Licht ins Dunkel

Monitoring mit Prometheus und Grafana.

Alexander Schreiber <als@thangorodrim.ch>

http://www.thangorodrim.ch/

Chemnitzer Linux-Tage 2019, 2019-03-17

Nicht alles was zählt, kann gezählt werden, und nicht alles was gezählt werden kann, zählt!

— *Albert Einstein*

Inhalt

Über den Autor

- beschäftigt sich seit über 20 Jahren mit Linux
- tätig als Systemingenieur bei Google Switzerland
- hat den einen oder anderen Computer im Haus
- disclaimer: Dieser Vortrag hat nichts mit meinem Arbeitgeber zu tun und entspringt reinem Privatvergnügen.

Um was geht es?

- Monitoring/Alerting
- Blackbox/Whitebox
- Graphen
- Alerting
- Vorführung

Monitoring?

- Warum eigentlich, läuft doch alles ...
- Wirklich?
  - Wieviele Anfrage/s? Wie sieht die Tageskurve aus?
  - Reicht die Kapazität oder brauchen wir mehr?
  - Gibt es auffällige Muster? Lastspitzen?
  - Sind die Latenzen mit der neuen Softwareversion anders?
  - Hinweise auf Speicherlecks?
- Alle diese Fragen und mehr beantwortet (gutes) Monitoring.

Alerting?

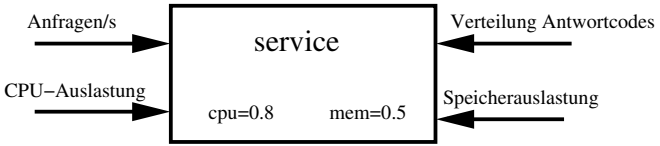
- Wir merken schon, wenn was nicht stimmt ...
  - Wie? Gelegentlich/dauerhaft Graphen anstarren?
  - Rechtzeitig?
  - Auch nachts? (Merke: Das Internet schläft nie)
- Bespiele:
  - Webserver-Anfragen über 20x des Normalwertes.
  - Latenzen im zweistelligen Sekundenbereich (von 300 ms)
  - Anteil Webserver-Fehlermeldungen (500) übersteigt 90%
  - ... und das schon seit einer halben Stunde
- Mit Alerting erfährt man davon, bevor Chef oder Kunde anruft ...

Blackbox monitoring

- interner Zustand des Dienstes nicht verfügbar
- Monitoring extern verfügbarer Signale (z.B. Antworten auf Anfragen) mit Probern
- nur begrenzte Rückschlüsse auf Zustand möglich
- Erkennung kritischer interner Zustände oft kaum/zu spät möglich
- manchmal aber die einzige Möglichkeit (proprietäre Dienste)

Whitebox

- interne Zustandsinformations des Dienstes verfügbar
- exportiert in (idealerweise) Echtzeit in verwertbarem Format
- interne Zustandsinformationen via push/pull ins Monitoring
- direkte Erkennung interner Zustände
- genaue Überwachung möglich



Motivation

- Heimnetzwerk mit 2 Dutzend Maschinen
- Monitoring (Graphen) mit Munin
- Monitoring (Alerting) mit Nagios & Icinga
- nicht so ganz zufriedenstellend:
  - Munin:
    - einfach aufzusetzen
    - skaliert nicht (sowohl collector als auch graph generation)
    - alles oder nichts Ansatz für Graphen
  - Nagios/Icinga:
    - white box monitoring für remote hosts ist Arbeit
    - kein brauchbares silencing für flapping alerts → alert spam

Hintergrund

- entwickelt seit 2012 bei SoundCloud
- Grund: vorhandene Lösungen (StatsD, Graphite) nicht ausreichend
- inspiriert von Borgmon (Google internes Monitoring)
- Ziele:
  - mehrdimensionales Datenmodel
  - skalierbare Datensammlung
  - leistungsstarke Abfragesprache
  - einfacher Betrieb
- 2013 intern bei SoundCloud in Betrieb, 2015 veröffentlicht
- Prometheus 1.0: Juli 2016
- aktuell (März 2019): Prometheus 2.8.0

Übersicht

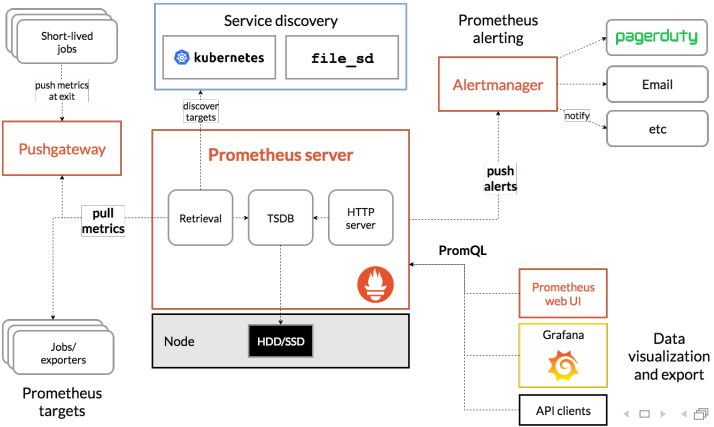
- Prometheus Komponenten:
  - Datenquellen: Exporter (z.B. `node_exporter`)
  - Datensammler: Prometheus
  - Datenspeicher: TSDB (Prometheus integriert)
  - Graphen: Prometheus, aber Grafana empfohlen
  - Alerting: Alertmanager
- überwiegend in Go geschrieben
- Plattformen direkt unterstützt: Linux/amd64, Linux/arm, Darwin/amd64
- weitere Plattformen: Go verfügbar? Selber bauen.
- alternativ: spezielle Exporter als Brücken (später)

Exporter

- Exporter stellen Daten für Prometheus bereit
- scraping via http, z.B. http://a.host.com:9100/metrics
- Mehrdimensionalität durch Label, z.B. cpu=0, cpu=1
- Metrik-Typen: gauge, counter, histogram, summary (letzte 2 u.a. für Φ Quantile)
- Basisformat 2+n Zeilen, n $\geq$ 1:
  - HELP metric helptext: beschreibt Metrik inhaltlich ("Number of connection states.")
  - TYPE metric type: Typ der Metrik ("gauge")
  - metric{label=value} Wert: ("tcp_connection_states{state="listen"} 32")

```
# HELP node_tcp_connection_states Number of connection states.
# TYPE node_tcp_connection_states gauge
node_tcp_connection_states{state="established"} 26
node_tcp_connection_states{state="listen"} 32
node_tcp_connection_states{state="time_wait"} 7
```

Prometheus Architektur



servicebasiertes Monitoring

- Prometheus Monitoring basiert auf dem Service-Konzept
- überwachte Einheit ist ein Service: Maschine, Webserver, Datenbank, ...
- Service bietet Metriken via Exporter an, Prometheus holt sie ab
- Standard scrape Intervall: 60s
- Merke: Auch eine Maschine ist nur ein Service (`node_exporter`)
- Metriken mit servicespecifischem Prefix (`node_total_memory`)

Basiskonzepte 1/2

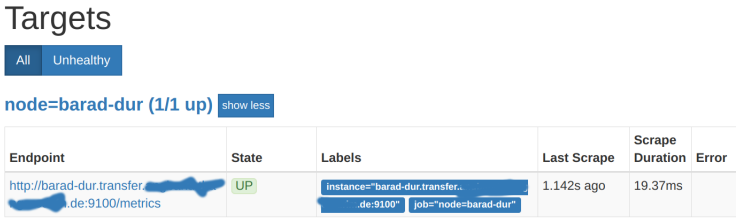
- Instanzen & Jobs:
  - überwachter endpoint: Instanz
  - Sammlung zusammengehöriger Instanzen: Job
- Beispiel:
  - job: `web-server`
    - Instanz 1: web1.prod.firma.de:7501
    - Instanz 2: web1.prod.firma.de:7502
    - Instanz 3: web2.prod.firma.de:7501
    - Instanz 4: web2.prod.firma.de:7502

Basiskonzepte 2/2

- automatische Labels & Timeseries
- `up{job="<job>", instance="<instance>"}`: 1: Instanz erreichbar
  - 1: Instanz erreichbar und scraping erfolgreich
  - 0: Instanz nicht erreichbar oder scraping fehlgeschlagen
  - eignet sich für Verfügbarkeitsüberwachung
- `scrape_duration_seconds{job="<job>", instance="<instance>"}`: Scrapingdauer
- `scrape_samples_scraped{job="<job>", instance="<instance>"}`: Anzahl Samples

Zustand des Prometheus-Servers

- Server lauscht auf 9090/tcp (HTTP)
- Konfiguration
- Alarme
- graph explorer
- Zustand der überwachten Ziele



Exporter, fertig und zum Selberschnitzen

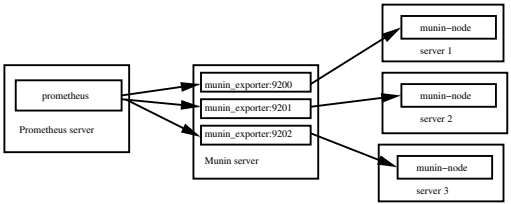
- Eine grosse Anzahl fertiger Exporter ist verfügbar
- von `prometheus.io`: blackbox (Prober), Consul, Graphite, HAProxy, Memcached, MySQLd, Statsd, node
- noch viel mehr: `https://prometheus.io/docs/instrumenting/exporters/`
- Client-Bibliotheken ermöglichen Integration in eigene Dienste
  - offizielle Client-Bibliotheken für Go, Java, Scala, Python, Ruby
  - inoffizielle für noch deutlich mehr `https://prometheus.io/docs/instrumenting/clientlibs/`

Exporter-Integration, Python Beispiel

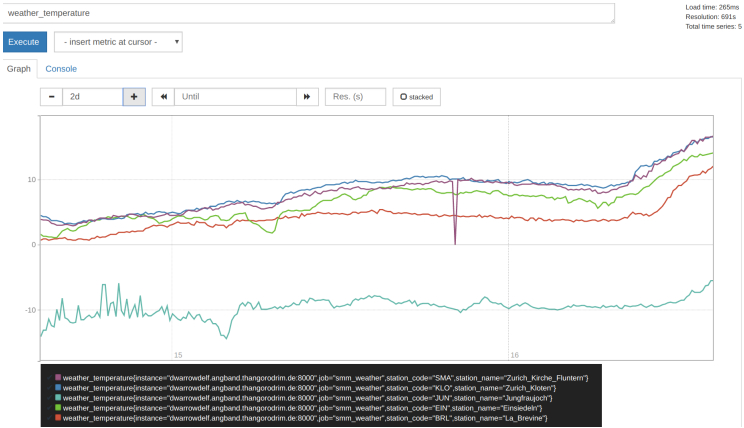
```
import prometheus_client as p_c

weather_temperature = p_c.Gauge('weather_temperature', 'temperature',
                                ['station_name', 'station_code'])

REQUEST_TIME = p_c.Summary('weather_request_processing_seconds',
                           'Time spent processing request')

@REQUEST_TIME.time()
def update_weather():
    weather = fetch_weather('SMA')
    weather_temperature.labels(station_name='Zurich_Kirche_Fluntern',
      station_code='SMA').set(weather['temperature'])

def main():
  p_c.start_http_server(8000)
  while True:
    update_weather()
    time.sleep(300)

if __name__ == '__main__':
  main()
```

munin_exporter

- Server nicht geeignet für `node_exporter` & Co (RAM, CPU, Architektur)
- aber: Munin (munin-node) verfügbar/installiert
- die Lösung: `munin_exporter` auf dem Munin-Server als Proxy
- https://github.com/pvdh/munin_exporter.git



Graphen - Prometheus (Graph Explorer)

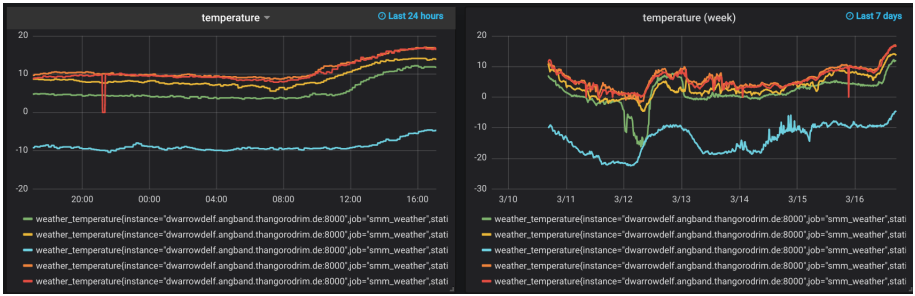


Graphen - Grafana

- Grafana - Open Source Graphing/Dashboarding Lösung
- unterstützt zahlreiche TSDB Datenquellen, u.a. Prometheus
- zahlreiche Möglichkeiten & Formate für Graphen & Dashboards
- kann auch Alerting
- von `Prometheus.io` empfohlen
- sehr einfach aufzusetzen
- Graphen & Dashboard sehr schnell und einfach definierbar

Grafana Dashboard

- basierend auf Prometheus Wetterdaten (Prometheus Beispiel von vorher)



Alertmanager Übersicht

- Alertmanager: alerting Komponente von Prometheus
- Alerts definiert in Prometheus, and Alertmanager geschickt

```
groups:
- name: example
  rules:
  - alert: HighErrorRate
    expr: job:request_latency_seconds:mean5m{job="myjob"} > 0.5
    for: 10m
    labels:
      severity: page
    annotations:
      summary: High request latency
```

Alertmanager Features

- Grouping: Zusammenfassen ähnlicher Alerts
- Deduplikation: identische Alerts parallel aktiv → ein Alert
- Inhibition: unterdrücke Alarm C wenn Alarm A aktiv
- Silencing: Alert für bestimmte Zeit unterdrücken
- viele Benachrichtigungsmethoden: PagerDuty, WeeChat, Slack, EMail, HipChat, OpsGenie, ...

Fragen?

Vielen Dank für Euer Interesse!