

MultiNET Services GmbH

Linux-HA Version 2

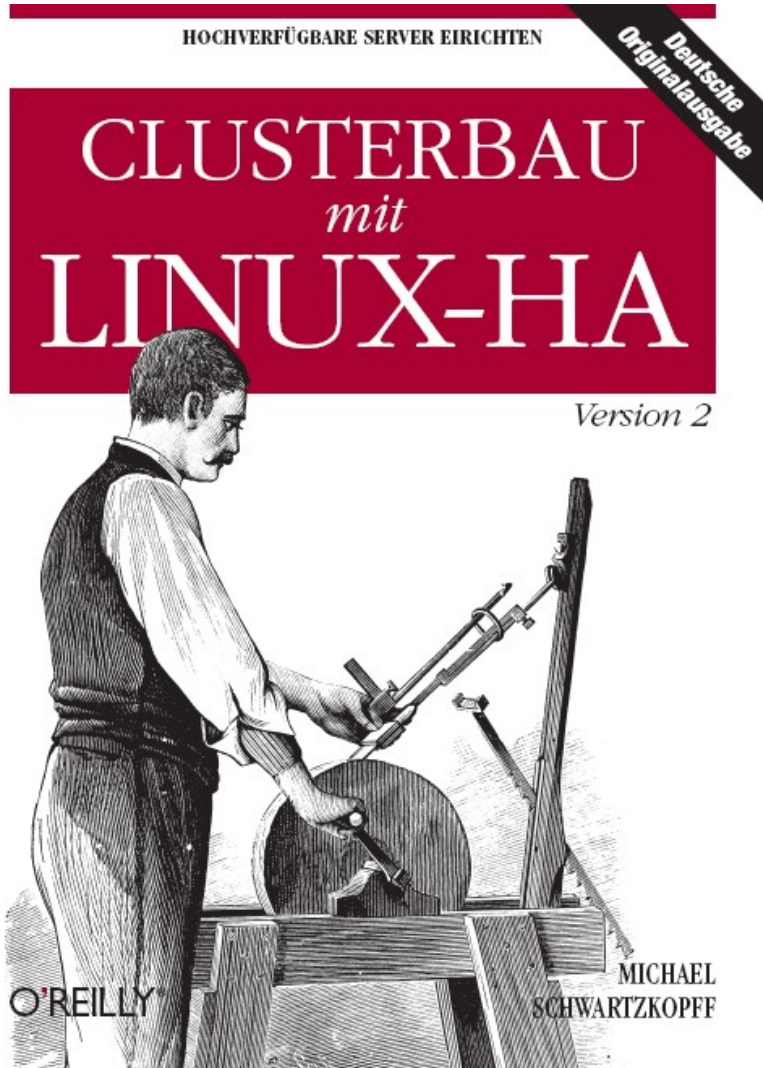
LUG Erding, 25.6.2008

Dr. Michael Schwartzkopff, MultiNET Services GmbH

Inhalt

- Hochverfügbarkeit allgemein
- Grundlagen und Architektur von Clustern
- Ressourcen und Bedingungen
- Administration: GUI und Kommandozeile
- Betrieb

Hardlink



- misch@multinet.de
- Linux-HA Mailingliste

Was ist Hochverfügbarkeit?

„Die IT soll Dienste immer anbieten.“

- Was sind „Dienste“?
- Was bedeutet „immer“?

Was bedeutet „immer“?

- 7 x 24 ist Standard, also 365,2425 Tage/Jahr.
- Tatsächlich wird diese Verfügbarkeit nur annähernd erreicht. Der Grad wird in Prozent oder Anzahl „9“ angegeben.
- 100% Verfügbarkeit gibt es nicht.
- Beispiele
 - 99% ~ 7,2 h / Monat Ausfall
 - 99,999% ~ 5,26 Min / Jahr (!)

Redundanz unter Linux

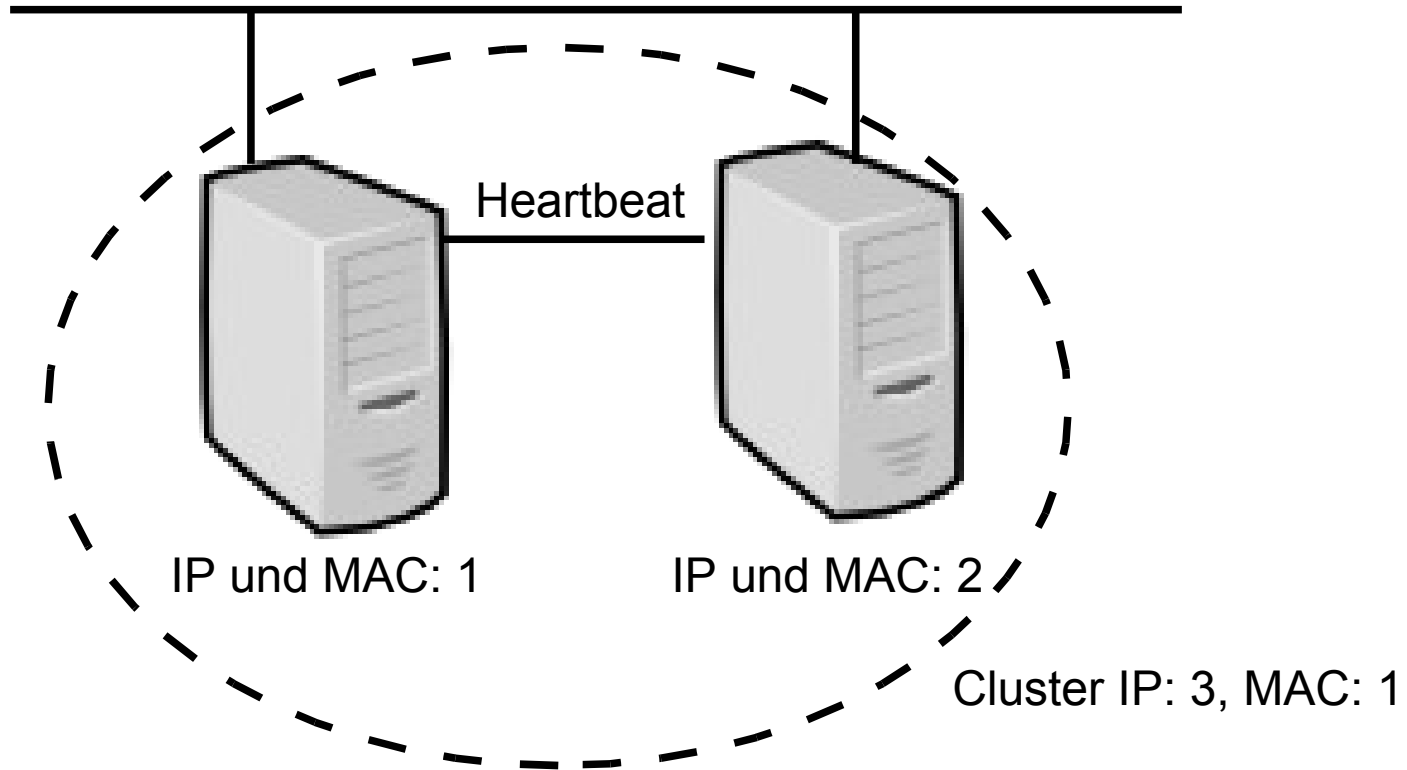
- Linux-HA (Heartbeat)
 - Ein Ersatzserver übernimmt im Fehlerfall die Aufgabe des primären Servers
 - Dies beinhaltet die Übernahme von IP Adressen, uvm.
 - Lastverteilung: Im Prinzip nein, aber ...
 - Aktiv / Passiv – System: Zwei Rechner.
 - Alternativen: Aktiv / Aktiv: Die Ressourcen sind auf zwei Rechnern verteilt.
 - oder „n to m“ Systeme: Die Ressourcen sind auf mehreren Rechnern verteilt.

Was leistet Linux HA?

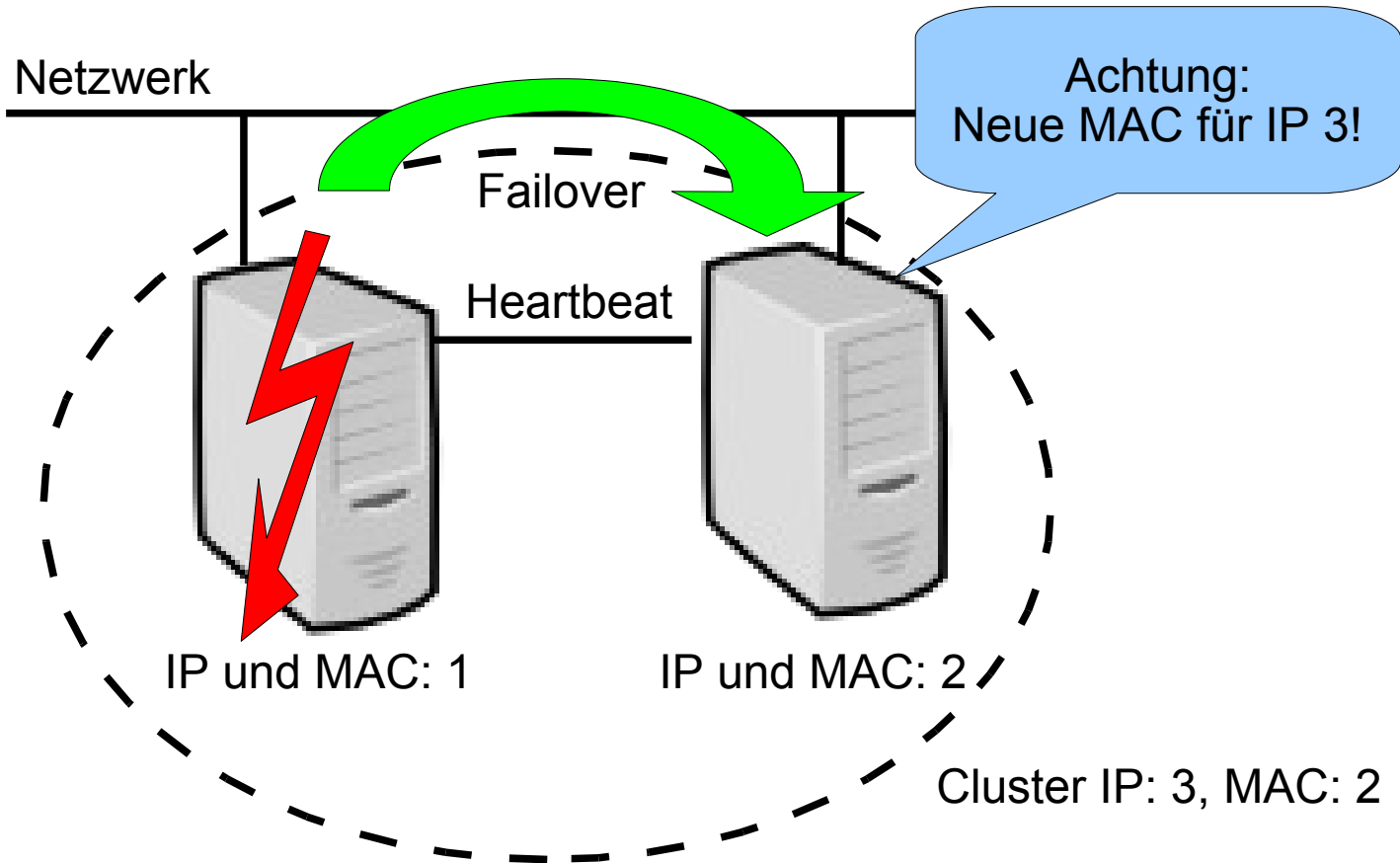
- Keine 100%ige Verfügbarkeit!
- Cluster schützen vor *einzelnen* Fehlern, nicht vor *mehreren gleichzeitigen* Fehlern.
- Auszeiten werden kurz: Sekunden bis Minuten
 - Wie beim Zaubern: „Wenn alles gut geht, war die Hand schneller als das Auge“
- Üblicherweise kann man eine „9“ an die Basisverfügbarkeit anhängen.
- Achtung: Komplexität ist der Feind der Zuverlässigkeit!

Linux – HA Prinzip I

Netzwerk



Linux – HA Prinzip I



Bekanntes ...

- Funktioniert so ähnlich wie die bekannten init startup Scripte
- Erweiterungen
 - Übergabe von Parametern an das Script (optional)
 - Läuft auf mehr als einem Computer
 - Regeln für
 - Reihenfolge für Start / Stop von Diensten
 - Beziehungen von Dienste untereinander
 - zeitliche Begrenzungen für Dienste
- HA Systeme sind ein bisschen wie „init on steroids“

- Clusterarchitekturen haben folgende Probleme zur Folge:
 - Split – Brain
 - Quorum
 - Fencing
- Gemeinsam genutzte Daten sind bei Einzelservern kein Problem, bei Clustern ist diese Funktion kritisch.

Split - Brain

- Probleme in der Kommunikation zwischen den Knoten eines Clusters können zu separaten Teilen des Clusters führen.
- Wenn jeder dieser Teile versucht, die Kontrolle über den Cluster zu erhalten nennt man das eine „Split-Brain“ Situation.
- GAU im Cluster, bis hin zum totalen Datenverlust.

Quorum

- Das „Quorum“ Konzept ist ein Versuch Split – Brain Situationen in die meisten Problemfällen zu verhindern.
- Üblicherweise wird versucht sicherzustellen, dass nur einen Teilcluster die Kontrolle übernimmt.
- Übliche Methode ist die Abstimmung: Nur eine Partition mit *mehr als* $N/2$ Knoten des ursprünglichen Clusters kann die Kontrolle übernehmen.
- Quorum funktioniert nicht mit 2 Knoten im Cluster

Fencing (I)

- Fencing versucht fehlerhafte Knoten vom Zugriff auf Ressourcen abzuschneiden (Speicher, Netz, Strom).
- Auf die Annahme eines korrekten Verhaltens kann verzichtet werden.
- Linux HA nutzt STONITH
 - Shoot The Other Node In The Head
 - Self-Fencing, z.B. IBM ServeRAID
- Gute deutsche Übersetzung für „Fencing“ gesucht!

Fencing (II)

- Garantiert die Integrität des Gesamtsystems „Cluster“ in problematischen Fällen
- Da ein Teilcluster nichts über den tatsächlichen Zustand andere Teilcluster wissen kann wird mit STONITH dafür gesorgt, dass der tatsächliche Zustand den anderen Teils der Vorstellung des ersten Teils entspricht.
- Einfachste Implementation: Strom abschalten.

Gemeinsame Daten

- Keine: In vielen Szenarien werden keine Daten zwischen den Knoten eines Clusters gemeinsam genutzt.
- Replikation: In vielen Fällen reicht eine „langsame“ Replikation der Daten z.B. mit `rsync`.
- Volle Synchronisation: Distributed Redundant Block Devices (DRBD). Ein bisschen wie RAID, nur auf zwei Rechnern.
- Outsourcen des Problems: NAS, SAN

Linux HA: Hintergrund

- Entwicklung seit 1998
- ca. 30.000 Systeme im Einsatz
- Aktive Entwicklung durch IBM und SuSE/Novell
- In den meisten Distributionen enthalten
- Keine zusätzliche Hardware notwendig, alles Userspace, d.h. kein Kernelmodul
- Alle Releases werden durch automatische Tests überprüft.

Linux HAv2: Fähigkeiten (I)

- Bis zu 16 Knoten im Cluster
- Clusterkommunikation: Seriell, udp (bcast, mcast, ucast). Redundanz möglich.
- Failover bei Problem des Knoten oder des Dienstes
- Failover bei Problemen im Netzwerk oder bei anderen frei definierten Kriterien
- Aktiv / Passiv oder volles Aktiv / Aktiv
- Eingebaute Überwachung der Ressourcen

Linux HAv2: Fähigkeiten (II)

- Unterstützung des Open Cluster Framework (OCF) Standards.
- Modell für Bedingungen und Abhängigkeiten
- Konfiguration basiert auf XML
- GUI zur Konfiguration und Überwachung
- Unterstützung für das OCFSv2
- Multi-State (Master/Slave) Ressourcen
- Split-Site Cluster mit Quorum

Installation

- Bei vielen Distribution ist `heartbeat` enthalten
- **SuSE Build Service:**
 - `http://download.opensuse.org/repositories/server:/ha-clustering/`
 - Pakete für debian, Fedora, Mandriva, SLES, openSUSE, Ubuntu
 - z. Zt. 2.1.3-18
- Aktuelle Version ($\geq 2.1.3$) wichtig!
- SLES Dokumentation auf:
 - `http://www.novell.com/documentation/sles10/heartbeat/data/heartbeat.html`

Neu: Pacemaker

- Aufspaltung des Projekts im Dezember 2007
- Hintergründe dazu: Mailingliste und http://www.clusterlabs.org/#Project_History
- Installation siehe: <http://www.clusterlabs.org/mw/Install>
- Änderungen: Cluster Resource Manager (CRM) wird ersetzt durch `pacemaker`
- Es ist einfach ein zusätzliches Paket zu installieren.

Erste Konfiguration

- Die Dateien `ha.cf` und `authkeys` werden benötigt.
- Beispiele aus `/usr/share/doc/heartbeat` nach `/etc/ha.d` kopieren.
- Knoten und Cluster – Kommunikation in `ha.cf` einrichten.
- Signatur (Passwort) in `authkeys` einrichten.

Der Rest der Konfiguration ...

- ... steht in der CIB
- **Datei:** `/var/lib/heartbeat/crm/cib.xml`
- **Niemals direkt editieren!**
- Beim erstem Systemstart wird automatisch eine leere CIB angelegt.
- **Ausgabe der CIB:** `cibadmin -Q`

Die CIB

- Die CIB enthält:
 - Information über die Konfiguration
 - Informationen über die Knoten
 - Informationen über die Ressourcen
 - Informationen über die Bedingungen
 - Informationen über den Status des Clusters
 - Welche Knoten sind up / down?
 - Alle Attribute der Knoten
 - Welche Ressourcen laufen auf welchem Knoten
- Der Administrator gibt nur die Konfiguration ein.

Variablen in der CIB

- An vielen Stellen der CIB werden Namen Werte zugewiesen
- Linux HA nutzt `<nvpair>` Tags
- Die Syntax ist:
`<nvpair id="uuid" name="Name" value="Wert"/>`
- In anderen Programmiersprachen heißt das:
`Name = Wert`

Ressourcen: Einfache Ressourcen

- *Ressourcen* sind alles, was von heartbeat verwaltet wird.
- *Agenten* sind Verbindung zwischen heartbeat und der tatsächlichen Ressource. Vergleichbar sind die Scripte, die von init beim Systemstart aufgerufen werden. Die Abkürzung ist RA.
- Einfache Ressourcen werden als „primitive Resources“ bezeichnet
- Im Gegensatz dazu existieren noch komplexere, zusammengesetzte Ressourcen.

Ressource Objekte

- Version 2 unterstützt folgende “Ressource Objekte”:
 - Einfache (primitive) Ressourcen
 - OCF, heartbeat, oder LSB resource agent Scripte
 - Klone: “*n*” Ressource Objekte irgendwo im Cluster
 - Gruppen: Eine Gruppe von einfachen Ressourcen mit impliziten Bedingungen
 - Multi-state Ressourcen (master/slave)
 - Entwickelt, um Master/Slave Ressourcen abzubilden, die dies zur Replikation benötigen (DRBD, et al)

OCF Resource Agenten

- OCF: Open Cluster Framework
- OCF RAs sind erweiterte init Skripte
 - Sie können mittels Umgebungsvariablen gesteuert werden
 - Sie haben zusätzliche Aktionen:
 - monitor: Überwachung der Ressource
 - meta-data: Liefert Informationen über den RA. Wird in der GUI genutzt.
 - validate-all: Überprüfung der Parameter der Ressource
 - OCF RAs finden sich in
`/usr/lib/ocf/resource.d/provider-name/`
- Siehe auch
<http://linux-ha.org/OCFResourceAgent>

Einfache Bedingungen

- Es gibt keine Vorgaben, wo welche Ressourcen laufen sollen.
- Falls der Administrator das nicht wünscht, können Bedingungen für Ressourcen definiert werden:
 - Anordnung: Eine Ressource wird nach einer anderen gestartet
 - Co-Lokation: Eine Ressourcen läuft auf dem selben Knoten, wie eine Andere.
 - Platzierung: Eine Ressource soll auf einem bestimmten Knoten laufen.

Punktesystem

- Der Administrator vergibt Punkte für erfüllte Bedingungen. (Co-Lokation, ...(!))
- `heartbeat` wertet diese Bedingungen im Kontext jedes Knotens aus und die zählt entsprechenden Punkte pro Ressource und pro Knoten.
- Eine Ressource wird auf dem Knoten ausgeführt, wo sie am meisten Punkte sammeln kann
- Falls gleiche Punktezahl erreicht wird, verteilt `heartbeat` die Ressourcen gleichmäßig auf die Knoten.

Bedingungen für Fortgeschrittene


- Beliebige Bedingungen können konfiguriert werden.
- Beispiele:
 - Bedingungen in Zusammenhang mit Multi-State Ressourcen.
 - Bedingungen, die sich auf Attribute von Knoten beziehen.
 - Zeitliche Begrenzungen für Bedingungen.
 - Erreichbarkeit im Netz
 - Beliebige Bedingungen
 - Failover erst nach „N“ Fehlern

Bedienung: Die GUI

- Zur Bedienung / Konfiguration gibt es eine GUI.
- Ressourcen und Bedingungen können über die GUI konfiguriert werden.
- Die GUI hat ihre Grenzen! Diese Grenzen sind teilweise sehr schnell erreicht.
- Gut für die Überwachung. Aber: Es gibt *kein* Rollenkonzept!
- Für Firewalls: Die GUI nutzt Port tcp/5560.
- Ideal sind identische Versionen von GUI und heartbeat.
- Aufrufen mit `hb_gui`.

Die GUI

Connection Resources Nodes



Name	Status
linux-ha	with quorum
Nodes	
xen02	running
resource_IP:1	running on ['xen02']
xen01	running(dc)
resource_IP:0	running on ['xen01']
Resources	
cl_ip	clone
resource_IP:0	running on ['xen01']
resource_IP:1	running on ['xen02']
Constraints	
Locations	
Orders	
Colocations	

Current Running on ['xen01']

Attributes Parameters Operations

Name	Value
ip	192.168.188.105
nic	eth0
cidr_netmask	24
clusterip_hash	sourceip-sourceport

Connected to 127.0.0.1

Die Kommandozeile

- Die wichtigsten Kommandos:
 - `crm_mon`: Status des Clusters
 - `cibadmin`: Schnittstelle zur CIB
 - `crm_resource`: Schnittstelle zum CRM
 - `crm_failcount`: Schnittstelle zum Fehlerzähler
 - `ocf-tester`: Testet eine ocf Ressource
 - `crm_verify`: Überprüft die CIB auf Probleme
 - `stonith`: STONITH Kommando

Warum Version 2?

- Einfache Konfigurationen, entsprechend Version 1, sind möglich.
- Schwierige Konfigurationen ebenfalls.
- Interne Überwachung der Ressourcen durch den Cluster.
- Dynamische Modifikation am Cluster während der Laufzeit.

Überwachung eines Clusters

- Externe Überwachung notwendig.
- Netzwerk Management nutzen (oder aufbauen)!
- Simple Network Management Protocol (SNMP) ist der Klassiker.
- Linux HA bringt einen eigenen Subagenten mit.

MIB von Linux-HA

```
LINUX-HA-MIB::LHATotalNodeCount.0 = Counter32: 2
LINUX-HA-MIB::LHALiveNodeCount.0 = Counter32: 2
(...)
LINUX-HA-MIB::LHANodeName.1 = STRING: xen02
LINUX-HA-MIB::LHANodeName.2 = STRING: xen01
LINUX-HA-MIB::LHANodeType.1 = INTEGER: normal(1)
LINUX-HA-MIB::LHANodeType.2 = INTEGER: normal(1)
LINUX-HA-MIB::LHANodeStatus.1 = INTEGER: active(3)
LINUX-HA-MIB::LHANodeStatus.2 = INTEGER: active(3)
(...)
LINUX-HA-MIB::LHANodeIFCount.1 = Counter32: 1
LINUX-HA-MIB::LHANodeIFCount.2 = Counter32: 1
LINUX-HA-MIB::LHAIFName.1.1 = STRING: eth0
LINUX-HA-MIB::LHAIFName.2.1 = STRING: eth0
LINUX-HA-MIB::LHAIFStatus.1.1 = INTEGER: up(1)
LINUX-HA-MIB::LHAIFStatus.2.1 = INTEGER: up(1)
(...)
```

Traps von heartbeat

Wichtige Traps von hbagent:

```
LHANodeStatusUpdate NOTIFICATION-TYPE  
  OBJECTS { LHANodeName LHANodeStatus }  
  DESCRIPTION "A node status change event just happened."
```

```
LHAIFStatusUpdate NOTIFICATION-TYPE  
  OBJECTS { LHANodeName LHAIFName LHAIFStatus }  
  DESCRIPTION "A link status just changed."
```

```
LHAMembershipChange NOTIFICATION-TYPE  
  OBJECTS { LHANodeName LHAMemberLastChange }  
  DESCRIPTION "A node just changed it membership. "
```

Überwachung durch `nagios`

- Abfrage von `LHALiveNodeCount` und Vergleich mit der Anzahl der Knoten im Cluster
- Definition eines neuen Kommandos in `checkcommands.conf`:

```
define command {  
    command_name check_snmp_cluster  
    command_line $USER1$/check_snmp \  
        -H $HOSTADDRESS$ \  
        -o .1.3.6.1.4.1.4682.1.2.0 \  
        -w 2: -c 2:  
}
```

wenn der Cluster aus 2 Knoten besteht.

Vielen Dank für die Aufmerksamkeit!

Rückfragen jederzeit:

- misch@multinet.de
- Tipps und Tricks: <http://clusterbau.blogspot.com>
- Praxis – Seminar bei Heinlein-Support