DATADOG

Container monitoring challenges

Xavier Vello - Datadog

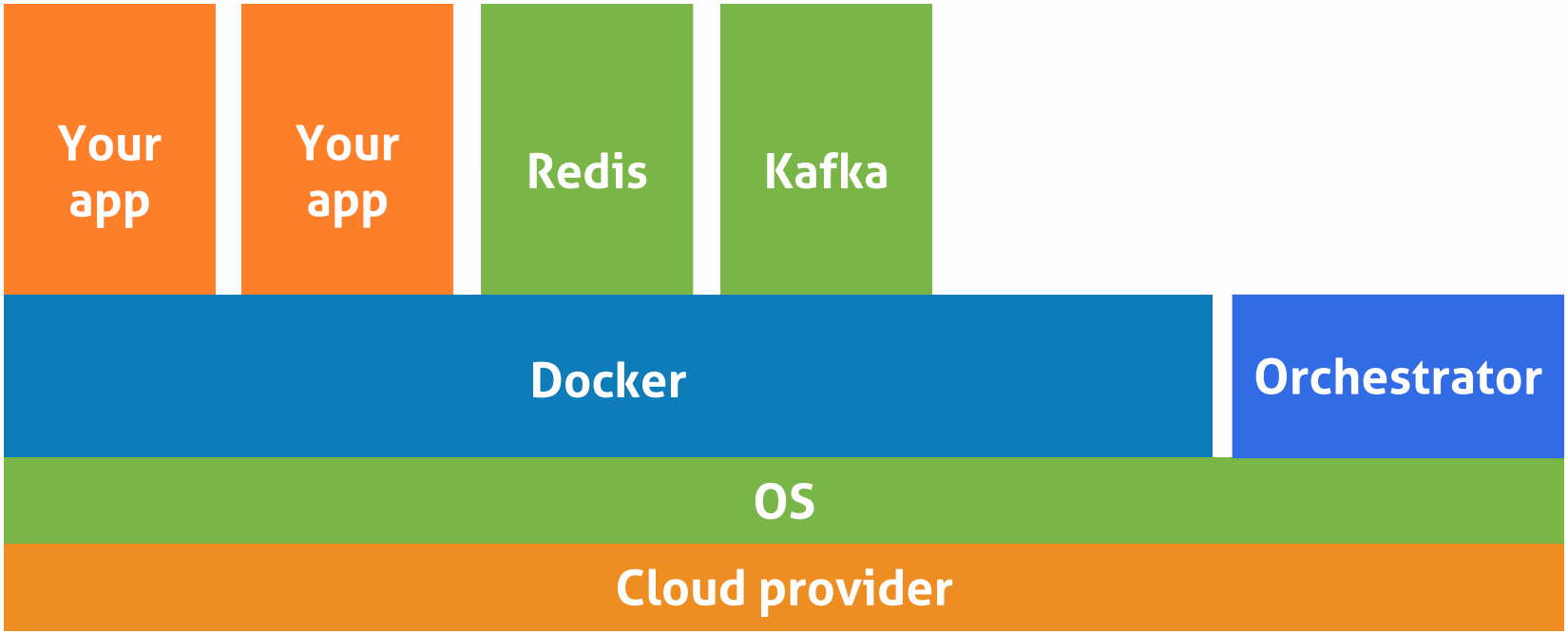
whois xvello.net

- Former Debian Maintainer
- (sporadic) KDE contributor
- Industrial Engineer for 5 years
- Software Engineer at Datadog
- Container Monitoring Team

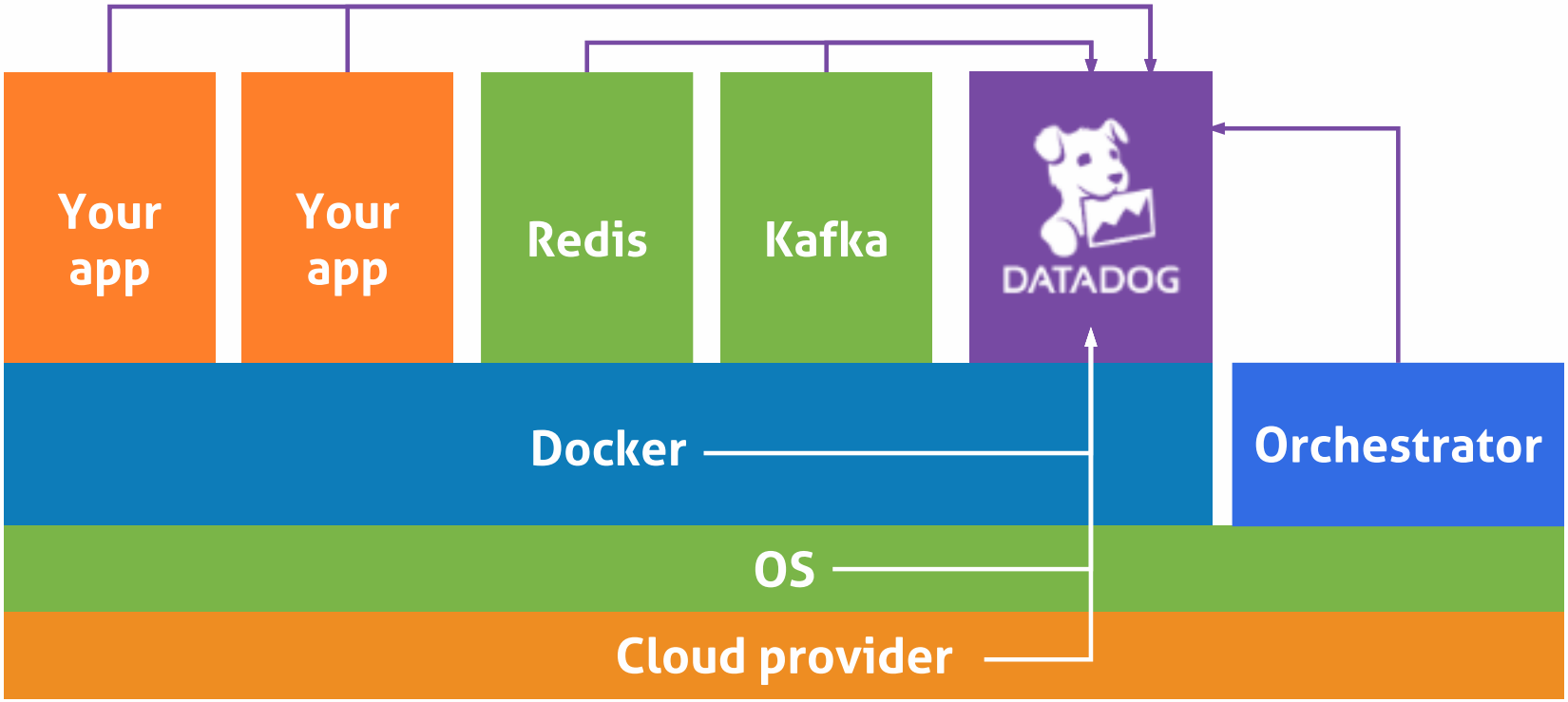
xavier.vello@datadoghq.com

http://lkdin.com/in/xaviervello

whois datadoghq.com



whois datadoghq.com




Filter by
cloud_provider ×
Fill by: % CPU utilized
avg
Size by:
--
aws
gcp
azure
% CPU utilized
0
50
100


$scope * $replset_name *
mongoDB
Available hosts
403
MongoDB events
db.northwest.dd.com is SECONDARY for hop
17 secs ago
db3.northwest.dd.com is SECONDARY for hop
4 mins ago
db.northwest.dd.com is STARTUP2 for golia
4 mins ago
db4c.northwest.dd.com
14 mins ago
dumbdb2.northwest.dd.io is PRIMARY for golia
3 hours ago
dumbdb2.northwest.dd.io is SECONDARY for golia
3 hours ago
care3.northwest.dd.io is PRIMARY for clog
3 hours ago
kobe.northwest.dd.io is SECONDARY for clog
3 hours ago
cab.northwest.dd.io is SECONDARY for hopper
3 hours ago
gito.northwest.dd.io is PRIMARY for ravi
3 hours ago
golia.northwest.dd.io is SECONDARY for danny
3 hours ago
coney.northwest.dd.io is STARTUP2 for hop
3 hours ago
kary.northwest.dd.io is PRIMARY for hopper
4 hours ago
kinoa.northwest.dd.io is SECONDARY for boom
4 hours ago
shardo.northwest.dd.io is SECONDARY for shard_p
19 hours ago
Throughput
Resource saturation
Read requests per second
Most used replica sets for reads
Queued read and write requests per second
Replica sets with largest read queues
Write requests per second
Most used replica sets for writes
Replica sets with largest write queu...
Number of clients with reads in progress or queued
Number of clients with writes in progress or queued
Resource utilization
Percentage of client connections used (%)
Max memory usage
Number of page faults per host
Data size per replica set
Performance & Errors
Oplog window per replica set
Number of open cursors per replica set
Average duration of background flushes per replica set (se...
Max Replication Lag per replica set (seconds)
Number of timed out cursors per replica set
Number of asserts per second

Application latency is high [bbd]

Edit | Status

OK since 2 HOURS AGO (4 Dec, 15:10:07) Created by: Mute

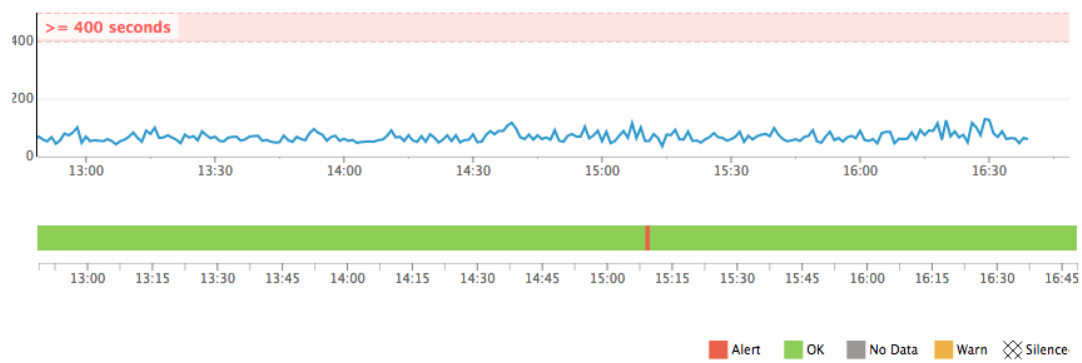
```
max(last_5m):avg:trace.dogweb.base.before.duration{*} + avg:cassandra.db.recent_read_latency_micros{*} + avg:cassandra.db.recent_write_latency_micros{*}
+ avg:redis.info.latency_ms{*} + avg:aws.elb.latency{*} >= 400
```

{{#is_alert}}
Application latency is rising above a threshold of {{threshold}} at {{value}}
...

Monitor History

Triggered Groups | All Groups

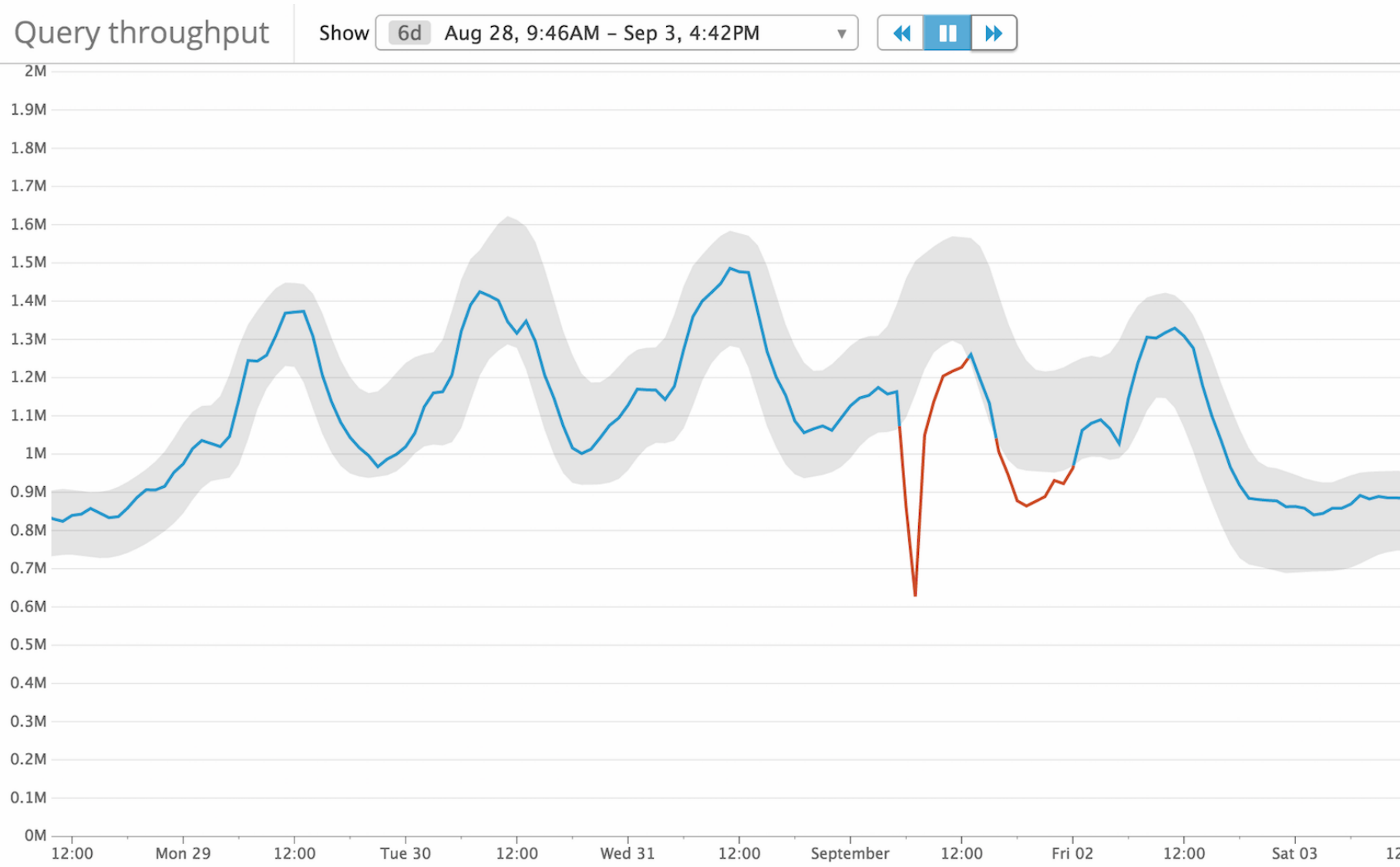
Show History over * (Entire Infrastructure) — 4h The Past 4 Hours

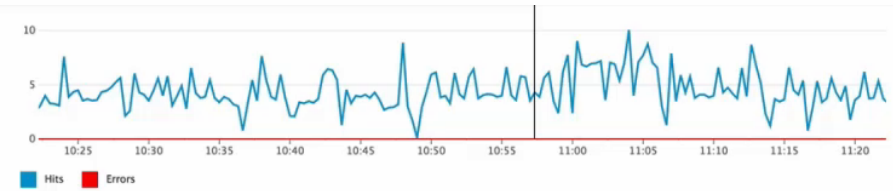


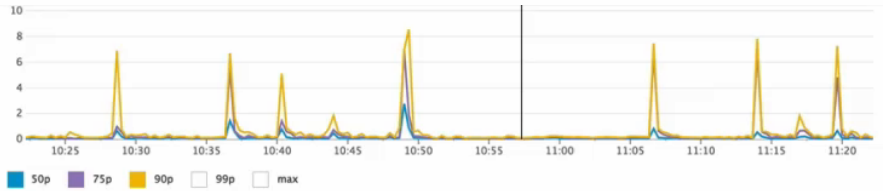
Related Monitors

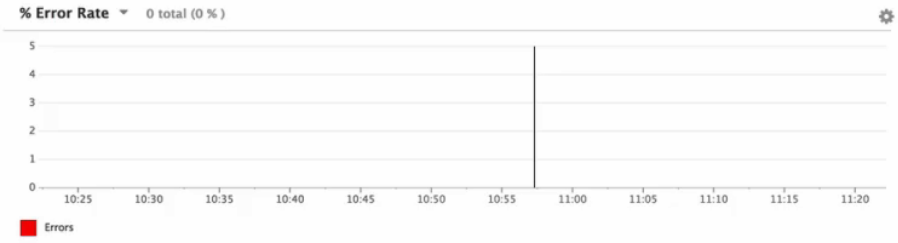
Similar Hosts | Similar Metrics

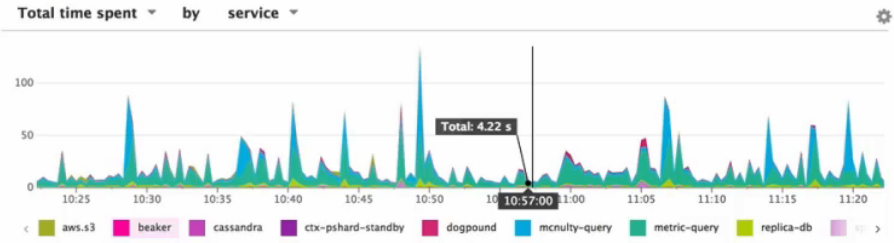
STATUS	NAME	DEFINITION	TAGS
OK	#estib test monitor	dd.app.errors	* random:test random:test random:test ...
ALERT	[AWS Stockholm] CPU is running high on a demo host	aws.ec2.cpuutilization	
OK	[boyan] Application latency is high	trace.dogweb.base.before.duration,cassandra.db.rec...	*

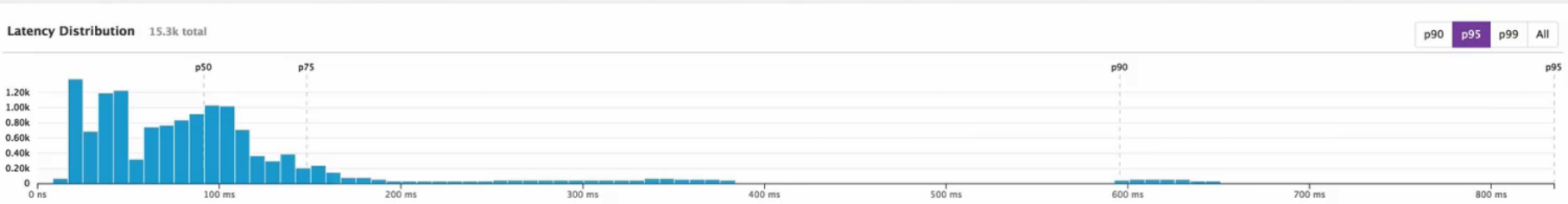

Query throughput
Show
6d
Aug 28, 9:46AM – Sep 3, 4:42PM
2M
1.9M
1.8M
1.7M
1.6M
1.5M
1.4M
1.3M
1.2M
1.1M
1M
0.9M
0.8M
0.7M
0.6M
0.5M
0.4M
0.3M
0.2M
0.1M
0M
12:00
Mon 29
12:00
Tue 30
12:00
Wed 31
12:00
September
12:00
Fri 02
12:00
Sat 03





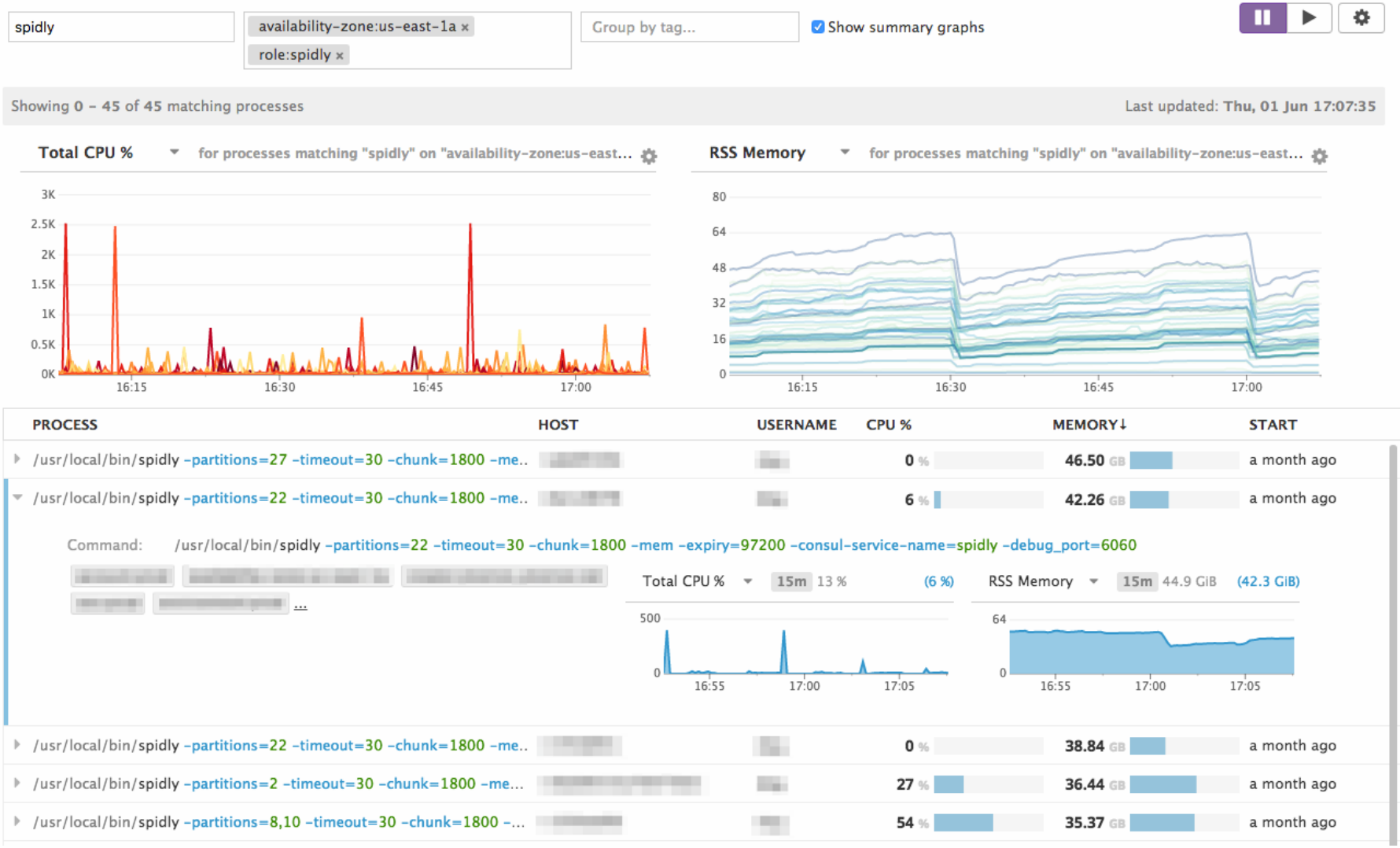


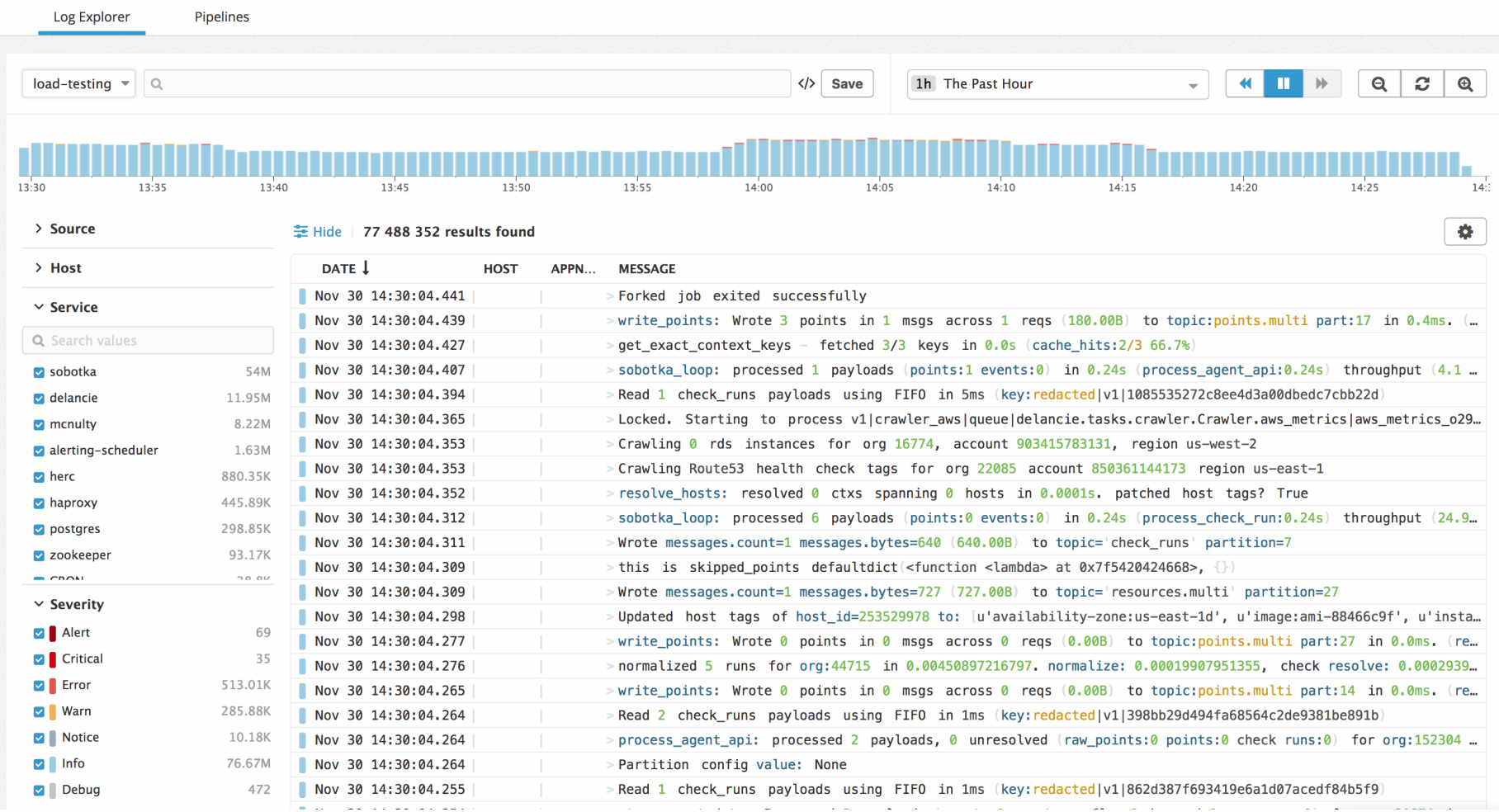




All | OK | Errors — Duration between **0 ms** and **13.6 s** — *798 sampled traces found. Click on a row to view trace*

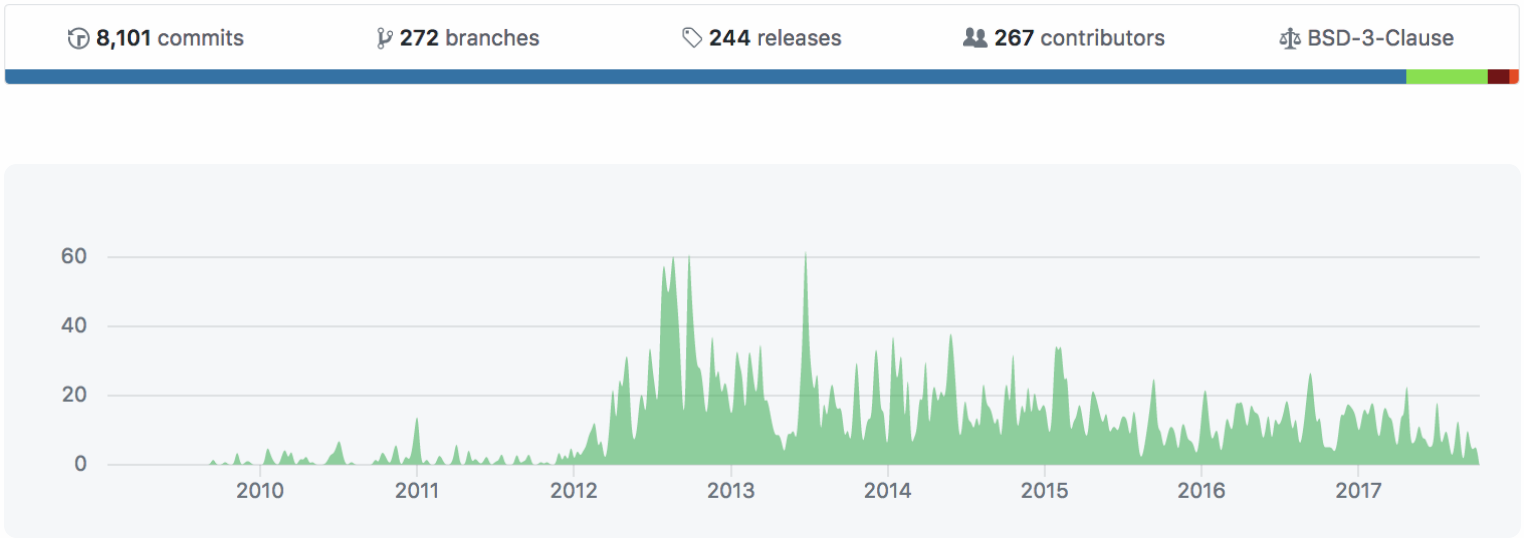
Date ↓	Duration	Status	Latency Breakdown by %	Hostname
11:22:11 AM 6 mins ago	49.6 ms	200		i-0f16e0db77b3d2d28
11:22:02 AM 6 mins ago	73.5 ms	200		i-0d2cfa1d52c67bde4
11:21:58 AM 6 mins ago	43.1 ms	200		i-0c503a32a17093762
11:21:57 AM 6 mins ago	94.7 ms	200		i-0f16e0db77b3d2d28
11:21:56 AM 6 mins ago	212 ms	200		i-0f16e0db77b3d2d28
11:21:55 AM 6 mins ago	59.8 ms	200		i-0f5ec59800dd4b3a5
11:21:41 AM 6 mins ago	56.5 ms	200		i-0928f985f3e2029a9


spidly
availability-zone:us-east-1a ×
role:spidly ×
Group by tag...
Show summary graphs
Showing 0 - 45 of 45 matching processes
Last updated: Thu, 01 Jun 17:07:35
Total CPU %
for processes matching "spidly" on "availability-zone:us-east...
RSS Memory
for processes matching "spidly" on "availability-zone:us-east...
3K
2.5K
2K
1.5K
1K
0.5K
0K
16:15
16:30
16:45
17:00
80
64
48
32
16
0
PROCESS
HOST
USERNAME
CPU %
MEMORY↓
START
/usr/local/bin/spidly -partitions=27 -timeout=30 -chunk=1800 -me..
0 %
46.50 GB
a month ago
/usr/local/bin/spidly -partitions=22 -timeout=30 -chunk=1800 -me..
6 %
42.26 GB
a month ago
Command:
/usr/local/bin/spidly -partitions=22 -timeout=30 -chunk=1800 -mem -expiry=97200 -consul-service-name=spidly -debug_port=6060
Total CPU %
15m 13 %
(6 %)
RSS Memory
15m 44.9 GiB
(42.3 GiB)
500
0
16:55
17:00
17:05
64
0
16:55
17:00
17:05
/usr/local/bin/spidly -partitions=22 -timeout=30 -chunk=1800 -me..
0 %
38.84 GB
a month ago
/usr/local/bin/spidly -partitions=2 -timeout=30 -chunk=1800 -me...
27 %
36.44 GB
a month ago
/usr/local/bin/spidly -partitions=8,10 -timeout=30 -chunk=1800 -...
54 %
35.37 GB
a month ago


Log Explorer
Pipelines
load-testing
Save
1h The Past Hour
13:30
13:35
13:40
13:45
13:50
13:55
14:00
14:05
14:10
14:15
14:20
14:25
Source
Host
Service
Search values
sobotka 54M
delancie 11.95M
mcnulty 8.22M
alerting-scheduler 1.63M
herc 880.35K
haproxy 445.89K
postgres 298.85K
zookeeper 93.17K
Severity
Alert 69
Critical 35
Error 513.01K
Warn 285.88K
Notice 10.18K
Info 76.67M
Debug 472
Hide
77 488 352 results found
DATE
HOST
APPN...
MESSAGE
Nov 30 14:30:04.441 Forked job exited successfully
Nov 30 14:30:04.439 write_points: Wrote 3 points in 1 msgs across 1 reqs (180.00B) to topic:points.multi part:17 in 0.4ms.
Nov 30 14:30:04.427 get_exact_context_keys - fetched 3/3 keys in 0.0s (cache_hits:2/3 66.7%)
Nov 30 14:30:04.407 sobotka_loop: processed 1 payloads (points:1 events:0) in 0.24s (process_agent_api:0.24s) throughput (4.1
Nov 30 14:30:04.394 Read 1 check_runs payloads using FIFO in 5ms (key:redacted|v1|1085535272c8ee4d3a00dbedc7cbb22d)
Nov 30 14:30:04.365 Locked. Starting to process v1|crawler_aws|queue|delancie.tasks.crawler.Crawler.aws_metrics|aws_metrics_o29
Nov 30 14:30:04.353 Crawling 0 rds instances for org 16774, account 903415783131, region us-west-2
Nov 30 14:30:04.353 Crawling Route53 health check tags for org 22085 account 850361144173 region us-east-1
Nov 30 14:30:04.352 resolve_hosts: resolved 0 ctxs spanning 0 hosts in 0.0001s. patched host tags? True
Nov 30 14:30:04.312 sobotka_loop: processed 6 payloads (points:0 events:0) in 0.24s (process_check_run:0.24s) throughput (24.9
Nov 30 14:30:04.311 Wrote messages.count=1 messages.bytes=640 (640.00B) to topic='check_runs' partition=7
Nov 30 14:30:04.309 this is skipped_points defaultdict(<function <lambda> at 0x7f5420424668>, {})
Nov 30 14:30:04.309 Wrote messages.count=1 messages.bytes=727 (727.00B) to topic='resources.multi' partition=27
Nov 30 14:30:04.298 Updated host tags of host_id=253529978 to: [u'availability-zone:us-east-1d', u'image:ami-88466c9f', u'insta
Nov 30 14:30:04.277 write_points: Wrote 0 points in 0 msgs across 0 reqs (0.00B) to topic:points.multi part:27 in 0.0ms. (re
Nov 30 14:30:04.276 normalized 5 runs for org:44715 in 0.00450897216797. normalize: 0.00019907951355, check resolve: 0.0002939
Nov 30 14:30:04.265 write_points: Wrote 0 points in 0 msgs across 0 reqs (0.00B) to topic:points.multi part:14 in 0.0ms. (re
Nov 30 14:30:04.264 Read 2 check_runs payloads using FIFO in 1ms (key:redacted|v1|398bb29d494fa68564c2de9381be891b)
Nov 30 14:30:04.264 process_agent_api: processed 2 payloads, 0 unresolved (raw_points:0 points:0 check runs:0) for org:152304
Nov 30 14:30:04.264 Partition config value: None
Nov 30 14:30:04.255 Read 1 check_runs payloads using FIFO in 1ms (key:redacted|v1|862d387f693419e6a1d07acedf84b5f9)

Our collection agent

A 7 year-old codebase, **24 458 Python SLOCs**



87 integrations (**32 274 Python SLOCs**) in a different repo

The Agent6 project

A rewrite

- With **containers** support from the start
- With **Autodiscovery** at the core

The Agent6 project

A rewrite

- With **containers** support from the start
- With **Autodiscovery** at the core

A rewrite in Go

- **Reduce** memory and CPU usage
- **Better** reliability and maintainability
- True **concurrency**

Agenda

Share our challenges and discoveries:

- Namespaces & cgroups
- DogStatsD traffic: using Unix Domain Sockets
- Secure usage of the Docker Socket?

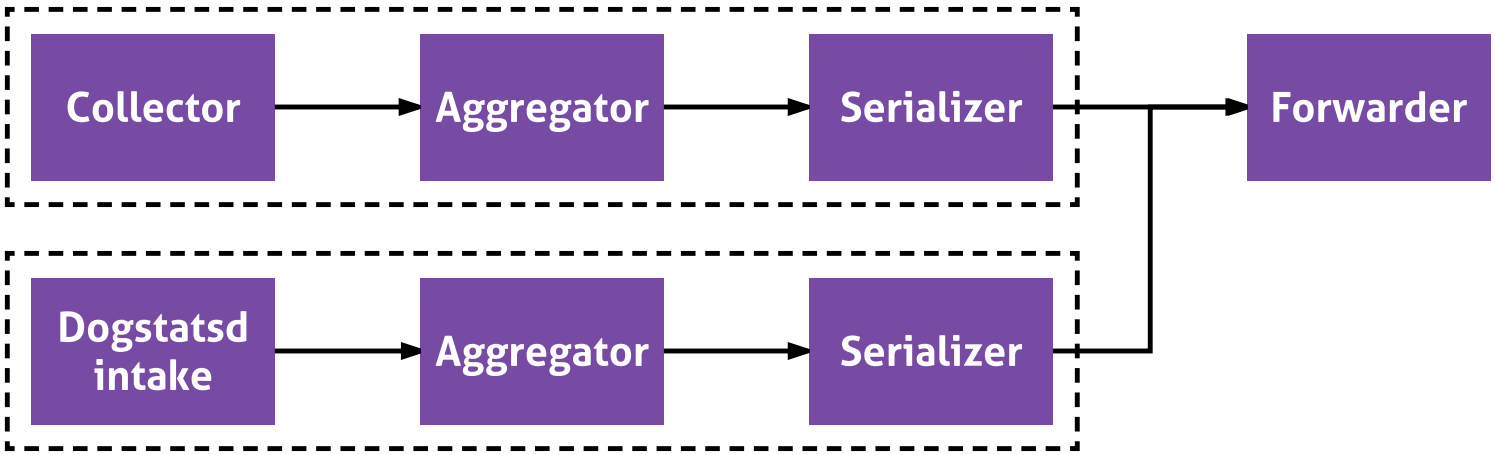
Namespaces & cgroups

"The dead forwarder" case of 2017

Network metrics

"The dead forwarder" case

```
supervisord supervisord -n -c /etc/dd-agent/supervisor.conf
  |
  ├─python agent.py foreground --use-local-forwarder
  |   └─{python}
  ├─python dogstatsd.py --use-local-forwarder
  |   └─{python}
  └─python ddagent.py
```



"The dead forwarder" case

- Support case: "metrics not coming through although the collection works OK"

"The dead forwarder" case

- Support case: "metrics not coming through although the collection works OK"
- Collector logs: errors connecting to `localhost:17123`

"The dead forwarder" case

- Support case: "metrics not coming through although the collection works OK"
- Collector logs: errors connecting to `localhost:17123`
- Supervisord logs: unexpected `exit` of the forwarder, retries 5 times then gives up

"The dead forwarder" case

- Support case: "metrics not coming through although the collection works OK"
- Collector logs: errors connecting to `localhost:17123`
- Supervisord logs: unexpected `exit` of the forwarder, retries 5 times then gives up
- Forwarder logs: `no error`

`docker-dd-agent` process tree

```
supervisord supervisord -n -c /etc/dd-agent/supervisor.conf
  |
  ├─python agent.py foreground --use-local-forwarder
  |   └─{python}
  ├─python dogstatsd.py --use-local-forwarder
  |   └─{python}
  └─python ddagent.py
```

docker-dd-agent process tree

```
supervisord supervisord -n -c /etc/dd-agent/supervisor.conf
  |
  ├─python agent.py foreground --use-local-forwarder
  |   └─{python}
  ├─python dogstatsd.py --use-local-forwarder
  |   └─{python}
  └─
```

`mem` cgroup

- Limit tells how much RAM / RAM+swap a cgroup can use
- If limit exceeded, ...

`mem` cgroup

- Limit tells how much RAM / RAM+swap a cgroup can use
- If limit exceeded, **oomkiller** descends upon your cgroup
  - Can kill the PID 1 (which leads to a container OOM exit)
  - Or not… which can leave the container **stuck in a non-working state** 🤷‍♀️
- Must pre-emptively monitor `docker.mem.in_use` and `docker.mem.sw_in_use` to see if that could happen

Let's test it

```
services:
  memleak-pid1:
    image: alpine:3.6
    command: "ash -c 'for i in `seq 1 10000000`; do true; done'"
    mem_limit: 10000000
    restart: on-failure
  memleak-forked:
    image: alpine:3.6
    command: "ash -c \"ash -c 'for i in `seq 1 10000000`; do true; done'
              & sleep 20\""
    mem_limit: 10000000
    restart: on-failure
```

Result

```
ubuntu@ci-xaviervello:~$ docker-compose -f composes/memlimit.compose up
memleak-pid1_1    | Killed
composes_memleak-pid1_1 exited with code 137 💥
memleak-pid1_1    | Killed
memleak-forked_1  | Killed
...
composes_memleak-forked_1 exited with code 0 ✅
composes_memleak-pid1_1 exited with code 137 💥
```

Result

```
ubuntu@ci-xaviervello:~$ docker-compose -f composes/memlimit.compose up
memleak-pid1_1    | Killed
composes_memleak-pid1_1 exited with code 137 💥
memleak-pid1_1    | Killed
memleak-forked_1  | Killed
...
composes_memleak-forked_1 exited with code 0 ✅
composes_memleak-pid1_1 exited with code 137 💥
```

What should I do?

- Ideally, run only one program per container
- Or use a robust supervisor, check exit codes
- Have a thorough healthcheck

`net` namespace

- Every container has their own network namespace
- Their own virtual `eth0` that is bridged by the host
- Allows isolation and routing
- Allows us to collect per-container metrics

```
$ docker exec agent5 cat /host/proc/30828/net/dev
Inter-|
 face | bytes       packets  errs drop fifo frame compressed multicast
    lo: 6961740     50619    0    0    0     0          0         0
  eth0: 19558170    37932    0    0    0     0          0         0
```

Getting the host's net stats

```
$ cat /proc/net/dev
Inter-|
 face | bytes      packets   errs drop fifo frame compressed multicast
enp0s8: 61509004   104768    0    0    0     0          0         0
enp0s3: 523131054  862084    0    0    0     0          0         0
    lo: 2952       46    0   0    0     0          0         0
...
```

```
$ docker exec agent5 cat /proc/net/dev
Inter-|
 face | bytes      packets  errs drop fifo frame compressed multicast
    lo: 6656783    48403    0    0    0     0          0         0
  eth0: 18699348   36269    0    0    0     0          0         0
```

Getting the host's net stats

```
$ docker exec agent5 cat /host/proc/net/dev
```

Getting the host's net stats

```
$ docker exec agent5 cat /host/proc/net/dev
```

```
Inter-|
 face | bytes      packets  errs drop fifo frame compressed multicast
    lo: 6697512    48702    0    0    0     0          0         0
  eth0: 18812503   36489    0    0    0     0          0         0
```

Getting the host's net stats

```
$ docker exec agent5 cat /host/proc/net/dev
```

```
Inter-|
 face |bytes      packets  errs drop fifo frame compressed multicast
    lo: 6697512    48702    0    0    0     0          0         0
  eth0: 18812503   36489    0    0    0     0          0         0
```

- We need to run with `net=host` 😞

Getting the host's net stats

```
$ docker exec agent5 cat /host/proc/net/dev
```

```
Inter-|
 face | bytes      packets  errs drop fifo frame compressed multicast
    lo: 6697512    48702    0    0    0     0          0         0
  eth0: 18812503   36489    0    0    0     0          0         0
```

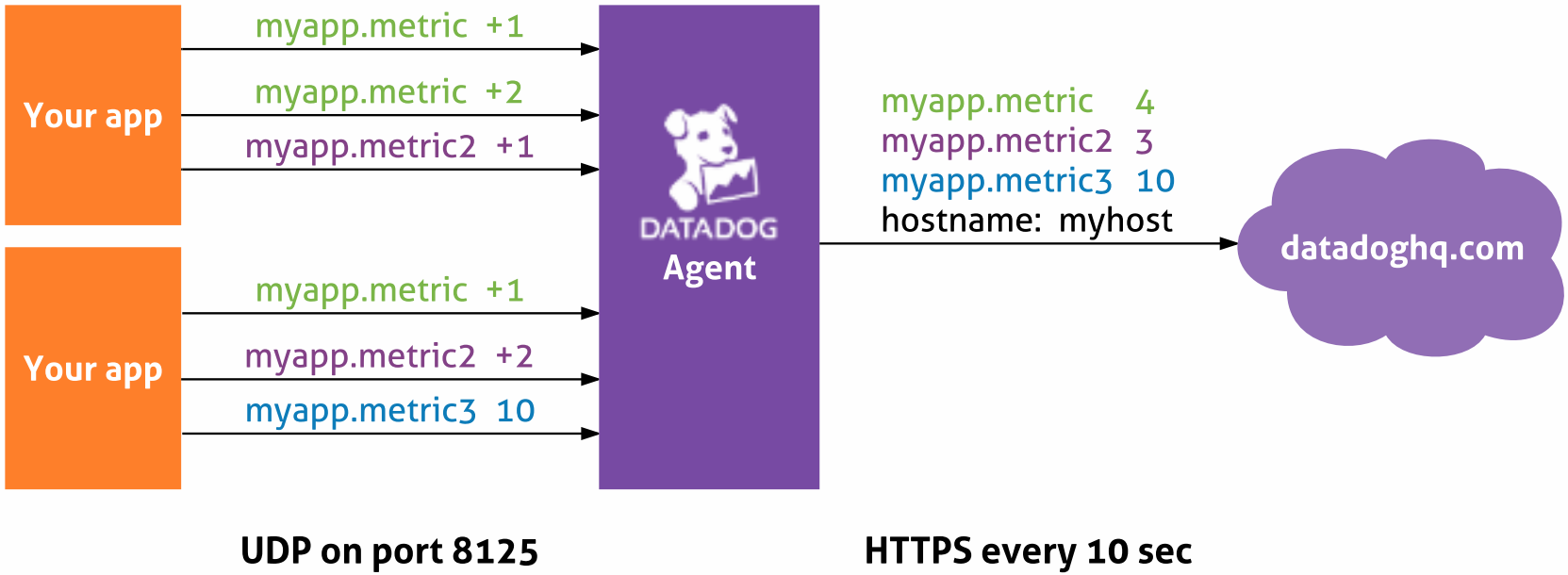
- We need to run with `net=host` 😞
- Investigating monitoring `pid 1`'s net stats

```
$ docker exec agent5 cat /host/proc/1/net/dev
Inter-|
 face | bytes      packets   errs drop fifo frame compressed multicast
enp0s8: 61509280   104771    0    0    0     0          0         0
enp0s3: 523996193  864400    0    0    0     0          0         0
    lo: 2952       46        0    0    0     0          0         0
...
```

DogStatsD traffic: using Unix Domain Sockets

What is DogStatsd?

Collection, tagging, aggregation of your custom metrics



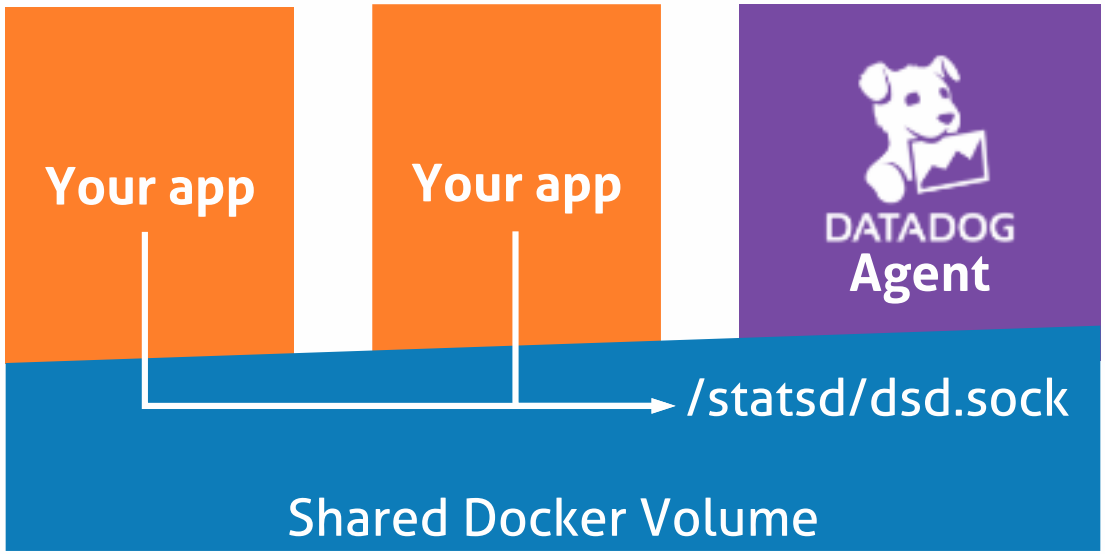
Issues with UDP

- Poor **discoverability** of the endpoint
- No error handling makes **automatic detection** impossible
- **Network overhead** (lots of small UDP packets)
- Detecting **source container** per IP is error-prone

So, we're just trying to talk to `localhost`, right?

Datagram Unix Sockets

- Host-local traffic
- Same semantics as UDP
- Smaller CPU overhead than iptables
- Endpoint is a **socket file in a docker volume**



Container tagging is easy!

```
unix.SetsockoptInt(fd, unix.SOL_SOCKET, unix.SO_PASSCRED, 1)
```

Unix Credentials in anciallary data, provided by the kernel

```
type Ucred struct {
    Pid int32
    Uid uint32
    Gid uint32
}
```

Need to run in `pid=host` mode

Container tagging is easy!

You can slice and compare your custom metrics by:

- Availability zone
- Host
- Chef role

as before, but also now by:

- **ECS task**
- **Kubernetes deployment**
- **Docker labels**

Secure usage of the Docker Socket?

Secure Docker Socket?

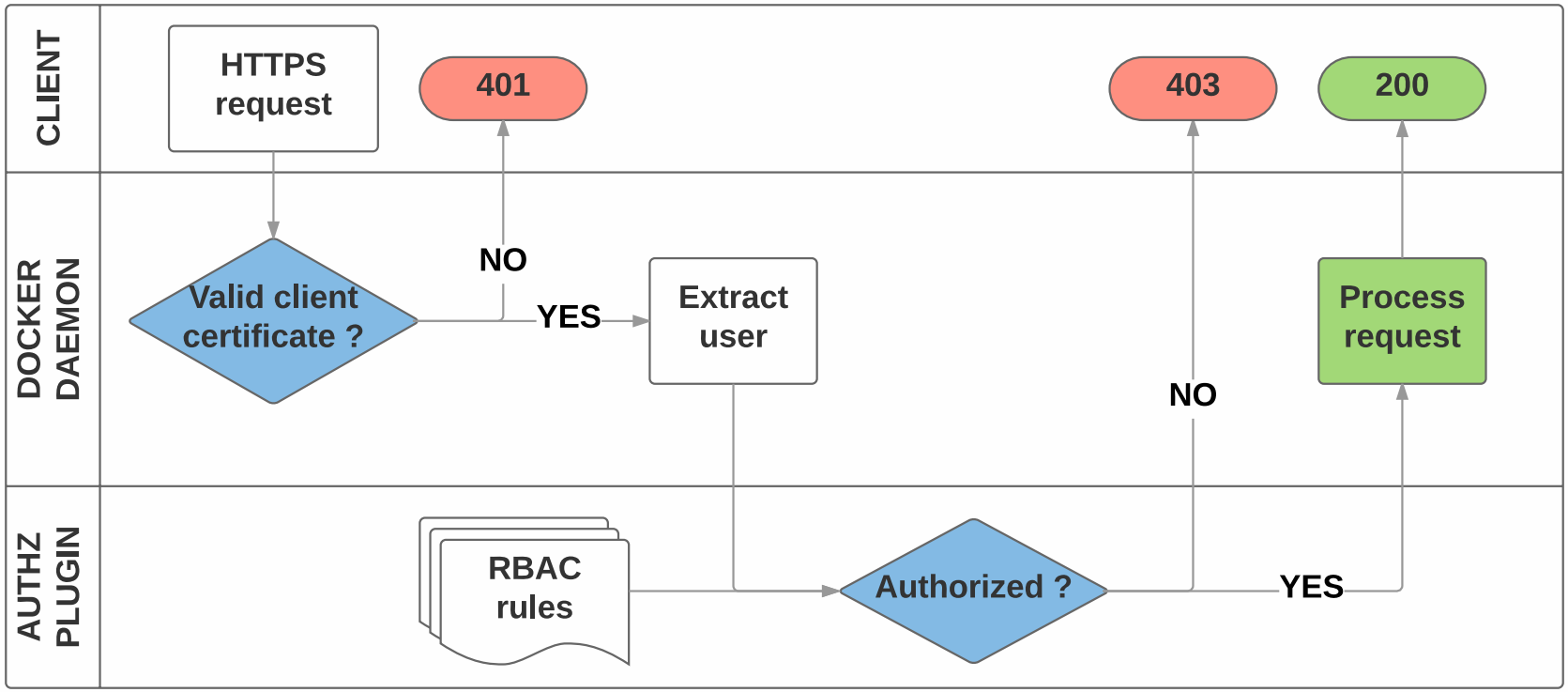
Monitoring is intrusive, but we strive to make it secure and seamless

Secure Docker Socket?

Monitoring is intrusive, but we strive to make it secure and seamless

- We need to list and inspect containers, images, volumes
- Docker does not provide a **monitoring interface**
- It provides **monitoring endpoints** in its **management interface**

Plan A: use AuthN and AuthZ



Plan A: use AuthN and AuthZ

Pros:

- Already upstream since 1.11
- Solid authentication mechanism
- Similar to the Kubernetes AuthN/AuthZ system

Plan A: use AuthN and AuthZ

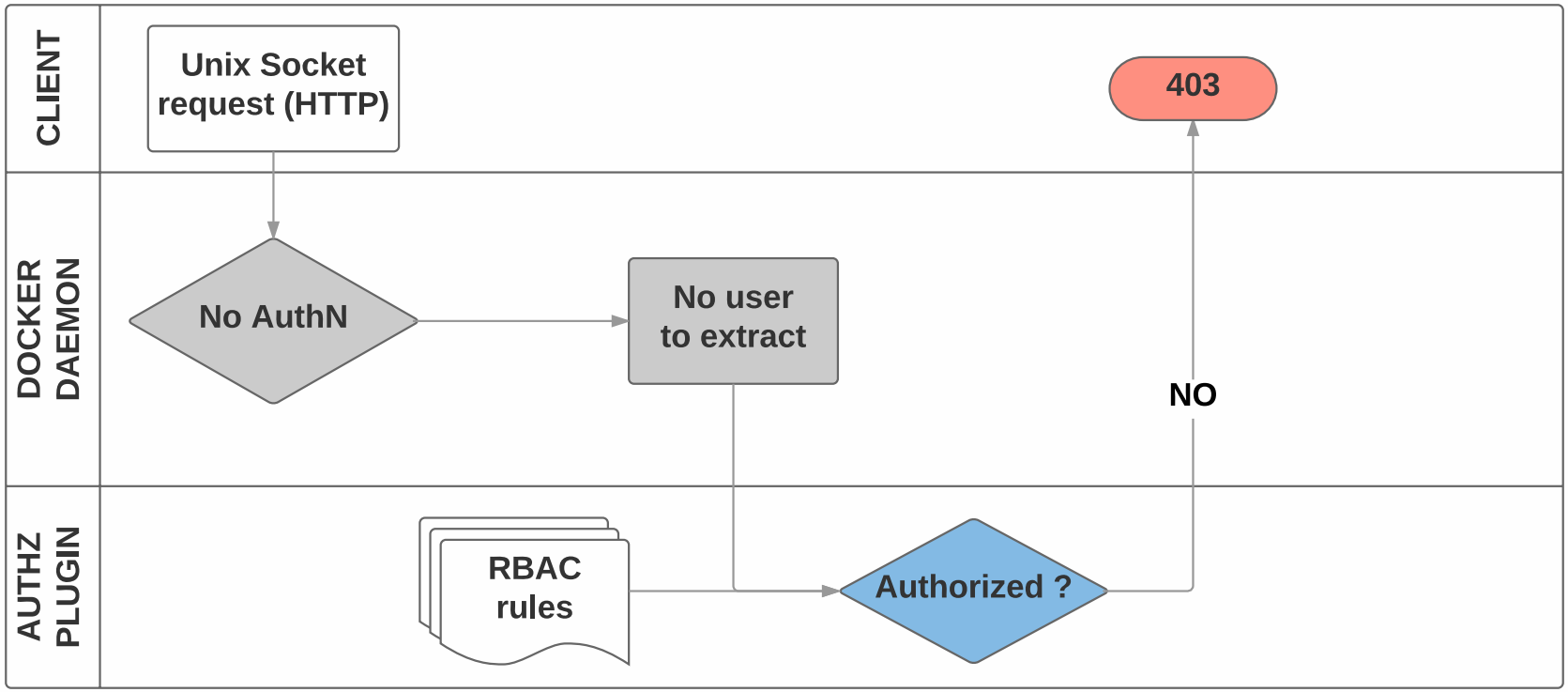
Pros:

- Already upstream since 1.11
- Solid authentication mechanism
- Similar to the Kubernetes AuthN/AuthZ system

Cons:

- Need to setup a PKI
- Requires a third-party AuthZ plugin
- `/var/run/docker.sock` unusable (it's HTTP-only for now)

Plan A: use AuthN and AuthZ



Plan B1: HTTPS on UDS

Pros:

- Minimal changes in Moby

Plan B1: HTTPS on UDS

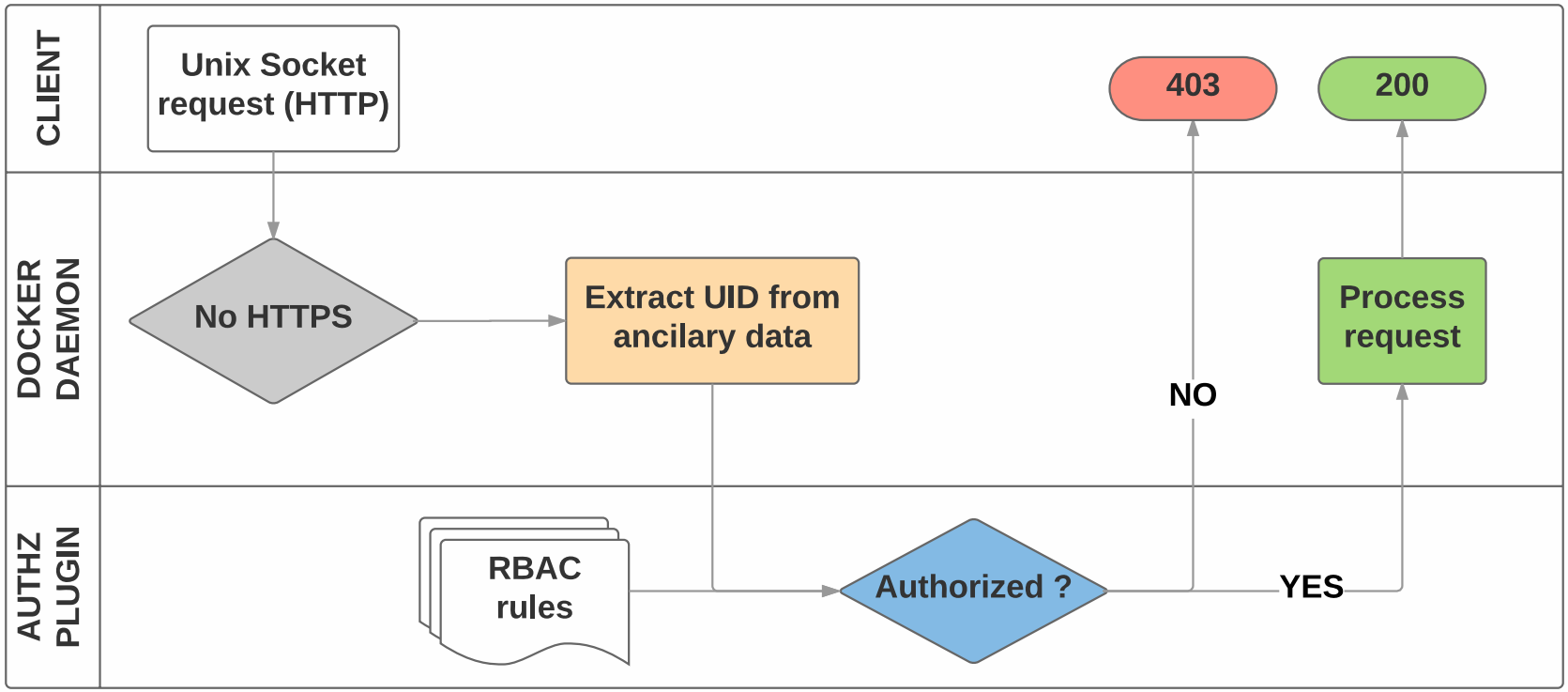
Pros:

- Minimal changes in Moby

Cons:

- Still need to setup a PKI
- **Still breaks most orchestrators**

Plan B2: use UID for AuthZ



Plan B2: use UID for AuthZ

Pros:

- Minimal changes
- No PKI
- Transparent to orchestrators (just whitelist `unix:root`)

Plan B2: use UID for AuthZ

Pros:

- Minimal changes
- No PKI
- Transparent to orchestrators (just whitelist `unix:root`)

Cons:

- Linux-specific logic
- Different code path for AuthN

Plan B2: use UID for AuthZ

Great idea!

Let's discuss it on:
https://github.com/moby/moby/issues/35711

RFC: enable AuthZ on the unix socket

Plan B2: use UID for AuthZ

Great idea!

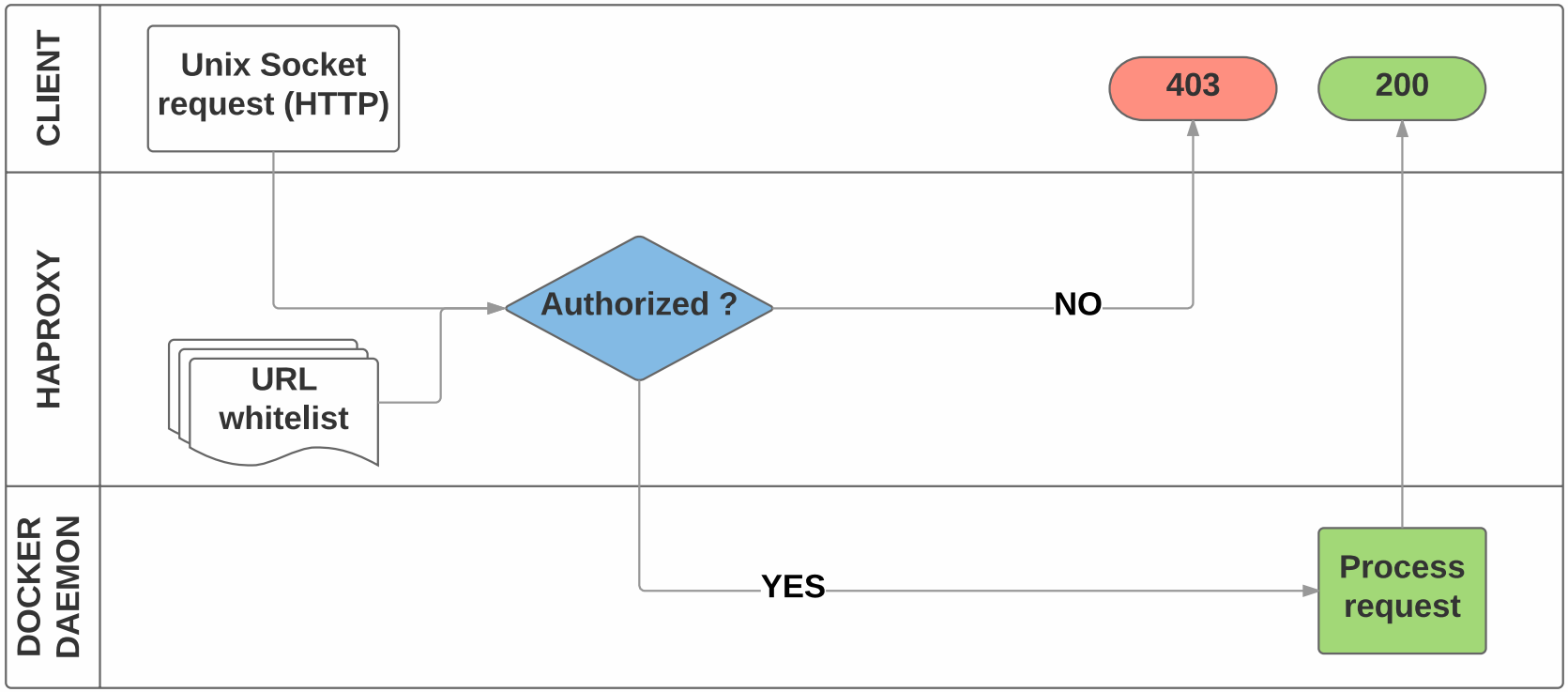
Let's discuss it on:
https://github.com/moby/moby/issues/35711

RFC: enable AuthZ on the unix socket

Wait! 😿

- ECS ships `1.12`
- GKE ships `1.13`

Plan C: use a filtering proxy



Plan C: use a filtering proxy

```
services:
  dockerproxy:
    image: "datadog/docker-filter"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - safe-socket:/safe-socket:rw
    network_mode: "none"

  agent5:
    image: "datadog/docker-dd-agent:latest"
    volumes:
      - safe-socket:/var/run:ro
      - /proc/:/host/proc/:ro
      - /sys/fs/cgroup/:/host/sys/fs/cgroup:ro
```

Thanks!

Questions?

xavier.vello@datadoghq.com

DATADOG

We are hiring!

Paris, NYC, Remote

http://jobs.datadoghq.com