

# Erweiterungen schreiben für Check\_MK

# Check\_MK

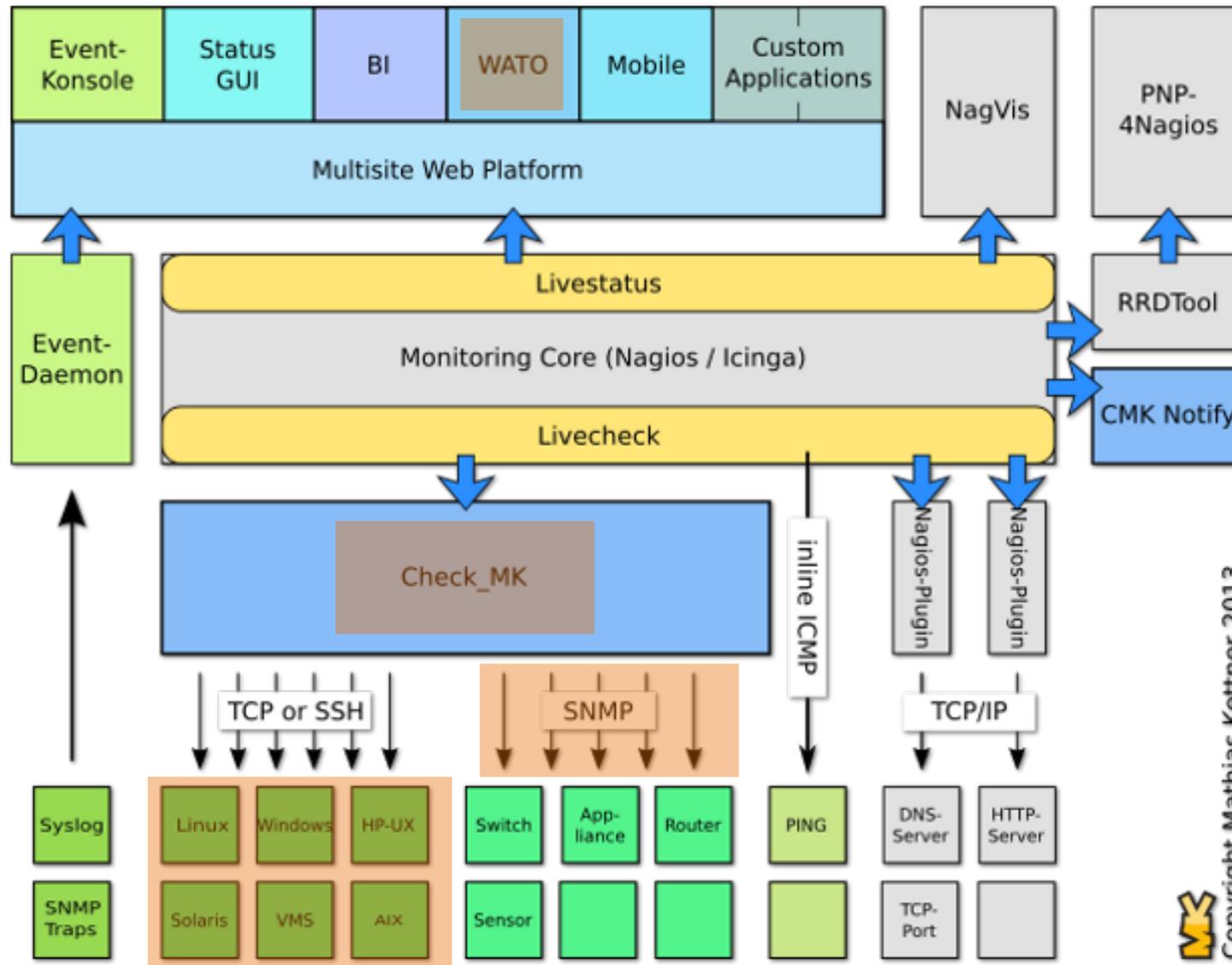
## Check\_MK

- entwickelt von Mathias Kettner & Team
  - [https://mathias-kettner.de/check\\_mk.html](https://mathias-kettner.de/check_mk.html)
- entstanden aus einem Nagios-Plugin
  - inzwischen eigenständiges Monitoring-System
- kompatibel zu Nagios / Icinga
- über 1250 mitgelieferte Checks
  - [https://mathias-kettner.de/cms\\_check\\_plugins\\_catalog.html](https://mathias-kettner.de/cms_check_plugins_catalog.html)
- regelbasierte Konfiguration mit Web-GUI WATO
- OMD-Konzept mit mehreren Instanzen
  - verteiltes Monitoring
  - Mandantenfähigkeit

# Datenmodell

- bekannt aus der Nagios-Welt
- Host
  - UP 0
  - DOWN 1
- Service
  - OK 0
  - WARN 1
  - CRIT 2
  - UNKNOWN 3
- Ein Service ist immer genau einem Host zugeordnet

## Architektur



Copyright Mathias Kettner 2013



## Host-Agent

- Unix-Systeme: Shell-Skript
  - mit xinetd an Port 6556 lauschend
  - oder Abfrage z.B. via SSH möglich
- Windows: Service
- read-only Zugriff
  - nimmt keine Parameter oder andere Daten entgegen
- liefert Informationen
  - sammelt alles ein, was bekannt ist
- optimiert auf schnelle Ausführung
  - Abfragedauer üblicherweise < 1s
- übliche Abfragefrequenz: 60s

## Agentendaten

```
<<<check_mk>>>
Version: 1.2.8p16
AgentOS: linux
Hostname: defiant
AgentDirectory: /etc/check_mk
DataDirectory: /var/lib/check_mk_agent
SpoolDirectory: /var/lib/check_mk_agent/spool
PluginsDirectory: /usr/lib/check_mk_agent/plugins
LocalDirectory: /usr/lib/check_mk_agent/local
<<<df>>>
udev                devtmpfs            8183928              0          8183928            0% /dev
tmpfs               tmpfs               1638988             10864      1628124            1% /run
/dev/mapper/system-root ext4                30832636           21130248   8113140           73% /
/dev/sda1            ext4                1998672             75068      1802364            4% /boot
<<<ps>>>
(root,120400,6596,00:00:01/06:33:16,1) /sbin/init
(root,0,0,00:00:00/06:33:16,2) [kthreadd]
(root,0,0,00:00:00/06:33:16,3) [ksoftirqd/0]
(root,0,0,00:00:00/06:33:16,5) [kworker/0:0H]
...
<<<mem>>>
MemTotal:          16389876 kB
MemFree:           3190608 kB
MemAvailable:     8933832 kB
Buffers:          1167128 kB
...
```

## Raw Edition ↔ Enterprise Edition

- Raw Edition ist die Open Source Variante
  - alle Checks enthalten
  - Nagios / Icinga als Kern
- Enterprise Edition hat zusätzlich
  - Micro Core als Nagios-Ersatz → bessere Performance
  - Agent-Bakery und Agent-Updates
  - Reporting mit PDF-Engine
  - Neues Metriken-System als Ersatz von PNP4Nagios
  - Graphite-Anbindung
  - Web-Interface für Erweiterungspakete
  - Innovation-Releases
- [http://mathias-kettner.de/check\\_mk\\_introduction.html](http://mathias-kettner.de/check_mk_introduction.html)

# Erweiterungsmöglichkeiten

## Checks

- klassisches Nagios-Plugin
- MRPE
  - Mathias Kettner's Remote Plugin Executor
  - Nagios-Plugin vom Agenten auf dem Host ausgeführt
- Local Check
- Agenten-Plugin mit
  - Check-Plugin
  - WATO-Plugin
- SNMP
- Spool-Files
  - wenn die eigene Applikation Monitoring-Daten erzeugt

## klassisches Nagios-Plugin

- Nagios Plugin-API
  - <https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/pluginapi.html>
- läuft auf dem Monitoring-Server
  - aktiver Check
  - eigener Schedule
  - eigene Konfiguration
- prüft „irgendetwas“ über das Netzwerk
- gibt als Returncode 0, 1, 2 oder 3 zurück
- auf stdout noch Check Output und Performance-Daten
- große Sammlung bereits vorhandener Checks
- Use-Case: Application-Monitoring

## MRPE

- Nagios-Plugin durch Agenten ausgeführt
  - [https://mathias-kettner.de/checkmk\\_mrpe.html](https://mathias-kettner.de/checkmk_mrpe.html)
- läuft auf dem Host
  - Konfiguration auf dem Host in `/etc/check_mk/mrpe.cfg`
  - sequenziell mit weiteren MRPE-Checks
- Laufzeit beachten
  - darf insgesamt nicht zu lang werden
- Entscheidung über Ergebnis auf dem Host
- Rückgabe inkl. Performance-Daten
- Use-Case: spezialisiertes Nagios-Plugin auf einem Host

## Local Check

- Eigenes Skript vom Agenten ausgeführt
  - [https://mathias-kettner.de/checkmk\\_localchecks.html](https://mathias-kettner.de/checkmk_localchecks.html)
- läuft auf dem Host
  - /usr/lib/check\_mk\_agent/local
  - eigene Konfiguration
- Asynchroner Aufruf möglich
  - z.B. alle 5 Minuten
  - nur Linux-Agent
- Entscheidung über Ergebnis auf dem Host
- Spezifisches Ausgabeformat („API“)
  - Status \_ Item\_Name \_ PerfData \_ Check Output
- Use-Case: sehr spezifischer Check

## Agent Plugin

- eigentlich mehrere Plugins
  - Agent, Check, WATO
  - [https://mathias-kettner.de/checkmk\\_writing\\_checks.html](https://mathias-kettner.de/checkmk_writing_checks.html)
- Agenten-Plugin
  - auf dem Host in `/usr/lib/check_mk_agent/plugin`
  - erzeugt eigene Datensektion → eindeutiger **Name**
  - wird vom Agenten gestartet, asynchron möglich
- Check-Plugin
  - läuft im Check\_MK-Kontext auf dem Monitoring-Server
  - wertet die Daten der Sektion aus (Implementation in Python)
- WATO-Plugin
  - erzeugt Parameter-Regeln und anderes in Python

# SNMP

- Check-Plugin
  - läuft im Check\_MK-Kontext
  - gibt an, welche OIDs abgefragt werden müssen
    - `check_info['snmp_info']` und `check_info['snmp_scan_function']`
  - bekommt die SNMP-Daten aufbereitet in `info[][]`
- Einfache Implementierung
- Keine Wiederholung der immergleichen SNMP-Kommunikation
  - das macht Check\_MK
  - kein Ärger mit `snmpget` oder SNMP-Bibliotheken ☐
- Komplette transparent für Entwickler und Nutzer
  - [https://mathias-kettner.de/checkmk\\_devel\\_snmpbased.html](https://mathias-kettner.de/checkmk_devel_snmpbased.html)
- Use Case: SNMP-fähiges Gerät

## Spool-Files

- Applikation schreibt selber Monitoring-Daten
  - [https://mathias-kettner.de/check\\_mk\\_werks.php?werk\\_id=0016](https://mathias-kettner.de/check_mk_werks.php?werk_id=0016)
- Datei in `/var/lib/check_mk_agent/spool`
  - eigene Sektion definieren
  - oder z.B. <<<local>>> verwenden
- ggfs Nummer als Namensprefix
  - dann weiß der Agent, wie alt die Datei max. sein darf
- Agent fügt Inhalt der Datei eigener Ausgabe hinzu
- Check-Plugin
  - auf Monitoring-Server
  - wertet Daten in Sektion aus
- Use-Case: Eigenentwickelte Anwendung überwachen

# Check-Plugin & WATO

## Check Plugin

- Liegt in `/omd/sites/<sitename>/local/share/check_mk/checks`
  - definiert 2 Funktionen und 1 Dictionary
  
- `def inventory_checkname(info):`  
    `return []`
  
- `def check_checkname(item, params, info):`  
    `return (state, message, perfdata)`
  
- `check_info['checkname'] = {`  
    `"inventory_function" : inventory_checkname,`  
    `"check_function" : check_checkname,`  
    `"service_description" : "Check Name %s",`  
    `"group" : "checkname"`  
    `}`

## **check\_info['check\_name'] Dictionary**

- inventory\_function
- check\_function
- service\_description
- parse\_function
  - eine optionale Funktion, die info für check + inventory Funktion aufbereitet
- has\_perfdata (True, False)
- group
  - welche WATO Konfigurationsregel gilt
- default\_levels\_variable
  - Name der Variablen mit Default-Parametern
- includes

## inventory\_function(info)

- erhält in info den Inhalt der passenden Agentensektion
- info ist Liste von Listen
  - äußere Liste sind die Zeilen, innere Liste die Wörter einer Zeile
  - Sektion wird in eine Tabelle zerlegt
- aus:

```
/dev/mapper/system-root  ext4    30832636 21130152 8113236   73% /  
/dev/sda1                ext4    1998672  75068   1802364   4% /boot
```
- wird:
  - [ [ '/dev/mapper/system-root', 'ext4', '30832636', '21130152', '8113236', '73%', '/', ],  
[ '/dev/sda1', 'ext4', '1998672', '75068', '1802364', '4%', '/boot' ] ]
- liefert Liste von Items mit Default-Parametern zurück
  - z.B.: [ ( '/', None ), ('/boot', None) ]

## **check\_function(item, params, info)**

- wird für jedes inventarisierte Item einmal aufgerufen
- erhält
  - das aktuelle Item
  - die für das aktuelle Item geltenden Parameter
  - die kompletten Daten der Sektion in info als Tabelle
- muß in info[][] das aktuelle Item finden
- ggfs Meßwerte mit Parametern vergleichen
- liefert zurück
  - Status (0, 1, 2, 3)
  - Message
  - ggfs Performancedaten

# WATO

## Check Parameter Regel

- „modelliert“ Regel-Formular in Python
  - liegt in `/omd/sites/<sitename>/local/share/check_mk/web/plugins/wato/<checkname>.py`
  - mitgelieferte Parameter-Regeln in `/omd/sites/<sitename>/share/check_mk/web/plugins/wato/check_parameters.py`
- eine Funktion wird aufgerufen: `register_check_parameters(`
  - `subgroup,`
  - `checkgroup,`
  - `title,`
  - `valuespec,`
  - `itemspec,`
  - `match_type, ...)`
- Am besten Beispiele anschauen

# WATO

## valuespec & itemspec

- Python-Klassen für verschiedene Formular-Datentypen
  - `/omd/versions/default/share/check_mk/web/htdocs/valuespec.py`
- gemeinsame Attribute `title`, `help`, `default_value`
- z.B.
  - `Integer()`
  - `TextAscii()`
  - `EmailAddress()`
  - `IPv4Network()`
  - `Filename()`
  - `ListOf()`
  - `Dictionary()`
  - `Tuple()`
- Können verschachtelt werden: `ListOf(Tuple(elements=[Integer(), ...`

# Demo

[https://github.com/HeinleinSupport/check\\_mk](https://github.com/HeinleinSupport/check_mk)

## CLI Praxistips

- `cmk -ll --debug -v --checks <checkname> <hostname>`
- `cmk --debug -vp --checks <checkname> <hostname>`
- `cmk --snmpwalk <hostname>`
- `cmk --translatesnmp <hostname>`
- `mkp find`
- `mkp create`
- `mkp pack`

**Soweit, so gut.**

**Gleich sind Sie am Zug:  
Fragen und Diskussionen!**

- Natürlich und gerne stehe ich Ihnen jederzeit mit Rat und Tat zur Verfügung und freue mich auf neue Kontakte.
  - Robert Sander
  - Mail: [r.sander@heinlein-support.de](mailto:r.sander@heinlein-support.de)
  - Telefon: 030/40 50 51 - 43
  
- Wenn's brennt:
  - Heinlein Support 24/7 Notfall-Hotline: 030/40 505 - 110