Monitoring with Grafana

Marcus Olsson Developer Advocate

While you wait ...





- \$ # Install Docker and Docker Engine
- \$ git clone https://github.com/grafana/tutorial-environment
- \$ cd tutorial-environment
- \$ docker-compose up -d

Who am I?

- Developer Advocate at Grafana
 - Developer Experience for plugin authors
 - Educating and supporting the Grafana community
- Software Developer since 2011
 - Frontend, backend, leftend and rightend
 - Product development, data engineering, SRE





What we'll cover today

- What is monitoring?
- Types of data
- Exploring data
- Building dashboards
- Annotations, variables, and links
- Creating alerts
- Best practices for dashboard design



. . .

What do you hope to learn today? Grafana Labs

What is monitoring?

- Graphs on a TV on the walls of Engineering?
- Waking up to ~20 emails about CPU usage?
- Pinging services to make sure they're alive?





"Monitoring tells you whether the system works. Observability lets you ask why it's not working."

- Baron Schwartz



Why do we monitor?

- Make sure system works as intended
- Get insights in how the system is being used
- Fix problems before customers tell us about them
- Outage cost is more expensive than investing in quality
- Make knowledge available to the rest of the organization
- Make decisions on data rather the gut feeling



000

Monitoring is changing

- Infrastructure is becoming more dynamic
 - Servers are becoming cattle, rather that pets
- Teams are deploying changes several times a day
 - Less up-front testing
- Monitoring tools need to keep up





Why do you want to start monitoring?

If you already do, why?











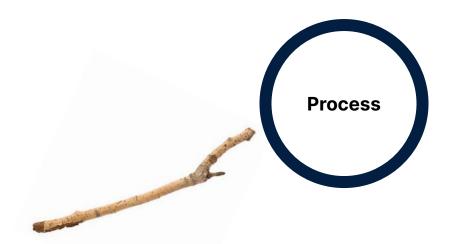
How do I know my process is working?





You can poke it with a stick ...

(black box monitoring)





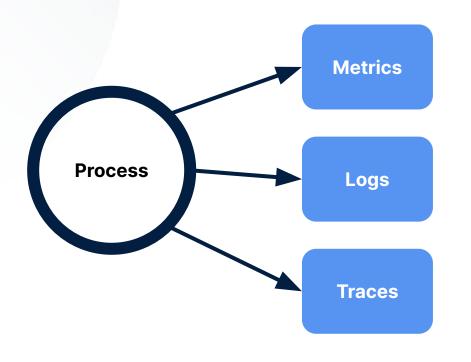
... or have it tell you how it's doing

(white box monitoring)





Extracting data from a running process



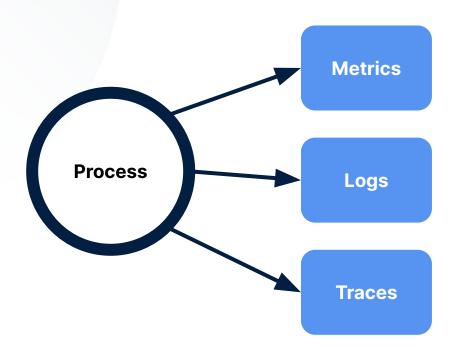
Numeric data for easy aggregation

Textual data for understanding what happened

Execution paths for individual requests



Extracting data from a running process



Pros: Lets you see trends and patterns. Less

data, faster queries

Cons: Needs to be configured up-front

Pros: Lets you see exactly what the process was

doing at a given time

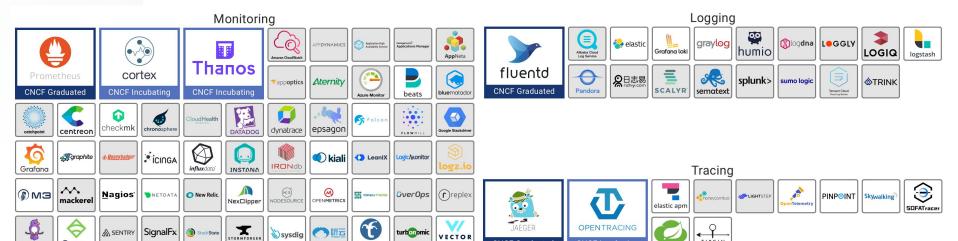
Cons: Costly to store and to query

Pros: Lets you see the path a request took through the system, and where time was spent

Cons: Costly to store and query







turbonomic

VECTOR By Timber.io

CNCF Graduated

OPENTRACING

CNCF Incubating

https://landscape.cncf.io

SignalFx

weave scope

⋒ SENTRY

weave cloud

Sensu

WAVEFRONT

ROOKOUT

VICTORIA METRICS

STORMFORGER

ZABBIX

WhaTap

sysdig

What data are you monitoring on?

How do you get that data?









Grafana

An open composable observability platform



An open composable observability platform

- Aims to integrate rather than replace
- Ships with integrations for popular projects
- Offers a plugin platform for integrating with other projects
 - Browse the plugins written by the Grafana community at grafana.com/plugins



Choose your own stack





Our application

- \$ # Install Docker and Docker Engine
- \$ git clone https://github.com/grafana/tutorial-environment
- \$ cd tutorial-environment
- \$ docker-compose up -d

Browse to http://localhost:8081



Demo

A tour of Grafana





Data sources

- Data sources bring your data into Grafana
- Data source options configures how to connect to a data source
- The Query editor configures what data you want to display



Panels

- Panels consist of a query and a visualization
- Display options configure the currently selected visualization type
 - For example, whether you want to show a table header or not
- Field options configure how the data is displayed
 - For example, if the data ranges from 0–1, you want to display it as percentage (12%) regardless of the visualization type



Dashboards

- Dashboards consist of multiple panels
- All panels in a dashboard share time range
 - Zooming into one changes the time range for all panels



Metrics



Metrics

- A quantifiable, single type of data that's changing over time
- For example:
 - Temperature
 - Churn rate
 - Logged-in users





Metric type: Counter

- A counter starts at zero and is only incremented
- The rate of change is often more useful, and can be calculated using the counter value
 - Example: Requests per second for the last 24 hours







Metric type: Gauge

- A gauge is a snapshot in time of the current state
- Single numerical value that can go up and down
- For example:
 - 21°C at 13:00
 - 25°C at 14:00
 - 23°C at 15:00





Time series



Time series

- Measuring a metric over time results in one or more time series.
- A sequence of measurements ordered in time.
- Each measurement consists of a point in time, and a value.
- Usually measurements are taken at regular intervals, such as every 30 second, hourly, or every quarter
 - Otherwise, we're probably looking at event data, more fitting as logs

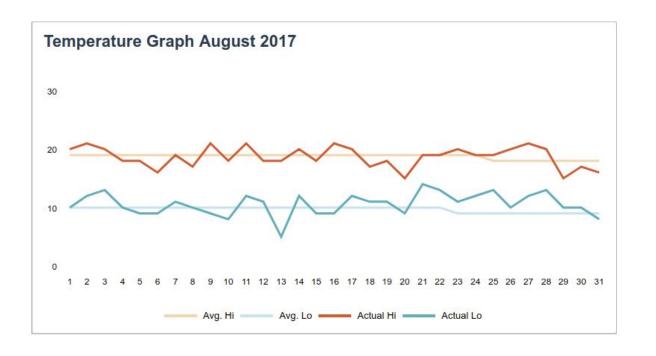


Raw time series data

Series	Time	Value
apps	2019-09-13 2:00:00	7651
apps	2019-09-14 2:00:00	37523
apps	2019-09-15 2:00:00	37668
datasources	2019-09-13 2:00:00	63021
datasources	2019-09-14 2:00:00	284467
datasources	2019-09-15 2:00:00	286193
panels	2019-09-13 2:00:00	62368
panels	2019-09-14 2:00:00	282907
panels	2019-09-15 2:00:00	284612
all	2019-09-13 2:00:00	66566
all	2019-09-14 2:00:00	302986
all	2019-09-15 2:00:00	304785

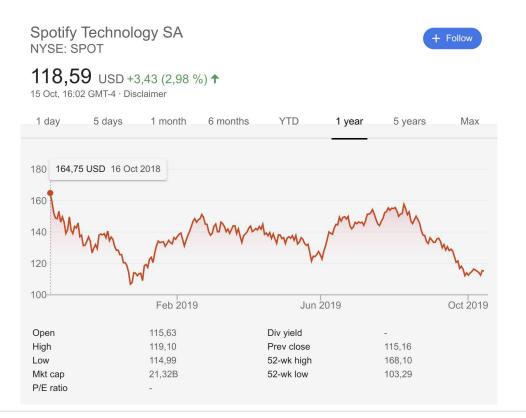


Example: Temperature





Example: Spotify share price





The cardinality problem

- Each time series has a unique name
 - o stats.sweden.stockholm.temperature
 - temperature{country="sweden", city="stockholm"}
- Changing part of the identifier means creating a new time series
- Making things like user id part of the name could cause an explosion of time series, affecting performance.
 - Cardinality problem



Time series databases (TSDB)

- Not relational data (no joins)
- Specialized and optimized
- More efficient at storing time series data
 - Prometheus uses around 1-2 bytes per measurement on average
- Better at querying time series data











Metrics using Prometheus





Exercise: Create a dashboard to display Prometheus metrics

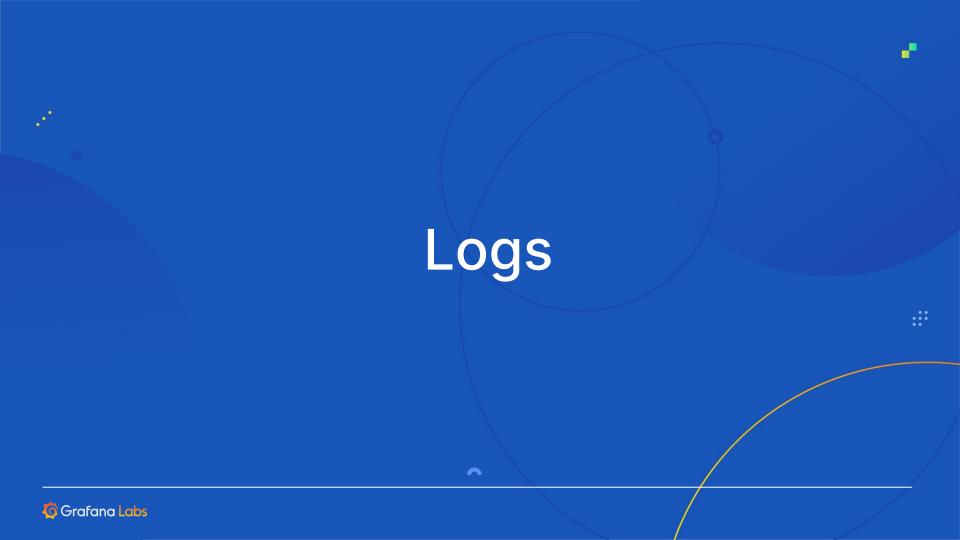
Data source URL: http://prometheus:9090

```
Useful queries:
```

```
rate(tns_request_duration_seconds_count[1m])
histogram_quantile(0.95, rate(tns_request_duration_seconds_bucket[1m]))
```







Logs

- Shows you what's happening inside the application
- Append-only text
- Typically large volumes



Logs

- Usually for applications, not so much for resources, like CPU or disk utilization.
- Be mindful about what you log
 - Avoid excessive logging
 - Makes it difficult to find the logs that matters
 - Storage may be cheap, but your time is not





Unstructured logs

User 752 bought 3 tickets

Structured logs

```
{msg: "bought tickets", user: "752", count: 3}
```







Logs using Loki





Exercise: Add a Logs panel to display Loki logs in the dashboard

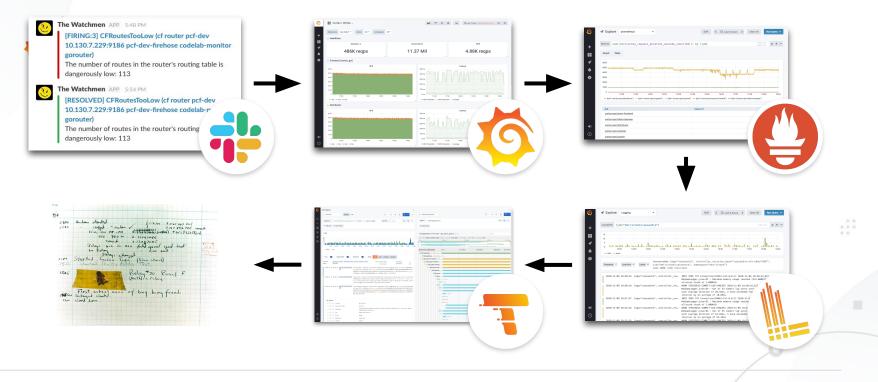
Data source URL: http://loki:3100

```
Useful queries:
{filename="/var/log/tns-app.log"}
{filename="/var/log/tns-app.log"} |= "error"
```





Troubleshoot without leaving Grafana





Annotations



Annotations

- Add context to a visualization by annotating it
- Annotate events or entire regions
- Query annotations from data sources





Annotations





Exercise: Annotate the graph panel with errors from Loki

Create an annotation query

```
{filename="/var/log/tns-app.log"} |= "error"
```





Dynamic dashboards



Dynamic dashboards

- In practice, most services will be monitored in similar ways
- Avoid duplicated dashboards by using variables
- Lets you create templated queries and panels





Variables





Exercise: Create a variable for selecting status code

- 1. Create a **Query** variable called status_code
- 2. Select Prometheus as the data source
- 3. Enter the following query
 label_values(tns_request_duration_seconds_count, status_code)
- 4. Click Add and save the dashboard
- 5. Change the panel query to

```
rate(tns_request_duration_seconds_count{status_code="${status_code}"}[1m])
```





Repeated panels

 Avoid duplicated panels by repeating them for every value in a variable



Repeated panels





Exercise: Repeat panel for multiple status codes

- Update the status_code variable to be Multi-value
- 2. Click **Update**
- 3. Enter Edit mode for the panel
- 4. In the Panel editor, select **Repeat options**
- 5. Select your variable and click **Apply**





Dashboard design



Keep your user in mind

- Consider your target audience
- How much details do they need?
- Questions to ask yourself:
 - What issue let them to open this dashboard?
 - What questions should this dashboard answer?
 - When would the user want this information?





Monitor with intent

- Start small
 - Resist the urge to fill up your dashboard at first
 - Make sure you understand each panel before adding another
- Keep it simple
 - Every detail you add, adds to the complexity of the dashboard
 - Avoid putting to much information on a single dashboard
- Sort by importance
 - Not all panels are equally important



0 0

Will you understand the dashboard in the middle of the night during an outage?









Links



Links

- Prefer smaller dashboards with clear focus and link them together
- Grafana lets you link dashboards together using three types of links: dashboard links, panel links, and data links.







Links

- Dashboard links
 - Link to other dashboards or external websites
- Panel links
 - Same as dashboard link but in the context of a panel
- Data links
 - Use values from your query result in your link





Comparing link types





Alerting



What makes a good alert?

- Alerts are not warnings: They are calls for help
 - Only alert on real problems
- Avoid alert fatigue
- They're simple
- Requires a human
- Includes actionable information



. . .

Alert on symptoms, not causes

- There are thousands of reasons for a website to not respond
- Alerting on symptoms always catch the problem
- Alerting on causes might catch the problem





The life of an alert in Grafana



Alerts can be at any time Paused



Create an alert





Exercise: Create an alert

- Create a request box at https://rbox.app
- 2. Create a webhook notification channel with the URL to your request box
- 3. Enter panel edit mode and click the Alerts tab under the graph
- 4. Create an alert that evaluates every 5s for 5s
- 5. Select your webhook as notification channel
- 6. Make the alert trigger and watch the request box





What would you alert on? Grafana Labs

What should you monitor?



USE

- Created by <u>Brendan Gregg</u>
- For every **resource**:
 - Utilization
 - Saturation
 - Errors





RED

- Created by Tom Wilkie
- For every **service**:
 - Rate
 - Errors
 - Duration





Golden signals

- Created at Google
- For every system:
 - Latency
 - Traffic
 - Errors
 - Saturation



What would you monitor? Grafana Labs

Dashboard design



Dashboards

- Be consistent
 - Use same colors and styles for the same concepts across dashboards
 - During an outage, you're not always thinking straight. Make it easy to understand.





Panels

- Will I still understand this panel a month from now?
- Use units
- Label your axes
- Give your panels proper titles
- Consider shared crosshairs
- Keep metrics of different scales in separate panels
 - Reads can be order of magnitudes larger than writes
- Combine aggregates for better insights

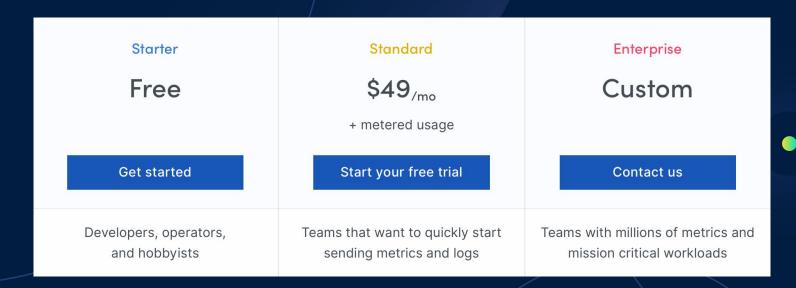


Colors

- Avoid overusing colors
- Colors have meaning
 - Traffic lights
 - Green indicates something good, such as free disk space
 - Red indicates something bad, grabs your attention, and should be used for critical things
- Consider color blindness



Grafana Cloud



https://grafana.com/signup











Monitoring doesn't fix your problems.

It shows them to you.









Thank you!

marcus.olsson@grafana.com

@marcusolsson



Bonus: Plugins



Plugins

- Three types of plugins
 - Data sources
 - Panels
 - Apps
- Browse published plugins on https://grafana.com/plugins



Demo

Install a plugin





Exercise: Install a plugin

Find a plugin you find interesting on https://grafana.com/plugins

Install in the tutorial environment:

docker-compose exec grafana /bin/bash

grafana-cli plugins install <plugin id>



