Give Your PXE Wings!

It’s not magic! How booting actually works.

Presentation for virtual SREcon 2020
By Rob Hirschfeld, RackN
Rob Hirschfeld
@zehicle
Co-Founder of RackN

We created Digital Rebar
Bare Metal Provisioning ++

@L8istSh9y Podcast on PXE:
In concept, Provisioning is Easy!

We’re just installing an operating system on a server or switch!

Why is that so hard?!

- Bootstrapping
- Firmware Limitations
- Variation
- Networking
- Security
- Performance
- Post configuration
In concept, Provisioning is Easy!

We’re just installing an operating system on a server or switch!

Why is that so hard?!

- Bootstrapping
- Firmware Limitations
- Variation
- Networking
- Security
- Performance
- Post configuration

And that’s not even including:

- System Inventory
- System Validation
- Hardware Configuration
- Naming & Addressing
- Credentials Injection
## Exploring Provisioning Approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netboot</td>
<td>(25 min)</td>
</tr>
<tr>
<td>● PXE</td>
<td></td>
</tr>
<tr>
<td>● iPXE</td>
<td></td>
</tr>
<tr>
<td>● ONIE</td>
<td></td>
</tr>
<tr>
<td>● Kickstart</td>
<td></td>
</tr>
<tr>
<td>● Preseed</td>
<td></td>
</tr>
<tr>
<td>Image Deploy</td>
<td>(10 min)</td>
</tr>
<tr>
<td>● Packer</td>
<td></td>
</tr>
<tr>
<td>● Write Boot Part</td>
<td></td>
</tr>
<tr>
<td>● Cloud Init</td>
<td></td>
</tr>
<tr>
<td>Esoteric Flavors</td>
<td>(5 min)</td>
</tr>
<tr>
<td>● kexec</td>
<td></td>
</tr>
<tr>
<td>● Secure Boot</td>
<td></td>
</tr>
<tr>
<td>● BMC Boot</td>
<td></td>
</tr>
</tbody>
</table>
Exploring Provisioning Approaches

Netboot (25 min)
- PXE
- iPXE
- ONIE
- Kickstart
- Preseed

Image Deploy (10 min)
- Packer
- Write Boot Part
- Cloud Init

Esoteric Flavors (5 min)
- kexec
- Secure Boot
- BMC Boot

All roads lead to a kernel init process
**PXE**

DHCP - Get Address and Boot Info

DHCP exchange in PXE is usually a 2 sets of two packets: Discover / Offer and then Request / Ack. The DHCP server responds with the bootfile and the next server to contact through TFTP. We are currently working out what should work here. We currently use ipxelinux.0 and the DRP Endpoint IP. This process is repeated often.

The DRP DHCP can also act as a Proxy DHCP server that will handle file management for the existing DHCP server that hands out IPs. This allows for DRP to handle files without managing IPs.

TFTP - Get initial bootloader

If DRP is the DHCP server, the bootfile is specified based upon the DHCP client options (EFI or legacy, arch support). Also, if local boot is requested, the DRP DHCP server will NOT respond with a boot file.

Additionally, the DRP DHCP server will attempt to use ipxe where possible. The DHCP client options are parsed to determine if ipxe with the correct options is available. If so, then the bootfile is set to default.ipxe (it must be provided by the unknown bootenv). If not, then the correct ipxe (EFI or legacy BIOS) is sent to chain boot. If all else fails or overridden by configuration, ipxelinux.0 is sent as the bootfile.

BOOT - Boot the initial bootloader and get configuration

Ipxelinux 0 starts a progression of file requests until it gets one that works. These include IP and MAC based filenames and then a default filename. Ipxe also uses a similar progression. These are all over TFTP to start. This process may be upto 12 file requests. These files allow for a machine specific or default fall through configuration.
Let’s PXE!
Bootstrapping is a multi-stage process

- **Server**
  - Firmware
    - PXE
  - Bootloader
    - Stage 1
  - Bootloader
    - Stage 2
  - O/S Kernel

- **Provisioning Service(s)**
  - DHCP
  - NextServer
    & Options
  - TFTP
    - ipxelinux.0
  - HTTP(S)
    - ipxe.efi
  - HTTP(S)
    - O/S ISO

- **O/S ISO**
First: Get on the network
Then: Download a Bootloader
Then get a BETTER Bootloader
Finally load a “real” operating system
Each stage is actually a NEW O/S Load

Server
- Firmware PXE

Bootloader Stage 1
- PXE
- DHCP
- HTTP(S)
- TFTP
- ipxelinux.0

Bootloader Stage 2
- DHCP
- HTTP(S)
- ipxe.efi

O/S Kernel
- HTTP(S)
- O/S ISO

Provisioning Service(s)
- HTTP(S)
- O/S ISO
- DHCP
And modern servers can skip TFTP!
So... technically, no longer PXE
Yay! We’re done, right?
Provisioning is more than PXE
Hardware varies, so Install must be guided by templates
ISOs are minimal and stale
So they must be updated
And then you can actually connect to start configuring the system!
Automating Provisioning means Connecting all these steps together
But wait.... There's more to consider!

Server
- O/S Kernel
- Kickstart
- Installation
- Post-Config

Out of Band Management (BMC, IPMI, Redfish, etc)

Infrastrastructure as Code

Provisioning Service(s)
- iPXE
- Bootloader
- O/S ISO
- Config Templates
- Download Packages
- Access & Apps
IaC?
Show us some templates!
Typical PXE Questions

Why is this so fragile?
What about PXE over Wifi?
What about using a VLAN?
Can I dockerize this?
What about setting BIOS & RAID?
How can I make this faster?
How can we simplify that?!!

At RackN, we've been using an in memory operating system, “sledgehammer,” based on CentOS.

It’s highly optimized to

- Run on nearly any hardware
- Load very quickly
- Collect deep inventory
- Have built-in tools for system tasks like hardware config
Image Based Deployment (10x faster!)

- **Server**
  - PXE
  - RAM only O/S
  - Write O/S To Drive(s)
  - reboot
  - Informed Installation

- **PXE/iPXE**
  - Small Footprint
  - O/S Image as Archive
  - Machine Init

- **Provisioning Service(s)**
Immutable Provisioning

Server
- PXE
- RAM only O/S
- Config & Attach Disks
- Load Apps & Containers

Highly Available Provisioning Service(s)
- Minimal Footprint
- Machine Initialize
- Container Registries

Load Apps & Containers
And now...

Advanced Provisioning!
ESXi Provisioning

Server

- O/S Kernel
- weasel
- restricted CLI/python
- ESXi

Provisioning Service(s)

- iPXE
- Bootloader
- O/S ISO
- Config Templates
- Access & Apps
- Control via VMw APIs

VMware Tooling
ONIE: Open Network Install Environment

Designed for Embedded Systems where we’re replacing the O/S as a complete image.

Does have DHCP options for a startup script.
kexec (kernel execute)

- Server
  - Running O/S
  - New Kernel
    - kexec
  - New O/S
- Normal Provision
- Download New Kernel
- Starts Without Rebooting
- Provisioning Service(s)
kexec (kernel execute)
Secure Boot Required SIGNED Bootloaders

Server

- Secured Firmware
- Trusted Bootloader
- Trusted O/S Kernel

TPM

- Enabled
- Verified
- Verified

DHCP

- NextServer & Options
- Signed ipxe.efi
- Signed O/S ISO

HTTP(S)

Provisioning Service(s)
BMC Boot option 1

Server

- Firmware
- Attached O/S Install
- Kickstart
- Installation
- Post-Config

BMC

Install Media

Provisioning Service(s)

Download Packages
Access & Apps
Thanks!

Contact us:
Rob Hirschfeld, RackN.com

Digital Rebar
Behind the Firewall, Self-Service Infrastructure as Code

Self-Trials: rebar.digital