

Incus + FOG (titouan)

Sommaire

Sommaire.....	1
Schéma réseau :.....	2
Introduction.....	3
Enjeux.....	3
Architecture cible.....	13
Choix technologiques.....	13
Mise en place du serveur d'infrastructure.....	13
Mise en place du serveur FOG.....	20
Création de la machine de référence.....	31
Capture de l'image.....	31
Déploiement sur machine vierge.....	32
Difficultés techniques rencontrées et solutions adoptées.....	32
Tests de validation et résultats.....	32
Conclusion technique.....	32

1 — Introduction & Enjeux

Dans ce projet, j'ai mis en place une solution de déploiement automatisé de systèmes via le réseau à l'aide de **FOG Project**. L'objectif était de pouvoir créer une machine de référence (master), la capturer, puis la déployer sur des machines vierges sans intervention manuelle et de manière reproductible.

Ce type de mécanisme permet d'industrialiser l'installation de postes clients. Il évite le reformatage manuel poste par poste, réduit le temps d'installation, standardise les configurations et limite les erreurs humaines.

Pour isoler l'environnement et le rendre reproductible, l'infrastructure a été hébergée sur un **serveur physique** exécutant **Incus**, dans lequel j'ai créé le **PXE**, le **DHCP**, et **FOG**.

Ce rapport décrit **comment j'ai mis en place** l'infrastructure, **pourquoi**, **les difficultés rencontrées**, et **les preuves du fonctionnement**.

Vidéo youtube de la preuve du résultat: <https://youtu.be/ScFOFoE4QDQ>

2 — Architecture cible

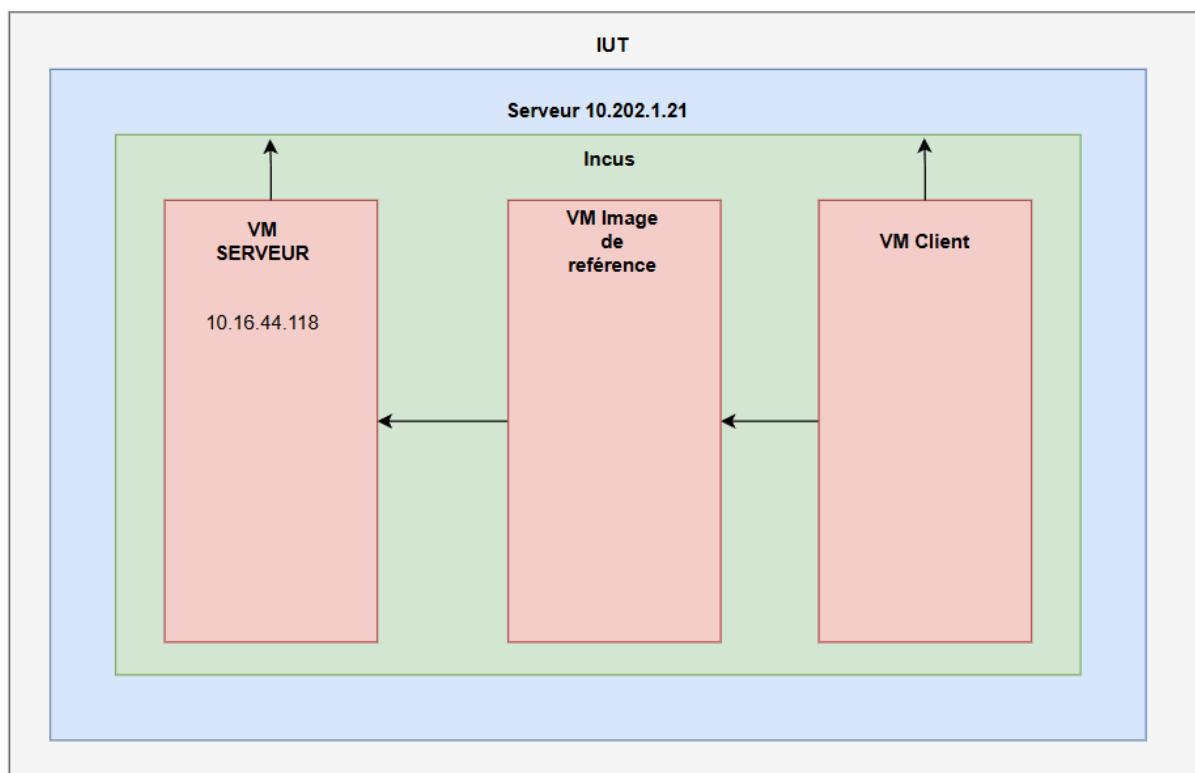
L'infrastructure a été conçue de manière à isoler entièrement le cycle de capture/déploiement tout en restant reproductible. Le serveur physique héberge un hyperviseur **Incus**, qui sert à créer et gérer les différentes machines virtuelles nécessaires au projet.

L'architecture finale se compose de trois machines virtuelles distinctes :

- **Serveur FOG** : infrastructure de déploiement (DHCP + TFTP + PXE + stockage d'images)
- **Master Debian 12** : machine de référence à capturer
- **Client vierge** : machine sans OS destinée à recevoir l'image

L'ensemble communique via un **bridge réseau Incus**, ce qui permet aux machines virtuelles d'échanger directement les paquets nécessaires au PXE et au DHCP, sans sortir du serveur physique.

Dans un contexte d'entreprise, ce type d'architecture permettrait d'éviter d'impacter le réseau de production (ex : éviter un conflit DHCP), mais dans ce projet le bridge servait essentiellement à reproduire le comportement d'un réseau isolé.



3 — Choix technologiques

3.1 – Hyperviseur : Incus

J'ai utilisé **Incus** comme hyperviseur, car c'est une solution orientée serveur, administrable en ligne de commande, capable d'héberger à la fois des conteneurs et des machines virtuelles. Contrairement à VirtualBox ou VMware Workstation, Incus ne nécessite pas d'interface graphique sur l'hôte, s'intègre bien à des workflows d'infrastructure et permet la création d'un réseau virtuel cohérent via bridge et vue que j'avais commencer a prendre en main durant la première SAE.

L'approche Incus → VMs permet de reproduire plus fidèlement un comportement proche d'un environnement d'entreprise sans dépendre de postes clients physiques.

3.2 – Solution de déploiement : FOG Project

Pour la partie capture/déploiement, j'ai choisi **FOG Project**.

FOG est une solution open-source complète incluant ,un serveur **DHCP** ,un serveur **TFTP**, un **boot PXE**, un stockage d'images, un système de capture basé sur **Partclone** et un moteur de déploiement automatisé.

Ce choix était pertinent, car FOG permet de **maîtriser toute la chaîne PXE**.

3.3 – OS master : Debian 12

Pour la machine de référence, j'ai utilisé **Debian 12**.

Ce choix présente plusieurs avantages techniques car Debian est **stable** et largement utilisé en entreprise et serveurs et contrairement à un master Windows, il n'était pas nécessaire d'utiliser Sysprep ou d'intégrer des drivers WinPE.

Et également le choix que j'ai fait pour mon pc personnel.

3.4 – Virtualisation réseau : Bridge Incus

J'ai choisi un mode réseau en **bridge** plutôt qu'en NAT ou Host-Only, car le PXE nécessite la réception directe des broadcasts DHCP (trames DISCOVER). En NAT, ces trames ne traversent pas la translation, ce qui rendrait le PXE inopérant.

Ce choix était donc une **contrainte technique obligatoire**.

4 — Mise en place du serveur d'infrastructure

4.1 Installation de l'hyperviseur Incus

J'ai installé **Incus** directement sur le serveur afin de disposer d'un hyperviseur sans interface graphique et intégrable dans une logique d'infrastructure. Ce choix permet de créer des VMs tout en contrôlant précisément le réseau virtuel, ce qui est indispensable pour le PXE.

Une fois installé, Incus a été initialisé via la commande :`incus init`

```
titouan@srvv1:~$  
titouan@srvv1:~$  
titouan@srvv1:~$  
titouan@srvv1:~$ sudo incus admin init  
[sudo] password for titouan:  
Would you like to use clustering? (yes/no) [default=no]: yes  
What IP address or DNS name should be used to reach this server? [default=10.202.4.69]: |
```

Ce qui a permis de créer les pools de stockage et la configuration de base.

4.2 Cration des machines virtuelles

Trois VMs ont été créées sur Incus :

une VM serveur FOG

une VM master Debian 12

une VM client vierge

Ces VMs sont visibles via : `incus list`

incus ls						
NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS	
client	RUNNING	10.16.44.128 (enp5s0)	fd42:c629:d551:8637:216:3eff:feae:6942 (enp5s0)	VIRTUAL-MACHINE	0	
client-recup	RUNNING	10.16.44.77 (enp5s0)	fd42:c629:d551:8637:216:3eff:fe4e:567d (enp5s0)	VIRTUAL-MACHINE	0	
client-video	RUNNING	10.16.44.64 (enp5s0)	fd42:c629:d551:8637:216:3eff:fed7:9867 (enp5s0)	VIRTUAL-MACHINE	0	
vm3	RUNNING	10.16.44.118 (enp5s0)	fd42:c629:d551:8637:216:3eff:fedf:83db (enp5s0)	VIRTUAL-MACHINE	0	

Ce qui permet de vérifier l'existence des VMs, leur état et leur adresse réseau.
Donc ici :

VM3 = Serveur FOG

client-recup = master Debian 12

Client et client-video étaient des clients vierges auquel le serveur PXE à déployer des images

4.3 Mise en place du réseau Incus

Pour permettre la capture et le déploiement via PXE, il fallait que les VMs communiquent au niveau **layer 2** (broadcast DHCP). J'ai donc configuré Incus pour utiliser un **bridge** réseau, faisant office de switch virtuel.

```
titouan@pxe-server:~$ incus network show incusbr-1000
config:
  ipv4.address: 10.16.44.1/24
  ipv4.dhcp: "true"
  ipv4.dhcp.ranges: 10.16.44.50-10.16.44.200
  ipv4.nat: "true"
  ipv6.address: fd42:c629:d551:8637::1/64
  ipv6.nat: "true"
description: Network for user restricted project user-user-1000
name: incusbr-1000
type: bridge
used_by:
- /1.0/instances/client-recup?project=user-1000
- /1.0/instances/client-video?project=user-1000
- /1.0/instances/client?project=user-1000
- /1.0/instances/vm3?project=user-1000
- /1.0/profiles/default?project=user-1000
managed: true
status: Created
locations:
- none
titouan@pxe-server:~$ |
```

Ce choix était nécessaire pour deux raisons :

1. PXE utilise des trames DHCP **DISCOVER**, qui ne passent pas en NAT
2. Le transfert d'image via Partclone nécessite un trafic direct entre le client et FOG

Sans bridge, le PXE ne fonctionne pas.

5 — Installation du serveur FOG et configuration PXE

5.1 Mise en place du serveur FOG

J'ai déployé le serveur FOG sur une machine virtuelle dédiée sous Linux. L'objectif de cette VM était d'assurer l'ensemble de l'infrastructure nécessaire au déploiement PXE, à savoir :

- DHCP (attribution IP aux clients)
- TFTP (envoi du bootloader PXE)
- HTTP/NFS (transfert des images via Partclone)
- Base de données (MySQL/MariaDB)
- Interface d'administration web

FOG a été installé en utilisant le script officiel fourni par le projet. Ce script automatise l'installation des dépendances et la configuration des services nécessaires :

```
git clone https://github.com/FOGProject/fogproject
cd fogproject/bin
sudo ./installfog.sh
[roo...@fog ~]# git clone https://github.com/FOGProject/fogproject.git
Cloning into 'fogproject'...
remote: Enumerating objects: 157898, done.
remote: Counting objects: 100% (3900/3900), done.
remote: Compressing objects: 100% (1056/1056), done.
remote: Total 157898 (delta 2858), reused 3706 (delta 2732), pack-reused 153998
Receiving objects: 100% (157898/157898), 849.04 MiB / 6.48 MiB/s, done.
Resolving deltas: 100% (112669/112669), done.
[roo...@fog ~]#
```

Pendant l'installation, le script demande quels services doivent être activés.

J'ai validé le mode : **DHCP + PXE via le serveur FOG** car dans mon architecture, FOG devait être le **seul serveur DHCP** du réseau virtuel.

5.2 Désactivation du Secure Boot

Le **Secure Boot** empêche le boot PXE avec iPXE, car le bootloader d'iPXE n'est pas signé Microsoft.

Lors des premiers essais, la machine virtualisée refusait le boot PXE tant que Secure Boot était actif.

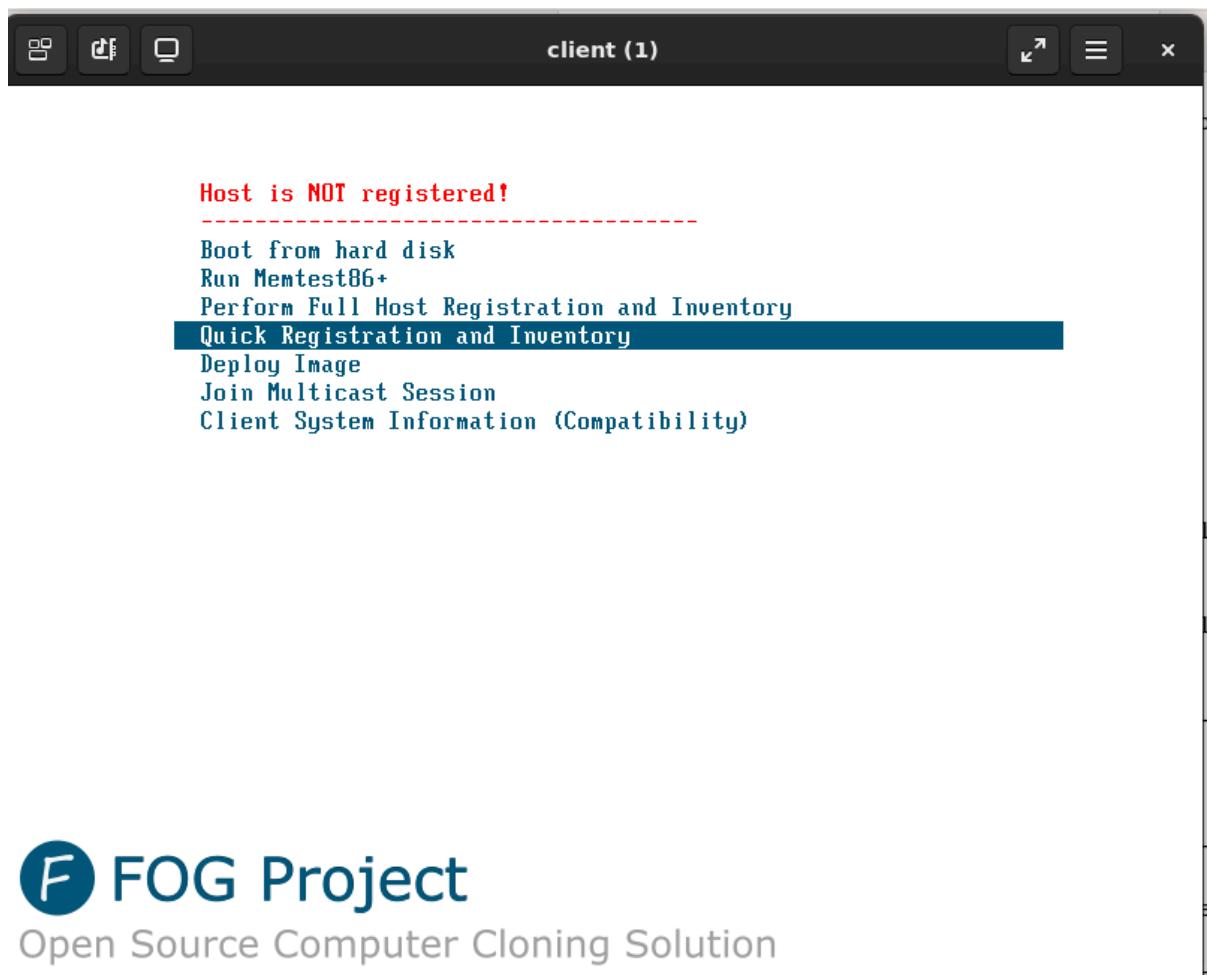
La solution a été de **désactiver Secure Boot**

```
titouan@pxe-server:~$ incus config set client-video security.secureboot=false  
titouan@pxe-server:~$
```

Cette étape est importante car PXE + Secure Boot ne fonctionne pas nativement.

5.5 Validation du service PXE

Une fois FOG installé et DHCP activé, j'ai effectué un premier test de boot PXE avec la VM cliente vierge.



Lors du premier boot PXE réussi, la VM a affiché le menu FOG avec les différentes options (Deploy, Registration, etc.), ce qui valide :

- DHCP opérationnel
- TFTP opérationnel
- Loader PXE fonctionnel
- Communication réseau entre client et FOG

5.6 Difficulté rencontrée — DHCP externe

Une première difficulté rencontrée avec FOG provient du fait qu'un réseau ne peut pas avoir **deux serveurs DHCP simultanés**.

Sur un réseau d'entreprise réel, il faudrait isoler FOG ou modifier les options DHCP existantes.

Dans mon cas, le réseau Incus était **isolé**, ce qui permettait à FOG d'être le seul DHCP. Ce choix simplifie énormément la configuration et permet de travailler dans un environnement contrôlé.

5.7 Interface d'administration

FOG fournit une interface Web permettant :

- l'enregistrement des clients
- la création d'images
- le déclenchement de captures ou déploiements

Après installation, l'interface était accessible via HTTP à l'adresse :

`http://<adresse_fog>/fog/`

6 — Cration de la machine de reference (Debian 12)

Pour constituer la machine de référence, j'ai créé une VM Incus et j'y ai installé **Debian 12**. Cette VM servait de master, c'est-à-dire que son image allait être capturée par FOG puis redéployée sur d'autres machines vierges. L'installation a été réalisée classiquement via ISO, avec un partitionnement automatisé et une configuration réseau en DHCP. Le fait que le bridge Incus disposait d'un accès Internet a permis d'installer des mises à jour et des paquets supplémentaires sans limitation.

L'usage de Debian a simplifié la phase de mastering : contrairement à Windows, il n'y a pas de problématique d'activation ni de Sysprep. Une capture Linux avec Partclone est propre tant que le système est cohérent et ne contient pas de secrets ou d'identifiants persistants. Avant la capture, j'ai surtout vérifié que la VM obtenait bien une adresse via le DHCP de FOG (ce point est indispensable, car FOG communique avec le master via PXE lors de la capture).

Une fois le système prêt, j'ai redémarré la VM en PXE et je l'ai **enregistrée** dans FOG via la fonction de **Host Registration**. Cette étape permet à FOG d'associer la machine au stockage d'images en fonction de son adresse MAC. Ensuite, via l'interface web de FOG, j'ai créé l'image correspondant au master en la configurant en mode :

Single Disk — Resizable / Linux / Partclone

Ce mode permet un redimensionnement automatique lors du déploiement et est compatible avec Debian.

La capture a ensuite été déclenchée depuis FOG. La VM a rebooté en PXE et FOG a lancé **Partclone**, qui a transféré le contenu du disque vers le serveur via NFS/HTTP. À la fin du processus, FOG a indiqué que l'upload était terminé et l'image Debian est alors devenue disponible pour les déploiements. À ce stade, la machine de référence était finalisée.

7 — Déploiement sur machine vierge

Pour valider la capture, j'ai créé une nouvelle VM Incus vierge, sans système d'exploitation. L'objectif était de vérifier si FOG était capable de réinstaller automatiquement l'image Debian précédemment capturée et de rendre la machine opérationnelle sans intervention manuelle.

Au premier démarrage, la VM a booté en PXE et a récupéré une adresse IP via le DHCP du serveur FOG. Le menu PXE est apparu, ce qui confirmait que le réseau et le chainload iPXE fonctionnaient correctement. La machine n'était pas encore enregistrée, donc j'ai utilisé l'option d'enregistrement afin que FOG puisse associer l'adresse MAC de ce poste à l'image à déployer.

Une fois la machine enregistrée, j'ai déclenché la tâche de déploiement depuis l'interface web de FOG en sélectionnant l'image Debian précédemment capturée. Après redémarrage en PXE, FOG a automatiquement lancé **Partclone** en mode restauration. Le disque de la machine vierge a été réécrit avec l'image Debian 12 issue du master, en utilisant le même mécanisme NFS/HTTP que pour la capture, mais dans le sens inverse.

À la fin du processus, la machine a redémarré seule. Debian 12 s'est lancé normalement, preuve que l'image avait été déployée correctement et que le système était fonctionnel. À ce stade, la machine vierge était une copie conforme du master. Comme le master était Debian, il n'y avait pas de problématique d'activation, de licence ou de Sysprep, ce qui simplifie la validation.

Ce test valide l'ensemble de la chaîne :

- PXE → fonctionnel
- DHCP → fonctionnel
- TFTP / iPXE → fonctionnel
- Capture → fonctionnelle
- Déploiement → fonctionnel
- Boot post-install → fonctionnel

Preuves attendues pour cette partie :

- [SCREEN – Boot PXE sur machine vierge]
- [SCREEN – Enregistrement du host dans FOG]
- [SCREEN – Lancement de la tâche Deploy dans FOG]
- [SCREEN – Partclone en restauration]
- [SCREEN – Boot Debian post-déploiement]

8 — Validation du déploiement

Après le déploiement de l'image sur la machine vierge, Debian 12 a démarré normalement. Le système était fonctionnel, identique au master, et utilisable immédiatement. Aucun correctif n'a été nécessaire. La machine a obtenu une adresse IP via DHCP et pouvait communiquer sur le réseau, ce qui confirme que la capture et le déploiement via FOG ont bien fonctionné.

Preuves attendues :

- [SCREEN – Boot Debian post-déploiement]
- [SCREEN – ip a sur machine déployée]

9 — Problèmes rencontrés et corrections

Le boot PXE ne fonctionnait pas en UEFI à cause du Secure Boot qui bloquait iPXE → il a été désactivé.

La configuration réseau Incus refusait l'ajout d'interface avec la commande classique des conteneurs → la bonne commande `network=<bridge>` a été utilisée.

Pour éviter les conflits DHCP, FOG a été configuré comme seul DHCP du bridge → le PXE a pu distribuer l'image correctement.

10 — Conclusion technique

L'infrastructure FOG installée sur une VM Incus a permis de capturer une image Debian 12 puis de la redéployer correctement sur une machine vierge via PXE. La chaîne complète du déploiement automatisé a fonctionné sans intervention manuelle après correction des prérequis (boot UEFI, DHCP unique, bridge Incus). Le résultat final est un master Debian reproductible et exploitable immédiatement après déploiement.

