Documentation IPXE

Sommaire :

Introduction

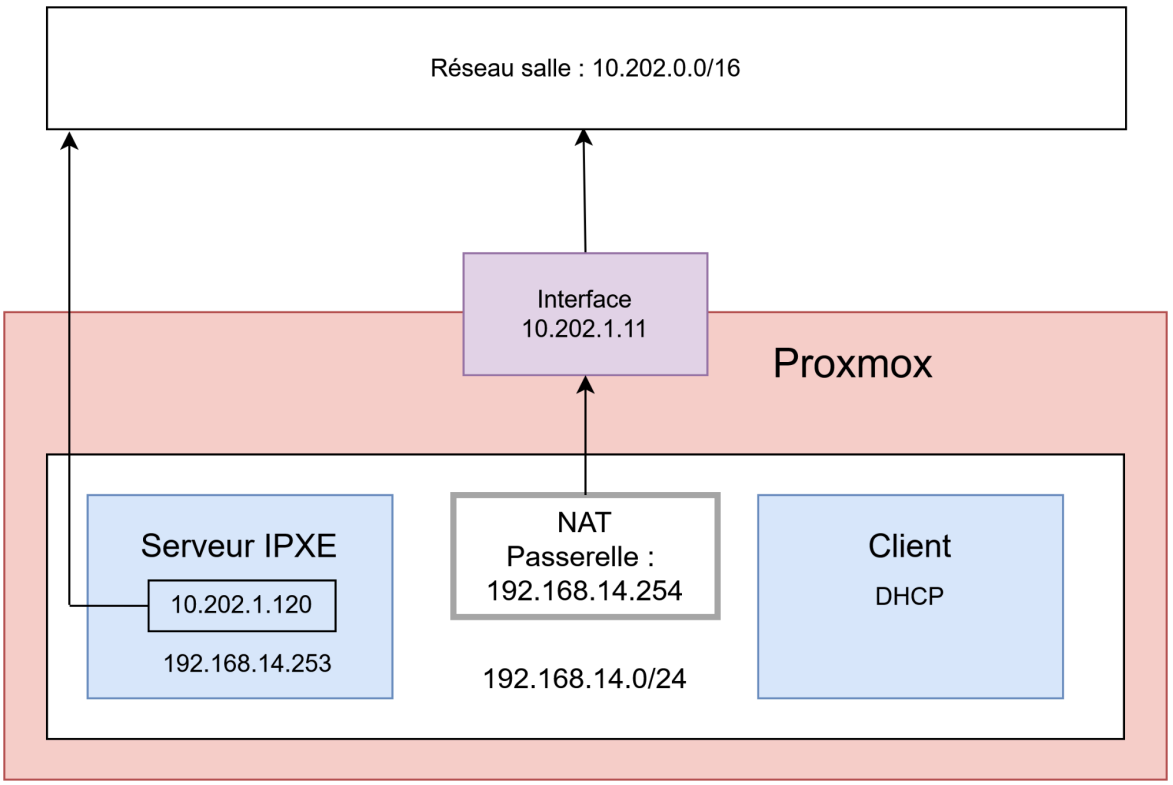
IPXE est un système de démarrage/d'amorcage. Il peut être comparé avec grub, Windows Deployment Service.
Son but est qu'au démarrage de la machine il récupère le système d'exploitation par le réseau.
Il est facile de l'utiliser afin d'utiliser différent protocole de téléchargement par http, nfs ou tftp.
Dans ce document nous allons utiliser pour serveur un ubuntu server 24.04 LTS. Nous y installerons un serveur dhcp, un tftp, un http, un nfs et l'outils ipxe.

Voici l'organisation :
root@ubuntu:/srv/tftp/boot# tree

```
.
├── configs
│   └── ubuntu-24.04.3-desktop-amd64
│       ├── meta-data
│       ├── user-data
│       └── vendor-data
├── images
│   └── ubuntu-24.04.3-desktop-amd64.iso
├── ipxe.efi
├── menu.ipxe
└── ubuntu-24.04.3-desktop-amd64
    ├── initrd
    └── vmlinuz

5 directories, 8 files
```

Schéma



Résumé de ce qu'il ce passe entre le serveur et le client :

Résumé de l'architecture

- **DHCP (Kea)** : Donne l'IP et dit au client "Charge ipxe.efi via TFTP".
- **TFTP** : Envoie ipxe.efi au client.
- **iPXE** : S'exécute sur le client, voit qu'il est "iPXE" et demande alors boot.ipxe via **HTTP**.
- **HTTP (Nginx)** : Envoie le script de boot, le Kernel et l'Initrd (très rapide).
- **NFS ou HTTP**: Fournit le système de fichiers (rootfs) pour faire tourner l'installateur Ubuntu.

Installation

Une fois l'installation de la machine Ubuntu server, voici les commande à effectuer

```
apt update && apt upgrade
apt install kea tftp-hpa nginx nfs-kernel git
```

Nous installerons dans un répertoire unique tout les fichiers pour le serveur IPXE.

```
mkdir -p /srv/ipxe/boot/
```

Installation du serveur DHCP Kea

Voici la configuration exact à entrer dans le serveur DHCP

```
{
  "Dhcp4": {
    "interfaces-config": {
      "interfaces": [
        "ens18"
      ]
    },
    "control-socket": {
      "socket-type": "unix",
      "socket-name": "/tmp/kea4-ctrl-socket"
    },
    "lease-database": {
      "type": "memfile",
      "persist": true,
      "name": "/var/lib/kea/dhcp4.leases",
      "lfc-interval": 3600
    },
    "expired-leases-processing": {
      "reclaim-timer-wait-time": 10,
      "flush-reclaimed-timer-wait-time": 25,
      "hold-reclaimed-time": 3600,
      "max-reclaim-leases": 100,
      "max-reclaim-time": 250,
      "unwarned-reclaim-cycles": 5
    },
    "renew-timer": 1800,
    "rebind-timer": 3150,
    "valid-lifetime": 3600,
    "option-data": [],
    "client-classes": [
      {
        "name": "ipxe",
        "test": "option[77].hex == 'iPXE'",
        "next-server": "192.168.12.253",
        "boot-file-name": "/boot/menu.ipxe"
      },
      {
        "name": "LegacyBIOS",
        "test": "substring(option[60].hex,0,20) == 'PXEClient:Arch:00000'",
        "next-server": "192.168.12.253",
        "boot-file-name": "http://192.168.12.253/boot/menu.ipxe"
      },
      {
        "name": "UEFIx64_PXE",
        "test": "not (option[77].hex == 'iPXE') and (option[93].hex == 0x0007 or option[93].hex == 0x0009)",
        "boot-file-name": "/boot/ipxe.efi"
      },
      {
        "name": "HTTP_UEFI_x64",
        "test": "option[93].hex == 0x0010",
        "boot-file-name": "http://192.168.12.253/boot/ipxe.efi",
        "option-data": [
```

```
            {
              "name": "vendor-class-identifier",
              "data": "HTTPClient"
            }
        ]
      }
    ],
    "subnet4": [
      {
        "id": 1,
        "subnet": "192.168.12.0/24",
        "pools": [
          {
            "pool": "192.168.12.1 - 192.168.12.99"
          }
        ],
        "option-data": [
          {
            "name": "routers",
            "data": "192.168.12.254"
          },
          {
            "name": "domain-name-servers",
            "data": "10.255.255.200"
          },
          {
            "name": "domain-search",
            "data": "iutbeziers.fr"
          }
        ],
        "reservations": []
      }
    ],
    "loggers": [
      {
        "name": "kea-dhcp4",
        "output_options": [
          {
            "output": "/var/log/kea/dhcp4.log"
          }
        ],
        "severity": "INFO",
        "debuglevel": 0
      }
    ]
  }
}
```

Nous pouvons voir les test qui détermine si la machine cliente utilise du BIOS ou de l'UEFI PXE ou de l'UEFI HTTP.

```
systemctl restart kea-dhcp4-server
```

Installation de HTTP

```
apt install nginx
nano /etc/nginx/nginx.conf
```

```
ubuntu@ubuntu:~$ cat /etc/nginx/nginx.conf
http {

    include             mime.types;
    default_type        application/octet-stream;

    charset             utf-8;
    sendfile            on;
    keepalive_timeout   65;
```

```
    server {
        listen          80  default_server;
        listen     [::]:80  default_server;

        location / {
            root   /srv/www/htdocs/;
            index  index.html index.htm;
        }

        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   /srv/www/htdocs/;
        }

        # Expose TFTP boot directory for HTTP boot
        location /boot {
            alias      /srv/ipxe/boot;
            autoindex  on;
        }
    }
}

events {
    worker_connections  1024;
}
```

```
systemctl restart nginx
chown -R www-data:www-data /srv/ipxe/
chmod 644 -R /srv/ipxe/
```

Installation de IPXE

Nous allons git clone le répo du projet IPXE puis nous allons compilé le fichier afin de ne pas utiliser un fichier en .efi d'une image linux.

```
sudo apt install build-essential liblzma-dev git -y
```

```
cd /home/ubuntu
git clone https://github.com/ipxe/ipxe.git
cd ipxe/src
for f in config/*.h; do touch "config/local/$(basename "$f")"; done
```

(Elle parcourt tous les fichiers .h du dossier config et crée un fichier vide du même nom dans config/local/. Cela "débloque" toutes les inclusions conditionnelles du code source)

S'il y a eu des tentatives échoué je conseil de remettre au propre le git avec :
make clean

Une fois installés je crée un fichier abc.ipxe

```
cat <<EOF > abc.ipxe
#!ipxe
dhcp
chain http://192.168.12.253/boot/menu.ipxe
EOF
```

Ce fichier sera importer dans le fichier que les client UEFI vont récupérer, il va pouvoir à ces client de récupérer le fichier menu.ipxe qui contiendra le choix des images à boot.
Cela nous évitera de recompiler à chaque fois le fichier des clients UEFI lors d'une modification dans menu.ipxe.

Pour les architecture x86_64 :

```
make bin-x86_64-efi/ipxe.efi EMBED=abc.ipxe
cp bin-x86_64-efi/ipxe.efi /srv/ipxe/boot/ipxe.efi
```

```
#!ipxe

# Définition des variables
set server_ip 192.168.12.253
set base_url http://192.168.12.253/boot
set nfs_path /srv/install/ubuntu
set config_url http://192.168.12.253/boot/configs/

:start
menu Installation Ubuntu 24.04 LTS
item --gap --             ------------------------ Gestion des Installations
------------------------
item --key u ubuntu_nfs   [U] Installer Ubuntu Desktop 24.04 (Via NFS)
item --key h ubuntu_http  [H] Installer Ubuntu Desktop 24.04 (Via HTTP/ISO)
item --gap --             ------------------------ Outils Avances
-----------------------------------
item --key s shell        [S] iPXE Shell (Ligne de commande)
item --key r reboot       [R] Redemarrer l'ordinateur
item --gap --
-------------------------------------------------------------------------------
choose --default ubuntu_nfs --timeout 15000 selected || goto start
#goto ${selected}



:ubuntu_install_nfs
echo Demarrage de l'installateur Ubuntu...
# Via NFS
kernel http://192.168.12.253/boot/ubuntu-24.04.3-desktop-amd64/vmlinuz
imgargs vmlinuz initrd=initrd boot=casper ip=dhcp netcfg/choose_interface=ens18 netboot=nfs
nfsroot=192.168.12.253:/srv/install/ubuntu autoinstall "ds=nocloud-net;s=${base_url}/configs/"
cloud-config-url=/dev/null rw --
initrd http://192.168.12.253/boot/ubuntu-24.04.3-desktop-amd64/initrd
boot
```

```
:ubuntu_install_http
# On charge le noyau et l'initrd via HTTP
kernel ${base_url}/ubuntu-24.04.3-desktop-amd64/vmlinuz
# Paramètres de boot pour l'ISO (nécessite que l'ISO soit accessible via HTTP ou montée)
imgargs vmlinuz initrd=initrd ip=dhcp cloud-config-url=/dev/null
url=http://192.168.12.253/boot/images/ubuntu-24.04.3-desktop-amd64.iso autoinstall
ds=nocloud-net;s=${base_url}/configs/ --

initrd ${base_url}/ubuntu-24.04.3-desktop-amd64/initrd
boot

:shell
shell
goto start

:reboot
reboot
```

Note cruciale sur l'argument ds : L'argument ds=nocloud-net;s=${config_url} indique à l'installateur de chercher user-data et meta-data à l'adresse spécifiée. Le ; est important et l'URL doit se terminer par un /

Installation du serveur TFTP

```
apt install tftpd-hpa
nano /etc/default/tftpd-hpa
```

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/ipxe"  # Votre dossier racine
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

```
chown -R tftp:tftp /srv/ipxe
chmod -R 755 /srv/ipxe

systemctl restart tftpd-hpa
systemctl enable tftpd-hpa
```

On peut vérifier si le service écoute bien sur le port 69 :

```
ss -uln | grep :69
```

Installation de NFS

```
apt install nfs-kernel-server
```

```
echo "/srv/ipxe/ 192.168.12.0/24(ro,sync,no_subtree_check)" | sudo tee
-a /etc/exports
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

Déploiement

Erreur

Si un Access Denied s’affiche sur l’écran il faut au préalable désactivé le sécure boot en mode UEFI.

Nous pouvons à présent profiter du serveur PXE linux comme on peut le voir dans la vidéo