Installation PXE Automatisée – Ubuntu 25.04 avec Miroir APT Local Minimal

Ethan Rogge, Melnic Alexandru, Flament Franklin, Bollengier Thomas, Bui The Dat

15 mai 2025

Résumé

Ce document décrit pas à pas l'implémentation d'un serveur PXE complet (DHCP + TFTP + HTTP + NFS) capable de déployer Ubuntu 25.04 Desktop via des profils *autoinstall* et un miroir APT local minimal géré par `aptly`. Il sert :

— de mémo rapide pour réinstaller l'infrastructure en un minimum de temps ;
— de référence pour diagnostiquer les pannes (où regarder, quels logs, quels ports) ;
— de procédure de changement (ajout de paquets, nouvelle version d'Ubuntu, nouveaux profils, etc.).

Les fichiers complets (`.conf`, `.yaml`, scripts) sont fournis en pièces jointes ; seules les parties importantes sont commentées ici.

1 Architecture et choix techniques

1.1 Rôles des machines

Serveur PXE Ubuntu 25.04 Server (nom : `pxe`), adresse IP fixe `192.168.0.58`. Elle regroupe plusieurs services essentiels :

— **DHCP** : via `isc-dhcp-server`, pour l'attribution automatique d'adresses IP aux clients ;
— **TFTP** : via `tftpd-hpa`, pour fournir le fichier de boot `pxelinux.0` ;
— **HTTP** : via `Apache2`, pour héberger les fichiers `user-data`, `meta-data` et l'image ISO Ubuntu ;
— **NFS** : pour permettre la lecture directe de l'ISO sans copie locale ;
— **Miroir APT** : via `aptly`, servant de source locale pour les paquets et leurs dépendances.

Clients Machines configurées pour démarrer en réseau via PXE. Elles chargent `pxelinux.0`, puis sélectionnent un profil `autoinstall` parmi les suivants : `admin`, `dev`, `sec`.

1.2 Interaction entre les machines et déroulement du FAI

— Le client démarre et, ne disposant ni de système d'exploitation ni de configuration disque, tente un boot réseau (PXE).
— Le client envoie une requête DHCP sur le réseau. Le serveur PXE lui attribue une adresse IP et lui fournit l'adresse du serveur TFTP.
— Le client contacte alors le serveur TFTP et télécharge le fichier `pxelinux.0`, ainsi que les fichiers de configuration associés.

— Grâce à ces fichiers, le client est redirigé vers le serveur HTTP pour récupérer une image ISO Ubuntu et les fichiers d'installation automatisée (`user-data` et `meta-data`).
— Le système s'installe automatiquement selon le profil choisi. Les fichiers autoinstall définissent notamment :
  — la configuration du système (partitionnement, langue, utilisateur, etc.) ;
  — les paquets à installer automatiquement ;
  — les commandes `late-commands` (comme l'ajout du miroir local APT).
— L'installation s'effectue via le miroir local APT hébergé sur le serveur, évitant ainsi l'utilisation d'Internet et accélérant les installations.

1.3 Principaux choix expliqués en détail

— **Utilisation d'Aptly plutôt que `dpkg-scanpackages` :** Aptly offre des fonctionnalités avancées essentielles pour la gestion efficace d'un miroir APT local. Contrairement à `dpkg-scanpackages`, Aptly permet :
  — la gestion simplifiée de plusieurs composants (main, dev, admin, sec) via une publication unique ;
  — la création automatisée de snapshots quotidiens, facilitant la gestion d'historique et les rollbacks éventuels ;
  — un nettoyage automatique et planifié des anciennes versions des paquets, réduisant ainsi l'espace disque utilisé et optimisant les performances.
— **Méthode d'installation hors ligne avec montage local ISO :** Plutôt que de télécharger à chaque installation les fichiers nécessaires depuis Internet, l'ISO complète d'Ubuntu Desktop est téléchargée une fois sur le serveur, montée localement et pris via NFS (Network File System), ce dernier a été utilisé pour éviter de charger le fichier iso dans la RAM, en raison de sa taille.. Cette méthode permet :
  — d'éviter toute dépendance à une connexion Internet durant l'installation ;
  — de réduire drastiquement la consommation de bande passante réseau ;
  — d'assurer une disponibilité et une rapidité accrues lors des déploiements massifs. ł'utilisation
— **Création de groupes utilisateurs spécifiques :** Afin de renforcer la sécurité et de prévenir les accès non autorisés à des privilèges élevés, chaque profil utilisateur (`adminsys`, `devuser`, `secretary`) est configuré dans des groupes spécifiques limitant strictement leurs droits :
  — L'utilisateur `adminsys` a un accès complet,étant dans le groupe sudo sans autres restrictions, nous avons choisi de ne pas en faire un utilisateur root complet, car ce n'est pas une bonne pratique , tandis que le `devuser` a des privilèges sudo, mais limités pour ne pas pouvoir accéder au shell root, enfin l'utilisateur `secretary` n'a pas de privilèges sudo, les paramètres détaillés pour chaque profil peuvent être vus dans les fichiers autoinstall.yaml .
  — Un utilisateur spécifique `sysadmin` est cependant présent sur chaque poste (notamment Dev et Secrétaire), pour les besoins ponctuels d'administration, disposant lui d'un accès complet à `sudo`.Nous avons décidé qu'il devrait être présent avec le compte admin_lead, y compris pour les postes destinés aux administrateurs système, pour pouvoir assurer le support et la maintenance pour chacun, s'il aura ce rôle.
— **Meta-paquets personnalisés avec `equivs` :** L'utilisation de méta-paquets sim-

plifie considérablement la gestion logicielle pour chaque profil (dev, adminsys, secrétaire). Chaque méta-paquet (`dev-profile`, `admin-profile`, `sec-profile`) permet :
  - une gestion unifiée des dépendances logicielles pour chaque type de poste ;
  - une simplification des mises à jour logicielles ultérieures : il suffit d'ajuster le méta-paquet et de republier sur Aptly pour mettre à jour automatiquement tous les clients concernés. Un script a été créé à cet effet.
- **Ansible :** Il est également possible d'utiliser ansible pour installer rapidement et instantanément certains paquets pour un groupe d'utilisateurs spécifique, comme dev par exemple. Lors de l'installation, l'adresse IP de chaque machine est enregistrée sur le serveur, dans un fichier correspondant au type d'utilisateur. Finalement, un script a été créé qui prend ces adresses, utilise un playbook, se connecte aux clients via ssh, et a pour fonction d'installer un package ou d'effectuer une mise à niveau ou une mise à jour.
- **Utilisation d'Autoinstall plutôt que Preseed** Le choix d'utiliser la solution moderne `autoinstall` est motivé par plusieurs avantages majeurs sur les approches traditionnelles telles que Preseed ou Ansible :
  - `Autoinstall` est nativement supporté et recommandé par Ubuntu depuis la version 20.04, offrant une intégration parfaite et un support à long terme par la communauté Ubuntu.
  - Contrairement à Preseed, qui nécessite des configurations complexes et est souvent moins clair, `autoinstall` utilise un format YAML beaucoup plus lisible et facile à maintenir.
  - Il convient de noter que l'installation automatique peut être utilisée pour Ubuntu Desktop à partir de la version 24.04 et n'est pas prise en charge pour les versions antérieures. Pour les versions précédentes, Ubuntu Live Server peut être utilisé avec des packages supplémentaires pour créer un environnement de Desktop. Pour une meilleure expérience utilisateur, cette dernière méthode n'est pas la meilleure solution et nécessite une configuration plus étendue pour créer un environnement de Desktop approprié.

2 Nom des fichiers de configuration et description de leur contenu

`/etc/dhcp/dhcpd.conf` : Définit les paramètres réseau pour les clients PXE (plage IP, serveur DNS, fichier de démarrage PXE et adresse du serveur TFTP).

`/etc/default/isc-dhcp-server` : Spécifie l'interface réseau utilisée par le service DHCP.

`/etc/default/tftpd-hpa` : Configure le serveur TFTP (répertoire racine, adresse et options de sécurité).

`/srv/tftp/pxelinux.cfg/default` : Fichier définissant le menu de boot PXE, incluant les profils disponibles (Admin, Dev, Sec) et leurs paramètres.

`/etc/exports` : Configure l'export NFS pour le partage de l'ISO Ubuntu montée en local vers les clients.

`/var/www/html/profiles/*/autoinstall.yaml` : Fichiers YAML définissant les configurations automatiques (création de comptes utilisateurs, installation de paquets

spécifiques, configuration APT, commandes post-installation).

`/var/www/html/profiles/*/custom-aptly.sources` : Fichiers APT configurant chaque profil pour utiliser le miroir local avec les bons composants (main, dev, admin, sec).

`/var/www/html/profiles/rc-local.service` : Permet la création d'un service qui sera lancé au premier démarrage pour que l'utilisateur puisse définir son propre mot de passe, ce fichier est créé sur le serveur, et mis sur le système de l'utilisateur via http dans la postinstallation.

`/var/www/html/profiles/*/rc.local` : Le script correspondant qui oblige l'utilisateur à définir son propre mot de passe.

`/usr/local/bin/update_aptly.sh` : Script automatisant la mise à jour quotidienne du miroir Aptly, incluant la création de snapshots, la fusion de profils, la publication, et le nettoyage des anciens snapshots.

`/etc/apt/apt.conf.d/20auto-upgrades` : Configure les mises à jour automatiques quotidiennes côté client (unattended-upgrades), un fichier qui est configuré directement pendant le processus d'installation sur le système de l'utilisateur.

`/var/www/html/report.php` : Le fichier est un script PHP exécuté sur le serveur Apache. Il permet de collecter et enregistrer dynamiquement dans des fichiers sur le serveur l'adresse IP et le type d'utilisateur de chaque machine cliente au moment de son installation. Hors de l'étape late-command de l'installation, chaque client exécute une commande curl qui appelle le fichier report.php.

`/install_package.sh` :

`/install_package.yml` : Le fichier est un playbook Ansible qui permet d'automatiser la gestion des paquets logiciels sur un ensemble de machines distantes. Il est conçu pour exécuter différentes opérations en fonction d'un paramètre transmis (pkg), tel que l'installation d'un paquet, une mise à jour (update) ou une mise à niveau (upgrade) du système.

3 Commandes utiles

3.1 Vérification rapide des services serveur PXE

```
sudo systemctl status isc-dhcp-server tftpd-hpa apache2 nfs-server
```

3.2 Vérification des ports ouverts

```
sudo ss -lupnt | grep -E ':67|:69|:80|:2049|:22'
```

3.3 Contrôle des journaux système

```
sudo journalctl -u isc-dhcp-server
sudo tail -f /var/log/syslog | grep tftp
sudo tail -f /var/log/apache2/access.log
```

3.4 Script publication et update du miroir Aptly

```
APTLY_HOME=/home/username/.aptly /usr/local/bin/update_aptly.sh
```

3.5 Dépannage côté client

```
sudo systemctl status unattended-upgrades
sudo unattended-upgrade --dry-run --debug
```

3.6 Redémarrage rapide des services

```
sudo systemctl restart isc-dhcp-server tftpd-hpa apache2 nfs-kernel
   -server
```

4 Processus en services sur le server

Lors du fonctionnement normal du serveur PXE, les processus suivants doivent être actifs en permanence pour assurer un fonctionnement optimal :

- **dhcpd (isc-dhcp-server)** : service DHCP qui fournit des adresses IP et les paramètres nécessaires pour le démarrage réseau PXE.
- **in.tftpd (tftpd-hpa)** : serveur TFTP responsable du transfert initial des fichiers de boot (`pxelinux.0`, kernel, initrd).
- **apache2** : serveur HTTP servant les profils `autoinstall.yaml` et le miroir APT local.
- **rpc.nfsd** : serveur NFS exportant l'ISO Ubuntu Desktop, monté en lecture seule par les clients.
- **cron** : lance quotidiennement le script `update_aptly.sh` qui met à jour le miroir APT local et gère les snapshots de paquets.
- -

Ces processus doivent être régulièrement surveillés via `systemctl status` pour prévenir d'éventuels dysfonctionnements.

5 Service tournant en arrière-plan côté client

Chaque client déployé exécute automatiquement un service essentiel assurant les mises à jour automatiques depuis le miroir APT local :

`unattended-upgrades` permet l'installation silencieuse et quotidienne des mises à jour de sécurité et des paquets via le miroir local.

5.1 Fichier de configuration principal

```
/etc/apt/apt.conf.d/20auto-upgrades
```

5.2 Contenu du fichier

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
APT::Periodic::AutocleanInterval "7";
```

Ce fichier est créé lors du processus de post-installation directement sur le système client.

5.3 Commandes utiles pour vérifier le service côté client

```
sudo systemctl status unattended-upgrades
sudo unattended-upgrade --dry-run --debug
```

Ce service garantit que tous les postes clients restent constamment à jour sans nécessiter d'intervention manuelle régulière.

6 Ports Réseau Utilisés

Les ports réseau suivants doivent être ouverts et accessibles sur le serveur PXE pour assurer son bon fonctionnement :

- **67/UDP :** Service DHCP utilisé pour l'attribution automatique d'adresses IP aux clients PXE.
- **69/UDP :** TFTP, utilisé pour la récupération initiale des fichiers de boot.
- **80/TCP :** Apache2, nécessaire à la distribution des fichiers de configuration `autoinstall.yaml` et du miroir APT local.
- **2049/TCP :** NFS, utilisé par les clients pour monter l'ISO Ubuntu Desktop en lecture seule.
- **22/TCP :** SSH, utilisé pour l'administration distante du serveur PXE et la gestion des dépôts Aptly.

7 Scripts de démarrage

Tous les services critiques (DHCP, TFTP, Apache, NFS) sont configurés pour démarrer automatiquement au lancement du serveur via `systemd`. Exemple de vérification rapide des services :

```
sudo systemctl enable isc-dhcp-server tftpd-hpa apache2 nfs-server
sudo systemctl start isc-dhcp-server tftpd-hpa apache2 nfs-server
sudo systemctl status isc-dhcp-server tftpd-hpa apache2 nfs-server
```

8 Liste détaillée des étapes d'implémentation

8.1 Installation initiale des paquets sur le serveur PXE

```
sudo apt update
sudo apt install -y isc-dhcp-server tftpd-hpa apache2 wget curl
   dpkg-dev xorriso openssh-server nfs-kernel-server aptly gnupg2
   pxelinux syslinux-common equivs ansible php libapache2-mod-php
```

8.2 Configuration du serveur DHCP

Exécutez la commande **ip addr :** pour voir l'adresse du serveur

Fichier : `/etc/dhcp/dhcpd.conf`

Modifier le fichier suivant avec les paramètres réseau appropriés, le fichier complété sera joint, voici un exemple, doit être modifié en fonction de l'adresse IP du serveur :

```
# option definitions common to all supported networks...
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;
# Allow booting and bootp (PXE) clients
allow booting;
allow bootp;

default-lease-time 600;
max-lease-time 7200;
authoritative;

subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.100 192.168.0.200;
  option routers 192.168.0.1;
  option domain-name-servers 8.8.8.8;
  filename "pxelinux.0";
  next-server 192.168.0.58;
}
ddns-update-style none;
```

Interface DHCP : `/etc/default/isc-dhcp-server`

```
sudo nano /etc/default/isc-dhcp-server
INTERFACESv4="enp0s8"
```

8.3 Configuration du serveur TFTP

Modifier le fichier suivant et créer les répertoires :

Fichier : `/etc/default/tftpd-hpa`

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

```
sudo mkdir -p /srv/tftp
sudo chmod -R 777 /srv/tftp
sudo systemctl restart tftpd-hpa
```

8.4 Préparation et montage local de l'ISO Ubuntu Desktop

```
sudo mkdir -p /srv/iso /mnt/ubuntu
cd /srv/iso
sudo wget https://releases.ubuntu.com/plucky/ubuntu-25.04-desktop-
   amd64.iso -O ubuntu-desktop.iso
sudo mount -o loop ubuntu-desktop.iso /mnt/ubuntu

sudo mkdir -p /srv/tftp/ubuntu/casper
sudo cp /mnt/ubuntu/casper/{vmlinuz,initrd} /srv/tftp/ubuntu/casper
   /
```

Il faut refaire le mount à chaque redémarrage du serveur.

8.5 Configuration du menu PXE et export NFS

```
sudo cp /usr/lib/PXELINUX/pxelinux.0 /srv/tftp/
sudo cp /usr/lib/syslinux/modules/bios/* /srv/tftp/
sudo mkdir -p /srv/tftp/pxelinux.cfg
```

Créer et modifier les fichiers suivants :
- `/srv/tftp/pxelinux.cfg/default` - entièrement disponible dans l'archive de fichiers
- `/etc/exports`

Puis activer l'export NFS :

```
# dans le fichier /etc/exports
/mnt/ubuntu *(ro,sync,no_subtree_check,no_root_squash)
#
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

8.6 Clé ssh

Pour que l'administrateur système puisse se connecter aux machines des utilisateurs, il est nécessaire de configurer une clé ssh, il convient de mentionner que tous les utilisateurs auront openssh-server installé, et la clé publique appartiendra au compte adminsys sur chaque machine.

Exécutez la commande suivante pour générer une nouvelle paire de clés SSH :

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Appuyez sur Enter pour accepter l'emplacement par défaut (/.ssh/id_rsa) : Vous serez invité à saisir une phrase secrète pour sécuriser la clé privée. Laissez vide (en appuyant sur Enter).

```
cat ~/.ssh/id_rsa.pub
```

Copiez tout le contenu du fichier, c'est la clé, et placez-le dans le fichier autoinstall.yaml pour chaque profil.

```
ssh -i ~/.ssh/id_rsa adminsys@client_machine_ip
```

Il s'agit de la commande permettant de se connecter à la machine cliente en tant qu'utilisateur adminsys.

9 Instalation automatique des paquets avec ansible, aussi upgrade et update.

9.1 IP

Il faut creer un script PHP qui reprendre l'addrese IP depuis la machine client, qui va faire un curl via HTTP sur ce fichier dans la post-instalation. Le script prend l'adresse IP et l'insère dans les fichiers correspondant au type d'utilisateurs Repeter ces commands pour tous les profiles

```
sudo touch /var/log/dev.txt     #admin.txt #sec.txt
sudo chown www-data:www-data /var/log/dev.txt
sudo chmod 664 /var/log/dev.txt
```

Fichier : `/var/www/html/report.php`

```
<?php
$client_ip = $_SERVER['REMOTE_ADDR'] ?? 'unknown';
$hostname = $_GET['hostname'] ?? 'unknown';
$line = $hostname . ' ' . $client_ip . "\n";
file_put_contents('/var/log/' . $hostname . '.txt', $line,
   FILE_APPEND | LOCK_EX);
header('Content-Type: text/plain');
echo "OK\n";
?>
```

9.2 Configuration ansible.

Fichier : `/install_package.yml`
Configurer un playbook, avec les fonctions nécessaires pour installer un package, une mise à jour ou une mise à niveau.
Fichier : `/install_package.sh`

```
chmod +x install_package.sh
```

Le script qui est appelé lors de la commande curl par le client, prend l'adresse IP et l'enregistre dans le fichier correspondant au profil. Les fichiers sont disponibles dans l'archive.

9.3 Création des profils *autoinstall*

Créer les répertoires et fichiers YAML :

```
sudo mkdir -p /var/www/html/profiles/{admin,dev,sec}
sudo nano /var/www/html/profiles/{admin,dev,sec}/autoinstall.yaml
```

10 configuration pour que l'utilisateur pouvoir mettre son password au premiere boot

```
sudo nano /var/www/html/profiles/dev/rc.local    la meme pour les
   autres profils .

#   mettre dedans , adminsys pour le profil admin, sec pour le
   profil secretaire.

#!/bin/bash
usermod -p '' devuser
passwd --expire devuser
rm -f /etc/rc.local
exit 0
#EOF

sudo nano /var/www/html/profiles/rc-local.service

#   mettre dedans

Description=/etc/rc.local Compatibility
ConditionPathExists=/etc/rc.local
After=network.target

[Service]
Type=forking
ExecStart=/etc/rc.local start
TimeoutSec=0
RemainAfterExit=yes
GuessMainPID=no

[Install]
WantedBy=multi-user.target
#EOF
```

10.1 Automatisation des mises à jour Aptly

Créer le script et configurer cron :
Script : `/usr/local/bin/update_aptly.sh`
Planification cron quotidienne (2h du matin) :

```
0 2 * * * APTLY_HOME=/home/username/.aptly /usr/local/bin/
   update_aptly.sh >> /var/log/aptly-update.log 2>&1
```

10.2 Création du miroir APT local minimal

Création de la clé pour permettre la connexion du miroir à internet :

```
gpg --no-default-keyring --keyring trustedkeys.gpg --keyserver
   keyserver.ubuntu.com --recv-keys 871920D1991BC93C
```

Création du fichier txt contenant la liste de tous les paquets qu'il faut à la création du miroir :

```
sudo nano packages.txt
```

Création initiale du miroir avec Aptly :

```
FILTER=$(paste -sd'|' packages.txt)

aptly mirror create -architectures=amd64   -filter=$FILTER -filter-
   with-deps plucky http://archive.ubuntu.com/ubuntu plucky
aptly repo create dev-extra
aptly repo create admin-extra
aptly repo create sec-extra
aptly mirror update plucky
```

Création de la clé pour publish :

```
gpg --full-generate-key

-Choose the key type: RSA and RSA (option 1).

-Set the key size: 2048 or 4096 bits (2048 is fine for most use
   cases).

-Set the key expiration: choose a suitable expiration date.

-Enter your name and email address when prompted.

-You may be asked to provide a passphrase (make sure to remember
   this).
```

10.3 Création de méta-paquets par profil

Utiliser `equivs` pour créer des méta-paquets :

```
mkdir -p ~/meta && cd ~/meta

sudo nano common-packages-control + dev-profile-control + admin-
   profile-control + sec-profile-control (voir fichier en annexe
   pour le contenu)

equivs-build dev-profile-control
aptly repo add dev-extra ./dev-profile_1.0.1_all.deb

equivs-build admin-profile-control
aptly repo add admin-extra ./admin-profile_1.0.1_all.deb

equivs-build sec-profile-control
aptly repo add sec-extra ./sec-profile_1.0.1_all.deb
```

Mise à jour du miroir et publication après l'ajout des méta-paquets :

```
/usr/local/bin/update_aptly.sh
```

10.4 Fichier `.sources`

Fichier à créer :

```
sudo nano /var/www/html/profiles/{dev,sec,admin}/custom-aptly.
   sources

#   mettre dedans

Types: deb
URIs: http://192.168.0.58/ubuntu
Suites: plucky
Components: main common dev          #     change     dev    par
    admin    ou    sec    selon le profil
Signed-By: /etc/apt/trusted.gpg.d/aptly.gpg
Trusted: yes
```

10.5 Redémarrage des services après configuration

Redémarrer tous les services impliqués :

```
sudo systemctl restart isc-dhcp-server tftpd-hpa apache2 nfs-kernel
   -server
```

Après ces étapes, l'infrastructure PXE automatisée est prête à fonctionner et à déployer automatiquement les postes de travail souhaités.

11 Maintenance et Procédures de Changement

11.1 Ajout de nouveaux paquets avec l'aide du miroir

Pour ajouter des paquets spécifiques à un profil particulier :

1. Copier le paquet `.deb` sur le serveur :

   ```
   aptly repo add nom\_repo /chemin/vers/paquet.deb
   ```

   ou, si le paquet est déjà présent sur le miroir créer, mettez à jour le meta-paquet correspondant en augmentant la version et en ajoutant à dépends le paquet voulu.

2. Exécuter le script de mise à jour Aptly :

   ```
   /usr/local/bin/update_aptly.sh
   ```

3. Les clients récupèrent automatiquement les paquets via `unattended-upgrades` ou à l'aide d'ansible qui se connecte à ssh, comme on a les ip des clients sur le server trié par type de client, nous pouvons forcer la mise à jour manuellement, avant l'automatique,

4. En même temps, avec l'aide d'ansible, il est possible d'installer directement un paquet pour les utilisateurs dev, secretary ou adminsys,sans utiliser aptlty, à condition que ce paquet soit dans le miroir local.

11.2 Update, upgrade, ajout de nouveaux paquets avec l'aide d'ansible

Faire directement via ansible comme expliqué juste avant mais il faut juste effectuer au choix l'une des commandes suivantes selon ce que vous voulez faire.

```
Mode d'utilisation: remplacer "dev" par "admin" ou "sec".

./install_package.sh upgrade dev
./install_package.sh update dev
./install_package.sh nodejs dev
```

11.3 Changer de version Ubuntu

1. Télécharger la nouvelle ISO dans `/srv/iso`.
2. Monter l'ISO et copier les fichiers nécessaires vers le répertoire TFTP :

```
sudo mount -o loop nouvelle.iso /mnt/ubuntu
sudo cp /mnt/ubuntu/casper/{vmlinuz,initrd} /srv/tftp/ubuntu/
   casper/
```

3. Mettre à jour le script Aptly pour cibler la nouvelle version.
4. Relancer le script Aptly et redémarrer les services :

```
/usr/local/bin/update_aptly.sh
sudo systemctl restart isc-dhcp-server tftpd-hpa apache2 nfs-
   server
```

12 Opérations de dépannage rapide

```

Verifier la configuration DHCP

sudo journalctl -u isc-dhcp-server

Verifier les transferts TFTP

sudo tail -f /var/log/syslog | grep tftp

Consulter les acces Apache

sudo tail -f /var/log/apache2/access.log

Verifier la publication Aptly

aptly publish list
```

13 Annexe : Liste des fichiers créés ou modifiés sur le serveur

- /etc/dhcp/dhcpd.conf
- /etc/default/isc-dhcp-server
- /etc/default/tftpd-hpa
- /srv/tftp/pxelinux.cfg/default
- /var/www/html/profiles/admin,dev,sec/autoinstall.yaml
- /var/www/html/profiles/admin,dev,sec/custom-aptly.sources
- /usr/local/bin/update_aptly.sh
- /etc/cron.d/update_aptly
- /install_package.sh
- /install_package.yml
- /var/www/html/report.php
- /var/www/html/profiles/admin,dev,sec/rc-local.service
- /var/www/html/profiles/admin,dev,sec/rc.local
- /meta/common,dev,sec,admin-profile-control