



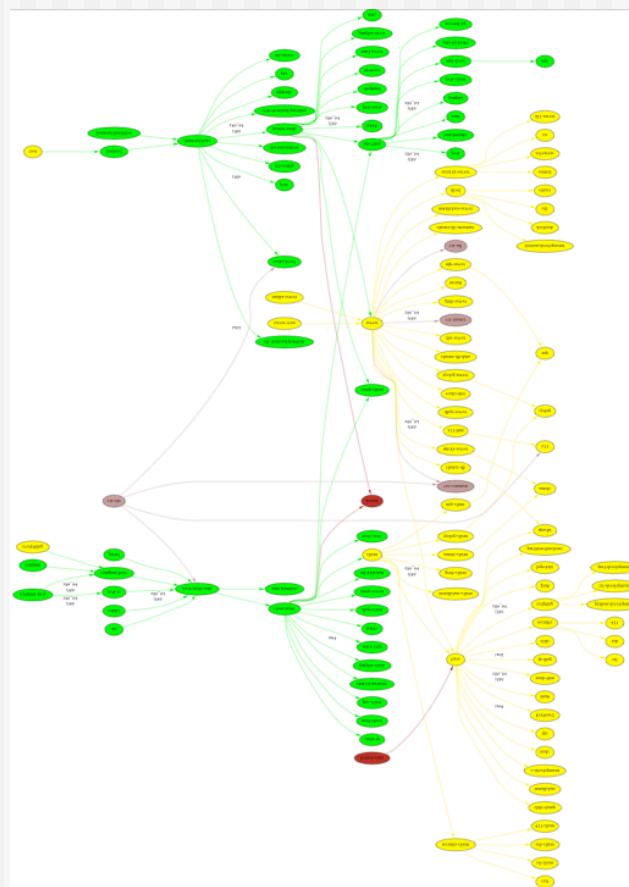
# Inside the Viz Roll

---

Mason Katz  
[mjk@sdsc.edu](mailto:mjk@sdsc.edu)

ROCKS

# { base, hpc, kernel, viz }



# Early Work: NCSA

## ◆ LCD Cluster

- ⇒ Custom framing
- ⇒ One PC / tile
- ⇒ Portable (luggable)
- ⇒ SC 2001 Demo

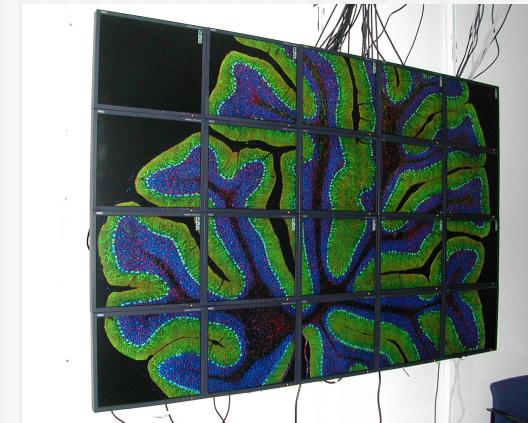
## ◆ NCSA Software

- ⇒ Pixel Blaster
- ⇒ Display Wall In-A-Box
  - OSCAR based
  - Never fully released



# NCMIR

- ◆ Using Rocks
- ◆ Hand configured a visualization cluster
- ◆ “Administered the machine to the point of instability”
  - David Lee
- ◆ Automation is needed





# COTS Vis: GeoWall

- ◆ LCD Clusters
  - ➲ One PC / tile
  - ➲ Gigabit Ethernet
  - ➲ Optional Stereo Glasses
  - ➲ Portable
  - ➲ Commercial Frame (Reason)
- ◆ Applications
  - ➲ Large remote sensing
  - ➲ Volume Rendering
  - ➲ Seismic Interpretation
  - ➲ Brain mapping (NCMIR)
- ◆ Electronic Visualization Lab
  - ➲ Jason Leigh (UIC)





# OptIPortal (SAGE)





# One Node per Display





# OptIPortal





# Nodes Behind the Wall





# Use of OptIPortal to Interactively View Microbial Genome (CGView)

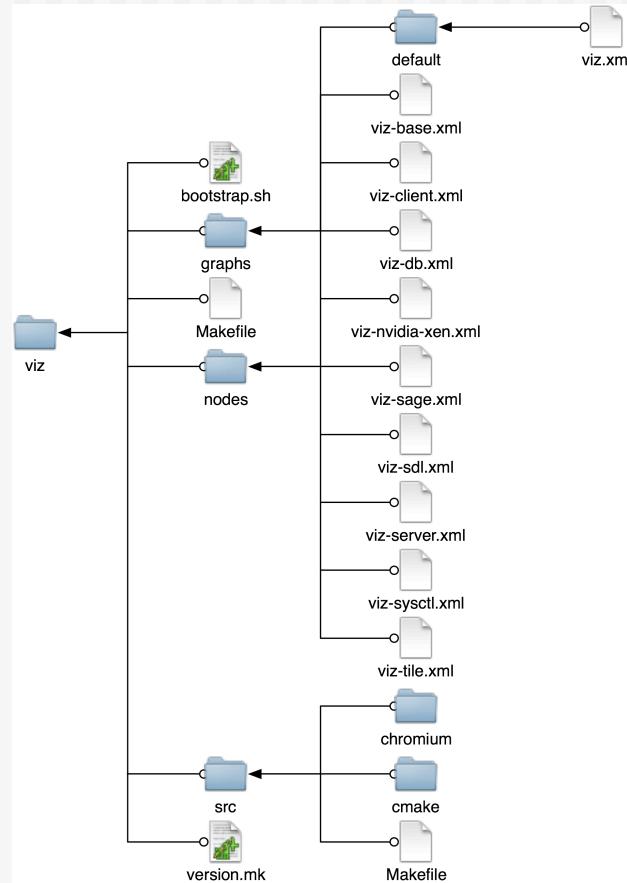
Acidobacteria  
Bacterium Ellin345  
(NCBI)  
Soil Bacterium 5.6 Mb



Source: Raj Singh, UCSD

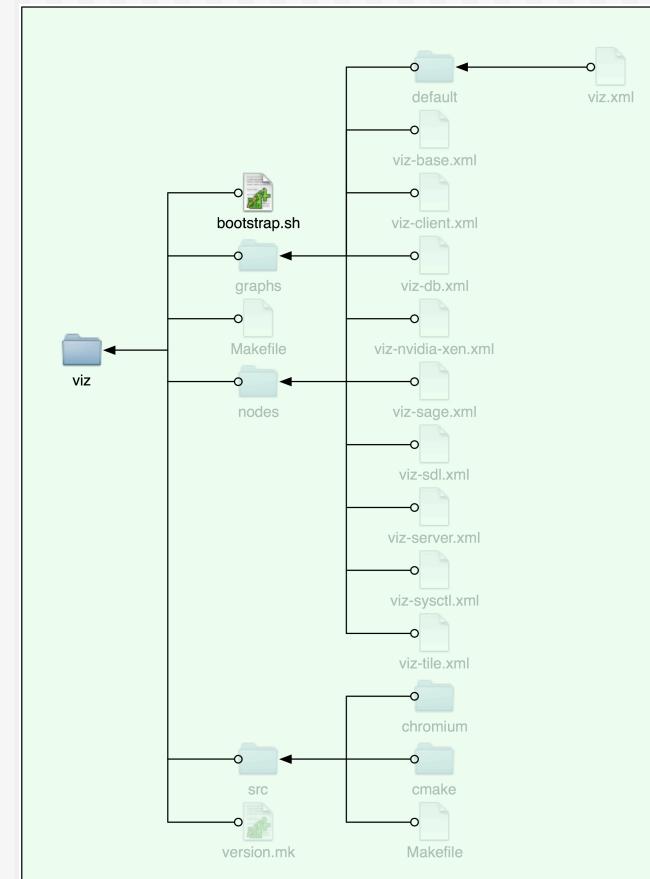
Acidobacteria bacterium Ellin345 (NCBI)  
© 2009 UC Regents  
Soil Bacterium 5.6 Mb

# Viz Roll Source Tree



# Bootstrapping

- ◆ Roll need to be build on a Rocks Frontend
- ◆ The Viz Roll needs to be build on a Rocks Frontend that has the Viz Roll install
- ◆ Is that a problem?



# bootstrap.sh

---

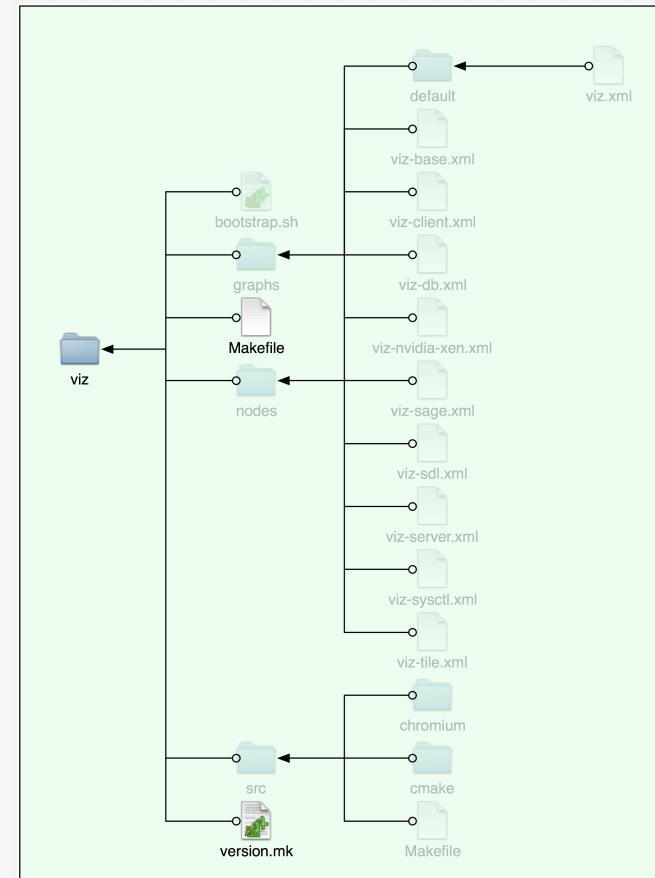
- ◆ Every Roll has one
- ◆ Used install building dependencies
  - ⇒ From OS
  - ⇒ From Roll install
- ◆ Simple shell script
- ◆ Easy to read

# For Example

```
1:#!/bin/sh
2:#
3: # $Id: bootstrap.sh,v 1.18 2009/05/01 19:07:24 mjk Exp $
4:
5: . $ROLLSROOT/etc/bootstrap-functions.sh
6:
7: install_os_packages viz-server
8:
9: install libdmx
10: install libdmx-devel
11:
12: compile SRPMs
13: install glut
14: install glut-devel
15: install joy2key
16: install joystick
17:
18: compute_and_install cmake
19:
20: compile_and_install foundation-sdl
21: compile_and_install foundation-sdl-image
22: compile_and_install foundation-sdl-mixer
23: compile_and_install foundation-sdl-net
24: compile_and_install foundation-sdl-ttf
25:
26: compile_and_install chromium
27: compile_and_install libraw1394
28: compile_and_install openal
29: compile_and_install freealut
30: compile_and_install plib
31: compile_and_install SimGear
32:
33: compile_and_install libmad
34: compile_and_install libdvdread
35: compile_and_install faac
36: compile_and_install faad2
37: compile_and_install ffmpeg
38: compile_and_install mpeg2dec
39: compile_and_install wx
40: compile_and_install portaudio
41: compile_add_install glew
42:
43: compile quanta
44: install QUANTA
45:
```

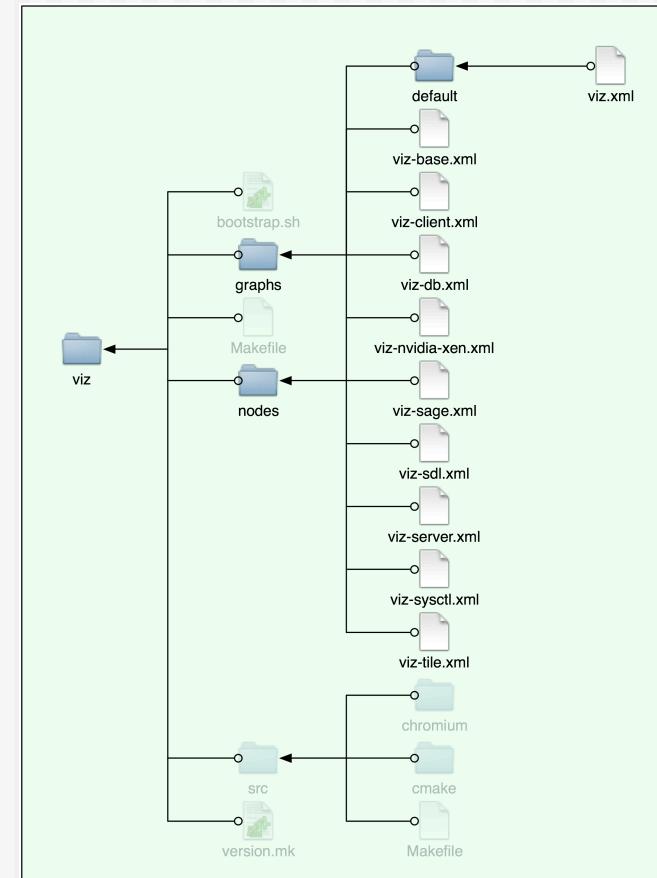
# makefiles

- ◆ Every Rocks Frontend has all our build environment installed
- ◆ Roll Makefile almost empty
  - ⦿ Just copy from another Roll
  - ⦿ No need to change anything
- ◆ Version.mk can
  - ⦿ Set version
  - ⦿ Set color (graph)
  - ⦿ A few other minor things



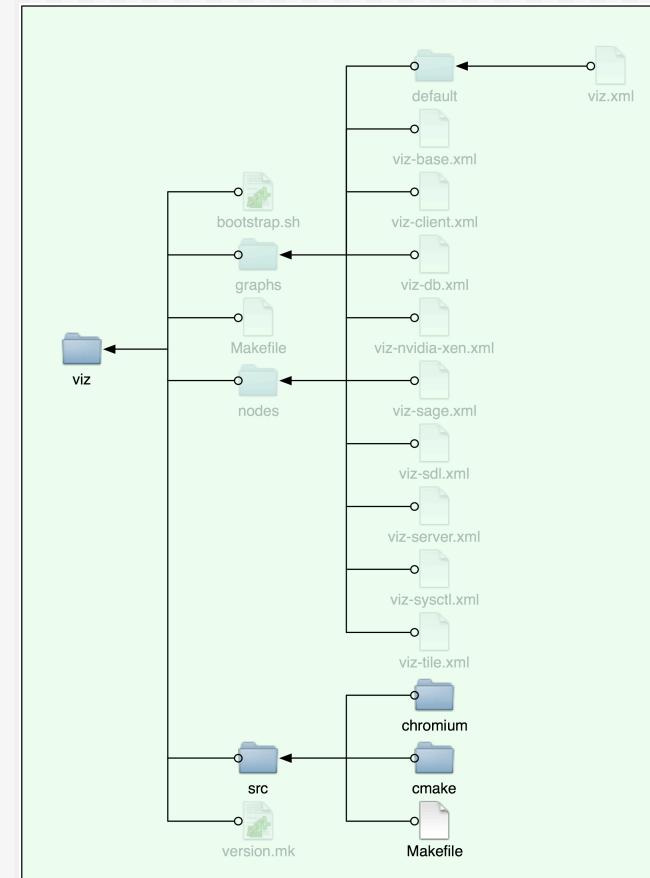
# Viz Roll Sub-Graph

- ◆ Single Graph File
- ◆ 9 Nodes Files



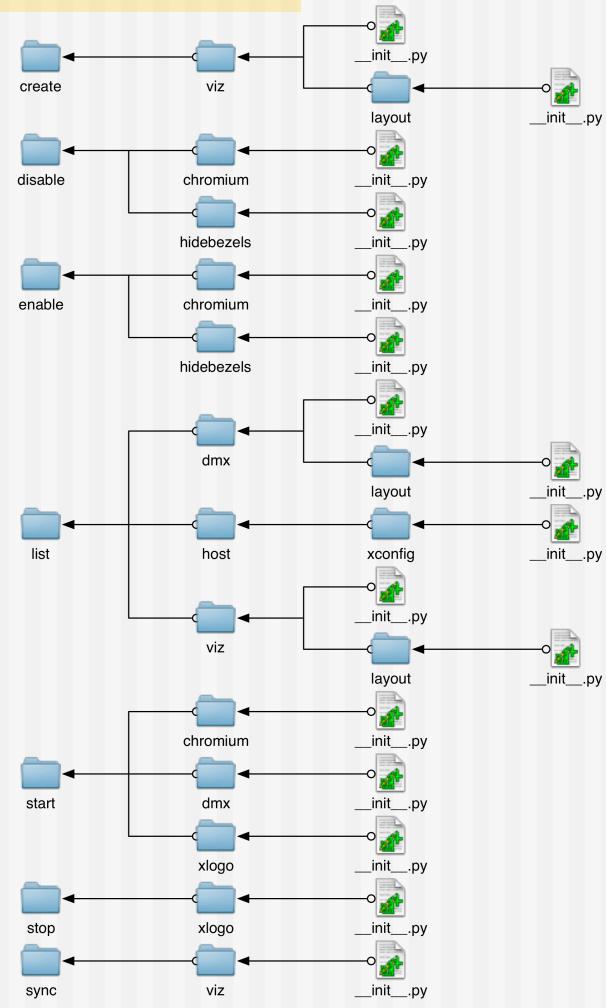
# Source Code

- ◆ Almost 50 packages



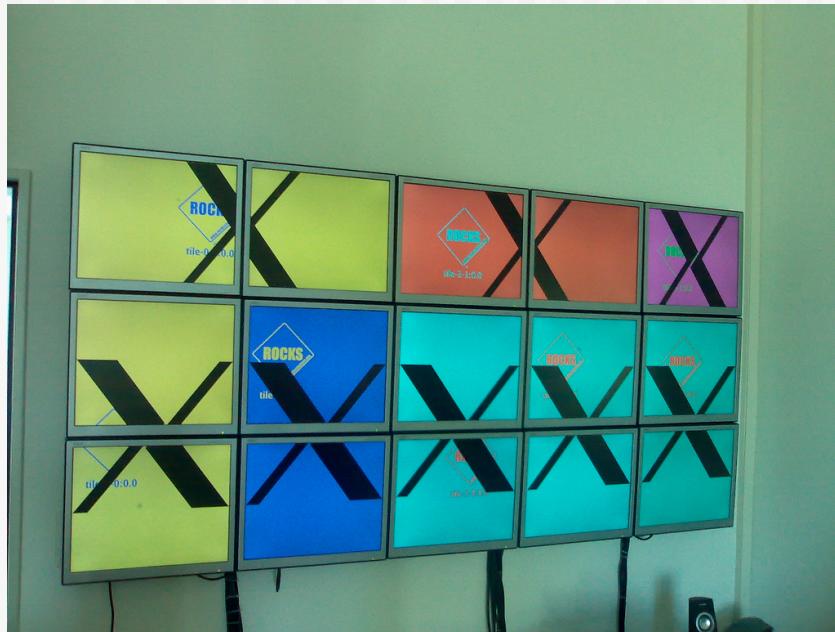
# Dozen+ Command

```
# rocks list roll command viz
COMMAND
create viz layout
disable chromium
disable hidebezels
enable chromium
enable hidebezels
list dmx layout
list host xconfig
list viz layout
start chromium
start dmx
start xlogo
stop xlogo
sync viz
```



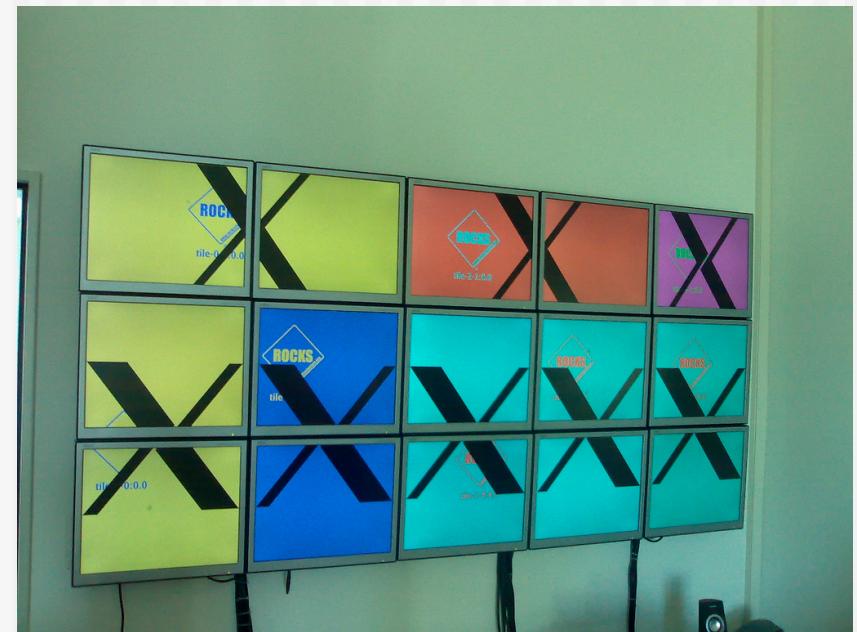


# LCD Bezels



# rocks enable hidebezels

- ◆ Draws pixels behind the bezels (mullions) of the LCD monitors
- ◆ Calculated offset for TwinView and normal modes
- ◆ Reset the X11 configuration on all nodes
- ◆ Great mode for moving graphics



```

1: import rocks.commands.enable
2: import os
3:
4: class Command(rocks.commands.enable.Command):
5:     """
6:     Enable Bezel Hiding mode.
7:
8:     <example cmd="enable hidebezels">
9:     </example>
10:    """
11:
12:    MustBeRoot = 0
13:
14:    def run(self, params, args):
15:
16:        os.system('touch ~/.hidebezels')
17:
18:        # If the database videowall layout has two (or more) entries
19:        # for the same host and card we know we are in twinview
20:        # mode. In this case we need to reconfigure and restart
21:        # X11 for this host.
22:
23:        self.db.execute("""select n.name, v.cardid
24:                        from nodes n, videowall v where v.node=n.id""")
25:        dict = {}
26:        for key in self.db.fetchall():
27:            if dict.has_key(key):
28:                dict[key] = 1 # TwinView host
29:            else:
30:                dict[key] = 0 # NonTwinView host (so far)
31:
32:        for (host, card) in dict.keys():
33:            if dict[(host, card)]:
34:                os.system('ssh -f '
35:                         '%s /usr/X11R6/bin/xrandr -d :0 -s 1'
36:                         % host)

```

# rocks disable hidebezels

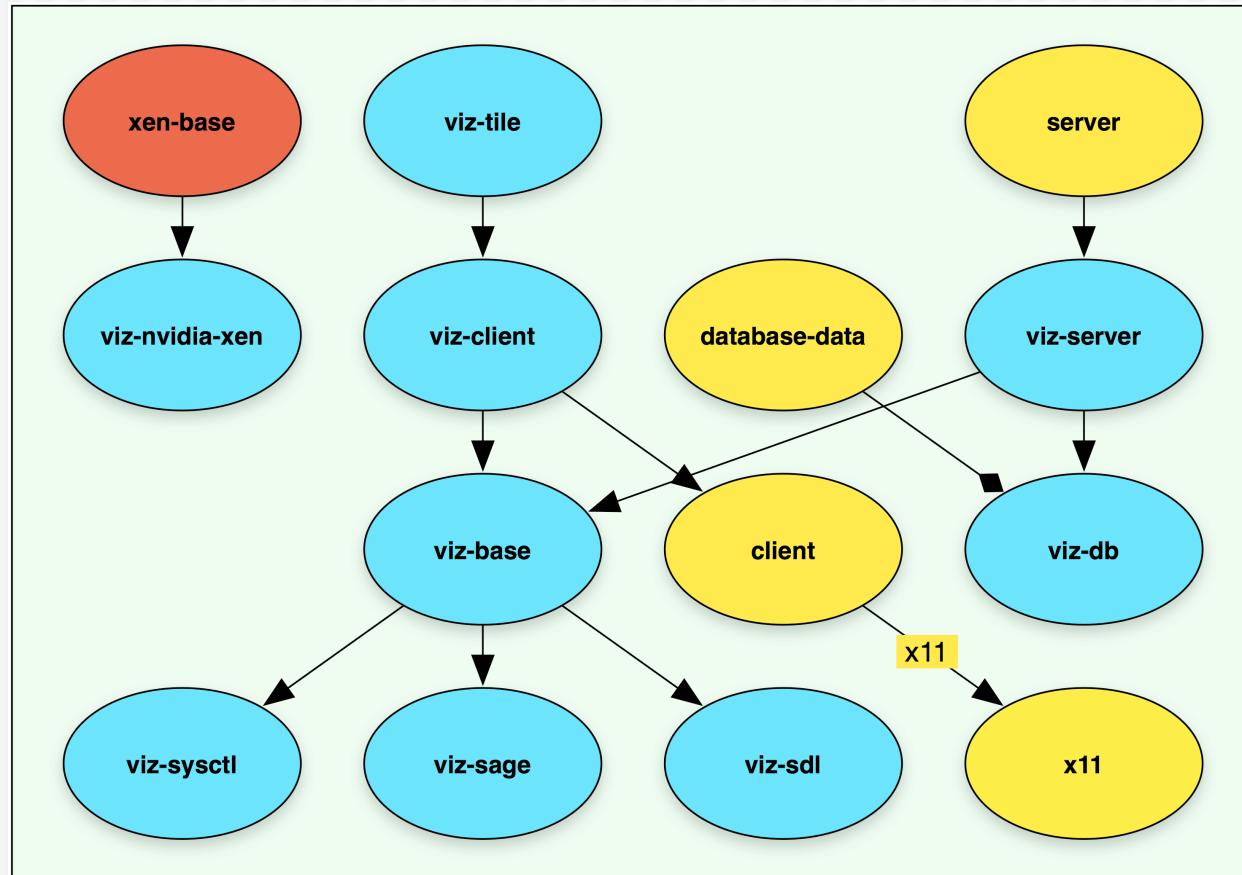
- ◆ All pixels are drawn and the bezels break apart the image
- ◆ Removes any offset from previous mode
- ◆ Resets the X11 configuration on all nodes
- ◆ Great for static images and text



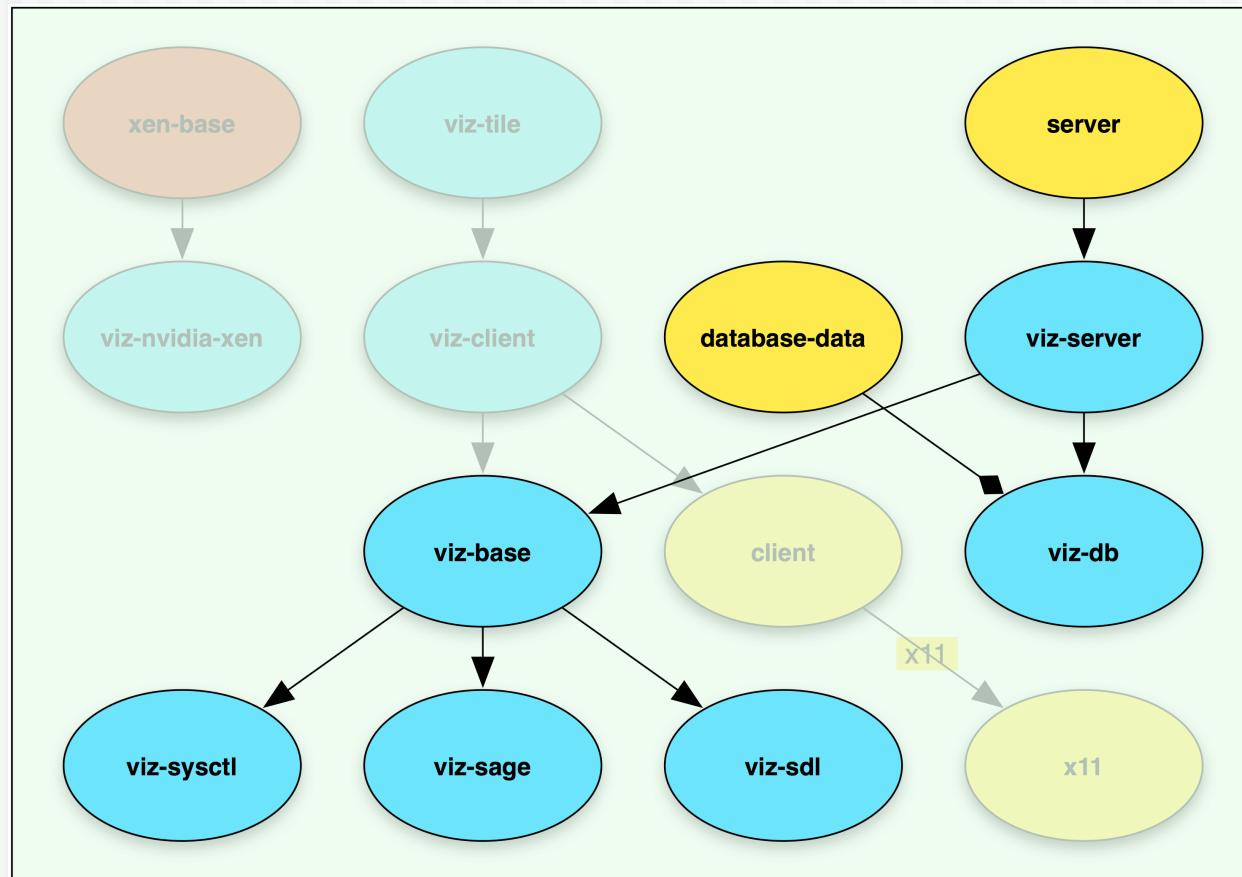
```

1: import rocks.commands.enable
2: import os
3:
4: class Command(rocks.commands.disable.Command):
5:     """
6:     Disable Bezel Hiding mode.
7:
8:     <example cmd="disable hidebezels">
9:     </example>
10:    """
11:
12:    MustBeRoot = 0
13:
14:    def run(self, params, args):
15:
16:        os.system('/bin/rm ~/hidebezels')
17:
18:        # If the database videowall layout has two (or more) entries
19:        # for the same host and card we know we are in twinview
20:        # mode. In this case we need to reconfigure and restart
21:        # X11 for this host.
22:
23:        self.db.execute("""select n.name, v.cardid
24:                        from nodes n, videowall v where v.node=n.id""")
25:        dict = {}
26:        for key in self.db.fetchall():
27:            if dict.has_key(key):
28:                dict[key] = 1 # TwinView host
29:            else:
30:                dict[key] = 0 # NonTwinView host (so far)
31:
32:        for (host, card) in dict.keys():
33:            if dict[(host, card)]:
34:                os.system('ssh -f '
35:                         '%s /usr/X11R6/bin/xrandr -d :0 -s 0'
36:                         % host)
37:
```

# Viz Roll Sub-Graph



# Frontend Profile

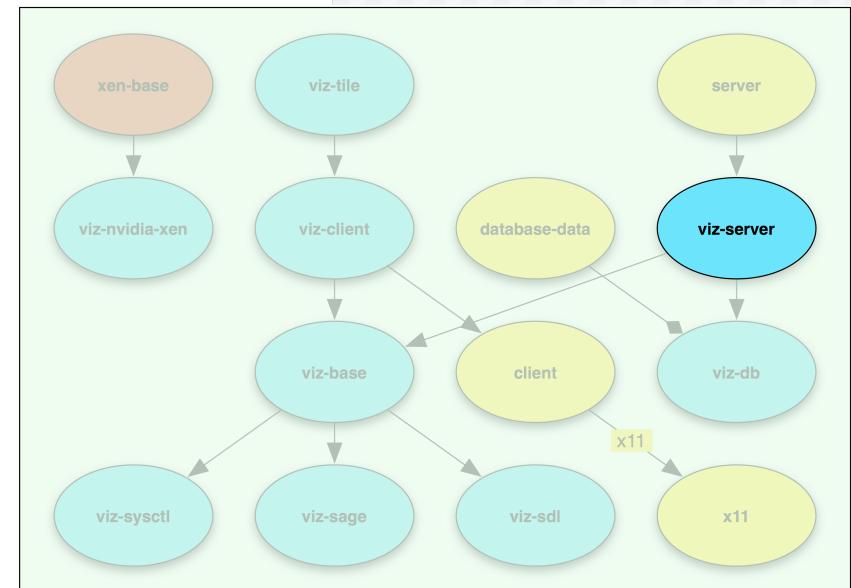


# viz-server

```

1: <kickstart interface="public">
2:
3:   <description>
4:     Visualization Cluster Frontend support
5:   </description>
6:
7:   <package>curl</package>
8:   <package>curl-devel</package>
9:
10:  ...
11:
12:  <package arch="i386">cmake</package>
13:
14: <post>
15:
16: /opt/rocks/bin/rocks add appliance tile \
17:     membership=Tile node=viz-tile
18:
19: /opt/rocks/bin/rocks set attr HideBezels false
20:
21: /opt/rocks/bin/rocks set appliance attr tile x11 true
22:
23: </post>
24:
25: </kickstart>

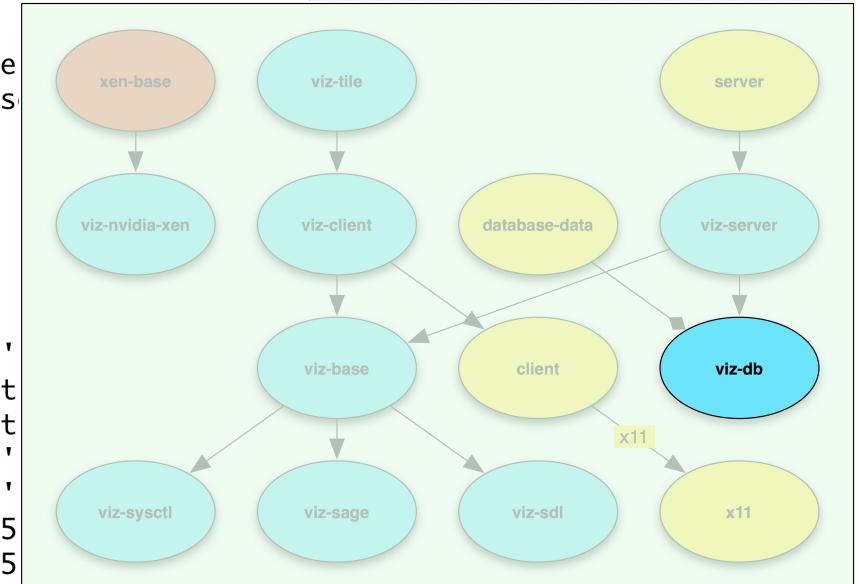
```



# viz-db

```

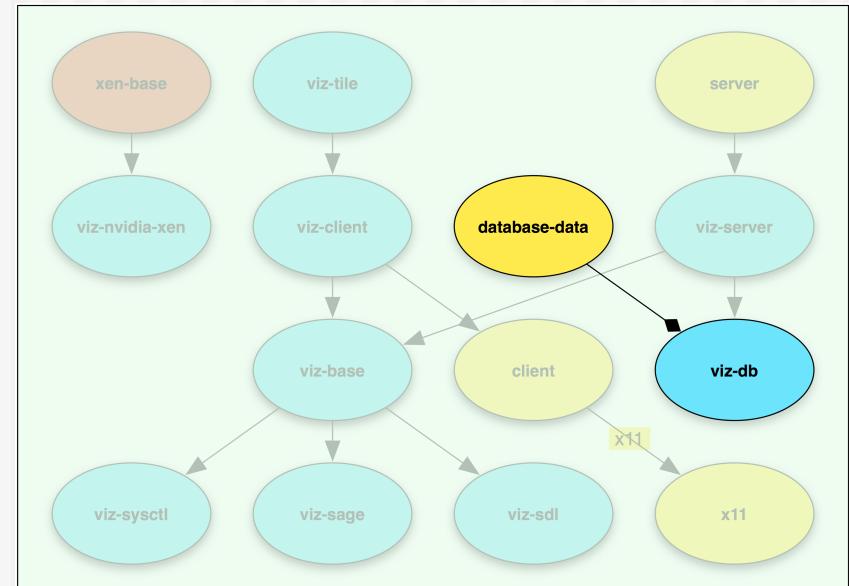
1: <kickstart>
2:
3:     <description>
4: Visualization Cluster Database Tables
5:     </description>
6:
7:     <copyright>
8: Copyright (c) 2000 - 2009 The Regents of the
9: All rights reserved. Rocks(r) v5.2 www.rocks
10:    </copyright>
11:
12: <post>
13:
14: <file name="/tmp/viz-tables.sql">
15: DROP TABLE IF EXISTS videowall;
16: CREATE TABLE videowall (
17:     Node          int(11) NOT NULL default '0'
18:     Display       varchar(8) NOT NULL default
19:     Resolution    varchar(32) NOT NULL default
20:     X             int(11) NOT NULL default '0'
21:     Y             int(11) NOT NULL default '0'
22:     LeftBorder   int(11) NOT NULL default '75'
23:     RightBorder  int(11) NOT NULL default '75'
24:     TopBorder    int(11) NOT NULL default '75',
25:     BottomBorder int(11) NOT NULL default '75'
26: ) TYPE=MyISAM;
27: </file>
28:
29: /opt/rocks/bin/mysql --user=root --password=' cluster &lt; /tmp/viz-
30: tables.sql
31: </post>
32:
33: </kickstart>
```



# Ordering

- ◆ Added DB Schema
- ◆ But, need DB up first

```
<order head="database-data">
    <tail>viz-db</tail>
</order>
```

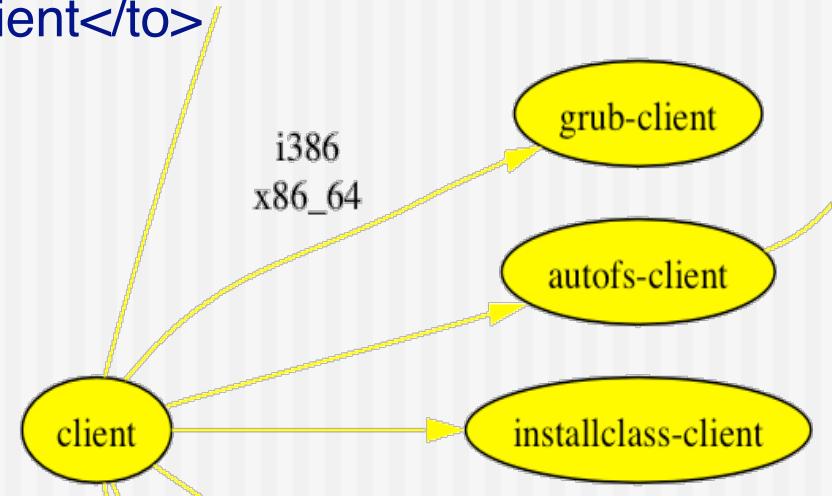


# Another Look at XML

```
<graph>
```

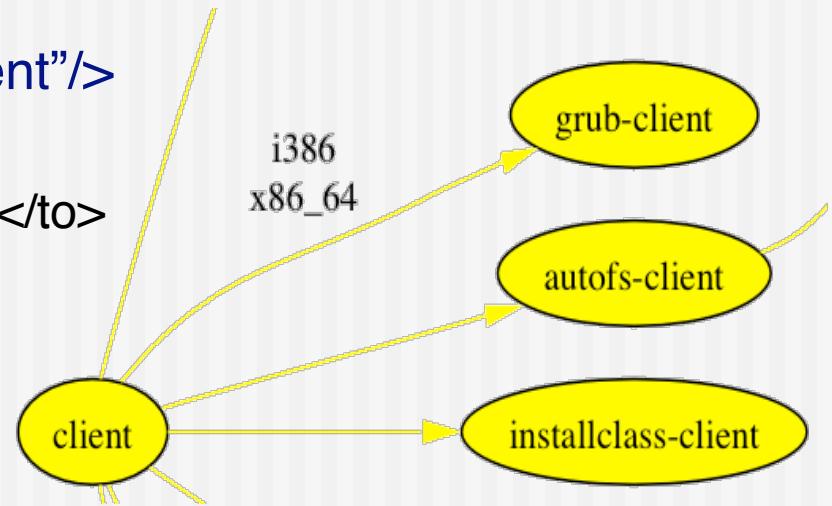
```
  <edge from="client">
    <to arch="i386,x86_64">grub-client</to>
    <to>autofs-client</to>
    <to>installclass-client</to>
  </edge>
```

```
</graph>
```



# Partial Ordering

```
<graph>
  <order head="autofs-client" tail="client"/>
  <edge from="client">
    <to arch="i386,x86_64">grub-client</to>
    <to>autofs-client</to>
    <to>installclass-client</to>
  </edge>
</graph>
```

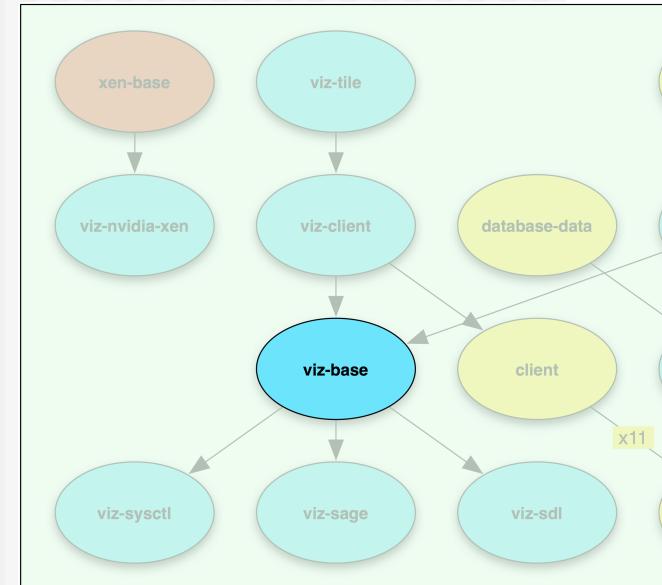


- ◆ Forces autofs-client <post> section to run before client's <post> section
- ◆ In order graph traversal enforces a partial ordering
- ◆ Applying standard graph theory to system installation

# viz-base

```

1: <kickstart interface="public">
2:
3:
4:     <description>
5:         Visualization Cluster support
6:     </description>
7:
8:     <package>rocks-viz</package>
9:     <package>fvwm</package>
10:    <package>nvidia-driver</package>
11:    <package>freeglut</package>
12:    <package>chromium</package>
13:    <package>Cg</package>
14:    <package>wx</package>
15:    ...
16:
17: <post>
18:
19: <!-- Create the viz account on each node (avoid 411) -->
20: /usr/sbin/useradd -M -u402 -c "Viz X11 Owner" -d /opt/viz/etc viz
21:
22: chown -R viz.viz /opt/viz/etc
23:
24: /sbin/chkconfig --add nvidia
25:
26: <file name="/etc/inittab"/>
27: gawk \
28:     '/id:3:initdefault:/ { print "id:5:initdefault:"; next; } \
29:     { print; }' /etc/inittab &gt; /tmp/inittab
30: mv /tmp/inittab /etc/inittab
31:
32: </post>
33:
34: </kickstart>
```



# useradd

---

- ◆ Create a new user named viz
- ◆ Used to auto-login X11 and xhost+ tiles
- ◆ Home is /opt/viz/etc
  - ⇒ Directory is local to every node
  - ⇒ Required for .xauthority files
- ◆ UID is hard coded
  - ⇒ Avoids 411 lookups during install
  - ⇒ Roll developers can request a RESERVED-UID

# rocks/src/roll/RESERVED-UIDS

```
# $Id: RESERVED-UIDS,v 1.17 2009/04/21 00:32:49 anoop Exp $
#
# Register all UIDs created by Rolls here. Third party developers may
# request an ID allocation from devel@rocksclusters.org.
```

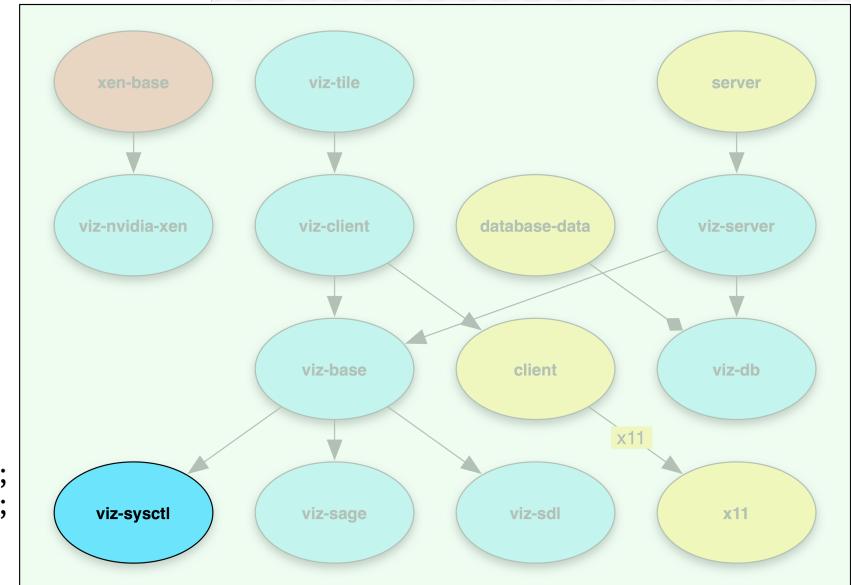
ID	NAME	SOURCE ROLL	NOTES
400	sge	sge	
401	globus	grid	
402	viz	viz	owner of X11 session
403	Rocksdbs	base	foundation mysql user
404	BioUser	bio	user associated with the bioroll
405	jboss	java	jboss not started by default
406	gxmap	gx-map	
407	Condor	condor	condor master user
408	pbs	pbs	pbs batch
409	postgres	postgres8	postgres version 8 roll
410	cvs	cvs-server	
411	cvsanon	cvs-server	
412	tomcat	java	tomcat not started by default
413	nagios	nagios	
414	ssoadmin	camera	SSO admin user
415	wsuser	camera	ws user for dropbox
416	hg	-----	mercurial user
417	boinc	boinc	boinc user
418	moab	moab	moab scheduler user
419	zfsuser	thumper-conf	ZFS Replication user

# viz-sysctl

```

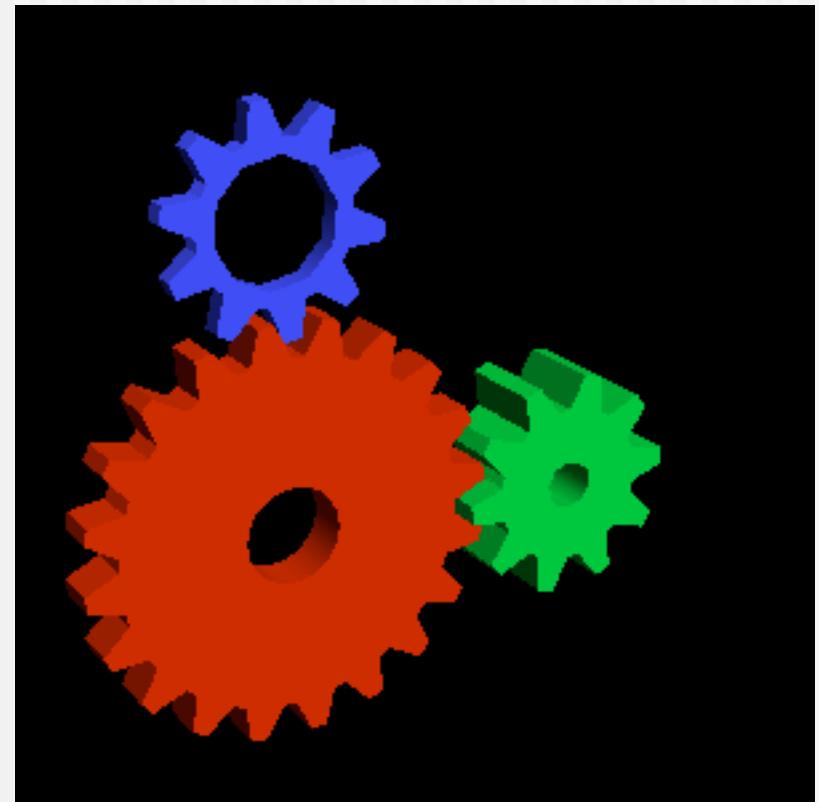
1: <kickstart>
2:
3:
4:     <description>
5:         Increase network buffers
6:     </description>
7:
8: <post>
9:
10: <file name="/etc/sysctl.conf"/>
11: <file name="/tmp/sysctl.awk">
12: /net\.\core\.\rmem_max/          { next; }
13: /net\.\core\.\wmem_max/         { next; }
14: /net\.\ipv4\.\tcp_rmem/        { next; }
15: /net\.\ipv4\.\tcp_wmem/        { next; }
16: /net\.\ipv4\.route\.flush/    { next; }
17: /net\.\core\.\rmem_max/          { next; }
18: { print; }
19: END {
20:     print "net.core.rmem_max = 16777216";
21:     print "net.core.wmem_max = 16777216";
22:     print "net.ipv4.tcp_rmem = 4096 87380 16777216";
23:     print "net.ipv4.tcp_wmem = 4096 16384 16777216";
24:     print "net.ipv4.route.flush = 1";
25: }
26: </file>
27: awk -f /tmp/sysctl.awk /etc/sysctl.conf &gt; /tmp/sysctl.conf
28: cp /tmp/sysctl.conf /etc/sysctl.conf
29: rm /tmp/sysctl.awk /tmp/sysctl.conf
30:
31: </post>
32:
33: </kickstart>

```



# Deep Pipes

- ◆ Performance Tuning
- ◆ Glxgears was choppy
- ◆ Re-wrote Chromium network stack
- ◆ Still broken
- ◆ Months later increased network buffers
- ◆ Everything worked

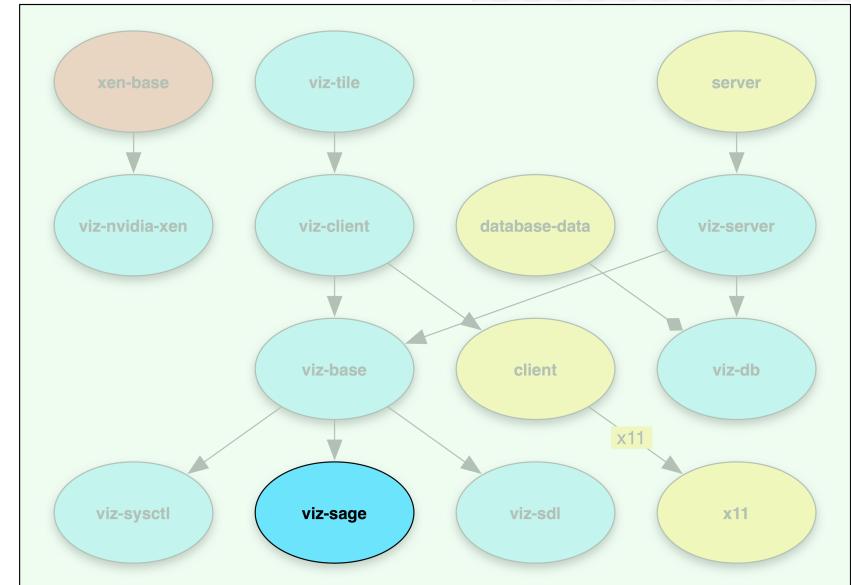


# viz-sage

```

1: <kickstart>
2:
3:   <description>
4:     Common Setup for all SAGE nodes
5:   </description>
6:
7:
8:   <package>wx</package>
9:
10:  <package>libtool-libs</package>
11:  <package>libtool</package>
12:
13:  <package>mpeg2dec</package>
14:  <package>libraw1394</package>
15:  <package>libiec61883</package>
16:  <package>QUANTA</package>
17:  <package>sage</package>
18:
19: </kickstart>

```

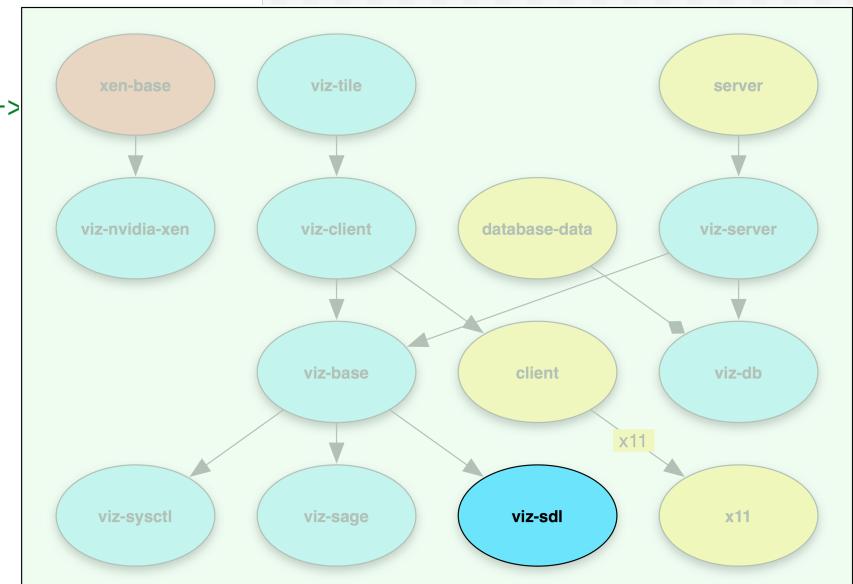


# viz-sdl

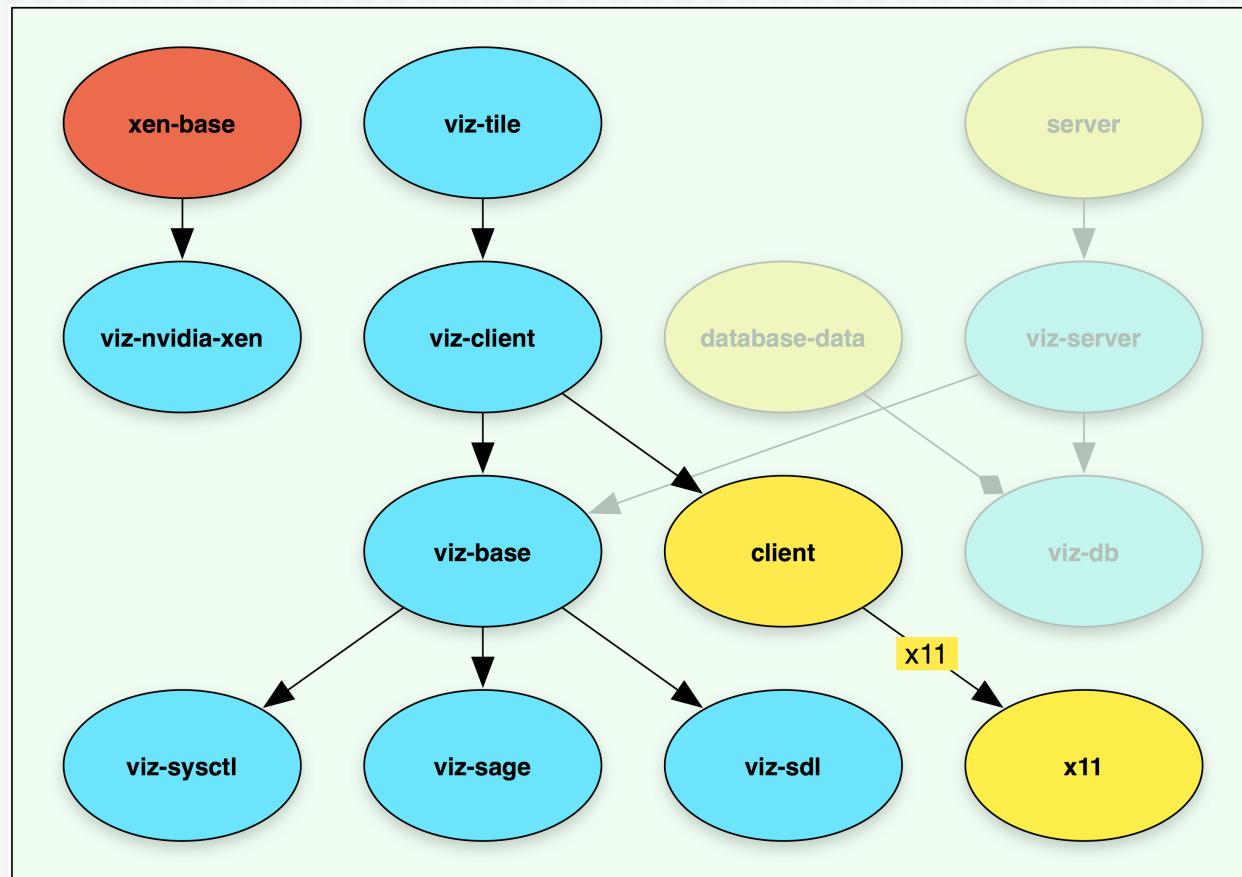
```

1: <kickstart>
2:
3:   <description>
4:     SDL stuff
5:   </description>
6:
7:   <!-- SDL needs stuff we trim off the base CD -->
8:   <package>arts</package>
9:   <package>qt</package>
10:
11:  <!-- SDL is now in the foundation -->
12:  <package>foundation-sdl</package>
13:  <package>foundation-sdl-image</package>
14:  <package>foundation-sdl-mixer</package>
15:  <package>foundation-sdl-net</package>
16:  <package>foundation-sdl-ttf</package>
17:
18: </kickstart>

```



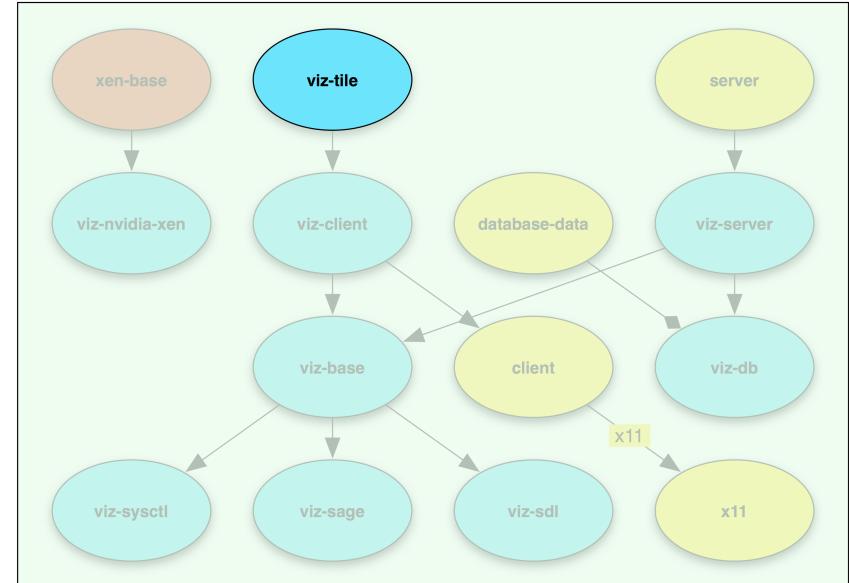
# Tile Profile



# viz-tile

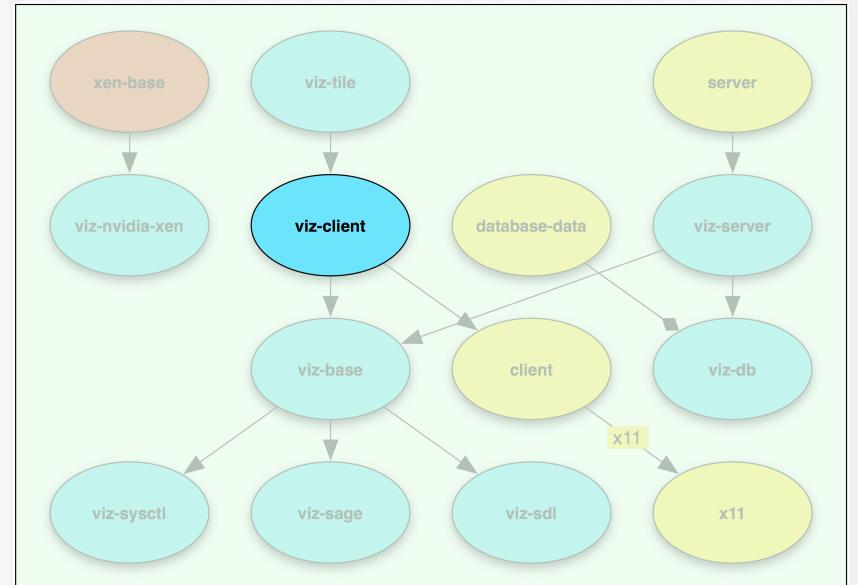
```

1: <kickstart interface="public">
2:
3:   <description>
4:     Display Node
5:   </description>
6:
7:   <post>
8:
9:     <!-- setup autologin for viz user -->
10:    <file name="/etc/gdm/custom.conf">
11:      [daemon]
12:      TimedLoginEnable=true
13:      TimedLogin=viz
14:      TimedLoginDelay=5
15:
16:      [security]
17:      DisallowTCP=false
18:    </file>
19:
20:   </post>
21:
22: </kickstart>
```

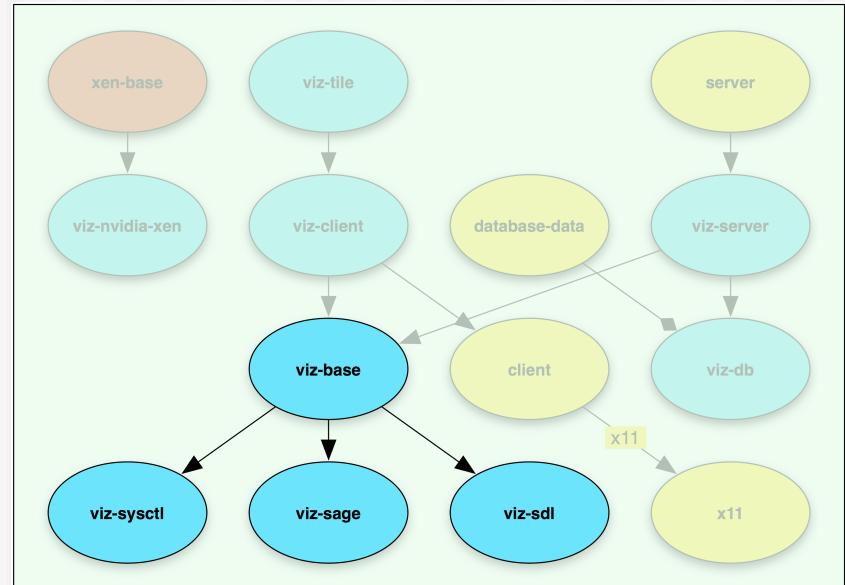


ROCKS

# viz-client



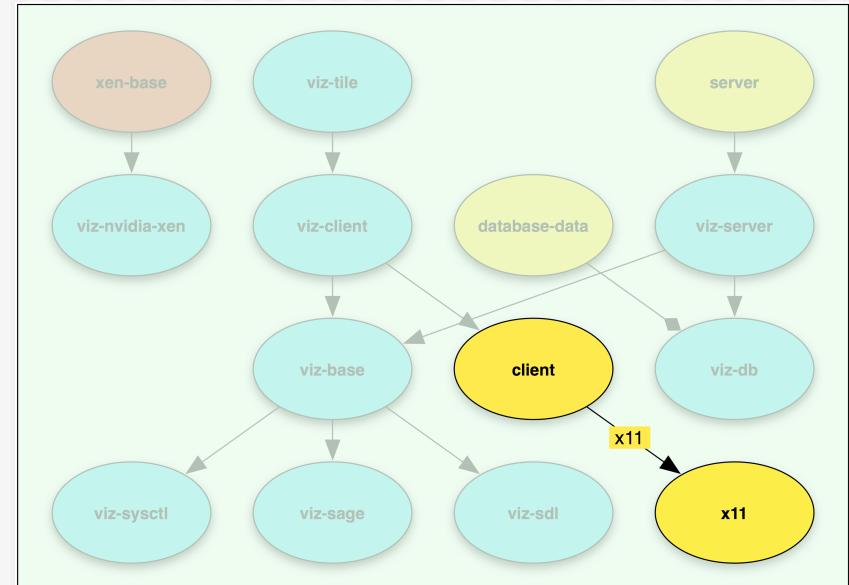
# viz-base ...



# client -> x11

- ◆ Conditional Edge from Base Roll
- ◆ Uses x11 attribute
- ◆ Set in viz-server

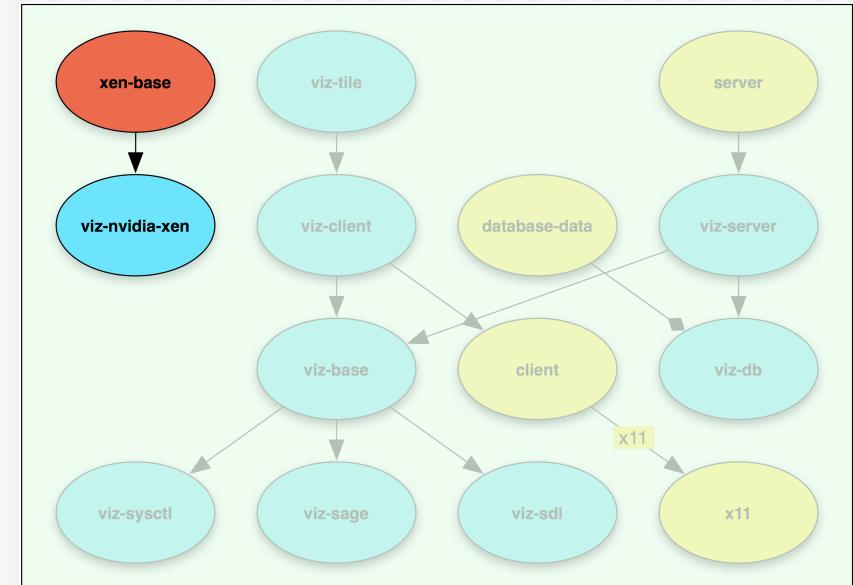
```
<edge from="client"
      to="x11" cond="x11" />
```



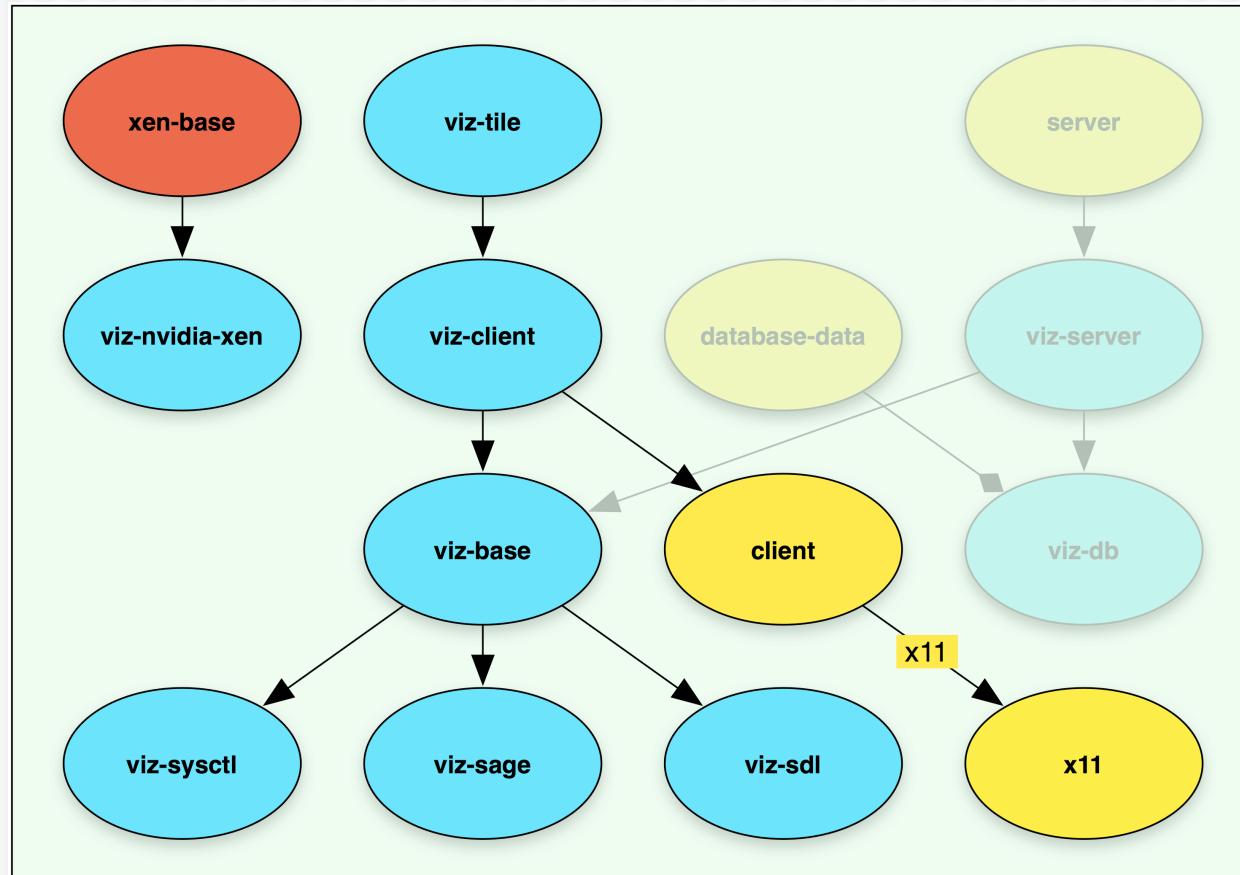
# xen-base

```

1: <kickstart>
2:
3:     <description>
4:         Patch the nVidia driver to make it work in dom0
5:     </description>
6:
7:     <package>nvidia-xen</package>
8:
9: <post>
10:
11: <file name="/opt/viz/drivers/nvidia/install-driver" perms="755">
12: <![CDATA[
13: #!/bin/bash
14: if [ -e /usr/src/linux-2.6 ]
15: then
16:     ln -s /usr/src/linux-2.6 /usr/src/linux
17: elif [ -e /usr/src/linux-2.4 ]
18: then
19:     ln -s /usr/src/linux-2.4 /usr/src/linux
20: fi
21:
22: /opt/viz/drivers/nvidia/NVIDIA*run --no-network -x > /dev/null 2>&1
23:
24: cd NVIDIA*/usr/src/nv
25: zcat /opt/viz/drivers/nvidia/nvidia_xenpatch.gz | \
26:     patch --forward -p4 --quiet > /dev/null 2>&1
27:
28: make SYSSRC=/usr/src/linux module > /dev/null 2>&1
29:
30: install -D -o root -g root -m 0644 nvidia.ko \
31:         /lib/modules/`uname -r`/video/nvidia.ko
32:
33: depmod -a
34:
35: cd ../../..
36: ./nvidia-installer --no-kernel-module -s > /dev/null 2>&1
37: ]]>
38: </file>
39:
40: </post>
41:
42: </kickstart>
43:
```

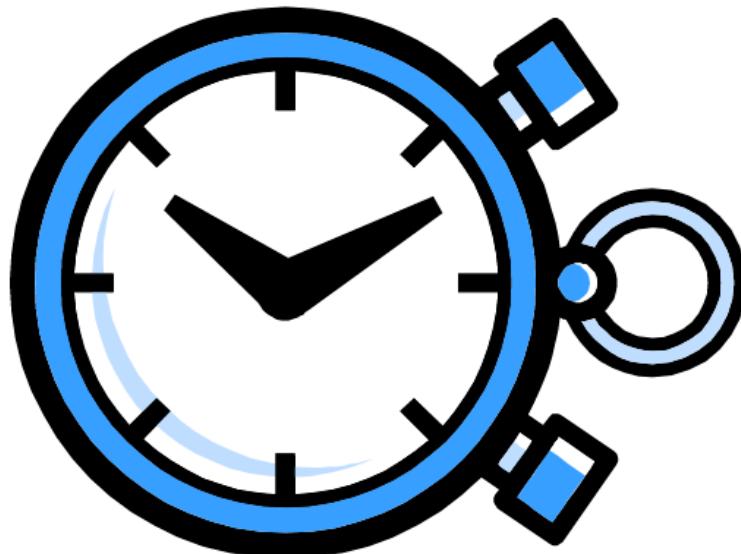


**ROCKS**



ROCKS

# It's Python Time!





# Motivation

---

- ◆ Lack of consistency in Rocks commands
  - ⇒ add-extra-nic (15 flags)
  - ⇒ 411put
  - ⇒ rocks-dist
  - ⇒ dbreport (~ a dozen reports)
- ◆ Extensible to other groups
  - ⇒ How do I add a flag to an existing command?
  - ⇒ How do I add a new command?
  - ⇒ How do I document my command?

# Verb Based

- ◆ “add”, “set”, “enable”, ...
  - ⇒ Modify the cluster database
- ◆ “list”, “dump”, “report”
  - ⇒ Inspect the cluster database
- ◆ About 20 verbs in the command line so far
- ◆ You can even add your own



# Grammar

- ◆ rocks <verb> <object...> <subject> <params...>
  - ⌚ Object is general to specific
    - “host” “interface”
    - “network” “subnet”
    - “viz” “layout”
  - ⌚ Subject is typed
    - host
    - appliance
    - Network
- ◆ Example:

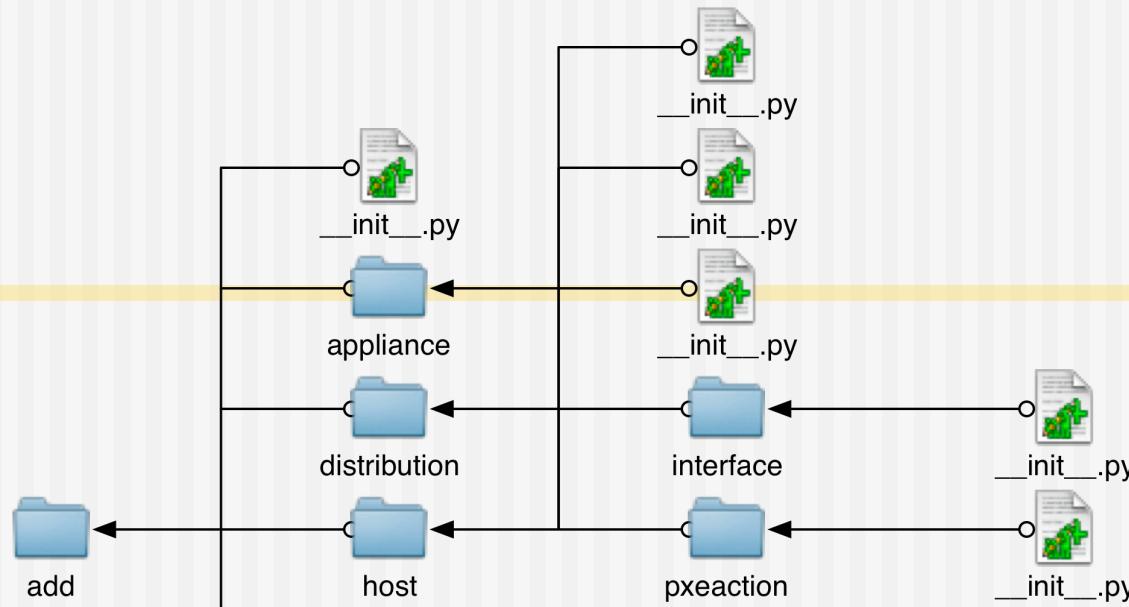
```
# rocks list host interface tile-0-0
SUBNET  IFACE MAC          IP          NETMASK   GATEWAY MODULE NAME
private eth0  00:13:72:ba:be:42 10.255.255.254 255.0.0.0 ----- tg3    tile-0-0
----- eth1  00:0a:5e:1a:6d:64 ----- ----- ----- skge  -----
```

# Implementation

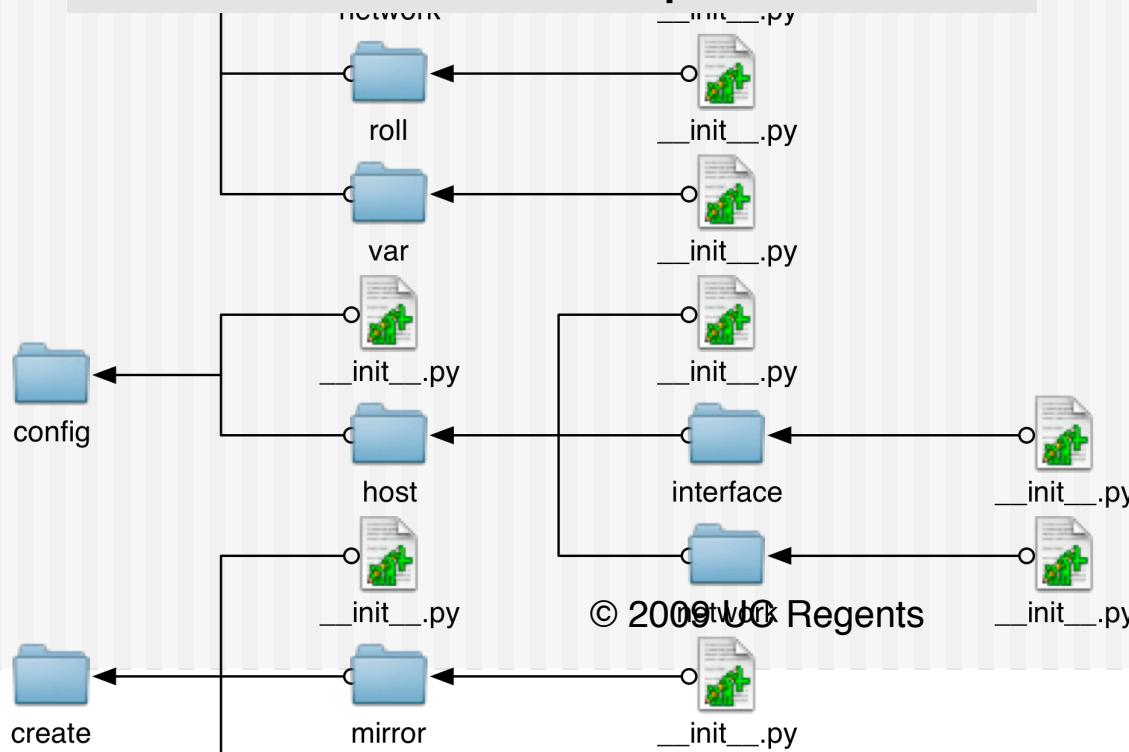
---

- ◆ Python
  - ⇒ Similar to existing dbreport code
  - ⇒ Very small modules
- ◆ Command line is identical to the directory hierarchy
  - ⇒ Verbs are directories
  - ⇒ Objects are directories
  - ⇒ Subjects are `__init__.py` files
- ◆ Commands are added by adding directories

**ROCKS**

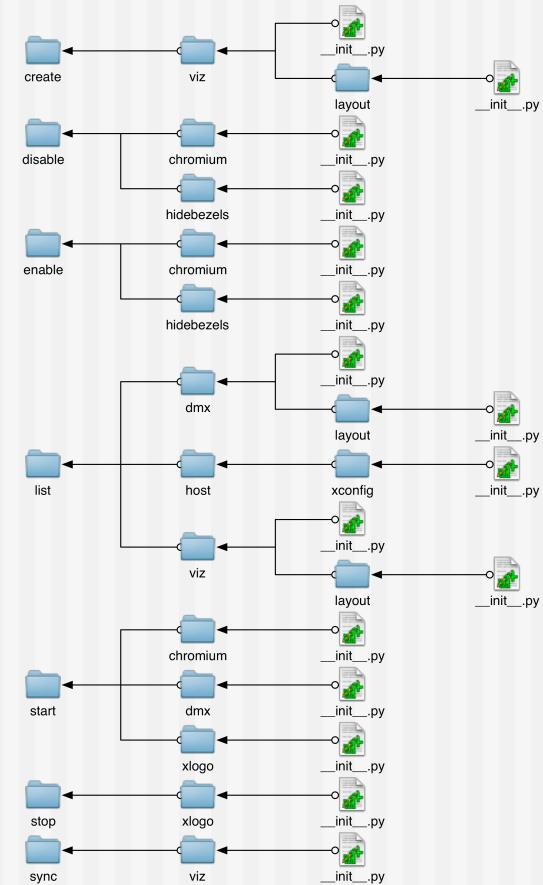


**rocks add host pxeaction**



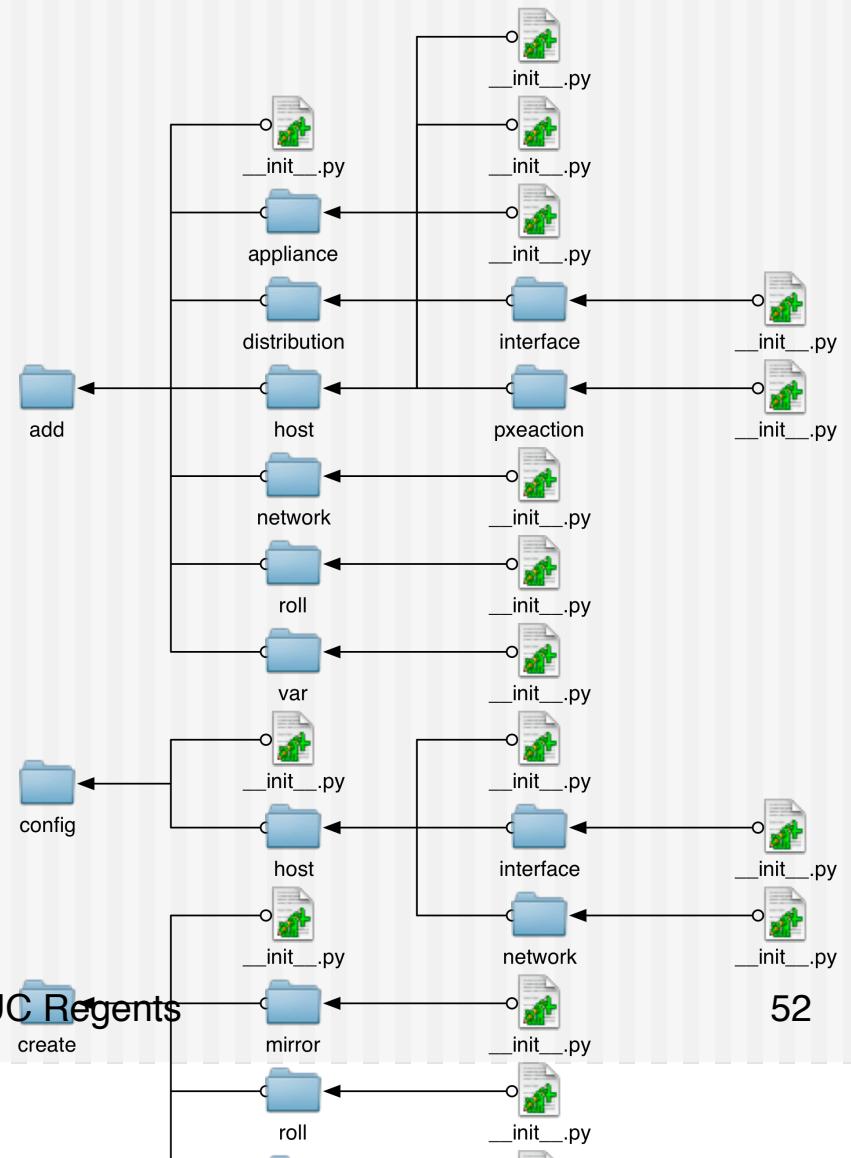
# Rolls Can Add Commands

- ◆ Similar to the configuration graph
- ◆ Rolls can add command line
  - ↳ Files : commands
  - ↳ Directories : verbs and objects
- ◆ Think hard before adding another verb



# add

- ◆ Creates new entries in the cluster database
- ◆ Examples:
  - ➲ Hosts
  - ➲ Appliances
  - ➲ Rolls



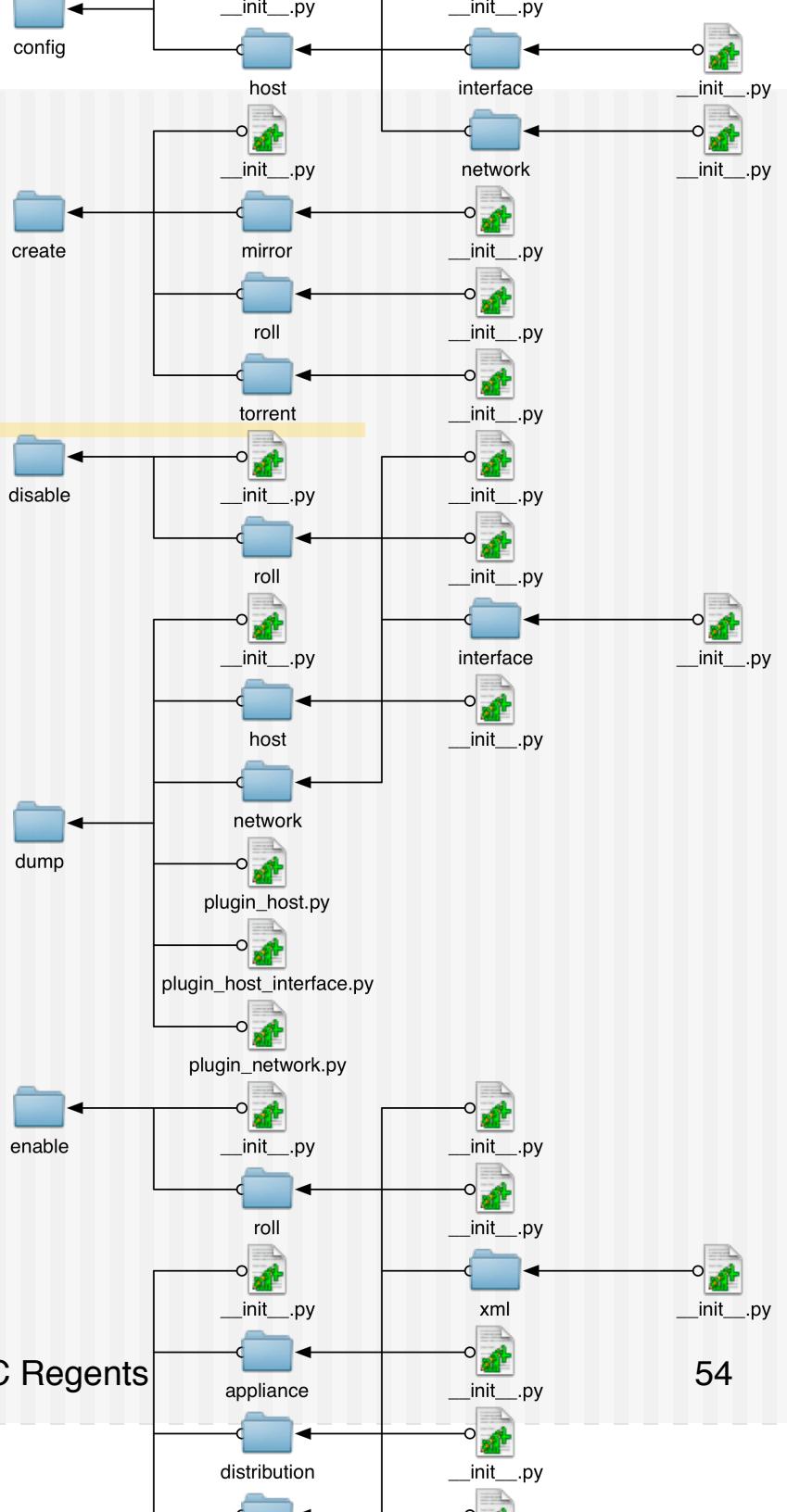
# rocks add distribution

```
1: import rocks.commands
2:
3: class Command(rocks.commands.DistributionArgumentProcessor,
4:               rocks.commands.add.command):
5:     """
6:     Add a distribution specification to the database.
7:
8:     <arg type='string' name='distribution'>
9:         Name of the new distribution.
10:    </arg>
11:
12:    <example cmd='add distribution rocks-dist'>
13:        Adds the distribution named "rocks-dist" into the database.
14:    </example>
15:    """
16:
17:    def run(self, params, args):
18:
19:        if len(args) != 1:
20:            self.abort('must supply one distribution')
21:        dist = args[0]
22:
23:        if dist in self.getDistributionNames():
24:            self.abort('distribution "%s" exists' % dist)
25:
26:        self.db.execute("""insert into distributions (name) values
27:                         ('%s')""" % dist)
28:
29:
```

ROCKS

# dump

- ◆ Returns cluster database information in the form of rocks command lines
- ◆ Examples:
  - ↳ Hosts
  - ↳ Network
- ◆ Same as –dump flag on insert-ethers



# rocks dump host

```
# rocks dump host
/opt/rocks/bin/rocks add host vizagra cpus=1 rack=0 rank=0 membership="Frontend"
/opt/rocks/bin/rocks add host tile-0-1 cpus=2 rack=0 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-0 cpus=2 rack=0 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-2 cpus=2 rack=0 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-3 cpus=2 rack=0 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-3 cpus=2 rack=1 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-2 cpus=2 rack=1 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-1 cpus=2 rack=1 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-0 cpus=2 rack=1 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-0 cpus=2 rack=2 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-1 cpus=2 rack=2 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-2 cpus=2 rack=2 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-3 cpus=2 rack=2 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-0 cpus=2 rack=3 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-1 cpus=2 rack=3 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-2 cpus=2 rack=3 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-3 cpus=2 rack=3 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-0 cpus=2 rack=4 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-1 cpus=2 rack=4 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-2 cpus=2 rack=4 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-3 cpus=2 rack=4 rank=3 membership="Tile"
```

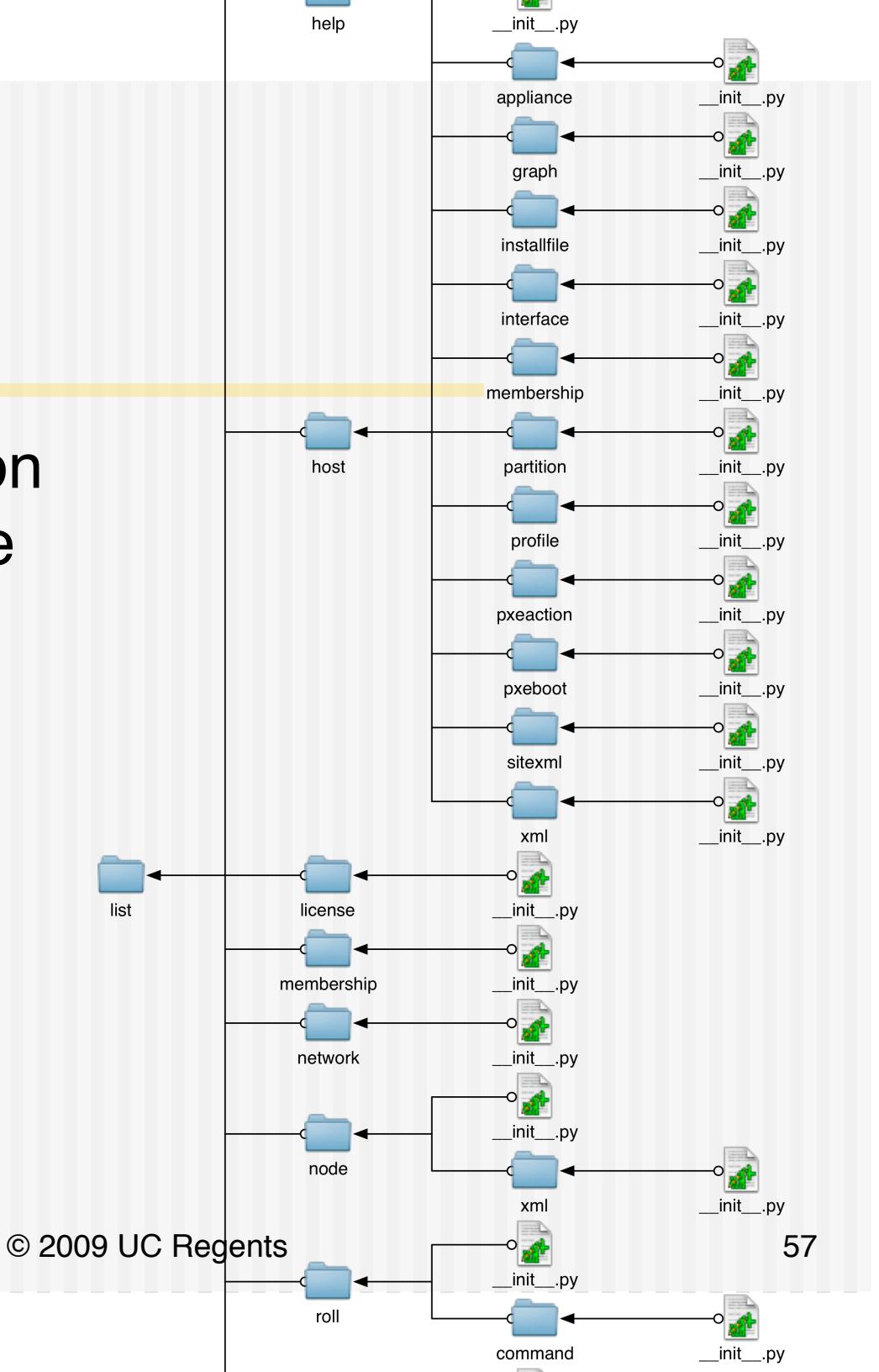
# rocks dump host

```
1: import os
2: import sys
3: import string
4: import rocks.commands
5:
6: class command(rocks.commands.HostArgumentProcessor,
7:                 rocks.commands.dump.command):
8:     pass
9:
10: class Command(command):
11:     """
12:     Dump the host information as rocks commands.
13:
14:     <arg optional='1' type='string' name='host' repeat='1'>
15:         Zero, one or more host names. If no host names are supplied,
16:         information for all hosts will be listed.
17:     </arg>
18:
19:     <example cmd='dump host compute-0-0'>
20:         Dump host compute-0-0 information.
21:     </example>
22:
23:     <example cmd='dump host compute-0-0 compute-0-1'>
24:         Dump host compute-0-0 and compute-0-1 information.
25:     </example>
26:
27:     <example cmd='dump host'>
28:         Dump all hosts.
29:     </example>
30:     """
31:
32:     def run(self, params, args):
33:         for host in self.getHostnames(args):
34:             self.db.execute("""select
35:                             n.cpus, n.rack, n.rank, m.name
36:                             from nodes n, memberships m where
37:                             n.membership=m.id and n.name='%s'" % host)
38:             (cpus, rack, rank, membership) = self.db.fetchone()
39:             self.dump('add host %s cpus=%s rack=%s rank=%s '
40:                      'membership="%s"' %
41:                      (host, cpus, rack, rank, membership))
42:
43:
```



# list

- ◆ Reports information in human readable format
- ◆ No side-effects on the database
- ◆ Examples:
  - ⌚ Hosts
  - ⌚ Appliances
  - ⌚ Rolls



# rocks list host

```
# rocks list host
HOST      MEMBERSHIP CPUS RACK RANK COMMENT
vizagra: Frontend   1    0    0    -----
tile-0-1: Tile     2    0    1    -----
tile-0-0: Tile     2    0    0    -----
tile-0-2: Tile     2    0    2    -----
tile-0-3: Tile     2    0    3    -----
tile-1-3: Tile     2    1    3    -----
tile-1-2: Tile     2    1    2    -----
tile-1-1: Tile     2    1    1    -----
tile-1-0: Tile     2    1    0    -----
tile-2-0: Tile     2    2    0    -----
tile-2-1: Tile     2    2    1    -----
tile-2-2: Tile     2    2    2    -----
tile-2-3: Tile     2    2    3    -----
tile-3-0: Tile     2    3    0    -----
tile-3-1: Tile     2    3    1    -----
tile-3-2: Tile     2    3    2    -----
tile-3-3: Tile     2    3    3    -----
tile-4-0: Tile     2    4    0    -----
tile-4-1: Tile     2    4    1    -----
tile-4-2: Tile     2    4    2    -----
tile-4-3: Tile     2    4    3    -----
```

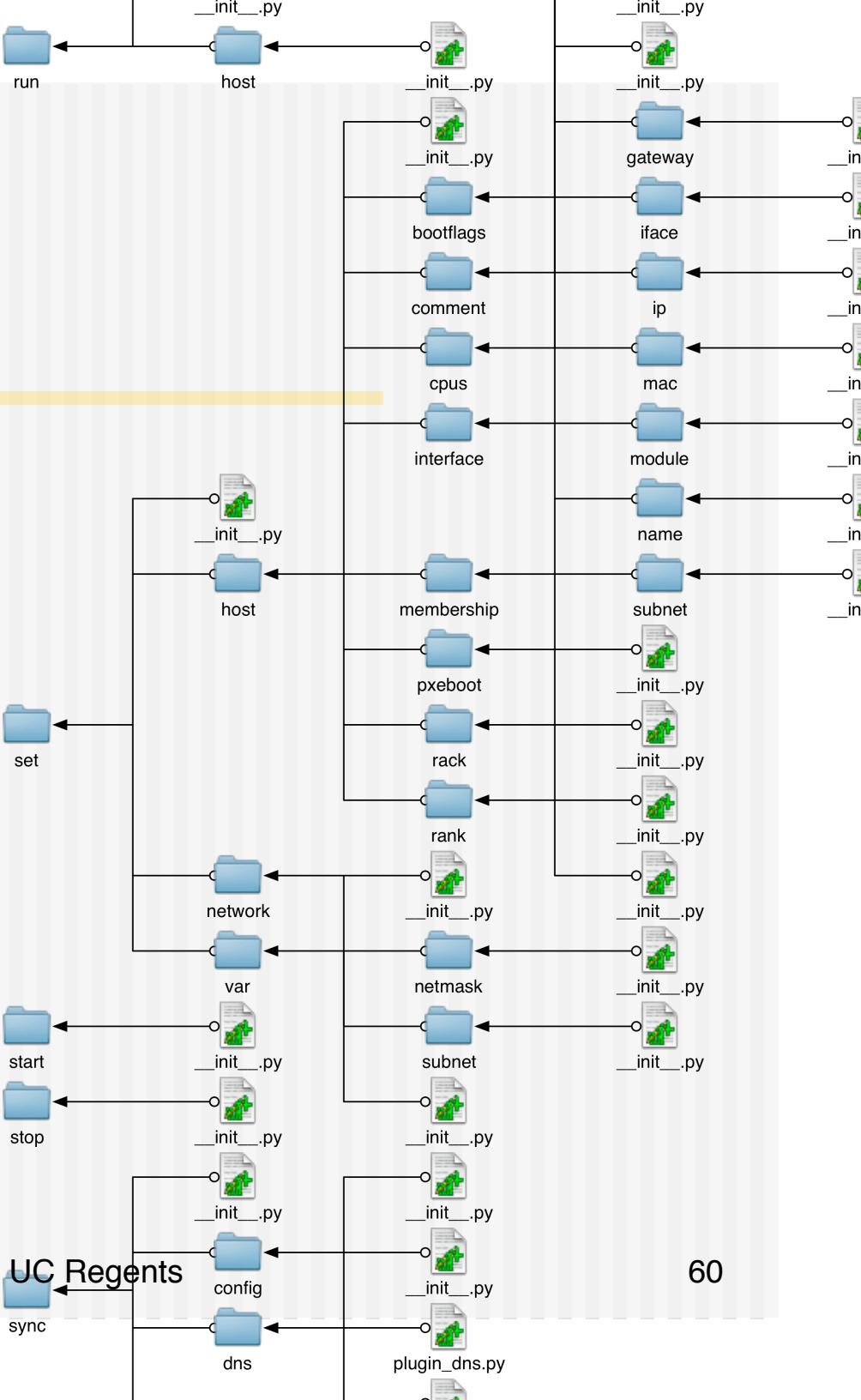
# rocks list host

```
1: import rocks.commands
2:
3: class command(rocks.commands.HostArgumentProcessor,
4:                 rocks.commands.list.command):
5:     pass
6:
7: class Command(command):
8:     """
9:     List the membership, CPU count, physical position info and comment for
10:    a list of hosts.
11:
12:    <arg optional='1' type='string' name='host' repeat='1'>
13:        Zero, one or more host names. If no host names are supplied, info about
14:        all the known hosts is listed.
15:    </arg>
16:
17:    <example cmd='list host compute-0-0'>
18:        List info for compute-0-0.
19:    </example>
20:
21:    <example cmd='list host'>
22:        List info for all known hosts.
23:    </example>
24:    """
25:
26:    def run(self, params, args):
27:        self.beginOutput()
28:
29:        for host in self.getHostnames(args):
30:            self.db.execute("""select m.name, n.cpus,
31:                            n.rack, n.rank, n.comment from
32:                            nodes n, memberships m where
33:                            n.membership=m.id and n.name=%s""", % host)
34:            self.addOutput(host, self.db.fetchone())
35:
36:        self.endOutput(header=['host', 'membership',
37:                              'cpus', 'rack', 'rank', 'comment'])
```

**ROCKS**

# set

- ◆ Modifies entries in the cluster database
- ◆ Examples:
  - ↳ Network Interfaces
  - ↳ Appliance Assignment
  - ↳ Rack / Rank
- ◆ add-extra-nic
  - ↳ Rocks add host interface
  - ↳ Rocks set host interface



# add-extra-nic is now ...

## ◆ Start

```
# rocks list host interface compute-1-1
SUBNET  IFACE MAC           IP           NETMASK     GATEWAY MODULE NAME
private  eth0   00:0e:0c:5d:7e:5e 10.255.255.251 255.0.0.0 ----- e1000  compute-1-1
----- eth1   00:30:1b:b2:ea:61 ----- ----- ----- tg3      -----
```

## ◆ Configure

```
# rocks set host interface ip compute-1-1 eth1 192.168.1.1
# rocks set host interface gateway compute-1-1 eth1 192.168.1.254
# rocks set host interface name compute-1-1 eth1 fast-1-1
# rocks set host interface subnet compute-1-1 eth1 public
```

## ◆ Verify

```
# rocks list host interface compute-1-1
SUBNET  IFACE MAC           IP           NETMASK     GATEWAY      MODULE NAME
private  eth0   00:0e:0c:5d:7e:5e 10.255.255.251 255.0.0.0 ----- e1000  compute-1-1
public   eth1   00:30:1b:b2:ea:61 192.168.1.1    255.255.255.0 192.168.1.254 tg3      fast-1-1
```

# rocks set host interface ip

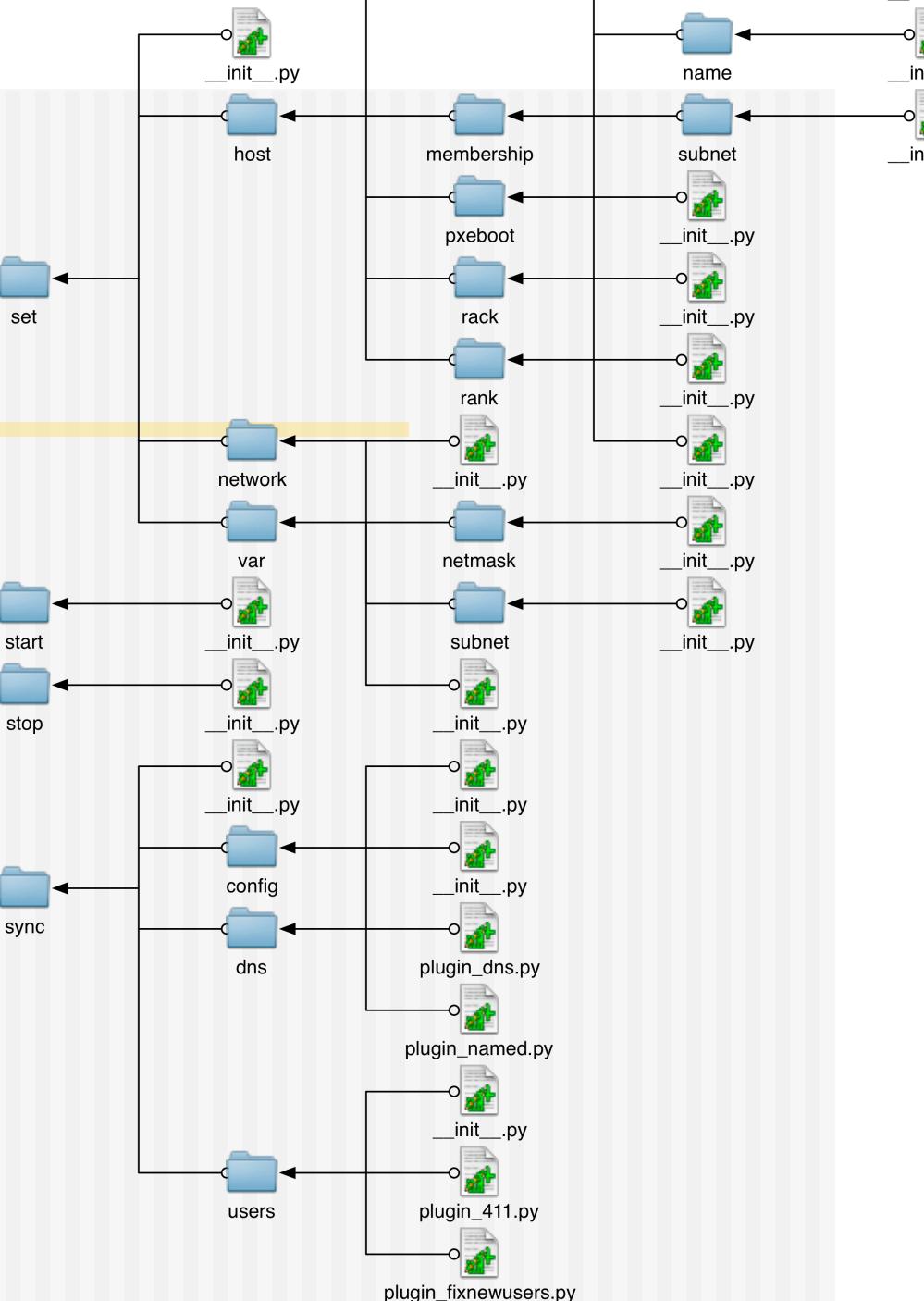
```

1: import rocks.commands
2:
3: class Command(rocks.commands.set.host.command):
4:     """
5:     Sets the IP address for the named interface for one host.
6:
7:     <arg type='string' name='host'>
8:         Host name.
9:     </arg>
10:
11:    <arg type='string' name='iface'>
12:        Interface that should be updated. This may be a logical interface
13:        or the physical address of the interface.
14:    </arg>
15:
16:    <arg type='string' name='ip'>
17:        The IP address of the interface. Usually of the form nnn.nnn.nnn.nnn
18:        where n is a decimal digit. This format is not enforced. Use IP=NULL
19:        to clear.
20:    </arg>
21:
22:    <param type='string' name='iface'>
23:        Can be used in place of the iface argument.
24:    </param>
25:
26:    <param type='string' name='ip'>
27:        Can be used in place of the ip argument.
28:    </param>
29:
30:
31: <example cmd='set host interface ip compute-0-0 eth1 192.168.0.10'>
32:     Sets the IP Address for the eth1 device on host compute-0-0.
33: </example>
34:
35: <example cmd='set host interface ip compute-0-0 iface=eth1 ip=192.168.0.10'>
36:     Same as above.
37: </example>
38:
39: <related>set host interface iface</related>
40: <related>set host interface ip</related>
41: <related>set host interface gateway</related>
42: <related>set host interface module</related>
43: <related>add host</related>
44: """
45:
46: def run(self, params, args):
47:
48:     (args, iface, ip) = self.fillPositionalArgs(('iface', 'ip'))
49:
50:     hosts = self.getHostnames(args)
51:
52:     if len(hosts) != 1:
53:         self.abort('must supply one host')
54:     if not iface:
55:         self.abort('must supply iface')
56:     if not ip:
57:         self.abort('must supply ip')
58:
59:     ip = ip.upper() # null -> NULL
60:
61:     for host in hosts:
62:         self.db.execute("""update networks, nodes set
63:             networks.ip=NULLIF(%s,'NULL') where
64:             nodes.name=%s and networks.node=nodes.id and
65:             (networks.device=%s or networks.mac=%s)""", %
66:             (ip, host, iface, iface))
67:
```

**ROCKS**

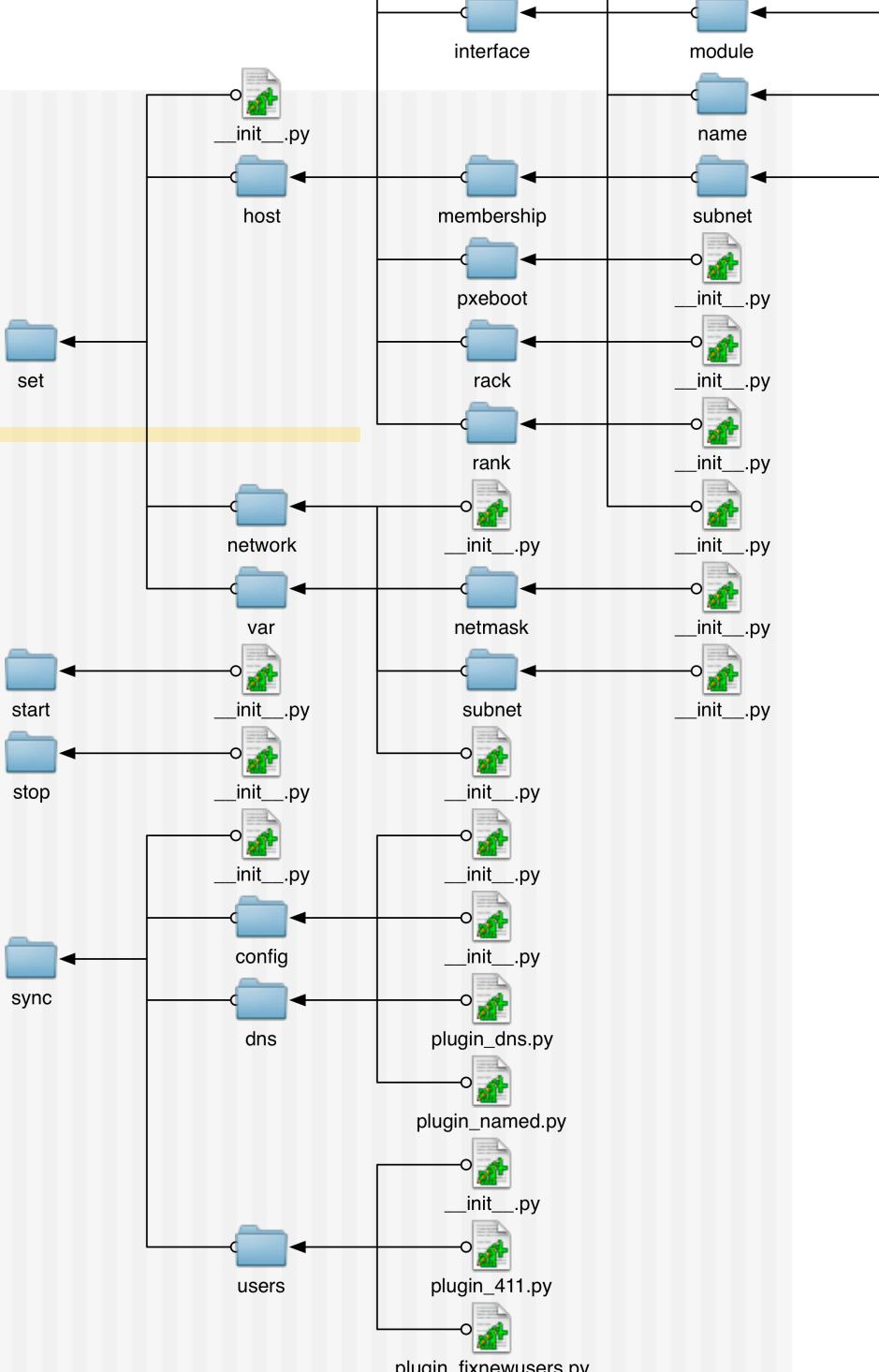
# start / stop

- ◆ Start and stop something
- ◆ NULL commands
- ◆ Reserve the verbs for use on other Rolls
- ◆ Think “abstract base class”



# sync

- ◆ Synchronizes the database state to software configuration files
- ◆ Similar to the old “insert-ethers – update”
- ◆ Not complete yet, see Rocks VI (5.1)



# rocks sync config

```
1: import os
2: import sys
3: import string
4: import rocks.file
5: import rocks.commands
6:
7:
8: class Command(rocks.commands.sync.command):
9:     """
10:         For each system configuration file controlled by Rocks, first
11:         rebuild the configuration file by extracting data from the
12:         database, then restart the relevant services.
13:
14:         <example cmd='sync config'>
15:             Rebuild all configuration files and restart relevant services.
16:         </example>
17:         """
18:
19:     def run(self, params, args):
20:         cmd = '/opt/rocks/sbin/insert-ethers --update'
21:         for line in os.popen(cmd).readlines():
22:             self.owner.addText(line)
23:
```

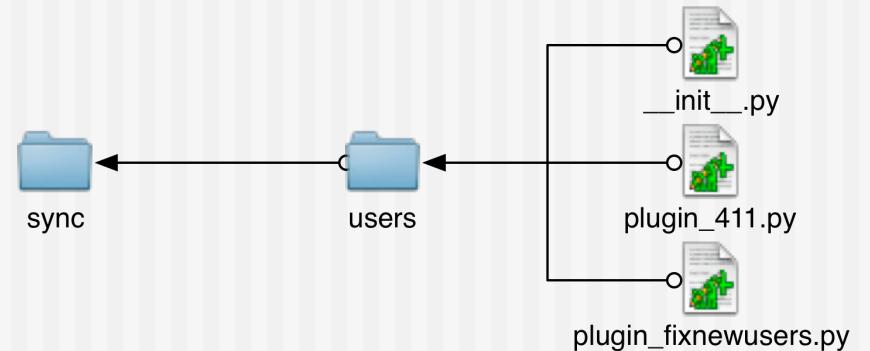
# Extensibility

---

- ◆ New commands
  - ➲ Add directories
  - ➲ Add `__init__.py` code
- ◆ Existing commands
  - ➲ Some commands can be extended
  - ➲ Plugins

# rocks sync users

- ◆ Run after useradd
  - Populate auto.home
  - Cleanup password file
  - Send 411 files
- ◆ Two plugins
  - Fixnewusers
  - 411
- ◆ Partial Ordering
- ◆ Other Rolls can add more plugins to this command
- ◆ Command must be design for plugins (not default)



# \_\_init\_\_.py

```
1: import rocks.commands
2:
3:
4: class Command(rocks.commands.sync.command):
5:     """
6:     Update all user-related files (e.g., /etc/passwd, /etc/shadow, etc.)
7:     on all known hosts. Also, restart autofs on all known hosts.
8:
9:     <example cmd='sync users'>
10:        Send all user info to all known hosts.
11:    </example>
12:    """
13:
14:    def run(self, params, args):
15:        self.runPlugins()
```

# 411 plugin

```
1: import os
2: import rocks.commands
3:
4: class Plugin(rocks.commands.Plugin):
5:     """Force a 411 update and re-load autofs on all nodes"""
6:
7:     def provides(self):
8:         return '411'
9:
10:    def requires(self):
11:        return ['fixnewusers']
12:
13:    def run(self, args):
14:        #
15:        # force the rebuild of all files under 411's control
16:        #
17:        for line in os.popen('make -C /var/411 force').readlines():
18:            self.owner.addText(line)
19:
20:        #
21:        # restart autofs on all known hosts
22:        #
23:        cmd = '/opt/rocks/bin/tentakel "service autofs reload"'
24:        for line in os.popen(cmd).readlines():
25:            self.owner.addText(line)
26:
27:
```

# auto.home / passwd plugin

```

1: import os
2: import string
3: import rocks.commands
4:
5: class Plugin(rocks.commands.Plugin):
6:     """Relocates home directories to /export and fixes autofs.home"""
7:
8:     def provides(self):
9:         return 'fixnewusers'
10:
11    def run(self, args):
12        # scan the password file for any '/export/home' entries
13        # this is the default entry as setup by useradd
14        new_users = []
15        default_dir = '/export/home/'
16:
17        file = open('/etc/passwd', 'r')
18:
19        for line in file.readlines():
20            l = string.split(line[:-1], ':')
21:
22            if len(l) < 6:
23                continue
24:
25            username = l[0]
26            homedir = l[5]
27:
28            if homedir[:len(default_dir)] == default_dir:
29                new_users.append(username)
30        file.close()
31:
32        hostname = '%s.%s' % \
33                    (self.db.getGlobalVar('Kickstart', 'PrivateHostname'),
34                     self.db.getGlobalVar('Kickstart', 'PrivateDNSDomain'))
35:
36        for user in new_users:
37:
38            # for each new user, change their default directory to
39            # /home/<username>
40            cmd = '/usr/sbin/usermod -d %s %s' % \
41                  (os.path.join('/home', user), user)
42            for line in os.popen(cmd).readlines():
43                self.owner.addText(line)
44:
45            # then update the auto.home file

```

# Argument Processing

- ◆ `rocks <verb> <object...> <subject>`  
`<params...>`
- ◆ Subject is typed by first object
  - ⌚ host -> one or more hostname
  - ⌚ roll -> one or more roll names
- ◆ Params are in key=value form
- ◆ Same as `-flag=value` but easier to read

# Helper classes and functions

- ◆ ArgumentProcessors

- ↳ Class to parse the subject in a standard way
  - ↳ Exists for hosts, rolls, appliances, ...

- ◆ Parameters Parsing

- ↳ fillPositionalArgs
  - ↳ fillParams

# HostArgumentProcessor

- ◆ Command must derive from  
`rocks.commands.HostArgumentProcessor`
- ◆ `self.getHostnames(args)`
  - ↳ Return a list of hostname as they appear in the cluster database
  - ↳ If `args = None` all the host in the cluster are returned
  - ↳ `args` can also be a group
    - Rack0, rack1
  - ↳ Or an appliance type
    - Compute, Tile, ...

```
1: import rocks.commands
2:
3: class command(rocks.commands.HostArgumentProcessor,
4:                 rocks.commands.list.command):
5:     pass
6:
7: class Command(command):
8:     """
9:     List the membership, CPU count, physical position info and comment for
10:    a list of hosts.
11:
12:    <arg optional='1' type='string' name='host' repeat='1'>
13:        Zero, one or more host names. If no host names are supplied, info about
14:        all the known hosts is listed.
15:    </arg>
16:
17:    <example cmd='list host compute-0-0'>
18:        List info for compute-0-0.
19:    </example>
20:
21:    <example cmd='list host'>
22:        List info for all known hosts.
23:    </example>
24:    """
25:
26:    def run(self, params, args):
27:        self.beginOutput()
28:
29:        for host in self.getHostnames(args):
30:            self.db.execute("""select m.name, n.cpus,
31:                            n.rack, n.rank, n.comment from
32:                            nodes n, memberships m where
33:                            n.membership=m.id and n.name=%s""", % host)
34:            self.addOutput(host, self.db.fetchone())
35:
36:        self.endOutput(header=['host', 'membership',
37:                              'cpus', 'rack', 'rank', 'comment'])
```

# args = None

```
# rocks list host
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	COMMENT
vizagra:	Frontend	1	0	0	-----
tile-0-1:	Tile	2	0	1	-----
tile-0-0:	Tile	2	0	0	-----
tile-0-2:	Tile	2	0	2	-----
tile-0-3:	Tile	2	0	3	-----
tile-1-3:	Tile	2	1	3	-----
tile-1-2:	Tile	2	1	2	-----
tile-1-1:	Tile	2	1	1	-----
tile-1-0:	Tile	2	1	0	-----
tile-2-0:	Tile	2	2	0	-----
tile-2-1:	Tile	2	2	1	-----
tile-2-2:	Tile	2	2	2	-----
tile-2-3:	Tile	2	2	3	-----

# args = list of hosts

```
# rocks list host tile-0-0 10.255.255.253 tile-3-0.local
HOST      MEMBERSHIP CPUS RACK RANK COMMENT
tile-0-0: Tile        2     0     0     ----- 
tile-0-1: Tile        2     0     1     ----- 
tile-3-0: Tile        2     3     0     -----
```

# args = rack

```
# rocks list host rack2
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	COMMENT
tile-2-0: Tile	2	2	0		-----
tile-2-1: Tile	2	2	1		-----
tile-2-2: Tile	2	2	2		-----
tile-2-3: Tile	2	2	3		-----

# args = appliance type

```
# rocks list host tile
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	COMMENT
tile-0-0: Tile	2	0	0	0	-----
tile-0-1: Tile	2	0	1	1	-----
tile-0-2: Tile	2	0	2	2	-----
tile-0-3: Tile	2	0	3	3	-----
tile-1-0: Tile	2	1	0	0	-----
tile-1-1: Tile	2	1	1	1	-----
tile-1-2: Tile	2	1	2	2	-----
tile-1-3: Tile	2	1	3	3	-----
tile-2-0: Tile	2	2	0	0	-----78
tile-2-1: Tile	2	2	1	1	-----
tile-2-2: Tile	2	2	2	2	-----

# Any combination is fine

```
# rocks list host tile-2-0 rack1 frontend
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	COMMENT
tile-1-0: Tile	2	1	0	0	-----
tile-1-1: Tile	2	1	1	1	-----
tile-1-2: Tile	2	1	2	2	-----
tile-1-3: Tile	2	1	3	3	-----
tile-2-0: Tile	2	2	0	0	-----
vizagra: Frontend	1	0	0	0	-----

# ArgumentProcessors

Class Name	Helper Function
ApplianceArgumentProcessor	getApplianceNames
DistributionArgumentProcessor	getDistributionNames
HostArgumentProcessors	getHostnames
MembershipArgumentProcessor	getMembershipNames
NetworkArgumentProcessor	getNetworkNames
RollArgumentProcessor	getRollNames

# RollArgumentProcessor

```
1: import os
2: import stat
3: import time
4: import sys
5: import string
6: import rocks.commands
7:
8:
9: class Command(rocks.commands.RollArgumentProcessor,
10:               rocks.commands.list.command):
11:     """
12:     List the status of available rolls.
13:
14:     <arg optional='1' type='string' name='roll' repeat='1'>
15:         List of rolls. This should be the roll base name (e.g., base, hpc,
16:         kernel). If no rolls are listed, then status for all the rolls are
17:         listed.
18:     </arg>
19:
20:     <example cmd='list roll kernel'>
21:         List the status of the kernel roll
22:     </example>
23:
24:     <example cmd='list roll'>
25:         List the status of all the available rolls
26:     </example>
27:     """
28:
29:     def run(self, params, args):
30:
31:         self.beginOutput()
32:         for (roll, version) in self.getRollNames(args, params):
33:             self.db.execute("""select version, arch, enabled from
34:                             rolls where name=%s and version=%s""", %
35:                             (roll, version))
36:             for row in self.db.fetchall():
37:                 self.addOutput(roll, row)
38:
39:         self.endOutput(header=['name', 'version', 'arch', 'enabled'],
40:                       trimOwner=0)
```

# No Parameter

```
# rocks list roll
```

NAME	VERSION	ARCH	ENABLED
viz:	5.0	i386	yes
sge:	5.0	i386	yes
kernel:	5.0	i386	yes
updates:	5.1	i386	yes
java:	4.3.2	i386	yes
xen:	5.0	i386	yes
CentOS:	5.1	i386	yes
ganglia:	5.0	i386	yes
web-server:	5.0	i386	yes
base:	5.0	i386	yes

# Version Parameter

```
# rocks list roll version=4.3.2
NAME  VERSION ARCH ENABLED
java: 4.3.2    i386 yes
```

# Summary

---

- ◆ ArgumentProcessors standardize the handling of command line subjects
- ◆ Calling the helper function with an empty list returns all subject in the database
- ◆ HostArgumentProcessor knows about more than just host names
- ◆ RollArgumentProcessor can filter on versions

# fillParams

---

- ◆ Create local variables based on command parameters (key=value)
- ◆ Argument a list of (key, default) tuples
- ◆ If the parameter is not found on the command line the default value is used

```

32: generates torrent files for every file in the KIMS directory.
33: </example>
34: """
35:
36:     def maketorrent(self, filename, data):
37:         info = {}
38:         info['length'] = os.stat(filename)[stat.ST_SIZE]
39:         info['name'] = os.path.basename(filename)
40:
41:         data['info'] = info
42:         encoded = BitTorrent.bencode.bencode(data)
43:
44:         file = open('%s.torrent' % (filename), 'w')
45:         file.write(encoded)
46:         file.close()
47:
48:
49:
50:     def run(self, params, args):
51:
52:         if len(args) != 1:
53:             self.abort('must supply one file')
54:         filename = args[0]
55:
56:         (timestamp, ) = self.fillParams([('timestamp', time.time())])
57:         try:
58:             creation_date = int(timestamp)
59:         except:
60:             creation_date = int(time.time())
61:
62:         data = {}
63:
64:         #
65:         # announce string
66:         #
67:         localhost = self.db.getGlobalVar('Kickstart', 'PrivateAddress')
68:         data['announce'] = 'http://%s:7625/announce' % (localhost)
69:
70:         data['creation date'] = creation_date
71:
72:         #

```

# rocks create torrent

```

73:             basename, rack, rank = host.split('-')
74:             self.db.execute("""select m.name from
75:                 appliances a, memberships m where
76:                 a.name=%s and m.appliance=a.id""") % basename)
77:             membership, = self.db.fetchone()
78:             rack = int(rack)
79:             rank = int(rank)
80:             except: membership = None
81:             rack = None
82:             rank = None
83:
84:
85:             # fillParams with the above default values
86:
87:             (membership, numCPUs, rack, rank) = self.fillParams(
88:                 [('membership', membership),
89:                  ('cpus', 1),
90:                  ('rack', rack),
91:                  ('rank', rank)])
92:
93:             if not membership:
94:                 self.abort('membership not specified')
95:             if rack == None:
96:                 self.abort('rack not specified')
97:             if rank == None:
98:                 self.abort('rank not specified')
99:
100:            self.db.execute("""insert into nodes
101:                (site, name, membership, cpus, rack, rank)
102:                values
103:                (%s,
104:                 '%s',
105:                 (select id from memberships where name=%s),
106:                 %d,
107:                 %d,
108:                 %d)""") %
109:                 (host, membership, int(numCPUs), int(rack), int(rank)))
110:
111:

```

# rocks add host

# fillPositionalArgs

- ◆ Allows for parameters to have implied keys (just values on command line)
- ◆ This is an optimization for ease of use, not ease of software
- ◆ Argument is a list of keys
  - ↳ No default value processing, if a key is specified it is required
  - ↳ Use this only when a parameter is required
- ◆ Example:

```
# rocks set network netmask optiputer netmask=255.255.255.0
# rocks set network netmask optiputer 255.255.0.0
```

```

11: </arg>
12:
13: <arg type='string' name='netmask'>
14: Netmask that named networks should have.
15: </arg>
16:
17: <param type='string' name='netmask'>
18: Can be used in place of netmask argument.
19: </param>
20: rocks set network netmask
21: <example cmd='set network netmask optiputer 255.255.255.0'>
22: Sets the netmask for the "optiputer" network to a class-c address
23: space.
24: </example>
25:
26: <example cmd='set network netmask optiputer netmask=255.255.255.0'>
27: Same as above.
28: </example>
29:
30: <example cmd='set network netmask optiputer cavewave 255.255.0.0'>
31: Sets the netmask for the "optiputer" and "cavewave" networks to
32: a class-b address space.
33: </example>
34:
35: <related>add network</related>
36: <related>set network subnet</related>
37: """
38:
39: def run(self, params, args):
40:     (args, netmask) = self.fillPositionalArgs(('netmask',))
41:
42:     if not len(args):
43:         self.abort('must supply network')
44:     if not netmask:
45:         self.abort('must supply netmask')
46:
47:     for network in self.getNetworkNames(args):
48:         self.db.execute("""update subnets set netmask='%s' where
49:                         subnets.name='%s'""" % (netmask, network))
50:

```

```

32: Sets the MAC Address for the eth1 device on host compute-0-0.
33: </example>
34:
35: <example cmd='set host interface mac compute-0-0 iface=eth1 mac=00:0e:0c:a7:5d:ff'>
36: Same as above.
37: </example>
38:
39: <example cmd='set host interface mac compute-0-0 iface=eth1 mac=NULL'>
40: clears the mac address from the database
41: </example>
42:
43: <!-- cross refs do not exist yet
44: <related>set host interface iface</related>
45: <related>set host interface ip</related>
46: <related>set host interface gateway</related>
47: <related>set host interface module</related>
48: -->
49: <related>add host</related>
50: """
51:
52: def run(self, params, args):
53:
54:     (args, iface, mac) = self.fillPositionalArgs(('iface', 'mac'))
55:
56:     hosts = self.getHostnames(args)
57:
58:     if len(hosts) != 1:
59:         self.abort('must supply one host')
60:     if not iface:
61:         self.abort('must supply iface')
62:     if not mac:
63:         self.abort('must supply mac')
64:
65:     for host in hosts:
66:         self.db.execute("""update networks, nodes set
67:             networks.mac=NULLIF('%s', 'NULL') where
68:             nodes.name='%s' and networks.node=nodes.id and
69:             (networks.device='%s' or networks.mac='%s')"""\ %
70:             (mac, host, iface, iface))
71:
```

# rocks set host interface

# Help and Docstrings

- ◆ The command line is the documentation
  - ➲ No more out of date man pages
  - ➲ Still needs a cookbook document, but reference is part of the code
- ◆ We've been looking at this all session
- ◆ Class docstring “””text”””
- ◆ Command line has an XML format



```
# rocks list roll help  
rocks list roll [roll]...
```

Description:

---

List the status of available rolls.

Arguments:

[roll]

List of rolls. This should be the roll base name (e.g., base, hpc, kernel). If no rolls are listed, then status for all the rolls are listed.

Examples:

```
$ rocks list roll kernel
```

List the status of the kernel roll

```
$ rocks list roll
```

List the status of all the available rolls

```

1: import os
2: import stat
3: import time
4: import sys
5: import string
6: import rocks.commands
7: 

# rocks list roll help


8:
9: class Command(rocks.commands.RollArgumentProcessor,
10:               rocks.commands.list.command):
11:     """
12:     List the status of available rolls.
13:
14:     <arg optional='1' type='string' name='roll' repeat='1'>
15:         List of rolls. This should be the roll base name (e.g., base, hpc,
16:         kernel). If no rolls are listed, then status for all the rolls are
17:         listed.
18:     </arg>
19:
20:     <example cmd='list roll kernel'>
21:         List the status of the kernel roll
22:     </example>
23:
24:     <example cmd='list roll'>
25:         List the status of all the available rolls
26:     </example>
27:     """
28:
29:     def run(self, params, args):
30:
31:         self.beginOutput()
32:         for (roll, version) in self.getRollNames(args, params):
33:             self.db.execute("""select version, arch, enabled from
34:                             rolls where name=%s and version=%s"""\ %
35:                             (roll, version))
36:             for row in self.db.fetchall():
37:                 self.addOutput(roll, row)
38:             © 2009 UC Regents
39:             self.endOutput(header=['name', 'version', 'arch', 'enabled'],
40:                           trimOwner=0)
41:
```

# <arg>

## ◆ Attributes

- ➲ name (required)
- ➲ optional (default = “0”)
- ➲ type (default = “string”)
- ➲ repeat (default = “0”)

## ◆ Example:

```
<arg type='string' name='network' repeat='1'>
  One or more named networks that should have the
  defined netmask.
</arg>
```

# <param>

## ◆ Attributes

- ➲ name (required)
- ➲ optional (default = “1”)
- ➲ type (default = “string”)
- ➲ repeat (default = “0”)

## ◆ Example:

```
<param type='string' name='iface'>
```

Can be used in place of the iface argument.

```
</param>
```

# <example>

- ◆ Attributes

- ↳ cmd(required)

- ◆ Example:

```
<example cmd='set host interface mac compute-0-0  
eth1 00:0e:0c:a7:5d:ff'>
```

Sets the MAC Address for the eth1 device on host  
compute-0-0.

```
</example>
```

# <related>

---

## ◆ Example

```
<related>set host interface iface</related>
<related>set host interface ip</related>
<related>set host interface gateway</related>
<related>set host interface module</related>
```

# Help

- ◆ `rocks <verb> <object...> <subject> help`
  - ⌚ Loads the command module
  - ⌚ Parses the XML docstring
  - ⌚ Format and output help as 80 column text
- ◆ Debug syntax with `format=` parameter

# help format=raw

```
# rocks list roll help format=raw
1:
2: List the status of available rolls.
3:
4: <arg optional='1' type='string' name='roll' repeat='1'>
5: List of rolls. This should be the roll base name (e.g., base, hpc,
6: kernel). If no rolls are listed, then status for all the rolls are
7: listed.
8: </arg>
9:
10:   <example cmd='list roll kernel'>
11:     List the status of the kernel roll
12:   </example>
13:
14:   <example cmd='list roll'>
15:     List the status of all the available rolls
16:   </example>
```

# Help format=parsed

```
# rocks list roll help format=parsed
{'related': [], 'example': [(u'list roll kernel', u'\t\t\n\tList the
    status of the kernel roll\n\t'), (u'list roll', u'\n\tList the status
    of all the available rolls\n\t')], 'description': u'\n\tList the status
    of available rolls.\n\t', 'param': [], 'arg': [(u'roll',
    u'string', 1, 1), u'\n\tList of rolls. This should be the roll base
    name (e.g., base, hpc,\n\tkernel). If no rolls are listed, then status
    for all the rolls are\n\tlisted.\n\t')]}
```

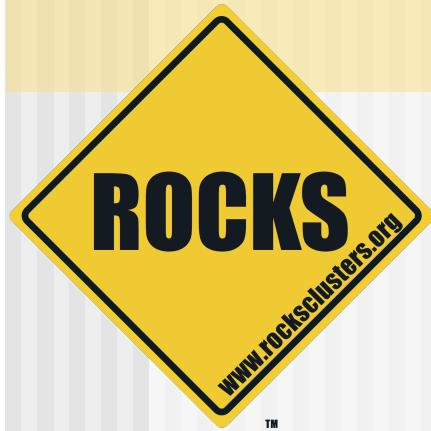
# Docbook

- ◆ Roll Usersguide Command Reference is generated automatically

```
# rocks list roll help format=docbook
<section id="rocks-list-roll" xreflabel="list roll">
<title>list roll</title>
<cmdsynopsis>
  <command>rocks list roll</command>
  <arg rep="repeat" choice="opt">roll</arg>
</cmdsynopsis>
<para>
```

List the status of available rolls.

```
</para>
<variablelist><title>arguments</title>
<varlistentry>
```



© 2009 UC Regents

102