Fast, Cheap, DIY Monitoring with Open Source Analytics and Visualization

Robert Hodges - Altinity

Conf42: DevSecOps 2023

Let’s make some introductions

Robert Hodges

Database geek with 30+ years on DBMS. Kubernaut since 2018. Day job: Altinity CEO

Altinity Engineering

Database geeks with centuries of experience in DBMS and applications

Altinity

ClickHouse support and services including Altinity.Cloud
Authors of Altinity Kubernetes Operator for ClickHouse
and other open source projects

Monitoring is for answering questions

- Why users are seeing performance problems?
- When did it start?
- How many users are affected?
- Which service is at fault?

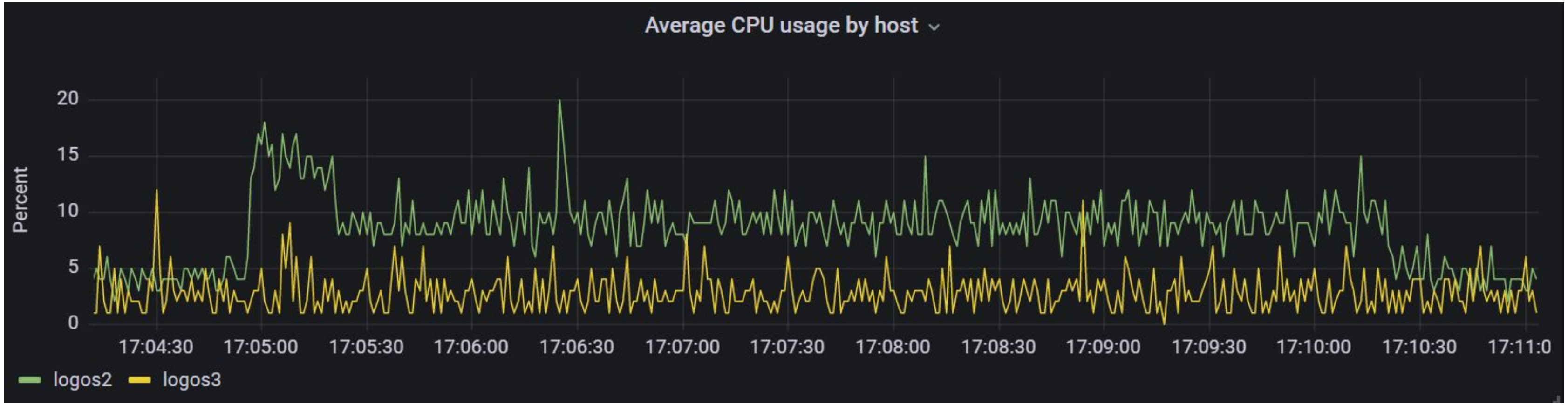
What’s the best way to answer these questions?

This…

```
$ vmstat -n 2 10
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 343296 21690808 2290104 6897160    0    0     3   187    0    2  4  2 94  0  0
 0  0 343296 21690800 2290104 6897160    0    0     0    60 2989 7688  2  1 97  0  0
 0  0 343296 21690140 2290104 6897164    0    0     0    72 4704 13677  3  2 95  0  0
 0  0 343296 21689888 2290104 6897164    0    0     0    14 3132 9364  2  1 97  0  0
 0  0 343296 21690220 2290104 6897168    0    0     0    86 3014 7995  1  1 97  0  0
 0  0 343296 21690448 2290104 6897176    0    0     0    20 2660 7297  1  1 98  0  0
 0  0 343296 21690268 2290104 6897176    0    0     0    12 2695 7222  1  1 98  0  0
 1  0 343296 21690196 2290104 6897180    0    0     0    80 3641 10419  2  1 97  0  0
 0  0 343296 21689696 2290104 6897180    0    0     0    14 4108 12605  3  2 95  0  0
 0  0 343296 21689900 2290104 6897184    0    0     0    60 2688 7270  2  1 97  0  0
```

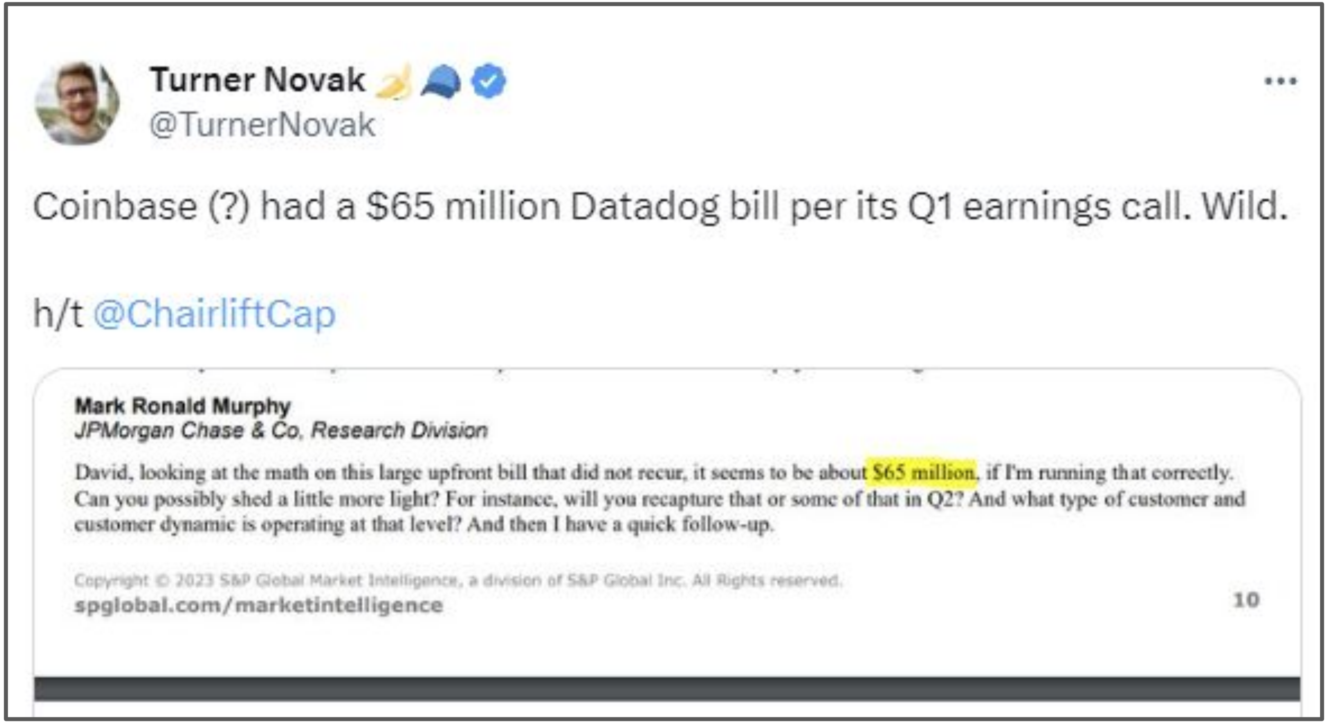
What’s the best way to answer these questions?

Or this…



Altinity

Off-the-shelf solutions? Perhaps not for you…



Turner Novak 🍌🧢
@TurnerNovak

Coinbase (?) had a $65 million Datadog bill per its Q1 earnings call. Wild.

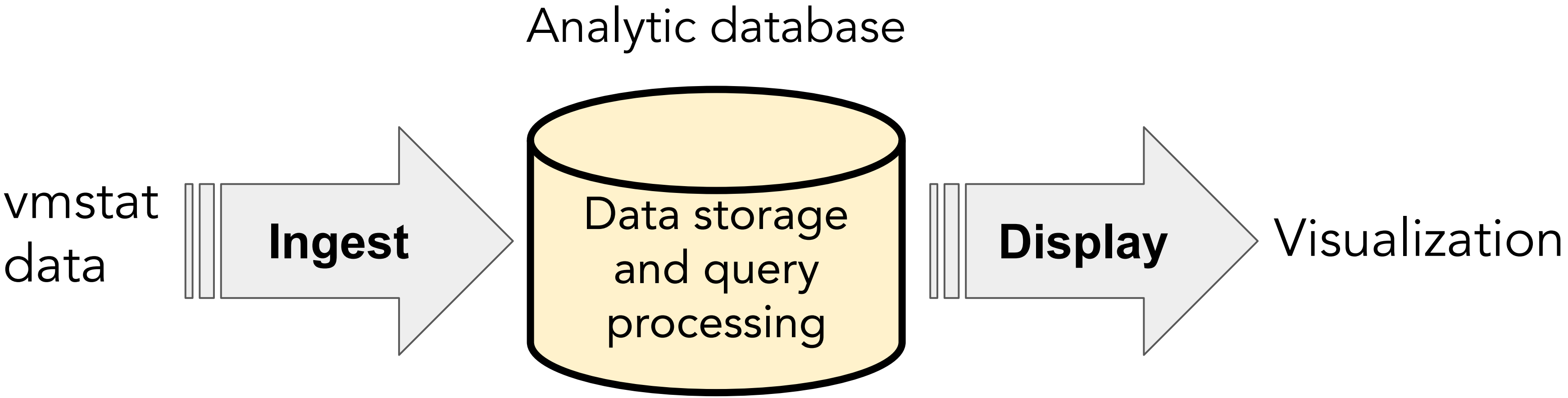
h/t @ChairliftCap

Mark Ronald Murphy
JPMorgan Chase & Co, Research Division

David, looking at the math on this large upfront bill that did not recur, it seems to be about $65 million, if I'm running that correctly. Can you possibly shed a little more light? For instance, will you recapture that or some of that in Q2? And what type of customer and customer dynamic is operating at that level? And then I have a quick follow-up.

Copyright © 2023 S&P Global Market Intelligence, a division of S&P Global Inc. All Rights reserved.
spglobal.com/marketintelligence 10

Let’s build a monitoring system with open source



Analytic database

vmstat data

Ingest

Data storage and query processing

Display

Visualization

Pick an open source analytic database

Query and search on semi-structured data

OpenSearch
Apache 2.0

Full-text search, log analytics

Real-time analytics on structured data

ClickHouse
Apache 2.0

Web analytics, network flow logs, observability, financial asset valuation, security event & incident management, …

Federated query on data lakes and DBMS

Presto
Apache 2.0

Enterprise analytics on large volumes of data across disparate sources

A short list of reasons why ClickHouse is popular

Understands SQL

Runs on bare metal to cloud

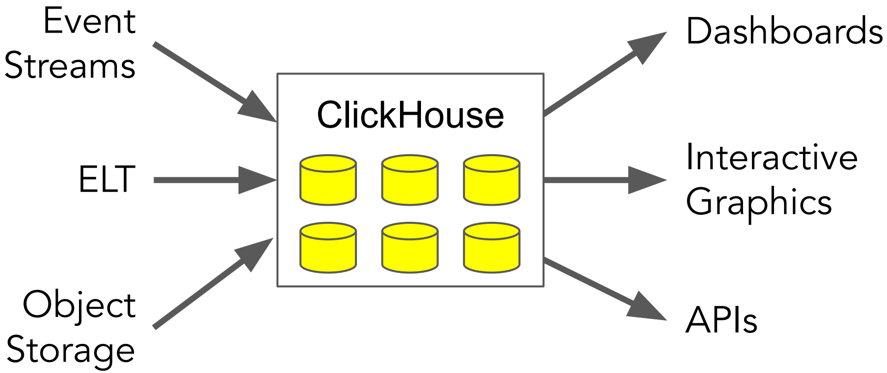
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

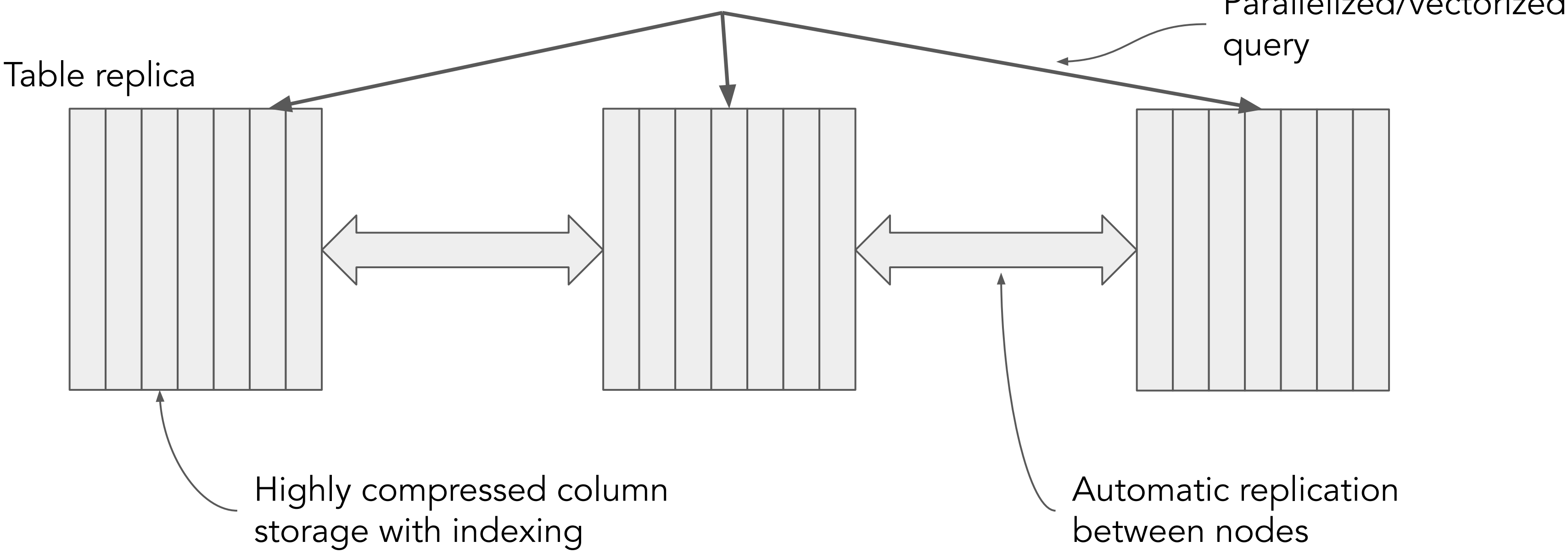
Scales to many petabytes

Is Open source (Apache 2.0)



It’s the core engine for real-time analytics

ClickHouse optimizes for fast response on large datasets



```
SELECT host, avg(idle)
FROM vmstat GROUP BY host
```

Parallelized/vectorized query

Table replica

Highly compressed column storage with indexing

Automatic replication between nodes

…And supports [many] dozens of input formats

```
INSERT INTO some_table Format <format>

      TabSeparated
      TabSeparatedWithNames
      CSV
      CSVWithNames
      CustomSeparated
      Values
      JSON
      JSONEachRow
      Protobuf
      Parquet
      ...
```

It also has great support for time-ordered data

Date -- Precision to day

DateTime -- Precision to second

DateTime64 -- Precision to nanosecond

BI tools like Grafana like DateTime values

toYear(), toMonth(), toWeek(), toDayOfWeek, toDay(), toHour(), ...

toStartOfYear(), toStartOfQuarter(), toStartOfMonth(), toStartOfHour(), toStartOfMinute(), …, toStartOfInterval()

toYYYYMM()

toYYYYMMDD()

toYYYYMMDDhhmmsss()

And many more!

Grafana pairs well with ClickHouse for observability apps

Understands time series data

Simple installation

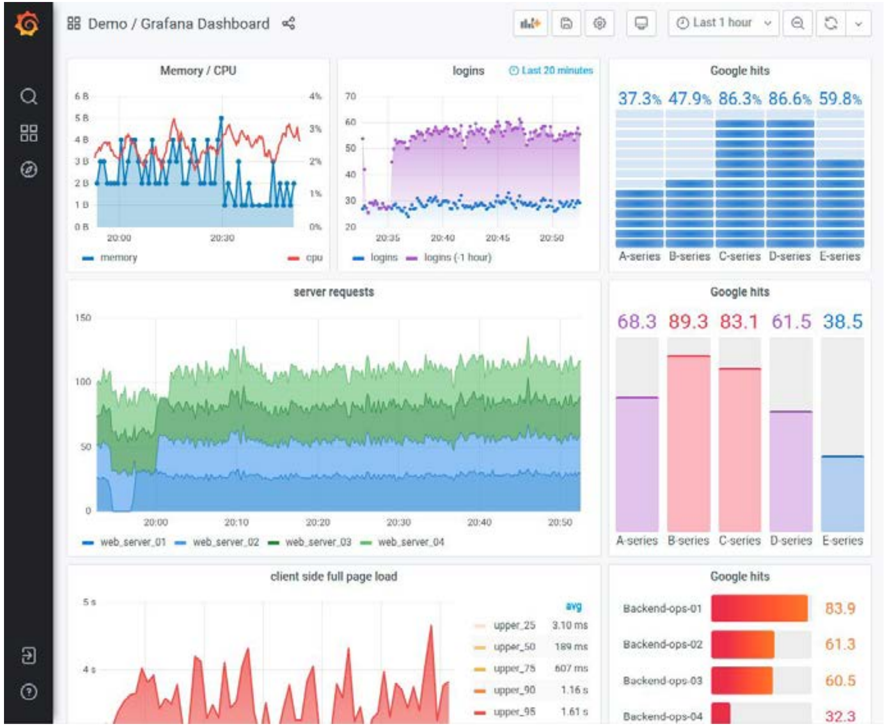
Many data sources

Lots of display plugins

Interactive zoom-in/zoom-out

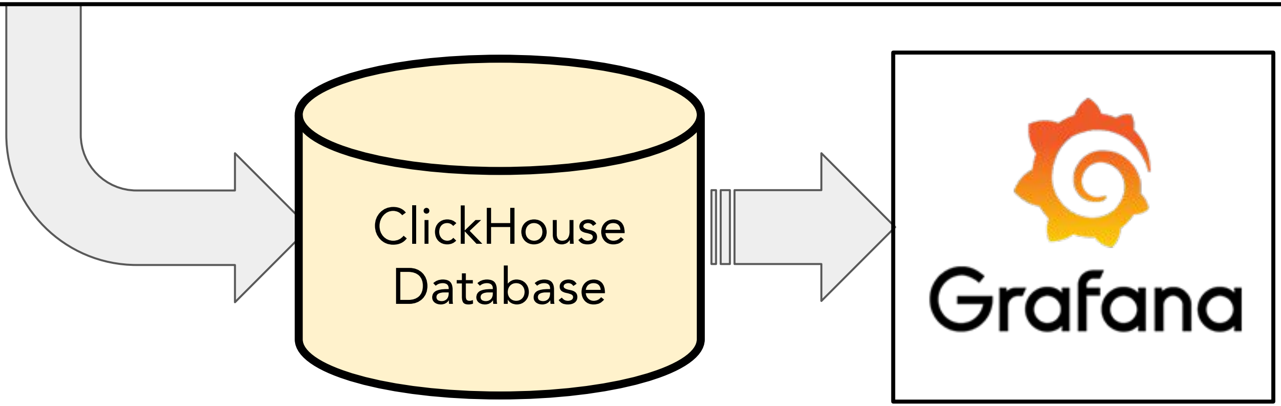
Great for monitoring dashboards

Is open source (AGPL 3.0)



Sooo…How do we ingest vmstat data and display it?

```
$ vmstat 1 -n
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 166912 2645740  36792 3360652    0    0     3   101    1    1  2  1 98  0  0
 1  0 166912 2645360  36792 3360652    0    0     0     0 1182 3986  7  1 93  0  0
```



Step 1: Generate vmstat data

```
#!/usr/bin/env python3
import datetime, json, socket, subprocess
host = socket.gethostname()
with subprocess.Popen(['vmstat', '-n', '1'], stdout=subprocess.PIPE) as proc:
    proc.stdout.readline() # discard first line
    header_names = proc.stdout.readline().decode().split()
    values = proc.stdout.readline().decode()
    while values != '' and proc.poll() is None:
        dict = {}
        dict['timestamp'] = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        dict['host'] = host
        for (header, value) in zip(header_names, values.split()):
            dict[header] = int(value)
        print(json.dumps(dict), flush=True)
        values = proc.stdout.readline().decode()
```

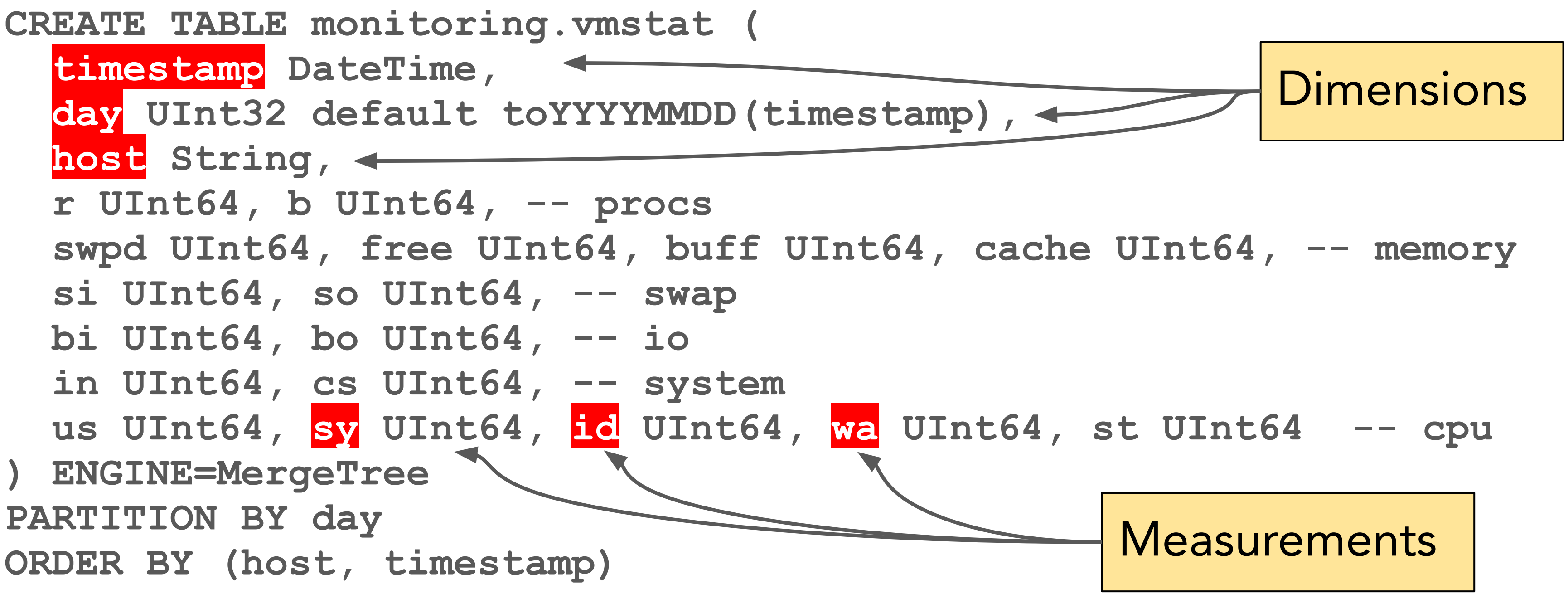
Here’s the output

```
{"timestamp": "2023-01-22 18:13:16", "host": "logos3", "r": 0, "b":
0, "swpd": 166912, "free": 2523688, "buff": 41412, "cache": 3408292,
"si": 0, "so": 0, "bi": 3, "bo": 101, "in": 1, "cs": 0, "us": 2,
"sy": 1, "id": 98, "wa": 0, "st": 0}

{"timestamp": "2023-01-22 18:13:17", "host": "logos3", "r": 0, "b":
0, "swpd": 166912, "free": 2523696, "buff": 41412, "cache": 3408316,
"si": 0, "so": 0, "bi": 0, "bo": 216, "in": 1214, "cs": 4320, "us":
1, "sy": 1, "id": 98, "wa": 0, "st": 0}

{"timestamp": "2023-01-22 18:13:18", "host": "logos3", "r": 0, "b":
0, "swpd": 166912, "free": 2527120, "buff": 41412, "cache": 3408572,
"si": 0, "so": 0, "bi": 0, "bo": 0, "in": 1172, "cs": 4162, "us": 2,
"sy": 1, "id": 98, "wa": 0, "st": 0}
```

Step 2: Design a ClickHouse table to hold data



```
CREATE TABLE monitoring.vmstat (
  timestamp DateTime,
  day UInt32 default toYYYYMMDD(timestamp),
  host String,
  r UInt64, b UInt64, -- procs
  swpd UInt64, free UInt64, buff UInt64, cache UInt64, -- memory
  si UInt64, so UInt64, -- swap
  bi UInt64, bo UInt64, -- io
  in UInt64, cs UInt64, -- system
  us UInt64, sy UInt64, id UInt64, wa UInt64, st UInt64  -- cpu
) ENGINE=MergeTree
PARTITION BY day
ORDER BY (host, timestamp)
```

Dimensions

Measurements

Step 3: Load data into ClickHouse

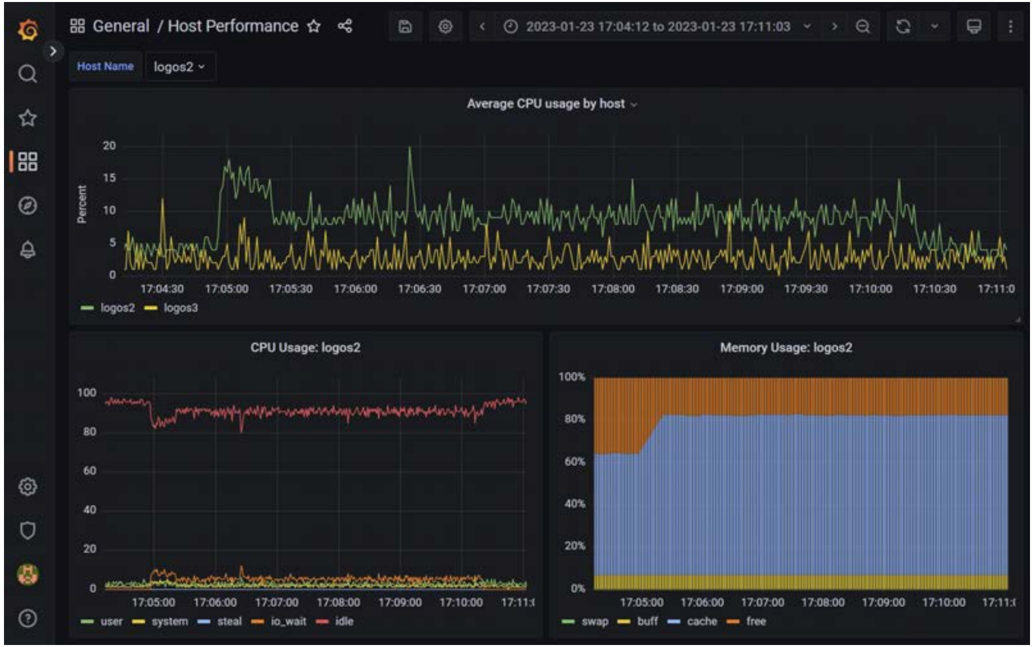
```
INSERT INTO vmstat Format JSONEachRow
```

E.g.

```
INSERT='INSERT%20INTO%20vmstat%20Format%20JSONEachRow'
cat vmstat.dat | curl -X POST --data-binary @- \
  "http://logos3:8123/?database=monitoring&query=${INSERT}"
```

(Or a Python script)

Step 4: Build a Grafana dashboard to show results



Host Name logos2

Altinity plugin for ClickHouse

ClickHouse data source for Grafana

Altinity

Step 5: Go crazy!

```
SELECT host, count() AS loaded_minutes
FROM (
    SELECT
        toStartOfMinute(timestamp) AS minute, host, avg(100 - id) AS load
    FROM monitoring.vmstat
    WHERE timestamp > (now() - toIntervalDay(1))
    GROUP BY minute, host HAVING load > 25
)
GROUP BY host ORDER BY loaded_minutes DESC
```

```
┌─host───┬─loaded_minutes─┐
│ logos3 │              6 │
│ logos2 │              5 │
└────────┴────────────────┘
```

2 hosts had > 25% load for at least a minute in the last 24 hours

DEMO TIME!

Where’s the code?

https://github.com/Altinity/clickhouse-sql-examples

More software to build monitoring on ClickHouse

OPEN SOURCE

Event streaming

- Apache Kafka
- Apache Pulsar
- Vectorized Redpanda

ELT

- Apache Airflow
- Rudderstack

Rendering/Display

- Apache Superset
- Cube.js
- Grafana

Client Libraries

- C++ - ClickHouse CPP
- Golang - ClickHouse Go
- Java - ClickHouse JDBC
- Javascript/Node.js - Apla
- ODBC - ODBC Driver for ClickHouse
- Python - ClickHouse Driver, ClickHouse SQLAlchemy

More client library links HERE

Kubernetes

- Altinity Operator for ClickHouse

Where can I find out more?

ClickHouse official docs – https://clickhouse.com/docs/

Grafana official docs – https://grafana.com/docs/grafana

Altinity Blog – https://altinity.com/blog/

Altinity Youtube Channel – https://www.youtube.com/channel/UCE3Y2lDKl_ZfjaCrh62onYA

Altinity Knowledge Base – https://kb.altinity.com/

Meetups, other blogs, and external resources. Use your powers of Search!

Conf42:
DevSecOps 2023

Thank you and have fun!

Robert Hodges - Altinity
https://altinity.com

Altinity.Cloud
Altinity Stable Builds for ClickHouse
Altinity Kubernetes Operator for ClickHouse