Linux-HA Release 2 Tutorial

Alan Robertson
Project Leader – Linux-HA project
alanr@unix.sh

IBM Systems & Technology Group
Industry Technology Leadership Team
HA Subject Matter Expert

Tutorial Overview

- HA Principles
- Installing Linux-HA
- Basic Linux-HA configuration
- Configuring Linux-HA
- Sample HA Configurations
- Testing Clusters
- Advanced features

Part I

- General HA principles
- Architectural overview of Linux-HA
- ~~Compilation and installation of the Linux-HA ("heartbeat") software~~

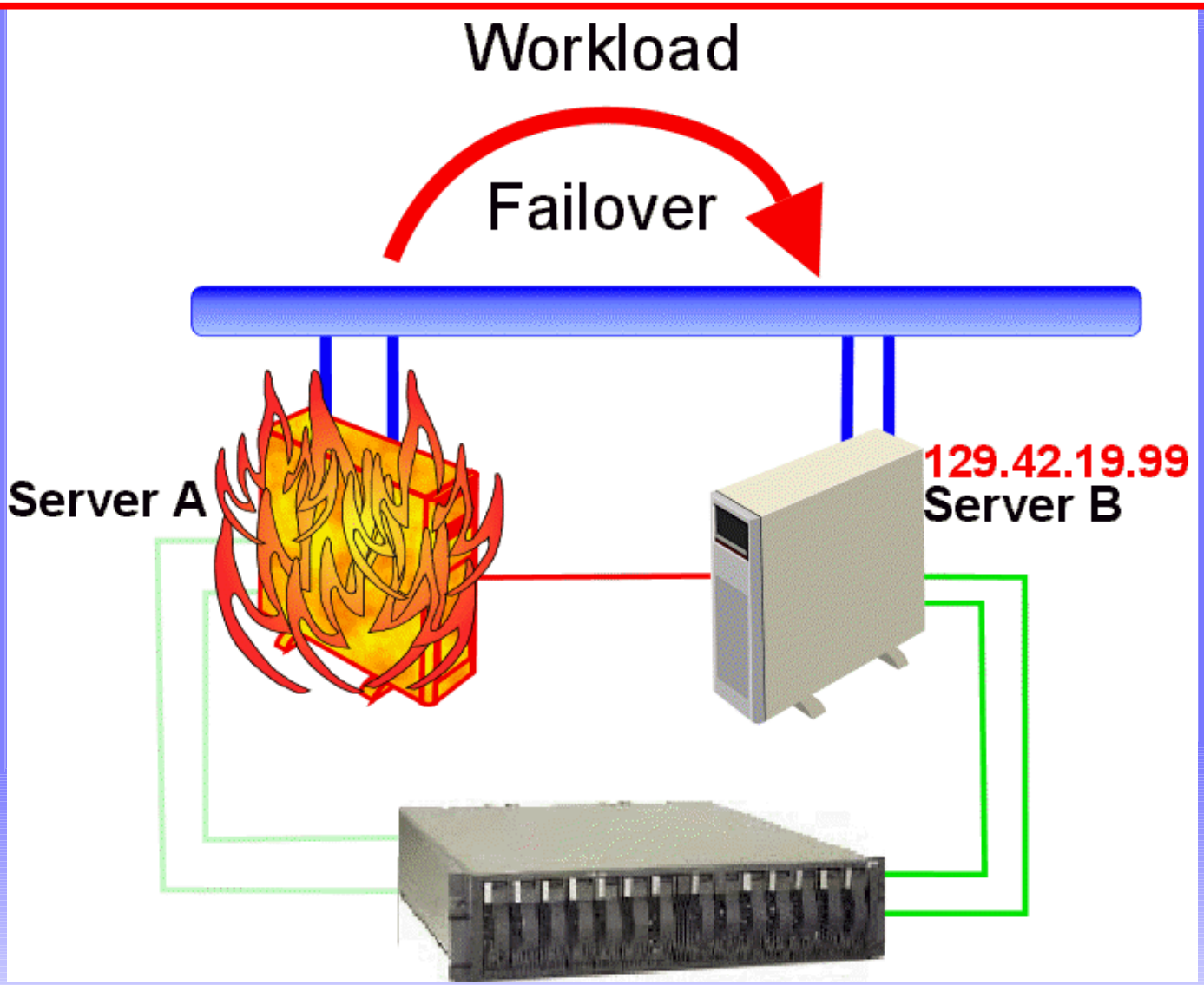
What Is HA Clustering?

- Putting together a group of computers which trust each other to provide a service even when system components fail
- When one machine goes down, others take over its work
- This involves IP address takeover, service takeover, etc.
- New work comes to the “takeover” machine
- Not primarily designed for high-performance

What Can HA Clustering Do For You?

- **It cannot achieve 100% availability** – *nothing can.*
- HA Clustering designed to recover from single faults
- It can make your outages very short
  - From about a second to a few minutes
- It is like a Magician's (Illusionist's) trick:
  - When it goes well, the hand is faster than the eye
  - When it goes not-so-well, it can be reasonably visible
- A good HA clustering system adds a “9” to your base availability
  - 99->99.9, 99.9->99.99, 99.99->99.999, etc.
- **Complexity is the enemy of reliability!**

High-Availability Workload Failover



Lies, Damn Lies, and Statistics

Counting nines

99.9999%	30 sec
99.999%	5 min
99.99%	52 min
99.9%	9 hr
99%	3.5 day

How is this like what you know?

- It's a lot like the current init startup scripts extended by
  - (optionally) adding parameters to them
  - running on a more than one computer
  - adding policies for
    - what order to do things
    - how services relate to each other
    - when to run them
- HA systems are a lot like “init on steroids”

What is different?

- HA Clusters introduce concepts and complications around
  - Split-Brain
  - Quorum
  - Fencing
- Data sharing isn't usually an issue with a single server – it's critically important in clusters

Split-Brain

- Communications failures can lead to separated partitions of the cluster
- If those partitions each try and take control of the cluster, then it's called a split-brain condition
- If this happens, then bad things will happen
  - http://linux-ha.org/BadThingsWillHappen

Quorum

- Quorum is an attempt to avoid split brain for most kinds of failures
- Typically one tries to make sure only one partition can be active
- Quorum is term for methods for ensuring this
- Most common kind of quorum is voting – and only a partition with > n/2 nodes can run the cluster
- This doesn't work very well for 2 nodes :-(

Fencing

- Fencing tries to put a fence around an errant node or nodes to keep them from accessing cluster resources
- This way one doesn't have to rely on correct behavior or timing of the errant node.
- We use STONITH to do this
  - STONITH: Shoot The Other Node In The Head
- Other techniques also work
  - Fiber channel switch lockout
  - etc

How is HA Clustering Different from Disaster Recovery ("geographic clustering")?

- HA:
  - Failover is cheap
  - Failover times measured in seconds
  - Reliable inter-node communication
- DR:
  - Failover is expensive
  - Failover times often measured in hours
  - Unreliable inter-node communication assumed
- Linux-HA provides special features to deal with "geographic clustering" (aka disaster recovery)

Single Points of Failure (SPOFs)

- A single point of failure is a component whose failure will cause near-immediate failure of an entire system or service
- Good HA design eliminates of single points of failure

Non-Obvious SPOFs

- Replication links are rarely single points of failure
  - The system may fail when another failure happens
- Some disk controllers have SPOFs inside them which aren't obvious without schematics
- Redundant links buried in the same wire run have a common SPOF
- **Non-Obvious SPOFs can require deep expertise to spot**

The "*Three R's*" of High-Availability

- **R**edundancy
- **R**edundancy
- **R**edundancy

- If this sounds redundant, that's probably appropriate... ;-)
- **Most SPOFs are eliminated by redundancy**
- **HA Clustering is a good way of providing and managing redundancy**

Redundant Communications

- Intra-cluster communication is critical to HA system operation
  - Most HA clustering systems provide mechanisms for redundant internal communication for heartbeats, etc.
- External communications is usually essential to provision of service
  - Exernal communication redundancy is usually accomplished through routing tricks
  - Having an expert in BGP or OSPF is a help

Fencing

- Guarantees resource integrity in the case of certain difficult cases
- Three Common Methods:
  - FiberChannel Switch lockouts
  - SCSI Reserve/Release (difficult to make reliable)
  - **Self-Fencing (like IBM ServeRAID)**
  - **STONITH – Shoot The Other Node In The Head**
- Linux-HA supports the last two models

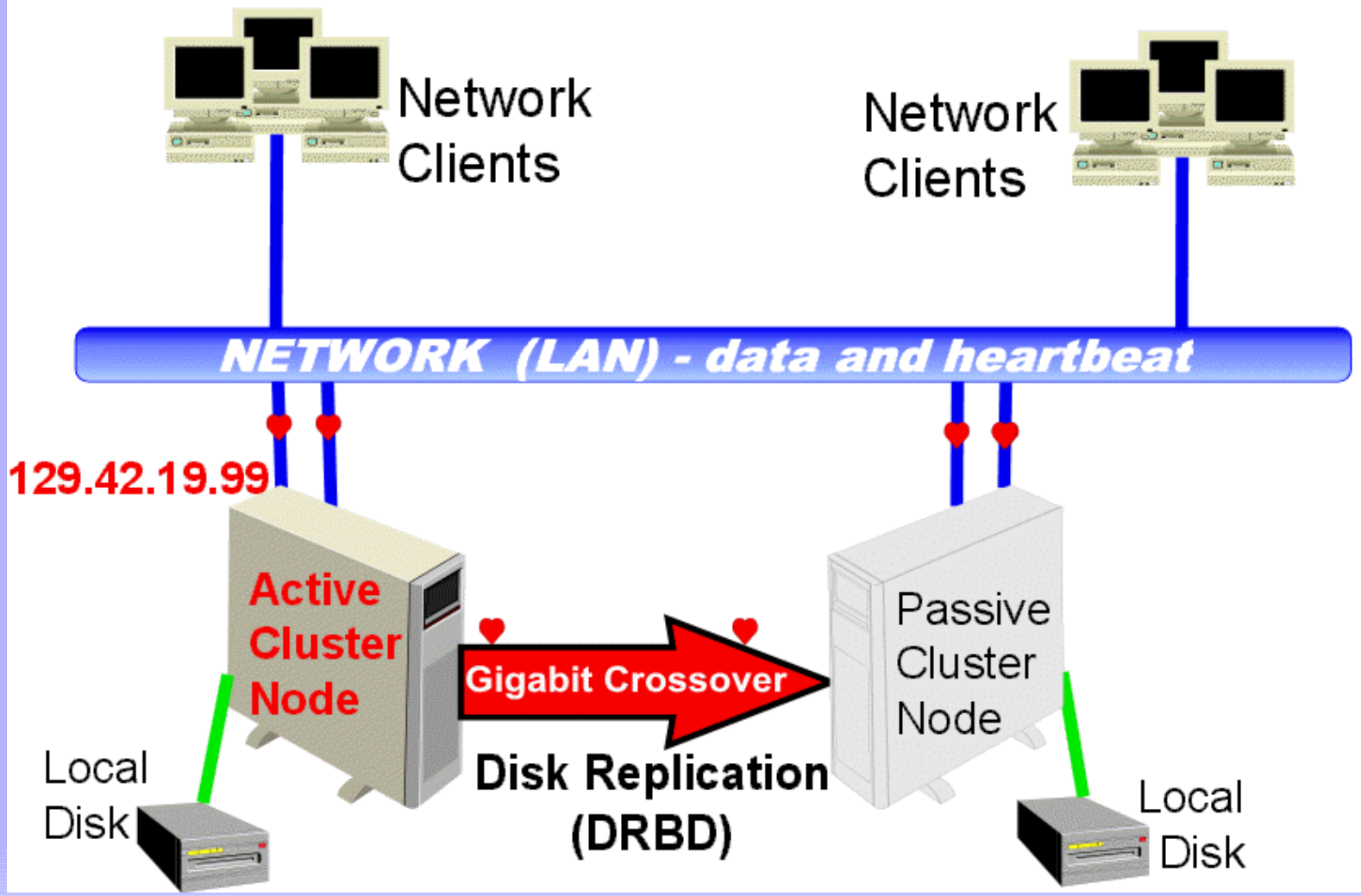
Data Sharing - *None*

- Strangely enough, some HA configurations don't need any formal disk data sharing
  - Firewalls
  - Load Balancers
  - (Caching) Proxy Servers
  - Static web servers whose content is copied from a single source

Data Sharing – *Replication*

- Some applications provide their own replication
  - DNS, DHCP, LDAP, DB2, etc.
- Linux has excellent disk replication methods available
  - DRBD is my favorite
  - ***DRBD-based HA clusters are extremely affordable***
- Some environments can live with less “precise” replication methods – rsync, etc.
- Often does not support parallel access
- Fencing highly desirable, but not always necessary
- ***EXTREMELY cost effective***

Two node Active/Passive HA Cluster Real-Time Disk Replication (*DRBD*)



Data Sharing – *ServeRAID*

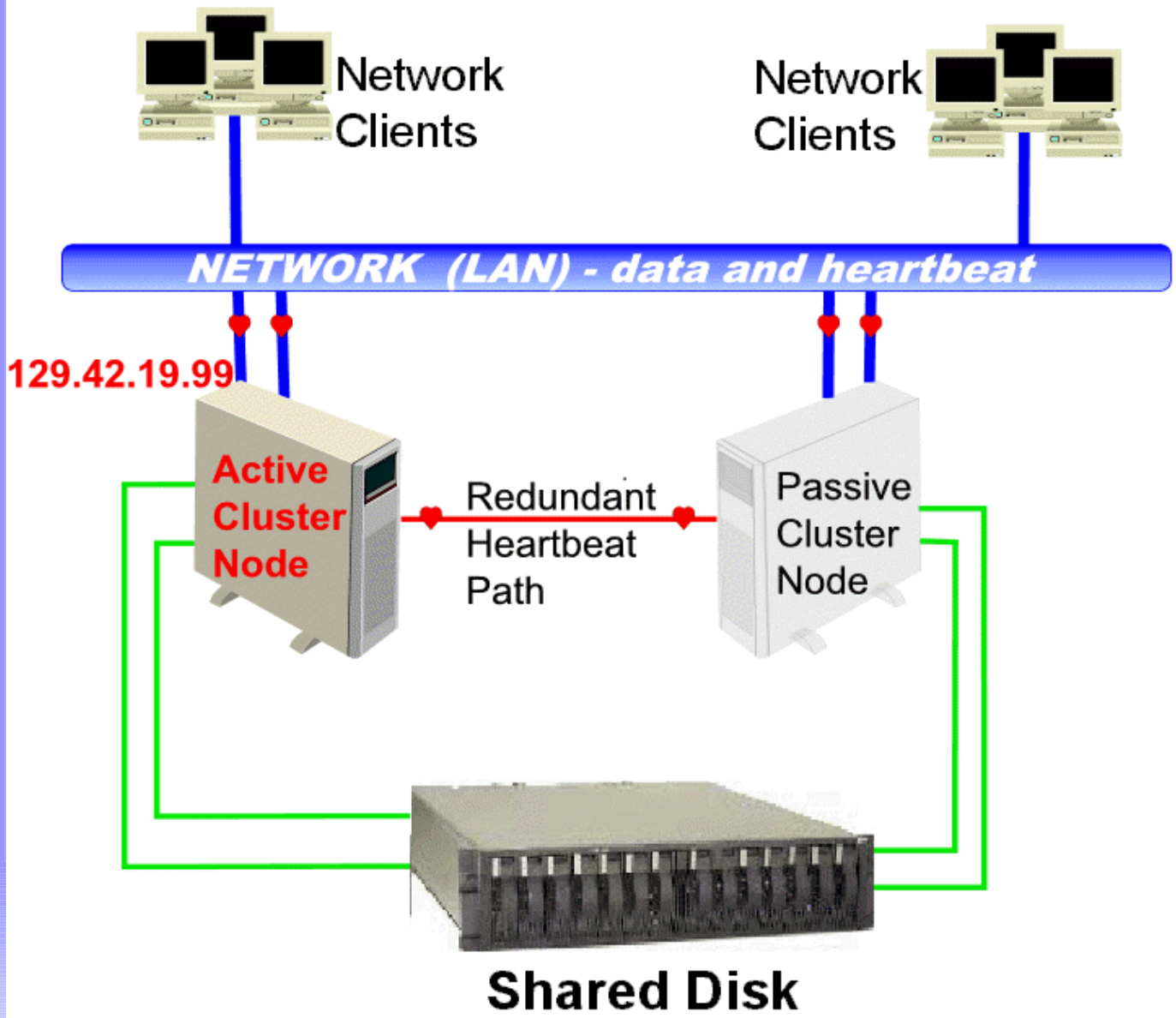
- IBM ServeRAID disk is **self-fencing**
- This helps integrity in failover environments
- This makes cluster filesystems, etc. impossible
  - No Oracle RAC, no GPFS, etc.
- ServeRAID failover requires a script to perform volume handover
- Linux-HA provides such a script in open source
- Linux-HA is ServerProven with IBM ServeRAID

Data Sharing – *FiberChannel*

- The most classic data sharing mechanism
- Allows for failover mode
- Allows for true parallel access
  - Oracle RAC, Cluster filesystems, etc.
- Fencing **always** required with FiberChannel
- Linux-HA is certified ServerProven with IBM storage

Two node Active/Passive HA Cluster

Shared Disk (FasTT, ESS, etc.)



Data Sharing – *Back-End*

- Network Attached Storage can act as a data sharing method
- Existing Back End databases can also act as a data sharing mechanism
- Both make reliable and redundant data sharing Somebody Else's Problem (SEP).
- If they did a good job, you can benefit from them.
- Beware SPOFs in your local network

Linux-HA Background

- The oldest and most well-known open-community HA project - providing sophisticated fail over and restart capabilities for Linux (and other OSes)
- In existence since 1998; ~ 30k mission-critical clusters in production since 1999
- Active, open development community led by IBM and Novell
- Wide variety of industries, applications supported
- Shipped with most Linux distributions (all but Red Hat)
- No special hardware requirements; no kernel dependencies, all user space
- All releases tested by automated test suites

Linux-HA Capabilities

- Supports n-node clusters – where 'n' <= something like 16
- Can use serial, UDP bcast, mcast, ucast comm.
- Fails over on node failure, or on service failure
- Fails over on loss of IP connectivity, or arbitrary criteria
- Active/Passive or full Active/Active
- Built-in resource monitoring
- Support for the OCF resource standard
- Sophisticated dependency model with rich constraint support (resources, groups, incarnations, master/slave) (needed for SAP)
- XML-based resource configuration
- Configuration and monitoring GUI
- Support for OCFS cluster filesystem
- Multi-state (master/slave) resource support
- Split-site (stretch) cluster support with quorum daemon

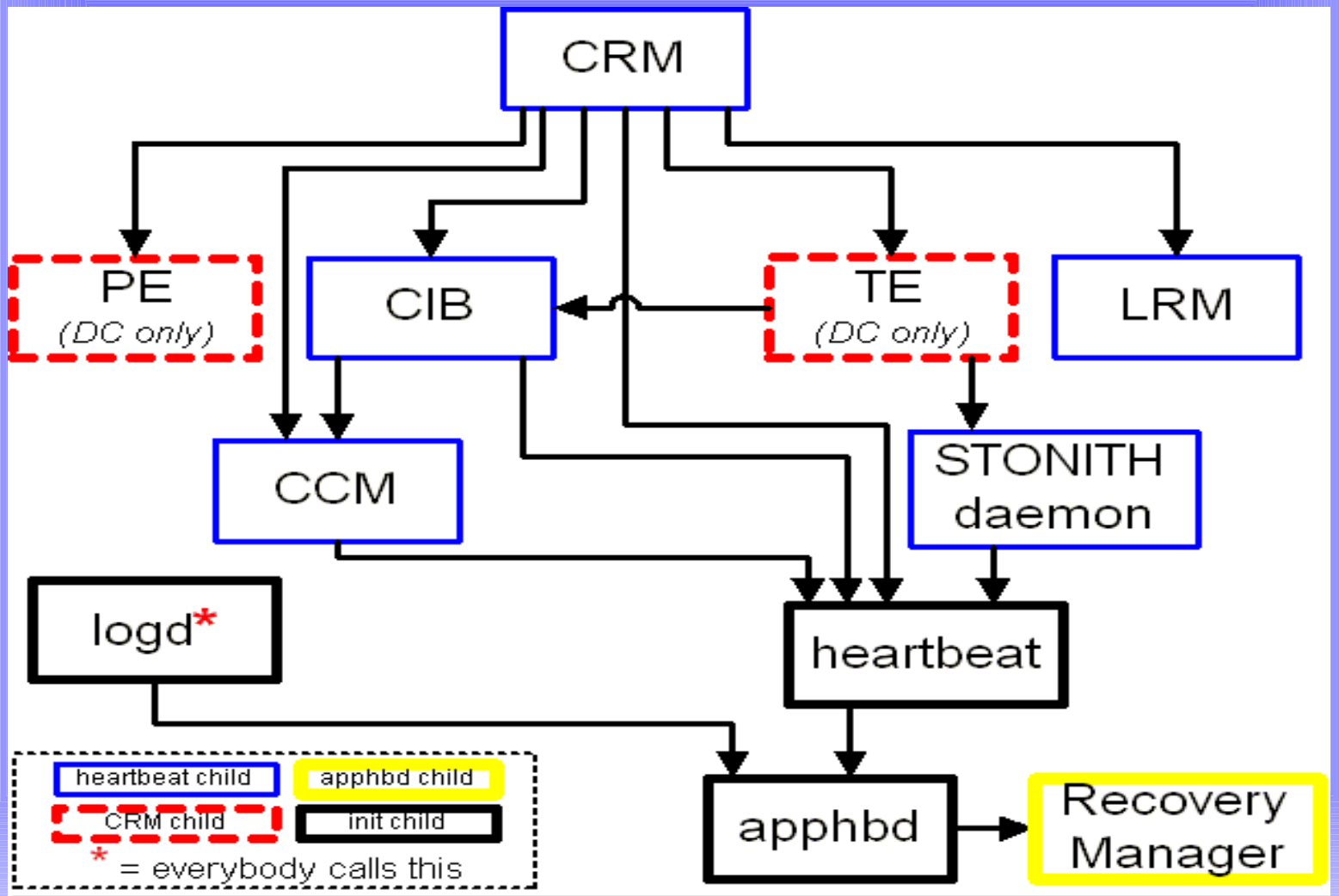
Some Linux-HA Terminology

- **Node** – a computer (real or virtual) which is part of the cluster and running our cluster software stack
- **Resource** – something we manage – a service, or IP address, or disk drive, or whatever. If we manage it and it's not a node, it's a resource
- **Resource Agent** – a script which acts as a proxy to control a resource. Most are closely modelled after standard system init scripts.
- **DC** – Designated Coordinator – the “master node” in the cluster
- **STONITH** – Acronym for **Shoot The Other Node In The Head** – a method of fencing out nodes which are misbehaving by resetting them
- **Partitioned cluster** or **Split-Brain** – a condition where the cluster is split into two or more pieces which don't know about each other through hardware or software failure. Prevented from doing BadThings by STONITH
- **Quorum** – normally assigned to at most one single partition in a cluster to keep split-brain from causing damage. Typically determined by a voting protocol

Key Linux-HA Processes

- **CRM** – Cluster Resource Manager – The main management entity in the cluster
- **CIB** – The cluster Information Base – keeper of information about resources, nodes. Also used to refer to the information managed by the CIB process. The CIB is XML-based.
- **PE** – Policy Engine – determines what should be done given the current policy in effect – creates a graph for the TE containing the things that need to be done to bring the cluster back in line with policy *(only runs on the DC)*
- **TE** – Carries out the directives created by the PE – through it's graph *(only runs on the DC)*
- **CCM** – Consensus Cluster Membership – determines who is in the cluster, and who is not. A sort of gatekeeper for cluster nodes.
- **LRM** – Local Resource Manager – low level process that does everything that needs doing – not cluster-aware – no knowledge of policy – ultimately driven by the TE (through the various CRM processes)
- **stonithd** – daemon carrying out STONITH directives
- **heartbeat** – low level initialization and communication module

Linux-HA Release 2 Architecture



Compiling and Installing Linux-HA from source via RPM or .deb

- Grab a recent stable tar ball >= 2.0.7 from: http://linux-ha.org/download/index.html
- untar it with: `tar tzf heartbeat-2.0.x.tar.gz`
- `cd heartbeat-2.0.x`
- `./ConfigureMe package`
- `rpm -install full-RPM-pathnames`
- `./ConfigureMe package` produces packages appropriate to the current environment (including Debian, Solaris, FreeBSD, etc.)

Pre-built Packages

- The Linux-HA download site includes SUSE-compatible packages
- Debian includes heartbeat packages – for Sid and Sarge
- Fedora users can use yum to get packages
  - `$ sudo yum install heartbeat`
- RHEL-compatible versions are available from CentOS
  - `http://dev.centos.org/centos/4/testing/i386/RPMS/`
  - `http://dev.centos.org/centos/4/testing/x86_64/RPMS/`

RPM Package names

- **heartbeat-pils** – plugin loading system
- **heartbeat-stonith** – STONITH libraries and binaries
- **heartbeat** – main heartbeat package
- **heartbeat-ldirectord** – code for managing Linux Virtual Server installations

- The ldirectord subpackage is optional
- All other subpackages are mandatory. Fedora dropped the heartbeat prefix from the pils and stonith subpackages.

Installing RPMs

- rpm –install heartbeat-2.0.*x*-1.*xxx*.rpm \
  heartbeat-pils-2.0.*x*-1.*xxx*.rpm \
  heartbeat-stonith-2.0.*x*-1.*xxx*.rpm
- That was simple, wasn't it?

Initial configuration

- Create the following files by copying templates found in your system's documentation directory `/usr/share/doc/`*heartbeat-version* into /etc/ha.d
  - `ha.cf  -> /etc/ha.d/ha.cf`
  - `authkeys -> /etc/ha.d/authkeys`

Fixing up /etc/ha.d/ha.cf

- Add the following directives to your ha.cf file:

```
node node1 node2 node3 # or enable autojoin
bcast eth0 # could use mcast or ucast
crm on  # this is the minimum set
```

- For complete documentation on the ha.cf file see:

http://linux-ha.org/ha.cf

Fixing up /etc/ha.d/authkeys

- Authkeys provides a shared authentication key for the cluster. Each cluster should have a different key.
- Add 2 lines a lot like these to authkeys:

```
auth 1
1 sha1 PutYourSuperSecretKeyHere
```

- File ***MUST*** be mode **0600** or **0400**
- Be sure and change your signature key ;-)
- Complete documentation on authkeys is here:

http://linux-ha.org/authkeys

Creating /var/lib/heartbeat/crm/cib.xml

- It has to be owned by haclient:hacluster
- It should be mode **0600**
- When the cluster is running, it is managed by the CIB process – don't mess with it directly!
- **cib.xml** *can* be updated while the cluster is running – either completely or incrementally
- **cib.xml** is described here:

http://linux-ha.org/ClusterResourceManager/DTD1.0/Annotated

- We will spend much of the rest of the class talking about what to put in cib.xml :-)

Part II

- System Concepts
- Introducing CIB configuration

Resource Objects in Release 2

- Release 2 supports “resource objects” which can be any of the following:
  - Primitive Resources
    - OCF, heartbeat-style, or LSB resource agent scripts
  - Resource Clones – need *“n”* resource objects - somewhere
  - Resource Groups – a group of primitive resources with implied co-location and linear ordering constraints
  - Multi-state resources (master/slave)
    - Designed to model master/slave (replication) resources (DRBD, et al)

OCF Class Resource Agents

- OCF == Open Cluster Framework
- OCF Resource agents are the most powerful type of resource agent we support
- OCF RAs are extended init scripts
  - They accept parameters from the environment
  - They have additional actions:
    - monitor – for monitoring resource health
    - meta-data – for providing information about the RA for GUI
    - validate-all – for validating resource parameters
- OCF RAs are located in **`/usr/lib/ocf/resource.d/`*`provider-name`*`/`**
- See **`http://linux-ha.org/OCFResourceAgent`**

LSB Class Resource Agents

- LSB == Linux Standards Base
- LSB resource agents are standard System V-style init scripts commonly used on Linux and other UNIX-like OSes
- LSB init scripts are stored under **`/etc/init.d/`**
- This enables Linux-HA to immediately support nearly every service that comes with your system, and most packages which come with their own init script
- It's straightforward to change an LSB script to an OCF script
- See **`http://linux-ha.org/LSBResourceAgent`**

'heartbeat' (R1) Class Resource Agents

- Similar to LSB init scripts except they take command line parameters
- **status** operation used for resource monitoring
- Typically not interesting for R2 installations
- Provided for compatibility with R1 versions of heartbeat for customers who wrote their own resource agents

http://linux-ha.org/HeartbeatResourceAgent

stonith Resource Agents

- Provide a wrapper for STONITH reset modules
- Very similar to LSB resource agents – from a configuration point of view
- STONITH reset modules can be written in 'C' or any scripting language
- STONITH reset modules follow the STONITH API, not a resource agent API

Basic Dependencies in Release 2

- Ordering Dependencies
  - start before *(normally implies stop after)*
  - start after *(normally implies stop before)*
- Mandatory Co-location Dependencies
  - must be co-located with
  - cannot be co-located with

Resource Location Constraints

- **Mandatory Constraints:**
  - Resource Objects can be constrained to run on any selected subset of nodes. Default depends on setting of *symmetric_cluster*.
- **Preferential Constraints:**
  - Resource Objects can also be preferentially constrained to run on specified nodes by providing weightings for arbitrary logical conditions
  - The resource object is run on the node which has the highest weight (score)

Resource Clones

- Resource Clones allow one to have a resource which runs multiple (*"n"*) times on the cluster
- This is useful for managing
  - load balancing clusters where you want *"n"* of them to be slave servers
  - Cluster filesystems
  - Cluster Alias IP addresses

Resource Groups

Resource Groups provide a simple method for creating ordering and co-location dependencies

- Each resource object in the group is declared to have linear *start-after* ordering relationships
- Each resource object in the group is declared to have co-location dependencies on each other
- This is an easy way of converting release 1 resource groups to release 2

Multi-State (master/slave) Resources

- Normal resources can be in one of two stable states:
  - **`started`**
  - **`stopped`**
- Multi-state resources can have more than two stable states. For example:
  - **`stopped`**
  - **`running-as-master`**
  - **`running-as-slave`**
- This is ideal for modelling replication resources like DRBD, HADR (IBM DB2) and Oracle DataGuard

Advanced Constraints

- Nodes can have arbitrary attributes associated with them in name=value form
- Attributes have types: **int**, **string**, **version**
- Constraint expressions can use these attributes as well as node names, etc. in largely arbitrary ways
- Operators:
  - =, **!=,** <, >, <=, >=
  - **defined**(*attrname*), **undefined**(*attrname*),
  - **colocated**(*resource id*), **not colocated**(*resource id*)

Advanced Constraints (cont'd)

- Each constraint is associated with particular resource, and is evaluated in the context of a particular node.
- A given constraint has a boolean predicate associated with it according to the expressions before, and is associated with a weight, and condition.
- If the predicate is true, then the condition is used to compute the weight associated with locating the given resource on the given node.
- All conditions are given weights, positive or negative. Additionally there are special values for modeling must-have conditions
  - +INFINITY
  - -INFINITY

Cluster Information Base (CIB) Intro

- The CIB is an XML file containing:
  - Configuration Information
    - Cluster Node information
    - Resource Information
    - Resource Constraints
  - Status Information
    - Which nodes are up / down
    - Attributes of nodes
    - Which resources are running where
- We only provide configuration information

About 'id's

- As you will soon see, many/most tags in our XML DTD require 'id' attributes
- These values absolutely must be unique among all other tags of the same type
- It is good practice to make them globally unique
- These tags are used by the cib_admin in order to specify exactly which part of the CIB is being modified

About <nvpair>s

- Many places in the CIB, we need to have allow an arbitrary set of name/value pairs
- In those places, we use <**nvpair**> tags.
- Basic syntax is:

<**nvpair** id="some-unique-id" name="some_name"
value="some-value"/>

This is XML's verbose way of saying:
some_name="some-value"

An Empty CIB

```
<cib>
   <configuration>
     <crm_config/>
     <nodes/>
     <resources/>
     <constraints/>
   </configuration>
   <status/>
</cib>
```

The **crm_config** CIB section

```
<crm_config>
   <cluster_property_set
            id="cib-bootstrap-options">

    <attributes>
      <nvpair/>
    </attributes>

  </cluster_property_set>
</crm_config>
```

crm_config Global Cluster Properties

- transition_idle_timeout
- symmetric_cluster
- no_quorum_policy
- stonith_enabled
- stonith_action
- startup_fencing
- default_resource_stickiness
- default_resource_failure_stickiness
- is_managed_default
- stop_orphan_resources
- stop_orphan_actions
- short_resource_names

crm_config: transition_idle_timeout

- **interval, default=60s**
  - Provides the default global timeout for actions
  - Any action which has a defined timeout automatically uses the higher timeout

crm_config: symmetric_cluster

- **boolean, default=TRUE**
  - If true, resources are permitted to run anywhere by default.
  - Otherwise, explicit constraints must be created to specify where they can run.
- Typically set to **TRUE**

crm_config: default_resource_stickiness

- Do we prefer to run on the existing node or be moved to a "better" one?
  - **0** : resources will be placed optimally in the system. This may mean they are moved when a "better" or less loaded node becomes available. This option is almost equivalent to the old **auto_failback on** option
  - **value > 0** : resources will prefer to remain in their current location but may be moved if a more suitable node is available. Higher values indicate a stronger preference for resources to stay where they are.
  - **value < 0** : resources prefer to move away from their current location. Higher absolute values indicate a stronger preference for resources to be moved.

default_resource_stickiness (cont'd)

- Special cases:
  - **INFINITY** : resources will always remain in their current locations until forced off because the node is no longer eligible to run the resource (node shutdown, node standby or configuration change). This option is almost equivalent to the old **auto_failback off** option.
  - **-INFINITY** : resources will always move away from their current location.

crm_config: is_managed_default

- **boolean, default=TRUE**
  - **TRUE** : resources will be started, stopped, monitored and moved as necessary/required
  - **FALSE** : resources will not be started if stopped, stopped if started nor have any recurring actions scheduled.
- Can be overridden by the resource's definition
- Handy for disabling management of resources for software maintenance

crm_config: no_quorum_policy

- **enum, default=stop**
  - **stop** Stop all running resources in our partition requiring quorum. Fencing is disabled
  - **ignore** Pretend we have quorum
  - **freeze** Do not start any resources not currently in our partition. Resources in our partition may be moved to another node within the partition. Fencing is disabled

crm_config: stonith_enabled

- **boolean, default=FALSE**
  - If **TRUE**, failed nodes will be fenced.
- A setting of **TRUE** requires STONITH-class resources to be configured for correct operation.

crm_config: stonith_action

- **enum {reboot,off}, default=reboot**
  - If set to **reboot**, nodes are rebooted when they are fenced
  - If set to **off**, nodes are shut off when they are fenced
- Typically defaulted to **reboot**

crm_config: startup_fencing

- **boolean, default=TRUE**
  - If true, nodes we have never heard from are fenced
  - Otherwise, we only fence nodes that leave the cluster after having been members of it first
- *Potentially dangerous* to set to **FALSE**

crm_config: stop_orphan_resources

- **boolean, default=TRUE** (as of release 2.0.6)
- Defines the action to take on running resources for which we currently have no definition:
  - **TRUE** : Stop the resource
  - **FALSE** : Ignore the resource
- This defines the CRM's behavior when a resource is deleted by an admin without it first being stopped.

crm_config: stop_orphan_actions

- **boolean, default=TRUE**
- What to do with a recurring action for which we have no definition:
  - **TRUE** : Stop the action
  - **FALSE** : Ignore the action
- This defines the CRM's behavior when the interval for a recurring action is changed.

crm_config: short_resource_names

- **boolean, default=FALSE, recommended=TRUE**
  - This option is for backwards compatibility with versions earlier than 2.0.2 which could not enforce id-uniqueness for a given tag type.
- It is highly recommended that you set this to **TRUE**.
- **WARNING:** The cluster must be completely stopped before changing this value

The `nodes` section of the CIB

- We let the CRM get the nodes information from the membership layer (and some from the 'heartbeat' layer)
- This makes things much easier on us :-)

The **resources** section of the CIB

- The resources section is one of the most important sections.
- It consists of a set of individual resource records
- Each resource record represents a single resource

```
<resources>
   <primitive/>
   <primitive/>
   ...
</resources>
```

Classes of Resource Agents in R2

- **OCF** – Open Cluster Framework - http://opencf.org/
  - take parameters as name/value pairs through the environment
  - Can be monitored well by R2
- **Heartbeat** – R1-style heartbeat resources
  - Take parameters as command line arguments
  - Can be monitored by **`status`** action
- **LSB** – Standard LSB Init scripts
  - Take no parameters
  - Can be monitored by **`status`** action
- **Stonith** – Node Reset Capability
  - Very similar to OCF resources

An OCF **primitive** object

```
<primitive id="WebIP" class="ocf"
    type="IPaddr" provider="heartbeat">
 <instance_attributes>
  <attributes>
    <nvpair
        name="ip"
        value="192.168.224.5"/>
  </attributes>
 </instance_attributes>
</primitive>
```

A STONITH **primitive** object

```
<primitive id="st" class="stonith"
    type="ibmhmc">
 <instance_attributes>
  <attributes>
    <nvpair
        name="ip"
        value="192.168.224.99"/>
  </attributes>
 </instance_attributes>
</primitive>
```

An LSB **primitive** object
(i. e., an init script)

```
<primitive id="samba-smb"
    class="lsb"
    type="smb">
  <instance_attributes>
    <attributes/>
  </instance_attributes>
</primitive>
```

meta_attributes of Primitives

- **is_managed** – **FALSE** means heartbeat ignores it
- **resource_stickiness** – how badly do we want to stay where we are (if possible) when nodes return
- **resource_failure_stickiness** – ditto for resource failure
- **restart_type** - *dependency* {**restart**, ignore} ??
- **multiple_active**: {stop_only, block, **stop/start}**
- **start_prereq** {nothing, quorum, **fencing**}
- **priority** – resource placement ordering - defaults to **0**
- **target_role** – {Started,Stopped,Master, Slave, **default}**

attributes of Resource Actions

- **timeout** – how long can action take before timing out
- **interval** – how long to delay before repeating monitor action
- **start_delay** – how long to wait before starting the action (typically monitor)
- **on_fail** (action: stop) – {block,**fence**,stop, restart,nothing}
  - *What is default? Is there a global value for this?*

Setting monitor check level

```
<op id="apache_a1_mon" interval="120s"
        name="monitor" timeout="60s">
  <instance_attributes id="apache_a1_mon_attr">
    <attributes>
      <nvpair id="apache_a1_mon_attr_0"
              name="OCF_CHECK_LEVEL"
              value="20"/>
     </attributes>
  </instance_attributes>
</op>
```

The DTD allows **<attributes>** to be preceded by a **<rule>** so that 'deep' checking can be restricted to run on any criteria (like time of day)

Resource Groups

- Resources can be put together in groups a lot like R1 resource groups or those of other HA systems
- Groups are simple to manage, but less powerful than individual resources with constraints

```
<group id="webserver">
   <primitive/>
   <primitive/>
</group>
```

- By default, groups imply co-location and ordering, these properties are optional

meta_attributes of Groups

- **ordered** – boolean – defaults to TRUE
  - TRUE means the group physical ordering implies start-after ordering constraints
  - FALSE means no such start-after ordering is implied
- **collocated** – boolean – defaults to TRUE
  - TRUE means all members of the group must be co-located
  - FALSE means no such co-location is implied

Disabling both makes the group a naming convenience

- **target_role**: same as for primitives – inherited by contained resources

Resource "clone" Units

- If you want a resource to run in several places, then you can "clone" the resource

```
<clone id="MyID">
   <instance_attributes>
     <attributes/>
   </instance_attributes>
   <primitive>
     <operations/>
     <instance_attributes/>
   </primitive/>
</clone>
```

meta_attributes of Clones

- **clone_max** – the maximum number of clones running total
- **clone_node_max** – maximum number of clones running on a single node
- **notify** – **TRUE** means peer notification is to be given
- **globally_unique** – **TRUE** means the clone number is unique across the entire cluster, **FALSE** means its only locally unique
- **ordered** – means don't overlap clone operations (start, etc.)
- **interleave** – means start clones with their respective operations interleaved. Otherwise, start each clone completely before going on to resources in the next (only meaningful with ordered=**TRUE**)
- See also **`http://linux-ha.org/v2/Concepts/Clones`**

STONITH "clone" resource(s)

```
<clone id="fencing">
  <instance_attributes>
    <attributes>
      <nvpair id="1" name="clone_max" value="2"/>
      <nvpair id="2" name="globally_unique" value="false"/>
    </attributes>
  </instance_attributes>
  <primitive id="fencing_op" class="stonith" type="ibmhmc">
    <operations>
      <op id="1" name="monitor" interval="5s" timeout="20s"
                 prereq="nothing"/>
      <op id="2" name="start" timeout="20s" prereq="nothing"/>
    </operations>
    <instance_attributes>
      <attributes>
        <nvpair id="1" name="ip" value="192.168.224.99"/>
      </attributes>
    </instance_attributes>
  </primitive>
</clone>
```

Useful tools for checking your CIB

- `crm_verify -LV` – checks your CIB for errors
  - Everyone should run this!
- `ptest` – tells what will happen when various failure events occur
  - ptest uses the same decision engine libraries as the CRM does
  - ptest is also used for regression testing during heartbeat development

Part III

- Sample configurations and rules

CIB **constraints**

```
<constraints>
   <rsc_location>
     <rule/>
     <rule/>
   </rsc_location>
</constraints>
```

rsc_location information

- We prefer to run on host `cl01lp2`

```
<rsc_location id="run_webserver"
              group="webserver">
  <rule id="rule_webserver" score=100>
    <expression attribute="#uname"
       operation="eq" value="cl01lp2"/>
  </rule>
</rsc_location>
```

Managing init (LSB) services

- LSB services are monitored using their status operation – this is usually a pretty wimpy monitor
- It is important that they conform to the LSB exit code behavior
  - status operation actually implemented
  - status operation when stopped exits with 1, 2, or 3
  - stopping when stopped returns exit code 0
  - starting when started returns exit code 0

Managing init (LSB) services – cont'd

- Make sure the data needed by the service is on a shared or replicated filesystem
- Don't forget to put the configuration files on shared media too
- Symlinks are your friend!
  - Sean Reifschneider's **`drbdlinks`** resource agent is really handy for managing symbolic links.
    You don't need to be running DRBD to use it :-D

LSB -> OCF conversion

- Add a **`monitor`** action to monitor the service in detail
- Add environment parameters **`OCF_RESKEY_xxx`** to allow more than one copy to be running, or get their data or configuration from 'non-standard' places
- Add a **`meta-data`** action to deliver meta data to stdout describing the resource, and its parameters (here-documents are convenient for this)
- add a **`validate-all`** action to validate the parameters given to the service and give an error exit if they're invalid
- Exit codes for new actions follow the LSB conventions

drbdlinks

- DRBDlinks is a handy tool for managing symbolic links for filesystem mounts
- It is useful with filesystem replication (DRBD) or shared disk arrangements
- You need one drbdlinks resource for each filesystem you want to manage with it
- It is currently only available as a Heartbeat classic style resource (not yet as an OCF resource)
- Find it here: http://tummy.com/Community/software/drbdlinks/

drbdlinks configuration

```
restartSyslog(1)
mountpoint('/drbd1')
link('/etc/dhcpd.conf')
link('/etc/postfix')
link('/etc/named.conf')
link('/etc/named.conf.include')
link('/var/lib/dhcp')
link('/var/lib/named')
```

Sample DNS Configuration

```
<primitive id="Rnamed" class="lsb"
type="named">
  <operations>
    <op id="named_mon" interval="30s"
      name="monitor" timeout="60s"/>
  </operations>
  <instance_attributes/>
</primitive>
```

- named (DNS) needs to have the following symlinked onto shared disk to make it work when failing over:
  - **/etc/dhcpd.conf**
  - **/var/lib/dhcp**

Sample DHCP Configuration

```
<primitive id="Rdhcp" class="lsb" type="dhcpd">
 <operations>
   <op id="dhcp_mon"
       interval="30s" name="monitor"
       timeout="60s"/>
  </operations>
  <instance_attributes/>
</primitive>
```

- DHCP needs to have the following symlinked onto shared disk to make it work when failing over:
  - **/etc/dhcpd.conf**
  - **/var/lib/dhcp**

Sample Apache Configuration

```
<primitive id="Rdhcp" class="ocf"
           type="apache" provider="heartbeat">
 <operations>
   <op id="apache_mon"
       interval="30s" name="monitor"
       timeout="60s"/>
  </operations>
  <instance_attributes id="apache_inst_attrs>
   <attributes>
    <nvpair id="apache_config" name="configfile"
     value="/etc/apache2/httpd.conf"/>
   </attributes>
  </instance_attributes>
</primitive>
```

Sample Apache Configuration (continued)

- Apache needs to have the following symlinked onto shared disk to make it work when failing over:
  - `/etc/apache2`
  - *all the content directories, CGI scripts, etc. needed by the configuration*
- These file names, etc. vary from release to release, and distribution to distribution
- *Of course, changing the 'configfile'* option changes where /etc/apache2 (or equivalent) is to be found.
- You can set up apache config so no symlinks are needed
- MAKE SURE each apache config uses a different pid file (!)

Sample NFS Configuration

```
<primitive id="Rnfs" class="lsb" type="nfsserver">
 <operations>
  <op id="nfsmon" interval="30s" name="monitor"
                  timeout="60s"/>
 </operations>
 <instance_attributes/>
</primitive>
```

- NFS needs to have the following symlinked onto shared media to make it work when failing over:
  - **/etc/exports**
  - **/var/lib/nfs**
- Making the (major,minor) of disk devices match can be a bit tricky – newer versions of NFS can be convinced to not require this

Sample Samba Configuration

```
<primitive id="samba-smb"
    class="lsb"
    type="smb">  <operations>
   <op id="samba_mon"
       interval="30s" name="monitor"
       timeout="60s"/>
  </operations>
  <instance_attributes/>
</primitive>
```

- Samba needs to have the following symlinked onto shared disk to make it work when failing over:
  - **/etc/samba**
  - **/var/lib/samba**

Testing HA Configurations

- A configuration which has not been thoroughly tested will not be highly available - *for certain*.
- Be sure and keep at least a test cluster around for testing future changes and upgrades
- If you can't have a full copy of your production environment, a small copy is better than nothing!
- Virtual machines can help a lot here
- A significant percentage of avoidable outages occur because of untested operational procedures. Practice on your test environment.

Testing HA Configurations - 2

- Make sure you test at least these things:
  - Failure of every node
  - Failure of each resource (application)
  - Failure of the customer network to the active server
- Each should be tested multiple times, including with failback
- If you are using time-based rules, be sure and test during each time period
- Be sure and test under full load, overload and no load conditions

Testing HA Configurations - 3

- If you are have multi-path fiber channel disks be sure you test all these conditions:
  - Failure of a path into the disk controller
  - Failure of a path into the active host
  - Failure of a path into the passive host
- Be sure and test them under full load, overload, and no load conditions
- These kinds of failures can affect the timing of monitor operations

Testing HA Configurations - 4

- ***Systems with shared disks need to be configured with STONITH enabled***
- STONITH setup needs to be fully tested
  - Test by hand using the stonith command line tool
  - Force a STONITH by killing heartbeat on one of the active servers
  - Force a STONITH by powering off an active server
  - Force a STONITH by resetting an active server
- Make sure failure of one of your hosts doesn't systematically cause failure of its STONITH device

The crm_resource command

- crm_resource can be used to
  - force a resource to migrate to a particular node
  - un-migrate a resource
  - delete a resource
  - force re-probing for 'rogue' resources
  - retrieve and set properties for a resource
  - retrieve parameters for a resource
  - locate which node a resource is running on
  - reset failure counters

The crm_standby Command

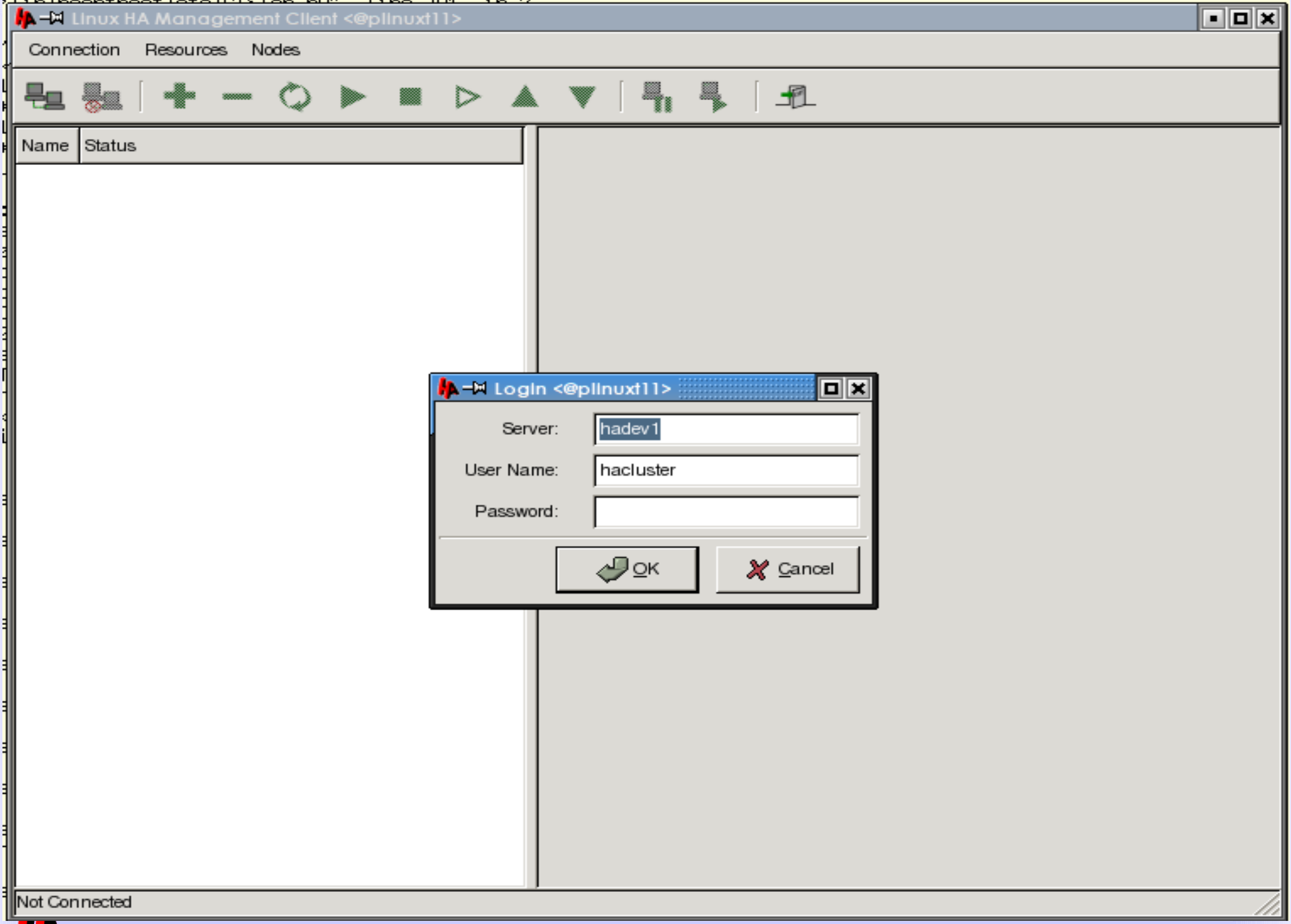
- crm_standby can be used to:
  - put a node into standby mode
  - remove a node from standby status
  - retrieve the standby status of a node
- A node can be put into or taken out of standby status either indefinitely, or until next reboot

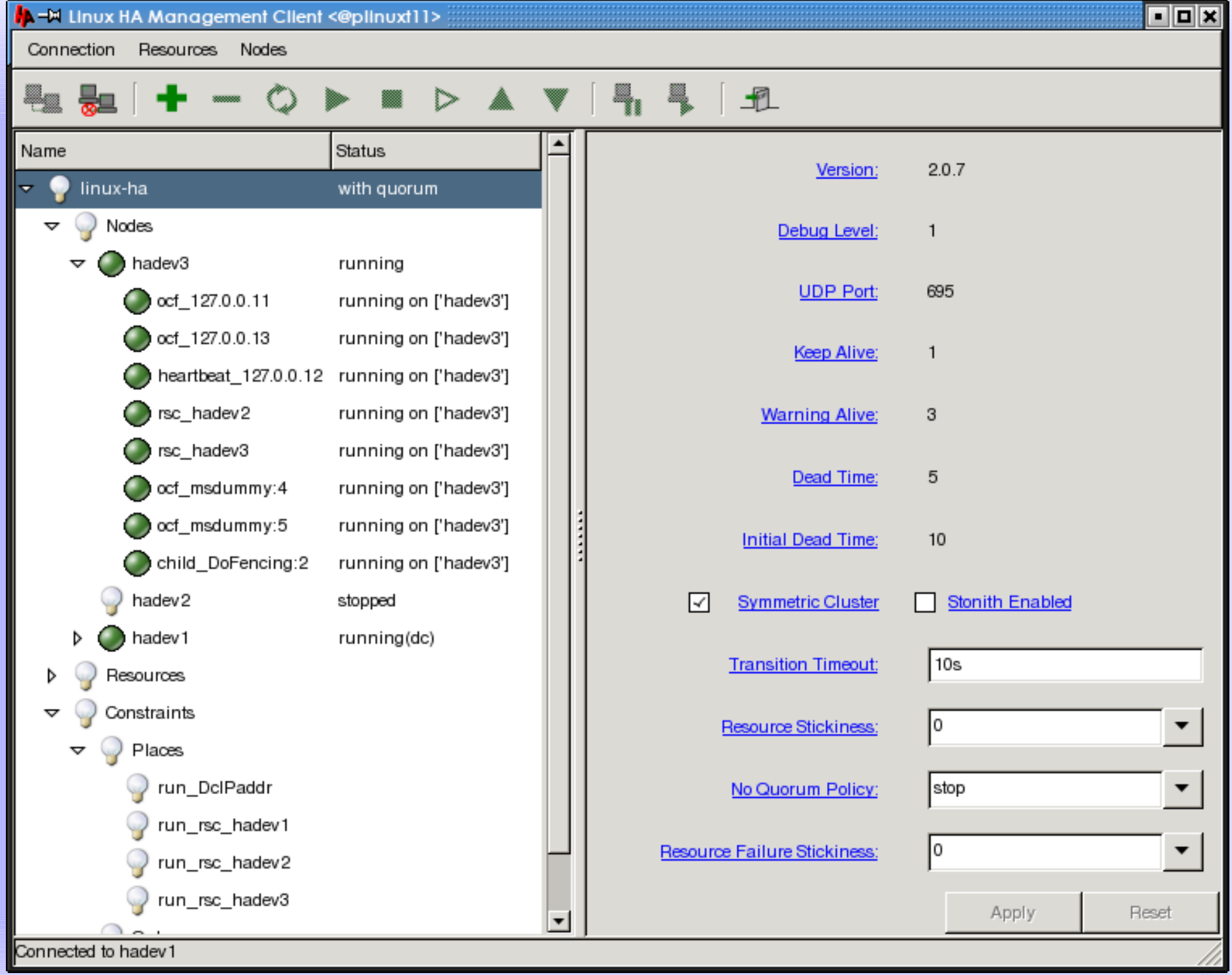
The cibadmin command

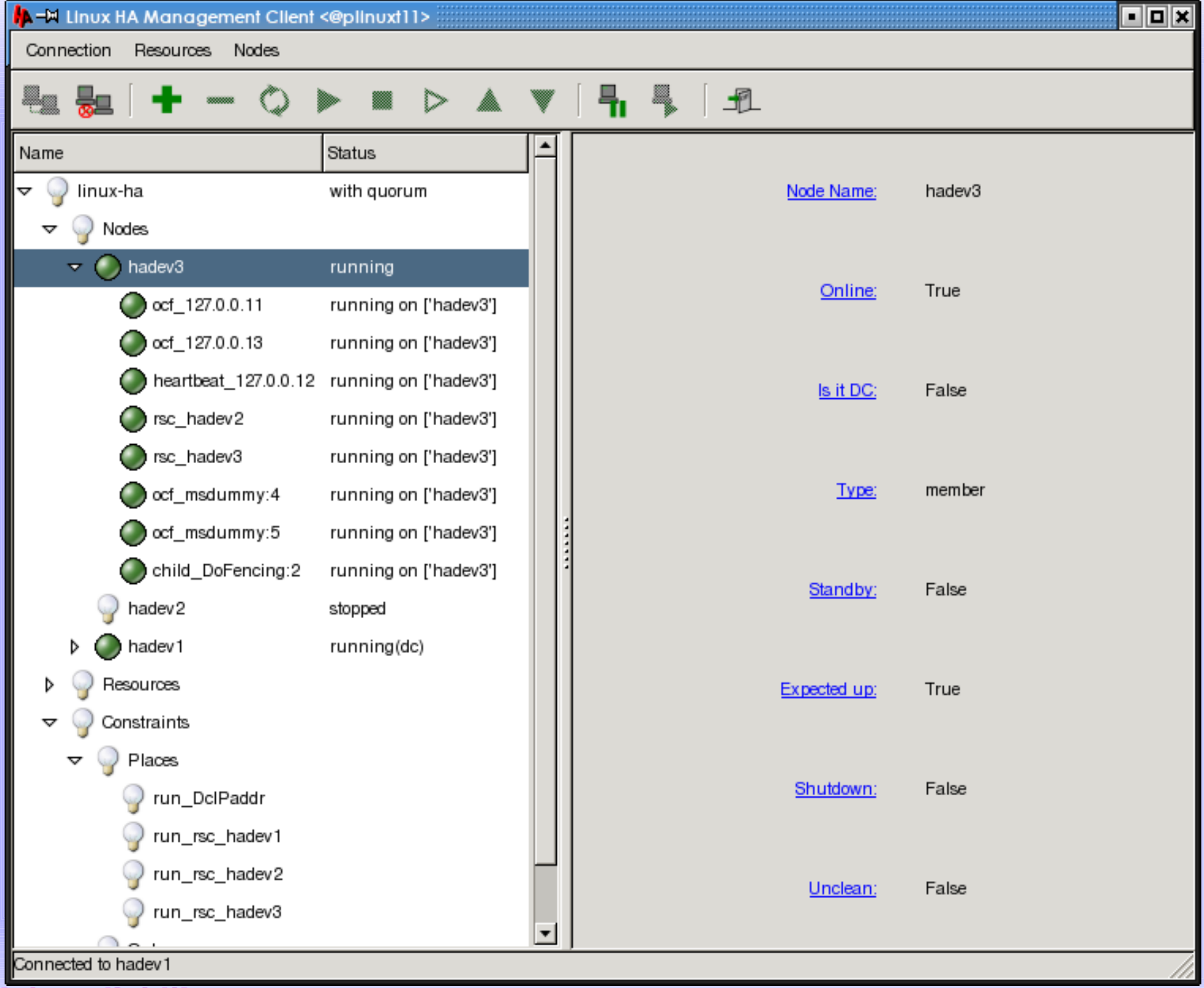
- **`cibadmin`** can be used to do a wide variety of potentially dangerous things to your CIB:
  - Dump out the current live CIB or selected sections of it
  - add XML to a specific part of the XML subtree
  - remove a specific XML subtree or leaf node
  - modify an attribute in a particular XML element
  - replace a particular XML element or subtree
  - indicate whether current CIB is master CIB or not
  - force a resync of the CIB from the master CIB
- cibadmin has a reasonable man page

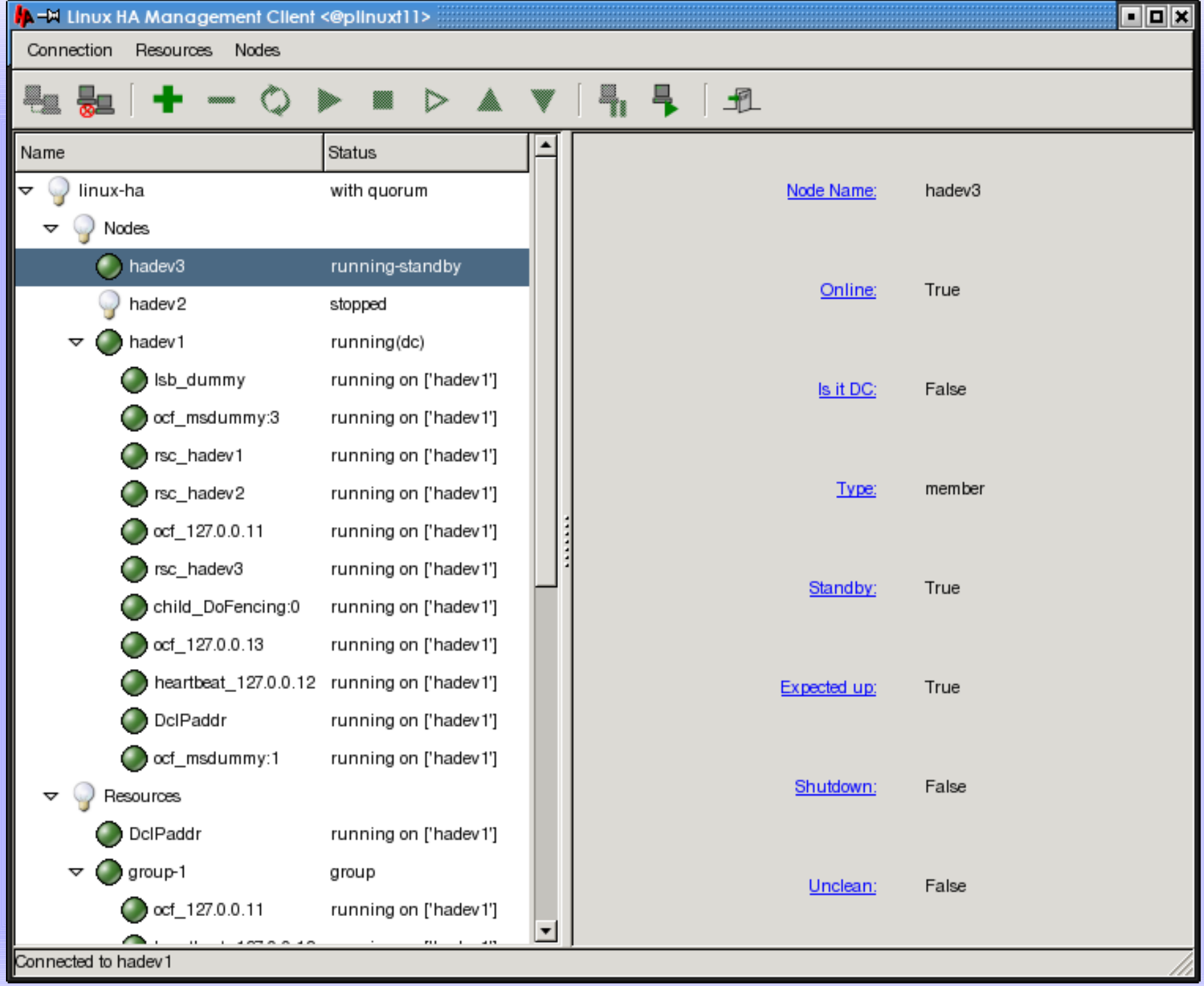
Using the Heartbeat GUI (hb_gui)

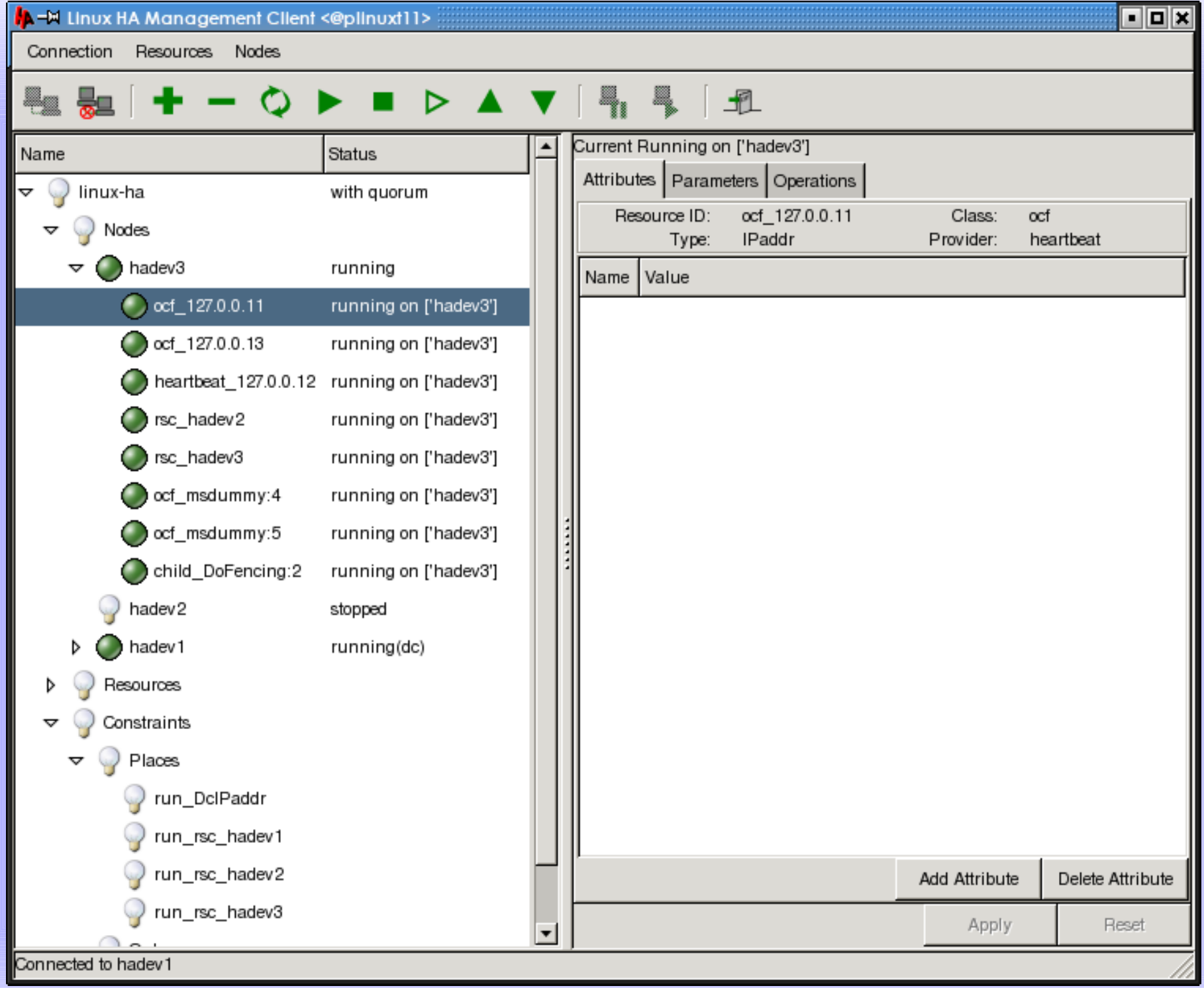
- hb_gui allows configuration and monitoring through the same interface
- It provides both node-centric and resource-centric views
- Although it supports a significant portion of what the CRM supports, it is a work-in-progress at this time, and does not yet allow for expressing the full power found in the CIB

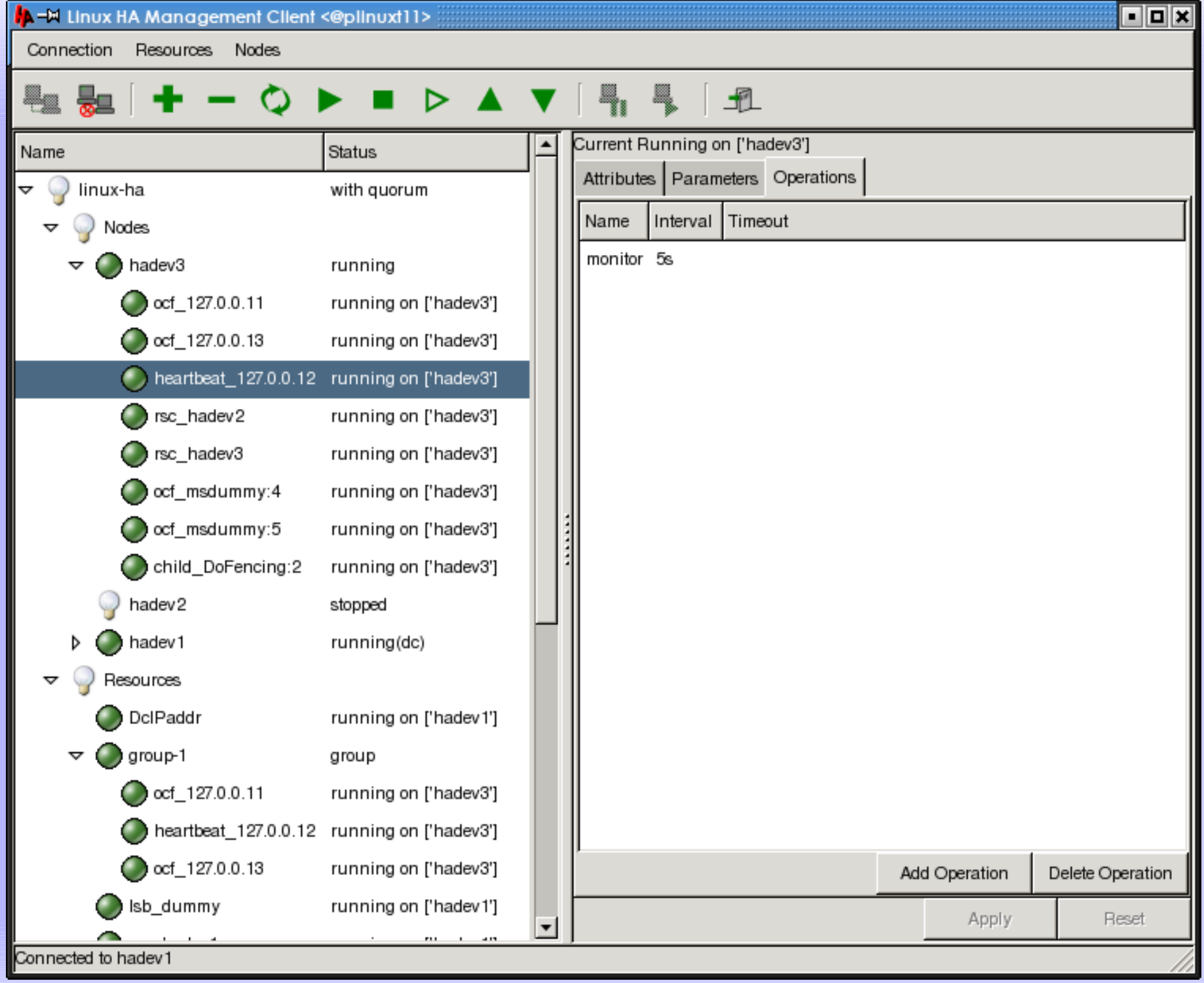

Linux HA Management Client <@plinuxt11>
Connection
Resources
Nodes
Name
Status
Login <@plinuxt11>
Server:
hadev1
User Name:
hacluster
Password:
OK
Cancel
Not Connected

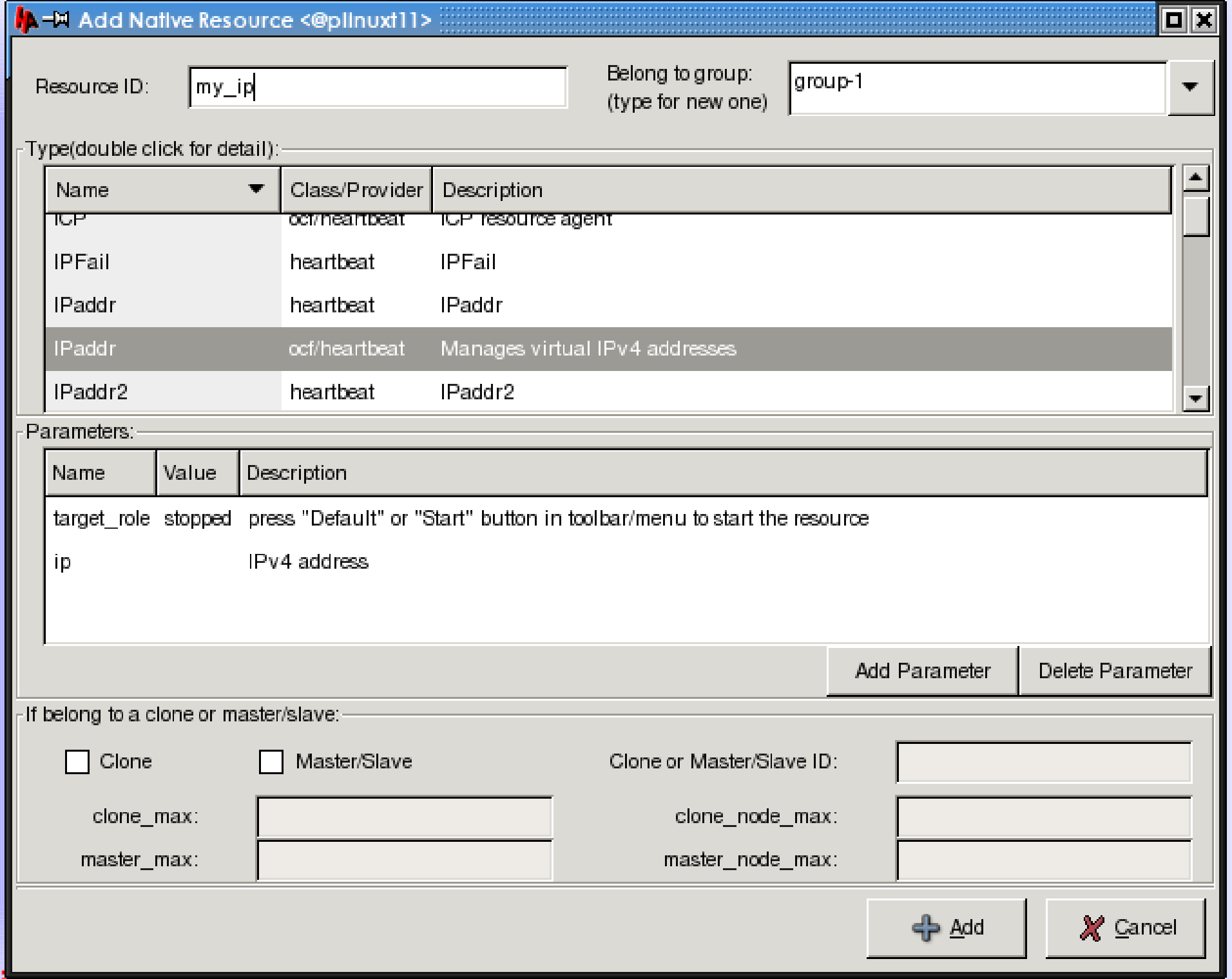

Linux HA Management Client <@plinuxf11>
Connection
Resources
Nodes
Name
Status
linux-ha
with quorum
Nodes
hadev3
running
ocf_127.0.0.11
running on ['hadev3']
ocf_127.0.0.13
running on ['hadev3']
heartbeat_127.0.0.12
running on ['hadev3']
rsc_hadev2
running on ['hadev3']
rsc_hadev3
running on ['hadev3']
ocf_msdummy:4
running on ['hadev3']
ocf_msdummy:5
running on ['hadev3']
child_DoFencing:2
running on ['hadev3']
hadev2
stopped
hadev1
running(dc)
Resources
Constraints
Places
run_DcIPaddr
run_rsc_hadev1
run_rsc_hadev2
run_rsc_hadev3
Version:
2.0.7
Debug Level:
1
UDP Port:
695
Keep Alive:
1
Warning Alive:
3
Dead Time:
5
Initial Dead Time:
10
Symmetric Cluster
Stonith Enabled
Transition Timeout:
10s
Resource Stickiness:
0
No Quorum Policy:
stop
Resource Failure Stickiness:
0
Apply
Reset
Connected to hadev1

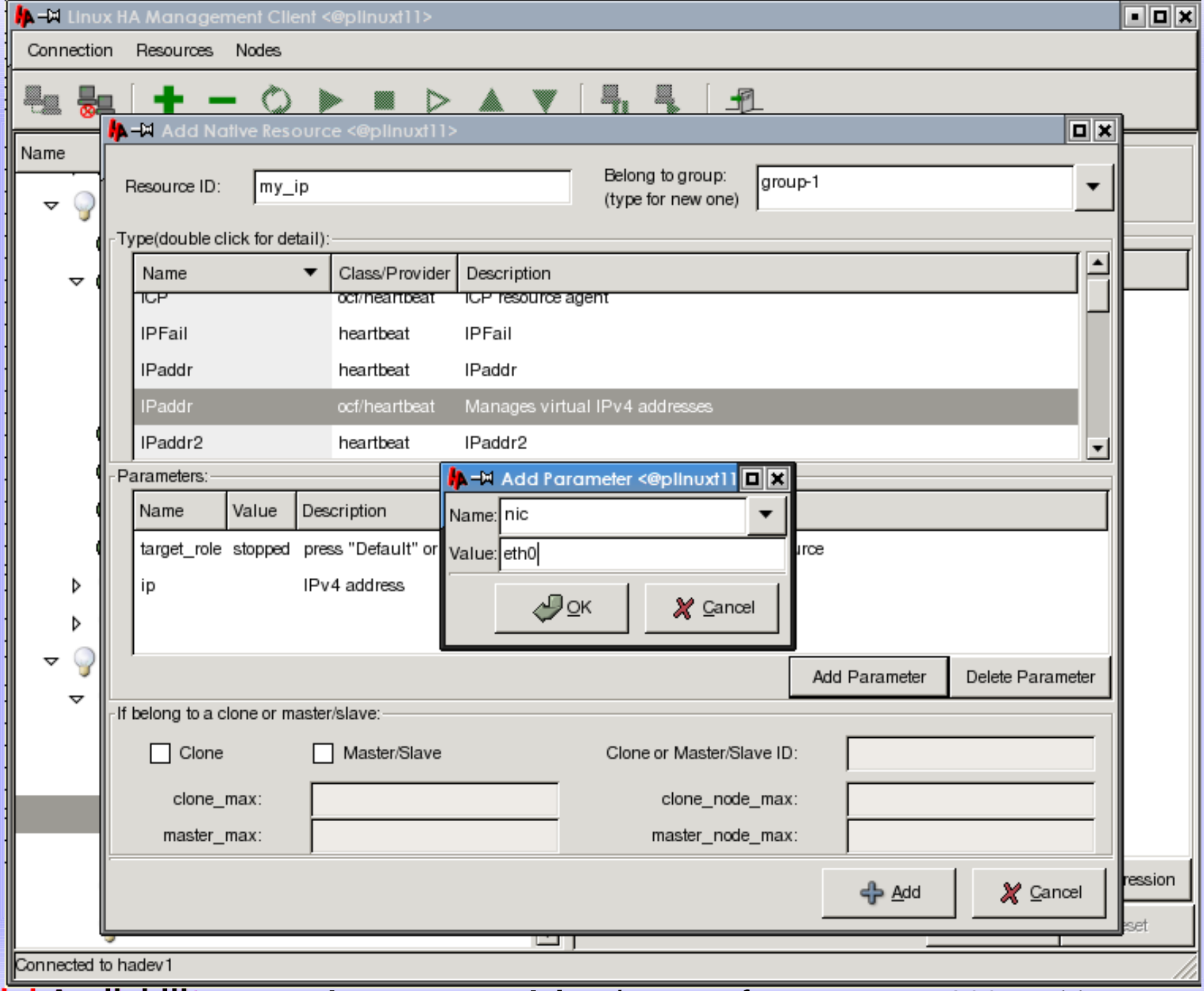

Linux HA Management Client <@plinux11>
Connection
Resources
Nodes
Name
Status
linux-ha
with quorum
Nodes
hadev3
running
ocf_127.0.0.11
running on ['hadev3']
ocf_127.0.0.13
running on ['hadev3']
heartbeat_127.0.0.12
running on ['hadev3']
rsc_hadev2
running on ['hadev3']
rsc_hadev3
running on ['hadev3']
ocf_msdummy:4
running on ['hadev3']
ocf_msdummy:5
running on ['hadev3']
child_DoFencing:2
running on ['hadev3']
hadev2
stopped
hadev1
running(dc)
Resources
Constraints
Places
run_DcIPaddr
run_rsc_hadev1
run_rsc_hadev2
run_rsc_hadev3
Node Name:
hadev3
Online:
True
Is it DC:
False
Type:
member
Standby:
False
Expected up:
True
Shutdown:
False
Unclean:
False
Connected to hadev1

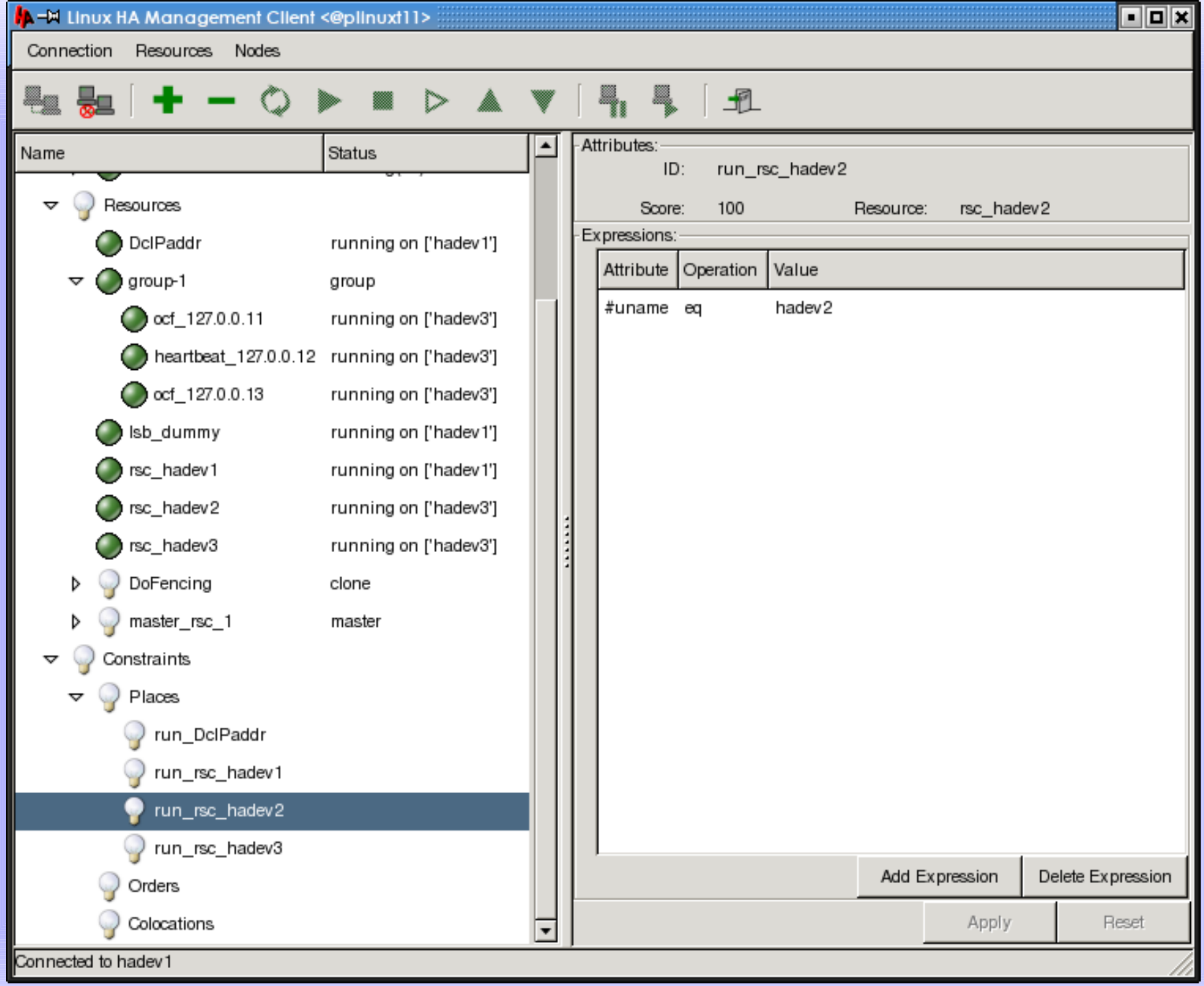

Linux HA Management Client <@plinuxt11>
Connection
Resources
Nodes
Name
Status
linux-ha
with quorum
Nodes
hadev3
running-standby
hadev2
stopped
hadev1
running(dc)
lsb_dummy
running on ['hadev1']
ocf_msdummy:3
running on ['hadev1']
rsc_hadev1
running on ['hadev1']
rsc_hadev2
running on ['hadev1']
ocf_127.0.0.11
running on ['hadev1']
rsc_hadev3
running on ['hadev1']
child_DoFencing:0
running on ['hadev1']
ocf_127.0.0.13
running on ['hadev1']
heartbeat_127.0.0.12
running on ['hadev1']
DcIPaddr
running on ['hadev1']
ocf_msdummy:1
running on ['hadev1']
Resources
DcIPaddr
running on ['hadev1']
group-1
group
ocf_127.0.0.11
running on ['hadev1']
Node Name:
hadev3
Online:
True
Is it DC:
False
Type:
member
Standby:
True
Expected up:
True
Shutdown:
False
Unclean:
False
Connected to hadev1

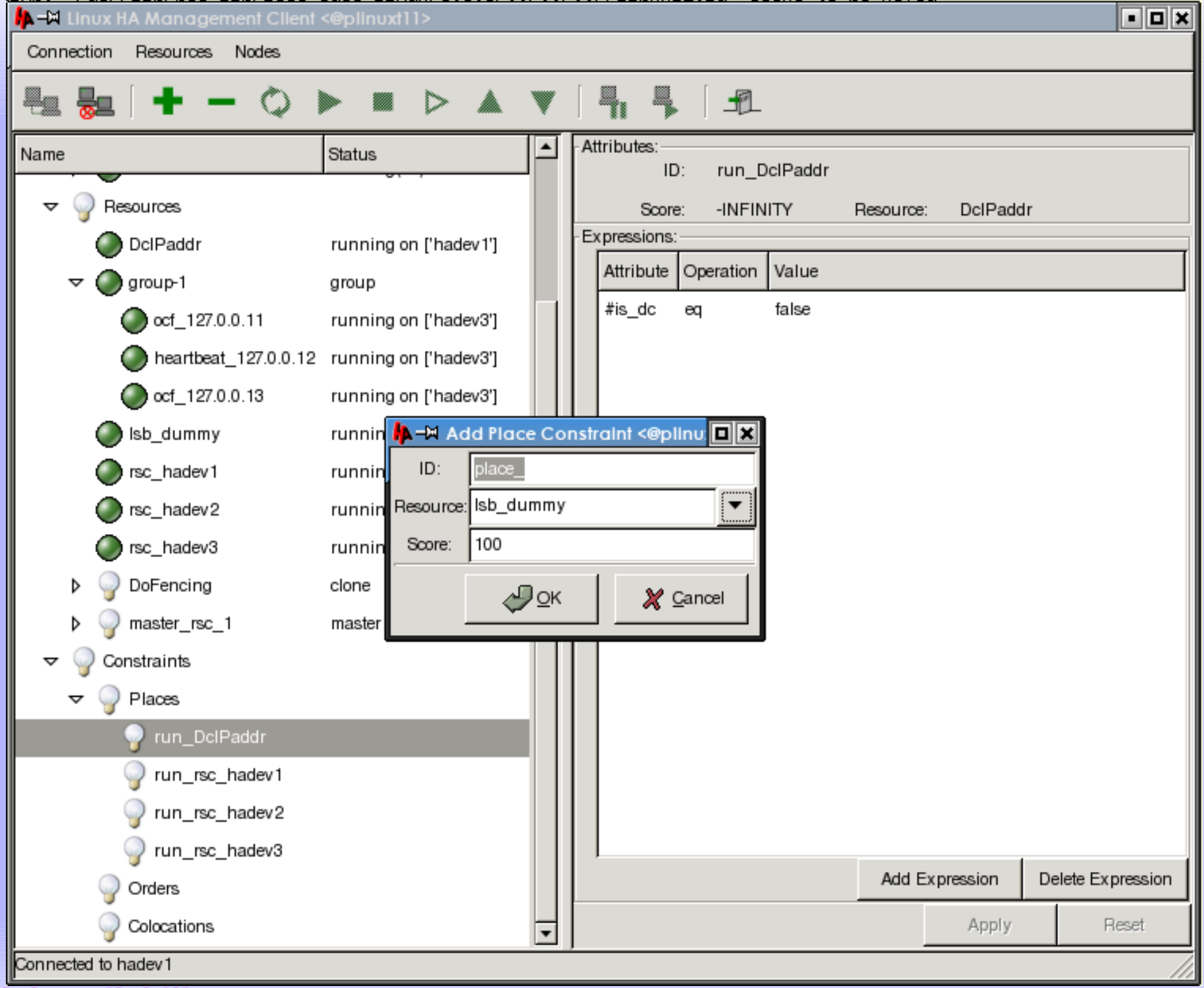

Linux HA Management Client <@plinux11>
Connection
Resources
Nodes
Name
Status
linux-ha
with quorum
Nodes
hadev3
running
ocf_127.0.0.11
running on ['hadev3']
ocf_127.0.0.13
running on ['hadev3']
heartbeat_127.0.0.12
running on ['hadev3']
rsc_hadev2
running on ['hadev3']
rsc_hadev3
running on ['hadev3']
ocf_msdummy:4
running on ['hadev3']
ocf_msdummy:5
running on ['hadev3']
child_DoFencing:2
running on ['hadev3']
hadev2
stopped
hadev1
running(dc)
Resources
Constraints
Places
run_DcIPaddr
run_rsc_hadev1
run_rsc_hadev2
run_rsc_hadev3
Current Running on ['hadev3']
Attributes
Parameters
Operations
Resource ID:
ocf_127.0.0.11
Class:
ocf
Type:
IPaddr
Provider:
heartbeat
Name
Value
Add Attribute
Delete Attribute
Apply
Reset
Connected to hadev1

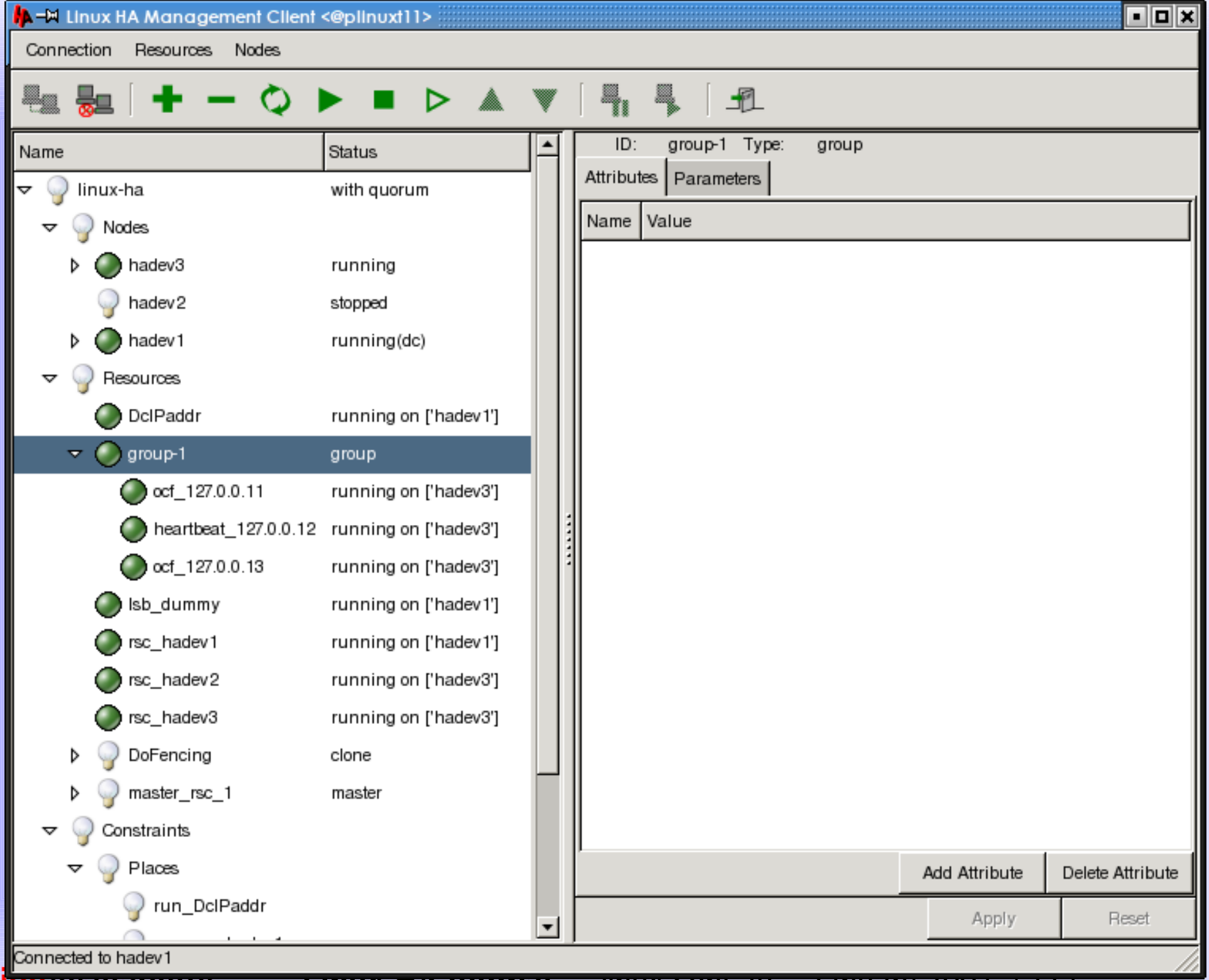

Linux HA Management Client <@plinuxf11>
Connection
Resources
Nodes
Name
Status
linux-ha
with quorum
Nodes
hadev3
running
ocf_127.0.0.11
running on ['hadev3']
ocf_127.0.0.13
running on ['hadev3']
heartbeat_127.0.0.12
running on ['hadev3']
rsc_hadev2
running on ['hadev3']
rsc_hadev3
running on ['hadev3']
ocf_msdummy:4
running on ['hadev3']
ocf_msdummy:5
running on ['hadev3']
child_DoFencing:2
running on ['hadev3']
hadev2
stopped
hadev1
running(dc)
Resources
DcIPaddr
running on ['hadev1']
group-1
group
ocf_127.0.0.11
running on ['hadev3']
heartbeat_127.0.0.12
running on ['hadev3']
ocf_127.0.0.13
running on ['hadev3']
lsb_dummy
running on ['hadev1']
Current Running on ['hadev3']
Attributes
Parameters
Operations
Name
Interval
Timeout
monitor
5s
Add Operation
Delete Operation
Apply
Reset
Connected to hadev1

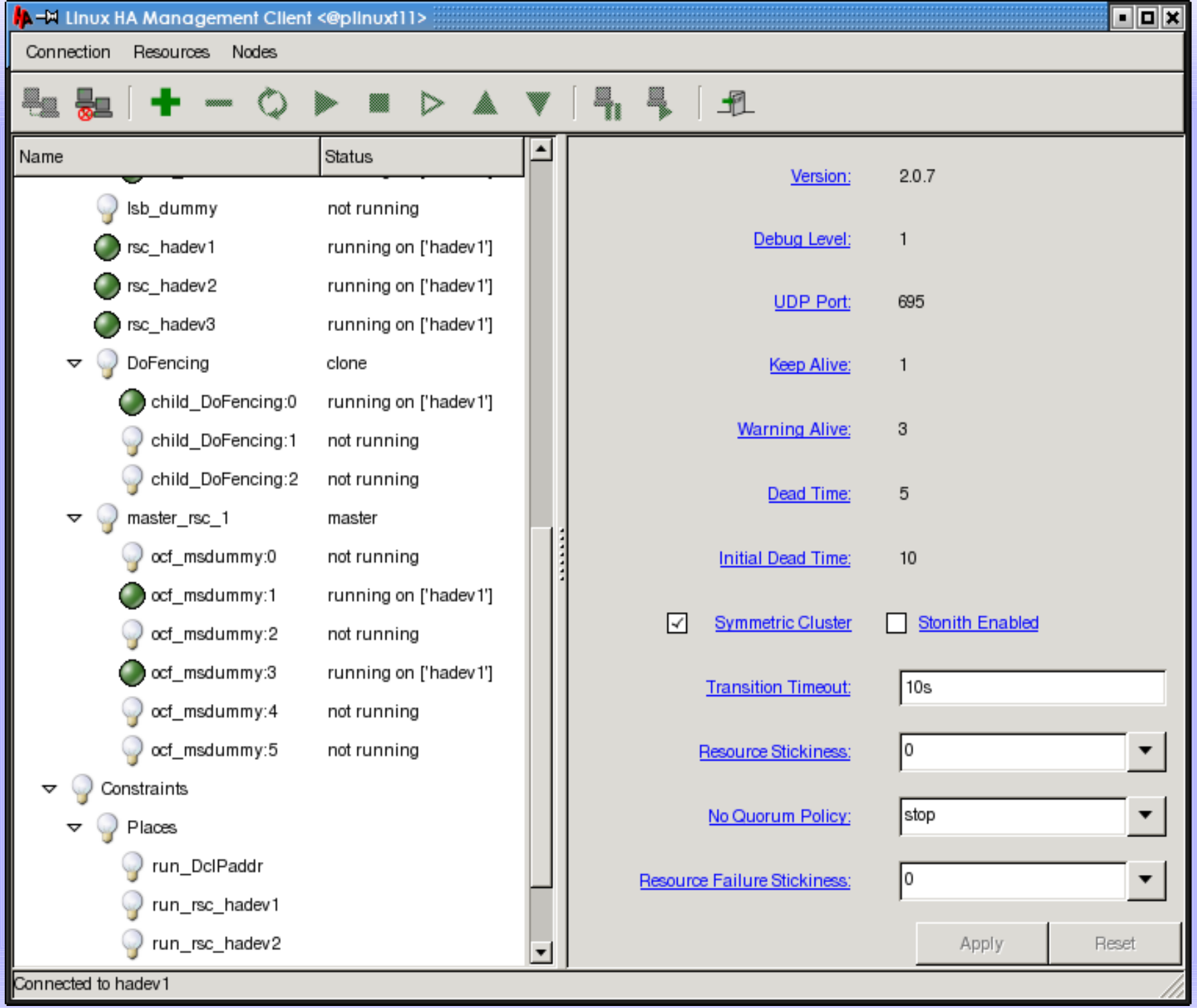

Add Native Resource <@plinux11>
Resource ID:
my_ip
Belong to group:
(type for new one)
group-1
Type(double click for detail):
Name
Class/Provider
Description
ICP
ocf/heartbeat
ICP resource agent
IPFail
heartbeat
IPFail
IPaddr
heartbeat
IPaddr
IPaddr
ocf/heartbeat
Manages virtual IPv4 addresses
IPaddr2
heartbeat
IPaddr2
Parameters:
Name
Value
Description
target_role
stopped
press "Default" or "Start" button in toolbar/menu to start the resource
ip
IPv4 address
Add Parameter
Delete Parameter
If belong to a clone or master/slave:
Clone
Master/Slave
Clone or Master/Slave ID:
clone_max:
clone_node_max:
master_max:
master_node_max:
Add
Cancel


Linux HA Management Client <@plinuxf11>
Connection
Resources
Nodes
Add Native Resource <@plinuxf11>
Resource ID:
my_ip
Belong to group:
(type for new one)
group-1
Type(double click for detail):
Name
Class/Provider
Description
IPFail
heartbeat
IPFail
IPaddr
heartbeat
IPaddr
IPaddr
ocf/heartbeat
Manages virtual IPv4 addresses
IPaddr2
heartbeat
IPaddr2
Parameters:
Name
Value
Description
target_role
stopped
press "Default" or
ip
IPv4 address
Add Parameter
Name:
nic
Value:
eth0
OK
Cancel
Add Parameter
Delete Parameter
If belong to a clone or master/slave:
Clone
Master/Slave
Clone or Master/Slave ID:
clone_max:
clone_node_max:
master_max:
master_node_max:
Add
Cancel
Connected to hadev1


Linux HA Management Client <@plinux11>
Connection
Resources
Nodes
Name
Status
Resources
DclPaddr running on ['hadev1']
group-1 group
ocf_127.0.0.11 running on ['hadev3']
heartbeat_127.0.0.12 running on ['hadev3']
ocf_127.0.0.13 running on ['hadev3']
lsb_dummy running on ['hadev1']
rsc_hadev1 running on ['hadev1']
rsc_hadev2 running on ['hadev3']
rsc_hadev3 running on ['hadev3']
DoFencing clone
master_rsc_1 master
Constraints
Places
run_DclPaddr
run_rsc_hadev1
run_rsc_hadev2
run_rsc_hadev3
Orders
Colocations
Attributes:
ID: run_rsc_hadev2
Score: 100
Resource: rsc_hadev2
Expressions:
Attribute
Operation
Value
#uname eq hadev2
Add Expression
Delete Expression
Apply
Reset
Connected to hadev1


Linux HA Management Client <@plinux11>
Connection
Resources
Nodes
Name
Status
Resources
DclPaddr
running on ['hadev1']
group-1
group
ocf_127.0.0.11
running on ['hadev3']
heartbeat_127.0.0.12
running on ['hadev3']
ocf_127.0.0.13
running on ['hadev3']
lsb_dummy
rsc_hadev1
rsc_hadev2
rsc_hadev3
DoFencing
clone
master_rsc_1
master
Constraints
Places
run_DclPaddr
run_rsc_hadev1
run_rsc_hadev2
run_rsc_hadev3
Orders
Colocations
Attributes:
ID: run_DclPaddr
Score: -INFINITY
Resource: DclPaddr
Expressions:
Attribute
Operation
Value
#is_dc
eq
false
Add Place Constraint <@plinu
ID:
place_
Resource:
lsb_dummy
Score:
100
OK
Cancel
Add Expression
Delete Expression
Apply
Reset
Connected to hadev1

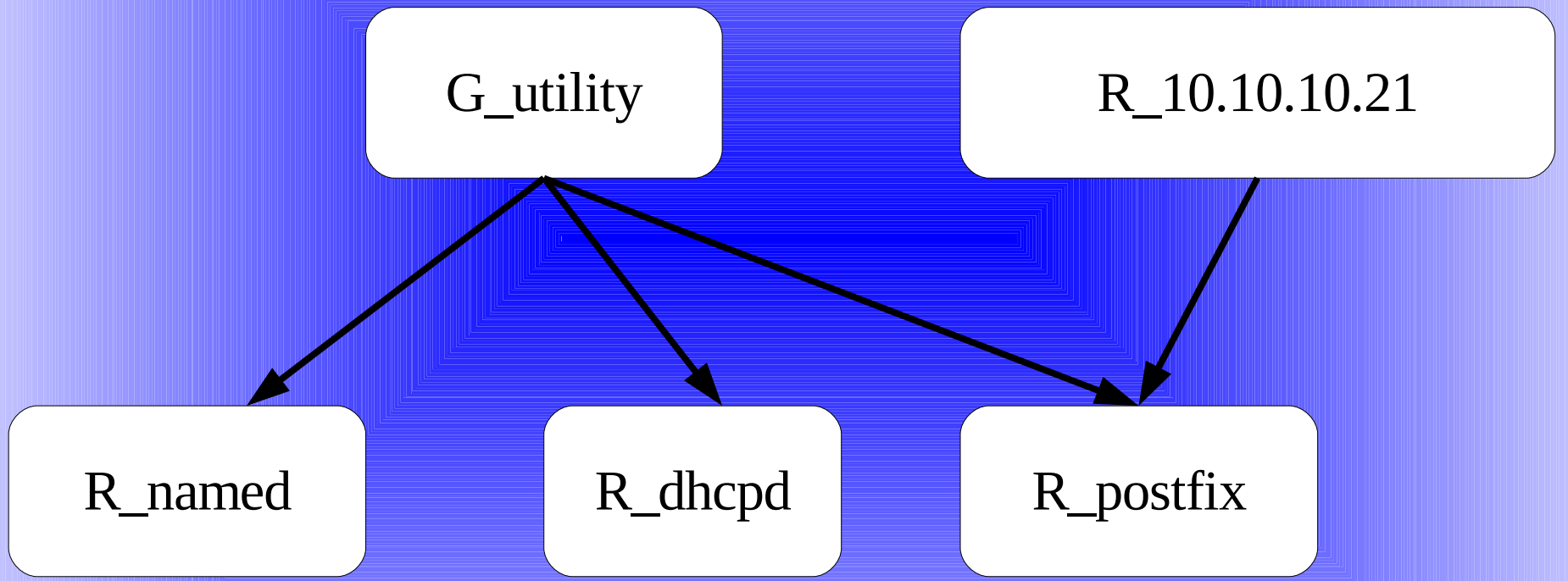

Linux HA Management Client <@plinuxt11>
Connection
Resources
Nodes
Name
Status
linux-ha
with quorum
Nodes
hadev3
running
hadev2
stopped
hadev1
running(dc)
Resources
DcIPaddr
running on ['hadev1']
group-1
group
ocf_127.0.0.11
running on ['hadev3']
heartbeat_127.0.0.12
running on ['hadev3']
ocf_127.0.0.13
running on ['hadev3']
lsb_dummy
running on ['hadev1']
rsc_hadev1
running on ['hadev1']
rsc_hadev2
running on ['hadev3']
rsc_hadev3
running on ['hadev3']
DoFencing
clone
master_rsc_1
master
Constraints
Places
run_DcIPaddr
ID: group-1 Type: group
Attributes
Parameters
Name
Value
Add Attribute
Delete Attribute
Apply
Reset
Connected to hadev1


Linux HA Management Client <@plinux11>
Connection
Resources
Nodes
Name
Status
lsb_dummy not running
rsc_hadev1 running on ['hadev1']
rsc_hadev2 running on ['hadev1']
rsc_hadev3 running on ['hadev1']
DoFencing clone
child_DoFencing:0 running on ['hadev1']
child_DoFencing:1 not running
child_DoFencing:2 not running
master_rsc_1 master
ocf_msdummy:0 not running
ocf_msdummy:1 running on ['hadev1']
ocf_msdummy:2 not running
ocf_msdummy:3 running on ['hadev1']
ocf_msdummy:4 not running
ocf_msdummy:5 not running
Constraints
Places
run_DcIPaddr
run_rsc_hadev1
run_rsc_hadev2
Version: 2.0.7
Debug Level: 1
UDP Port: 695
Keep Alive: 1
Warning Alive: 3
Dead Time: 5
Initial Dead Time: 10
Symmetric Cluster
Stonith Enabled
Transition Timeout: 10s
Resource Stickiness: 0
No Quorum Policy: stop
Resource Failure Stickiness: 0
Apply
Reset
Connected to hadev1

Creating Detailed Ordering Constraints

- Ordering constraints can apply between any two resource objects – primitive, group or clone
  - The main kind of ordering constraint that is used is start_after
    - There is also a start_before constraint
    - There may also be stop_after, and stop_before constraints :-D
  - Although these others provide flexibility, they're not commonly used
- Ordering constraints can make start and stop actions complete faster than groups

Sample Ordering Constraint Graph



Sample Ordering Constraint XML

```
<rsc_order id="O_dhcpd"
 from="R_dhcpd" type="after"        to="G_utility"/>

<rsc_order id="O_named"
     from="R_named" type="after"    to="G_utility"/>

<rsc_order id="O_postfix"
     from="R_postfix" type="after" to="G_utility"/>

<rsc_order id="O_postfix_ip"
     from="R_postfix" type="after" to="R_10.10.10.21"/>
```

Part IV

- More sophisticated usages
- Writing Resource Agents

Co-location Constraints

- The XML DTD permits both mandatory and optional co-location constraints
- As of 2.0.8, both mandatory co-location constraints are supported.
- As of 2.0.8, co-location constraints are fully asymmetric.

Sample Co-location Constraints

```
<rsc_co-location id="C_10.10.10.21"
from="R_10.10.10.21" to="G_utility" score="INFINITY"/>

<rsc_co-location id="C_postfix"
from="R_postfix" to="G_utility" score="INFINITY"/>

<rsc_co-location id="C_dhcpd"
from="R_dhcpd" to="G_utility" score="INFINITY"/>

<rsc_co-location id="C_named"
from="R_named" to="G_utility" score="INFINITY"/>
```

Writing and testing OCF Resource Agents

- If you have an init script for the resource available under a suitable license, start with that
- If possible, make your resource agent reusable by others.
  - Avoid things that are distribution-specific
  - Avoid hard-coding local conventions (use parameters)
  - Think generally

Writing and testing OCF Resource Agents - Methodology

- Locate init script to use as template (if any)
  - Your final script can serve dual duty as an LSB init script and an OCF Resource Agent with a little effort
- Decide what parameters you want to pass the agent
- Support these actions in the script:
  - start, stop, status, monitor, meta-data, validate-all
- Test the script manually
- Test the script with Andrew's test tool (ocf-tester)

OCF Resource Agents – Parameters

- Decide what parameters your resource agent needs to have configurable. Examples:
  - location of data for service
  - Direct configuration information (IP address, etc.)
  - location of configuration file (if configurable)
  - location of binaries
  - user id to run as
  - other parameters to issue when starting
  - It's better to parse configuration files rather than duplicating configuration information in parameters

OCF Resource Agents – Parameters

- Choose reasonably intuitive parameter names like 'ip' or 'configfile', etc.
- Whatever names you choose, the OCF standard prepends OCF_RESKEY_ to them. ip becomes OCF_RESKEY_ip, etc.
- Provide reasonable defaults – if possible
  - If you do this for all parameters, and you support the status operation (with LSB status exit codes), then your script can also be used as an LSB init script.

OCF RAs – Return Codes

- Proper **monitor** return codes:
  - **0** running
  - **7** stopped (follows the LSB convention)
  - **other** something bad happened
- If resource is started, start operation must succeed (return code 0)
- If resource is stopped, stop operation must succeed (return code 0)
- **status** return codes are different from **monitor** return codes (to make them LSB compatible...)

OCF meta-data and validate-all

- **validate-all** checks the parameters supplied and exits with 0 if they're correct, and non-zero (LSB conventions) if they can be determined to be incorrect
- **meta-data** operation just delivers a fixed blob of XML to standard output describing this resource agent, and exits 0. The meta-data operation replaces the structured comments provided for by the LSB. This meta-data is used by the GUI and is useful for humans doing configuration by hand.

OCF stop, start, monitor actions

- **start** initiates or activates the resource.
- **stop** deactivates, stops, or terminates the resource
- **monitor** examines the resource to see if it is running correctly
  - The monitor action can implement different levels of checking quality or difficulty
  - The better the quality of monitoring, the more likely service outages are to be noticed and recovered from
  - The desired level(s) of checking can then be selected by the administrator through the CIB configuration for the monitor action.

OCF Meta-data example

```
<?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">

<resource-agent name="IPaddr">
<version>1.0</version>

<longdesc lang="en">
IPaddr manages aliased IP addresses.  It will add an IP alias
when started, and remove it when stopped.
</longdesc>

<shortdesc lang="en">Manage virtual IPv4
addresses</shortdesc>
```

OCF Meta-data example

<parameters>

<parameter name="ip" unique="1" required="1">
<longdesc lang="en">
The IPv4 address to be configured in dotted quad notation, for example "192.168.1.1".
</longdesc>

<shortdesc lang="en">IPv4 address</shortdesc>
<content type="string" default=""/>
</parameter>
</parameters>

OCF Meta-data example

```
<actions>

<action name="start"   timeout="90s" />
<action name="stop"    timeout="100s" />
<action name="monitor" depth="10"  timeout="20s"
            interval="5s" start-delay="1s" />
<action name="validate-all"  timeout="30s" />
<action name="meta-data"  timeout="5s" />

</actions>


</resource-agent>
</xml>
```

OCF RAs – Manual Testing - 1

- Install relevant software, and create test data to go with it
- Test *at least* this order of actions:
  - monitor (must exit with return code 7 -- stopped)
  - start (should succeed)
  - start (must succeed)
  - monitor (must succeed)
  - stop (must succeed)
  - stop (must succeed)
  - monitor (must exit with return code 7 -- stopped)

OCF RAs – Manual Testing - 2

- After starting it, try to impair the resource somehow
  - kill a daemon, or ifdown an interface, remove a database or config file, or other action harmful to the resource in question
- Make sure 'monitor' reports failure
- Use this experience to improve your 'monitor' testing
- Heartbeat's can only recover from things your monitor action reports

OCF RA testing with ocf-tester

- **ocf_tester** provides basic testing of OCF Resource Agents for certain key conditions heartbeat is especially concerned with.

```
ocf-tester -v -n my_ip_rsc    \
   -o ip=127.0.10.1           \
   -o netmask=255.255.0.0     \
   /usr/lib/ocf/resource.d/heartbeat/IPaddr
```

  - **-v** verbose
  - **-n** resource id (name)
  - **-o** resource option (parameter)

Introducing node attributes

- Nodes can be assigned arbitrary attributes, which can then be used in resource location rules

```
<node id="uuid1" uname="nodeA" type="normal">
  <instance_attributes id="uuid1:attrs">
    <attributes>
      <nvpair id="uuid1:installed_ram"
              name="installed_ram" value="1024"/>
      <nvpair id="uuid1:pingcount”
              name="pingcount" value="2"/>
    </attributes>
  </instance_attributes>
</node>
```

Using pingd to fail over on loss of network connectivity

- **`pingd`** is a daemon which sets node attributes in the CIB based on how many different destinations are reachable from the current node.
- To use **`pingd`**:
  - Direct heartbeat to ping your routers or whatever addresses you've selected using the **`ping`** or **`ping_group`** directives
  - Configure pingd to run on whatever nodes you wish specifying the attribute value you want, and the values to set into it
  - Incorporate these attributes into your CIB location contstraints

Starting pingd as an OCF clone resource (1/2)

```
<clone id="pingd">
  <instance_attributes id="pingd">
    <meta_attributes>
      <nvpair id="pingd-clone_max"
              name="clone_max" value="10"/>
      <nvpair id="pingd-clone_node_max"
              name="clone_node_max" value="1"/>
      <nvpair id="pingd-dampen"
              name="dampen" value="5s"/>
      <nvpair id="pingd-multiplier"
              name="multiplier" value="100"/>
    </meta_attributes>
  </instance_attributes>
```

Starting pingd as an OCF clone resource (2/2)

```
 <primitive id="pingd-child" provider="heartbeat"
            class="OCF" type="pingd">
  <operations>
     <op id="pingd-child-monitor"
         name="monitor" interval="20s"
         timeout="40s"prereq="nothing"/>
     <op id="pingd-child-start" name="start"
         prereq="nothing"/>
   </operations>
 </primitive>
</clone>
```

Starting pingd from ha.cf

- Insert something similar to this into your ha.cf files:

```
respawn hacluster /usr/lib/heartbeat/pingd -m 100 -d 5s
```

  - **-m**: multiplier factor for number of ping nodes
  - **-d**: hysteresis (settling) time delay
- This example sets the attribute '**pingd**' to 100 times the number of ping nodes reachable from the current machine, and delays 5 seconds before modifying the **pingd** attribute in the CIB

See also:

http://www.linux-ha.org/ha.cf/PingDirective
and **http://www.linux-ha.org/v2/faq/pingd**

Using pingd attributes in rules

- Previous examples defaulted the attribute value to '**pingd**'

```
<rsc_location id="my_resource:connected"
              rsc="my_resource">
  <rule id="my_resource:connected:rule"
        score_attribute="pingd" >
    <expression id="my_resource:connected:expr:defined"
                attribute="pingd"
                operation="defined"/>
  </rule>
</rsc_location>
```

- This rule causes the value of the node attribute **pingd** to be added to the value of every node on which its defined
- Previous examples set it to 100*ping_count

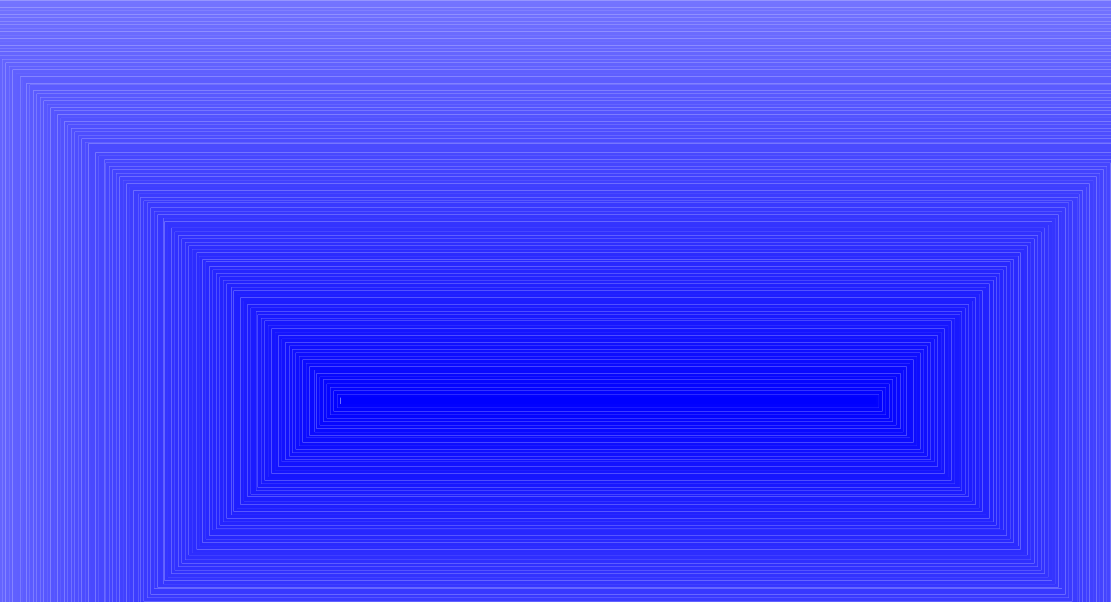
Failing over on arbitrary conditions

- **pingd** is a worked example of how to fail over on arbitrary conditions
- **attrd_updater** is what **pingd** uses to modify the CIB
- **attrd** implements the idea of hysteresis in setting values into the CIB – allowing things to settle out into stable configurations before failing over – to avoid false failovers
- **pingd** asks heartbeat to notify it when ping nodes come and go. When they do, it invokes **attrd_updater** to make the change, and **attrd** updates the CIB – after a delay
- You can use **attrd_updater** yourself to do this for any condition you can observe

Using attrd_updater

- **`attrd_updater`** command line arguments:
  - **-n** *name* name of attribute to set
  - **-v** *value* value to set attribute *name* to
  - **-s** *attribute-set* which attribute set does *name* reside in
  - **-d** *dampen time* time delay before updating CIB
- To use **`attrd`**:
  - Write code to observe something
  - Invoke **`attrd_updater`** to update some attribute value when it changes
  - Write CIB rules to use the attribute value you set

Master/Slave Resources

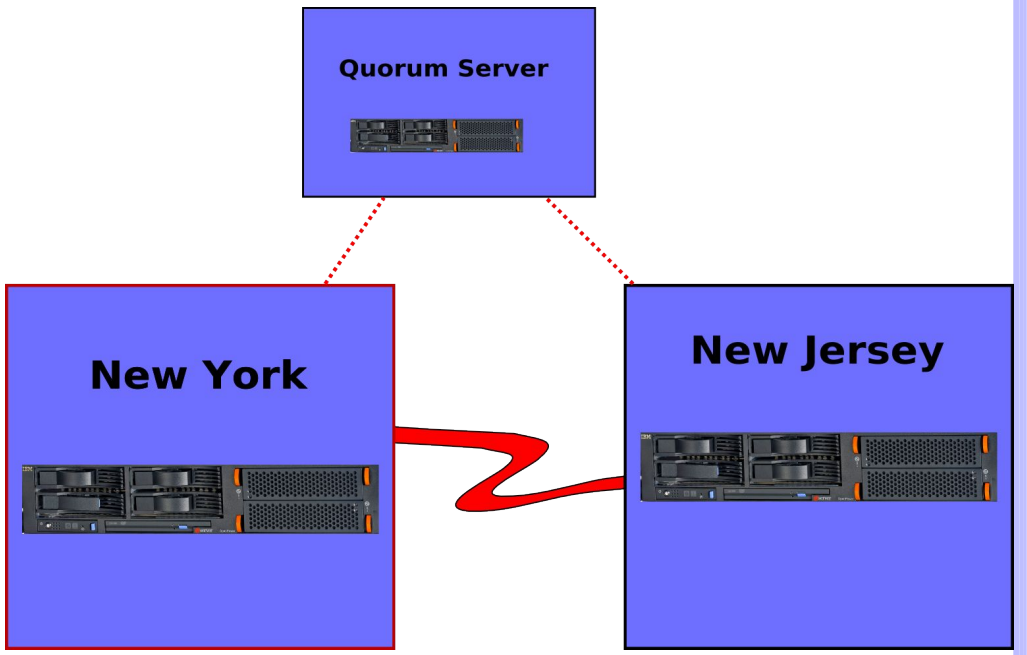


Split-site (“stretch”) clusters

- Geographic-scale communications are never as reliable as local communications
- Fencing techniques (STONITH, SCSI reserve) all require highly reliable communications, don't work remotely
- Split-site clusters cannot rely on fencing in most cases
- Quorum without fencing must be used instead
- Two-site quorum without fencing is problematic
- Linux-HA introduces a quorum server to solve this problem

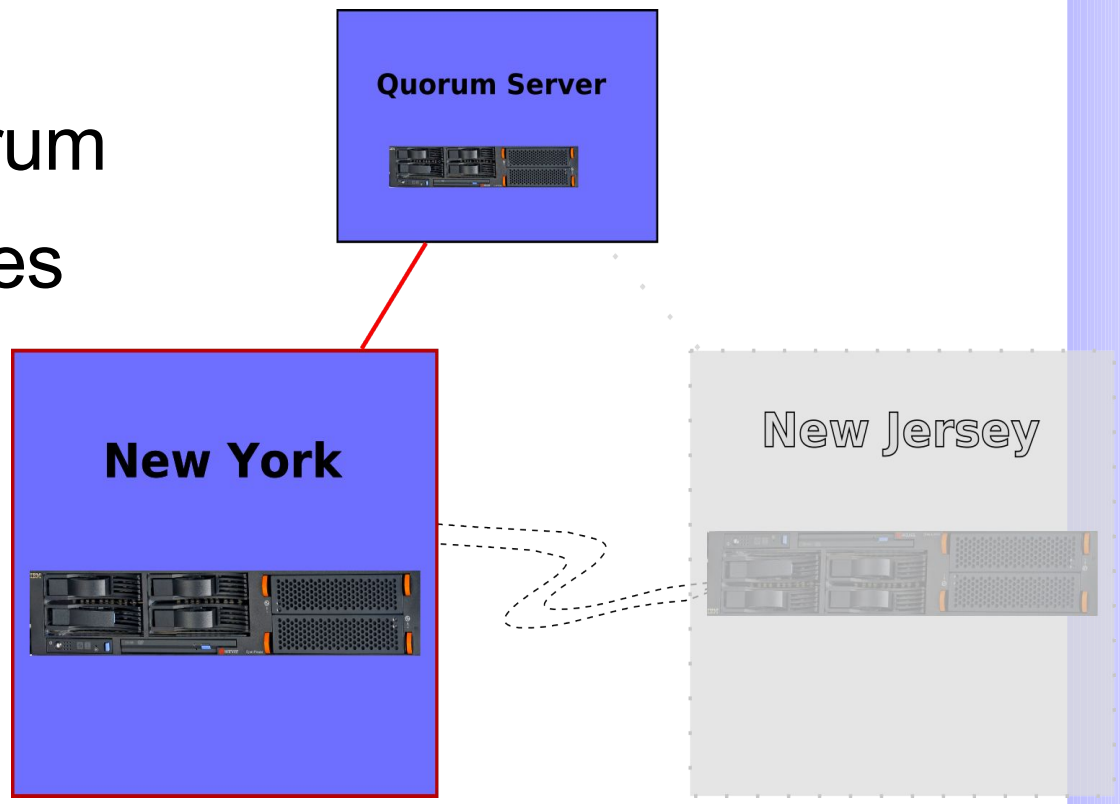
Quorum Server basics

- Quorum Server provides an extra quorum vote
- Quorum server not a cluster member
- Quorum server does not require special networking
- Reliability of quorum server and links to it are important



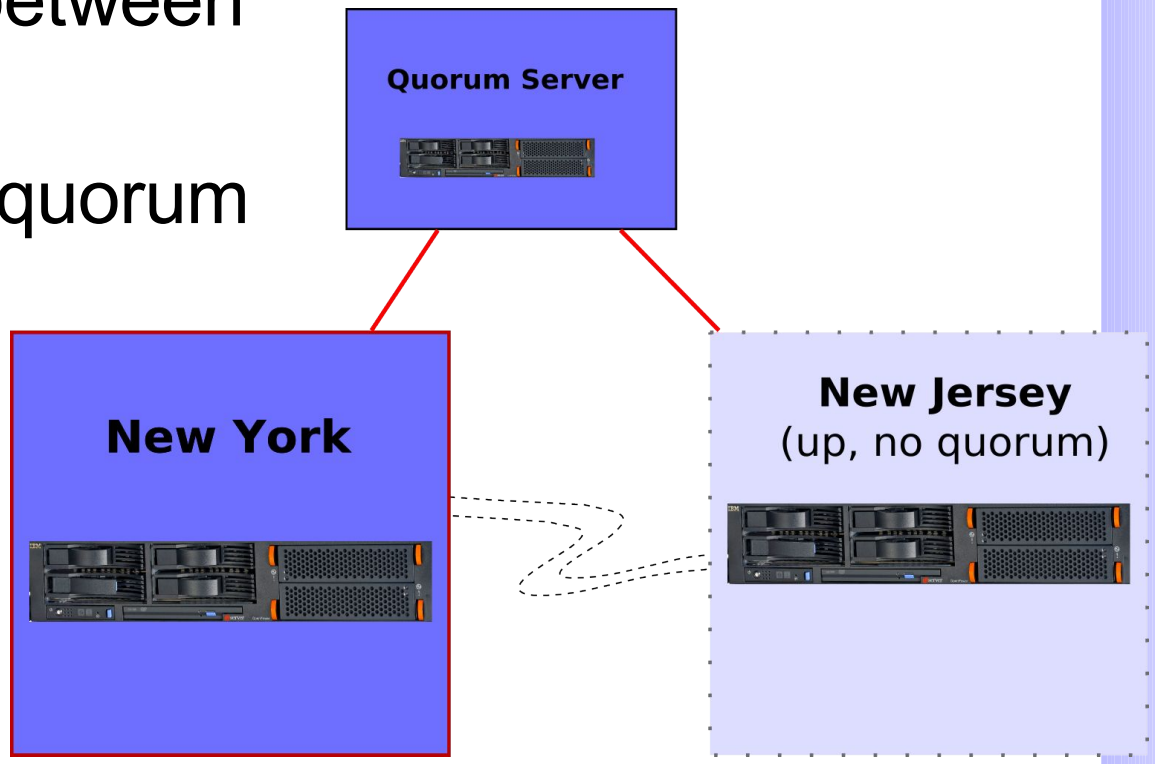
Quorum Server: Single Site failure

- “New Jersey” is down
- Quorum server supplies extra quorum vote
- Cluster retains quorum
- “New York” continues to provide service



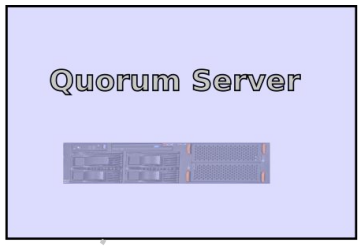
Quorum Server prevents Split-Brain

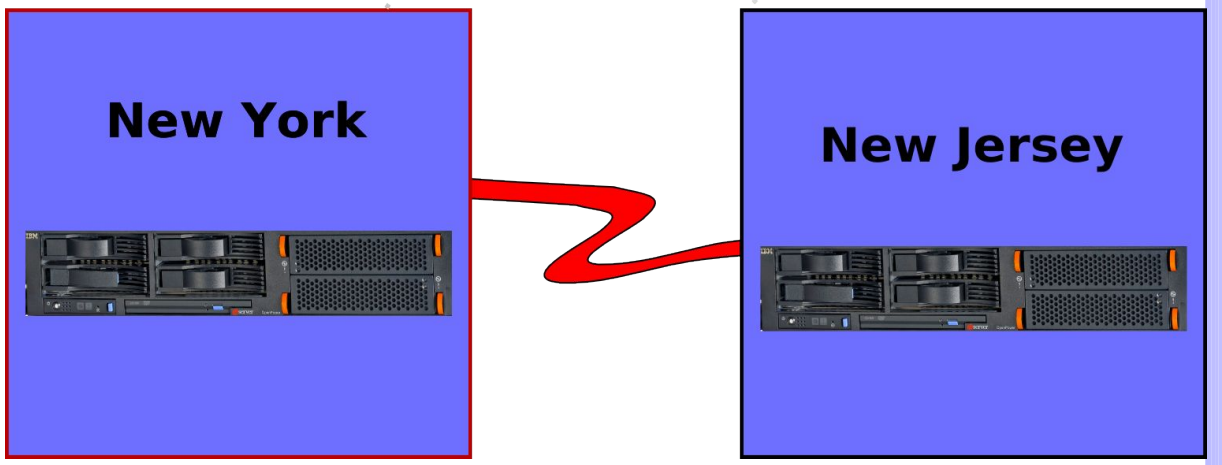
- Communications between sites goes down
- Both sites contact quorum server
- Quorum server gives quorum to New York ***ONLY***
- New Jersey site: no quorum -> no services



Quorum Server Not a SPOF

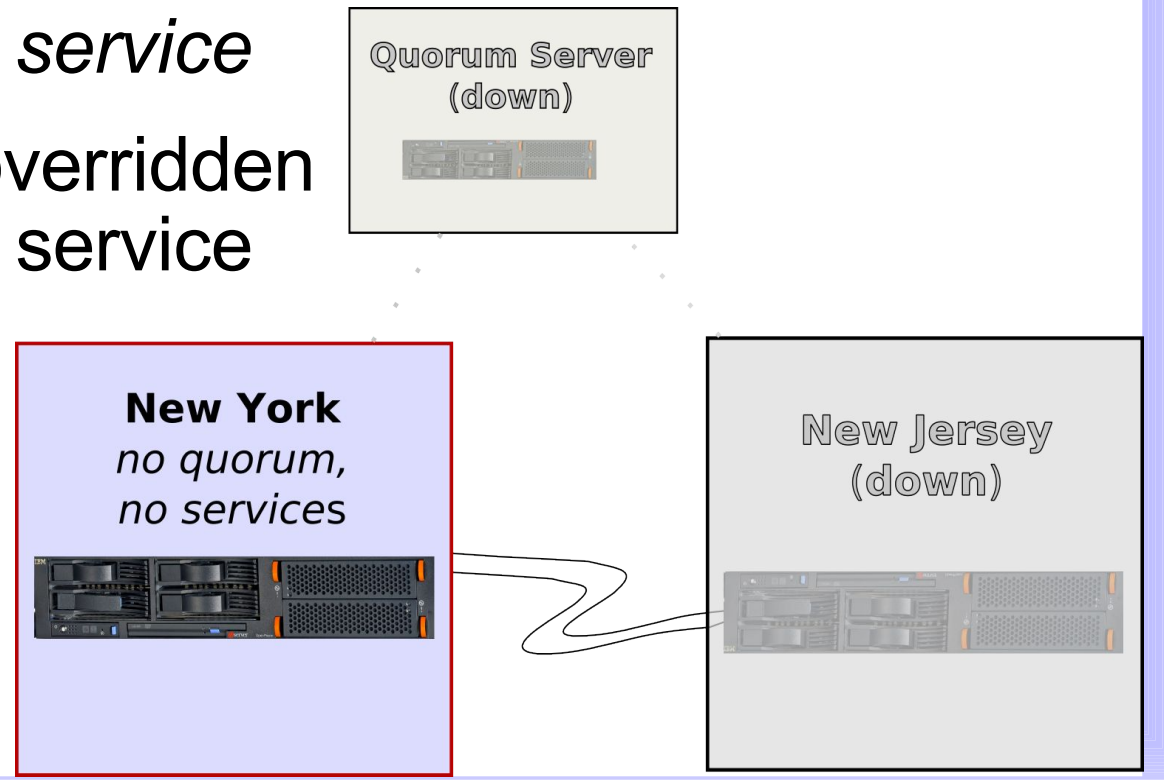
- Quorum server goes down
- Cluster retains quorum
- Services are still supplied
- Service is uninterrupted





Multiple Failures *Can* Lead To No Service

- Quorum server: down
- New Jersey site: down
- New York site: up
  no quorum => no service
- Quorum can be overridden manually to force service at New York



Time Based Configuration Rules

- The CRM can be given different rules for different periods of time – by the hour, day of week, etc.
- These can either be default rule parameters or rule parameters for specific resources
- The most common and obvious use of these are to allow "failback" only during certain times when workload is expected to be light
- The concept is quite general and can be used for virtually any set of **<attributes>** in the CIB
- start and end times follow the ISO8601 standard
- **<date_spec>** notation is cron-like

Allowing fail-back of an IP address only on weekends

```
<primitive id="my_ip" provider="heartbeat"
           class="OCF" type="IPaddr">
   <instance_attributes id="my_ip:weekend_override" score="100">
     <rule id="my_ip:failover" boolean_op="and">
       <date_expression id="my_ip:days" operation="date_spec">
         <date_spec id="my_ip:days" weekdays="6-7"/>
       </date_expression>
     </rule>
     <meta_attributes>
       <nvpair id="sat-sun-sticky" name="resource_stickiness"
                                   value="0"/>
     </meta_attributes>
   </instance_attributes>
   <instance_attributes id="my_ip" score="10">
      <meta_attributes>
        <nvpair id="default-sticky" name="resource_stickiness"
                                    value="INFINITY"/>
      </meta_attributes>
   </instance_attributes>
</primitive>
```

Setting default_resource_stickiness to default to fail back on weekends

```
<crm_config >
 <cluster_property_set id="weekend_override" score="100">
  <rule id="my_ip:failover" boolean_op="and">
    <date_expression id="my_ip:days" operation="date_spec">
      <date_spec id="my_ip:days" weekdays="6-7"/>
    </date_expression>
  </rule>
  <attributes>
    <nvpair id="sat-sun-stick" name="default_resource_stickiness"
                               value="0"/>
  </attributes>
 </cluster_property_set>
 <cluster_property_set id="default_cluster_properties" score="10">
  <attributes>
   <nvpair id="default-sticky" name="default_resource_stickiness"
                               value="INFINITY"/>
  </attributes>
 </cluster_property_set>
   ...
</crm_config>
```

More about Time-Based rules

- `http://linux-ha.org/v2/faq/time_based_failback`
- `http://linux-ha.org/ClusterResourceManager/DTD1.0/Annotated#date_expression`
- `http://en.wikipedia.org/wiki/ISO_8601`
- **Time-based rules can be sensitive to the phase of the moon (for implementing werewolf HA ;-))**

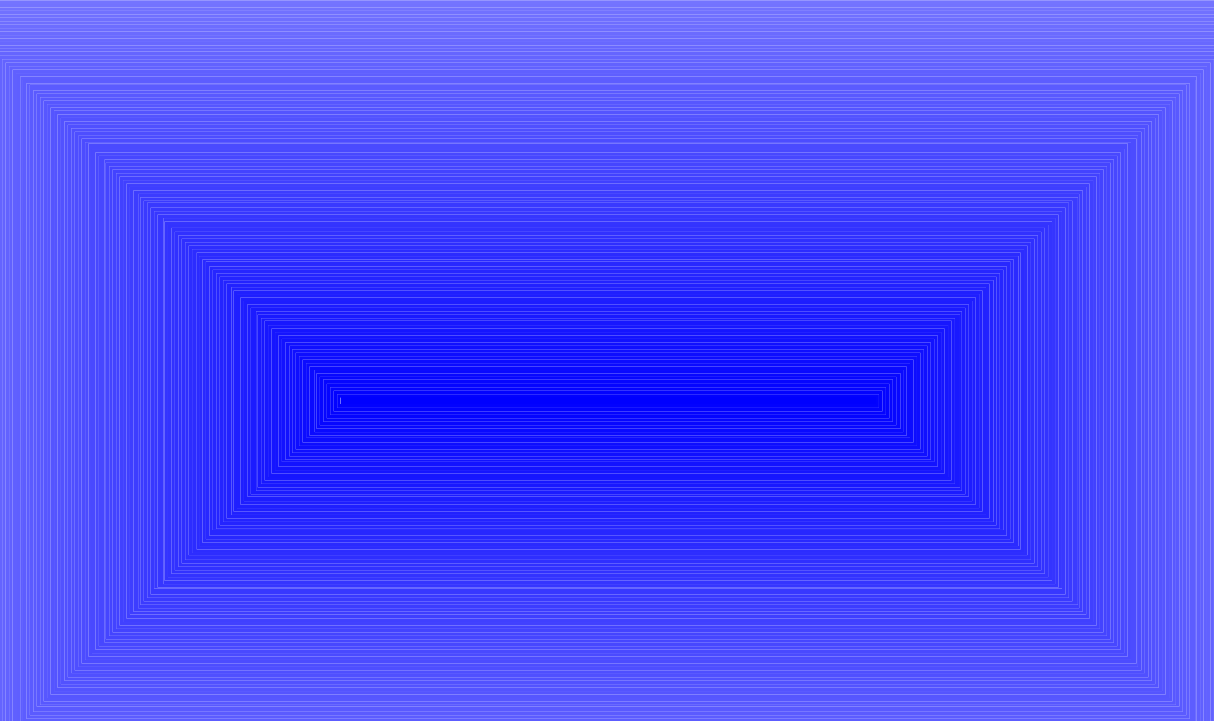
References

- http://linux-ha.org/
- http://linux-ha.org/download/
- http://linux-ha.org/SuccessStories
- http://linux-ha.org/Certifications
- http://linux-ha.org/ClusterResourceManager/DTD1.0/Annotated

Legal Statements

- IBM is a trademark of International Business Machines Corporation.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.
- This work represents the views of the author and does not necessarily reflect the views of the IBM Corporation.

Backup Slides



IPaddr resource Agent

- Class: **OCF**
- Parameters:
  - ***ip*** – IP address to bring up
  - ***nic*** – NIC to bring address up on (*optional*)
  - ***cidr_netmask*** – netmask for ip in CIDR form *(optional)*
  - ***broadcast*** – broadcast address *(optional)*
- If you don't specify **nic**, then heartbeat will figure out which interface serves the subnet that **ip** is on – which is quite handy. The same is true for cidr_netmask.

Filesystem resource Agent

- Class: **OCF**
- Parameters:
  - ***device*** – "devicename" to mount
  - ***directory*** – where to mount the filesystem
  - ***fstype*** – type of filesystem to mount
  - ***options*** – mount options *(optional)*
- This is essentially an /etc/fstab entry – expressed as a resource

ClusterMon resource Agent

- Class: **OCF**
- Parameters:
  - ***htmlfile*** – name of output file
  - ***update*** – how often to update the HTML file *(required)*
  - ***user*** – who to run crm_mon as
  - ***extra_options*** – Extra options to pass to crm_mon *(optional)*
- Update must be in seconds
- htmlfile must be located in the Apache docroot
- Suggested value for extra_options: "*-n -r*"

Apache resource Agent

- Class: **OCF**
- Parameters:
  - ***configfile*** – name of apache configuration file *(required)*
  - ***port*** – the port the server is running on *(optional)*
  - ***statusurl*** – URL to use in monitor operation *(optional)*
- Values for optional parameters are deduced from reading the configuration file.
- Configfile and html directories must go on shared media

smb and nmb resources

- Class: **LSB** (i. e., normal init script)
- They take no parameters
- Must be started *after* the IP address resource is started
- Must be started after the filesystem they are exporting is started
- Their configuration files should go on shared or replicated media

nfslock and nfsserver Resources

- Class: **LSB** (i. e., normal init script)
- Neither takes any parameters
- NFS config and lock info must be on shared media
- NFS filesystem data must be on shared media
- Inodes of mount devices and all files must match (!)
- Must be started *before* IP address is acquired
- Newer versions of NFS don't have separate nfslock service

ibmhmc STONITH Resource

- Class: **stonith**
- Parameters:
  - ***ip*** – IP address of the HMC controlling the node in question

This resource talks to the “management console” for IBM's POWER architecture machines