

# Real-time-first metrics

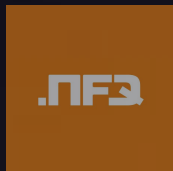
Aurelijus Banelis



# Aurelijus Banelis

Software developer  
[aurelijus.banelis.lt](mailto:aurelijus.banelis.lt)  
[aurelijus@banelis.lt](mailto:aurelijus@banelis.lt)

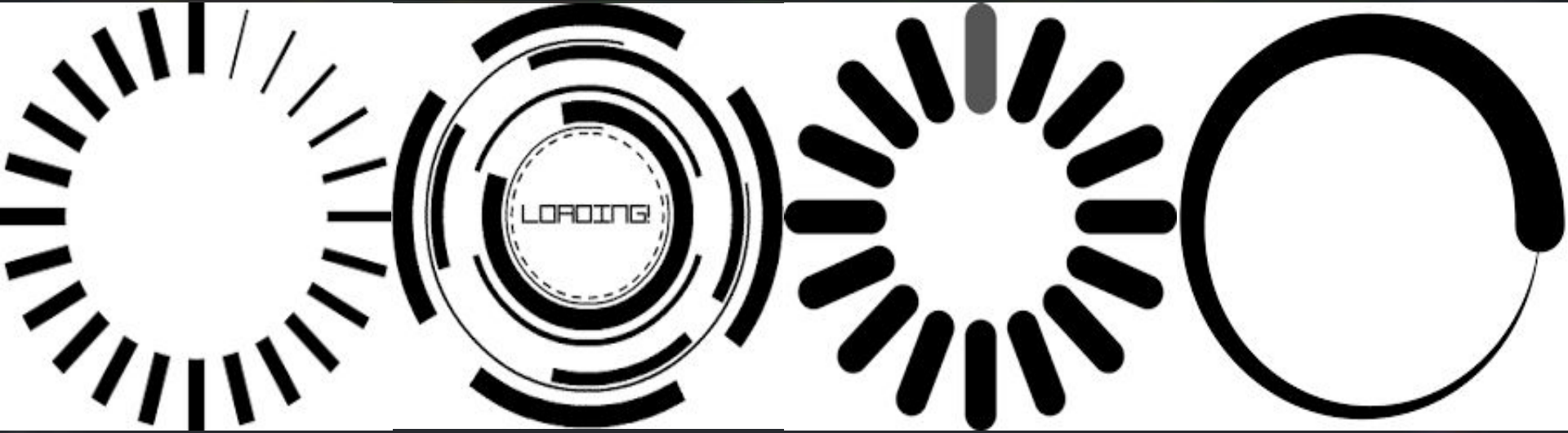
PGP public key      rsa2048/ **539B6203**  
Key fingerprint = 130D C446 1F1A 2E50 D6E3  
                    3DA8 3202 05E7 539B 6203





# Real-time-first metrics

# What is going on?



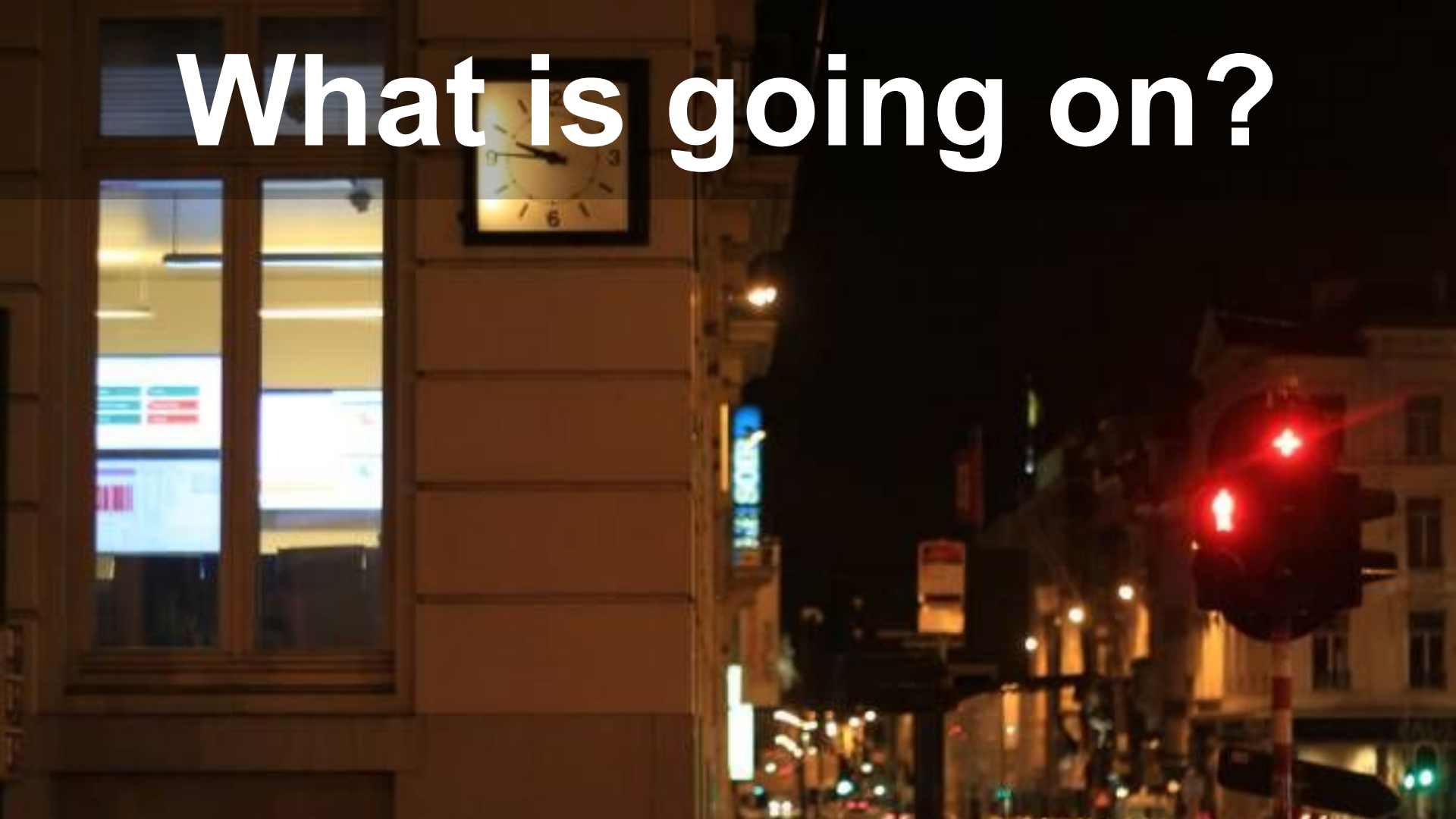
# Until it is too late

## **504 Gateway Time-out**

---

nginx/1.11.1

# What is going on?



# Metrics can tell you





**Metrics**

**Compromises choosing  
types of monitoring**

**Standard**

**Examples of tools to  
capture and store**

**Exotic**

**Examples of nonstandard  
ways to monitor**



**Metrics**

**Compromises choosing  
types of monitoring**

**Standard**

**Examples of tools to  
capture and store**

**Exotic**

**Examples of nonstandard  
ways to monitor**

# Logs vs metrics

## cpu

Total CPU utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the **CPUs** section and per application usage at the **Applications Monitoring** section.

Keep an eye on **iowait** (0%). If it is constantly high, your disks are a bottleneck and they slow your system down.

Another important metric worth monitoring, is **softirq** (0%). A constantly high percentage of softirq may indicate network drivers issues.



SQS (Ohio)

SQS (Oregon)

S3 (Montreal)

S3 (N. California)

S3 (N. Virginia)

S3 (Ohio)

S3 (Oregon)



# Durable vs real time

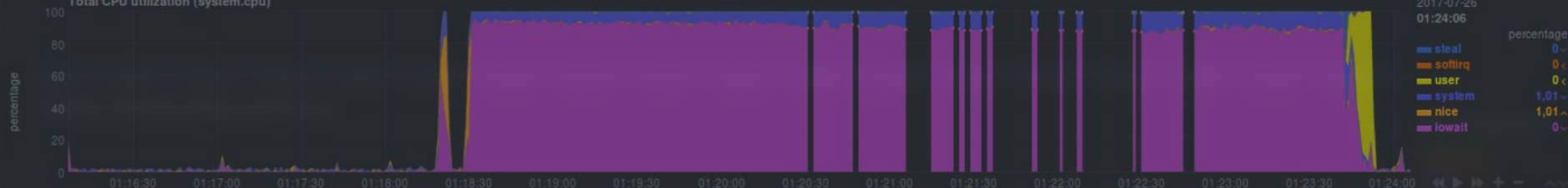
cpu

Total CPU utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the **CPUs** section and per application usage at the **Applications Monitoring** section.

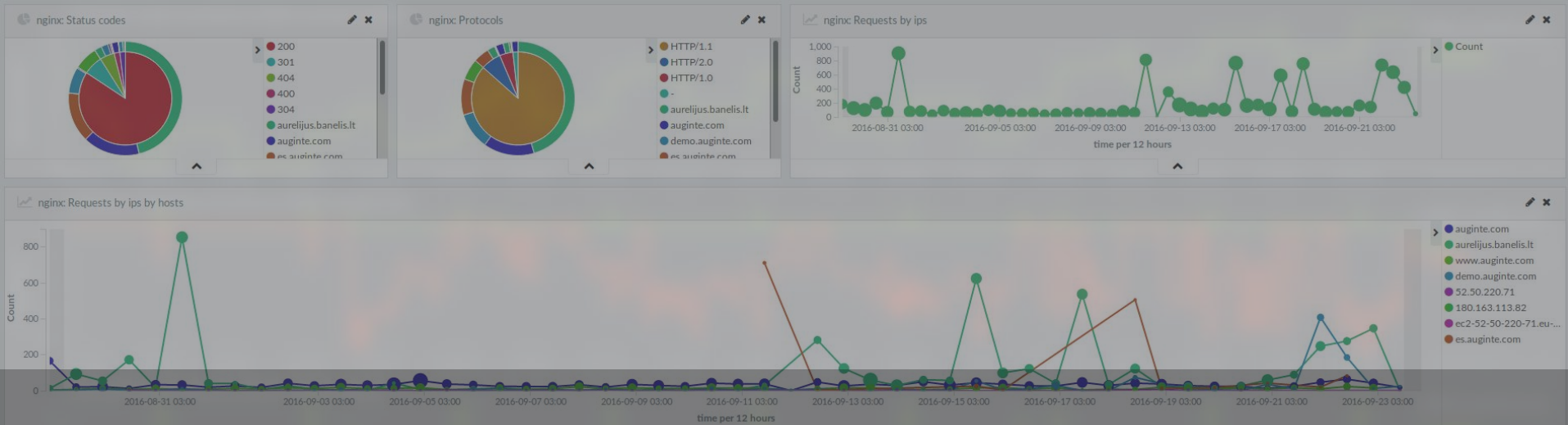
Keep an eye on **iowait** (0%). If it is constantly high, your disks are a bottleneck and they slow your system down.

Another important metric worth monitoring, is **softirq** (0%). A constantly high percentage of softirq may indicate network drivers issues.

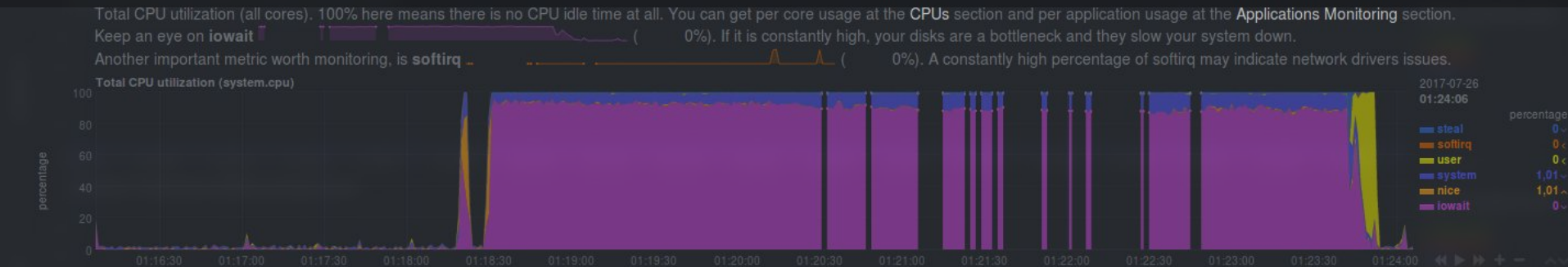
Total CPU utilization (system.cpu)







# Long term vs real time



# Amazon Glacier Vaults

Create Vault

Delete Vault

Settings



Filter By Name:

Name ▲

Inventory Last Updated

Size (as of last inventory)

# of Archives (as of last inventory)

Jul 25, 2017 8:11:33 PM

1.8 GB

1436

Jul 25, 2017 7:38:10 M

2.1 GB

89

Jun 13, 2017 5:45:37 AM

55.3 kB

1

# Unlimited vs

# Fixed resources

`mem-buf-size()`

Type:

number (bytes)

Default:

163840000

**Description:** Use this option if the option `reliable()` is set to `yes`. This option contains the size of the messages in bytes that is used in the memory part of the disk buffer. It replaces the old `log-fifo-size()` option. It does not inherit the

**Metrics**

**Details, aggregation,  
durability, price**

**Standard**

**Examples of tools to  
capture and store**

**Exotic**

**Examples of nonstandard  
ways to monitor**

**Metrics**

Details, aggregation,  
durability, price

**Standard**

Examples of tools to  
capture and store

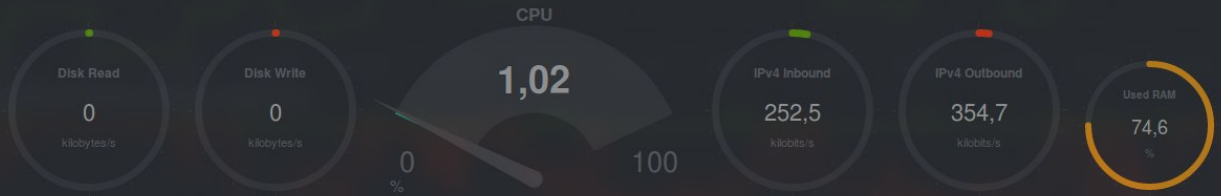
**Exotic**

Examples of nonstandard  
ways to monitor



# System Overview

Overview of the key system metrics.



## cpu

Total CPU utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the [CPUs](#) section and per application usage at the [Applications Monitoring](#) section. Keep an eye on [iowait](#) ( 0,0%). If it is constantly high, your disks are a bottleneck and they slow your system down. An important metric worth monitoring, is [softirq](#) ( 0,00%). A constantly high percentage of softirq may indicate network driver issues.

Total CPU utilization (system.cpu)



## load

Current system load, i.e. the number of processes using CPU or waiting for system resources (usually CPU and disk). The 3 metrics refer to 1, 5 and 15 minute averages. The system calculates this once every 5 seconds. For more information check [this wikipedia article](#)

System Load Average (system.load)



## disk

Total Disk I/O, for all disks. You can get detailed information about each disk at the [Disks](#) section and per application Disk usage at the [Applications Monitoring](#) section.

### System Overview

- cpu
- load
- disk
- ram
- network
- processes
- interrupts
- softirqs
- softnet
- entropy
- ipc semaphores
- uptime

### Memory

### CPUs

### Disks

### IPv4 Networking

### IPv6 Networking

### Firewall (netfilter)

### Network Interfaces

### Applications

### User Groups

### Users

5b285b9762fb

auginte.blog

auginte.blog.mysql

auginte.java.eventsourced

java demoapp

java website

mysql ab it

orientdb demoapp

php ab it

php dc

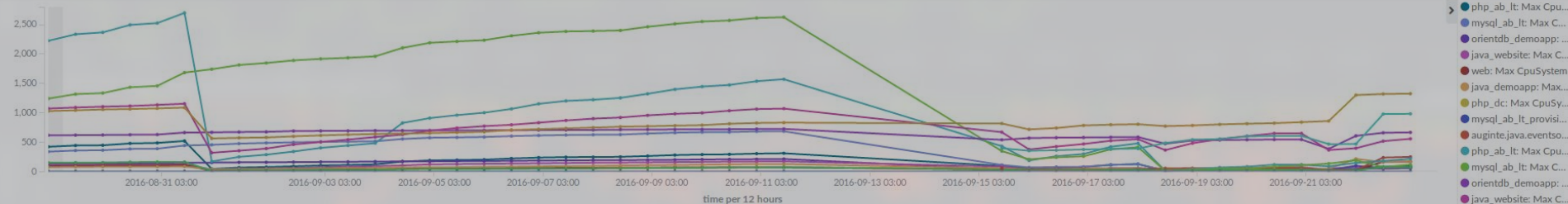
web

Netdata Monitoring

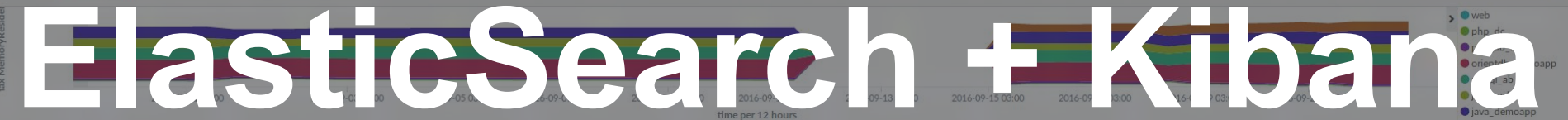
+ add more charts

+ add more alarms

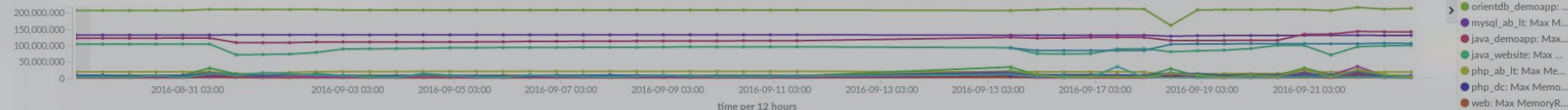
Docker: CPU



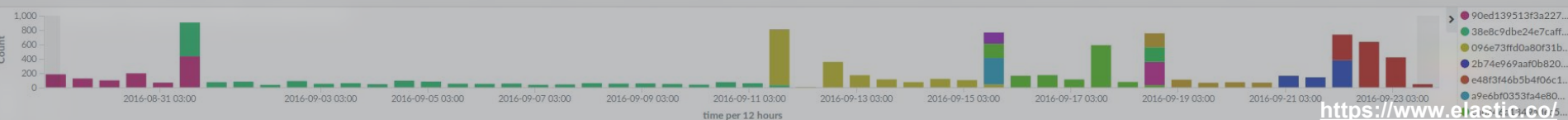
Docker: Memory: Sum



Docker: Memory



Requests by Git revision



2017-07-26 00:12:35 7faa1bac5700 INNODB MONITOR OUTPUT

Per second averages calculated from the last 4 seconds

BACKGROUND THREAD

srv\_master\_thread loops: 0 srv\_active, 0 srv\_shutdown, 6135 srv\_idle  
srv\_master\_thread log flush and writes: 6133

SEMAPHORES

OS WAIT ARRAY INFO: reservation count 4  
OS WAIT ARRAY INFO: signal count 4  
Mutex spin waits 3, rounds 90, OS waits 2  
RW-shared spins 2, rounds 60, OS waits 2  
RW-excl spins 0, rounds 0, OS waits 0  
Spin rounds per wait: 30.00 mutex, 30.00 RW-shared, 0.00 RW-excl

TRANSACTIONS

Trx id counter 7940  
Purge done for trx's n:o < 0 undo n:o < 0 state: running but idle  
History list length 0  
LIST OF TRANSACTIONS FOR EACH SESSION:

---TRANSACTION 0, not started  
MySQL thread id 27, OS thread handle 0x7faa1bac5700, query id 5687  
SHOW ENGINE INNODB STATUS

FILE I/O

I/O thread 0 state: waiting for completed aio requests (insert buffer thread)  
I/O thread 1 state: waiting for completed aio requests (log thread)  
I/O thread 2 state: waiting for completed aio requests (read thread)  
I/O thread 3 state: waiting for completed aio requests (read thread)  
I/O thread 4 state: waiting for completed aio requests (read thread)  
I/O thread 5 state: waiting for completed aio requests (read thread)  
I/O thread 6 state: waiting for completed aio requests (write thread)  
I/O thread 7 state: waiting for completed aio requests (write thread)  
I/O thread 8 state: waiting for completed aio requests (write thread)  
I/O thread 9 state: waiting for completed aio requests (write thread)  
Pending normal aio reads: 0 [0, 0, 0, 0] , aio writes: 0 [0, 0, 0, 0] ,  
ibuf aio reads: 0, log i/o's: 0, sync i/o's: 0  
Pending flushes (fsync) log: 0; buffer pool: 0  
300 OS file reads, 7 OS file writes, 5 OS fsyncs  
0.00 reads/s, 0 avg bytes/read, 0.00 writes/s, 0.00 fsyncs/s

INSERT BUFFER AND ADAPTIVE HASH INDEX

**SHOW ENGINE INNODB STATUS;**



# MySQL



Aurelijus Banelis

Local Guide · Level 4

278 points >

250

300

500

CONTRIBUTE

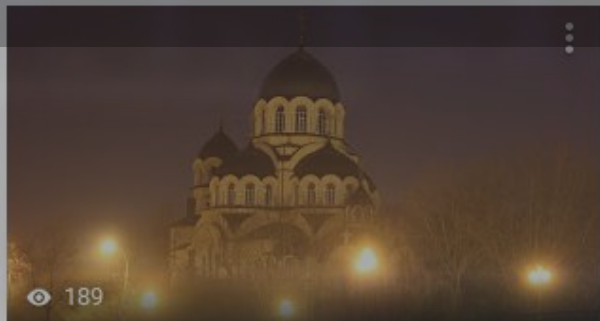
REVIEWS

PHOTOS

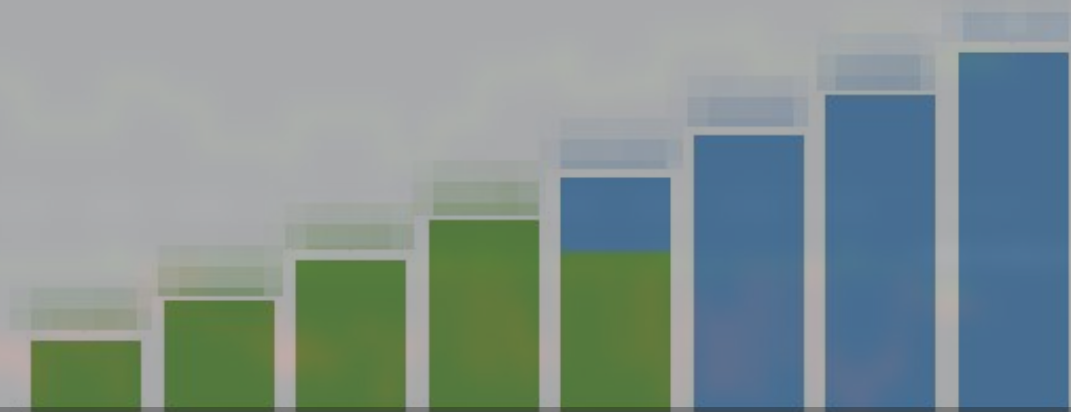
EDITS



Vilniaus Dievo Motinos šventovės „Žemkliai iš dangaus“ ...  
Vytauto g. 21, 06019



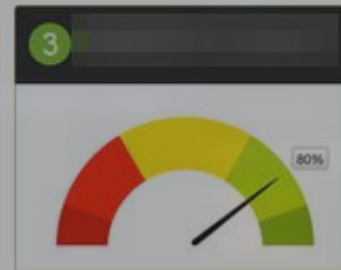
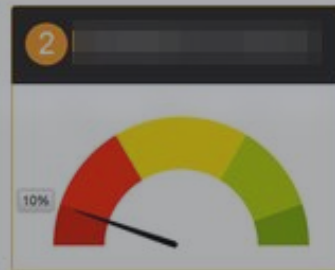
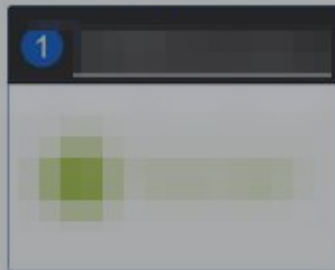
Bistro @ Sapiegos Vilnius Tech Park  
Antakalnio g. 17, Vilnius 10312



Month 12 Month 13 Month 14 Month 15 Month 16 Month 17 Month 18 Month 19

# End user: badges

YOUR STATUS:



**Metrics**

Details, aggregation,  
durability, price

**Standard**

Netdata, Elasticsearch,  
syslog-ng, MySQL status

**Exotic**

Examples of nonstandard  
ways to monitor



**Metrics**

Details, aggregation,  
durability, price

**Standard**

Netdata, Elasticsearch,  
syslog-ng, MySQL status

**Exotic**

**Examples of nonstandard  
ways to monitor**

## The Directed Graph Shell (DGSH)

build passing

The directed graph shell, *dgsh*, allows the expressive expression of efficient big data set and streams processing pipelines using existing Unix tools as well as custom-built components. It is a Unix-style shell allowing the specification of pipelines with non-linear scatter-gather operations. These form a directed acyclic process graph, which is typically executed by multiple processor cores, thus increasing the operation's processing throughput.

You can find a complete introduction, reference documentation, and illustrative examples in the project's [GitHub site](https://github.com/dspinellis/dgsh).

# Split to I/O applications



# μWebSockets

μWS is one of the most lightweight, efficient & scalable WebSocket & HTTP server implementations available. It features an easy-to-use, fully async object-oriented interface and scales to millions of connections using only a fraction of memory compared to the competition. While performance and scalability are two of our top priorities, we consider security, stability and standards compliance paramount. License is zlib/libpng (very permissive & suits commercial applications).

## Real-time applications

- One million WebSockets require ~111mb of user space memory (104 bytes per WebSocket).
- Single-threaded throughput of up to 5 million HTTP req/sec or 20 million WebSocket echoes/sec.
- Linux, OS X, Windows & [Node.js](#) support.
- Runs with raw epoll, libuv or ASIO (C++17-ready).
- Valgrind & AddressSanitizer clean.
- Permessage-deflate, SSL/TLS support & integrates with foreign HTTP(S) servers.
- Multi-core friendly & optionally thread-safe via compiler flag UWS\_THREADSAFE.

build passing

**Metrics**

Details, aggregation,  
durability, price

**Standard**

Netdata, Elasticsearch,  
syslog-ng, MySQL status

**Exotic**

Using only I/O streams,  
fixed memory storage

**If debugging is hard**

**Consider  
monitoring differently**

**Metrics**

Details, aggregation,  
durability, price

**Standard**

**Questions?**

Netdata, Elasticsearch,  
syslog-ng, MySQL status

**Exotic**

Examples of nonstandard  
ways to monitor

# References

- <https://my-netdata.io/>
- <https://www.elastic.co/>
- <https://www.elastic.co/>
-