Robert Sander <r.sander@heinlein-support.de>

Monitoring automatisieren mit CheckMK 2.0

Robert Sander <r.sander@heinlein-support.de>

- **Heinlein Support**
  - IT-Consulting und 24/7 Linux-Support mit ~40 Mitarbeitern
  - Eigener Betrieb eines ISPs seit 1992
  - Täglich tiefe Einblicke in die Herzen der IT aller Unternehmensgrößen

- 24/7-Notfall-Hotline: 030 / 40 50 5 - 110
  - 28 Spezialisten mit LPIC-2 und LPIC-3
  - Für alles rund um Linux & Server & DMZ
  - Akutes: Downtimes, Performanceprobleme, Hackereinbrüche, Datenverlust
  - Strategisches: Revision, Planung, Beratung, Konfigurationshilfe

Robert Sander <r.sander@heinlein-support.de>

CheckMK – Monitoring-System

- Entwicklung in München durch tribe29 GmbH
- Agentenbasiert
- Raw Edition 100% Open Source
- Nagios als Monitoring-Kern, in Enterprise Edition eigener Kern
- eigene Weboberfläche
- eigenes Konfigurationstool
- durch Plugin-Framework einfach erweiterbar

Robert Sander <r.sander@heinlein-support.de>

CheckMK – Monitoring-System

- → Instanzenkonzept aus der Open Monitoring Distribution (†)

- → Datenmodell von Nagios übernommen
- → Host & Service Checks
- → Service Check immer einem Host zugeordnet

- → Durch Agentennutzung Auto-Discovery von Service Checks

Robert Sander <r.sander@heinlein-support.de>

CheckMK – Version 2.0

- Neue Weboberfläche
  - soll gerade Einsteiger:innen das Arbeiten vereinfachen
  - neue grafische Elemente
- Drop-Down Menüs statt endloser Button-Liste
- Enterprise Graphen-System auch in RAW-Edition
- Vorhersage-Graphen
- CheckMK Micro-Core der Enterprise Edition verbessert
- Neue API in der Check-Engine
- Neue REST-API für die Konfiguration und Statusabfrage
- Python 3.8

Warum Automatisieren?

Robert Sander <r.sander@heinlein-support.de>

Automatisierung macht das Leben einfacher

- Konfiguration ist reproduzierbar
- Aus dem Deployment automatisch Hosts erzeugen
  - keine unüberwachten Systeme mehr
- Zentrale Datenquelle anzapfen
  - Inventardatenbank als „Quelle der Wahrheit“
- Weniger Handarbeit → weniger Fehler
  - dafür wurde der Computer erfunden

Robert Sander <r.sander@heinlein-support.de>

Datenquellen für Automatisierung

- CMDB, Inventardatenbank
- Deployment
  - Ansible
  - Terraform
  - SaltStack
  - Puppet
  - Chef
  - ...
- Netzwerkscan / Piggybackdaten von VMs und Containern

Robert Sander <r.sander@heinlein-support.de>

Themen für Automatisierung

- Konfiguration
  - Hosts
  - Ordner
  - Host Tags
  - Hostgruppen
  - Servicegruppen
  - Service Discovery
  - Agenten backen & signieren
  - Time Periods
  - Nutzer
  - Kontaktgruppen
  - Business Intelligence
  - Passwörter
- Operating
  - Probleme quittieren
  - Wartungszeiten einrichten
- Status abfragen
  - Host
  - Service Check
- Änderungen aktivieren

Alte APIs: WATO & Multisite

- WATO-API wird durch REST-API abgelöst
  - in Übergangsphase noch vorhanden
- noch nicht möglich:
  - Regelsätze bearbeiten
  - Instanzen verwalten
  - Metriken & Graphen anzeigen
  - SLAs anzeigen
- wird aber noch implementiert
- Multisite-API noch vorhanden
- Export von Views
  - CSV
  - JSON
- Einfach Daten extrahieren
  - auch aus HW/SW-Inventory

Robert Sander <r.sander@heinlein-support.de>

Wie Automatisieren?

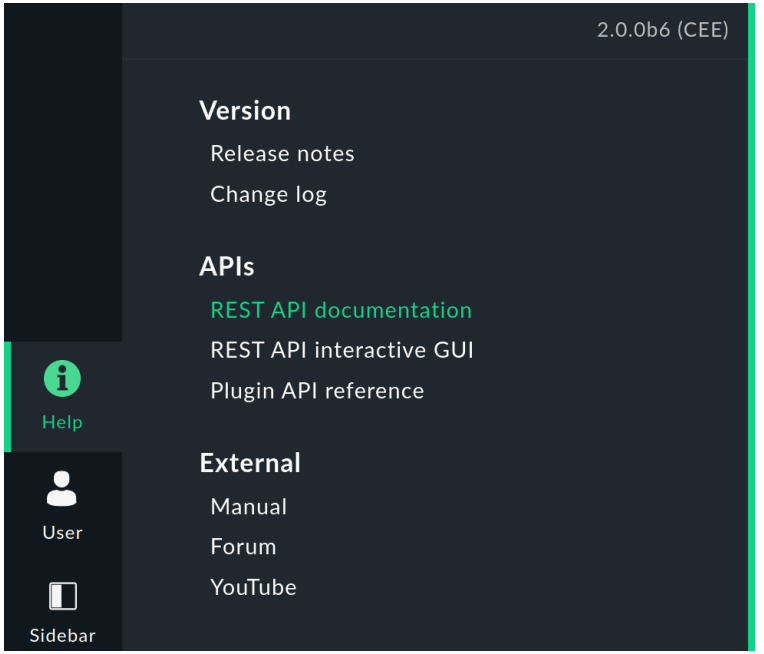
Robert Sander <r.sander@heinlein-support.de>

CheckMK – die neue REST-API

- **RE**presentational **S**tate **T**ransfer
- folgt OpenAPI Spezifikation OAS 3.x
- Protokoll: HTTP
- Kodierung: JSON, UTF-8, ISO-8601
- Authentifizierung: Basic oder Bearer
- Versionierung
- Dokumentation eingebaut inkl. Beispiele
- Fehler via HTTP Status-Codes
- Zusatzinformationen: „Hypermedia as the Engine of Application State“ (HATEOAS)

CheckMK – die neue REST-API

- Dokumentation direkt erreichbar
- mehr in Demo
- Online im Handbuch:
  - https://docs.checkmk.com/2.0.0/de/rest_api.html



REST-API – Authentifizierung

- Bearer-Auth
  - per zusätzlichem HTTP-Header „Authorization“ übermittelt
  - Zugangsdaten von CheckMK Automation-Account
  - `Authorization: Bearer $Username $Secret`
- HTTP Basic-Auth
  - muss eigens im Apache-Webserver der Instanz eingerichtet werden
  - für spezielle Installationen gedacht

REST-API – Beispiele – Host anlegen

```
→ API_URL="https://$HOST_NAME/$SITE_NAME/check_mk/api/v0"
  curl \
      --request POST \
      --header "Authorization: Bearer $USERNAME $PASSWORD" \
      --header "Accept: application/json" \
      --header "Content-Type: application/json" \
      --data '{
            "attributes": { "ipaddress": "192.168.0.123" },
            "folder": "\/",
            "host_name": "example.com"
          }' \
      "$API_URL/domain-types/host_config/collections/all"
```

REST-API - Beispiele - Service-Erkennung durchführen

→
```
API_URL="https://$HOST_NAME/$SITE_NAME/check_mk/api/v0"
curl \
    --request POST \
    --header "Authorization: Bearer $USERNAME $PASSWORD" \
    --header "Accept: application/json" \
    "$API_URL/objects/host/example.com/actions/discover-services/mode/
tabula-rasa"
```

REST-API – Beispiele – Änderungen aktivieren

```
→ API_URL="https://$HOST_NAME/$SITE_NAME/check_mk/api/v0"
  curl \
      --request POST \
      --header "Authorization: Bearer $USERNAME $PASSWORD" \
      --header "Accept: application/json" \
      --header "Content-Type: application/json" \
      --data '{
            "redirect": false,
            "sites": [ "mysite" ]
          }' \
      "$API_URL/domain-types/activation_run/actions/activate-changes/invoke"
```

Robert Sander <r.sander@heinlein-support.de>

Einfacher Automatisieren

API-Wrapper für Python

- https://github.com/HeinleinSupport/check_mk_extensions/tree/cmk2.0/check_mk_api
- Momentan:
  - add_host()
  - get_host()
  - get_all_hosts()
  - delete_host()
  - edit_host()
  - disc_host()
  - activate()
  - bake_agents()
  - set_downtime()
  - revoke_downtime()
  - view()

API-Wrapper – Beispiel

```
→ import checkmkapi
→ cmk = checkmkapi.CMKRESTAPI(url, username, secret)
→ cmk.add_host('exmaple.com',
                '/',
                attributes={'ipaddress': '192.168.0.123'})
→ cmk.disc_host('example.com')
→ cmk.activate()
```

API-Wrapper – Beispiel

- https://github.com/HeinleinSupport/check_mk_extensions/tree/cmk2.0/helper/bin
- `vm_parent.py`
  - liest den Service Check „ESX Hostsystem“ aus
  - nimmt den ESX-Hostnamen von dort
  - trägt ihn als Parent in VM-Hostkonfiguration ein
- `container_parent.py`
  - liest den Service Check „Docker container status aus
  - nimmt den Docker-Node-Namen von dort
  - trägt ihn als Parent in Container-Hostkonfiguration ein

API-Wrapper – Beispiel

- https://github.com/HeinleinSupport/check_mk_extensions/tree/cmk2.0/data2tag
- `data2tag.py`
  - Setzt Host-Tags aus Monitoring-Daten
  - SW-Inventory Betriebssystem → Tag-Gruppe „opsys“
  - SW-Inventory Softwarepakete → Tags für installierte Software

Robert Sander <r.sander@heinlein-support.de>

API-Wrapper – Beispiel

- https://github.com/HeinleinSupport/check_mk_extensions/tree/cmk2.0/data2label
- `data2label.py`
  - Setzt Host-Labels aus Monitoring-Daten
  - SW-Inventory Betriebssystem → Label „opsys/linux:yes“ oder „opsys/windows:yes“
  - SW-Inventory Softwarepakete → Label „software/apache:installed“ oder „software/docker:installed“

Demo

Robert Sander <r.sander@heinlein-support.de>

Soweit, so gut.

Gleich sind Sie am Zug:
Fragen und Diskussionen!

Wir suchen:
Admins, Consultants, Trainer!

Wir bieten:
Spannende Projekte, Kundenlob, eigenständige Arbeit, ein tolles Team, Work-Life-Balance

...und natürlich: Linux, Linux, Linux...

http://www.heinlein-support.de/jobs

Robert Sander <r.sander@heinlein-support.de>

Heinlein Support hilft bei allen Fragen rund um Linux-Server

HEINLEIN AKADEMIE

Von Profis für Profis: Wir vermitteln die oberen 10% Wissen: geballtes Wissen und umfangreiche Praxiserfahrung.

HEINLEIN HOSTING

Individuelles Business-Hosting mit perfekter Maintenance durch unsere Profis. Sicherheit und Verfügbarkeit stehen an erster Stelle.

HEINLEIN CONSULTING

Das Backup für Ihre Linux-Administration: LPIC-2-Profis lösen im CompetenceCall Notfälle, auch in SLAs mit 24/7-Verfügbarkeit.