Guidelines

- We are a small group
- Interrupt the speaker
- Ask unrelated question
- Help us keep this fluid

- We can re-tool the agenda on-the-fly
Community Development
Topics

◆ Community Status
  ✐ Why this is Important
  ✐ Open Source Licensing
  ✐ Contributors and Working Groups
  ✐ Source Code

◆ Becoming a Developer
  ✐ Graph and Rolls
  ✐ Attributes
  ✐ Command Line

◆ Avoiding becoming a Developer
Community Status
WHY?
Rocks is almost 10 years old!

- Mission accomplished
- Rocks is the *de facto* open-source clustering solution
- Great user community
  - 2000+ on mailing list
  - Amazing signal to noise ratio
- Everything from 2 nodes cluster to top 10 supercomputers

© 2010 UC Regents
Rocks is almost 10 years old!

- 90% of development is
  - NSF (and other grant) funded
  - Located at UC San Diego

- Need to diversify development
  - More ideas, passion, and focus areas
  - More secure funding
Licensing / Copyrights
http://www.rocksclusters.org/wordpress/?page_id=48

- Rocks is entirely open-source
- BSD Attribution License
  - Standard UNIX open-source
  - Very friendly for derived works
- We have not changed to the more recent non-attribution BSD license
- Copyrights are owned by University of California Regents
- 3rd party code is a mix of licenses and copyrights
  - Most of Rocks bits are 3rd party!

© 2010 UC Regents
Attribution Clause

This product includes software developed by the Rocks® Cluster Group at the San Diego Supercomputer Center at the University of California, San Diego and its contributors.
Trademark
invent@ucsd.edu

- The Rocks name and logo are registered trademarks.
- For fee licensing is available
  - Standard usage
  - Derivative usage (e.g. “ACME Rocks”)
Summary

- Rocks is open-source and free
- Use it any way you wish
- Make billions of dollars with it without even buying us a single beer

- Give us attribution
- License the name for commercial use

- These two things help keep us funded
EARLY COMMUNITY ROLLS
Sun Grid Engine

- Way back in 2004
- Rocks supported PBS
- Scalable Systems added SGE support
  - Laurence Liew, Najib Ninaba
  - 1st external developers for Rocks
  - Based in Singapore
- SGE Roll created from this
Torque Roll

- Way back in 2006
- SGE Roll was the favorite of core team
- Threatened to drop PBS Support
- The Computer Center, University of Tromsø
  - Roy Dragseth
  - 2nd major external developer
  - Based in Norway
- Continues to develop and support Torque Roll
We were “big” overseas
First major attempt for actively recruit developers

WORKING GROUPS
Working Groups

◆ Purpose
  ➢ Fill gaps from core development team
  ➢ Handle issues off the core road map
  ➢ Make Rocks a more flexible solution

◆ Success Metrics
  ➢ Number of Rolls produced
  ➢ Amount of new Documentation (Wiki, …)
Software Update WG

- Started early 2008
  - Threads on yum updates increased
  - Core team said “don’t do it”
  - Advocates said “it works for me”
  - WG was recruited to address the issue

- Best practices defined
  - Exclude lists
  - Additional docs on custom restore Rolls
Status

- Community Interest: High
- Documentation: Moderate
- Rolls Produced: None

Summary: Some real interest but needs leadership.
Began with Rocks Solaris port
  - Sun funded
  - How to Manage ZFS NAS appliances
  - Specifically Sun Thumper

Core team lead effort

Used by several groups at UCSD

Software is released
Status

- Community Interest: Low
- Documentation: Good
- Rolls Produced: Jumpstart

Summary: Excellent activity with a small UCSD audience. Needs to build a larger user base.
Rolls Working Group

- Started early 2009
  - Developing free versions of commercial Rolls
  - Organized by Stanford University
- Self-organized group of a 3 individuals
- Good initial offering of Rolls
- Struggled with mailing list support

© 2010 UC Regents
Status

- Community Interest: High
- Documentation: Average
- Rolls Produced: Good

Summary: Excellent start, needs help with user support and keeping current with Rocks releases.
Triton Working Group
http://tritonresource.sdsc.edu/

- Started 2009
  - Developing Roll for large production cluster
  - Every piece of SW on system is part of a Roll
  - Includes commercial software
- Amazing set of Rolls (20+) to be released
- Triton group is here at SDSC
- No organized presence on Rocks list

© 2010 UC Regents
Status

- Community Interest: Good
- Documentation: Good
- Rolls Produced: Excellent

- Summary: Highly productive group, but meets weekly with member(s) of Rocks core team. Phil is also their boss.
Great idea, some good traction, but not what we want

OVERALL GRADE: C-
What can UCSD do better?

- WG phone/video conference
  - WG to Core team
  - WG All hands
- Need to communicate roadmaps between WGs and Core team
  - Ease release tracking
  - No surprises (e.g. Rocks Command Line)
- Where should support issues go?
  - Main list
  - A new WG list
Ideas?

- Docs in dev process
  - Devel guide out of date
  - Mine mailing list for solution
- Developer Cloning Process
  - Jumpstart guide to development
- RESOLVED tag on mailing list
- Bug/Issue searchable database
  - RH is a good example of this
- IRC
- AIM Address Book (non-indexed)
What can you do better

- Tell us what you want
  - Complain
  - A lot
  - But, nicely

- Ask for help to start a new working group
- Join an existing working group
- We are starting this **today**
SOURCE CODE
Version Control

◆ Core team uses CVS
  ✅ That’s what we started with
  ✅ We aren’t changing anytime soon

◆ CVS access available to very few people
  ✅ Too risky
  ✅ Access control is a pain
  ✅ Release management difficult

◆ We use Mecurial for all non-core development

◆ Mecurial synced to CVS every 10 minutes!
Workflow
Example

$ hg clone http://fyp.rocksclusters.org/hg/rocks-5.3
destination directory: rocks-5.3
real URL is http://fyp.rocksclusters.org/hg/
rocks-5.3/
requesting all changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 2815 changes to 2815 files
2815 files updated, 0 files merged, 0 files removed, 0 files unresolved
Issues

- Mecurial is slow
- Transaction based
  - Any aborted operation rolls back
  - Do not stop the clone
- Patch sets can be tedious
Advantages - Freedom

- Publish your own repository
- No need to even commit back to core
- Commit broke code and only hurt yourself

- Core Rocks remains stable
- HG clones innovate
Notes

- For code older than 5.0
  - rocks-2.3 to rocks-4.3
- We are not tied to this workflow
- We are not tied to HG
- Other workflow suggestions are solicited
Becoming a Developer

Resume @ 11:20
The Rocks engine

GRAPH AND ROLLS
Rocks Configuration Graph
The XML Graph Includes

◆ Nodes
  ➤ Single purpose modules
  ➤ Kickstart file snippets (XML tags map to kickstart commands)
  ➤ Approximately 200 node files in Rocks

◆ Graph
  ➤ Defines interconnections for nodes
  ➤ Think OOP or dependencies (class, #include)
  ➤ A single default graph file in Rocks

◆ Macros
  ➤ SQL Database holds site and node specific state
  ➤ Node files may contain &state; entities (attributes)
Composition

- Aggregate Functionality

- scripting IsA
  - perl-development
  - python-development
  - tcl-development
Traverse by Attributes

- if x11 == TRUE
  - client IsA x11

- if rsh == FALSE
  - client IsNotA rsh

- Most important slide in this session

- RCL allows you to control the graph
Think of this as Cluster DNA
ROLL FUNDAMENTALS
Cluster Software Stack

- Parallel Code / WebFarm / Grid / Computer Lab
- Message Passing / Communication Layer
- Job Scheduling and Launching
- Cluster Software Management
- Cluster State Management / Monitoring
- Linux Environment
- HPC Device Drivers (e.g., Interconnect and Storage)
- Linux Kernel

© 2009 UC Regents
Rolls Break Apart Rocks


© 2009 UC Regents
Our Graph Has Colors
Rolls are sub-graphs

- A graph makes it easy to ‘splice’ in new nodes
- Each Roll contains its own nodes and splices them into the system graph file
STARTING FROM THE EMPTY SET
{ base }
\{ \text{base, hpc} \}
{ base, hpc, kernel, sge }
Simplified Example

{base, hpc, sge, bio}
Two different Clusters

MPI Cluster::{base, hpc, kernel, sge}

Protein Databank::{base, hpc, kernel, pdb}
ATTRIBUTES
Attributes

- Attributes can be set at 4 levels:
  - Globally
    - ‘rocks set attr’
  - By appliance type
    - ‘rocks set appliance attr’
  - By OS (linux or sunos)
    - ‘rocks set os attr’
  - By host
    - ‘rocks set host attr’
Attributes

- Example, set the public IP address of a remote frontend that is used during a ‘central’ installation:

```
# rocks set host attr vi-1.rocksclusters.org \ Kickstart_PublicAddress 137.110.119.118
```
# rocks list host attr tile-0-0

<table>
<thead>
<tr>
<th>HOST</th>
<th>ATTR</th>
<th>VALUE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-0-0:</td>
<td>Info_CertificateCountry</td>
<td>US</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Info_CertificateLocality</td>
<td>San Diego</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Info_CertificateOrganization</td>
<td>CalIT2</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_DistroDir</td>
<td>/export/rocks</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PrivateAddress</td>
<td>10.1.1.1</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PrivateBroadcast</td>
<td>10.1.255.255</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PrivateDNSDomain</td>
<td>local</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PrivateDNSServers</td>
<td>10.1.1.1</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PrivateGateway</td>
<td>10.1.1.1</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicDNSServers</td>
<td>132.239.0.252</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicGateway</td>
<td>137.110.119.1</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicHostname</td>
<td>vizagra.rocksclusters.org</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicKickstartHost</td>
<td>central.rocksclusters.org</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicNTPHost</td>
<td>pool.ntp.org</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicNetmask</td>
<td>255.255.255.0</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicNetmaskCIDR</td>
<td>24</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_PublicNetwork</td>
<td>137.110.119.0</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Kickstart_Timezone</td>
<td>America/Los_Angeles</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Server_Partitioning</td>
<td>force-default-root-disk-only</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>arch</td>
<td>x86_64</td>
<td>H</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>hostname</td>
<td>tile-0-0</td>
<td>I</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>rack</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>rank</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>rocks_version</td>
<td>5.2</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>HideBezels</td>
<td>false</td>
<td>G</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>HttpConf</td>
<td>/etc/httpd/conf</td>
<td>O</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>HttpConfigDirExt</td>
<td>/etc/httpd/conf.d</td>
<td>O</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>HttpRoot</td>
<td>/var/www/html</td>
<td>O</td>
</tr>
</tbody>
</table>
Edge Conditionals

- Use attributes to conditionally traverse edges of the configuration graph

```xml
<edge from="client" cond="rsh">
  <to>rsh</to>
</edge>
```

- If ‘rsh’ evaluates to ‘true’, then the edge from ‘client’ to ‘rsh’ will be traversed
  - Default value is ‘false’
Edge Conditionals

- To set a conditional attribute:

  # rocks set attr rsh true

- Edge conditionals are attributes
- Can also be set at 4 levels:
  - Globally
  - By appliance type
  - By OS (linux or sunos)
  - By host
COMMAND LINE
Evil Commands

[--list-rcfiles] [--list-project-info] [--verbose] [--dump] [--del] [--list]
[--password password] [--db database] [--user host]
[--if interface (default: eth1)] [--mac mac address]
[--module linux driver module name] [--ip ip address]
[--netmask netmask (default /24)] [--gateway ip address of gateway]
[--name hostname on new interface] [--site client ip] node

[-d dirname] [-g path] [-l lang] [-r release] [--help] [--list-rcfiles]
[--graph-draw-order] [--graph-draw-edges] [--graph-draw-key] [--graph-draw-all]
[--notorrent] [--rcfile arg] [--host host] [--password password]
[--db database] [--user host] [--arch architecture] [--comps path]
[--dist dirname] [--graph-draw-size arg] [--graph-draw-format arg]
[--mirror-dir dirname] [--mirror-host hostname] [--root dirname]
[--cdrom /mnt/cdrom] [--with-roll rollname-rollback]
[--path single path item] command

Available commands:
dist dvd makecontrib makesitenodes copycd usb copyroll cdrom paths graph dist2mirror

© 2009 UC Regents
Command Line as API

- Lack of consistency in Rocks commands
  - `add-extra-nic` (15 flags)
  - `411put`
  - `rocks-dist`
  - `dbreport` (~ a dozen reports)

- Not 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?
Do Over

- Consistent
  - Interface
  - Argument parsing
  - Usage / Help

- Extensible
  - Easy to add commands (3rd party rolls)
  - Easy to modify commands

- Easy to guess the right command
- Purge all –flags from Rocks
- Hide the SQL database (and underlying schema)
- Inspired by Trac

© 2009 UC Regents
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

© 2009 UC Regents
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 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

import rocks.commands

class Command(rocks.commands.DistributionArgumentProcessor,
              rocks.commands.add.command):
    
    Add a distribution specification to the database.

    <arg type='string' name='distribution'>
    Name of the new distribution.
    </arg>

    <example cmd='add distribution rocks-dist'>
    Adds the distribution named "rocks-dist" into the database.
    </example>

    def run(self, params, args):
        
        if len(args) != 1:
            self.abort('must supply one distribution')
        dist = args[0]

        if dist in self.getDistributionNames():
            self.abort('distribution "%s" exists' % dist)

        self.db.execute("insert into distributions (name) values ('%s')" % dist)
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

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

© 2008 UC Regents
list

- Reports information in human readable format
- No side-effects on the database
- Examples:
  - Hosts
  - Appliances
  - Rolls
# rocks list host

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vizagra:</td>
<td>Frontend</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-------</td>
</tr>
<tr>
<td>tile-0-1:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>-------</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-------</td>
</tr>
<tr>
<td>tile-0-2:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>-------</td>
</tr>
<tr>
<td>tile-0-3:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>-------</td>
</tr>
<tr>
<td>tile-1-3:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>-------</td>
</tr>
<tr>
<td>tile-1-2:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-------</td>
</tr>
<tr>
<td>tile-1-1:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>-------</td>
</tr>
<tr>
<td>tile-1-0:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-------</td>
</tr>
<tr>
<td>tile-2-0:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>-------</td>
</tr>
<tr>
<td>tile-2-1:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-------</td>
</tr>
<tr>
<td>tile-2-2:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-------</td>
</tr>
<tr>
<td>tile-2-3:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>-------</td>
</tr>
<tr>
<td>tile-3-0:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>-------</td>
</tr>
<tr>
<td>tile-3-1:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>-------</td>
</tr>
<tr>
<td>tile-3-2:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-------</td>
</tr>
<tr>
<td>tile-3-3:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>-------</td>
</tr>
<tr>
<td>tile-4-0:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>-------</td>
</tr>
<tr>
<td>tile-4-1:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>-------</td>
</tr>
<tr>
<td>tile-4-2:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>-------</td>
</tr>
<tr>
<td>tile-4-3:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>-------</td>
</tr>
</tbody>
</table>
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
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”
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)
```python
import rocks.commands

class Command(rocks.commands.sync.command):
    """
    Update all user-related files (e.g., /etc/passwd, /etc/shadow, etc.)
on all known hosts. Also, restart autoofs on all known hosts.
    """

    <example cmd='sync users'>
    Send all user info to all known hosts.
    </example>

    def run(self, params, args):
        self.runPlugins()
```

from rocks import rocks.commands

class Plugin(rocks.commands.Plugin):
    """Force a 411 update and re-load autofs on all nodes"""

    def provides(self):
        return '411'

    def requires(self):
        return ['fixnewusers']

    def run(self, args):
        
        # force the rebuild of all files under 411's control
        
        for line in os.popen('make -C /var/411 force').readlines():
            self.owner.addText(line)

        
        # restart autofs on all known hosts
        
        cmd = '/opt/rocks/bin/tentakel "service autofs reload"'
        for line in os.popen(cmd).readlines():
            self.owner.addText(line)
auto.home / passwd plugin

```python
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:     def provides(self):
8:         return 'fixnewusers'
9:     def run(self, args):
10:         # scan the password file for any '/export/home' entries
11:         # this is the default entry as setup by useradd
12:         new_users = []
13:         default_dir = '/export/home/
14:         file = open('/etc/passwd', 'r')
15:         for line in file.readlines():
16:             l = string.split(line[:-1], ':')
17:             if len(l) < 6:
18:                 continue
19:             username = l[0]
20:             homedir = l[5]
21:             if homedir[len(default_dir)] == default_dir:
22:                 new_users.append(username)
23:         file.close()
24:         hostname = '\%s' % 
25:         (self.db.getGlobalVar("Kickstart", 'PrivateHostname'),
26:          self.db.getGlobalVar("Kickstart", 'PrivateDNSDomain'))
27:         for user in new_users:
28:             # for each new user, change their default directory to
29:             # /home/username
30:             cmd = 'lsusermod -d %s %s' % (os.path.join('/home', user), user)
31:             os.system(cmd)
32:             for line in os.popen(cmd).readlines():
33:                 if line[0] == 'c':
34:                     self.owner.addText(line)
35:     
36: ```
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, …
import rocks.commands

class command(rocks.commands.HostArgumentProcessor,
    rocks.commands.list.command):
    pass

class Command(command):
    ""
    List the membership, CPU count, physical position info and comment for
    a list of hosts.
    ""

    <arg optional='1' type='string' name='host' repeat='1'>
    Zero, one or more host names. If no host names are supplied, info about
    all the known hosts is listed.
    </arg>

    <example cmd='list host compute-0-0'>
    List info for compute-0-0.
    </example>

    <example cmd='list host'>
    List info for all known hosts.
    </example>
    ""

def run(self, params, args):
    self.beginOutput()
    for host in self.getHostnames(args):
        self.db.execute("""\n        select m.name, n cpus, 
        n.rack, n.rank, n.comment from 
        nodes n, memberships m where 
        n.membership=m.id and n.name="""%s"""" % host) 
        self.addOutput(host, self.db.fetchone())
    self.endOutput(header=['host', 'membership',
        'cpus', 'rack', 'rank', 'comment'])
args = None

# rocks list host

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vizagra:</td>
<td>Frontend</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-1:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-2:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-3:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-3:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-2:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-1:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-0:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-0:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-1:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-2:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-3:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>--------</td>
</tr>
</tbody>
</table>
args = list of hosts

```
# rocks list host tile-0-0 10.255.255.253 tile-3-0.local

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-0-0:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-1:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-3-0:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>--------</td>
</tr>
</tbody>
</table>
```

© 2008 UC Regents
args = rack

# rocks list host rack2

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-2-0: Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td></td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-1: Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-2: Tile</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-3: Tile</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td>--------</td>
</tr>
</tbody>
</table>
\textbf{args} = \textbf{appliance type}

```plaintext
# rocks list host tile

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-0-0: Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>----------</td>
</tr>
<tr>
<td>tile-0-1: Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>----------</td>
</tr>
<tr>
<td>tile-0-2: Tile</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>----------</td>
</tr>
<tr>
<td>tile-0-3: Tile</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>----------</td>
</tr>
<tr>
<td>tile-1-0: Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>----------</td>
</tr>
<tr>
<td>tile-1-1: Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>----------</td>
</tr>
<tr>
<td>tile-1-2: Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>----------</td>
</tr>
<tr>
<td>tile-1-3: Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>----------</td>
</tr>
<tr>
<td>tile-2-0: Tile</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>----------</td>
</tr>
<tr>
<td>tile-2-1: Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>----------</td>
</tr>
</tbody>
</table>
```

© 2008 UC Regents
Any combination is fine

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

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-1-0:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-1:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-2:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-3:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-0:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>vizagra:</td>
<td>Frontend</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
</tbody>
</table>
```

© 2008 UC Regents
# ArgumentProcessors

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Helper Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplianceArgumentProcessor</td>
<td>getApplianceNames</td>
</tr>
<tr>
<td>DistributionArgumentProcessor</td>
<td>getDistributionNames</td>
</tr>
<tr>
<td>HostArgumentProcessors</td>
<td>getHostnames</td>
</tr>
<tr>
<td>MembershipArgumentProcessor</td>
<td>getMembershipNames</td>
</tr>
<tr>
<td>NetworkArgumentProcessor</td>
<td>getNetworkNames</td>
</tr>
<tr>
<td>RollArgumentProcessor</td>
<td>getRollNames</td>
</tr>
</tbody>
</table>
import os
import stat
import time
import sys
import string
import rocks.commands

class Command(rocks.commands.RollbackArgumentProcessor,
               rocks.commands.list.command):
    ""
    List the status of available rolls.
    ""
    def run(self, params, args):
        self.beginOutput()
        for (roll, version) in self.getRollNames(args, params):
            self.db.execute("select version, arch, enabled from rolls where name='%s' and version='%s'" % (roll, version))
            for row in self.db.fetchall():
                self.addOutput(roll, row)
        self.endOutput(header=['name', 'version', 'arch', 'enabled'],
                       trimOwner=0)
# rocks list roll

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>ARCH</th>
<th>ENABLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>viz:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>sge:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>kernel:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>updates:</td>
<td>5.1</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>java:</td>
<td>4.3.2</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>xen:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>CentOS:</td>
<td>5.1</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>ganglia:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>web-server:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>base:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
</tbody>
</table>

© 2008 UC Regents
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
rocks create torrent

def maketorrent(self, filename, data):
    info = {}
    info['length'] = os.stat(filename)[stat.ST_SIZE]
    info['name'] = os.path.basename(filename)
    data['info'] = info

    encoder = BitTorrent.encoding.bencode

    file = open('%s.torrent' % (filename), 'w')
    file.write(encoded)
    file.close()

def run(self, params, args):
    if len(args) != 1:
        self.abort('must supply one file')
    filename = args[0]
    
    (timestamp, ) = self.fillParams([(timestamp, time.time())])
    try:
        creation_date = int(timestamp)
    except:
        creation_date = int(time.time())

    data = {}

    #
    # announce string
    #
    localhost = self.db.getGlobalVar('Kickstart', 'PrivateAddress')
    data['announce'] = 'http://%s:7625/announce' % (localhost)

    data['creation date'] = creation_date
rocks add host

```python
rocks add host

73:     basename, rack, rank = host.split('-')
74:     self.db.execute('''
75:         select m.name from
76:             appliances a, memberships m where
77:             a.name='%%%s' and m.appliance=a.id''' % basename)
78:     membership, = self.db.fetchone()
79:     rack = int(rack)
80:     rank = int(rank)
81:     membership = None
82:     rack = None
83:     rank = None
84:     # fillParams with the above default values
85:     (membership, numCPUs, rack, rank) = self.fillParams(
86:         [('membership', membership),
87:          ('cpus', 1),
88:          ('rack', rack),
89:          ('rank', rank)])
90:     if not membership:
91:         self.abort('membership not specified')
92:     if rack == None:
93:         self.abort('rack not specified')
94:     if rank == None:
95:         self.abort('rank not specified')
96:     self.db.execute('''
97:         insert into nodes
98:         (site, name, membership, cpus, rack, rank)
99:         values
100:             (0,
101:             '%%%s',
102:             (select id from memberships where name='%%%s'),
103:             '%%%d',
104:             '%%%d',
105:             '%%%d')''' %
106:             (host, membership, int(numCPUs), int(rack), int(rank)))
107: ```
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
rocks set network netmask

Netmask that named networks should have.

Can be used in place of netmask argument.

Sets the netmask for the "optiputer" network to a class-c address space.

Same as above.

Sets the netmask for the "optiputer" and "cavewave" networks to a class-b address space.

add network

set network subnet

---

def run(self, params, args):
    (args, netmask) = self.fillPositionalArgs(('netmask',))
    if not len(args):
        self.abort('must supply network')
    if not netmask:
        self.abort('must supply netmask')
    for network in self.getNetworkNames(args):
        self.db.execute("update subnets set netmask='%s' where subnets.name='%s'", (netmask, network))
rocks set host interface

def run(self, params, args):
    (args, iface, mac) = self.fillPositionalArgs(('iface', 'mac'))
    hosts = self.getHostnames(args)
    if len(hosts) != 1:
        self.abort('must supply one host')
    if not iface:
        self.abort('must supply iface')
    if not mac:
        self.abort('must supply mac')
    for host in hosts:
        self.db.execute("update networks, nodes set
            networks.mac=NULLIF('%s','NULL') where
            nodes.name='%s' and networks.node=nodes.id and
            (networks.device='%s' or networks.mac='%s')
            % (mac, host, iface, iface))
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 [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
import os
import stat
import time
import sys
import string
import rocks.commands

class Command(rocks.commands.RollbackArgumentProcessor,
              rocks.commands.list.command):
    """
    List the status of available rolls.
    """
    
    <arg optional='1' type='string' name='roll' repeat='1'>
    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.
    </arg>

    <example cmd='list roll kernel'>
    List the status of the kernel roll
    </example>

    <example cmd='list roll'>
    List the status of all the available rolls
    </example>

    """

def run(self, params, args):
    
    self.beginOutput()
    for (roll, version) in self.getRollNames(args, params):
        self.db.execute("""select version, arch, enabled from rolls where name='%s' and version='%s'"""" % (roll, version))
        for row in self.db.fetchall():
            self.addOutput(roll, row)
    self.endOutput(header=['name', 'version', 'arch', 'enabled'],
                   trimOwner=0)
<arg>

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

- **Example:**
  ```xml
  <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>
  ```
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, kernel). If no rolls are listed, then status for all the rolls are listed.
6: </arg>
7: <example cmd='list roll kernel'>
8: List the status of the kernel roll
9: </example>
10:<example cmd='list roll'>
11:List the status of all the available rolls
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'	List the status of the kernel roll
'), (u'list roll', u'	List the status of all the available rolls
')], 'description': u'	List the status of available rolls.

', 'param': [], 'arg': [(u'roll', u'string', 1, 1), u'
List of rolls. This should be the roll base name (e.g., base, hpc). If no rolls are listed, then status for all the rolls are listed.

')}
Roll Usersguide Command Reference is generated automatically

# rocks list roll help format=docbook

<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>

© 2008 UC Regents
AVOIDING BECOMING A DEVELOPER
Philosophy

- All software is installed on the local disk
- Does not require NFS or non-scalable diskless technologies
- Use the native OS packager for everything
  - Linux = rpm
  - Solaris = pkg
Violate the Rules

- You just need a few packages added and cannot find or build packages
- You want this only on your cluster and not on several clusters
- You still want to avoid NFS and benefit from Rocks management
Get a Directory Tree

- Build your software from source and install on the frontend
  - configure
  - make
  - install

- Or, just untar a binary bundle
CREATE PACKAGE

rocks create package
  <path>
  <package-name>

rocks create package
  /opt/mx
mx
# rpm -qip mx-1.0-1.x86_64.rpm
Name        : mx                           Relocations: (not relocatable)
Version     : 1.0                               Vendor: Rocks Clusters
Release     : 1                             Build Date: Tue 12 May 2009 04:40:00 PM
            PDT                           Build Host: vizagra.rocksclusters.org
Install Date: (not installed)               Source RPM: mx-1.0-1.src.rpm
Group       : System Environment/Base       License: University of California
Size        : 17588899
Signature   : (none)
Summary     : A collection of Python software tools.
Description :
The mx extensions for Python are a collection of Python software tools
which enhance Python's usability in many areas.
ADDING YOUR PACKAGE TO COMPUTE NODES
Step 1: Contribute the RPM

- Your distribution looks for packages from Rolls and in a contrib area
- Copy your RPMS into contrib

```
cp mx-1.0-1.x86_64.rpm /export/rocks/install/contrib/5.2/x86_64/RPMS
```
Step 2: Extend XML

cd /export/rocks/install/site-profiles/5.2/nodes/

cp skeleton.xml extend-compute.xml

vi extend-compute.xml
Add Package Tag

**original**

```xml
<kickstart>
  <description>
  Skeleton XML Node
  </description>
  <changelog>
  </changelog>
  <!--
  <package></package>
  -->
  <post>
  </post>
</kickstart>
```

**modified (with mx)**

```xml
<kickstart>
  <description>
  Skeleton XML Node
  </description>
  <changelog>
  </changelog>
  <package>mx</package>
  <post>
  </post>
</kickstart>
```
Step 3: Rebuild Distribution

- RPM package is already contributes
- XML node file is already extended
- Now we need to rebuild the dist
- Must be done in /export/rocks/install
CREATE DISTRO

cd /export/rocks/install

rocks create distro
Step 4: Re-install
(repeated material 3 slides)

- PXE Boot
  - Network Boot is first in BIOS boot order
  - Set Rocks Boot action to install
  - Reboot the host

- Otherwise use old rocks commands or just hard power cycle the host.
SET HOST BOOT

rocks set host boot

<host>

action=<boot-action>

rocks set host boot

compute-0-0

action=install
RUN HOST

rocks run host

<host>

<command>

rocks run host

compute-0-0

/sbin/init 6
## Should I Build a Roll?

<table>
<thead>
<tr>
<th>contrib &amp; site-profiles</th>
<th>Roll</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast and Easy</td>
<td>Takes about 1 day</td>
</tr>
<tr>
<td>Admin Friendly</td>
<td>Developer Friendly</td>
</tr>
<tr>
<td>Difficult to share</td>
<td>Easy to share (.iso)</td>
</tr>
<tr>
<td>Difficult to backup/restore</td>
<td>Easy to backup/restore</td>
</tr>
<tr>
<td>Frontend is your development host</td>
<td>Frontend is your development host</td>
</tr>
</tbody>
</table>
Break Time