équipe AES de GRDF dans le cadre du déploiement de la solution.

Le chapitre suivant synthétise les choix retenus lors des ateliers qui se sont déroulés en février 2021 avec les architectes de Red Hat.

## Présentation de Ansible Tower

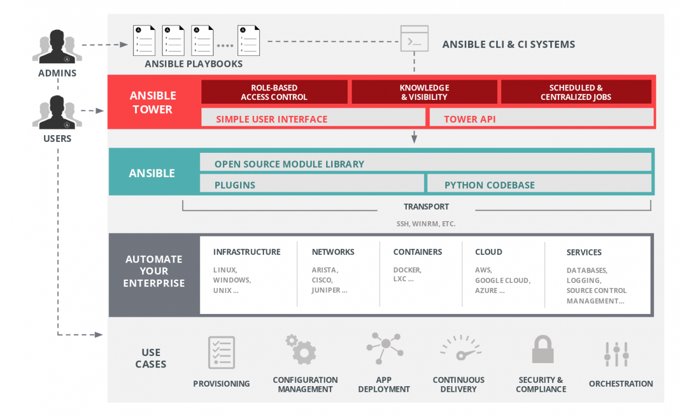
Ansible est à la fois un outil d’automatisation (*control node*) et un langage d’automatisation (*playbook*). Il permet de décrire les opérations (*tasks*) que l’on souhaite exécuter sur un ou plusieurs serveurs (Linux ou Windows, physiques ou virtuels, équipement réseau…) ou sur une API. Cette description représente l’état (*Desired State*) que l’utilisateur souhaite obtenir à la fin de son opération (déploiement, provisioning, orchestration, intégration continue…). Pour cela, le playbook est qualifié de langage *descriptif* – à l’inverse du langage *impératif*, comme le sont les scripts shell ou powershell. Néanmoins ceci est vrai seulement si le développeur a écrit le playbook de manière *idempotente*, signifiant alors qu’il a le même effet qu'on l'exécute une ou plusieurs fois.

L’architecture d’Ansible est représentée dans le schéma suivant. Le control node (appelé ici *automation engine*) s’appuient sur une collection de *modules* et de *plugins* (code développé en Python). A travers une interface en ligne de commande (*CLI*) il offre aux utilisateurs la possibilité d’exécuter leurs playbooks sur des serveurs Linux (en SSH) ou Windows (en WinRM) contenus dans des fichiers d’inventaires (*inventory*). Cela signifie que ce serveur a un accès privilégié à ces serveurs.



L’interface (CLI) et la gestion des accès, mais aussi la scalabilité sont des points faibles à Ansible.

Ansible Tower offre une interface Web (*Web UI*) simple d’utilisation avec une gestion des accès et des délégations parfaitement intégrés. Cette surcouche à Ansible est également *scalable* car elle se déploie en configuration HA avec répartition de charge (*Load Balancing*) et permet aussi de piloter des zones réseaux restreintes avec l’utilisation d’*Isolated Node*(s). Son API est conçue pour que Tower puisse être piloté par un orchestrateur tiers, type CMP (*Cloud Management Platform*) ou CI/CD (GitLab ou Jenkins par exemple).



Parmi la terminologie employée sur Ansible et Tower, on retrouve notamment :

| Terme / composant | Description |
| --- | --- |
| Ansible Playbooks | Fichiers YAML dans lesquels sont mentionnés toutes les tâches qu'Ansible doit exécuter |
| Ansible CLI | L’outil en ligne de commande qui permet d’exécuter du code Ansible sans Tower |
| Web UI | Interface Web simple permettant de piloter l’ensemble de l’infrastructure gérée par Tower, d’exécuter des Job template et d’avoir une vue globale |
| Tower API | Inclut un service API RESTful qui permet aux administrateurs système et aux développeurs d'écrire des scripts et applications de tierce-partie personnalisés pouvant interagir avec Ansible Tower. |
| Inventory | Statique ou dynamique permettant de lister/classer les hosts sur lesquels ansible interviendra |
| Credential | Les credentials sont utilisés par Ansible Tower pour se connecter aux différentes sources extérieures. |
| Project | Les projects sont une collection logique de playbooks qu’utilisent Ansible Tower. Ils représentent généralement les repository SCM (GitLab, Subversion, etc…) contenant les playbooks attenant au job\_template |
| Job Template | Ils permettent aux playbooks d’être contrôler, déléguer et d’être scalable en termes d’organisation.  Ils nécessitent un inventaire, un credential pour se connecter aux VM et un project (SCM ou local) dans lequel seront stockés les playbooks |
| Workflows | Possibilité de lancer un séquencement de playbooks pour aboutir in fine aux actions désirées |
| RBAC | RBAC (Role Based Access Control) permet de gérer les permissions des utilisateurs sur les objets Tower (Inventory, Credential, Projects, Job Templates …) |
| Centralisation des logs | Les logs d’exécution de Ansible Tower peuvent être centralisés sur un puit de log (splunk, logstash, etc ...) |
| Modules | Bouts de codes écrits principalement en python pour modifier l’état de propriété d’un host. Les modules sont invoqués par l’exécution de taches soit directement en ligne de commande soit via des playbooks |
| Plugins | Les Plugins apportent des fonctionnalités complémentaires à Ansible (become ; action ; …) |

## Architecture générale

Ansible Tower se décompose principalement en trois composants :

* Un frontal web Nginx
* Une partie applicative avec notamment le Control Node Ansible
* Une base de données PostgreSQL

Ansible Tower est déployé « On-premises », sur des VMs déployées selon les standards GRDF en vigueur. Pour rendre Ansible Tower hautement disponible, on distingue au minimum 3 VMs :

* Un cluster de 2 nœuds pour la partie frontale et applicative ;
* Ainsi qu’une base de données externalisée sur une VM dédié à cet usage

Cette architecture standard et hautement disponible est supportée par l’éditeur Red Hat et elle est fortement recommandée pour les environnements de production. Elle permet une répartition de la charge (Load-balancing) des connexions utilisateurs et API. Le fonctionnement interne à Tower assure également la répartition de l’exécution des jobs Ansible sur l’ensemble des nœuds configurés.

Note : Parce que non supportée par Red Hat, la réplication de la base de données PostgreSQL n’est pas prévue. Seule la sauvegarde de celle-ci selon la procédure standard livré par Red Hat est utilisée.

Bien que non identifié comme un besoin au moment de la rédaction de document, l’ajout d’un Isolated Node est également possible. Ceci permet à Tower de déléguer l’exécution de Job Ansible à un nœud situé dans une zone réseau différente. Ce type de serveur, ayant un fonctionnement proche d’un rebond SSH ne permet pas de répartir la charge mais plutôt de communiquer avec des serveurs clients situés dans des zones réseau restreintes et de répondre à des contraintes de sécurité.

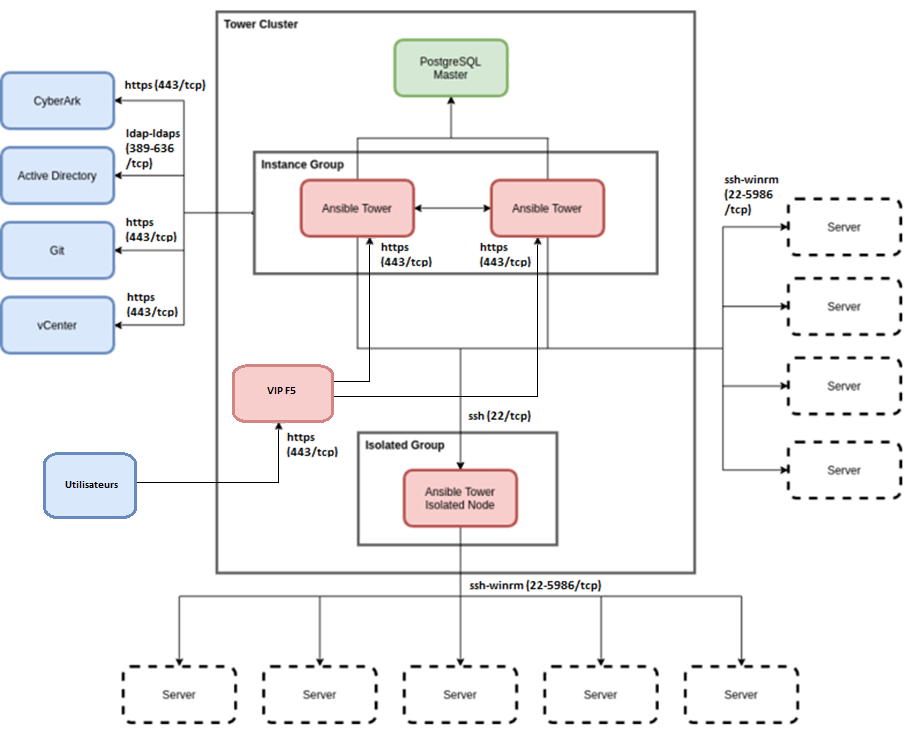
Note : Hors environnement d’Innovation (Bac à sable), il n’est pas nécessaire de déployer un Isolated Node tant que le besoin de son utilisation, pour couvrir une nouvelle zone réseau restreinte normalement, n’est pas exprimé.

La mise à l’échelle (scalabilité) de Tower est également prévue. Avec l’architecture retenue, il sera possible :

* En cas de monter en charge, d’augmenter le nombre de nœuds du cluster Tower ;
* Et, en cas de nouvelle zone réseau restreinte, d’ajouter un Isolated Node

Le schéma ci-après représente l’architecture générale Ansible Tower retenue chez GRDF. On y retrouve :

* Un « Instance Group » comprenant 2 nœuds Ansible Tower
* Une base donnée PostgreSQL dédiée
* Un « Isolated Group » comprenant 1 Isolated Node
* Une VIP portée par le Load Balancer interne GRDF pour permettre l’accès aux utilisateurs



Les composants « Base de données » et « nœuds Tower » sont déployés dans le même sous réseau. L’Isolated Node est lui déployé dans un autre sous réseau, il est accessible en SSH depuis les nœuds Tower.

D’une manière générale, les clients d’Ansible Tower, principalement des serveurs Linux et Windows sont accédés respectivement en SSH et WinRM pour y exécuter les tâches d’automatisation décrites dans les Playbooks Ansible. Dans certains cas, l’Isolated Node sert d’intermédiaire exécuter ces tâches

Pour répondre aux différents cas d’utilisation de la solution Tower (développement, qualification et validation), 3 environnements Tower sont déployés :

* Un cluster de Production
* Un cluster d’Homologation
* Un cluster d’Innovation (anciennement bac à sable)

Note : Il n’y a pas de cluster Tower de Conception. Le cluster d’Homologation est utilisé pour le développement des workflows Ansible ainsi que les tests de montée de version

Chaque cluster se base sur la même architecture telle que décrite précédemment.

### Ansible Tower en Innovation

Ce cluster est temporaire. Installé à des fins de bac à sable, il a pour rôle de qualifier la solution Tower dans l’environnement GRDF sur une souche RHEL8 standard. Après avoir servi pour tester les différents cas d’usage envisagés, il est décommissionné courant 2021.

### Ansible Tower en Production

Ce cluster est déployé sur un socle RHEL8 GRDF standard. Il s’agit de la réelle plateforme de production. Ce cluster a pour rôle de gérer l’ensemble des serveurs GRDF sur tous les environnements (Production, Homologation, Conception, Innovation) et également les environnements Legacy S4. Il est le point d’entrée par défaut. Ce service est ouvert aux équipes d’expertise, d’exploitation et applicatives.

### Ansible Tower en Homologation

Ce cluster, également déployé à partir d’un socle RHEL8 GRDF standard, permet de qualifier les évolutions d’architecture et de configuration de la plateforme. Elle sert aussi au développement des Job Templates et différents workflows.

### Interactions avec Ansible Tower

Pour répondre aux différents Use Case et pouvoir exécuter les tâches automatisées décrites dans ceux-là, Ansible Tower intègre le SI GRDF avec certains privilèges. Il doit pouvoir se connecter à l’ensemble des VMs Linux et Windows mais aussi aux outils de virtualisation (VMware Vsphere), de gestion du code (Gitlab), d’authentification centralisé (Active Directory). Pour permettre l’exécution de requêtes API et la mise en place de workflow externes à Tower, ce dernier doit également être accessible depuis l’orchestrateur vRO de la CMP de GRDF.

|  | Protocole | Tower Homologation | Tower Production |
| --- | --- | --- | --- |
| Vers les VMs Linux de Test | SSH | X |  |
| Vers les VMs Linux d’Innovation | SSH | X | X |
| Vers les VMs Linux de Conception | SSH | X | X |
| Vers les VMs Linux d’Homologation | SSH | X | X |
| Vers les VMs Linux de Production | SSH |  | X |
| Vers les VMs Linux S4 | SSH |  | X |
| Vers les VMs Windows de Test | WinRM | X |  |
| Vers les VMs Windows d’Innovation | WinRM | X | X |
| Vers les VMs Windows de Conception | WinRM | X | X |
| Vers les VMs Windows d’Homologation | WinRM | X | X |
| Vers les VMs Windows de Production | WinRM |  | X |
| Vers les VMs Windows S4 | WinRM |  | X |
| Vers le Vcenter d’administration (inventaire dynamique) | HTTPS | X | X |
| Vers le Vsphere de production/homologation (inventaire dynamique) | HTTPS | X | X |
| Vers le Vsphere de conception/innovation (inventaire dynamique) | HTTPS | X | X |
| Vers le Vsphere de Test (inventaire dynamique) | HTTPS | X |  |
| Vers le serveur Gitlab | HTTPS | X | X |
| Vers l’Active Directory Nova Prod | LDAP | X | X |
| Depuis le vRO d’Homologation | HTTPS | X |  |
| Depuis le vRO de Production | HTTPS |  | X |

## Architecture détaillée

### Version et mise à jour

Les clusters Tower sont initialement déployés en version 3.8.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date de sortie | Fin de support complet | Fin de support étendu n°1 | Fin de support étendu n°2 |
| 3.8 | 18/11/2020 | 17/05/2021 | 18/11/2021 | 18/11/2022 |

La phase de support complet comprend :

* La fourniture de correctifs de sécurité qualifiés comme critique ou important
* La fourniture de correctifs de bugs de priorité urgente et haute
* La fourniture de certaines évolutions des fonctionnalités

La phase de support étendu n°1 comprend :

* La fourniture de correctifs de sécurité qualifiés comme critique
* La fourniture de correctifs de bugs de priorité urgente

La phase de support étendu n°2 comprend :

* La fourniture de correctifs de sécurité qualifiés comme critique

Compte tenu du cycle de vie de Tower qui prévoit une nouvelle version tous les 6 mois environ, une à 2 mises à jour annuelles sont prévues.

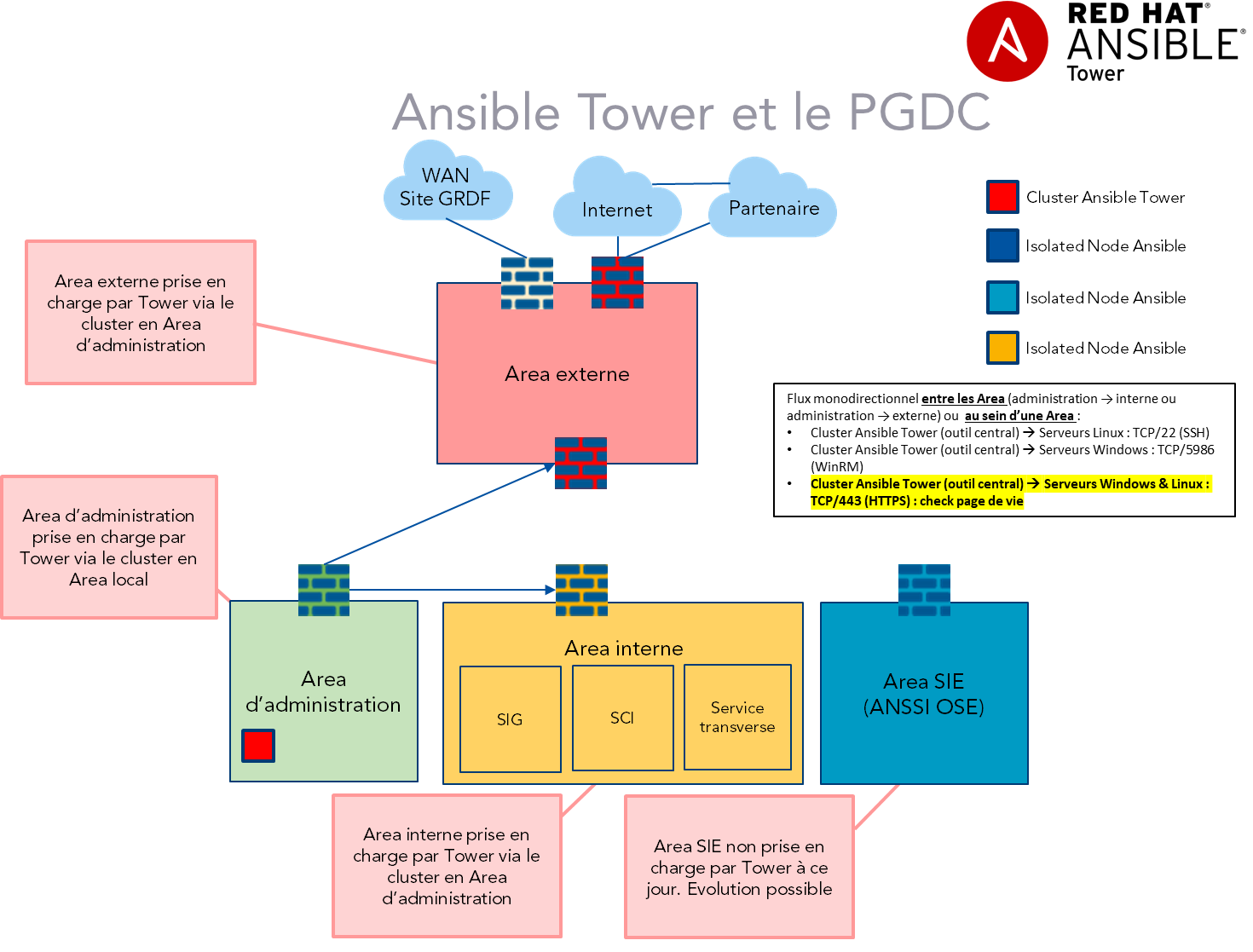
### Liste des serveurs Tower par environnement

La liste ci-dessous comprend les serveurs de type nœuds Ansible Tower et base donnée PostgreSQL. Aucun Isolated Node n’étant déployé initialement, ils n’y apparaissent pas. Cela fera l’objet d’une évolution de l’architecture si nécessaire.

| Environnement | Nom du serveur | Fonction | CPU | Mémoire | Stockage | VLAN | Adresse IP |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Homologation | vlh1tow01.dc.infra-grdf.com | Nœud Tower | 4 | 8 | 50 | Front-3056 | 10.40.11.37 |
| Homologation | vlh1tow02.dc.infra-grdf.com | Nœud Tower | 4 | 8 | 50 | Front-3056 | 10.40.11.38 |
| Homologation | vlh1tow03.dc.infra-grdf.com | BDD | 4 | 8 | 100 | Front-3056 | 10.40.11.39 |
| Production | vlp1tow01.dc.infra-grdf.com | Nœud Tower | 4 | 8 | 50 | Front-3050 | 10.40.9.37 |
| Production | vlp1tow02.dc.infra-grdf.com | Nœud Tower | 4 | 8 | 50 | Front-3050 | 10.40.9.38 |
| Production | vlp1tow03.dc.infra-grdf.com | BDD | 4 | 8 | 100 | Front-3050 | 10.40.9.39 |

### Localisation réseau

Les clusters Tower sont déployés dans l’Area d’administration. Chaque cluster communique vers l’ensemble de ses clients Linux (par le protocole SSH) et Windows (par le protocole WinRM) sans proxy.



### Matrice de flux

Les flux listés ci-dessous sont propres au fonctionnement de Tower dans l’environnement GRDF. Les VMs Tower étant des serveurs RHEL8, les flux standard identifiés pour tout serveur RHEL8 doivent être également autorisés (DNS/NTP/Satellite/Supervision/Sauvegarde…)

#### Tower Homologation

| Source(s) | IP(s) Source | Destination(s) | IP(s) Destination | Protocole(s) | Motif |
| --- | --- | --- | --- | --- | --- |
| vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com | 10.40.11.37  10.40.11.38 | vlh1tow03.dc.infra-grdf.com | 10.40.11.39 | 5432/TCP, UDP | Accès à la base de données PGSQL d’homologation |
| vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com | 10.40.11.37  10.40.11.38 | rhsat.dc.infra-grdf.com  vlp1rhs02.dc.infra-grdf.com | 10.236.197.160  10.40.9.64 | 443/TCP | Permet de récupérer les souscriptions «  Red Hat Ansible Automation » |
| vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com | 10.40.11.37  10.40.11.38 | AREA ADMIN  AREA INTERNE HML  AREA INTERNE HML  AREA INTERNE CCPT  AREA INTERNE CCPT  AREA INTERNE ST HOMOL  AREA INTERNE ST HOMOL  AREA INTERNE ST HOMOL  AREA INTERNE ST HOMOL  AREA EXTERNE HML | 10.40.0.0/20  10.40.64.0/19  10.40.96.0/20  10.40.112.0/20  10.40.128.0/20  10.40.152.0/24  10.40.153.0/24  10.40.155.0/24  10.40.158.0/24  10.40.192.0/19 | 22/TCP  5986/TCP | Accès aux serveurs Linux (SSH) et Windows (WinRM) PGDC et S4 |
| vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com | 10.40.11.37  10.40.11.38 | vxh1vce01.dc.infra-grdf.com  vxp1vce03.dc.infra-grdf.com  vxh1vce01.dc.infra-grdf.com | 10.40.4.134  10.40.4.136  10.40.4.137 | 443/TCP | Génération des inventaires dynamiques sur les Vcenter d’administration, de production/homologation, de conception/innovation et de test |
| vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com | 10.40.11.37  10.40.11.38 | chef-gitlab.grdf.net  vlp1scm01.dc.infra-grdf.com | 10.236.22.145  10.40.150.10 | 443/TCP  443/TCP | Permet de synchroniser les projets Gitlab |
| vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com | 10.40.11.37  10.40.11.38 | nova.infra-grdf.com | 10.236.13.11  10.236.13.12  10.236.13.13  10.236.13.14  10.236.13.69  10.236.13.70  10.236.13.71  10.236.13.72 | 389/TCP + UDP | Permet l’authentification des utilisateurs présents dans l’annuaire LDAP |
| vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com | 10.40.11.37  10.40.11.38 | mailapp.grdf.net | 10.236.30.11 | 25/TCP | Envoi de mail via SMTP (notification, reporting) |
| vxh1vau03.dc.infra-grdf.com | 10.40.11.8 | adam-homol.grdf.net | <IP> | 443/TCP | Permet d’exécuter des requêtes API pour piloter Tower |
| Accès utilisateur (Jump) | 10.236.196.222 | adam-homol.grdf.net | <IP> | 443/TCP | Permet aux utilisateurs de se connecter aux Web UI de Tower |

#### Tower Production

| Source(s) | IP(s) Source | Destination(s) | IP(s) Destination | Protocole(s) | Motif |
| --- | --- | --- | --- | --- | --- |
| vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | 10.40.9.37  10.40.9.38 | vlp1tow03.dc.infra-grdf.com | 10.40.9.39 | 5432/TCP, UDP | Accès à la base de données PGSQL de production |
| vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | 10.40.9.37  10.40.9.38 | rhsat.dc.infra-grdf.com  vlp1rhs02.dc.infra-grdf.com | 10.236.197.160  10.40.9.64 | 443/TCP | Permet de récupérer les souscriptions «  Red Hat Ansible Automation » |
| vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | 10.40.9.37  10.40.9.38 | AREA ADMIN  AREA INTERNE PROD  AREA INTERNE PROD  AREA INTERNE HML  AREA INTERNE HML  AREA INTERNE CCPT  AREA INTERNE CCPT  AREA INTERNE ST  AREA EXTERNE PROD  AREA EXTERNE HML  AREA EXTERNE ST  S4 groupe 1  S4 groupe 2 | 10.40.0.0/20  10.40.16.0/20  10.40.32.0/19  10.40.64.0/19  10.40.96.0/20  10.40.112.0/20  10.40.128.0/20  10.40.144.0/20  10.40.160.0/19  10.40.192.0/19  10.40.240.0/20  10.146.0/16  10.236.0/16 | 22/TCP  5986/TCP | Accès aux serveurs Linux (SSH) et Windows (WinRM) PGDC et S4 |
| vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | 10.40.9.37  10.40.9.38 | vxh1vce01.dc.infra-grdf.com  vxp1vce02.dc.infra-grdf.com  vxp1vce03.dc.infra-grdf.com | 10.40.4.134  10.40.4.135  10.40.4.136 | 443/TCP | Génération des inventaires dynamiques sur les Vcenter d’administration, de production/homologation et de conception/innovation |
| vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | 10.40.9.37  10.40.9.38 | chef-gitlab.grdf.net  vlp1scm01.dc.infra-grdf.com | 10.236.22.145  10.40.150.10 | 443/TCP | Permet de synchroniser les projets Gitlab |
| vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | 10.40.9.37  10.40.9.38 | nova.infra-grdf.com | 10.236.13.11  10.236.13.12  10.236.13.13  10.236.13.14  10.236.13.69  10.236.13.70  10.236.13.71  10.236.13.72 | 389/TCP + UDP | Permet l’authentification des utilisateurs présents dans l’annuaire LDAP |
| vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | 10.40.9.37  10.40.9.38 | mailapp.grdf.net | 10.236.30.11 | 25/TCP | Envoi de mail via SMTP (notification, reporting) |
| vxp1vau03.dc.infra-grdf.com | 10.40.9.25 | adam-homol.grdf.net | <IP> | 443/TCP | Permet d’exécuter des requêtes API pour piloter Tower |
| Accès utilisateur (Jump) | - | adam-homol.grdf.net | <IP> | 443/TCP | Permet aux utilisateurs de se connecter aux Web UI de Tower |

# Installation de la solution

L’installation d’Ansible Tower est réalisée à l’identique en Homologation et en Production. Elle peut se décomposer principalement en 2 étapes, l’installation de base, telle que décrite dans la documentation officielle livrée par Red Hat puis le paramétrage spécifique GRDF, décrite dans la suite de ce chapitre.

## Installation de base

L’installation de base est réalisée à partir d’un socle RHEL8 GRDF standard puis :

* La communication entre les serveurs Tower est mise en place.
  + Création d’une paire de clé SSH sur le premier nœud et la copie de la clé publique vers les autres nœuds
  + Modification de la configuration SSHD pour permettre la connexion en tant que root (sans mot de passe, avec clé SSH)
  + Ajout de l’utilisateur root au groupe ssh-trusted pour autoriser les connexions SSH entrantes.
* Adaptation des RI pour les 2 nœuds Tower (hors BDD)
  + Retrait de l’option « noexec » sur les Filesystem /var/tmp et /tmp (provoque un échec dans l’exécution du playbook
  + Désactivation permanente du booléen SELinux deny\_execmem (bloque le chargement du manifest de la souscription Tower)
* Installation d’Ansible Tower en mode déconnecté
  + Copie du bundle d’installation Tower
  + Modification du fichier d’inventaire
    1. Ajout des nœuds Tower dans le groupe « tower »
    2. Ajout de la Base de données dans le groupe « database » et dans la variable « pg\_host » et du port par défaut « 4321 »
    3. Positionnement des mots de passe des comptes locaux « admin » et de la base de données postgresql
  + Lancement de l’installation sans option (./setup.sh)
* L’ajout du manifest (souscriptions Tower) : 3000 Hosts disponibles à l’écriture de cette documentation répartis de la manière suivante :
  + 2900 Hosts pour le cluster de Production
  + 100 Hosts pour le cluster d’Homologation

## Installation spécifique GRDF

L’installation spécifique prévoit principalement le paramétrage des éléments suivants :

* Déploiement du script de sauvegarde
* Configuration des modules disponibles pour les commandes Ad’hoc :
  + Modules communs : debug, meta, setup
  + Modules RHEL : command, group, mount, ping, raw, service, service\_facts, shell, user, yum
  + Modules Windows : win\_feature, win\_group, win\_ping, win\_regedit, win\_service, win\_updates, win\_user,
* Configuration de l’intervalle de communication de l’Isolated Node à 30 secondes
* Désactivation de la vérification SSL pour les communications avec Gitlab
* Configuration de la durée de rétention du Fact Cache à 24h
* Configuration de l’authentification LDAP avec l’Active directory
  + Authentification avec l’AD Nova Prod
  + Configuration d’un Team mapping pour les équipes admin de Tower (AES)
* Création des Credentials de base (Ansible Galaxy, VMware Vcenter, Gitlab)
* Création des inventaires de base (VMware, statiques pour serveurs physiques…)
* Création des projets SCM (GitLab) de base

L’installation spécifique est décrite sous forme de Playbook/Role Ansible dans le projet Gitlab « postinstall\_tower » qui permet de déployer de manière idempotente le paramétrage décrit précédemment sur les Cluster de Production et d’Homologation.

## Nommage des objets Tower

Dans son fonctionnement normal, Tower manipule différents objets (ressources), tels que les inventaires, les *credentials*, les projets et les *job templates*. Pour des besoins organisationnels, la norme de nommage suivante est utilisée pour chaque type d’objets.

| Type d’objet | Norme de nommage | Remarques |
| --- | --- | --- |
| Credential | *crd\_<type\_credential>\_<description\_credential*  Avec :   * type\_credential: linux, windows, scm, vcenter, vault, cyberark, galaxy, ou tower * description\_credential: champ libre sans séparateur   Exemple :   * crd\_linux\_homologation * crd\_vcenter\_vxp1vce01 | Un quatrième champ peut être ajouté dans certains cas pour préciser d’avantage la description de l’objet |
| Inventory | *inv\_<type\_inventaire>\_<description\_inventaire>*  Avec :   * type\_inventaire : dynamic, static ou smart * description\_inventaire : champ libre sans séparateur   Exemple :   * inv\_dynamic\_vmware * inv\_static\_rhel8physical | Un quatrième champ, voire un cinquième, peuvent être ajoutés dans certains cas pour préciser d’avantage la description de l’objet |
| Project | *prj\_<nom\_gitserver>\_<repo\_gitserver>*  Avec :   * nom\_gitserver : nom réseau du serveur Git * repo\_gitserver : nom du repository Git   Exemple :   * prj\_chefgitlab\_provisioning * prj\_chefpgdc\_deploy\_snmp | Un quatrième champ peut être ajouté dans certains cas pour préciser d’avantage la description de l’objet |
| Job Template | *[sjt,wjt]\_<champ1>\_<champ2>*  Avec :   * sjt/wjt : [simple|workflow] Job Template * champ1 : use case/projet du Job Template * champ2 : fonction du Job Template (facultatif)   Exemple :   * sjt\_provisioning\_rollback * wjt\_customfacts\_deploy | Un quatrième champ peut être ajouté dans certains cas pour préciser d’avantage la description de l’objet |

# Cas d’utilisation

Ces cas d’utilisation s’appliquent à la gestion des VM hébergées sur le socle VMware mais pourra être étendu aux serveurs physiques si possible.

## Post-provisioning d’une VM

Tower est piloté par vRO (VMware vRealize Orchestrator) pour exécuter les actions de Post-provisioning sur les VMs Linux et Windows.

Le graphique ci-dessous décrit chaque étape du Post-provisioning. Les pastilles jaunes sont celles déclenchées par vRA et ne sont pas détaillées ici. Les pastilles bleues sont des actions orchestrées par Tower et sont déclenchées par vRO via l’API de Tower.

| Etape | Description |
| --- | --- |
| Création d'un inventaire temporaire | vRO envoie une requête API à Tower pour créer un inventaire vide. Celui-ci porte le nom de la VM qui sera créée et est destiné à être supprimé à la fin du traitement  L’utilisation de l’inventaire temporaire a plusieurs avantages :   * Il permet de ne pas dépendre de la synchronisation de l’inventaire dynamique * Il permet de cibler précisément la VM sur laquelle exécuter le Job Template de Post-provisioning, indépendamment d’une limite d’inventaire |
| Ajout de la VM à l'inventaire temporaire | vRO envoie une requête API à Tower pour ajouter la VM (en utilisant son FQDN/adresse IP) dans l’inventaire temporaire créé lors de l’étape précédente |
| Exécution du JT de Post-provisioning | vRO envoie une requête API à Tower pour exécuter le Job Template de Post-provisioning sur l’inventaire temporaire créé lors des étapes précédentes.  vRO envoie les extra-vars nécessaires à l’exécution du Job Template de Post-provisioning (nom de la VM, offre/environnement/taille du disque de données…).  Ce Job Template comprend notamment les tâches de configuration de la VM et d’intégration de celle-ci aux différents outils d’exploitations (agents) |
| Suppression de l'inventaire temporaire | vRO envoie une requête API à Tower pour supprimer définitivement l’inventaire temporaire |

Si l’une des tâches exécutées via l’API de Tower renvoie une erreur (code HTTP différent de 201 ou 204), le Rollback (retour arrière) sur le Provisioning s’exécute alors, comme décrit dans le graphique ci-après.

| Etape | Description |
| --- | --- |
| Exécution du JT de Rollback sur le Provisioning | vRO envoie une requête API à Tower pour exécuter le Job Template de Rollback sur le Provisioning sur l’inventaire temporaire créé lors des étapes précédentes.  vRO envoie les extra-vars nécessaires à l’exécution de ce Job Template.  Ce Job Template comprend la suppression de la VM parmi les différents outils d’exploitations (agents)  Si un échec survient lors du rollback, la notification envoyée à l’utilisateur précisera que des actions manuelles devront être réalisées pour terminer le nettoyage |

## Décommissionnement d’une VM

Tower est piloté par vRO pour exécuter les actions de décommissionnement des VMs Linux et Windows.

Note : Comme pour le Post-provisioning, l’utilisation d’un inventaire temporaire est privilégiée lors du décommissionnement d’une VM. Cela permet d’exécuter le Job Template de décommissionnement même dans le cas où la VM n’est pas présente dans l’inventaire dynamiquement (cas du décommissionnement appelé dans les heures de la création de la VM)

Le graphique ci-dessous décrit chaque étape du décommissionnement d’une VM. Les pastilles jaunes sont celles déclenchées par vRA et ne sont pas détaillées ici. Les pastilles bleues sont des actions orchestrées par Tower et sont déclenchées par vRO via l’API de Tower.

| Etape | Description |
| --- | --- |
| Création d'un inventaire temporaire | vRO envoie une requête API à Tower pour créer un inventaire vide. Celui-ci porte le nom de la VM qui sera créé et est destiné à être supprimé à la fin du traitement |
| Ajout de la VM à l'inventaire temporaire | vRO envoie une requête API à Tower pour ajouter la VM, en utilisant son FQDN/adresse IP, à l’inventaire temporaire créé lors de l’étape précédente |
| Exécution du JT de décommissionnement | vRO envoie une requête API à Tower pour exécuter le Job Template de Décommissionnement sur l’inventaire temporaire créé lors des étapes précédentes.  vRO envoie les extra-vars nécessaires à l’exécution de ce Job Template (nom de la VM…)  Ce Job Template comprend les tâches de nettoyage de la VM et des différents outils d’exploitations (agents)  Si un échec survient lors du décommissionnement, la notification envoyée à l’utilisateur précisera que des actions manuelles devront être réalisées pour terminer le nettoyage |
| Suppression de l'inventaire temporaire | vRO envoie une requête API à Tower pour supprimer définitivement l’inventaire temporaire |

## Déploiement de middleware

Les middlewares concernés par ces déploiements sont ceux de la politique industrielle GRDF, à savoir :

* Apache Httpd
* Nginx +
* Tomcat
* OpenJDK
* PostgreSQL
* EDB (Enterprise DB)

Les déploiements de ces middlewares sont pilotés par vRO dans le cadre du Post-provisioning avec middleware et peuvent être exécutés sur une VM existante

### Déploiement de middleware lors du Post-provisioning

Dans le premier cas, voici le graphique qui décrit chaque étape de déploiement d’un middleware dans la continuité du Post-provisioning d’une VM.

| Etape | Description |
| --- | --- |
| Création d'un inventaire temporaire | vRO envoie une requête API à Tower pour créer un inventaire vide. Celui-ci porte le nom de la VM qui sera créée et est destiné à être supprimé à la fin du traitement  L’utilisation de l’inventaire temporaire a plusieurs avantages :   * Il permet de ne pas dépendre de la synchronisation de l’inventaire dynamique   Il permet de cibler précisément la VM sur laquelle exécuter le Job Template de Post-provisioning, indépendamment d’une limite d’inventaire |
| Ajout de la VM à l'inventaire temporaire | vRO envoie une requête API à Tower pour ajouter la VM (en utilisant son FQDN/adresse IP) dans l’inventaire temporaire créé lors de l’étape précédente |
| Exécution du JT de Post-provisioning | vRO envoie une requête API à Tower pour exécuter le Job Template de Post-provisioning sur l’inventaire temporaire créé lors des étapes précédentes.  vRO envoie les extra-vars nécessaires à l’exécution du Job Template de Post-provisioning (nom de la VM, offre/environnement/taille du disque de données…).  Ce Job Template comprend notamment les tâches de configuration de la VM et d’intégration de celle-ci aux différents outils d’exploitations (agents) |
| Exécution du JT de déploiement du middleware | vRO envoie une requête API à Tower pour exécuter le Job Template de déploiement du middleware sur l’inventaire temporaire créé lors des étapes précédentes.  vRO envoie les extra-vars nécessaires à l’exécution de ce Job Template (nom d’une instance, offre/environnement/taille du disque de données…). |
| Suppression de l'inventaire temporaire | vRO envoie une requête API à Tower pour supprimer définitivement l’inventaire temporaire |

Si l’une des tâches exécutées via l’API de Tower renvoie une erreur (code HTTP différent de 201 ou 204), le Rollback (retour arrière) sur le Provisioning s’exécute alors (identique à celui vu précédemment pour le cas d’utilisation : Post-provisioning d’une VM).

### Déploiement de middleware sur une VM existante

Il est également possible de déployer les middlewares sur une VM existante. Pour cela un Job Template par middleware est présent avec un Survey (questionnaire) pour renseigner la valeur des différentes variables (versions du middleware, nom d’une instance, répertoire de déploiement …). Ce déploiement de middleware peut être initié par vRA ou directement à partir de Tower.

Note : Le déploiement des middlewares hors politique industrielle peut être orchestré directement depuis Tower

## Conformité

La conformité permet de vérifier que les VMs Linux et Windows sont conformes aux règles d’ingénierie GRDF. Certains écarts de configurations sont rectifiés automatiquement, d’autres génèrent des alertes uniquement.

Ce cas d’utilisation est traité par Tower directement (aucune interaction avec vRO) à travers le Job Template de Conformité avec deux paramètres :

1. L’inventaire sur lequel est réalisé est ces vérifications (Test, Conception, Homologation et Production)
2. L’éventuelle « limite » sur cet inventaire pour restreindre l’exécution sur des VMs ciblées

Note : Le choix de l’inventaire est déterminant lors de l’exécution du Job Template de conformité. Un inventaire dynamique est prévu par offre (Conception, Homologation, Production…). Ceux-ci doivent être utilisés par exécuter le Job Template de conformité.

Un Job Template par type d’OS (RHEL et Windows) est présent.

## Livraison applicative

A instruire

# Maintenance, Sécurité et Exploitation

## Supervision

En plus des métriques systèmes classiques générales à toutes les VMs chez GRDF, les services suivants sont supervisés. Ceci permet de réagir rapidement lorsqu’un composant principal de Tower est hors service (frontal web, application, base de données).

| Fonction Tower | VM Tower | Services à superviser |
| --- | --- | --- |
| Nœud Tower | vlh1tow01.dc.infra-grdf.com  vlh1tow02.dc.infra-grdf.com  vlp1tow01.dc.infra-grdf.com  vlp1tow02.dc.infra-grdf.com | ansible-tower.service  nginx.service  redis.service  supervisord.service |
| Base de données Tower | vlh1tow03.dc.infra-grdf.com  vlp1tow03.dc.infra-grdf.com | postgresql.service |

Si l’un de ces services n’est pas dans l’état active ou qu’un code de retour est différent de 0 lors de l’interrogation de son état, une alerte remonte dans la console de supervision et une alerte est envoyée par courriel au support système Niveau 3.

## Accès

Le tableau ci-dessous permet d’identifier les accès entrants et sortants et la méthode de connexions des utilisateurs et/ou automates.

| Type d’accès | Source / Destination | Description | Méthode de connexion | Identifiant |
| --- | --- | --- | --- | --- |
| Entrant | Utilisateurs administrateurs | Permet aux administrateurs de se connecter aux Web UI de Tower | * Compte local * Connexion sécurisée HTTPS | **admin** |
| Entrant | Utilisateurs administrateurs | Permet aux administrateurs de se connecter aux Web UI de Tower | * Compte personnel centralisé dans l’Active Directory Nova ou Compte de service via Cyberark * Connexion sécurisée HTTPS | Compte nominatif |
| Entrant | Utilisateurs sans privilèges | Permet aux utilisateurs de se connecter aux Web UI de Tower uniquement en tant qu’une « Team » ciblée | * Compte personnel centralisé dans l’Active Directory Nova ou compte de service via Cyberark * Connexion sécurisée HTTPS | Compte nominatif |
| Entrant | VMware vRealize Orchestrator Production | Permet d’exécuter des requêtes API pour piloter Tower | * Compte de service centralisé dans l’Active Directory Nova * Mot de passe sauvegardé dans vRA/vRO | **S-TOW-PR1-VAU @nova.infra-grdf.com** |
| Entrant | VMware vRealize Orchestrator Homologation | Permet d’exécuter des requêtes API pour piloter Tower | * Compte de service centralisé dans l’Active Directory Nova * Mot de passe sauvegardé dans vRA/vRO | **S-TOW-HM1-VAU @nova.infra-grdf.com** |
| Sortant | VMs Linux | Permet d’exécuter des playbooks | * Compte de service Linux local * Connexion sécurisée SSH * Clé privée avec passphrase sauvegardée dans BDD Tower (Credential) | **tower**  Clé publique SSH propre à chaque offre :   * Innovation * Conception * Homologation * Production |
| Sortant | VMs Windows | Permet d’exécuter des playbooks | * Compte de service centralisé dans l’Active Directory Nova * Connexion sécurisée WinRM/HTTPS * Mot de passe sauvegardé dans BDD Tower (Credential) | **S-DIO-ANS-PROD@nova.infra-grdf.com**  **S-DIO-ANS-HPROD@nova-preprod.infra-grdf.com** |
| Sortant | VMware Vcenter | Permet de synchroniser les inventaires dynamiques VMware (lecture seule) | * Compte de service centralisé dans l’Active Directory Nova * Connexion sécurisée HTTPS * Mot de passe sauvegardé dans Tower (Credential) | **S-VMW-PR1-RO@nova.infra-grdf.com** |
| Sortant | Serveur Gitlab | Permet de synchroniser les projets Gitlab | * Compte personnel centralisé dans l’Active Directory Nova * Connexion sécurisée HTTPS * Mot de passe sauvegardé dans Tower (Credential) | **S-GIT-PR1-TOW@nova.infra-grdf.com** |
| Sortant | Active Directory | Permet l’authentification des utilisateurs présents dans l’annuaire LDAP | * Compte de service centralisé dans l’Active Directory Nova * Connexion LDAP * Mot de passe sauvegardé dans Tower (Settings/Authentication) | **S-COM-P1-AD\_Read@nova.infra-grdf.com** |
| Sortant | Red Hat Satellite 6 | Permet de récupérer les souscriptions «  Red Hat Ansible Automation » | * Compte de service centralisé dans l’Active Directory Nova * Connexion LDAP * Mot de passe sauvegardé dans Tower (Settings/Licence) | **S-RHS-PR1-TOW@nova.infra-grdf.com** |

## Sauvegarde et restauration

Deux types de sauvegardes sont mises en place.

### Via Hyperviseur

Comme toutes les VMs RHEL chez GRDF, une sauvegarde régulière est effectuée en utilisant la technologie Commvault.

Cette sauvegarde a l'avantage de travailler au niveau "macro" et de sauvegarder l'intégralité pour une reprise rapide sur incident. En revanche, il n'assure pas la cohérence des données.

Il faudrait arrêter les services de l'outil Tower (Nœuds + Base de données) durant une plage horaire et effectuer les sauvegardes (Nœuds + Base de données) durant cette plage pour éviter tout problème de cohérence des données. Pour cela, les fichiers de sauvegarde logicielle (détaillée ci-après) pourront être récupérés avec les outils Commvault et utilisés en cas de restauration Tower.

L’agent Commvault n’est pas nécessaire à la mise en place de cette sauvegarde.

### Logicielle

Une seconde sauvegarde doit être effectuée via l’outil mis à disposition de Tower.

Cet outil permet d'effectuer des sauvegardes full. Il intègre l’arrêt des services (Nœuds + Base de données), la sauvegarde de tous les éléments Tower puis le redémarrage des services.

Il est exécuté via un script planifié (cron) sur l’un des nœuds Tower. Son exécution est quotidienne et prévoit la conservation des 10 dernières sauvegardes. Le script utilisé est disponible dans GitLab et déployé via le Role Ansible role\_tower\_config.

Ces backups seront utilisés dans le cas de reprise sur un incident pour la reconstruction d'un cluster Tower complet (Nœuds + Base de données) à H-24 maximum pour la sauvegarde la plus récente et J-10 pour la plus ancienne.

Cette sauvegarde est exécutée sur chaque cluster Tower (Homologation et Production). Ces sauvegardes n’ont pas besoin d’être externalisées.

## Maintenance et arrêt

Pourront être planifiés chaque année plusieurs créneaux d’une demi-journée pour les mises à jour et évolutions de l’architecture de Tower. Ces créneaux de maintenance seront communiqués à l’avance et se dérouleront en semaine. Pendant ces opérations, Tower ne sera pas disponible aux utilisateurs et autres service tiers.

Note : Idéalement, cette opération de maintenance sera signalée au niveau de la VIP de chaque cluster pour que les utilisateurs ne puissent se connecter le temps nécessaire.

Avant chaque opération de maintenance, une sauvegarde logicielle est réalisée

A des fins de qualifications, les mêmes opérations de maintenance sont réalisées sur les 2 clusters, d’abord sur le celui d’Homologation, puis celui de Production :

* Montée de version mineure ou majeure
* Mise à l’échelle (ajout d’un ou plusieurs Node) ou ajout d’un Isolated Node

## Reprise sur incident et PCI

Les serveurs Ansible Tower s’inscrivent dans le plan de continuité informatique par le socle de virtualisation VMware.

### Sinistre local (SL)

En cas d’incident sur l’un des nœuds Tower, la continuité informatique est garantie par la fonctionnalité « HA » de Tower. L’autre nœud (ou les autres nœuds dans le cas d’une extension du cluster dans le futur) gèrera-ont l’ensemble des accès des utilisateurs (Web UI ou API) jusqu’à la remise en fonction du nœud faisant défaut. La configuration de la VIP de chaque cluster Tower permet de détecter une erreur de communication avec les nœuds pour désactiver le(s) nœud(s) en erreur.

En cas d’incident sur la base de données Tower, une restauration de la VM Tower (Via Hyperviseur) devra être réalisée, et également, si besoin une restauration de la base de données. Ce type d’incident provoquera forcément une perte de service pouvant aller à plusieurs heures.

### Sinistre majeur (SM)

En cas de perte d’un hyperviseur, les mécanismes de haute disponibilité VMware se chargent automatiquement de la continuité de services des VMs Tower sur un autre hyperviseur.

Hébergées par le POD VMware d’Admin, les VMs Tower sont exécutées par défaut sur le site E3. En cas de la perte de ce site, les mécanismes de haute disponibilité VMware se chargent automatiquement de la continuité de services des VMs Tower sur le site distant (E4).

Faire un lien vers le ISP

Perte de données admissible

Temps d’indisponibilité admissible