Diskless CentOS Install

Authors:
Jennifer Beattie
Edmund Sutcliffe

Summary

This document provides detailed steps to set up a diskless CentOS image that can be booted over a local Ethernet.

We assume that a basic PXE environment has already been set up as per another document in this series, the Hive PXE Install document.

This setup will allow multiple nodes to be booted from the same NFS read-only root image, so that a single root image can be used for multiple servers. Each server still has a portion of per-server writable space. This approach is popular in larger ISPs, and we aim to provide a particular enterprise-oriented formulation of it.

Overview

Creating a diskless boot image is an established process in a UNIX environment, and the technical requirements fall into several fundamental stages:

- Pre-reqs: ensure a basic PXE environment and install image is available on the build machine.
- Create the basic diskless root image: create a directory to hold the image, and copy all required packages into it.
- Customize the root image: fixup config files, set a root password, and turn off unwanted services.
- Create an NFS bootable kernel: on CentOS, use the system-config-netboot tools to generate a kernel and initrd image.
- Make the install image network-bootable: Configure NFS and PXE to point at the newly generated kernel and root image, and boot a diskless VM using it.

Prerequisites

You must have a build machine with a PXE boot environment installed as per an earlier document in this series, the Hive PXE Install document.

You must have a PXE bootable target server to disklessly boot, and have DNS set up with records for the boot server and target server names.

Filesystem Paths and Network Addresses

Several of the pathnames used in this install process are configurable. This document assumes particular locations for these, and for ease of reading, the explanations below use these paths without qualifying them as potentially variable names.

The following paths are used as standard:

TFTP root directory	`/tftpboot`
NFS base directory	`/data/bootimage`
HTTP boot server filesystem location	`/data/bootserv`
CentOS HTTP repository root	`/data/bootserv/centos`

The network topology is assumed to be on a private network `10.1.4.0/24` in this example. The machines being created are intended to be part of a domain `test.example.com`.

We further assume that the build machine will be acting as the DNS nameserver for the local network, and this is reflected in the resolv.conf we create. The build machine itself is named `core` in this example (ie `core.test.example.com`) and is assumed to be on `10.1.4.1`.

Process

The following steps should be followed in sequence, once the pre-requisites have been met.

Create the basic diskless image

In this guide all our diskless data is placed under `/data` .
Inside this container, we create a base directory to hold the diskless boot files we generate. Inside *this*, we create a directory to hold the readonly root image, which must be called `root` when working with the CentOS builtin utilities.

```
mkdir -p /data/bootimage/root
```

Now, in order to ensure that we can create an image from the specific set of packages in the ISO, we need to work round CentOS' default use of network mirrors, by creating our own local repository for yum to use. We do this by adding a file in `/etc/yum.repos.d` .

```
cat > /etc/yum.repos.d/core.conf  <<END
[core]
name=CentOS Core Install Package
baseurl=http://core/bootserv/centos
END
```

We have assumed that the repository has been made available over HTTP, as per the original Hive PXE Install document, to allow other network kickstart usage, although for the purposes of this document a local `file://` repository would work just as well.

Next we need to fix up a couple of files inside our root image so that the package install scripts do not complain.

```
touch /data/bootimage/root/etc/fstab
mkdir /data/bootimage/root/proc
mount -t proc /proc /data/bootimage/root/proc
```

Now we can use yum to install our desired set of packages into the root image.
In this guide we install everything for simplicity; other builds can be constructed using the `Base` package group to form a minimal install.
We turn off every yum repository except for our `core` repo, to control the install sources.

```
yum -y --disablerepo=\* --enablerepo=core \
    --installroot=/data/bootimage/root groupinstall '*'
```

After this step, you should find a fairly complete root image (with `bin`, `usr`, `etc`, `var` directories and so on inside `/data/bootimage/root` .

Customize the diskless image

Now that we have a basic set of binaries, there are some configuration files that we need to edit.

First of all, we need to turn on readonly root support:

```
cat > /etc/sysconfig/readonly-root <<END
READONLY=yes
TEMPORARY_STATE=yes
RW_MOUNT=/var/lib/stateless/writable
RW_LABEL=stateless-rw
STATE_LABEL=stateless-state
STATE_MOUNT=/.snapshot
END
```

The above ensures that the various init scripts do not try and write to files marked readonly at boot time. For example, `/etc/rc.d/rc.sysinit` will not try and update the local random seed file in `/var/lib/random-seed` if `READONLY=yes` .

Next, in this build, we turn off SELinux support, because we want standard UNIX security behavior:

```
cat  > /data/bootimage/root/etc/selinux/config <<END
SELINUX=disabled
SELINUXTYPE=targeted
SETLOCALDEFS=0
END
```

We must also ensure that a bare bones DHCP network setup and DNS config are available:

```
cat > /data/bootimage/root/etc/resolv.conf <<END
search test.example.com
nameserver 10.1.4.1
END

cat > \
/data/bootimage/root/etc/sysconfig/network-scripts/ifcfg-eth0 <<END
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
END
```

Note that the file /etc/sysconfig/network will be created by the diskless utilities later, with the value `keep_old_ip=yes` in it.

Next we need to set a root password inside the image. We need to know the crypted value of the root password. If required, you can use the program `grub-md5-crypt` to produce one from a given password.

For example, to set a crypt value `$1$LUOTZ/$yF5BEnm2lGU8XUhMdshx/1` in the image:

```
/sbin/chroot /data/bootimage/root usermod \
  -p  '$1$LUOTZ/$yF5BEnm2lGU8XUhMdshx/1' root
```

Now we need to turn off services which are not relevant to a diskless install, or which may cause problems.

```
for i in iscsi iscsid pcscd avahi-daemon avahi-dnsconfd       \
  firstboot yum-updatesd xend xen-domains libvirtd smartd; do \
  /sbin/chroot /data/bootimage/root chkconfig $i off
```

If desired, you can also edit `/etc/inittab` to change the default runlevel from 3 (text mode) to 5 (graphical).

```
perl -pi -e's,id:3:initdefault:,id:5:initdefault,' /etc/inittab
```

Create a bootable diskless kernel

Now that we have prepared our diskless root, we can use the CentOS/RedHat utilities to build an initrd that is NFS aware and can locate and boot into the given image when network loaded.

We must ensure that `busybox-anaconda` is available inside the root image, and we need the `system-config-netboot` package available in the build machine:

```
yum -y --disable-repo=\* --enable-repo=core \
    --installroot=/data/bootimage/root install busybox-anaconda
yum -y --disable-repo=\* --enable-repo=core \
    install system-config-netboot
```

Now we can use the `mkdiskless` and `updateDiskless` tools to create the initial ramdisk. When we invoke these tools, we are passing the following arguments:

- `/data/bootimage/root` is the root of our readonly image
- `2.6.18-164.el5` is the version string of the kernel, as seen on the end of the kernel image we want to build from in: `/data/bootimage/boot/vmlinuz-2.6.18-164.el5` .
- `/data/bootimage` is the directory where we want to place the resulting kernel and initrd files.

```
/usr/share/system-config-netboot/diskless/mkdiskless \
    /data/bootimage/root
/usr/share/system-config-netboot/diskless/updateDiskless \
    /data/bootimage/root 2.6.18-164.el5 /data/bootimage/
```

Once this step is complete, you should see files `vmlinuz` and `initrd.img` inside `/data/bootimage`. You should also see a new subdirectory named `snapshot` as a peer of the root image directory. It contains a text file called files . This holds the name of every file which needs to be writable (and therefore a separate copy kept per diskless server):

```
/data/
      bootimage/
            root/
            snapshot/
                  files
                  files.custom
```

We can add additional per-server writable files to this list by creating files.custom in the snapshot directory. Here we only want to add the machine-id file which the dbus service uses to get a unique identifier per running OS instance:

```
cat >> /data/bootimage/root/snapshot/files.custom <<END
/var/lib/dbus/machine-id
END
```

Now we have a root image which should be capable of booting over NFS without errors.

Make the diskless image network bootable

Now, we need to make sure that our network-bootable kernel and initial ramdisk can be loaded via the TFTP server:

```
ln -s /data/bootimage/vmlinuz /tftpboot/vmlinuz
ln -s /data/bootimage/initrd.img /tftpboot/initrd.img
```

Next we need to ensure that the NFS server is available, configured and reachable on the build server. This requires configuring the daemons to use certain ports, and then opening only those ports in the firewall. That detail is available in a separate document but omitted here for simplicity. If on a private test network you can *completely* open the firewall temporarily with `iptables -F` , naturally taking care of the risks involved.

```
yum -y install portmap nfs-utils
chkconfig --level 345 portmap on
chkconfig --level 345 nfs on
service nfs start
service portmap start
```

Now we come to a crucial stage - we must configure the NFS exports file to grant access to the diskless filesystem. Since we need a read-only mount for the shared root filesystem, and a read-write mount for the per-server writable files, we add two lines to the exports file. In the following example, we allow access to any host on the private subnet:

```
cat >> /etc/exports <<END
/data/bootimage/root        10.1.4.0/24(ro,async,no_root_squash)
/data/bootimage/snapshot    10.1.4.0/24(rw,async,no_root_squash)
END

/usr/sbin/exportfs -ra
```

Note in particular that there is no space between the subnet declarations and the options contained within brackets; this is an important point to remember about the exports file syntax.

Now we can configure our PXE boot menu file to point at the NFS bootable kernel and add the NFS parameters to the kernel boot line. For example:

```
cat >> /tftpboot/pxelinux.cfg/default <<END
label centos
  kernel vmlinuz
  append initrd=initrd.img load_ramdisk=1 network \
         root=/dev/nfs NFSROOT=10.1.4.1:/data/bootimage
END
```

Notice in particular that the NFS root parameter ***must*** be in upper case, and the value provided for the NFS root directory must be the common parent of the `root` and `snapshot` directories.

Presuming that the target machine is configured in dhcpd.conf as per the earlier PXE Install document, then you can now network boot the target, guide it to the `centos` boot option (or make that the default), and test a boot. The target system should come up and allow root login using the root password you defined.