1 Einleitung

Mit dem Begriff »Monitoring« werden Systeme bezeichnet, die den Zustand von Geräten und Programmen überwachen und beim Abweichen vom gewünschten Zustand Alarmmeldungen verschicken.[1]

Ohne Monitoring können die Betreuer von IT-Systemen nur dann reagieren, wenn ein Anwender eine Störung meldet oder sie selbst durch sehr zeitaufwendige manuelle Kontrolle oder per Zufall ein Problem feststellen. Beim heute üblichen Zahlenverhältnis zwischen Systemen und Betreuern ist eine manuelle Kontrolle nicht mehr zu realisieren und wenn ein Anwender ein Problem feststellt, ist es eigentlich schon zu spät, da genau diese Probleme ja verhindert werden sollen.

Um diese Problematik zu lösen, werden Monitoring-Systeme benötigt, die versuchen, Schwierigkeiten frühzeitig zu bemerken, damit sie behoben werden können, bevor sie größeren Schaden anrichten können oder bevor Benutzer davon betroffen sind. Dafür gibt es verschiedene Ansätze:

- Ermitteln von Ergebnissen einer eigenen Statusprüfung und Logs von Anwendungen oder Hardware.
- Ende-zu-Ende-Checks wie das Versenden einer E-Mail und Prüfen, ob sie auch ankommt.
- Simulieren einer Nutzung eines Dienstes wie das Aufrufen einer Website.
- Prüfen von Eckdaten eines Systems wie das Abfragen der Festplattenauslastung mit Bordmitteln.

Icinga 2 bietet dabei die Möglichkeit, alle oben genannten Verfahren abzudecken, spezialisiert sich jedoch auf die letzten beiden Ansätze. Diese Flexibilität rührt daher, dass Icinga 2 selbst keine Funktionalität zur direkten Überwachung enthält, aber sehr gut darin ist, »Plugins«, also ausführbare Dateien, die ein paar Anforderungen erfüllen, laufen zu lassen und die Ausgabe zu interpretieren. Falls sich also eine Überwachungsmöglichkeit in einer ausführbaren Datei realisieren lässt, kann sie auch in Icinga integriert werden.

[1] Dies ist keine offizielle Definition, zeigt aber Sinn und Zweck des in diesem Buch beschriebenen Tools Icinga 2.

1.1 Es war einmal...

Icinga 2 ist ein sehr modernes Monitoring-System, das keinen Code von seinen geistigen Vorgängern enthält, aber fast alle bewährten Konzepte weiterverwendet. Diese Ähnlichkeit erleichtert vor allem Umsteigern die Arbeit mit Icinga 2 deutlich. Die folgenden Zeilen sollen einen kurzen Überblick über die bisherige Entwicklung geben.

1.1.1 Icinga

Im April 2009 wurde das Icinga-Projekt gegründet, das einen Fork des Nagios-Monitoring-Systems entwickeln sollte, da viele Community-Mitglieder unglücklich über den Umgang mit eingereichten Patches und den Umgang von Nagios Enterprises mit der Nagios-Community waren. Bei einem Fork wird der aktuelle Stand des Quellcodes eines Programms von verschiedenen Entwicklerteams in unterschiedliche Richtungen weiterentwickelt. Häufig wird dabei das bisherige Team aufgespalten und je ein Teil entwickelt vom gleichen Ausgangspunkt aus in verschiedene Richtungen weiter. Üblicherweise behält dabei ein Team den Namen des Produkts bei und das andere sucht sich einen neuen. Beim Icinga-Team fiel die Wahl auf »Icinga«, das Zulu-Wort für »Ausschau halten«.

Der Fork, und damit auch die weiteren Schritte wie die Evolution von Icinga zu Icinga 2, war insbesondere auch möglich, da einige Firmen bereit waren, Entwicklern Zeit zu sponsern, um die Entwicklung von Icinga voranzutreiben. Dies geschah meist aus Eigeninteresse, da sie Monitoring auf Nagios-Basis bei sich oder ihren Kunden erfolgreich einsetzten und eine Lösung für die schleppende Entwicklung suchten. Der Fork war nötig, da zuvor zwar entwickelt wurde, die daraus resultierenden Patches jedoch nicht in Nagios einflossen. Das Icinga-Team bemüht sich, schneller auf Input aus der Community zu reagieren, und hat auch genug »Stammmitglieder«, um die Entwicklung voranzutreiben. Eine dieser Firmen ist die NETWAYS GmbH[2] aus Nürnberg in Deutschland, die einige der Icinga-Kernentwickler angestellt hat, die entweder im Kundenauftrag oder als Dienst an der Community weiter an Icinga, Icinga 2 und anderen Tools aus diesem Ökosystem arbeiten. Auch die Autoren dieses Buchs haben von der NETWAYS GmbH Zeit gesponsert bekommen, um dieses Projekt zu verwirklichen.

[2] http://www.netways.de

1.1.2 Icinga 2

Um einige sehr tief im Code verwurzelte Einschränkungen aufzulösen, hat sich das Icinga-Team entschlossen, mit dem Code bei null zu beginnen und alles komplett neu zu schreiben. Das Resultat ist das Release von Icinga 2 Version 2.0 vom 16. Juni 2014. Darin wurden die bekannten Konzepte in der Bedienung weitergeführt, aber auch einige sehr wichtige Neuerungen eingeführt.

Diese Neuerungen umfassen unter anderem:

- eine Domain Specific Language (DSL) zur Konfigurationi,
- den integrierten Cluster-Stack, zur Kommunikation von verschiedenen Icinga-Instanzen untereinander,
- das Verteilen von Teilen der Konfiguration an beliebig viele andere Icinga 2 Instanzen,
- besseres Ausnutzen der verfügbaren Ressourcen durch Multithreading,
- der Einsatz als Agent,
- die API, mit der u. a. zu überwachende Objekte zur Laufzeit hinzugefügt werden können.

Die neue regelbasierte Sprache erlaubt eine deutlich flexiblere Konfiguration. Viele Objekte, die früher einzeln definiert werden mussten, können nun automatisch durch Auswerten von Regeln und Funktionen angegeben werden. Das spart nicht nur sehr viel Tipparbeit, sondern schützt auch vor Fehlern, da einmal definierte Regeln natürlich auch für neue Objekte gelten.

Um Demilitarized Zones (DMZs) oder entfernte Netzwerke zu überwachen, wurden auch bei den Vorgängern von Icinga 2 schon mehrere Instanzen verwendet, die miteinander kommunizieren. Allerdings enthielten diese keinen nativen Weg, mehrere Instanzen miteinander zu verbinden, weshalb verschiedenste Lösungen dafür entwickelt wurden. Diese waren jedoch teilweise kompliziert zu konfigurieren und oft auch nicht performant genug, um die Zeit zwischen Auftreten einer Störung und der Alarmierung ausreichend kurz zu halten. Dieses Verbinden von Instanzen wurde auch genutzt, um die auftretende Last zwischen mehreren Hosts zu verteilen und in Verbindung mit Clustertools wie `corosync`[3] und `pacemaker`[4] Hochverfügbarkeit zu erreichen.

Durch die neue Clusterfunktionalität können nicht nur lastverteilte und hochverfügbare Cluster mit Icinga 2 Bordmitteln erreicht werden, auch untergeordnete Icinga 2 Instanzen in anderen Netzsegmenten können Ergeb-

[3] http://corosync.github.io/corosync
[4] http://clusterlabs.org/pacemaker

nisse von Checks, die sie ausführen, an zentrale Instanzen weiterschicken. Diese zeigen einen Gesamtüberblick an und alarmieren.

Über die Clusterverbindung kann auch von zentralen Systemen aus eine Konfiguration an untergeordnete Instanzen ausgebracht werden.

Die Vorgänger von Icinga 2 liefen als ein einziger Thread und konnten so auch nur einen CPU-Kern ausnutzen. Zwar waren die gestarteten Plugins immer schon eigene Threads, dennoch wurde dadurch einer Installation eine Obergrenze an zu überwachenden Systemen gesetzt. Icinga 2 ist nun multithreaded und kann die zur Verfügung stehenden Ressourcen tatsächlich ausnutzen.

Durch den geringen Ressourcenverbrauch des Icinga 2 Kerns, die Clusterverbindung und die Konfigurationsverteilung hat sich eine neue Anwendungsmöglichkeit als Agent ergeben. Dabei wird eine Icinga 2 Instanz auf einem zu überwachenden Host installiert, die nur eben diesen überwacht. Die Ergebnisse leitet sie über die Clusterverbindung an eine zentrale Instanz weiter, die die Checks aller angeschlossenen Icinga 2 Instanzen sammelt und eine Übersicht zur Verfügung stellt. Konfiguriert wird sie von der zentralen Instanz aus. Für einen Vergleich aus Anwendersicht bietet die Icinga 2 Onlinedokumentation[5] einen umfassenden Überblick an.

1.1.3 Namen und Versionen

Aktuell wird nur noch die Version 2.x weiterentwickelt. Dieses Buch behandelt ausschließlich den 2.x-Zweig der Entwicklung. Mit dem Versionssprung von 1.x auf 2.x geht aber auch eine Umbenennung einher, daher heißt das Tool nun tatsächlich »Icinga 2 Version 2.x«.

Gleiches gilt für das moderne Webinterface Icinga Web 2. Es ist übrigens zu beiden existierenden Icinga-Varianten kompatibel, weshalb man durchaus Icinga 2 2.8.1 wie auch Icinga 1.13.3 mit Icinga Web 2 2.5.1 betreiben kann.

Die hier genannten Versionen waren zum Zeitpunkt, als die zweite Auflage dieses Buchs entstand, die jeweils aktuellsten.

1.2 Software-Komponenten

Ein Monitoring-System auf Basis von Icinga 2 besteht aus mehreren Teilkomponenten, so bildet der Core »Icinga 2« das Grundgerüst. Er regelt alle Abläufe, unter anderem z. B., wann was wie zu überwachen ist.

Die eigentliche Arbeit der Überwachung wird dann durch sogenannte Plugins erledigt, die vom Core aufgerufen werden und deren Ergebnisse

[5] https://www.icinga.com/docs/icinga2/latest/doc/23-migrating-from-icinga-1x/

dann verarbeitet werden. Plugins werden gesondert angeboten und nicht vom Icinga-Team betreut.

Die Komponente Icinga Web 2 ist ein in Hypertext Preprocessor (PHP) geschriebenes Framework zur Darstellung der ermittelten Ergebnisse der Überwachung mit Icinga 2. Als Schnittstelle zwischen Core und dem in einem Webserver laufenden Icinga Web 2 muss eine Datenbank verwendet werden. Unterstützt werden ausschließlich MariaDB respektive MySQL oder PostgreSQL. Das Beschicken dieser Datenbank ist über ein explizit einzuschaltendes Core-Feature zu aktivieren.

- Icinga 2
- Plugins, z. B. die Monitoring-Plugins[6]
- MariaDB-, MySQL- oder PostgreSQL-Datenbank
- Icinga Web 2
- Webserver mit PHP, standardmäßig Apache

Alle Komponenten werden in eigenen Projekten gepflegt und weiterentwickelt, somit wird auch jede unter einer eigenen Versionierung geführt. Darüber hinaus existieren viele weitere Komponenten, die über das bis hierher Erwähnte hinausgehen. So existiert mit dem Director eine Integration in Icinga Web 2 zur grafisch unterstützten Konfiguration oder diverse Projekte, die zeitliche Verläufe von Daten aus anderen Projekten wie Graphite oder InfluxDB einbinden.

1.3 Grundlagen

Icinga 2 ist wie auch sein Vorgänger auf Verfügbarkeitsüberwachung ausgelegt. Hierbei wird primär mit aktiven Checks gearbeitet. Als ein Check wird der Test eines Hosts oder Services bezeichnet. Jedes Gerät mit einer IP-Adresse wird als Host-Objekt betrachtet. Ein Service ist immer einem Host zugeordnet, ist also z. B. ein auf einem Host laufender Dienst oder eine zu überwachende Hardwarekomponente. Die Logik, wie Tests zu erfolgen haben, ist in sogenannten Plugins implementiert. Plugins sind externe Programme, die für jeden aktiven Check aufgerufen werden und über ihren Returncode den ermittelten Status zurückgeben.

Plugins können somit auch durch den Benutzer zu Testzwecken aufgerufen werden. Zu beachten ist, dass dies durch den Benutzer erfolgen sollte, unter dem der Icinga 2 Prozess ausgeführt wird. Auf RedHat-Systemen ist das *icinga*, auf Debian hingegen historisch bedingt der Benutzer *nagios*.

```
$ sudo -u icinga /usr/lib64/nagios/plugins/check_procs
PROCS OK: 100 processes | procs=100;;;0;
```

[6] http://www.monitoring-plugins.org

So ermittelt z. B. das Plugin `check_procs` durch einen Aufruf ohne Parameter die momentan auf dem System lauffähigen Prozesse. Die Ausgabe informiert über den ermittelten Status und die Anzahl der Prozesse. Der Text nach dem senkrechten Strich (Pipe) enthält Metriken, die das Plugin ermittelt hat, die sogenannten Performance-Daten. Eine genaue Erläuterung erfolgt in Kapitel 20 ab Seite 470.

Plugins sollten auch immer eine Hilfsfunktion bieten, damit ein Anwender dessen Funktionsumfang kennenlernen kann.

```
$ cd /usr/lib64/nagios/plugins
$ sudo -u icinga ./check_procs --help
check_procs v2.1.4 (nagios-plugins 2.1.4)
Copyright (c) 1999 Ethan Galstad <nagios@nagios.org>
Copyright (c) 2000-2014 Nagios Plugin Development Team
  <devel@nagios-plugins.org>

  Checks all processes and generates WARNING or CRITICAL
  states if the specified metric is outside the required
  threshold ranges. The metric defaults to number of
  processes. Search filters can be applied to limit the
  processes to check.

Usage:
check_procs -w <range> -c <range> [-m metric] [-s state]
 [-p ppid] [-u user] [-r rss] [-z vsz] [-P %cpu]
 [-a argument-array] [-C command] [-k] [-t timeout] [-v]

Options:
 -h, --help
    Print detailed help screen
 -V, --version
    Print version information
 --extra-opts=[section][@file]
    Read options from an ini file. See
    https://www.nagios-plugins.org/doc/extra-opts.html
    for usage and examples.
 -w, --warning=RANGE
   Generate warning state if metric is outside this range
 -c, --critical=RANGE
   Generate critical state if metric is outside this range

[...]
```

Plugins müssen vier Zustände zurückliefern können, OK, WARNING, CRITICAL und UNKNOWN. Diese werden als Returncode vom Plugin zurück gegeben. Kann ein Zustand nicht ermittelt werden, weil z. B. das Plugin nicht ausführbar ist oder eine gewisse Laufzeit überschreitet und damit in

einen Timeout läuft, ist der Rückgabewert UNKNOWN. Wann die anderen drei Zustände eintreten, kann bei nahezu allen Plugins durch die Angabe von Schwellwerten[7] beeinflusst werden.

Returncode	Servicestatus	Hoststatus
0	OK	UP
1	WARNING	UP
2	CRITICAL	DOWN
3	UNKNOWN	DOWN oder UNREACHABLE wird aus Abhängigkeiten berechnet

Tabelle 1-1: Plugin-Returncodes, Service- und Hoststatus

Diese Schwellwerte beziehen sich immer auf die ermittelten Metriken, bei `check_procs` ist das z. B. die Anzahl der lauffähigen Prozesse. So liefert das folgende Beispiel ein WARNING, da der entsprechende Schwellwert auf 90 gesetzt wurde und der Schwellwert für ein CRITICAL erst bei 120 liegt. Liefen also 120 Prozesse oder mehr, meldete das Plugin den Status CRITICAL.

```
$ sudo -u icinga ./check_procs -w 90 -c 120
PROCS WARNING: 100 processes | procs=100;90;120;0;
```

Der Returncode des letzten gelaufenen Prozesses kann mit `echo` ausgegeben werden:

```
$ sudo -u icinga ./check_procs -w 80 -c 90
PROCS CRITICAL: 100 processes | procs=100;80;90;0;
$ echo $?
2
```


Ein Host kann drei unterschiedliche Zustände annehmen, ein Service hingegen vier. Das Plugin erkennt jedoch nicht, ob ein Host oder ein Service zu überwachen ist. Es liefert immer Zustände zwischen 0 und 3 zurück, also vier wie für Services. Handelt es sich aber um einen Host, interpretiert Icinga 2 die Werte entsprechend. Die Werte OK und WARNING bedeuten, dass sich der Host im Status UP befindet. Die Unterscheidung zwischen DOWN und UNREACHABLE muss berechnet werden und setzt voraus, dass zwischen Hosts Abhängigkeiten definiert sind. Wie dies zu realisieren ist, wird in Abschnitt 12.3 ab Seite 184 erläutert.

Ruft Icinga 2 ein Plugin auf, wird von aktiven Checks gesprochen. Dies geschieht regelmäßig in einem anzugebenden Intervall und ist von Icinga 2 fortwährend zeitlich einzuplanen. Definiert man in der Konfiguration den

[7] Die Autoren wurden freundlich darauf hingewiesen, dass »Schwellenwert« korrekt wäre. Da sich »Schwellwert« aber eingebürgert hat, wird weiterhin dieser, dem Duden unbekannte, Begriff verwendet.

zu überwachenden Host oder Service als passiv, wartet Icinga 2 auf Informationen über den Status, der von einer externen Quelle geliefert werden muss. Mehr Informationen zu passiven Checks finden sich unter Kapitel 21.5 ab Seite 549.

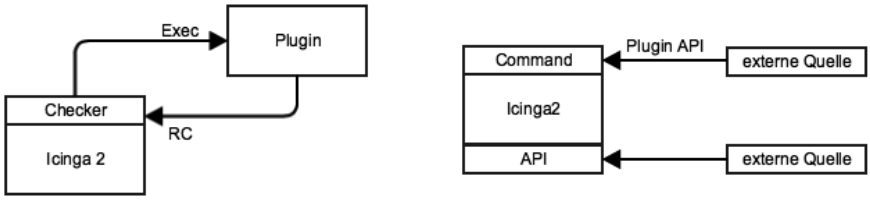

Abbildung 1-1: Aktive und passive Checks

Eine weitere wichtige Aufgabe des Monitorings ist die Benachrichtigung über erkannte Probleme. Hierbei ist es womöglich wünschenswert, über unterschiedliche Probleme verschiedene Personenkreise zu informieren. Diese Benachrichtigungen können je nach Grad der Wichtigkeit auf unterschiedlichen Wegen erfolgen; üblich sind Mails, Instant Messenger, SMS oder Voicecalls. Mehr Informationen zu Benachrichtigungen und wie sie konfiguriert werden, zeigt Kapitel 12 ab Seite 177.

Auch möchte man solche Benachrichtigungen gegebenenfalls unterdrücken, weil sie zeitlich in eine Downtime fallen und es sich damit um einen geplanten Ausfall handelt. Downtimes können bei Icinga 2 auf zwei Weisen angegeben werden: als wiederkehrendes Ereignis mit Scheduled Downtime bezeichnet, die in der Konfiguration hinterlegt werden muss, oder als einmaliges. Letzteres wird zur Laufzeit vom Benutzer erzeugt und Downtime genannt.

Icinga 2 ist modular aufgebaut und rüstet alle Funktionalität über sogenannte Features nach. So kümmert sich z. B. *notification* um die Benachrichtigung oder *checker* um den Aufruf von Plugins und damit um die Planung, wann aktive Checks auszuführen sind. Alle Features werden ausführlich in Abschnitt 4.3 ab Seite 49 vorgestellt.