PAGE DE GARDE DU DOSSIER PROFESSIONNEL

BREVET DE TECHNICIEN SUPÉRIEUR SERVICES INFORMATIQUES AUX ORGANISATIONS

Session 2026

DOSSIER PROFESSIONNEL

NOM : MOREAU

Prénom : Matt

Établissement de formation (sur un seul des deux exemplaires du dossier)

Visa du représentant de l'équipe pédagogique attestant la réalité des activités professionnelles décrites dans le dossier (sur un seul des deux exemplaires du dossier) :

Nom et qualité du signataire	Date	Signature

Attestation sur l'honneur pour les candidats individuels (sur un seul des deux exemplaires du dossier) :

Je soussigné(e), MOREAU , Matt , certifie que les activités décrites ainsi que les différentes informations reproduites dans ce dossier reflètent les activités professionnelles que j'ai personnellement réalisées au cours de ma formation.

Fait à Bouguenais
Date 21/04/2026

Signature

BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2026**

Fiche descriptive de réalisation professionnelle (recto)

Épreuve E6 - Administration des systèmes et des réseaux (option SISR)

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE	**N° réalisation : 2**
Nom, prénom : MOREAU Matt	**N° candidat : 02542583108**
Épreuve ponctuelle ☐ **Contrôle en cours de formation** ☒	**Date :** 28/05/2026

Organisation support de la réalisation professionnelle
Entreprise fictive **Oasis** et prestataire **NTxSystem**

Intitulé de la réalisation professionnelle
Mise en place d'une solution d'automatisation

Période de réalisation : 2024 - 2026 ***Lieu :*** *CFA Fab'Academy Bouguenais (UIMM)*
Modalité : ☐ *Seul(e)* ☒ *En équipe*

Compétences travaillées
- ☒ Concevoir une solution d'infrastructure réseau
- ☒ Installer, tester et déployer une solution d'infrastructure réseau
- ☒ Exploiter, dépanner et superviser une solution d'infrastructure réseau

Conditions de réalisation[1] (ressources fournies, résultats attendus)
Dans le cadre de la mission concernant la redondance et le déploiement, une solution d'automatisation et de déploiement de configurations sur des machines Linux à l'aide d'Ansible a été mise en place, afin de garantir une gestion centralisée, reproductible et cohérente des serveurs, tout en appliquant des configurations standardisées et sécurisées selon les besoins d'Oasis.

Description des ressources documentaires, matérielles et logicielles utilisées[2]

Différentes ressources ont été utilisées pour la mise en place de la solution d'automatisation, tout d'abord pour les ressources documentaires, la ressource principale utilisée a été la documentation officielle de Ansible, pour les ressources matérielles, un serveur HP avec l'hyperviseur VMware ESXi a été utilisé en tant que ressource matérielle, pour les ressources logicielles, VMware ESXi, GitLab, NetBox et Ansible ont été utilisés.

Modalités d'accès aux productions[3] et à leur documentation[4]
L'ensemble des documents liés à l'infrastructure est disponible sur le partage réseau accessible depuis le réseau BTS SIO. Cet emplacement est dédié au stockage des informations relatives à la section. Il contient notamment des documentations sur l'environnement virtuel déployé, l'ensemble de la configuration de l'infrastructure mise en place, les différentes solutions étudiées, le plan d'adressage ainsi que les différents schémas réalisés de l'infrastructure.

L'ensemble des mots de passe de l'infrastructure sont conservés dans notre gestionnaire de mot de passe Bitwarden.

Partage Réseau Documentation NTxSystem : \\partage.btssio.nte\fichiers\BAIES-PEDA\NTXSYSTEM

Identifiant Bitwarden : ntxsystem@proton.me
Mot de passe Bitwarden : NTxbitwarden44.
Lien Bitwarden : https://vault.bitwarden.com

[1] En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Administration des systèmes et des réseaux » prévues dans le référentiel de certification du BTS SIO.

[2] Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

[3] Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve.* ». Les éléments nécessaires peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2026**

Fiche descriptive de réalisation professionnelle
(verso, éventuellement pages suivantes)

Épreuve E6 - Administration des systèmes et des réseaux (option SISR)

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

A travers cette réalisation professionnelle portant sur l'automatisation du déploiement et la standardisation de configuration sur des serveurs Linux, différents outils ont été monté au sein de l'infrastructure, tout l'environnement est virtualisé sur deux serveurs HP utilisant VMware ESXI et il y a différentes machines virtuelles dédiées à différents services.

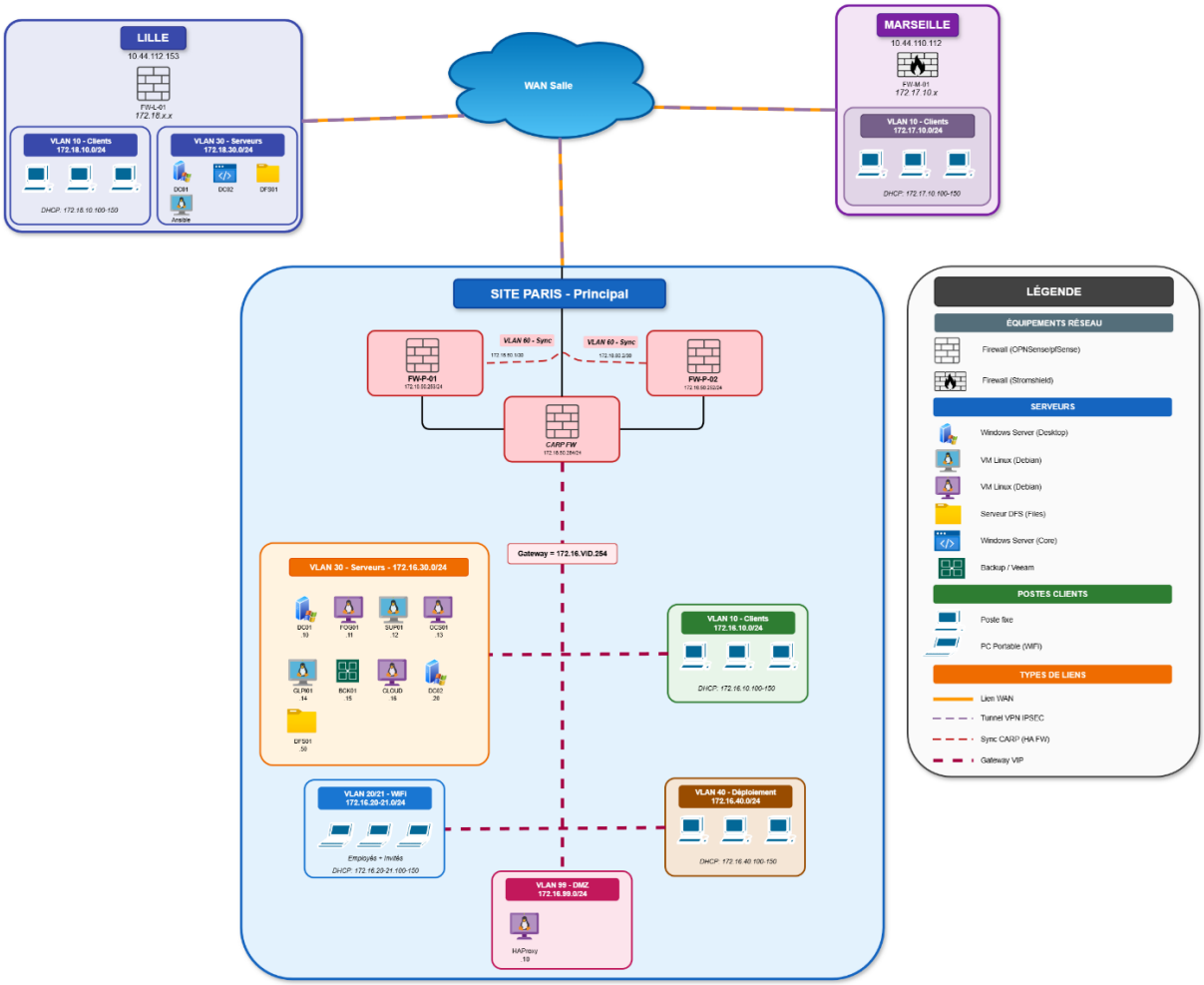
Cette infrastructure a été construite sur trois sites. Le premier correspondant au site commun au groupe NTxSystem, le site de Paris, le second correspondant au site de Marseille, sur lequel il est possible de retrouver des clients et le dernier correspondant à mon site personnel, se trouvant sur les Proxmox de la salle BTS SIO, le site de Lille.

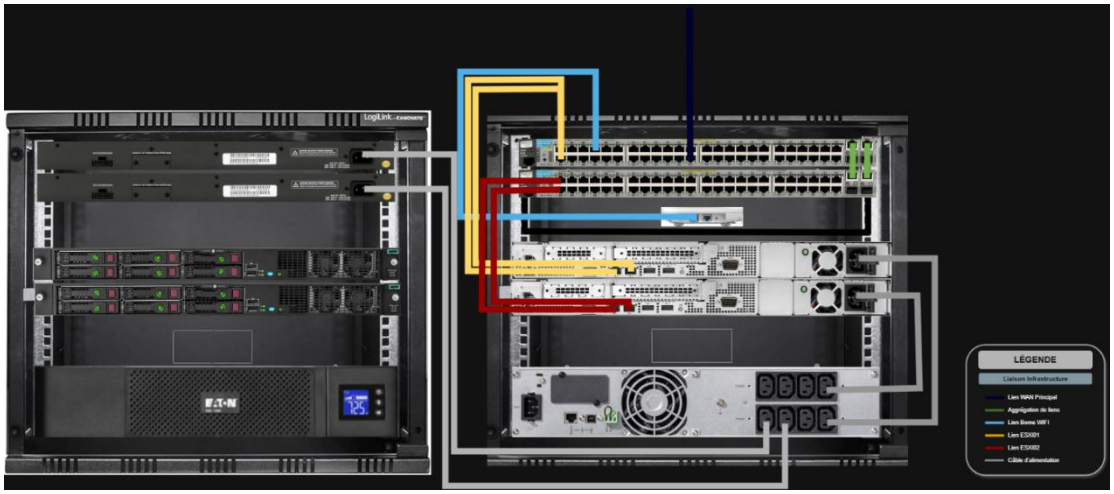
L'objectif principal de cette réalisation était de mettre en place une solution permettant déployer des configurations de manière centralisée, pour répondre à ce besoin, j'ai utilisé l'outil Ansible afin d'automatiser la configuration des serveurs, cette solution s'appuie sur un dépôt GitLab pour la gestion des fichiers de configuration et sur NetBox pour l'organisation des informations liées à l'infrastructure.

Cette solution permet donc de finalement simplifier l'administration des serveurs et réduire toutes configurations manuelles.

Ci-dessous les schémas logique et physique ainsi que le plan d'adressage de l'infrastructure.

[4] Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation, par exemples schéma complet de réseau mis en place et configurations des services.


LILLE
10.44.112.153
VLAN 10 - Clients
172.18.10.0/24
VLAN 30 - Serveurs
172.18.30.0/24
WAN Salle
MARSEILLE
10.44.110.112
VLAN 10 - Clients
172.17.10.0/24
SITE PARIS - Principal
FW-P-01
FW-P-02
CARP FW
VLAN 60 - Sync
Gateway = 172.16.VID.254
VLAN 30 - Serveurs - 172.16.30.0/24
VLAN 10 - Clients
172.16.10.0/24
VLAN 20/21 - WIFI
172.16.20-21.0/24
VLAN 40 - Déploiement
172.16.40.0/24
VLAN 99 - DMZ
172.16.99.0/24
LÉGENDE
ÉQUIPEMENTS RÉSEAU
Firewall (OPNSense/pfSense)
Firewall (Stormshield)
SERVEURS
Windows Server (Desktop)
VM Linux (Debian)
VM Linux (Debian)
Serveur DFS (Files)
Windows Server (Core)
Backup / Veeam
POSTES CLIENTS
Poste fixe
PC Portable (WIFI)
TYPES DE LIENS
Lien WAN
Tunnel VPN IPSEC
Sync CARP (HA FW)
Gateway VIP


LÉGENDE

Plage IP de base :	Description :
10.44.150.0/24	Plage IP du WAN

VLAN 10

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-02	172.16.10.252	255.255.255.0	172.16.10.0	172.16.10.254	IP FW-P-01 VLAN 10
FW-P-01	172.16.10.253	255.255.255.0	172.16.10.0	172.16.10.254	IP FW-P-02 VLAN 10
CARP Firewall	172.16.10.254	255.255.255.0	172.16.10.0	172.16.10.254	Passerelle du VLAN 10

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.10.100-150	172.16.10.254	172.16.30.10	172.16.30.20	Plage DHCP Client Paris

VLAN 20

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
B-P-WIFI	172.16.20.50	255.255.255.0	172.16.20.0	172.16.20.254	Administration borne Wifi
FW-P-02	172.16.20.252	255.255.255.0	172.16.20.0	172.16.20.254	IP FW-P-02 VLAN 20
FW-P-01	172.16.20.253	255.255.255.0	172.16.20.0	172.16.20.254	IP FW-P-01 VLAN 20
CARP Firewall	172.16.20.254	255.255.255.0	172.16.20.0	172.16.20.254	Passerelle du VLAN 20

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.20.100-150	172.16.20.254	172.16.30.10	172.16.30.20	Plage DHCP WIFI Employés

VLAN 21

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-02	172.16.21.252	255.255.255.0	172.16.21.0	172.16.21.254	IP FW-P-02 VLAN 21
FW-P-01	172.16.21.253	255.255.255.0	172.16.21.0	172.16.21.254	IP FW-P-01 VLAN 21
CARP Firewall	172.16.21.254	255.255.255.0	172.16.21.0	172.16.21.254	Passerelle du VLAN 21

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.21.100-150	172.16.21.254	172.16.30.10	172.16.30.20	Plage DHCP WIFI Invité

VLAN 30

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SRV-P-DC01	172.16.30.10	255.255.255.0	172.16.30.0	172.16.30.254	DC 1
SRV-P-DC02	172.16.30.20	255.255.255.0	172.16.30.0	172.16.30.254	DC 2
SRV-P-DFS01	172.16.30.50	255.255.255.0	172.16.30.0	172.16.30.254	DFS01
SRV-P-FOG01	172.16.30.11	255.255.255.0	172.16.30.0	172.16.30.254	Fog
SRV-P-OCS01	172.16.30.13	255.255.255.0	172.16.30.0	172.16.30.254	OCS Inventory
SRV-P-GLPI01	172.16.30.14	255.255.255.0	172.16.30.0	172.16.30.254	GLPI
SRV-P-BCK01	172.16.30.15	255.255.255.0	172.16.30.0	172.16.30.254	Veeam
SRV-P-CLOUD01	172.16.30.16	255.255.255.0	172.16.30.0	172.16.30.254	Nextcloud
SRV-P-RSAT-T0	172.16.30.30	255.255.255.0	172.16.30.0	172.16.30.254	RSAT T0
SRV-P-RSAT-T1	172.16.30.31	255.255.255.0	172.16.30.0	172.16.30.254	RSAT T1
SRV-P-RSAT-T2	172.16.30.32	255.255.255.0	172.16.30.0	172.16.30.254	RSAT T2
SRV-P-EDR01	172.16.30.19	255.255.255.0	172.16.30.0	172.16.30.254	EDR
SRV-P-ANS01	172.16.30.21	255.255.255.0	172.16.30.0	172.16.30.254	Ansible Lille
SRV-P-NETBOX01	172.16.30.22	255.255.255.0	172.16.30.0	172.16.30.254	Outil d'infrastructure
SRV-P-POL01	172.16.30.25	255.255.255.0	172.16.30.0	172.16.30.254	Centreon Poller
FW-P-02	172.16.30.252	255.255.255.0	172.16.30.0	172.16.30.254	IP FW-P-02 VLAN 30
FW-P-01	172.16.30.253	255.255.255.0	172.16.30.0	172.16.30.254	IP FW-P-01 VLAN 30
CARP Firewall	172.16.30.254	255.255.255.0	172.16.30.0	172.16.30.254	Passerelle du VLAN 30

VLAN 40

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-02	172.16.40.252	255.255.255.0	172.16.40.0	172.16.40.254	IP FW-P-02 VLAN 40
FW-P-01	172.16.40.253	255.255.255.0	172.16.40.0	172.16.40.254	IP FW-P-01 VLAN 40
CARP Firewall	172.16.40.254	255.255.255.0	172.16.40.0	172.16.40.254	Passerelle du VLAN 40

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.40.100-150	172.16.40.254	172.16.30.10	172.16.30.20	Plage DHCP Déploiement

VLAN 50

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SW-P-01	172.16.50.1	255.255.255.0	172.16.50.0	172.16.50.254	VLAN 50 Switch 1 Paris
SW-P-02	172.16.50.2	255.255.255.0	172.16.50.0	172.16.50.254	VLAN 50 Switch 2 Paris
SRV-P-ESXI01	172.16.50.10	255.255.255.0	172.16.50.0	172.16.50.254	IP d'administration hyperviseur
SRV-P-ESXI02	172.16.50.20	255.255.255.0	172.16.50.0	172.16.50.254	IP d'administration hyperviseur
PAW-P-T0	172.16.50.50	255.255.255.0	172.16.50.0	172.16.50.254	Machine d'administration
FW-P-02	172.16.50.252	255.255.255.0	172.16.50.0	172.16.50.254	IP FW-P-02 VLAN 50
FW-P-01	172.16.50.253	255.255.255.0	172.16.50.0	172.16.50.254	IP FW-P-01 VLAN 50
CARP Firewall	172.16.50.254	255.255.255.0	172.16.50.0	172.16.50.254	Passerelle du VLAN 50

VLAN 60

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-01	172.16.60.1	255.255.255.252	172.16.60.0	-	IP FW-P-01 VLAN 60
FW-P-02	172.16.60.2	255.255.255.252	172.16.60.0	-	IP FW-P-02 VLAN 60

VLAN 99

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SRV-P-HAProxy	172.16.99.10	255.255.255.0	172.16.99.0	172.16.99.254	HAProxy
FW-P-02	172.16.99.252	255.255.255.0	172.16.99.0	172.16.99.254	IP FW-P-02 VLAN 99
FW-P-01	172.16.99.253	255.255.255.0	172.16.99.0	172.16.99.254	IP FW-P-01 VLAN 99
CARP Firewall	172.16.99.254	255.255.255.0	172.16.99.0	172.16.99.254	Passerelle du VLAN 99

Marseille

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-M-01	172.17.10.254	255.255.255.0	172.17.10.0	172.17.10.254	IP FW-M-01 VLAN 10 Marseille
FW-M-01	10.44.110.112	255.255.255.0	10.44.110.0	10.44.110.254	IP WAN Marseille

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.17.10.100-150	172.17.10.254	172.16.30.10	172.16.30.20	Plage DHCP Client Marseille

Proxmox Matt (Lille)

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SRV-L-DC01	172.18.30.10	255.255.255.0	172.18.30.0	172.18.30.254	DC1 Lille
SRV-L-DC02	172.18.30.20	255.255.255.0	172.18.30.0	172.18.30.254	DC2 Core Lille
SRV-L-ANS01	172.18.30.15	255.255.255.0	172.18.30.0	172.18.30.254	Ansible Lille
SRV-L-NETBOX01	172.18.30.30	255.255.255.0	172.18.30.0	172.18.30.254	Netbox Infrastructure
FW-L-01	172.18.10.254	255.255.255.0	172.18.10.0	172.18.10.254	IP FW-L-01 LAN Lille
FW-L-02	172.18.10.252	255.255.255.0	172.18.10.0	172.18.10.254	IP FW-L-02 LAN Lille
CARP Firewall Lille	172.18.10.254	255.255.255.0	172.18.10.0	172.18.10.254	Passerelle du VLAN 10
FW-L-01	172.18.30.254	255.255.255.0	172.18.30.0	172.18.30.254	IP FW-L-01 SRV Lille
FW-L-02	172.18.30.252	255.255.255.0	172.18.30.0	172.18.30.254	IP FW-L-02 SRV Lille
CARP Firewall Lille	172.18.30.254	255.255.255.0	172.18.30.0	172.18.30.254	Passerelle du VLAN 30
FW-L-01	172.18.60.1	255.255.255.252	172.18.60.0	-	IP FW-L-01 SYNC_OPN
FW-L-02	172.18.60.2	255.255.255.252	172.18.60.0	-	IP FW-L-02 SYNC_OPN
FW-L-01	172.18.99.254	255.255.255.0	172.18.99.0	172.18.99.254	IP FW-L-01 DMZ Lille
FW-L-02	172.18.99.252	255.255.255.0	172.18.99.0	172.18.99.254	IP FW-L-02 DMZ Lille
CARP Firewall Lille	172.18.99.254	255.255.255.0	172.18.99.0	172.18.99.254	Passerelle du VLAN 99
FW-L-01	10.44.115.50	255.255.255.0	10.44.115.0	10.44.115.254	IP WAN Lille
FW-L-02	10.44.112.100	255.255.255.0	10.44.112.0	10.44.112.254	IP FW-L-02 WAN
CARP Firewall Lille	10.44.112.153	255.255.255.0	10.44.112.0	10.44.112.254	CARP WAN

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	SS	172.18.10.254	172.18.30.10	172.18.30.20	Plage DHCP Client Lille

Pour plus de détails, les deux schémas montrant l'ensemble de l'infrastructure ainsi que le plan d'adressage peuvent être trouvés en Annexe n°1 pour le schéma logique, en annexe n°2 pour le schéma physique et en annexe n°3 pour le plan d'adressage.

BTS Services informatiques aux organisations SESSION 2026
ANNEXE 10-A : Outil d'aide à l'appréciation de l'environnement technologique mobilisé par la personne candidate
Épreuve E6 - Administration des systèmes et des réseaux (option SISR)

CONTRÔLE DE L'ENVIRONNEMENT TECHNOLOGIQUE
En référence à l'annexe II.E – « Environnement technologique pour la certification » du référentiel du BTS SIO

Identification[5]	**Fab'Academy, 9 Rue de l'Halbrane, 44340 Bouguenais**	**SISR**

1. Environnement commun aux deux options
1.1 L'environnement technologique supportant le système d'information de l'organisation cliente comporte au moins :

Éléments	**Description de l'implantation dans le centre d'examen (nom du service ou de l'outil et caractéristiques techniques)**	**Remarques de la commission d'interrogation**
Un service d'authentification	**Active Directory Windows**	
Un SGBD	**MySQL / MariaDB**	
Un accès sécurisé à internet	**Firewall OPNsense, Stormshield**	
Un environnement de travail collaboratif	**DFS/DFSR (Partage de fichiers), Nextcloud**	
Deux serveurs, éventuellement virtualisés, basés sur des systèmes d'exploitation différents, dont l'un est un logiciel libre (*open source)*	**GLPI (Debian), Windows Server 2022**	

[5] Nom et adresse du centre d'examen ou identification de la personne candidate individuelle (numéro, nom, prénom)

ANNEXE 10-A (suite) : Modèle d'attestation de respect de l'annexe II.E – « Environnement technologique pour la certification » du référentiel Épreuve E6 - Administration des systèmes et des réseaux (option SISR)

Éléments	Description de l'implantation dans le centre d'examen (nom du service ou de l'outil et caractéristiques techniques)	Remarques de la commission d'interrogation
Une solution de sauvegarde	Veeam B&R	
Des ressources dont l'accès est sécurisé et soumis à habilitation	DFS/DFS/R (Partage de fichier), Nextcloud	
Deux types de terminaux dont un mobile (type *smartphone* ou encore tablette)	Tablette / PC Portable via connexion Wifi	

1.2 Des outils sont mobilisés pour la gestion de la sécurité :

Éléments	Description de l'implantation dans le centre d'examen (nom du service ou de l'outil et caractéristiques techniques)	Remarques de la commission d'interrogation
Gestion des incidents	GLPI	
Détection et prévention des intrusions	Wazuh, Stormshield	
Chiffrement	TLS, IPsec, SSH, PKI	
Analyse de trafic	Wireshark	

Rappel : les logiciels de simulation ou d'émulation sont utilisés en réponse à des besoins de l'organisation. Ils ne peuvent se substituer complètement à des équipements réels dans l'environnement technologique d'apprentissage.

ANNEXE 10-A (suite) : Modèle d'attestation de respect de l'annexe II.E « Environnement technologique pour la certification » du référentiel Épreuve E6 - Administration des systèmes et des réseaux (option SISR)

2. Éléments spécifiques à l'option « Solutions d'infrastructure, systèmes et réseaux » (SISR)

Rappel de l'annexe II.E du référentiel : « *Une solution d'infrastructure réduite à une simulation par un logiciel ne peut être acceptée.* »

2.1 L'environnement technologique supportant le système d'information de l'organisation cliente comporte au moins :

Éléments	Description de l'implantation dans le centre d'examen (nom du service ou de l'outil et caractéristiques techniques)	Remarques de la commission d'interrogation
Un réseau comportant plusieurs périmètres de sécurité	Segmentation VLANs via Switch	
Un service rendu à l'utilisateur final respectant un contrat de service comportant des contraintes en termes de sécurité et de haute disponibilité	Partage Réseau avec droits d'accès via DFS/R suivant méthode AGDLP	
Un logiciel d'analyse de trames	Wireshark	
Un logiciel de gestion des configurations	Ansible, GPO	
Une solution permettant l'administration à distance sécurisée de serveurs et de solutions techniques d'accès	SSH, RDP, HTTPS	
Une solution permettant la supervision de la qualité, de la sécurité et de la disponibilité des équipements d'interconnexion, serveurs, systèmes et services avec remontées d'alertes	Centreon	
Une solution garantissant des accès sécurisés à un service, internes au périmètre de sécurité de l'organisation (type intranet) ou externes (type internet ou extranet)	Firewall OPNsense, HaProxy, VPN	

Éléments	Description de l'implantation dans le centre d'examen (nom du service ou de l'outil et caractéristiques techniques)	Remarques de la commission d'interrogation
Une solution garantissant la continuité d'un service	**Veeam B&R, Haute disponibilité OPNsense**	
Une solution garantissant la tolérance de panne de systèmes serveurs ou d'éléments d'interconnexion	**RAID 1, redondance switch et Firewall OPNsense**	
Une solution permettant la répartition de charges entre services, serveurs ou éléments d'interconnexion	**DFS/R , DHCP, DNS, Firewall OPNsense**	

2.2 La structure et les activités de l'organisation s'appuient sur au moins une solution d'infrastructure opérationnelle parmi les suivantes :

Éléments	Description de l'implantation dans le centre d'examen (nom du service ou de l'outil et caractéristiques techniques)	Remarques de la commission d'interrogation
Une solution permettant la connexion sécurisée entre deux sites distants	**VPN IPsec**	
Une solution permettant le déploiement des solutions techniques d'accès	**FOG, Ansible**	
Une solution gérée à l'aide de procédures automatisées écrites avec un langage de *scripting*	**Ansible, Batch GPO**	
Une solution permettant la détection d'intrusions ou de comportements anormaux sur le réseau	**Stormshield IPS, Wazuh**	

Projet : Mise en place d'automatisation

a) Contexte

Dans le cadre de ma formation BTS SIO, j'ai eu l'opportunité d'intervenir sur un projet de mise en place d'infrastructure réseau et sécurité pour l'entreprise Oasis.

Ce projet consiste à accompagner Oasis dans la conception et le déploiement de son infrastructure informatique pour sa nouvelle agence, tout en assurant une communication sécurisée avec les autres sites. Pour répondre à ces besoins, j'ai été amené à mettre en place une solution d'automatisation de l'ensemble de l'infrastructure, Ansible afin de simplifier l'installation de nouveaux services et de standardiser l'ensemble des configurations réalisées.

b) Présentation de l'entreprise Oasis

L'entreprise Oasis est une société parisienne spécialisée dans la conception de voyages sur mesure pour une clientèle exigeante, à la recherche d'expériences uniques, loin des circuits touristiques classiques.

Créée en 2017, elle s'est rapidement imposée comme un acteur innovant dans le secteur du tourisme personnalisé, grâce à une approche centrée sur l'écoute client, la connaissance culturelle approfondie des destinations, et un réseau de partenaires locaux dans plus de 30 pays.

Après plusieurs années de forte croissance, Oasis a décidé d'ouvrir une nouvelle agence à Marseille, pour mieux couvrir le sud de la France et répondre à une demande croissante dans cette zone. L'agence parisienne reste le siège social et le cœur de la stratégie de conception et de relation client haut de gamme.

En 2024, Oasis a atteint un chiffre d'affaires de 2,3 millions d'euros, et ambitionne désormais de renforcer sa structure numérique afin d'améliorer la coordination entre les sites, la sécurité des données clients et la fluidité de l'expérience interne.

C'est dans ce contexte de croissance que NTxSystem a été sollicitée pour concevoir et déployer une infrastructure informatique adaptée aux besoins d'Oasis que ce soit pour l'agence parisienne, le siège social ou pour l'agence de Marseille.

c) Présentation de l'entreprise NTxSystem

NTxSystem est une entreprise prestataire spécialisée dans les solutions informatiques pour les professionnels. Dans le cadre de l'expansion d'Oasis, NTxSystem a été chargé de concevoir et déployer l'ensemble de l'infrastructure réseau des agences Paris et Marseille.

Les enjeux de ce projet sont multiples : centralisation des services, virtualisation des ressources, gestion des utilisateurs, sécurisation des communications inter-sites et mise en place d'un environnement stable et évolutif.

Pour répondre aux différentes exigences d'Oasis, l'ensemble de l'infrastructure est déployé dans un environnement virtualisé VMware ESXi.

d) Cahier des charges

L'entreprise Oasis a chargé NTxSystem de concevoir et de tester une infrastructure système et réseau virtualisée, capable de répondre aux besoins opérationnels et organisationnels de l'entité.

Les attentes techniques de la direction portent sur la centralisation des services, la virtualisation des ressources, la gestion des utilisateurs, la communication inter-sites, et la mise en place d'un environnement stable, évolutif et sécurisé, le tout dans un environnement de test isolé avant déploiement réel.

Dans ce cadre, j'ai eu la charge de mettre en place et d'automatiser le déploiement de configurations sur des machines Linux à l'aide d'Ansible, afin de garantir une gestion centralisée, reproductible et cohérente des serveurs, tout en appliquant des configurations standardisées et sécurisées selon les besoins d'Oasis.

e) Analyse du besoin

Avant de réaliser le projet, une veille sur les différentes solutions vues dans l'étude de faisabilité a été réalisée.

Pour cela, une analyse du besoin a été effectuée. Dans cette étude, trois solutions ont été comparées : Ansible, Terraform et Puppet en fonction de différents critères :

- Est-ce que les solutions sont toujours maintenu à jour ?
- Quelle est la difficulté de mise en place pour chaque solution ?
- Y'a-t-il de la documentation sur chaque solution ?
- Quel est la solution la plus adapté à notre petite infrastructure ?

Ces différents critères permettent de juger quel est la solution la plus adapté par rapport au besoin général et par rapport à l'infrastructure déjà mit en place.

f) Etude de solutions

Dans un premier temps, pour le premier critère par rapport à la maintenance de la solution, nous avons pu étudier si chaque solution était toujours maintenue à partir des sites officiels de chaque application et de GitHub.

Ensuite, pour le second critère, une évaluation de la difficulté de mise en place de chaque solution a été réalisée en fonction des différentes documentations.

Après, pour le troisième critère, la documentation de chaque solution a été évaluée en regardant à quel point chaque outil allait loin et à quels points ils pouvaient correspondre à nos besoins.

Enfin, pour le dernier critère, la solution la plus adaptée à l'infrastructure a été évaluée en fonction de la difficulté de mise en place de chaque outil et en fonction de la simplicité d'utilisation.

Les solutions	Solution 1	Solution 2	Solution 3
Intitulé	Ansible	Terraform	Puppet
Faisabilité technique *(Oui / Non, en précisant pourquoi)*	Ansible est une solution d'automatisation et de gestion de configuration, utilisée pour déployer des serveurs, installer des logiciels et appliquer des configurations sur plusieurs machines.	Terraform est une solution d'Infrastructure qui permet de créer et gérer automatiquement des infrastructures à partir de fichiers de configuration.	Puppet est une solution de gestion de configuration permettant d'automatiser la configuration et l'administration des serveurs à grande échelle.
Besoins RH *(Internes et/ou Externes)*	Interne	Interne	Interne
Besoin Matériel et Immatériel	Serveur Physique, machine virtuelle	Serveur Physique, machine virtuelle	Serveur Physique, Puppet Master + Agents su les machines
Coût total estimé	0	0	0
Temps Jours / Hommes	1	3	3
Durée de réalisation estimée	5h	7h	7h
Points forts **Points faibles**	• PF: Facile à utiliser, pas d'agents sur les machines (connexion SSH), automatisation facile et de nombreux plugins • Pf: Moins adaptés aux grosses infrastructures s'il y a beaucoup de machines à gérer, notamment au niveau de l'inventaire	• PF: Excellent pour créer une infrastructure complète et créer des machines de façon basique • Pf: Pas fait pour configurer les machines en détail, peut-être complexe à utiliser	• PF: Permet de gérer des machines sur de grandes infrastructures de façon stable, gestion centralisée • Pf: Plus complexe à mettre en place, nécessite d'avoir un agent sur les machines, plus lourd en termes d'administration

On peut voir dans le tableau ci-dessus que les besoins matériels et immatériels sont globalement similaires pour chaque solution. Uniquement Puppet nécessite davantage de ressources, notamment car il fonctionne avec un serveur et des agents sur les machines.

Pour le temps de mise en place (jours/hommes) et la durée de réalisation estimée, Terraform et Puppet sont plus complexes à installer et à prendre en main qu'Ansible.

Enfin, au niveau des points forts et des points faibles :

- Ansible est simple et rapide à utiliser, mais il est moins adapté aux très grandes infrastructures.
- Terraform permet de créer facilement des ressources et des machines, mais il ne sert pas à configurer les machines en détail.
- Puppet permet une gestion complète et centralisée des machines, mais sa mise en place et son utilisation sont plus complexes.

Grâce à ces différentes comparaisons et à la veille réalisée sur les documentations de chaque solution, il a été conclu que la solution Ansible correspondait le plus à nos besoins car celle-ci est gratuite, non gourmande en ressources et elle est facile à mettre en place et à utiliser.

g) Planification

Le projet pour la mise en place de la solution Ansible a été séparé en diverses étapes, ces étapes sont détaillées dans les différentes parties.

1) Discussion de la problématique

Cette partie correspond à l'étude de la mission donnée par Oasis afin de comprendre comment gérer de façon sécurisée les différentes machines Linux de l'infrastructure.

2) Réalisation de l'étude de solutions

Dans cette partie, j'ai pu étudier les différents besoins de l'automatisation afin de voir quelles étaient les solutions à ma disposition pour répondre au cahier des charges.

3) Réalisation du GANTT

Pour cette étape, j'ai pu lister les différentes tâches pour la mise en place de l'automatisation avec Ansible et j'ai pu formuler l'ensemble des tâches sous la forme d'un GANTT afin de savoir tout ce qu'il y a à faire.

4) Installation de la machine virtuelle

L'installation correspond à la création de la machine virtuelle, l'ajout des différentes ressources pour celle-ci et la configuration de la machine Debian de base.

5) Installation Ansible

L'installation d'Ansible correspond à l'ensemble des processus qu'il faut mettre en place pour que l'outil fonctionne correctement.

6) Mise en place inventaire dynamique NetBox

La mise en place de l'inventaire dynamique correspond aux différentes configurations nécessaires afin qu'Ansible puisse récupérer automatiquement l'inventaire des machines depuis NetBox.

7) Mise en place tâche automatisée avec GitLab et NetBox

La mise en place de tâche automatisée avec Gitlab et NetBox correspond à la configuration de déclencheur d'événement afin de pouvoir exécuter des Playbooks Ansible directement depuis l'interface de NetBox.

8) Test lancement de playbooks

Ce premier test correspond au fait d'exécuter des playbooks sur des machines ciblées afin de voir s'ils sont corrects.

9) Test de l'inventaire dynamique

Ce second test est le fait de lancer des playbooks via l'inventaire dynamique NetBox afin de voir si Ansible peut correctement aller chercher les informations se trouvant dans l'application.

10) Test de playbooks via NetBox

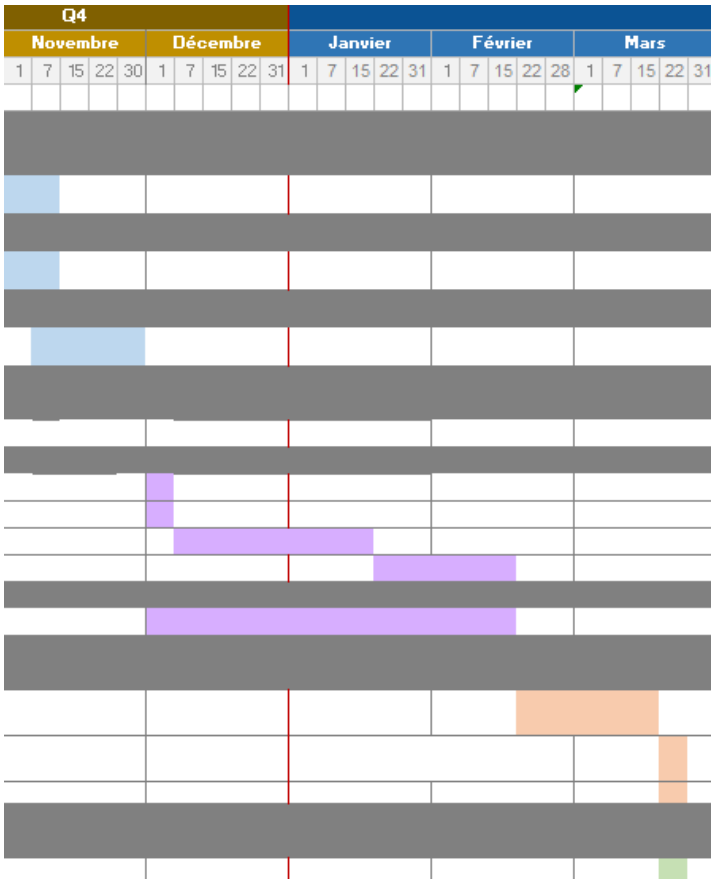
Ce dernier test correspond au lancement de playbook Ansible via l'interface de NetBox.

11) Clôture du projet

Pour terminer, la clôture du projet correspond à la fin du projet, cela permet de vérifier que le projet a bien répondu au cahier des charges.

Ci-dessous, les tâches du diagramme de Gantt ainsi que le diagramme correspondant. Celui-ci permet de voir et de comprendre l'ensemble des tâches du projet. On peut également voir à qui elles sont attribuées et le temps de réalisation de ces différentes tâches.

LOT	INTITULE DE LA PHASE	INTERVENANT(S)	TEMPS J/H	DATE DE DÉBUT	DEADLINE	DATE DE FIN RÉELLE
Phase 1 : Initialisation			**3h30**	**03-nov.-25**	**28-nov.-25**	**28-nov.-25**
	Phase 1 : Initialisation					
	Discussion de la problématique	NTxSystem	1h	3-nov.-25	7-nov.-25	7-nov.-25
	Phase 2 : Faisabilité					
	Réalisation de l'étude de solutions	Matt MOREAU	1h30	3-nov.-25	7-nov.-25	7-nov.-25
	Phase 3 : Etude et Cadrage					
	Réalisation du GANTT	Matt MOREAU	1h	24-nov.-25	28-nov.-25	28-nov.-25
Phase 2 : Réalisation, Déploiement & Recette			**8h30**	**1-déc.-2025**	**20-févr.-2026**	**20-févr.-2026**
	Phase 4 : Matériel					
	-	-	-	-	-	-
	Phas 5 : Installation et Configuration solution					
	Installation Machine Virtuelle	Matt MOREAU	30min	1-déc.-2025	5-déc.-2025	5-déc.-2025
	Installation Ansible	Matt MOREAU	30min	1-déc.-2025	5-déc.-2025	5-déc.-2025
	Mise en place inventaire dynamique NetBox	Matt MOREAU	4h	5-déc.-2026	20-janv.-2026	20-janv.-2026
	Mise en place tâche automatisée avec GitLab et NetBox	Matt MOREAU	3h30	20-janv.-2026	20-févr.-2026	20-févr.-2026
	Phase 6 : Documentation					
	Réalisation documentation complète de la solution	Matt MOREAU	4h	1-déc.-2025	20-févr.-2026	20-févr.-2026
Phase 3 : Exploitation			**40min**	**20-févr.-2026**	**20-mars-2026**	**20-mars-2026**
	Phase 7 : Test de la solution					
	Test lancement de playbooks	Matt MOREAU	10min	20-févr.-2026	16-mars-2026	16-mars-2026
	Test de l'inventaire dynamique	Matt MOREAU	30min	16-mars-2026	20-mars-2026	20-mars-2026
	Test de playbooks via NetBox	Matt MOREAU	30min	16-mars-2026	20-mars-2026	20-mars-2026
Phase 4 : Clôture du Projet			**10min**	**20-mars-26**	**20-mars-26**	**20-mars-26**
	Phase 8 : Clôture du Projet					
	Clôture du Projet	Matt MOREAU	10min	20-mars-2026	20-mars-2026	20-mars-2026



h) Définitions et abréviations

Playbook : Un playbook Ansible est un fichier en YAML qui permet de décrire un ensemble de tâches à exécuter sur un ou plusieurs hôtes

Rôle : Un rôle Ansible est une structure organisée qui regroupe des tâches (ainsi que des variables, handlers, templates, fichiers, etc.) afin de les réutiliser facilement dans plusieurs playbooks.

Ansible lint : Outil permettant de vérifier la syntaxe des playbooks, des rôles et des différents fichiers de configuration Ansible

Webhook NetBox : Un Webhook est une communication qui transmet de façon automatique des données entre les applications via HTTP

Pipeline Trigger token : Un pipeline trigger token est un jeton secret qui sert à déclencher un pipeline automatiquement via une requête API

Pipeline CI / CD : Un pipeline CI/CD est une suite d'étapes permettant d'automatiser certains processus tels que la création et le déploiement

Gitlab Runner : Le Gitlab Runner est le processus qui exécute les jobs CI/CD, sans lui les pipelines ne peuvent pas être exécutés

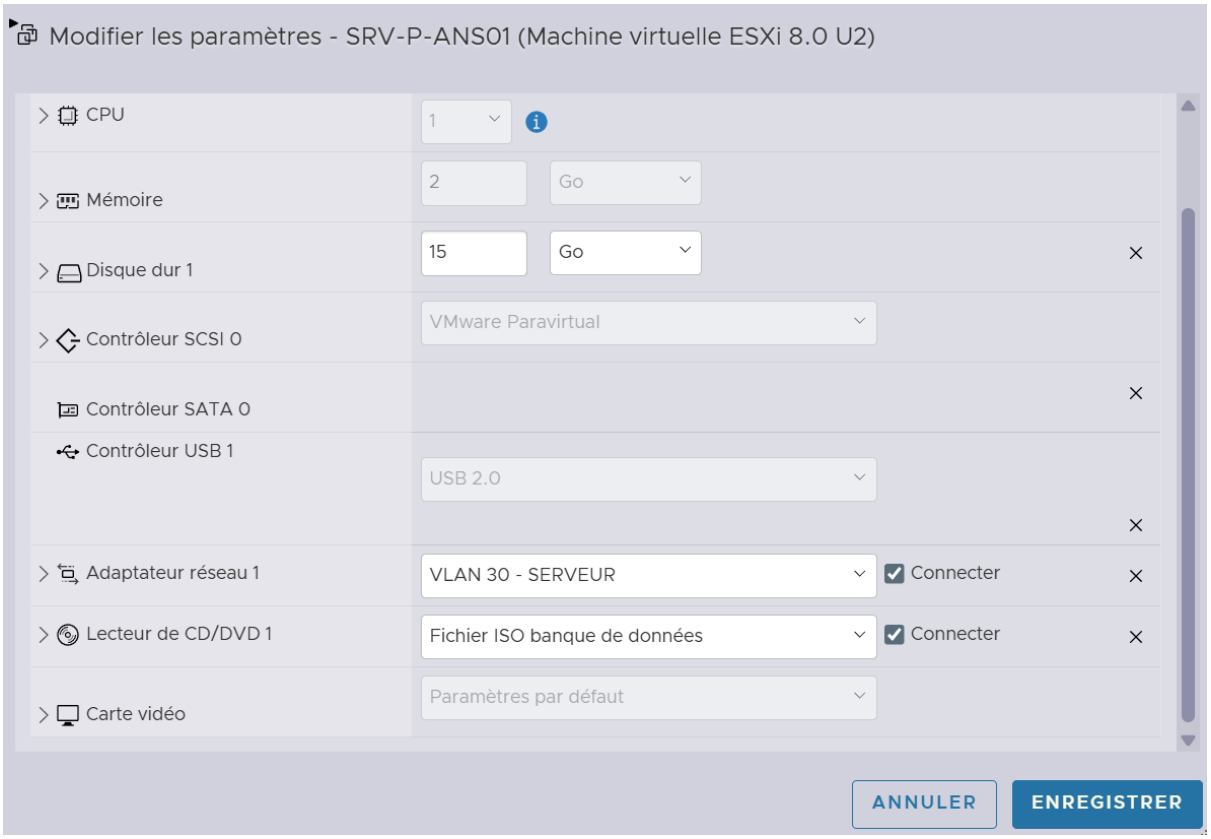
i) Liste du matériel à disposition

- 2 Hyperviseurs VMWare
- 2 Switch
- Câbles Réseaux
- Câbles d'alimentation

j) Installation et Configuration Ansible

1) Installation de la machine virtuelle

Pour l'installation de la machine virtuelle Ansible, j'ai pu créer une machine virtuelle sur notre ESXI02. J'ai pu mettre 15 Go de stockage pour cette machine, 2 Go de RAM et une carte réseau sur le VLAN 30 qui correspond à notre VLAN pour les serveurs. J'ai également pu ajouter l'ISO correspondant à Debian 12. Pour cette machine, il y a besoin de peu de ressources car elle sert uniquement pour le déploiement.



2) Installation Ansible

Une fois Debian installée sur la machine virtuelle, il faut tout d'abord mettre les paquets à jour comme ci-dessous.

```
apt update && apt upgrade -y
```

```
root@SRV-P-ANS01:~/ansible-oasis# apt update && apt upgrade -y
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian bookworm-updates InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@SRV-P-ANS01:~/ansible-oasis#
```

Ensuite, pour utiliser Ansible, il faut installer le paquet correspondant. Le paquet Git est également installé, car il servira à interagir avec GitLab pour la mise en place de la structure Ansible.

```
apt install ansible git -y
```

```
root@SRV-P-ANS01:~# apt install ansible git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  cowsay sshpass git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  ansible git
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 24.2 MB of archives.
After this operation, 311 MB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 ansible all 7.7.0+dfsg-3+deb12u1 [16.9 MB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 git amd64 1:2.39.5-0+deb12u3 [7264 kB]
Fetched 24.2 MB in 17s (1428 kB/s)
Selecting previously unselected package ansible.
(Reading database ... 23726 files and directories currently installed.)
Preparing to unpack .../ansible_7.7.0+dfsg-3+deb12u1_all.deb ...
Unpacking ansible (7.7.0+dfsg-3+deb12u1) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.39.5-0+deb12u3_amd64.deb ...
Unpacking git (1:2.39.5-0+deb12u3) ...
Setting up git (1:2.39.5-0+deb12u3) ...
Setting up ansible (7.7.0+dfsg-3+deb12u1) ...

Progress: [ 78%] [#########################################################################################.....................]
```

Après, il est possible de vérifier que le paquet Ansible soit bien installé en vérifiant la version du paquet.

```
ansible --version
```

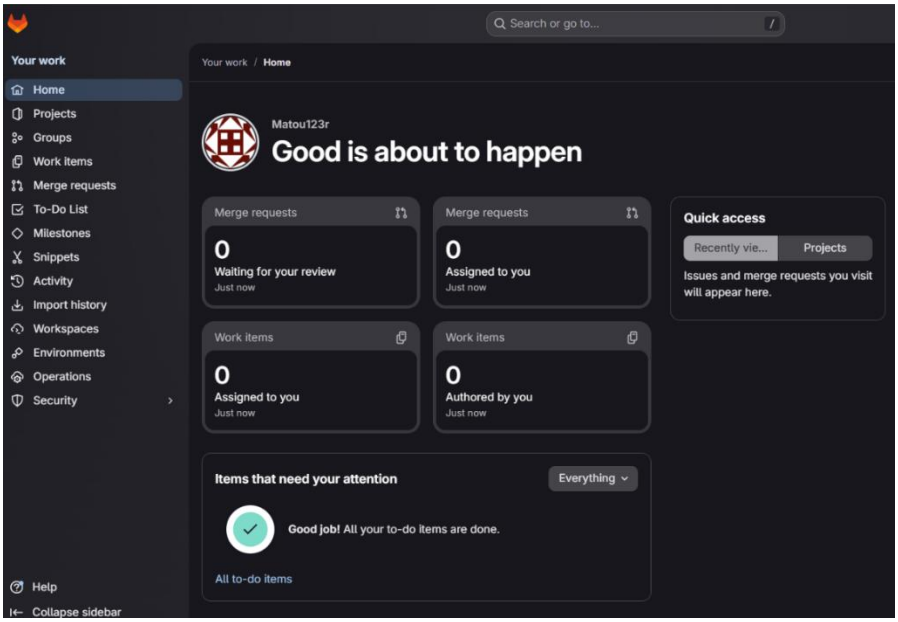
```
root@SRV-P-ANS01:~# ansible --version
ansible [core 2.14.18]
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.11.2 (main, Apr 28 2025, 14:11:48) [GCC 12.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

3) Configuration structure Ansible

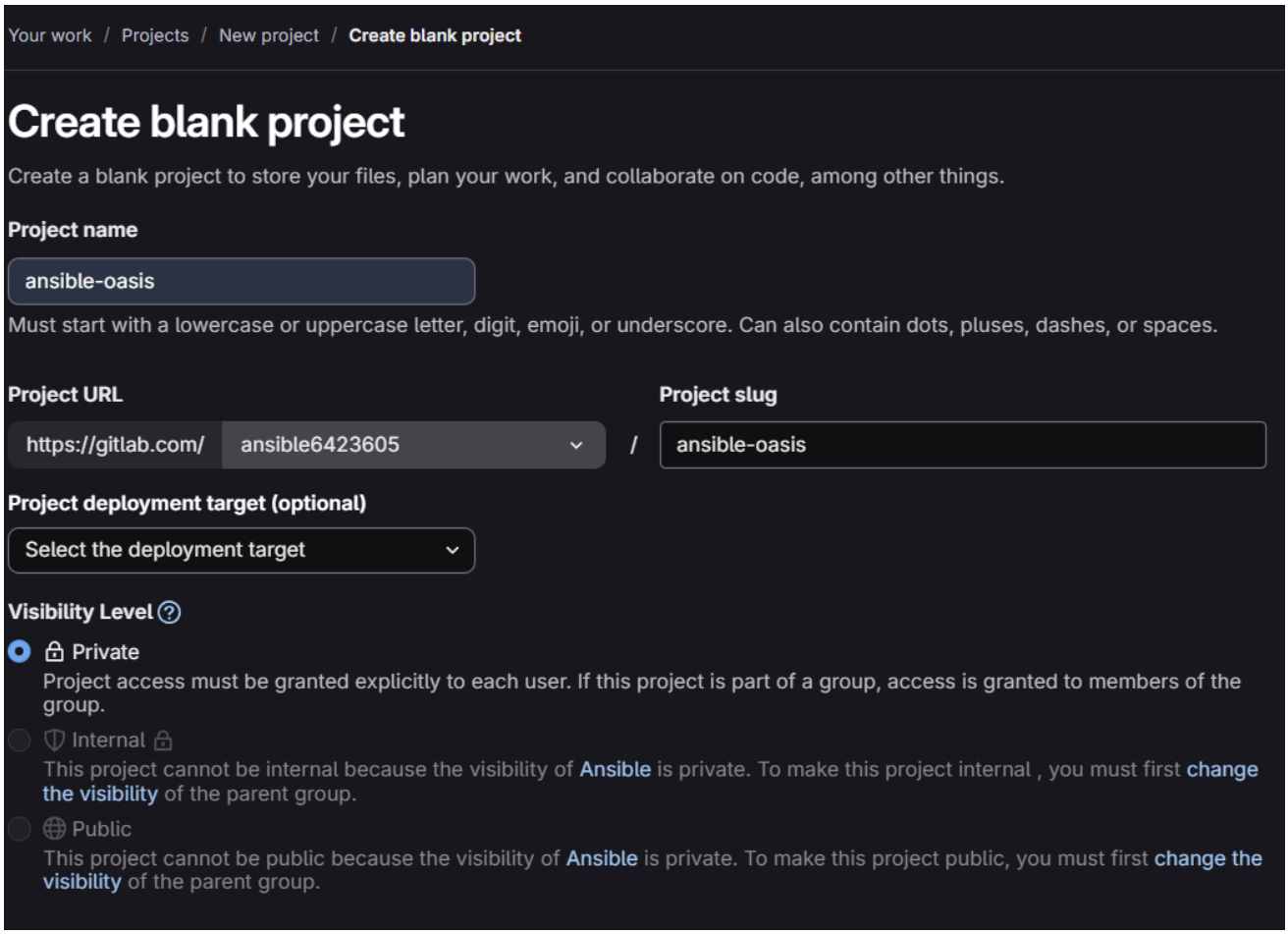
Une fois l'installation d'Ansible effectuée, il est possible de réaliser la configuration de la structure pour le fonctionnement de la solution. Pour cela, GitLab est utilisé afin d'avoir l'ensemble du code gérable en ligne de façon simple et de disposer d'un système de versionnement grâce à Git. GitLab est accessible via l'URL ci-dessous ; la solution n'a pas été installée sur l'infrastructure interne pour des raisons de simplicité, au vu de la capacité de celle-ci.

https://gitlab.com/

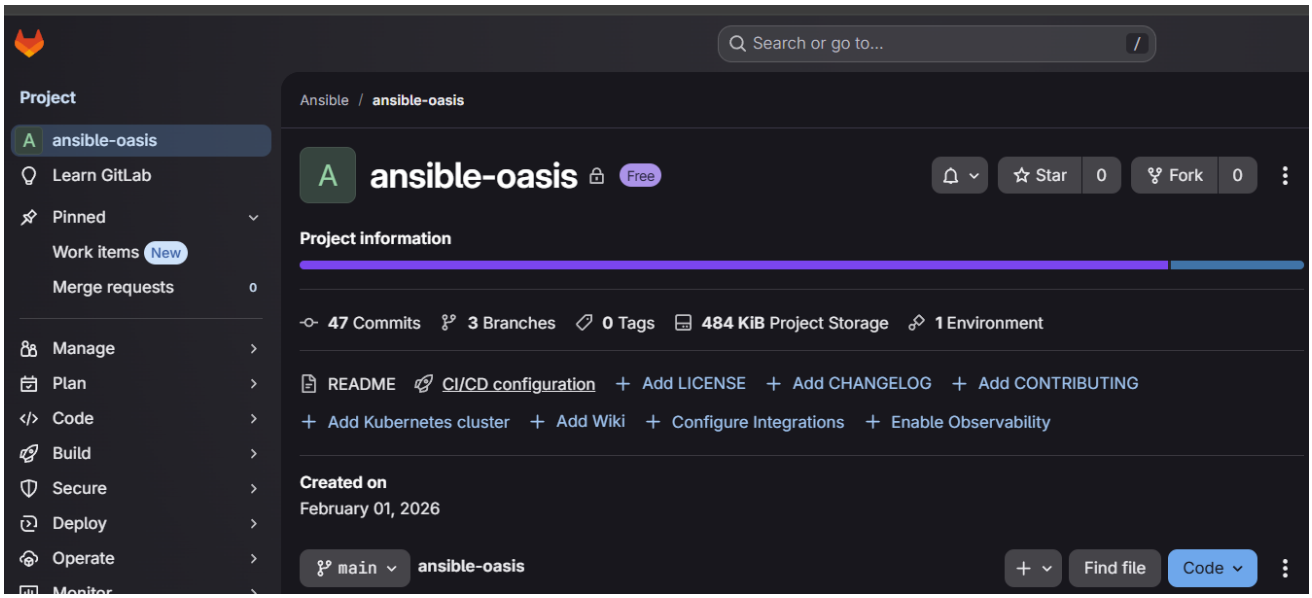
Pour cela, il suffit tout d'abord d'aller sur GitLab.



Ensuite, il faut créer un projet afin d'accueillir l'ensemble de la configuration Ansible en allant dans l'onglet « Projects ». Dans la page de création, il faut renseigner le nom du projet, l'URL ainsi que le niveau de visibilité. Ici, la visibilité est définie en interne car l'outil est uniquement utilisé par les administrateurs de l'entreprise Oasis, personne d'autre ne doit y avoir accès.



Lorsque le projet est créé, cette page s'affiche, cela veut dire que le projet est bien créé et qu'il peut maintenant être édité.



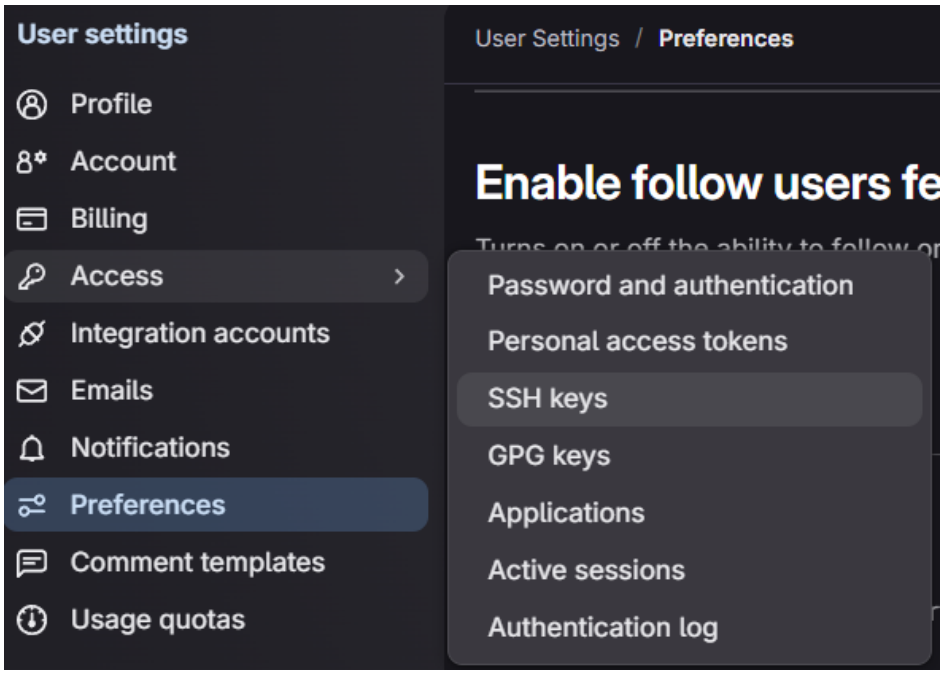
Afin de gérer la structure du projet depuis la machine Ansible créée précédemment, il faut cloner le projet. Pour cela, il faut créer une clé SSH sur la machine dédiée à Ansible. Ici, une clé ED25519 a été créée car c'est le type de clé le plus sécurisé à l'heure actuelle. Nous pouvons voir ci-dessous la création de la clé SSH.

```
root@SRV-P-ANS01:~# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:9IOAnbi7i/KF6VTpB2rWtdqqOaNkMDvP1LYztVCFnEk root@SRV-P-ANS01
The key's randomart image is:
+--[ED25519 256]--+
|     oE+         |
|     +=..        |
|    o +..        |
|     o.o o       |
|o   =.. S o      |
|.o O.+..   .     |
|ooO Boo.         |
|+Xo=o*.          |
|.+O+*=.          |
+----[SHA256]-----+
```

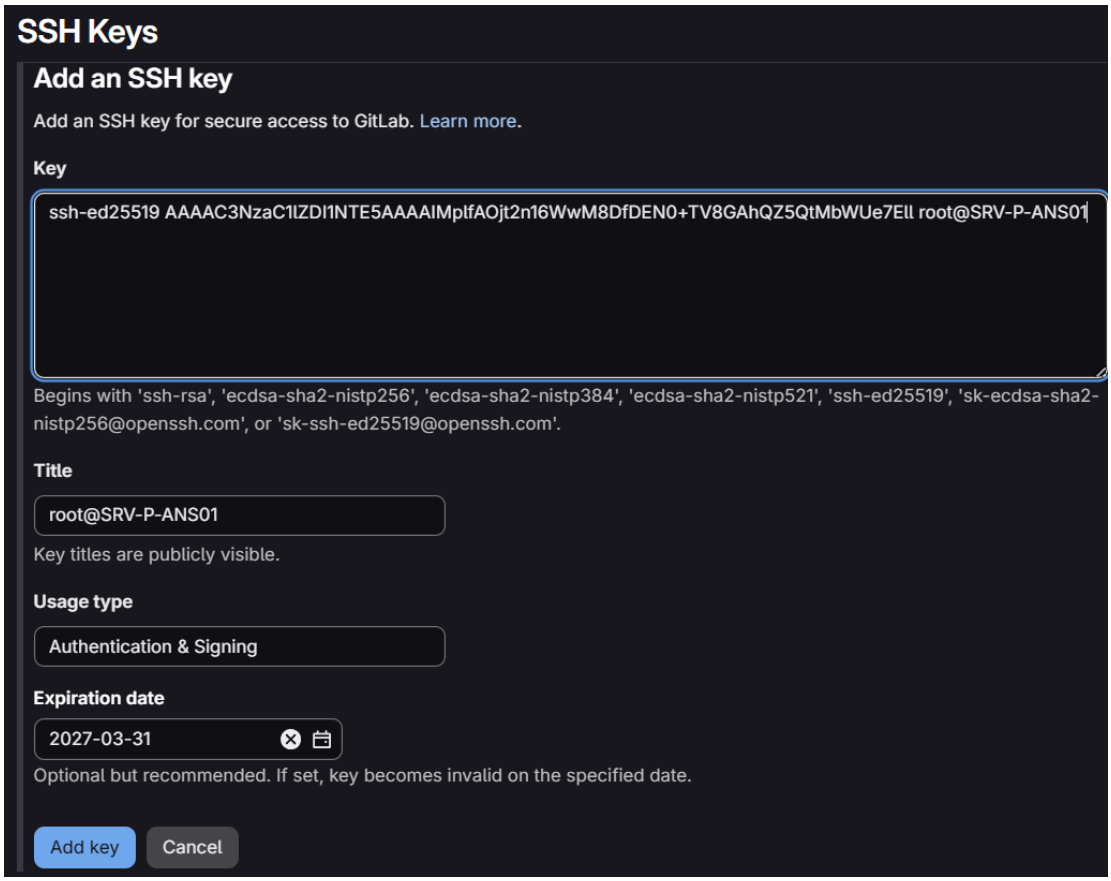
Une fois la clé créée, il est possible de vérifier que celle-ci existe bien dans le dossier de l'utilisateur correspondant, comme ci-dessous pour la clé publique.

```
root@SRV-P-ANS01:~# cat .ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMplfAOjt2n16WwM8DfDEN0+TV8GAhQZ5QtMbWUe7Ell root@SRV-P-ANS01
```

Pour cloner et donc récupérer le projet Gitlab créé précédemment, il faut ajouter la clé SSH au profil Gitlab connecté afin que celle-ci y soit liée, pour cela, il faut aller dans « Preferences », « SSH keys ».



Puis il faut ajouter la clé publique qui a été générée précédemment comme ci-dessous.

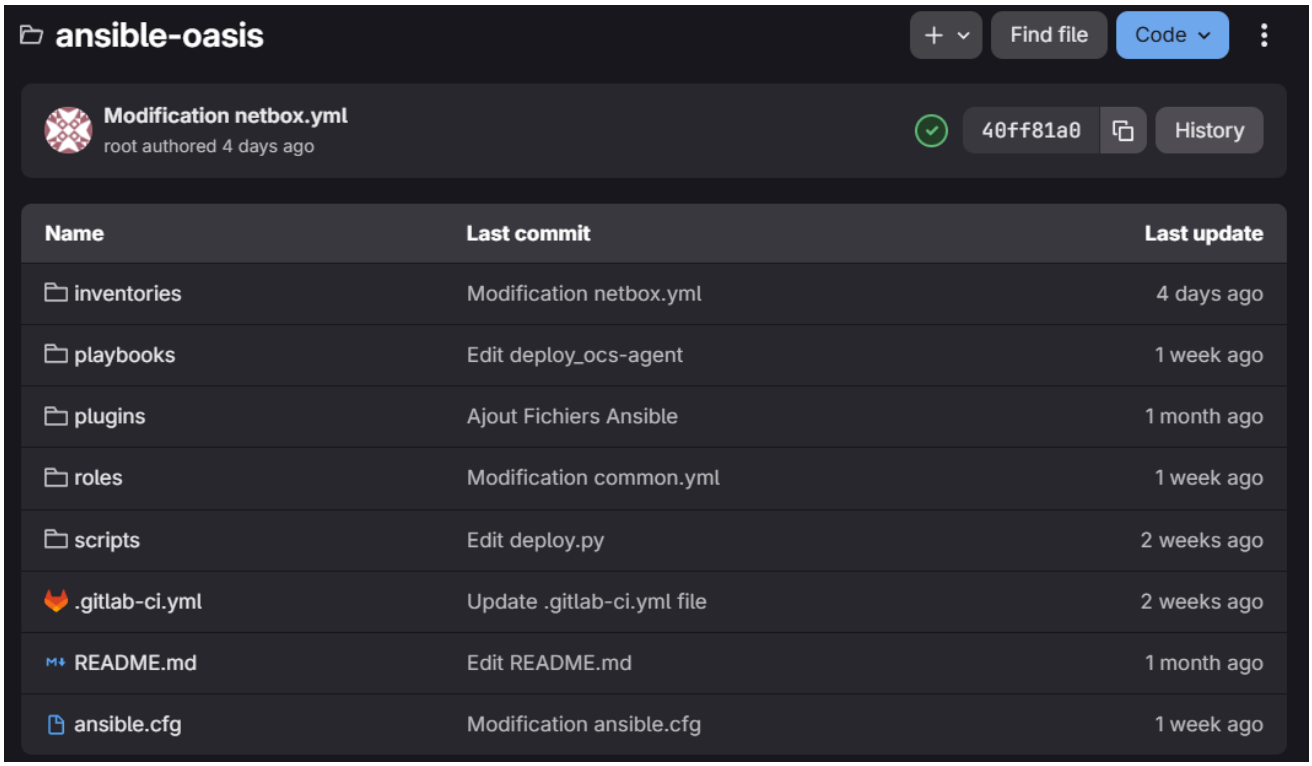


Après avoir ajouté la clé sur Gitlab, il est possible de cloner le dépôt ansible-oasis créé précédemment, comme ci-dessous.

```
root@SRV-P-ANS01:~# git clone git@gitlab.com:ansible6423605/ansible-oasis.git
Cloning into 'ansible-oasis'...
remote: Enumerating objects: 296, done.
remote: Counting objects: 100% (193/193), done.
remote: Compressing objects: 100% (168/168), done.
remote: Total 296 (delta 74), reused 0 (delta 0), pack-reused 103 (from 1)
Receiving objects: 100% (296/296), 39.49 KiB | 13.16 MiB/s, done.
Resolving deltas: 100% (88/88), done.
root@SRV-P-ANS01:~#
```

Une fois le dépôt cloné, il est possible de créer les dossiers nécessaires à la mise en place de la structure Ansible.

La structure Ansible comporte quelques dossiers.



Dans celle-ci, il y a tout d'abord, le dossier « inventories », dans ce dossier, il est possible de retrouver la liste des hôtes / serveurs gérés par Ansible.

Ensuite, il y a le dossier « playbooks », ce dossier décrit une suite de tâches ou de rôles écrits dans un fichier au format YAML.

Après, il y a le dossier plugins, celui-ci contient des extensions personnalisées pour Ansible, il permet d'étendre les fonctionnalités natives de Ansible.

Puis, la partie rôles, dans ce dossier de configuration, il y a des tâches écrites en YAML qui peuvent être réutilisées dans les différents playbooks.

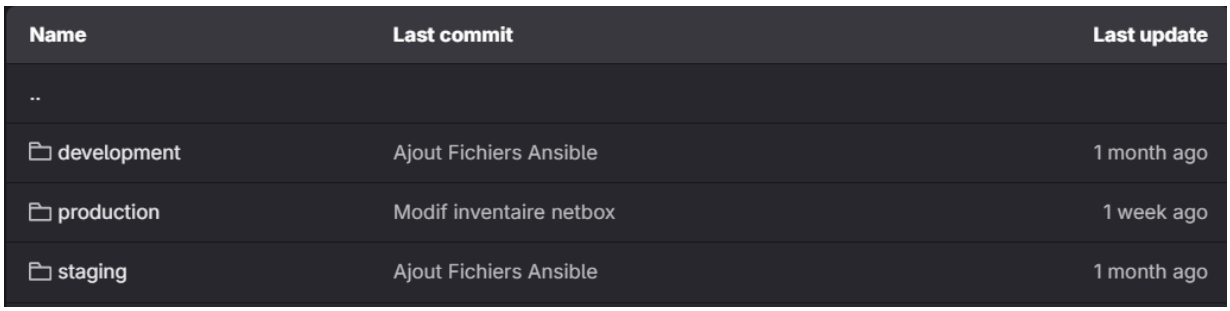
Ensuite, le dossier scripts, ce dossier contient des scripts utilitaires complémentaires à Ansible.

Enfin, le fichier ansible.cfg, ce fichier est le fichier de configuration global pour le projet Ansible.

4) Configuration Inventories

Pour la configuration des inventaires Netbox, les inventaires ont été séparés en trois catégories comme nous pouvons le voir ci-dessous.

- Development : Utilisé pour les développeurs, permet de contenir des machines dédiées au développement
- Production : Utilisé directement pour les machines Linux en production
- Staging : Utilisé pour la pré-production, c'est le miroir de la production



Pour la construction d'un inventaire statique, les hôtes sont organisés en groupes. Cela permet ensuite de lancer les rôles Ansible sur le groupe voulu. Ci-dessous la configuration du fichier « host.yml » se trouvant dans le dossier production.

```yaml
---
all:
  children:

    monitoring:
      hosts:
        SRV-P-SUP01:
          ansible_host: centreon.oasis.local
        SRV-P-EDR01:
          ansible_host: wazuh.oasis.local

    gestion:
      hosts:
        SRV-P-GLPI01:
          ansible_host: glpi.oasis.local
        SRV-P-OCS01:
          ansible_host: ocs.oasis.local
        SRV-P-FOG01:
          ansible_host: fog.oasis.local

    cloud:
      hosts:
        SRV-P-CLOUD01:
          ansible_host: nextcloud.oasis.local
```

Dans l'inventaire de production, il y a également un dossier « group_vars », dans ce dossier sont stockés des fichiers permettant de définir des variables globales par rapport aux groupes définis dans le host.yml.

5) Configuration Ansible.cfg

Le fichier ansible.cfg est le fichier de configuration central d'Ansible. Il définit le comportement par défaut de l'outil lors de l'exécution des playbooks. Il est lu automatiquement par Ansible.

Dans celui-ci, il y a différents paramètres :

ansible_managed : Permet de définir un message précis lorsque la variable ansible_managed est utilisée

remote_user : Définit l'utilisateur SSH utilisé pour se connecter aux machines distantes

roles_path : Indique le répertoire où Ansible recherche les rôles à exécuter

retry_files_enabled : Désactive la génération de fichiers .retry après un échec

timeout : Délai maximum pour établir une connexion SSH

forks : Définit le nombre de machines traitées en même temps

log_path : Chemin du fichier de log

pipelining : Réduit le nombre de connexions SSH par tâches

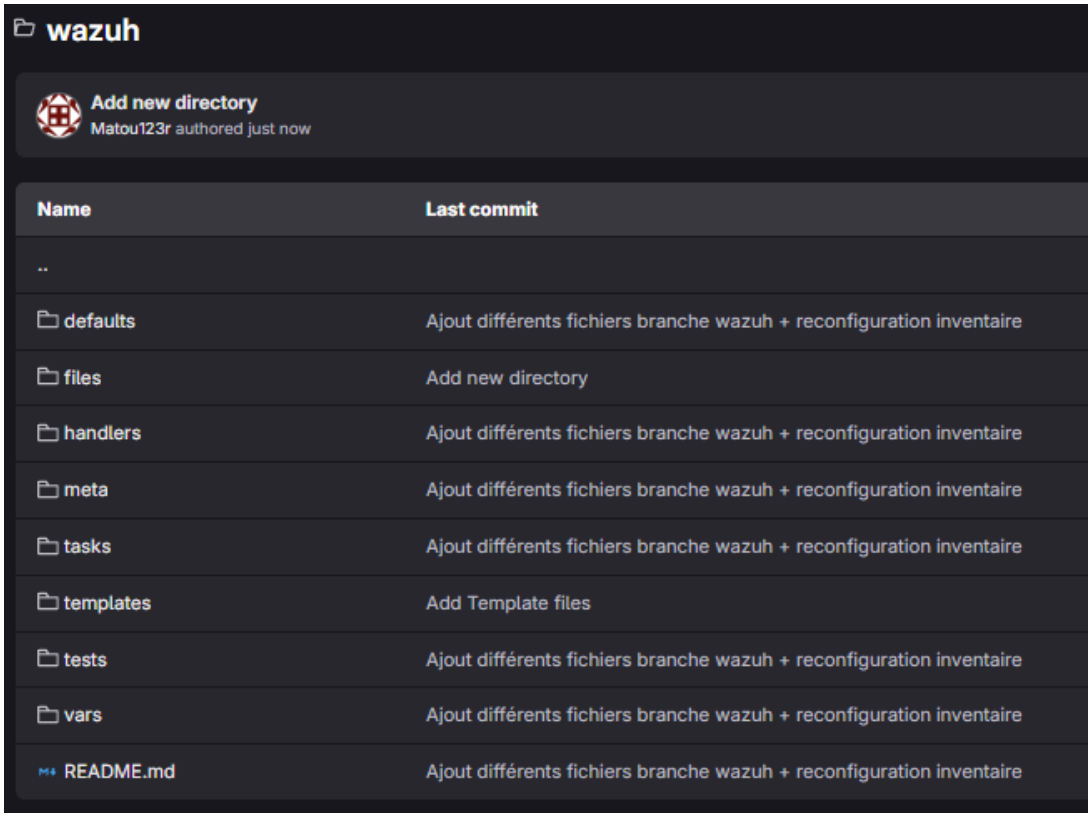
ssh_args : Permet de réutiliser une seule connexion SSH pour plusieurs tâches

```
[defaults]
# Configuration ansible managed
ansible_managed = Ansible managed file, do not edit directly
# Define distant user
remote_user = root
# Define the path for the roles
roles_path = roles
# Desactive the creation of .retry files
retry_files_enabled = False
# Define the maximum time for a ssh connection
timeout = 30
# Numbers of machines used in the same time
forks = 20
# File for logs
log_path = /var/log/ansible/ansible.log

[ssh_connection]
# Reduce the number of ssh  connection
pipelining = True
# Reuse the same ssh connection for some tasks
ssh_args = -o ControlMaster=auto -o ControlPersist=60s
```

Dans cette configuration ansible.cfg, l'inventaire à utiliser n'est pas défini directement car les trois inventaires définis dans le dossier « inventories » peuvent être utilisés selon les situations.

6) Création de rôles

Dans le dossier rôles, il est possible de retrouver les rôles réutilisables dans les différents playbooks. Dans un rôle, par exemple le rôle Wazuh permettant d'installer et de configurer l'agent Wazuh sur des clients Debian, il y a d'autres dossiers permettant de définir les différents paramètres.

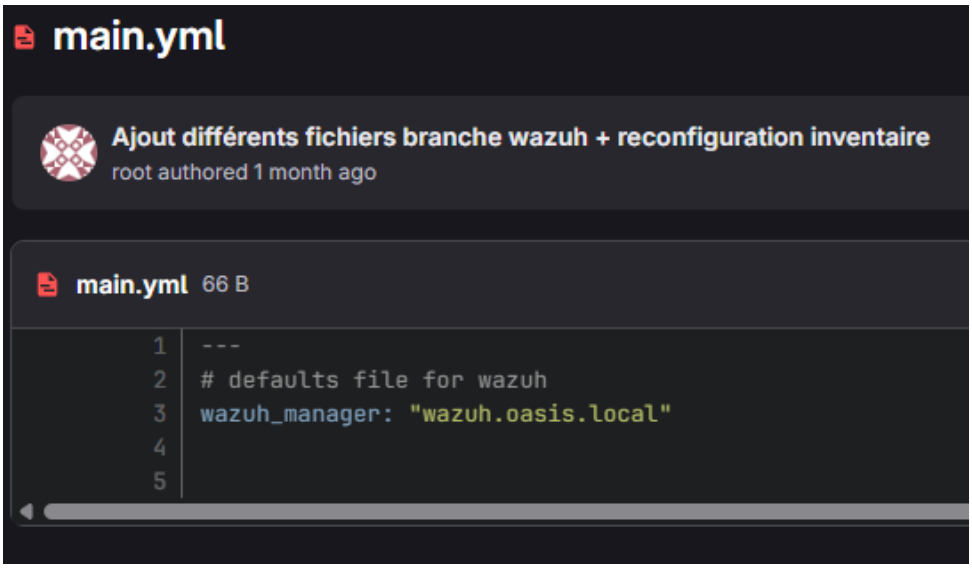


Les différents dossiers du rôle sont expliqués dans le tableau ci-dessous.

Dossier	Rôle
defaults/	Contient les variables du rôle, régulièrement les variables qui sont destinées à être modifiées
files/	Contient les fichiers statiques à copier tels quels sur les machines distantes, sans aucune modification de contenu
handlers/	Contient les actions déclenchées par une notification, par exemple le démarrage ou l'arrêt d'un service
meta/	Contient les métadonnées du rôle et l'importation de potentiel rôle
tasks/	Contient la liste des tâches à exécuter sur un hôte distant
templates/	Contient les fichiers de configuration dynamiques au format Jinja2
tests/	Contient un playbook minimum de test permettant de vérifier le bon fonctionnement du rôle de manière isolée
vars/	Contient les variables spécifiques au rôle. Ces valeurs ne sont pas destinées à être modifiées par l'utilisateur du rôle

Dans ce rôle Wazuh, il est possible de retrouver différents fichiers. Tout d'abord, dans le dossier defaults, il y a le fichier main.yml.

Dans ce fichier sont configurées les variables qui peuvent être utilisées par l'ensemble du rôle Wazuh, ici il y a seulement une variable, « wazuh_manager » qui permet de définir le serveur Wazuh pour les différents agents.



Dans le dossier handlers, il est également possible de retrouver un fichier main.yml. Celui-ci contient des tâches déclenchées conditionnellement, ici le redémarrage du service wazuh-agent. Le handler peut être appelé depuis n'importe quelle tâche via la directive « notify ».

main.yml

Ajout différents fichiers branche wazuh
root authored 1 month ago

main.yml 104 B

```yaml
---
# Handlers Wazuh
- name: restart wazuh-agent
  systemd:
    name: wazuh-agent
    state: restarted


```

Ensuite, dans le dossier meta, il est également possible de retrouver un fichier main.yml. Celui-ci contient les métadonnées du rôle Wazuh, comme l'auteur, le nom de l'entreprise, la version minimale d'Ansible requise ainsi que les plateformes destinées à recevoir ce rôle. Il est également possible de déclarer des dépendances vers d'autres rôles, mais cela n'est pas nécessaire ici.

main.yml

Add meta/main.yml
root authored 57 seconds ago

main.yml 229 B

```yaml
galaxy_info:
  author: Matt MOREAU
  description: Install and configure Wazuh agent
  company: Oasis
  license: None
  min_ansible_version: "2.1"

  platforms:
    - name: Debian
      versions:
        - all


  galaxy_tags: []

```

Après, dans le dossier templates, pour ce rôle Wazuh, il y a un template nommé ossec.conf.j2.

Ce template correspond au fichier de configuration utilisé par le service wazuh-agent.

Dans ce fichier, il est indiqué qu'il a été configuré par Ansible grâce à ansible_managed, qui permet d'afficher une phrase précise (comme défini dans le fichier ansible.cfg).

Il contient également la version de Debian via la variable ansible_distribution_version, qui est une variable par défaut d'Ansible.

Enfin, l'adresse du serveur Wazuh est définie grâce à la variable wazuh_manager, présente dans le fichier situé dans le dossier defaults.

ossec.conf.j2 5.52 KiB

```
# {{ ansible_managed }}
<!--
  Wazuh - Agent - Default configuration for debian {{ ansible_distribution_version }}
  More info at: https://documentation.wazuh.com
  Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <client>
    <server>
      <address>{{ wazuh_manager }}</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>debian, debian12</config-profile>
    <notify_time>20</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
  </client>
```

Dans chaque dossier, il est possible de retrouver différents fichiers.

Tout d'abord, dans le dossier tasks, il est possible de retrouver un fichier main.yml dans lequel sont définies les différentes tâches qui devront être utilisées lorsque l'on va lancer le rôle.

Dans un premier temps, il y a l'installation des paquets « gnupg » et « apt-transport-https », gnupg sert à vérifier l'authenticité des paquets téléchargés depuis le dépôt Wazuh et apt-transport-https est un plugin qui permet à apt de supporter le HTTPS pour le téléchargement de paquets pour toute Debian inférieure à 10, pour les versions supérieures, cela est intégré par défaut.

Ensuite, le téléchargement de la clé GPG de Wazuh (clé publique).

Après, l'ajout des sources pour le téléchargement des paquets Wazuh par rapport à la clé GPG téléchargée précédemment.

Puis l'installation du paquet wazuh-agent.

Une fois cela fait, il y a l'utilisation du template « ossec.conf.j2 » qui contient la configuration qui est mise à la place du fichier « ossec.conf » afin que l'agent Wazuh puisse avoir la bonne configuration.

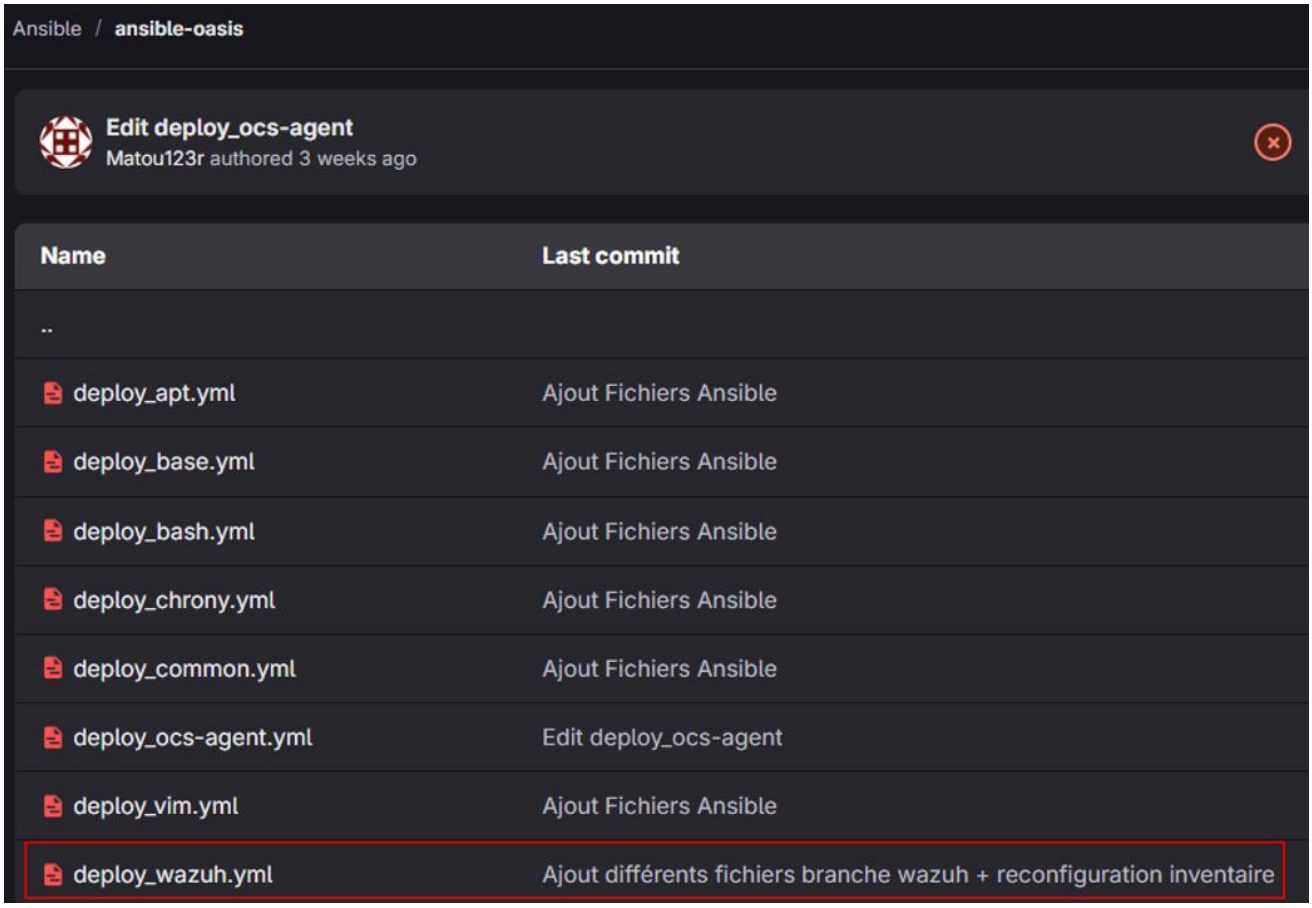
Enfin, le service wazuh-agent est activé et démarré sur la machine et le daemon systemd est relancé.

```yaml
- name: Install gnupg and apt-transport-https
  ansible.builtin.apt:
    name:
      - gnupg
      - apt-transport-https
    state: present
    update_cache: yes

- name: Download Wazuh GPG key
  ansible.builtin.get_url:
    url: https://packages.wazuh.com/key/GPG-KEY-WAZUH
    dest: /usr/share/keyrings/wazuh.gpg
    mode: '0644'

- name: Add Wazuh apt repository
  ansible.builtin.apt_repository:
    repo: "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main"
    state: present
    filename: wazuh
    update_cache: yes

- name: Install Wazuh agent
  ansible.builtin.apt:
    name: wazuh-agent
    state: present
    update_cache: yes

- name: Configure Wazuh agent
  ansible.builtin.template:
    src: ossec.conf.j2
    dest: /var/ossec/etc/ossec.conf
    owner: root
    group: wazuh
    mode: '660'
  notify: restart wazuh-agent

- name: Enable and start Wazuh agent service
  ansible.builtin.systemd:
    name: wazuh-agent
    enabled: yes
    state: started
    daemon_reload: yes
```

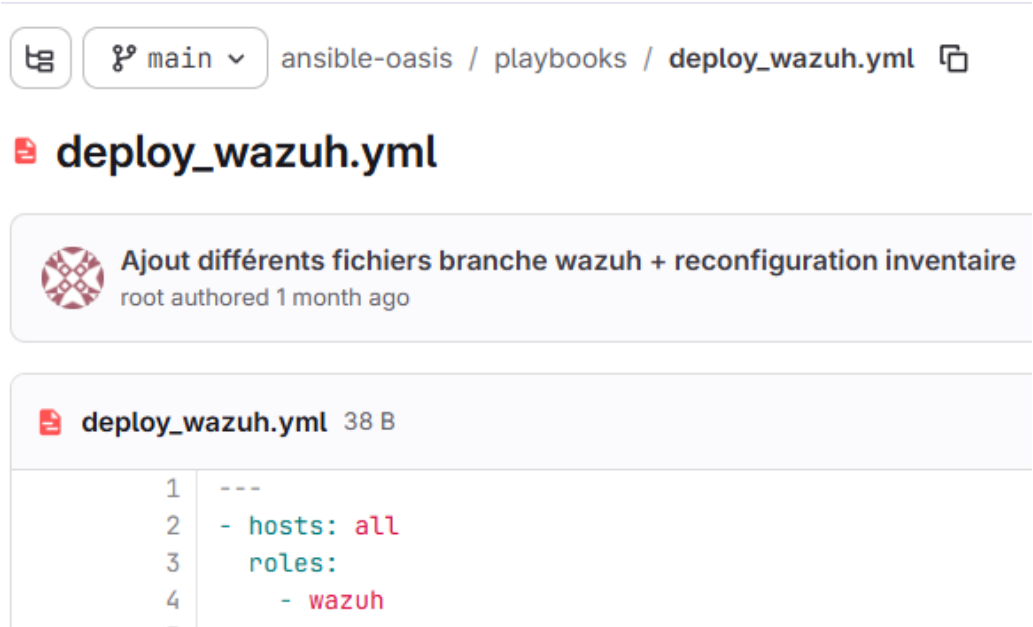
Avec la configuration de ces différents fichiers, le playbook est terminé.

7) Création de playbooks

Une fois le rôle réalisé, il faut créer le playbook correspondant, pour cela, il suffit d'aller dans le dossier « playbooks » à la racine du projet Gitlab ansible-oasis puis créer un playbook correspondant au rôle, ici pour le rôle Wazuh, le playbook « deploy_wazuh.yml ».



Dans ce playbook, la configuration est assez simple, on ne définit pas directement les machines ciblées, car elles sont déclarées dans l'inventaire. Le champ « hosts : all » indique simplement que le playbook doit s'exécuter sur tous les hôtes présents dans cet inventaire. Ensuite, on définit le rôle à appliquer, dans notre cas le rôle Wazuh.

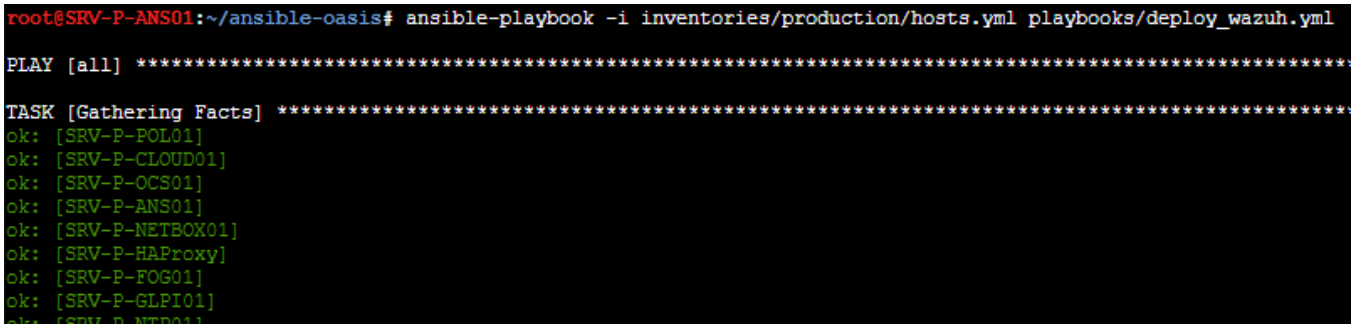


8) Lancement de playbooks

Une fois le playbook créé, il est possible de l'exécuter, pour cela il faut aller sur la machine permettant d'exécuter Ansible, sur cette machine, il faut utiliser la commande « ansible-playbook »

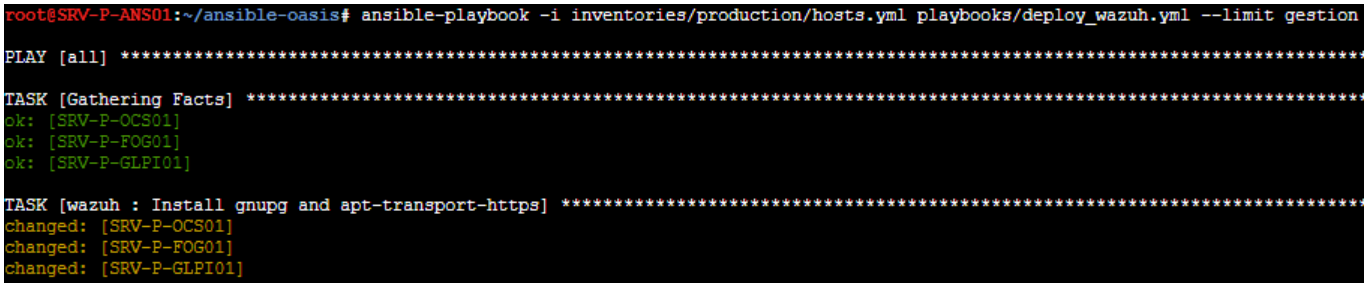
Cette commande a différentes options, il est tout d'abord possible d'exécuter un playbook sur toutes les machines d'un inventaire en déterminant l'inventaire à utiliser et rien de plus comme ci-dessous sur l'inventaire de production.

```
ansible-playbook -i inventories/production/hosts.yml playbooks/deploy_wazuh.yml
```



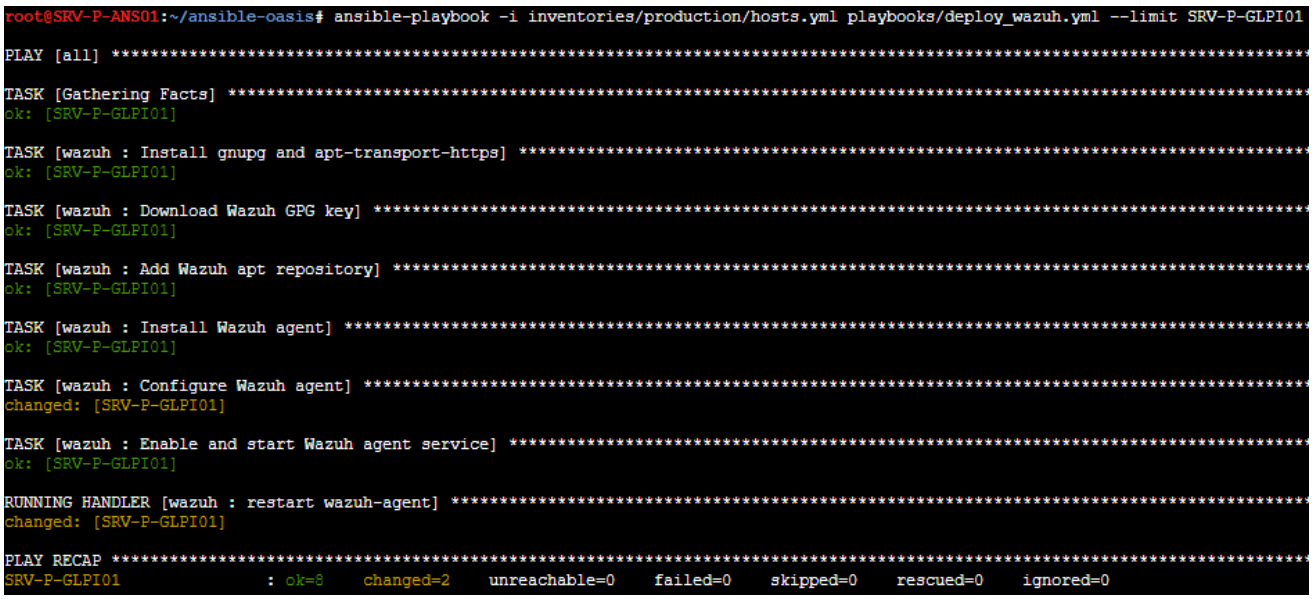
Ensuite, il est possible d'exécuter le playbook sur un groupe d'host définit, pour cela, il faut utiliser l'option « --limit » comme ci-dessous avec le groupe gestion.

```
ansible-playbook -i inventories/production/hosts.yml playbooks/deploy_wazuh.yml --limit gestion
```



Enfin, il est également possible d'exécuter le playbook sur un host précis avec l'option « --limit » mais avec le nom exactement défini dans l'inventaire, comme ci-dessous pour la machine GLPI. Il est également possible de voir sur cette image que la tâche réussit bien et qu'il n'y a pas d'erreur.

```
ansible-playbook -i inventories/production/hosts.yml playbooks/deploy_wazuh.yml --limit SRV-P-GLPI01
```



Les options --check permettent d'exécuter le playbook en mode simulation, sans appliquer de changements sur les machines. L'option --diff permet d'afficher les différences entre l'état actuel et l'état souhaité.

9) Rôles et Playbooks infrastructure Oasis

Au sein de l'infrastructure d'OASIS, différents rôles sont utilisés sur les machines Linux, chaque rôle correspond à un playbook du même nom. Le playbook se contente d'appeler le rôle voulu, cela permet quand même de pouvoir réutiliser les rôles selon le besoin.

Ci-dessous l'ensemble des rôles disponibles :

Rôle / Playbook	Description	Fichier playbook
apt	Gestion des dépôts et paquets APT. Configure les sources des machines Debian.	roles/apt
base	Configuration de base commune à tous les serveurs : installe les paquets de base voulu dans l'infrastructure.	roles/base
bash	Configuration de l'environnement shell Bash avec le bash_rc + installation bash-completion.	roles/bash
chrony	Synchronisation NTP via chrony. Assure la coherence de l'heure sur l'ensemble des systèmes Linux.	roles/chrony
common	Tâche appliquée à tous les hôtes, réutilise différents rôles à appliquer par défaut sur les machines	roles/common
ocs-agent	Déploiement et configuration de l'agent OCS inventory. Permet l'inventaire automatisé du parc.	roles/ocs-agent
vim	Installation et configuration de l'éditeur Vim	roles/vim
wazuh	Déploiement de l'agent Wazuh, permet de remonté les différentes faille de sécurité	roles/wazuh

k) Mise en place de l'inventaire dynamique NetBox

1) Présentation de l'inventaire dynamique

L'inventaire dynamique Ansible permet de ne pas gérer manuellement la liste des machines ciblées par les playbooks. Contrairement à l'inventaire statique et basique de Ansible où les hôtes sont définis dans un fichier fixe.

Pour cet inventaire dynamique, l'outil NetBox est utilisé comme source de vérité. Ansible interagit avec l'API de NetBox pour récupérer les informations sur les différentes machines voulues. Grâce à cela, il est possible de centraliser toutes les informations relatives à l'infrastructure dans un seul outil de façon simple.

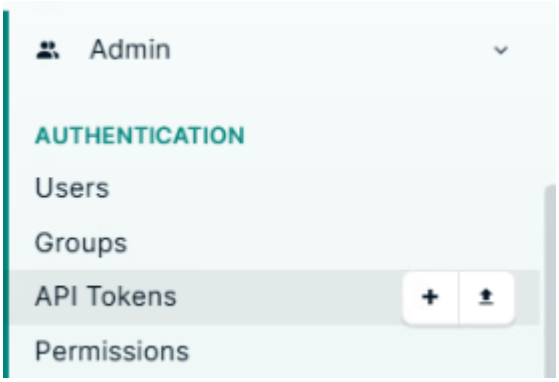
Ce fonctionnement permet de garantir un certain niveau de cohérence au niveau des données, comme la même source d'informations est utilisée de façon constante.

Pour que l'inventaire dynamique soit fonctionnel, il faut configurer un token afin que la machine qui exécute Ansible puisse aller chercher les informations dans NetBox, ensuite il faut configurer des tags permettant de séparer les différents inventaires puis ajouter un fichier de configuration pour l'inventaire dynamique.

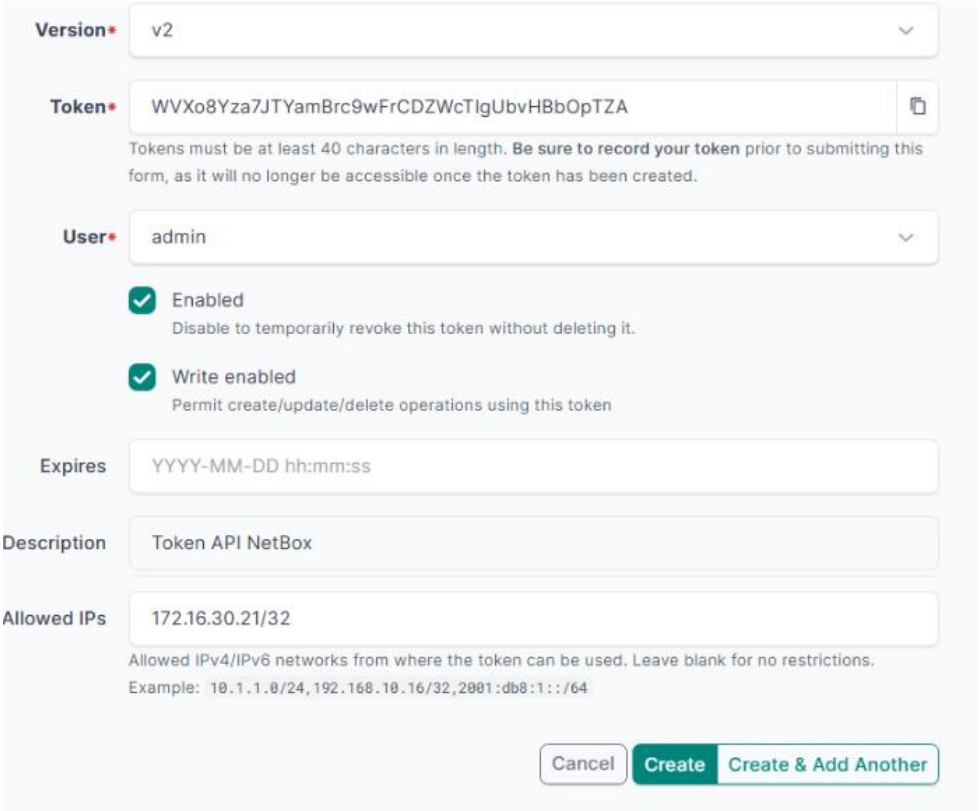
2) Création de token Netbox

Au sein de cette configuration, il est considéré que l'installation de l'outil NetBox a déjà été effectuée, il est tout de même possible de retrouver cette mise en place en Annexe 4.

Pour la création d'un token NetBox, il faut aller sur l'interface Web de NetBox et se connecter dessus avec un compte administrateur. Une fois connecté, il est possible d'aller dans l'onglet « Admins » puis « API Tokens » comme ci-dessous.



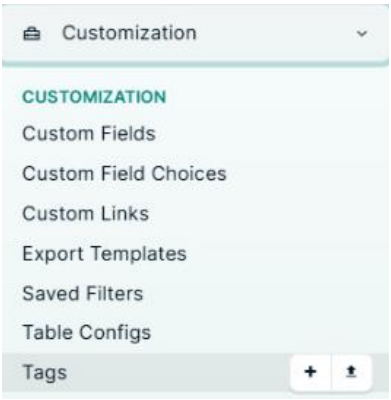
Dans cet onglet, il faut ajouter un token, l'attribuer à un utilisateur spécifique, activer le token et définir sur quelle adresse IP celui-ci peut être utilisé, l'adresse IP définie correspond à la machine dédiée à l'exécution de Ansible afin de limiter les risques. (le token de la capture ci-dessous est un token fictif)



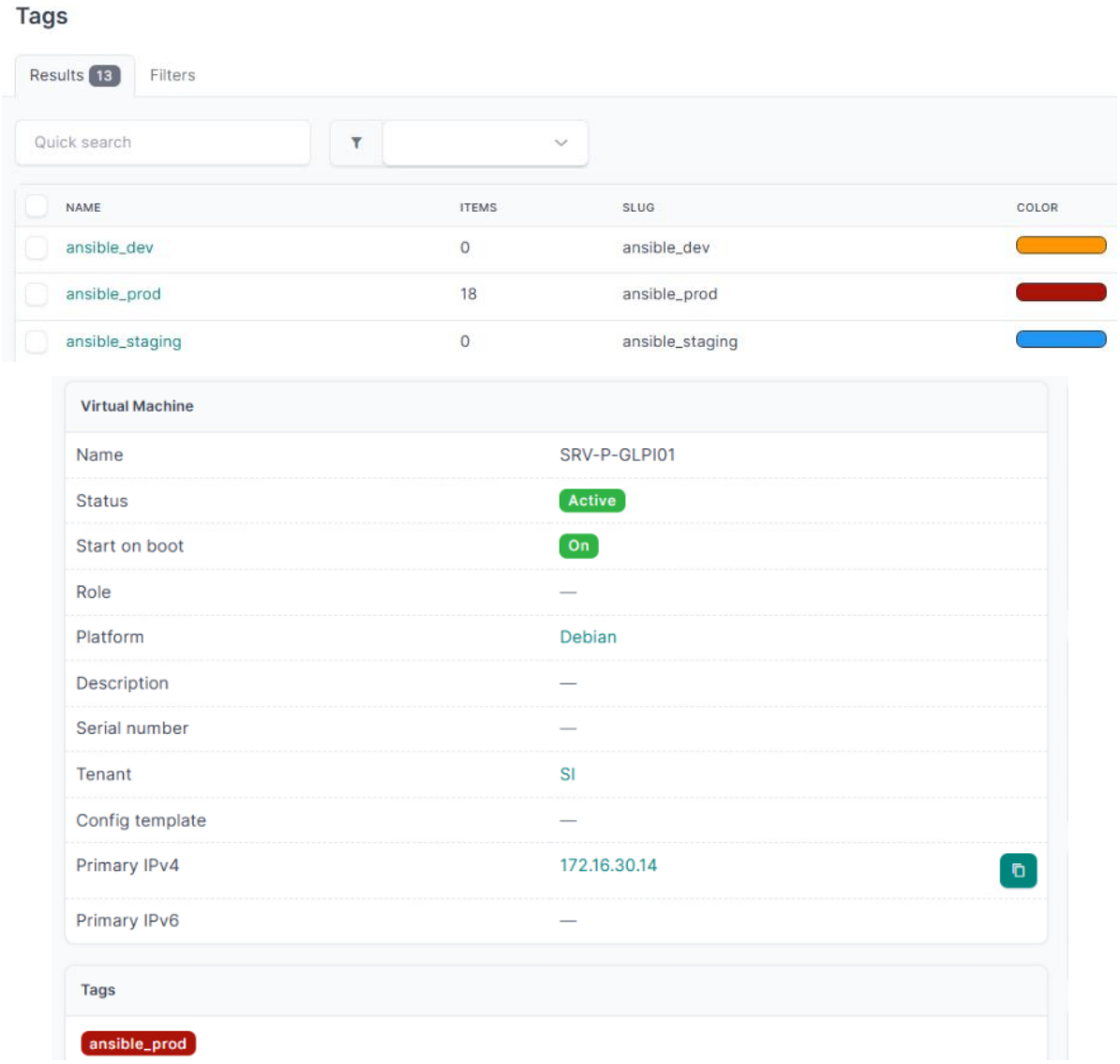
Une fois le token ajouté, celui-ci peut être utilisé.

3) Création de Tags Netbox

Maintenant que le token pour l'inventaire dynamique est créé, il faut ajouter les différents tags permettant de séparer les différents inventaires dynamiques. Pour cela, il faut aller dans l'onglet « Customization » puis « Tags » sur Netbox, comme ci-dessous.



Dans cet onglet, il faut ajouter 3 tags avec différentes couleurs afin d'avoir un tag différent pour chaque inventaire. Au moins un de ces tags doit toujours être attribué à une machine Debian qui est ajoutée dans NetBox afin d'avoir une configuration correcte de la machine avec Ansible ensuite, comme sur la seconde image ci-dessous pour la machine « SRV-P-GLPI01 » qui possède le tag « ansible_prod ».

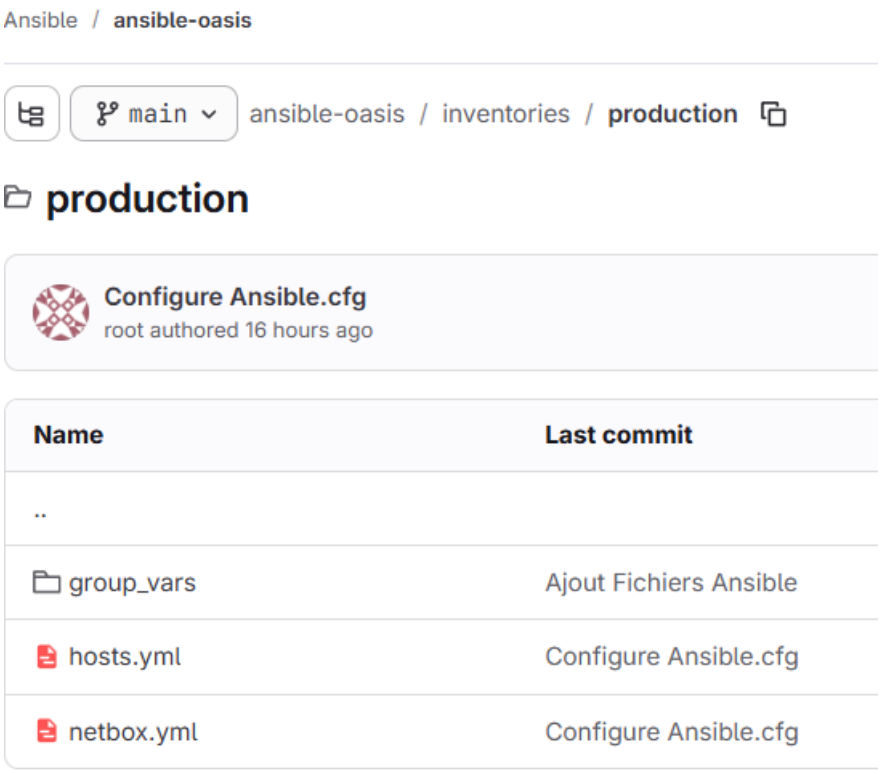


Une fois les tags ajoutés, il faut réaliser la configuration des fichiers d'inventaire dynamique.

4) Configuration des fichiers Netbox.yml

Pour la configuration des fichiers d'inventaire dynamique, il faut les ajouter sur Gitlab dans le dossier inventories de Ansible.

Dans ce dossier, il y a les dossiers production, development et staging. Dans chacun de ces dossiers, un fichier netbox.yml a été créé, comme ci-dessous dans le dossier « production ».



Dans ce fichier, il y a différents paramètres :

plugin : Effectue la connexion à NetBox via le plugin netbox.netbox.nb_inventory

api_endpoint : Définis l'url de connexion à NetBox

validate_certs : Définis la gestion des certificats SSL pour la connexion à NetBox

config_context : Récupère les variables NetBox

flatten_config_context : Simplifie le format des variables données pour Ansible

context_namespace : Place les variables sous un namespace Ansible

strict : Échoue si incohérence au niveau de l'inventaire

query_filters : Filtre les équipements avec le tag ansible_prod pour la production

group_by : Créer des groupes Ansible automatiquement par rapport au rôle des équipements et aux tags

rename_variables : Renomme les variables exportées par NetBox afin d'avoir un préfixe NetBox

compose : Créer une variable personnalisée à partir de champs personnalisés

netbox.yml 457 B

```yaml
plugin: netbox.netbox.nb_inventory
api_endpoint: https://netbox.oasis.local
validate_certs: false

# Config context settings
config_context: true
flatten_config_context: true
context_namespace: ansible

# make invalid entries raise
strict: true

query_filters:
  - tag: ansible_prod

group_by:
  - device_roles
  - tags

rename_variables:
  - pattern: tags
    repl: netbox_tags
  - pattern: serial
    repl: netbox_serial

compose:
  netbox_object_id: id
```

Les inventaires pour la production, le développement et le staging sont les mêmes, la seule chose qui change est le tag utilisé pour filtrer les machines : ansible_prod, ansible_dev ou ansible_staging.

5) Test de l'inventaire dynamique

Pour tester le bon fonctionnement de l'inventaire dynamique, il faut aller sur la machine dédiée à exécuter ansible et avoir le répertoire GitLab correspondant à ansible-oasis.

Ensuite, il faut installer le plugin Ansible permettant de faire simplement la connexion à l'API de NetBox.

```
ansible-galaxy collection install netbox.netbox
```

```
root@SRV-P-ANS01:~/ansible-oasis# ansible-galaxy collection install netbox.netbox
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/api/v3/plugin/ansible/content/published/collections/artifact
Installing 'netbox.netbox:3.22.0' to '/root/.ansible/collections/ansible_collections/netbox/netbox'
netbox.netbox:3.22.0 was installed successfully
root@SRV-P-ANS01:~/ansible-oasis#
```

Une fois la collection installée, il faut installer le module Python « pynetbox » afin que la collection ansible puisse fonctionner correctement.

```
apt install python3-pynetbox
```

```
root@SRV-P-ANS01:~/ansible-oasis# apt install python3-pynetbox
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les NOUVEAUX paquets suivants seront installés :
  python3-pynetbox
0 mis à jour, 1 nouvellement installés, 0 à enlever et 19 non mis à jour.
Il est nécessaire de prendre 20,9 ko dans les archives.
Après cette opération, 120 ko d'espace disque supplémentaires seront utilisés.
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 python3-pynetbox all 7.0.0-1 [20,9 kB]
20,9 ko réceptionnés en 0s (202 ko/s)
Sélection du paquet python3-pynetbox précédemment désélectionné.
(Lecture de la base de données... 73573 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../python3-pynetbox_7.0.0-1_all.deb ...
Dépaquetage de python3-pynetbox (7.0.0-1) ...
Paramétrage de python3-pynetbox (7.0.0-1) ...
root@SRV-P-ANS01:~/ansible-oasis#
```

De plus, comme le token n'est pas défini explicitement dans l'inventaire pour des raisons de sécurité, il faut l'importer via une variable d'environnement. Afin que celle-ci reste disponible sur la machine et ne soit pas effacée à chaque redémarrage, il faut l'ajouter dans le fichier .bashrc de l'utilisateur avec lequel on est connecté pour exécuter Ansible, comme indiqué ci-dessous. Pour des raisons de sécurité, le token utilisé, créé dans NetBox, a été remplacé par « MY_TOKEN » sur la capture d'écran ci-dessous.

```
vim ~/.bashrc
```

```
# /etc/bash.bashrc - Configuration bash globale

# Si non interactif, ne rien faire
[ -z "$PS1" ] && return

##############
# HISTORIQUE
##############
HISTSIZE=10000
HISTFILESIZE=20000
HISTCONTROL=ignoredups:erasedups
HISTTIMEFORMAT="%F %T "
shopt -s histappend

##############
# OPTIONS
##############
shopt -s checkwinsize
shopt -s cdspell
shopt -s globstar 2>/dev/null
shopt -s nocaseglob

stty -ixon

##############
# VARIABLES
##############
export EDITOR=vim
export VISUAL=vim
export PAGER=less
export LESS='-R -i'
export NETBOX_TOKEN=MY_TOKEN
```

Afin que la configuration soit prise en compte sans redémarrer, il faut recharger le contenu du fichier « .bashrc », comme ci-dessous.

```
source ~/.bashrc
```

```
root@SRV-P-ANS01:~# source ~/.bashrc
root@SRV-P-ANS01:~#
```

Enfin, pour tester le bon fonctionnement de l'inventaire dynamique, il est possible de lister les machines trouvées dans l'inventaire, comme ci-dessous. On peut constater qu'Ansible récupère l'ensemble des informations des machines présentes dans NetBox et qu'il ne prend en compte que celles portant le tag ansible_prod, puisque l'inventaire de production est utilisé.

```
ansible-inventory -i inventories/production/netbox.yml
```

```
root@SRV-P-ANS01:~/ansible-oasis# ansible-inventory -i inventories/production/netbox.yml --list
[WARNING]: Collection netbox.netbox does not support Ansible version 2.14.18
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
{
    "_meta": {
        "hostvars": {
            "SRV-N-POL01": {
                "ansible_host": "172.19.30.25",
                "cluster": "SRV-P-PROX01",
                "cluster_type": "proxmox",
                "custom_fields": {},
                "disk": 20000,
                "is_virtual": true,
                "local_context_data": [
                    null
                ],
                "locations": [],
                "memory": 2000,
                "netbox_object_id": 23,
                "netbox_serial": "",
                "netbox_tags": [
                    "ansible_prod",
                    "vim"
                ],
                "platforms": [
                    "debian"
                ],
                "primary_ip4": "172.19.30.25",
                "regions": [],
                "services": [],
                "site_groups": [],
                "sites": [
                    "nantes"
                ],
                "status": {
                    "label": "Active",
                    "value": "active"
                },
                "tenants": [
                    "si"
                ],
                "vcpus": 1.0
```

Avec cela, l'inventaire dynamique est fonctionnel, il est possible de lancer des playbooks avec cet inventaire dynamique.

I) Mise en place de tâche automatisée avec Gitlab et Netbox

1) Fonctionnement Inventaire Gitlab, Netbox et Ansible

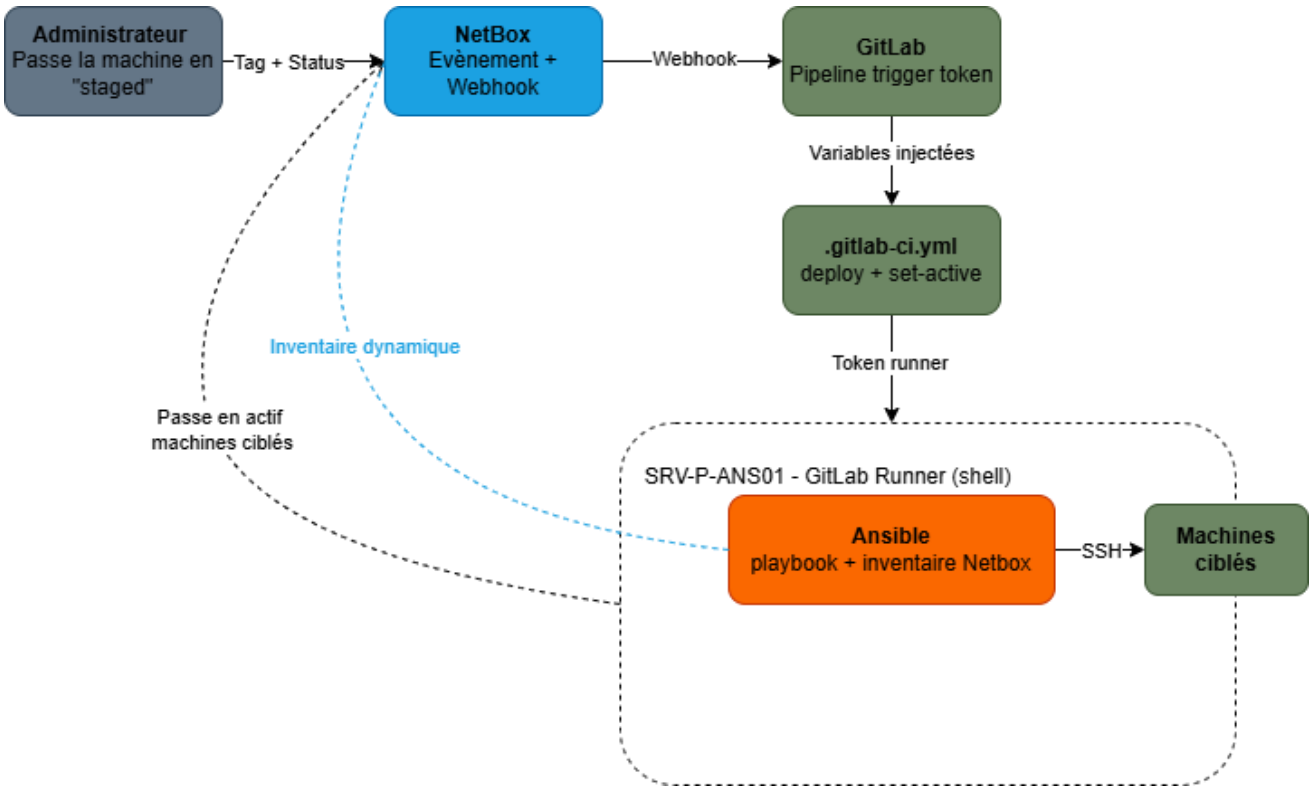
L'inventaire dynamique Ansible permet de ne pas gérer manuellement la liste des machines ciblées par les playbooks mais avec seulement un inventaire dynamique, il faut quand même être sur une machine pouvant exécuter Ansible afin de lancer les différents playbooks.

Afin de pouvoir exécuter les playbooks simplement, il est possible d'utiliser l'API de NetBox avec GitLab. Avec cela, si une machine est ajoutée ou modifiée dans NetBox, un Webhook est déclenché. Celui-ci appelle un trigger de pipeline GitLab via un token défini au préalable. GitLab lance alors automatiquement un pipeline CI/CD.

Ce pipeline est exécuté sur un Gitlab Runner hébergé dans l'infrastructure. Le runner exécute les jobs définis dans le pipeline.

La machine exécutant Ansible utilise ensuite les informations de NetBox comme inventaire dynamique afin d'exécuter les bons playbooks sur les machines ciblées.

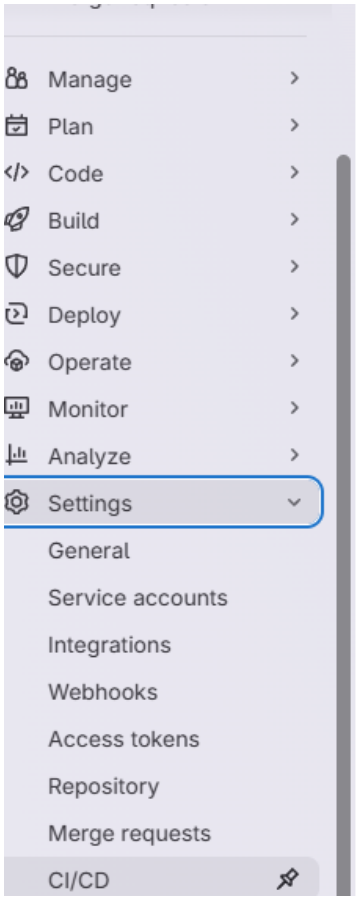
Ce fonctionnement permet de centraliser la gestion de l'infrastructure, d'automatiser les déploiements et de garantir une cohérence des données entre NetBox, Gitlab et Ansible. Ci-dessous un schéma du fonctionnement de cette infrastructure automatisée.

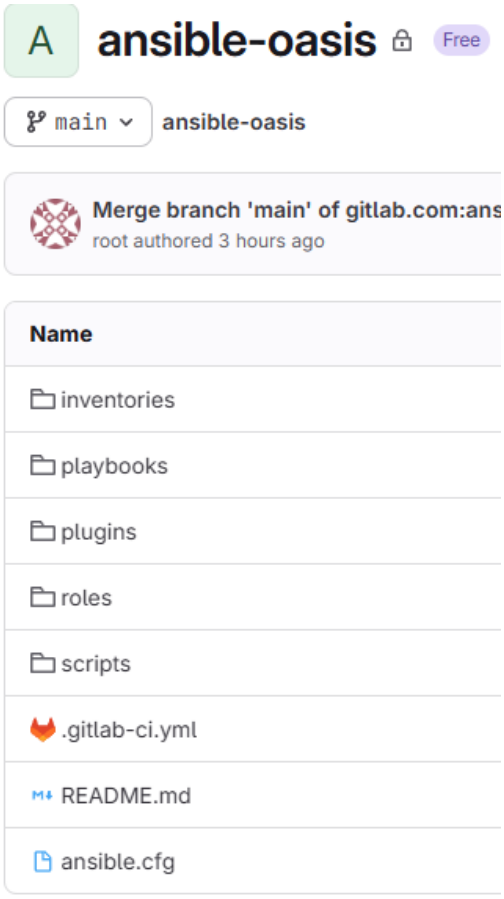


Pour que cette infrastructure soit fonctionnelle, il faut configurer le déclencheur d'évènement Netbox

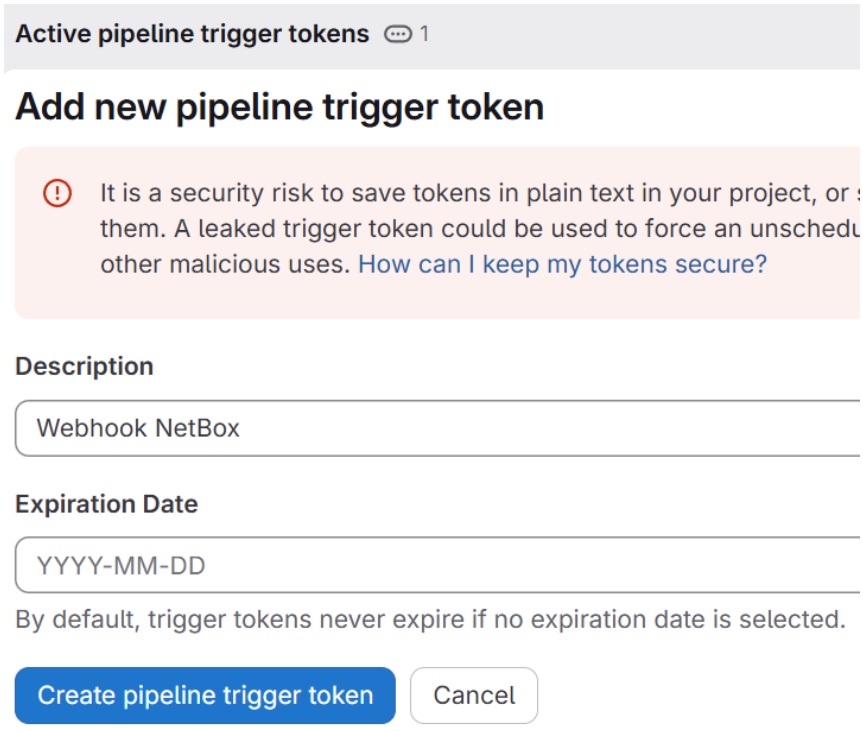
2) Configuration pipeline trigger token + variables cachées

Le token pipeline trigger est ce qui va permettre à NetBox de déclencher le .gitlab-ci.yml avec le fonctionnel final, pour le configurer, il faut aller sur le projet « ansible-oasis » dans GitLab et aller dans « Settings » puis « CI/CD ».

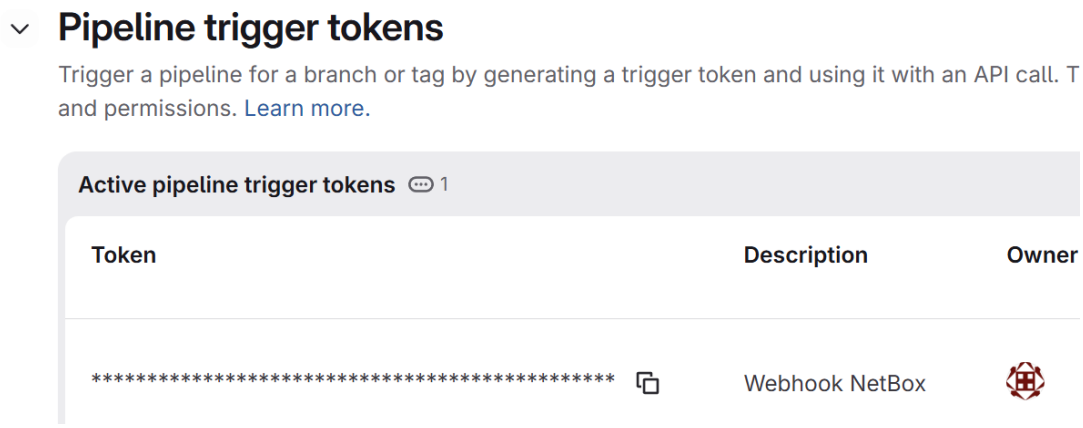




Sur la page de configuration, il faut ajouter un nouveau token comme ci-dessous.



Une fois le token créé, il est possible de le copier en allant dessus afin de l'ajouter dans NetBox dans les étapes suivantes.



Ensuite, pour l'ajout des variables cachées dans GitLab afin de ne pas mettre d'informations sensibles directement dans la tâche .gitlab-ci.yml, il faut rester dans le même onglet mais aller dans « Variables ».

La variable cachée qu'il faut ajouter pour le fonctionnement futur du .gitlab-ci.yml est une variable correspondant à l'url de NetBox et une variable correspondant au token de connexion NetBox afin que la tâche puisse repasser les machines en « active » sur l'outil de gestion d'infrastructure.

Pour cela, il suffit de cliquer sur « Add variable » et il faut renseigner les éléments suivants :

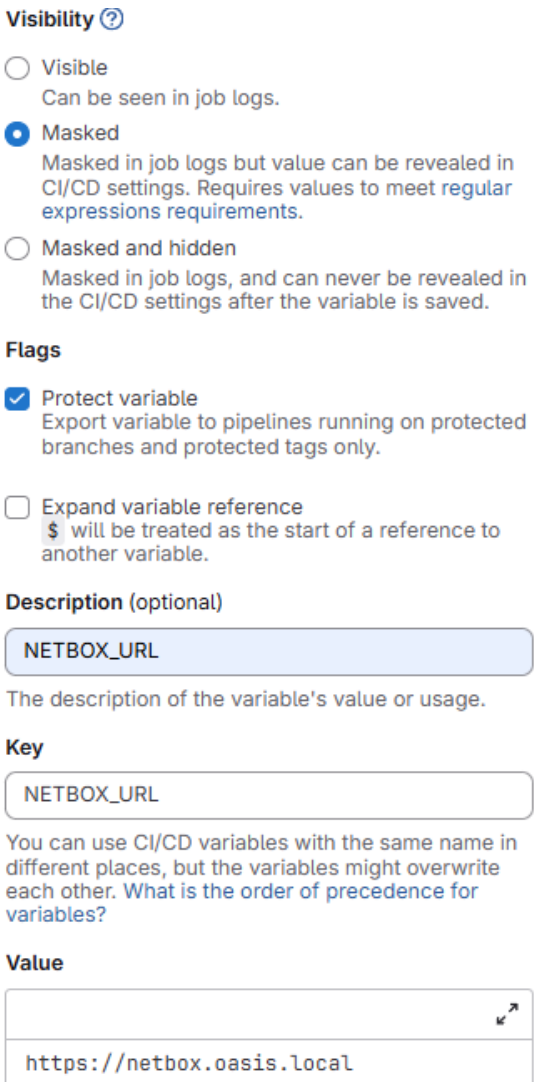
Visibility : Correspond à la visibilité de la variable

Flags : Le flag correspond au périmètre de la variable, ici elle doit être une variable protégée

Description : Description de l'usage de la variable

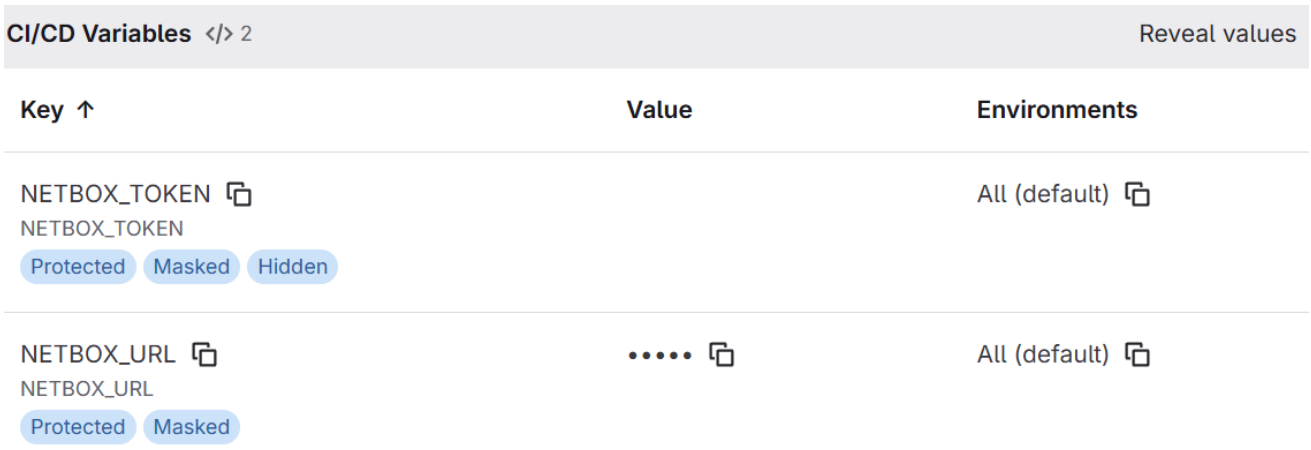
Key : La clé est le nom de la variable, c'est ce qui va permettre d'utiliser la variable

Value : La valeur est ce qu'il doit être caché, ici, « https://netbox.oasis.local »



Il faut également ajouter la variable correspondant au token NetBox, pour celle-ci, il faut mettre la visibilité « Masked and hidden ».

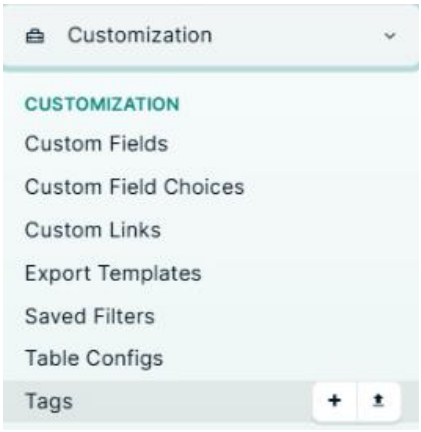
Une fois les deux variables ajoutées, il est possible de les voir, dans l'onglet variables, comme ci-dessous.



3) Configuration Tags NetBox

Après avoir configuré le token pipeline trigger et les variables cachées GitLab, il faut configurer les tags correspondant au playbook sur NetBox.

Pour cela, il faut aller dans l'onglet « Customization » puis « Tags » comme ci-dessous.

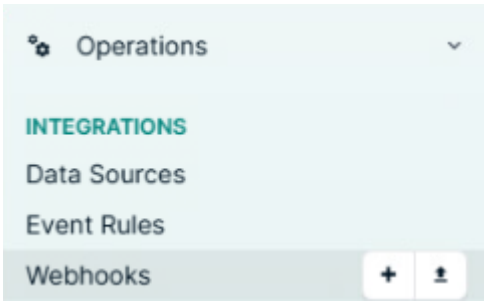


Dans cet onglet, il faut ajouter un tag correspondant à chaque rôle ajouté dans Ansible, comme ci-dessous.

Une fois les tags ajoutés, il faut faire la configuration du déclencheur d'événement .

4) Configuration Webhook NetBox

Pour la configuration du Webhook permettant de déclencher le .gitlab-ci.yml grâce au token pipeline trigger, il faut aller dans « Operations » puis « Webhooks » sur NetBox.



Dans cet Webhook est configuré différents éléments :

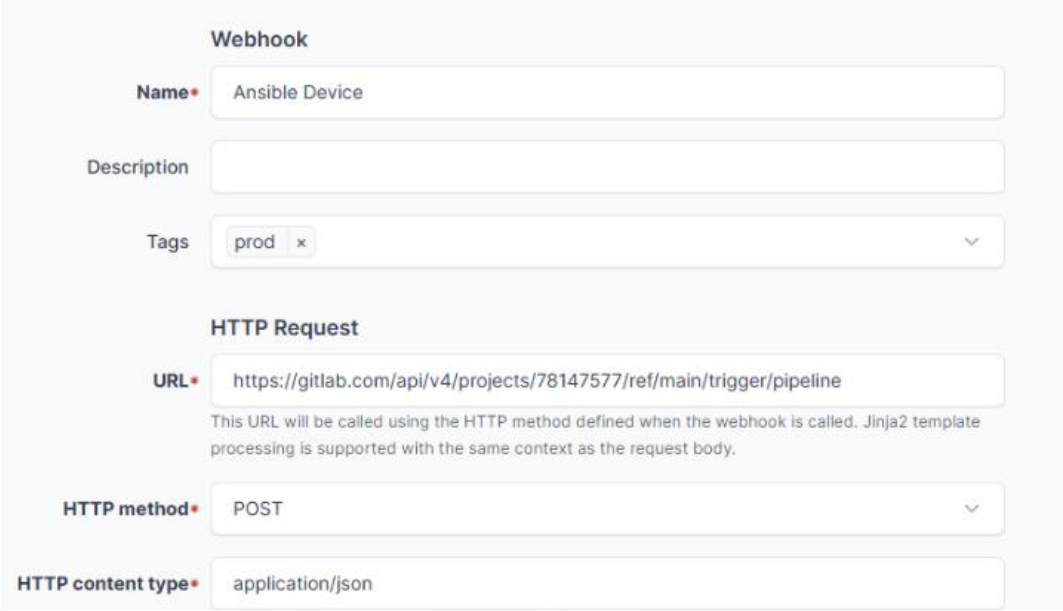
Name : Nom du Webhook NetBox

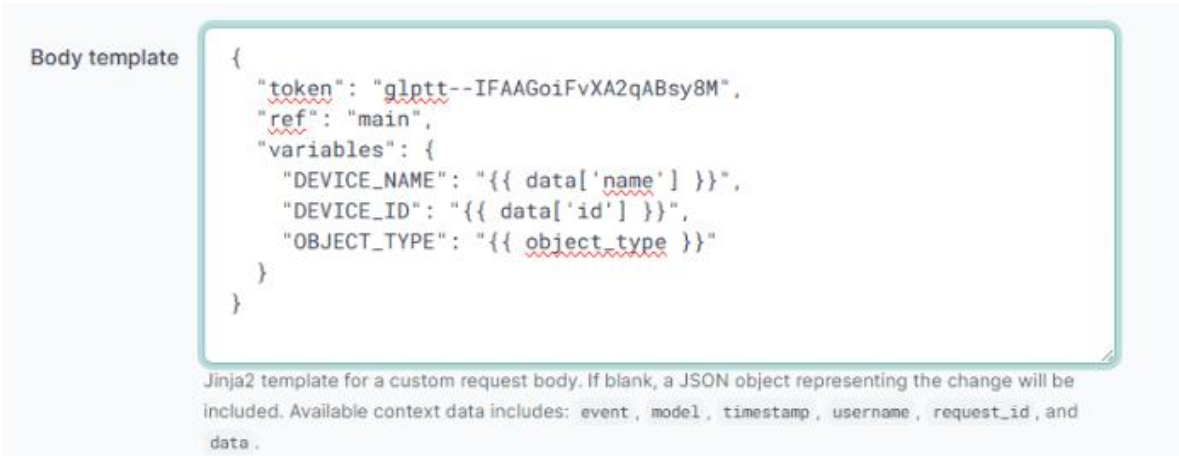
URL : Adresse que NetBox va appeler, contient l'ID du projet GitLab et indique quel projet doit être déclenché

HTTP method : POST est utilisé car on envoie des données à GitLab

HTTP content type : Indique à GitLab que les données sont envoyées au format JSON

Body template : Contenu envoyé à GitLab, le token pour prouver que la requête est autorisée, main pour indiquer la branche sur laquelle déclencher le pipeline et les différentes variables extraites de NetBox pour les informations de la machine cible





5) Configuration déclencheur d'événements NetBox

Ensuite, pour le bon fonctionnement du déclencheur d'événements dans NetBox qui va permettre d'exécuter le Webhook créé précédemment si une machine passe en statut « staged » sur NetBox, il faut aller dans l'onglet « Operation » puis « Event Rules », il faut ajouter une règle d'événement.

Dans cette règle d'événement est configurée différentes valeurs :

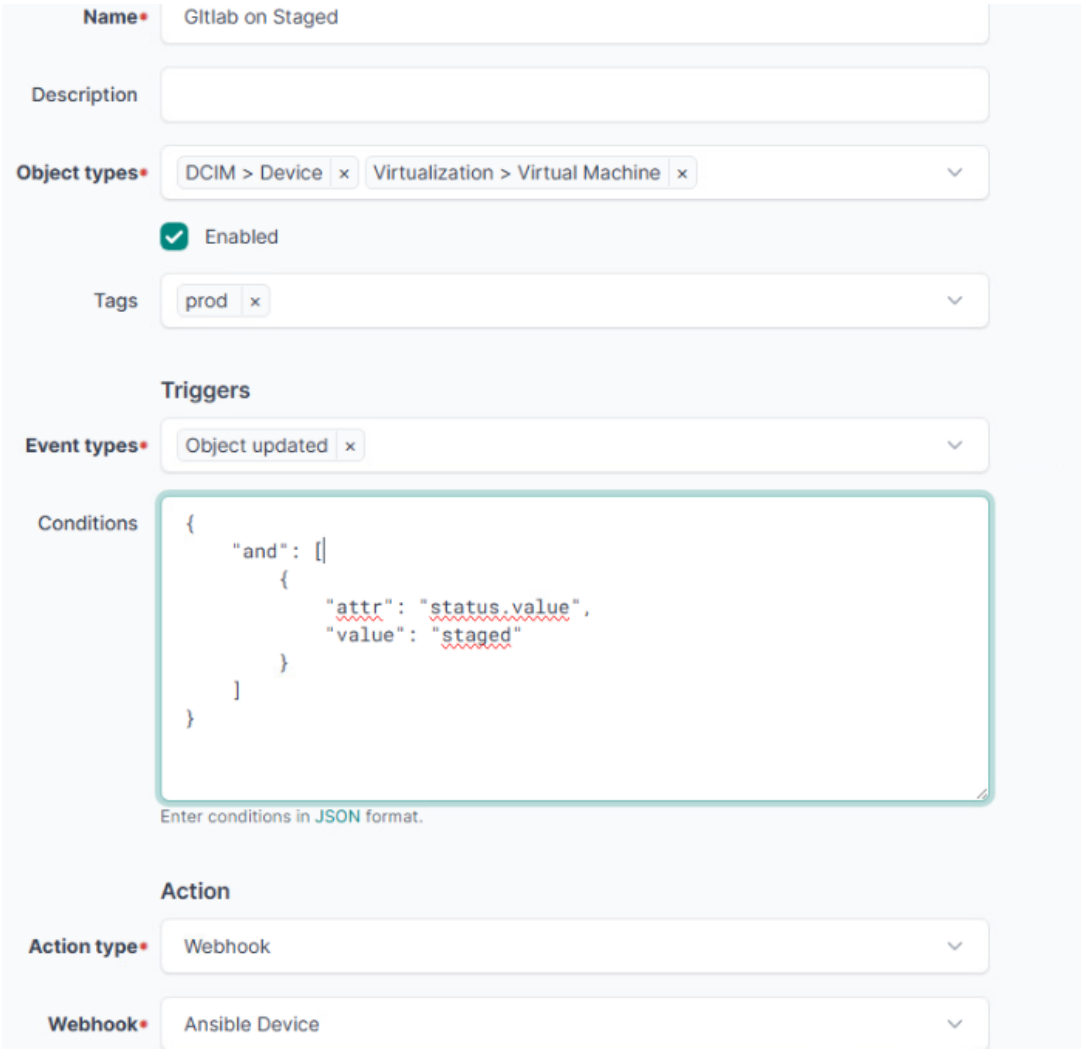
Object Types : Types d'objets sur lesquels cette règle doit agir, les serveurs physiques et les machines virtuelles

Event types : Type d'événement permettant de déclencher la règle, lorsqu'un objet est mis à jour

Conditions : Condition pour déclencher la règle, si le statut de la machine est en « staged » (configuration)

Action type : Type d'actions à réaliser lors du déclenchement de la règle, lancement d'un Webhook

Webhook : Définis le Webhook à lancer lors du déclenchement de la règle



6) Configuration Gitlab runner

Avec les configurations précédentes, la configuration NetBox est terminée, afin que la configuration .gitlab-ci.yml puisse être exécutée sur l'infrastructure Ansible, il faut un runner permettant d'exécuter les différentes tâches.

Pour la mise en place de celui-ci, il faut tout d'abord ajouter le dépôt GitLab Runner comme ci-dessous.

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | bash
```

```
root@SRV-P-ANS01:~# curl -L "https://packages.gitlab.com/install/repositories/runner/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5315  100  5315    0     0  11402      0 --:--:-- --:--:-- --:--:-- 11405
Atteint :1 http://deb.debian.org/debian bookworm InRelease
Atteint :2 http://deb.debian.org/debian bookworm-updates InRelease
Atteint :3 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :4 https://packages.wazuh.com/4.x/apt stable InRelease
Atteint :5 https://packages.gitlab.com/runner/gitlab-runner/debian bookworm InRelease
Lecture des listes de paquets... Fait
Repository configured successfully.
Ready to install packages.
```

Une fois le dépôt GitLab Runner installé, il faut installer le paquet correspondant comme ci-dessous.

```
apt install -y gitlab-runner
```

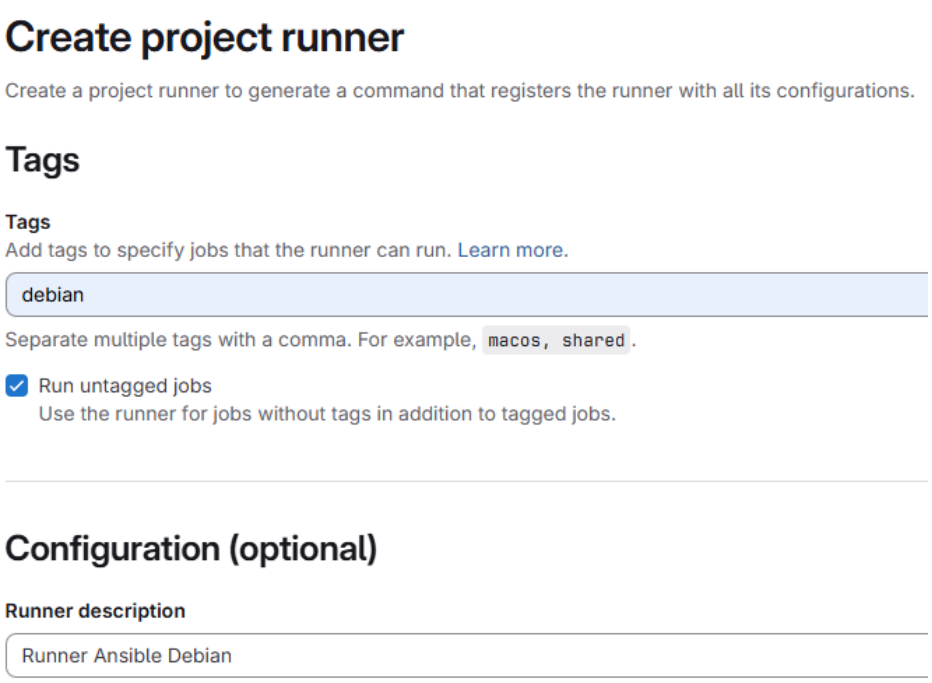
```
root@SRV-P-ANS01:~# apt install -y gitlab-runner
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  gitlab-runner-helper-images
Paquets suggérés :
  docker-engine
Les NOUVEAUX paquets suivants seront installés :
  gitlab-runner
Les paquets suivants seront mis à jour :
  gitlab-runner-helper-images
1 mis à jour, 1 nouvellement installés, 0 à enlever et 17 non mis à jour.
Il est nécessaire de prendre 562 Mo dans les archives.
Après cette opération, 120 Mo d'espace disque supplémentaires seront utilisés.
Réception de :1 https://packages.gitlab.com/runner/gitlab-runner/debian bookworm/main
0% [1 gitlab-runner-helper-images 832 kB/533 MB 0%]
```

Ensuite, il faut se rendre sur le projet « ansible-oasis » sur GitLab, puis aller dans « Settings », « CI/CD », puis « Runners ». Sur cette page, il est nécessaire de créer un nouveau runner pour le projet. Lors de la configuration du runner, il faut spécifier différents paramètres :

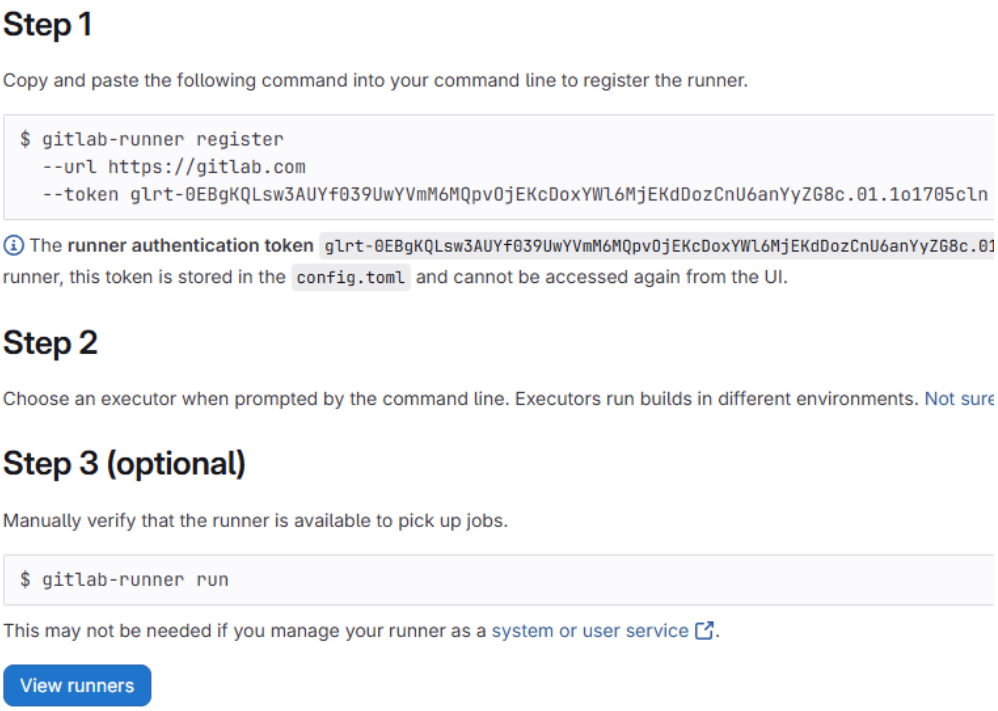
Tags : Définit les tags à spécifier pour les tâches que le runner peut exécuter

Run untagged jobs : Cette case à cocher permet d'autoriser ce runner à exécuter les tâches ne disposant d'aucun tag

Runner Description : Définit la description du runner



Lorsque la page précédente est validée, la page ci-dessous devrait s'afficher, sur celle-ci il faut copier ce qui se trouve dans l'étape 1 afin de coller la commande sur la machine qui va être runner.



La commande « gitlab-runner register » permet d'enregistrer le runner grâce au token fourni, cet ajout va configurer le fichier config.toml dans « /etc/gitlab-runner/ ». (Ce token est fictif et non utilisé en production)

```
gitlab-runner register  --url https://gitlab.com  --token glrt-
0EBgKQLsw3AUZJ2JFS9ffz92FJS9SFNzqAAQf8c.01.1of105cfn
```

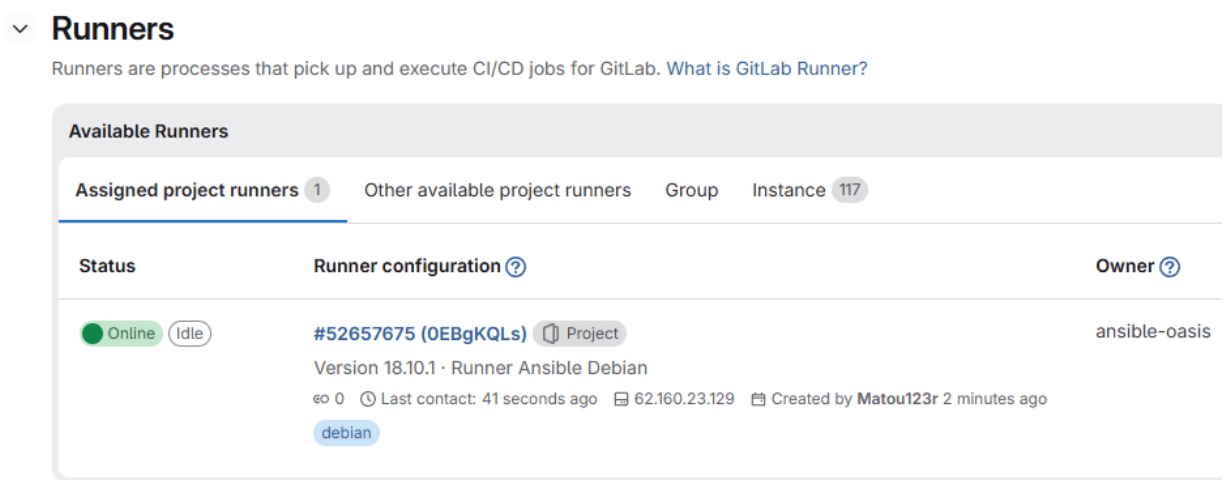
```
root@SRV-P-ANS01:~# gitlab-runner register  --url https://gitlab.com  --token glrt-0EBgKQLsw
Runtime platform                                    arch=amd64 os=linux pid=4397 revision=3b
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
[https://gitlab.com]:
Verifying runner... is valid                        correlation_id=9eb35c04ff85d494-CDG runn
Enter a name for the runner. This is stored only in the local config.toml file:
[SRV-P-ANS01]:
Enter an executor: ssh, virtualbox, kubernetes, custom, shell, parallels, docker, docker-win
shell
Runner registered successfully. Feel free to start it, but if it's running already the confi

Configuration (with the authentication token) was saved in "/etc/gitlab-runner/config.toml"
```

Une fois l'enregistrement fait, il est possible de regarder le statut du service correspondant au gitlab-runner, celui-ci devrait être redémarré, comme ci-dessous.

```
systemctl status gitlab-runner
```

```
root@SRV-P-ANS01:~# systemctl status gitlab-runner
● gitlab-runner.service - GitLab Runner
     Loaded: loaded (/etc/systemd/system/gitlab-runner.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-04-12 17:47:28 CEST; 41s ago
   Main PID: 629 (gitlab-runner)
      Tasks: 7 (limit: 2298)
     Memory: 91.6M
        CPU: 249ms
     CGroup: /system.slice/gitlab-runner.service
             └─629 /usr/bin/gitlab-runner run --config /etc/gitlab-runner/config.toml --w

avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]:   - Request bottleneck: 1 runners have
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: This can cause job delays matching your
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: Recommended solutions:
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]:   1. Increase 'request_concurrency' to
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: Note: The 'FF_USE_ADAPTIVE_REQUEST_CONC
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: This message will be printed each time
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: See documentation: https://docs.gitlab.
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: listen_address not defined, metrics & d
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: [session_server].listen_address not def
avril 12 17:47:33 SRV-P-ANS01 gitlab-runner[629]: Initializing executor providers
```

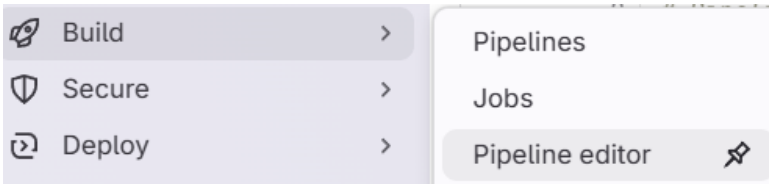
Il est également possible de voir sur GitLab si le runner est correctement configuré en allant dans les « Runners », celui configuré précédemment devrait être en « Online »



Avec cette dernière vérification, le runner est complètement configuré.

7) Configuration .gitlab-ci.yml

Avec l'ensemble de la configuration des étapes précédentes, tout le processus devrait être fonctionnel, il reste uniquement à réaliser la configuration du fichier .gitlab-ci.yml, pour cela, il faut aller sur le projet « ansible-oasis » dans GitLab et aller dans « Build » puis « Pipeline editor ».



Dans ce fichier, il y a différents blocs :

Stage : Définis l'ordre d'exécution des tâches

Tags : Force l'exécution sur le runner mis en place précédemment, qui porte le tag Debian

deploy : Contient du scripting Bash avec une première condition permettant de définir si c'est une machine virtuelle ou un équipement physique (les URLs changent), une seconde partie permettant de récupérer les tags et de définir l'inventaire à utiliser, production, development ou staging et une dernière partie permettant de lancer les tags sur la machine cible par rapport au playbook définit dans Ansible

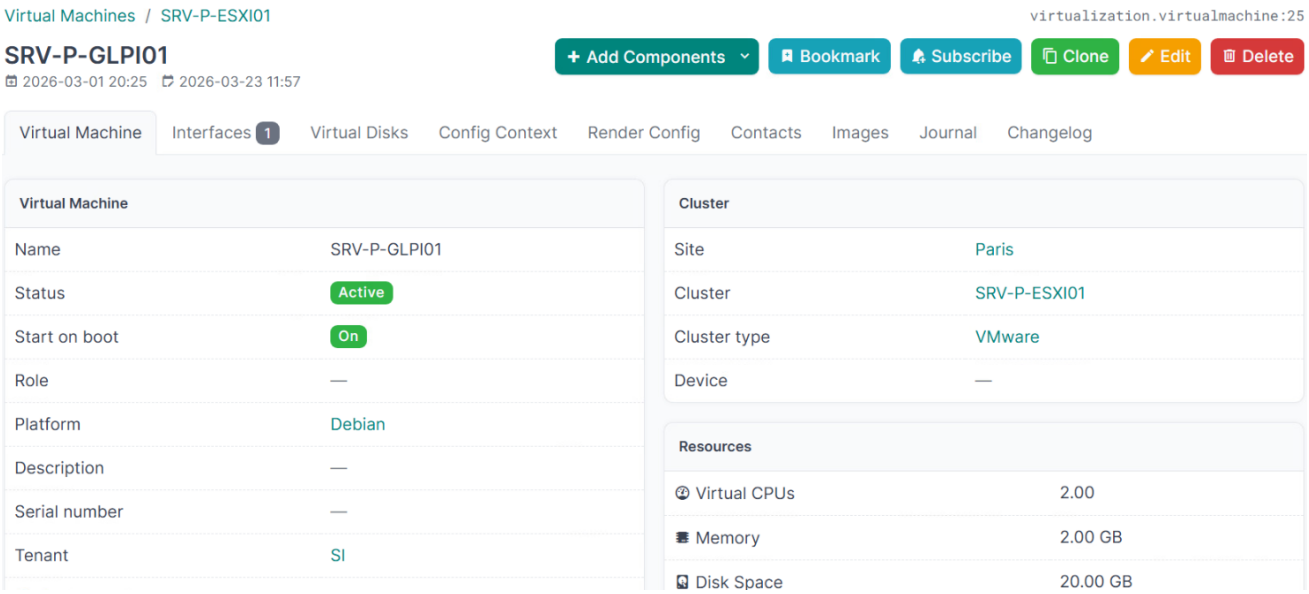
set-active : Contient du scripting bash avec une première condition permettant de définir si c'est une machine virtuelle ou un équipement physique (les urls changent) puis une seconde partie qui permet de passer le statut de la machine cible en « active » au lieu de « staged ».

```
stages:
  - deploy
  - finalize

# ================================================================================
# ÉTAPE 1 — Récupère les tags depuis Netbox et lance les playbooks Ansible
# ================================================================================
deploy:
  stage: deploy
  tags:
    - debian
  script:
    - |
      # Choisit le bon endpoint selon le type d'objet
      if [ "$OBJECT_TYPE" = "virtualization.virtualmachine" ]; then
        API_URL="$NETBOX_URL/api/virtualization/virtual-machines/$DEVICE_ID/"
      else
        API_URL="$NETBOX_URL/api/dcim/devices/$DEVICE_ID/"
      fi

      # Récupère les tags du device depuis Netbox
      TAGS=$(curl -s -k -H "Authorization: Token $NETBOX_TOKEN" "$API_URL" | jq -r '.tags[].slug')

      echo "Device : $DEVICE_NAME"
      echo "Tags   : $TAGS"
```

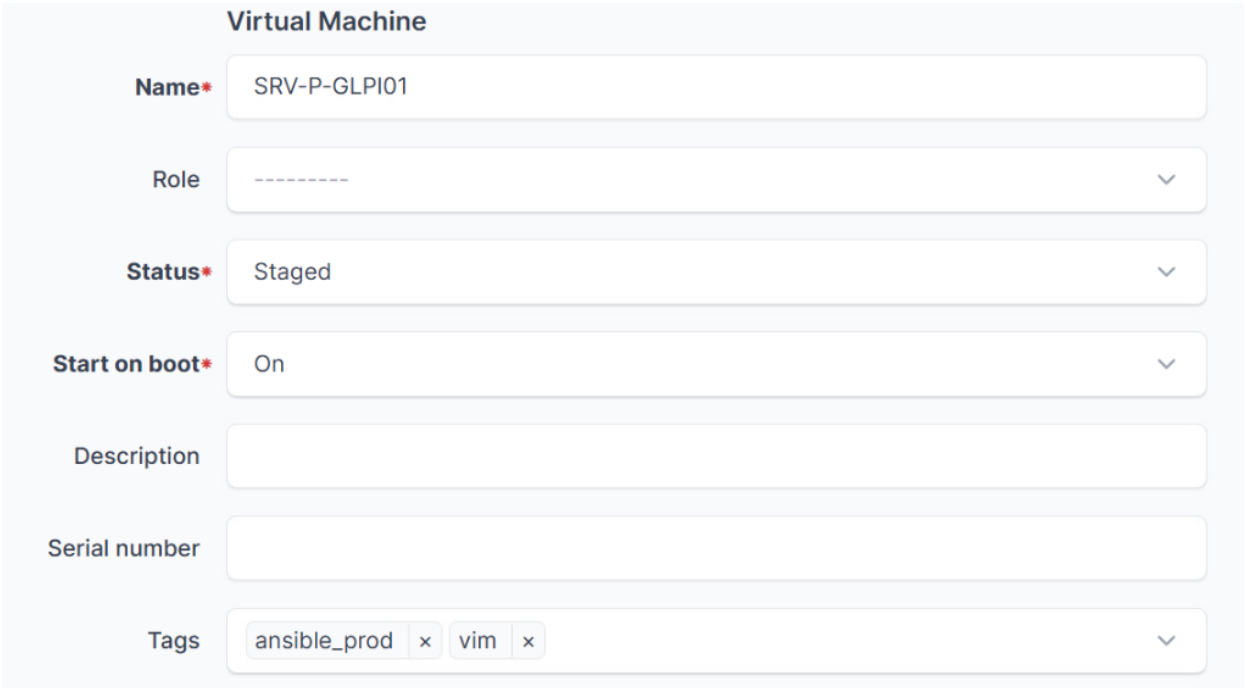
```
      # Choisit l'inventaire selon le tag d'environnement
      if echo "$TAGS" | grep -q "ansible_prod"; then
        INVENTORY="inventories/production/netbox.yml"
      elif echo "$TAGS" | grep -q "ansible_staging"; then
        INVENTORY="inventories/staging/netbox.yml"
      else
        INVENTORY="inventories/development/netbox.yml"
      fi

      echo "Inventaire : $INVENTORY"

      # Lance le playbook correspondant à chaque tag
      for TAG in $TAGS; do
        PLAYBOOK="playbooks/deploy_${TAG}.yml"
        if [ -f "$PLAYBOOK" ]; then
          echo "Lancement de $PLAYBOOK..."
          ansible-playbook "$PLAYBOOK" -i "$INVENTORY" -l "$DEVICE_NAME"
        else
          echo "Pas de playbook pour le tag '$TAG', ignoré."
        fi
      done

set-active:
  stage: finalize
  tags:
    - debian
  script:
    - |
      if [ "$OBJECT_TYPE" = "virtualization.virtualmachine" ]; then
        API_URL="$NETBOX_URL/api/virtualization/virtual-machines/$DEVICE_ID/"
      else
        API_URL="$NETBOX_URL/api/dcim/devices/$DEVICE_ID/"
      fi

      curl -s -k -X PATCH "$API_URL" \
        -H "Authorization: Token $NETBOX_TOKEN" \
        -H "Content-Type: application/json" \
        -d '{"status": "active"}'

      echo "Machine $DEVICE_NAME repassée en active sur Netbox."
  when: on_success
```

8) Lancement d'un playbook via Netbox

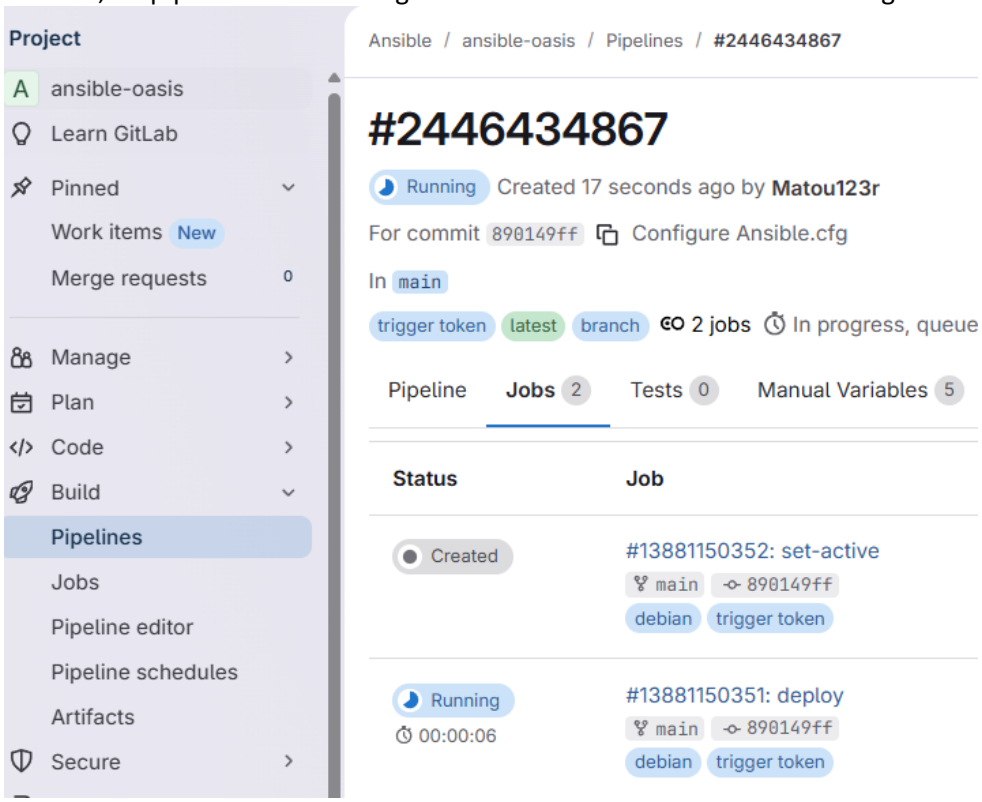
Pour le lancement d'un playbook via Netbox, c'est plutôt simple, il faut aller sur une machine virtuelle ou un device sous Debian sur l'outil.



Ensuite, il faut modifier la machine concernée dans NetBox et lui ajouter le tag correspondant au playbook à exécuter, tout en conservant le tag « ansible_prod », « ansible_dev » ou « ansible_staging », selon l'environnement d'utilisation de la machine. Il faut ensuite passer la machine en statut « staged ». Dans l'exemple ci-dessous, le tag « vim » est utilisé.

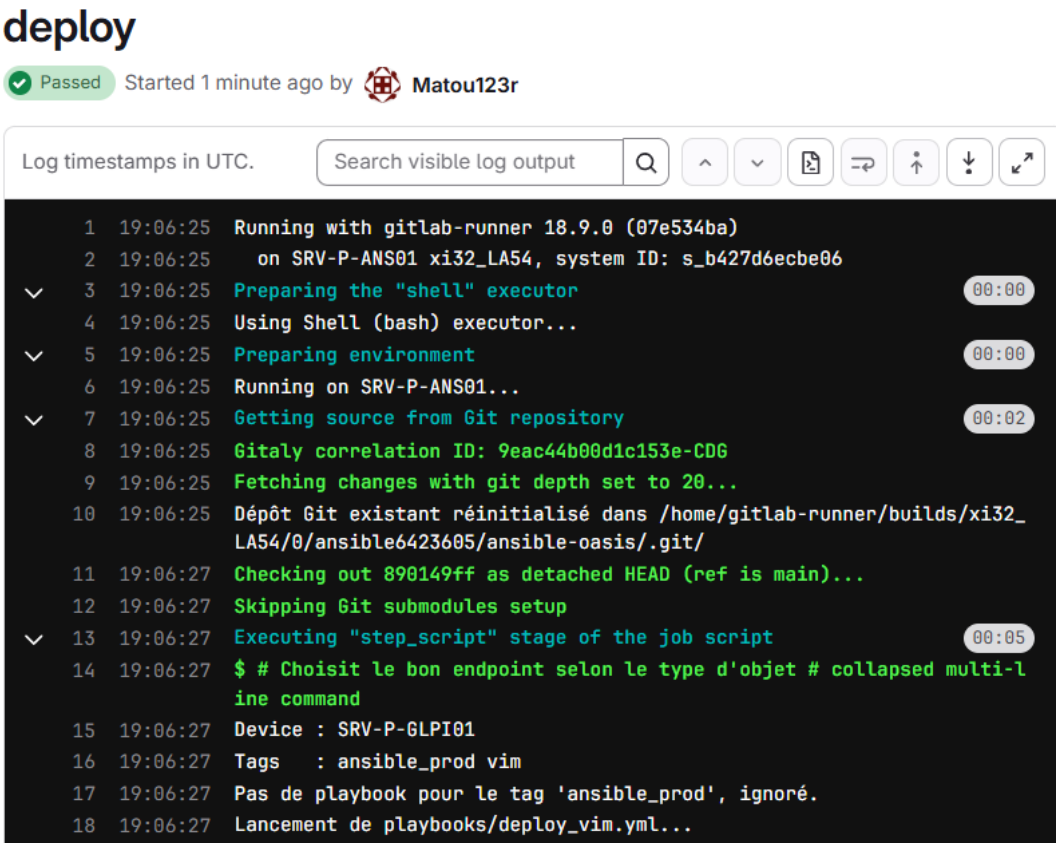


Une fois cela effectué, un pipeline est exécuté grâce au déclencheur d'évènement configuré dans Netbox.

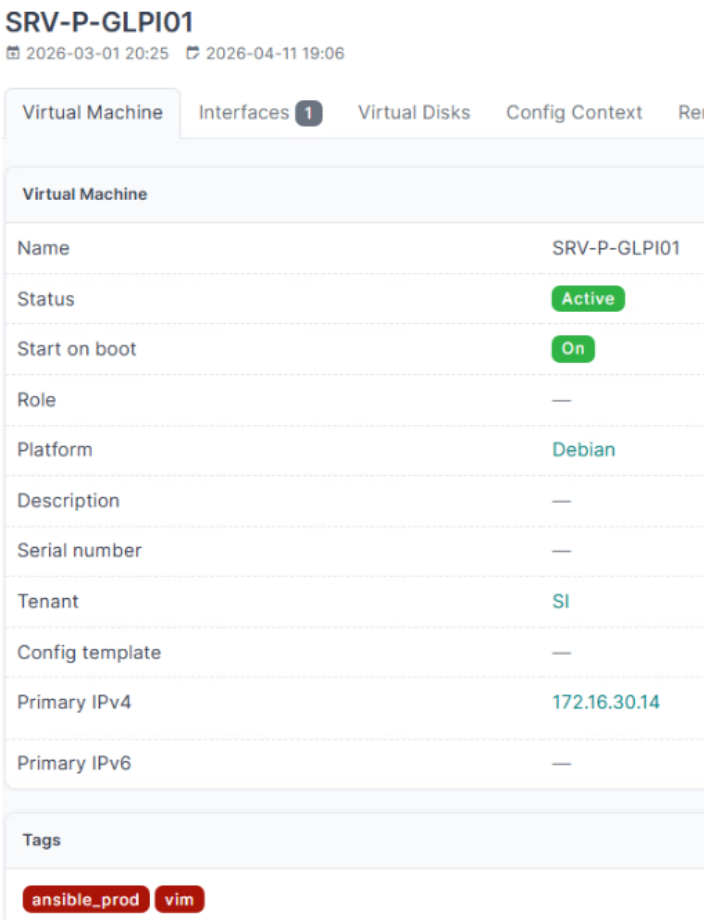


Dans ce pipeline, il est possible de suivre l'exécution de la tâche Ansible, de vérifier qu'elle s'exécute correctement et que les différentes étapes se déroulent comme prévu.

Une fois le pipeline terminé, si celui-ci s'est correctement déroulé, il passera en statut « Passed ».



Il est également possible de constater que la machine sur laquelle le pipeline a été exécuté passe en statut « Active » sur NetBox au lieu de « Staged », tout en conservant le ou les tags qui lui ont été attribués.


SRV-P-GLPI01
2026-03-01 20:25
2026-04-11 19:06
Virtual Machine
Interfaces 1
Virtual Disks
Config Context
Virtual Machine
Name
SRV-P-GLPI01
Status
Active
Start on boot
On
Role
—
Platform
Debian
Description
—
Serial number
—
Tenant
SI
Config template
—
Primary IPv4
172.16.30.14
Primary IPv6
—
Tags
ansible_prod
vim

m) Axes d’amélioration

Pour ce projet, je pense qu'il y a différents axes d'amélioration. Tout d'abord, j'aurais pu implémenter un stage de lint dans le pipeline GitLab afin de bloquer toute exécution de playbook si la qualité du code n'est pas satisfaisante, ou simplement afficher un avertissement dans ce cas.

Il aurait également été possible de mettre en place un inventaire dynamique avec des filtres plus précis, c'est-à-dire de configurer un filtrage permettant de lancer des playbooks en fonction des rôles des machines définis dans NetBox, afin d'exécuter chaque playbook de façon simplifiée selon le besoin.

Enfin, il aurait été possible de mettre en place un système de clés SSH plus sécurisé, en configurant une clé SSH individuelle par machine ou par rôle, ou encore en attribuant une clé SSH par utilisateur plutôt que d'utiliser une machine dédiée à Ansible, ou en créant un utilisateur dédié aux déploiements au lieu d'utiliser root.

n) Conclusion

En conclusion, ce projet m'a permis d'apprendre de nombreuses choses. J'ai pu réaliser des tâches qui m'ont permis d'acquérir de nouvelles compétences sur l'automatisation et l'utilisation de GitLab de façon générale.

Au niveau des difficultés rencontrées, j'ai éprouvé des difficultés lors de la configuration de l'inventaire dynamique. En effet, il m'a été difficile au départ de comprendre la structure générale d'Ansible et comment construire un rôle proprement afin que celui-ci soit idempotent.

J'ai également rencontré des difficultés lors de la mise en place du lancement de playbooks via NetBox. Il m'a été difficile de mettre en place cette fonctionnalité car c'est quelque chose que je n'avais jamais fait auparavant. De plus, je n'étais pas forcément familier avec les différents concepts de GitLab, ce qui m'a rendu la tâche complexe.

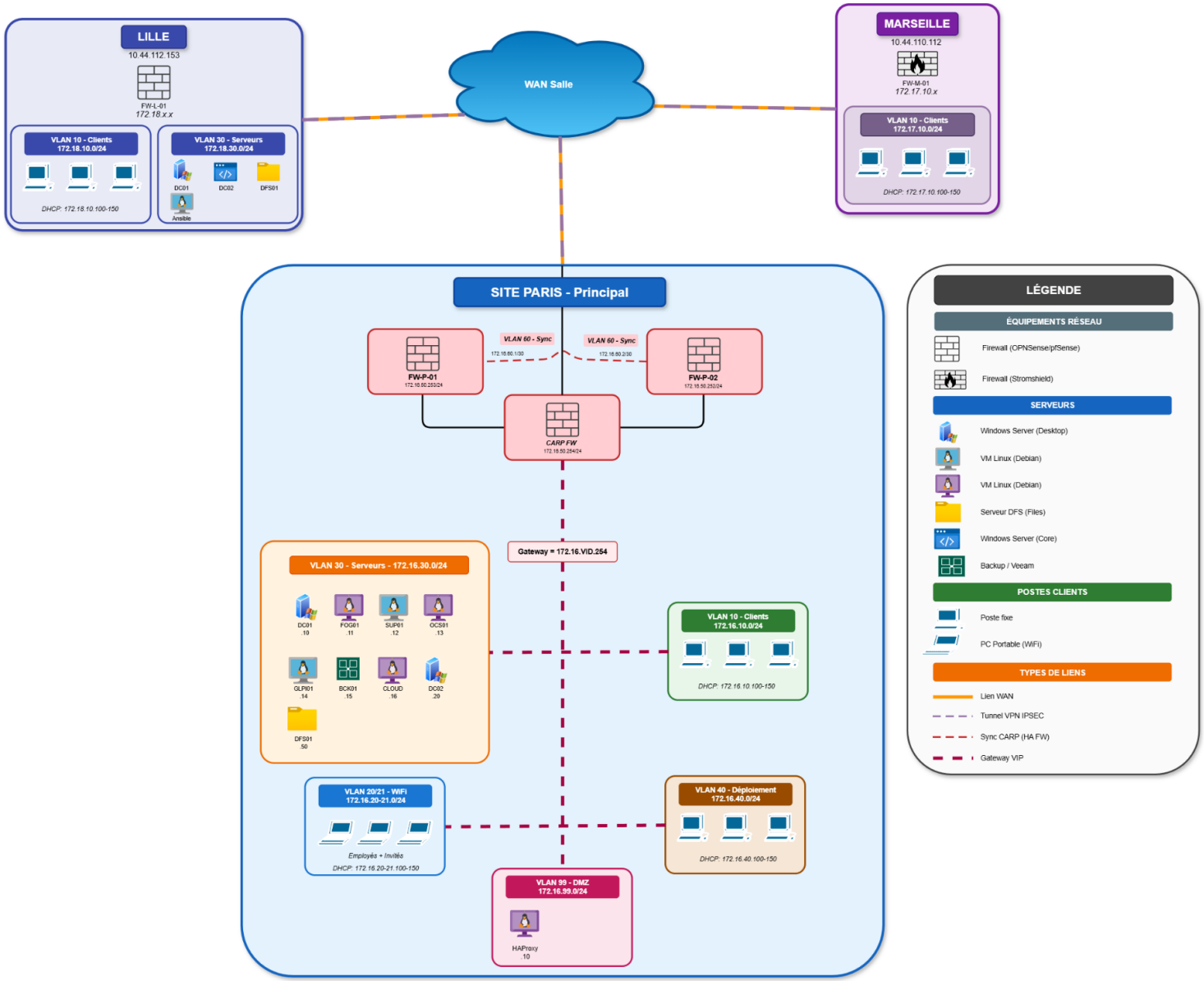
Malgré ces difficultés, je pense avoir répondu au cahier des charges ainsi qu'à l'objectif de la mission qui m'a été confiée, en mettant en place une solution permettant de réaliser de l'automatisation sur l'ensemble des machines Linux.

Pour terminer, ce projet m’a permis de valider plusieurs compétences du référentiel BTS SIO Option SISR :

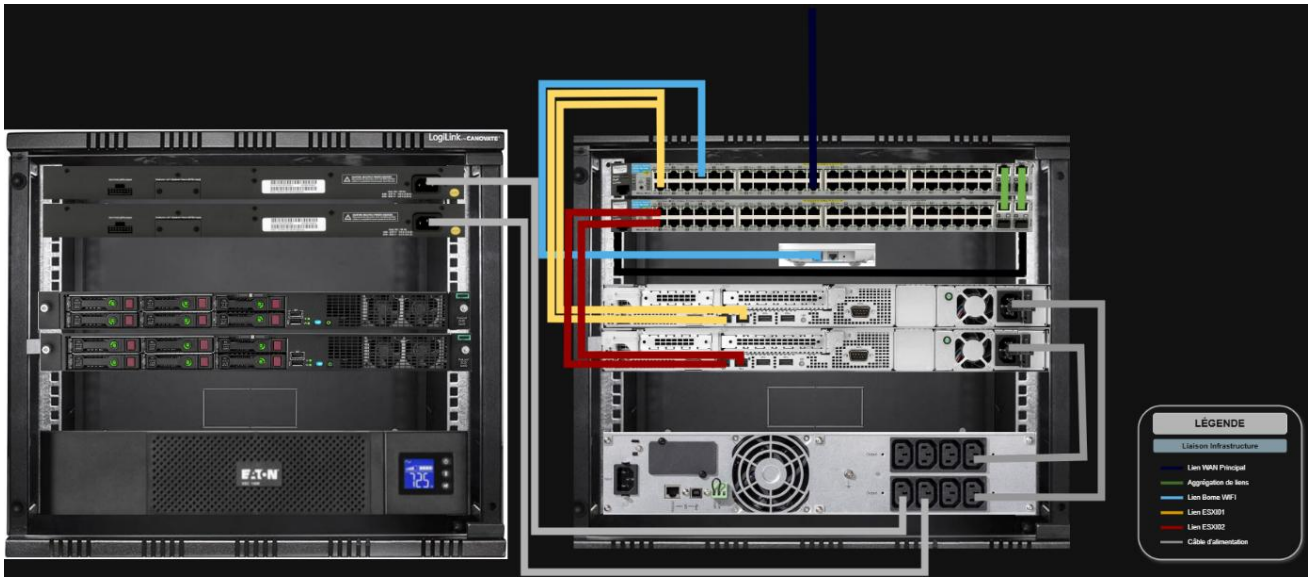
- Gérer le patrimoine informatique
- Répondre aux incidents et aux demandes d'assistance et d'évolution
- Travailler en mode projet
- Mettre à disposition des utilisateurs un service informatique

Annexes

a) Annexe n°1 : Schéma logique



b) Annexe n°2 : Schéma physique



c) Annexe n°3 : Plan d'adressage

VLAN 10

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-02	172.16.10.252	255.255.255.0	172.16.10.0	172.16.10.254	IP FW-P-01 VLAN 10
FW-P-01	172.16.10.253	255.255.255.0	172.16.10.0	172.16.10.254	IP FW-P-02 VLAN 10
CARP Firewall	172.16.10.254	255.255.255.0	172.16.10.0	172.16.10.254	Passerelle du VLAN 10

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.10.100-150	172.16.10.254	172.16.30.10	172.16.30.20	Plage DHCP Client Paris

VLAN 20

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
B-P-WIFI	172.16.20.50	255.255.255.0	172.16.20.0	172.16.20.254	Administration borne Wifi
FW-P-02	172.16.20.252	255.255.255.0	172.16.20.0	172.16.20.254	IP FW-P-02 VLAN 20
FW-P-01	172.16.20.253	255.255.255.0	172.16.20.0	172.16.20.254	IP FW-P-01 VLAN 20
CARP Firewall	172.16.20.254	255.255.255.0	172.16.20.0	172.16.20.254	Passerelle du VLAN 20

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.20.100-150	172.16.20.254	172.16.30.10	172.16.30.20	Plage DHCP WIFI Employés

VLAN 21

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-02	172.16.21.252	255.255.255.0	172.16.21.0	172.16.21.254	IP FW-P-02 VLAN 21
FW-P-01	172.16.21.253	255.255.255.0	172.16.21.0	172.16.21.254	IP FW-P-01 VLAN 21
CARP Firewall	172.16.21.254	255.255.255.0	172.16.21.0	172.16.21.254	Passerelle du VLAN 21

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.21.100-150	172.16.21.254	172.16.30.10	172.16.30.20	Plage DHCP WIFI Invité

VLAN 30

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SRV-P-DC01	172.16.30.10	255.255.255.0	172.16.30.0	172.16.30.254	DC 1
SRV-P-DC02	172.16.30.20	255.255.255.0	172.16.30.0	172.16.30.254	DC 2
SRV-P-DFS01	172.16.30.50	255.255.255.0	172.16.30.0	172.16.30.254	DFS01
SRV-P-FOG01	172.16.30.11	255.255.255.0	172.16.30.0	172.16.30.254	Fog
SRV-P-OCS01	172.16.30.13	255.255.255.0	172.16.30.0	172.16.30.254	OCS Inventory
SRV-P-GLPI01	172.16.30.14	255.255.255.0	172.16.30.0	172.16.30.254	GLPI
SRV-P-BCK01	172.16.30.15	255.255.255.0	172.16.30.0	172.16.30.254	Veeam
SRV-P-CLOUD01	172.16.30.16	255.255.255.0	172.16.30.0	172.16.30.254	Nextcloud
SRV-P-RSAT-T0	172.16.30.30	255.255.255.0	172.16.30.0	172.16.30.254	RSAT T0
SRV-P-RSAT-T1	172.16.30.31	255.255.255.0	172.16.30.0	172.16.30.254	RSAT T1
SRV-P-RSAT-T2	172.16.30.32	255.255.255.0	172.16.30.0	172.16.30.254	RSAT T2
SRV-P-EDR01	172.16.30.19	255.255.255.0	172.16.30.0	172.16.30.254	EDR
SRV-P-ANS01	172.16.30.21	255.255.255.0	172.16.30.0	172.16.30.254	Ansible Lille
SRV-P-NETBOX01	172.16.30.22	255.255.255.0	172.16.30.0	172.16.30.254	Outil d'infrastructure
SRV-P-POL01	172.16.30.25	255.255.255.0	172.16.30.0	172.16.30.254	Centreon Poller
FW-P-02	172.16.30.252	255.255.255.0	172.16.30.0	172.16.30.254	IP FW-P-02 VLAN 30
FW-P-01	172.16.30.253	255.255.255.0	172.16.30.0	172.16.30.254	IP FW-P-01 VLAN 30
CARP Firewall	172.16.30.254	255.255.255.0	172.16.30.0	172.16.30.254	Passerelle du VLAN 30

VLAN 40

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-02	172.16.40.252	255.255.255.0	172.16.40.0	172.16.40.254	IP FW-P-02 VLAN 40
FW-P-01	172.16.40.253	255.255.255.0	172.16.40.0	172.16.40.254	IP FW-P-01 VLAN 40
CARP Firewall	172.16.40.254	255.255.255.0	172.16.40.0	172.16.40.254	Passerelle du VLAN 40

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.16.40.100-150	172.16.40.254	172.16.30.10	172.16.30.20	Plage DHCP Déploiement

Proxmox Matt (Lille)

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SRV-L-DC01	172.18.30.10	255.255.255.0	172.18.30.0	172.18.30.254	DC1 Lille
SRV-L-DC02	172.18.30.20	255.255.255.0	172.18.30.0	172.18.30.254	DC2 Core Lille
SRV-L-ANS01	172.18.30.15	255.255.255.0	172.18.30.0	172.18.30.254	Ansible Lille
SRV-L-NETBOX01	172.18.30.30	255.255.255.0	172.18.30.0	172.18.30.254	Netbox Infrastructure
FW-L-01	172.18.10.254	255.255.255.0	172.18.10.0	172.18.10.254	IP FW-L-01 LAN Lille
FW-L-02	172.18.10.252	255.255.255.0	172.18.10.0	172.18.10.254	IP FW-L-02 LAN Lille
CARP Firewall Lille	172.18.10.254	255.255.255.0	172.18.10.0	172.18.10.254	Passerelle du VLAN 10
FW-L-01	172.18.30.254	255.255.255.0	172.18.30.0	172.18.30.254	IP FW-L-01 SRV Lille
FW-L-02	172.18.30.252	255.255.255.0	172.18.30.0	172.18.30.254	IP FW-L-02 SRV Lille
CARP Firewall Lille	172.18.30.254	255.255.255.0	172.18.30.0	172.18.30.254	Passerelle du VLAN 30
FW-L-01	172.18.60.1	255.255.255.252	172.18.60.0	-	IP FW-L-01 SYNC_OPN
FW-L-02	172.18.60.2	255.255.255.252	172.18.60.0	-	IP FW-L-02 SYNC_OPN
FW-L-01	172.18.99.254	255.255.255.0	172.18.99.0	172.18.99.254	IP FW-L-01 DMZ Lille
FW-L-02	172.18.99.252	255.255.255.0	172.18.99.0	172.18.99.254	IP FW-L-02 DMZ Lille
CARP Firewall Lille	172.18.99.254	255.255.255.0	172.18.99.0	172.18.99.254	Passerelle du VLAN 99
FW-L-01	10.44.115.50	255.255.255.0	10.44.115.0	10.44.115.254	IP WAN Lille
FW-L-02	10.44.112.100	255.255.255.0	10.44.112.0	10.44.112.254	IP FW-L-02 WAN
CARP Firewall Lille	10.44.112.153	255.255.255.0	10.44.112.0	10.44.112.254	CARP WAN

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	SS	172.18.10.254	172.18.30.10	172.18.30.20	Plage DHCP Client Lille

VLAN 50

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SW-P-01	172.16.50.1	255.255.255.0	172.16.50.0	172.16.50.254	VLAN 50 Switch 1 Paris
SW-P-02	172.16.50.2	255.255.255.0	172.16.50.0	172.16.50.254	VLAN 50 Switch 2 Paris
SRV-P-ESXI01	172.16.50.10	255.255.255.0	172.16.50.0	172.16.50.254	IP d'administration hyperviseur
SRV-P-ESXI02	172.16.50.20	255.255.255.0	172.16.50.0	172.16.50.254	IP d'administration hyperviseur
PAW-P-T0	172.16.50.50	255.255.255.0	172.16.50.0	172.16.50.254	Machine d'administration
FW-P-02	172.16.50.252	255.255.255.0	172.16.50.0	172.16.50.254	IP FW-P-02 VLAN 50
FW-P-01	172.16.50.253	255.255.255.0	172.16.50.0	172.16.50.254	IP FW-P-01 VLAN 50
CARP Firewall	172.16.50.254	255.255.255.0	172.16.50.0	172.16.50.254	Passerelle du VLAN 50

VLAN 60

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-P-01	172.16.60.1	255.255.255.252	172.16.60.0	-	IP FW-P-01 VLAN 60
FW-P-02	172.16.60.2	255.255.255.252	172.16.60.0	-	IP FW-P-02 VLAN 60

VLAN 99

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
SRV-P-HAProxy	172.16.99.10	255.255.255.0	172.16.99.0	172.16.99.254	HAProxy
FW-P-02	172.16.99.252	255.255.255.0	172.16.99.0	172.16.99.254	IP FW-P-02 VLAN 99
FW-P-01	172.16.99.253	255.255.255.0	172.16.99.0	172.16.99.254	IP FW-P-01 VLAN 99
CARP Firewall	172.16.99.254	255.255.255.0	172.16.99.0	172.16.99.254	Passerelle du VLAN 99

Marseille

Nom Machine	IP	MSR	Adresse Réseau	Passerelle	Description
FW-M-01	172.17.10.254	255.255.255.0	172.17.10.0	172.17.10.254	IP FW-M-01 VLAN 10 Marseille
FW-M-01	10.44.110.112	255.255.255.0	10.44.110.0	10.44.110.254	IP WAN Marseille

DHCP	Plage	Passerelle	DNS1	DNS2	Description
	172.17.10.100-150	172.17.10.254	172.16.30.10	172.16.30.20	Plage DHCP Client Marseille

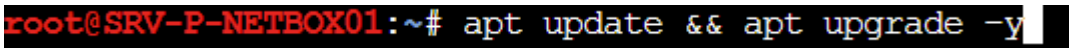
d) Annexe 4 : Mise en place de NetBox

1) Configuration base de données Netbox

Netbox est un outil permettant de faire de la gestion d'infrastructure avec Ansible. Nous pouvons installer cet outil sur Debian 12.

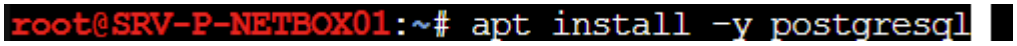
Dans un premier temps, avant d'installer l'outil, il faut mettre à jour les paquets sur la machine concernée comme ci-dessous.

```
apt update && apt upgrade -y
```



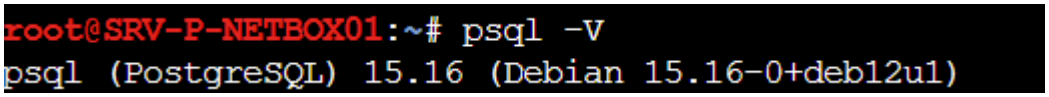
Une fois cela fait, nous pouvons installer le paquet PostgreSQL car c'est ce type de base de données qui est utilisé par Netbox.

```
apt install -y postgresql
```



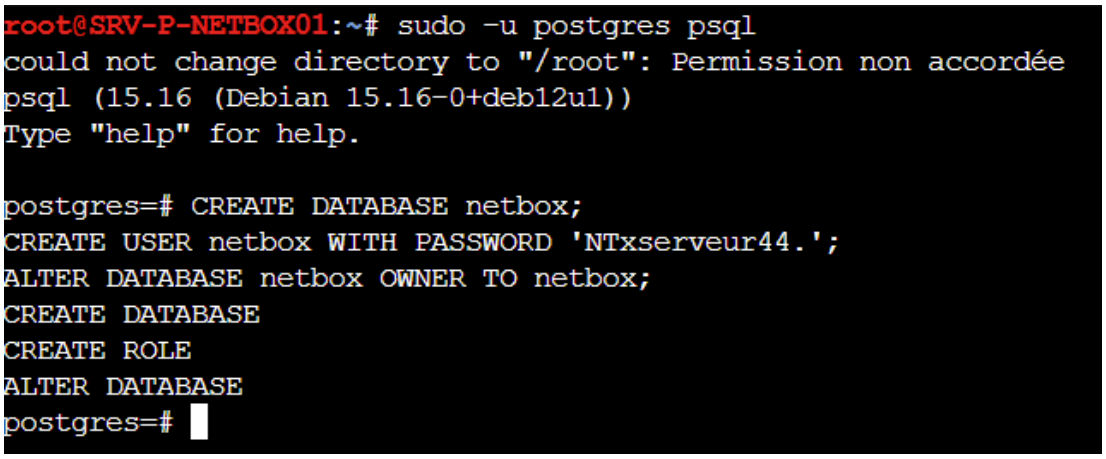
Une fois le paquet installé, nous pouvons vérifier la version de celui-ci, il faut que PostgresSQL soit en version 15.x sous Debian 12 et 17.x sous Debian 13. Dans notre cas, nous sommes sous Debian 12, c'est pour cela que l'on est en 15.x

```
psql -V
```



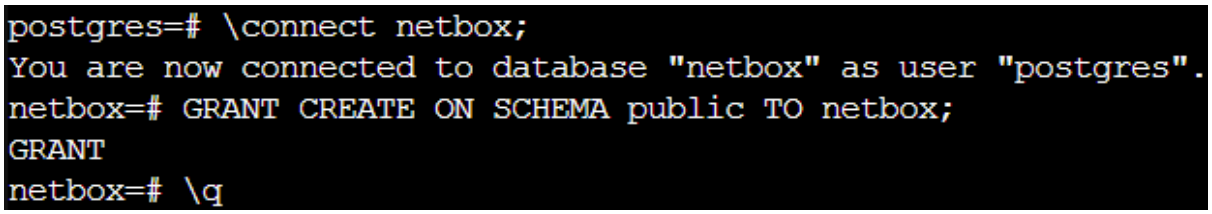
Après avoir correctement installé PostgreSQL, il faut procéder à la création de la base de données pour NetBox, lui attribuer un nom d'utilisateur ainsi qu'un mot de passe pour l'authentification comme ci-dessous. Le mot de passe donné ci-dessous est un mot de passe fictif.

```
sudo -u postgres psql
CREATE DATABASE netbox;
CREATE USER netbox WITH PASSWORD 'NTxserveur44.';
ALTER DATABASE netbox OWNER TO netbox;
```



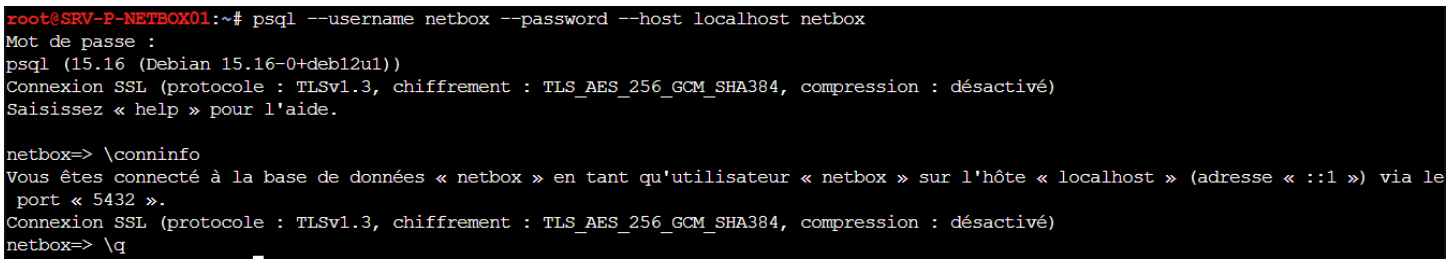
Une fois la base de données créée, nous pouvons nous connecter dessus puis donner des privilèges de création à l'utilisateur NetBox comme sur l'image ci-dessous.

```
\connect netbox;
GRANT CREAT ON SCHEMA public TO netbox;
\q
```



Après avoir fait ces différentes étapes sur la base de données, il faut vérifier l'accès à la base de données avec l'utilisateur NetBox qui a été créé et vérifier les informations de connexion de l'utilisateur comme ci-dessous.

```
psql --username netbox --password --host localhost netbox
\conninfo
\q
```



2) Configuration Redis Netbox

Redis est un outil essentiel pour le bon fonctionnement de l'application Netbox, c'est un système de stockage clé-valeur en mémoire utilisé par Netbox pour la mise en cache et la gestion des files d'attente.

Il y a différentes étapes pour l'installation de cet outil, tout d'abord, il faut installer le paquet correspondant à Redis.

```
apt install -y redis-server
```

```
root@SRV-P-NETBOX01:~# apt install -y redis-server
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Le paquet suivant a été installé automatiquement et n'est plus nécessaire :
  linux-image-6.1.0-27-amd64
Veuillez utiliser « apt autoremove » pour le supprimer.
Les paquets supplémentaires suivants seront installés :
  libatomic1 liblzf1 redis-tools
```

Une fois le paquet installé, il faut vérifier que la version de redis est supérieure à la v4.0 car Netbox ne supporte pas les versions antérieures. Puis utiliser l'utilitaire pour s'assurer que le service est fonctionnel. Nous pouvons voir que l'outil répond « PONG », cela confirme le bon fonctionnement de l'outil.

```
redis-server -V
redis-cli ping
```

```
root@SRV-P-NETBOX01:~# redis-server -v
Redis server v=7.0.15 sha=00000000:0 malloc=jemalloc-5.3.0 bits=64 build=5281cccdf7ef82d6
root@SRV-P-NETBOX01:~# redis-cli ping
PONG
root@SRV-P-NETBOX01:~#
```

3) Installation de NetBox

Maintenant que l'installation des outils permettant le fonctionnement de l'application Netbox est terminée, nous pouvons passer à l'installation de l'application elle-même. Pour cela, il faut commencer par installer tous les services utilisés par NetBox comme ci-dessous. Attention, la version de python3 qui doit être installée est la version python3.12, 3.13 ou 3.14.

```
apt install -y python3 python3-pip python3-venv python3-dev \ build-essential libxml2-dev libxslt1-dev libffi-dev libpq-dev \ libssl-dev zlib1g-dev
```

```
root@SRV-P-NETBOX01:~# apt install -y python3 python3-pip python3-venv python3-dev \
build-essential libxml2-dev libxslt1-dev libffi-dev libpq-dev \
libssl-dev zlib1g-dev
```

Une fois cela fait, il faut créer les répertoires permettant de réaliser l'installation de Netbox, dans notre cas, /opt/netbox, puis installer le paquet git afin de pouvoir cloner le dépôt de Netbox dans un second temps.

```
mkdir -p /opt/netbox
cd /opt/netbox
apt install -y git
```

```
root@SRV-P-NETBOX01:~# mkdir -p /opt/netbox/
root@SRV-P-NETBOX01:~# cd /opt/netbox/
root@SRV-P-NETBOX01:/opt/netbox# apt install -y git
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Le paquet suivant a été installé automatiquement et n'est plus nécessaire :
  linux-image-6.1.0-27-amd64
Veuillez utiliser « apt autoremove » pour le supprimer.
Les paquets supplémentaires suivants seront installés :
  git-man liberror-perl
Paquets suggérés :
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
Les NOUVEAUX paquets suivants seront installés :
  git git-man liberror-perl
0 mis à jour, 3 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 9 347 ko dans les archives.
Après cette opération, 48,2 Mo d'espace disque supplémentaires seront utilisés.
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 liberror-perl all 0.17029-2 [29,0 kB]
Réception de :2 http://deb.debian.org/debian bookworm/main amd64 git-man all 1:2.39.5-0+deb12u3 [2 053 kB]
Réception de :3 http://deb.debian.org/debian bookworm/main amd64 git amd64 1:2.39.5-0+deb12u3 [7 264 kB]
59% [3 git 3 251 kB/7 264 kB 45%]
```

Après avoir installé le paquet Git, nous pouvons cloner le répertoire de NetBox, comme ci-dessous.

```
git clone https://github.com/netbox-community/netbox.git .
```

```
root@SRV-P-NETBOX01:/opt/netbox#
root@SRV-P-NETBOX01:/opt/netbox# git clone https://github.com/netbox-community/netbox.git .
```

Une fois cela fait, nous pouvons choisir la version à installer en changeant de branche sur git, dans notre cas, nous avons choisi d'installer la dernière version current, la version 4.5.3, comme ci-dessous.

```
git switch v4.5.3
```

```
root@SRV-P-NETBOX01:/opt/netbox# git switch v4.5.3
```

Ensuite, il faut créer l'utilisateur NetBox qui sera utilisé pour les différents services de l'application puis lui donner des droits sur certains dossiers de l'outil.

```
adduser --system --group netbox
chown --recursive netbox /opt/netbox/netbox/media/
chown --recursive netbox /opt/netbox/netbox/reports/
chown --recursive netbox /opt/netbox/netbox/scripts/
```

```
root@SRV-P-NETBOX01:/opt/netbox# sudo adduser --system --group netbox
sudo chown --recursive netbox /opt/netbox/netbox/media/
sudo chown --recursive netbox /opt/netbox/netbox/reports/
sudo chown --recursive netbox /opt/netbox/netbox/scripts/
```

Par la suite, il faut aller dans le répertoire de configuration Netbox afin de réaliser certaines modifications au niveau du script Python. Tout d'abord, il faut copier le script de configuration afin de le modifier.

```
cd /opt/netbox/netbox/netbox/
cp configuration_example.py configuration.py
```

Une fois le script dupliqué, il faut modifier quelques paramètres d'installation, dans notre cas, avec l'éditeur vim.

```
vim configuration.py
```

Dans un premier temps, le paramètre « ALLOWED_HOST », celui-ci correspond à l'adresse IP de Netbox, dans notre cas, nous mettons le nom FQDN (Fully Qualified Domain Name) de Netbox.

```
#########################
#                       #
#   Required settings   #
#                       #
#########################

# This is a list of valid fully-qualified domain n
server. NetBox will not permit write
# access to the server via any other hostnames. Th
be treated as the preferred name.
#
# Example: ALLOWED_HOSTS = ['netbox.example.com',
ALLOWED_HOSTS = ['netbox.oasis.local']
```

Ensuite, il faut configurer les paramètres correspondant à la base de données Netbox dans le même fichier, le nom d'utilisateur, son mot de passe, le nom de la base et le port de connexion s'il a été modifié. Ci-dessous la configuration réalisée dans le fichier.

```
# PostgreSQL database configuration. See the Django documentation for a complet
list of available parameters:
#   https://docs.djangoproject.com/en/stable/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',  # Database engine
        'NAME': 'netbox',                           # Database name
        'USER': 'netbox',                           # PostgreSQL username
        'PASSWORD': 'NTxserveur44.',                # PostgreSQL password
        'HOST': 'localhost',                        # Database server
        'PORT': '',                                 # Database port (leave blank for default)
        'CONN_MAX_AGE': 300,                        # Max database connection age
    }
}
```

Une fois ces deux étapes réalisées, nous devons générer la clé qui va être utilisée par NetBox, celle-ci peut être générée via un script python qui est dans le même dossier que le fichier configuration, une fois générée, il faut copier la clé affichée à l'écran puis la mettre dans le fichier configuration.py comme ci-dessous. La clé utilisée est une clé de test, celle-ci n'est actuellement pas utilisé en production

```
python3 ../generate_secret_key.py
```

```
root@SRV-P-NETBOX01:/opt/netbox/netbox/netbox# python3 ../generate_secret_key.py
VGUD_jy#$IlR^Zx+_x04tvFI&%cJT$2XA9iY2#8WRhlSs+PGbA
root@SRV-P-NETBOX01:/opt/netbox/netbox/netbox#
```

```
# https://docs.djangoproject.com/en/stable/ref/settings/#std:setting-SECRET_K
SECRET_KEY = 'VGUD_jy#$IlR^Zx+_x04tvFI&%cJT$2XA9iY2#8WRhlSs+PGbA'
```

Maintenant que le script a la bonne configuration, nous pouvons procéder à l'installation de l'outil en exécutant le script « upgrade.sh » qui va réaliser toute l'installation de Netbox. Ce script doit être exécuté avec python3.12 installé au préalable.

```
/opt/netbox/upgrade.sh
```

```
root@SRV-P-NETBOX01:/opt/netbox/netbox/netbox# /opt/netbox/upgrade.sh
```

Après que le script s'est terminé, il faut créer un super utilisateur pour pouvoir se connecter à Netbox, par défaut, Netbox ne crée pas de comptes utilisateur. Pour cela, il faut rentrer dans l'environnement virtuel Python créé par le script « upgrade.sh ».

```
source /opt/netbox/venv/bin/activate
```

```
root@SRV-P-NETBOX01:/opt/netbox/netbox/netbox# source /opt/netbox/venv/bin/activate
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox/netbox#
```

Ensuite, il faut aller dans le dossier où se trouve le script pour la gestion des utilisateurs puis l'exécuter et suivre les étapes de création, comme ci-dessous.

```
cd /opt/netbox/netbox
python3 manage.py createsuperuser
```

```
(venv) root@SRV-P-NETBOX01:~# cd /opt/netbox/netbox/
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox#
```

```
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox# python3 manage.py createsuperuser
/opt/netbox/netbox/netbox/settings.py:226: UserWarning: API_TOKEN_PEPPERS is not defined. v2 API
tokens cannot be used.
  warnings.warn("API_TOKEN_PEPPERS is not defined. v2 API tokens cannot be used.")
Username: root
Email address:
Password:
Password (again):
Superuser created successfully.
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox#
```

4) Installation Gunicorn

Une fois cela fait, nous allons passer à la configuration de Gunicorn qui va nous permettre de faire le pont entre notre serveur web (apache2) et l'application Netbox.

Pour cela, il faut copier le script gunicorn.py dans le dossier directement utilisé par NetBox puis copier le service dans le système afin qu'il soit considéré directement comme un service système comme ci-dessous. Après avoir fait ces deux étapes, il faut redémarrer le daemon.

```
cp /opt/netbox/contrib/gunicorn.py /opt/netbox/gunicorn.py
cp -v /opt/netbox/contrib/*.service /etc/systemd/system/ sudo systemctl daemon-reload
```

```
^C(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox cp /opt/netbox/contrib/gunicorn.py /opt/netbox/gunicorn.pypy
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox#  cp -v /opt/netbox/contrib/*.service /etc/systemd/system
'/opt/netbox/contrib/netbox-rq.service' -> '/etc/systemd/system/netbox-rq.service'
'/opt/netbox/contrib/netbox.service' -> '/etc/systemd/system/netbox.service'
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox#
```

```
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox# systemctl daemon-reload
```

Ensuite, il faut démarrer les services et les configurer afin qu'ils se lancent automatiquement lors du démarrage de la machine.

```
systemctl enable --now netbox netbox-rq
systemctl status netbox.service
```

```
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox# systemctl enable --now netbox netbox-rq
Created symlink /etc/systemd/system/multi-user.target.wants/netbox.service → /etc/systemd/system/netbox.service.
Created symlink /etc/systemd/system/multi-user.target.wants/netbox-rq.service → /etc/systemd/system/netbox-rq.service.
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox# systemctl status netbox.service
● netbox.service - NetBox WSGI Service
     Loaded: loaded (/etc/systemd/system/netbox.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-02-22 19:24:25 CET; 9s ago
       Docs: https://docs.netbox.dev/
   Main PID: 82749 (gunicorn)
      Tasks: 6 (limit: 4636)
     Memory: 561.0M
        CPU: 6.002s
     CGroup: /system.slice/netbox.service
             ├─82749 /opt/netbox/venv/bin/python3.12 /opt/netbox/venv/bin/gunicorn --pid /var/tmp/netbox.pid --pythonpath /opt/netbo
             ├─82750 /opt/netbox/venv/bin/python3.12 /opt/netbox/venv/bin/gunicorn --pid /var/tmp/netbox.pid --pythonpath /opt/netbo
             ├─82751 /opt/netbox/venv/bin/python3.12 /opt/netbox/venv/bin/gunicorn --pid /var/tmp/netbox.pid --pythonpath /opt/netbo
             ├─82752 /opt/netbox/venv/bin/python3.12 /opt/netbox/venv/bin/gunicorn --pid /var/tmp/netbox.pid --pythonpath /opt/netbo
             ├─82753 /opt/netbox/venv/bin/python3.12 /opt/netbox/venv/bin/gunicorn --pid /var/tmp/netbox.pid --pythonpath /opt/netbo
             └─82754 /opt/netbox/venv/bin/python3.12 /opt/netbox/venv/bin/gunicorn --pid /var/tmp/netbox.pid --pythonpath /opt/netbo
```

5) Installation du serveur Web

Pour la configuration du serveur Web en HTTPS, nous utilisons le serveur Web Apache2. Tout d'abord, il faut installer le paquet correspondant à celui-ci.

```
apt install -y apache2
```

```
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox#  apt install -y apache2
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Le paquet suivant a été installé automatiquement et n'est plus nécessaire :
  linux-image-6.1.0-27-amd64
Veuillez utiliser « apt autoremove » pour le supprimer.
Les paquets supplémentaires suivants seront installés :
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
Paquets suggérés :
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
Les NOUVEAUX paquets suivants seront installés :
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 mis à jour, 8 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 2 220 ko dans les archives.
```

Après avoir installé le paquet, nous pouvons copier la configuration qui est donnée dans les fichiers de Netbox et la mettre sur nos sites apache2 puis modifier celle-ci, comme ci-dessous.

```
cp /opt/netbox/contrib/apache.conf /etc/apache2/sites-available/netbox.conf
vim /etc/apache2/sites-available/netbox.conf
```

```
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox# cp /opt/netbox/contrib/apache.conf /etc/apache2/sites-available/netbox.conf
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox# vim /etc/apache2/sites-available/netbox.conf
```

Dans le fichier, il faut modifier le nom du serveur « ServerName », puis modifier les chemins pour les certificats HTTPS et HTTP utilisés, ces certificats peuvent être générés via l'autorité de certification.

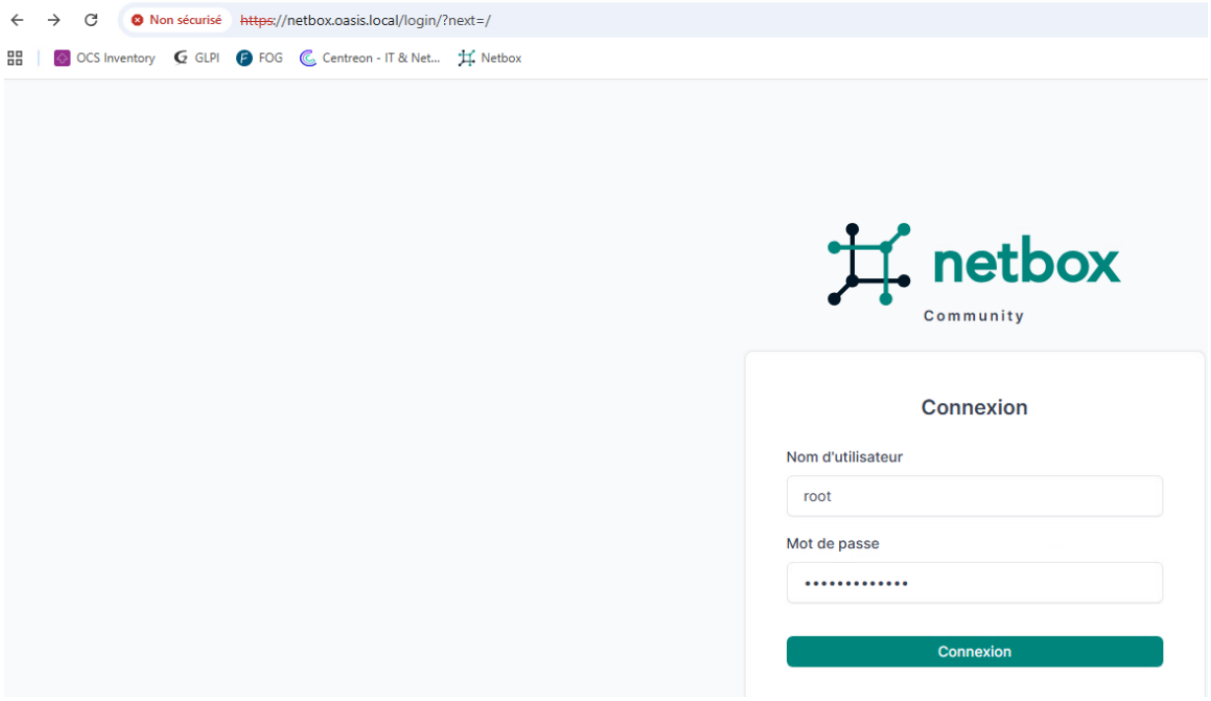
```
/e/a/s/netbox.conf
<VirtualHost *:80>
    # CHANGE THIS TO YOUR SERVER'S NAME
    ServerName SRV-P-NETBOX01

    RewriteEngine On
    RewriteCond %{HTTPS} !=on
    RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
</VirtualHost>

<VirtualHost *:443>
    ProxyPreserveHost On

    # CHANGE THIS TO YOUR SERVER'S NAME
    ServerName SRV-P-NETBOX01

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/netbox.crt
    SSLCertificateKeyFile /etc/ssl/private/netbox.key

    Alias /static /opt/netbox/netbox/static

    <Directory /opt/netbox/netbox/static>
        Options FollowSymLinks MultiViews
        AllowOverride None
        Require all granted
    </Directory>

    <Location /static>
        ProxyPass !
    </Location>

    RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}
    ProxyPass / http://127.0.0.1:8001/
    ProxyPassReverse / http://127.0.0.1:8001/
</VirtualHost>
```

Une fois la configuration réalisée, il faut activer les différents modules Apache2 pour ce site, puis ajouter ce site dans les sites disponibles et enfin redémarrer le service Apache2 afin que les modifications soient prises en compte.

```
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox# sudo a2enmod ssl proxy proxy_http headers rewrite
sudo a2ensite netbox
sudo systemctl restart apache2
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates
Enabling module proxy.
Considering dependency proxy for proxy_http:
Module proxy already enabled
Enabling module proxy_http.
Enabling module headers.
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
Enabling site netbox.
To activate the new configuration, you need to run:
  systemctl reload apache2
(venv) root@SRV-P-NETBOX01:/opt/netbox/netbox#
```

6) Utilisation Netbox

Pour aller sur l'interface Web de Netbox, il faut taper son nom FQDN dans un navigateur Web puis se connecter dessus avec l'utilisateur qui a été créé précédemment comme ci-dessous.



Une fois connecté, cette page s'affichera avec les différents menus de NetBox, avec ces différentes étapes, l'installation de NetBox est terminée.