PXE

Askar Khabibullin

January 2020

1 Introduction

The following document contains the whole process of PXE server configuration to provide Ubuntu 18.04 Server installation as well as some details and specifications for the provided setup.

2 What is PXE?

It is an environment specification that boots a software assembly, retrieved from a network, on PXE-enabled clients. On the client side it requires only a PXE-capable network interface controller (NIC), and uses a small set of industry-standard network protocols such as DHCP and TFTP.

3 Typical PXE setup

Looking closer at TFTP definition, one can find out what's the purpose of the protocol. Trivial File Transfer Protocol (TFTP) is a simple lockstep File Transfer Protocol which allows a client to get a file from or put a file onto a remote host. One of its primary uses is in the early stages of nodes booting from a LAN. One of the main reasons I considered TFTP applicable is simplicity and wide amount of material provided on the topic.

4 Where this setup has been tested?

This setup was tried at my home LAN and has proven to work correctly. BIOS legacy boot device was connected to the LAN with working router and Internet connection. The server was a machine with Ubuntu 19.04, also connected to the same LAN. The setup can be reproduced by anyone having Legacy boot sequence, and network. The setup showed that all the requirements that were given were fully met. The system installation process was fully unattended and could be then checked by connecting to the client machine via ssh. (root login: root, pass: 12345678)

5 PXE chain

What happens at PXE boot in my configuration?

1) The PXE boot routine starts a DHCP discovery to finally get an IP address from the running DHCP Server

2) The PXE boot routine downloads the bootstrap program from the TFTP server. (The location of the bootstrap program is set in a DHCP-option)

3) The bootstrap program looks for a bootfile with the boot parameters for the current client.

4) Kernel and initial ram disk image(initrd.gz) are downloaded from the TFTP server and executed.

5) Apart from providing the kernel, the system also provides a preseed configuration file. The task was to provide a completely unattended installation, and the preseed.cfg serves this purpose.

*Initially the system ran an apache alongside the dnsmasq, to serve some additional files on the run, but this functionality was found redundant and it is not going to appear in the final version.

6 Assumptions

This setup was chosen due to some assumptions made about the possible target system.

```
1)  There has to exist a router which provides running DHCP
service, that can and ought not be changed. The server and
client machines are connected to some LAN of that router.
```

The LAN, that the sever is on has to have Internet connection, so that the server can download necessary files at runtime. In such scenario, we have to configure our DHCP to serve together with the initial working DHCP on LAN, providing the ranges, and managing the overlapping regions, but that must be done manually for each existing network configuration and might slow down the process of PXE deployment. It was chosen to either do this or use proxy in dnsmasq. Dnsmasq is a lightweight, easy to configure DNS forwarder, designed to provide DNS (and optionally DHCP and TFTP) services to a small-scale network. It can serve the names of local machines which are not in the global DNS. It is easy to install and configure on the server. File dnsmasq.conf is the only part that must be carefully considered and can provide one with a wide range of configurations, depending on what we're trying to achieve. In the setup that is provided the configuration is done for the user, all the user has to tell is the proper server's network interface that will be used to serve installation files. The dnsmasq.conf file is one of the essential parts of the system and can be found on the project's repository.

```
2)  The target system can be of different architectures.
```

As for the real system, there should be many configurations provided for the setup to work properly on different architectures. Mainly the configuration process will not change the setup, but some configuration files. For i686, arm and many other popular architectures the netboot files are easily found at archive.ubuntu.com, and if the user wants to add any other configurations, only the contents of dnsmasq.conf and setup files should be changed. For the given task, I included x86amd, but any other kernel might be found and added easily.

```
3)  DHCP regions in the initial setup can overlap with
existing DHCP server
```

For the current setup, the feature of dynamically adjusting dnsmasq was not implemented for DHCP ranges, as the system assigns IP addresses of a certain range. One can fix this by loading the dnsmasq.conf file and assigning the ranges considering the existing LAN setup manually.

```
4)  The user has root privileges and can build docker images
on the user side.
```

For the service to initialize and work, user must install some additional software, such as docker-io, and have access to execution root commands.

```
5) Client machine is better to be BIOS.
```

As for now, the system has been put to test and showed it's performance only on BIOS setting. Although some work has been done in order to provide the client with UEFI setting, it has not shown itself to work properly during the test time.

7 How to run the system?

0) Connect your sever PC to the router, that provides your network with access to the Internet using ethernet cable.

0.1) Connect your target PC to the same existing LAN.

1) Download one and the only file script.sh file and allow its execution via

```
wget https://raw.githubusercontent.com/Asterka/PXE-automated
-installation/master/setup

chmod a+wxr setup
```

2) Execute the file with a working network interface as a parameter. The network interface should be attached to the computer that is going to work as a server. The network should also have another computer, which we will install our system, to be a part of it. The example for enp0s25 is:

```
./setup enp0s25
```

3) User should be privileged when executing the main configuration files, and if step one was done correctly and assumptions from step 2 are not broken, the system should do all the configuration after running the ./setup

8 How does the setup file work?

The setup file does the following:

1) **Downloads the kernel** files from the vendor for UEFI and Legacy installation. Unzips the files into a current directory for two types of architectures : ARM, AMD.

2) Downloads configuration files, namely **'dnsmasq.conf', 'preseed.cfg', 'Dockerfile', and 'default'** files. This document will consider the purpose of those a little later.

3) **Patches the configuration files** with the details of the system. (Assigns a static IPv4 address, that is going to be used in the configs later.)

4) Copies essential configuration files into the directories where they will be **grabbed by the docker** from.

5) **Creates a new docker image** with the provided configurations, without using cached files. (this turned out to be a cornerstone at some point)

6) **Stops all the containers** that could stop it from running correctly. (That is, the container that could be running occupying the **:53 port**, mainly the previous versions of this server)

7) **Runs the image in a new container**, providing the network configuration of the host machine.

9 Brief explanation of the work in the background

What is the purpose of dnsmasq.conf file? It contains both TFTP and DHCP configurations for the target server. When running the './setup enp0s3' we pass the interface as a parameter, which is then read and taken IP from. This IP is put into some placeholders (gateway, dhcp-boot sevices), along with the interface name itself, in the dnsmasq.conf file.

```
interface=interface_name,lo #specifies the interface
bind-interfaces #allows to run process copies
domain=mydom.local #specifies the domain

#-- Set dhcp scope (could have been better,
see Assumptions 3) )
dhcp-range=192.168.0.160,192.168.0.200,255.255.255.0,2h

#-- Set gateway option
```

```
dhcp-option=3,192.168.0.155 - use preconfigured address as a gateway

Allow different boot configurations for both BIOS and UEFI:

dhcp-match=x86PC, option:client-arch, 0 #BIOS x86
dhcp-match=BC_EFI, option:client-arch, 7 #EFI x86-64

# Load different PXE boot image depending on client architecture
pxe-service=tag:x86PC,X86PC, "Install Linux on x86 legacy BIOS", /boot/boot_amd/pxelinux
pxe-service=tag:X86-64_EFI,X86-64_EFI, "Install Linux on x86-64 UEFI", bootx64.efi


# Set boot file name only when tag is "bios" or "uefi"
dhcp-boot=tag:x86PC,pxelinux.0  # for Legacy BIOS detected by dhcp-match above
dhcp-boot=tag:X86-64_EFI,bootx64.efi # for UEFI arch detected by dhcp-match above

#--enable tftp service
enable-tftp

#-- Root folder for tftp
tftp-root=/tftp
```

What is the purpose of 'default', and 'grub.cfg' file? The 'default' and grub.cfg file contains setup screen entries as well as system file paths that are going to be considered at boot time by the remote machine. It also contains a path to retrieve the preseed file from.

What's the purpose of the 'preseed' file? It is very essential that the process of installation is unattended, meaning the user does not have to interact with the remote PC. **preseed** file contains the parameters that the system will initialize itself with at boot time. It also contains hashed passwords that will be used in the system, additional software installation procedures, and last, but not the least **late install system configuration**. For this task it was necessary to access ssh after the installation.

10 Github page

Configuration files can be found on the project's Github repository:
https://github.com/Asterka/PXE-automated-installation