PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

MORE Kickstart Tips & Tricks

Chip Shabazian
Vice President / Consultant II
Bank of America
June 23rd, 2010

What is kickstart

Kickstart is an automated method to build servers that utilize the anaconda build tools. By utilizing ks.cfg files, server builds can be consistent with no user interaction required to bring a machine from bare metal to fully functional and ready for deployment

Why use kickstart

- Streamline builds
- Enforce build consistency
  - Packages
  - Partitions
  - Configurations
  - Monitoring
  - Security
- Rapid bare metal deployment
- Reduce human errors

Kickstart sections

A kickstart file is comprised of four sections:

- The Command Section
  - System configuration
- %packages
  - Package manifest
- %pre
  - Commands run before the build
- %post
  - Commands run after the build

User input during a kickstart

```
%pre
#!/bin/sh
exec < /dev/tty3 > /dev/tty3 2>&1
chvt 3
install="no"
while [ "$install" != "yes" ]; do
   clear
   echo
   echo '********************************************************************************'
   echo '*                                  W A R N I N G                               *'
   echo '*                                                                              *'
   echo '*This process will install a completely new operating system and             *'
   echo '*                                                                              *'
   echo '*  Do you wish to continue?  (Type the entire word "yes" to proceed.)          *'
   echo '*                                                                              *'
   echo '********************************************************************************'
   echo
   read -p "Proceed with install? " install done clear chvt 1 #%end
```

More instructions at http://www.trueblade.com/techblog/user-input-during-a-fedora-kickstart

The command section

This is where system configuration items are set. Required items for new builds include:

- auth

–Authentication options for the system

- bootloader

–bootloader options (grub is now the only bootloader used)

- keyboard

–keyboard type, eg: us

The command section (continued)

- part
  - Disk partitioning scheme
- rootpw --iscrypted
  - root password. Create encrypted hash using grub-md5-crypt
- timezone
  - Timezone for this server. --utc or specific TZ
- lang
  - language for the install, eg: en_US

The command section (continued)

Other non-required but typical kickstart entries include:

- clearpart

–This will remove existing partitions for a clean install

- driverdisk

–Add drivers for additional components

- firewall

–--enabled or --disabled

–Specific rules can also be included

The command section (continued)

- install
  - Do a fresh install, not an upgrade
  - Specify the install source
- network
  - Configure one or more NICs
- reboot
  - Reboot the server after the install
- repo
  - Add additional repo's for build currency and additional packages

The command section (continued)

- selinux

–--disabled | --enforcing | --permissive

- skipx

–Don't configure X

- text

–Don't use the GUI

The command section %include

%include can be one of the most powerful additions to any part of your kickstart file, and can be used in the command section to include information that is either dynamically generated or retrieved in the %pre

%packages

The %package section is used to create your manifest of rpm packages:

- Default is Base
- Use --nobase to skip base packages
  - Not recommended without adding back in AT LEAST
    - @core
    - yum
    - openssh-server
- Use --ignoremissing to avoid user interaction for missing packages and groups. Best to resolve all dependencies though.

%packages (Continued)

- Use @ for groups
  - Both Base and groups can be viewed by examining the comps.xml file

- Add or remove specific packages
  - add_this_packagename
  - -don't_install_this_packagename

- To install multiple architectures of an rpm, use the arch in %packages
  - EX: beecrypt.i386

To query and see what the architecture of an installed package is, use queryformat

EX: : rpm -q --queryformat '%{NAME}-%{VERSION}-%{RELEASE}-%{ARCH}\n‘

- Yes, you can use %include in %packages too

%pre

- --interpreter
  - To change from default bash
  - /usr/bin/perl
  - /usr/bin/python

LOTS of customization and automagic can be performed in the %pre section.

- Create the network line
- Partition the disks
- Retrieve configurations over the network
- What do YOU need to do?

%pre - Create the network line

If you use dhcp, or even if you assign the IP address on the boot: line, you can statically assign the address to the server:

In the command section

```
%include /tmp/buildnet
```

In %PRE

```
ETH=`grep DEVICE /tmp/netinfo | cut -d = -f 2`

IP=`ifconfig $ETH | grep inet | cut -d : -f 2 | cut -d " " -f 1`

NETMASK=`ifconfig $ETH | grep inet | cut -d : -f 4`

GATEWAY=`route | grep default | cut -b 17-32 | cut -d " " -f 1`

HOSTNAME=`grep HOSTNAME /tmp/netinfo | cut -d = -f 2 | cut -d . -f 1`


cat << EOF > /tmp/buildnet

network --device $ETH --bootproto static --ip=$IP --netmask=$NETMASK

--gateway=$GATEWAY --hostname=$HOSTNAME

EOF
```


%pre - Partition the disks

You can decision based on number/type/size of drives:

In the command section

```
%include /tmp/buildnet
```

In %PRE

```
# Determine how many number/type/size of drives we have
set $(list-harddrives)
let numd=$#/2 # This will provide the total # of drives
d1=$1               # This is the device of disk 1
d2=$3               # This is the device of disk 2, etc.
S1=$2               # This is the size of disk 1
S2=$4               # This is the size of disk 2, etc.
```


%pre - Partition the disks (continued)

You can decision based on number/type/size of drives:

In %PRE (continued)

```
# This would be a partition scheme for two or more drives
if [ $numd -ge 2 ] ; then cat << EOF >> /tmp/partinfo part pv.01 --size=1 --grow --fstype=ext3
    --ondisk=$d1 volgroup volgrp01 pv.01 part pv.02 --size=1 --grow --fstype=ext3 --ondisk=$d2
    volgroup volgrp02 pv.02 #HOWEVER_YOU_WANT_TO_PARTITION EOF

else

cat << EOF >> /tmp/partinfo part pv.01 --size=1 --grow --fstype=ext3 --ondisk=$d1 volgroup volgrp01
    pv.01 #HOWEVER_YOU_WANT_TO_PARTITION EOF

fi
```


%pre - Retrieve configurations over the network

Using tools such as wget, you can retrieve items to be included in the command section

In the command section

```
%include /tmp/file_to_include
```

In %PRE

```
wget http://server/info_to_include -O /tmp/file_to_include
```


%pre – More %pre

Get values from your kickstart file

grep /tmp/ks.cfg file

Get values from the kickstart commandline

grep /proc/cmdline

touch /tmp/kickstarting

Use this as a trigger in your rpms

Getting values from the command line in %pre

Stick this at the top of the %pre section, and it will take anything of the form var=value from /proc/cmdline and turn it into a variable you can use directly

```
set -- `cat /proc/cmdline`

for I in $*; do case "$I" in *=*) eval $I;; esac; done
```

Example: ks system=hp san=yes

Booting the System

You may kickstart your system using any method to boot the system:

- Local Drive

–Copy vmlinuz and initrd.img from boot.iso and add it to grub using grubby. Great for upgrades

```
grubby --title=rebuild5 --add-kernel=/boot/vmlinuz5 \

-c /etc/grub.conf --initrd=/boot/initrd5.img \

--args="ks=http://path/to/ks.cfg text ksdevice=eth0 \

ramdisk_size=8192 noipv6"
```

- PXEBoot

–This is the preferred method for kickstarting large numbers of systems. PXEBoot is built into most modern systems and allows bare metal builds without the need for media.

Booting the System (Continued)

- CD
  - Disk 1 has a minimal boot.iso image that can be used to boot a system. If network connectivity is an issue, you can embed your kickstart file in the initrd:

```
mkdir initrd
cd initrd
gzip -dc path/to/unzipped/initrd.img | cpio -id
# Make the changes you need
find . | cpio -c -o | gzip -9 > /path/to/new/initrd.img
```

  - You will need to use mkisofs to regenerate your boot.iso image after embedding a ks.cfg

Booting the System (Continued)

- USB
  - Disk 1 also contains diskboot.img which can be copied to a USB drive using dd:

```
dd if=/path/to/diskboot.img of=/dev/your_usb_drive
```

(if your usb drive is sda, use /dev/sda)

  - Once the usb boot media is created, you can mount it and copy across your kickstart files without the need to embed them in the initrd

Useful boot: options

- text
  - Launches a text mode installation
- ks=
  - Location of the kickstart file
- nofb
  - Turns off frame buffer. Needed for HP DL installs
- dd=
  - Load device drivers over the network for the build
- nostorage
  - Prevents disk from loading. One solution to SAN issues

Other boot: options

Documented options can be found in Appendix A. These boot: options aren't found in the online documentation, but may be useful:

- nicdelay and linksleep
  - nicdelay=50 linksleep=50
  - Supposed to delay bringing up the NIC for XX seconds to allow portfast negotiation. YMMV
- ethtool
  - eth0_ethtool="autoneg=on speed=1000 duplex=full"
- latefcload
  - Supposed to load fiber channel last to ensure you don't build over existing SAN data. YMMV

Where to get your kickstart file

- NFS
  - ks=nfs:ip_address:/path/to/ks.cfg
- Local Disk
  - ks=hd:sda3:/ks.cfg (must be vfat or ext2)
- Floppy / CDROM
  - ks={floppy,cdrom}:/ks.cfg # use floppy OR cdrom
- HTTP / FTP
  - ks={http,ftp}://your.server.com/path/to/ks.cfg # use http OR ftp
- Embedded in initrd
  - ks=file:/ks.cfg

Yummy builds

New in RHEL 5

- Allows a stable base install, while providing a current build
- Can be used to load non distribution RPMs in %packages instead of %post

```
repo --name=updates --baseurl=http://your.server.com/updates/5

repo --name=my_extras --baseurl=http://your.server.com/extras/5
```


%post

--nochroot

- Used in the pre-chrooted environement. Useful for copying over information from the build environment to the built environment

- --interpreter

- /usr/bin/python
- /usr/bin/perl

NOTE: You can run multiple %post sections. This is useful if you want to first run a --nochroot to copy information from the build environment to the built environment, Then want to run scripts in the chrooted environment.

%Post

Since Anaconda doesn't keep a log of what you do in post (or copy %post into the archive kickstart file it creates in /root) lets do it ourselves:

```
# Let's log everything:

tail -F /root/post-install.log &

(

All of your post commands

) 2>&1 >> /root/post-install.log
```

The hostname isn't set in the built environment until after the reboot:

```
# set the hostname for apps that need it

export HOSTNAME=`grep HOSTNAME /etc/sysconfig/network | awk -F= '{print $2}'`

hostname $HOSTNAME
```


%Post (Continued)

Make sure to rev your kickstart file, and log it on the servers you build:

```
echo 2010062301 > /etc/my_release_file
```

Set the speed and duplex on your nics

```
for i in `ls /etc/sysconfig/network-scripts/ifcfg-eth*`
 do
  INTERFACE=`ls $i | awk -F - '{ print $3 }'`
  SPEED=`ethtool $INTERFACE  | grep Speed | awk '{ print $2 }' | awk -F Mb \
'{ print $1 }'`
   if [ "$SPEED" = "1000" ] ; then
    echo 'ETHTOOL_OPTS="autoneg on speed 1000 duplex full"' >> $i
   elif [ "$SPEED" = "100" ] ; then
    echo 'ETHTOOL_OPTS="autoneg off speed 100 duplex full"' >> $i
   fi
 done
```


Getting values from the command line in %post

Stick this at the top of the %post section, and it will take anything of the form var=value from /proc/cmdline and turn it into a variable you can use directly

```
set -- `cat /proc/cmdline`

for I in $*; do case "$I" in *=*) eval $I;; esac; done
```

Example: ks update=yes dmz=yes

Troubleshooting

- Bus Enumeration RHEL 4.0 – 4.4. Changed back in 4.5
- put your %pre or %post to sleep
  - sleep 999999999
  - open a terminal with alt-f2 and manually walk through your build
- Alt-f{2,3,4}
- Portfast issues

Troubleshooting

RHEL 5 now requires cpio to edit the initrd

```
mkdir initrd
cd initrd
gzip -dc path/to/unzipped/initrd.img | cpio -id
# Make the changes you need
find . | cpio -c -o | gzip -9 > /path/to/new/initrd.img
```


Common problems

- Portfast issue
  - Compounded by the NIC being cycled three times
- Drivers in the initrd need to be added or removed
  - Bust open your initrd and edit the /modules/module-info file
- updating loader
  - Allows a stable base build while taking advantage of newer anaconda options
- 255 character limit on boot: line
  - Anything beyond 255 characters will simply be ignored. Use wget in %pre to work around

Don’t wipe out your SAN

When rebuilding a server with SAN attached storage, your SAN disks will have a higher priority than your internal disks, and anaconda will install the OS onto the SAN wiping out anything that is there unless you make the SAN inaccessible.

There are two ways to avoid wiping out your SAN

1) Use the nostorage option on the kickstart command line and add the storage that you need
    1) `In ks.cfg add “device scsi cciss”`
    2) `In %pre, modprobe cciss`
2) Remove the driver modules from modules.cgz in the initrd
    1) `Requires a rebuild of the initrd.img`

Tools to make life easier

- RPM
  - Package your stuff
- Cobbler
  - Setup a powerful kickstart infrastructure
- Revisor
  - Roll your own
- mRepo
  - Setup and manage your repositories

Tools to make life easier

- Kickstart Configurator
  - Somewhere to start
- Yum
  - Creating and managing repo's
- PXEBoot
  - A must for scalable deployments
- ksvalidator
  - Script to test your ks.cfg for validity

RPM

Packaging your deployments is an excellent way to deploy and manage system configuration. Company specific configuration items can be wrapped into an rpm and deployed at build to ensure consistency and security from the outset.

Good packaging guidelines can be found at:

http://fedoraproject.org/wiki/Packaging/Guidelines

Cobbler

Probably the best package available for building your own provisioning infrastructure:

- Imports a build tree
- Configures PXEBoot
- Can manage DHCP
- Can build via profiles
- Able to kickstart xen instances
- Remotely rebuild “enchant” systems
- Very active development

http://cobbler.et.redhat.com/documentation.php

Revisor

From http://revisor.fedoraunity.org/[1]

Revisor enables you to customize and compose your own Fedora based installation and live media. It does so by presenting you a GUI with all options you can click you way through, and a CLI and extended configuration files for the more advanced users. Features that Revisor has vary from customizing the packages available during the installation and/or installed on the live media, to fully customizing the desktop environment for the live media.

Video demos: http://revisor.fedoraunity.org/media

mRepo

From http://dag.wieers.com/home-made/mrepo/[2]

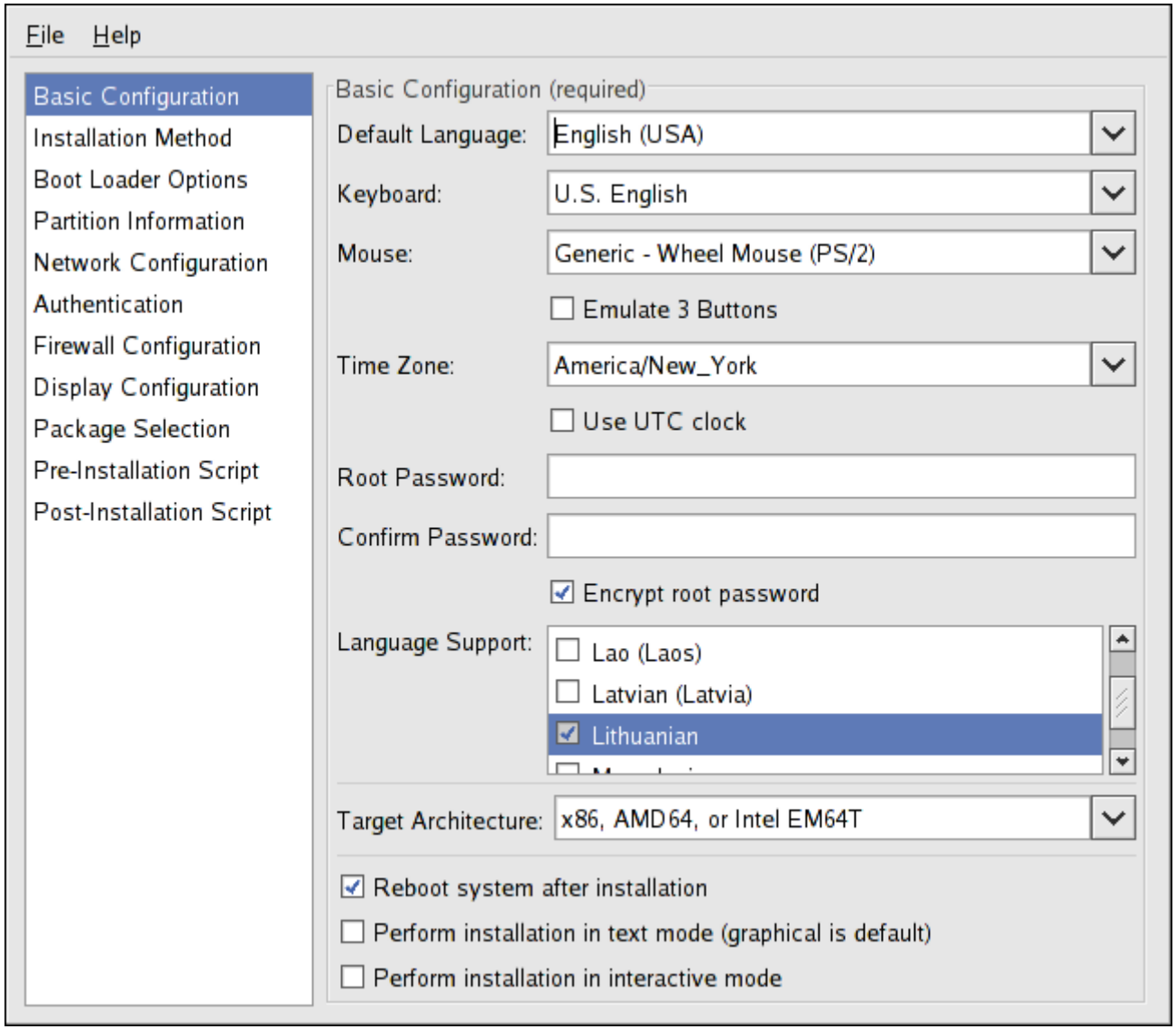
mrepo builds a local APT/Yum RPM repository from local ISO files, downloaded updates, and extra packages from 3rd party repositories. It takes care of setting up the ISO files, downloading the RPMs, configuring HTTP access and providing PXE/TFTP resources for remote network installations.

Features

- Easy Yum-alike configuration
- Supports mirroring using FISH, FTP, HTTP, RSYNC, SFTP and RHN
- Supports Smart, Apt, Yum and up2date (as well as synaptic, yumgui and other derivatives)
- Can download and distribute updates from RHN channels
- Can work directly from ISO images (so you don't need extra diskspace to store ISOs or copy RPMs)
- Supports Red Hat, Fedora Core, Red Hat Enterprise (TaoLinux, CentOS) and Yellow Dog Linux out of the box
- Will probably work with other RPM based distributions (feedback needed, please mail me)
- Allows for remote network installation (using a PXE-enabled NIC on target systems)
- Support for 3rd party repositories and vendor packages
- Allows to maintain your own customized (corporate) repository
- Allow for chaining mrepo servers in large organisations with remote sites
- Can hardlink duplicate packages (to save precious diskspace)

Kickstart Configurator

May be useful in creating a starting point for your kickstart file. The real power comes in your customizations



Yum

From http://linux.duke.edu/projects/yum/[3]

Yum is an automatic updater and package installer/remover for rpm systems. It automatically computes dependencies and figures out what things should occur to install packages. It makes it easier to maintain groups of machines without having to manually update each one using rpm.

NOTE: As of RHEL 5, Red Hat has moved from up2date to yum as the primary repository system. up2date tools now are a front end to yum

PXEBoot

From Wikipedia:[4]

The Preboot Execution Environment (PXE, aka Pre-Execution Environment, or 'pixie') is an environment to boot computers using a network interface card independently of available data storage devices (like hard disks) or installed operating systems.

PXEBoot DHCP Setup

Uses pxelinux.0 from:

http://syslinux.zytor.com/pxe.php[5]

DHCP Configuration:

```
allow booting;
        allow bootp;
   # Standard configuration directives...
        option domain-name "<domain name>";
        option subnet-mask <subnet mask>;
        option broadcast-address <broadcast address>;
        option domain-name-servers <dns servers>;
        option routers <default router>;
   # Group the PXE bootable hosts together
   group {
    # PXE-specific configuration directives...
    next-server <TFTP server address>;
    filename "/tftpboot/pxelinux.0";
    # You need an entry like this for every host
    # unless you're using dynamic addresses
           host <hostname> {
             hardware ethernet <ethernet address>;
           fixed-address <hostname>;
    }
   }
```


PXEBoot tftp Setup

- Install tftp-server
- Easiest is to run kickstart-configurator to build tree
- Change tftp-server to it's own daemon instead of an xinetd service if appropriate
- tftpboot directory structure:

```
/tftpboot

pxelinux.0        # This is the initial pxe bootstrap

/pxelinux.cfg     # Contains individual and group pxeboot entries

/linux-install

/RHEL5

/x86
initrd.img
vmlinuz
/x86_64
initrd.img
vmlinuz
```


PXEBoot Entries

Config files are searched for using the system UUID, followed by the MAC address, then the IP in hex, removing one character at a time, and finally loading default. For a system with a UUID of b8945908-d6a6-41a9-611d-74a6ab80b83d, a MAC address of 88:99:AA:BB:CC:DD and an IP of 192.0.2.91 (C000025B), the following search order would be used[5]:

```
/tftboot/pxelinux.cfg/b8945908-d6a6-41a9-611d-74a6ab80b83d
/tftboot/pxelinux.cfg/01-88-99-aa-bb-cc-dd
/tftboot/pxelinux.cfg/C000025B
/tftboot/pxelinux.cfg/C000025
/tftboot/pxelinux.cfg/C00002
/tftboot/pxelinux.cfg/C0000
/tftboot/pxelinux.cfg/C000
/tftboot/pxelinux.cfg/C00
/tftboot/pxelinux.cfg/C0
/tftboot/pxelinux.cfg/C
/tftboot/pxelinux.cfg/default
```


PXEBoot Config File

The PXEBoot config file contains the same options you would pass at the boot: prompt if you were going to build a system manually:

```
default mypxebuild

label mypxebuild

kernel RHEL5/x86/vmlinuz

append initrd=RHEL5/x86/initrd.img nofb ramdisk_size=10000
ks="http://your.server.com/path/to/your/ks.cfg" ksdevice=eth0 eth0_ethtool="autoneg=on speed=1000
duplex=full"
```


PXEBoot Menuing

You can create simple to powerful, nested menuing sytems using PXEBoot and syslinux. In fact, Cobbler will help you do this to avoid manually creating these menu's. For custom menus, refer to the syslinux page:

http://syslinux.zytor.com/faq.php[6]

Other provisioning technologies

- System Imager
  - Cloned installations
- Satellite
  - Provisioning module does bare metal builds
- Levanta
  - Intrepid product line
- Bladelogic
  - Multiple modules and platforms

System Imager

From http://systemimager.org/[7]

SystemImager is software which automates Linux installs, software distribution, and production deployment. SystemImager is a part of System Installation Suite.

SystemImager makes it easy to do automated installs (clones), software distribution, content or data distribution, configuration changes, and operating system updates to your network of Linux machines. You can even update from one Linux release version to another!

It can also be used to ensure safe production deployments. By saving your current production image before updating to your new production image, you have a highly reliable contingency mechanism. If the new production enviroment is found to be flawed, simply roll-back to the last production image with a simple update command!

Some typical environments include: Internet server farms, database server farms, high performance clusters, computer labs, and corporate desktop environments.

Red Hat Satellite with Provisioning

From https://www.redhat.com/rhn/rhndetails/provisioning/[8]

The Provisioning Module allows you to:

- Manage the complete life cycle of your Linux infrastructure.
- Deploy, configure, manage, update, and then re-deploy your Linux systems, all from a single GUI console complete with all the necessary enterprise functionality and controls.

Functionality

- Bare metal provisioning
- Existing state provisioning
- Multi-state rollback (includes snapshot based recovery)
- Configuration management
- RPM based application provisioning
- Kickstart configuration writer

Levanta Intrepid M

From http://www.levanta.com/[9]

Levanta's Turn-Key Linux Management Appliance

The Intrepid M combines data center-proven Linux management technology with shared storage and Open Source software to create an all-in-one solution to your Linux systems management needs. The Intrepid M is preconfigured so you can be up and running in less than one hour.

Features

- Rapidly provision Linux OS and complete application stacks on a variety of hardware, including servers, diskless blades and virtual machines
- Track all changes made to your Linux systems without changing your current processes
- Recover broken systems, even unbootable ones, quickly and safely using our file-level restore
- Replace failed hardware by moving entire systems, in minutes
- Seamlessly manage servers, diskless blades and virtual machines using a single tool

rom http://www.bmc.com/products/offering/bmc-bladelogic-operations-manager.html

BladeLogic

ladeLogic consists of three modules that address the three most pressing needs in the data center: Configuration, Compliance, and Virtualization.

onfiguration Module

he BMC BladeLogic Configuration Module for Servers (part of the BladeLogic Operations Manager) uses a policy-based approach to provision, patch, configure, and update servers across platforms. Changes are applied to a policy and then synchronized with the target servers. This bi-directional approach lowers the cost and errors associated with managing server infrastructure. It also features a cross-platform, command line interface (Network Shell™) that supports single-sign on using a range of authentication protocols. All user communication is encrypted, and all user actions are logged and can be authorized based on a user's role.

ompliance Module

T compliance measurement is often a costly, manual process that requires domain experts. With the BMC BladeLogic Compliance Module for Servers, you can measure compliance with internal and external standards and regulations. You can create policies based on internal best practices, vendor recommendations (like PCI DSSS), or frameworks like COBIT. The policies enforce compliance through prevention and remediation. With the reporting capabilities, you'll be able to demonstrate compliance with standards and the presence of appropriate controls.

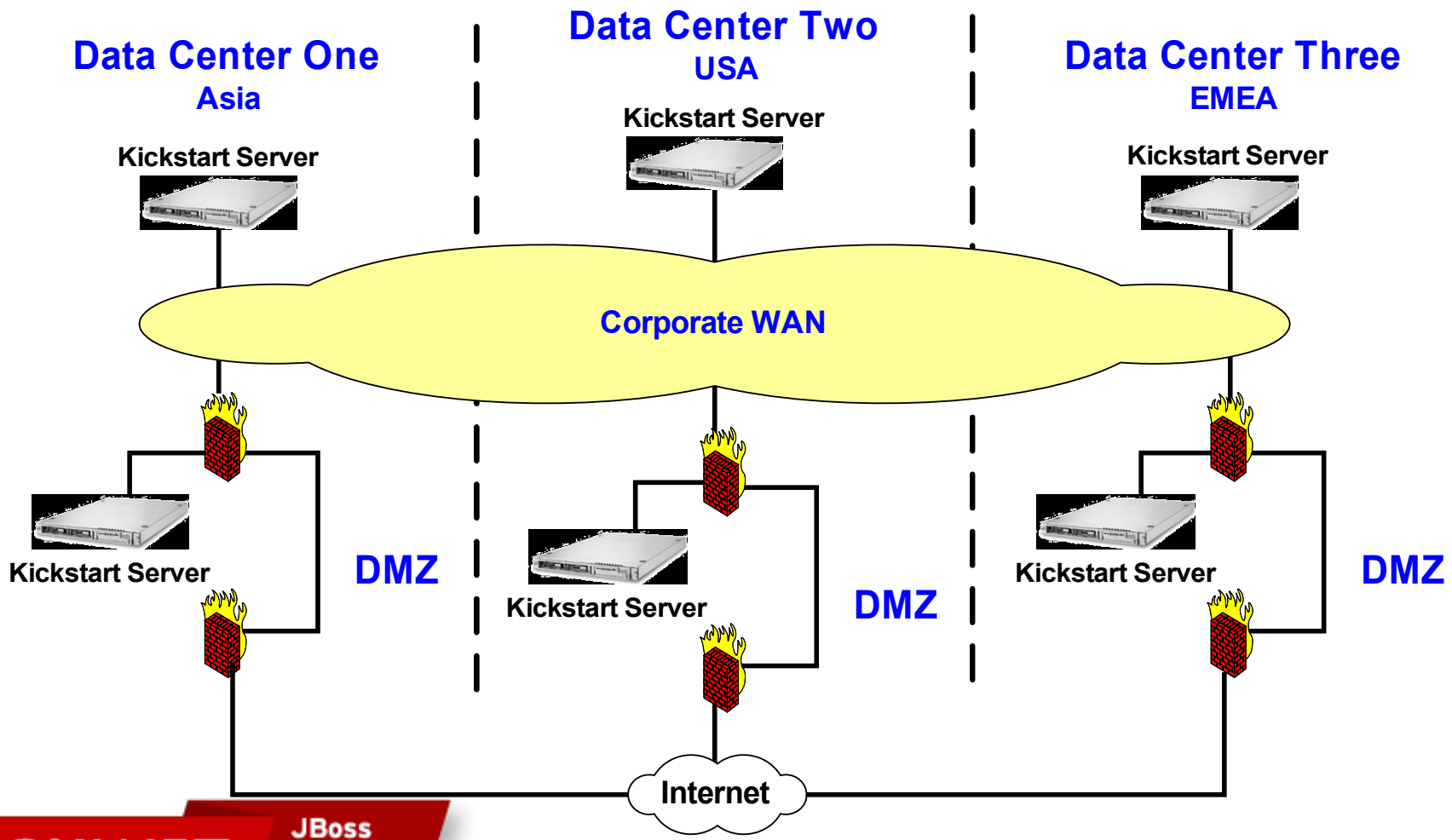
irtualization Module

MC BladeLogic Virtualization Module for Servers enables IT organizations to manage both physical and virtual environments from a unified management platform resulting in greater infrastructure consistency and reduced downtime. Through BladeLogic Virtualization Manager's integrated approach, it is transparent to an end user whether a given server or application service being managed is running on a physical machine or on a virtual instance residing in a virtual container.

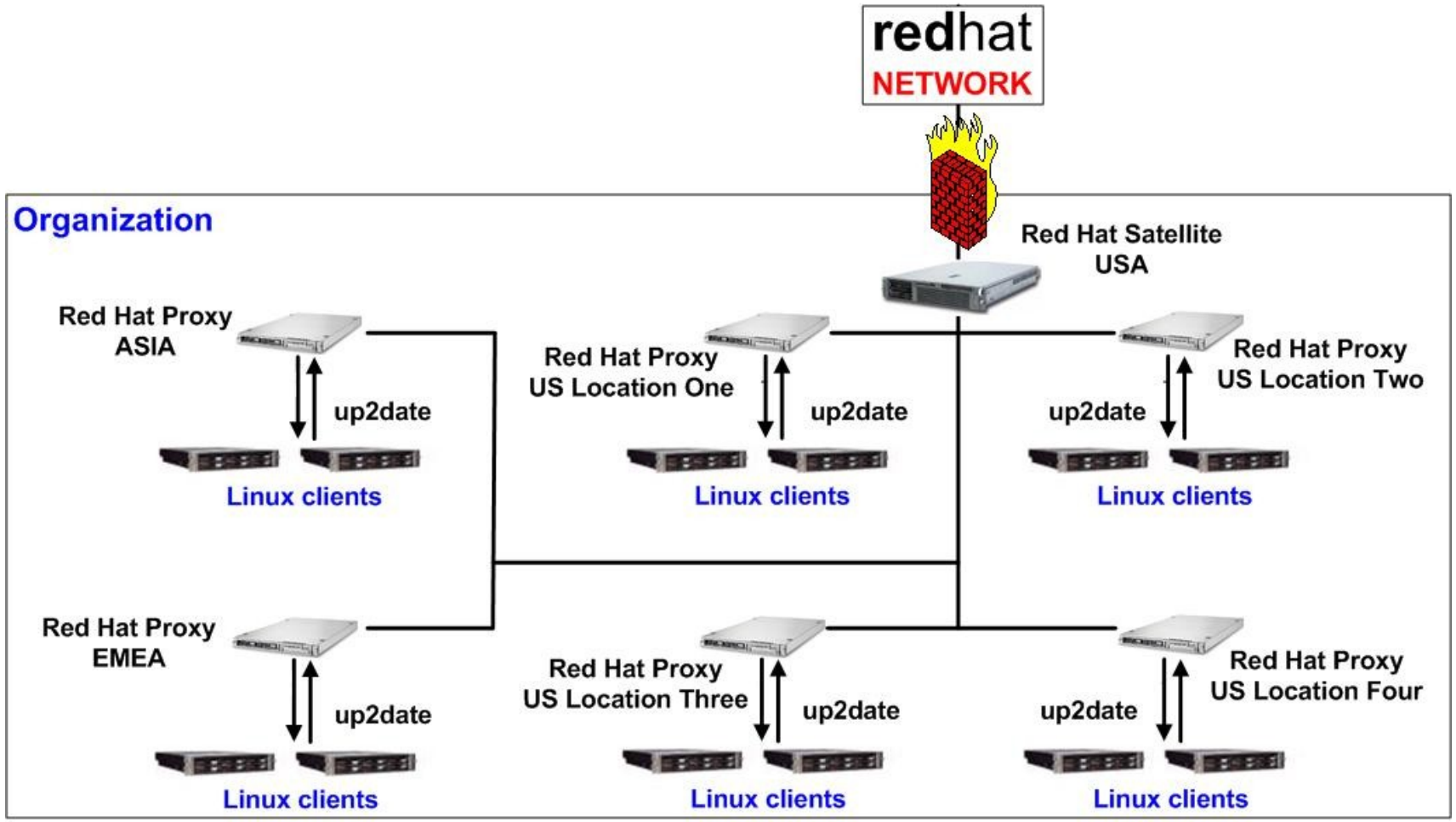
Best Practices

- Core Image should be minimal
- Secure from build via RPM
- Geographically distributed kickstart servers
- Web based kickstart interface for consistency
- Don't reinvent the wheel, leverage existing projects
- Log installation
- Copy across full ks.cfg file (remove password using perl/sed)

Global Build Example



Global Maintenance Example



Resources

- Kickstart mailing list
  - https://www.redhat.com/mailman/listinfo/kickstart-list
- Official Red Hat Documentation
  - https://www.redhat.com/docs/manuals/enterprise/
- RPM Packaging
  - http://fedoraproject.org/wiki/Packaging/Guidelines
- Puppet
  - http://reductivelabs.com/projects/puppet
- CFT
  - http://cft.et.redhat.com/
- Managing RPM-Based Systems with Kickstart and Yum
  - http://www.oreilly.com/catalog/9780596513825/index.html
- Extending Kickstart
  - http://fedoraproject.org/wiki/AnacondaExtendingKickstart

Resources (Continued)

- Fedora “Other Technical Documentation”
  - http://docs.fedoraproject.org/install-guide/f7/en_US/ap-techref.html
- Kickstart Configurator
  - https://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/ch-redhat-config-kickstart.ht
- Anaconda Network Issues
  - http://fedoraproject.org/wiki/AnacondaNetworkIssues
- Owl River tips page
  - http://www.owlriver.com/tips/
- Yum
  - http://docs.fedoraproject.org/yum/en/
- CFEngine
  - http://www.cfengine.org/
- Red Hat Training
  - https://www.redhat.com/training/

Disclaimer

The views expressed herein are the views of the author and are based upon direct and indirect experiences with Kickstart and other provisioning technologies. They do not necessarily represent the views of the Bank of America Corporation.

Chip Shabazian

chip.shabazian@bankofamerica.com

chip@shabazian.com

Thanks

- Red Hat, Inc
  - Red Hat Training
  - All the developers on Anaconda / Kickstart
- CentOS
  - Russ Herrold (Owl River)
- Bank of America
  - For allowing me to give this presentation and providing me the opportunity to learn kickstart so well
  - Brad Crochet for additional kickstart tips
- Everyone on the kickstart mailing list

References

1 Welcome to Revisor - Revisor. Retrieved June 27, 2007, from Welcome to Revisor - Revisor Web site: http://revisor.fedoraunity.org/

2 Wieers, Dag DAG: mrepo: Yum/Apt repository mirroring (fka yam). Retrieved June 27, 2007, from DAG: mrepo: Yum/Apt repository mirroring (fka yam) Web site: http://dag.wieers.com/home-made/mrepo/

3 Linux@DUKE: Yum: Yellow dog Updater, Modified. Retrieved June 27, 2007, from Linux@DUKE: Yum: Yellow dog Updater, Modified Web site: http://linux.duke.edu/projects/yum/

4 Preboot Execution Environment - Wikipedia, the free encyclopedia. Retrieved June 27, 2007, from Preboot Execution Environment - Wikipedia, the free encyclopedia Web site: http://en.wikipedia.org/wiki/Preboot_Execution_Environment

5 Anvin, H. Peter PXELINUX - SYSLINUX for network boot. Retrieved June 27, 2007, from SYSLINUX - The Easy-to-use Linux Bootloader Web site: http://syslinux.zytor.com/pxe.php

6 Anvin, H. Peter SYSLINUX - common documentation. Retrieved June 27, 2007, from SYSLINUX - The Easy-to-use Linux Bootloader Web site: http://syslinux.zytor.com/faq.php

7 Main Page - SystemImager. Retrieved June 27, 2007, from Main Page - SystemImager Web site: http://wiki.systemimager.org/index.php/Main_Page

8 redhat.com | Provisioning. Retrieved June 27, 2007, from redhat.com | Home Web site: https://www.redhat.com/rhn/rhndetails/provisioning/

9 Levanta Products - Levanta Intrepid M. Retrieved June 27, 2007, from Levanta - Linux Management On Demand Web site: http://www.levanta.com/products/intrepid-m.shtml

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#summitjbw

READ THE BLOG

http://summitblog.redhat.com/