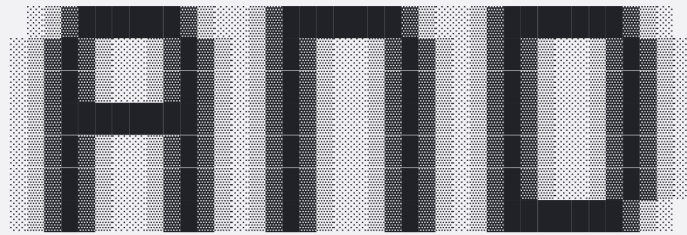
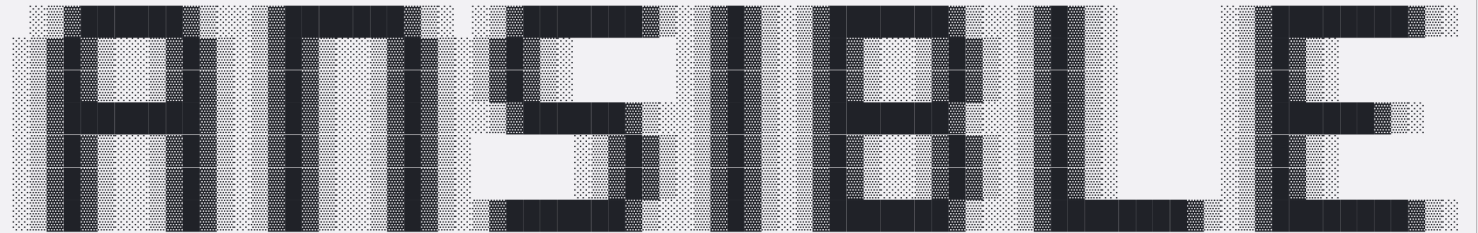


IPXE



AND



ANSIBLE

How I stopped carrying USB and love the network

Cenk Yildiz // SB-SPC IT  
[cenk.yildiz@epfl.ch](mailto:cenk.yildiz@epfl.ch)

# **Installing linux on a laptop**

## **a use case**

# Classical way

- Download iso image on a USB key
- Insert USB key in laptop
- Boot into installer, click, tap
- After first boot, clicks more, tap more

## Problem?

- Manual, error prone

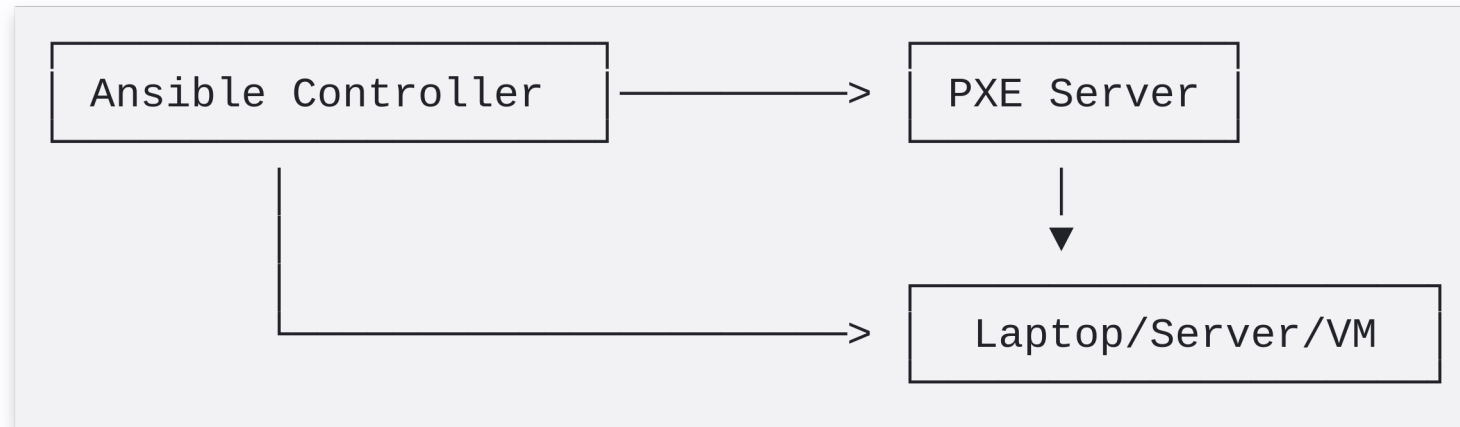


# Automated/Reproducible way

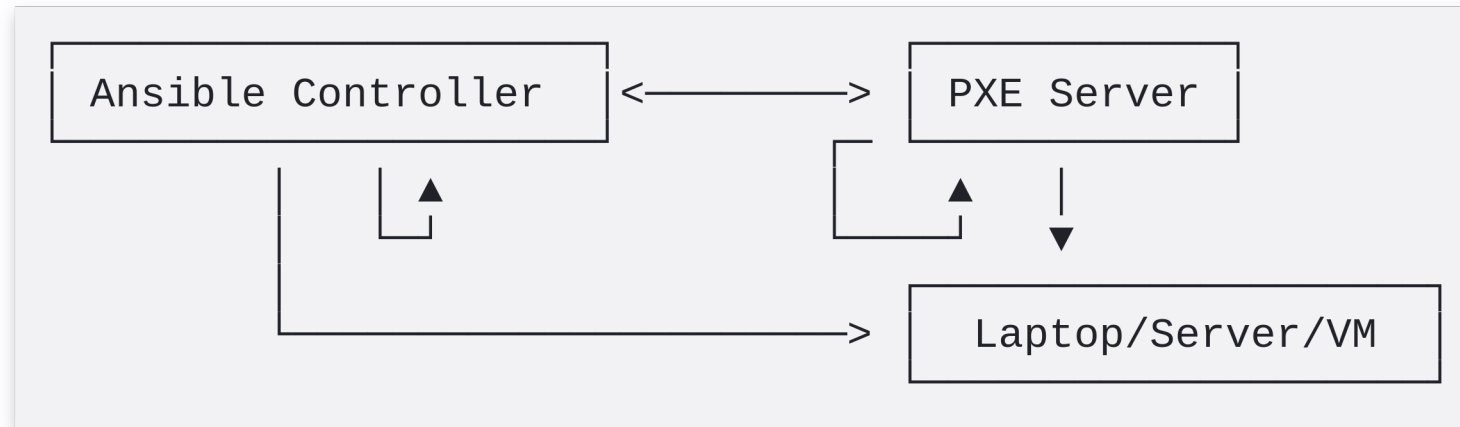
- Create a config file (user setup, custom software, language ...)
- Boot host into network, pick "Ubuntu Autoinstall" from the menu
- After first boot, provision with ansible (from config file)

# **Configuring a PXE Installation server with Ansible**

# Components



# Components - Reality



# Ansible Role: pxeserver

## Features

- Target: redhat 8/9 compatible host
- Configure networking (multiple interfaces)
- Download OS images (redhat, ubuntu, memtest, clonezilla)
- Install/start service: NFS, tftp, ipxe, web server, dnsmasq (dhcp/dns)
- Set up firewalld
- Install admin scripts, configure sudo rules

## Create config files from jinja2 templates

- dnsmasq configuration
- ipxe menu
- Automated installation files for RedHat/Ubuntu



# Ansible Project structure

From ansible best practices doc:

```
.
├── ansible.cfg
├── hosts
├── bin
│   └── install_ubuntu.sh
├── group_vars
│   ├── all.yaml
│   └── ubuntu-pc.yaml
├── host_vars
│   ├── host1.yaml
│   └── host2.yaml
├── playbooks
│   ├── pxe-server.yaml
│   ├── site.yaml
│   └── ubuntu-example.yaml
└── roles
    ├── common                -> Common role used in other roles
    ├── pxe-server            -> Provisions pxe server
    └── ubuntu                 -> Provisions Ubuntu Laptops/PCs
```

# Playbook to provision the PXE Server

```
# playbooks/pxeserver.yaml
- hosts: pxe01
  roles:
    - {role: pxe01}
```

Run as:

```
ansible-playbook playbooks/pxeserver.yaml
```

## Maintenance

```
# Replace red hat images with latest ones in the defaults file
ansible-playbook playbooks/pxeserver.yaml --tag=update-redhat

# Regenerate menu
ansible-playbook playbooks/pxeserver.yaml --tag=ipxe-menu
```

# Configuration Parameters

```
pxenetworks:
  - zone: 18
    interface: enp0s7
    ip: 10.0.18.0/24
  - zone: 16
    interface: enp0s8
    ip: 10.0.16.0/24
  - zone: 15
  ...

ubuntu_releases:
  - shortname: u2404
    longname: Ubuntu 24.04
    isourl: http://cdimage.ubuntu.com/ubuntu-server/noble/daily-live/current/
    isoname: noble-live-server-amd64.iso
    desktop: False
  - shortname: u2404d
  ...

redhat_releases:
  - shortname: rocky9
    longname: Rocky Linux 9
    isourl: http://mirror.init7.net/rockylinux/9/isos/x86_64/
    isoname: Rocky-9.3-x86_64-minimal.iso
    remoteurl: http://mirror.init7.net/rockylinux/9/BaseOS/x86_64/os/
  - shortname: alma9
  ...
```

# **Components of PXE server:**

**Dnsmasq**

**iPXE**

**Automated Installation**

# Dnsmasq

- Lightweight DHCP/DNS server
- Configuration built from Jinja2 template
- Sends the "ipxe" rom with DHCP to known MAC addresses

---

<https://thekelleys.org.uk/dnsmasq/doc.html>

## Example Jinja2 Template for dnsmasq configuration

```
# By default we have a single tag, where we show a general menu with all options (ubuntu, redhat ...)
dhcp-boot=tag:spcpxe,tag:ipxe,http://{{ pxeserver_ip }}:{{ pxehttpport }}/main_menu.php

# Any new boot script should be added here with a new tag
#dhcp-boot=tag:u2204auto,tag:ipxe,http://{{ pxeserver_ip }}:{{ pxehttpport }}/path/to/ubuntu2204_auto.php

{% for n in pxenetworks %}
dhcp-range={{ n.interface }},.178.{{ n.zone }}.0,static,1h
dhcp-option={{ n.interface }},option:router,10.0.{{ n.zone }}.1
dhcp-option={{ n.interface }},option:netmask,255.255.255.0

{% endfor %}
```

## Example dnsmasq configuration

```
# By default we have a single tag, where we show a general menu with all options (ubuntu, redhat ...)
dhcp-boot=tag:spcpxe,tag:ipxe,http://10.0.18.14:8080/main_menu.php

# Any new boot script should be added here with a new tag
#dhcp-boot=tag:u2204auto,tag:ipxe,http://10.0.18.14:8080/path/to/ubuntu2204_auto.php

dhcp-range=enp0s3,10.0.18.0,static,1h
dhcp-option=enp0s3,option:router,10.0.18.1
dhcp-option=enp0s3,option:netmask,255.255.255.0

dhcp-range=enp0s8,10.0.16.0,static,1h
dhcp-option=enp0s8,option:router,10.0.16.1
dhcp-option=enp0s8,option:netmask,255.255.255.0

dhcp-range=enp0s9,10.0.15.0,static,1h
dhcp-option=enp0s9,option:router,10.0.15.1
dhcp-option=enp0s9,option:netmask,255.255.255.0
```

# **iPXE**

## **(GIF incoming)**



# iPXE

```
VirtualBox temporary boot device selection
```

```
Detected Hard disks:
```

```
    AHCI controller:
```

```
      1) Hard disk
```

```
Other boot devices:
```

```
  f) Floppy
```

```
  c) CD-ROM
```

```
  1) LAN
```

```
  b) Continue booting
```

## PXE

- Preboot Execution Environment
- Uses DHCP and tftp
- Requires a PXE capable NIC

## iPXE

- Extends PXE
- Open source, loads via PXE or other sources (USB key), then takes over
- HTTP, NFS, iSCSI

### Example dnsmasq configuration

```
# If network card doesn't have ipxe tag, send undionly.kpxe for loading ipxe
dhcp-boot=tag:!ipxe,tag:!nopxe,undionly.kpxe
dhcp-match=set:ipxe,175 # iPXE sends a 175 option

# Special options for EFI clients https://tools.ietf.org/html/rfc4578#section-2.1
dhcp-match=set:x86_EFI,option:client-arch,7

# chainloading with ipxe.efi, after this, the ipxe should send the next image (undionly.kpxe)
dhcp-boot=tag:x86_EFI,tag:!nopxe,ipxe.efi

enable-tftp
tftp-root=/pxe/tftp
```

# **iPXE menu (GIF incoming)**

Please choose an operating system to install/run

Install RedHat flavored OSes

Ubuntu flavored OSes

Install Windows

Other OSes (Toolboxes/Clonezilla etc...)

Reboot

Boot from local disk

(24)

## jinja2 template

```
:MENU_REDHAT
menu Install/Run RedHat flavored OS
  item
  item MENU_TOP          return to main menu
  item
{% for item in redhat_releases %}
  item {{ item.shortname }}local      Install {{ item.longname }} from local repository
  item {{ item.shortname }}ks         Install {{ item.longname }} from local repository, using kickstart file (Installs base system)
{% if item.remoteurl|default("") != "" %}
  item {{ item.shortname }}remote     Install {{ item.longname }} from remote repository (may be more up to date)
{% endif %}
  item
{% endfor %}
choose --default MENU_TOP --timeout 25000 target && goto ${target}
```

```

:MENU_REDHAT
menu Install/Run RedHat flavored OS
  item
  item MENU_TOP          return to main menu
  item
  item rocky8local       Install Rocky Linux 8 from local repository
  item rocky8ks           Install Rocky Linux 8 from local repository, using kickstart file (Installs base system)
  item rocky8remote      Install Rocky Linux 8 from remote repository (may be more up to date)
  item
  item rocky9local       Install Rocky Linux 9 from local repository
  item rocky9ks           Install Rocky Linux 9 from local repository, using kickstart file (Installs base system)
  item rocky9remote      Install Rocky Linux 9 from remote repository (may be more up to date)
  item
  item alma8local        Install Alma Linux 8 from local repository
  item alma8ks            Install Alma Linux 8 from local repository, using kickstart file (Installs base system)
  item alma8remote       Install Alma Linux 8 from remote repository (may be more up to date)
  item
  item alma9local        Install Alma Linux 9 from local repository
  item alma9ks            Install Alma Linux 9 from local repository, using kickstart file (Installs base system)
  item alma9remote       Install Alma Linux 9 from remote repository (may be more up to date)
  item
  item rhel8local        Install Red Hat Enterprise Linux 8 from local repository
  item rhel8ks            Install Red Hat Enterprise Linux 8 from local repository, using kickstart file (Installs base system)
  item
  item rhel9local        Install Red Hat Enterprise Linux 9 from local repository
  item rhel9ks            Install Red Hat Enterprise Linux 9 from local repository, using kickstart file (Installs base system)
  item
  choose --default MENU_TOP --timeout 25000 target && goto ${target}

```

# Automated install

# Automated install - how does it work?

- Pass special options and a configuration file in the kernel line
- Base image is very minimal. Makes host ansible controllable: python, admin user, ssh keys...
- It could fully provision a host
  - less flexible than ansible
  - testing and error recovery is harder



# Automated PXE install - RedHat

- iPXE entry for passing Kickstart file

```
:rocky9ks
set url ${redhat_url}/rocky9
echo !..... This may delete everything on your disk .....!
echo !..... continue only if you are sure .....!

prompt --key c --timeout 10000 Press 'c' to continue before timeout || exit

kernel ${url}/isolinux/vmlinuz initrd=initrd.img ip=dhcp inst.repo=${url} \
      inst.ks=${redhat_url}/auto/rocky9_ks.cfg

initrd ${url}/isolinux/initrd.img

boot || goto MENU_REDHAT
```

# Automated PXE install - Ubuntu

- iPXE entry for passing Ubuntu server autoinstall file

```
:u2404auto
set url ${ubuntu_url}/u2404
set isourl ${ubuntu_url}/noble-live-server-amd64.iso
echo !..... This may delete everything on your disk .....!
echo !..... continue only if you are sure .....!

prompt --key c --timeout 10000 Press 'c' to continue before timeout || exit

kernel ${url}/vmlinuz initrd=initrd root=/dev/ram0 ramdisk_size=1500000 ip=dhcp url=${isourl} \
    autoinstall ds=nocloud-net;s=${ubuntu_url}/auto/generic/

initrd ${url}/initrd

boot || goto MENU_UBUNTU
```

# What else can one do?

- Rescue by booting into a custom rescue OS
- Diskless iPXE booted hosts with readonly-root from NFS
- Install OS on a old hardware, without mainline kernel support for certain drivers  
(See backup slides for examples)

# Pitfalls/Challenges

- Unless you have remote management(ipmi/idrac/ilo), you still have to setup bios for PXE booting
  - Let your hosts boot into network by default? Dangerous unless you have full control of network
- Laptops: MAC Address passthrough (MAC pre/post boot are not the same)

# Going back to Ubuntu Installation

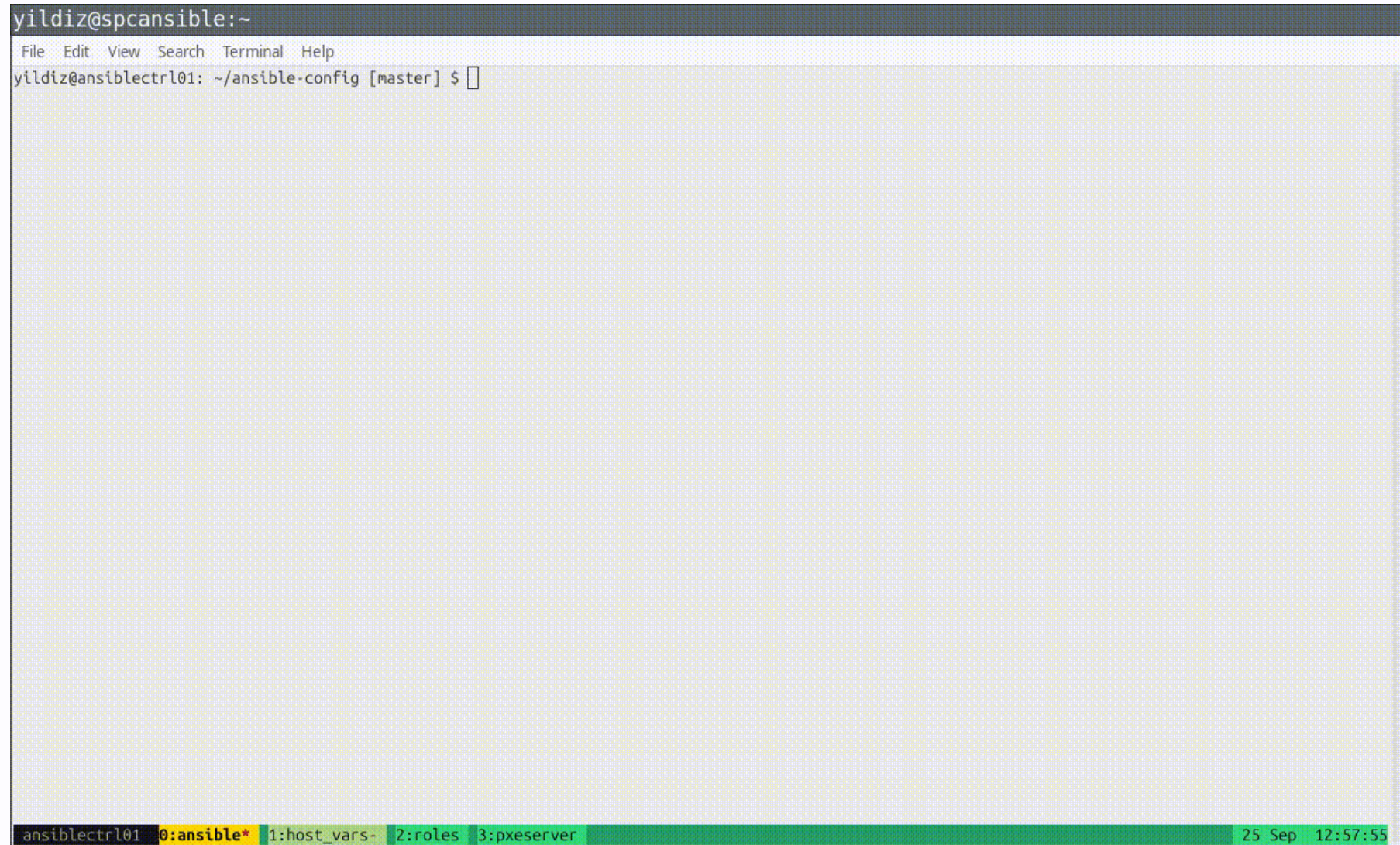
```
ssh admin@ansiblectrl01  
./bin/install_ubuntu.sh -M 01:23:45:a1:b2:c3 -u yildiz -h somehost.epfl.ch -l en
```

- Interact with dnsmasq on `pxesrv01` to add DHCP directive
- Instruct user to boot host/select "Ubuntu 2X.04 autoinstall" from iPXE menu
- After base os intall, generates/runs ansible configuration through:

```
hosts_custom  
host_vars/somehost.epfl.ch.yaml  
playbooks/somehost_ubuntu.yaml
```

```
- hosts: somehost.epfl.ch  
  roles:  
    - ubuntu
```

```
yildiz@spcansible:~  
File Edit View Search Terminal Help  
yildiz@ansiblectrl01: ~/ansible-config [master] $
```



ansiblectrl01 0:ansible\* 1:host\_vars- 2:roles 3:pxeserver 25 Sep 12:57:55

# Thank you

☀ slides by Marp ☀

Markdown Presentation  
Ecosystem

<https://marp.app/>

# Backup Slides



# Rescuing a remote system

- Remote rescue by booting into a different OS
- Change boot order (in linux if you have access, or via ipmi/idrac/ilo etc...)

```
# If you can still log in to OS
efibootmgr -o 0003,0001,0002
reboot

# otherwise, if you have remote console
ipmitool -H hosttorescue-mgt -U root -P insecurePass raw 0x0 0x8 0x5 0xC0 0x4 0x0 0x0
ipmitool -H hosttorescue-mgt -U root -P insecurePass power cycle
```

- Send dedicated rescue os image

```
dhcp-host=AA:BB:CC:00:11:22,10.0.18.5,hosttorescue,set:rescue
dhcp-boot=tag:rescue,tag:ipxe,http://pxesrv01/rescue.php
```

- Once booted, ssh to your rescue image (granted you have set up ssh keys, or password)

# Diskless hosts

```
#!/ipxe
# IPXE Boot file for Rocky Linux 9 based x-terminal.
# Its allows booting kernel from NFS

echo "Starting PXE boot for XTermR9"

set server_ip 10.0.18.1
set top_url http://${server_ip}:8080

set rootdir /pxe/nfs/XTermR9

set netboot9_nfs ${server_ip}:${rootdir}
set netboot9_nfs_url nfs://${server_ip}${rootdir}
set kernel_version 5.14.0-284.30.1.el9_2.x86_64
set initrd_base initramfs-${kernel_version}.img

# Direct NFS url, removes need to copy files to http server
set vmlinuz ${netboot9_nfs_url}/lib/modules/${kernel_version}/vmlinuz
set initrd ${netboot9_nfs_url}/boot/${initrd_base}

kernel ${vmlinuz} initrd=${initrd_base} ro ip=dhcp root=nfs4:${netboot9_nfs}:ro audit=1 LANG=en_US.UTF-8 selinux=0
initrd ${initrd}

boot || exit
END
```

# Auto Ubuntu Installer

```
ssh admin@ansiblectrl01  
./bin/install_ubuntu.sh -M 01:23:45:a1:b2:c3 -u yildiz -h somehost.epfl.ch -l en
```

- Interact with dnsmasq on `pxesrv01` to add DHCP directive
- Instruct user to boot host/select "Ubuntu 2X.04 autoinstall" from iPXE menu
- After base os intall, generates/runs ansible configuration through:

```
hosts_custom  
host_vars/somehost.epfl.ch.yaml  
playbooks/somehost_ubuntu.yaml
```

```
- hosts: somehost.epfl.ch  
  roles:  
    - ubuntu
```

# What else can one do?

- Rescue by booting into a custom rescue OS
  - efibootmgr
  - ipmi/idrac/ilo
- Diskless iPXE booted hosts with readonly-root
  - kernel, initrd from NFS, rootfs as (ro) NFS mount
  - rw volumes (/home etc.) mounted as tmpfs
- Install OS on a old hardware (see: `inst.dd=https://...` )

```
# For redhat, change kernel line in ipxe menu:  
kernel ${url}/isolinux/vmlinuz initrd=initrd.img ip=dhcp inst.repo=${url} inst.dd=https://address.of/driver.iso
```

