

Prometheus & Grafana

Autoren: Morris Tichy, Lukas Freudensprung

Inhaltsverzeichnis

1. Prometheus/Grafana	2
1.1. Theorie	2
1.1.1. Was ist Prometheus?	2
1.1.2. Was ist Grafana?	2
1.1.3. Komponenten	2
1.1.4. Überwachungs- und Fehlermanagement	3
1.1.5. Wie werden Daten ausgewertet?	3
1.2. Installation	3
1.2.1. Grundkonfiguration Prometheus Server	3
1.2.2. Installation Prometheus	4
1.2.2.1. Überprüfen des Web-Zugriffs	5
1.2.3. Installation Grafana	6
1.2.3.1. Überprüfung des Web-Zugriffs	6
1.2.4. Node Explorer Ubuntu	7
1.2.5. Node Explorer Windows	7
1.3. Dashboard einrichten	8
1.4. Alerts	8
1.4.1. Zustände von Alarminstanzen:	9
1.4.2. Zustände von Alarmregeln:	9
1.5. Alert Manager	11
1.6. Testen	12

1. Prometheus/Grafana

1.1. Theorie

1.1.1. Was ist Prometheus?

Prometheus ist ein Open-Source-Überwachungstool, das auf Metriken basiert und speziell für die Überwachung von Anwendungen und Infrastruktur entwickelt wurde. Es ermöglicht die Erfassung und Analyse von Leistungsdaten wie CPU-Auslastung, Speichernutzung und Netzwerkauslastung, was dabei hilft, die Effizienz und Stabilität von IT-Systemen zu gewährleisten. Durch seine flexible Architektur und die Möglichkeit der Integration in verschiedene Systeme ist Prometheus ein wertvolles Werkzeug für Entwickler und IT-Administratoren, um die Gesundheit ihrer Systeme zu überwachen und Probleme frühzeitig zu erkennen.

1.1.2. Was ist Grafana?

Grafana ist ein Open-Source-Tool, das speziell dafür entwickelt wurde, Daten aus Prometheus grafisch darzustellen. Es ermöglicht die Erstellung detaillierter Dashboards, die in Echtzeit Einblicke in die Leistung von Systemen wie CPU-Auslastung und Speichernutzung bieten. Mit Grafana können Benutzer Alarme einrichten, um schnell auf ungewöhnliche Aktivitäten reagieren zu können. Es ist das perfekte grafische Tool zur Ergänzung von Prometheus und ein unverzichtbares Werkzeug für IT-Administratoren und Entwickler, die ihre Systemüberwachung optimieren möchten.

1.1.3. Komponenten

Prometheus Server speichert und wertet Metriken ab, indem er ein Pull-Modell verwendet und Metriken von den zu überwachenden Zielen über das HTTP-basierte Exporter-Protokoll abrufen.

Exporter Metriken die von den überwachten Zielen gesammelt und über HTTP-Endpunkte bereitgestellt werden, dabei wird das Exporter-Protokoll verwendet

Targets überwachenden Ziele, von denen Prometheus Metriken sammelt, indem es deren HTTP-Endpunkte abfragt

Grafana Visualisierung von Metriken in Dashboards

Alertmanager (optional) Benachrichtigen von Benutzern oder Systemen

1.1.4. Überwachungs- und Fehlermanagement

Alerting: Wenn ein Fehler auftritt, wird eine Benachrichtigung (Alert) gesendet, um jemanden zu alarmieren.

Debugging: Sobald jemand benachrichtigt wurde, muss der Fehler gefunden und behoben werden.

Trending: Langzeitüberwachung zur Ressourcenplanung, um die Nutzung und Bedürfnisse besser zu verstehen.

Plumbing: Überwachungssysteme fungieren als Datenverarbeitungspipelines und können manchmal für andere Zwecke genutzt werden, anstatt separate maßgeschneiderte Lösungen zu entwickeln.

1.1.5. Wie werden Daten ausgewertet?

Profiling: sammelt begrenzte Kontextinformationen für eine begrenzte Zeit und wird für taktisches Debugging verwendet, erfordert jedoch oft eine Verringerung des Datenvolumens, um in andere Überwachungskategorien zu passen. Tracing: selektiert einen Prozentsatz von Ereignissen, um Einblicke in Codepfade und Latenzzeiten zu erhalten, mit der Möglichkeit zur verteilten Nachverfolgung in Mikroservice-Architekturen. Logging: zeichnet begrenzte Kontextinformationen für eine bestimmte Anzahl von Ereignissen auf und wird in Kategorien wie Transaktionsprotokollen, Anforderungsprotokollen, Anwendungsprotokollen und Debug-Protokollen unterteilt. Metrics: erfassen aggregierte Daten über verschiedene Ereignisse im Laufe der Zeit und bieten Einblicke in die Leistung und das Verhalten eines Systems, wobei die Kontextinformationen begrenzt sind, um das Datenvolumen und die Verarbeitungsanforderungen zu optimieren.

1.2. Installation

1.2.1. Grundkonfiguration Prometheus Server

Die folgenden Konfigurationen wurden auf dem Ubuntu Server bzw. auf dem Prometheus eingefügt.

```
1  hostnamectl hostname Prometheus bash
2
3  sudo nano /etc/netplan/00-installer-config.yaml
4  network:
```

```
5 version: 2
6 renderer: networkd
7 ethernets:
8   LAN1:
9     addresses:
10      - 192.168.10.100/24
11     routes:
12      - to: default
13        via: 192.168.10.1
14     match:
15       macaddress: 00:0c:29:46:22:de
16     set-name: LAN1
17
18 sudo netplan apply
19
```

Mit dieser Konfiguration werden die Netzwerk Einstellungen festlegt.

1.2.2. Installation Prometheus

Anschließend wird auf dem Prometheusserver die verschiedenen Dienste für Prometheus installiert.

```
wget https://github.com/prometheus/prometheus/releases/download/v2.
1 51.1/prometheus-2.51.1.linux-amd64.tar.gz # Installieren von bash
Prometheus
2 tar xzf prometheus-2.51.1.linux-amd64.tar.gz # entpacken der Datei
3 mv prometheus-2.51.1.linux-amd64 /etc/prometheus # verschieben der Datei
in den Ordner /etc/prometheus
4
5 nano /etc/systemd/system/prometheus.service
6 [Unit]
7 Description=Prometheus
8 Wants=network-online.target
9 After=network-online.target
10 [Service]
11 ExecStart=/etc/prometheus/prometheus --config.file=/etc/prometheus/
prometheus.yml
12 Restart=always
13 [Install]
```

```
14 WantedBy=multi-user.target
```

Hier wird der Prometheus Server installiert und die Konfiguration wird in der Datei /etc/prometheus/prometheus.yml festgelegt. Anschließend wird ein Service erstellt, der den Prometheus Server startet.

Mit den Befehlen:

```
1 systemctl daemon-reload
2 systemctl restart prometheus
3 systemctl enable prometheus
4 systemctl status prometheus
```

werden die **Services** neugestartet.

```
/etc/prometheus/prometheus --config.file=/etc/prometheus/
1 prometheus.yml # Starte den Prometheus Server mit einer bestimmten
Konfigurationsdatei
```

Anschließend kann in der Konfigurationsdatei Jobs angelegt werden auf denen der Node Explorer installiert worden ist.

```
nano /etc/prometheus/prometheus.yml # A scrape configuration
1 containing exactly one endpoint to scrape from node_exporter running
on a host:
2 scrape_configs: # The job name is added as a label `job=<job_name>` to any
timeseries scraped from this config.
3 - job_name: 'node'
4   # metrics_path defaults to '/metrics'
5   # scheme defaults to 'http'.
6   static_configs:
7     - targets: ['localhost:9100']
8 systemctl restart prometheus
9 systemctl status prometheus
```

1.2.2.1. Überprüfen des Web-Zugriffs

Danach sollte unter der :9090 der Prometheus Server erreichbar sein.

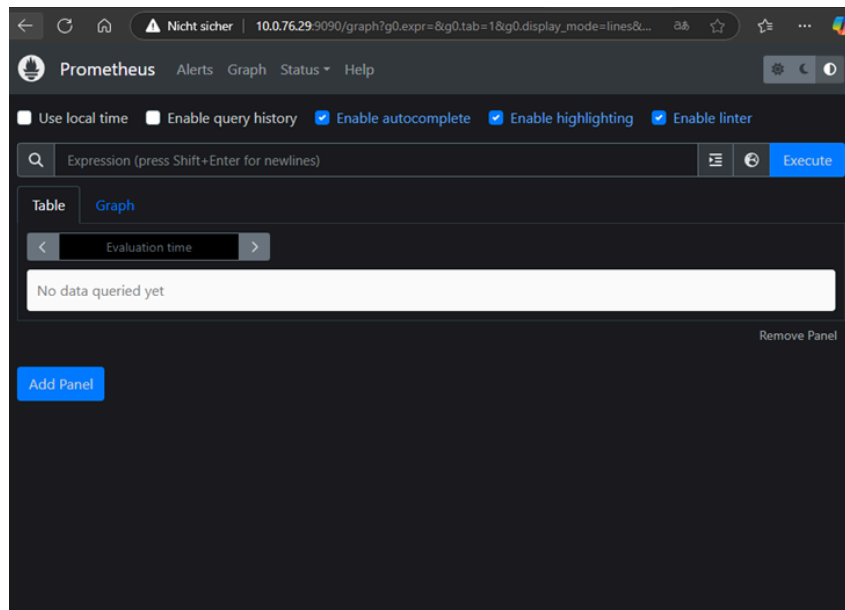


Abbildung 1: Prometheus Web Access

1.2.3. Installation Grafana

```
1 sudo apt-get install -y adduser libfontconfig musl # Installation
   von Installationn der Abhängigkeiten für Grafana
2 wget https://dl.grafana.com/enterprise/release/grafana-enterprise_10.4.1_
   amd64.deb # installation von Grafana
3
4 sudo apt install musl # Installation von musl
5
6 sudo dpkg -i grafana-enterprise_10.4.1_amd64.deb # Entpacken der Datei
```

Durch folgende Befehle wird der Grafana Server gestartet und aktiviert.

```
1 sudo systemctl restart grafana-server
2 sudo systemctl enable grafana-server
3 sudo systemctl status grafana-server
```

1.2.3.1. Überprüfung des Web-Zugriffs

Danach sollte unter der <IP-Adresse>:3000 der Grafana Server erreichbar sein. Die Anmeldedaten für den Web-Zugriffs sind User: „admin“, Passwort: „admin“. Nach Eingabe von Benutzername und Passwort wird man aufgefordert ein neues Passwort festzulegen.

1.2.4. Node Explorer Ubuntu

Damit wir Server auf dem Dashboard anzeigen lassen können, müssen wir auf den Geräten den Node_exporter einrichten. Folgenden Befehle werden dafür verwendet.

```
wget https://github.com/prometheus/node_exporter/releases/download/v
1 1.7.0/node_exporter-1.7.0.linux-amd64.tar.gz # Download von Node_exporter bash
2
3 tar xzf node_exporter-1.7.0.linux-amd64.tar.gz # Entpackung der Datei
4 mv node_exporter-1.7.0.linux-amd64 /etc/node_exporter # Verschieben der Datei in den Ordner /etc/node_exporter
```

```
1 nano /etc/systemd/system/node_exporter.service # Erstellen der Service Datei bash
2 [Unit]
3 Description=Node Exporter
4 Wants=network-online.target
5 After=network-online.target
6 [Service]
7 ExecStart=/etc/node_exporter/node_exporter
8 Restart=always
9 [Install]
10 WantedBy=multi-user.target
```

Durch folgende Befehle wird der Node Exporter gestartet und aktiviert.

```
1 systemctl daemon-reload bash
2 systemctl restart node_exporter
3 systemctl enable node_exporter
4 systemctl status node_exporter
```

1.2.5. Node Explorer Windows

Damit wir Server auf dem Dashboard anzeigen lassen können, müssen wir auf den Geräten den Node_exporter einrichten. Folgenden Befehle werden dafür verwendet.

```
1 https://github.com/prometheus-community/windows_exporter/releases bash
2 -> .msi installieren # Download von Node_exporter
3
```

```
4 msiexec /i C:\\Users\\Administrator\\Downloads\\windows_exporter-0.25.1-  
arm64.msi ENABLED_COLLECTORS="ad,cpu"
```

Bei Windows ist es wichtig zu beachten, dass der Standardport für den Node-Exporter 9182 ist.

1.3. Dashboard einrichten

Damit die Metriken von den Maschinen in numerische Werte umgewandelt werden, muss ein neues Dashboard angelegt werden. Dazu wird unter **Dashboard > New > Import** kann ein benutzerdefiniertes Dashboard oder ein vorgefertigtes Dash-board importierte werden. Mit der Dashboard ID: 1860 sieht das Dashboard wie folgt aus:

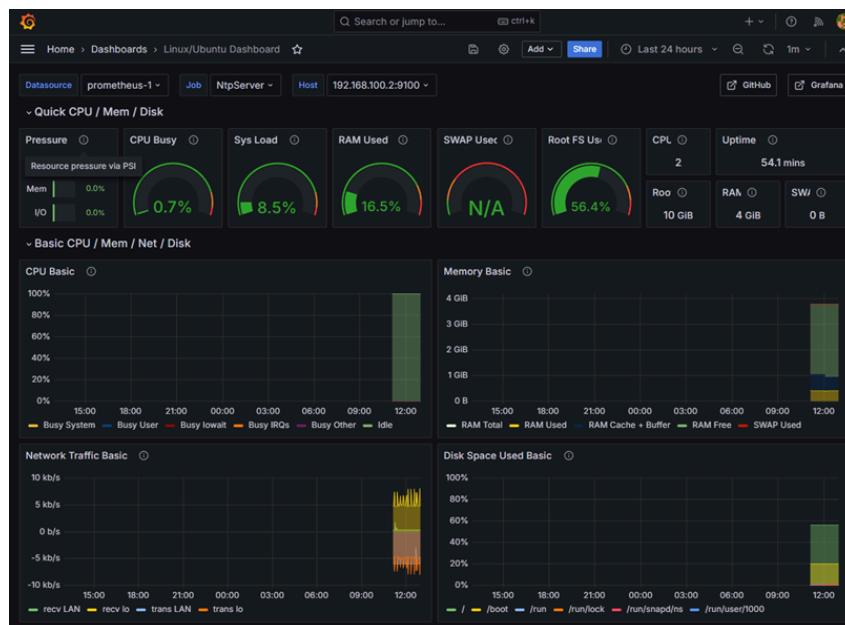


Abbildung 2: Linux/Ubuntu Dashboard

Unter dem Reiter Jobs können die verschiedenen Geräte ausgewählt werden, die davor in der prometheus.yml Datei angelegt wurden. Nach der Konfiguration des Dashboards kann man Alerts festlegen, die in unserem Fall die CPU-Auslastung überwacht.

1.4. Alerts

In Grafana gibt es verschiedene Zustände für Alarmregeln und deren Instanzen, die den aktuellen Status eines Alarms widerspiegeln:

1.4.1. Zustände von Alarminstanzen:

Normal: Der Zustand eines Alarms, wenn die definierte Bedingung (Schwellenwert) nicht erfüllt ist.

Pending: Der Alarm hat den Schwellenwert überschritten, jedoch für weniger als die definierte Wartezeit.

Alerting: Der Alarm hat den Schwellenwert für länger als die definierte Wartezeit überschritten und wird als aktiv betrachtet.

No Data: Der Alarm erhält keine Daten oder alle Werte sind null.

Error: Bei der Auswertung der Alarmregel ist ein Fehler oder ein Timeout aufgetreten.

1.4.2. Zustände von Alarmregeln:

Normal: Keine der Alarminstanzen befindet sich im Zustand „Pending“ oder „Alerting“.

Pending: Mindestens eine Alarminstanz befindet sich im Zustand „Pending“.

Firing: Mindestens eine Alarminstanz befindet sich im Zustand „Alerting“.

In unserem Fall konfigurieren wir einen Alert für die CPU-Auslastung. Falls die Auslastung über 0.3 wird, ein Alert erstellt und dieser an den Alert Manager weitergeschickt. Die Konfiguration dieses Alerts beschreibt die nachfolgenden Screenshots.

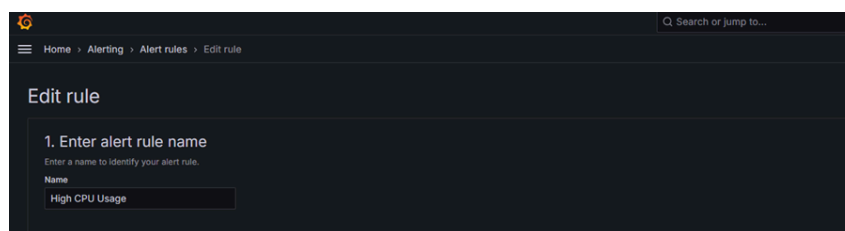


Abbildung 3: Regel Namen festlegen

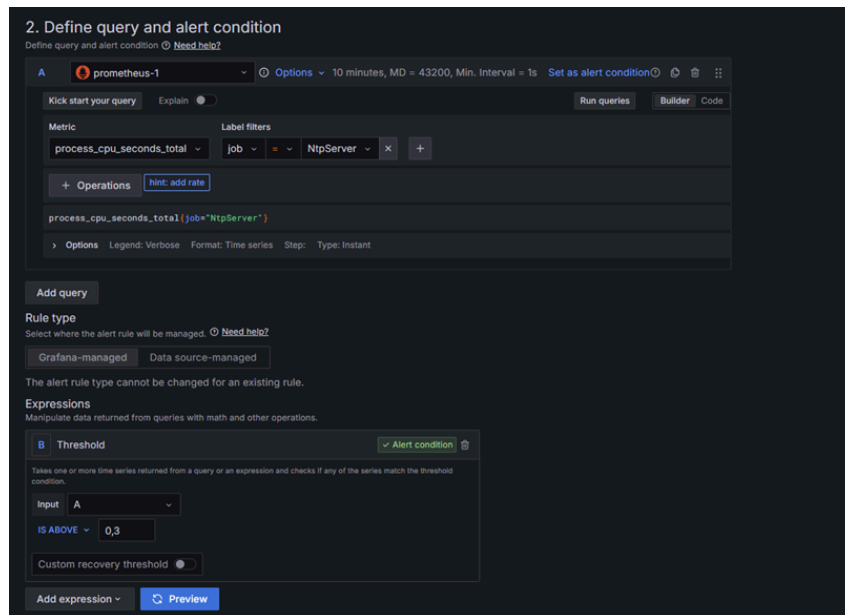


Abbildung 4: Regel definieren

In dieser Abbildung ist zu erkennen, dass die CPU Auslastung ausgewertet wird. Falls die Auslastung über 0.3 liegt, wird ein Alert erstellt.

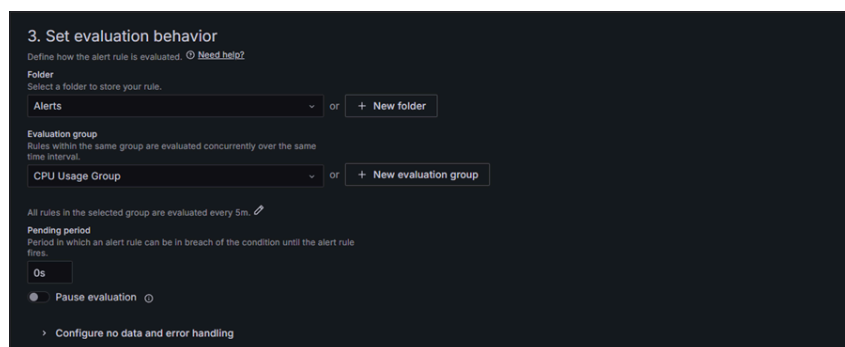


Abbildung 5: Ordner festlegen

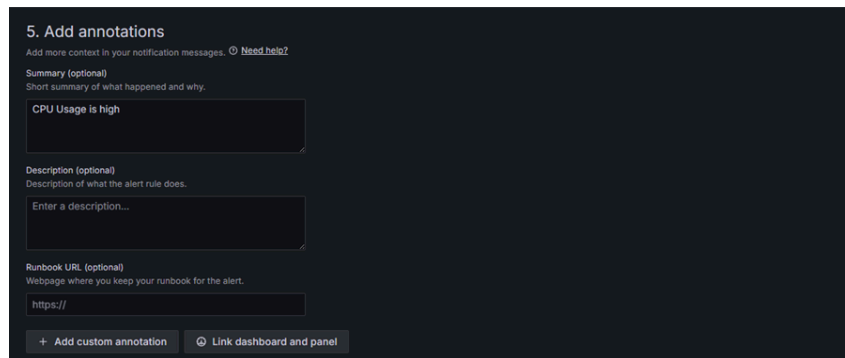


Abbildung 6: Dashboard und Link festlegen

Nachdem die Regel konfiguriert kann anschließend noch ein Alert Manager eingebunden werden. Der Alert Manager ist ein System, das es ermöglicht, Benachrichtigungen über Alerts zu senden.

1.5. Alert Manager

Der Alert Manager wurde lokal auf der Prometheus Maschine installiert. Die Konfiguration des Alert Managers wird in der Datei prometheus.yml beschrieben.

```
wget https://github.com/prometheus/alertmanager/releases/download/v
1 0.23.0/alertmanager-0.23.0.linux-amd64.tar.gz # Download von Alert Manager bash
2 tar -xvf alertmanager-0.23.0.linux-amd64.tar.gz # Entpacken
3
4 sudo mkdir -p /alertmanager-data /etc/alertmanager
5
6 sudo mv alertmanager-0.23.0.linux-amd64/alertmanager /usr/local/bin/ # Verschieben der Datei
7 sudo mv alertmanager-0.23.0.linux-amd64/alertmanager.yml /etc/alertmanager/ # Verschieben der Datei
```

Service erstellen

```
1 sudo nano /etc/systemd/system/alertmanager.service bash
2 [Unit]
3 Description=Alertmanager
4 Wants=network-online.target
5 After=network-online.target
```

```
6 [Service]
7 ExecStart=/usr/local/bin/alertmanager --storage.path=/alertmanager-data --
  config.file=/etc/alertmanager/alertmanager.ymlRestart=always
8 [Install]
9 WantedBy=multi-user.target
```

Durch folgende Befehle wird der Alert Manager gestartet und aktiviert.

```
1 sudo systemctl enable alertmanager.service
2 sudo systemctl start alertmanager.service
3 sudo systemctl status alertmanager.service
```

AlertManager in Prometheus integrieren

```
1 sudo nano /etc/prometheus/prometheus.yml
2 # my global config
3 global:
4     scrape_interval: 15s # Set the scrape interval to every 15 seconds.
  Default is every 1 minute.
5     evaluation_interval: 15s # Evaluate rules every 15 seconds. The default
  is every 1 minute.
6     # scrape_timeout is set to the global default (10s).
7
8 # Alertmanager configuration
9 alerting:
10 alertmanagers:
11     - static_configs:
12         - targets:
13             - alertmanager:9093
```

Abschließend muss das Prometheus Service neu gestartet werden.

```
1 sudo systemctl restart prometheus.service
```

1.6. Testen

Wenn man jetzt den NTP-Server neustartet geht die CPU Auslastung hoch und wir bekommen zweimal einen Alert. In Grafana steht Firing, das beschreibt das sich eine Alarminstanz in Alerting befindet. In Grafana ist diese Nachricht bei den Alerts zu sehen.

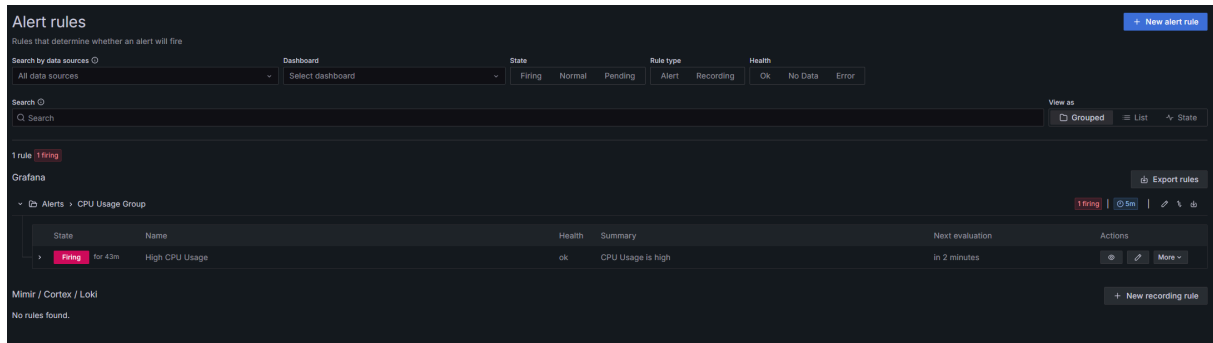


Abbildung 7: Alert ist im Zustand Firing

Unter der <IP-Adresse>:9093 können wir sehen, dass es einen neuen Alert gibt unter der Instanz 192.168.100.2. Der nachfolgende Screenshot zeigt diesen Alert.

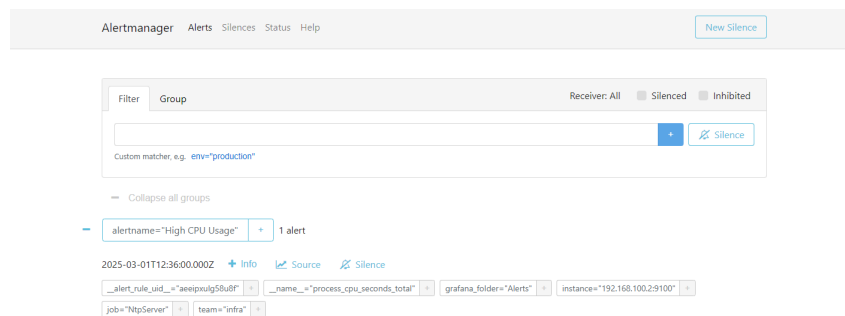


Abbildung 8: Alertmanager Alert