

Implementierung eines Monitoring-Systems mit Prometheus und Grafana

1. Einleitung

Im Rahmen des Moduls 321-07A habe ich ein umfassendes Monitoring-System für meine Server-Infrastruktur implementiert. Dieses Dokument beschreibt die Installation und Konfiguration der eingesetzten Komponenten Prometheus, Node-Exporter und Grafana sowie die Erstellung eines Dashboards zur Visualisierung der wichtigsten Systemmetriken.

1.1 Überblick der Komponenten

- **Prometheus:** Zeitreihendatenbank und Monitoring-System, das Metriken sammelt und speichert
- **Node-Exporter:** Agent, der Systemmetriken (CPU, RAM, Disk, etc.) sammelt und für Prometheus bereitstellt
- **Grafana:** Visualisierungsplattform zur Darstellung der in Prometheus gespeicherten Metriken

1.2 Ziele der Implementation

Das Ziel der Implementation war die Erstellung eines Monitoring-Dashboards, das folgende Informationen auf einen Blick anzeigt:

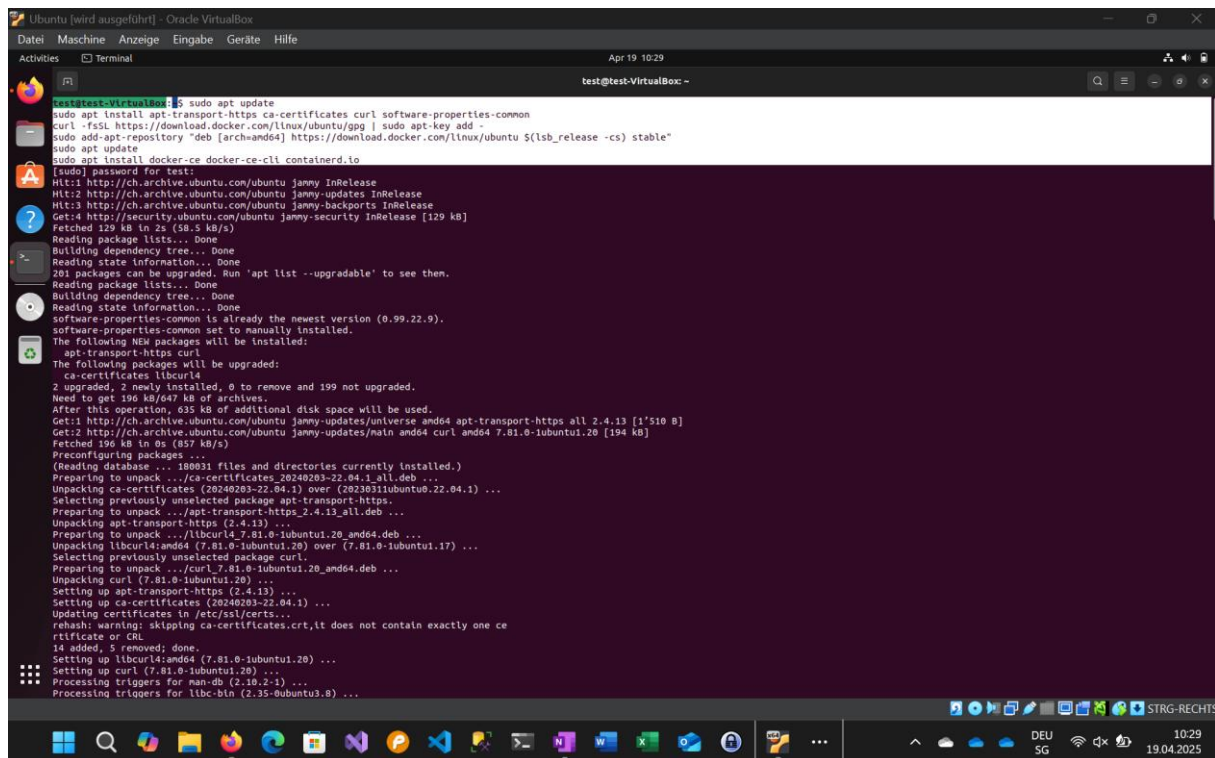
- Up/Down-Status der einzelnen Server
- Memory-Verbrauch von wichtigen Servern
- Uptime der Server
- Disk-Auslastung aller Server

2. Installation der Komponenten

Die Installation erfolgte auf Ubuntu 22.04 mit Docker als Container-Lösung.

2.1 Vorbereitungen

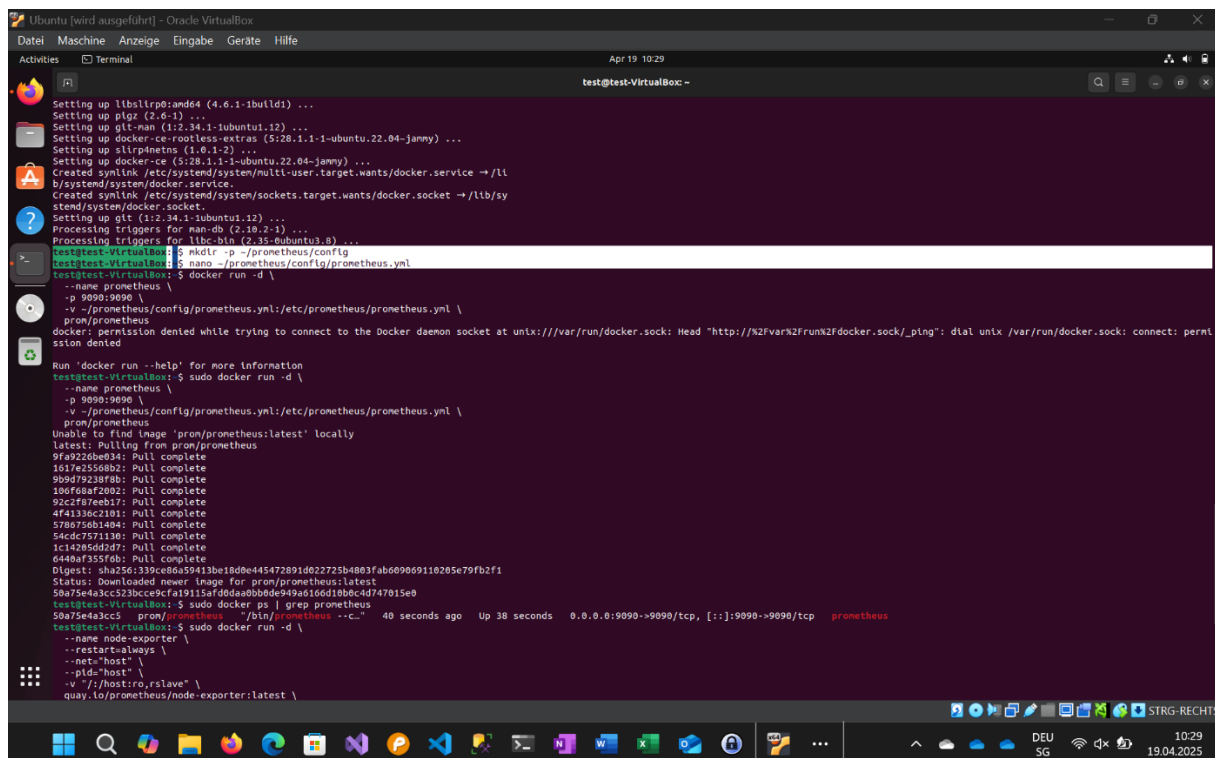
Zunächst habe ich sichergestellt, dass Docker installiert und korrekt konfiguriert ist:



```
test@test-VirtualBox:~$ sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io

[sudo] password for test:
Hit:1 http://ch.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ch.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ch.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Fetched 129 kB in 2s (58.5 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
201 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.99.22.9).
software-properties-common set to manually installed.
The following NEW packages will be installed:
  apt-transport-https curl
The following packages will be upgraded:
  ca-certificates libcurl4
2 upgraded, 2 newly installed, 0 to remove and 199 not upgraded.
Need to get 196 kB/407 kB of archives.
After this operation, 685 kB of additional disk space will be used.
Get:1 http://ch.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.13 [1'510 B]
Get:2 http://ch.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.20 [194 kB]
Fetched 196 kB in 0s (887 kB/s)
Preconfiguring packages ...
(Reading database ... 180031 files and directories currently installed.)
Preparing to unpack .../ca-certificates_20240203-22.04-1_all.deb ...
Unpacking ca-certificates (20240203-22.04-1) over (20230111ubuntu22.04-1) ...
Selecting previously unselected package apt-transport-https.
Preparing to unpack .../apt-transport-https_2.4.13_all.deb ...
Unpacking apt-transport-https (2.4.13) ...
Preparing to unpack .../libcurl4_7.81.0-1ubuntu1.20_amd64.deb ...
Unpacking libcurl4:amd64 (7.81.0-1ubuntu1.20) over (7.81.0-1ubuntu1.17) ...
Selecting previously unselected package curl.
Preparing to unpack .../curl_7.81.0-1ubuntu1.20_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.20) ...
Setting up apt-transport-https (2.4.13) ...
Setting up ca-certificates (20240203-22.04-1) ...
Updating certificates in /etc/ssl/certs...
rehash: warning: skipping ca-certificates.crt, it does not contain exactly one certificate or CRL
14 added, 5 removed; done.
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.20) ...
Setting up curl (7.81.0-1ubuntu1.20) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.0) ...
```

Anschliessend habe ich ein Verzeichnis für die Prometheus-Konfiguration erstellt:



```
Setting up libltp:amd64 (4.6.1-1build1) ...
Setting up pigz (2.6-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.12) ...
Setting up docker-ce-rootless-extras (5:28.1.1-1-ubuntu-22.04-jammy) ...
Setting up slirp4netns (1.0.1-2) ...
Setting up docker-ce (5:28.1.1-1-ubuntu-22.04-jammy) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Setting up git (1:2.34.1-1ubuntu1.12) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.0) ...
test@test-VirtualBox:~$ mkdir -p ~/prometheus/config
test@test-VirtualBox:~$ nano ~/prometheus/config/prometheus.yml
test@test-VirtualBox:~$ docker run -d \
  --name prometheus \
  -p 9090:9090 \
  -v ~/prometheus/config/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker.sock%2Fping": dial unix /var/run/docker.sock: connect: permission denied
Run 'docker run --help' for more information
test@test-VirtualBox:~$ sudo docker run -d \
  --name prometheus \
  -p 9090:9090 \
  -v ~/prometheus/config/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
9fa9220eb34: Pull complete
1617e25568b2: Pull complete
9b9d79238f8b: Pull complete
106f08af2002: Pull complete
92c4ff7ee517: Pull complete
4f4133ec2101: Pull complete
578875b1404: Pull complete
54c6787113b: Pull complete
1c14265d2d7: Pull complete
6448af355f6b: Pull complete
Digest: sha256:339ce86a59413be18d0e45472891d622723b4803fab090909110205e79fb2f1
Status: Downloaded newer image for prom/prometheus:latest
50a75e43c523bce9cf1a1915afdd0a0bbde949a6166d18b6c4d747815e0
test@test-VirtualBox:~$ sudo docker ps | grep prometheus
50a75e43c5  prom/prometheus  "/bin/prometheus -c..." 40 seconds ago  Up 38 seconds  0.0.0.0:9090->9090/tcp, [::]:9090->9090/tcp  prometheus
test@test-VirtualBox:~$ sudo docker run -d \
  --name node-exporter \
  --restart=always \
  --net="host" \
  --pid="host" \
  -v /:/host:ro,rsslave \
  quay.io/prometheus/node-exporter:latest \
```

2.2 Installation von Prometheus

2.2.1 Erstellung der Prometheus-Konfigurationsdatei

```
nano ~/prometheus/config/prometheus.yml
```

Folgende Konfiguration habe ich erstellt:

global:

scrape_interval: 15s

evaluation_interval: 15s

scrape_configs:

- job_name: "prometheus"

static_configs:

- targets: ["localhost:9090"]

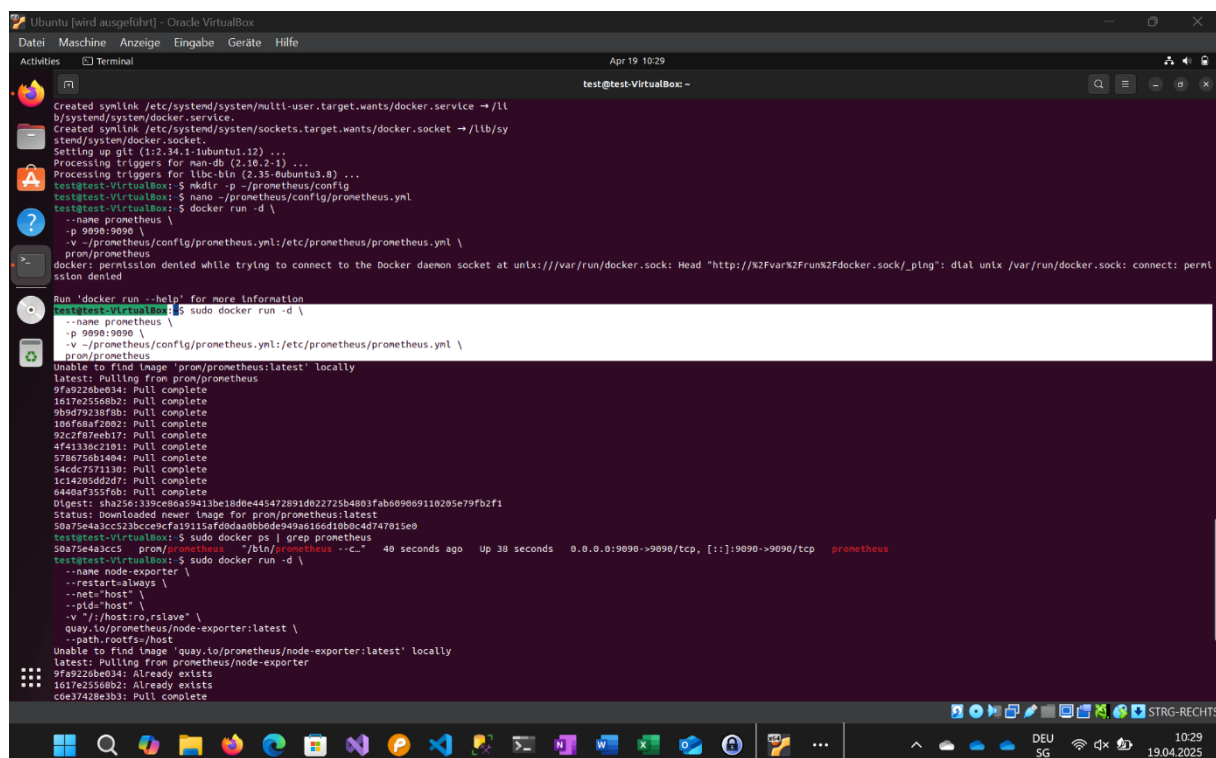
- job_name: "node-exporter"

static_configs:

- targets: ["localhost:9100"]

2.2.2 Start des Prometheus-Containers

Prometheus habe ich im Host-Netzwerkmodus gestartet, um Netzwerkprobleme zu vermeiden:



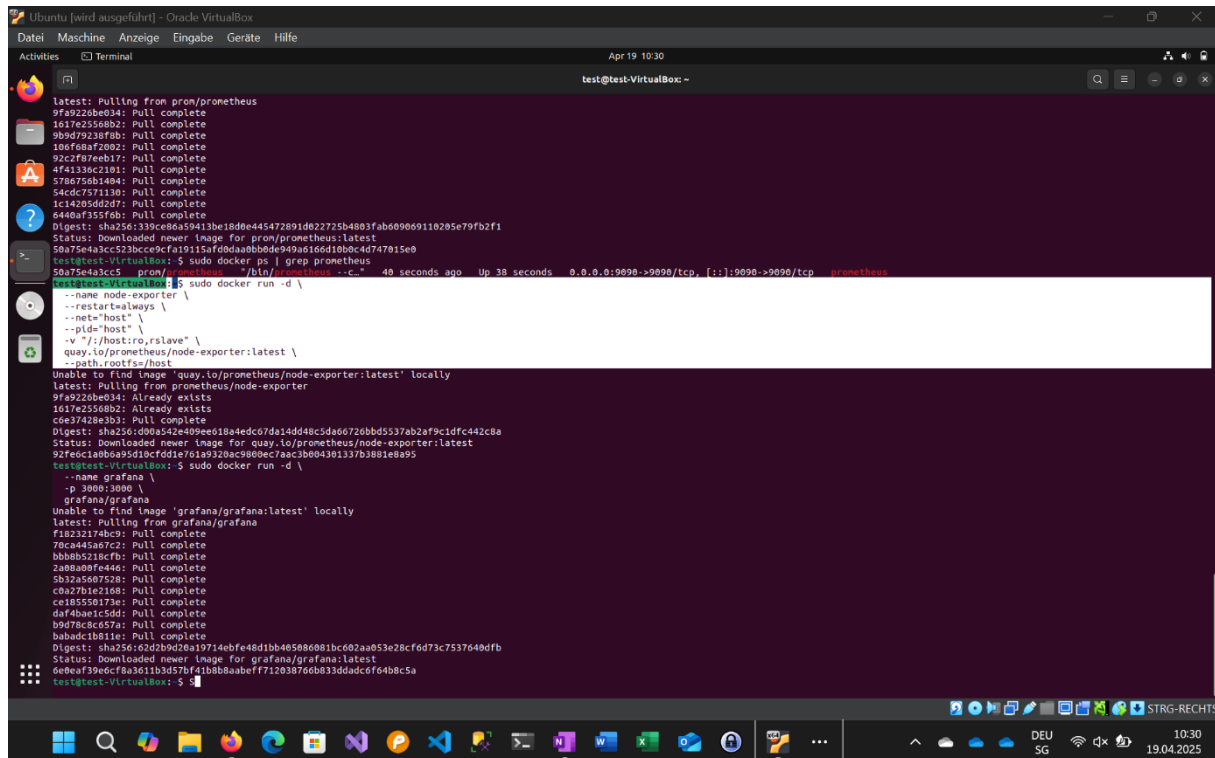
```
Ubuntu [wird ausgeführt] - Oracle VirtualBox
Datei Maschine Anzeige Eingabe Geräte Hilfe
Activities Terminal Apr 19 10:29 test@test-VirtualBox: ~

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/
b/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/sy
stemd/system/docker.socket.
Setting up git (1:2.34.1-1ubuntu1.12) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
test@test-VirtualBox:~$ mkdir -p ~/prometheus/config
test@test-VirtualBox:~$ nano ~/prometheus/config/prometheus.yml
test@test-VirtualBox:~$ docker run -d \
  --name prometheus \
  -p 9090:9090 \
  -v ~/prometheus/config/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://x2Fvarx2Frunx2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permil
sston denied

Run 'docker run --help' for more information.
test@test-VirtualBox:~$ sudo docker run -d \
  --name prometheus \
  -p 9090:9090 \
  -v ~/prometheus/config/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
9fa9228be034: Pull complete
1617e25568b2: Pull complete
9b0d723f8fab: Pull complete
106f68af2802: Pull complete
92c2f87eeb17: Pull complete
4f41336c2101: Pull complete
578d758b1044: Pull complete
54cdc7571130: Pull complete
1c14205d6d2d: Pull complete
6440af355f0b: Pull complete
Digest: sha256:339ce80a59413be18d0e445472891d622725b4803fab09069110205e79fb2f1
Status: Downloaded newer image for prom/prometheus:latest
50a75e4a3cc523bccc9cf81915af0daa8bb0de949a3a6d10b0c4d747815e0
test@test-VirtualBox:~$ sudo docker ps | grep prometheus
50a75e4a3cc5 prom/prometheus "/bin/prometheus --c..." 40 seconds ago Up 38 seconds 0.0.0.0:9090->9090/tcp, [::]:9090->9090/tcp prometheus
test@test-VirtualBox:~$ sudo docker run -d \
  --name node-exporter \
  --restart=always \
  --net=host \
  --pid=host \
  -v /:/host:ro,rslave \
  quay.io/prometheus/node-exporter:latest \
  --path.rootfs=/host
Unable to find image 'quay.io/prometheus/node-exporter:latest' locally
latest: Pulling from prometheus/node-exporter
9fa9228be034: Already exists
1617e25568b2: Already exists
c6e37428e3b3: Pull complete
```

2.3 Installation des Node-Exporters

Den Node-Exporter habe ich ebenfalls im Host-Netzwerkmodus gestartet:

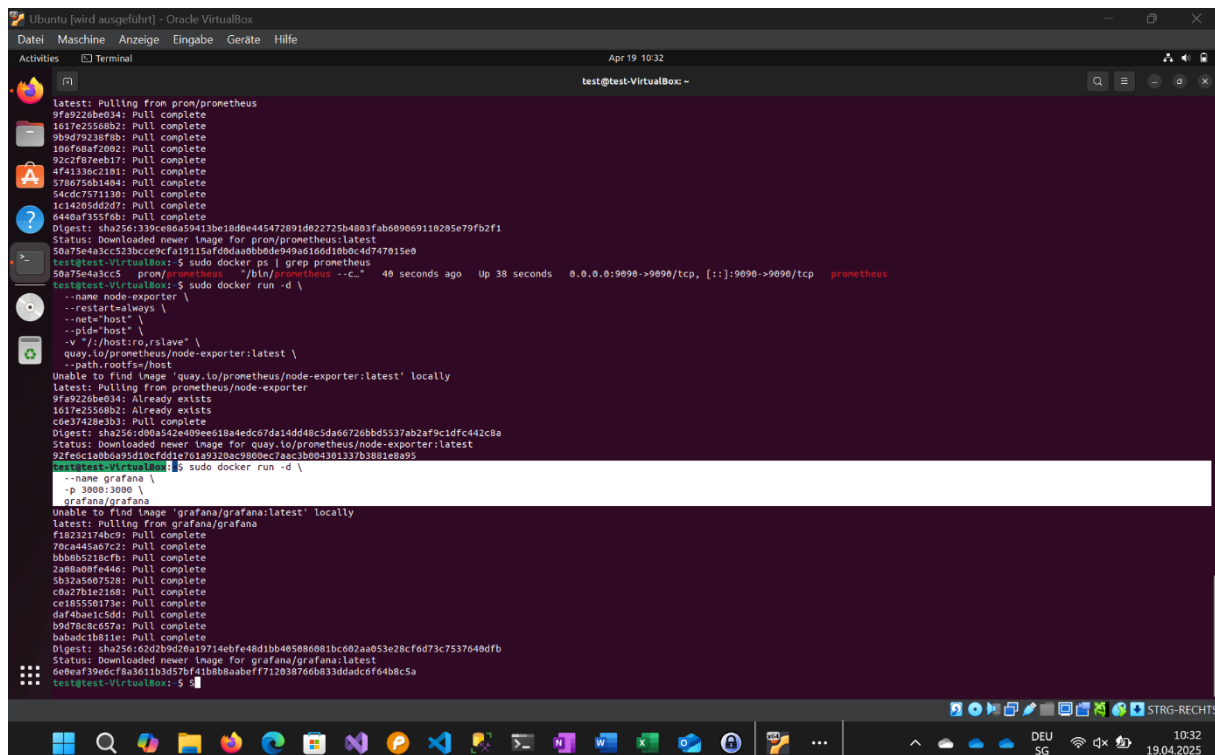


```
test@test-VirtualBox: ~$ docker pull quay.io/prometheus/node-exporter:latest
latest: Pulling from prom/prometheus
9fa9220be034: Pull complete
1617e25560b2: Pull complete
9b9d79238f8b: Pull complete
100f68af2002: Pull complete
92c2f8feeb17: Pull complete
4f41336c2101: Pull complete
5786750b1404: Pull complete
54cdc7571130: Pull complete
1c14205dd2d7: Pull complete
6440af355f6b: Pull complete
Digest: sha256:339ce6a59413be18d0e445472091d622725b4003fab09069110205e79fb2f1
Status: Downloaded newer image for prom/prometheus:latest
50a75e43c523bcc9cfa19115afddaa8bbde9a9a16dd10b0c4d747815e0
test@test-VirtualBox: ~$ docker ps | grep prometheus
50a75e43c523bcc9cfa19115afddaa8bbde9a9a16dd10b0c4d747815e0 40 seconds ago Up 38 seconds 0.0.0.0:9090->9090/tcp, [::]:9090->9090/tcp prometheus

test@test-VirtualBox: ~$ sudo docker run -d \
--name node-exporter \
--restart=always \
--net=host \
--pid=host \
-v "/:/host:ro,rslave" \
quay.io/prometheus/node-exporter:latest \
--path.rootfs=/host
Unable to find image 'quay.io/prometheus/node-exporter:latest' locally
latest: Pulling from prometheus/node-exporter
9fa9220be034: Already exists
1617e25560b2: Already exists
c6e37428e3b3: Pull complete
Digest: sha256:d00a542e409ee018a4edc67da14dd48c5da66720bd5537ab2af9c1dfc442c8a
Status: Downloaded newer image for quay.io/prometheus/node-exporter:latest
92fec1a0b0a95d10cfd9d1e701a9320ac9800ec7aac3b004301337b3881e8a95
test@test-VirtualBox: ~$ sudo docker run -d \
--name grafana \
-p 3000:3000 \
grafana/grafana
Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
f18232174bc9: Pull complete
70ca445a07c2: Pull complete
bb0b5218c9b: Pull complete
2a08a09fe44e: Pull complete
5b32a5007528: Pull complete
c0a27b1e2108: Pull complete
ce1855d0173e: Pull complete
daf4ba1c5dd: Pull complete
b0d78c6c57a: Pull complete
babad1b11e: Pull complete
Digest: sha256:62d2b9d20a19714ebfe48d1bb40508e081bc602aa053e28cf6d73c7537640dfb
Status: Downloaded newer image for grafana/grafana:latest
6e0eaf396cf8a3611b3d57bf41b8b8a8eff712038760b33ddadc6f64b8c5a
test@test-VirtualBox: ~$
```

2.4 Installation von Grafana

Grafana habe ich auch im Host-Netzwerkmodus gestartet:



```
test@test-VirtualBox: ~$ docker pull quay.io/prometheus/node-exporter:latest
latest: Pulling from prom/prometheus
9fa9220be034: Pull complete
1617e25560b2: Pull complete
9b9d79238f8b: Pull complete
100f68af2002: Pull complete
92c2f8feeb17: Pull complete
4f41336c2101: Pull complete
5786750b1404: Pull complete
54cdc7571130: Pull complete
1c14205dd2d7: Pull complete
6440af355f6b: Pull complete
Digest: sha256:339ce6a59413be18d0e445472091d622725b4003fab09069110205e79fb2f1
Status: Downloaded newer image for prom/prometheus:latest
50a75e43c523bcc9cfa19115afddaa8bbde9a9a16dd10b0c4d747815e0
test@test-VirtualBox: ~$ docker ps | grep prometheus
50a75e43c523bcc9cfa19115afddaa8bbde9a9a16dd10b0c4d747815e0 40 seconds ago Up 38 seconds 0.0.0.0:9090->9090/tcp, [::]:9090->9090/tcp prometheus

test@test-VirtualBox: ~$ sudo docker run -d \
--name node-exporter \
--restart=always \
--net=host \
--pid=host \
-v "/:/host:ro,rslave" \
quay.io/prometheus/node-exporter:latest \
--path.rootfs=/host
Unable to find image 'quay.io/prometheus/node-exporter:latest' locally
latest: Pulling from prometheus/node-exporter
9fa9220be034: Already exists
1617e25560b2: Already exists
c6e37428e3b3: Pull complete
Digest: sha256:d00a542e409ee018a4edc67da14dd48c5da66720bd5537ab2af9c1dfc442c8a
Status: Downloaded newer image for quay.io/prometheus/node-exporter:latest
92fec1a0b0a95d10cfd9d1e701a9320ac9800ec7aac3b004301337b3881e8a95
test@test-VirtualBox: ~$ sudo docker run -d \
--name grafana \
-p 3000:3000 \
grafana/grafana
Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
f18232174bc9: Pull complete
70ca445a07c2: Pull complete
bb0b5218c9b: Pull complete
2a08a09fe44e: Pull complete
5b32a5007528: Pull complete
c0a27b1e2108: Pull complete
ce1855d0173e: Pull complete
daf4ba1c5dd: Pull complete
b0d78c6c57a: Pull complete
babad1b11e: Pull complete
Digest: sha256:62d2b9d20a19714ebfe48d1bb40508e081bc602aa053e28cf6d73c7537640dfb
Status: Downloaded newer image for grafana/grafana:latest
6e0eaf396cf8a3611b3d57bf41b8b8a8eff712038760b33ddadc6f64b8c5a
test@test-VirtualBox: ~$
```

3. Konfiguration der Komponenten

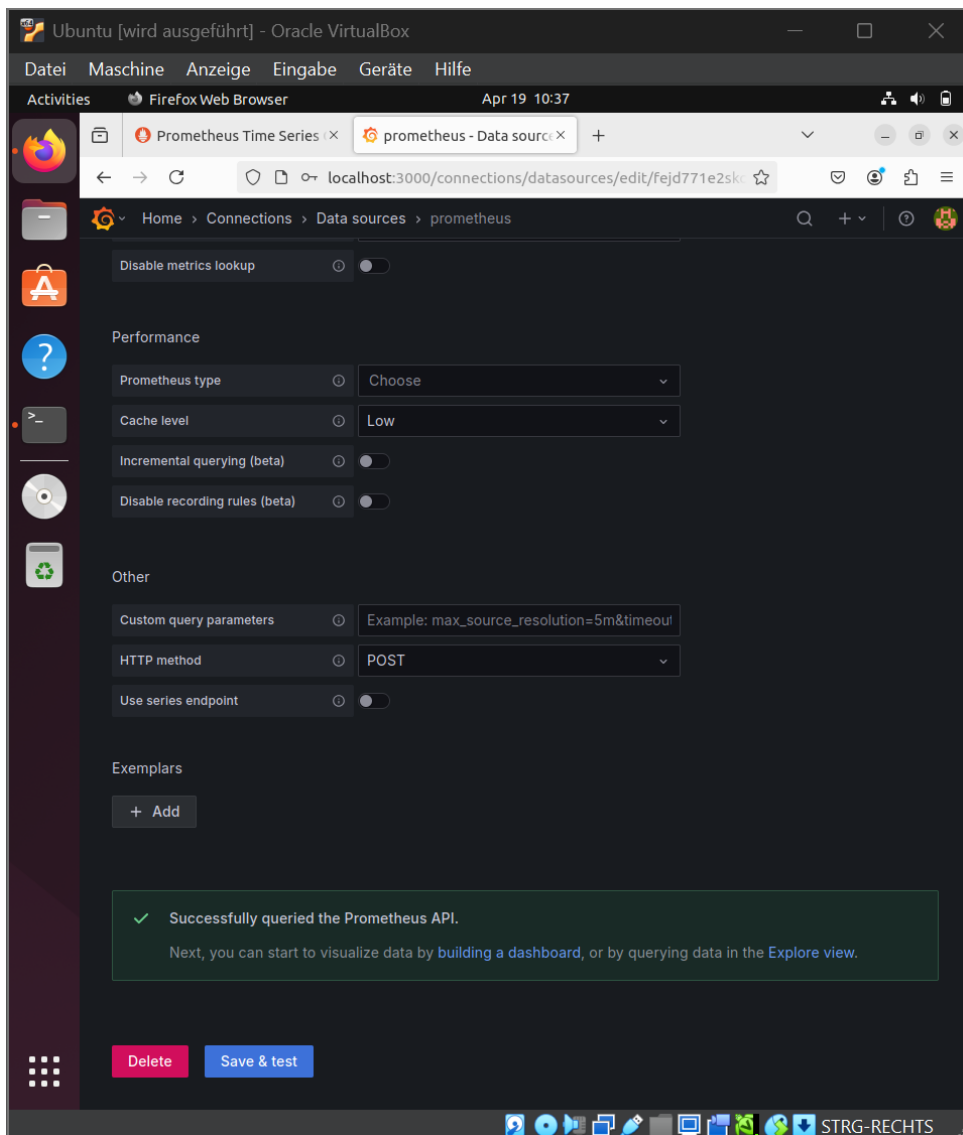
3.1 Zugriff auf die Web-Oberflächen

Nach erfolgreicher Installation waren die Web-Oberflächen unter folgenden URLs erreichbar:

- Prometheus: <http://localhost:9090>
- Grafana: <http://localhost:3000> (Standard-Login: admin/admin)

3.2 Konfiguration der Prometheus-Datenquelle in Grafana

In Grafana habe ich Prometheus als Datenquelle hinzugefügt:



4. Behobene Probleme während der Installation

4.1 Docker-Berechtigungen

Beim ersten Versuch, Docker-Container zu starten, trat ein Berechtigungsproblem auf:

docker: permission denied while trying to connect to the Docker daemon socket

Dieses Problem habe ich durch Hinzufügen von sudo vor den Docker-Befehlen gelöst.

4.2 Prometheus-Konfigurationsfehler

Bei der Konfiguration von Prometheus trat ein YAML-Parsing-Fehler auf:

parsing YAML file /etc/prometheus/prometheus.yml: yaml: line 9: did not find expected key

Dieses Problem habe ich durch Neuschreiben der Konfiguration mit korrekter Einrückung behoben.

4.3 Netzwerkprobleme zwischen Containern

Es gab Probleme bei der Kommunikation zwischen Prometheus und dem Node-Exporter sowie zwischen Grafana und Prometheus:

Post "http://localhost:9090/api/v1/query": dial tcp [::1]:9090: connect: connection refused

Dieses Problem habe ich durch den Start aller Container im Host-Netzwerkmodus (--net="host") gelöst, wodurch alle Container direkt auf das Host-Netzwerk zugreifen können.

5. Erstellung des Dashboards

5.1 Neue Dashboard-Erstellung

In Grafana habe ich ein neues Dashboard erstellt:

1. Klick auf "+" in der Seitenleiste
2. Auswahl von "Dashboard"
3. Klick auf "Add a new panel"

5.2 Konfiguration der Panels

5.2.1 Server-Status-Panel

- **Abfrage:** `up{job="node-exporter"}`
- **Visualisierungstyp:** Stat
- **Wertzuordnung:** 0 = Offline (rot), 1 = Online (grün)
- **Titel:** Server Status (Up/Down)

5.2.2 Memory-Nutzungs-Panel

- **Abfrage:** $100 - ((\text{node_memory_MemAvailable_bytes} * 100) / \text{node_memory_MemTotal_bytes})$
- **Visualisierungstyp:** Time series
- **Einheit:** Percent (0-100)

- **Titel:** Memory Usage (%)

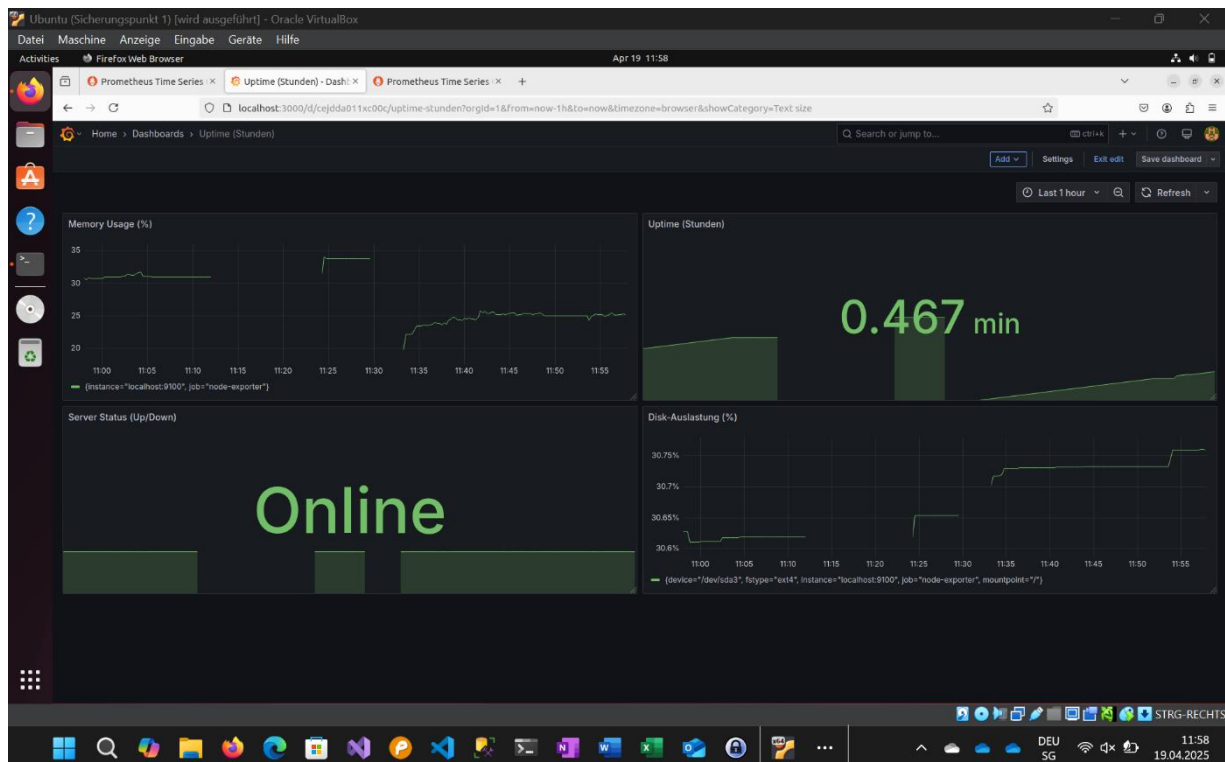
5.2.3 Uptime-Panel

- **Abfrage:** $(\text{node_time_seconds} - \text{node_boot_time_seconds}) / 3600$
- **Visualisierungstyp:** Stat
- **Einheit:** Hours (h)
- **Titel:** Uptime (Stunden)

5.2.4 Disk-Auslastungs-Panel

- **Abfrage:** $100 - ((\text{node_filesystem_avail_bytes}\{\text{mountpoint}="/" \} * 100) / \text{node_filesystem_size_bytes}\{\text{mountpoint}="/" \})$
- **Visualisierungstyp:** Gauge
- **Einheit:** Percent (0-100)
- **Schwellenwerte:** 0% (grün), 75% (orange), 90% (rot)
- **Titel:** Disk-Auslastung (%)

5.3 Dashboard-Layout



6. Ergebnis und Zusammenfassung

6.1 Erreichte Ziele

Mit der Implementation des Monitoring-Systems habe ich folgende Ziele erreicht:

- Echtzeit-Überwachung der Server-Verfügbarkeit
- Visualisierung des Speicherverbrauchs
- Überwachung der Uptime der Server
- Visualisierung der Festplattenauslastung