

RAPPORT D'ACTIVITE PROFESSIONNELLE

Licence informatique générale (L3)

APPRENTI :

NOM : BARRILLET

PRENOM : Lyssandre

ENCADRANTS :

- **RESPONSABLE ENTREPRISE : M. MARTIN BASTIEN**
- **RESPONSABLE FORMATION : M. MAZZONI PHILIPPE**

ENTREPRISE D'ACCUEIL :



Rapport réalisé dans le cadre d'un contrat d'apprentissage
en licence informatique générale troisième année dispensée par le CNAM.

IDENTITÉ

M, Mme, Mlle : BARRILLET
Prénoms : Lyssandre
Date et lieu de naissance : 15/05/1997 Saint Germain en Laye
Nationalité : Française
Adresse : 38 Cours Jean Jaurès
Code postal : 38000
Ville : GRENONLE
Téléphone personnel : 06 58 40 34 71
Adresse e-mail :
N° de la carte d'auditeur du CNAM : ARA572637

VOTRE FORMATION

1. VOTRE DIPLOME OU TITRE OBTENU, LE PLUS ÉLEVÉ

BTS Service informatique aux organisations Systèmes et Réseaux

Date de l'obtention : 2022

Diplôme : OUI

Titre homologué : OUI

- D'établissement public

Niveau du diplôme : II

(Reportez-vous au glossaire page 10 et entourez le chiffre ci-dessus du bon niveau)

GLOSSAIRE DES NIVEAUX

- *niveau I : 3ème cycle universitaire, diplôme d'ingénieur, d'école supérieure de commerce...*
- *niveau II : 2^{ème} cycle d'études supérieures = bac + 3 ou 4*
- *niveau III : 1^{er} cycle d'études supérieures = bac + 2*
- *niveau IV : bac, brevet professionnel...*
- *niveau V : CAP, BEP...*

VOTRE EXPÉRIENCE PROFESSIONNELLE





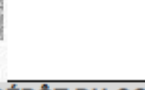
- Durée totale de votre expérience professionnelle : 4 ans
- Durée de votre expérience en informatique : 3 ans
- Succession des postes occupés, du plus ancien au plus récent. :

Précisez le nom de l'entreprise, le poste occupé (celui inscrit sur la fiche de paye, niveau de qualification), la fonction tenue, la durée de l'activité.

Emploi occupé	Entreprise et secteur d'activité	Durée de l'activité	Fonctions occupées
Technicien son	Audiovisuel - TitraFilm - Rejoyce - StudioCaptiale	1 an	<ul style="list-style-type: none">- Maintenance- Post-production
Technicien Systèmes et Réseaux - alternance	TitraFilm - Post-production de film	2 ans	<ul style="list-style-type: none">- Support utilisateur- Intervention incidents système et réseaux- Administration systèmes
DevOps - alternance	Ivès	1 an	<ul style="list-style-type: none">- Mise en place de scénario d'automatisation

Mode contractuel de l'apprentissage 111

L'EMPLOYEUR		<input checked="" type="checkbox"/> [X] employeur privé	<input type="checkbox"/> [] employeur « public »*
Nom et prénom ou dénomination : INTERACTIVITE VIDEO ET SYSTEME		N°SIRET de l'établissement d'exécution du contrat : 49137099500061	
Adresse de l'établissement d'exécution du contrat : 1080 CHEMIN DE LA CROIX VERTE		Type d'employeur : 12	
Complément :		Employeur spécifique : 0	
Code postal : 38330		Code activité de l'entreprise (NAF) : 6201Z	
Commune : MONTBONNOT ST MARTIN		Effectif total salariés de l'entreprise : 30	
Téléphone : 04-86-68-89-46		Convention collective applicable : Convention collective nationale applicable au personnel des bureaux d'études techniques, des cabinets d'ingénieurs-conseils et des sociétés de conseils(BET, SYNTEC)	
Courriel : rh@ives.fr		Code IDCC de la convention 1486	
*Pour les employeurs du secteur public, adhésion de l'apprenti au régime spécifique d'assurance chômage : <input type="checkbox"/>			
L'APPRENTI(E)			
Nom de naissance de l'apprenti(e) : BARRILLET			
Prénom de l'apprenti(e) : Lyssandre			
NIR de l'apprenti(e)* :1970578551345 <i>*Pour les employeurs de secteur privé dans le cadre L.6353-10 du code du travail</i>		Date de naissance : 15/05/1997	
Adresse de l'apprenti(e) : 1 RUE DES 21 APPELÉS		Sexe : <input type="checkbox"/> M <input checked="" type="checkbox"/> F	
Complément :		Département de naissance : 78	
Code postal : 93300		Commune de naissance : SAINT GERMAIN EN LAYE	
Commune : AUBERVILLIERS		Nationalité : 1 Régime social : 2	
Téléphone : 06-58-40-34-71		Déclare être inscrit sur la liste des sportifs, entraîneurs, arbitres et juges sportifs de haut niveau : <input type="checkbox"/> oui <input checked="" type="checkbox"/> non	
Courriel : lyssandre.barrillet@edu.itescia.fr		Déclare bénéficier de la reconnaissance travailleur handicapé : <input type="checkbox"/> oui <input checked="" type="checkbox"/> non	
Représentant légal (renseigner si l'apprenti est mineur non émancipé)		Situation avant ce contrat : 4	
Nom de naissance et prénom :		Dernier diplôme ou titre préparé : 54	
Adresse du représentant légal :		Dernière classe / année suivie : 01	
Complément :		Intitulé précis du dernier diplôme ou titre préparé : BTS SERVICES INFORMATIQUES AUX ORGANIS	
Code postal :		Diplôme ou titre le plus élevé obtenu : 54	
Commune :			
LE MAÎTRE D'APPRENTISSAGE			
Maître d'apprentissage n°1		Maître d'apprentissage n°2 :	
Nom de naissance : MARTIN		Nom de naissance :	
Prénom : Bastien		Prénom :	
Date de naissance : 09/06/1990		Date de naissance :	
[X] L'employeur atteste sur l'honneur que le maître d'apprentissage répond à l'ensemble des critères d'éligibilité à cette fonction.			

LE CONTRAT		
Type de contrat ou d'avenant : 22		Type de dérogation : 60 à renseigner si une dérogation existe pour ce contrat
Numéro du contrat précédent ou du contrat sur lequel porte l'avenant : 092202010041113		
Date de conclusion : <small>(Date de signature du présent contrat)</small> 06/09/2022	Date de début d'exécution du contrat : 12/09/2022	Si avenant, date d'effet :
Date de fin du contrat ou de la période d'apprentissage : 12/07/2023		Durée hebdomadaire du travail : 35 heures 00 minutes
Travail sur machines dangereuses ou exposition à des risques particuliers : <input type="checkbox"/> oui <input checked="" type="checkbox"/> non		
Rémunération <small>* Indiquer SMIC ou SMC (salaire minimum conventionnel)</small> 1 ^{re} année, du 12/09/2022 au 31/05/2023 : 80 % du SMC * ; du 01/06/2023 au 12/07/2023 : 100 % du SMC * 2 ^e année, du ____/____/____ au ____/____/____ : ____ % du ____ * ; du ____/____/____ au ____/____/____ : ____ % du ____ * 3 ^e année, du ____/____/____ au ____/____/____ : ____ % du ____ * ; du ____/____/____ au ____/____/____ : ____ % du ____ * 4 ^e année, du ____/____/____ au ____/____/____ : ____ % du ____ * ; du ____/____/____ au ____/____/____ : ____ % du ____ *		
Salaire brut mensuel à l'embauche : 1280 €		Caisse de retraite complémentaire : KLESIA
Avantages en nature, le cas échéant : Nourriture : ____ € / repas Logement : ____ € / mois Autre : <input type="checkbox"/>		
LA FORMATION		
CFA d'entreprise : <input type="checkbox"/> oui <input checked="" type="checkbox"/> non		Diplôme ou titre visé par l'apprenti : 63
Dénomination du CFA responsable : OGE C CHARMILLES CFA		Intitulé précis : LICENCE STS MENTION INFORMATIQUE PARCOURS INFORMATIQUE GENERALE
N° UAI du CFA : 0383474V		Code du diplôme : 20532618
N° SIRET du CFA : 78837731500031		Code RNCP : 24514
Adresse du CFA responsable : 8 RUE DU TOUR DE L'EAU		Organisation de la formation en CFA : Date de début du cycle de formation : 12/09/2022
Complément : Code postal : 38400		Date prévue de fin des épreuves ou examens : 04/07/2023
Commune : SAINT MARTIN D'HERES		Durée de la formation : 450 heures
Visa du CFA (cachet et signature du directeur)		
<div style="display: flex; justify-content: space-around; align-items: center;">   </div>		
[X] L'employeur atteste disposer de l'ensemble des pièces justificatives nécessaires au dépôt du contrat Fait à MONTBONNOT ST MARTIN		
Signature de l'employeur	Signature de l'apprenti(e)	Signature du représentant légal de l'apprenti(e) mineur(e)
		
CADRE RÉSERVÉ À L'ORGANISME EN CHARGE DU DÉPÔT DU CONTRAT		
Nom de l'organisme : *OPCO ATLAS		N° SIRET de l'organisme : 35321361400137
Date de réception du dossier complet :		Date de la décision :
N° de dépôt :		Numéro d'avenant :
<small>Pour remplir le contrat et pour plus d'informations sur le traitement des données reportez-vous à la notice FA 14</small>		

REMERCIEMENTS

Je tiens avant toute chose à remercier les équipes d'IVES pour leur accueil et mon intégration du début de mon alternance jusqu'à aujourd'hui.

Je souhaiterais plus particulièrement remercier M. Martin Bastien de m'avoir recruté en tant qu'apprenti au sein de son service. Plus particulièrement pour la qualité de sa pédagogie et de sa disponibilité. L'encadrement fut de qualité ayant permis une réelle montée en compétences.

Je voudrais également remercier tout le corps enseignant des charmilles pour la qualité de leur cours. Plus particulièrement M. Mazzoni Philippe, responsable de la formation pour sa disponibilité.

SOMMAIRE

Glossaire	10
Introduction	13
Présentation de l'entreprise	
Ivès	14
Mon rôle	14
Mission 1 : Mise en place d'Ansible	
Attente d'Ansible et historique	17
L'infrastructure	17
Mon environnement de travail	19
Installation et fonctionnement d'Ansible	19
Fonctionnement de nos certificats openSSL	21
Playbook 'list and register'	22
Résultat du playbook 'list and register'	23
Mission 2 : Création de conteneur grâce à Ansible	
Attente	24
Création de conteneur simplifié	25
Script SHELL/BASH	25
Configuration automatique du fichier /etc/ansible/host	29
Notification	32
Bilan du projet	
Les points forts	34
Axe d'amélioration	34
Conclusion	35
Bibliographie	36
Sitographie	36
Annexes	37

Glossaire :

Almalinux : système d'exploitation open source basé sur la distribution Red Hat Entreprise.

API : interface de programmation d'une application qui permet à d'autres applications de communiquer entre elles.

Argument : Valeur envoyée à un programme.

Centos6-7 : système d'exploitation Linux open source.

CIDR : acronyme pour Classless Inter-Domain Routing, méthode permettant d'allouer une adresse IP.

Clé asymétrique : type de cryptographie utilisant une paire de clés publique et privée afin de chiffrer et déchiffrer des données.

Clé ED25519 : algorithme de cryptographie.

Clé privée : Clé devant être gardée secrète permettant le déchiffrement des données à l'aide de la clé publique.

Clé publique : Clé devant être partagée au destinataire permettant le déchiffrement des données.

Clé RSA : algorithme de cryptographie.

Cloud : modèle permettant d'accéder à des ressources informatiques via internet.

Concatener : fusion d'éléments informatiques en un seul élément.

Conteneur : technologie d'isolation de processus permettant de faire fonctionner plusieurs applications sur un même système sans qu'elles n'interfèrent entre elles.

Dépôt : espace de stockage pour les fichiers ou les paquets de logiciel

Dictionnaire : Structure de données en développement permettant d'y stocker des valeurs.

Fichier CSV: fichier de données

Hyperviseur type 1 : hyperviseur s'exécutant directement sur la machine, les ressources de la machine lui sont exclusivement dédiées

Image : copie de l'état d'un système sauvegardé sur un support

IP : identifiant unique attribué à une machine sur le réseau

Jira : outil de gestion de projet

JSON : format de données

Lab : environnement de test isolé

LXC : technologie de virtualisation

Noeuds : ensemble de machines connectées à un réseau.

Online.net : entreprise d'hébergement de système dans le cloud

OpenSSL : bibliothèque de cryptographie

OPENVZ : technologie de virtualisation

Oracle VirtualBox : logiciel de virtualisation open source

Paquet : ensemble de fichiers regroupés afin de faciliter leur installation et désinstallation

Playbook : fichier contenant des étapes nécessaires à automatiser une tâche

Programmation web : programmation d'applications exécutées par les navigateurs web

Python : langage de programmation interprété

Serveur Apache : serveur web open source

Services : application ou processus exécutés en arrière-plan

Shell Bash : l'interpréteur de commande utilisé par Linux

SSH : protocole de communication sécurisée pour accéder à des systèmes à distance

Système d'information : ensemble de tous les éléments interconnectés d'une organisation afin de permettre la disponibilité, la confidentialité et l'intégrité de ces données.

SysAdmin : personne responsable de l'infrastructure informatique.

YAML : format de fichier utilisé pour configurer des applications ou services.

Introduction

Suite à l'obtention du BTS Service informatique des organisations, orienté système et réseaux, il fut logique pour moi de me diriger vers la licence informatique générale afin de diversifier mes compétences et obtenir un socle de compétences plus solide. J'ai choisi la voie de l'alternance car ayant réalisé les deux précédentes années en alternance, c'est pour moi la meilleure façon d'allier théorie et pratique et d'obtenir une expérience professionnelle concrète. Ces deux précédentes années, je les ai réalisées en tant que technicien système et réseaux. Cette expérience me permettra d'approfondir mes compétences en système et réseau, mais surtout d'en acquérir de nouvelles, notamment en développement logiciel.

C'est l'entreprise IVES, basée à Montbonnot Saint Martin que j'ai choisi pour réaliser cette alternance. Il s'agit d'un éditeur de logiciel. Ce qui m'a attiré en premier lieu chez Ives c'est qu'il s'agit d'une entreprise mettant la technologie au service de l'Homme, favorisant un usage pertinent de celle-ci. Pleinement implantée dans le secteur du numérique, son activité consiste à développer une plateforme permettant de mettre en relation une personne sourde avec une personne entendante. Cette mise en relation est supervisée par un interprète permettant la traduction de la conversation, en direct et en langue des signes. Cette plateforme est accessible sous système : MacOS, Windows, ainsi qu'en accès web.

Afin d'assurer la mise en production de ces services, ainsi que l'infrastructure interne de l'entreprise, nous disposons d'un parc d'environ 200 **conteneurs**. Un conteneur est un moyen d'isoler des éléments nécessaires pour faire fonctionner une application.

Et à l'heure où la conteneurisation offre de nombreux avantages : efficacité, fiabilité, disponibilité. Il a fallu trouver un outil permettant la configuration et la gestion centralisée de ces **conteneurs**. Notre choix s'est porté sur l'outil Ansible, il s'agit d'un outil open source d'automatisation. Il se connecte au système que l'on souhaite administrer et effectue des instructions que l'on aura prédéfinies afin de ne plus avoir à les effectuer manuellement.

La mise en production de cet outil m'a alors été confiée, ainsi que la création des premiers scénarios souhaitée afin d'automatiser les premières tâches.

Je commencerais par présenter l'entreprise IVES ainsi que son organisation, s'en suivront la présentation de l'état initial du contexte dans lequel s'inscrit la problématique. J'aborderai ensuite les différentes phases techniques de la mise en place de l'outil Ansible ainsi que la création des premiers scripts permettant de mettre en place des scénarios d'automatisation.

Présentation de l'entreprise

1) Ivès

Fondée en 2005, Ivès est une PME située près de Grenoble à Montbonnot Saint Martin. Elle met en place des solutions permettant aux personnes sourdes et malentendantes de communiquer librement et de façon accessible. Nous développons des solutions permettant à des milliers de sourds et malentendants une meilleure autonomie. Ces solutions se présentent sous la forme de plateformes ou logiciels. Ils permettent la mise en relation tripartite entre la personne présentant l'un de ces handicaps, son interlocuteur et un interprète supervisant l'échange, et traduit celui-ci en langue des signes.

L'entreprise est née de l'association par les co-gérants Pascal Dupuy, Didier Chabanol et Emmanuel Buu et du constat suivant : en France, une personne sourde ou malentendante est confrontée à de multiples obstacles lorsqu'il s'agit de communiquer au quotidien. Dès lors, une première version de produit est apparue. Aujourd'hui, le produit logiciel d'Ivès est pensé pour différentes utilisations. La principale est l'accueil téléphonique, mais il est également possible de superviser l'accueil physique via notre plateforme. Nos clients principaux sont donc les entreprises, qui, pour mettre en place une politique d'inclusion, font appel à nos services. De plus, depuis le 7 octobre 2018, une loi oblige les entreprises dont le chiffre d'affaires est de plus de 250 millions d'euros une accessibilité téléphonique pour les personnes sourdes et malentendantes.

Depuis, avec la mise en place de cette nouvelle loi, ainsi que la prise en considération des problématiques d'inclusion plus importantes, l'entreprise connaît une croissance lui ayant permis de doubler ses effectifs depuis 2020 ainsi que d'investir dans la recherche, afin de proposer des produits plus performants. C'est notamment le cas en se positionnant sur le marché de l'intelligence artificielle en proposant un chatBox à destination des sourds et malentendants, mais aussi en innovant avec le projet d'un avatar 3D capable de traduire l'audio en langue des signes, en temps réel. Cette croissance a notamment permis à la société le rachat de la société Eliaz, spécialisée dans la langue des signes ainsi que d'ouvrir une filiale au Canada.

Aujourd'hui, l'entreprise Ivès France compte 33 collaborateurs et centralise le pôle technique et informatique. Celui-ci a été divisé en différents services, nous retrouvons en premier lieu les développeurs dont l'objectif est de concevoir les fonctionnalités des applications. La qualité (QA) testant et assurant la mise en production de celles-ci. L'administration système permettant d'assurer la continuité des services ainsi que le maintien de l'infrastructure informatique. La R&D (recherche et

développement) afin d'assurer les besoins futurs de la société, ainsi que le service DevOps dont je fais partie.

Le DevOps est une approche conciliant développement et **opération**. L'objectif principal est de fluidifier et d'automatiser les échanges entre ces deux services. Les développeurs conçoivent les fonctionnalités des applications, celles-ci doivent être intégrées à l'infrastructure afin d'être testées, puis une fois validées mise en production. Il existe donc une multitude de tâches afin d'intégrer le code à une application ou infrastructure pour le rendre exploitable. L'infrastructure désigne l'ensemble des systèmes permettant d'héberger les outils nécessaires à l'élaboration de nos applications ainsi qu'à leur mise en production. Une des composantes clé du DevOps consiste à automatiser les processus de développement, de test ainsi que son déploiement. Ceci permet une mise en production plus fiable et plus performante des outils. Chez Ivès, le service DevOps a été créé il y a deux ans et est né du besoin de fiabiliser les livraisons de la part des développeurs vers l'équipe QA. Aujourd'hui, nous sommes deux au service DevOps, Bastien mon tuteur, et moi-même.

2) Mon rôle

Ayant été issue d'une formation orientée système et réseaux, il m'a fallu comprendre rapidement les bases du développement. Pour cela j'ai été encouragé à travailler sur divers projets pensés et encadrés par mon tuteur afin de me familiariser avec les outils permettant une approche DevOps. Je travaille donc en mode projet sur ces premières missions, nous utilisons l'outil Jira afin d'effectuer et de collaborer sur le suivi des projets. Mon Tuteur Bastien m'assigne le projet en prenant soin de décomposer les différentes étapes clés ainsi que les attendus, puis me l'assigne sous forme de ticket dans **Jira**. Ce ticket que je complète au fil du temps, me sert de tram quant à mon avancée sur le projet. Nous faisons par la suite des réunions quotidiennes afin de suivre l'avancement des projets qui me sont assignés ainsi que des besoins futurs de l'organisation. La culture DevOps, regroupant une grande variété d'outils et de compétences, mes missions l'ont été tout autant : diverses et variées. Cela m'a permis de me familiariser avec l'infrastructure informatique de l'organisation tout en progressant sur des langages de programmation, tels que la **programmation web**, **python** et le scripting sur **Shell bash**.

Je travaille dès lors sur la création d'outils permettant d'automatiser des tâches qui demandent à ce jour des actions manuelles, et donc longues, n'apportant pas ou peu de valeur. La première mission fut tout de suite très intéressante, j'ai pu commencer par créer un **serveur Apache** s'actualisant automatiquement. Celui-ci permet de lister toutes les fonctionnalités ainsi que leur

version et serveur associé de notre infrastructure. Puis d'autres scripts permettant d'effectuer des inventaires sur les configurations mails, ou d'effectuer des rapports sur les **paquets** installés sur nos **conteneurs**. Ces outils sont actuellement fonctionnels et utilisés par les équipes en interne.

Une des principales missions qui m'a été confiée, et qui fera l'objet de la suite de ce rapport, est la mise en place de l'outil Ansible. Cet outil est conçu pour simplifier et automatiser la gestion d'une infrastructure informatique. Il s'agit d'un outil open source offrant une grande flexibilité sur l'automatisation, le déploiement ainsi que la gestion de systèmes.

Mission 1 : Mise en place d'Ansible

1) Attente d'Ansible et historique

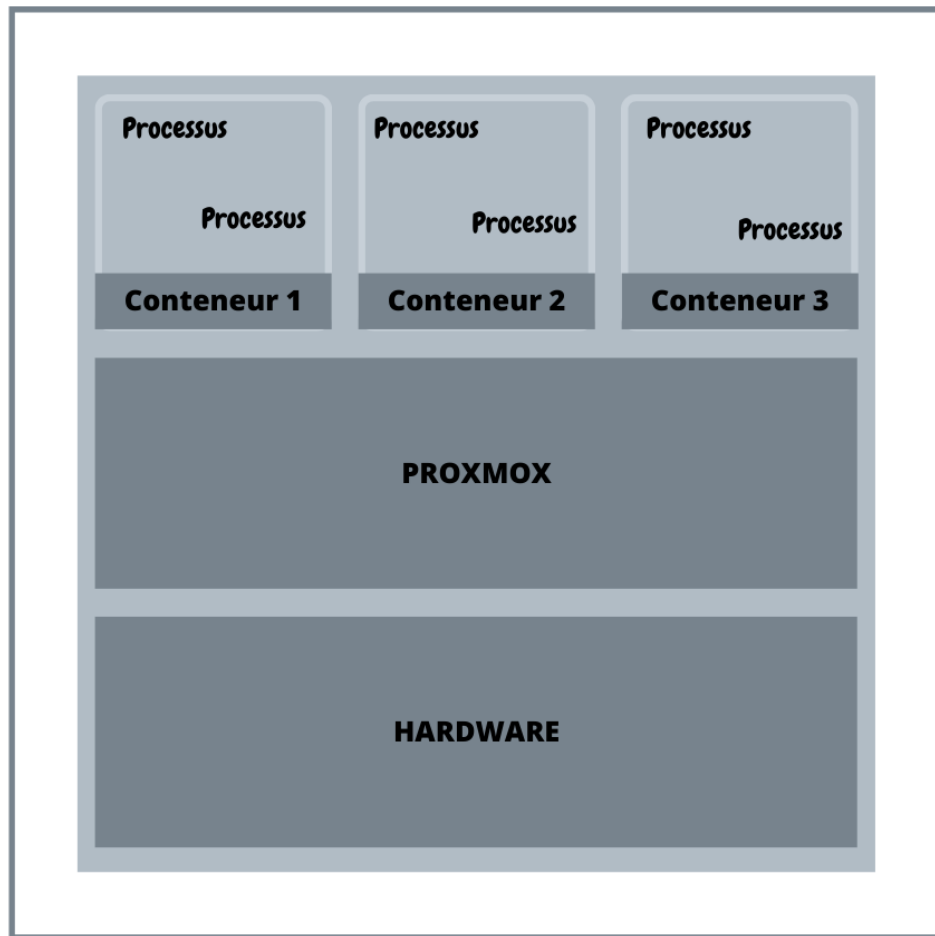
L'objectif de la mise en place de l'outil Ansible est de permettre une gestion centralisée de tous nos conteneurs. Un exemple concret est lorsque nous souhaitons installer un **paquet** sur un **conteneur**, nous pouvons le déployer sur un ensemble et non répéter la manipulation pour chacun. Aujourd'hui, chaque conteneur étant géré individuellement, cela peut nécessiter un investissement en temps et en ressources. Travaillant en mode projet, je pourrais consacrer la majeure partie de mon activité à la mise en place de cet outil, qui de plus, désigne un besoin réel de l'entreprise. J'apprécie la confiance qui m'est accordée quant à la réalisation de ce projet, nécessitant les accès à l'ensemble du **système d'information** d'Ivès.

Ce projet va se décomposer en plusieurs parties car il est nécessaire d'établir un certain nombre de prérequis afin d'exploiter Ansible, et ne pas faire de mauvaise manipulation. Cet outil sera utilisé pour administrer nos outils en interne, mais également en production, son utilisation doit être millimétrée.

2) L'infrastructure

Ivès est une entreprise évoluant sur une infrastructure basée sur proxmox, il s'agit d'un **hyperviseur type 1**, open source, pouvant héberger un certain nombre de machines virtuelles et conteneurs de manière centralisée grâce à une interface web. Nos hyperviseurs sont hébergés en interne ainsi que dans le **cloud** chez **OVH** et **online.net**. Nos conteneurs servent à héberger l'ensemble des **services** permettant le bon fonctionnement des applications Ivès, à ce jour, nous en disposons d'environ 200, répartis sur une dizaine d'hyperviseurs. Pour mieux comprendre le concept d'hyperviseur, voici un schéma récapitulatif :

HYPERVISEUR TYPE 1 - PROXMOX



Proxmox est le système d'exploitation permettant de créer des environnements isolés : les conteneurs. Chacun dispose de son propre système de fichiers, et de son propre environnement d'exécution, mais il partage le même noyau du système d'exploitation. Ces conteneurs permettent d'isoler les processus nécessaires aux fonctionnements de nos applications.

Pour exploiter l'outil Ansible, je dois m'assurer d'avoir une vision et une compréhension claire de notre infrastructure système. Il est à noter que chaque conteneur dispose de sa propre image. Chez Ivès, nous utilisons des images exclusivement sous distribution **Linux**: Centos6, Centos7 ainsi que Almalinux. Ce choix, historique, a été d'utiliser ces distributions car elles sont stables, open source, et gérées par la communauté. Actuellement, une longue migration est prévue de nos distributions Centos vers Almalinux, car les **dépôts** Centos ne sont plus maintenus.

Nous utilisons deux technologies de conteneurisation : **LXC** et **OPENVZ**. Il m'est important de comprendre les spécificités de ces technologies de conteneurisation ainsi que des distributions Linux utilisées car mes futurs scripts devront s'y adapter, signifiant que beaucoup de conditions seront effectuées selon ces paramètres.

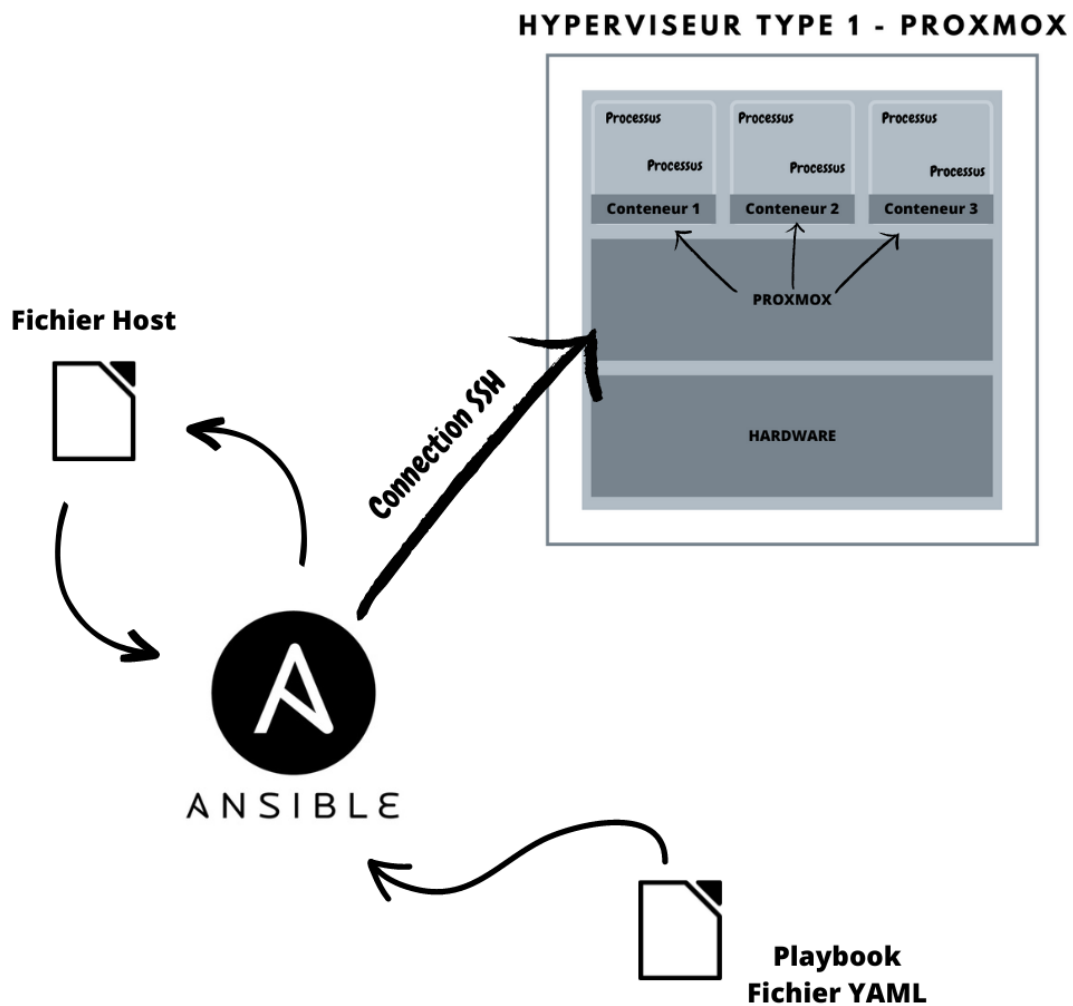
3) Mon environnement de travail

Je dispose de mon propre serveur de développement afin de réaliser les projets qui me sont assignés. Pour celui-ci, je mets donc en place mon environnement '**lab**' qui permettra de concevoir mes premiers tests, ainsi que de mettre en place les premières versions des procédures qui seront utiles par la suite, afin de migrer le projet cette fois-ci en production. Réaliser de la documentation quotidienne et la mettre à jour fait partie des bonnes pratiques d'Ivès. Cela est particulièrement important dans le secteur de l'informatique, et permet un transfert de compétences ainsi qu'une meilleure autonomie de la part de tous les collaborateurs. Dans mon cas, étant en alternance c'est un réel avantage quant à ma montée en compétences de disposer de documentations mises à jour.

4) Installation et fonctionnement d'Ansible

J'installe les **dépôts** nécessaires à l'installation d'Ansible puis, je peux commencer à prendre en main l'outil. Son fonctionnement est basé sur l'écriture de **playbooks**. Ce sont des fichiers **YAML** qui contiendront les actions à suivre sur l'ensemble des **nœuds** que nous lui aurons spécifié. Ces **nœuds**, nous les spécifions dans le fichier de configuration host d'Ansible. Ils permettront d'indiquer quels conteneurs seront éligibles aux actions prévues par le playbook.

FONCTIONNEMENT ANSIBLE



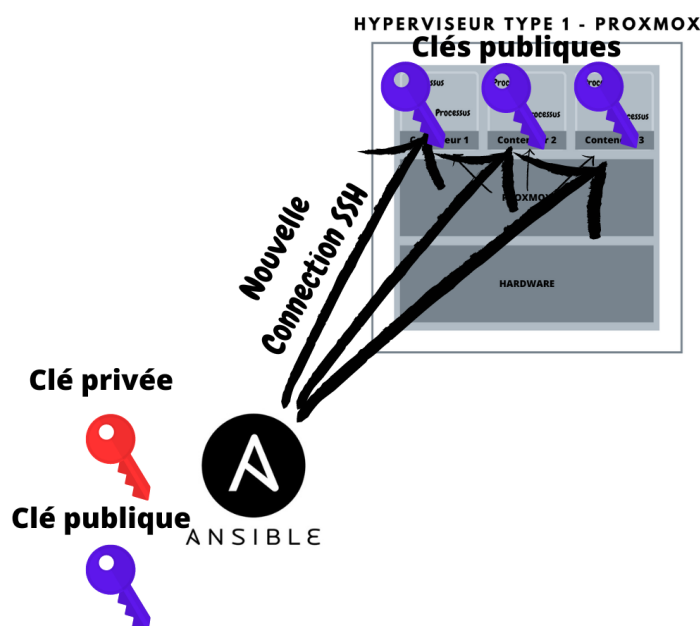
La seule contrainte d'Ansible, afin qu'il se connecte aux serveurs cibles est que celui-ci doit être accessible via **SSH**. Ce qu'il n'est actuellement le cas d'aucun de nos conteneurs. L'idée est alors d'atteindre nos conteneurs via proxmox l'hyperviseur, qui dans notre cas est accessible en SSH, puis d'accéder à ses conteneurs. Depuis proxmox, il est possible, grâce à certaines commandes, d'écrire dans ses conteneurs les clés nécessaires à établir la connexion.

5) Fonctionnement de nos certificats OpenSSL

OpenSSL permet de générer des **clés publiques et privées**, requises pour l'authentification client-serveur utilisant le protocole SSH. Cette partie du projet va me permet donc de mieux saisir l'importance d'établir une politique de sécurité, ainsi que des bonnes pratiques quant à son exécution. Je peux ainsi mettre en pratique un cas concret en rapport avec le module Cybersécurité de la licence. Ce premier **playbook** (annexe 1) que je réalise, sera donc exécuté sur tous nos hyperviseurs **proxmox**, et permettra d'écrire dans tous ses **conteneurs** une clé préalablement générée. Ces clés sont générées grâce à une commande linux "ssh-keyscann" sur le serveur hôte. Il est indispensable, pour assurer la confidentialité et l'intégrité de nos données, que la clé privée reste secrète. Notre serveur hôte va ainsi disposer des deux clés, aussi appelées **clés asymétriques**. L'objectif est de déployer la clé publique sur nos 200 conteneurs en une fois. Ceci permettant d'établir le chiffrement de la connexion entre les conteneurs et Ansible, et ainsi ne plus utiliser l'authentification par mot de passe. Ceci aurait demandé une saisie manuelle lors de l'exécution des prochains playbooks, et est moins sécurisé qu'un chiffrement par **clé asymétrique**.

La décision a été de générer deux types de clé : une **clé RSA** pour nos conteneurs sous Centos, ainsi qu'une **clé ED25519** pour ceux sous Almalinux. Ce choix est fait étant donné que la dernière version **openSSL** de Centos n'est pas éligible à **ED25519**. Ces clés sont obtenues grâce à un algorithme de cryptographie. Il est à noter que l'algorithme utilisé par les clés **ED25519** est plus récent et plus complexe, que celui utilisé par les clés **RSA**, qui dans un futur seront remplacées.

CLÉS ASYMÉTRIQUES



En comparaison avec le schéma précédent, le souhait est de communiquer via SSH directement avec le conteneur et non plus avec proxmox.

6) Premier playbook “list and register”

Après consultation de l'administrateur Système, car les **playbooks** vont pleinement toucher à l'infrastructure qu'il manage (besoin de sa permission), nous allons établir avec mon Tuteur Bastien les différentes étapes de ce playbook permettant d'initier la connexion **SSH** et qui me sera attribuée. Pour nos 200 **conteneurs**, suivant la technologie de conteneurisation utilisée (**LXC** ou **OPENVZ**), chacun doit avoir un nouvel utilisateur dédié, la bonne **clé publique** suivant sa version **d'openssl**. Cela signifie que je dois réaliser une phase de test afin de m'assurer que le conteneur ne dispose pas déjà d'une clé quelconque, ou de l'utilisateur. Ensuite un mail créé automatiquement, grâce à un autre playbook (annexe 2) avertira notre administrateur Système de toutes les modifications opérées.

Le **playbook** (annexe 1) va donc permettre ces étapes. Il est déployé sur l'ensemble de nos hyperviseurs, va lister puis boucler sur l'ensemble des conteneurs **LXC** et **OPENVZ** afin d'y effectuer les tâches suivantes :

- Sauvegarder les noms d'hôtes qui permettront d'organiser la notification mail
- Tester si l'utilisateur existe
- Créer l'utilisateur
- Créer le bon répertoire quant à l'hébergement de la clé publique
- Tester la version d'**OpenSSL**
- Vérifier si la clé n'existe pas déjà
- Copier la bonne clé suivant la version d'**openssl**.
- Enregistrer toutes les modifications dans un **fichier csv** et l'envoyer automatiquement par mail.

Par la suite, je mets en place un script simple bash (annexe 3) permettant de **concatener** et mettre en forme le fichier à destination de l'administrateur système ainsi que d'exécuter les deux **playbooks**, le premier servant à effectuer toutes les tâches (annexe 1), le deuxième est réservé à l'envoi du mail (annexe 2).

7) Résultat du playbook “list and register”

Désormais, plus aucune manipulation n'est à effectuer de notre côté, le processus est automatisé. Le script **bash** sera exécuté toutes les nuits. Tous nos **conteneurs** sont accessibles via **SSH**, grâce à leur adresse **IP** ou nom d'hôtes. Nous pouvons leur assigner un groupe écrit dans le fichier host de ansible. Ces groupes permettent de segmenter les conteneurs sur lesquels exécuter les actions définies par le playbook. Je peux désormais pleinement exploiter les outils offerts par ansible. Le deuxième playbook permettant l'envoi du mail notification, voici son contenu :

À : devops

Cc : SysAdmin

Ven 10/03/2023 10:14

[modifications.csv](#)

440 octets

Hello,

You'll find in the attached file, the list of all modifications done by Ansible.

Best Regards.

Avec pour exemple de contenu du fichier csv :

```
192.168.92.111;950;oss-950.internal; user added
192.168.92.111;950;oss-950.internal;RSA key added
192.168.92.111;951;oss-951.internal; user added
192.168.92.111;951;oss-951.internal;ED25519 key added
```

Mission 2 : Création de conteneurs grâce à Ansible

1) Attente du projet

L'une des possibilités d'action qu'offre Ansible, est de permettre la création et le déploiement automatisés de **conteneur** sur **Proxmox**. L'attente du prochain projet est d'exécuter ceci sans effectuer d'action manuelle. L'objectif est, en saisissant en tant qu'**argument** le nom du **proxmox** ainsi que l'**image** à déployer, y créer un nouveau **conteneur**. Ce **conteneur** nous souhaitons le personnaliser. Une liste m'est alors établie au début de projet (annexe 4), par Bastien, de toutes les spécifications souhaitées.

Ce nouveau script, dont une partie sera écrite en **bash**, en **yaml** ainsi qu'en **python**, va se décomposer en plusieurs étapes. Il s'agit d'un plus gros projet que le premier, dans le sens où il est plus complexe, et dispose de plus d'étapes à mettre en place. Il me faut donc bien saisir ces différentes étapes : un grand nombre de **playbooks** sera exécuté afin d'y segmenter les actions. Cette segmentation est nécessaire car il sera nécessaire de séparer les **playbooks** selon l'**image** du **conteneur** cible. La complexité étant que je dois adapter le comportement des **playbooks** suivant l'image choisie. En effet, Almalinux, Centos 6 ou Centos 7 sont assez différentes, les actions envisagées pour l'une ne seront pas les mêmes que pour l'autre. Ainsi, je dois adapter les actions des playbooks en fonction de ces images. En clair, toutes les actions présentées dans la liste (annexe 4) sont exécutables mais selon l'image, la méthode d'exécution peut différer.

Je travaillerais tout au long de ce projet, sur un proxmox **lab**, qui me permettra de réaliser tous les tests nécessaires avant une mise en production. Je virtualise donc celui-ci grâce à **Oracle Virtualbox** afin de pouvoir le manipuler sans risque pour la production. Une fois créé, je lui envoie les **images** nécessaires au déploiement des futurs **conteneurs**, les mêmes que nous utilisons déjà, soit **Centos6**, **Centos7** et **Almalinux**.

2) Création d'un conteneur simple

Dans un premier temps, je souhaite simplement créer un nouveau **conteneur** grâce à l'exécution d'un seul **playbook**. Ce **playbook**, (annexe 5) va se décomposer en 3 tâches : la création du conteneur, le lancement du **conteneur**, ainsi que l'installation **d'openssh** permettant d'exécuter le service **ssh**. Ce **playbook** fonctionne, mais, était écrit en "dur" les éléments nécessaires à sa configuration. En effet l'adresse **IP**, le mot de passe, le nom d'hôte ainsi que de nombreux paramètres doivent être soit unique, soit modulable (image, numéro du conteneur, proxmox). Actuellement, pour effectuer cette personnalisation je dois modifier le fichier **YAML** du **playbook**. La prochaine étape consistera à déterminer tous ces paramètres de façon optimisée. Pour ce faire, je crée un script **SHELL/BASH** qui va permettre d'exécuter les **playbooks** dans l'ordre indiqué, mais surtout d'envoyer tous les paramètres sous forme de variables, que l'on aura déterminées dans ce script.

3) Script SHELL/BASH

La première étape de ce script (annexe 6) va être d'extraire certaines données en accédant à l'**API** proxmox. Nous souhaitons connaître les conteneurs que celui-ci contient. Dans un premier temps, il est nécessaire d'établir l'authentification à cette API, le mot de passe ainsi que le nom du proxmox saisi en argument vont permettre d'initier la connexion à cette API. Je vais donc pouvoir extraire un certain nombre de données au format **JSON** et une fois triées vont permettre d'extraire une liste contenant les conteneurs déjà présents. Ces conteneurs sont définis par un numéro appelé CTID. Voici un exemple des données récupérées en accédant à une API proxmox, contenant deux conteneurs.

```
{
  "data": [
    {
      "name": "oss-950.internal",
      "maxswap": 20971520,
      "type": "lxc",
      "disk": 382840832,
      "maxmem": 536870912,
      "swap": 0,
      "mem": 8224768,
      "status": "running",
      "cpu": 0.000123397946211673,
      "cpus": "1",
      "pid": 219233,
      "maxdisk": 8350298112,
      "ctid": "950",
      "uptime": 3304,

```



```

        "netin":13769943,
        "diskwrite":319488,
        "diskread":34750464,
        "netout":6681
    },
    {
        "netin":23000003,
        "diskwrite":56737792,
        "ctid":"951",
        "maxdisk":8350298112,
        "uptime":453,
        "netout":243703,
        "diskread":106786816,
        "cpu":0,
        "pid":227799,
        "cpus":"1",
        "swap":0,
        "mem":20516864,
        "disk":753938432,
        "type":"lxc",
        "maxswap":20971520,
        "maxmem":536870912,
        "status":"running",
        "name":"oss-951.internal"
    }
  ]
}

```

Ainsi, en bash grâce aux commandes :

```
jq '.data[].vmid' | sort | tr '\n' ' ' | tr -d ' '
```

Je peux obtenir les numéros des conteneurs, soit ici : “950, 951”

Cette liste va permettre de déterminer quel nouveau conteneur créer. Il a été déterminé que la plage des CTID de 950 à 999 sera réservée à la création des nouveaux conteneurs. Ainsi grâce à une boucle je peux déterminer le premier CTID de libre dans cette plage. À partir de ce CTID, je peux lui assigner une IP. Utilisant un **CIDR** /24, seul le dernier octet change d’un conteneur à l’autre. Nous choisissons d’attribuer la plage IP allant de 192.168.102.150 à 192.168.102.199, ainsi en effectuant la soustraction de 800 par le numéro du CTID, nous obtenons le dernier octet de notre conteneur.

```

# Determine the first available CT_ID
for CT_ID in $(seq 950 999)
do
    if [[ ! "${NEW_LIST}" =~ "${CT_ID}" ]]
    then
        break
    fi
done

```

```

LAST_IP_BYTE=$(( ${CT_ID}-800 ))
HOSTNAME="oss-${CT_ID}.internal"

echo "Container ID: ${CT_ID}"
echo "Container IP: 192.168.102.${LAST_IP_BYTE}"
echo "Container hostname: ${HOSTNAME}"

```

L'étape suivante du script est la saisie des **arguments**, comme expliqué précédemment nous souhaitons pouvoir saisir en **argument** , le nom du proxmox, l'**image** souhaitée à envoyer à notre **conteneur**, ainsi que le mot de passe du proxmox.

```

usage() {
    echo "script usage: $0 -n proxmox_name -o OS [ -p proxmox_password
]"
    exit 1
}

while getopts 'n:p:o:' OPTION
do
    case "$OPTION" in
        n)
            PROXMOX_NAME=${OPTARG}
            ;;

        p)
            PROXMOX_PASSWORD=${OPTARG}
            ;;

        o)
            OS_PARAMETER=${OPTARG}
            if [ ${OS_PARAMETER} = "centos6" ]
            then
                OS='local:vztmpl/centos-6-default_20191016_amd64.tar.xz'
            elif [ ${OS_PARAMETER} = "centos7" ]
            then
                OS='local:vztmpl/centos-7-default_20190926_amd64.tar.xz'
            elif [ ${OS_PARAMETER} = "almalinux8" ]
            then
                OS='local:vztmpl/almalinux-8-default_20210928_amd64.tar.xz'
            fi
            ;;
    esac
done

```

```

*)
    usage
    ;;

esac
done

```

La suite du script va permettre la création de mot de passe pour les futurs utilisateurs du **conteneur**. Il y en aura pour l'instant 3.

```

OPERATION_PASSWORD=$(openssl rand -base64 48)
ANSIBLE_PASSWORD=$(openssl rand -base64 48)
ROOT_PASSWORD=$(openssl rand -base64 48)

```

A ce stade du script, toutes les valeurs nécessaires à l'exécution du playbook permettant de créer le prochain conteneur sont enregistrées dans des variables.

Nous disposons :

- Du serveur proxmox et de son mot de passe
- Du CTID du prochain conteneur
- De l'adresse IP du prochain conteneur
- De l'image souhaitée
- Des mots de passe des utilisateurs root, operation, et ansible_user

Désormais, il est possible d'appeler le playbook précédent (annexe 5) en y spécifiant toutes les variables précédentes grâce à la commande de lancement de playbook suivante :

```

ansible-playbook --extra-vars "host='${PROXMOX_NAME}'
password='${PROXMOX_PASSWORD}' container='${CT_ID}'
last_ip_byte='${LAST_IP_BYTE}' hostname='${HOSTNAME}' os='${OS}'
root_password='${ROOT_PASSWORD}' "
${script_path}/playbooks/create_lxc_container.yaml

```

Le playbook va donc se servir de ces variables afin de personnaliser le nouveau conteneur, il va ensuite le démarrer, puis lancer l'installation d'**openSSH**. S'en suivront l'appel du premier

playbook ‘list and register’ créé précédemment, ce qui permettra de rendre opérationnelle la connexion **SSH**.

La difficulté pour la suite de ce projet va être la personnalisation des conteneurs suivant son image. En effet, il existe de fortes spécificités entre Centos et Almalinux, ainsi il va falloir adapter le comportement du playbook suivant l’image préalablement indiquée en argument du script. L’idée est d’utiliser la variable “OS” contenant le nom de l’**image** afin qu’elle pointe vers d’autres fichiers **YAML**.

Ansible prévoit un module: ‘include_taks’ permettant, depuis un playbook d’inclure des tâches provenant d’autres fichiers **yaml**. Dans notre cas, c’est une bonne méthode afin de segmenter les actions selon l’**image**. Nous aurons donc un playbook “principal” qui va déléguer certaines actions à d’autres fichiers **YAML**. Ainsi, suivant l’**image**, certains fichiers seront appelés et d’autres non.

Voici un exemple permettant d’installer openssh-server :

```
# Déterminer l'OS et le sauvegarder dans une variable Ansible.
- name: change os input
  set_fact: os_name="{{ (os.split("/") [1]).split('-') [:2] |
join('-') }}"

# Installer OpenSSH-server
- name: Install openssh-server
  include_tasks: "{{ os_name }}_install_openssh-server.yaml"
```

Ici, “{{ os_name }}_install_openssh-server.yaml” pointe vers le playbook pour installer openssh-server pour centos, ou pour almalinux.

Ainsi, je vais créer un playbook (annexe 7) permettant d’exécuter la liste (annexe 4) m’ayant été fourni afin de personnaliser notre conteneur. Dès qu’une action n’est pas réalisable sur les deux **images**, cette étape contiendra un “include_tasks” permettant de personnaliser l’action selon l’image.

4) Configuration automatique du fichier “host”

Nous avons vu précédemment que Ansible se sert d’un fichier de configuration ‘host’ afin de déterminer sa zone d’action, c’est-à-dire sur quel conteneur s’exécuter.

Ce fichier est divisé en plusieurs sections permettant d’effectuer des groupes. Lorsqu’un playbook est exécuté, il agit sur le groupe spécifié. Ce fichier ressemble à ceci :

```
[proxmox_lab]
192.168.92.111 ansible_user=root ansible_become_pass="*****"
```

```
ansible_ssh_pass="*****"

[containers_europe]
oss-950.internal ansible_user=ansible
oss-951.internal ansible_user=ansible
```

L'enjeu désormais, afin de continuer sur une approche DevOps est de maintenir et mettre à jour automatiquement ce fichier. Jusqu'à maintenant il est nécessaire d'écrire manuellement le host dans le fichier. Il m'est donc demandé lors de la création des nouveaux **conteneurs**, que cette liste soit actualisée avec ce nouveau **conteneur**, et dans la section dédiée, dans notre cas "[containers_europe]" Cette étape sera réalisée en **python** et est pour moi une occasion d'un peu plus me familiariser avec ce langage. C'est l'occasion pour moi de mettre en pratique les notions acquises durant le module paradigme de programmation. De plus, python est l'une des compétences clé du DevOps. L'idée est, à partir d'un fichier **csv** contenant les noms de nos **conteneurs**, de mettre à jour le fichier "hosts".

Le script python commence par importer les modules nécessaires (pathlib, argparse et sys) et instancie un objet argparse.ArgumentParser() pour gérer les **arguments** de la ligne de commande. L'argument -f est défini comme étant le nom du fichier **CSV** contenant les noms d'hôtes à ajouter.

```
import pathlib
import argparse
import sys

parser = argparse.ArgumentParser(
    prog=sys.argv[0],
    add_help=True
)
parser.add_argument('-f',
                    action='store',
                    dest='hostname_csv',
                    required=True,
                    help='CSV file containing all hostnames to add to
ansible hosts file.')
args_parsed = parser.parse_args()

Dictionnary = {}
```

Il lit ensuite le contenu du fichier /etc/ansible/hosts et stocke son contenu dans un **dictionnaire** nommé Dictionary préalablement créé. Cette section permet de définir les groupes, ces

groupes sont des lignes commençant par “[” et se terminant par “]”.

```
with open("/etc/ansible/hosts", "r") as file:
    for line in file:
        if line.startswith("\n"):
            continue
        if line.startswith("["):
            Table = []
            Section = line[1:line.index(")]")
            Dictionnary[Section] = Table
            continue
        Table.append(line.rstrip('\n'))
```

Puis, il ouvre le fichier **CSV** dont le nom est fourni en **argument**, lit son contenu et stocke les noms des **conteneurs** dans une liste nommée `hostnames_list`. Cette liste est nettoyée en supprimant les doublons et les éléments vides. Il boucle sur chaque élément de la liste `hostnames_list`. Pour chaque élément, il vérifie s'il est déjà présent dans la section `containers_europe` du dictionnaire. Si ce n'est pas le cas, il ajoute une nouvelle entrée à cette section avec le nom du **conteneur** et l'utilisateur Ansible associé.

```
with open("/etc/ansible/hosts", "r") as file:
    for line in file:
        if line.startswith("\n"):
            continue
        if line.startswith("["):
            Table = []
            Section = line[1:line.index(")]")
            Dictionnary[Section] = Table
            continue
        Table.append(line.rstrip('\n'))

# Récupérer le fichier hostname_containers.csv en argument du script
python.

srv = open(args_parsed.hostname_csv, 'r')
data = srv.read()
hostnames_list = list(set(data.split("\n")))
srv.close()

for element in hostnames_list:
    new_ansible_host = True
    if element != "":
```

```

for entry in Dictionnary[section_name]:
    if element in entry:
        new_ansible_host = False
        break
if new_ansible_host:
    Dictionnary[section_name].append(f"{element}
ansible_user=4n51bl3")

```

Enfin, le script écrit le contenu mis à jour du **dictionnaire** dans le fichier `/etc/ansible/hosts`. Chaque section est écrite en tant que ligne commençant par `[nom_de_la_section]`, suivie des lignes correspondantes. Les lignes vides sont ignorées.

```

with open('/etc/ansible/hosts', 'w') as file:
    for key, value in Dictionnary.items():
        file.write('\n' + '[' + key + ']' + '\n')
        for ligne in value:
            if ligne != '':
                file.write(ligne + '\n')

```

Ce script **Python** permet donc d'ajouter des noms d'hôtes à un fichier de configuration Ansible, en utilisant un fichier **CSV** comme source de données.

5) Notification

Toujours dans la logique d'obtenir une trace des actions effectuées, la dernière étape consiste à réaliser un playbook (annexe 8) afin d'avertir le service **SysAdmin** de la création d'un nouveau **conteneur**. Ce playbook permettra la création d'un mail automatique à chaque exécution du script **SHELL/BASH** regroupant toutes les actions précédentes. Cette notification inclura tous les mot de passe liés à cette création générés.

À : devops

Ven 21/04/2023 16:30

Hello,

You'll find here, the list of all modifications done by Ansible.

New container created:

Container: 'oss-958.internal'

OS: 'local:vztmpl/almaLinux-8-default_20210928_amd64.tar.xz'

IP: 192.168.102.158

ROOT password:

'f3PjbJpSBkJK2xQzhGoMH+URaVsKYY1yZ8ew3kQOCupyAK26diu8VB1NrJp9x
dGW'

ANSIBLE password

'IZFAR80srWJvVWDMHRJpr5wj6Ftuthv1IWL9LRkhPCCa/7aYVQm3UVsXaR4Z
dT8'

OPERATION password:

'gcYTpkgO2LEvyT5e9TC46+NFmVKcQdqT1snm2YYjHfjVfKrWymwu0KzrdxfRRx
MW'

Best Regards,

Bilan

1) Les points forts

Pour Ivès, un des points forts du projet est la mise en place d'un outil permettant une meilleure gestion des ressources. Les scripts créés permettent de déployer rapidement les nouveaux conteneurs, ce qui représente un gain dans l'efficacité et la productivité de l'équipe. De plus, l'avantage de l'automatisation est la réduction des risques d'erreurs humaines, ceci permet d'éviter des problèmes de sécurité ou de fiabilité à l'avenir.

De mon côté, ce projet m'a permis de travailler sur un environnement exclusivement Linux, en ligne de commande, ainsi que de me familiariser avec cette manière de travailler. J'ai appris beaucoup de notions, que ce soit en système ou en développement, que j'ai mis en pratique dans des projets. J'ai pu prendre, pour chaque projet, le temps de me documenter sur les technologies. Bastien mon tuteur ayant beaucoup d'expérience a très bien su me déléguer le travail, m'accompagner dans la réalisation de celui-ci, ainsi que de prendre le temps de m'encadrer tout au long de mon contrat.

2) Axe d'amélioration

Ayant suivi une licence informatique, couvrant de nombreux domaines de ce secteur et en parallèle un poste d'apprenti DevOps couvrant lui aussi de nombreux aspect en système et en développement, je n'ai pas réellement pu me spécialiser, me laissant parfois l'impression de survoler certain concepts avant de passer à un autre.

Conclusion

Ce projet m'a permis de développer mes compétences en automatisation et en gestion d'infrastructure. J'ai appris à écrire des playbooks efficaces en utilisant Ansible pour gérer les opérations de déploiements et de configuration des conteneurs. En mettant en place cette solution, j'ai contribué à réduire et optimiser le temps que nécessite la maintenance de notre infrastructure. Dans le futur, de nombreux nouveaux playbooks verront le jour. Ce projet continuera d'évoluer.

La licence m'a apporté les bases qu'il me manquaient dans plusieurs domaines. Au fil de l'année, j'ai pu aborder les projets avec plus de sérénité, et avoir une meilleure compréhension technique et humaine de l'environnement dans lequel je me trouve. Sans oublier le module d'anglais m'ayant fortement servi, mes recherches étant dorénavant exclusivement en anglais.

Je suis heureux d'avoir pu intégrer le service DevOps d'Ivès, et travaillé sur des projets enrichissants et stimulants. J'ai pu observer l'étendue et la complexité du métier à travers divers projets concrets. Je peux désormais mieux orienter mon apprentissage afin de combler mes lacunes. En effet, je vais continuer et me spécialiser dans l'univers du DevOps / SysOps car aujourd'hui, le besoin d'acquérir une expertise se fait ressentir.

Bibliographie

Rohaut Sébastien (2021) - Linux Maitriser l'administration système - édition ENI

Rohaut Sébastien (2020) - Les fondamentaux du langage - édition ENI

Sitographie

Formation :

OpenClassRoom : <https://openclassrooms.com/fr/>

Udemy : <https://www.udemy.com/>

Documentation :

Documentation ansible : <https://docs.ansible.com/>

Bash : <https://devdocs.io/bash/>

Python : <https://docs.python.org/3/>

Articles :

Ansible : <https://geekflare.com/tag/ansible/>

Linux : <https://www.redhat.com/>

Annexes

Annexe 1 :

Playbook “list and register”

```
# "{{host}}" is used with ansible-playbook command, by adding
--extra-vars "host=whatever"
- hosts: "{{ host }}"
  become: yes
  tasks:

# Dans le run.sh, ajouter automatiquement les hosts manquants

#####
# USE KEY IN VARIABLE                                     #
#####

- name: Register content of /sources_files/ansible_RSA_key.pub into
variable
  set_fact:
    file_contents_RSA: "{{ lookup('pipe', 'cat
../source_files/ansible_rsa_key.pub') }}"

- name: Register content of /sources_files/ansible_ed25519_key.pub
into variable
  set_fact:
    file_contents_ed25519: "{{ lookup('pipe', 'cat
../source_files/ansible_ed25519_key.pub') }}"

#####
# LXC                                                     #
#####

- name: Get LXC CT_ID list
  shell: /usr/sbin/pct list | awk 'NR>1' | tr -s ' ' / | cut -d '/'
-f1
  register: lxclist_proxmox

- name: Get LXC CT_ID hostnames
  shell: /usr/sbin/pct exec {{ item }} -- hostname
  loop: "{{ lxclist_proxmox.stdout_lines }}"
  ignore_errors: true
```

```

    register: lxc_hostname_list

- name: Save hostname to CSV file
  local_action:
    module: copy
    content: "{{ lxc_hostname_list.results |
map(attribute='stdout') | join('\n') }}"
    dest: ../retrieved_files/proxmox/lxclist_{{ inventory_hostname
}}.csv

- name: Check if ansible user exist for LXC container
  shell : /usr/sbin/pct exec {{ item }} -- id ansible
  loop: "{{ lxclist_proxmox.stdout_lines }}"
  ignore_errors: true
  register: lxc_ansible_user

- name: Create new user for LXC container
  shell: /usr/sbin/pct exec {{ item.item }} -- useradd -m -s
/bin/bash ansible
  loop: "{{ lxc_ansible_user.results }}"
  ignore_errors: true
  when: item.stdout | length == 0
  register: new_user_lxc

- name: register LXC container's user added
  local_action: shell echo "{{ inventory_hostname }};{{
item.item.item }};$(ansible {{ inventory_hostname }} -b -u operation
-m shell -a '/usr/sbin/pct exec {{ item.item.item }} -- hostname' |
grep -v CHANGED); user added" >> ../retrieved_files/modifications.csv
  loop: "{{ new_user_lxc.results }}"
  ignore_errors: true
  when: item.item.stderr is search("no such user", ignorecase=True)
  register: container_modified

- name: Get OpenSSL version
  shell: /usr/sbin/pct exec {{ item }} -- openssl version | cut -c
1-11
  loop: "{{ lxclist_proxmox.stdout_lines }}"
  register: openssl_versions

- name: Create .ssh directory and authorized_keys if they don't
exist

```

```

    shell: |
        /usr/sbin/pct exec {{ item }} -- ls
/home/ansible/.ssh/authorized_keys
    if [ $? -ne 0 ]
    then
        /usr/sbin/pct exec {{ item }} -- mkdir -p /home/ansible/.ssh
        /usr/sbin/pct exec {{ item }} -- touch
/home/ansible/.ssh/authorized_keys
    fi
    loop: "{{ lxclist_proxmox.stdout_lines }}"
    ignore_errors: true
    register: file_and_directory_added_lxc

- name: Check if RSA SSH Key exists
    # If grep find nothing in the file, it return 1 as error code and
    # Ansible does not expect error code >= 0 it's why some steps are in
    # error.
    shell: /usr/sbin/pct exec {{ item.item }} -- grep {{
file_contents_RSA.split()[1] }} /home/ansible/.ssh/authorized_keys
    loop: "{{ openssl_versions.results }}"
    when: item.stdout <= "OpenSSL 1.0"
    ignore_errors: true
    register: rsa_ssh_key_exists

- name: Add RSA SSH key
    shell: /usr/sbin/pct exec {{ item.item.item }} -- bash -c 'echo
"ssh-rsa {{ file_contents_RSA.split()[1] }} ansible" >>
/home/ansible/.ssh/authorized_keys'
    loop: "{{ rsa_ssh_key_exists.results }}"
    when: item.skipped is not defined and item.stdout | length == 0
    ignore_errors: true
    register: RSA_lxc_key_added

- name: register RSA LXC container modified
    local_action: shell echo "{{ inventory_hostname }};{{
item.item.item.item }};$(ansible {{ inventory_hostname }} -b -u
operation -m shell -a '/usr/sbin/pct exec {{ item.item.item.item }}
-- hostname' | grep -v CHANGE);RSA key added" >>
../retrieved_files/modifications.csv
    loop: "{{ RSA_lxc_key_added.results }}"
    when: item.item.stdout is defined and item.item.stdout | length
== 0

```

```

    ignore_errors: true
    register: container_lxc_RSA_modified

- name: Check if ED25519 LXC SSH Key exists
  # If grep find nothing in the file, it return 1 as error code and
  # Ansible does not expect error code >= 0 it's why some steps are in
  # error.
  shell: /usr/sbin/pct exec {{ item.item }} -- grep {{
file_contents_ed25519.split()[1] }}
/home/ansible/.ssh/authorized_keys
  loop: "{{ openssl_versions.results }}"
  when: item.stdout >= "OpenSSL 1.1"
  ignore_errors: true
  register: ed25519_LXC_ssh_key_exists

- name: Add ED25519 LXC SSH key
  shell: /usr/sbin/pct exec {{ item.item.item }} -- bash -c 'echo
"ssh-ed25519 {{ file_contents_ed25519.split()[1] }} ansible" >>
/home/ansible/.ssh/authorized_keys'
  loop: "{{ ed25519_LXC_ssh_key_exists.results }}"
  when: item.skipped is not defined and item.stdout | length == 0
  ignore_errors: true
  register: ED25519_LXC_key_added

- name: register ED25519 LXC container modified
  local_action: shell echo "{{ inventory_hostname }};{{
item.item.item.item }};$(ansible {{ inventory_hostname }} -b -u
operation -m shell -a '/usr/sbin/pct exec {{ item.item.item.item }}
-- hostname' | grep -v CHANGED);ED25519 key added" >>
../retrieved_files/modifications.csv
  loop: "{{ ED25519_LXC_key_added.results }}"
  when: item.item.stdout is defined and item.item.stdout | length
== 0
  ignore_errors: true
  register: container_LXC_ED25519_modified

#####
# OPENVZ #
#####

- name: Get OPENVZ CT_ID list
  shell: /usr/sbin/vzlist --no-header | tr -s ' ' / | cut -d '/'

```

```

-f2
  register: openvzlist_proxmox

- name: Get OPENVZ CT_ID hostnames
  shell: /usr/sbin/vzctl exec {{ item }} hostname
  loop: "{{ openvzlist_proxmox.stdout_lines }}"
  register: openvz_hostname_list

- name: Save hostname to CSV file
  local_action:
    module: copy
    content: "{{ openvz_hostname_list.results |
map(attribute='stdout') | join('\n') }}"
    dest: ../retrieved_files/proxmox/openvzlist_{{
inventory_hostname }}.csv

- name: Check if ansible user exist for openvz container
  shell : /usr/sbin/vzctl exec {{ item }} id ansible
  loop: "{{ openvzlist_proxmox.stdout_lines }}"
  ignore_errors: true
  register: openvz_ansible_user

- name: Create new user for openvz container
  shell: /usr/sbin/vzctl exec {{ item.item }} useradd -m -s
/bin/bash ansible
  loop: "{{ openvz_ansible_user.results }}"
  ignore_errors: true
  when: item.stdout | length == 0
  register: new_user_openvz

- name: register OPENVZ container's user added
  local_action: shell echo "{{ inventory_hostname }};{{
item.item.item }};$(ansible {{ inventory_hostname }} -b -u operation
-m shell -a '/usr/sbin/vzctl exec {{ item.item.item }} hostname' |
grep -v CHANGED);user added" >> ../retrieved_files/modifications.csv
  loop: "{{ new_user_openvz.results }}"
  ignore_errors: true
  when: item.item.stderr is search("no such user", ignorecase=True)
  register: container_modified

- name: Get OPENVZ OpenSSL version
  shell: /usr/sbin/vzctl exec {{ item }} openssl version | cut -c

```



```

1-11
    loop: "{{ openvzlist_proxmox.stdout_lines }}"
    register: openssl_openvz_versions

- name: Create .ssh directory and authorized_keys if they don't
exist
    shell: |
        /usr/sbin/vzctl exec {{ item }} ls
/home/ansible/.ssh/authorized_keys
        if [ $? -ne 0 ]
        then
            /usr/sbin/vzctl exec {{item}} mkdir -p /home/ansible/.ssh
            /usr/sbin/vzctl exec {{item}} touch
/home/ansible/.ssh/authorized_keys
        fi
    loop: "{{ openvzlist_proxmox.stdout_lines }}"
    register: file_and_directory_added_openvz

- name: Check if RSA SSH Key exists
    # If grep find nothing in the file, it return 1 as error code and
Ansible does not expect error code >= 0 it's why some steps are in
error.
    shell: /usr/sbin/vzctl exec {{ item.item }} grep {{
file_contents_RSA.split()[1] }} /home/ansible/.ssh/authorized_keys
    loop: "{{ openssl_openvz_versions.results }}"
    when: item.stdout <= "OpenSSL 1.0"
    ignore_errors: true
    register: rsa_openvz_ssh_key_exists

- name: Add RSA SSH key
    shell: /usr/sbin/vzctl exec {{ item.item.item }} 'printf "%s\n"
"ssh-rsa {{ file_contents_RSA.split()[1] }} ansible" >>
/home/ansible/.ssh/authorized_keys'
    loop: "{{ rsa_openvz_ssh_key_exists.results }}"
    when: item.skipped is not defined and item.stdout | length == 0
    ignore_errors: true
    register: RSA_key_added

- name: register RSA container modified
    local_action: shell echo "{{ inventory_hostname }}";{{
item.item.item.item }};$(ansible {{ inventory_hostname }} -b -u
operation -m shell -a '/usr/sbin/vzctl exec {{ item.item.item.item }}

```

```

hostname' | grep -v CHANGE);RSA key added" >>
../retrieved_files/modifications.csv
    loop: "{{ RSA_key_added.results }}"
    when: item.item.stdout is defined and item.item.stdout | length
== 0
    ignore_errors: true
    register: container_RSA_modified

- name: Check if ED25519 SSH Key exists
  # If grep find nothing in the file, it return 1 as error code and
Ansible does not expect error code >= 0 it's why some steps are in
error.
  shell: /usr/sbin/vzctl exec {{ item.item }} grep {{
file_contents_ed25519.split()[1] }}
/home/ansible/.ssh/authorized_keys
  loop: "{{ openssl_openvz_versions.results }}"
  when: item.stdout >= "OpenSSL 1.1"
  ignore_errors: true
  register: ed25519_openvz_ssh_key_exists

- name: Add ED25519 SSH key
  shell: /usr/sbin/vzctl exec {{ item.item.item }} 'printf "%s\n"
"ssh-ed25519 {{ file_contents_ed25519.split()[1] }} ansible" >>
home/ansible/.ssh/authorized_keys'
  loop: "{{ ed25519_openvz_ssh_key_exists.results }}"
  when: item.skipped is not defined and item.stdout | length == 0
  register: ED25519_key_added

- name: register ED25519 container modified
  local_action: shell echo "{{ inventory_hostname }};{{
item.item.item.item }};$(ansible {{ inventory_hostname }} -b -u
operation -m shell -a '/usr/sbin/vzctl exec {{ item.item.item.item }}
hostname' | grep -v CHANGED);ED25519 key added"
>>../retrieved_files/modifications.csv
    loop: "{{ ED25519_key_added.results }}"
    when: item.item.stdout is defined and item.item.stdout | length
== 0
    ignore_errors: true
    register: container_ED25519_modified

```

Annexe 2 :

Notification par mail

```
- hosts: localhost
  tasks:
    - name: Test if modifications.csv file exists
      stat: path=../retrieved_files/modifications.csv
      register: file_exists
    - name: Send emails to a bunch of users, with playbook report as an
      attachment.
      mail:
        host: localhost
        port: 25
        subject: "[Ansible] SSH creation report"
        body: "Hello,\n\nYou'll find in the attached file, the list of
all modifications done by Ansible.\n\nBest Regards,"
        attach:
          - ../retrieved_files/modifications.csv
        charset: us-ascii
      when: file_exists.stat.exists
```

Annexe 3

Script run.sh

```
#!/bin/bash

#ADD SSH KEY AND REGISTER MODIFICATION
ansible-playbook --extra-vars "host=all_proxmox_europe"
playbooks/list_and_register.yaml

for file in retrieved_files/*list_*.csv
do
    echo -e "$(cat $file)\n" >> retrieved_files/all_hostnames.csv
done

#Delete lines breaks
sed -i '/^$/d' retrieved_files/all_hostnames.csv

cat retrieved_files/all_hostnames.csv | while read LINE
do
    ssh-keyscan ${LINE} >> /home/ansible/.ssh/known_hosts
done
```

```
#SENDING MAIL
ansible-playbook --extra-vars "host=all_proxmox_europe"
playbooks/send_email.yaml

# Cleaning temporary files
rm -f retrieved_files/*list_*.csv retrieved_files/all_hostnames.csv
retrieved_files/modifications.csv
```

Annexe 4:

Liste afin de personnaliser les conteneurs

1. Installation des dépôts linux suivants :
 1. epel-release
 2. centos-release-scl
2. Faire un yum update de l'OS
3. Installation des paquets suivants :
 1. bc
 2. deltarpm
 3. file
 4. haproxy
 5. less
 6. mailx
 7. man-pages
 8. man
 9. multitail
 10. nano
 11. net-tools
 12. patch
 13. screen
 14. sendmail
 15. sendmail-cf
 16. strace
 17. tcpdump
 18. telnet
 19. tree
 20. unzip
 21. vim
 22. wget
 23. yum-utils
 24. zip
 25. bind-utils
 26. iftop
 27. iotop
 28. iptraf-ng
 29. monit
 30. mysql
 31. nagios-plugins-disk
 32. nagios-plugins-load
 33. nagios-plugins-procs

- 34. net-snmp
- 35. nethogs
- 36. nmap
- 37. nrpe
- 38. perf
- 39. psmisc
- 40. wireshark
- 4. Set locale to en_US.utf8
- 5. Change timezone to UTC
- 6. Change YUM configuration
 - 1. sed -i 's/installonly_limit=5/installonly_limit=15/g' /etc/yum.conf
- 7. Customizing SECURITY LIMITS
 - 1. sed --in-place '/^# End of file/i * soft core unlimited' /etc/security/limits.conf
- 8. Customizing SSH config
 - 1. sed --in-place 's/^#PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config
 - 2. sed --in-place 's/^#UseDNS yes/UseDNS no/' /etc/ssh/sshd_config
 - 3. sed --in-place 's/^#ClientAliveInterval 0/ClientAliveInterval 120/' /etc/ssh/sshd_config
 - 4. sed --in-place 's/^PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config
 - 5. sed --in-place 's/^GSSAPIAuthentication yes/GSSAPIAuthentication no/' /etc/ssh/sshd_config
- 9. Customizing HISTORY config
 - 1. sed --in-place '/HISTSIZE=1000/a HISTFILESIZE=1000000000' /etc/profile
 - 2. sed --in-place '/HISTFILESIZE/a PROMPT_COMMAND="history -a; history -c; history -r; \\${PROMPT_COMMAND}"' /etc/profile
 - 3. sed --in-place '/HISTSIZE=1000/i export HISTTIMEFORMAT "[%F %T] '" /etc/profile
- 10. Creating user Operation
 - 1. useradd --create-home operation
 - 2. sed --in-place '/^root/a operation ALL=(ALL) ALL' /etc/sudoers
- 11. Retrieving RPMs list
 - 1. repoquery --all --installed | sort > base_rpms_list.txt

Annexe 5:

Création de conteneurs

```
# "{{host}}" is used with ansible-playbook command, by adding
--extra-vars "host=whatever"
- hosts: "{{ host }}"
  become: yes
  tasks:

- name: New LXC container creation
  community.general.proxmox:
    vmid: "{{ container }}"
    node: pve
    api_user: root@pam
```

```

    api_password: "{{ password }}"
    api_host: "{{ host }}"
    hostname: "{{ hostname }}"
    ostemplate: "{{ os }}"
    memory: 512
    swap: 20
    cpus: 1
    disk: 8
    password: "{{ root_password }}"
    netif: '{"net0": "name=eth0,gw=192.168.0.1,ip=192.168.102.{{
last_ip_byte }}/16,bridge=vbr0"}'

- name: Start container
  community.general.proxmox:
    vmid: "{{ container }}"
    api_user: root@pam
    api_password: "{{ password }}"
    api_host: "{{ host }}"
    state: started

- name: Pause for 5 seconds
  ansible.builtin.pause:
    seconds: 5

# Déterminer l'OS et le sauvegarder dans une variable Ansible.
- name: change os input
  set_fact: os_name="{{ (os.split("/") [1]).split('-')[:2] |
join('-') }}"

# Installer OpenSSH-server
- name: Install openssh-server
  include_tasks: "{{ os_name }}_install_openssh-server.yaml"

```

Annexe 6:

Script SHELL/BASH

```

#!/bin/bash

script_path=$(dirname "$(readlink --canonicalize "$0")")

```

```

usage() {
    echo "script usage: $0 -n proxmox_name -o OS [ -p proxmox_password
]"
    exit 1
}

while getopts 'n:p:o:' OPTION
do
    case "$OPTION" in
        n)
            PROXMOX_NAME=${OPTARG}
            ;;

        p)
            PROXMOX_PASSWORD=${OPTARG}
            ;;

        o)
            OS_PARAMETER=${OPTARG}
            if [ ${OS_PARAMETER} = "centos6" ]
            then
                OS='local:vztmpl/centos-6-default_20191016_amd64.tar.xz'
            elif [ ${OS_PARAMETER} = "centos7" ]
            then
                OS='local:vztmpl/centos-7-default_20190926_amd64.tar.xz'
            elif [ ${OS_PARAMETER} = "almalinux8" ]
            then
                OS='local:vztmpl/almalinux-8-default_20210928_amd64.tar.xz'
            fi
            ;;

        *)
            usage
            ;;

    esac
done

# If the script was run without argument
# OR
# If the script does not use mandatory argument (-n and -o)

```

```

if [ $# -lt 1 ] || [ -z "${PROXMOX_NAME}" ] || [ -z "${OS}" ]
then
    usage
fi

# Check if python3 is installed
if [ -z "$(command -v python3)" ]
then
    echo "Python 3 is not installed, please install it first."
    exit 1
fi

# Check if the proxmox password was given as argument (-p)
if [ -z "${PROXMOX_PASSWORD}" ]
then
    read -r -s -p "${PROXMOX_NAME}'s password: " PROXMOX_PASSWORD
    echo ""
fi

# Create PVE API token
TOKEN=$(curl --silent --insecure --data "username=root" --data
"realm=pam" --data-urlencode "password=${PROXMOX_PASSWORD}"
https://${PROXMOX_NAME}:8006/api2/json/access/ticket)
echo "$TOKEN" | jq --raw-output '.data.ticket' | sed
's/^/PVEAuthCookie=/ ' > ${script_path}/source_files/cookie
echo "$TOKEN" | jq --raw-output '.data.CSRFPreventionToken' | sed
's/^/CSRFPreventionToken:/ ' > ${script_path}/source_files/csrf_token

# Check if the PVE API token is valid
if [ -z "$(curl --silent --insecure --cookie "$(cat
${script_path}/source_files/cookie)"
https://${PROXMOX_NAME}:8006/api2/json/nodes)" ]
then
    echo "PVE API token is not valid. Check the proxmox password you
have entered."
fi

# Retrieve CT_ID list
NEW_LIST=$(curl --silent --insecure --cookie "$(cat
${script_path}/source_files/cookie)"
https://${PROXMOX_NAME}:8006/api2/json/nodes/pve/lxc | jq

```



```

'.data[].vmid' | sort | tr '\n' ' ' | tr -d '"')

# Determine the first available CT_ID
for CT_ID in $(seq 950 999)
do
    if [[ ! "${NEW_LIST}" =~ "${CT_ID}" ]]
    then
        break
    fi
done

LAST_IP_BYTE=$(( ${CT_ID} - 800 ))
HOSTNAME="oss-${CT_ID}.internal"

echo "Container ID: ${CT_ID}"
echo "Container IP: 192.168.102.${LAST_IP_BYTE}"
echo "Container hostname: ${HOSTNAME}"

OPERATION_PASSWORD=$(openssl rand -base64 48)
ANSIBLE_PASSWORD=$(openssl rand -base64 48)
ROOT_PASSWORD=$(openssl rand -base64 48)

# Create the container
ansible-playbook --extra-vars "host='${PROXMOX_NAME}'
password='${PROXMOX_PASSWORD}' container='${CT_ID}'
last_ip_byte='${LAST_IP_BYTE}' hostname='${HOSTNAME}' os='${OS}'
root_password='${ROOT_PASSWORD}'"
${script_path}/playbooks/create_lxc_container.yaml

# Create ansible user into the container
ansible-playbook --extra-vars "host='${PROXMOX_NAME}'"
${script_path}/../install_ansible_ssh_keys/playbooks/list_and_register.yaml
ssh-keyscan ${HOSTNAME} >> ~/.ssh/known_hosts
ssh -q -o BatchMode=yes -o StrictHostKeyChecking=no -o
ConnectTimeout=5 ansible@${HOSTNAME} 'exit 0'

# Add the new container into Ansible hosts file
echo "${HOSTNAME}" >>
${script_path}/retrieved_files/hostname_container.csv
sudo python3 ${script_path}/source_files/fill_ansible_hosts_file.py

```

```
-f ${script_path}/retrieved_files/hostname_container.csv

# Install
ansible-playbook --extra-vars "host='${HOSTNAME}' ansible_user=ansible
os='${OS}' ansible_password='${ANSIBLE_PASSWORD}'
operation_password='${OPERATION_PASSWORD}'"
${script_path}/playbooks/all_install_template.yaml

ansible-playbook --extra-vars "host='${HOSTNAME}' os='${OS}'
ansible_password='${ANSIBLE_PASSWORD}'
operation_password='${OPERATION_PASSWORD}'
root_password='${ROOT_PASSWORD}' ip='${LAST_IP_BYTE}'"
${script_path}/playbooks/send_email.yaml
```

Annexe 7

Template conteneur

```
# "{{host}}" is used with ansible-playbook command, by adding
--extra-vars "host=whatever"
- hosts: "{{ host }}"
  become: yes
  tasks:

  # Determine OS
  - name: Determine OS
    set_fact: os_name="{{ (os.split("/") [1]).split('-')[:2] |
join('-') }}"

  - name: Change ansible password
    user:
      name: "ansible"
      password: "{{ aaa_password }}"

  - name: Install RPM
    include_tasks: "{{ os_name }}_install_rpm.yaml"

  - name: Update the OS
    include_tasks: "{{ os_name }}_update_os.yaml"

  - name: Set locale to en_US.utf8
    include_tasks: "{{ os_name }}_set_utf8.yaml"
    when: os_name == "centos-6"
```

```

- name: Update yum.conf & dnf.conf
  include_tasks: "{{ os_name }}_update_yum&dnf_conf.yaml"

- name: Update limits.conf
  lineinfile:
    path: /etc/security/limits.conf
    line: '* soft core unlimited'
    insertbefore: '^# End of file'

- name: Customizing SSH config
  replace:
    path: /etc/ssh/sshd_config
    regexp: "{{ item.regexp }}"
    replace: "{{ item.replace }}"
  with_items:
    - { regexp: 'PermitRootLogin yes', replace: 'PermitRootLogin
no' } #OK sur almalinux
    - { regexp: '^.*UseDNS.*$', replace: 'UseDNS no' } #Déjà de
base en no sur almalinux
    - { regexp: '#ClientAliveInterval 0', replace:
'ClientAliveInterval 120' } #OK sur almalinux
    - { regexp: 'PasswordAuthentication yes', replace:
'PasswordAuthentication no' } #OK sur almalinux
    - { regexp: 'GSSAPIAuthentication yes', replace:
'GSSAPIAuthentication no' } #OK sur almalinux

- name: Customizing HISTORY config
  lineinfile:
    path: /etc/profile
    line: "{{ item.line }}"
    insertafter: "{{ item.insertafter }}"
  with_items:
    - { line: 'HISTFILESIZE=1000000000', insertafter:
'HISTSIZE=1000' }
    - { line: 'PROMPT_COMMAND="history -a; history -c; history -r;
$PROMPT_COMMAND"', insertafter: 'HISTFILESIZE' }
    - { line: 'HISTTIMEFORMAT "[%F %T] "', insertafter:
'HISTSIZE=1000' }

- name: Create 'operation' user
  user:

```

```

    name: operation
    password: "{{ operation_password }}"
    createhome: yes

- name: Update SUDOERS conf
  lineinfile:
    path: /etc/sudoers
    line: 'operation ALL=(ALL) ALL'
    insertafter: '^root'

```

Annexe 8:

Notification mail

```

- hosts: localhost
  tasks:

# Déterminer l'OS et le sauvegarder dans une variable Ansible.
- name: change os input
  set_fact: os_name="{{ (os.split("/") [1]).split('-') [:2] |
join('-') }}"

- name: Send emails to a bunch of users, with playbook report as an
attachment.
  mail:
    host: localhost
    port: 25
    subject: "[Test] [Ansible] container creation report"
    body: "Hello,\n\nYou'll find here, the list of all
modifications done by Ansible.\n\nNew container
created:\n\nContainer: '{{ host }}'\nOS: '{{ os_name }}'\nIP:
192.168.102.{{ ip }}'\nRoot password: '{{ root_password }}'\nansible
password '{{ ansible_password }}'\nOperation password: '{{
operation_password }}'\n\nBest Regards,"
    charset: us-ascii

```