

Rocks Development

A Day in the Trenches with Greg Bruno





Post Install Script Debugging

◆ Example extend-compute.xml:

```
<?xml version="1.0" standalone="no"?>

<kickstart>

<post>
ech "yo" 2&>1 > /tmp/fun
</post>

</kickstart>
```



Post Install Script Debugging

◆ Produces an XML syntax error

```
# rocks list host profile compute-0-1 > /tmp/ks.cfg
Traceback (most recent call last):
  File "/opt/rocks/bin/rocks", line 264, in ?
    command.runWrapper(name, args[i:])
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py", line 1868, in runWrapper
    self.run(self._params, self._args)
.
.
.
.

    parser.feed(line)
  File "/opt/rocks/lib/python2.4/site-packages/_xmlplus/sax/expatreader.py", line 220, in feed
    self._err_handler.fatalError(exc)
  File "/opt/rocks/lib/python2.4/site-packages/_xmlplus/sax/handler.py", line 38, in fatalError
    raise exception
xml.sax._exceptions.SAXParseException: <unknown>:78:11: not well-formed (invalid token)
```



Post Install Script Debugging

◆ One way to debug the error

```
# ROCKSDEBUG=y rocks list host profile compute-0-1 > /tmp/ks.cfg  
  
.  
.  
.  
  
</kickstart>[parse1]  
[parse1]<kickstart>  
[parse1]  
[parse1]<post>  
[parse1]ech "yo" 2&>1 /tmp/fun  
Traceback (most recent call last):  
  File "/opt/rocks/bin/rocks", line 264, in ?  
    command.runWrapper(name, args[i:])  
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py",  
    line 1868, in runWrapper
```



Post Install Script Debugging

◆ Can also use xmllint

```
<post>  
ech "yo" 2&>1 > /tmp/fun  
</post>
```

```
# xmllint extend-compute.xml  
extend-compute.xml:6: parser error :xmlParseEntityRef: no name  
ech "yo" 2&>1 > /tmp/fun  
                ^
```



Post Install Script Debugging

- ◆ Fix your syntax error in your XML file

```
<post>  
ech "yo" > /tmp/fun  
</post>
```

- ◆ But there still is a problem with the post script



Post Install Script Debugging

- ◆ After a node installs, there are several log files saved on the host
 - ➔ A key log file is: `/var/log/rocks-install.log`



Post Install Script Debugging

- ◆ Execute:

```
# rocks list host profile compute-0-0 > /tmp/ks.cfg
```

- ◆ Then examine /tmp/ks.cfg:

```
%post --log=/mnt/sysimage/var/log/rocks-install.log
```

```
ech "yo" > /tmp/fun
```

- ◆ Output from the commands in the post section will be saved to /var/log/rocks-install.log on the node's hard disk



Post Install Script Debugging

- ◆ Huh?
 - ⇒ The filename says `/mnt/sysimage/var/log/rocks-install.log`
- ◆ The installer runs inside a ramdisk
 - ⇒ The 'root' (e.g., '/') for the installer is a ramdisk
 - ⇒ The installer mounts the hard disk partitions under `/mnt/sysimage`



Post Install Script Debugging

- ◆ After the node installs, examine `/var/log/rocks-install.log`

```
./nodes/extend-compute.xml: begin post section  
/tmp/ks-script-MP2uTd: line 2: ech: command not found  
./nodes/extend-compute.xml: end post section
```



Post Install Script Debugging

- ◆ After the node installs, examine `/var/log/rocks-install.log`

```
./nodes/extend-compute.xml: begin post section  
/tmp/ks-script-MP2uTd: line 2: ech: command not found  
./nodes/extend-compute.xml: end post section
```



Post Install Script Debugging

- ◆ Put a “stall” in a post section

```
<post>  
ech "yo" > /tmp/fun  
  
while [ -f /tmp/fun ]  
do  
    sleep 1  
done  
  
</post>
```



Post Install Script Debugging

- ◆ When the node reinstalls, it will stall at the “Running post-install scripts” section
- ◆ Login to the installing node:

```
# ssh compute-0-0 -p 2200
```



Post Install Script Debugging

◆ Get list of processes:

```
# ps aux
root      3783  0.0  0.0      0      0 ?        S      10:30   0:00 [pdflush]
root      4934  0.0  0.0  65888  1304 pts/0    S+    10:30   0:01 /bin/sh /tmp/ks-script-THsZwm
root      13548 0.3  0.0  62616  2980 ?        Ss    12:54   0:00 sshd: root@pts/1
root      13549 0.0  0.0  10764  1328 pts/1    Ss    12:54   0:00 -bash
```



Post Install Script Debugging

- ◆ Let's look at the /tmp/ks-script-* file

```
# cat /mnt/sysimage/tmp/ks-script-THsZwm  
  
ech "yo" > /tmp/fun  
  
while [ -f /tmp/fun ]  
do  
    sleep 1  
done
```

- ◆ This is the contents of our post script



Post Install Script Debugging

- ◆ We can see what anaconda is doing by looking at `/tmp/anaconda.log`

```
# tail -f /tmp/anaconda.log
10:30:16 INFO      : moving (1) to step dopostaction
10:30:16 INFO      : Running kickstart %%post script(s)
10:30:20 ERROR     : Error code 127 encountered running a kickstart


```
pre/%post script
10:30:26 ERROR : Error code 9 encountered running a kickstart
pre/%post script
```


```




Post Install Script Debugging

- ◆ Remove the file `/mnt/sysimage/tmp/fun`

```
# rm -f /mnt/sysimage/tmp/fun
```

- ◆ And the installation will continue



Holding An Installation After All Post Scripts Have Executed

- ◆ Add the “rocks-debug” kernel parameter to the installing kernel
- ◆ List all “bootactions”

```
# rocks list bootaction
```

ACTION	KERNEL	RAMDISK
install:	vmlinuz-5.3-x86_64	initrd.img-5.3-x86_64
ks ramdisk_size=150000 lang= devfs=nomount pxe kssendmac selinux=0		noipv6 ksdevice=bootif
install headless:	vmlinuz-5.3-x86_64	initrd.img-5.3-x86_64
ks ramdisk_size=150000 lang= devfs=nomount pxe kssendmac selinux=0		noipv6 headless vnc



Holding An Installation After All Post Scripts Have Executed

- ◆ Create a new bootaction called “install debug” based on the “install” bootaction

```
# rocks add bootaction action="install debug" \  
kernel="vmlinuz-5.3-x86_64" ramdisk="initrd.img-5.3-x86_64" \  
args="ks ramdisk_size=150000 lang= devfs=nomount pxe \  
kssendmac selinux=0 noipv6 rocks-debug"
```



Holding An Installation After All Post Scripts Have Executed

- ◆ Assign “install debug” installation to a node

```
# rocks list host
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	RUNACTION	INSTALLACTION
build-x86-64:	Frontend	8	0	0	os	install
compute-0-1:	Compute	8	0	1	os	install
compute-0-0:	Compute	8	0	0	os	install

```
# rocks set host installaction compute-0-0 action="install debug"
```

```
# rocks list host
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	RUNACTION	INSTALLACTION
build-x86-64:	Frontend	8	0	0	os	install
compute-0-1:	Compute	8	0	1	os	install
compute-0-0:	Compute	8	0	0	os	install debug



Holding An Installation After All Post Scripts Have Executed

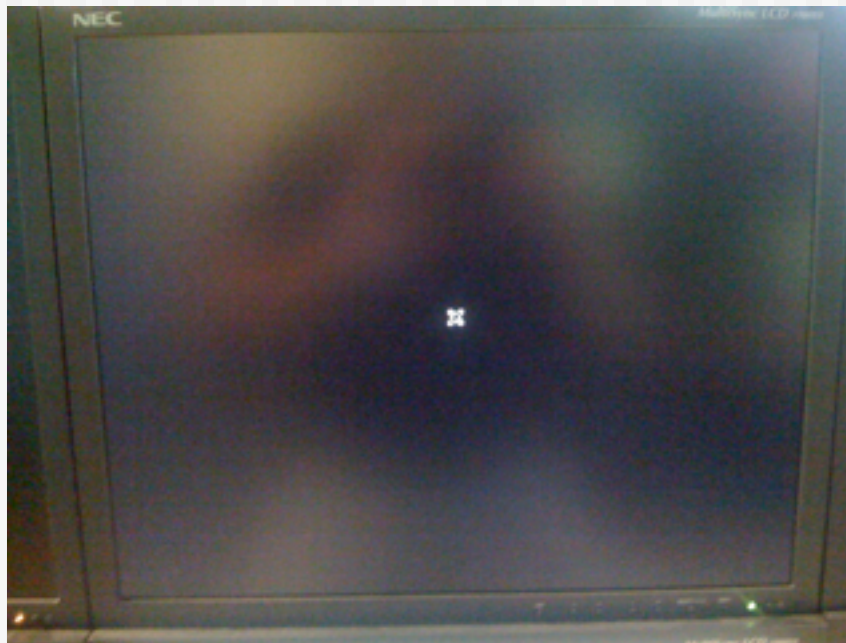
- ◆ The PXE configuration file is updated with the “rocks-debug” flag

```
# cat /tftpboot/pxelinux/pxelinux.cfg/0A01FFFE
default rocks
prompt 0
label rocks
    kernel vmlinuz-5.3-x86_64
    append ks ramdisk_size=150000 lang= devfs=nomount pxe
    kssendmac selinux=0 noipv6 rocks-debug
    initrd=initrd.img-5.3-x86_64
```



Post Install Script Debugging

- ◆ When the node reinstalls, it will stall at the “initial X screen”





Post Install Script Debugging

- ◆ At this point, all post scripts have executed and the installation is stalled
 - ➔ Login to the node and you'll see:

```
# ssh compute-0-0 -p 2200
```

```
-bash-3.2# tail -f /tmp/anaconda.log
```

```
10:54:01 ERROR      : Error code 127 encountered running a kickstart %pre/%post script
10:54:07 ERROR      : Error code 9 encountered running a kickstart %pre/%post script
10:54:08 INFO       : All kickstart %%post script(s) have been run
```



Post Install Script Debugging

- ◆ You now have unlimited time to poke around on the node
- ◆ To complete the installation, you need to reboot the node (e.g., <Ctrl><Alt>)
 - Or, if you logged in via “ssh compute-0-0 -p 2200” execute:

```
# reboot -f
```




Other Log Files on an *Installing* Node

- ◆ Look at other log files on an installing node

```
# ls /tmp/*log
/tmp/anaconda-http.log /tmp/httpd.log /tmp/syslog
/tmp/anaconda.log      /tmp/ks-pre.log
```

- ◆ /tmp/httpd.log
 - ⇒ Log messages from lighttpd
- ◆ /tmp/syslog
 - ⇒ Typical syslog
- ◆ /tmp/anaconda-httpd.log
 - ⇒ RPM download timing information
- ◆ /tmp/ks-pre.log
 - ⇒ Output from “pre” scripts



/tmp/httpd.log

```
127.0.0.1 127.0.0.1 - [28/May/2010:10:45:11 -0700]
"GET /install/rocks-dist/x86_64/RedHat/RPMS/dmraid-
events-1.0.0.rc13-53.el5.x86_64.rpm HTTP/1.1" 200 18083
 "-" "urlgrabber/3.1.0 yum/3.2.22"
```

```
127.0.0.1 127.0.0.1 - [28/May/2010:10:45:13 -0700]
"GET /install/rocks-dist/x86_64/RedHat/RPMS/compat-
libf2c-34-3.4.6-4.i386.rpm HTTP/1.1" 200 2674 "-"
"urlgrabber/3.1.0 yum/3.2.22"
```



/tmp/syslog

```
<4>raid6: int64x4      1929 MB/s
<4>raid6: int64x8      1585 MB/s
<4>raid6: sse2x1       3195 MB/s
<4>raid6: sse2x2       5332 MB/s
<4>raid6: sse2x4       6058 MB/s
<4>raid6: using algorithm sse2x4 (6058 MB/s)
<6>md: raid6 personality registered for level 6
<6>md: raid5 personality registered for level 5
<6>md: raid4 personality registered for level 4
```



/tmp/anaconda-httpd.log

```
1275068709.09: RedHat/RPMS/dmraid-events-1.0.0.rc13-53.e15.x86_64.rpm start
1275068711.1: RedHat/RPMS/dmraid-events-1.0.0.rc13-53.e15.x86_64.rpm end
1275068711.1: RedHat/RPMS/compat-libf2c-34-3.4.6-4.i386.rpm start
1275068713.1: RedHat/RPMS/compat-libf2c-34-3.4.6-4.i386.rpm end
```



/tmp/ks-pre.log

```
/tmp/ks-script-mvCzD5: line 12: /opt/rocks/bin/ci: No such file or directory  
/tmp/ks-script-mvCzD5: line 13: /opt/rocks/bin/co: No such file or directory
```



Log Files on an *Installed* Node

- ◆ After a node is installed, several log files are saved under “/root”

```
# ls /root/*log
/root/httpd.log      /root/install.log.syslog  /root/rocks-pre.log
/root/install.log   /root/rocks-post.log
```

- ◆ /root/httpd.log
 - A copy of the log messages from lighttpd
- ◆ /root/install.log
 - A list of RPMs installed by anaconda
- ◆ /root/install.log.syslog
 - Post script error messages
- ◆ /root/rocks-pre.log, /root/rocks-post.log
 - Output from rocks-pre.sh and rocks.post.sh scripts



/root/install.log

```
Installing hal-0.5.8.1-52.el5.i386
Installing kudzu-1.2.57.1.21-1.el5.centos.x86_64
Installing hal-0.5.8.1-52.el5.x86_64
Installing system-config-network-tui-1.3.99.18-1.el5.noarch
Installing firstboot-tui-1.4.27.7-1.el5.centos.x86_64
```

- ◆ Shows the packages that were installed
 - ⇒ And it shows the “architecture” of the package



/root/install.log.syslog

```
<86>May 28 10:53:53 useradd[3907]: new group: name=nfsnobody, GID=4294967294  
<86>May 28 10:53:53 useradd[3907]: new user: name=nfsnobody, UID=4294967294,  
GID=4294967294, home=/var/lib/nfs, shell=/sbin/nologin  
<86>May 28 10:53:53 useradd[3925]: new group: name=haldaemon, GID=68  
<86>May 28 10:53:53 useradd[3925]: new user: name=haldaemon, UID=68, GID=68,  
home=/, shell=/sbin/nologin
```

◆ I never look at this file!



/root/rocks-pre.log /root/rocks-post.log

- ◆ On the first boot after an installation, the following scripts are run:

```
/root/rocks-pre.sh  
/root/rocks-post.sh
```

- ◆ The scripts ensure the file permissions set with “<file>” tags are maintained

Node XML file:

```
<file name="/etc/rc.d/init.d/mdmonitor" perms="755">
```

/root/rocks-pre.sh:

```
if [ -f /etc/rc.d/init.d/mdmonitor ]; then  
    echo "rocks" | /opt/rocks/bin/ci /etc/rc.d/init.d/mdmonitor;  
    /opt/rocks/bin/co -f -l /etc/rc.d/init.d/mdmonitor;  
fi  
chmod 755 /etc/rc.d/init.d/mdmonitor
```



/root/rocks-pre.log /root/rocks-post.log

- ◆ The log files mainly contain RCS status messages
 - ➔ All files modified with a “<file>” tag are managed with RCS

```
/etc/rc.d/init.d/RCS/mdmonitor,v <-- /etc/rc.d/init.d/mdmonitor
new revision: 1.2; previous revision: 1.1
done
/etc/rc.d/init.d/RCS/mdmonitor,v --> /etc/rc.d/init.d/mdmonitor
revision 1.2 (locked)
done
```

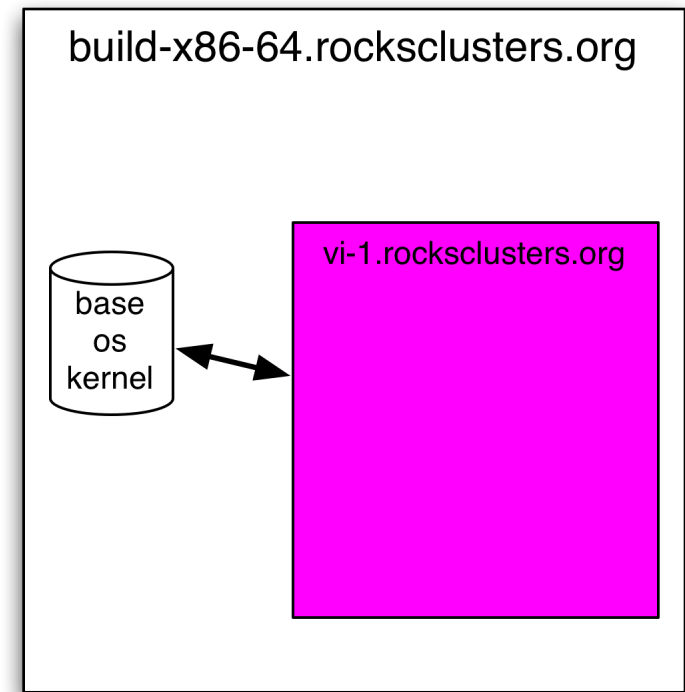


“Lights Out” Frontend Installation



“Lights Out” Frontend Install

- ◆ Fully-automated frontend install
- ◆ Install a virtual frontend (vi-1) inside a physical frontend (build-x86-64)





“Lights Out” Frontend Install

◆ How it works:

⇒ On a normal installation:

- After the user selects the rolls, a file named `/tmp/rolls.xml` is created
- After the user inputs all data in the user “screens” (e.g., “Enter Private Network Configuration”), a file named `/tmp/site.attrs` is created

- ⇒ So, if the files `/tmp/site.attrs` and `/tmp/rolls.xml` already exist on the ramdisk, then the machine will bypass the user screens and automatically install



Lights Out Frontend Installation

◆ Create a virtual frontend

```
# rocks add cluster vi-1.rocksclusters.org 137.110.119.118 0
```

◆ Set attributes

```
# rocks set host attr vi-1.rocksclusters.org \  
  Kickstart_PrivateHostname vi-1
```

```
# rocks set host attr vi-1.rocksclusters.org \  
  Kickstart_PublicAddress 137.110.119.118
```

```
# rocks set host attr vi-1.rocksclusters.org \  
  Kickstart_PublicHostname vi-1.rocksclusters.org
```

```
# rocks set host attr vi-1.rocksclusters.org \  
  Kickstart_PublicKickstartHost build-x86-64.rocksclusters.org
```



Assign Rolls to the New Frontend

```
# rocks set host roll vi-1.rocksclusters.org os 5.3 x86_64
# rocks set host roll vi-1.rocksclusters.org base 5.3 x86_64
# rocks set host roll vi-1.rocksclusters.org kernel 5.3 x86_64
```

```
# rocks list host roll vi-1.rocksclusters.org
HOST          NAME    VERSION ARCH  OS
frontend-0-0-0: os      5.3    x86_64 linux
frontend-0-0-0: base    5.3    x86_64 linux
frontend-0-0-0: kernel 5.3    x86_64 linux
```



/tmp/rolls.xml

```
# rocks report host roll vi-1.rocksclusters.org
<rolls>
<roll
    name="os"
    version="5.3"
    arch="x86_64"
    url="http://build-x86-64.rocksclusters.org/install/rolls/"
    diskid=""
/>
<roll
    name="base"
    version="5.3"
    arch="x86_64"
    url="http://build-x86-64.rocksclusters.org/install/rolls/"
    diskid=""
/>
<roll
    name="kernel"
    version="5.3"
    arch="x86_64"
    url="http://build-x86-64.rocksclusters.org/install/rolls/"
    diskid=""
/>
</rolls>
```




/tmp/site.attrs

```
# rocks report host attr vi-1.rocksclusters.org
.
.
.
Kickstart_PublicAddress:137.110.119.118
Kickstart_PublicBroadcast:137.110.119.255
Kickstart_PublicDNSDomain:rocksclusters.org
Kickstart_PublicDNSServers:8.8.8.8
Kickstart_PublicGateway:137.110.119.1
Kickstart_PublicHostname:vi-1.rocksclusters.org
Kickstart_PublicKickstartHost:build-x86-64.rocksclusters.org
Kickstart_PublicNTPHost:pool.ntp.org
.
.
.
```



Need to Create rolls.xml and site.attrs

- ◆ Create node XML file 'extend-wan.xml'
 - ⇒ In the directory:
 - /export/rocks/install/site-profiles/5.3/nodes



Need to Create rolls.xml and site.attrs Files in the Installer

```
<?xml version="1.0" standalone="no"?>

<kickstart>

<pre>

<file name="/tmp/site.attrs">
<eval>/opt/rocks/bin/rocks report host attr &hostname;</eval>
</file>

<file name="/tmp/rolls.xml">
<eval>/opt/rocks/bin/rocks report host roll &hostname;</eval>
</file>

</pre>

</kickstart>
```



Rebuild the Distro and Boot the VM

```
# cd /export/rocks/install  
# rocks create distro  
  
# rocks start host vm vi-1.rocksclusters.org
```

- ◆ 10 minutes later, you've got a new frontend!



Fin



Partitioning



Partitioning

- ◆ You can see what the “programmatically partitioning” will do
- ◆ Login to compute node and execute:

```
# ssh compute-0-0 p 2200

-bash-3.2# /tmp/product/do_partition.py
part / --fstype ext3 --onpart sda1
part /var --fstype ext3 --onpart sda2
part swap --noformat --onpart sda3
part /state/partition1 --noformat --onpart sda5
```



Partitioning

- ◆ If you have a bug in your programmatic partitioning code, you'll need to recreate the code from the 'pre' script

- ⇒ On the frontend, create /tmp/ks.cfg

```
# rocks list host profile compute-0-0 > /tmp/ks.cfg
```

- ⇒ Find the 'pre' script in /tmp/ks.cfg with your partitioning code

- ◆ Make a new file named /tmp/a.py from the partitioning code



/tmp/a.py

```
#!/opt/rocks/bin/python

import rocks_partition

membership = 'Compute'
nodename = 'compute-0-1'

def doDisk(file, disk):
    file.write('clearpart --all --initlabel --drives=%s\n' % disk)
    file.write('part / --size=6000 --fstype=ext3 --ondisk=%s\n' % disk)
    file.write('part /var --size=2000 --fstype=ext3 --ondisk=%s\n' % disk)
    file.write('part swap --size=2000 --ondisk=%s\n' % disk)
    file.write('part /mydata --size=1 --grow --fstype=ext3 --ondisk=%s\n'
               % disk)

#
# main
#
p = rocks_partition.RocksPartition()
disks = p.getDisks()

if len(disks) == 1:
    file = open('/tmp/user_partition_info', 'w')
    doDisk(file, disks[0])
    file.close()
```



Copy the Script to the Compute Node

- ◆ Put the script in /tmp/product on the compute node

```
-bash-3.2# scp 10.1.1.1:/tmp/a.py /tmp/product/a.py
```

- ◆ Run it:

```
-bash-3.2# ./a.py
```

```
-bash-3.2# cat /tmp/user_partition_info  
clearpart --all --initlabel --drives=sda  
part / --size=6000 --fstype=ext3 --ondisk=sda  
part /var --size=2000 --fstype=ext3 --ondisk=sda  
part swap --size=2000 --ondisk=sda  
part /mydata --size=1 --grow --fstype=ext3 --ondisk=sda
```