Linux auf Filialgeräten— ein Use Case mit Foreman, Puppet und Gitlab

Timo Goebel | 04.10.19
Who am I

- Timo Goebel
- Foreman Core Committer
- Rubyist
- Sysadmin
- github.com/timogoebel
Agenda

- Project Goals
- Tools
- Control repo
- Boot process
Project Goals
Business Goals

- Deploy a Linux operating system on point of sale hardware all over Europe
- Manage the lifecycle of these 15,000 devices
- No budget for software licenses

Scenario:

- Cyberdyne Systems has 4,000 stores in Europe
- Sells robots
- Every store has
  - 2 to 8 points of sale that should run Linux
  - A broadband connection to the corporate datacenter
• We can’t migrate to the cloud as the hardware has to run locally in the stores
Technical Goals

- Automate everything
- Test everything
- Use open source tools
Challenges

- No management of the DHCP server possible
- Possibly unreliable Network
Tools
Foreman

- Inventory Database
- handles the OS installation via rich templating engine
- Integrates with
  - DNS
  - Puppet
- Reporting
<table>
<thead>
<tr>
<th>Power</th>
<th>Name</th>
<th>Operating system</th>
<th>Puppet Environment</th>
<th>Model</th>
<th>Host group</th>
<th>Last report</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>allan-deschner.example.com</td>
<td>RHEL 7.1</td>
<td>ESX-LX-LAN</td>
<td>Test</td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>alton-buttner.example.com</td>
<td>RHEL 7.1</td>
<td>ESX-LX-LAN</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>anna-massella.example.com</td>
<td>RHEL 7.1</td>
<td>ESX-LX-LAN</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>blossom.example.com</td>
<td>redhat 7.4</td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>cara-kotba.lxbx.ka.de.dm-drogeriema...</td>
<td>RHEL 7.1</td>
<td>production</td>
<td>ESX-LX-LAN</td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>ejujuj.example.com</td>
<td>RHEL 7.1</td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>ffff-fffff-01.lxbx.ka.de.dm-drogeriema...</td>
<td>RHEL 7.1</td>
<td></td>
<td>ESX-LX-LAN</td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>flora-klenke.example.com</td>
<td>RHEL 7.1</td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>floyd-falker</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>garry-hodgkiss.example.com</td>
<td>RHEL 7.1</td>
<td>RollingUpdate</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>gene-valorie.example.com</td>
<td>CoreOS 1445.7.0</td>
<td>ESX-LX-LAN</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>host-sample-01.example.com</td>
<td>RHEL 7.1</td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>irvin-erteita.lxbx.ka.de.dm-drogeriema...</td>
<td>RHEL 7.1</td>
<td></td>
<td>ESX-LX-LAN</td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>jake-nippert.example.com</td>
<td>RHEL 7.1</td>
<td>RollingUpdate</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>jess-aasvedo.example.com</td>
<td>RHEL 7.2</td>
<td>ESX-LX-LAN</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
<tr>
<td></td>
<td>jon-jamieson.example.com</td>
<td>RHEL 7.1</td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td>xEdit</td>
</tr>
</tbody>
</table>
kind: iPXE
name: Kickstart default iPXE
model: ProvisioningTemplate
oses:
- CentOS
- Fedora
- RedHat

%% subnet = @host.subnet -%%
%% if subnet.respond_to?(:dhcp_boot_mode?) && subnet.dhcp_boot_mode? -%%
%% static = '' -%%
%% else -%%
%% static_arg = 'static=yes' -%%
%% static = (@host.token.nil? ? '?' : '&') + static_arg -%%
%% end -%%

%% stage2 = host_param('kickstart_liveimg') ? 'inst.stage2=' + @host.operatingsystem.medium_uri(@host).to_s : '' -%%

kernel "[@host.url_for_boot(:kernel)]" initrd=initrd.img ks="foreman_url('provision')" inst.stage2="@host.operatingsystem.medium_uri(@host).to_s" instinitrd="[@host.url_for_boot(:initrd)]"
Puppet

- Config management

- Uses a pull approach, which scales better – especially with an unreliable network
Gitlab

- Hosts the git repositories
- Runs CI Pipelines
Test Kitchen

- Can run serverspec tests in a docker container
Docker

- Provides a container to simulate an endpoint and run tests inside
iPXE

• Boot Firmware

• We want to do as little TFTP as possible

• Can retrieve Kernel/Initrd via HTTP(s)
The control repo
Single point of control

- Contains
  - Puppet Code
  - Test definitions
  - CI definitions
  - Data (Hiera)
Test driven

11 before_script:
12   - apk add bash build-base git
13   - bundle install --without kitchen
14
15 puppet-syntax:
16     stage: syntax
17     script:
18        - /bin/puppet_check_syntax.sh all
verify-centos-7:
  services:
    - docker:dind
  before_script:
    - echo "$\{PUPPET_EYAML_PRIVATE_KEY\}" > hiera_keys/private_key.pkcs7.pem
    - cat hiera_keys/private_key.pkcs7.pem | head -n 1
    - apk add build-base docker libffi-dev patch git
    - docker info
    - bundle install --with kitchen
  stage: functional
  artifacts:
    expire_in: 7d
    when: on_failure
    paths:
      - .kitchen/logs
  script:
    - kitchen test default-centos-7 -d always
Test driven

```ruby
require 'serverspec'

# Required by serverspec
set :backend, :exec

describe package('vim-enhanced') do
  it { should be_installed }
end

describe package('tree') do
  it { should be_installed }
end
```
Test driven

- We can easily test
  - if our config works with different OS versions
  - If a code change breaks a special configuration
  - in a local (debug) environment

```yaml
platforms:
  - name: centos-7
    driver_config:
      image: centos:7
      platform: centos
      run_command: /usr/sbin/init

verifier:
  name: serverspec
  default_pattern: true

suites:
  - name: default
```
The site folder

- Contains
  - Custom Puppet Modules
  - Roles
  - Profiles

  - profile
  - role/manifests
  - trustanchor
The Puppetfile

- Contains

- Puppet Modules from the forge

```plaintext
forge 'https://forgeapi.puppetlabs.com'
mod 'blackknight36-chrony', '0.2.3'
mod 'camptocamp-systemd', '2.1.0'
mod 'puppet-yum', '4.0.0'
mod 'puppetlabs-inifile', '2.4.0'
mod 'puppetlabs/stdlib'
mod 'saz-timezone', '5.0.2'
mod 'firewalld',
git: 'https://github.com/crayfishx/puppet-firewalld.git',
ref: 'master'
```
Code deploy

- Deploy to Gitlab Pages is easy
- No branch support, yet.
- Won’t support a phased rollout.
Code deploy

- Deploy Code to Puppetmaster(s)
- Create Puppet environment in Foreman
  
  hammer --fetch-ca-cert https://foreman.example.com/

  hammer environment create --name #{env} --organizations #{organization} --locations #{location}

  hammer proxy import-classes --environment #{env} --name #{puppetmaster_name}
The boot process
The boot process

- Store computers boot via PXE
  - DHCP Request to (local) router
  - Router tells the network card firmware
    - The IP-Address of a TFTP server
    - A filename on the TFTP server
  - TFTP is very trivial, not a good fit for WAN
Prerequisites

• Stores have a router running Cisco iOS
• Router is a pretty dumb DHCP server
• Router can server files locally via TFTP
• Let’s serve „undionly.kpxe“ (iPXE) via TFTP, distribute via Ansible

Router#sho run | begin pool 1
ip dhcp pool 1
network 192.168.1.0 255.255.255.0
bootfile undionly.kpxe
next-server 192.168.1.1
default-router 192.168.1.1
dns-server 192.168.1.1 192.168.1.2
domain-name store.de.example.com
lease 0 4
Breaking the infinite loop

- When iPXE starts, it issues a DHCP request to determine what to do next
- The DHCP server (being dumb) tells iPXE to load iPXE via tftp
- We’re stuck in an infinite loop

- We can use an embedded script to break the loop
Infinite Loop Com Flow

NIC → iPXE → Router

DHCP-Request

Reply: Boot iPXE

DHCP-Request

Reply: Boot iPXE
Embedded Script

- We need to build our own iPXE image

```plaintext
#!/ipxe

dhcp
chain http://boot.ipxe.org/demo/boot.php

make bin/undionly.kpxe EMBED=demo.ipxe
```
We need to build our own iPXE image

```plaintext
#!ipxe
dhcp
chain
http://foreman.example.com/unattended/iPXE?mac=${net0/mac}
```
iPXE Templates

• Install SLES Kernel/Initrd
• Boot from disk
• Unknown Host
• What we bake into iPXE
• Install RHEL Kernel/Initrd
• Install Debian Kernel/Initrd
Limitations

- How to configure the iOS device to serve either "undionly.kpxe" for BIOS boots or "ipxe.efi" for EFI boots?
Thank you.