@InfluxDB

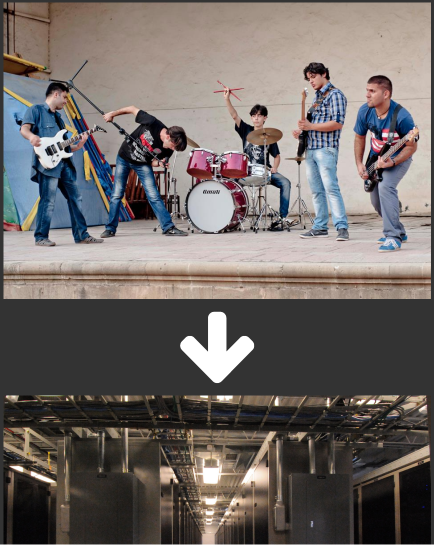
David Norton (@dgnorton)

david@influxdb.com

LISA15 Nov. 8 – 13, 2015 | Washington, D.C.
Sponsored by USENIX in cooperation with LOPSA

Instrumenting a Data Center

Amati


Amati

The problem:

Efficiently monitor hundreds or thousands of servers

The solution:

Automate it!

Automate what?

data collection, storage, analysis, & alerting

Easy!

Write a few scripts & store the data in SQL.

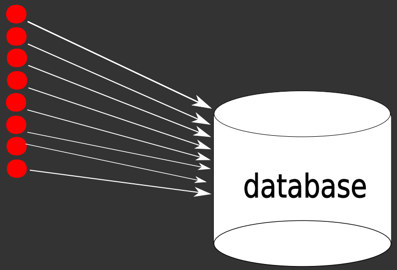
1 server

100 measurements

8,640 per day (once every 10s)

365 days

= 315 million records (points) per year



10 servers

100 measurements per server

8,640 per day (once every 10s)

365 days

= 3.2 billion records (points) per year

2,000 servers

200 measurements per server

17,280 per day (once every 5s)

365 days

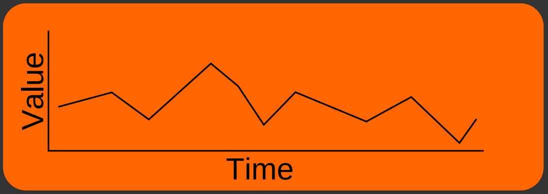
= 2.5 trillion records (points) per year

Time series database is ideal for this type of data & workload

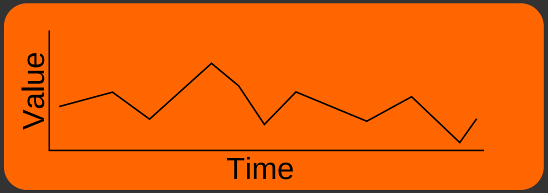
"time series"

Values and time stamps

?

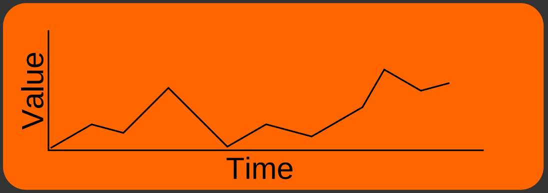


Values and time stamps

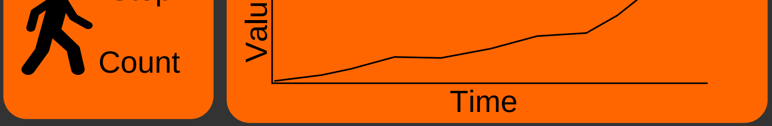


time	value
2015-04-22 5:00 PM	10,0
2015-04-22 6:00 PM	20,0

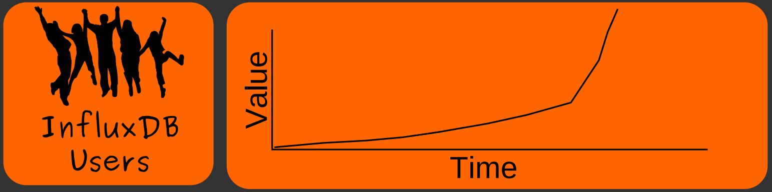
CPU usage every 10 seconds...



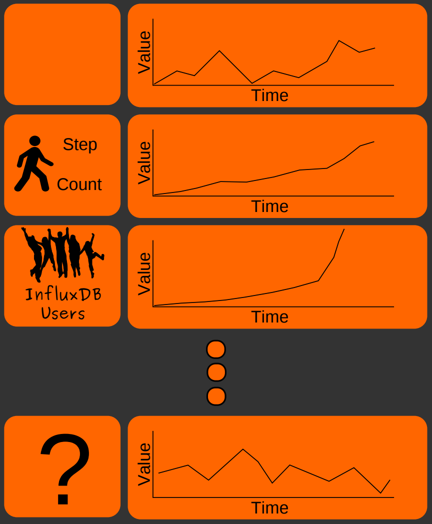
Cumulative steps taken...



Happy InfluxDB users...



Time series database



InfluxDB

InfluxDB

InfluxDB

- time series database

InfluxDB

- time series database

- no external dependencies

InfluxDB

- time series database

- no external dependencies

- distributed & scalable

InfluxDB

- time series database

- no external dependencies

- distributed & scalable

- easy to install, use, & maintain

InfluxDB

- time series database
- no external dependencies
- distributed & scalable
- easy to install, use, & maintain
- open source (MIT license)

InfluxDB

- time series database

- no external dependencies

- distributed & scalable

- easy to install, use, & maintain

- open source (MIT license)

- written in Go

Features

- SQL-like query language

- HTTP(S) API for writes & queries

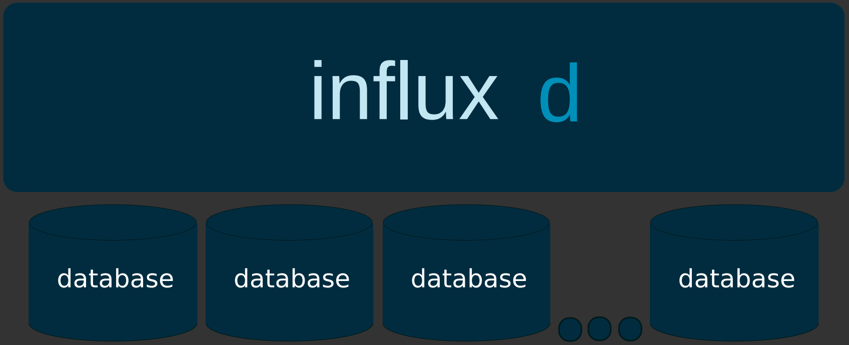
- Supports other protocols (collectd, graphite, opentsdb)

- Automated data retention policies

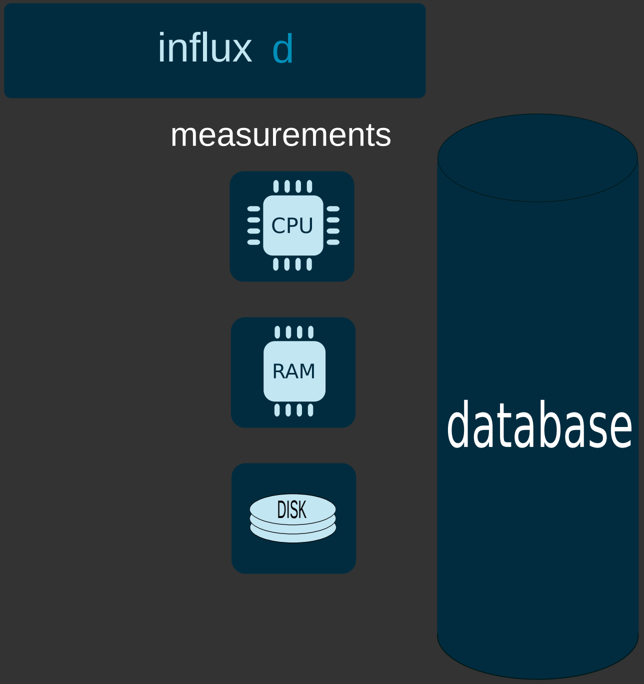
- Aggregate data on-the-fly

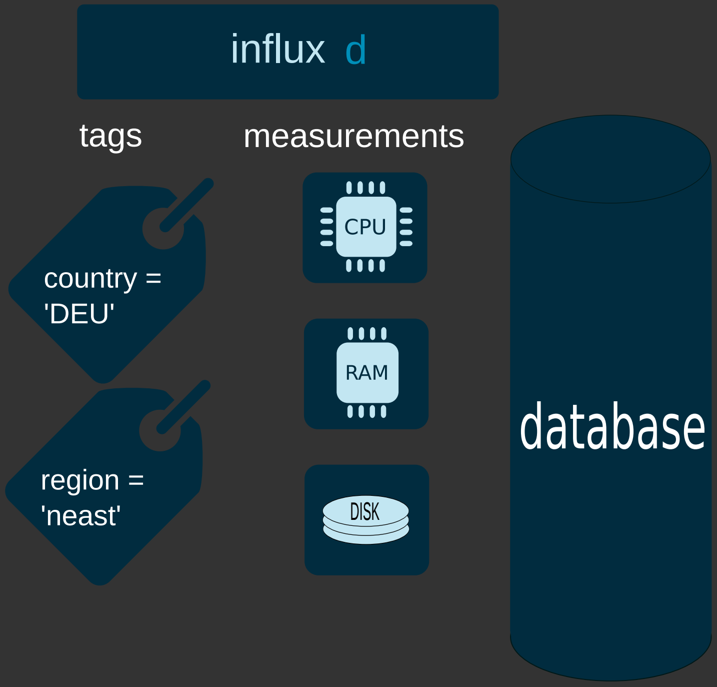
Data model...

influx d


influx d
database
database
database
database

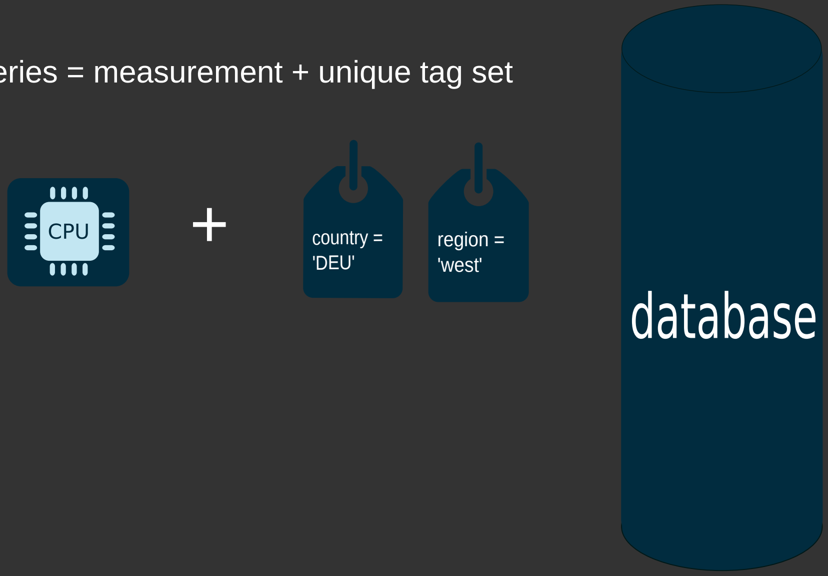
influx d
database


influx d
measurements
CPU
RAM
DISK
database


influx d
tags
measurements
country = 'DEU'
region = 'neast'
CPU
RAM
DISK
database

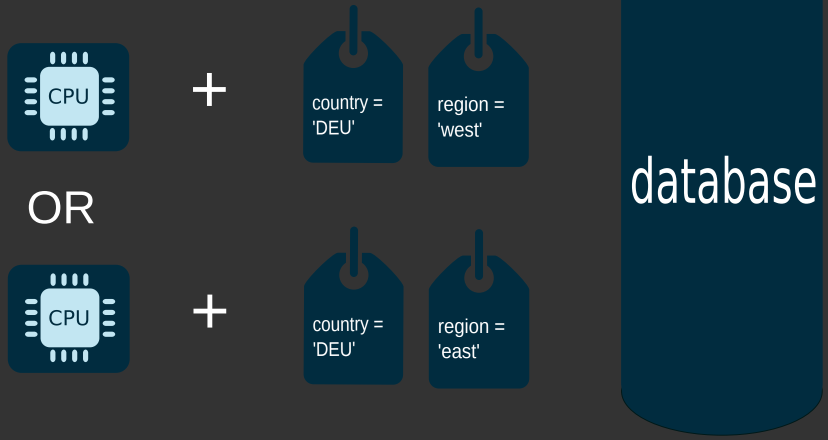
influx d

series = measurement + unique tag set



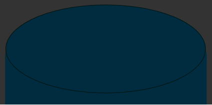
influx d

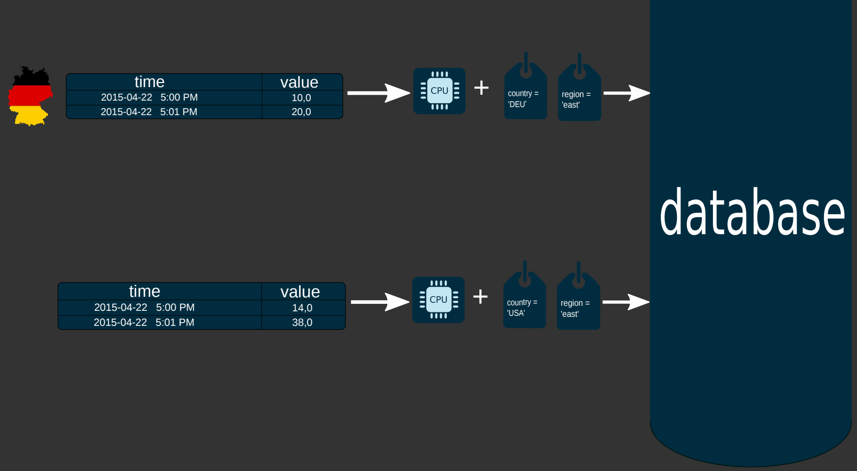
series = measurement + unique tag set



influx d

Points: values in a series





time	value
2015-04-22 5:00 PM	10,0
2015-04-22 5:01 PM	20,0

time	value
2015-04-22 5:00 PM	14,0
2015-04-22 5:01 PM	38,0

What's indexed?

What's indexed?

Metadata:

- Measurement names

What's indexed?

Metadata:

- Measurement names

- Tag keys & values

What's indexed?

Metadata:

- Measurement names

- Tag keys & values

- Field keys

What's indexed?

Metadata:

- Measurement names

- Tag keys & values

- Field keys

- Field values by time*

    WHERE time > now() - 1m

Metadata index is held in-memory at run time

What's not indexed?

What's not indexed?

- Field values by value

```
WHERE value > 2.718
```

What's not indexed?

- Field values by value

    WHERE value > 2.718

- Metadata by time

```
SHOW MEASUREMENTS WHERE time > now() - 1h
```

Data retention

InfluxDB has Retention Policies

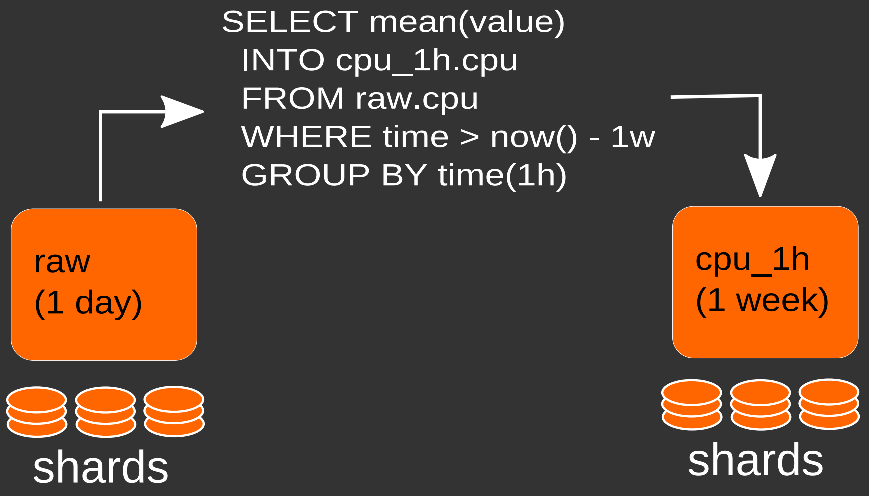
How do they work?

All data is written to a retention policy

Each retention policy has a duration specified

(between 1 hour and infinite)

(between 1 hour and infinite)

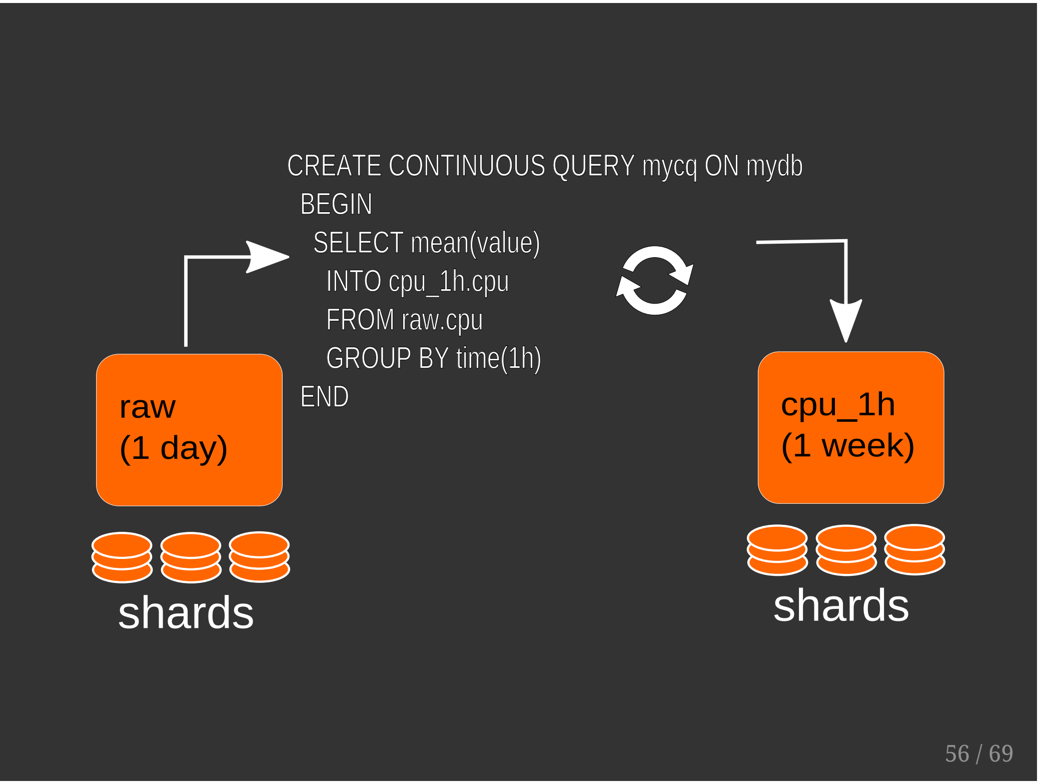


```
SELECT mean(value)
  INTO cpu_1h.cpu
  FROM raw.cpu
  WHERE time > now() - 1w
  GROUP BY time(1h)
```

How to automate downsampling?

Continuous Queries

Stored in the database and run periodically in the background



```
CREATE CONTINUOUS QUERY mycq ON mydb
 BEGIN
  SELECT mean(value)
   INTO cpu_1h.cpu
   FROM raw.cpu
   GROUP BY time(1h)
 END
```

Replication is handled through Retention Policies

Functions

Aggregations:

count, distinct, mean, median, sum

Functions

Aggregations:

count, distinct, mean, median, sum

Selectors:

bottom, first, last, max, min, percentile, top

Functions

Aggregations:

count, distinct, mean, median, sum

Selectors:

bottom, first, last, max, min, percentile, top

Transformations:

ceiling, derivative, difference, floor, histogram,

stddev

Data ingestion

InfluxDB supports:

- collectd
- openTSDB
- graphite

Telegraf

- http://github.com/influxdb/telegraf

- Open Source (MIT License)

- Also written in Go

- Plugin based (~30 currently and growing)

- Cross platform

Client libraries

- Go

- Ruby

- Java

- Python

- etc. (http://github.com/influxdb)

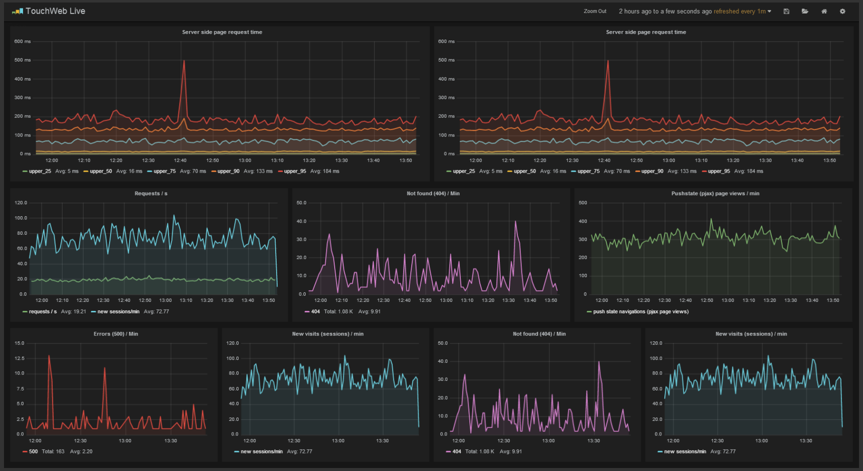
HTTP(S) API

Write data via HTTP using a simple text-based protocol

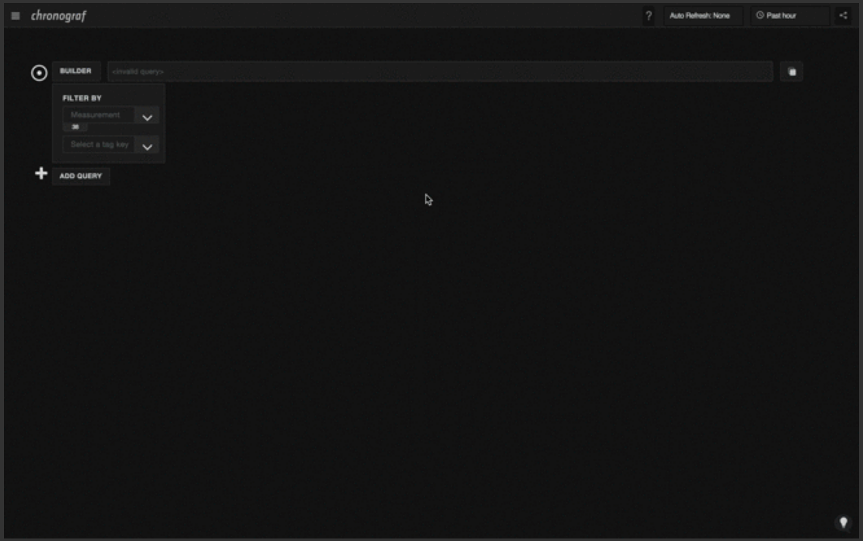
```
cpu,host=serverA value=10.0 1234567890
```

Visualization

Grafana


TouchWeb Live
Zoom Out
2 hours ago to a few seconds ago refreshed every 1m
Server side page request time
upper_25 Avg: 5 ms upper_50 Avg: 16 ms upper_75 Avg: 70 ms upper_90 Avg: 133 ms upper_95 Avg: 184 ms
Server side page request time
upper_25 Avg: 5 ms upper_50 Avg: 16 ms upper_75 Avg: 70 ms upper_90 Avg: 133 ms upper_95 Avg: 184 ms
Requests / s
requests / s Avg: 19.21 new sessions/min Avg: 72.77
Not found (404) / Min
404 Total: 1.08 K Avg: 9.91
Pushstate (pjax) page views / min
push state navigations (pjax page views)
Errors (500) / Min
500 Total: 163 Avg: 2.20
New visits (sessions) / min
new sessions/min Avg: 72.77
Not found (404) / Min
404 Total: 1.08 K Avg: 9.91
New visits (sessions) / min
new sessions/min Avg: 72.77

Chronograf


chronograf
Auto Refresh: None
Past hour
BUILDER
<invalid query>
FILTER BY
Measurement
Select a tag key
ADD QUERY

Thank You!

@InfluxDB

David Norton (@dgnorton)

david@influxdb.com