Projektdokumentation

Fachinformatiker Systemintegration

Nagios-Migration zu Icinga in einer Krankenhaus-IT

Prüflingsnummer: 90064
Prüfungsausschuss: FISI Han 11 111
Projektzeitraum: 05.10.2015 – 09.10.2015

Ausbildungsbetrieb:
Siemens AG
Am Brabrinke 14
30519 Hannover

Projektdurchführender:
Sven Sperling
Bahnhofstr. 30
30900 Wedemark

Inhaltsverzeichnis

1. Einleitung 1
2. Ist-Zustand 1
3. Soll-Zustand 1
4. Nachweis des Zeitaufwandes 1
5. Projektdurchführung 1
5.1 Installation und Konfiguration der virtuellen Maschine im Xen Center 1
5.2 Installation des Betriebssystems Debian 7 64 Bit 2
5.3 Konfiguration des Betriebssystems Debian 7 64 Bit 3
5.3.1 Proxyeinstellungen 3
5.3.2 Update von Debian 7 über das Programm „apt-get“ 3
5.3.3 Vergabe von statischer IP 3
5.3.4 Installation von SSH und Konfiguration der Software „PuTTY“ auf Windows 7 4
5.4 Installation von Icinga 2 4
5.5 Icinga 2 Classic UI 4
5.5.1 Installation und Konfiguration der Weboberfläche 5
5.5.2 Neuen Benutzer hinzufügen und Passwort für bestehenden Benutzer ändern 5
5.6 Migration der Nagios-Überwachungen 6
5.6.1 User und Usergroups 6
5.6.2 Hosts 6
5.6.3 Hostgroups 6
5.6.4 Servicegroups 7
5.6.5 Services 7
5.6.6 Commands 7
5.6.7 Custom Plugins 7
5.7 Mail 9
6. Fehlerbehebung 9
7. Funktionstest 10
8. Fazit 10

Glossar I
Tabellenverzeichnis II
Abbildungsverzeichnis II
Quellenverzeichnis II
Anhang III
A. Projektphasen mit Zeitaufwand in Stunden III
B. Verzeichnisstruktur Icinga 2 Konfigurationen IV
C. user.conf V
D. Host Konfiguration V
E. groups.conf VI
F. servicegroup_switch.conf VI
G. Service Konfiguration VII
H. Command Konfiguration VII
I. Check MySQL Health Konfiguration VIII
J. Check NetApp Konfiguration IX
K. Check NT Konfiguration X
L. NRPE Konfiguration XI
M. Workaround Script für “Segmentation Fault” Fehler XII
N. Restart Script falls Icinga 2 abgestürzt XII
Bestätigung über die durchgeführte betriebliche Projektarbeit XIII

1. Einleitung

Im Rahmen meiner Umschulung bei der Siemens Professional Education in Hannover, absolvierte ich ein Praktikum im Vinzenzkrankenhaus Hannover. Dieses gehört zu der Vinzenz-Verbund Hildesheim GmbH, welcher mit Krankenhäusern, Altenpflegeheimen und einer Kindertagesstätte, sowie zwei Gesundheits- und Krankenpflegeschulen und Medizinische Versorgungszentren an sieben Standorten in Südniedersachsen und Nordhessen beauftragt ist. Das IHK-Projekt wird in den Räumen der IT-Abteilung des Vinzenzkrankenhauses in einer Testumgebung realisiert und getestet, bevor es schließlich in die Realumgebung integriert wird. In dieser Dokumentation wurden alle IP-Adressen, Benutzernamen, Passwörter und E-Mail Adressen aufgrund von Datenschutz anonymisiert. Die in kursiv dargestellten Begriffe werden im Glossar (siehe Seite I) erklärt.

2. Ist-Zustand

Um einen reibungslosen Ablauf im Krankenhausalltag zu gewährleisten, wird das gesamte IT-Netzwerk per Monitoring-Software überwacht. Für das Überwachen der Hosts, VPN, Switche, Festplattenkapazitäten, der Firewall und anderer Dienste wird die Monitoring-Software „Nagios Version 3.2.3“ verwendet. Laufen Warnungen und / oder Fehler bei der Überwachung auf, werden diese dem Administrator per Mail mitgeteilt. Daraufhin kann der Administrator entsprechend reagieren.

3. Soll-Zustand

Innerhalb des Vinzenzkrankenhauses soll ausnahmslos die Monitoring-Software „Icinga 2“ eingesetzt werden, da wichtige Updates der Nagios Software nicht mehr kompatibel zur virtuellen Umgebung sind. Um die Überwachung aufrechtzuerhalten und Datenverlust zu verhindern, wird zunächst eine Testumgebung aufgebaut. Dazu wird auf einem Citrix Xen-Server eine *virtuelle Maschine* mit dem Betriebssystem „Debian 7 / 64 Bit“ eingerichtet. Auf dem dann vorhandenen Betriebssystem wird die Monitoring-Software „Icinga 2“ installiert und konfiguriert.
Die vorhandenen Konfigurationen der Nagios-Überwachungen werden in die Icinga 2 Software migriert. Danach wird ein Funktionstest durchgeführt, um eventuelle Fehler zu erkennen und zu beheben. Nach erfolgreicher Migration der Nagios-Überwachungen zu Icinga 2 im Testbetrieb soll diese zu einem späteren Zeitpunkt auch im Realbetrieb vorgenommen werden.

4. Nachweis des Zeitaufwandes

Es entstand ein erheblicher Verzug bei der Migration der Nagios-Überwachungen zu Icinga 2, da der Aufbau der Konfigurationen in Icinga 2 vollkommen verschieden war. Die Online-Dokumentation der neuen Monitoring-Software war teilweise unvollständig und nicht nachvollziehbar. Attribute von Nagios waren in Icinga 2 nicht mehr vorhanden, wodurch einige zeitintensive Recherchen nötig wurden.
Die beiden Tabellen (siehe Seite III – A. Projektphasen mit Zeitaufwand in Stunden) stellen den geplanten und den tatsächlich benötigten Zeitaufwand für die einzelnen Projektphasen gegenüber.

5. Projektdurchführung

5.1 Installation und Konfiguration der virtuellen Maschine im Xen Center

Bevor das Betriebssystem auf dem Citrix Xen Center installiert werden konnte, musste eine *virtuelle Maschine* erstellt werden. Dazu wurde eine neue *virtuelle Maschine* mit Hilfe des Xen Center Assistenten (Reiter „VM“ -> New VM) erstellt. Es wurde das VM Template „Debian 7.0 Wheezy 64 Bit“ ausgewählt, ein Name für die *virtuelle Maschine* (VKH-Icinga) vergeben und das Debian 7 / 64 Bit Installationsmedium aus der ISO Bibliothek ausgewählt. Beim nächsten Schritt wurde folgende Auswahl getroffen: „Don´t assign this VM a home server. The VM will be started on any server with the necessary

resources“. Dadurch startet die *virtuelle Maschine* nicht auf einem bestimmten Server, sondern immer auf einem Server, der erreichbar ist und die notwendigen Ressourcen besitzt.
Es wurden 2 vCPUs (1 Socket mit 2 Kernen per Socket) und der RAM mit 8192 MB festgelegt. Um dem Betriebssystem und späteren Programmen das Installieren zu gewährleisten, wurde die Speichergröße der Festplatte auf 20 GB festgelegt. Auf der nächsten Seite wurde das virtuelle Netzwerkinterface konfiguriert, in welchem man die MAC Adresse automatisch und das Netzwerk „Bond 0+1“ manuell auswählen konnte. Nach den vorher genannten Vorgaben konnte die *virtuelle Maschine* erstellt werden. Nach Abschluss startete sie automatisch neu.

5.2 Installation des Betriebssystems Debian 7 64 Bit

Über den Reiter „Console“ im Citrix *XenCenter* konnte nun die Installation des Betriebssystems „Debian 7 / 64 Bit“ erfolgen. Es erschien der Installationsassistent, welcher durch verschiedene Optionen führte. Als erstes wurde nach der Sprache für den Installationsassistenten gefragt, die auch gleichzeitig die Standardsprache für das installierte System war. Es konnte nur „English“ als Sprache ausgewählt werden. In den nächsten Schritten wurde die Zeitzone auf Berlin und das Tastaturlayout auf „deutsch“ eingestellt. Der Hostname wurde auf „vkh-monitoring-icinga2“ und der Domänenname auf „vinzenz.local“ eingestellt. Das Passwort für den Benutzer „root“ wurde erstellt und auf der nächsten Seite noch einmal bestätigt. Für nicht administrative Aktivitäten wurde ein neuer Account mit Name und Passwort erstellt.

Es wurde die geführte Methode (Guided – use entired disk) und separate Partitionen für /home, /usr, /var und /tmp ausgewählt. Die erstellten Partitionen wurden auf die Festplatte geschrieben.

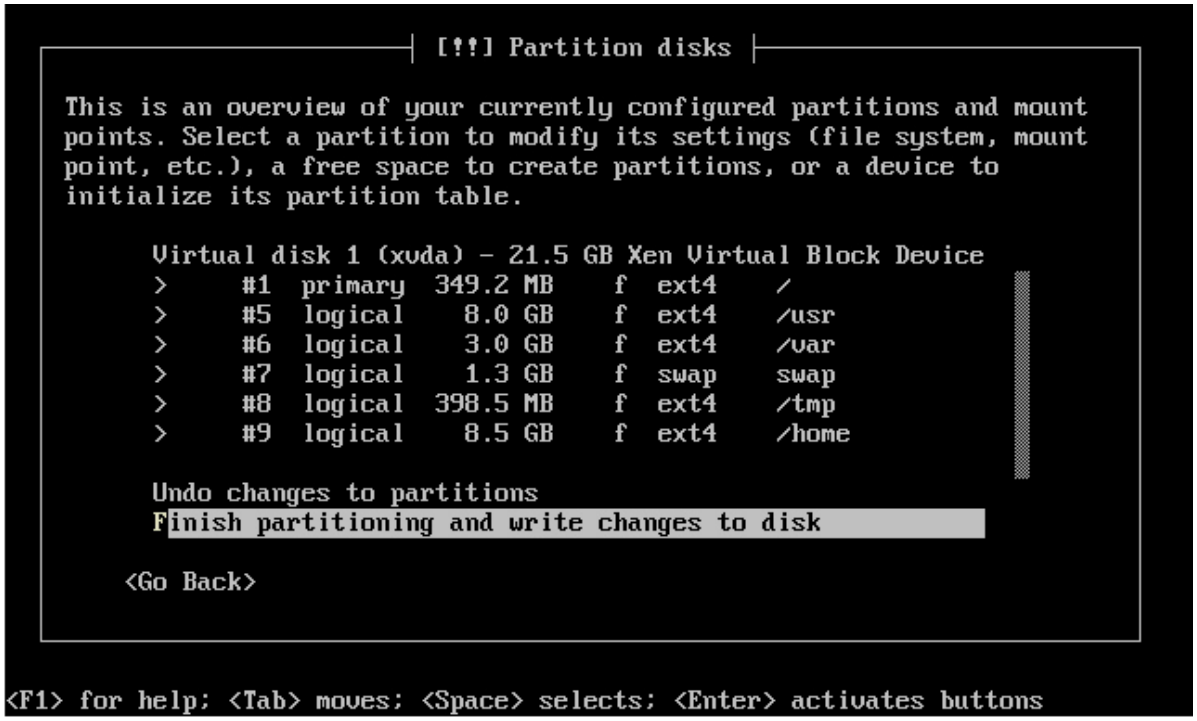


Abbildung 1: Partitionsübersicht

Vorteile von separaten Partitionen sind unter anderen:

- Verhinderung, dass Root Partition vollläuft und das System unbrauchbar wird
- bessere Performance
- Datensicherheit (z.B. Verschlüsselung der gesamten Partition, die sensitive Daten enthält)

Das Einstellen des Debian Archive Mirrors wurde übersprungen und zu einem späteren Zeitpunkt manuell konfiguriert. Abschließend wurde das Betriebssystem mit den ausgewählten Optionen installiert und war nach einiger Installationszeit einsatzbereit.

5.3 Konfiguration des Betriebssystems Debian 7 64 Bit

Um mit administrativen Rechten zu starten, musste sich der Benutzer mit dem Benutzernamen „root“ und dem dazu gehörigen Passwort anmelden.

5.3.1 Proxyeinstellungen

Da das gesamte IT-Netzwerk durch einen Proxy geschützt war, mussten einige Einstellungen vorgenommen werden, um die *virtuelle Maschine* auch für das Internet funktionstüchtig zu machen. Für wichtige Updates und um neue Programme installieren zu können, musste der Proxy für den Debian Archive Mirror eingestellt werden. Dazu wurde die Datei „apt.conf“, welche sich im Ordner „/etc/apt/“ befindet, mittels dem Editor „Nano“ geändert.

Code: /etc/apt/apt.conf

```
Acquire::http::Proxy "http://<name>:<passwort>@192.168.95.***:8080/";
```

Um Dateien von einer Website herunterzuladen, musste folgender Befehl benutzt werden:

Code:

```
http_proxy=http://192.168.95.***:8080/ wget --proxy-user=<name> --proxy-password=<passwort> <link>
```

Für das Herunterladen von einer verschlüsselten Website musste folgendes eingegeben werden:

Code:

```
https_proxy=http://192.168.95.***:8080/ wget  --no-check-certificate --proxy-user=<name> --proxy-password=<passwort> <link>
```

5.3.2 Update von Debian 7 über das Programm „apt-get“

Für Updates und zum Installieren anderer Programme musste die Datei „sources.list“ geändert werden. Folgendes wurde in die Datei hinzugefügt:

Code: /etc/apt/sources.list

```
deb http://ftp.de.debian.org/debian/ wheezy main
```

Befehl	Beschreibung
apt-get update	Aktualisiert die lokal vorhandene Paketliste
apt-get upgrade	Aktualisiert die lokal vorhandenen Pakete (Software)
apt-get install <PaketName>	Installiert das Paket mit dem Namen “PaketName”
apt-get remove <PaketName>	Entfernt ein Paket. Konfigurationsdateien bleiben erhalten
apt-get purge <PaketName>	Entfernt ein Paket. Konfigurationsdateien werden gelöscht
apt-get autoremove	Entfernt alle ungenutzten Pakete

Tabelle 1: Apt Befehlsübersicht

Es wurde zuerst die lokal vorhandene Paketliste und dann die lokal vorhandenen Pakete aktualisiert. Danach war das Betriebssystem auf dem aktuellsten Stand.

Code:

```
apt-get update && apt-get upgrade
```

5.3.3 Vergabe von statischer IP

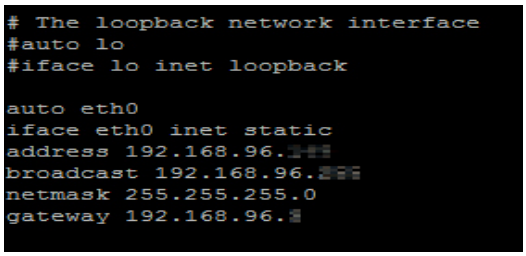


Abbildung 2: Konfiguration statische IP

Damit Icinga 2 Classic UI später im Webbrowser immer von der gleichen IP aufgerufen werden konnte, musste die bisher per DHCP automatisch zugewiesene IP Adresse auf eine statische IP Adresse geändert werden. Dieses wurde im Ordner „/etc/network/“ in der Datei „interfaces“ abgeändert.

5.3.4 Installation von SSH und Konfiguration der Software „PuTTY“ auf Windows 7

Um ein angenehmeres Arbeiten zu ermöglichen, z.B. durch mehrere Konsolenfenster, wurde die Remotesoftware „PuTTY“ verwendet. Dazu wurde auf dem Laborrechner, welcher mit dem Betriebssystem „Windows 7“ läuft, die *portable* Version von „PuTTY“ heruntergeladen und ausgeführt. Damit die Remotesoftware auf die *virtuelle Maschine* zugreifen konnte, musste der OpenSSH Server auf der virtuellen Maschine installiert und gestartet werden.

Code:

```
apt-get install openssh-server && /etc/init.d/ssh restart
```

PuTTY wurde mit folgenden Einstellungen ausgeführt:

- IP Adresse: 192.168.96.***
- Port: 22
- Connection Type: SSH

Beim erstmaligen Verbinden fragte PuTTY nach, ob dem SSH Schlüssel vertraut und der Schlüssel hinzugefügt werden kann. Dieses wurde mit einem „Yes“ akzeptiert. Danach war ein normales Arbeiten über die PuTTY Konsole möglich.

5.4 Installation von Icinga 2

Da sich die Monitoring-Software „Icinga 2“ nicht im normalen Debian Archive Mirror befand, musste der Update Mirror „debmon“ installiert werden. Dazu wurde über den Befehl „wget“ der Repository Key heruntergeladen und hinzugefügt:

Code:

```
cd /tmp && http_proxy=http://192.168.95.***:8080/ wget --proxy-user=<name> --proxy-password=<passwort>
http://debmon.org/debmon/repo.key && apt-key add repo.key
```

Danach wurden die Paketlisten aktualisiert, Icinga 2 heruntergeladen und installiert.

Code:

```
apt-get update && apt-get install icinga2
```

Icinga 2 startet automatisch bei einem Neustart. Es konnten aber auch verschiedene Befehle in die Konsole eingegeben werden, um Icinga 2 z.B. neu zu starten oder zu stoppen.

Befehl	Beschreibung
service icinga2 start	Startet die Monitoring-Software “Icinga 2”
service icinga2 stop	Stoppt die Monitoring-Software “Icinga 2”
service icinga2 restart	Stoppt und startet danach die Monitoring-Software “Icinga 2”
restart_icinga2	**Bitte zum Server Neustart verwenden!** (Seite 9 – 6. Fehlerbehebung)
service icinga2 checkconfig	Überprüft die Icinga 2 Konfigurationen auf Fehler

Tabelle 2: Icinga 2 Service Befehlsübersicht

5.5 Icinga 2 Classic UI

Da Icinga 2 ohne Frontend ausgeliefert wurde, musste eine Weboberfläche manuell installiert werden. Es wurde sich für die Weboberfläche „Icinga 2 Classic UI“ entschieden, da sie sehr dem alten Nagios Webinterface ähnelt und auch schon in einer anderen Abteilung des Vinzenzkrankenhauses verwendet wird. Durch das Frontend hat der Administrator die Möglichkeit, sich schnell einen Überblick über den Status der verschiedenen Hosts und Services zu verschaffen, sowie Servicechecks manuell zu starten oder auch Kommentare zu einzelnen Hosts / Servicechecks zu hinterlassen.

5.5.1 Installation und Konfiguration der Weboberfläche

Voraussetzung für das Installieren der Weboberfläche „Icinga 2 Classic UI“ war ein Webserver. Es wurde der „Apache 2“ Webserver gewählt. Danach konnte die Weboberfläche installiert werden.

Code:

```
apt-get install apache2 && apt-get install icinga2-classicui
```

Während der Installation von “Icinga 2 Classic UI” wurde das Passwort für den Benutzer „icingaadmin“ eingerichtet. Das Webinterface war unter „http://192.168.96.***/icinga2-classicui“ erreichbar, welches durch eine Authentifizierung mit dem Benutzernamen „icingaadmin“ und dem dazu gehörigen Passwort geschützt war.

Es wurden Logos für die Hosts eingefügt, um die Weboberfläche optisch aufzuwerten. Dazu wurde ein Logopaket heruntergeladen und in das Logoverzeichnis von Icinga 2 Classic UI entpackt. *Statusmap* Bilder wurden zu diesem Zeitpunkt nicht von Icinga 2 unterstützt**.**

Abbildung 3: Hosts mit Action Button und Logo

Code:

```
apt-get install unzip && http_proxy=http://192.168.95.***:8080/ wget --proxy-user=<name> --proxy-password=<passwort> http://ihkprojekt.huxxer.de/imagepak-andrade-2.0.zip && unzip -j imagepak-andrade-2.0.zip -d /usr/share/icinga2/classicui/images/logos/base && rm imagepak-andrade-2.0.zip
```

Die Verlinkung der „*Action URL*“ Buttons wurde mit in die Weboberfläche integriert. Da bei einer passwortgeschützten Administrationswebsite nur eine weiße Seite zu sehen war, wurde das Attribut „action_url_target“ von „main“ in „blank“ geändert. Somit war die Seite nicht mehr in die Weboberfläche integriert, sondern wurde als neuer Tab geöffnet. Sicherheitshalber wurde dieses bei dem Attribut „notes_url_target“ auch geändert. Außerdem wurde das „result_limit“ auf null gesetzt, damit alle Einträge der Services und Hosts auf der ersten Seite erscheinen.

Code: /etc/icinga2-classicui/cgi.conf

```
action_url_target = blank
notes_url_target = blank
result_limit=0
```

5.5.2 Neuen Benutzer hinzufügen und Passwort für bestehenden Benutzer ändern

Um das Passwort des bestehenden Benutzers „icingaadmin“ zu ändern, musste mit Hilfe des Befehls „htpasswd“ die Datei „htpasswd.users“ geändert werden.

Code:

```
htpasswd /etc/icinga2-classicui/htpasswd.users icingadmin
```

Nach Eingabe des Befehls wurde nach einem neuen Passwort gefragt und dieses musste bestätigt werden. Ein neuer Benutzer konnte mittels „htpasswd“ auch angelegt werden.

Code:

```
htpasswd /etc/icinga2-classicui/htpasswd.users <name>
```

Nach Eingabe wurde nach einem Passwort gefragt und dieses musste bestätigt werden.

5.5.3 Anpassen der Website Adresse

Eine HTML Weiterleitung wurde erstellt, damit die Weboberfläche auch geöffnet wird, falls nur die Webadresse „http://192.168.96.***“ im Webbrowser eingegeben wurde. Ansonsten hätte jedes mal die

komplette Icinga 2 Webadresse „http://192.168.96.***/icinga2-classicui“ eingegeben werden müssen. Dazu wurde in dem Ordner „/var/www“ die Datei „index.html“ geöffnet und abgeändert.

Code: /var/www/index.html

```
<html>
<head>
<meta http-equiv="refresh" content="0; URL=http://192.168.96.***/icinga2-classicui/">
</head>
<body>
</body>
</html
```

5.6 Migration der Nagios-Überwachungen

Die Konfigurationsdateien von Nagios wurden auf einen Fileserver exportiert. In der Nagiosumgebung hatten alle Konfigurationsdateien die Dateiendung „.cfg“. Dieses wurde bei Icinga 2 in die Dateiendung „.conf“ geändert. Sie wurden entsprechend der neuen Konfigurations -struktur und -syntax angepasst und in die *virtuelle Maschine* importiert. Es wurden Ordner für die Hosts, Services und Commands erstellt und die Konfigurationsdateien strukturiert eingegliedert (siehe Seite IV - B. Verzeichnisstruktur Icinga 2 Konfigurationen). Im Vergleich ist deutlich geworden, dass bei Icinga 2 eine etwas andere Syntax verwendet wird (Anführungszeichen und Gleichzeichen zwischen dem Attribut und dem dazugehörigen Wert), sowie teilweise andere Attributnamen verwendet werden. Im Allgemeinen wurde das Attribut „alias“ in „display_name“ und das Attribut „use“ in „import“ umbenannt. In den folgenden Punkten 5.6.1 – 5.6.6 werden die verschiedenen Konfigurationsdateien, sowie vorgenommene Änderungen beispielhaft erklärt.

5.6.1 User und Usergroups

Die User und Usergroups waren in Nagios unter der Datei „contacts.cfg“ gespeichert. Unter Icinga 2 wurden sie in der Datei „users.conf“ (siehe Seite V - C. user.conf) gespeichert. Das Objekt „User“ beinhaltete ein Kontakttemplate, Benutzernamen, den Anzeigename im Frontend, in welcher Gruppe der Benutzer ist, die E-Mail Adresse, sowie die Frage, ob der Benutzer bei Warnungen oder kritischen Fehlern über diese benachrichtigt werden sollte. Der „contact_name“ steht direkt vor den geschweiften Klammern und nicht mehr als einzelnes Attribut. In dem Objekt „UserGroup“ wurde nur die Benutzergruppe erstellt. Diese wurde später bei Bedarf in dem Objekt „User“ aktiviert. Hinzukommend wurde die Benutzergruppe direkt bei dem Benutzer eingetragen und nicht mehr im Gruppenobjekt.

5.6.2 Hosts

Physikalische Geräte wie Server, Router und Switches waren die so genannten Hosts (siehe Seite V – D. Host Konfiguration). Sie wurden durch ihre IP-Adresse identifiziert und konnten dadurch über die später erklärten Services überwacht werden. Die Attributnamen blieben weitestgehend gleich, bis auf die in Punkt 5.6 genannten Attributnamen. Das Attribut „alias“ fiel in der Hostdatei weg, da sonst der Hostname in der Weboberfläche mit dem Alias überschrieben worden wäre. Es wurde in einigen Hosts eine *Action URL* definiert. Dadurch konnte der Administrator in der Icinga 2 Classic UI durch einen Klick auf die entsprechende Schaltfläche z.B. auf die Administrationsseite eines Switches gelangen.

5.6.3 Hostgroups

In die Hostgroups (siehe Seite VI - E. groups.conf) wurden bestimmte Hosts zu Gruppen zusammengefasst. Somit konnten verschiedene Services auf den Gruppen angewendet werden und mussten nicht für jeden Host einzeln bearbeitet werden. In Nagios wurden die Mitglieder einer Gruppe unter „members“ zusammengefasst. In Icinga 2 wurden sie für einen einzelnen Host unter „assign where host.name == „Host1“ „ und bei mehreren Hosts unter „assign where host.name in [„Host1“, „Host2“, „Host3“]“ eingetragen.

5.6.4 Servicegroups

Es wurden bestimmte Services zu einer Gruppe (siehe Seite VI - F. servicegroup_switch.conf) zusammengefasst. Damit konnte wie bei den Hostsgroups doppelte Arbeit vermieden werden. Bei Nagios wurden die Service-Mitglieder unter „members“ zusammengefasst. In der neuen Monitoring-Software werden sie unter „assign where service.name in [„Service1“, „Service2“]“ zusammengefasst.

5.6.5 Services

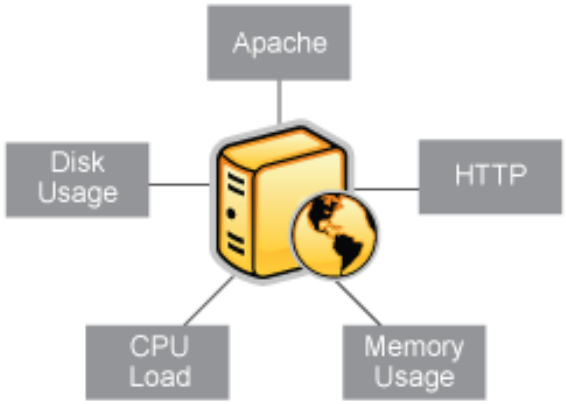


Abbildung 4: Arten von Services

Zentrale Objekte in der Überwachung sind die Services (siehe Seite VII - G. Service Konfiguration), welche eng mit den Hosts verbunden sind. Es wurden verschiedene Attribute von Hosts als Services definiert, z.B. CPU-Auslastung, laufende Prozesse, RAM-Auslastung und Festplatten-Belegungen. Zudem wurden verschiedene Services wie Ping Abfragen, DNS Abfragen oder auch http Abfragen festgelegt. Bei Nagios wurde das Attribut, um den Service in einer bestimmten Hostgruppe zu aktivieren als „hostgroup_name“ deklariert. In Icinga 2 wurde es in „assign where „hostgroupname“ in host.groups“ geändert. Ansonsten waren auch hier die üblich allgemeinen Attributnamensänderungen (siehe Punkt 5.4) angewandt worden. Angemerkt werden muss, dass mehrere Services niemals den gleichen Namen haben durften, da ansonsten Icinga 2 nicht gestartet werden konnte. Bei Nagios war dieses dagegen möglich.

5.6.6 Commands

Die Commands (siehe Seite VII - H. Command Konfiguration) teilten Icinga 2 mit, welche Programme, Skripte usw. ausgeführt werden sollten. Es definierte also einen Befehl. Dieses konnten Host- oder Serviceprüfungen und z.B. Benachrichtigungen sein. Es wurde ein Commandname, sowie andere zu importierende Befehle definiert. Auch wurden das jeweils benötigte Plugin und die dazugehörigen Befehlsargumente hineingeschrieben. Die einzig große Änderung war, dass die Argumente in einem extra Container innerhalb des Objektes geschrieben wurden und dass Zeiteinheiten wie z.B. „s“ für Sekunden oder „m“ für Minuten mit angegeben werden mussten.

5.6.7 Custom Plugins

Da es keine vorinstallierten Plugins für das Überprüfen der MySQL Datenbanken, des NetApp Speichers und kein *NRPE* (Nagios Remote Plugin Executor) Plugin gab, mussten diese manuell installiert werden. Es wurde ein Custom Plugins Ordner im Icinga 2 Plugins Ordner erstellt, um später genau nachvollziehen zu können, welche Plugins selbst hinzugefügt wurden.

Code:

```
mkdir /usr/lib/nagios/plugins/custom-plugins
```

Außerdem musste Icinga 2 der Custom Plugins Ordner in der Datei „constants.conf“ bekannt gegeben werden.

Code: /etc/icinga2/constants.conf

```
const CustomPluginDir = "/usr/lib/nagios/plugins/custom-plugins"
```

5.6.7.1 Plugin MySQL Health Check

Für die MySQL Datenbanken wurde das Plugin „check_mysql_health“ (siehe Seite VIII – I. Check MySQL Health Konfiguration) verwendet. Der Download lag nur im *Sourcecode* vor, ein so genannter „*Tarball*“,

weshalb zum Kompilieren und Erstellen des Plugins das Programm „make“ benötigt wurde. Außerdem war ein MySQL Client auf der virtuellen Maschine nötig.

Code:

```
apt-get install mysql-client make
```

Danach wurde der *Tarball* von der Projektwebsite heruntergeladen und im temporären Ordner entpackt, schließlich in das entpackte Archiv gewechselt und die heruntergeladene Datei gelöscht.

Code:

```
cd /tmp/ && https_proxy=http://192.168.95.***:8080/ wget --no-check-certificate --proxy-user=<name> --proxy-password=<passwort> https://labs.consol.de/assets/downloads/nagios/check_mysql_health-2.2.1.tar.gz && tar -xf check_mysql_health-2.2.1.tar.gz && cd check_mysql_health-2.2.1 && rm ../check_mysql_health-2.2.1.tar.gz
```

Der *Sourcecode* des MySQL Health Plugins musste kompiliert und das Plugin erstellt werden.

Code:

```
./configure && make && make install
```

Danach wurde das Plugin in den Custom Plugin Ordner verschoben und ausführbar gemacht.

Code:

```
mv /usr/local/nagios/libexec/check_mysql_health /usr/lib/nagios/plugins/custom-plugins/ && chmod +x /usr/lib/nagios/plugins/custom-plugins/check_mysql_health
```

5.6.7.2 Plugin Check NetApp

Das Netapp (*SAN-System*) Plugin (siehe Seite IX – J. Check NetApp Konfiguration) erforderte die Installation des Paketes „Perl“, sowie zweier Perl Schnittstellen. Danach wurde der NetApp Check *Tarball* auf die gleiche Weise wie das MySQL Health Check Plugin heruntergeladen und entpackt.

Code:

```
apt-get install perl libwww-perl libxml-libxml-perl
```

Code:

```
cd /tmp/ && https_proxy=http://192.168.95.***:8080/ wget --no-check-certificate --proxy-user=<name> --proxy-password=<passwort> https://github.com/willemdh/check_netapp_ontap/archive/v2.5.10.tar.gz && tar -xf v2.5.10.tar.gz && cd check_netapp_ontap-2.5.10 && rm ../v2.5.10.tar.gz
```

Das Plugin wurde in den Custom Plugins Ordner kopiert und ausführbar gemacht. Dann wurde die NetApp Perl SDK in den Perl Ordner kopiert.

Code:

```
cp check_netapp_ontap.pl /usr/lib/nagios/plugins/custom-plugins/ && chmod +x /usr/lib/nagios/plugins/custom-plugins/check_netapp_ontap.pl && cp NetApp/* /usr/lib/perl5
```

5.6.7.3 Plugin Check NT

Das Check NT Plugin (siehe Seite X – K. Check NT Konfiguration) erforderte nur das Herunterladen und Installieren eines Paketes aus dem Debian Repository. Danach war das Plugin einsatzbereit.

Code:

```
apt-get install monitoring-plugins
```

5.6.7.4 Plugin NRPE

Voraussetzung für das Benutzen des *NRPE* Plugins (siehe Seite XI – L. NRPE Konfiguration) war das Paket “openssl”. Folgend wurde das Plugin über das APT Programm heruntergeladen und installiert.

```
apt-get install openssl && apt-get --no-install-recommends install nagios-nrpe-plugin
```

5.7 Mail

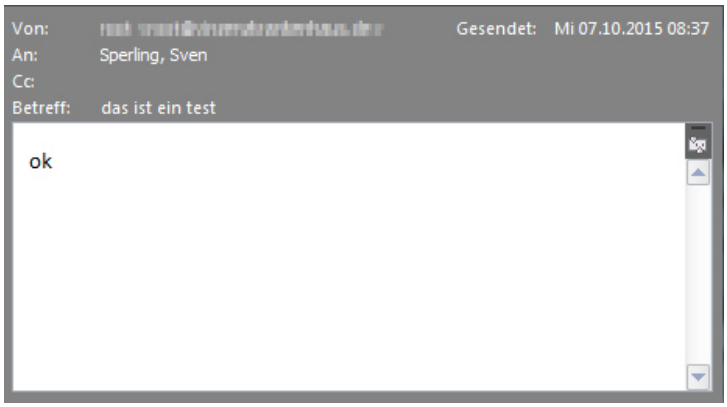


Abbildung 5: Erfolgreich zugestellte Testmail

Durch eine Mail werden die Administratoren bei Warnungen oder kritischen Fehlern benachrichtigt. Icinga 2 kann von Haus aus nicht selbstständig Mails versenden, sondern muss auf extra installierte Emailpakete zugreifen. Zum Versenden der Mails wurde das Paket „exim4“ installiert und konfiguriert. Bei der Installation kam es zu einigen Abfragen, die in der untenstehenden Tabelle 3 angegeben sind.

Code:

```
apt-get install exim4 && dpkg-reconfigure exim4-config
```

Abfrage	Eingabe
Type of mail configuration	mail sent by smarthost; no local mail
System mail name	webadresse.de
IP-addresses to listen on for incoming SMTP connections	127.0.0.1 ; ::1 ;
Other destinations for which mail is accepted	Bleibt leer
Visible domain name for local users	webadresse.de
IP address or host name of the outgoing smarthost	192.168.96.***
Keep number of DNS-queries minimal (Dial-on-Demand)	No
Split configuration into small files	No

Tabelle 3: Konfiguration Exim4

Der SMTP Server, sowie der Benutzername und das dazugehörige Passwort wurden für das Versenden von Mails in die Datei “/etc/exim4/passwd.client“ eingetragen.

Code:

```
<servername>:<benutzername>:<passwort>
```

Der Mailserver wurde neugestartet und danach die Konfiguration durch Senden einer Mail getestet.

Code:

```
invoke-rc.d exim4 restart && echo 'ok' | mail -s 'das ist ein test' name@webadresse.de
```

6. Fehlerbehebung

Es traten Probleme bei dem Plugin „MySQL Health Check“, sowie beim Neustart des Icinga 2 Servers auf. Desweiteren stürzte der Icinga 2 Server sporadisch ab. Bei Neustart des Icinga 2 Servers meldete die Konsole unregelmäßig den Fehler „Segmentation Fault“. Dadurch konnte der Server nicht neu gestartet werden. Die Überprüfung der Systemlogdatei und eine danach folgende Internetrecherche ergab, dass es sich wahrscheinlich um einen Fehler in der GNU Standard C++ Bibliothek und/oder in Icinga 2 handelte. Als *Workaround* wurde ein Shellscript mit dem Namen „restart_icinga.sh“ (siehe Seite XII – M. Workaround Script für “Segmentation Fault” Fehler) erstellt und mittels dem Befehl „chmod +x restart_icinga.sh“ ausführbar gemacht.

Nach Ausführen des Shellscripts wird der Server so oft neu gestartet, bis Icinga 2 ohne die Fehlermeldung erfolgreich durchgestartet ist. Der Fehler wurde mit diesem Skript nicht behoben, sondern lediglich umgangen. Folglich wird das Shellscript solange verwendet, bis der Fehler von den Softwareentwicklern entfernt wird.

Nachdem der *Sourcecode* des „MySQL Health Check“ Plugins kompiliert und installiert wurde, konnte es sich nach dem Ausführen nicht zur MySQL Datenbank verbinden. Es erschien folgende Fehlermeldung:

Code:

```
CRITICAL – cannot connect to information_schema. Host ‚vkh-monitoring-icinga2.vinzenz.local‘ is not allowed to connect to this MySQL server
```

Durch Hinzufügen der IP des Monitoring Servers in die Whitelist des MySQL Servers konnte dieses Problem gelöst werden.

Gründe für das Abstürzen des Icinga 2 Servers konnten nicht ermittelt werden. Ein Shellscript mit dem Namen „check_if_icinga2_started.sh „ (siehe Seite XII – N. Restart Script falls Icinga 2 abgestürzt) wurde erstellt und über den Befehl „chmod +x check_if_icinga2_started.sh“ ausführbar gemacht. Es wurde für das Shellscript ein Eintrag in die *Crontabdatei* vorgenommen. Dadurch wird das Shellscript jede Minute ausgeführt. Nach der Ausführung überprüft das Skript, ob der Icinga 2 Prozess gestartet ist und falls nicht wird der Prozess gestartet. Dadurch konnte sichergestellt werden, das Icinga 2 nicht länger als eine Minute außer Betrieb ist.

Code:

```
crontab -e
```

Code: crontab

```
SHELL=/bin/bash
PATH=/usr/local/bin:/usr/local/sbin:/sbin:/usr/sbin:/bin:/usr/bin:/usr/bin/X11

*/1 * * * * /etc/icinga2/check_if_icinga2_started.sh
```

7. Funktionstest

Nachdem die Nagioskonfigurationen erfolgreich in das Icinga 2 Format umgewandelt worden waren, konnte ein Funktionstest stattfinden. Da die Statusmeldungen aller Host- und Servicechecks auf „OK“ standen, musste ein Servicecheck manipuliert werden, um einen Test durchführen zu können. Testweise wurde dafür der Servicecheck „Check Core_Switch_2 Port A2 - Core-Switch“ genommen, so dass eine kritische Warnung ausgegeben wurde und eine Benachrichtigung per Mail erfolgen sollte.
Dazu wurde in der Datei „command_core_switch_2.conf“ in dem Objekt „check_hp_core_2_a2“, das Attribut „-c“ (kritischer Wert) von „1“ auf „0“ geändert. Der Icinga 2 Server wurde neugestartet. Nach kurzer Zeit meldete Icinga 2 eine kritische Statusmeldung. Es wurde automatisch eine E-Mail an den Icingaadmin gesendet, nachdem fünf Checkversuche durchgeführt wurden. Der Administrator hätte daraufhin im Ernstfall schnellstmöglich reagieren können. Somit wurde der Funktionstest erfolgreich absolviert.

8. Fazit

Nach anfänglichen Schwierigkeiten bei der Umstellung zu Icinga 2 wurden die Erwartungen an die neue Monitoring-Software vollkommen erfüllt. Icinga 2 mit der Classic UI Weboberfläche läuft stabil und ohne Probleme. Die Weboberfläche ist leicht verständlich und einfach zu bedienen. Neue Hosts und Services können leicht hinzugefügt werden. Somit ist die Software ein vollwertiger Ersatz für Nagios, die ausgebaut und aktualisiert werden kann.
Zwischenzeitlich wurde Icinga 2 in die Echtzeitumgebung erfolgreich integriert und läuft rund um die Uhr. Die alte Monitoring-Software „Nagios“ wurde abgeschaltet. Das Projekt wurde somit in allen Punkten erfolgreich umgesetzt.

Glossar

Action URL
Über die Weboberfläche können die Administrationsseiten von Switchen etc. erreicht werden.

Cron
Befehle aus der Crontab-Datei werden nach einem bestimmten Zeitplan ausgeführt.

Crontab
Auch Cron-Tabelle genannt, ist die Konfigurationsdatei von Cron. In der Cron-Tabelle werden alle auszuführenden Jobs eingetragen.

NRPE
Nagios Remote Plugin Executor (NRPE) ist ein Addon zum Ausführen von Nagios / Icinga Plugins auf einem Linux System.

Portable
Ohne Installation oder Anpassungen auf verschiedenen Computern lauffähig.

SAN-System
Ein Storage Area Network (SAN) ist ein Datenspeicher-Netzwerk in dem große Datenmengen gespeichert und bewegt werden. Im SAN wird der gesamte Speicher, unabhängig von Standort und Betriebssystem, zentral verwaltet und zu virtuellen Einheiten zusammengefasst.

Sourcecode
Quelltext einer Software in einer bestimmten Programmiersprache.

StatusMap
Grafischer Plan über alle Hosts im Monitoring-Netzwerk.

Tarball
Dateien die mit dem Unix Programm „Tar“ gepackt wurden.

Virtuelle Maschine
Die virtuelle Maschine (VM) bildet die Rechnerarchitektur eines real in Hardware existierenden oder hypothetischen Rechners nach.

Workaround
Eine Behelfslösung zur Vermeidung eines bekannten Fehlverhaltens. Es löst das Problem nicht, sondern umgeht dieses lediglich.

XenCenter
Hardware auf denen die virtuellen Maschinen im Betrieb sind.

Tabellenverzeichnis

Tabelle 1: Apt Befehlsübersicht....3
Tabelle 2: Icinga 2 Service Befehlsübersicht....4
Tabelle 3: Konfiguration Exim4....9

Abbildungsverzeichnis

Abbildung 1: Partitionsübersicht....2
Abbildung 2: Konfiguration statische IP....3
Abbildung 3: Hosts mit Action Button und Logo....5
Abbildung 4: Arten von Services....7
Abbildung 5: Erfolgreich zugestellte Testmail....9

Quellenverzeichnis

http://www.vdr-wiki.de/wiki/index.php/Debian_-_Befehle_zur_APT-Paket-Verwaltung
Stand: 05.10.2015

https://www.thomas-krenn.com/de/wiki/Installation_von_Icinga2_unter_Ubuntu_Server_14.04
Stand: 06.10.2015

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/toc
Stand: 06.10.2015

https://labs.consol.de/de/nagios/check_mysql_health/index.html
Stand: 07.10.2015

http://outsideit.net/check-netapp-ontap/
Stand: 07.10.2015

http://docs.icinga.org/latest/images/objects-services.png
Stand: 07.10.2015

http://www.akamaras.com/linux/linux-script-to-check-if-a-service-is-running-and-start-it-if-its-stopped
Stand: 08.10.2015

https://de.wikipedia.org/wiki/Workaround
Stand: 09.10.2015

https://de.wikipedia.org/wiki/Virtuelle_Maschine
Stand: 08.10.2015

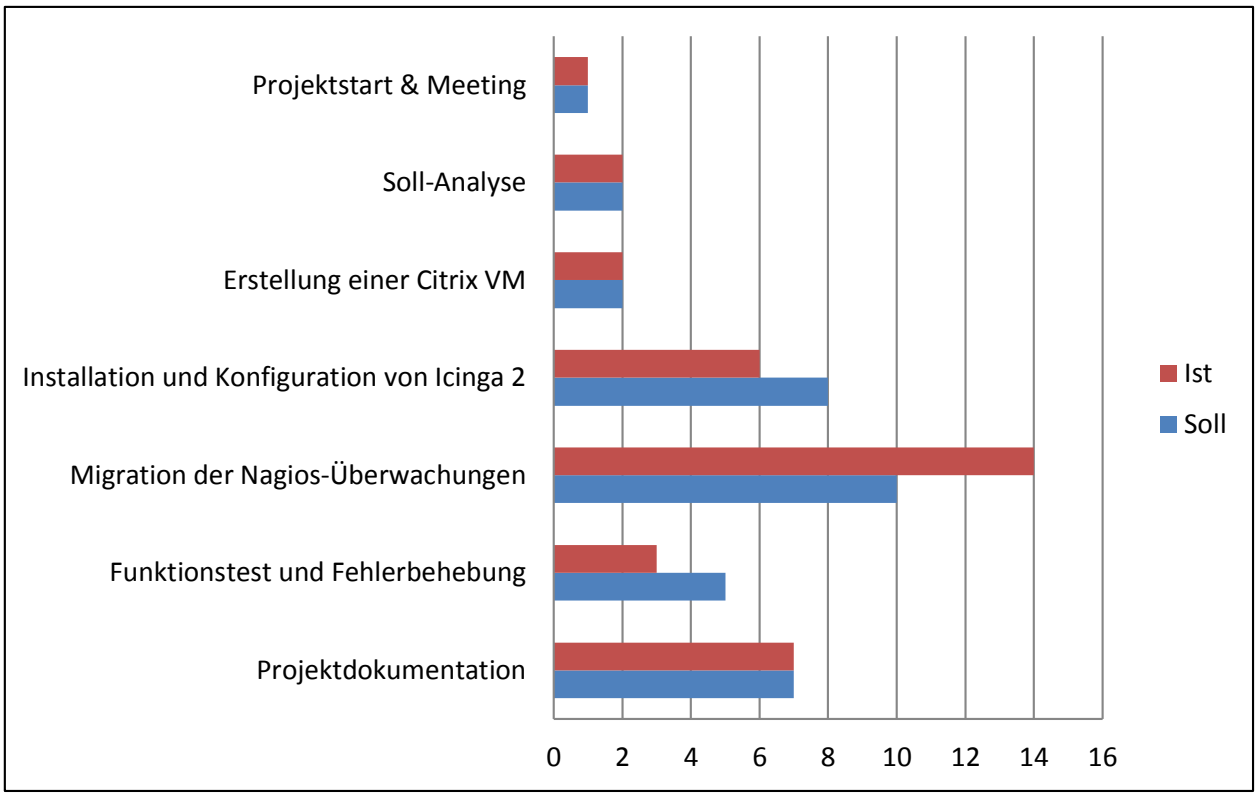
https://de.wikipedia.org/wiki/Cron
Stand: 08.10.2015

http://www.elektronik-kompendium.de/sites/net/0906071.htm
Stand: 08.10.2015

Anhang

A. Projektphasen mit Zeitaufwand in Stunden

Projektphasen	05.10	06.10	07.10	08.10	09.10	Soll	Ist
Projektstart & Meeting	1	-	-	-	-	1	1
Soll-Analyse	2	-	-	-	-	2	2
Erstellung Citrix VM	2	-				2	2
Installation und Konfiguration von Icinga 2	2	4	-	-	-	8	6
Migration der Nagios-Überwachungen	1	5	8	-	-	10	14
Funktionstest und Fehlerbehebung	-	-	-	3	-	5	3
Projektdokumentation					7	7	7
Gesamtaufwand	8	9	8	3	7	35	35



B. Verzeichnisstruktur Icinga 2 Konfigurationen

```
root@vkh-monitoring-icinga2:/etc/icinga2/conf.d# tree -C
.
├── commands
│   ├── command_core_switch_2.conf
│   ├── command_core_switch.conf
│   ├── command_etc.conf
│   ├── command_external.conf
│   ├── command_filecount_winhosts.conf
│   ├── command_localhost.conf
│   ├── command_mail.conf
│   ├── command_netapp.conf
│   ├── command_public.conf
│   ├── command_switch_xen1-3.conf
│   ├── command_switch_xen4-6.conf
│   ├── command_vs_switch.conf
│   ├── command_vs_switch_naf.conf
│   ├── command_watchguard.conf
│   └── command_win.conf
├── downtimes.conf
├── groups.conf
├── hosts
│   ├── hosts_blade.conf
│   ├── hosts_citrix_xen.conf
│   ├── hosts_etc.conf
│   ├── hosts_external.conf
│   ├── hosts_localhost.conf
│   ├── hosts_netapp_filer.conf
│   ├── hosts_netscaler.conf
│   ├── hosts_switch.conf
│   ├── hosts_watchguard.conf
│   └── hosts_windows.conf
├── notifications.conf
├── satellite.conf
├── servicegroups_switch.conf
├── services
│   ├── service_etc.conf
│   ├── service_external.conf
│   ├── service_filecount_winhosts.conf
│   ├── service_localhost.conf
│   ├── service_netapp.conf
│   ├── service_public.conf
│   ├── service_switch_core_2.conf
│   ├── service_switch_core.conf
│   ├── service_switch_naf.conf
│   ├── service_switch_xen1-3.conf
│   ├── service_switch_xen4-6.conf
│   ├── service_vs_switch.conf
│   ├── service_watchguard.conf
│   ├── service_win.conf
│   └── service_win_sql.conf
├── templates.conf
├── timeperiods.conf
└── users.conf

3 directories, 48 files
```

C. user.conf

Code: /etc/icinga2/conf.d/users.conf Icinga 2

```
object User "icingaadmin" {
    import "generic-user"
    display_name = "Icinga 2 Admin"
    groups = [ "icingaadmins" ]
    enable_notifications = true
    email = "user@webadresse.de"
 }

object UserGroup "icingaadmins" {
    display_name = "Icinga Administrators"
 }
```

Code: Nagios

```
define contact{
      contact_name      nagiosadmin
      use               generic-contact
      alias             Nagios Admin
      email             user@webadresse.de
  }

define contactgroup{
      contactgroup_name     admins
      alias                 Nagios Administrators
      members               nagiosadmin
      }
```

D. Host Konfiguration

Code: /etc/icinga2/conf.d/hosts/hosts_switch.conf Icinga 2

```
object Host "Core_Switch" {
    import "generic-switch"
    address = "192.168.95.***"
    action_url = "http://$HOSTADDRESS$"
    notes = "Core_Switch_1 Rechenzentrum"
}
```

Code: Nagios

```
define host{
    use             generic-switch
    host_name       Core_Switch
    alias           ProCurve Switch 8212zl
    address         192.168.95.***
    action_url      http://$HOSTADDRESS$
    notes           Core_Switch_1 Rechenzentrum
  }
```

E. groups.conf

Code: /etc/icinga2/conf.d/groups.conf **Icinga 2**

```
Einzelner Host:
object HostGroup "dhcp-server" {
    display_name = "DHCP Server"
    assign where host.name == “vkh-dc2"
 }

Mehrere Hosts:
object HostGroup "mssql-server" {
   display_name = "MSSQL DB"
   assign where host.name in [ "vkh-mssql", "vkh-mssql2" ]
 }
```

Code: **Nagios**

```
Einzelner Host:
define hostgroup{
   hostgroup_name      dhcp-server
   alias               DHCP-Server
   members             vkh-dc2
 }

Mehrere Hosts:
define hostgroup{
   hostgroup_name      mssql-server
   alias               MSSQL DB
   members             vkh-mssql,vkh-mssql2
 }
```

F. servicegroup_switch.conf

Code: /etc/icinga2/conf.d/servicegroups_switch.conf **Icinga 2**

```
object ServiceGroup "Core - VS 1-1" {
   display_name = "Netzwerkverbindung Core - VS 1-1"
   assign where service.name in [ "Check VS 1-1 Port 48", "Check Core_Switch Port A1 - VS1-1" ]
  }
```

Code: **Nagios**

```
define servicegroup{
   servicegroup_name    Core - VS 1-1
   alias                Netzwerkverbindung Core - VS 1-1
   members              VS 1-1,Check VS 1-1 Port 48,Core_Switch,Check Core_Switch Port A1 - VS1-1
 }
```

G. Service Konfiguration

Code: /etc/conf.d/services/service_public.conf **Icinga 2**

```
apply Service "DNS-Internet" {
    import "local-service"
    check_command = "check_dnsinet"
    assign where "dns-server" in host.groups
 }
```

Code: **Nagios**

```
define service{
    use                     local-service
    hostgroup_name          dns-server
    service_description     DNS-Internet
    check_command           check_dnsinet
 }
```

H. Command Konfiguration

Code: /etc/conf.d/commands/command_public.conf **Icinga 2**

```
object CheckCommand "check_dnsinet" {
import "plugin-check-command"
import "ipv4-or-ipv6"
command = [ PluginDir + "/check_dns" ]

arguments = {
            "-H" = "www.google.de"
            "-s" = "$check_address$"
            "-w" = 5s
            "-c" = 10s
            "-t" = 20s
            }
     }
```

Code: **Nagios**

```
define command{
            command_name check_dnsinet
            command_line    $USER1$/check_dns -H www.google.de -s $HOSTADDRESS$ -w 5 -c 10 -t 20
            }
```

I. Check MySQL Health Konfiguration

Code: **Icinga 2**

Code: /etc/conf.d/services/service_public.conf

```
apply Service "MySQL DB Uptime" {
    import "generic-service"
    check_command = "check_mysql_uptime"
    assign where "mysql-server" in host.groups
 }
```

Code: /etc/conf.d/commands/command_public.conf

```
object CheckCommand "check_mysql_uptime" {
    import "plugin-check-command"
    command = [ CustomPluginDir + "/check_mysql_health" ]
    arguments = {
            "--hostname" = "$address$"
            "-username" = "<name>"
            "-password" = "<passwort>"
            "-mode" = "uptime"
 }
}
```

Code: **Nagios**

```
define service{
    use                     generic-service
    hostgroup_name          mysql-server
    service_description     MySQL DB Uptime
    check_command           check_mysql_uptime
 }

define command{
    command_name      check_mysql_uptime
    command_line      $USER1$/check_mysql_health --hostname $HOSTADDRESS$ --username <name>
                      --password <passwort> --mode=uptime
 }
```

J. Check NetApp Konfiguration

Code: **Icinga 2**

Code: /etc/conf.d/services/service_netapp.conf

```
apply Service "Check Volume Health" {
        import "instant-service"
        assign where host.name == "Vinzenzfiler"
        check_command = "check_naf_health_volume"
        }

Code: /etc/conf.d/commands/command_netapp.conf
object CheckCommand "check_naf_health_volume" {
   import "plugin-check-command"
   command = [ CustomPluginDir + "/check_netapp_ontap.pl" ]
   arguments = {
        "-H" = "$address$"
        "-u" = "<name>"
        "-p" = "<passwort>"
        "-o" = "volume_health"
        "-w" = "90%i"
        "-c" = "95%i"
 }
}
```

Code: **Nagios**

```
define service{
   use                     instant-service
   host_name               Vinzenzfiler
   service_description     Check Volume Health
   check_command           check_naf_health_volume
        }

define command{
   command_name    check_naf_health_volume
   command_line    $USER1$/check_netapp_ontap.pl -H $HOSTADDRESS$ -user <name> -p <passwort>
                   -o volume_health -w 90%i -c 95%i
 }
```

K. Check NT Konfiguration

Code: Icinga 2

Code: /etc/conf.d/services/service_win.conf

```
apply Service "Memory Usage" {
    import "generic-service"
    check_command = "check_nt_win"
    assign where "windows-servers" in host.groups
    vars.nt_variable_arguments = "MEMUSE"
    vars.nt_warning_arguments = "90"
    vars.nt_critical_arguments = "95"
 }
```

Code: /etc/conf.d/commands/command_win.conf

```
object CheckCommand "check_nt_win" {
    import "plugin-check-command"
    command = [ PluginDir + "/check_nt" ]
    arguments = {
            "-H" = "$address$"
            "-p" = "<port>"
            "-s" = "<passwort>"
            "-d" = "$nt_showall_arguments$"
            "-w" = "$nt_warning_arguments$"
            "-c" = "$nt_critical_arguments$"
            "-v" = "$nt_variable_arguments$"
            "-l" = "$nt_parameter_arguments$"
 }
 }
```

Code: Nagios

```
define service{
    use                     generic-service
    hostgroup_name          windows-servers
    service_description     Memory Usage
    check_command           check_nt!MEMUSE!-w 90 -c 95
 }

define command{
    command_name      check_nt
    command_line      $USER1$/check_nt -H $HOSTADDRESS$ -p <port> -s <passwort> -v $ARG1$ $ARG2$
 }
```

L. NRPE Konfiguration

Code: Icinga 2

Code: /etc/conf.d/services/service_filecount_winhosts.conf

```
apply Service "Dateianzahl Befunde" {
    import "file-service"
    assign where host.name == "Hostname"
    check_command = "check_filecount"
    vars.nrpe_arguments = "\"C:\\Ordner\\Unterordner\\Unterordner\"!80!120!2!4"
 }
```

Code: /etc/conf.d/commands/command_filecount_winhosts.conf

```
object CheckCommand "check_filecount" {
    import "plugin-check-command"
    command = [ PluginDir + "/check_nrpe" ]
    arguments = {
            "-H" = "$address$"
            "-p" = "5666"
            "-c" = "check_fcount"
            "-a" = "$nrpe_arguments$"
            "-n" = ""
 }
}
```

Code: Nagios

```
define service{
    use                     file-service
    host_name               Hostname
    service_description     Dateianzahl Befunde
    check_command           check_filecount!"C:\\Ordner\\Unterordner\\Unterordner "!80!120!2!4
 }

define command{
    command_name    check_filecount
    command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -n -p <port> -c check_fcount -a $ARG1$ $ARG2$
                    $ARG3$ $ARG4$ $ARG5$
 }
```

M. Workaround Script für “Segmentation Fault” Fehler

Code:

```
#####################################################
##                    ICINGA RESTART SCRIPT                    ##
##   WORKAROUND FOR "SEGMENTATION FAULT" BUG    ##
#####################################################

#!/bin/bash
service icinga2 stop >>/dev/NULL
service=icinga2.err
while :
do
 service icinga2 start >>/dev/NULL
  sleep 1
  if (( $(ps -ef | grep -v grep | grep $service | wc -l) > 0 ))
  then
     echo -e "\e[32mIcinga 2 restarted!\e[39m"
     break           #Schleife abbrechen
  fi
done
```

N. Restart Script falls Icinga 2 abgestürzt

Code:

```
#################################################
##          Restart Icinga 2 If Stopped or Crashed          ##
##          check status every minute (crontab -e)          ##
#################################################

#!/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
if [ `/bin/ps ax | /bin/grep icinga2.err | /bin/grep -v /bin/grep | /usr/bin/wc -l` -eq 1 ]
then
exit
else
while :
do
 service icinga2 start >>/dev/NULL
  sleep 1
if [ `/bin/ps ax | /bin/grep icinga2.err | /bin/grep -v /bin/grep | /usr/bin/wc -l` -eq 1 ]
  then
  date >> /var/log/icinga2/restarted.after.crash.log
  break
  fi
done
fi
```