How many dashboards does your Grafana have?

Martin Polednik
Head of DevOps @ ThreatMark
Me

• Former Software Engineer (Infrastructure)
  • oVirt (RHEV), KubeVirt

• Enjoying the business / development / ops combination of DevOps (or SRE)
  • (not only) maintaining monitoring infrastructure @ ThreatMark

• Automation & Data Geek
Why Grafana

- Prometheus is awesome
- Grafana is good fit for Prometheus
- Fits cloud native landscape
- Beautiful! (more on this later)
- But... easy to mess up
Context Matters
${{startup}}
${startup}$

Weekend (obviously a lie)
Bistro in tech hub

![Graph showing lunch attendance]
Context Matters

• Companies prioritize different metrics
  • Being able to function?
  • Crazy SLAs?
  • Optimizing for cost?
• BI
• Focus on customer and business needs
Relevant Data != Nice Graphs
we're doing great!
totally didn't crash and violate SLAs
Relevant Data != Nice Graphs

• Grafana is terrible (you might hear this few more times)
  • Because it looks beautiful
  • It makes irrelevant metrics beautiful
• Track the scary stuff
Question Your Data
100 mbps line pushing 715 mbps
15k vs 4k IOPS

IOPS (read + write)

Disk IOPS Completed
Question Your Data

- When monitoring a metric, question the correctness
- Have someone review the data
- Care for double (or higher multiple) counting
- Physics can be handy!
CPU Cores: 40
Total RAM: 125.64 GiB
Total SWAP: 0 B
Total RootFS: N/A
System Load (1m avg): 0.68
Uptime: 19.1 hours

CPU Basic
Memory Basic
Network Traffic Basic
Disk Space Used Basic
Disk IOPS
I/O Usage Times
Clutter

- Grafana is terrible *(told you)*
  - It makes adding graphs and dashboards too easy!

- The struggle...
  - Seeing absolutely ridiculous amount of data
  - Not being able to analyze the system

- Can we fix that?
Clutter

• Think about dashboard space as limited resource
• Group correlated graphs
• Spacing
• Positioning
• Many graphs = slow dashboard (JS issue)
  • Pronounced with shared timeline selector
Overview at a Glance

- "Think about dashboard space as limited resource"

- We need a way to
  - Quickly determine the state of a system
  - Dig deeper / root cause analysis of an issue
Overview at a Glance

- Further analysis can be done via traces and logs
- What about multiple customers/deployments?
Overview at a Glance

- What about multiple customers/deployments?
- Grafana templating to the rescue!
- Maybe even repeating graphs?
production dashboard, showing only non-confidential stuff :)

*
Don't Leave a Mess

- Experiment with your dashboards!
- Try visualizing data in a new, better way
- Do crazy things
  - ... but clean up your Grafana :(
Don't Leave a Mess

• See the tags?
• Organize the dashboards, namespace them
• Have a naming + tagging policies in place
• Provision mission critical dashboards
Version Control

- Dashboard is yaml file, easy to store in git
- But reviewing changes hurts
  - Try moving one graph to different position...
bars = false

"gridPos": {
  "h": 5,
  "w": 12,
}

"x": 0,
"y": 21

"id": 4,
"id": 8,

"isNew": true,
"legend": {
  "alignAsTable": false,
Version Control

- Keep your dashboards in Python!
- Work around Grafana limitations
- Auto-generate graph IDs
- Code reviews are breeze (kinda)
- https://github.com/weaveworks/grafanalyzer
title='aborted clients/connections',
dataSource='Prometheus',
targets=[
    Target(
        expr="rate(mysql_global_status_aborted_connections{{filters},instance=~'\$dbnode'}[1m])",
        legendFormat="{{ instance }} aborted connections",
        refId="A",
    ),
    Target(
        expr="rate(mysql_global_status_aborted_clients{{filters},instance=~'\$dbnode'}[1m])",
        legendFormat="{{ instance }} aborted connections",
        refId="B",
    ),
],
fill=0,
legend=Legend(show=False),
yAxes=[
    YAxis(format=SHORT_FORMAT),
    YAxis(format=SHORT_FORMAT),
],
xAxis=XAxis(mode="time")
],

Version Control

- Doesn't fit nicely with pre-made dashboards
- Solution?
  - Re-implement them in Python (eww)
  - Consider them vendored (store as yaml)
- Other ideas?
Application Monitoring
Application Monitoring

- Gather stunning amount of data about app by observing infrastructure serving it
- Grafana also supports SQL data sources
MySQL Connection

- **Host**: localhost:3306
- **Database**: database name
- **User**: user
- **Password**: password

**User Permission**

The database user should only be granted SELECT permissions on the specified database & tables you want to query. Grafana does not validate that queries are safe so queries can contain any SQL statement. For example, statements like `USE otherdb;` and `DROP TABLE user;` would be executed. To protect against this we **Highly** recommend you create a specific MySQL user with restricted permissions. Checkout the MySQL Data Source Docs for more information.
SELECT MIN(UNIX_TIMESTAMP(creation_time)) as time_sec,
  count(case when master_score < 300 then 1 else null end)/count(*) * 100 as "pass (<300)",
  count(case when master_score > 300 and master_score < 900 then 1 else null end)/count(*) * 100 as "step_up (300<x<900)"
FROM table
WHERE $timeFilter(creation_time)
GROUP BY $timeGroup(creation_time, '5m')
ORDER BY creation_time ASC
Alerting

- Grafana is terrible (...)
- Alerting is extremely nice visually
- Not sophisticated enough to properly alert
Alerting

- Better handled by data source, e.g. Prometheus + Alertmanager

- Alert on symptoms
  - Anomaly detection works well
    - $\text{avg} + 2 \times \text{stddev}$
    - predict_linear (80% disk capacity vs disk capacity fill up rate)
  - Recording rules
predict_linear + constant
Summary

• Grafana is great
  • Maybe even good

• Don't overwhelm yourself with metrics just because it's easy

• Have fewer dashboards showing relevant metrics

• Keep overview and detail dashboards separated
Summary

• Focus on customer and business needs

• Monitoring new metric may be cheap

• But further implications could be expensive
Thank you!

Q/A