



Infrastructure Monitoring Basics with Telegraf, Grafana and InfluxDB

Anais Dotis-Georgiou
Developer Advocate, InfluxData



Anais Dotis-Georgiou

Developer Advocate



LinkedIn



At a glance

FOUNDED	<p>2013</p> <p>San Francisco HQ, 176 FTE's, 61+ in technical functions</p>
FOCUS	<p>Where developers build real-time applications for IoT, Analytics and Cloud native services</p>
DIFFERENCE	<p>One platform; one API across Multiple Clouds and On-Prem</p> <p>Ingest, query, story using common tools regardless of architecture</p>
OSS FOUNDATION	<p>1300+ Customers and 754,000 daily active OSS deployments;</p> <p>Google , Cisco, SAP, Comcast, Tesla, Siemens, PTC, Honeywell, JP Morgan Chase</p>
BUSINESS MODEL	<p>PLG Driven Usage and Subscription Model</p> <p>Pay for what you use; Pay how you want. Credit card, cloud provider, annual contract</p>

Agenda

Monitoring vs Observability

Let's break down what each area is and how they relate and differ.

Let's solve that problem

We will deploy open source tools such as Telegraf, InfluxDB, Grafana, OpenTelemetry & more to solve the problem.

●
1

2
●

●
3

4
●

Let's look at a problem

Problems drive learning. Let's create a scenario with an observability problem to solve.

Next steps

Get your hands on the source code and get involved with our community.

Monitoring vs Observability

Monitoring vs Observability



Collects and analyzes **metrics, logs, and events** to track system performance. Uses predefined rules and thresholds to detect issues, generating alerts when breached, helping maintain system health. This can be applied to various types of infrastructure, including **physical, digital**.



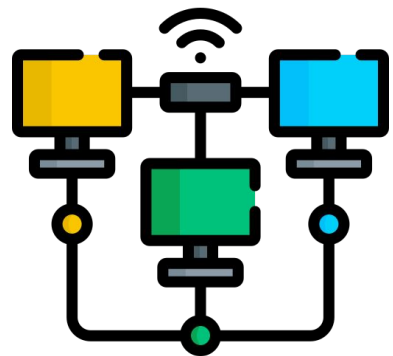
Instruments code and infrastructure to expose relevant data, enabling teams to **understand system behavior**. Correlates data from different sources to diagnose issues and **identify root causes**, providing insights for effective problem-solving. **Traces** are good example.

Monitoring + Observability Fields

Network Monitoring

Source: Routers, switches, and firewalls

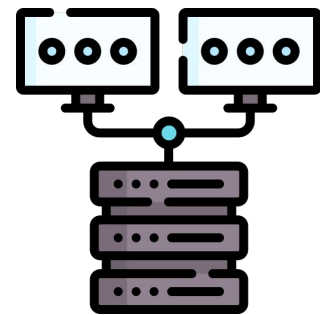
Monitor: Efficient data transmission, detect bottlenecks, status of devices



Server Monitoring

Source: CPU, memory, disk, processes

Monitor: CPU usage, memory consumption, disk space, active processes



Application Performance Monitoring

Source: Metrics, logs, and traces

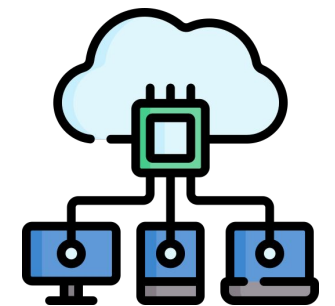
Monitor: Latency, code inefficiencies, errors



Cloud Infrastructure Monitoring

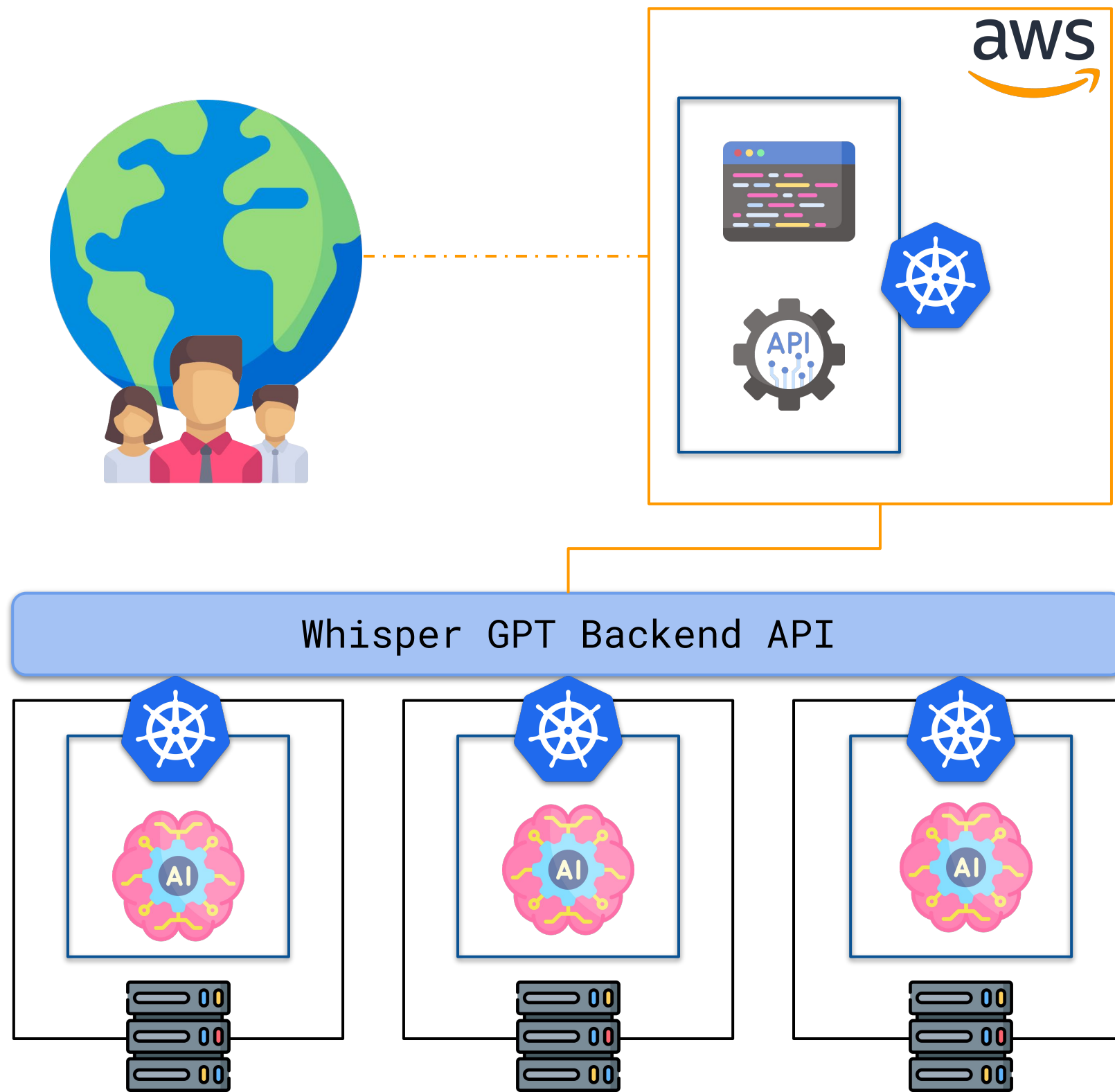
Source: Kubernetes, VM's microservices, services

Monitor: Uptime, cost, resource allocation



Let's look at a problem...

Whisper GPT



Product: Whisper GPT

Purpose: Natural language processing and machine learning techniques to provide users with highly accurate, context-aware, and personalized responses.

Problem: Unprecedented growth presents a few challenges, including potential bottlenecks, latency issues, and the need for seamless scalability to handle the influx of new users and requests.

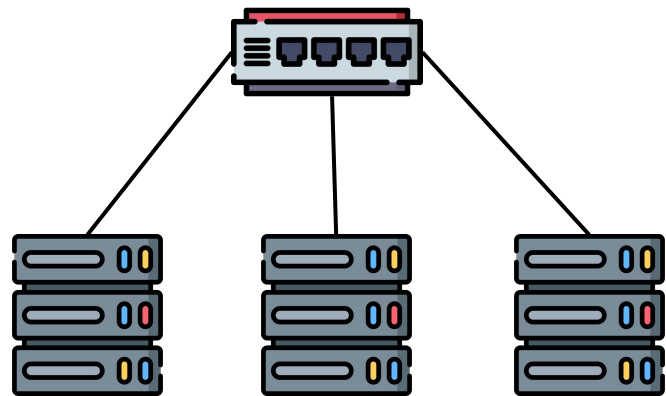
Question: How can the Whisper GPT team monitor and optimize their scaling solution's network, application, and cloud infrastructure to maintain optimal performance, reliability, and user experience?

Break it down

Network monitoring

Source: Routers, switches, and firewalls

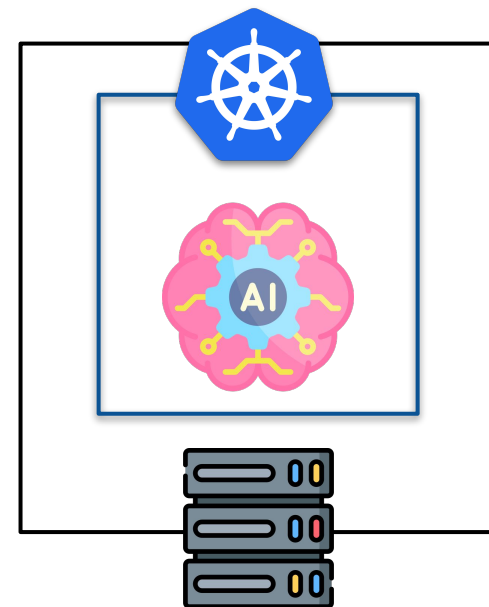
Monitor: Efficient data transmission, detect **bottlenecks, status of devices**



Server monitoring

Source: CPU, memory, disk, processes, **GPU**

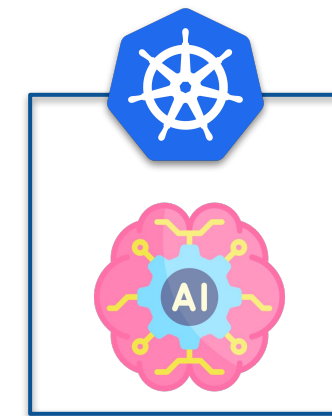
Monitor: **CPU & GPU** usage, **memory consumption**, disk space, active processes



Application performance monitoring

Source: Metrics, logs, and traces

Monitor: **Latency**, code inefficiencies, **errors**



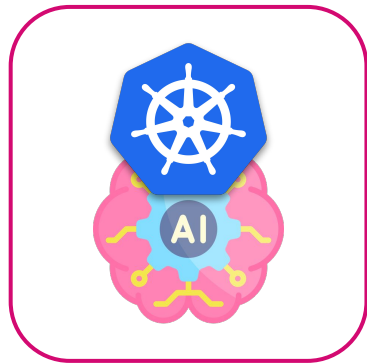
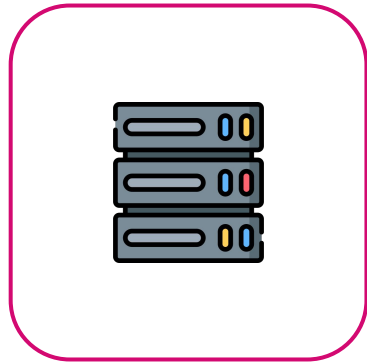
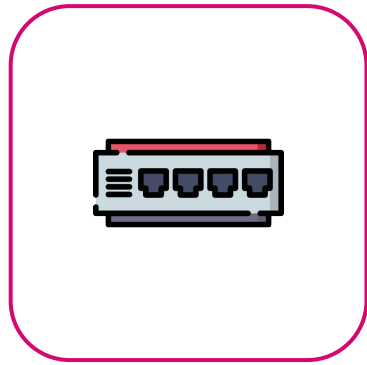
Cloud infrastructure monitoring

Source: **Kubernetes**, VM's microservices, **services**

Monitor: **Uptime**, **cost**, resource allocation



Let's solve that problem



Monitor



1

**Data
Collection**



2

**Data
Storage**



3

Data Action

Data Collection



Telegraf is our open source data collection agent for metrics and events.

With 300+ plugins for ingesting and outputting data, Telegraf is one of the most versatile ingest agents for time series data.

Input Plugins

activemq
aerospike
amqp_consumer
apache
apcupsd
aurora
azure_storage_queue
bcache
beanstalkd
bind
bond
burrow
cassandra
ceph
cgroup

chrony
cisco_telemetry_mdt
clickhouse
cloud_pubsub
cloud_pubsub_push
cloudwatch
contrack
consul
couchbase
couchdb
cpu
dcos
disk
diskio
disque

dmcache
dns_query
docker
docker_log
dovecot
ecs
elasticsearch
ethtool
eventhub_consumer
exec
execd
fail2ban
fibaro
file
filecount

filestat
fireboard
fluentd
github
gnmi
graylog
haproxy
hddtemp
http
http_listener_v2
http_response
httpjson
icinga2
infiniband
influxdb

Input Plugins

influxdb_listener
influxdb_v2_listener
intel_rdt
internal
interrupts
ipmi_sensor
ipset
iptables
ipvs
jenkins
jolokia
jolokia2
jti_openconfig_telemetry
kafka_consumer
kafka_consumer_legacy

kapacitor
kernel
kernel_vmstat
kibana
kinesis_consumer
kube_inventory
kubernetes
lanz
leofs
linux_sysctl_fs
logparser
logstash
lustre2
mailchimp
marklogic

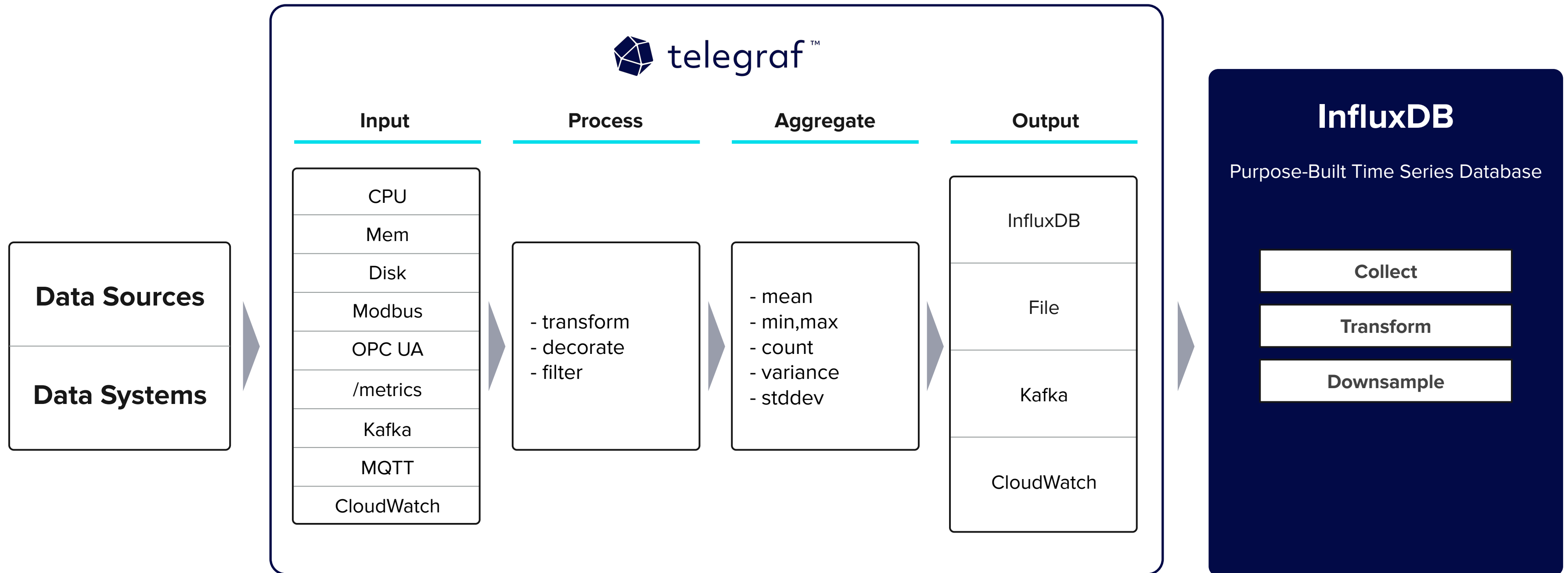
mcrouter
mem
memcached
mesos
minecraft
modbus
mongodb
monit
mqtt_consumer
multifile
mysql
nats
nats_consumer
neptune_apex
net

net_response
nginx
nginx_plus
nginx_plus_api
nginx_sts
nginx_upstream_check
nginx_vts
nsd
nsq
nsq_consumer
nstat
ntpq
nvidia_smi
opcua
openldap

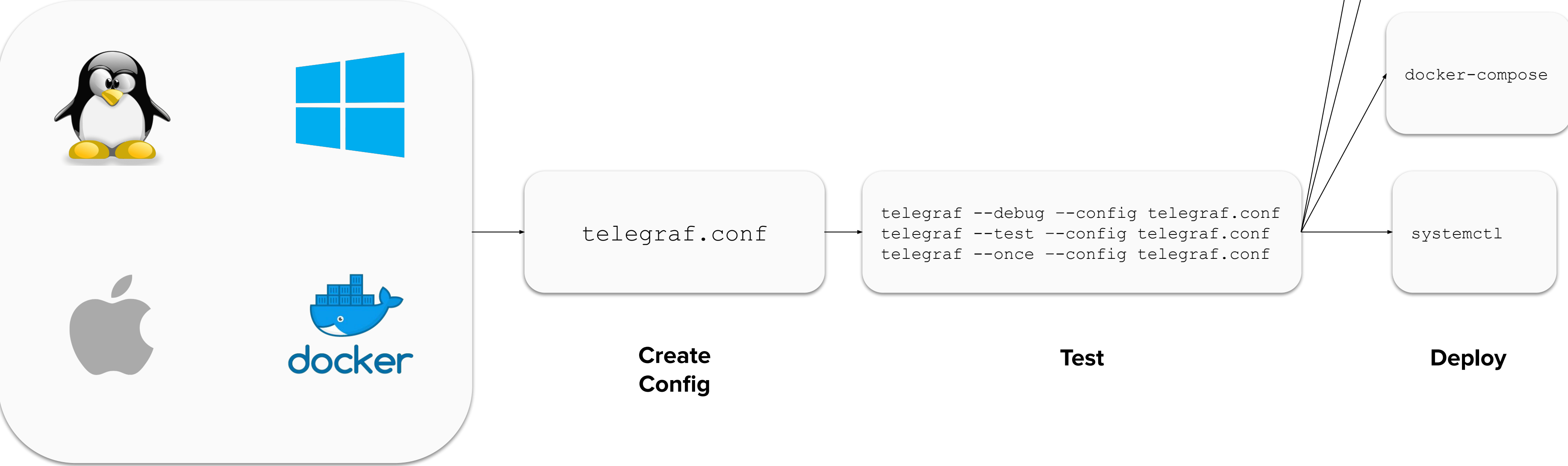
Input Plugins

opentelemetry	prometheus	snmp_legacy	tcp_listener	win_services
openntpd	proxmox	snmp_trap	teamspeak	wireguard
opensmtpd	puppetagent	socket_listener	temp	wireless
openweathermap	rabbitmq	solr	tengine	x509_cert
passenger	raindrops	sqlserver	tomcat	zfs
pf	ras	stackdriver	trig	zipkin
pgbouncer	redfish	statsd	twemproxy	zookeeper
phpfpm	redis	suricata	udp_listener	
ping	rethinkdb	swap	unbound	
postfix	riak	synproxy	uwsgi	
postgresql	salesforce	syslog	varnish	
postgresql_extensible	sensors	sysstat	vsphere	
powerdns	sflow	system	webhooks	
powerdns_recurser	smart	systemd_units	win_eventlog	
processes	snmp	tail	win_perf_counters	
procstat				

Telegraf Architecture



Telegraf Setup



Download & Install

telegraf.conf

Create Config

```
telegraf --debug --config telegraf.conf  
telegraf --test --config telegraf.conf  
telegraf --once --config telegraf.conf
```

Test

systemctl

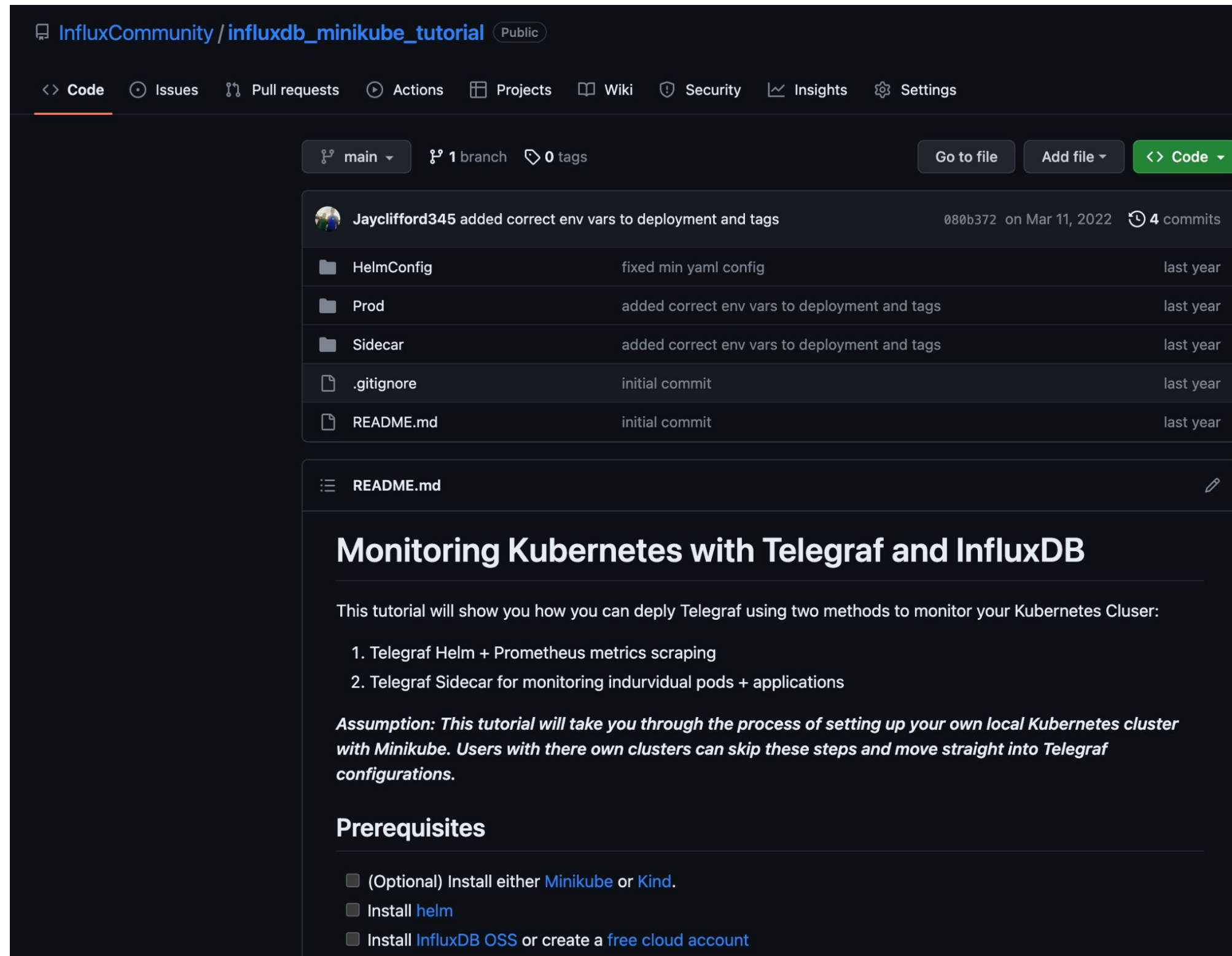
Deploy

Windows Service

kubernetes

docker-compose

Telegraf sidecar



The screenshot shows a GitHub repository page for 'InfluxCommunity/influxdb_minikube_tutorial'. The repository is public and has a 'main' branch with 1 branch and 0 tags. The commit history shows a commit by Jayclifford345 on Mar 11, 2022, with 4 commits. The commit message is 'added correct env vars to deployment and tags'. The files listed are HelmConfig, Prod, Sidecar, .gitignore, and README.md. The README.md file is open, showing the title 'Monitoring Kubernetes with Telegraf and InfluxDB'. The text in the README.md file reads: 'This tutorial will show you how you can deploy Telegraf using two methods to monitor your Kubernetes Cluster: 1. Telegraf Helm + Prometheus metrics scraping 2. Telegraf Sidecar for monitoring individual pods + applications. Assumption: This tutorial will take you through the process of setting up your own local Kubernetes cluster with Minikube. Users with their own clusters can skip these steps and move straight into Telegraf configurations. Prerequisites: (Optional) Install either Minikube or Kind. Install helm. Install InfluxDB OSS or create a free cloud account.'



https://github.com/InfluxCommunity/influxdb_minikube_tutorial

Telegraf Config

```
[global_tags]
# dc = "us-east-1" # will tag all metrics with dc=us-east-1
# rack = "1a"
# user = "$USER"
```

```
[agent]
interval = "10s"
round_interval = true
```

```
metric_batch_size = 1000
metric_buffer_limit = 10000
collection_jitter = "0s"
```

```
flush_interval = "10s"
flush_jitter = "0s"
precision = ""
```

```
# debug = false
# quiet = false
# logtarget = "file"
# logfile = ""
# logfile_rotation_interval = "0d"
# logfile_rotation_max_size = "0MB"
# logfile_rotation_max_archives = 5
```

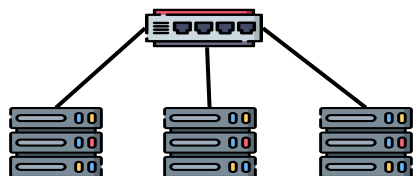
```
hostname = ""
omit_hostname = false
```


Input plugin configs

```
[[inputs.snmp]]
  agents = ["udp://127.0.0.1:161"].
  timeout = "15s"
  version = 2
  community = "SNMP"
  retries = 1
```

```
[[inputs.snmp.field]]
  oid =
  "SNMPv2-MIB::sysUpTime.0"
  name = "uptime"
  conversion = "float(2)"
```

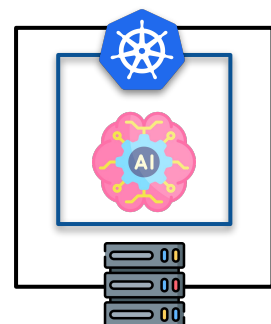
```
[[inputs.snmp.field]]
  oid =
  "SNMPv2-MIB::sysName.0"
  name = "source"
  is_tag = true
```



```
[[inputs.cpu]]
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = false
```

```
[[inputs.disk]]
  ignore_fs = ["tmpfs",
  "devtmpfs", "devfs",
  "iso9660", "overlay", "aufs",
  "squashfs"]
```

```
[[inputs.diskio]]
[[inputs.mem]]
[[inputs.processes]]
[[inputs.swap]]
[[inputs.system]]
[[nvidia-smi]]
```

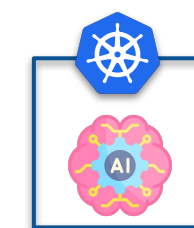


```
[[inputs.opentelemetry]]
  service_address =
  "0.0.0.0:4317"

  timeout = "5s"
```

```
metrics_schema =
  "prometheus-v2"

  tls_cert =
  "/etc/telegraf/cert.pem"
  tls_key =
  "/etc/telegraf/key.pem"
```



```
[[inputs.cloudwatch_metric_streams]]
```

```
  service_address = ":443"
```

```
[[inputs.cloudwatch]]
  region = "us-east-1"
```



Output Plugins

amon
amqp
application_insights
azure_monitor
cloud_pubsub
cloudwatch
cratedb
datadog
discard
dynatrace
elasticsearch
exec
execd
file
graphite

graylog
health
http
influxdb
influxdb_v2
instrumental
kafka
kinesis
librato
logzio
mqtt
nats
newrelic
nsq
opentsdb

prometheus_client
riemann
riemann_legacy
socket_writer
stackdriver
sumologic
syslog
timestream
warp10
wavefront
yandex_cloud_monitorin g

Telegraf Config

```
[[outputs.influxdb_v2]]
  urls = ["http://127.0.0.1:8086"]

  ## Token for authentication.
  token = ""

  ## Organization is the name of the organization you wish to write to.
  organization = ""

  ## Destination bucket to write into.
  bucket = ""

  ## The value of this tag will be used to determine the bucket.  If this
  ## tag is not set the 'bucket' option is used as the default.
  # bucket_tag = ""

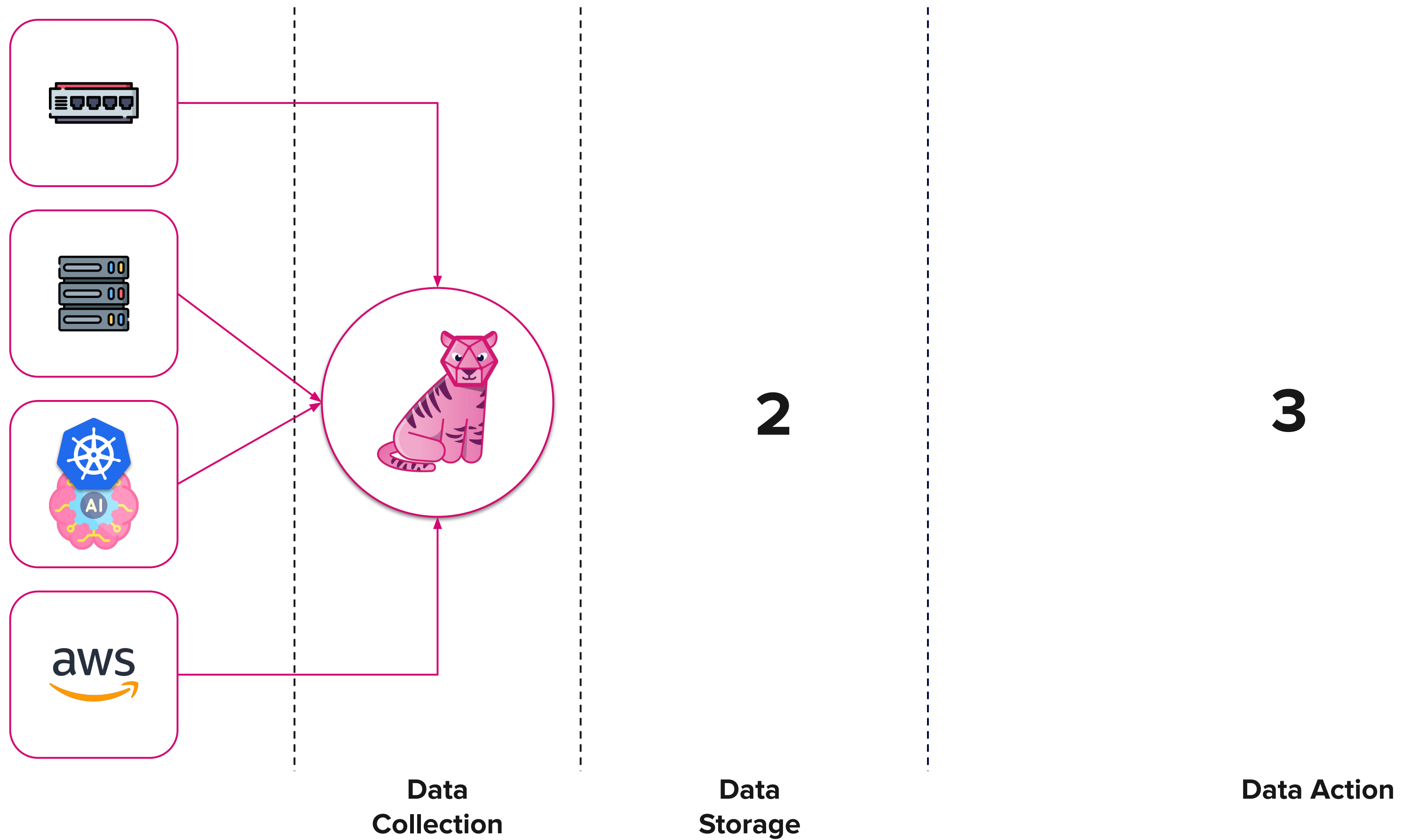
  ## If true, the bucket tag will not be added to the metric.
  # exclude_bucket_tag = false

  ## Timeout for HTTP messages.
  # timeout = "5s"

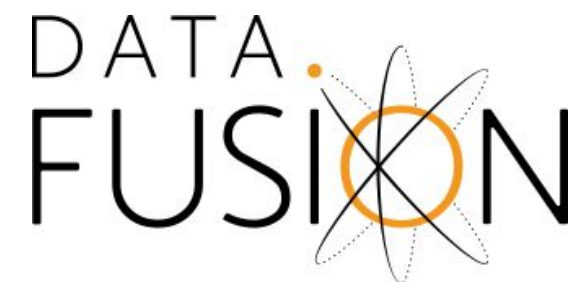
  ## Additional HTTP headers
  # http_headers = {"X-Special-Header" = "Special-Value"}

  ## HTTP Proxy override, if unset values the standard proxy environment
  ## variables are consulted to determine which proxy, if any, should be used.
  # http_proxy = "http://corporate.proxy:3128"
```





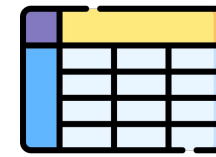
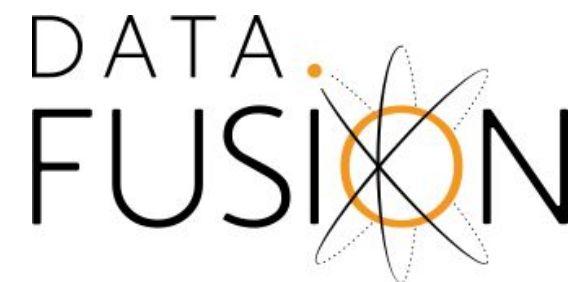
Data Storage



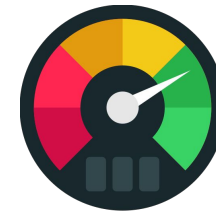
InfluxDB is a database purpose-built for handling time series data at massive scale for real-time analytics.

Developers can ingest, store, and analyze all types of time series data; metrics, events, traces in a single platform. Designed to handle high-speed, high-volume, and high-cardinality data.

InfluxDB 3.0



Schema on write



Write and query millions of rows per second

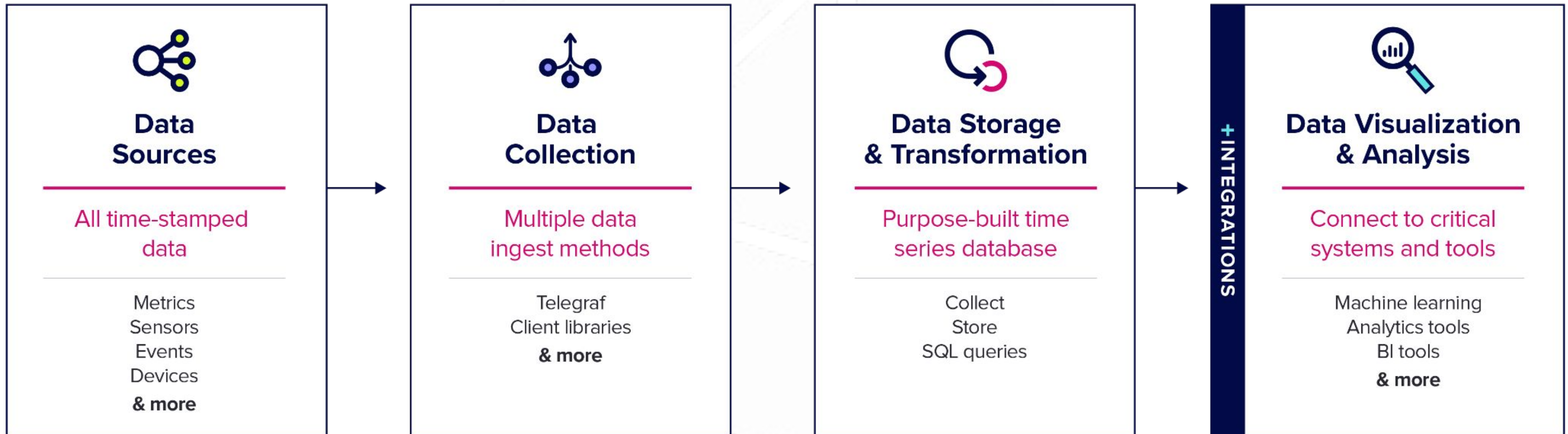


Single datastore for all time series data (metrics, logs, and traces)



SQL, InfluxQL Support

InfluxDB Platform



Concepts: Data Model

Bucket

- All InfluxDB data is stored in a bucket. A bucket combines the concept of a database and a retention period (the duration of time that each data point persists).

Measurement

- A name to a group of data at a high level (Table)

Tag set

- A set of key-value pairs to group data at a low level (values are strings)

Field set

- A set of key-value pairs to represent data (values are numerical & strings)

Timestamp

- Time of the data with nanosecond precision

Series

- A unique combination of measure+tags

Data Storage

- Writing points to InfluxDB uses Line Protocol, which takes the following format:

```
<measurement>[ , <tag-key>=<tag-value> ]  
[ <field-key>=<field-value> ]  
[ unix-nano-timestamp ]
```

	Measurement	Tag Set	Field Set	Timestamp
	server	, hostname=server02, us_west=az	cpu=24.5, mem=12.4	1234567890000000

Reference: <https://docs.influxdata.com/influxdb/cloud/reference/syntax/line-protocol/>

Schema Best Practises

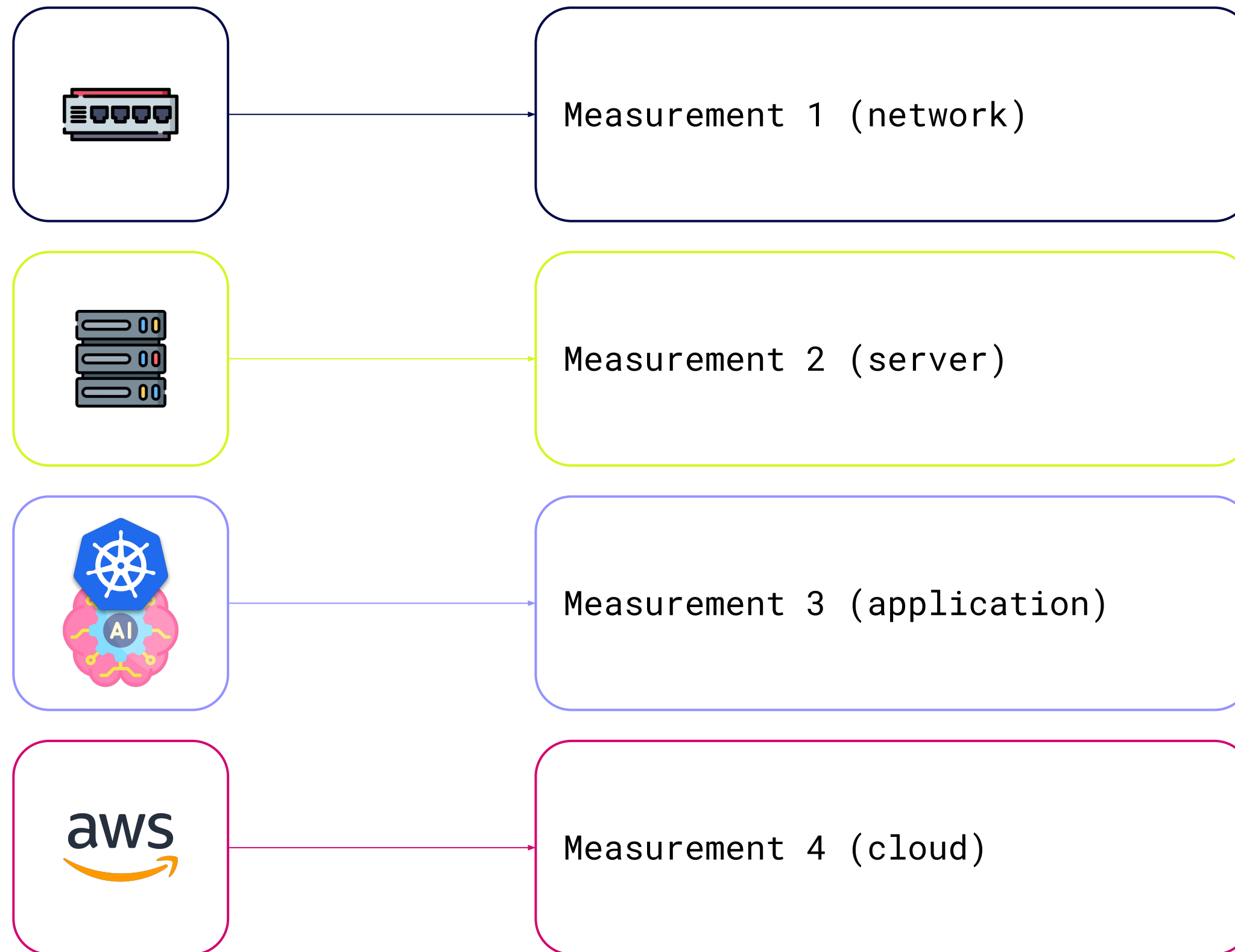
Design for performance

- Avoid Wide Schemas
- Avoid Sparse Schemas
- Homogeneous

Design for query simplicity

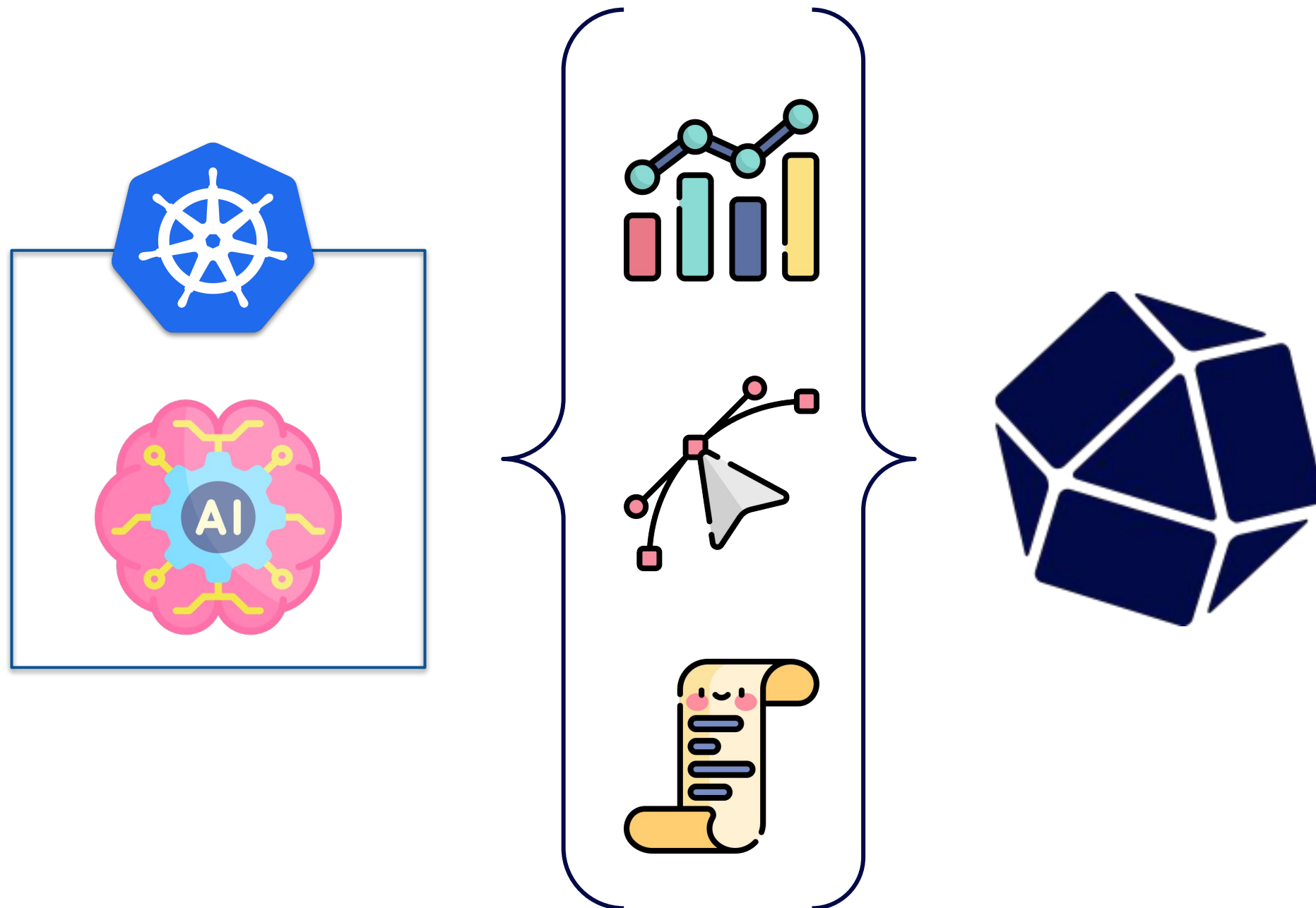
- Keep simple
- Avoid Special characters

Homogenous



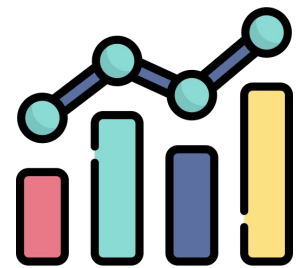
**“homogenous,”
meaning each row
should have the
same tag and field
keys.**

OpenTelemetry - Application



“**OpenTelemetry** is an open-source project for collecting, processing, and exporting observability data like **traces, metrics, and logs** from software applications, simplifying monitoring and performance optimization across languages and platforms.”

OpenTelemetry - Schema



Measurement ?

latency_ms_histogram

Search fields and tag keys

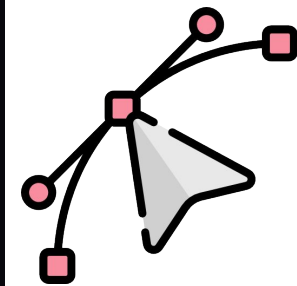
Fields ?

- count
- delta_10
- delta_100
- delta_1000
- delta_10000000
- delta_25
- delta_250
- delta_2500

+ Load more

Tag Keys ?

- http.method
- http.status_code
- operation
- service.name
- span.kind
- status.code



Measurement ?

spans

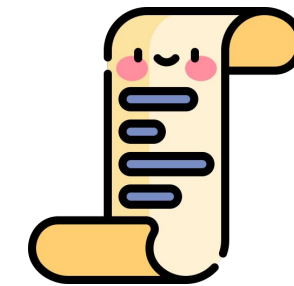
Search fields and tag keys

Fields ?

- attributes
- client-uuid
- duration_nano
- end_time_unix_nano
- host.name
- ip
- kind
- name
- opencensus.exporterversion
- otel.status_code
- parent_span_id
- service.name

Tag Keys ?

- span_id
- trace_id



Measurement ?

logs

Search fields and tag keys

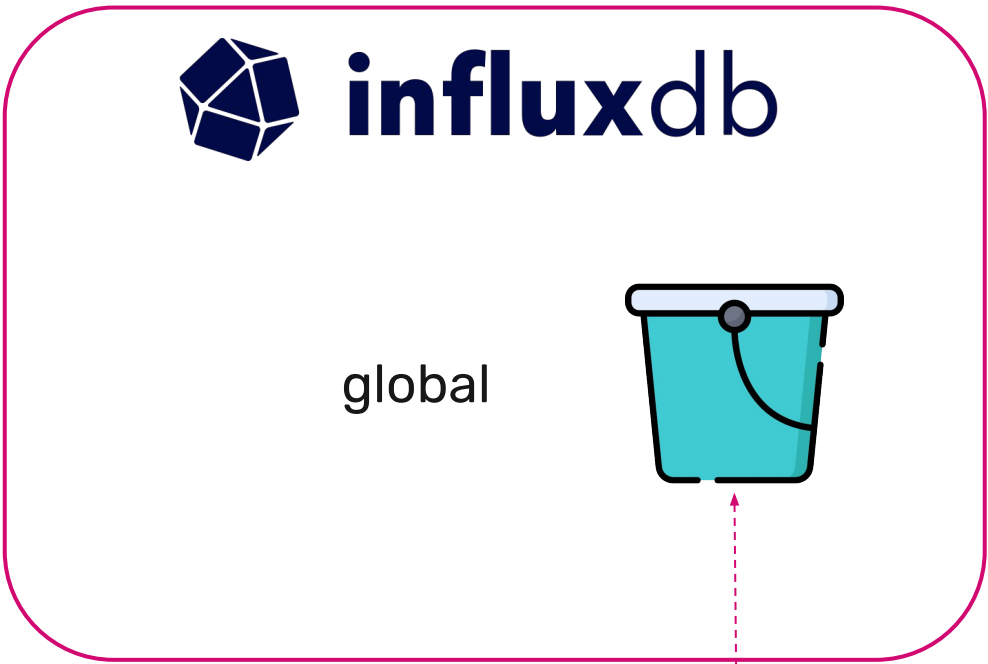
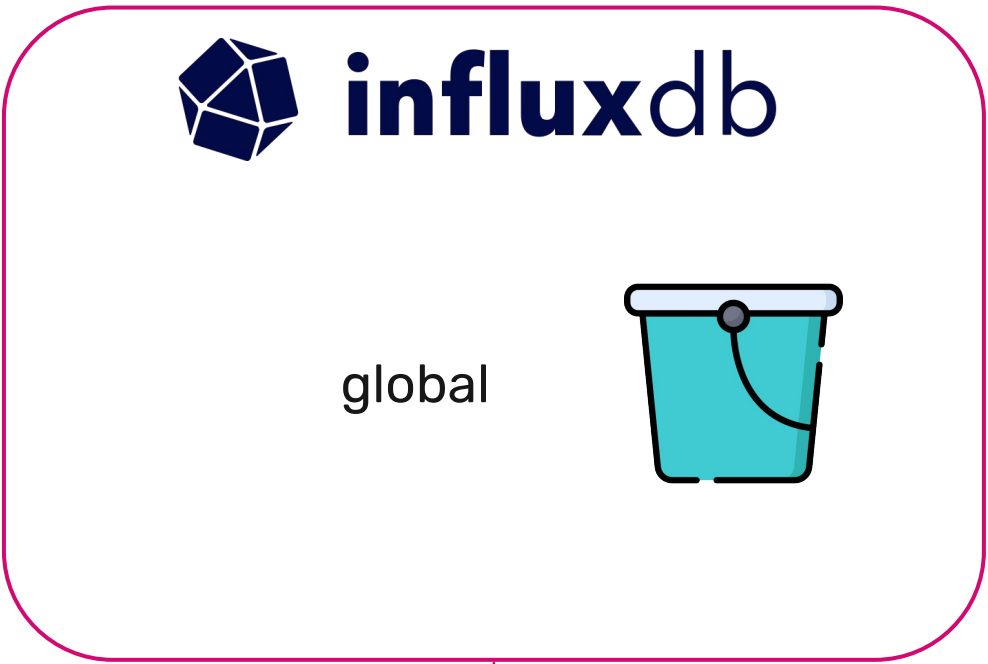
Fields ?

- attributes
- name

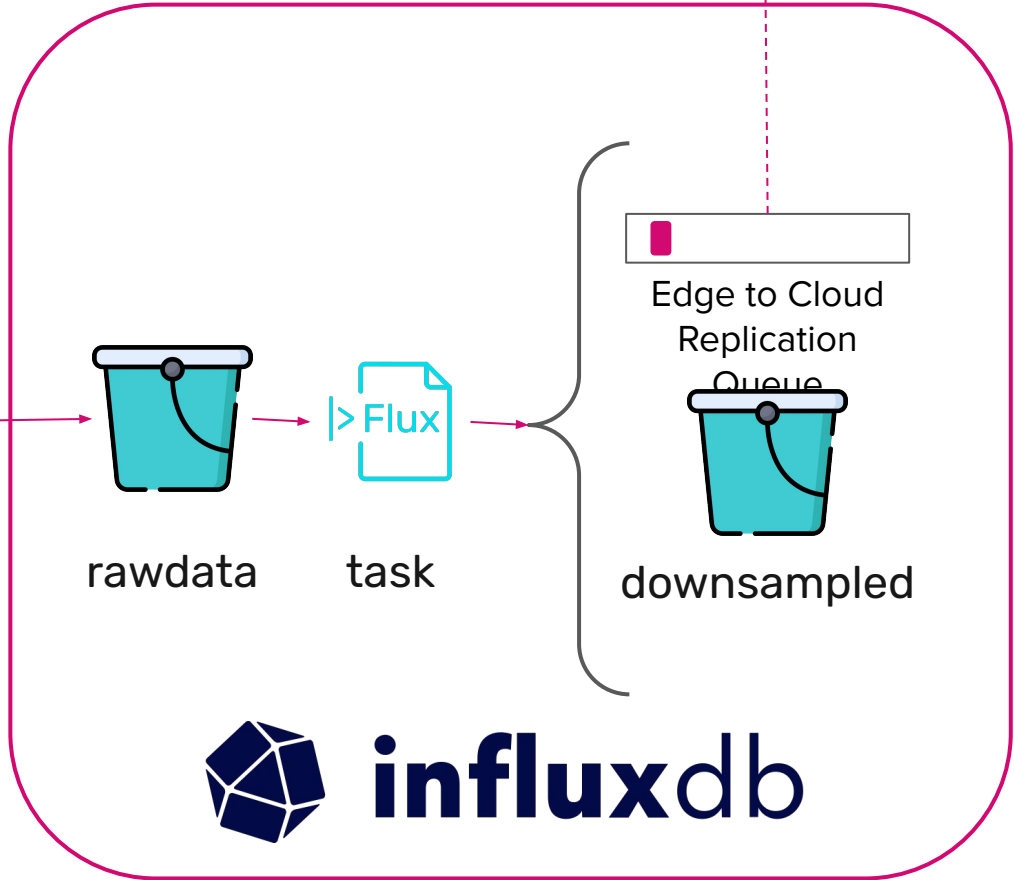
Tag Keys ?

- span_id
- trace_id

Hybrid InfluxDB Solutions

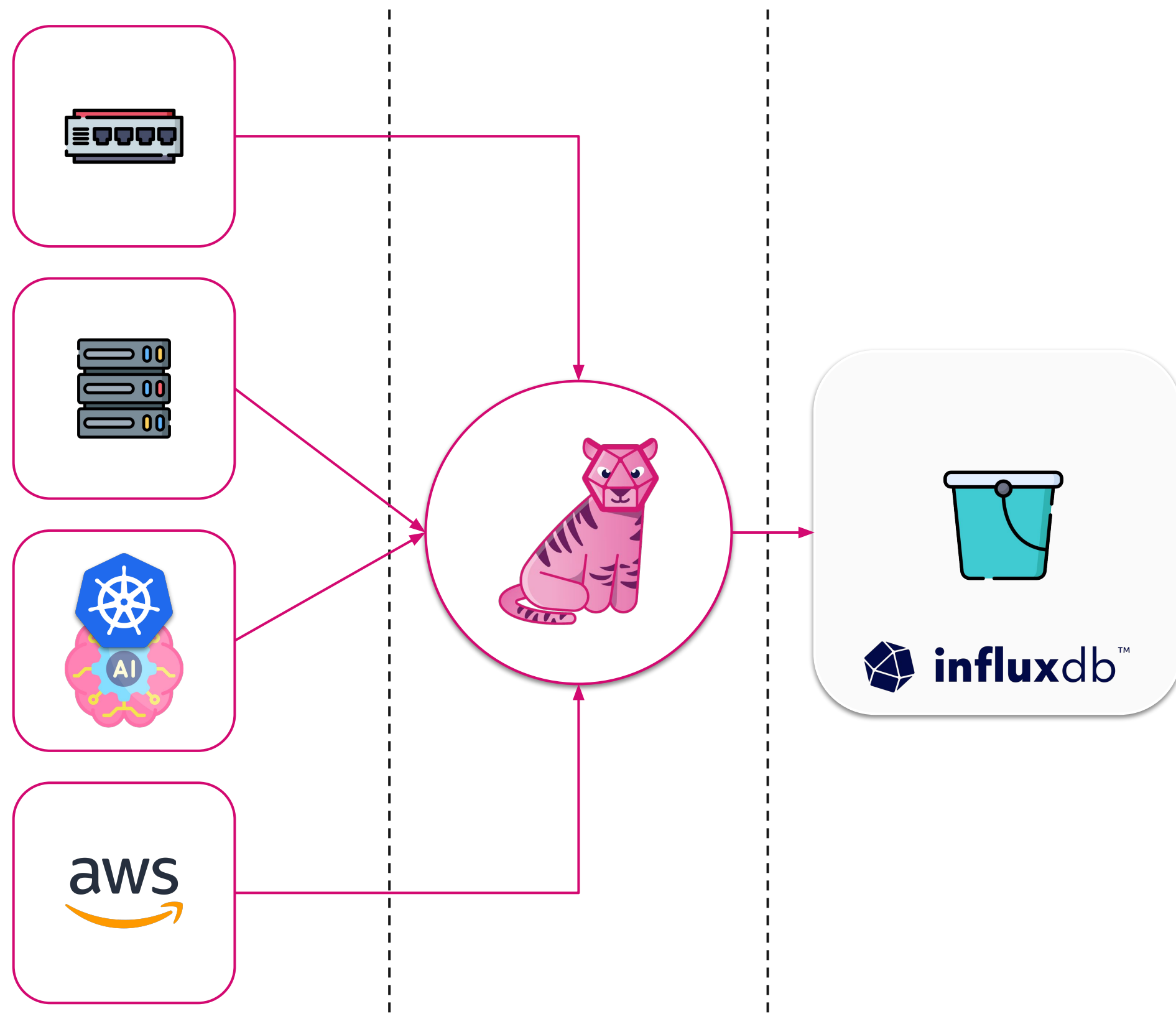


Cloud



Edge

Edge Data Replication



**Data
Collection**

**Data
Storage**

Data Action

3

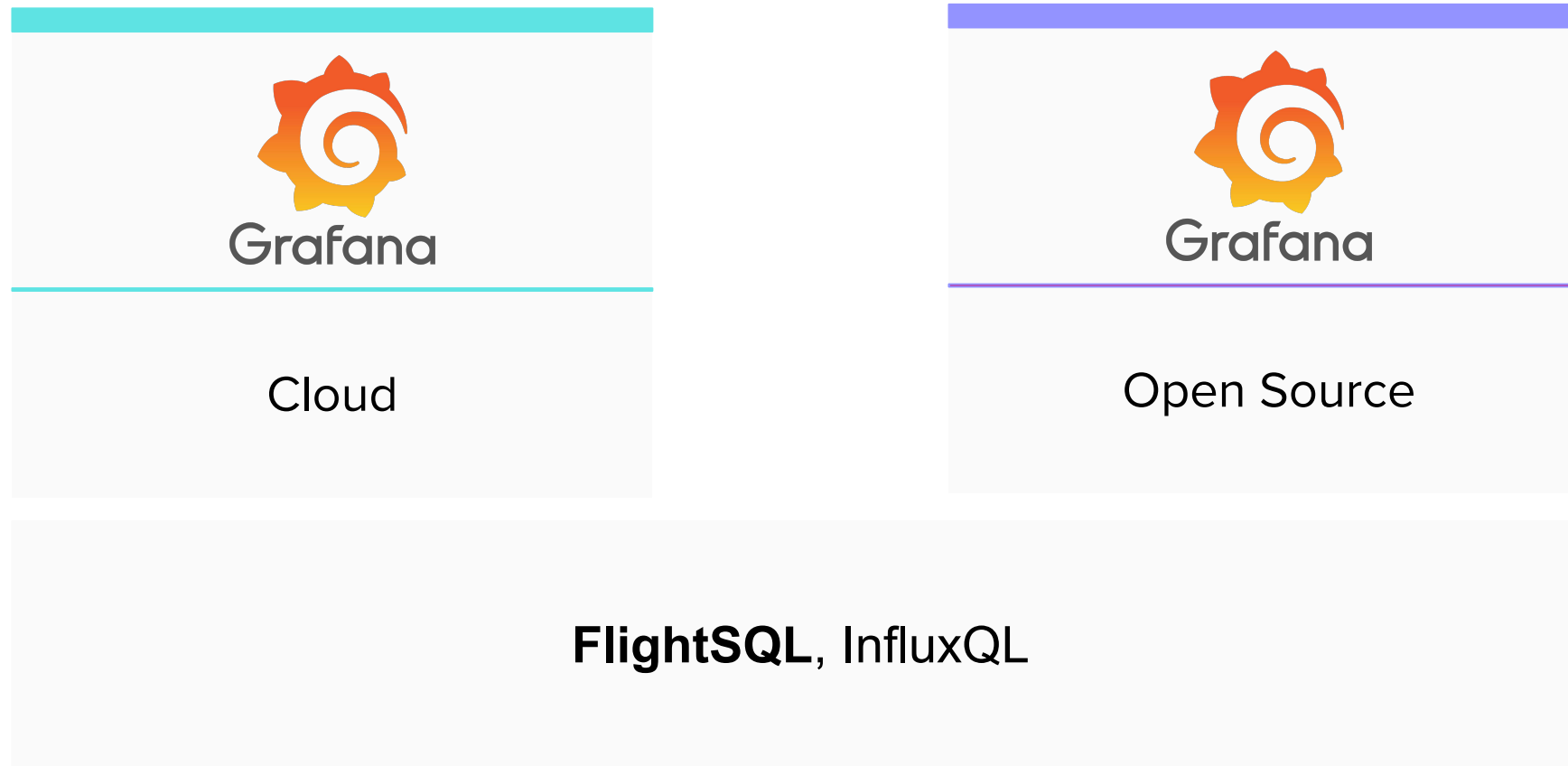
Data Action



Grafana is an open-source data visualization and monitoring platform.

Allows users to create interactive dashboards for real-time data analysis and tracking of metrics across various data sources.

Grafana Flavours



Grafana Flow

The screenshot shows the configuration page for a FlightSQL data source in Grafana. The page title is "FlightSQL" and it indicates the type is "FlightSQL". There is a "Settings" tab selected. A message states: "Provisioned data source. This data source was added by config and cannot be modified using the UI. Please contact your server admin to update this data source." Below this, it says "Alerting supported". The "Name" is set to "FlightSQL" and is the "Default" data source, indicated by a toggle switch. Under "FlightSQL Connection", the "Host:Port" is "eu-central-1-1.aws.cloud2.influxdata.com:443", "Auth Type" is a dropdown menu, and "Require TLS / SSL" is a toggle switch. The "MetaData" section has a table with one entry: Key "bucket-name" and Value "telegraf". At the bottom, there are buttons for "Back", "Explore", "Delete", and "Test".

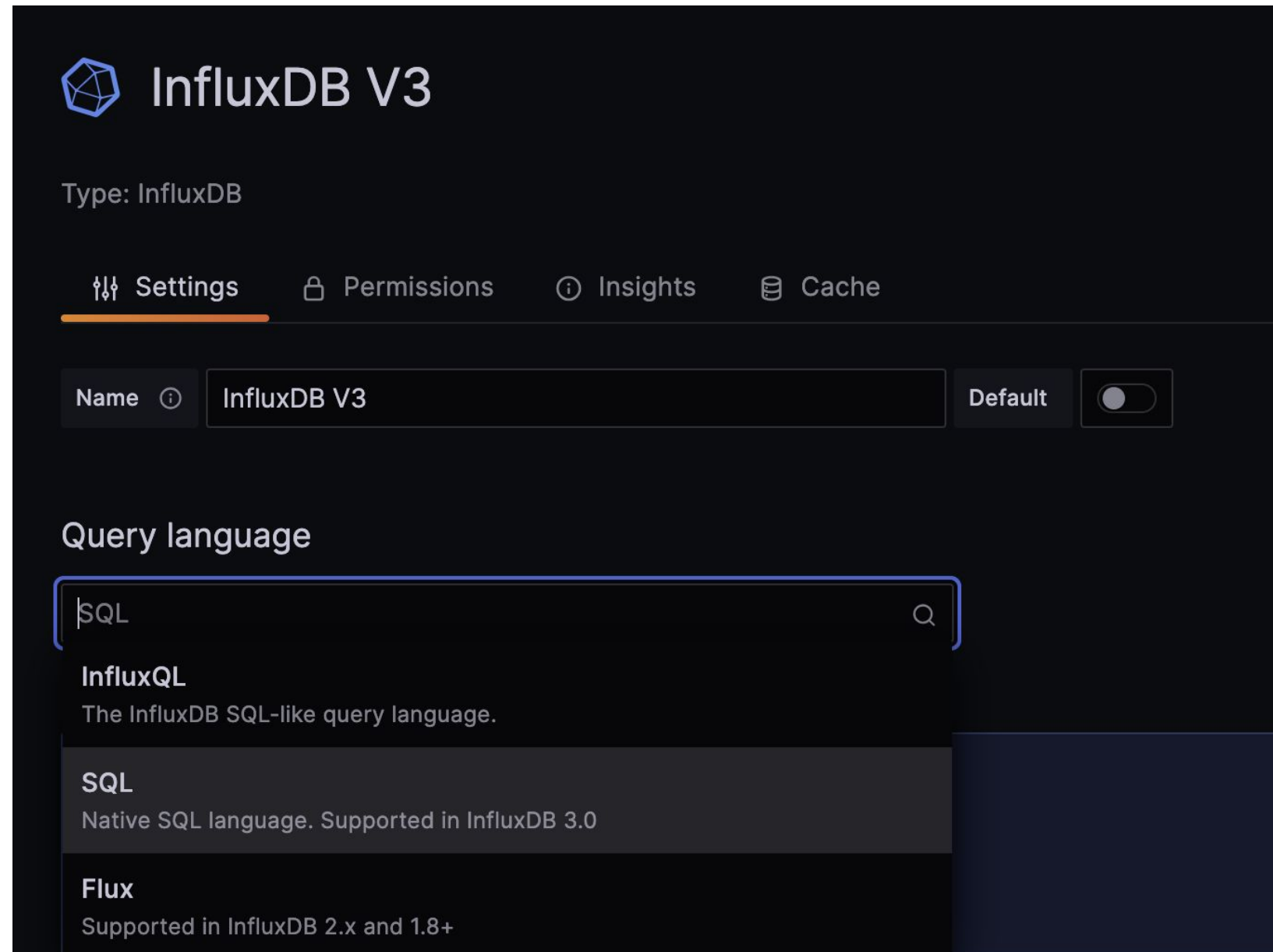
Datasource

The screenshot shows the query editor for the FlightSQL data source. The query is: `1 SELECT usage_idle, time, cpu FROM iox.cpu WHERE $__timeRange(time) order by time`. Below the query editor, there are buttons for "Format As", "Table", "Builder View", and "Show Query Help". There are also buttons for "+ Add query", "Query history", and "Inspector". Below the query editor, there is a table with the following data:

usage_idle	time	cpu
95.7	2023-05-02 14:02:57	cpu4
93.9	2023-05-02 14:02:57	cpu3

Explore

Grafana Official InfluxDB v3 Data Source



Grafana Flow



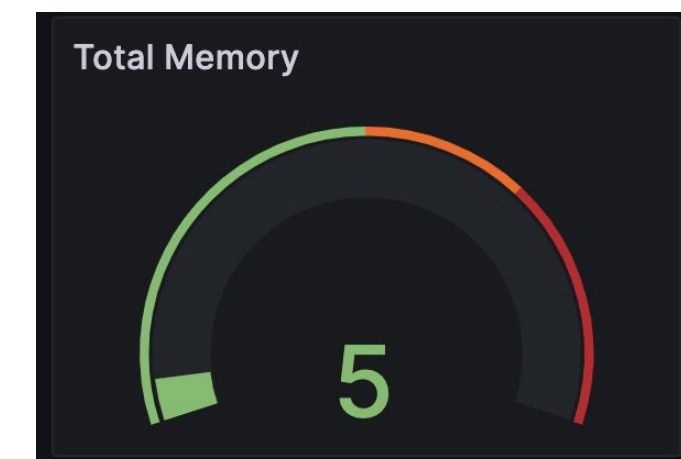
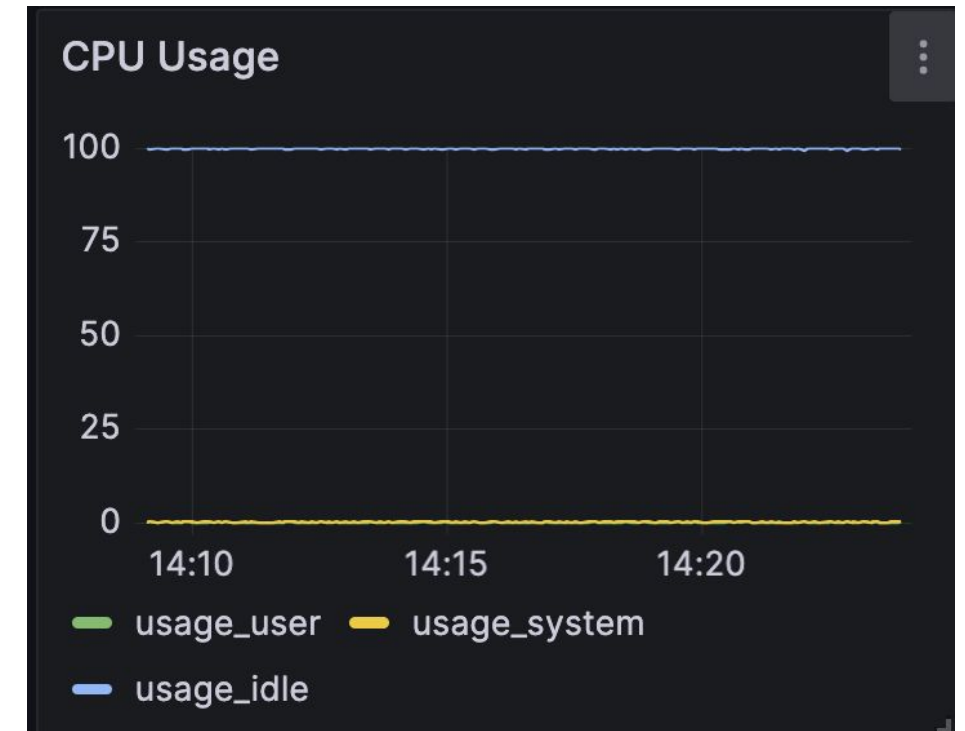
Visualize

Useful Queries

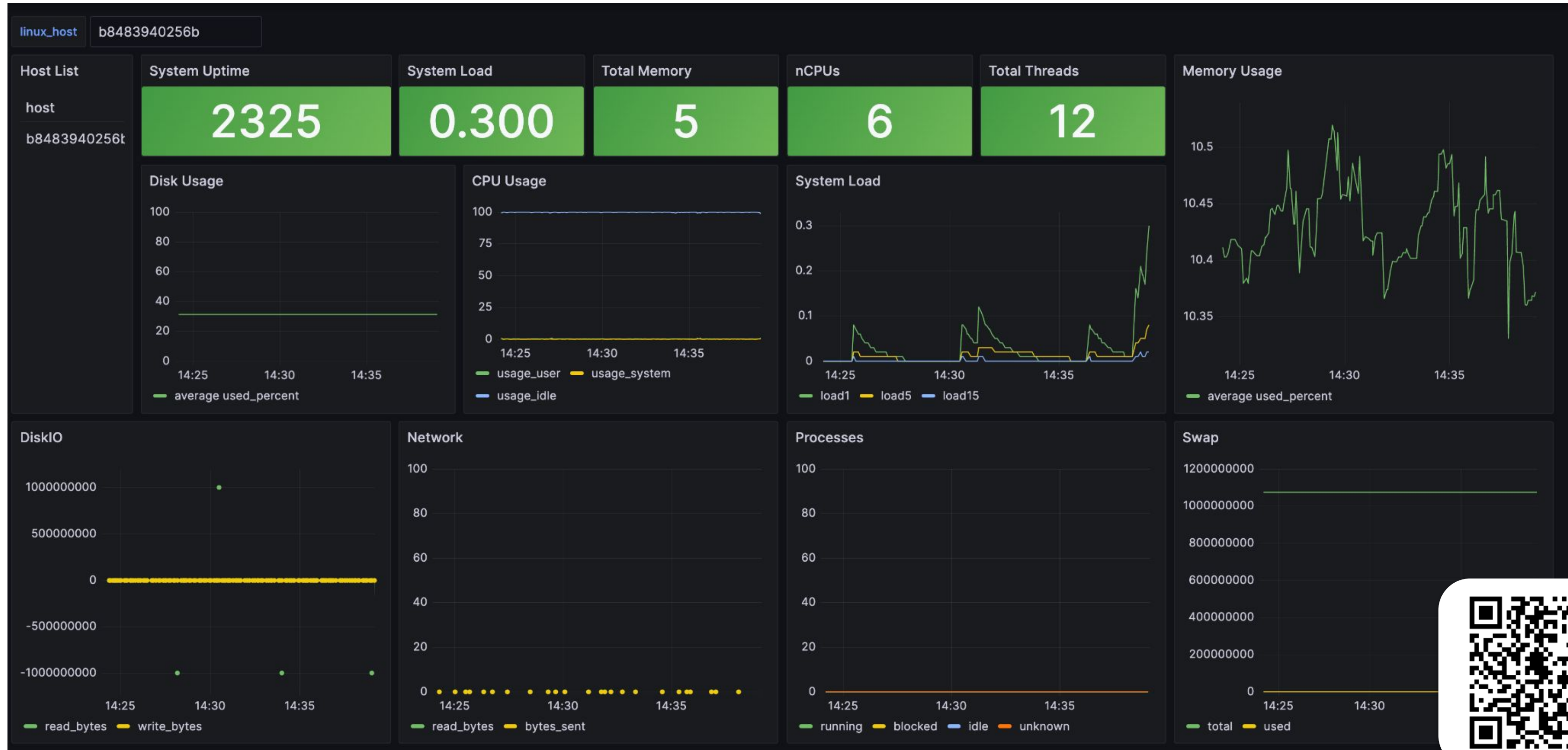
SQL Command

```
SELECT
  $__dateBin(time) ,
  avg(usage_user) AS 'usage_user',
  avg(usage_system) AS 'usage_system',
  avg(usage_idle) AS 'usage_idle'
FROM cpu
WHERE cpu='cpu-total' AND  $__timeRange(time)
GROUP BY 1
ORDER BY time
```

```
SELECT
  selector_last(total, time)['time'] AS time,
  selector_last(total, time)['value'] / 1024 / 1024 / 1024 As total
FROM mem
WHERE host='${linux_host}' AND  $__timeRange(time)
ORDER BY time
```



Example Dashboard



Alerting

The screenshot shows the InfluxDB alerting interface. At the top, the query editor contains the following SQL:

```
1 SELECT
2   $__dateBin(time) ,
3   avg(used_percent) AS 'average used_percent'
4 FROM mem
5 WHERE $__timeRange(time)
6 GROUP BY time
7 ORDER BY time
```

Below the query is a time series graph for 'average used_percent' from 14:34:00 to 14:43:00. A horizontal threshold line is set at 10.4. The current value is 10.56733. The graph shows a fluctuating line that crosses the threshold.

At the bottom, there are two configuration panels:

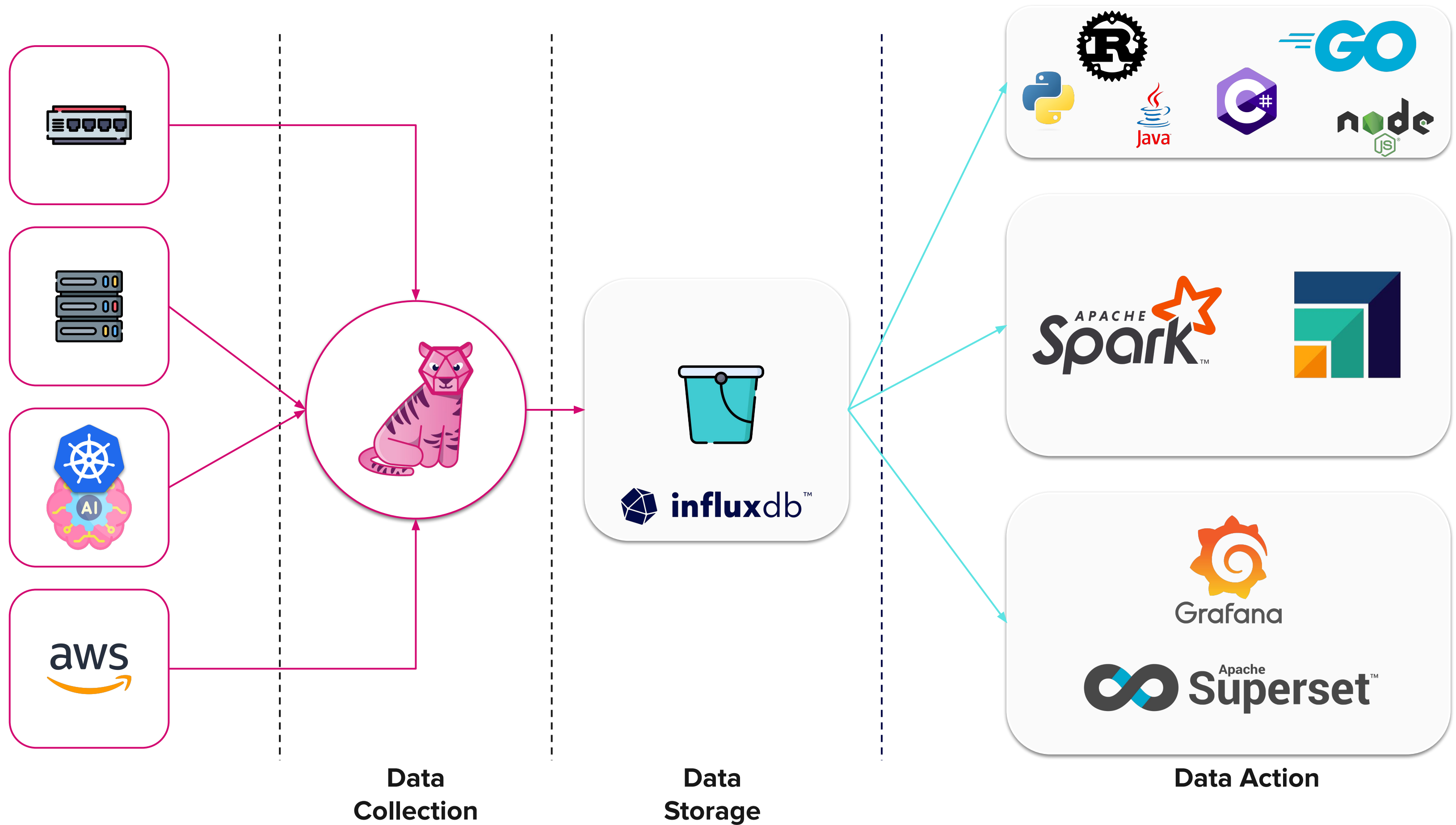
- Panel B (Reduce):** Function: Last, Input: A, Mode: Strict. Series 1 value: 10.56733.
- Panel C (Threshold):** Input: B, Operator: IS ABOVE, Value: 10.4. Status: 1 Firing, 0 normal.

A red arrow points from the 'Rules' section to the 'Endpoints' section.

Rules



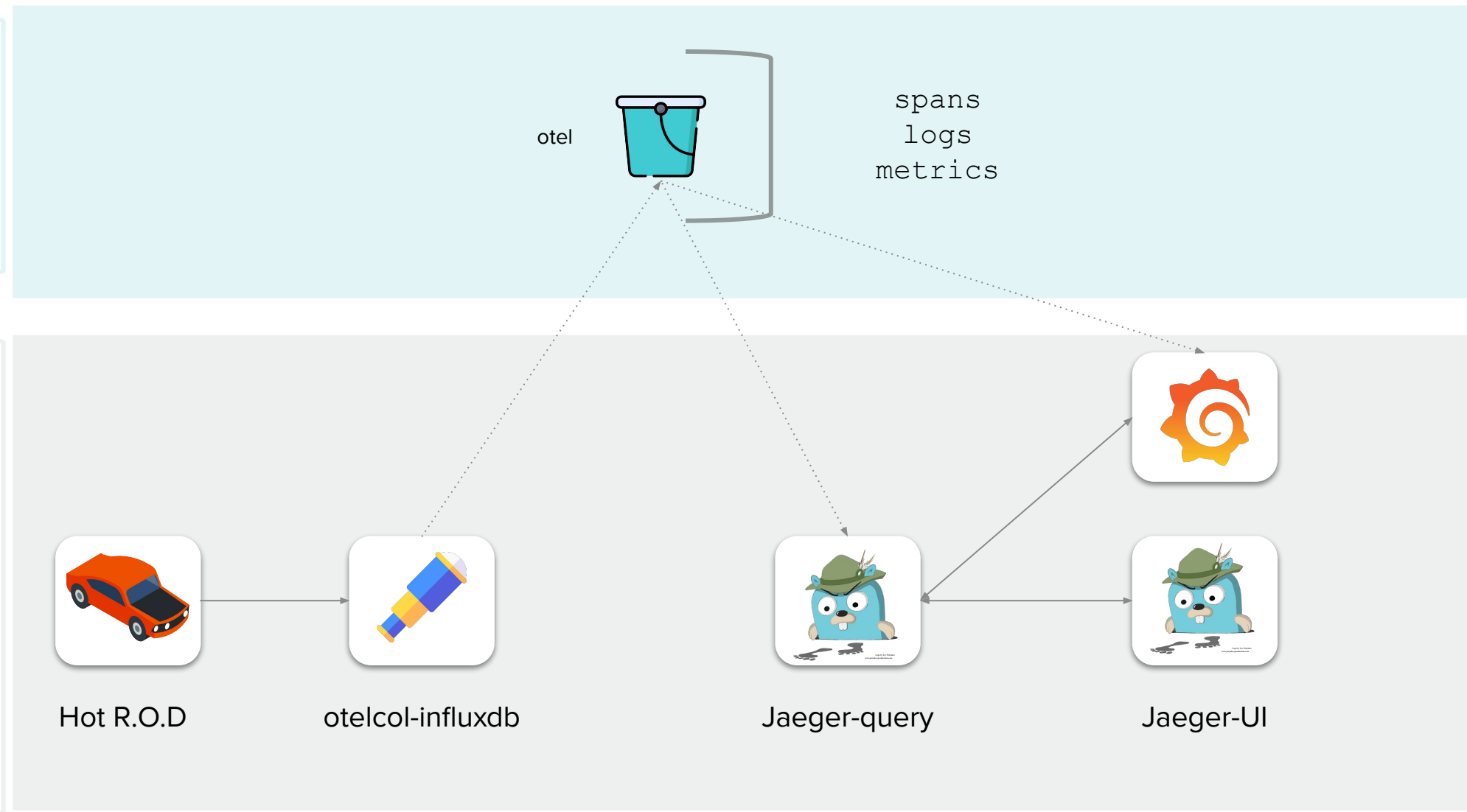
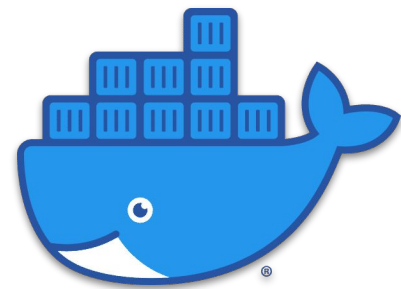
Endpoints



Observability POC/Demo

OpenTelemetry with InfluxDB

This demo provides a practical example of integrating InfluxDB, a high-performance time series database, with OpenTelemetry, an open-source observability framework, to achieve real-time monitoring and tracing of a distributed application.



<https://github.com/influxdata/influxdb-observability>

- Aims to provide a standard for converting **OTEL -> InfluxDB Schema** and **InfluxDB Schema -> OTEL**
- Parts of **otecol-influxdb** can be replaced with **Telegraf**

HotROD - Rides On Demand x Data Explorer | Jay-IDx | Influx x Open Telemetry - Dashboard x +

localhost:8080

Inbox (2) - jcliffor... Home - Confluence Developer DevRel Tools Travel Docs Particle Personal Workshops InfluxDB University Schema Best Prac... Schema Best Prac... Schema Best Prac... Schema Best Prac...

Your web client's id: 3333

Hot R.O.D.

 Rides On Demand 

Rachel's Floral Designs

Trom Chocolatier

Japanese Desserts

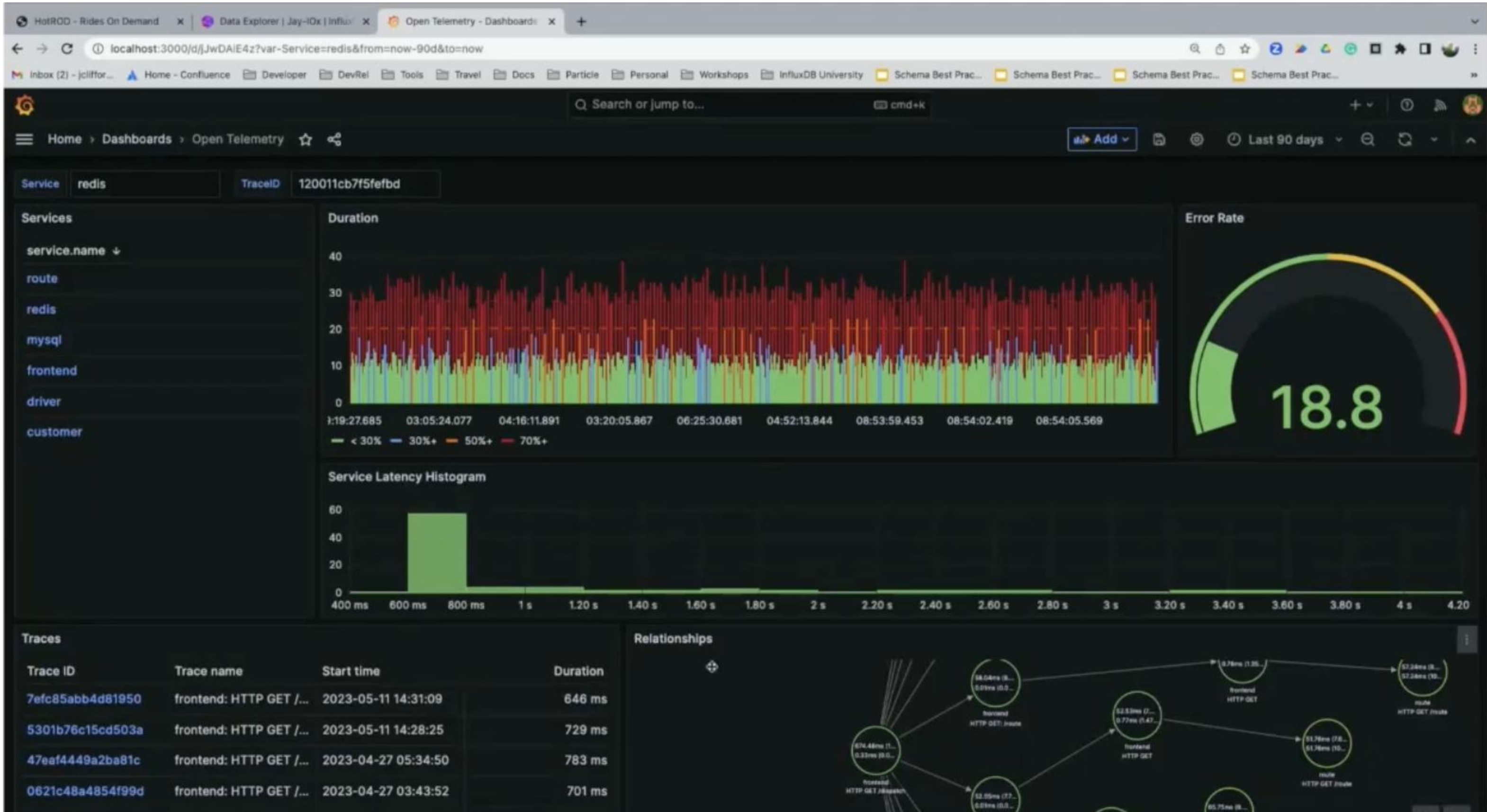
Amazing Coffee Roasters

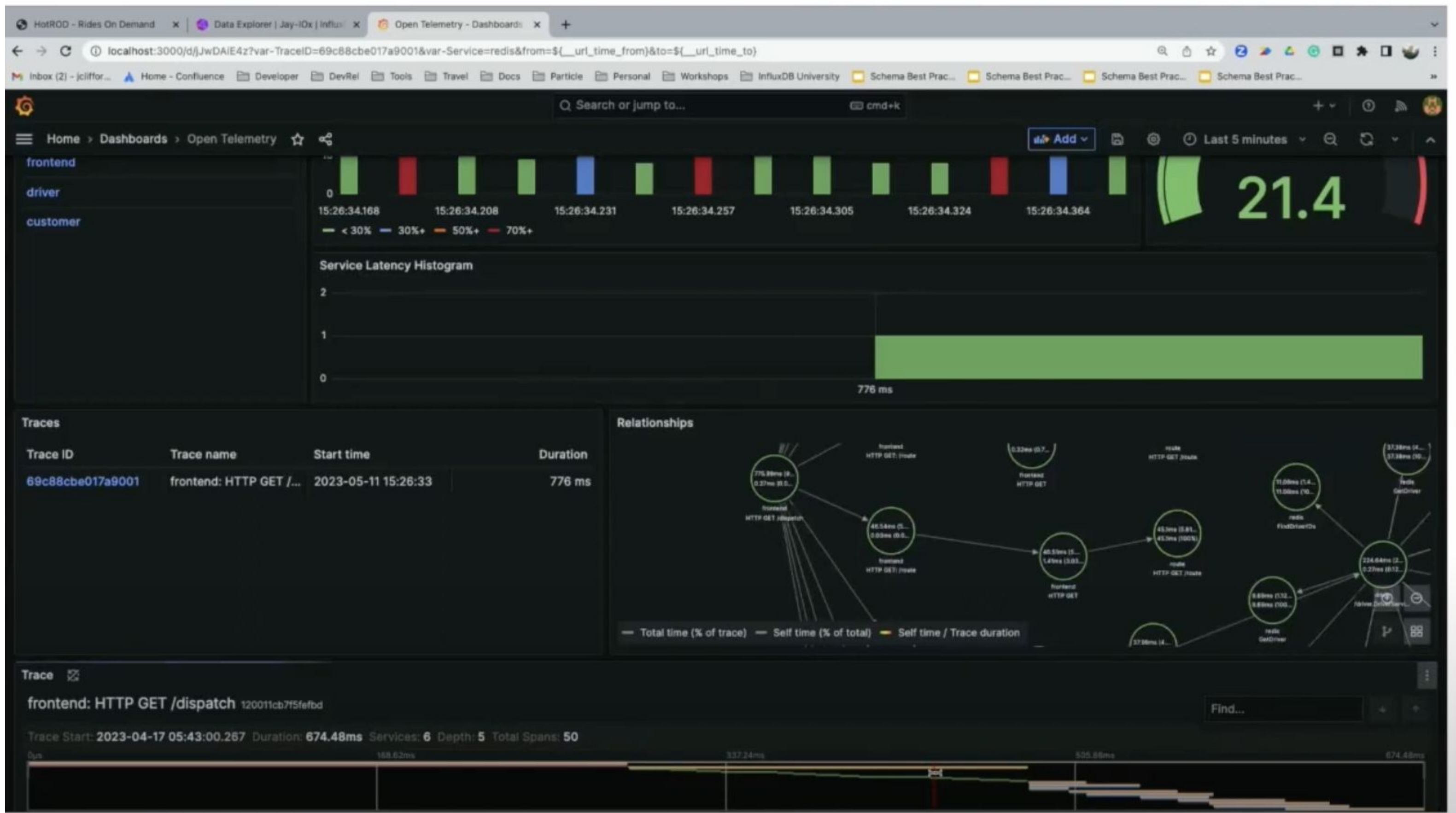
Click on customer name above to order a car.

HotROD **T716521C** arriving in 2min [req: 3333-3, latency: 783ms] [\[find trace\]](#)

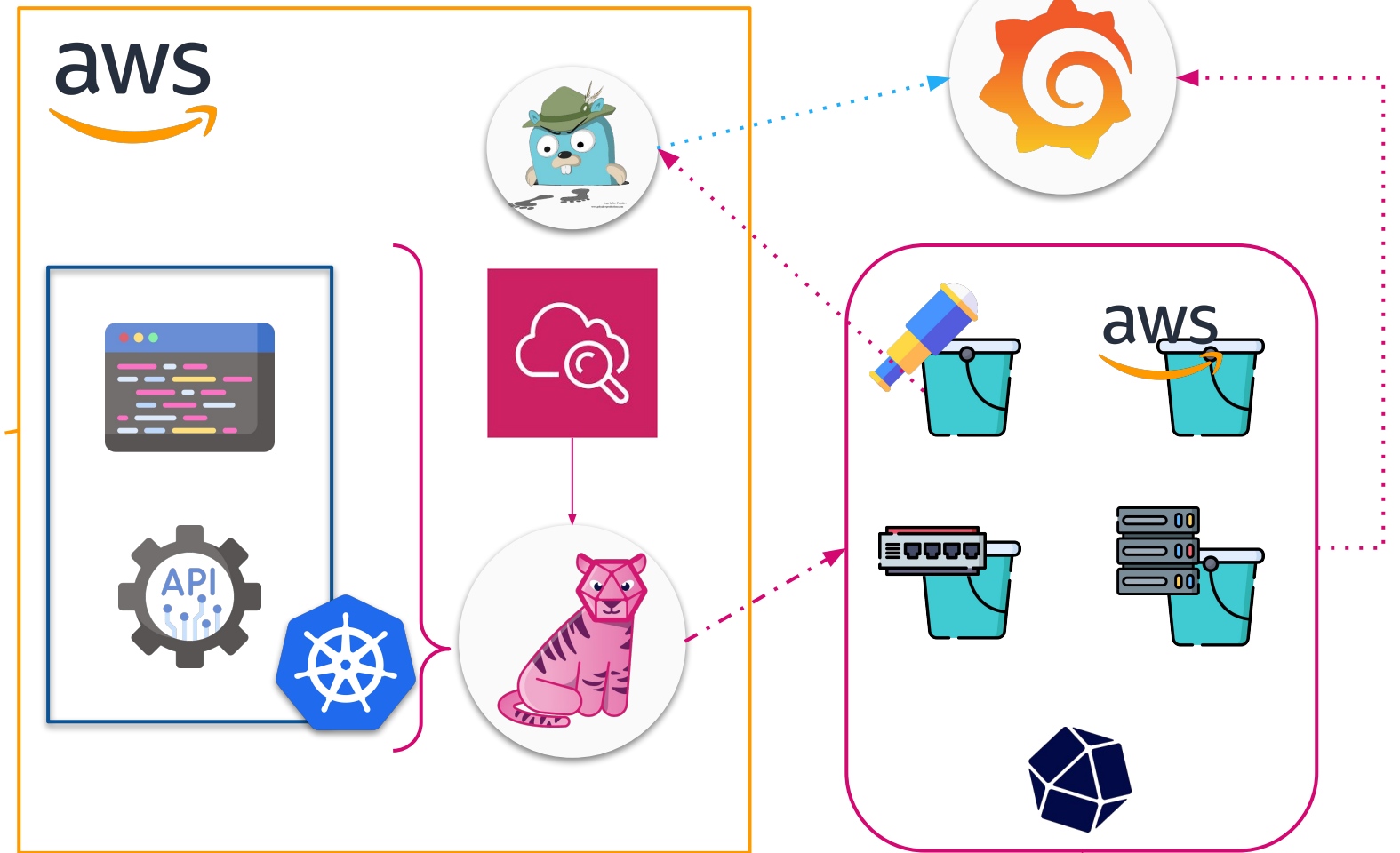
HotROD **T773074C** arriving in 2min [req: 3333-2, latency: 656ms] [\[find trace\]](#)

HotROD **T740861C** arriving in 2min [req: 3333-1, latency: 736ms] [\[find trace\]](#)





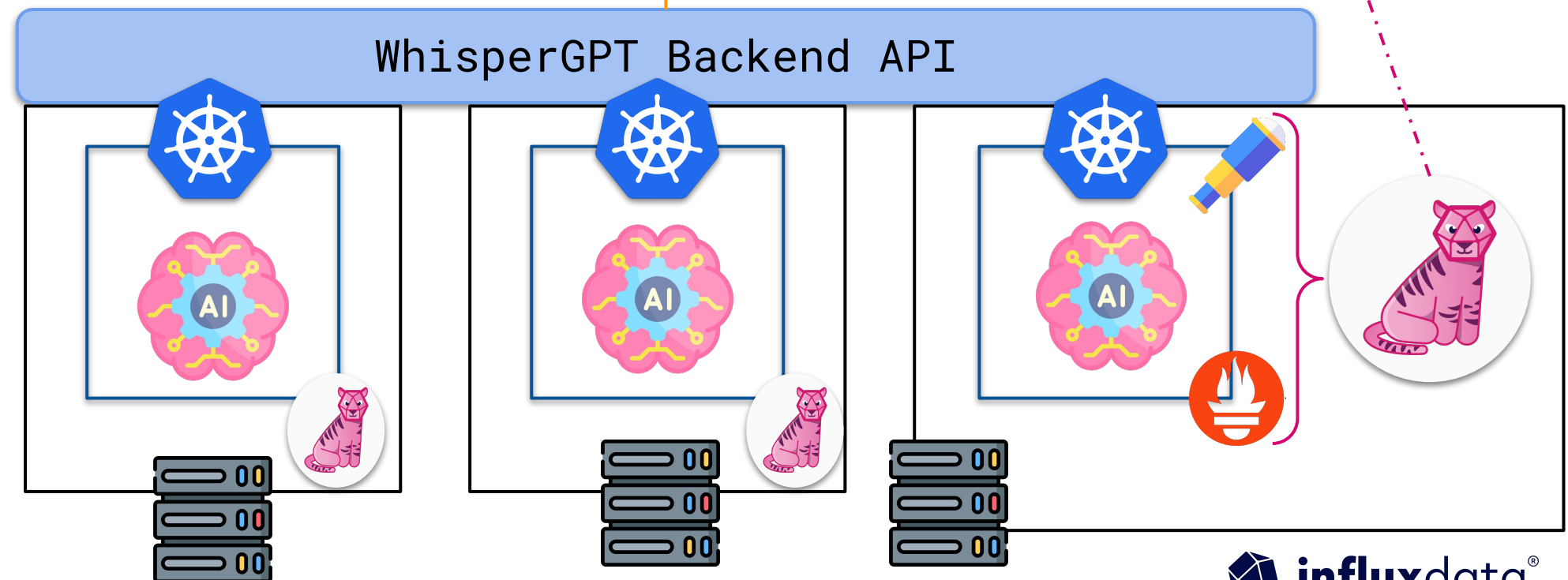
Whisper GPT Solution



Telegraf: Utilize Telegraf as our collection backbone. Deployed on all three servers and cloud infrastructure. Collects data from OTEL, Prometheus, CloudWatch, and raw server-based metrics.

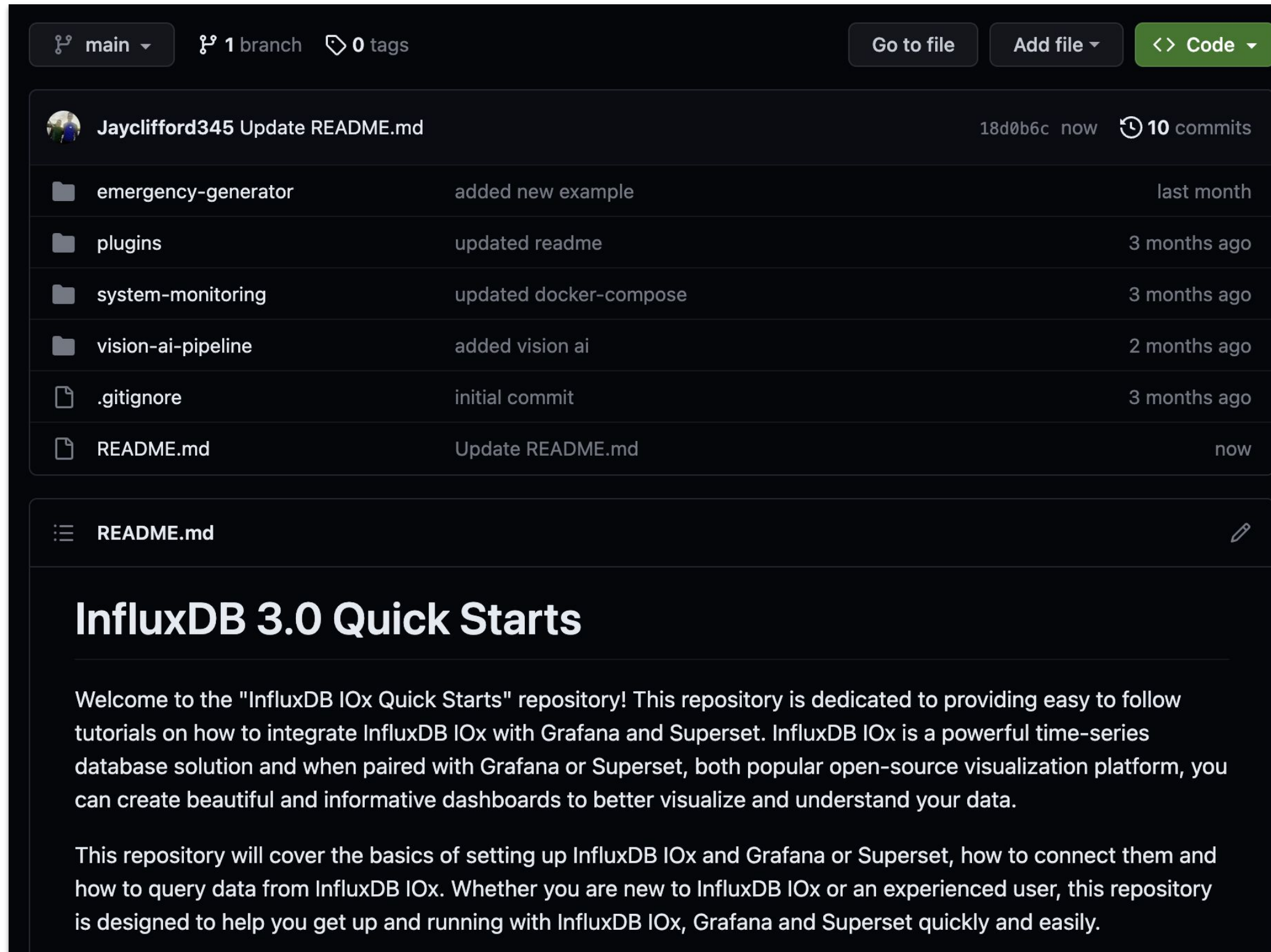
InfluxDB: InfluxDB 3.0 is setup with four repositories called buckets representing each of our datasources. InfluxDB allows us to store metrics, logs, and traces in one datastore.

Grafana: Grafana acts as the observability hub. We use both the FlightSQL and the Jaeger datasource to query our data from InfluxDB 3.0.



Next Steps

Try it yourself - Quick Starts



The screenshot shows a GitHub repository interface. At the top, it displays 'main' branch, '1 branch', and '0 tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a commit by 'Jayclifford345' is shown, titled 'Update README.md', with a commit hash '18d0b6c' and '10 commits' ago. A list of files and folders is visible, including 'emergency-generator', 'plugins', 'system-monitoring', 'vision-ai-pipeline', '.gitignore', and 'README.md'. The 'README.md' file is selected, showing its content. The title of the README is 'InfluxDB 3.0 Quick Starts'. The text in the README reads: 'Welcome to the "InfluxDB IOx Quick Starts" repository! This repository is dedicated to providing easy to follow tutorials on how to integrate InfluxDB IOx with Grafana and Superset. InfluxDB IOx is a powerful time-series database solution and when paired with Grafana or Superset, both popular open-source visualization platform, you can create beautiful and informative dashboards to better visualize and understand your data. This repository will cover the basics of setting up InfluxDB IOx and Grafana or Superset, how to connect them and how to query data from InfluxDB IOx. Whether you are new to InfluxDB IOx or an experienced user, this repository is designed to help you get up and running with InfluxDB IOx, Grafana and Superset quickly and easily.'



<https://github.com/InfluxCommunity/InfluxDB-3-Quick-Starts>

Try it yourself - OTEL

OpenTelemetry & InfluxDB

Welcome

Welcome to the InfluxDB OpenTelemetry Demo! In this demo, you will learn about OpenTelemetry and how it can be integrated with InfluxDB to collect, process, and store metrics, logs and traces.

What is OpenTelemetry?

OpenTelemetry is an observability framework for cloud-native software, designed to provide a single set of APIs for collecting and processing telemetry data such as metrics, logs, and traces. OpenTelemetry aims to simplify instrumentation by providing a consistent and vendor-neutral approach, allowing developers to build and deploy applications without being locked into a specific observability platform. It is a project within the Cloud Native Computing Foundation (CNCF) and is the result of a merger between OpenTracing and OpenCensus.

Architecture Overview

Here is a high level overview of the architecture

```
graph TD; App[Application] -- spans, logs, latency --> Collector[otecol-influxdb]; Collector --> Storage[InfluxDB]; Collector --> Jaeger[Jaeger-query]; Collector --> Grafana; Collector --> JaegerUI[Jaeger-UI];
```

influxdata

Editor Tab 1 +

InfluxDB OTEL Demo

ubuntu \$



<https://github.com/InfluxCommunity/influxdb-observability>

Getting started

Sign up

Influxdata.com

Get InfluxDB

Via cloud marketplace



Learn

- ✓ Self-service content
- ✓ Documentation
- ✓ InfluxDB University
- ✓ Community

<https://influxdbu.com/>

<https://influxcommunity.slack.com/>





THANK YOU

Any Questions?



influxdata.com