



HÖHERE TECHNISCHE BUNDESLEHRANSTALT Wien 3, Rennweg
IT & Mechatronik

HTL Rennweg :: Rennweg 89b
A-1030 Wien :: Tel +43 1 24215-10 :: Fax DW 18

Laborprotokoll

Kubernetes, VEEAM und PRTG

ausgeführt an der
Höheren Abteilung für Informationstechnologie
der Höheren Technischen Lehranstalt Wien 3 Rennweg

im Schuljahr 2019/2020

durch

Leon Kirschner
Michael Kudler

im Auftrag von

DI Bernhard Nickel

Wien, 19. April 2020

Inhaltsverzeichnis

1	Containerorchestrierung mit Kubernetes	1
1.1	Aufgabenstellung	1
1.1.1	Kubernetes	1
1.2	Installation	3
1.2.1	Validierung bevor es losgeht	3
1.3	Installation	4
1.3.1	Docker	4
1.3.2	Kubernetes	5
1.4	Erstellung eines Clusters	6
1.4.1	Einstellungen	6
1.4.2	Kubeadm	6
1.5	Troubleshooting	7
2	Netzwerküberwachung mit PRTG	9
2.1	Installation & Erste Schritte	9
2.2	LAB: Überwachung von Linux und Windows Server	10
2.2.1	Topologie	10
2.2.2	Überwachung Linux Server	10
2.2.3	Überwachung Windows Server	15
2.3	Lessons Learned	17

1 Containerorchestrierung mit Kubernetes

1.1 Aufgabenstellung

Die Idee des Projektes ist es einen Kubernetes Cluster auf 3 VMS zu deployen und die Möglichkeiten die dadurch entstehen zu erkunden.

Ein optionales Ziel ist es, diese Infrastruktur auf eine Web-Applikation mit Anbindung einer Datenbank anzuwenden und Aktivitäten zu überprüfen.

1.1.1 Kubernetes

Kubernetes ist ein Open-Source Orchestrierungs Tool, das dazu dient Container automatisiert Bereitzustellen und zu verwalten.

Der Name Kubernets kommt aus dem griechischen und steht für Steuer-mann.

Wir verwenden in unserem Projekt Docker als Container Runtime.

1.1.1.1 Aufbau und Architektur

Der Vorteil bei Kubernetes liegt darin, dass es *Pods* orchestriert. Sie stellen die kleinstmögliche steuerbare Einheit im Kubernetes Universum dar. Sie laufen auf *Nodes* - also VMs oder physischen Maschinen). Ein Pod kann einen oder mehrere Container beinhalten.

Die Architektur ist auf dem Master-Slave System aufgebaut.

Der Master ist die *Control Plane*, auf ihr wird Inventur über alle Objekte in einem Cluster geführt. Der Master steuert außerdem alle Slaves (Minions). Wir haben in

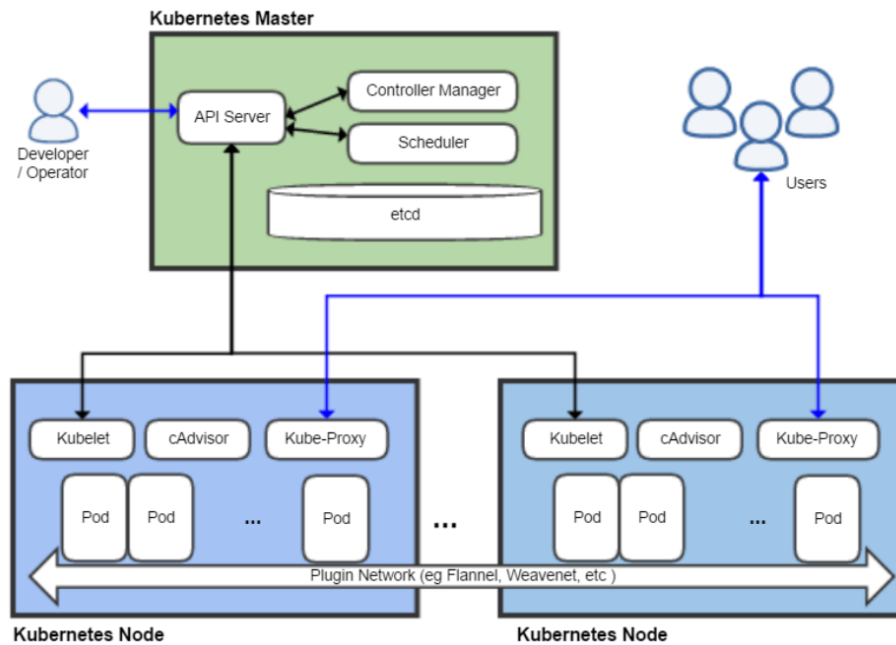


Abbildung 1.1: Kubernetes Architektur

unserem Fall nur einen Master-Node. Es können aber auch - zwecks Redundanz - mehrere Master-Nodes in einem Kubernetes Cluster konfiguriert werden.

Auf jedem Minion muss zusätzlich auch die zu verwendende Container-Runtime installiert werden.

1.2 Installation

1.2.1 Validierung bevor es losgeht

Bei jedem Node (VM oder physische Maschine), die im Cluster verwendet werden soll, müssen sich folgende Eigenschaften unterscheiden:

Da wir für dieses Beispiel Debian 10 verwenden, müssen wir zusätzlich noch iptables in den legacy-mode schalten.

```
update-alternatives --set iptables /usr/sbin/iptables-legacy  
update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy  
update-alternatives --set arptables /usr/sbin/arptables-legacy  
update-alternatives --set ebtables /usr/sbin/ebtables-legacy
```

1.3 Installation

1.3.1 Docker

Als Container Runtime benutzen wir für dieses Beispiel Docker.

Zuerst müssen wir alle Dependencies von Docker installieren.

```
apt update
apt -y install apt-transport-https ca-certificates curl gnupg2 software-properties-common
```

Anschließend fügen für den offiziellen GPG-Key von Docker zu unserem Package-Manager hinzu. Mit diesem Schlüssel sind die Pakete signiert.

```
curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
```

Nun können wir die offiziellen Docker-Repositories hinzufügen.

```
add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/debian \
    $(lsb_release -cs) \
    stable"
```

Die Installation selbst ist nun ziemlich einfach.

```
apt-get update && apt-get install -y \
    containerd.io=1.2.10-3 \
    docker-ce=5:19.03.4~3-0~debian-$(lsb_release -cs) \
    docker-ce-cli=5:19.03.4~3-0~debian-$(lsb_release -cs)
```

Zu guter Letzt passen wir noch die Einstellungen für Docker an.

```
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
```

```
mkdir -p /etc/systemd/system/docker.service.d
```

Einen kleinen Restart brauchen wir noch.


```
systemctl daemon-reload
systemctl restart docker
```

Wie jeder gute IT-Admin werden verifizieren wir natürlich noch am Ende die gelungene Installation.

```
docker version
```

Falls hier vernünftige Informationen angezeigt werden und keine Fehlermeldung ist die Installation gelungen.

1.3.2 Kubernetes

Als nächsten Schritt installieren wir Kubernetes. Oder besser gesagt die 3 Services, aus denen unsere Kubernetes Installation bestehen wird.

- kubeadm
- kubelet
- kubectl

Dafür benutzen wir einen ähnlichen Ablauf, wie bei Docker. Der Einfachheit halber ist nun nicht jeder Schritt kommentiert, sondern ein fertiges Skript zu sehen.

```
apt-get update && sudo apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl
```

Auch hier folgt natürlich wieder die Verifikation.

```
kubeadm version
kubelet --version
kubectl version
```

1.4 Erstellung eines Clusters

1.4.1 Einstellungen

Ein kleiner Schritt hält uns noch vom Cluster ab: Wir müssen swap abschalten.

```
swapoff -a
cp /etc/fstab /etc/fstab.orig
cat /etc/fstab.orig | grep -v 'swap' > /etc/fstab
```

Jetzt noch ein Reboot.

```
reboot 0
```

1.4.2 Kubeadm

Bevor man den Cluster installiert muss man sich für ein pod network add-on entscheiden. Die Auswahl ist da groß, weswegen wir uns schlichtweg für das beliebteste Add-On entschieden haben: Flannel

Auf unserem Master führen wir nun folgenden Befehl aus:

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

Mit den anderen Nodes können wir jetzt joinen. Den Befehl dafür finden wir im Output von kubeadm init am Master.

```
kubeadm join 10.0.0.88:6443 --token <token> \
  --discovery-token-ca-cert-hash sha256:<hash>
```

1.5 Troubleshooting

Bei der Verifizierung am Master mit `!kubectl get nodes!` kann es zu folgendem Fehler kommen:

The connection to the server localhost:8080 was refused – did you specify

Das lässt sich mit den folgenden Befehlen beheben. (Credits an den GitHub post von user csarora)

```
cp /etc/kubernetes/admin.conf $HOME/  
chown $(id -u):$(id -g) $HOME/admin.conf  
export KUBECONFIG=$HOME/admin.conf
```


2 Netzwerküberwachung mit PRTG

Die Software PRTG¹ der deutschen Firma Paessler GmbH bietet eine Vielzahl an Tools um nahezu jede Komponente eines Netzwerks zu überwachen. Einige wichtige Features bilden:

- Unterstützung aller gängige Agents: SNMP, WMI, SSH bzw. Agentless: SQL, Ping, REST APIs und viel mehr.
- Maps und Dashboard: Intuitive Erstellung von Dashboards zur echtzeit Überwachung mit Live-Statusinformationen
- Flexibler Alarm: PRTG alarmiert automatisch den Administrator, sobald Problem oder Anomalien entdeckt werden.

Paessler bietet mehrere Lizenzen an, die sich hauptsächlich durch die Anzahl der Sensoren unterscheiden. Angefangen bei 1.300€ mit 500 unterstützen Sensoren für kleine Unternehmen bis hin zu 12.500€ mit unbegrenzten Sensoren für größere Unternehmen, ist für jeden was dabei. Zu Testzwecken stellt Paessler jedem eine 30 Tage Testversion zur Verfügung.

2.1 Installation & Erste Schritte

Die Installation des Tools ist intuitiv und dauert nur wenige Minuten. Nachdem PRTG heruntergeladen ist, wird die Software installiert und automatisch gestartet. Es handelt sich hierbei um eine Webapplikation, die über die IP-Adresse des Computers erreichbar ist und über jeden Webbrowser erreichbar ist, der den Server erreichen kann.

Bevor man jedoch anfängt mit PRTG zu arbeiten, sollten einem einige Begriffe geläufig sein. Einer dieser Begriffe ist der Sensor. Sensoren sind das um und auf der PRTG Network Monitor Software. Ein Sensor ist ein Messpunkt, der einen *Aspekt* auf einem Gerät überwacht. Beispiele dafür wären CPU-Auslastung eines Servers oder die Portauslastung eines Interfaces auf einem Switch. So ermöglichen eine Vielzahl von Sensoren die Rundumüberwachung verschiedenster Geräte. Ein Sensor schleust Daten

¹ Früher Paessler Router Traffic Grapher

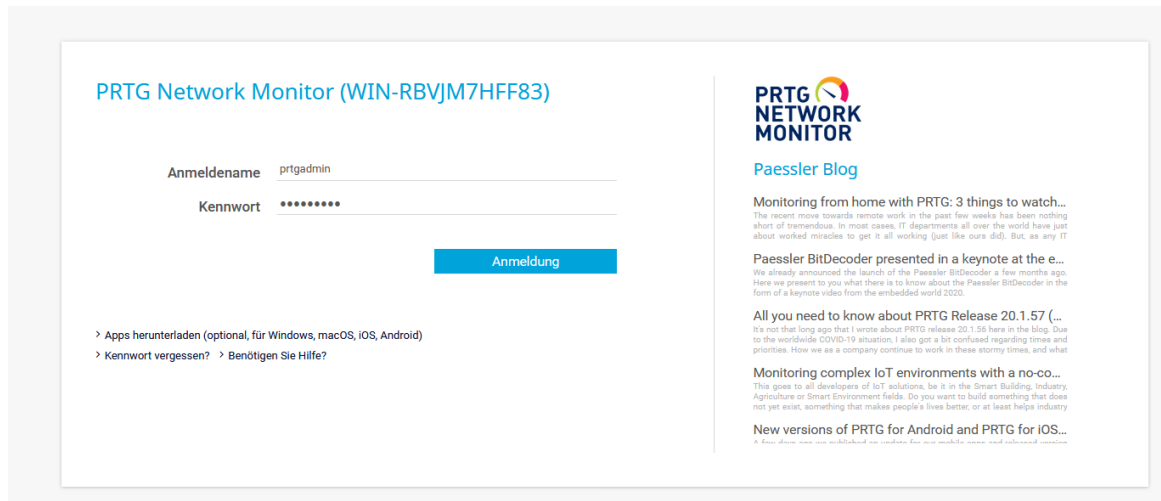


Abbildung 2.1: PRTG Anmeldemaske

in einen *Kanal*. So hat der Ping Sensor die Kanäle Latenz(ms), Paketverlust(%), etc.
.

2.2 LAB: Überwachung von Linux und Windows Server

Das folgende Kapitel beschäftigt sich mit der Überwachung einer einfachen Serverinfrastruktur.

2.2.1 Topologie

Die Infrastruktur besteht aus drei Servern:

- Windows Server (PRTG Host): Der Monitoring Server
- WINSRV: Windows Server
- LINUXSRV: Linux Server mit einem Apache Server

2.2.2 Überwachung Linux Server

Im ersten Schritt muss das Gerät registriert werden. Dazu im Reiter *Geräte* auf *Gerät hinzufügen* klicken und eine Gruppe nach Belieben auswählen. Wir haben die Gruppe *Linux/ macOS / Unix* gewählt. Danach erscheint es in der Geräteliste:

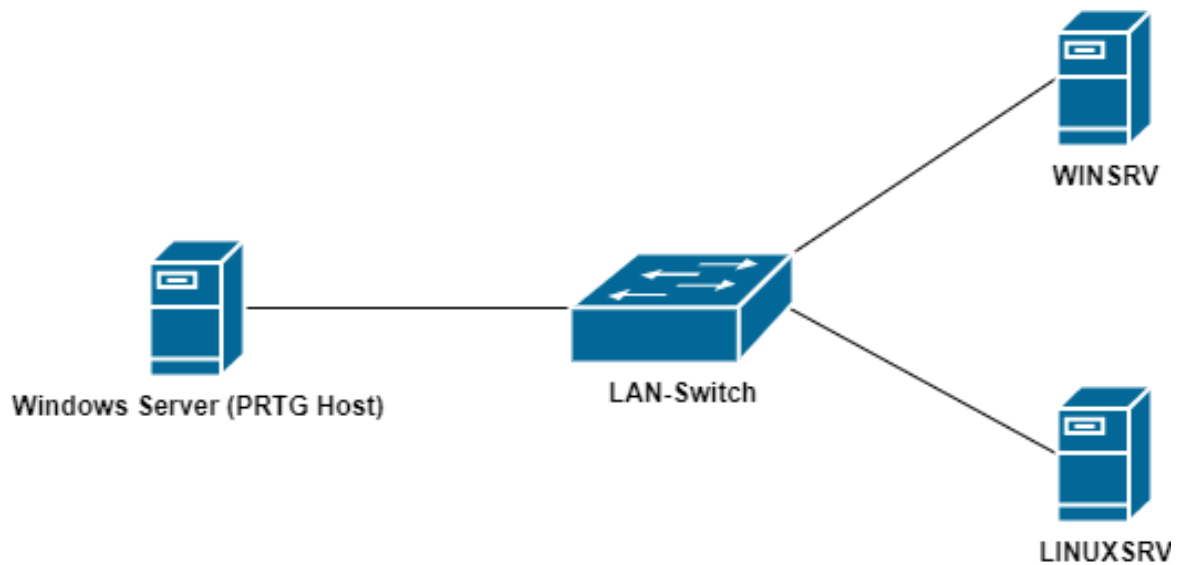


Abbildung 2.2: PRTG LAB Architektur

Geräte

Probe Gruppe Gerät ▾	Gerät ⓘ	Ortsangabe ⓘ	⚠ ⓘ
Local Probe (Getrennt)	🔴 Gerät der Probe		!! 1
Local Probe (Getrennt) » Netzwerk-Infrastruktur	🔴 Internet		
Local Probe (Getrennt) » Linux / macOS / Unix	🔴 Linux Server		

<< < 1 bis 3 von 3 > >>

Abbildung 2.3: PRTG Gerät: Linux Server

Die Überwachung eines Linux Servers funktioniert meist Agentless. Es muss also auf dem Zielgerät nichts installiert werden und Informationsabfrage findet über Protokolle wie SSH, HTTP oder SNMP statt. Dementsprechend ist die Konfiguration von Sensoren für Linux System ziemlich straight-forward. Nichtsdestotrotz mussten einige Vorbereitungen getroffen werden:

1. Open-SSH Server installieren²
2. Apache Server installieren³

² Meist vorinstalliert, bei uns (Linux Mint) jedoch nicht

³ apache2

2.2.2.1 Erster Sensor: Ping

Auf der Geräteseite des Linux Servers kann in der Sensorenbox via Klick auf das “+” ein Sensor hinzugefügt werden.

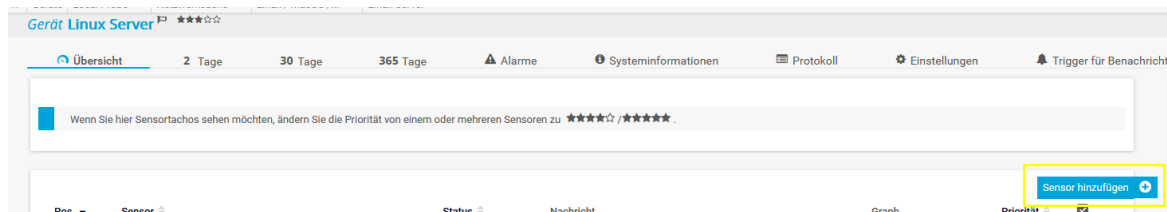


Abbildung 2.4: PRTG Sensor hinzufügen

Im nächsten Schritt ist der gewünschte Sensortyp auszuwählen, in unserem Fall Ping. Danach sind einige Parameter einzugeben, wir lassen sie auf den Defaulteinstellungen. Nach einer kurzen Wartezeit, kann man sich mit Klick auf den Ping Sensor eine Statistik über die abgesetzten Pings anschauen.

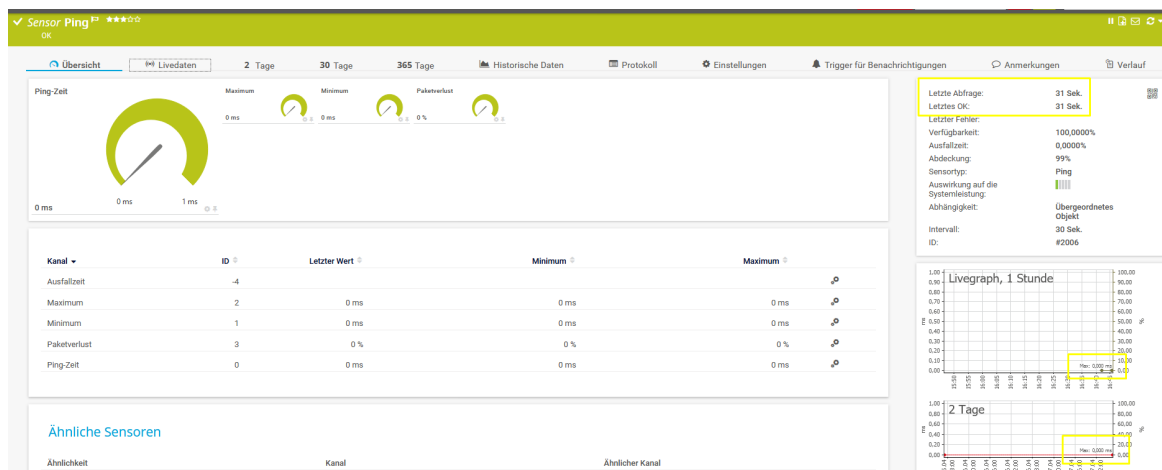


Abbildung 2.5: PRTG Sensor Statistiken

2.2.2.2 Apache Server überwachen

Auf dem Linux Server wurde kurzerhand ein Apache Server installiert. Über den HTTP Sensor lässt sich auch dieser ziemlich schnell und einfach überwachen. Dazu dem selben Prozedere wie zuvor folgen.

Diesmal wollen wir allerdings eine Benachrichtigung erhalten, sobald der Apache Server einen Fehler aufweist. Hierfür muss auf der Geräte Seite unter dem Tab *Trigger für Benachrichtigungen* ein neuer Trigger hinzugefügt werden. In unserem Fall soll der Trigger nach 20 Sekunden durchgehendem fehlerhaften Verhalten ein Ticket erstellt werden.

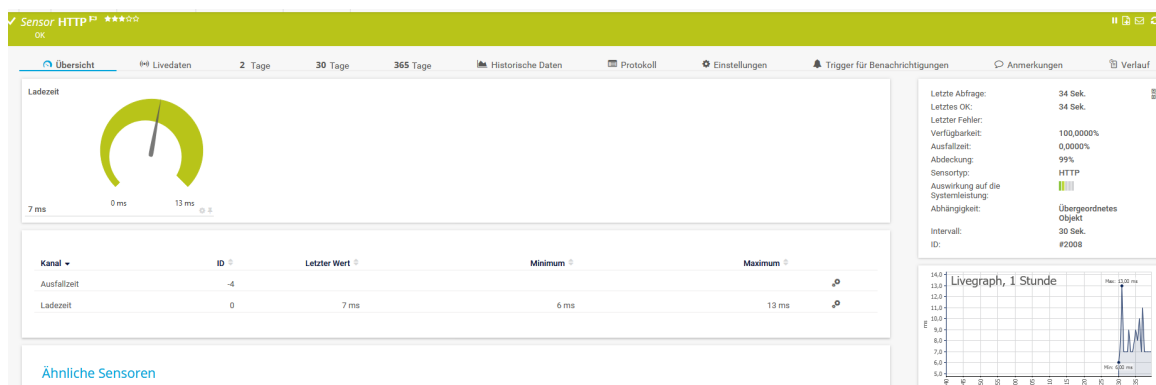


Abbildung 2.6: PRTG Überwachung des Apache Servers



Abbildung 2.7: PRTG Triggeraktion

Um zu testen wurde der Apache Server kurzerhand gestoppt und wie zu erwarten wurde ein wenig später ein Ticket erstellt.



Abbildung 2.8: PRTG Ticket

Mit einem Klick auf das Ticket können Details angezeigt werden (Abbildung 2.9).

2.2.2.3 Erweiterte Sensoren

Um einen - meines Erachtens - recht außergewöhnlichen Sensor zu testen/implementieren. Wurde auf dem Linux Server mittels Flask und Python eine REST-API programmiert um den PRTG REST Sensor zu testen.

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route("/")
def helloWorld():
    return jsonify({"hallo" : "welt"})
```

Benachrichtigung Ticket #4 ★★☆☆☆
Linux Server HTTP (HTTP) Fehler (Connection refused (Socketfehler # 10061))

Status: **offen** Zugewiesen an: **PRTG Administratoren** Verknüpftes Objekt: **HTTP** Typ: **Benachrichtigung** ID: **#4**

Letztes Update
Geöffnet von **PRTG System Administrator** + Zugewiesen an **PRTG Administratoren** 17.04.2020 18:40:48

Sensor: HTTP (HTTP)
Status: Fehler

Datum/Zeit: 17.04.2020 18:40:48 (W. Europe Standard Time)
Letztes Ergebnis:
Letzte Nachricht: Connection refused (Socketfehler # 10061)

Probe: Local Probe
Gruppe: Linux / macOS / Unix
Gerät: Linux Server (192.168.10.3)

Letzte Abfrage: 17.04.2020 18:40:48 [liegt 0 Sek. zurück]
Letztes OK: 17.04.2020 18:39:11 [liegt 97 Sek. zurück]
Letzter Fehler: 17.04.2020 18:40:48 [liegt 0 Sek. zurück]
Verfügbarkeit: 95,2978% [10 Min. 8 Sek.]
Ausfallzeit: 4,7022% [30 Sek.]
Akkumuliert seit: 17.04.2020 18:30:10
Ortsangabe:

Abbildung 2.9: PRTG Ticketdetails

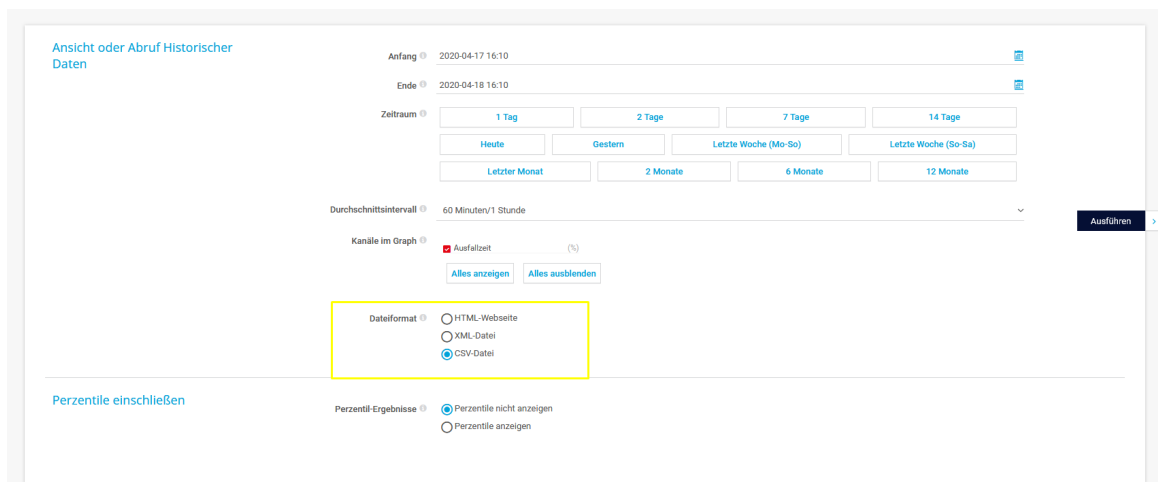
```
if __name__ == '__main__':  
    app.run(host='192.168.10.3', port=8080)
```

die mit REST API⁴ Sensor überprüft werden kann. Untr dem Reiter *Historische Daten* kann man die gesammelten Daten als CSV, kann man alle gesammelten Daten im CSV, XML oder HTML exportieren um daraus z.B. Grafiken zu generieren. Zusätzlich können Paremter wie der Zeitraum, das Durchschnittsinterval und die gesammelten Kanäle angepasst werden.

Mittels Excel können die Daten dann aufbereitet und weiterverarbeitet werden, sofern man das möchte. Wir haben zur Veranschaulichung ein Diagramm (Abbildung 2.11) über die Performance der ReST-API erstellt.

Mit PRTG ist es auch möglich über eine SSH-Session, Informationen über einen Server auszulesen wie z.B. die verfügbare Speicherauslastung.

⁴ noch in der BETA Version



Ansicht oder Abruf Historischer Daten

Anfang 2020-04-17 16:10
Ende 2020-04-18 16:10

Zeitraum: 1 Tag, 2 Tage, 7 Tage, 14 Tage
Heute, Gestern, Letzte Woche (Mo-Sa), Letzte Woche (So-Sa)
Letzter Monat, 2 Monate, 6 Monate, 12 Monate

Durchschnittsintervall: 60 Minuten/1 Stunde

Kanäle im Graph: ☒ Ausfallzeit (%)
Alles anzeigen, Alles ausblenden

Dateiformat: ☐ HTML-Webseite, ☐ XML-Datei, ☒ CSV-Datei

Perzentile einschließen

Perzentil-Ergebnisse: ☒ Perzentile nicht anzeigen, ☐ Perzentile anzeigen

Ausführen

Abbildung 2.10: PRTG Export von Daten

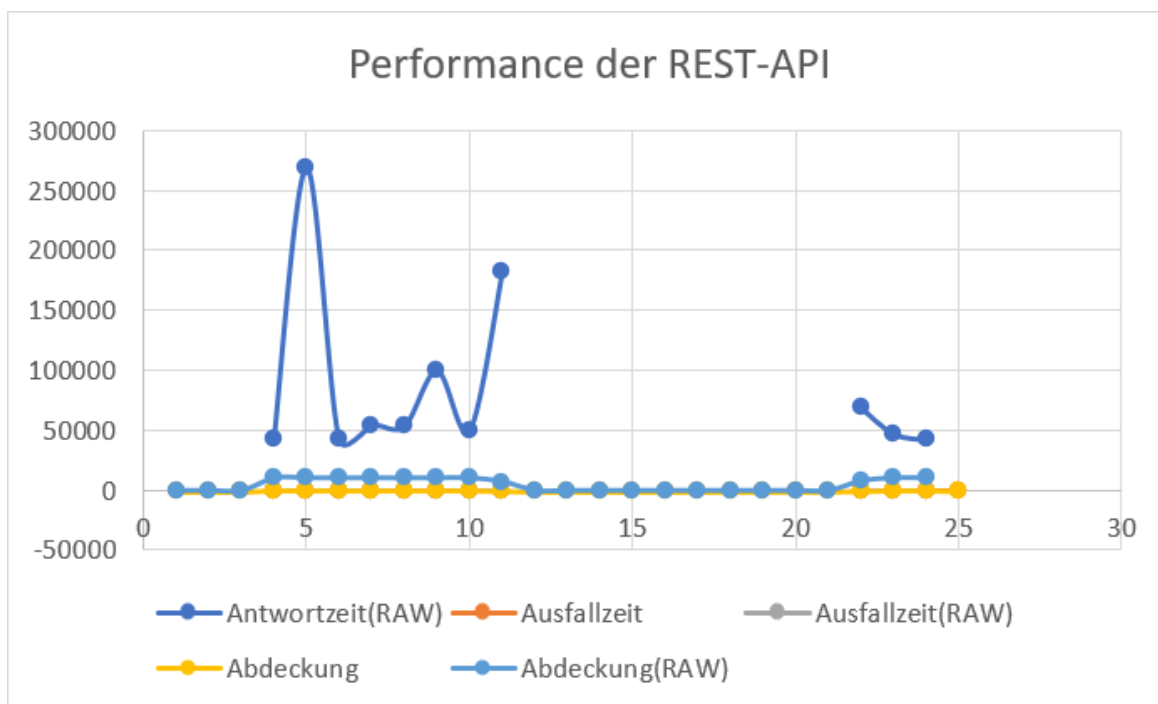


Abbildung 2.11: PRTG REST-Performance Diagramm

2.2.3 Überwachung Windows Server

Die Überwachung eines Windows-Servers geht ein wenig anders von statten. Hier findet der Informationsaustausch über WMI⁵, Performance Counters oder SNMP statt. Die Vorteile bei SNMP liegen darin, dass es eine deutlich geringere Last verursacht, als seine Microsoftproprietären Alternativen. Damit eht jedoch einher, dass WMI und Performance Counters eine größere Menge an Daten anbieten, was bei einer rundum

⁵ Windows Management Instrumentation

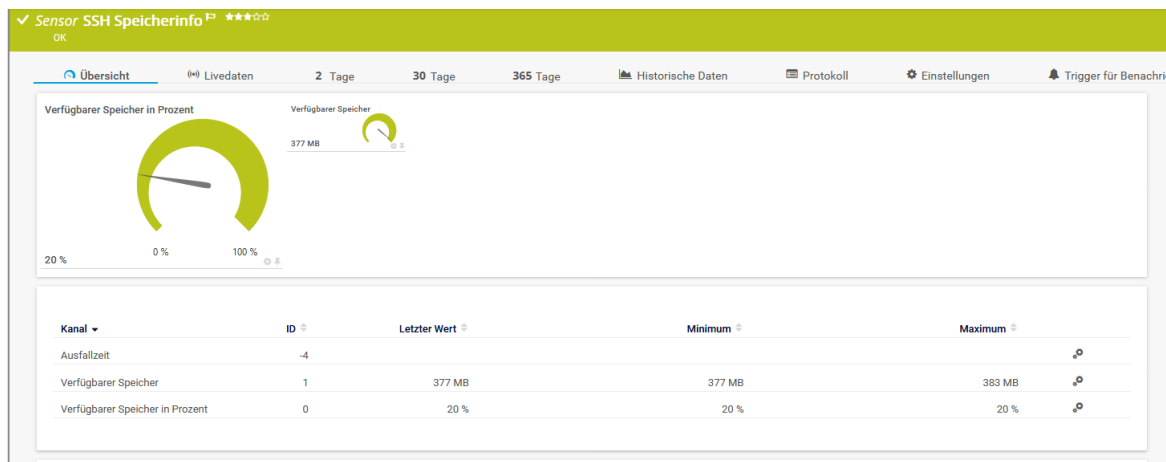


Abbildung 2.12: PRTG SSH-Speicherauslastung

Überwachung das um und auf ist.

In unserem Fall werden wir Log-Dateien über die Windows Management Instrumentations auswerten und einen modifizierten Sensor erstellen, der bei dem Überschreiten eines Schwellenwertes ein Ticket erstellt. Genauer der Ordner Desktop bei dem ab 4 Files ein neues Ticket erstellt wird.

Im ersten Schritt muss der Sensor hinzugefügt werden. Dazu dem vorherigen Prozedere folgen. Danach im Reiter *Trigger für Benachrichtigungen* ein neuer *Schwellenwerttrigger* hinzugefügt werden.

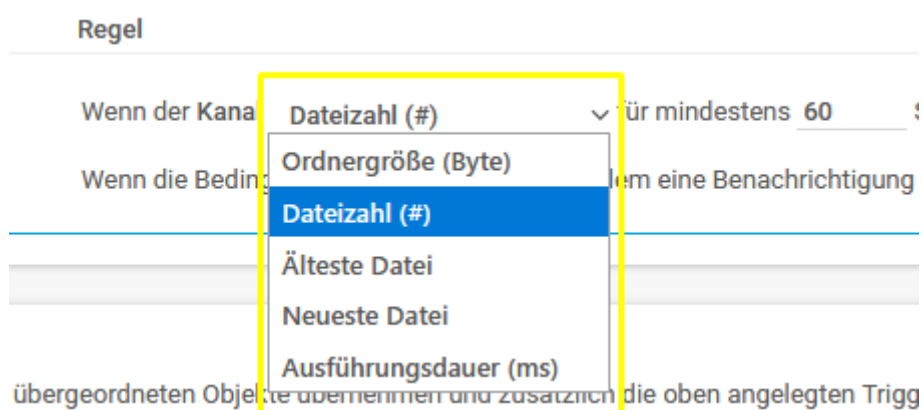


Abbildung 2.13: PRTG Dateikanäle

Wenn man nun einige Dateien erstellt und ein wenig wartet erscheint unter dem *Tickets* Reiter ein neues Ticket:

19.04.2020 20:00:20	Keine	Vorlage für Benachrichtigungen	⚠ E-Mail und Push-Benachrichtigung an Administrator	Benachrichtigungsinf...	Fehler beim Versenden von "Push-Benachrichtigung": Keir
19.04.2020 20:00:20	Keine	Gruppe	📁 Hauptgruppe	Benachrichtigungsinf...	Zustands-Trigger verschickt E-Mail/ Push-Benachrichtigung
19.04.2020 20:00:20	🐧 Linux Server	SSH Speicherinfo	➕ SSH Speicherinfo	Benachrichtigungsinf...	Zustands-Trigger aktiviert (Sensor/Quelle/ID: 2011/0/1)
19.04.2020 19:59:57	🐧 Linux Server	HTTP	➕ HTTP	Fehler	Connection refused (Socketfehler # 10061)
19.04.2020 19:59:29	🐧 Windows Server	Ordner	➕ Ordner	Benachrichtigung	Content changed
19.04.2020 19:59:00	Keine	Gruppe	📁 Hauptgruppe	Benachrichtigungsinf...	Status beim Versenden von E-Mail: "leon.kirschner@htl.re
19.04.2020 19:59:00	🐧 Linux Server	HTTP	➕ HTTP	Unbekannt	Dieser Sensor hat schon seit 93 Sekunden keine Daten m
19.04.2020 19:58:17	Keine	Gruppe	📁 Hauptgruppe	Benachrichtigungsinf...	Status beim Versenden von E-Mail: "leon.kirschner@htl.re
19.04.2020 19:58:00	Keine	Vorlage für Benachrichtigungen	🔖 Ticket-Benachrichtigung	Benachrichtigungsinf...	Gesendet Ticket: OK (Sensor/Source/ID: 2008/2005/1)
19.04.2020 19:58:00	🐧 Linux / macOS / Unix	Gerät	🐧 Linux Server	Benachrichtigungsinf...	Zustands-Trigger verschickt Ticket (Sensor/Quelle/ID: 200

Abbildung 2.14: PRTG Schwellenwertfolder

2.3 Lessons Learned

Die Arbeit und das Erkunden mit PRTG hat uns viel Spaß gemacht. Da unsere VMs in der Schule zwei Mal gelöscht wurden, ging jedoch leider viel Zeit für die Windows- bzw. Linux Installation drauf, die wir gerne in PRTG investiert hätten. Nichtsdestotrotz ist unser Fazit, dass PRTG ein sehr mächtiges - wenn auch teures - Tool zur Überwachung eines Netzwerkes ist. Der große Unterschied zu seinen (Open Source) Konkurrenten wie Nagios, wirbt es damit, keine Plugins anzubieten, weil es standardmäßig alles kann. Auch wenn wir nicht alles im Protokoll vermerken konnten, sind wir der Meinung, dass PRTG seinem Ruf gänzlich gerecht wird. Durch das intuitive Web-UI ist es auch möglich, relativ schnell und ohne viel Vorwissen Sensoren hinzuzufügen und ein IT-System überwachen kann.