RAPPORT DE STAGE

BTS Services Informatiques aux Organisations
Option Solutions Logicielles et Applications Métiers
2eme années

Evan AOUSTIN
janvier/février 2026

Tuteur de stage : Corinne NOVALES
Maître de stage (entreprise) : Yohann NICOLAS
Établissement : Lycée Honoré d'Estienne d'Orves
Entreprise d'accueil : Modulus (MC)

Sommaire

Présentation de l’entreprise

L’entreprise MODULUS évolue dans le secteur du casino terrestre (pas de jeux en ligne) et fournit un système de gestion et de supervision utilisé pour l’exploitation des machines et des opérations de jeu.

Un élément technique clé dans ce type d’architecture est l’interface côté machine. Bien que MODULUS ne crée pas de machines à sous et ne gère pas les systèmes de jeux, il est présent sur toutes les machines à sous du monde (sauf très anciennes) une SMIB (Slot Machine Interface Board).

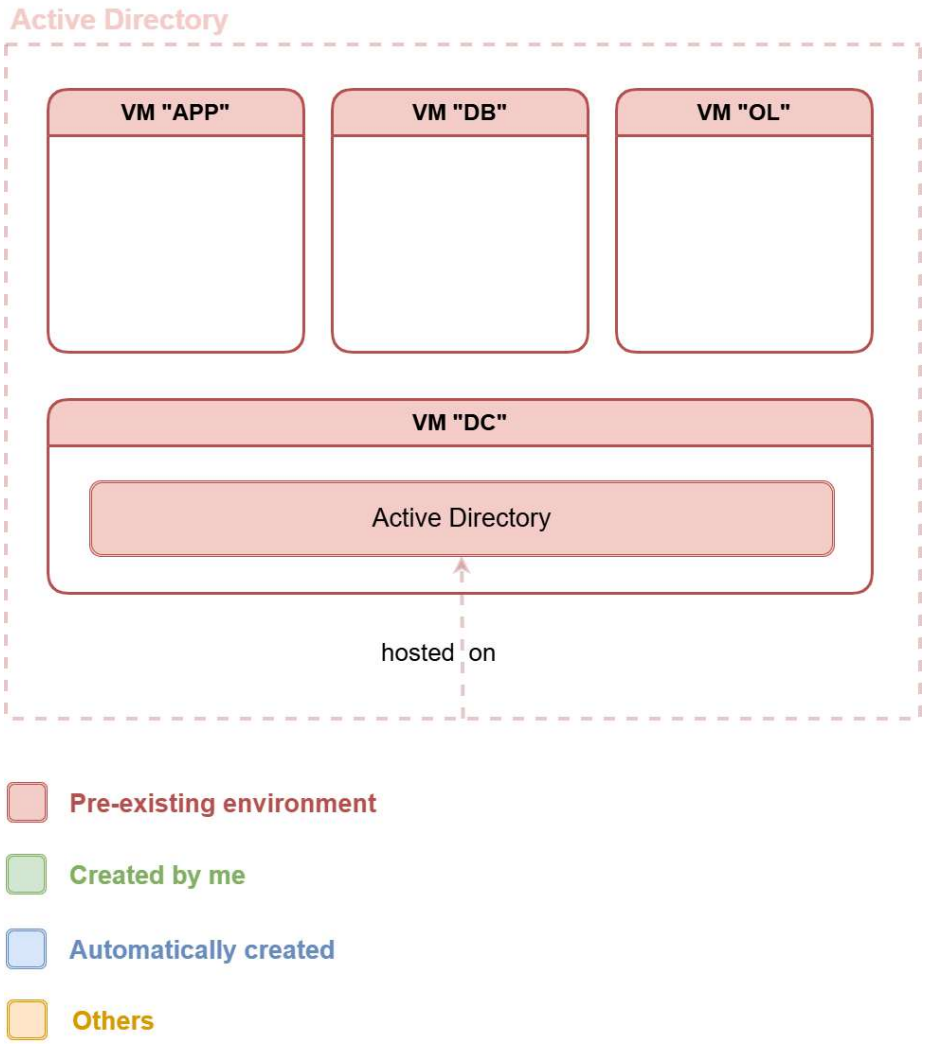
Une SMIB est une interface qui sert de pont de communication entre une machine à sous et le système de gestion du casino : elle facilite l’échange d’informations et l’intégration de machines pouvant avoir des protocoles différents.
Elle fournit et maintient un système central (casino management / gaming system) et son écosystème d’intégration, dont la SMIB constitue une brique terrain côté machine.

L’approche est orientée exploitation : fiabilité du fonctionnement, continuité de service et capacité à exploiter les données pour piloter l’activité.
MODULUS est présent dans plus de 320 casinos et a installé plus de 40 000 SMIB à travers le monde.

Présentation de la situation existante

L'environnement actuel, déployé chez le client (un client correspond à un casino), repose sur quatre machines virtuelles (VM), généralement virtualisées sur l'hyperviseur Proxmox.

- **VM Windows "APP"** : héberge l'ensemble des services applicatifs.
- **VM Windows "DB"** : héberge la base de données.
- **VM Windows "OL"** (Floor) : correspond au composant Floor, utilisé pour la gestion et le suivi des éléments liés au parc de machines / activité en salle.
- **VM Windows "DC"** joue le rôle de contrôleur de domaine et héberge Active Directory.



Présentation des résultats à obtenir

Les objectifs sont nombreux, mais peuvent se résumer à : **la conception d'un environnement dit : Infrastructure as Code (IaC) and Continuous Delivery (CD).**

Aujourd'hui, l'ensemble des services de MODULUS tourne chez le client (prod), ainsi que dans ses environnements internes (dev, test…), sur des machines Windows. Ces services sont développés en .NET, une plateforme historiquement centrée sur Windows. Depuis .NET 5, Linux est une cible officiellement supportée et peut être privilégiée dans certains cas, notamment pour des raisons de performance et d'optimisation des ressources.

Dans cette optique, certains services ont déjà été conteneurisés dans des conteneurs Docker par leurs équipes, afin de remplacer progressivement des services exécutés directement sur Windows par des services exécutés sur Linux, dans des conteneurs, et de simplifier la maintenance et la mise à jour des environnements.

La migration complète côté client n'est pas immédiate : dans un contexte aussi contraint qu'un casino, elle nécessite de valider les choix techniques et de mener de nombreux tests.

L'objectif ici est donc de réaliser un POC d'Infrastructure as Code permettant de provisionner de nouvelles VM Linux (OS : Rocky), de les configurer automatiquement, d'y déployer les composants nécessaires — dont le cœur applicatif : les conteneurs — et de garantir une mise à jour continue de l'environnement.

Organisation du travail et suivi des tâches

Avant d'arriver en stage, je savais déjà que j'allais travailler sur Terraform et Ansible (outils d'automatisation, de provisionnement et de configuration de VM, évoqués plus tard) et que j'allais effectuer tout cela sur l'hyperviseur Proxmox, j'ai donc pu me renseigner sur le concept de ces outils (vision haut niveau uniquement)

Les premiers jours de stage, j'ai pu avoir une vision plus bas niveau de chaque outil dont j'allais avoir besoin, consultation des docs de Terraform et Ansible, discussion et démonstration de Proxmox, de Docker, de Portainer, du concept de stacks (tout cela sera évoqué par la suite)

En bref, un apprentissage indispensable avant d'attaquer le cœur du métier, les semaines suivantes étaient dédiées au développement du projet, toujours aidé par les bonnes personnes en fonction des questions et problèmes rencontrés.

J'ai aussi pu assister à des réunions intéressantes avec toute l'équipe, notamment des réunions "sprint planning" toutes les deux semaines, le sprint planning est une étape des méthodologies Agile au cours de laquelle les équipes décident des tâches à accomplir lors du prochain sprint. Ainsi que d'autres réunions moins techniques, pour évoquer des problèmes, des idées… autour de l'entreprise.

Tout au long des semaines, j'ai donc pu être accompagné, j'ai aussi, pour la partie réalisation du projet, créé un diagramme d'architecture, afin de présenter le plus facilement possible mon projet, diagramme constamment amélioré au fur et à mesure des "fonctionnalités" nécessaires.

Liste des fonctionnalités ou missions / tâches

1) Mise en place de Terraform et Terragrunt

La première étape est la création et le maintien automatisés des VMs, avec une logique IaC.

Quand une VM Linux est créée, elle doit être jointe à un environnement cité précédemment. Nous passons donc de 4 à 5 VMs, avec donc deux VM dites “APP” qui hébergent l’ensemble des services applicatifs : une Rocky Linux pour les services conteneurisés “APP-ROCKY” (créée par Terraform) et une avec les services non conteneurisés “APP” (déjà existante donc).

Pour cela, le choix architectural s’est tourné vers Terraform. Cet outil permet d’écrire, via des fichiers de configuration texte de type (yml, hcl, tf), l’infrastructure souhaitée.
(VMs, réseaux, disques…)

Terraform est compatible avec de nombreuses APIs d’hyperviseurs, dont celle de Proxmox. À l’exécution (une seule commande), Terraform utilise donc l’API de Proxmox pour créer l’infrastructure souhaitée.

Ci-dessous, une version très simplifiée de la structure de Terraform.

```
resource "proxmox_vm_qemu" "rocky" {
  count        = 1
  name         = “vm-test”

      cpu {
            cores    = 3
            type     =  "x86-64-v3"
      }
      disk {
            size      = 50
      }
}
```

Une surcouche nommée Terragrunt a été ajoutée par la suite à Terraform, elle permet notamment de gérer plusieurs VMs, en ayant plusieurs configurations différentes, permettant ainsi de gérer plusieurs environnements (test, dev, acceptance, intégration…) via une configuration réseau, par exemple.

Exemple simplifié d'une possibilité de Terragrunt :

Environnement QA-ACCEPTANCE :

```
inputs = {
  vm_name_prefix = "vm-acceptance-app"

  vm_memory = 4096

  vm_bridge_1         = "bridge-acceptance"
}
```

Environnement QA-INTEGRATION :

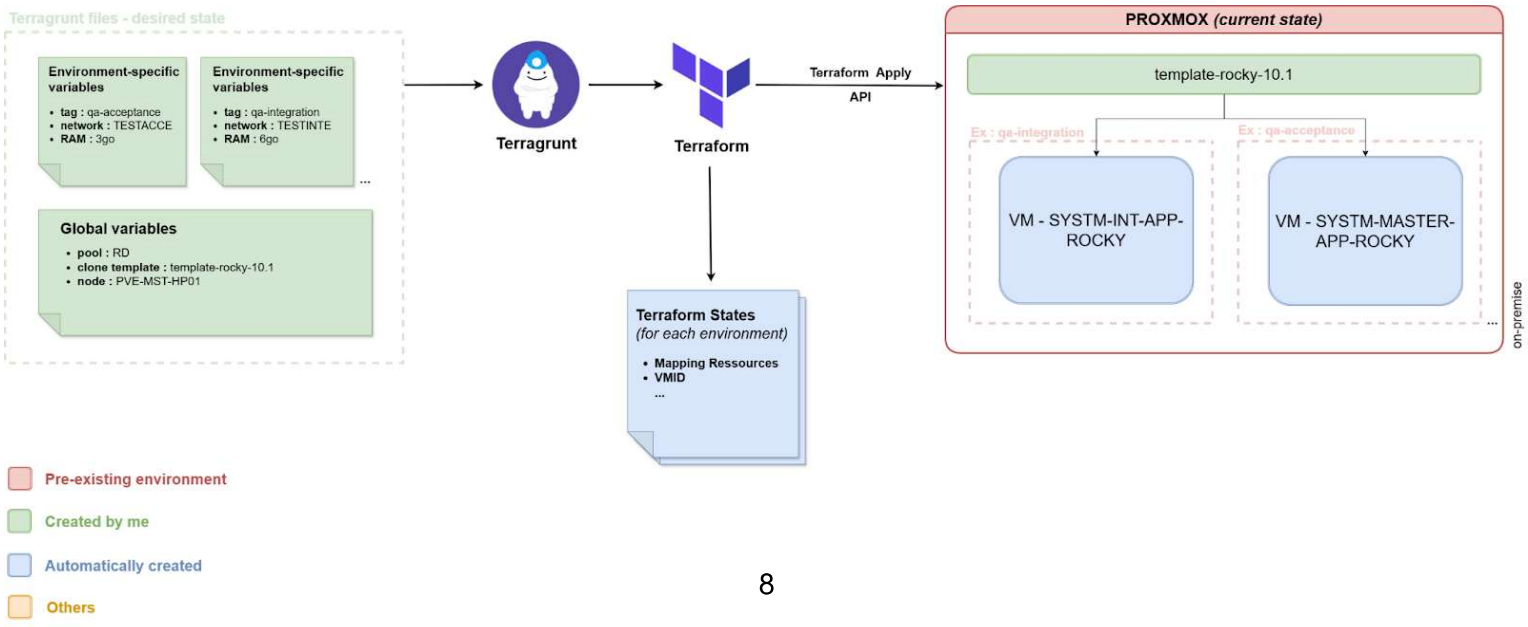
```
inputs = {
  vm_name_prefix = "vm-integration-app"

  vm_memory = 3072

  vm_bridge_1         = "bridge-integration"
}
```

Dans cet exemple, deux VM sont créées dans deux bridges réseau différents, et une a plus de RAM que l'autre.

Diagramme récapitulatif de ce que fait Terraform (et Terragrunt) :



2) Mise en place de Ansible

Une fois les VMs créées, celles-ci doivent être configurées, de manière 100 % automatique. Pour cela, le choix architectural s'est tourné vers Ansible.

Ansible permet d'automatiser la configuration en se connectant en SSH à la VM, il peut donc exécuter n'importe quelle tâche sur celle-ci.

En clair :

	Objectif	Exemple	Méthodes
Terraform	IaC / provisioning	créer l'infra : VMs, réseaux, disques…	API hyperviseur
Ansible	configuration / orchestration	configurer l'OS, installer des paquets, déployer applications…	ssh

Ansible a de nombreuses responsabilités : il doit créer des users, configurer un firewall, installer des packages indispensables et bien d'autres, mais nous ne nous attarderons pas là-dessus, et nous concentrerons sur le plus "important".

De manière simplifiée, Ansible :

- Installe tous les prérequis à son fonctionnement (exemple : Python)
- Crée les users, le firewall, l'ouverture des ports…
- Installe et configure Docker
- Installe et configure Portainer (interface de gestion de Docker)
- Crée des stacks Portainer à partir de fichiers docker-compose sur Bitbucket (un fichier docker-compose.yml est dans notre configuration stockée sur Bitbucket (dépôts Git, cloud) ; il sert à définir la configuration d'un ou plusieurs conteneurs. Dans notre configuration, il indique notamment où se situe l'image Docker, à savoir sur ProGet, utilisé ici comme Docker Registry)
- En créant ces stacks, Docker va automatiquement déployer les conteneurs
- Configure aussi le poll automatique sur Portainer, c'est-à-dire que Portainer va, toutes les minutes (durée configurable), vérifier si le docker-compose sur Bitbucket a été modifié. Le cas échéant, Portainer va redéployer Docker, et ainsi Docker va aller chercher la nouvelle version du service (conteneur) sur la Registry Docker (ProGet), mettant ainsi à jour l'environnement

- Faire la “jointure” avec l’environnement : comme vu précédemment, l’objectif est d’ajouter cette VM à un environnement déjà existant de 4 VMs et il ne suffit pas simplement d’être dans le même réseau. La VM Rocky Linux “APP-ROCKY” doit être jointe avec sa VM dite “compagne”, la VM Windows “APP”. Pour cela, Ansible fait un mount du disque “M:” de la VM Windows “APP”, et modifie des fichiers de configuration en inscrivant l’IP de la nouvelle VM Rocky Linux “APP-ROCKY”. Ainsi, le système pointera bien (pour les services conteneurisés uniquement) vers la nouvelle VM.

Exemple d’une “task” Ansible, ici pour l’installation de Docker (tâche simple) :

```
- name: Install Docker CE packages
  ansible.builtin.dnf:
    name:
      - docker-ce
      - docker-ce-cli
      - containerd.io
      - docker-buildx-plugin
      - docker-compose-plugin
    state: present
```

UI de Portainer (interface de gestion de Docker) **:**

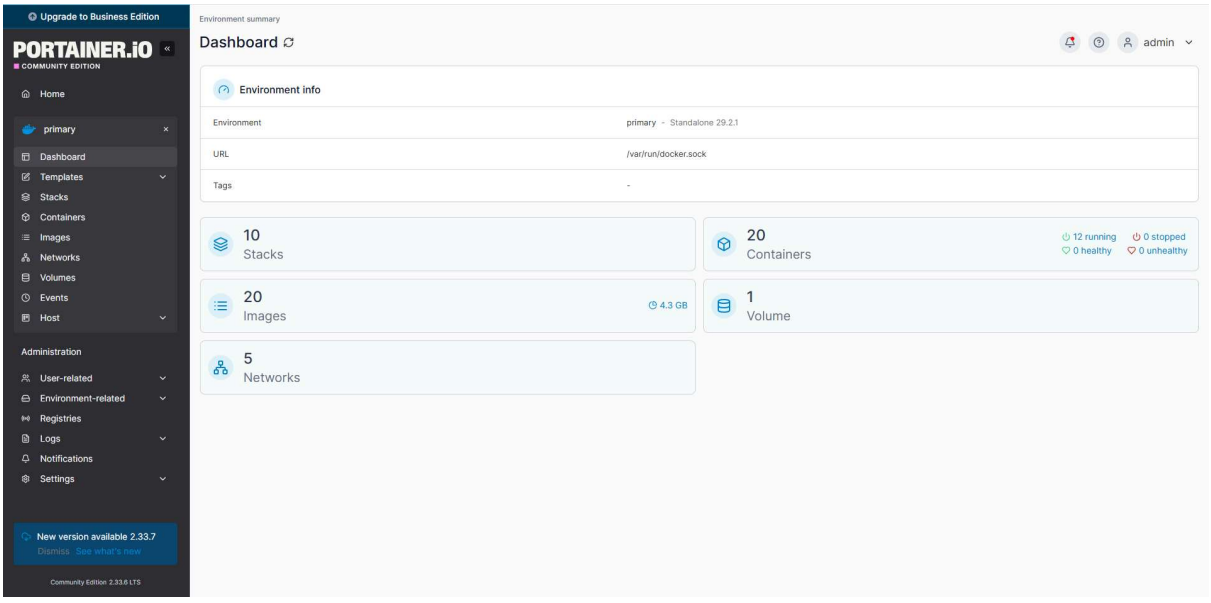
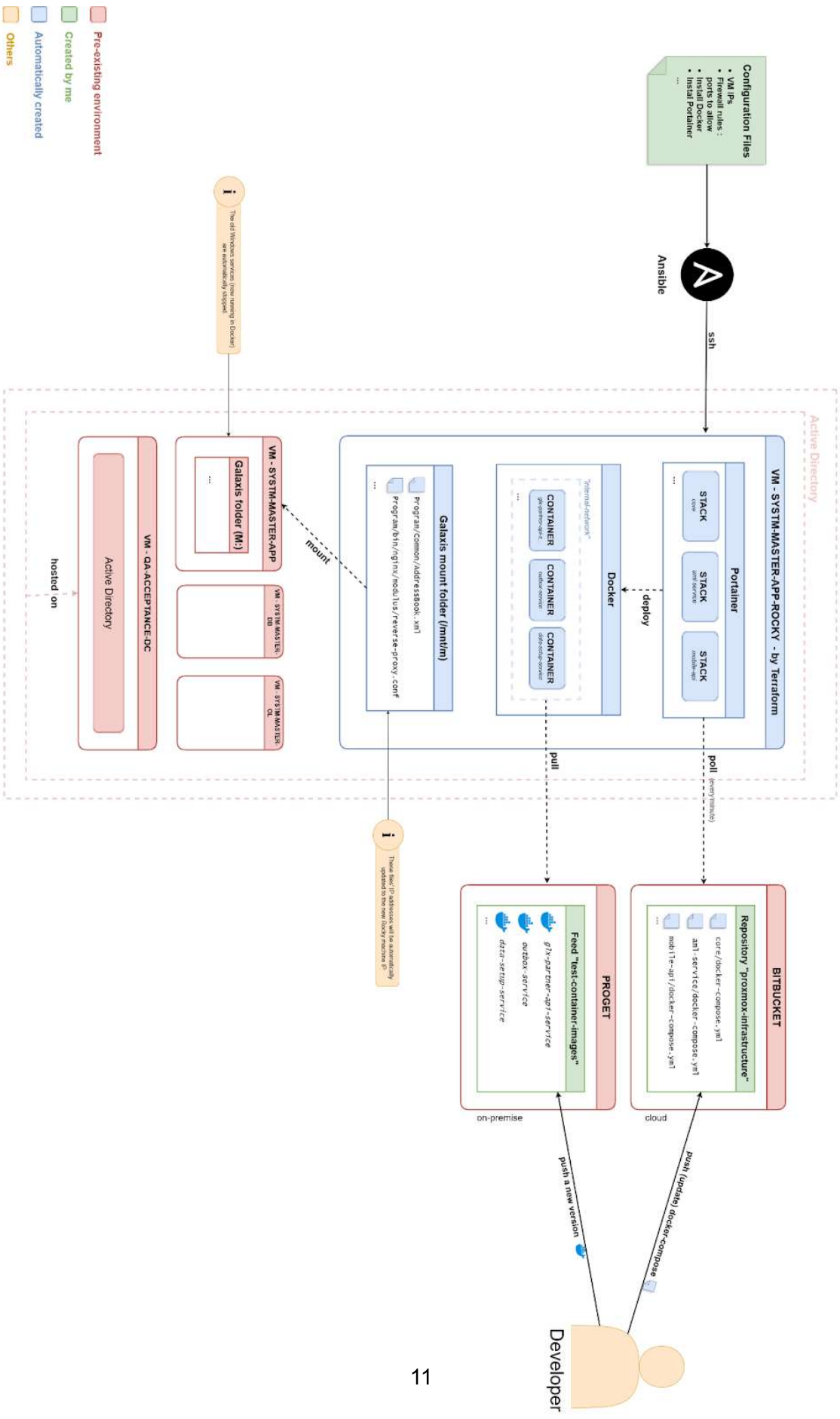


Diagramme récapitulatif de ce que fait Ansible :

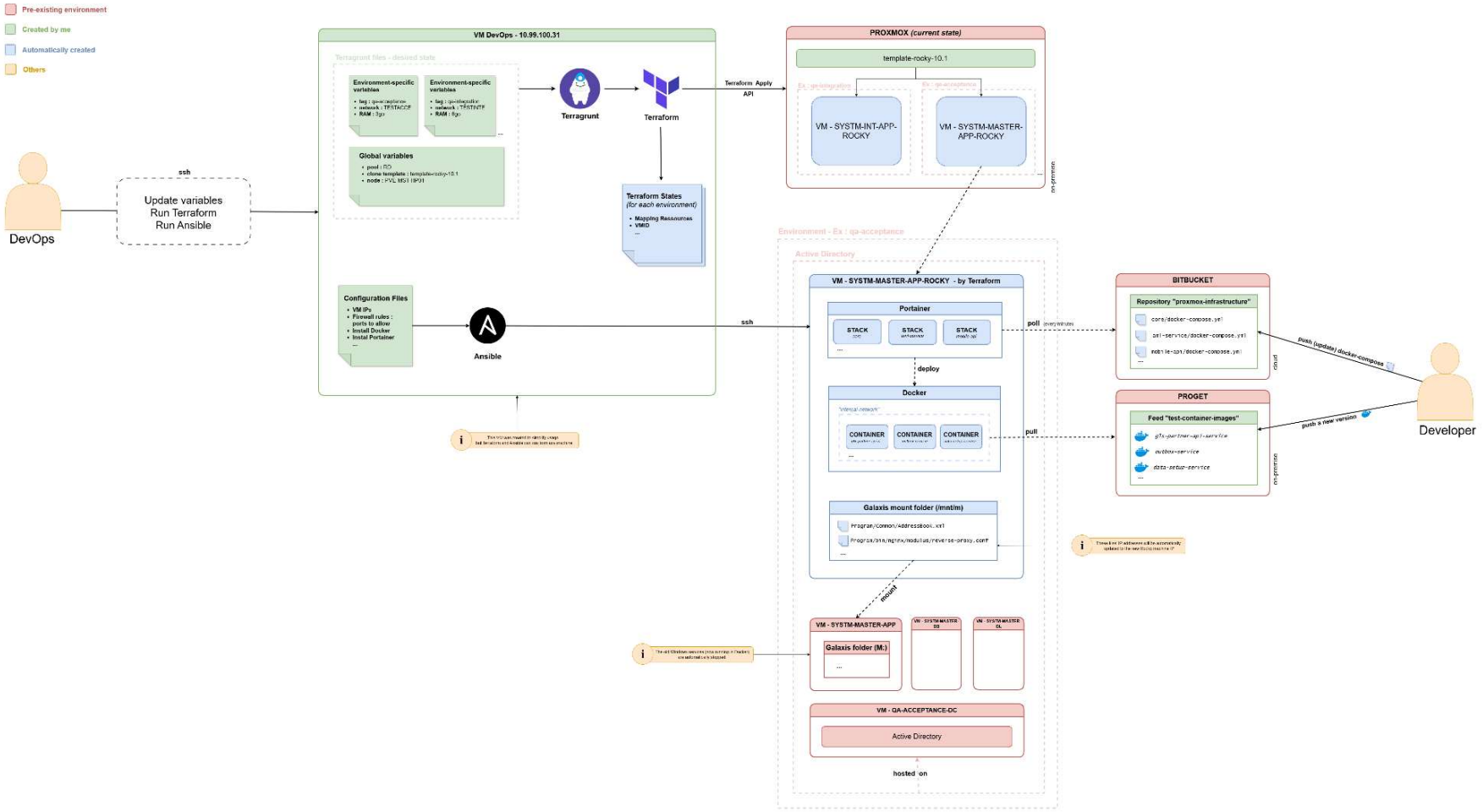


3) VM DevOps

Le code de tout ce projet est hébergé sur Bitbucket, donc récupérable pour utilisation, mais il peut tout de même nécessiter une certaine configuration de la machine hôte (celle qui exécutera le code), à savoir : installer Ansible, Terraform, Terragrunt et d'autres packages indispensables.
De plus, l'exécution demande pour la machine hôte un accès à l'IP du Proxmox pour Terraform (configuration firewall) et, pour Ansible, un accès au même réseau que la VM sur laquelle il doit se connecter.

Pour résoudre toutes ces contraintes (au moins dans l'environnement de dev), une VM dite "DevOps" a été créée, déjà configurée, et étant elle-même sur le Proxmox, il a été très simple de configurer les règles de firewall pour Terraform et d'accès aux autres réseaux pour Ansible.
Cette VM est matérialisée dans le diagramme suivant, représentant donc l'état final :

DevOps — Architecture diagram for Infrastructure as Code (IaC) and Continuous Delivery (CD) — Current state



Bilan du projet et conclusion personnelle

À la fin du stage, les tâches prévues ont toutes été réalisées, et même plus, puisque j'ai pu peaufiner des détails, simplifier l'utilisation, réaliser un README complet, une documentation, un diagramme d'architecture et d'autres éléments.

Ce stage a été bénéfique en tout point.
D'abord, il m'a permis de découvrir de nouvelles technologies comme Terraform, Ansible et Docker, et bien plus globalement des principes de DevOps et d'automatisation.

Ensuite, j'ai consolidé mes bases en Linux : utilisation quotidienne de la CLI, gestion des connexions SSH, des clés publiques/privées, etc. Le fait de manipuler ces outils tous les jours m'a fait gagner en aisance et en automatisme.

Il m'a aussi permis de renforcer ma compréhension de la virtualisation, un concept clé en informatique, ainsi que mes compétences en réseau, indispensables pour déployer et faire fonctionner des environnements virtualisés de manière propre et fiable.

Enfin, j'ai pu gagner en aisance dans le monde de l'entreprise : participation à des réunions, capacité à solliciter les bonnes personnes selon leurs compétences, échanges par e-mail avec des sociétés tierces (prestataires IT), rédaction de bilans, ainsi que création de diagrammes et de documents de synthèse.