

# 1

## MK Livestatus

Alle Rechte an diesen Unterlagen liegen bei Mathias Kettner, Preysingstr. 74, 81667 München. Vervielfältigung oder Reproduktion – auch auszugsweise – nur mit vorheriger schriftlicher Genehmigung. Die Nutzung ist beschränkt auf die Unterrichtsbegleitung für die von Mathias Kettner veranstaltete Schulung und deren spätere persönliche Nutzung durch die Teilnehmer. Insbesondere ist eine Nutzung der Unterlagen für weitere Schulungen oder ähnliche Veranstaltungen untersagt. Diese Unterlagen sind nur original mit Deckblatt, Datumshweis, in Farbe und auf 160-Gramm Papier. Melden Sie unerlaubte Kopien oder einen anderen Verstoß gegen das Urheberrecht bitte per Telefon unter 089 / 18 90 4210 oder per Email an mk@mathias-kettner.de.



### 1. Mathias Kettner

dozent\_kettner

#### Mathias Kettner



- Linux & Nagios-Experte
- Diplom Informatiker
- früher Entwickler bei SuSE
- Autor von „Fehlerdiagnose und Problembhebung unter Linux“
- seit 2000 selbständig als Dozent und Consultant

### 2. Firma MK

firma\_kettner



#### MK – Mathias Kettner

- Schulung, Consulting, Support
- Schwerpunktthema: Linux im Rechenzentrum, Nagios
- Sitz in München, Haidhausen

Kontakt:

Mathias Kettner  
Preysingstraße 74  
81667 München  
WWW: mathias-kettner.de  
E-Mail: mk@mathias-kettner.de

### 3. Livestatus

livestatus

Welche Wege gibt es zum Zugriff auf die Statusinformationen und historischen Ereignisse von Nagios? von Nagios?

#### status.dat (und nagios.log)

- klassischer Weg (Nagios CGIs)
- Jede Anfrage erfordert komplettes Parsen
- Größe bei 26.000 Services über 30 MB!
- Daten sind i.A. 10 - 30 Sekunden veraltet
- teilweise fehlen Informationen (z.B. Timeperiods)
- keine Unterstützung für verteiltes Monitoren

#### SQL-Datenbanken – NDO/Merlin

- Daten werden aktiv in Datenbank transportiert
- Zugriff über SQL
- teils recht komplexe Joins notwendig
- vor allem: Aktualhalten ist extrem CPU/IO-hungrig!
- große Probleme mit Housekeeping, etc.
- zusätzlicher Aufwand durch Datenbank-Administration
- verteiltes Monitoring kaum unterstützt

#### Check\_MK's Livestatus Modul

- Idee: Nagios hat alle Daten im Hauptspeicher
- diese sind untereinander optimal verknüpft
- Warum nicht diese direkt anzapfen...
- ...und an einem Socket zur Verfügung stellen?
- ☞ Genau das macht Livestatus!
- erste stabile Version: 27. November 2009

### Addons, die Livestatus unterstützen

- NagVis von Lars Michelsen: ab Version 1.4.5
- Thruk von Sven Nierlein: Nagios-CGI-Nachbau in Perl
- Nagios Business Process AddOns: ab Version 0.9.5
- und: Check\_MK Multisite: komplett neue GUI

Derzeit in Arbeit ist:

- Icinga Weboberfläche

#### Architektur

- livestatus.o wird als NEB-Modul geladen
- dieses öffnet ein UNIX-Socket und wartet auf Anfragen
- Anfragen werden direkt aus Nagios-Daten beantwortet
- Eigene Anfragesprache erlaubt effizientes Filtern
- → sehr schnelle Antworten und topaktuelle Daten
- → überhaupt kein Disk-I/O
- → minimaler CPU-Verbrauch, und das nur bei Abfragen
- → Nagios-Knowhow (z.B in\_notification\_period)

#### Verteiltes Monitoren mit Livestatus

- Livestatus horcht auf TCP Socket (per inetd)
- GUI fragt mehrere Sockets parallel ab...
- ...und baut daraus eine Gesamtsicht
- Das kann z. B. NagVis, Multisite und Thruk

### 4. Installation

livestatus\_installieren

Zwei Methoden, um Livestatus aufzusetzen:

- 1 Check\_MK installieren und bei Livestatus „yes“ sagen
- 2 Livestatus ohne Check\_MK von Hand kompilieren



## Kompilieren von Hand

Quellcodes von MK Livestatus:

[http://mathias-kettner.de/check\\_mk](http://mathias-kettner.de/check_mk) → Download

```
# tar xzf mk-livestatus-1.1.6.tar.gz
# cd mk-livestatus-1.1.6
# ./configure --prefix=/usr
# make -j 10
# make install
```

Ergebnis:

```
/usr/lib/mk-livestatus/livestatus.o Livestatusmodul
/usr/bin/unixcat Hilfsprogramm
```

Modul in Nagios integrieren:

```
_____ /etc/nagios/nagios.cfg _____
broker_module=/usr/lib/mk-livestatus/livestatu
➔s.o /var/lib/nagios/rw/live
event_broker_options=-1
```

- /var/lib/nagios/rw/live ist das UNIX-Socket.
- Soll im gleichen Verzeichnis liegen, wie Pipe

Hinten kann man Optionen anhängen, z. B.:

```
debug ..... Debuglevel (0, 1 oder 2)
num_client_threads ... Anzahl Threads
```

Beispiel: Debugmodus einschalten:

```
broker_module=/usr/lib/mk-livestatus/livestatu
➔s.o /var/lib/nagios/rw/live (debug=1)
```

Dokumentation:

[http://mathias-kettner.de/checkmk\\_livestatu](http://mathias-kettner.de/checkmk_livestatu)  
➔s.html

Danach Nagios neustarten. In nagios.log:

```
_____ nagios.log _____
[1268328639] livestatus: successfully finished.
➔ initialization
```

## Testabfrage

```
# echo -e 'GET status' | \
  unixcat /var/lib/nagios/rw/live
accept_passive_host_checks;accept_passive_
service_checks;cached_log_messages;check_e
xternal_commands;check_host_freshness;chec
1;1;0;1;0;1;8;0.1708863636;1;0;1;1;0;0;6
0;1268328839;0;1.1.6;18233;2;0.07944286369
;0;0;1;1268328639;3.2.1;7;0.1658863636;3;0
```

## 5. Livestatus LQL

livestatus\_lql

### Die Livestatus Query Language

Grundsätzlicher Aufbau:

- Text, Zeile für Zeile, Trenner ist Linefeed
- zuerst GET und den Namen einer Tabelle
- dann beliebig viele Header
- dann Verbindung schließen oder Leerzeile
- ähnlich, aber nicht gleich HTTP

### Beispiel

```
GET hosts
Columns: name alias
Filter: state != 0
```

### Tabellen (Version 1.1.3)

Statusobjekte:

```
hosts ..... alle Hosts
services ..... alle Services
hostgroups ..... alle Hostgruppen
servicegroups ..... alle Servicegruppen
comments ..... Host-/Servicecommentare
downtimes ..... Scheduled Downtimes
```

Joins mit Gruppen:

```
servicesbygroup ..... Servicegruppen mit Services
hostsbygroup ..... Hostgruppen mit Hosts
servicesbyhostgroup Hostgruppen mit Services
```

- Zweck: alle Elemente aller Gruppen in einer Anfrage
- Hosts/Services können mehrfach oder garnicht auftauchen

Konfigurationsdaten:

```
commands ..... Kommandos
contacts ..... Kontakte
contactgroups ..... Kontaktgruppen
timeperiods ..... Zeitperioden (nur Name und Alias)
```

Sonstiges:

```
log ..... Die Nagios Logdateien!
status ..... Programmstatus, Performance
columns ..... alle Spalten aller Tabellen
```

### Header: Columns

- wählt benötigte Spalten aus Tabelle aus
- wenn vorhanden → Header werden unterdrückt
- wenn fehlt → alle Spalten
- Achtung: teils sehr viele Spalten (z. B. 190 bei log)
- nicht gewählte Spalten kosten keine Performance!

### Header: Filter

- wählt bestimmte Zeilen aus Tabelle aus
- definiert Bedingung auf einer Spalte

Mögliche Operatoren:

```
= ..... Gleichheit
!= ..... Ungleichheit
> ..... größer
>= ..... größer oder gleich
< ..... kleiner
<= ..... kleiner oder gleich
~ ..... passt auf regulären Ausdruck (Teilstring)
```

```
=~ ..... gleich bis auf groß/klein
~~ ..... regulärer Ausdruck groß/klein
```

### Beispiele

Alle Hosts, die nicht OK sind:

```
GET hosts
Filter: state != 0
```

Services, deren Check länger als 3 sec. gedauert hat:

```
GET services
Filter: execution_time > 3
```

## 6. Livestatus per TCP

livelstatus\_tcp

### Idee

- Livestatussocket per TCP erreichbar machen
- von anderem Nagioshost aus abrufen
- komplett neuer Ansatz des verteilten Monitors!

### Vorgehen

☞ Wir nutzen unixcat und xinetd!

UNIX-Socket an Port 6557 binden:

```
_____ /etc/xinetd.d/livestatus _____
service livestatus
{
    type                = UNLISTED
    port                = 6557
    socket_type         = stream
    protocol            = tcp
    wait                = no
    cps                 = 100 3
    flags               = NODELAY
    user                = nagios
    server              = /usr/bin/unixcat
    server_args         = /var/lib/nagios/rw/live
```

```
only_from    = 10.0.0.0/24 127.0.0.1
disable      = no
}
```

Hinweise:

- cps setzt Connections pro Sekunde rauf
- NODELAY verbessert TCP Antwortzeiten
- only\_from immer verwenden!

### Test von anderem Rechner aus

Querydatei anlegen (Leerzeile am Ende wichtig):

```
_____ query1 _____
GET status
Columns: program_version
```

Query per netcat absenden:

```
> netcat nagioshost 6557 < query1
3.2.1
```

🔴 Auch bei Abfragen mit echo Leerzeile

```
> echo -e "GET hosts\n" | netcat klon 6557
```

## 7. Multisite

multisite

### Multisite - eine neue Nagios-GUI

Alleinstellungsmerkmale:

- 1 schnell
- 2 flexibel
- 3 ermöglicht verteiltes Monitoren

Multisite löst außerdem Multiadmin ab und übernimmt dessen Möglichkeiten der Massenadministration.

### 1 Schnell

Multisite baut direkt auf Livestatus auf → sehr schnelle Antwortzeiten, auch bei sehr großen Installationen mit vielen zigtausend Checks

### 2 Flexibel

- Erweiterbar in alle Richtungen via Plugins
- Dynamische Sidebar, pro User konfigurierbar
- am wichtigsten: frei definierbare Views

### 3 Verteiltes Monitoren mit Check\_MK Multisite

- direkter Zugriff auf mehrere Nagios-Instanzen
- Standortübergreifende Darstellung
- Darstellung als ein großes Monitoringsystem
- Einzelne Standorte bleiben aber erkennbar
- skaliert sehr gut, da kein zentraler Datenspeicher
- Standorte können flexibel zu- und weggeschaltet werden
- leicht auf bestehendes Nagios aufschaltbar

## 8. Multisite aufsetzen

multisite\_aufsetzen

Multisite wird mit Check\_MK automatisch mitinstalliert. Eventuell fehlt noch:

Python-Unterstützung für Apache installieren:

```
# aptitude install libapache2-mod-python
```

Dann Apache neu starten:

```
# /etc/init.d/apache2 restart
```

Und jetzt: ☞ [http://nagios/check\\_mk](http://nagios/check_mk)

### Konfiguration

Größtenteils per Web konfigurierbar, außer:

- Benutzerrollen zuordnen
- Standorte (Sites) = Nagiosinstanzen definieren

Konfiguration geschieht in `/etc/check_mk/multisite.mk`

### Sites

Eine Site ist eine Nagios-Installation.

- Default: nur lokales Nagios per UNIX-Socket

Konfiguration von lokalem und entferntem Site:

```

/etc/check_mk/multisite.mk
sites = {
    "muc" : { "alias" : "Muenchen" },
    "ham" : {
        "alias" : "Hamburg",
        "socket" : "tcp:10.22.128.11:6557",
    }
}

```

### Benutzerverwaltung

- Anmeldung per HTTP – genau wie bei Nagios
- kein HTTP user bekannt → kein Zugang
- Setup verwendet gleiche Userdatenbank
- Rollenzuordnung in `multisite.mk`

### Rollen

Defaultrechte der Rollen:

```

admin ..... Darf alles
guest ..... Darf alles sehen aber nix machen
user ..... Darf seine Objekte sehen und steuern

```

### Zuordnung zu Rollen

Admins in Variable `admin_users` listen:

```

/etc/check_mk/multisite.mk
admin_users = [ "nagiosadmin", "hh" ]

```

Gäste in `guest_users` listen:

```

guest_users = [ "welcome", "guest" ]

```

Alle andere mit HTTP login sind `user`. Nur bestimmte Benutzer überhaupt reinlassen:

```
users = [ "meier", "mueller" ]
```

→ Jetzt darf kein Unbekannter mehr rein.

## 9. Views

multisite\_views

View: Ansicht auf einen Teil der Statusdaten

### Merkmale einer View

- 1 Datenquelle: Hosts, Services, etc.
- 2 Layout: Tabelle, Blöcke, ...
- 3 Filter: Welche Datensätze werden gezeigt?
- 4 Sortierung: In welcher Reihenfolge?
- 5 Gruppierung: Nach was wird gruppiert?
- 6 Spalte: Welche Datenfelder werden gezeigt... und wie werden sie verlinkt?

☞ All dies kann frei editiert werden

### Views und Benutzer

- Jeder kann eigene Views erstellen
- er kann sie für andere freigeben
- Der Admin kann Systemviews abändern

### Sichtbarkeit von Views

View hat einen Namen, z. B. `allhosts`. Jeder Benutzer kann View dieses Namens definieren. Es gilt folgende Reihenfolge:

- 1 eigene Views
- 2 öffentliche Views von Administratoren
- 3 eingebaute Views
- 4 öffentliche Views anderer Benutzer

→ nur Admins können eingebaute Views überschreiben

### Filter

Filter wählt bestimmte Datensätze aus, z. B.:

- nur Hosts mit Problemen

- Services mit einem Text im Namen
- Services die nicht in einer Downtime sind

Vier Arten wie eine View einen Filter einsetzen kann:

- 1 fest verdrahtet in View
- 2 für Benutzer auswählbar
- 3 über HTML-Variablen in URL setzbar (Verlinken)
- 4 deaktiviert

### Beispiel für 3

View `host` hat Filter für Hostname zum Verlinken.

Aufruf dieser View für einen bestimmten Host:

☞ [/check\\_mk/view.py?view\\_name=host&host=nagios](/check_mk/view.py?view_name=host&host=nagios)

- Link-Filter dienen auch zum Verlinken von Views untereinander.

## 10. Vielen Dank

extro\_mk\_kompakt



Bis zum nächsten Kurs alles Gute und viel Spaß beim Erproben des Stoffs in der Praxis!

Mathias Kettner



Copyright 2010 Mathias Kettner



## MK Livestatus

Alle Rechte an diesen Unterlagen liegen bei Mathias Kettner, Preysingstr. 74, 81667 München. Vervielfältigung oder Reproduktion – auch auszugsweise – nur mit vorheriger schriftlicher Genehmigung. Die Nutzung ist beschränkt auf die Unterrichts-  
begleitung für die von Mathias Kettner veranstaltete Schulung und deren spätere persönliche Nutzung durch die Teilnehmer.  
Insbesondere ist eine Nutzung der Unterlagen für weitere Schulungen oder ähnliche Veranstaltungen untersagt. Diese Unterlagen  
sind nur original mit Deckblatt, Datumshinweis, in Farbe und auf 160-Gramm Papier. Melden Sie unerlaubte Kopien oder einen  
anderen Verstoß gegen das Urheberrecht bitte per Telefon unter 089 / 18 90 4210 oder per Email an [mk@mathias-kettner.de](mailto:mk@mathias-kettner.de).



Alle Rechte vorbehalten. Vervielfältigung  
- auch auszugsweise - nur mit vorheriger  
schriftlicher Genehmigung.

Mathias Kettner  
Preysingstraße 74  
81667 München  
Tel. 089 / 18 90 4210

[mk@mathias-kettner.de](mailto:mk@mathias-kettner.de)  
[www.mathias-kettner.de](http://www.mathias-kettner.de)

