Monitoring OVH: 300k servers, 28 DCs... and one Metrics platform

Horacio Gonzalez
@LostInBrittany

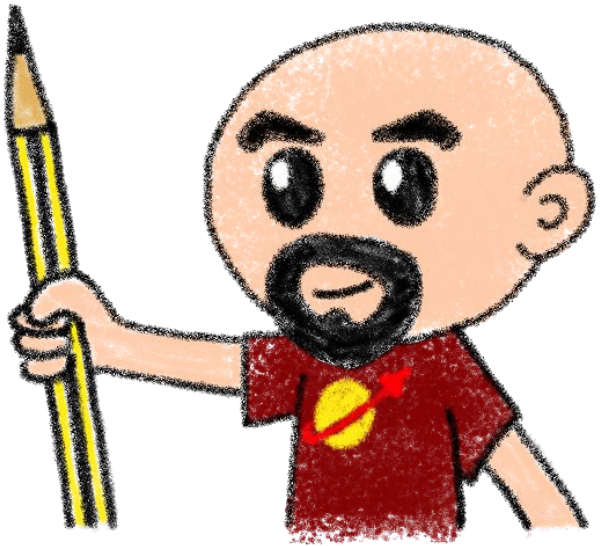
Who are we?

Introducing myself and introducing OVH

Horacio Gonzalez

@LostInBrittany

Spaniard lost in Brittany, developer, dreamer and all-around geek



OVH
Team DevRel

DevFest du Bout du Monde

Finist Devs

WARP 10 MEETUP

Google Developers Experts 2019
Web Technologies
GDE
Flutter

OVH : Key Figures

1.3M Customers worldwide in 138 Countries
1.5 Billions euros investment over five years
28 Datacenters (growing)
350k Dedicated Servers
200k Private cloud VMs running
650k Public cloud Instances created in a month
20TB bandwidth capacity
35 Points of presence
4TB Anti DDoS capacity
Hosting capacity : 1.3M Physical Servers

+ 2 500 Employees in 19 countries
20 Years of Innovation

OVH: Our solutions

Once upon a time...

Because I love telling tales

This talk is about a tale...

A true one nevertheless

And as in most tales

It begins with a mission

And a band of heroes

Engulfed into the adventure

They fight against mishaps

And all kind of foes

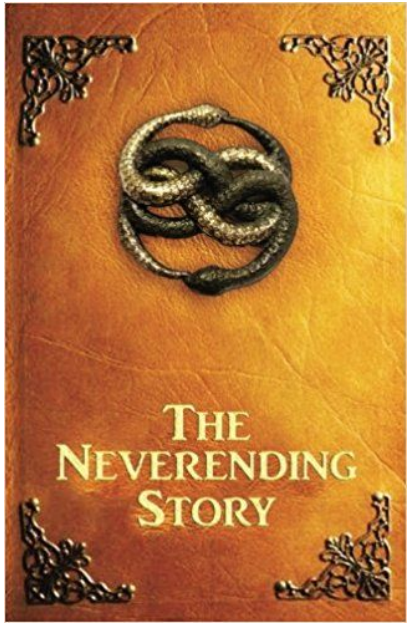
They build mighty fortresses

Pushing the limits of possible

And defend them day after day

Against all odds

But we don't know yet the end

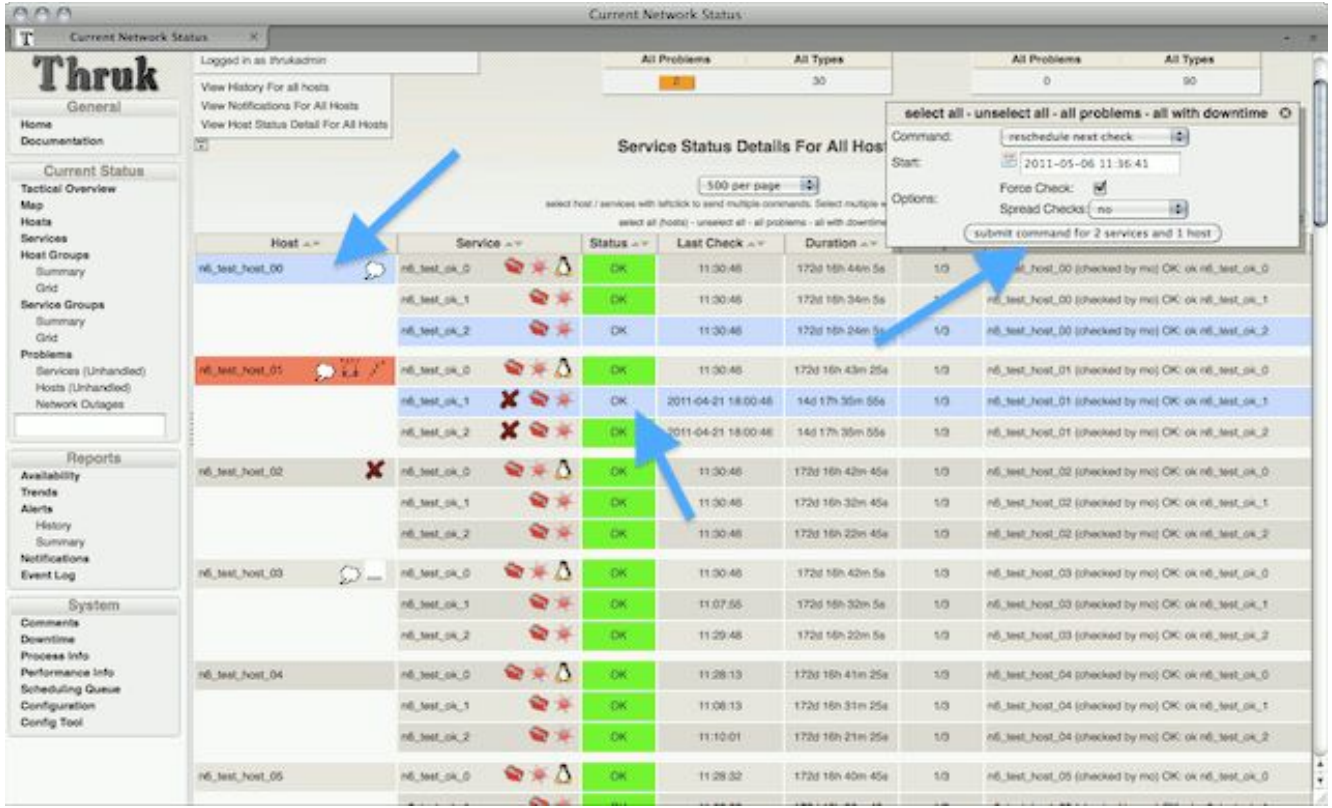


Because this tale isn't finished yet

It begins with a mission

Build a metrics platform for OVH

A long time ago...

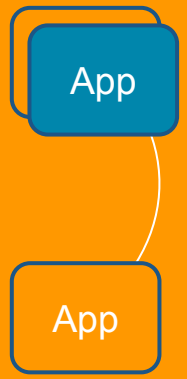


A long time ago...

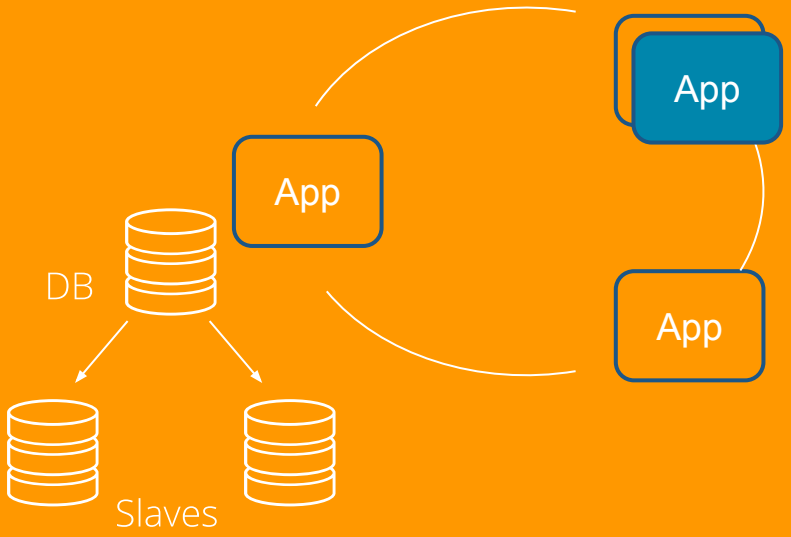
Monitoring: **Does** the system works?

Moving from monolith to µservices

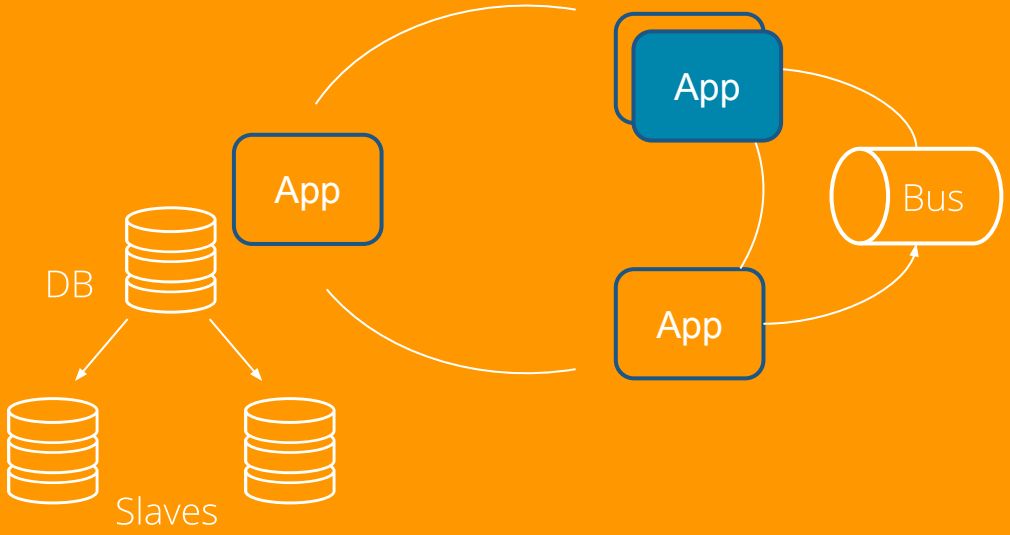
Moving from monolith to µservices



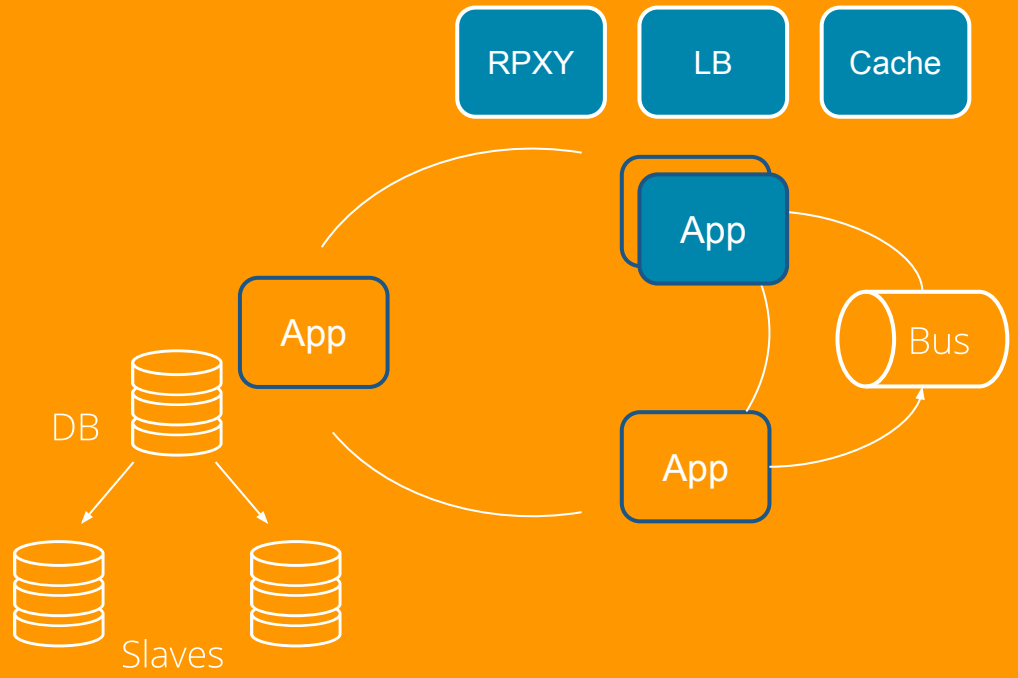
Moving from monolith to µservices



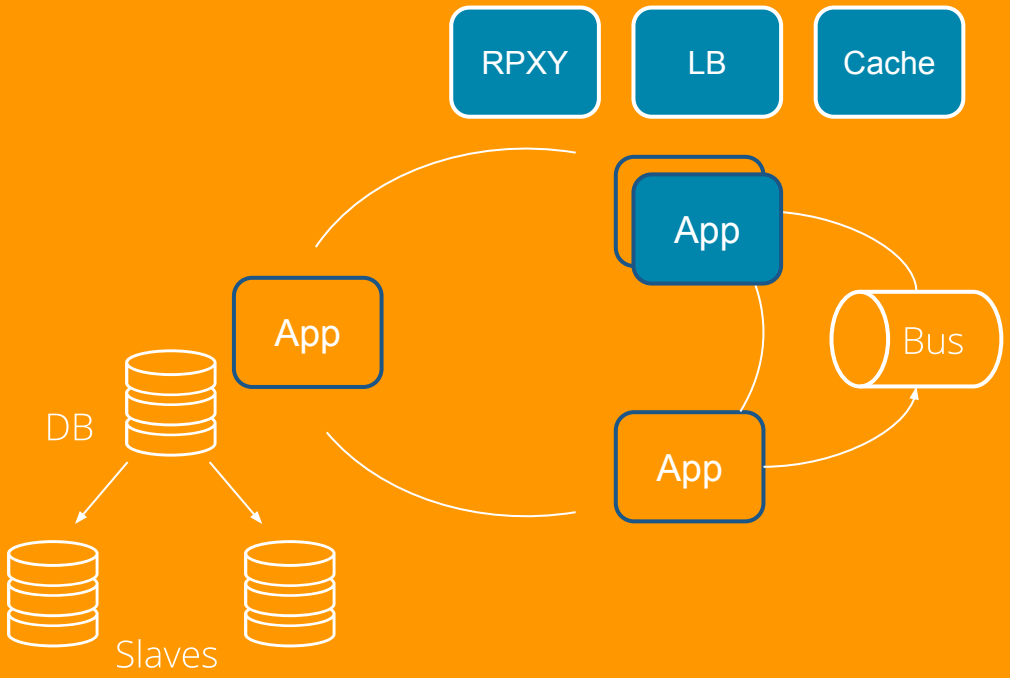
Moving from monolith to µservices



Moving from monolith to µservices

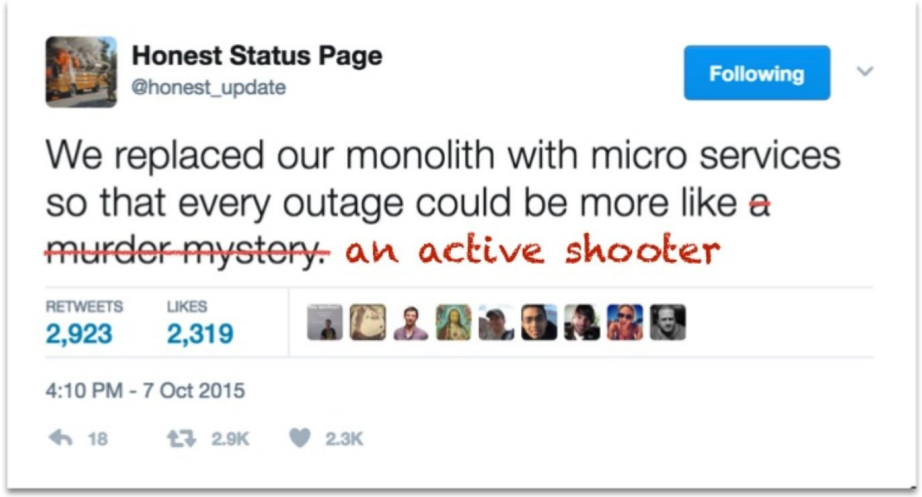


What could go wrong?



Microservices are a distributed system

The Microservices Complexity Paradox — Joyent

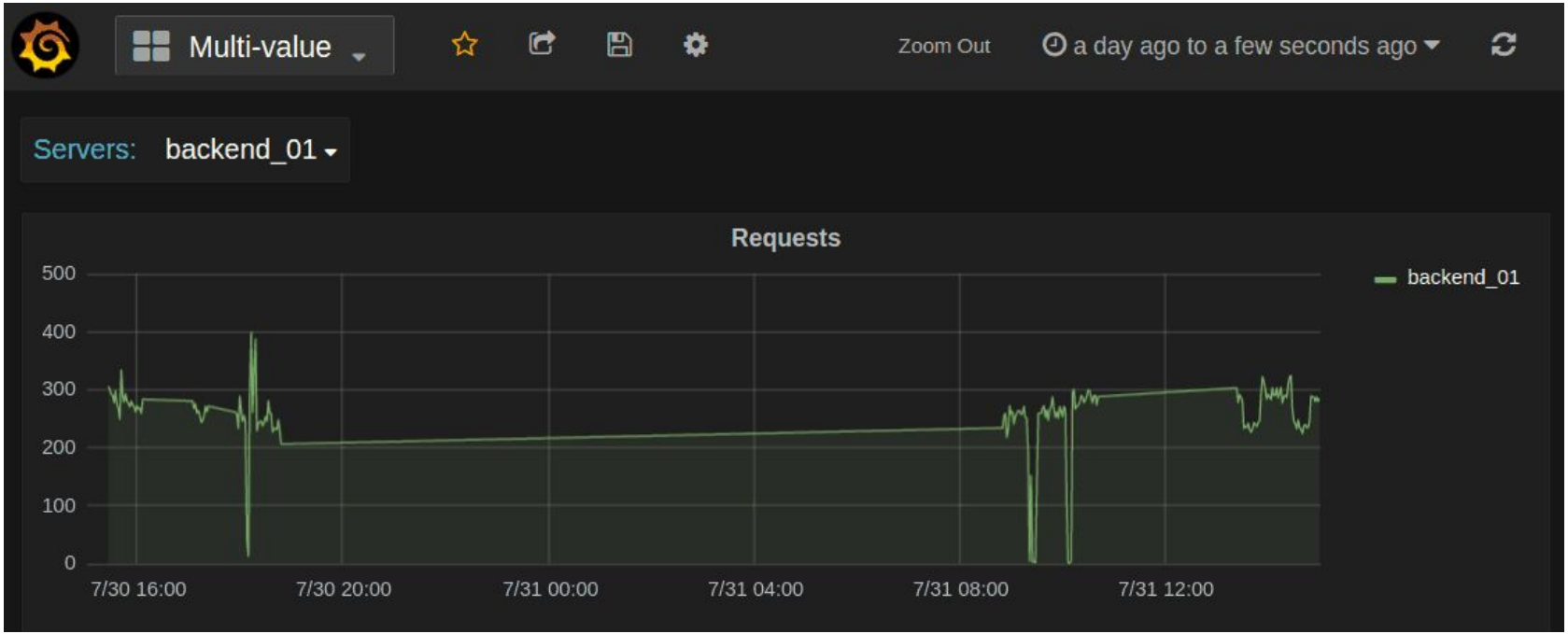


GOTO 2017 • Debugging Under Fire: Keep your Head when Systems have Lost their Mind • Bryan Cantrill

We need to have insights

Observability : Understand **how** it works

OVH decided go metrics-oriented



A metrics platform for OVH

For **all** OVH

Building OVH Metrics

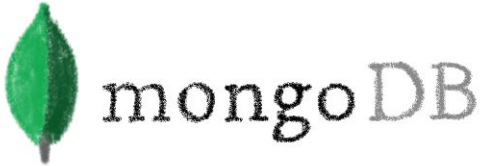
One Platform to unify them all,
One Platform to find them,
One Platform to bring them all
and in the Metrics monitor them

What is OVH Metrics?

Managed Cloud Platform
for Time Series

OVH monitoring story

We had lots of partial solutions...



OVH monitoring story

One Platform to unify them all

What should we build it on?

OVH monitoring story

Including a really big

OpenTSDB drawbacks

OpenTSDB RowKey Design

metrics timestamp tagk1 tagv1 tagk2 tagv2

OpenTSDB Rowkey design flaws

- .*regex.* => full table scans
- High cardinality issues (Query latencies)

We needed something able to manage

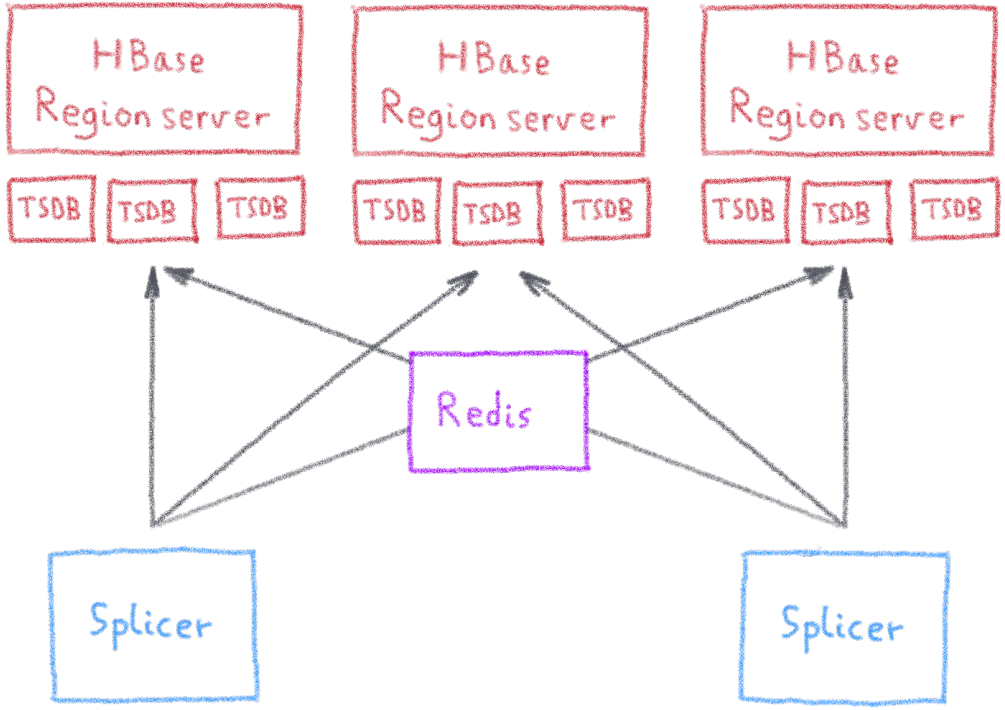
hundreds of millions time series

OpenTSBD didn't scale for us

OpenTSDB other flaws

- Compaction (or append writes)
- /api/query : 1 endpoint per function?
- Asynchronous
- Unauthenticated
- ...

Scaling OpenTSDB



Metrics needs

First **need**:

To be **massively** scalable

Analytics is the key to success

Fetching data is only the tip of the iceberg

Analysing metrics data

To be scalable, analysis must be done in the database, not in user's computer

Metrics needs

Second **need**:

To have **rich query** capabilities

Enter Warp 10...

Open-source
Time series
Database

More than a Time Series DB

Warp 10 is a software platform that

- Ingests and stores time series
- Manipulates and analyzes time series

Manipulating Time Series with Warp 10

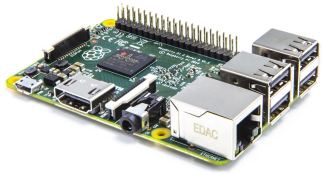
A true Time Series analysis toolbox

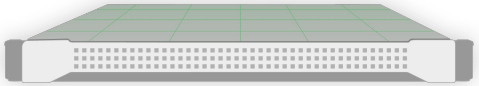
- Hundreds of functions
- Manipulation frameworks
- Analysis workflow

Manipulating Time Series with Warp 10

A Time Series manipulation language

Did you say scalability?





From the smallest to the largest...

More Warp 10 goodness

- Secured & multi tenant
- In memory Index
- No cardinality issues
- Lockfree ingestion
- WarpScript Query Language
- Support more data types
- Synchronous (transactions)
- Better Performance
- Better Scalability
- Versatile

  (standalone, distributed)

OVH Observability Metrics Platform

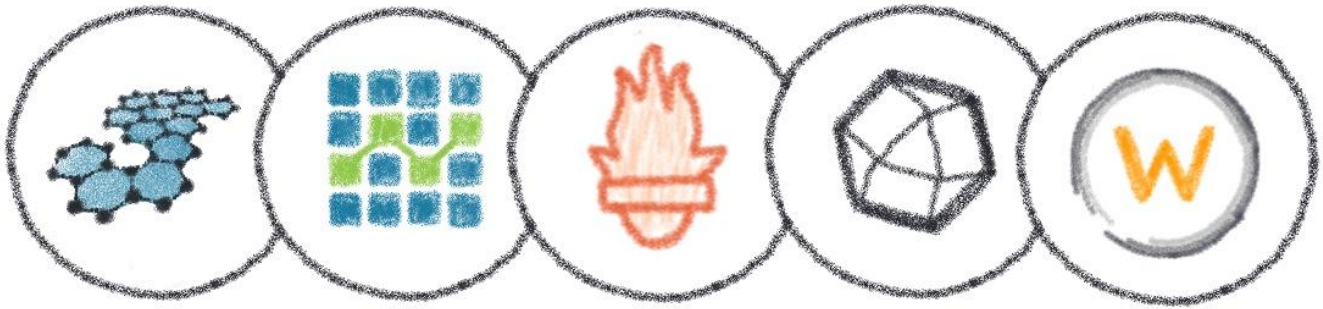
Building an ecosystem

From Warp 10 to OVH Metrics

What protocols should we support?

Who must do the effort?

Open source monitoring tools



Open source monitoring tools

Open source monitoring tools

Open source monitoring tools

Open source monitoring tools

Open source monitoring tools

Open source monitoring tools

Why choose?

Let’s support all of them!

Metrics Platform

Metrics Platform

https://

graphite

influx

opentsdb

prometheus

Warp10

tsl

...

.<region>.metrics.ovh.net

Metrics Platform

graphite

influx

https:// opentsdb .<region>.metrics.ovh.net

prometheus

Warp10

tsl

...

TSL

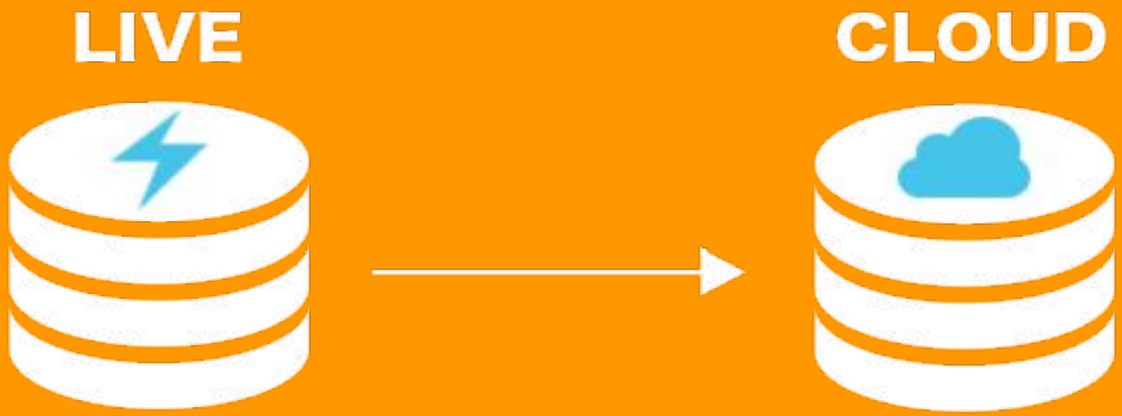
```
select("cpu.usage_system")
  .where("cpu~cpu[0-7]*")
  .last(12h)
  .sampleBy(5m,max)
  .groupBy(mean)
  .rate()
```

github.com/ovh/tsl

Metrics Live

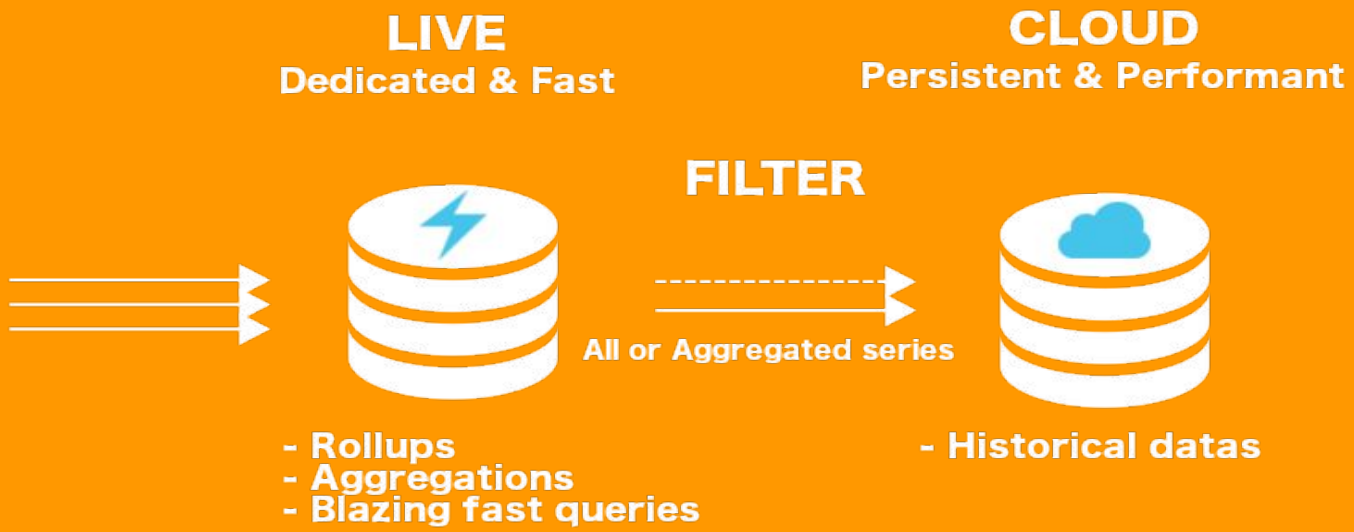
In-memory, high-performance Metrics instances

In-memory: Metrics live



millions of writes/s

In-memory: Metrics live

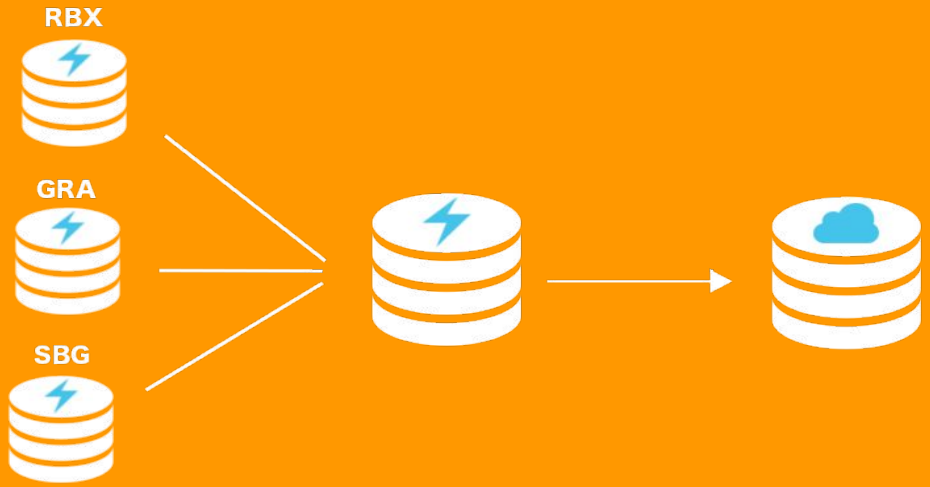


In-memory: Metrics live

STAGE 1
Short retention - hours
Fine grained monitoring
Raw data

STAGE 2
Short retention - days
Consolidated aggregations
Global infra monitoring

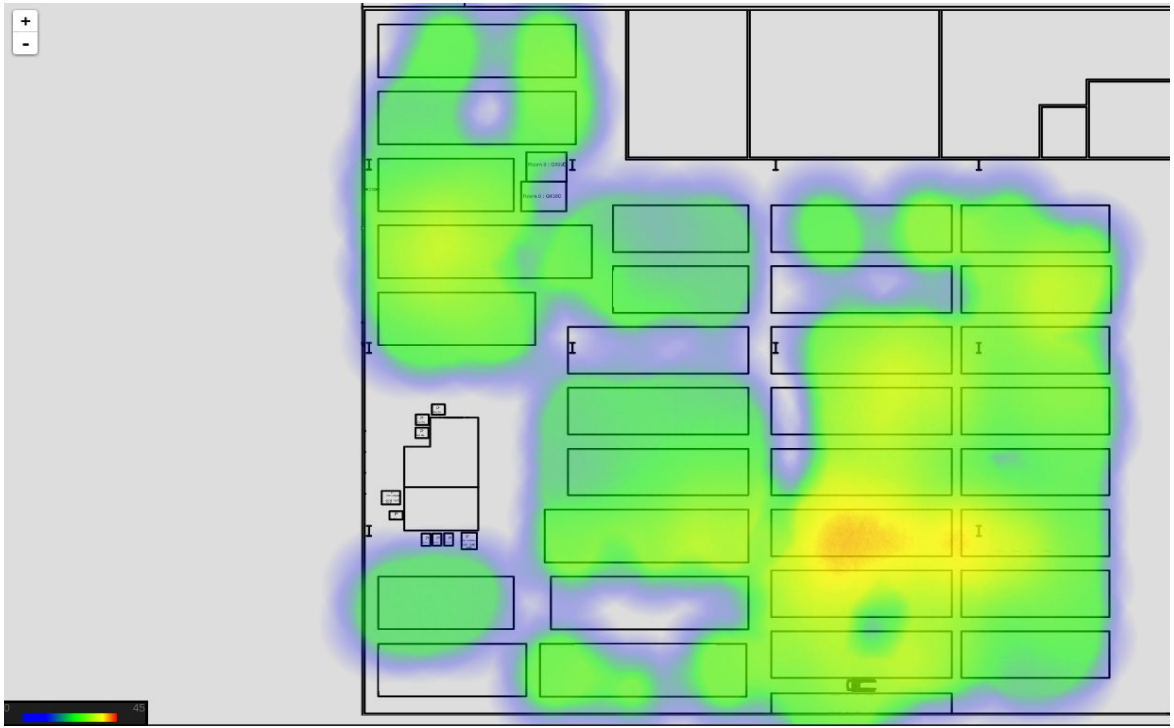
STAGE 3
Customer metrics
Historical datas



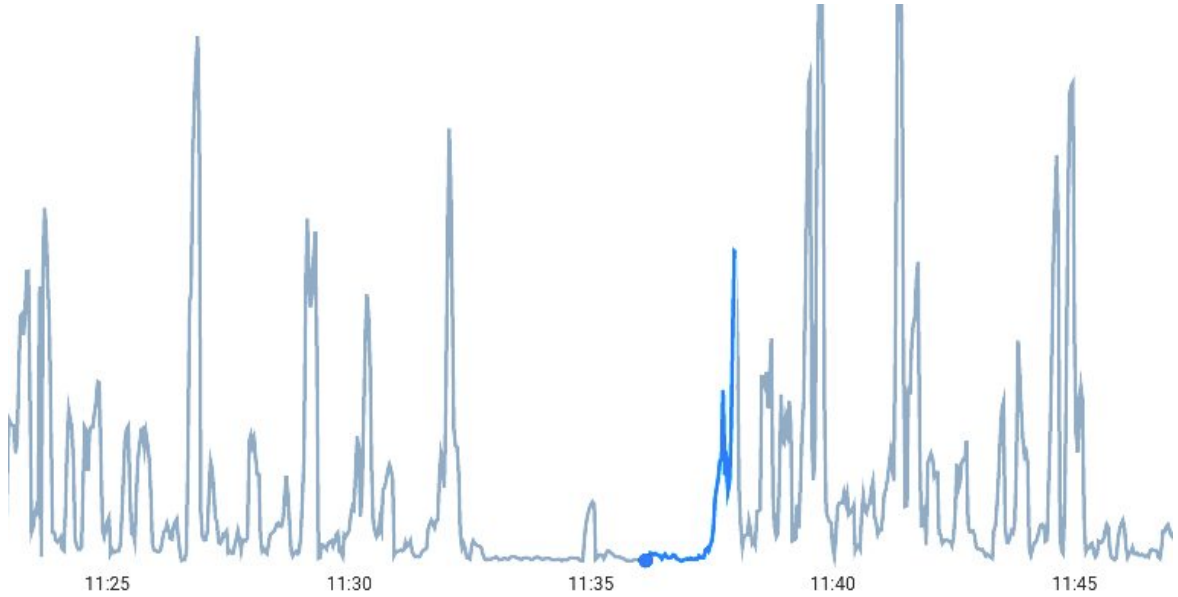
Monitoring is only the beginning

OVH Metrics answer to many other use cases

Graveline rack's temperature



Even medical research...



Metrics' Pattern Detection feature helped Gynaecology Research
to prove patterns on perinatal mortality

Use cases families

- Billing (e.g. bill on monthly max consumption)
- Monitoring (APM, infrastructure,appliances,...)
- IoT (Manage devices, operator integration, ...)
- Geo Location (Manage localized fleets)

Use cases

- DC Temperature/Elec/Cooling map
- Pay as you go billing (PCI/IPLB)
- GSCAN
- Monitoring
- ML Model scoring (Anti-Fraude)
- Pattern Detection for medical applications

SREing Metrics

With a great power comes a great responsibility

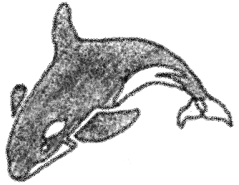
Metrics's metrics

432.000.000.000

datapoints / day

Our stack overview

- More than 650 machines operated by 5 people
- >95% dedicated servers
- No Docker, only SystemD
- Running many Apache projects:
  - Hadoop
  - HBase
  - Zookeeper
  - Flink
- And Warp 10

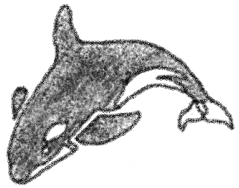


Warp 10™ by SenX

Our biggest Hadoop cluster

200 datanodes

2.3 PB of **capacity**
8.5Gb/s of **bandwidth**



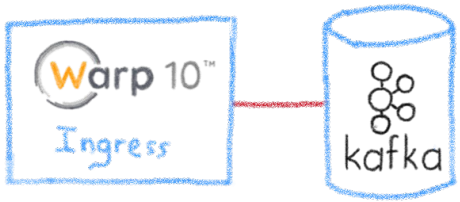
~60k regions of 10Gb

1.5M of **writes**/s
3M of **reads**/s

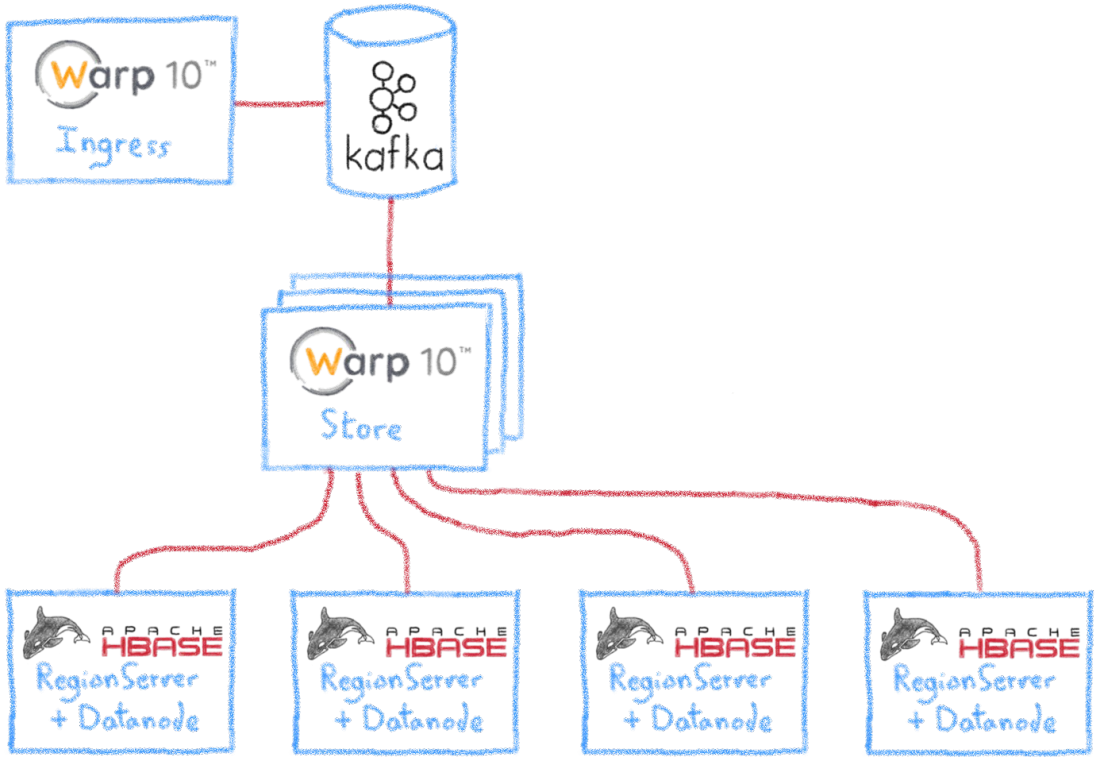
Hadoop need a lot of ❤

Warp10: distributed overview

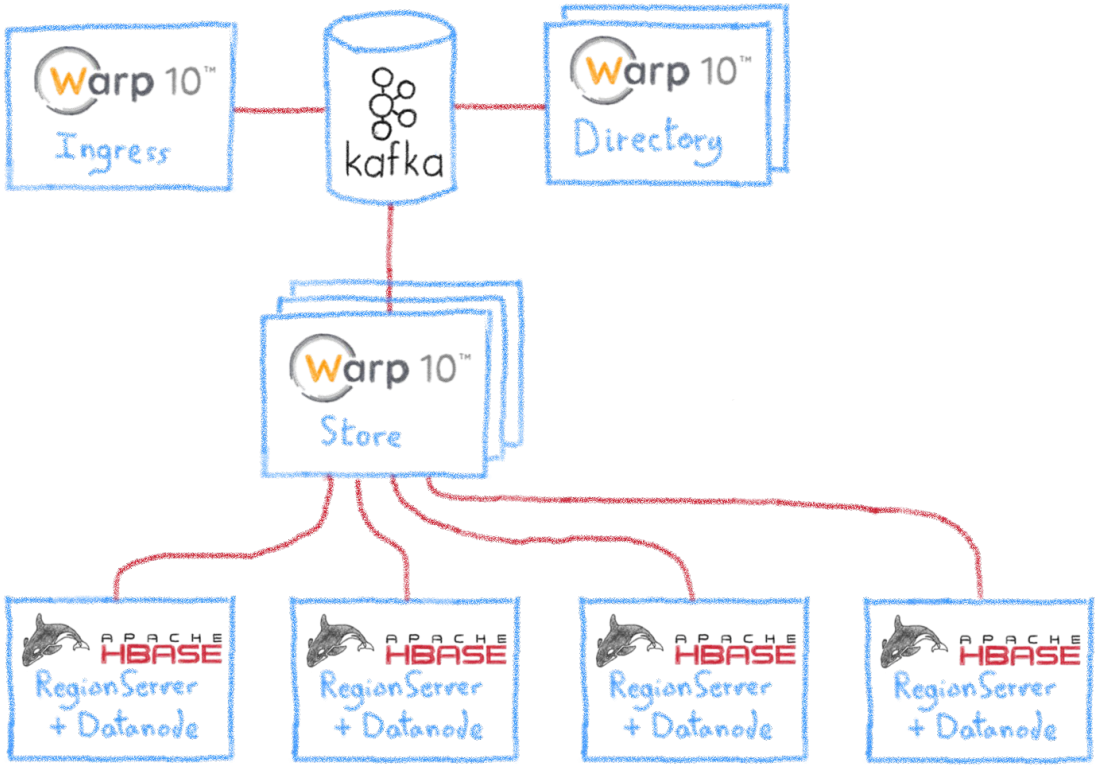
Warp10: distributed overview



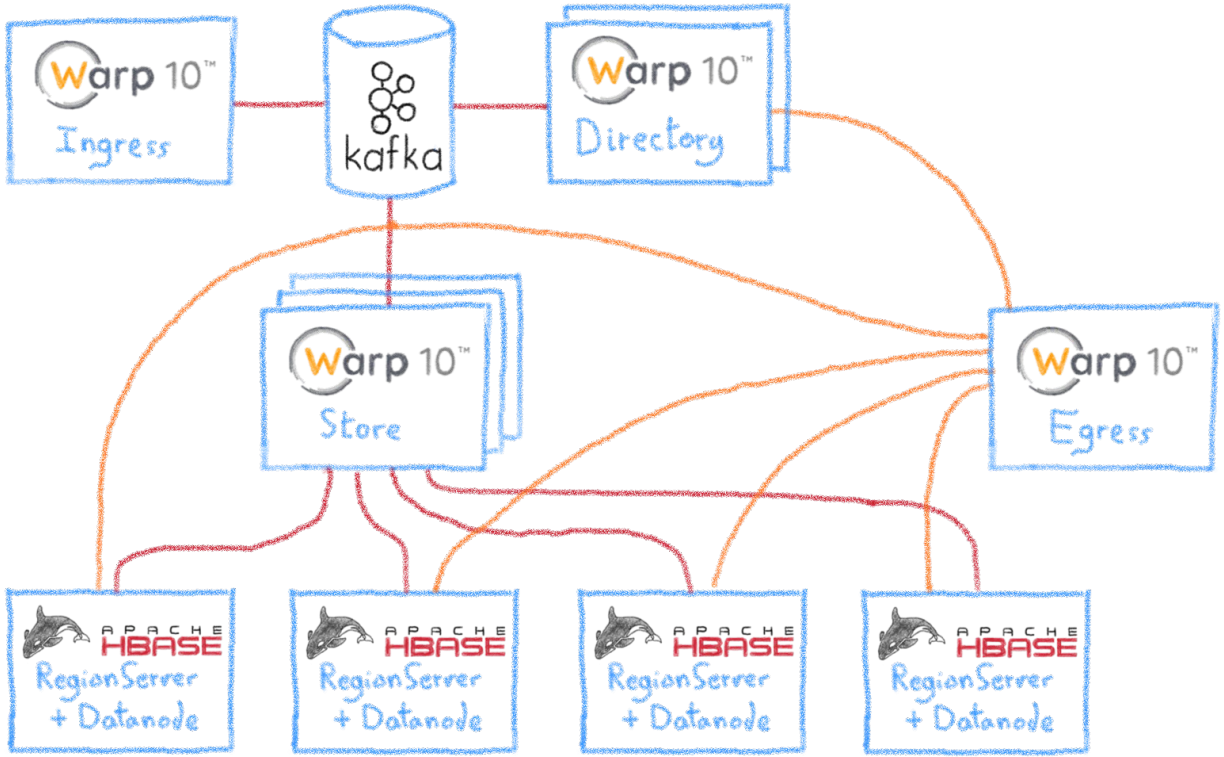
Warp10: distributed overview



Warp10: distributed overview



Warp10: distributed overview



Hadoop nodes

Most of the nodes are the following:

- 16 to 32 cores
- 64 to 128 GB of RAM 😱
- 12 to 16 TB

But, we also have some huge nodes:

- 2x 20 cores (xeon gold)
- 320 GB of RAM 😱 😱
- 12x 4TB of Disk

Warp10 nodes

Ingress (cpu-bound):

- 32 cores
- 128 GB of RAM 😱

Directory (ram-bound):

- 48 cores
- 512 GB of RAM 😱😱😱

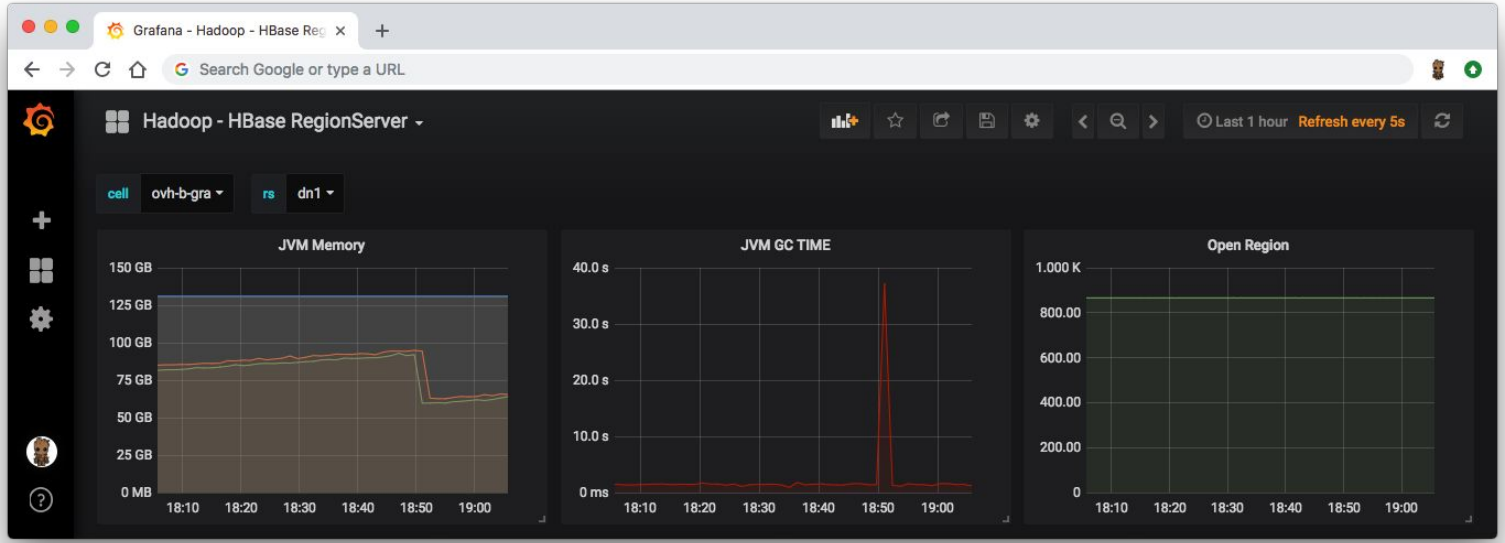
Egress (cpu-bound):

- 32 cores 😱
- 128 GB of RAM

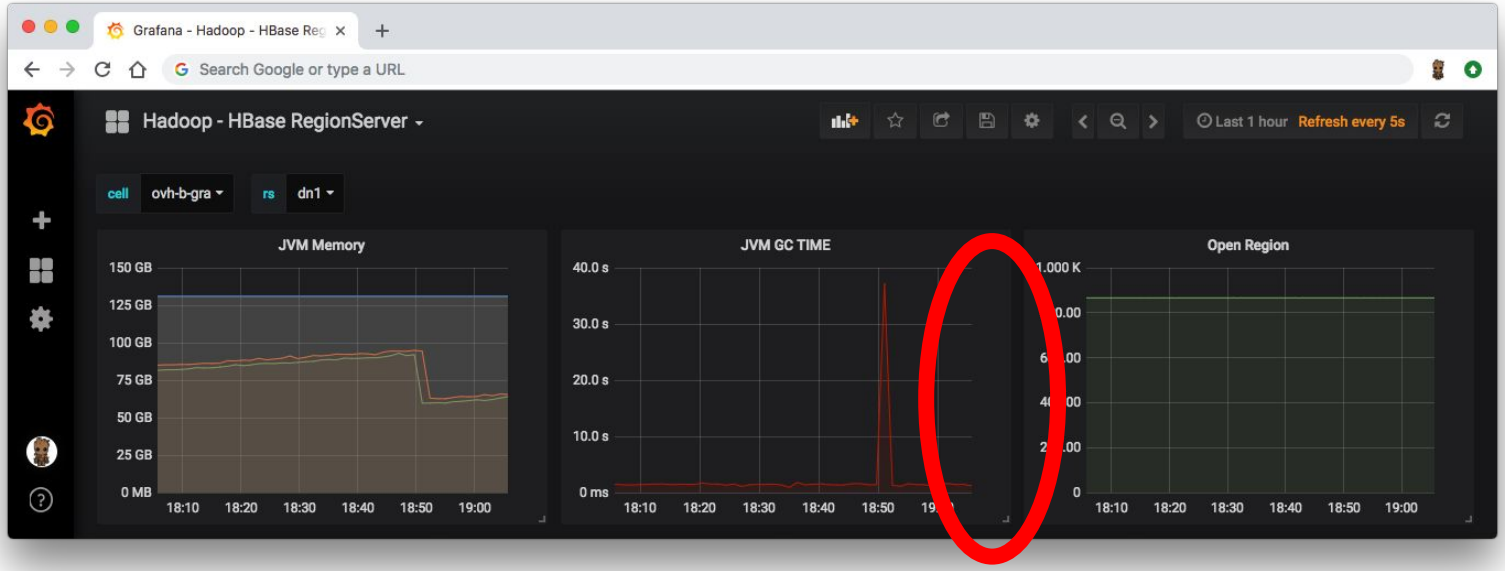
Store (cpu-bound):

- 32 cores 😱
- 128 GB of RAM

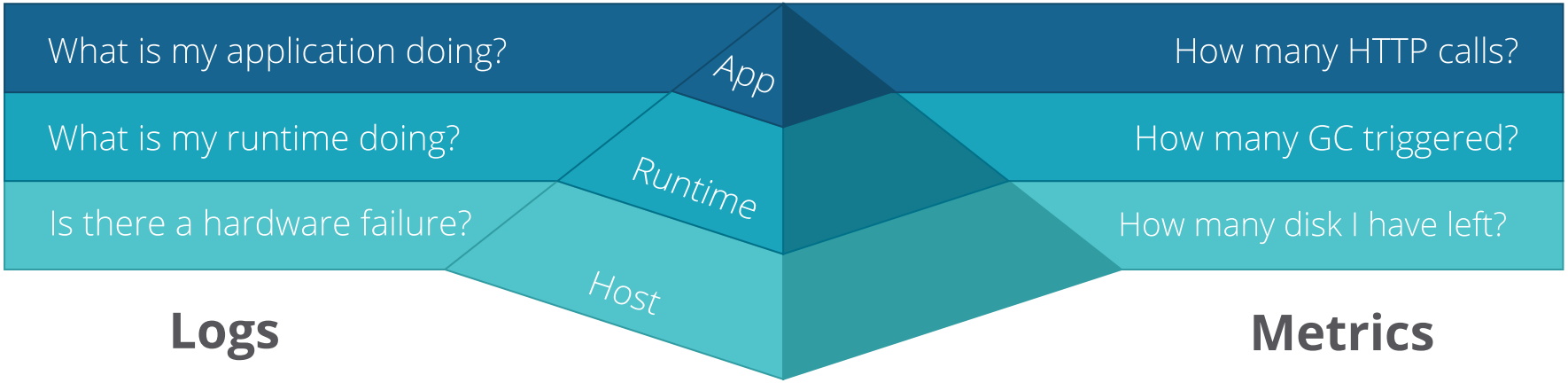
Why you should care?



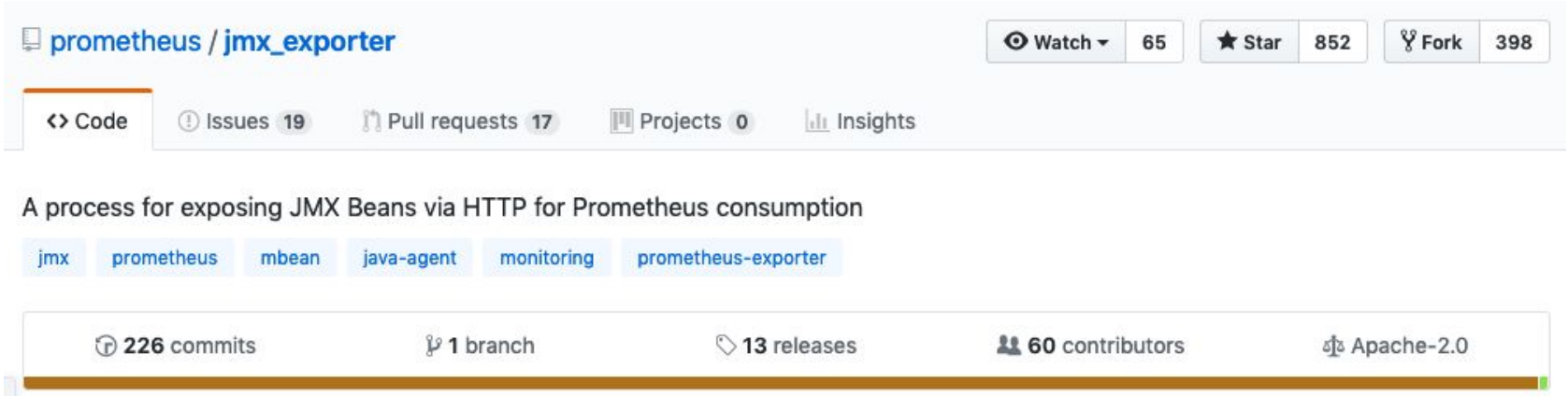
Why you should care? (>30s) 😱



The only way to optimize: measure



Monitoring JVM with metrics



Monitoring JVM with metrics

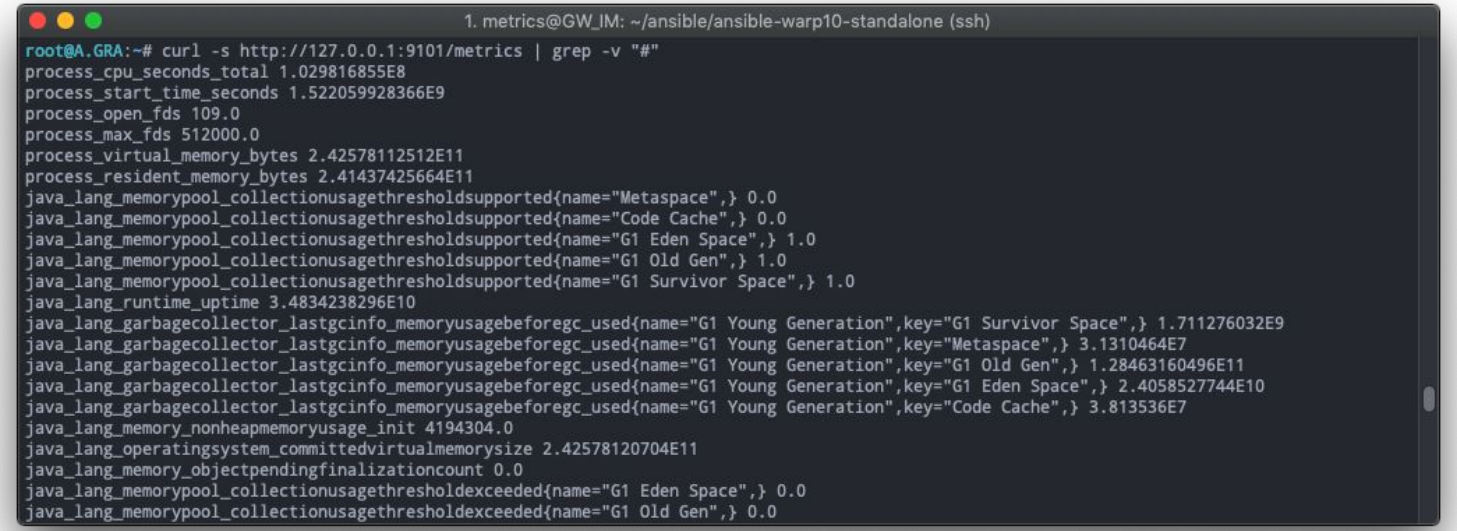
Running

To run as a javaagent download the jar and run:

```
java -javaagent:./jmx_prometheus_javaagent-0.11.0.jar=8080:config.yaml -jar yourJar.jar
```

Metrics will now be accessible at http://localhost:8080/metrics

Monitoring JVM with metrics



Monitoring JVM with metrics

Monitoring JVM with metrics

Tuning G1 is hard 😢

```
-Xms800g -Xmx800g \
-XX:+UseG1GC -XX:G1HeapRegionSize=64m \
-XX:MaxGCPauseMillis=500 \
-XX:ParallelGCThreads=36 \
-XX:ConcGCThreads=9 \
-XX:+UnlockExperimentalVMOptions \
-XX:G1NewSizePercent=8 \
-XX:G1MaxNewSizePercent=8 \
-XX:+ParallelRefProcEnabled \
-XX:+PerfDisableSharedMem \
-XX:-ResizePLAB \
-XX:-ReduceInitialCardMarks \
-XX:G1RSetRegionEntries=4096 \
-XX:InitiatingHeapOccupancyPercent=65 \
-XX:G1HeapWastePercent=10 \
-XX:G1MixedGCCountTarget=16 \
```

Tuning G1 is hard 😢😢

```
-Xms800g -Xmx800g \
-XX:+UseG1GC -XX:G1HeapRegionSize=64m \
-XX:MaxGCPauseMillis=500 \
-XX:ParallelGCThreads=36 \
-XX:ConcGCThreads=9 \
-XX:+UnlockExperimentalVMOptions \
-XX:G1NewSizePercent=8 \
-XX:G1MaxNewSizePercent=8 \
-XX:+ParallelRefProcEnabled \
-XX:+PerfDisableSharedMem \
-XX:-ResizePLAB \
-XX:-ReduceInitialCardMarks \
-XX:G1RSetRegionEntries=4096 \
-XX:InitiatingHeapOccupancyPercent=65 \
-XX:G1HeapWastePercent=10 \
-XX:G1MixedGCCountTarget=16 \
```

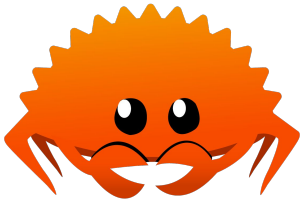
```
-XX:+HeapDumpOnOutOfMemoryError \
-XX:HeapDumpPath=/opt/warp/logs/heap.dump \
-verbose:gc \
-XX:+PrintGC \
-XX:+PrintGCDetails \
-XX:+PrintGCDateStamps \
-XX:+PrintGCTimeStamps \
-Xloggc:/opt/warp/logs/gc.log \
-XX:+UseGCLogFileRotation \
-XX:NumberOfGCLogFiles=10 \
-XX:GCLogFileSize=10M \

-XX:+AlwaysPreTouch \
-XX:+UseTransparentHugePages \
-XX:+UseNUMA \
-XX:-UseBiasedLocking \
```

Our programming stack

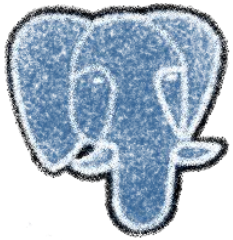
- We mostly use garbage collected languages as
  - Go
  - Java
  - JavaScript

Our programming stack

However, we are using non-garbage collected languages as Rust when needed



Our friends for µservices



PostgreSQL

We ❤ open-source

Code contribution:

- https://github.com/ovh/beamium
- https://github.com/ovh/noderig
- https://github.com/ovh/tsl
- https://github.com/ovh/ovh-warp10-datasource
- https://github.com/ovh/ovh-tsl-datasource
- ...

Involved in:

- Warp10 community
- Apache Hbase/Flink development
- Prometheus/InfluxData discussions
- TS Query Language Working group

Conclusion

That's all folks!