RED HAT
SUMMIT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

June 11-14, 2013
Boston, MA

redhat.

Introducing a New Linux Management API: OpenLMI

Russell Doty & Stephen Gallagher
Red Hat
June 13, 2013

Linux System Management Today

Is this you?

Powered by Google

Storage

- Done with local commands:
  - parted
  - pvcreate, lvcreate, vgcreate
  - mdraid
  - vgextend, resize2fs

Networking

- Edited /etc/sysconfig/networking scripts

Notice anything?

- Both methods require a local shell
- The methods had an entirely unrelated user experience

The list goes on...

- System services
- Installing software and patches
- Firewall
- Performance tuning
- ...

Training Costs

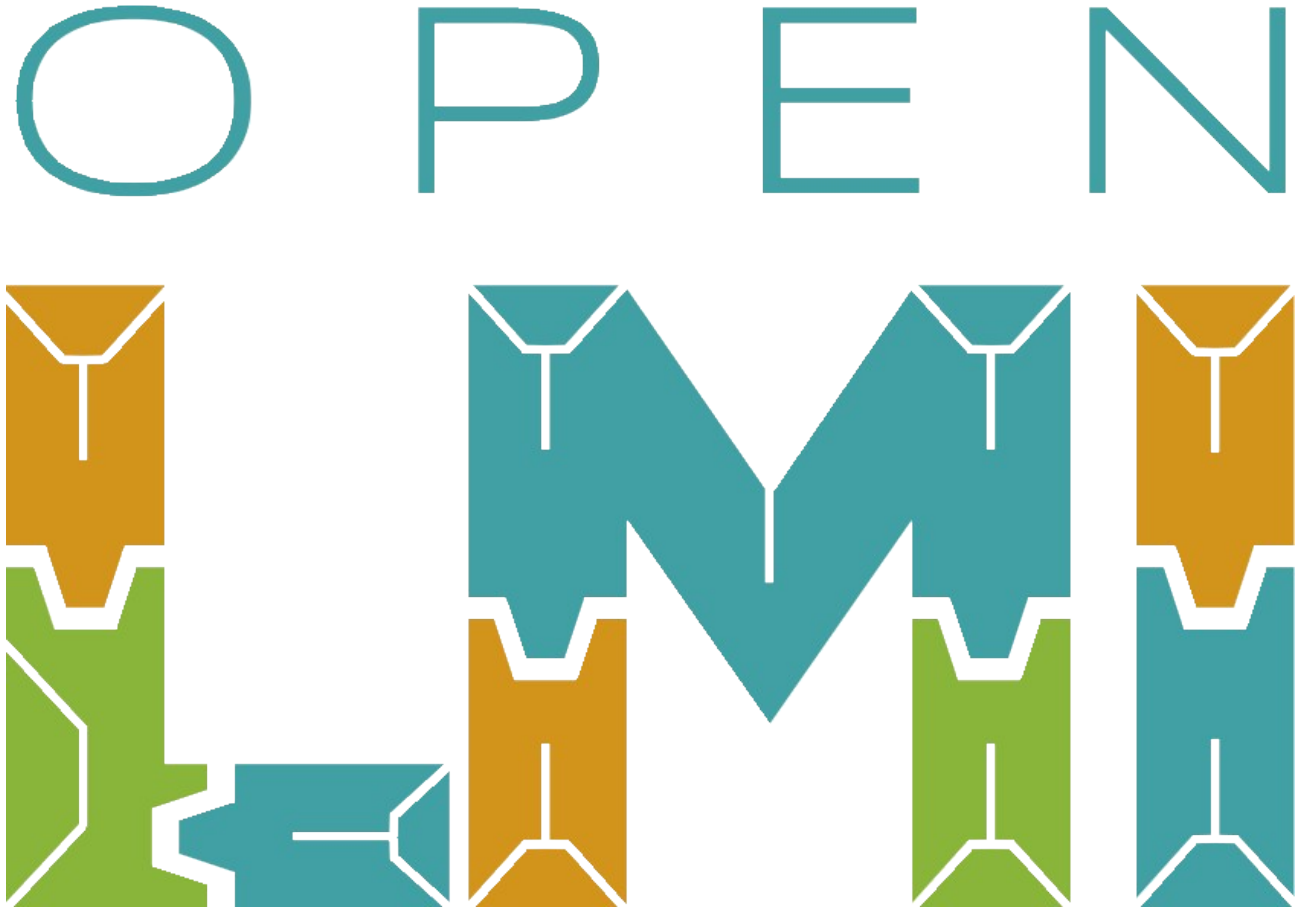
- There is a completely isolated path to learning administration of either approach
- Learning either task does not help you learn the other

Research

- "*The difficulty of managing these components is directly impacting our customers' ability to consume more Linux*" -- 2011 Red Hat Customer Survey Results (paraphrased)

Big Problem.

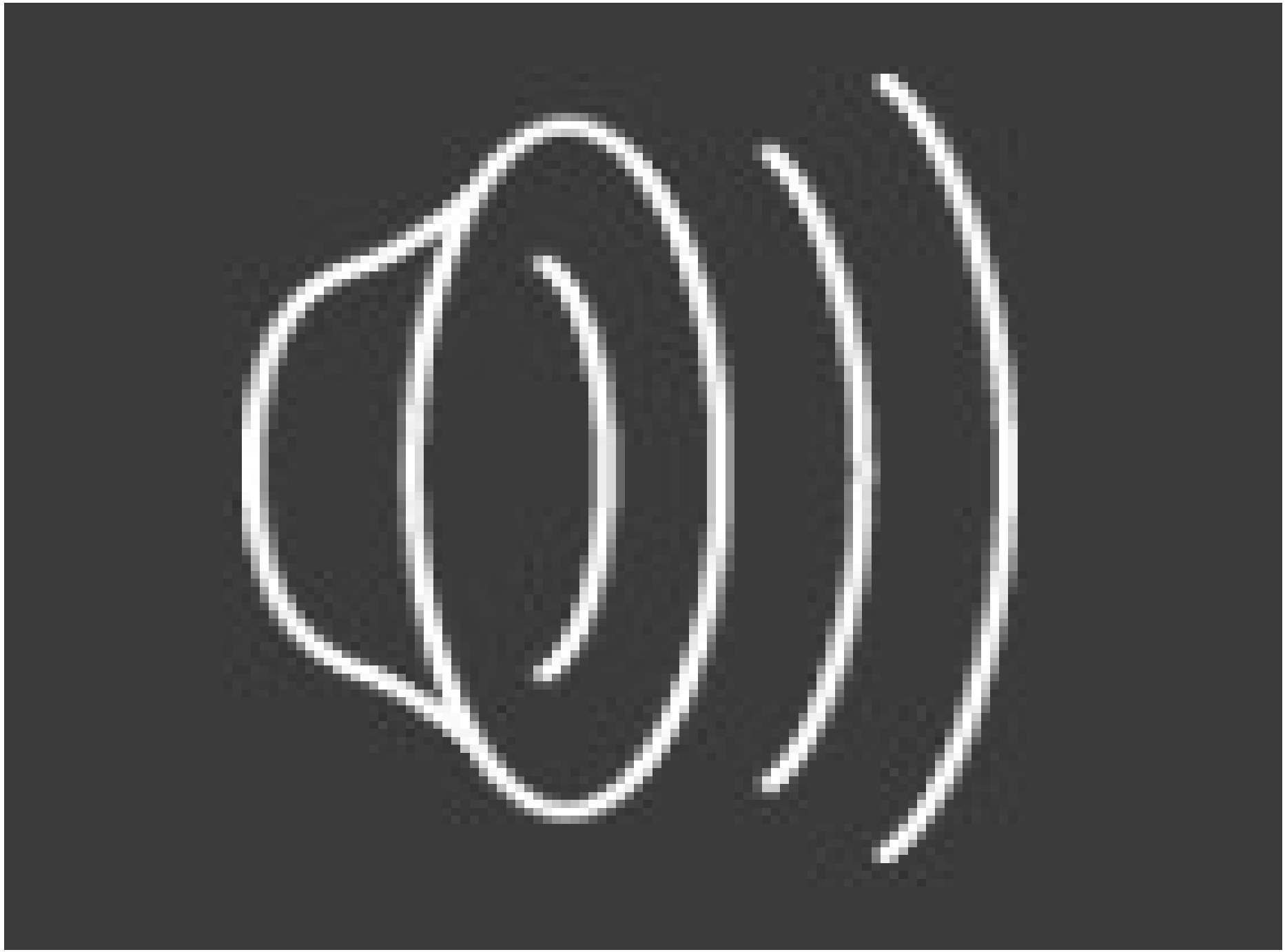
What are you going to do about it?


OPEN

Demo

- Create a 3-disk RAID array on a remote system
- Demo script:

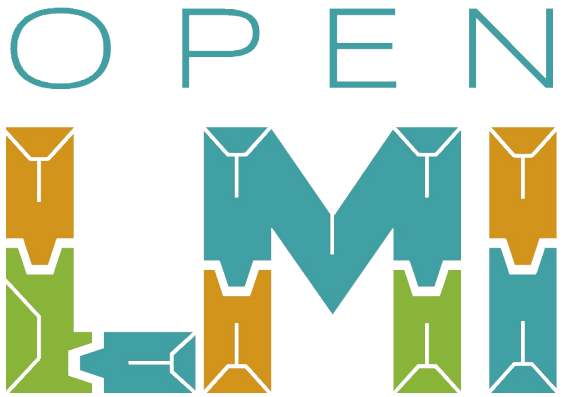
  `http://tinyurl.com/openlmi-demo`



```
# Identify “primordial” drives
drives = ns.wql("SELECT * FROM CIM_StorageExtent "
                "WHERE Primordial=True")

# Find partitions on the drives
dependent =
  extent.associators(AssocClass="CIM_BasedOn",
                     ResultRole="Dependent")

# Check for filesystem
filesystems = extent.associators(
                AssocClass="LMI_ResidesOnExtent",
                ResultClass="LMI_LocalFileSystem")

# Check for swap partition
swaps = extent.associators(
          AssocClass="LMI_ResidesOnExtent",
          ResultClass="LMI_DataFormat")
```

```
# Prep the first three drives
for drive in avail_drives:
  physical_device =
    ns.LMI_StorageExtent.first_instance(
      Key="Name", Value=drive)

  # Create a new GPT partition table on this disk
  job = LMISyncJob(ns,
          partitioning_service.SetPartitionStyle(
            Extent=physical_device,
            PartitionStyle=gpt_caps))
  job.process()

  # Create a single partition covering the whole
  # disk
  job = LMISyncJob(ns,
                   partitioning_service.
                     LMI_CreateOrModifyPartition(
                       extent=physical_device))
  job.process()
```

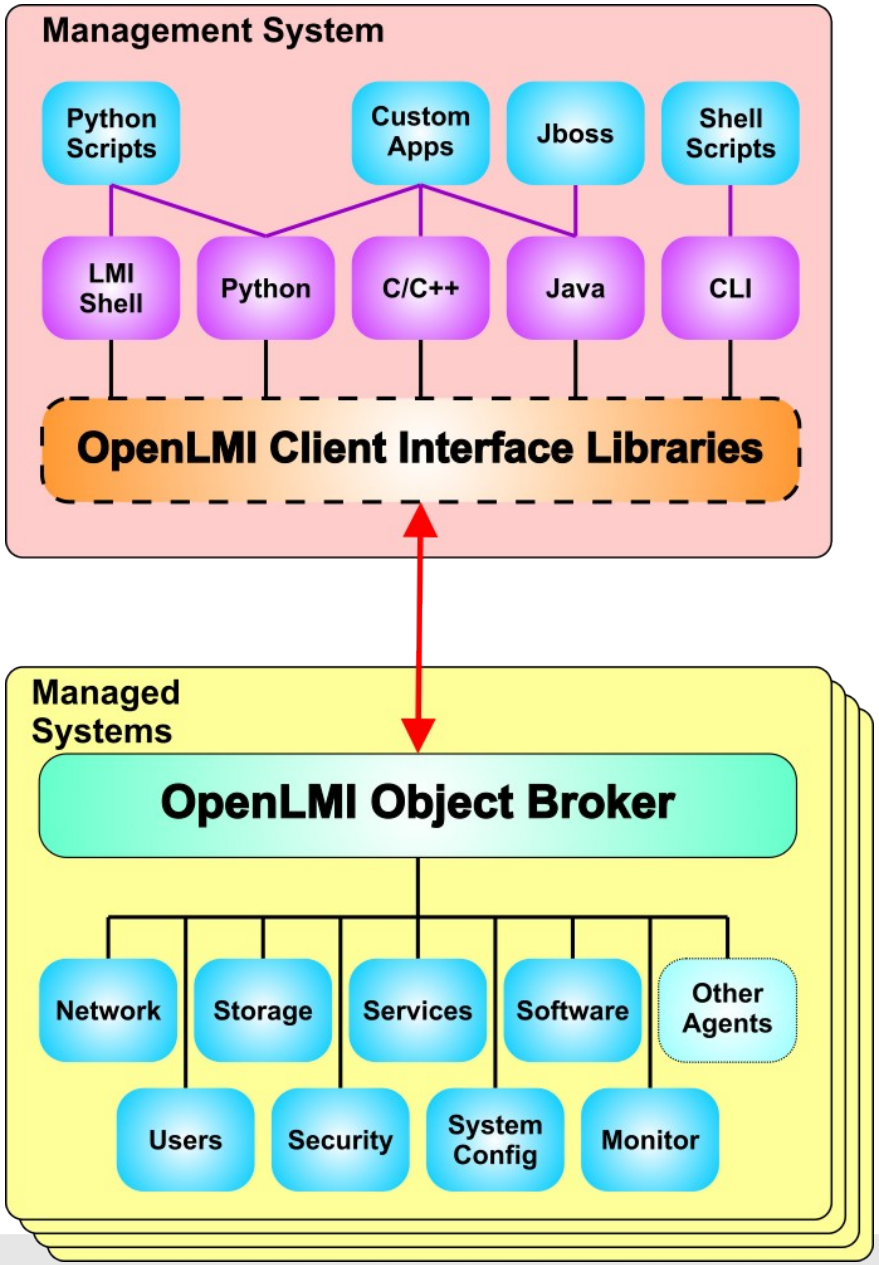
```
# Create the RAID set
job = LMISyncJob(ns,
        storage_service.CreateOrModifyMDRAID(
          ElementName = "myRAID",
          InExtents = [vdb1.path, vdc1.path,
                       vdd1.path],
          Level=5))
job.process()

# Create the EXT4 filesystem
raid = ns.LMI_StorageExtent.first_instance(
         Key="Name",
         Value="/dev/md/myRAID")
job = LMISyncJob(ns,
       filesystem_service.LMI_CreateFileSystem(
          FileSystemType = 32769,
          InExtents= [raid.path]))
job.process()
```



A platform for manageability

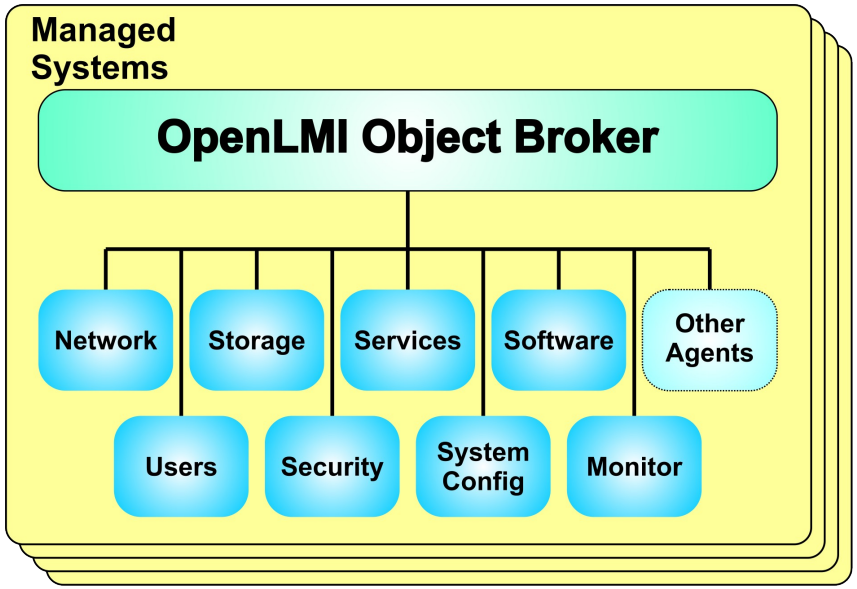
- **Low level functions to remotely *configure* and *manage* bare metal production servers (and virtual machine guests)**
- Monitoring infrastructure
- Standards based
- Open and extensible
- Open, upstream project: *www.openlmi.org*
- Delivered as part of RHEL; maintained and supported by Red Hat in RHEL

OpenLMI System Manageability Infrastructure



- Local agents installed on managed systems
- Agents and Object Broker supplied as part of OS
  - Support for future versions of RHEL
  - subsets for current versions of RHEL
- Remote API
  - Can also be used locally
- Agents and tools can be developed by Red Hat, 3rd parties, customers

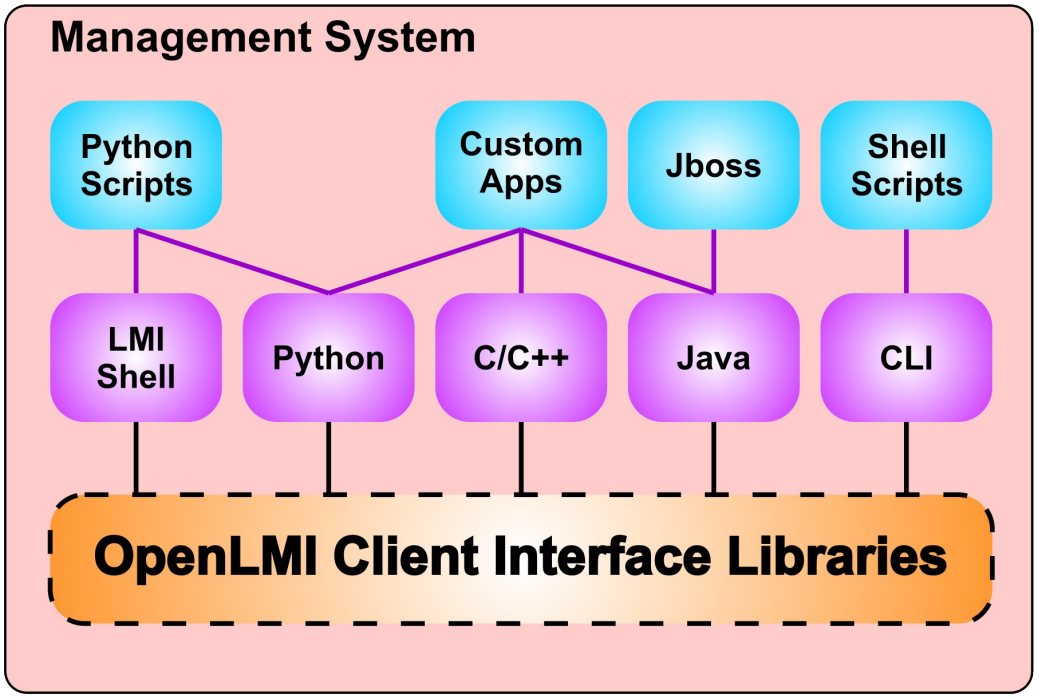
Agents



- Agents are Functional Modules
  - Get/Put attributes
  - Methods & relations
- Standard Interfaces
- Introspection
- Agents do all the work
- Toolchain for developing Agents
  - UML schema compile to produce skeleton
- Agents can be written in C/C++ or Python
- Agents can be call/response or asynchronous event driven

OpenLMI Interfaces

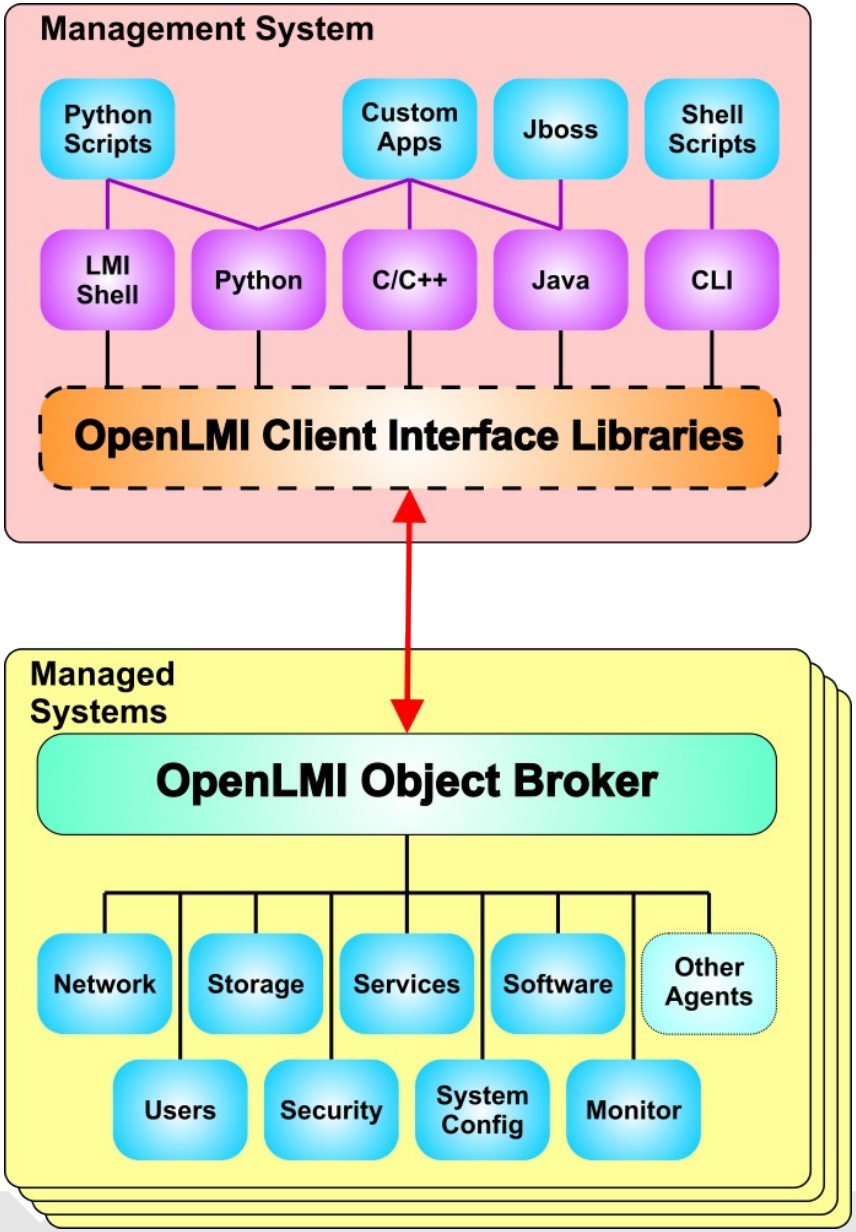
- LMI Shell
  - Enhanced CLI & scripting environment
  - Admin Friendly
- Python API
  - Use from Python modules
  - Good interface for OpenLMI Apps
- C/C++ API
  - Powerful interface for writing Apps or integrating with existing Apps



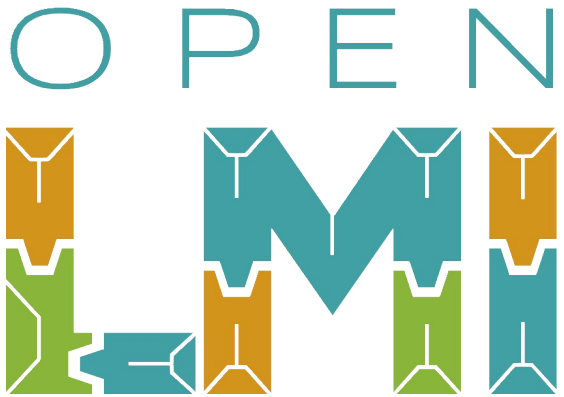
- Java
  - Write Java Apps
  - Easy interface with JBoss
- CLI

Can be used directly or from shell script

OpenLMI in Fedora 19



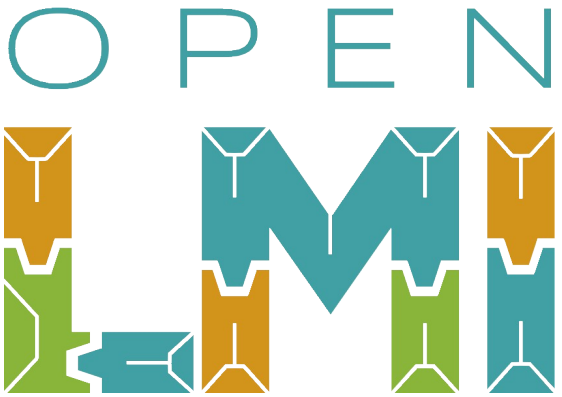
- **Implementation:**
  - DMTF/CIM technology stack
  - Https transport (no general Web server)
- **Included Agents:**
  - Storage
  - Network
  - System Services
  - Power Management
  - Local User Management (basic)
  - Software Management
  - System Monitoring (basic)
  - System Information & Configuration



What can you do with OpenLMI?

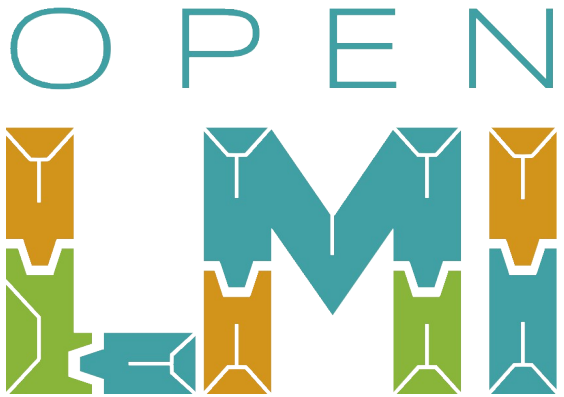
- Storage
- Networks
- Users
- Software
- Power
- System Services
- System configuration information
- Active Directory
- Firewall
- JBoss JON
- SCAP
- Monitoring & Alerts

More to come!



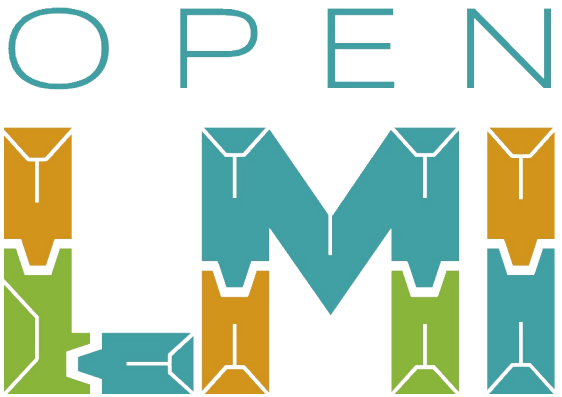
Benefits

- More Productive Sysadmins
  - Familiar environment
  - Standard API
  - Scripting friendly environment
  - Manage remote systems without logging in locally
- Shorter learning curve for Linux System Administration
  - Common framework across disparate subsystems
- Foundation for Automation
- Can be used by management platforms



Your Opportunity

- Give us requirements and feedback
- Evaluate OpenLMI in Fedora 19
- Get involved at OpenLMI.org
  - Testing
  - Use cases
  - Scripts
  - Agents
  - Tools



Key Information

- Russell Doty: rdoty@redhat.com
- Stephen Gallagher: sgallagh@redhat.com
- www.openlmi.org
- lists.fedorahosted.org/mailman/listinfo/openlmi-devel
- #openlmi on freenode