OPEN SOURCE MONITORING

Using TIG (Telegraf + InfluxDB + Grafana)

MONITORING BACKGROUND

These are the are 3 'legs' of the Monitoring Stool:

- Metrics
- Traces
- Logs

Each has its own history, toolings, and uses

Some platforms bring all of these together (known as *Unified Observability*)

It is possible to derive some from the other (e.g. determining metrics from Logs or Traces, searching Traces like Logs)

Logs

We all know what logs are - the granddaddy of the group

Generally a time-stamped 'message' written to a local file from an application, script or process

There can be varying structure to the log line, and/or the message itself can be structured

Using Logs in *Monitoring* involves shipping, transforming, storing, indexing, and analyzing these log messages

A well-known open-source platform for this is: ELK

- Elasticsearch+Logstash+Kibana from elastic.co

Traces / Application Tracing

Rise in popularity of *tracing* is tied to the use of services and microservices in building applications

Transactions *through* an application often touch multiple services

Tracing is a strategy to instrument all the services with the ability to emit *spans* for each of these transaction components in and out the service, and collectively multiple spans make up an application *trace*

Best known tooling in this area is Jaeger, but most practitioners are moving towards an accepted standard known as OTel or OpenTelemetry.

Metrics

We have all interacted with metrics, in one fashion or another.

For example - `top` - emits *metrics* - "numbers that have names"

That is what a metric is - a number. *"...that by which anything is measured."*

To organize the metric, we associate a name with it. That results in the generally accepted format of {"name" : "load_5", "value" : 0.223} or even ("load_5",0.223)

Imagine a system that would just send ("load_5",0.223) over UDP.

That is the essence of 'statsd' developed by an Etsy engineer in 2011.

Hey! No timestamp! That seems crazy.

Metrics

- in the original statsd, the timestamp was added by the receiver (keeping the concept of the sender CRAZY simple)
- the receiver could even do aggregations - sending the aggregations 'upstream' every minute and sender could send just whenever (flexible)
- even early on, emitting and receiving metrics was robust and open to innovation
- in general though, a standard 'metric' became something like:

  env=prod,region=us-east,name=sys-01 cpu_load_5=0.2445 1681179840

  0 or more *tags*, followed by a metric name, value, and lastly a timestamp.

Metrics

what the heck is a *tag*?

- like a category, or a grouping - something that doesn't really change.
- the 'by' in *slice and dice by*…
- it's how you'd answer "what is the average cpu_load_5 for all my prod servers?"

TIG (Telegraf - InfluxDB - Grafana)

This is an open-source 'stack' that executes all the important roles in a full monitoring system:

- get lots of great metrics from all sorts of systems
- store them in a tool that is designed for time-series data and has built-in query functions for stats and grouping
- display these metrics in useful and creative ways, using those queries and basic graphing functions
- alert on certain specified conditions of one or more metrics over time

TIG

Telegraf:

- compiled golang agent with 'plug-in' pipelined architecture for retrieving, transforming, filtering, and output of metrics

Influxdb:

- time-series database with advanced query and analysis functions

Grafana:

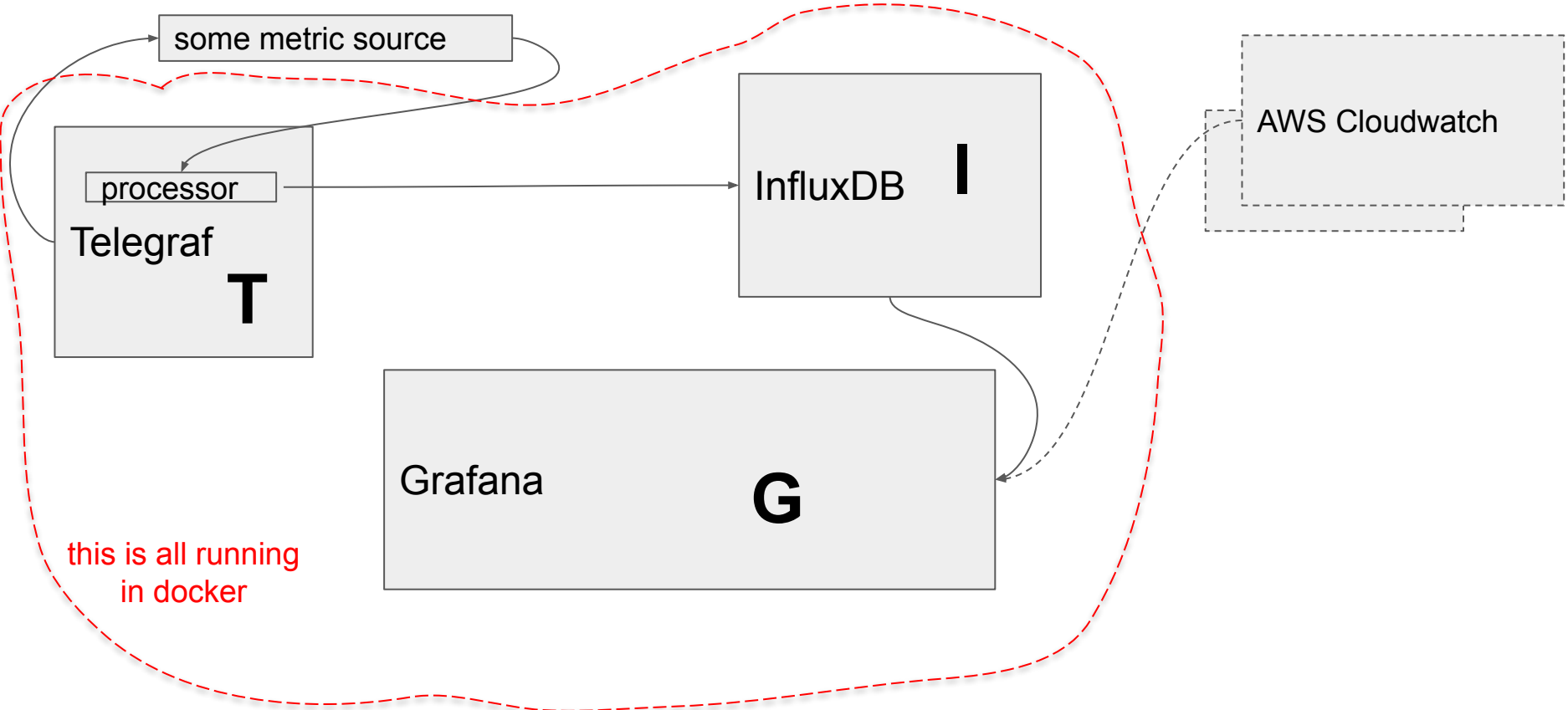
- dashboard and charting tool that also has alerting capabilities

TIG *stack*

When you get all those working *together*

inspired by this YouTube where it is all built using *docker-compose*

TIG architecture



docker-compose.yml

```
version: "3"

services:
 influxdb:
   image: influxdb:2.1.1
   volumes:
     - influxdb-storage:/var/lib/influxdb2:rw
   env_file:
     - .env
   entrypoint: ["./entrypoint.sh"]
   restart: on-failure:10
   ports:
     - ${DOCKER_INFLUXDB_INIT_PORT}:8086

 telegraf:
   image: telegraf:1.25.3
   volumes:
     - ${TELEGRAF_CFG_PATH}:/etc/telegraf/telegraf.conf:rw
#    networks:
#      mynetwork:
#        ipv4_address: 192.168.86.11
   env_file:
     - .env
   depends_on:
     - influxdb

 grafana:
   image: grafana/grafana-oss:8.4.3
   volumes:
     - grafana-storage:/var/lib/grafana:rw
   depends_on:
     - influxdb
   ports:
     - ${GRAFANA_PORT}:3000

volumes:
 grafana-storage:
 influxdb-storage:

# networks:
#   mynetwork:
#     driver: bridge
#    ipam:
#      config:
#        - subnet: 192.168.86.0/24
```

.env file

```
DOCKER_INFLUXDB_INIT_MODE=setup

## Environment variables used during the setup and operation
of the stack
#

# Primary InfluxDB admin/superuser credentials
#
DOCKER_INFLUXDB_INIT_USERNAME=admin
DOCKER_INFLUXDB_INIT_PASSWORD=password
DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=862010a1b8040e5f2eedc936543
58d2b24d42cdf1d83d571e2490a4556030a08

# Primary InfluxDB organization & bucket definitions
#
DOCKER_INFLUXDB_INIT_ORG=kyoss
DOCKER_INFLUXDB_INIT_BUCKET=telegraf

# Primary InfluxDB bucket retention period
#
# NOTE: Valid units are nanoseconds (ns), microseconds(us),
milliseconds (ms)
# seconds (s), minutes (m), hours (h), days (d), and weeks
(w).
DOCKER_INFLUXDB_INIT_RETENTION=4d

# InfluxDB port & hostname definitions
#
DOCKER_INFLUXDB_INIT_PORT=8086
DOCKER_INFLUXDB_INIT_HOST=influxdb

# Telegraf configuration file
#
# Will be mounted to container and used as telegraf
configuration
TELEGRAF_CFG_PATH=./telegraf/telegraf.conf

# Grafana port definition
GRAFANA_PORT=3000
```

Telegraf

- huge library of plug-ins
- input plug-ins
-

output plug-ins

this one is configured to send all metrics to influxdb

```
997
998   # # Configuration for sending metrics to InfluxDB
999   [[outputs.influxdb_v2]]
1000    ## The URLs of the InfluxDB cluster nodes.
1001    ##
1002    ## Multiple URLs can be specified for a single cluster, only ONE of the
1003    ## urls will be written to each interval.
1004    ##   ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
1005    urls = ["http://${DOCKER_INFLUXDB_INIT_HOST}:8086"]
1006
1007    ## Token for authentication.
1008    token = "$DOCKER_INFLUXDB_INIT_ADMIN_TOKEN"
1009
1010    ## Organization is the name of the organization you wish to write to; must exist.
1011    organization = "$DOCKER_INFLUXDB_INIT_ORG"
1012
1013    ## Destination bucket to write into.
1014    bucket = "$DOCKER_INFLUXDB_INIT_BUCKET"
1015
```