Eigene Checks erstellen

- Einleitung. Muss es immer ein echtes Plugin sein?
- Lokaler Check

Einleitung. Muss es immer ein echtes Plugin sein?

Beschreibung:

Checkmk umfasst fast 2.000 fertige Checkplugins für alle nur denkbare Hardware und Software. Diese werden vom Checkmk-Team gepflegt, und jede Woche kommen neue dazu. Daneben gibt es auf der Checkmk Exchange weitere Plugins, die von unseren Anwendern beigesteuert werden.

Und trotzdem gibt es immer wieder Situationen, in denen ein Gerät, eine Anwendung oder einfach nur eine bestimmte Metrik, die für Sie wichtig ist, noch von keinem dieser Plugins erfasst ist — vielleicht auch einfach deshalb, weil es sich dabei um etwas handelt, dass in Ihrer Firma entwickelt wurde und es daher niemand anders haben kann.

Methode	So geht’s	Vorteile	Nachteile
Lokaler Check	Checkmk-Agent um einfaches Skript erweitern	Geht sehr einfach, ist in allen Programmiersprachen möglich, welche das Betriebssystem des überwachten Hosts anbietet, unterstützt sogar Serviceerkennung	Konfiguration der Schwellwerte nur beim Agenten selbst, für komplexere Dinge unkomfortabel, keine Unterstützung für SNMP
Nagios-kompatibles Checkplugin	Plugin per *MRPE* vom Windows- oder Linux-Agenten aufrufen lassen	Zugriff auf alle vorhandenen Nagios-Plugins, auch hier freie Wahl der Programmiersprache	Konfiguration der Schwellwerte nur beim Agenten selbst, Keine SNMP-Unterstützung durch Checkmk, keine Serviceerkennung möglich
Logmeldungen auswerten	*Meldungen* überwachen per Event Console	Keine Entwicklung notwendig sondern nur aufstellen von Regeln in der Event Console	Geht nur, wenn passende Logmeldungen vorhanden sind, kein gesicherter aktueller Status, kein Erfassen von Metriken, keine konfigurierbaren Schwellwerte

Methode	So geht’s	Vorteile	Nachteile
Echtes Checkmk-Plugin	Wird in diesem Artikel erklärt	Fügt sich zu 100% in Checkmk ein, automatische Serviceerkennung, zentrale Konfiguration der Schwellwerte über die grafische Oberfläche, sehr performant, unterstützt SNMP, automatische Host- und Servicelabels möglich, unterstützt HW/SW-Inventur, Unterstützung durch Standardbibliotheken von Checkmk	Erfordert mehr Einarbeitungszeit sowie Kenntnisse in der Programmsprache Python

Lokaler Check

Skript erstellen

Sie können einen lokalen Check in jeder beliebigen Programmiersprache schreiben, die der Zielhost unterstützt. Das Skript muss so konstruiert sein, dass es pro Check eine Statuszeile ausgibt, die aus vier Teilen besteht. Hier ist ein Beispiel:

```
0 "My service" myvalue=73 My output text which may contain spaces
```

Die vier Teile sind durch Leerzeichen getrennt und haben folgende Bedeutung:

Beispielwert	Bedeutung	Beschreibung
`0`	Status	Der Zustand des Services wird als Ziffer angegeben: `0` für OK, `1` für WARN, `2` für CRIT und `3` für UNKNOWN. Alternativ ist es möglich, den Status dynamisch berechnen zu lassen: dann wird die Ziffer durch ein `P` ersetzt.
`My service`	Service-Name	Der Name des Services, wie er in Checkmk angezeigt wird, in der Ausgabe des Checks in doppelten Anführungszeichen. Falls der Service-Name keine Leerzeichen enthält, können Sie sich die Anführungszeichen sparen.
`myvalue=73;65;75`	Wert und Metriken	Metrikwerte zu den Daten. Sie finden im Kapitel zu den Metriken näheres zum Aufbau. Alternativ können Sie ein Minuszeichen setzen, wenn der Check keine Metriken ausgibt.
`My output text which may contain spaces`	Statusdetail	Details zum Status, wie sie in Checkmk angezeigt werden. Dieser Teil kann auch Leerzeichen enthalten.

Zwischen den einzelnen Teilen der Ausgabe und dem ersten Text des Statusdetails muss immer ein Leerzeichen stehen. Alles danach wird zum Statusdetail gezählt, weswegen dann auch Leerzeichen erlaubt sind.

Wenn Sie wegen einer möglichen Ausgabe unsicher sind, können Sie diese einfach testen, indem Sie ein kleines Skript mit dem Kommando `echo` schreiben. Fügen Sie in das `echo`-Kommando Ihre Ausgabe ein, die Sie testen möchten. Achten Sie darauf, die Anführungszeichen für den Service-Namen mit `\` zu maskieren, damit diese Zeichen nicht vom `echo`-Kommando interpretiert werden: Linux:

```
#!/bin/bash
echo "0 \"My 1st service\" - This static service is always OK"
```

Windows:

```
@echo off
echo 0 "My 1st service" - This static service is always OK
```

Beide Skripte führen in der Ausgabe zum gleichen Ergebnis:

```
0 "My 1st service" - This static service is always OK
```

Für Checkmk ist nur diese Ausgabe relevant, nicht wie Sie diese Ausgabe erzeugt haben.

Sie können übrigens beliebig viele Ausgaben in einem Skript erzeugen. Für jede ausgegebene Zeile wird dann ein eigener Service in Checkmk erstellt. Daher sind in der Ausgabe auch keine Zeilenumbruchzeichen erlaubt — es sei denn, sie sind maskiert, zum Beispiel für eine mehrzeilige Ausgabe in Checkmk.

Wie Sie prüfen, ob das lokale Skript vom Agenten richtig aufgerufen wird, sehen Sie in der Fehleranalyse.

Skript verteilen

Nachdem das Skript geschrieben ist, können Sie es an die entsprechenden Hosts verteilen. Der Pfad unterscheidet sich je nach Betriebssystem. Eine Liste der Pfade finden Sie in Dateien und Verzeichnisse weiter unten.

Vergessen Sie nicht, das Skript auf unixoiden Systemen ausführbar zu machen. Der Pfad in dem Beispiel bezieht sich auf Linux:

```
root@linux# chmod +x /usr/lib/check_mk_agent/local/mylocalcheck
```

Wenn Sie die Agentenbäckerei nutzen, können Sie das Skript auch regelbasiert verteilen. Mehr zur Regelerstellung erfahren Sie im Kapitel Verteilung über die Agentenbäckerei.

Den Service ins Monitoring aufnehmen

Bei jedem Aufruf des Checkmk-Agenten wird auch der im Skript enthaltene lokale Check ausgeführt und an die Ausgabe des Agenten angehängt. Die Service-Erkennung funktioniert also wie bei anderen Services auch automatisch: