Institut National de Radioélectricité et de Cinématographie

Enseignement technique secondaire de qualification

Accès aux études supérieures

Avenue Jupiter, 188

1190 Forest

WIndowsDeploy

Draouil Kamîl - Déploiement de Windows 11

Tuteur – Mr Kajoua

Pour l'obtention du certificat de qualification

Technicien(ne) en informatique

Année scolaire : 2025-2026

Remerciements

Je tiens tout d’abord à remercier l’ensemble de mes professeurs d’informatique pour leur disponibilité, les précieux conseils et l’accompagnement bienveillant durant la réalisation de ce projet mais aussi et surtout pour les trois années passées avec eux à l’Inraci.

Je remercie également ma famille et mes proches pour leur soutien indéfectible et sincère tant dans les bons que dans les mauvais moments. Je n’aurais certainement pas accompli ce projet sans leurs encouragements et leur présence quotidienne.

Résumé

Ce travail de fin d'études porte sur la mise en place d'une solution de déploiement automatisé de systèmes d'exploitation Windows au sein d'un environnement virtualisé. L'objectif principal est de concevoir, configurer et tester une infrastructure complète de déploiement reposant sur Microsoft Deployment Toolkit (MDT), installé sur un serveur Windows Server 2019, avec Windows Deployment Services (WDS) pour le boot PXE réseau, et un Raspberry Pi équipé d'un capteur DHT22 pour le monitoring.

La solution développée permet de déployer Windows 11 de manière entièrement automatisée sur des postes clients, en réduisant significativement le temps d'installation et en garantissant une configuration homogène et reproductible. Ce type de solution répond à un besoin concret dans les entreprises qui doivent gérer un parc informatique de plusieurs dizaines ou centaines de machines.

Table des matières

Remerciements 2

Résumé 2

Introduction 3

1. Analyse des besoins 4

1.1 Description du problème 4

1.2 Public cible 4

1.3 Cahier des charges 4

1.4 Contraintes 5

2. Etude préalable 5

2.1 Solutions existantes 5

2.2 Comparaison des solutions 6

2.3 Justification des choix technologiques 6

3. Conception du projet 7

3.1 Architecture globale 7

3.2 Schéma Déploiement 7

3.3 Architecture Raspberry Pi 8

4. Réalisation 8

4.1 Installation de Windows Server 2019 8

4.2 Installation de MDT et du Windows ADK 9

4.3 Création du Deployment Share 9

4.4 Importation de Windows 11 10

4.5 Création de la Task Sequence 10

4.6 Configuration de WDS et du boot PXE 11

4.7 Difficultés rencontrées et solutions apportées 11

4.8 Raspberry Pi — Monitoring, logs et capteur DHT22 12

5. Tests et validation 13

5.1 Test du boot PXE 13

5.2 Test du déploiement complet 13

5.3 Vérification post-déploiement 13

5.4 Test du Raspberry Pi 14

Conclusion 14

Sitographie 15

Annexes 16

Introduction

Aujourd'hui, la gestion d'un parc informatique est devenue un défi quotidien pour n'importe quelle structure, qu'il s'agisse d'une entreprise ou d'un établissement d'enseignement. Lorsqu'il faut équiper ou renouveler des dizaines, voire des centaines de postes de travail, installer chaque système d'exploitation à la main devient vite une tâche interminable et, avouons-le, une source évidente d'erreurs.

C'est précisément pour répondre à cette problématique que j'ai développé WindowsDeploy dans le cadre de mon travail de fin d'études. L'idée de départ est simple : automatiser entièrement le déploiement de Windows 11. À terme, un technicien doit simplement pouvoir brancher et allumer une machine pour que l'installation se fasse de bout en bout, sans qu'il n'ait à toucher au clavier.

Pour concrétiser ce projet, j'ai mis en place une infrastructure articulée autour de trois piliers :

- Le cœur du système : Un serveur Windows Server 2019 qui combine *Microsoft Deployment Toolkit (MDT)* et *Windows Deployment Services (WDS)* pour gérer le déploiement en réseau via le protocole PXE.
- La cible : Une machine virtuelle cliente, programmée pour recevoir et installer automatiquement Windows 11 dès son démarrage.
- L'optimisation : Un Raspberry Pi qui vient enrichir l'ensemble en offrant un suivi en temps réel, une centralisation des logs et une surveillance environnementale (température et humidité via un capteur DHT22) pour simuler la gestion d'une vraie salle informatique.

Ce rapport suit pas à pas la progression de mon travail. Après avoir analysé les besoins et comparé les solutions existantes pour valider mes choix technologiques, je vous présenterai la conception de l'infrastructure et les détails de sa réalisation technique. Enfin, nous verrons les tests effectués avant de dresser un bilan de ce projet et d'ouvrir la voie à de futures pistes d'amélioration.

1.Analyse des besoins

1.1 Description du problème

L'installation manuelle d'un système d'exploitation est une procédure longue et répétitive. Pour un seul poste, elle nécessite environ 30 à 60 minutes d'intervention directe : démarrage sur un support d'installation, sélection des paramètres, attente de l'installation, configuration post-installation. Multipliée par le nombre de machines d'un parc informatique, cette durée devient rapidement ingérable pour une équipe de taille réduite.

De plus, les installations manuelles sont sujettes à des erreurs humaines : mauvaise édition choisie, paramètres incorrects, oubli de logiciels, configuration non homogène entre les machines. Ces incohérences compliquent la maintenance et le support technique au quotidien.

Le projet WindowsDeploy vise à résoudre ces problèmes en automatisant entièrement le processus de déploiement, depuis le démarrage du poste jusqu'à la présentation du bureau Windows 11 configuré et prêt à l'emploi.

1.2 Public cible

Cette solution s'adresse principalement aux administrateurs système et techniciens informatiques chargés de gérer un parc de postes de travail dans une entreprise, une école ou une administration. Elle est particulièrement adaptée aux environnements où plusieurs machines doivent être configurées de manière identique et où les interventions doivent être rapides et fiables.

1.3 Cahier des charges

Le projet doit répondre aux exigences suivantes :

1. Déployer Windows 11 automatiquement via le réseau sans support physique (clé USB, DVD).
2. Le déploiement doit se déclencher au simple démarrage réseau du poste client.
3. L'installation doit être reproductible et identique sur chaque machine déployée.
4. Le serveur de déploiement doit être hébergé sur Windows Server 2019.
5. Un Raspberry Pi doit enrichir l'infrastructure avec du monitoring et de la collecte de logs.
6. Un capteur DHT22 sur le Raspberry Pi doit surveiller la température et l'humidité ambiante.

1.4 Contraintes

Le projet a été réalisé dans un environnement entièrement virtualisé avec VMware Workstation, ce qui a forcément imposé son lot de limites au quotidien. Le réseau NAT de VMware a notamment posé pas mal de défis lors de la configuration du boot PXE, m'obligeant à faire plusieurs ajustements pour que tout communique correctement. À cela s'est ajoutée une gestion des ressources assez serrée, puisque tout tournait sur un unique PC hôte, à côté du Raspberry Pi et de son capteur DHT22, ce qui demandait de bien surveiller les performances de la machine. Enfin, le facteur temps a été une vraie contrainte : avec un calendrier strictement limité à l'année scolaire, il a fallu avancer vite, prioriser les tâches et composer avec les imprévus pour réussir à boucler l'intégralité de l'infrastructure dans les délais impartis.

2 Etude préalable

2.1 Solutions existantes

Plusieurs solutions permettent de déployer automatiquement des systèmes d'exploitation en entreprise. Avant de faire un choix, j'ai comparé les principales d'entre elles.

2.1.1 Microsoft Deployment Toolkit (MDT)

MDT est l'outil de déploiement développé par Microsoft. Il est entièrement gratuit et spécifiquement conçu pour les environnements Windows. Il dispose d'une documentation très complète et est largement utilisé en entreprise. Il permet de gérer des images, des pilotes, des applications et des scripts de configuration dans un seul outil centralisé.

2.1.2 Microsoft SCCM/Intune

System Center Configuration Manager (SCCM) et Intune sont des solutions Microsoft plus avancées permettant une gestion complète du parc informatique. Très puissantes, elles sont cependant payantes, complexes à mettre en place et nécessitent une infrastructure Active Directory complète, ce qui dépasse le cadre de ce projet.

2.1.3 Clonezilla

Clonezilla est une solution open source de clonage de disque. Gratuite et simple d'utilisation, elle est cependant peu flexible : elle ne permet pas de personnaliser le déploiement selon les machines et ne dispose pas de mécanisme de gestion centralisée.

2.1.4 FOG Project

FOG Project est une solution open source de déploiement réseau. Bien que gratuite et supportant le PXE, elle est moins bien adaptée aux environnements Windows et dispose d'une communauté plus limitée que MDT.

2.2 Comparaison des solutions

Solution	Avantages	Inconvénients	Coût
MDT	Gratuit, flexible	Plus pris en charge par Windows	Gratuit
SSCM/INTUNE	Très complet	Complexe, nécessite AD	Payant
Clonezilla	Simple, open source	Peu flexible	Gratuit
FOG Project	Open source, PXE	Moins adapté Windows	Gratuit

2.3 Justification des choix technologiques

2.3.1 Microsoft Deployment Toolkit (MDT)

Après avoir comparé les différentes options, MDT s'est rapidement imposé comme le choix évident pour mon projet. Le gros point fort, c'est qu'il est totalement gratuit et développé sur mesure pour Windows. Il est super flexible, ce qui le rend aussi efficace pour faire des tests en labo que pour une mise en production réelle en entreprise. En plus, dès que je me loupais ou que je bloquais sur un problème technique, la documentation et l'aide de la communauté m'ont clairement sauvé la mise.

2.3.2 Windows Deployment Services (WDS)

WDS est un rôle de Windows Server qui permet le boot PXE, c'est-à-dire le démarrage d'un poste directement via le réseau, sans aucune clé USB ni support physique. Comme il est intégré de base à Windows Server, il est totalement gratuit et collabore parfaitement avec MDT. C'était donc le choix le plus logique pour mon projet.

2.3.3 Windows Server 2019

Windows Server 2019 est une version stable et ultra-répandue en entreprise. Elle est totalement compatible avec MDT et le Windows ADK, ce qui en fait la base solide idéale pour propulser toute mon infrastructure de déploiement.

2.3.4 Raspberry Pi avec capteur DHT22

De mon côté, le Raspberry Pi est la solution idéale pour ce projet. C’est un nano-ordinateur qui consomme très peu d’énergie mais qui est parfait pour gérer des tâches légères comme centraliser les logs ou afficher un dashboard de monitoring en temps réel. En plus, son coût est hyper abordable et sa communauté est énorme, ce qui m'a bien aidé pour trouver des ressources. Pour aller plus loin, j'ai ajouté un capteur DHT22 pour mesurer la température et l’humidité de la pièce. Ça m'a permis d'intégrer une vraie dimension IoT (Internet des Objets) au projet, en simulant la surveillance concrète d'une salle serveur.

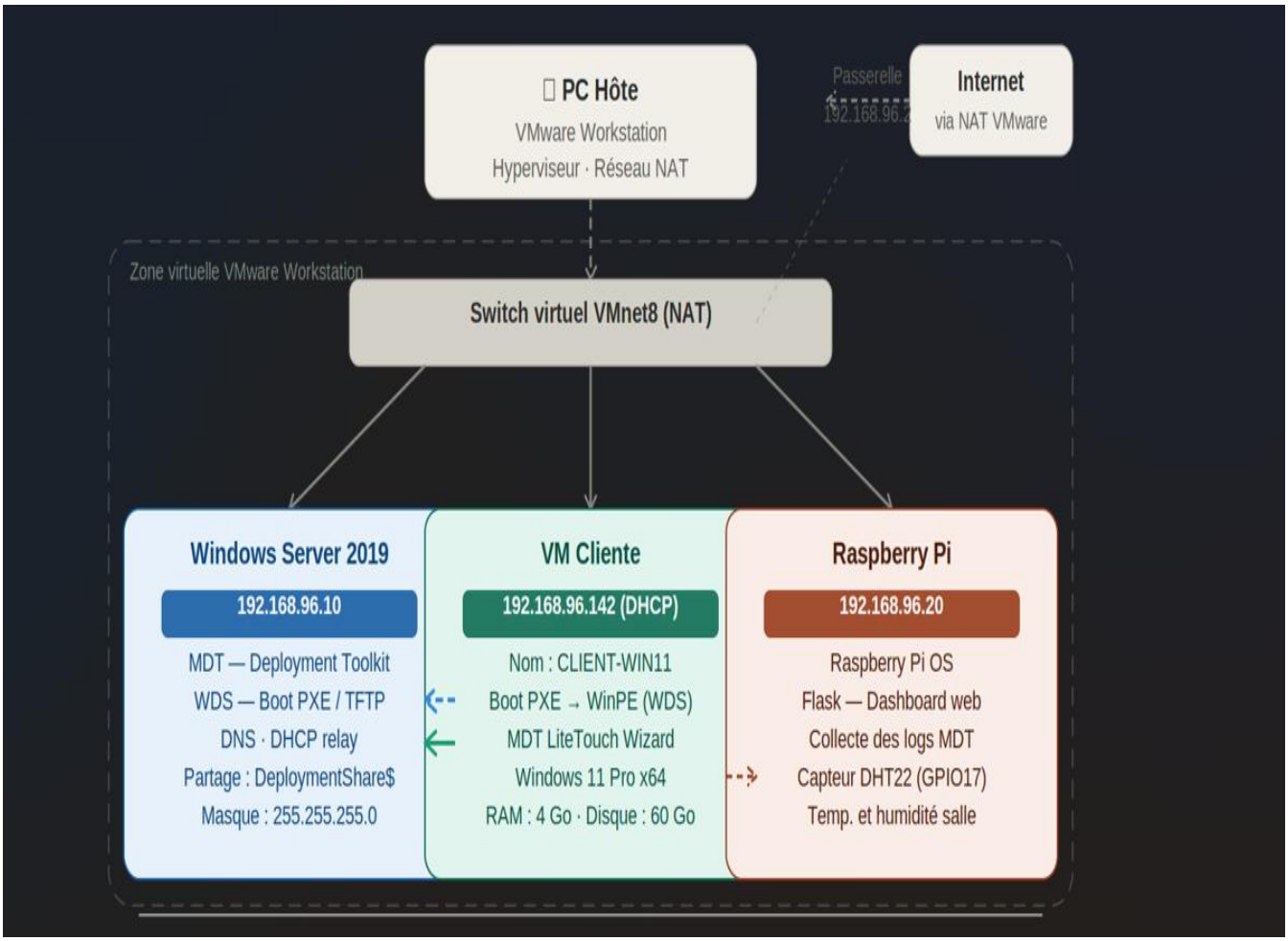
3. Conception du projet

3.1 Architecture globale

L'infrastructure WindowsDeploy repose sur trois composants principaux communiquant via un réseau local virtuel VMware NAT (192.168.96.0/24) :

Composant	Rôle	Adresse IP
Windows Server 2019	MDT+WDS+DNS	192.168.96.10
VM Cliente	Poste à déployer	192.168.96.x(DHCP)
Raspberry Pi	Monitoring+Logs+DHT22	192.168.96.20

3.2 Schéma Déploiement



3.2.1 Architecture MDT

Le Deployment Share est le dossier central de MDT, situé sur le serveur à l'emplacement C:\DeploymentShare et partagé sur le réseau sous le nom DeploymentShare$. C'est dans ce dossier que sont stockées toutes les ressources nécessaires au déploiement.

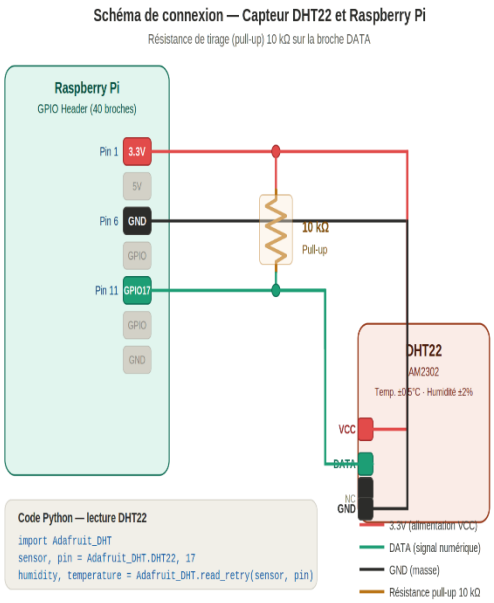
Il contient d'abord les images Windows 11, importées depuis l'ISO officiel Microsoft. C'est l'édition Windows 11 Pro x64 qui a été retenue pour ce projet.

Il contient ensuite la Task Sequence, appelée "Déploiement Windows 11". Il s'agit d'une liste d'étapes qui s'exécutent automatiquement lors du déploiement : formatage du disque, copie de Windows, configuration du système et finalisation. C'est elle qui pilote l'intégralité du processus sans intervention manuelle.

Deux fichiers de configuration complètent l'ensemble. Le fichier Bootstrap.ini indique au client PXE où se trouve le serveur MDT et définit la langue du clavier. Le fichier CustomSettings.ini personnalise le déroulement du déploiement en définissant à l'avance certains paramètres comme la langue du système ou le fuseau horaire, évitant ainsi d'avoir à répondre manuellement à ces questions lors de chaque installation.

3.3 Architecture Raspberry Pi

Pour faire tourner cette partie du projet, j'ai configuré mon Raspberry Pi pour qu'il gère deux services principaux que j'ai codés en Python. D'un côté, j'ai mis en place un serveur de logs. Son but est de récupérer et de ranger tous les rapports de déploiement de MDT dans des fichiers horodatés, ce qui est indispensable pour savoir exactement ce qui s'est passé en cas de bug. De l'autre, j'ai créé un Dashboard web avec Flask. L'avantage, c'est qu'il est accessible super facilement depuis n'importe quel appareil du réseau. Ça me permet de suivre en direct l'avancement des installations et de surveiller la température ou l'humidité de la pièce grâce au capteur DHT22. C’est un vrai centre de contrôle pour mon infrastructure.



4.1 Installation de Windows Server 2019

Pour commencer, j'ai installé Windows Server 2019 sur une machine virtuelle VMware Workstation. J'ai configuré la VM avec 4 Go de RAM, 2 cœurs de processeur, 80 Go de stockage et une carte réseau en mode NAT. Pour l'installation de Windows Server, j'ai choisi l'édition Standard avec expérience utilisateur, car elle inclut une interface graphique, ce qui est bien plus pratique pour administrer le serveur.

Une fois l'installation terminée, j'ai configuré quelques paramètres essentiels : j'ai attribué une adresse IP fixe (192.168.96.10) pour que le serveur soit toujours joignable à la même adresse sur le réseau, j'ai renommé le serveur pour le rendre facilement identifiable, et j'ai désactivé la configuration de sécurité renforcée d'Internet Explorer pour pouvoir télécharger les outils nécessaires sans problème.

4.2 Installation de MDT et du Windows ADK

Avant d'installer MDT, il faut d'abord installer deux outils fournis par Microsoft : le Windows Assessment and Deployment Kit (ADK) et son extension Windows PE. Ces outils sont indispensables car ils fournissent à MDT tout ce dont il a besoin pour créer les images de démarrage WinPE. J'ai donc téléchargé et installé dans l'ordre : l'ADK version 2004, l'extension Windows PE, et enfin MDT version 8450.

L'ordre d'installation a son importance : si MDT est installé avant l'ADK, il ne trouvera pas les composants dont il a besoin. Lors de l'installation de l'ADK, j'ai veillé à ne cocher que les composants utiles — les Outils de déploiement et le USMT — pour ne pas alourdir inutilement le système.

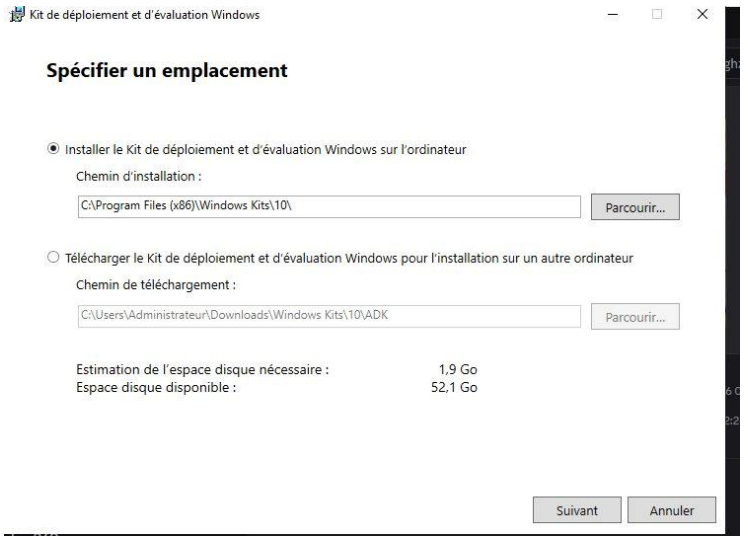


Figure 1 — Installation du Windows ADK

4.3 Création du Deployment Share

Une fois MDT installé, j'ai ouvert la console Deployment Workbench et créé un Deployment Share à l'emplacement C:\DeploymentShare. Ce dossier est le cœur de l'infrastructure MDT : c'est là que seront stockées les images Windows, les pilotes et les applications à déployer. Je

lui ai donné le nom de partage DeploymentShare$ — le $ à la fin rend le dossier invisible lors de la navigation réseau, ce qui est une bonne pratique de sécurité.

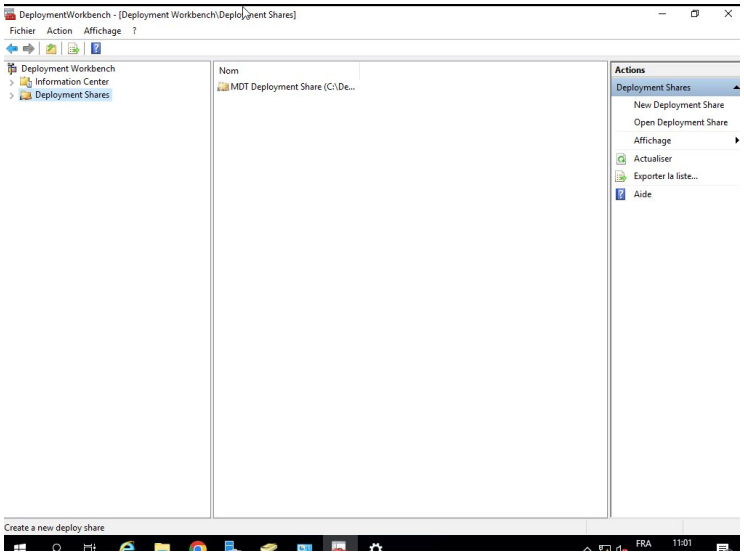


Figure 2 — Deployment Share créé dans Deployment Workbench

4.4 Importation de Windows 11

Pour importer Windows 11 dans MDT, j'ai d'abord téléchargé l'ISO officiel depuis le site de Microsoft. J'ai ensuite monté cet ISO comme second lecteur DVD dans la VM Windows Server, ce qui permet à MDT d'accéder directement aux fichiers d'installation. Via l'assistant Import Operating System, MDT a copié l'ensemble des fichiers dans le Deployment Share. Toutes les éditions présentes dans l'ISO ont été importées : Home, Pro, Education et leurs variantes N.

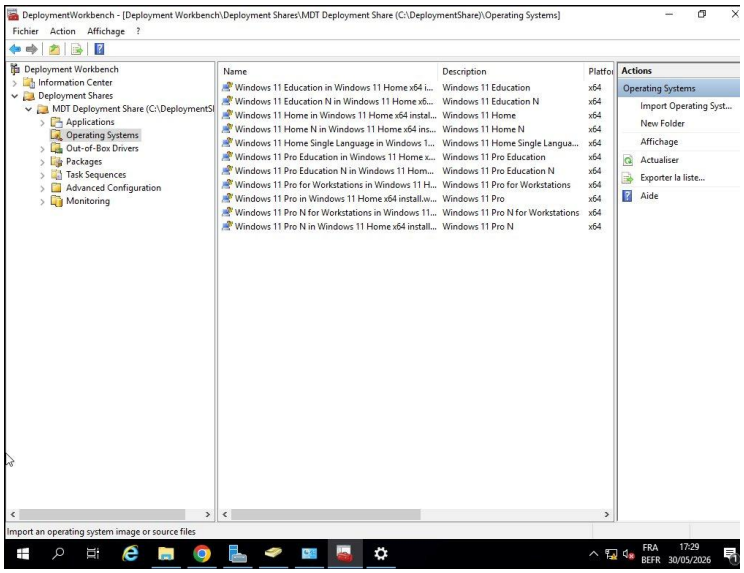


Figure 3 — Éditions Windows 11 importées dans MDT

4.5 Création de la Task Sequence

La Task Sequence, c'est ce qui fait vraiment la force de MDT. Concrètement, c'est une liste d'étapes qui s'enchaînent automatiquement lors du déploiement : formatage du disque, copie des fichiers Windows, configuration du système et finalisation. En gros, c'est elle qui remplace tout le travail manuel qu'un technicien devrait faire machine par machine.

J'ai créé une Task Sequence que j'ai appelée "Déploiement Windows 11", basée sur le template Standard Client Task Sequence proposé par MDT. Ce template est un bon point de départ car il inclut déjà toutes les étapes de base nécessaires pour une installation complète de Windows.

4.6 Configuration de WDS et du boot PXE

WDS (Windows Deployment Services) est le service qui permet à un poste client de démarrer depuis le réseau plutôt que depuis un disque ou une clé USB. Je l'ai installé comme rôle supplémentaire sur le serveur Windows via le Gestionnaire de serveur, ce qui est assez rapide. Une fois configuré en mode autonome, j'ai ajouté l'image de boot LiteTouchPE_x64.wim — générée par MDT — dans la section Images de démarrage de WDS.

J'ai ensuite ajusté les paramètres PXE pour que le serveur réponde automatiquement à toutes les machines qui tentent de démarrer depuis le réseau, et j'ai défini l'image LiteTouch comme image de démarrage par défaut pour l'architecture x64 UEFI.

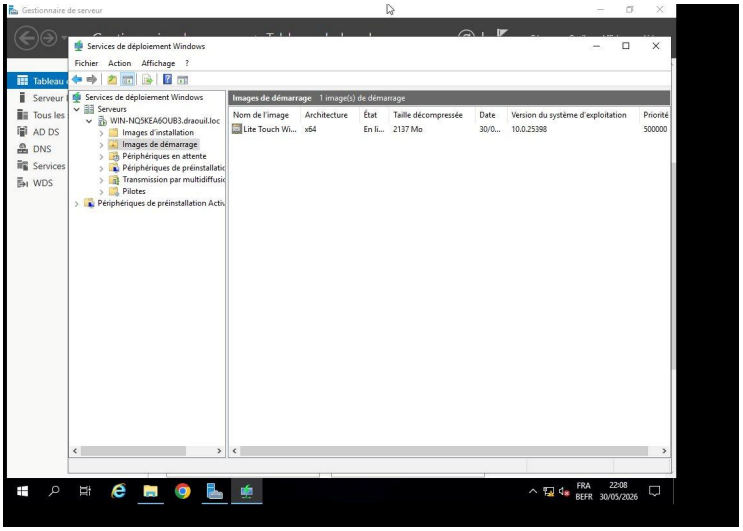


Figure 4 — Console WDS avec l'image de démarrage LiteTouch ajoutée

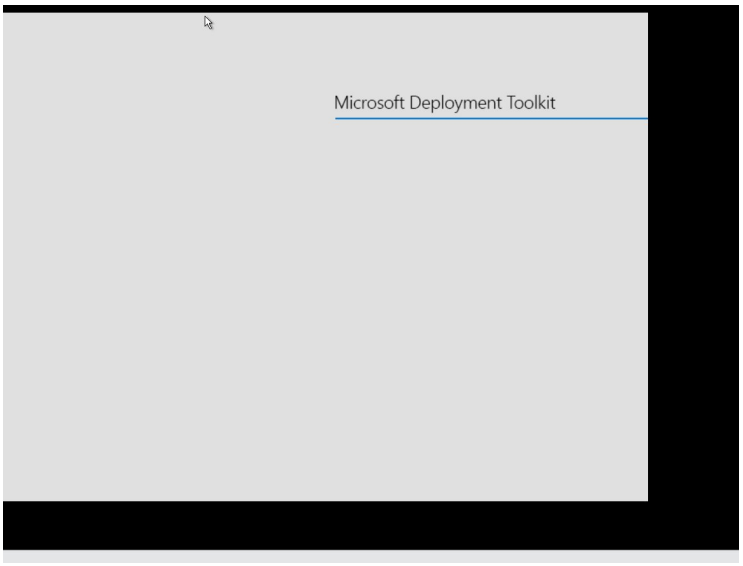


Figure 5 — Assistant MDT LiteTouch — boot PXE réussi

4.7 Difficultés rencontrées et solutions apportées

Ce projet n'a pas été sans obstacles, et la principale difficulté a été une erreur de compatibilité entre MDT 8450 et les versions récentes du Windows ADK. Au moment du boot PXE, un message d'erreur apparaissait : "The value for the attribute is not valid : language". Cette erreur empêchait les scripts MDT de se lancer dans WinPE et bloquait donc tout le déploiement.

Après pas mal de recherches et de tentatives, j'ai trouvé que le problème venait de l'incompatibilité entre MDT et l'ADK récent que j'avais installé. La solution a été de désinstaller cet ADK et de le remplacer par la version 2004, qui est parfaitement compatible avec MDT 8450 et qui est d'ailleurs celle recommandée par la communauté.

J'ai également rencontré un problème avec le boot PXE sur la VM cliente, qui ne démarrait pas automas configurer le serveur pour qu'il réponde à tous les clients inconnus et définir manuellement l'image de démarrage par défaut pour l'architecture x64 UEFI.

Figure 6 — Bureau Windows 11 après déploiement automatique réussi

4.8 Raspberry Pi — Monitoring, logs et capteur DHT22

4.8.1 Matériel et configuration de base

Pour la partie Raspberry Pi, j'ai utilisé un Raspberry Pi sous Raspberry Pi OS. J'y ai connecté un capteur DHT22 sur les broches GPIO afin de mesurer en continu la température et l'humidité ambiante. L'idée est de simuler ce qu'on trouverait dans une vraie salle serveur professionnelle, où ces paramètres sont surveillés en permanence pour éviter tout risque de surchauffe ou d'humidité excessive.

J'ai choisi le DHT22 notamment pour sa bonne précision : ±0,5°C en température et ±2% en humidité. Il est aussi très simple à utiliser avec Python grâce à la bibliothèque Adafruit, ce qui m'a permis d'intégrer rapidement les données du capteur dans le Dashboard.

4.8.2 Serveur de logs

Pour garder une trace de tous les déploiements, j'ai développé un petit service Python qui tourne en permanence sur le Raspberry Pi. Son rôle est simple : il écoute les connexions venant du serveur MDT et enregistre automatiquement chaque événement de déploiement dans des fichiers de logs. Ces fichiers sont horodatés, ce qui permet de savoir exactement quand chaque déploiement a eu lieu.

Pour chaque déploiement, le service enregistre plusieurs informations : le nom de la machine déployée, l'heure de début, l'heure de fin et le statut final (succès ou erreur). Ça permet d'avoir un historique complet et de pouvoir revenir dessus en cas de problème.

4.8.3 Dashboard de monitoring

Pour rendre tout ça visible et agréable à consulter, j'ai développé un dashboard web en Python avec le framework Flask. L'avantage de Flask, c'est qu'il permet de créer une interface web accessible depuis n'importe quel appareil connecté au réseau un PC, une tablette ou même un téléphone sans avoir à installer quoi que ce soit côté client.

Ce Dashboard affiche en temps réel plusieurs informations utiles : la liste des déploiements en cours et terminés avec leur statut, l'historique complet des installations avec les durées, la température et l'humidité relevées par le capteur DHT22, et des alertes automatiques qui se déclenchent si les valeurs sortent des seuils normaux. C'est ce genre d'outil qu'on retrouve dans les infrastructures informatiques professionnelles pour surveiller l'état d'un parc à distance.

5.1 Test du boot PXE

Pour commencer, j'ai vérifié que la VM cliente était bien capable de démarrer depuis le réseau. Au démarrage, elle a reçu automatiquement une adresse IP (192.168.96.142) via le DHCP, puis a contacté le serveur WDS à l'adresse 192.168.96.10. L'écran affichait clairement les informations de connexion : l'IP du client, l'IP du serveur et son nom. Après avoir confirmé le démarrage réseau, l'environnement WinPE s'est chargé depuis le serveur et l'assistant MDT LiteTouch s'est lancé sans problème. C'était une belle confirmation que toute la chaîne PXE fonctionnait correctement.

5.2 Test du déploiement complet

Une fois le boot PXE validé, j'ai lancé un déploiement complet de A à Z. J'ai sélectionné la Task Sequence "Déploiement Windows 11", configuré le nom de la machine (CLIENT-WIN11) et son groupe de travail, puis laissé MDT faire son travail. Il a enchaîné toutes les étapes automatiquement : formatage du disque, copie des fichiers Windows 11, redémarrages intermédiaires et configuration finale. Au bout du processus, le bureau Windows 11 est apparu sur la VM cliente sans que j'aie eu à toucher quoi que ce soit. C'est exactement le résultat attendu.

5.3 Vérification post-déploiement

Une fois le déploiement terminé, j'ai vérifié dans les Paramètres Windows que tout était en ordre. La machine portait bien le nom CLIENT-WIN11, Windows 11 était correctement installé et la connexion réseau était active. La VM était complètement opérationnelle et prête à être utilisée, sans aucune configuration manuelle supplémentaire. Ce test confirme que la solution est fiable et reproductible.

5.4 Test du Raspberry Pi

Pour valider la partie monitoring, j'ai accédé au Dashboard web du Raspberry Pi depuis un navigateur en entrant simplement son adresse IP. Les données de température et d'humidité relevées par le capteur DHT22 s'affichaient bien en temps réel et se mettaient à jour automatiquement. J'ai également pu consulter les logs des déploiements, qui étaient correctement enregistrés et présentés dans l'interface. En un coup d'œil, je pouvais voir l'historique complet de toutes les installations effectuées.

Boot PXE depuis le réseau	Succès
Chargement de WinPE via MDS	Succès
Lancement de l’assistant MDT LiteTouch	Succès
Déploiement complet Windows 11	Succès
Dashboard Raspberry Pi + DHT22	Succès

Conclusion

WindowsDeploy est sans conteste le projet le plus complet et le plus ambitieux que j'ai réalisé au cours de ma formation. Il m'a permis de mettre en pratique un grand nombre de compétences acquises durant ces années à l'INRACI : administration système Windows, configuration réseau, virtualisation, et même un peu de développement Python pour la partie Raspberry Pi.

Les objectifs fixés au départ ont tous été atteints. J'ai réussi à déployer Windows 11 automatiquement via le réseau PXE sur une machine virtuelle cliente, en partant d'un simple démarrage réseau jusqu'au bureau Windows 11 prêt à l'emploi. L'infrastructure MDT + WDS fonctionne correctement et est entièrement reproductible : il suffit d'allumer un nouveau poste sur le réseau pour que le déploiement se lance tout seul, sans qu'un technicien n'ait besoin d'intervenir.

L'ajout du Raspberry Pi avec le capteur DHT22 est venu enrichir le projet d'une dimension IoT concrète et professionnelle. Surveillance environnementale, collecte centralisée de logs, Dashboard accessible depuis n'importe quel appareil du réseau — c'est exactement le genre d'outils qu'on retrouve dans les vraies infrastructures informatiques d'entreprise.

Ce projet m'a aussi beaucoup appris sur le plan humain. Les problèmes de compatibilité entre MDT et l'ADK m'ont vraiment donné du fil à retordre, mais c'est justement là que j'ai le plus progressé : apprendre à lire les messages d'erreur avec méthode, chercher des solutions de façon structurée et ne pas lâcher même quand ça coince.

Pour aller plus loin, plusieurs améliorations seraient envisageables : l'ajout d'une étape d'installation automatique de logiciels dans la Task Sequence (suite Office, navigateur, antivirus), ou encore le test de la solution sur des machines physiques en plus des machines virtuelles, pour valider son fonctionnement en conditions réelles.

En définitive, WindowsDeploy est la concrétisation de ces années de formation en informatique et la confirmation que l'administration système est le domaine dans lequel je veux me spécialiser et construire ma carrière professionnelle.

Sitographie

Les sources suivantes ont ete consultees lors de la realisation de ce projet. Elles sont presentees selon les normes APA (7e edition).

Adafruit Industries. (s.d.). *DHT22 Temperature-Humidity Sensor tutorial.* Recupere le 1 juin 2026 de https://learn.adafruit.com/dht

Microsoft Corporation. (s.d.). *Microsoft Deployment Toolkit documentation.* Microsoft Learn. Recupere le 1 juin 2026 de https://learn.microsoft.com/fr-fr/mem/configmgr/mdt/

Microsoft Corporation. (s.d.). *Windows Assessment and Deployment Kit (Windows ADK).* Microsoft Learn. Recupere le 1 juin 2026 de https://learn.microsoft.com/fr-fr/windows-hardware/get-started/adk-install

Microsoft Corporation. (s.d.). *Windows Deployment Services overview.* Microsoft Learn. Recupere le 1 juin 2026 de https://learn.microsoft.com/fr-fr/windows-server/remote/remote-installation-services/

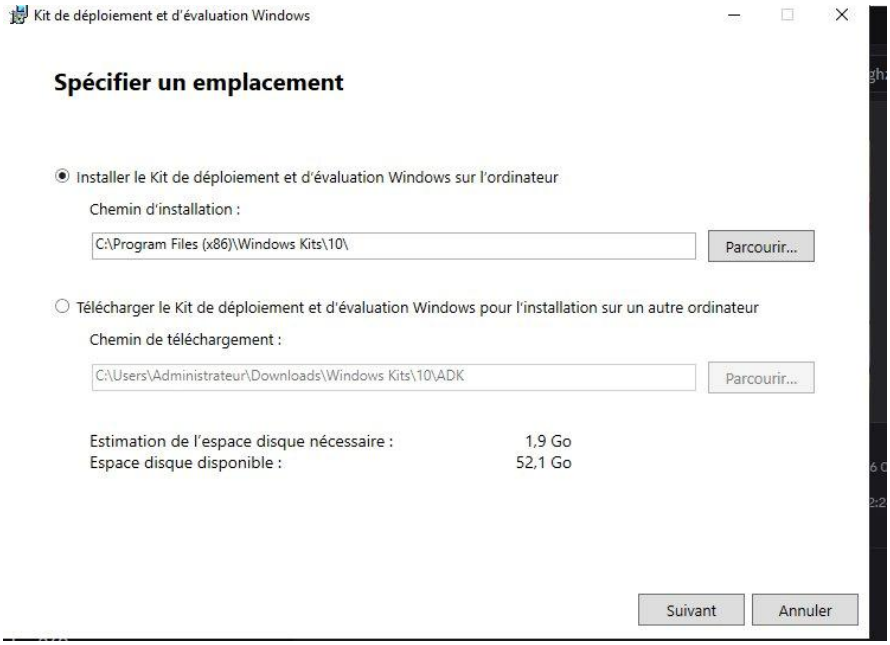
Pallets Projects. (s.d.). *Flask — Web development, one drop at a time.* Recupere le 1 juin 2026 de https://flask.palletsprojects.com/

Raspberry Pi Foundation. (s.d.). *Raspberry Pi documentation.* Recupere le 1 juin 2026 de https://www.raspberrypi.com/documentation/

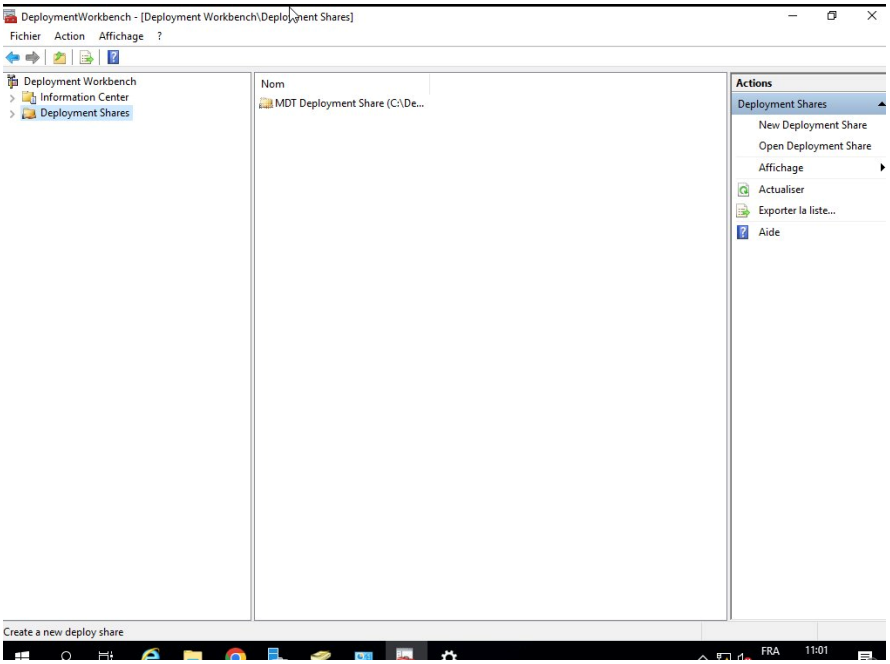
VMware. (s.d.). *VMware Workstation Pro documentation.* Recupere le 1 juin 2026 de https://docs.vmware.com/en/VMware-Workstation-Pro/

Annexes

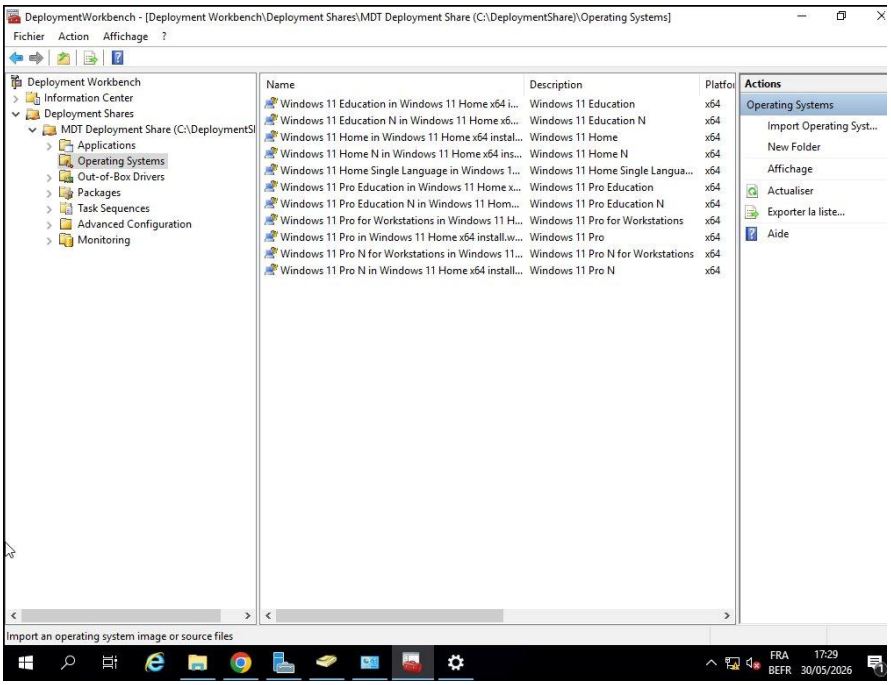
Annexe 1 — Captures d'ecran du deploiement



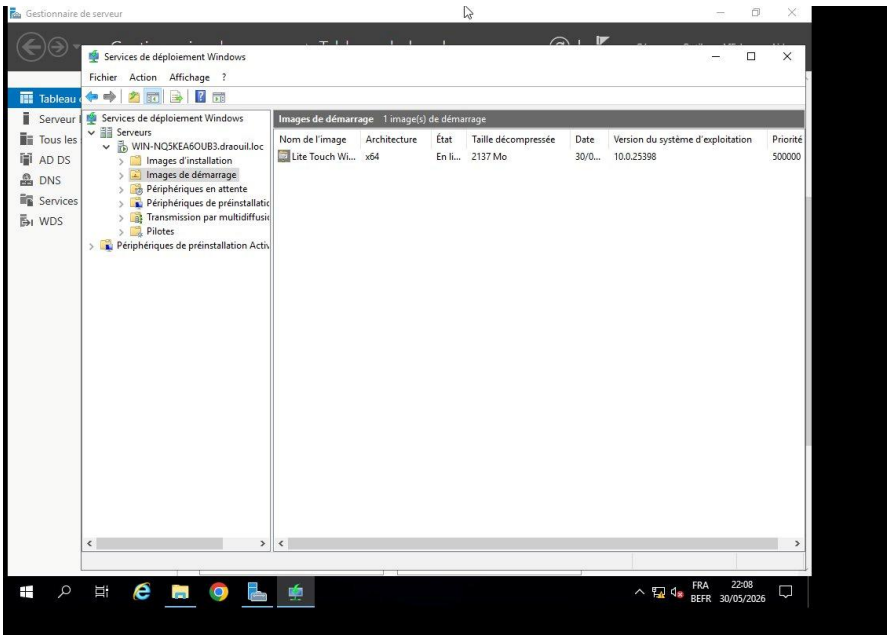
Installation du Windows ADK sur Windows Server 2019



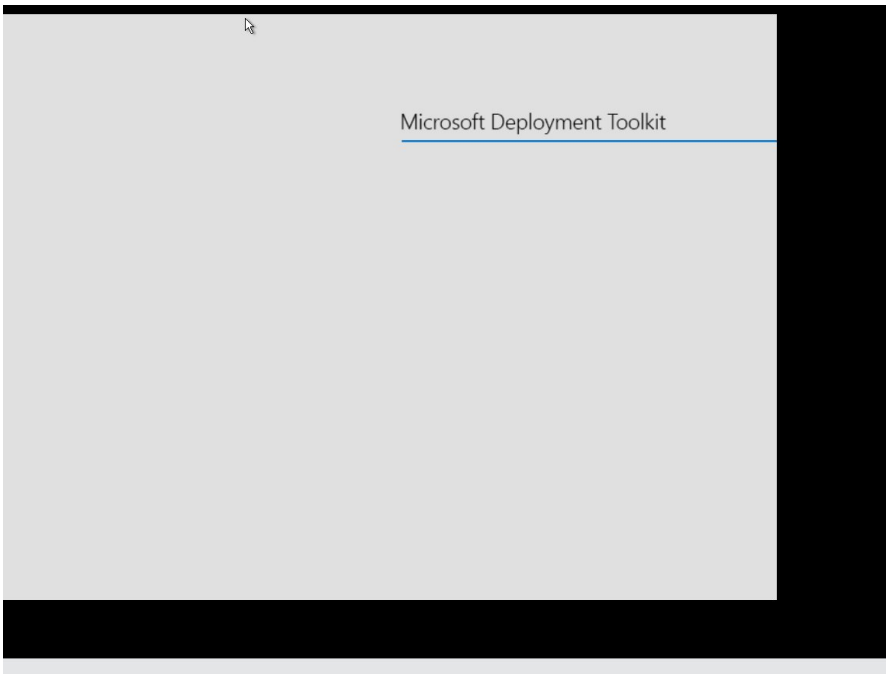
Deployment Share cree dans Deployment Workbench



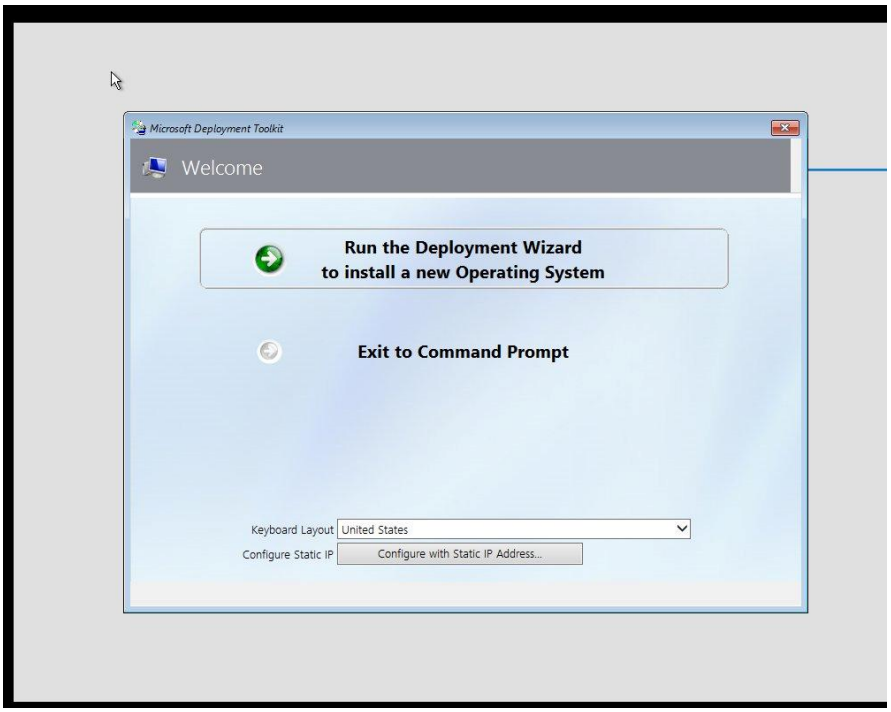
Editions Windows 11 importees dans MDT



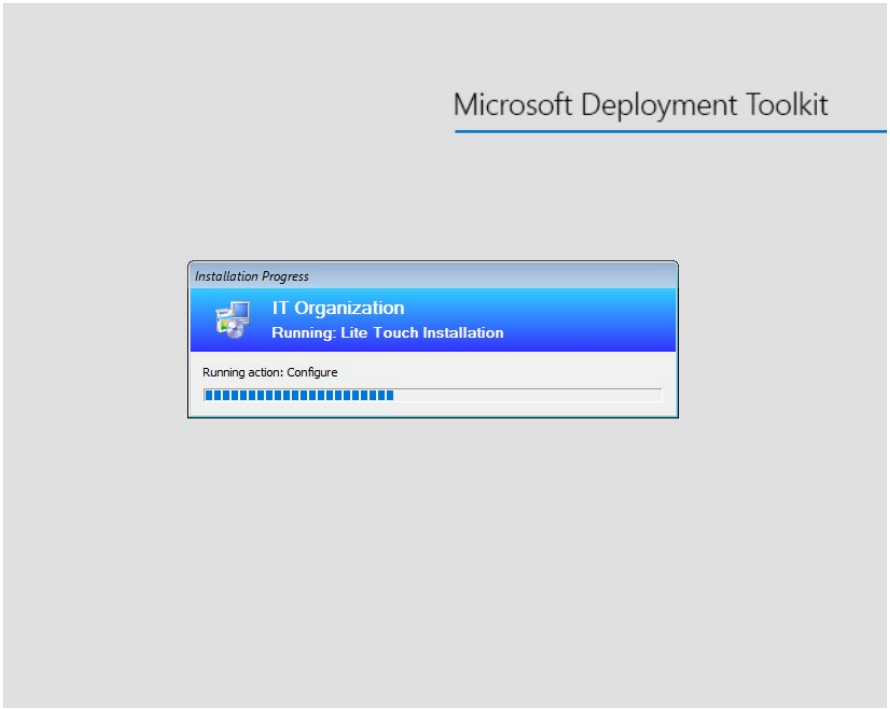
Console WDS avec l'image de demarrage LiteTouch ajoutee



Assistant MDT LiteTouch — boot PXE reussi

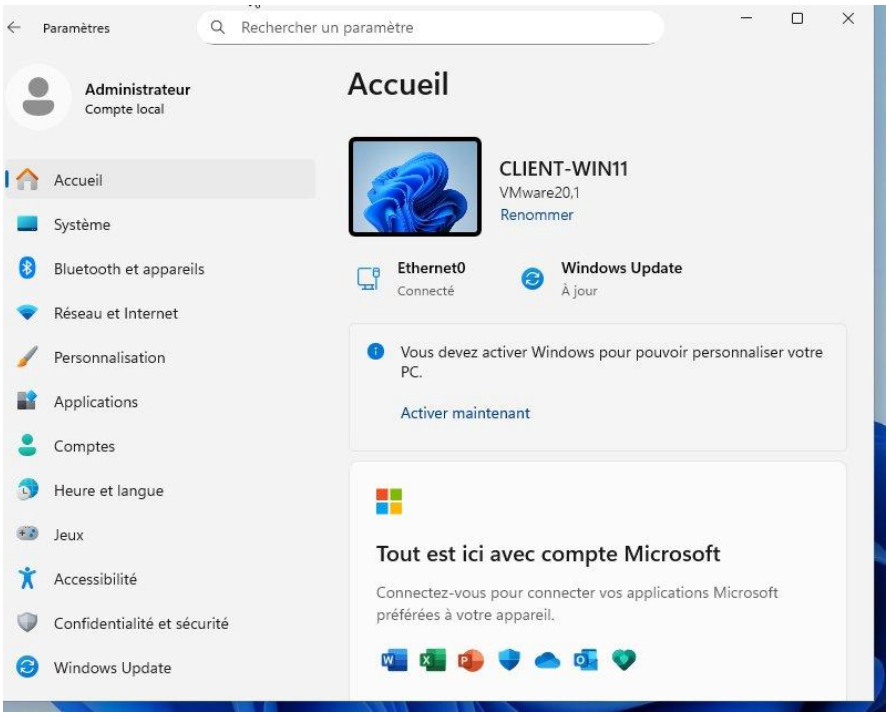


Selection de la Task Sequence dans l'assistant de deploiement



Installation en cours — MDT execute la Task Sequence

Bureau Windows 11 apres deploiement automatique reussi



Parametres Windows — CLIENT-WIN11 correctement configure

Annexe 2 — Code Python — Raspberry Pi

Les extraits de code suivants illustrent les deux composants Python developpes sur le Raspberry Pi.

```python
# dht22_sensor.py — Lecture capteur DHT22 sur Raspberry Pi
# Connexion : GPIO17  |  Bibliotheque : Adafruit_DHT

import Adafruit_DHT
import time
from datetime import datetime

SENSOR     = Adafruit_DHT.DHT22
GPIO_PIN   = 17
SEUIL_TEMP = 30.0   # Alerte si temperature > 30 degres Celsius
SEUIL_HUM  = 70.0   # Alerte si humidite > 70%

def lire_capteur():
    hum, temp = Adafruit_DHT.read_retry(SENSOR, GPIO_PIN)
    if temp is not None and hum is not None:
        return {
            'temperature' : round(temp, 1),
            'humidite'    : round(hum,  1),
            'horodatage'  : datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        }
    return None

def verifier_alertes(data):
    alertes = []
    if data['temperature'] > SEUIL_TEMP:
        alertes.append('ALERTE : Temperature elevee !')
    if data['humidite'] > SEUIL_HUM:
        alertes.append('ALERTE : Humidite excessive !')
    return alertes

while True:
    donnees = lire_capteur()
    if donnees:
        print(f"Temp: {donnees['temperature']} C  |  Hum: {donnees['humidite']} %")
        for alerte in verifier_alertes(donnees):
            print(alerte)
    time.sleep(10)
```

Code Python — Lecture et traitement des donnees du capteur DHT22

```python
# dashboard.py — Serveur Flask | Monitoring MDT + DHT22
# Accessible sur le reseau : http://192.168.96.20:5000

from flask import Flask, render_template, jsonify
import json, os
from dht22_sensor import lire_capteur

app      = Flask(__name__)
LOGS_DIR = '/home/pi/logs/deployments/'

def charger_logs():
    logs = []
    for f in os.listdir(LOGS_DIR):
        with open(LOGS_DIR + f) as fp:
            logs.append(json.load(fp))
    return sorted(logs, key=lambda x: x['debut'], reverse=True)

@app.route('/')
def accueil():
    return render_template('dashboard.html',
                           deployments=charger_logs())

@app.route('/api/capteur')
def api_capteur():
    return jsonify(lire_capteur())

@app.route('/api/logs')
def api_logs():
    return jsonify(charger_logs())

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=False)
```

Code Python — Serveur Flask du dashboard de monitoring