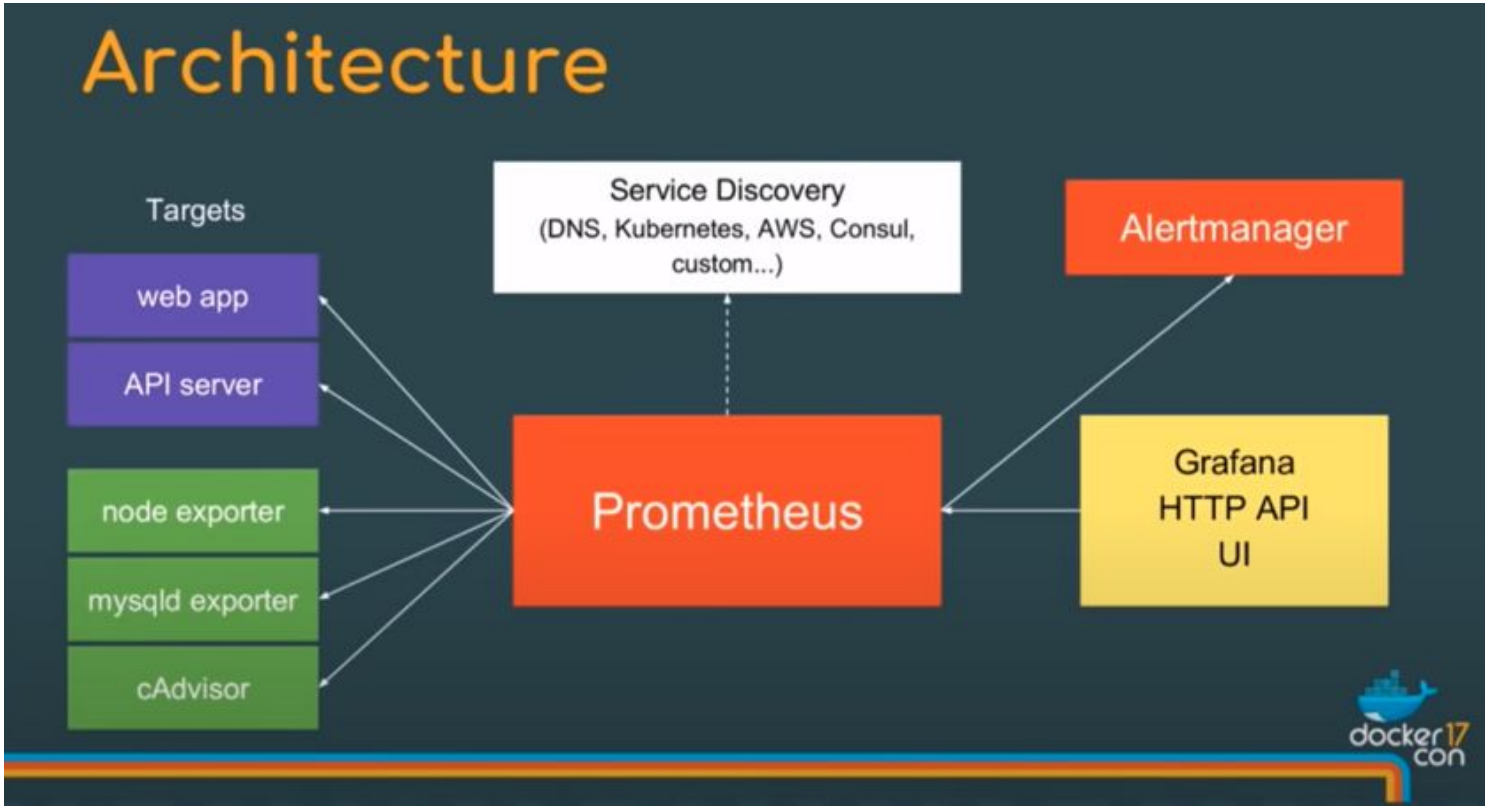
Prometheus

Next generation monitoring

Wat is prometheus?

- Open-source monitoring en ook alerting tool
- Ontworpen door Soundcloud, sinds 2012, beïnvloed door BorgMon (Google)
- Numerieke, multi dimensionaal, time series data
- Whitebox monitoring (ook “applicatie parameters”)
- Goed in een “volatile” omgeving (bijvoorbeeld een cloud omgeving als kubernetes)
- Eigen query taal: PromQL
- Pull systeem
- 1 enkele (Go) binary


Architecture
Targets
web app
API server
node exporter
mysqld exporter
cAdvisor
Service Discovery
(DNS, Kubernetes, AWS, Consul, custom...)
Prometheus
Alertmanager
Grafana
HTTP API
UI
docker17 con

Traditioneel (blackbox) monitoring

- Checkscripts (fail of ok)
- Vooral servers (disk, cpu, netwerk), niet services
- Soms wel verdieping in service: maar dan *specifiek* (bv. check 404 codes)
- Of achteraf verdieping na alert vanuit een hoek (drukke cpu -> mysqld als oorzaak)
- Statische omgevingen

Whitebox monitoring

- Idee: Als je *alle* metrics in de gaten houdt merk je “vanzelf” wat er mis gaat
  - -> om kunnen gaan met zeer veel metrics
  - -> prometheus!
- prometheus is efficient: **single node** kan heel veel metrics tegelijk aan, geen externe storage nodig (“by design”), kan overigens wel.
- multi dimensionale data -> detail!
- service discovery (dynamische omgevingen)
- eigen query language (PromQL)

De data

"**identifier**" -> [(t0, v0),(t1,v1), …]

(t*n* = int64 (milliseconde precisie), v*n* = float64)

Voorbeeld Graphite / StatsD identifiers:

- nginx.ip-1-2-3-4-80:home.200.http_requests_total
- nginx.ip-2-3-4-5-80:home.200.http_requests_total
- …

Voorbeeld Prometheus **identifier** : metric name + label based key/value pairs

- http_requests_total{job="nginx",instance="1.2.3.4:80",path="/home",status="200"}

  => multidimensionale time series data

Query:

.nginx..*.200.*.http_requests_total **vs** http_requests_total{job="nginx",status="200"}

PromQL 1/2

- nieuw
- goed met time series berekeningen
- niet SQL (maar functioneel)

Voorbeeld1: geef mij alle partities groter dan 100GB en die niet ”/” zijn:

node_filesystem_bytes_total{mountpoint!=”/”} / 1e9 > 100

{device=”sda4”, mountpoint=”/var”, instance=”10.0.0.3”} 203.4

{device=”sda3”, mountpoint=”/home”, instance=”10.0.0.4”} 102.6

PromQL 2/2

Voorbeeld2: geef mij de ***ratio*** (verhouding) van status 500 requests t.o.v. alle requests:

```
sum by(path) (rate(http_requests_total{status=”500”}[5m])) /
sum by(path) (rate(http_requests_total[5m]))
```

Geeft:

{path=”/status”}	0.0023
{path=”/”}	0.0016
{path=”/api”}	0.0384

PromQL moet je leren (maar is het waard)!

Waar komen de metrics vandaan?

Van “exporters”:

De exporters presenteren de voor prometheus geschikt gemaakt data op een http endpoint. De prometheus server “scraped” die endpoints.

Voor veel soorten services en servers is er wel een exporter:

Bijvoorveeld voor Linux hosts: node_exporter (package prometheus-node-exporter)

- exports cpu usage, disk-usage, network traffic, memory usage,
- maar ook “eigen data” in textfiles

Maar er zijn er veel meer. Bv. exporters die metrics van services (bv. mysqld, nginx, …) converteren naar “prometheus data”

Overzicht exporters op https://prometheus.io/docs/instrumenting/exporters/

Of je schrijft zelf een exporter

Bijvoorbeeld van json (veelal output van api's) -> prometheus format

{“temp”:[

{“Nijmegen”:12.6}

{“Arnhem”:12,.3}

…

]}

temp{“city”=”Nijmegen”} 12.6

temp{“city”=”Arnhem”} 12.3

...

```
jsondata = json.load(f)
for i in range(0, len(jsondata[data])):
  for key, value in jsondata[data][i].items():
    -> “prometheus format”: “identifier met key” value
```

Leuk metrics, maar hoe maak ik ze inzichtelijk?

Grafana!

De tool om time series data inzichtelijk weer te geven

exporters ← Prometheus ← Grafana dashboards

Grafana kent vele datasources waaronder Prometheus (maar ook MySQL, EleasticSearch, InfluxDb, …, …. ,.... ,....)

Grafana kent PromQL!