



CentOS

CentOS Install & maintain methods review

a look at anaconda/kickstart/tools

Fabian Arrotin
arrfab@centos.org



CentOS

/whois arrfab

- Belgian guy
- sysadmin by choice
- CentOS (ab)user for a long time
- CentOS Project member



Agenda

- Install/deployment (anaconda)
- Software installation (repo management)
- Maintaining (manual/distributed/cfg mgmt)
- Monitoring !



Install/deployment

- Anaconda ! *(because written in "python")*
- <http://fedoraproject.org/wiki/Anaconda>
- Boot kernel+initrd, loads anaconda
- Parses /proc/cmdline for options



Install/deployment

- Manual through local media
- Semi-automated through network/other means
- Automated (kickstart installations)
- Fully-automated (ks + machine setup)
- Cloud ! (ready to be consumed images)



Install/deployment - manual

- Once upon a time
- Floppy/cd/dvd/usb key
- All answers have to be provided through the installer
 - Easy to use
 - Not always reproducible



Install/deployment – semi-manual

- Wait, we have network !
- Boot from usb/boot.iso (smaller media)
- All answers have to be provided through the installer
 - Easy to use
 - Not always reproducible
 - Faster, no need to burn all media



Install/deployment – semi-manual

- Vnc setup !



Install/deployment – semi-manual

- From a running system, download vmlinuz/initrd.img (pxe images), load kernel/initrd and boot directly into installer mode and reconnect with vnc (or connect to vnc in –listen mode) !

```
yum install -y wget kexec-tools && cd /boot  
wget http://mirror.centos.org/centos/6/os/x86_64/images/pxeboot/{vmlinuz,initrd.img}
```

- Load the kernel/initrd.img (no need to hardware init machine)

```
kexec -l vmlinuz --append='ksdevice=eth0  
method=http://mirror.centos.org/centos/6/os/x86_64/ lang=en_GB keymap=be-latin1  
vnc vncpassword=d1X{5pgG ip=your.remote.ip netmask=your.remote.netmask  
gateway=your.remote.gw dns=8.8.8.8 hostname=hostname.fqdn' --initrd=initrd.img  
&& kexec -e
```



Install/deployment – semi-manual

- Note for CentOS 7
 - Not always easy to guess the remote ethernet device name (biosdevname)
 - Default to legacy/previous : *net.ifnames=0 biosdevname=0*

```
kexec -l vmlinuz --append='net.ifnames=0 biosdevname=0 ksdevice=eth0
inst.repo=http://mirror.centos.org/centos/7/os/x86_64/ inst.lang=en_GB
inst.keymap=be-latin1 inst.vnc inst.vncpassword=d1X{5pgG ip=your.remote.ip
netmask=your.remote.netmask gateway=your.remote.gw dns=8.8.8.8
hostname=hostname.fqdn' --initrd=initrd.img && kexec -e
```



Install/deployment – automated

- Wait, we have network !
- Boot from the network (pxe/dhcp)
- All answers are provided to the installer through a response file (kickstart : linux ks=)
 - Easy to use
 - always reproducible
 - Faster, no need to burn any media



Install/deployment – automated

- How to create kickstart file(s) ?
 - Vim ? (reading installation and deployment guide)
 - Install manually, use /root/anaconda-ks.cfg as a base
 - Minimal : @core packages group
 - Use %post to do more than just installation (bonus point)
 - Include the updates (or yours) repo ! (avoid a yum update && reboot after deployment)



Install/deployment – automated

- Where to host kickstart file(s) ?
 - On the network (http/ftp/nfs)
 - On cd !
 - On disk
 - In the installation initrd.img !



Install/deployment – automated

Hacking initrd.img for fun and profit

- With virt-install :
 - --initrd-inject=/path/to/your/ks.cfg
 - --extra-args "console=ttyS0 ks=file:/ks.cfg"



Install/deployment – automated

Hacking initrd.img for fun and profit

- Natively (example for CentOS 7) :

```
cd /boot/  
wget http://mirror.centos.org/centos/7/os/x86_64/images/pxeboot/{vmlinuz,initrd.img}  
#assuming we have copied the ks as ks.cfg in /boot already  
echo ks.cfg | cpio -c -o >> initrd.img  
kexec -l vmlinuz --append='net.ifnames=0 biosdevname=0 ksdevice=eth0  
inst.ks=file:/ks.cfg inst.lang=en_GB inst.keymap=be-latin1 ip=your.ip  
netmask=your.netmask gateway=your.gw dns=your.dns' --initrd=initrd.img && kexec -e
```



Install/deployment – fully automated

Combining kickstart files with existing tools

- Foreman
- Cobbler
- Custom scripts around pxe/dhcp + ks templates
- VM provisioner (whatever the virt platform)



Modifying anaconda “on the fly”

- No need to rebuild the installer
- Supply a updates.img file
 - Ext2 filesystem containing “to be patched” .py files
- Anaconda will use it if :
 - updates.img is in the images directory tree (automatic)
 - You give to anaconda path to updates.img
 - “updates=\$path/updates.img” or “inst.updates=\$path/updates.img”



Install/deployment – cloud !

Cloud images ready to be consumed

- Basic images
- Don't trust random images : build your own !
- Most of the ones you'll find have selinux off, etc
- Customization : cloud-init (or alternatives)



Install/deployment – containers !

- Container != virtual machine (repeat after me)
- Build your own :
 - Using CentOS base image and Dockerfile
 - Examples at <https://github.com/CentOS/CentOS-Dockerfiles>
 - From scratch (using kickstart – once again -) :
 - <https://github.com/CentOS/sig-cloud-instance-build/tree/master/docker>



Software installation

All through RPMs (cpio archive + header) !

- Don't build from source !
- Stick to one packages manager (rpm in our case)
- Don't mix with other tools (cpan/rubygem/npm/etc)
- Repositories :

<http://wiki.centos.org/AdditionalResources/Repositories>



Software installation

- Have a local mirror – in sync (rsync, reposync)
- Create additional repositories (cherry-pick what you want + createrepo)
- Use existing tools for that (pulp/katello/etc ...)
- (re)Build and sign your packages (not that hard)
- *<http://wiki.centos.org/HowTos/RebuildSRPM>*



Maintaining - *The “manual” way*

- ssh is your friend ... (with pub/private keys auth/tunnels/ProxyCommand)
- but doesn't scale at all (*in manual mode at least*)
- Parallel shells help (a little bit but ...) :
 - pdsh
 - clusterssh (cssh)
 - mussh
 - shmux , terminator, ansible -m shell, bash “for loops”, etc



Maintaining - *The “automated” way*

- ... enters “Configuration Management”
- Idea : describe the state the machine should be in
(*infrastructure as code/data*)
- Multiple solutions (pick the one that suits your needs) :
 - puppet
 - chef
 - ansible
 - Cfengine, bcfg2, SaltStack, etc ...



Maintaining - *The “automated” way*

Configuration management is a mindset, not only a tool !

- Whatever the tool, share, collaborate, use a vcs (git,svn,etc)
- Separate code from data (think hiera for puppet)
- Test your recipes/playbooks/modules (VM FTW)



Monitoring

- Often forgotten !
- Test your monitoring scenarios (including false positives !)
- Agentless vs standard protocols vs agent
- Nagios/icinga/zabbix/zenoss/etc ...
- Snmp/ipmi
- Log 2.0 (centralize your logs) : rsyslog/graylog2/elk (elasticsearch/logstash/kibana)/riemann, etc ..



Are you lost ?

We're happy to help you !



CentOS

Are you lost ?

We're happy to help you !

- Mailing lists : <http://lists.centos.org>
- Irc : #centos on irc.freenode.net
- Forum : <http://www.centos.org/forums>
- Twitter : @centos
- Wiki : <http://wiki.centos.org>



Q&A

Questions ?
Thank you !



CentOS