mesoform

SIMPLIFYING CONTAINERS AT SCALE

THE CONCIERGE PARADIGM

YOUR CONCIERGE FOR THE EVENING

- Gareth Brown
- Director and technologist at Mesoform
- Specialise in securely simplifying and streamlining
- DevOps back in early '00s
- Was running containers in production many years ago
- Built a self-service VM infrastructure..

Google

Azure

amazon
web services™

HISTORY OF CONTAINERS

- 1979: chroot
- Jails, Zones, LXC (2000, 2004, 2008)
- Along comes AWS
- Docked back in

FunnyWildLife.com

FLYING FISH

- Docker Engine in the Cloud
- Maintaining pets
- Monitoring
- Scheduling
- Auto-scaling
- Service discovery

LAYER CAKE

- New technologies (Kubernetes, Mesos)
- Complex
- Integrating different workloads and IaaS
- Up-skilling and support
- Tight coupling and dependency
- Keep It Stupidly Simple

THEY SHOULD REMAKE "BACK TO THE FUTURE 2"
WHERE THERE ARE NO FLYING CARS
SOCIAL DEMENTIA
AND PEOPLE JUST STARE AT THEIR PHONES ALL
THE TIME GETTING OFFENDED AT EVERYTHING

FUTURE OF CONTAINERS

- Standardisation
- Portability
- Performance
- Simplified management
- Resource Utilisation
- Cost!

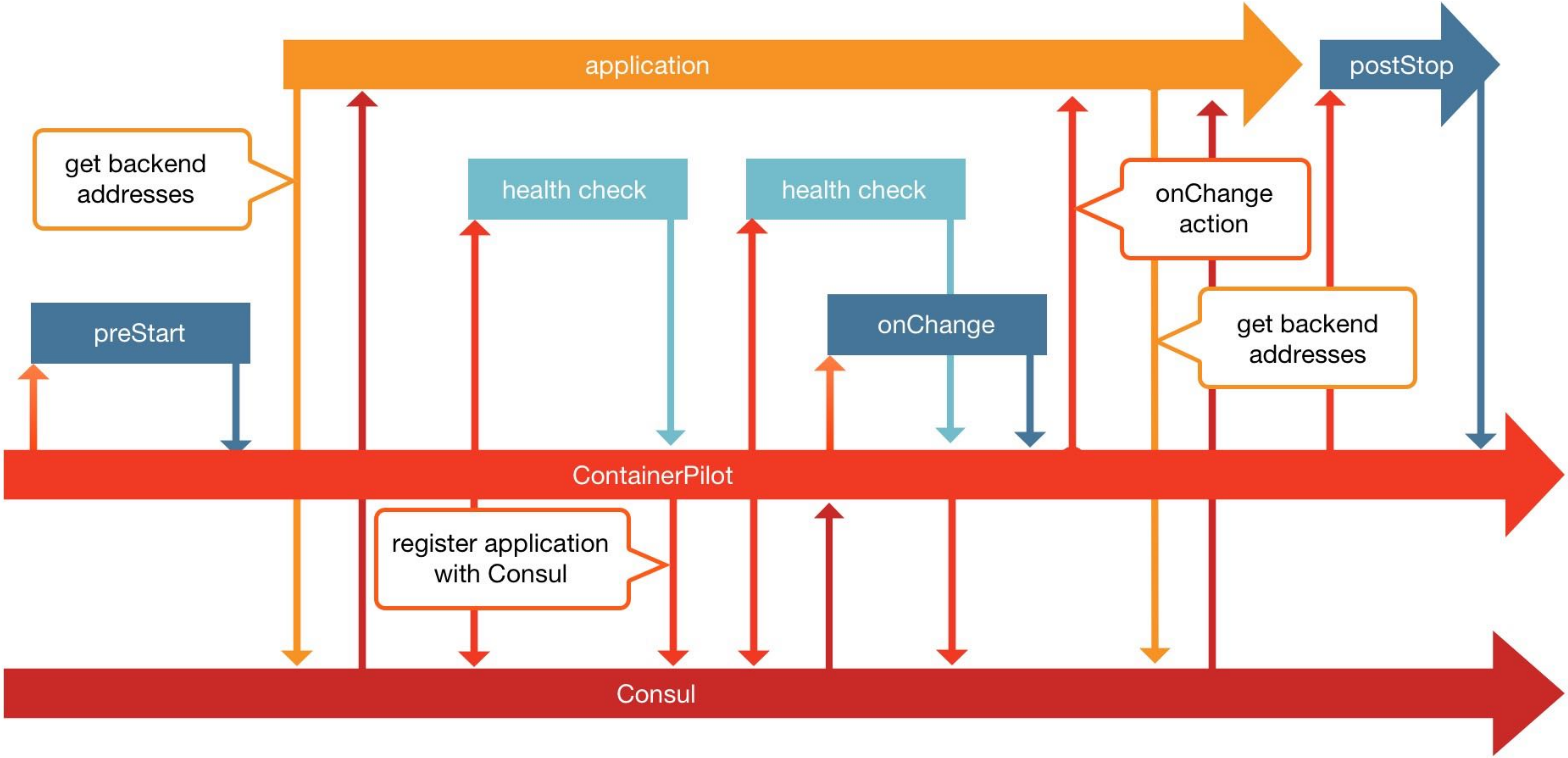
PUPPIES MAKE PAIN GO AWAY

OPERATING PAINS

- On-Premise, EC2, ECS, CoreOS, Kubernetes, other AWS services, Java, Python…
- Papertrail and Elastic Stack
- Zabbix and Librato
- Dropwizard with agents pulling from applications
- bumped all of the common issues
- Windowing and performance

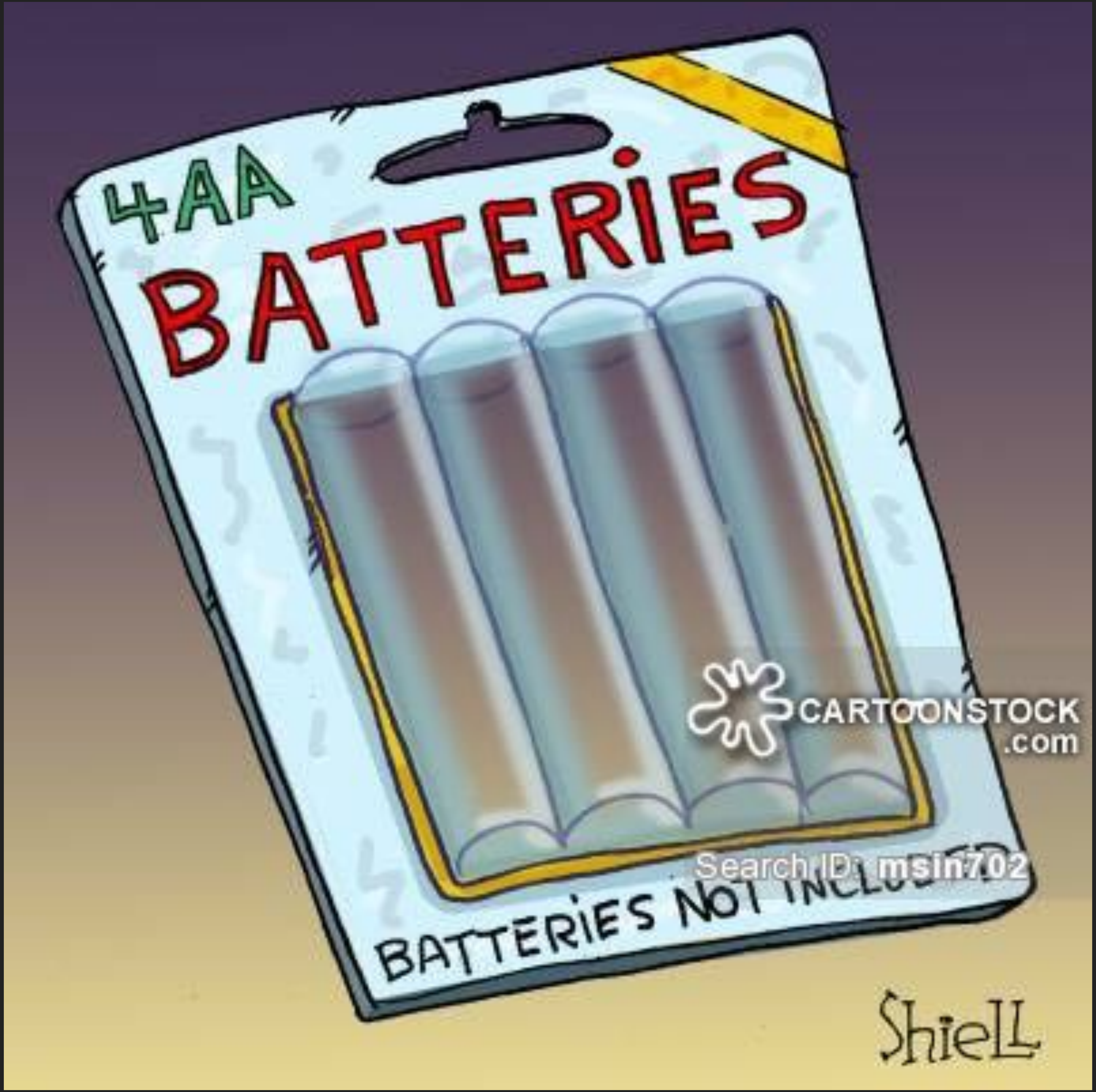
AUTOPILOT PATTERN

- No complex framework
- Service discovery
- Application orchestration
- Small piece of code to automate common actions


application
postStop
get backend addresses
health check
health check
onChange action
preStart
onChange
get backend addresses
ContainerPilot
register application with Consul
Consul

FLYING ON AUTOPILOT

- Scheduler agnostic
- Most things just work
- App-centric orchestration
- Drastically less management
- Production grade environment, test environment time
- Co-processes!


4AA
BATTERIES
BATTERIES NOT INCLUDED
Shiell

BATTERIES INCLUDED

- Loose-couple to well defined systems
- Automatically register our containers
- Automatically discover resources
- Self-healing or corrective actions
- Interact with legacy applications
- Compliance scanning

CONTAINERPILOT.JSON

```
jobs: [{
    name: 'scheduling-status-healthy',
    exec: ['zabbix_sender', '-c', '/etc/coprocesses/zabbix/zabbix_agentd.conf',
        '--key', 'container.state', '--value', '1'],
    when: { source: 'apache-fwdproxy', each: 'healthy' }
  },{
    name: 'zabbix-agent',
    exec: ['/usr/sbin/zabbix_agentd',
        '-fc', '/etc/coprocesses/zabbix/zabbix_agentd.conf'],
    restarts: 'unlimited',
    health: { exec: 'zabbix_agentd -t agent.ping', interval: 30, ttl: 60, timeout: 5 },
    when: { source: 'platform-integration-setup', once: 'exitSuccess' }
  },{
    name: 'post-stop',
    exec: ['zabbix_sender', '-c', '/etc/coprocesses/zabbix/zabbix_agentd.conf',
    '--key', 'container.state', '--value', '0'],
    when: { once: 'shutdown' }
  }]
```

© Daniel Leal-Olivas / i-Images

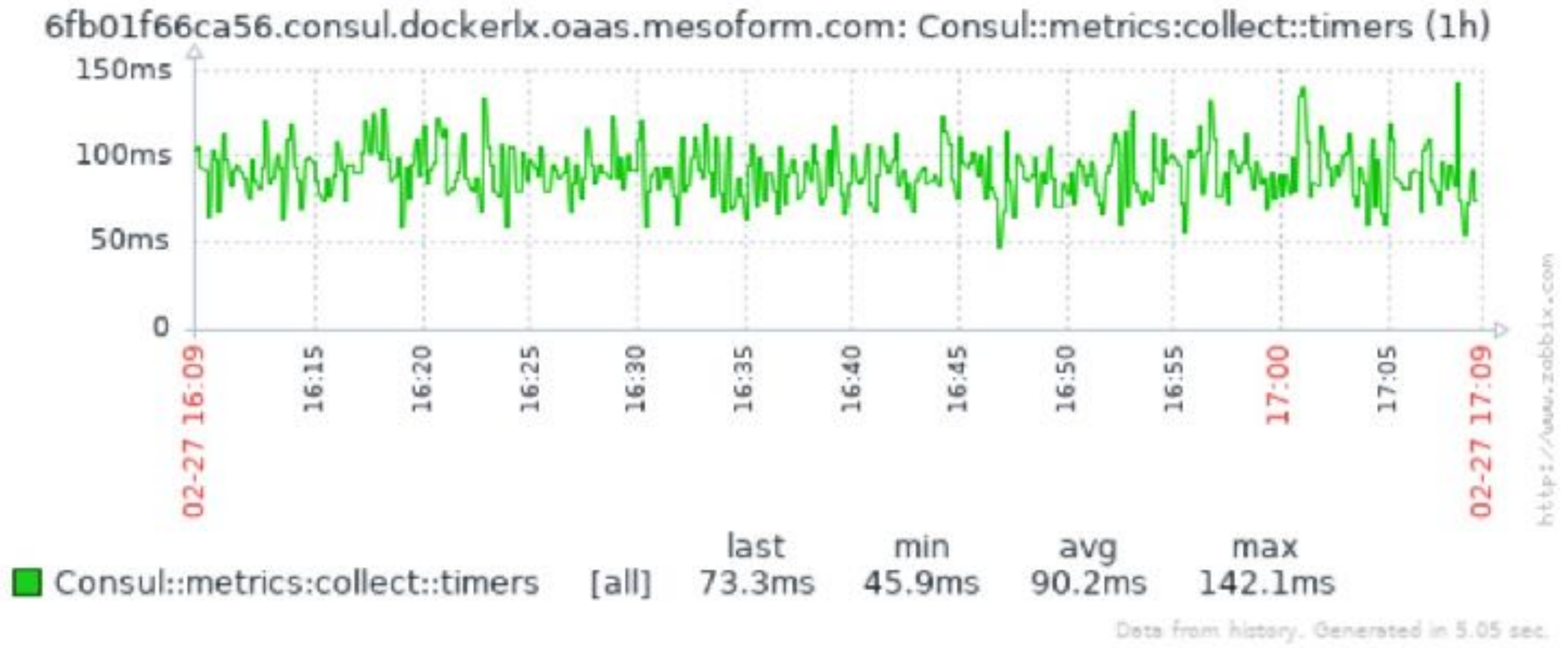
PUSH VS PULL

- Push method: auto-register but no confidence in instance state
- Pull method: centralised configuration but extra management
- Pull understands load and partitioning
- Processing poor performance
- Windowing
- Can we unify push and pull?

中和池
桑拿 冲浪 休闲
中和池浴室

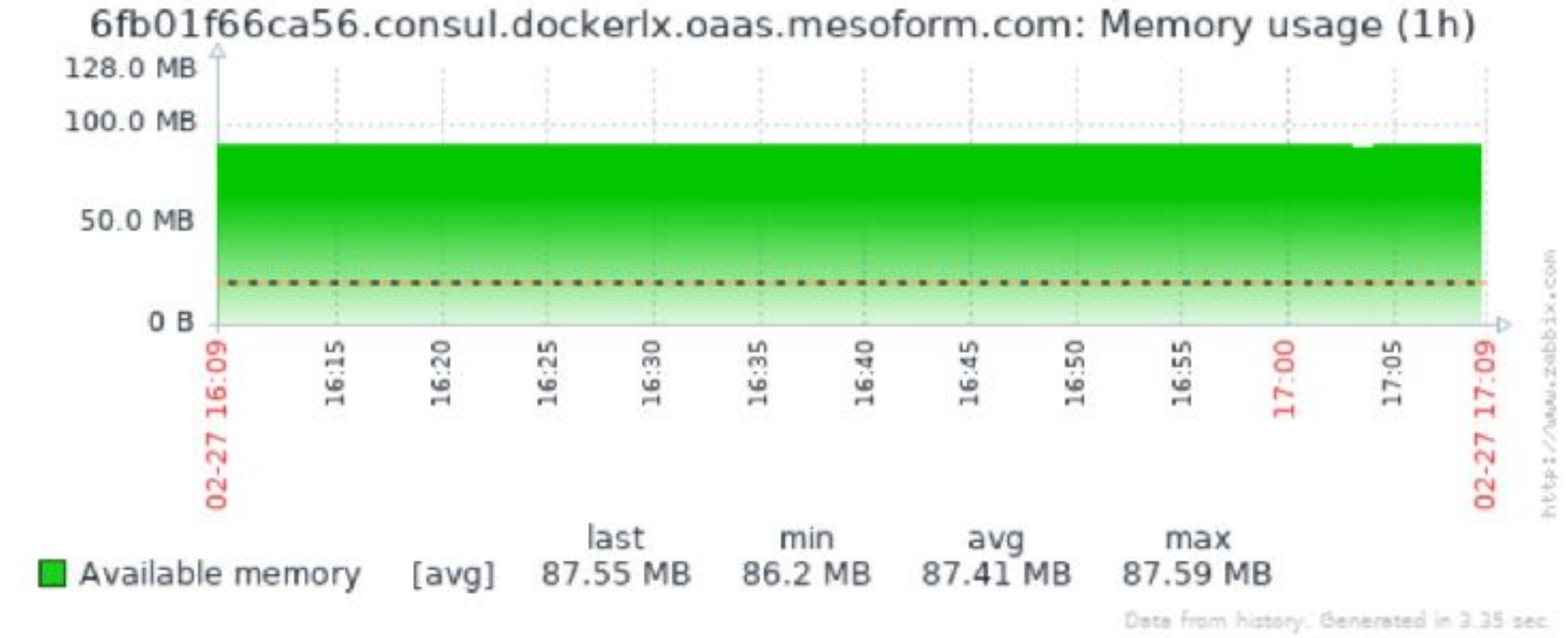
THE CONCIERGE COURIER

- Two purposes (discovery, delivery)
- Learns metrics
- Picks up metrics
- Delivers them
- Records delivery
- Performance?



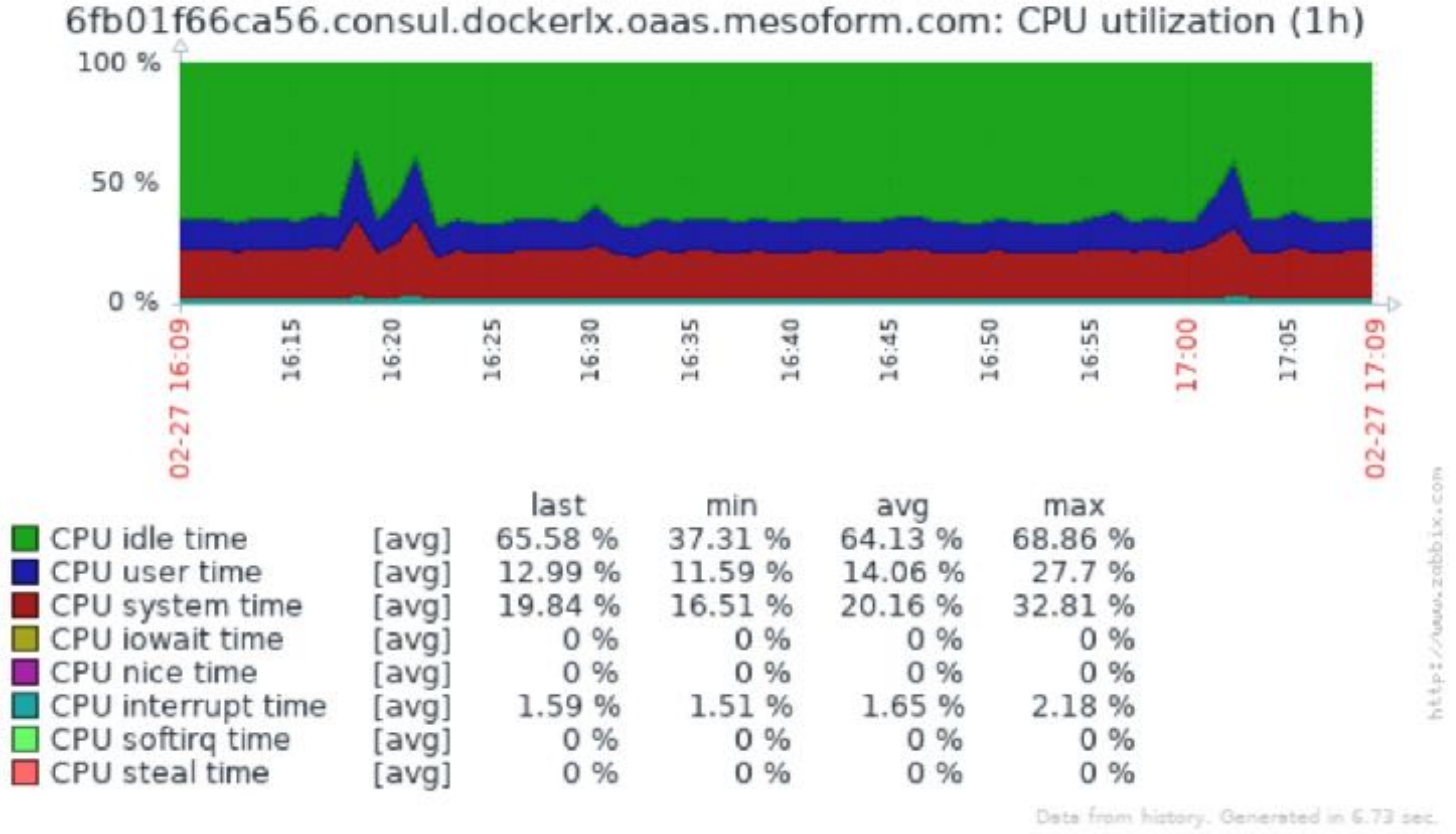
		last	min	avg	max
🟩 Consul::metrics:collect::timers	[all]	73.3ms	45.9ms	90.2ms	142.1ms

Data from history. Generated in 5.05 sec.



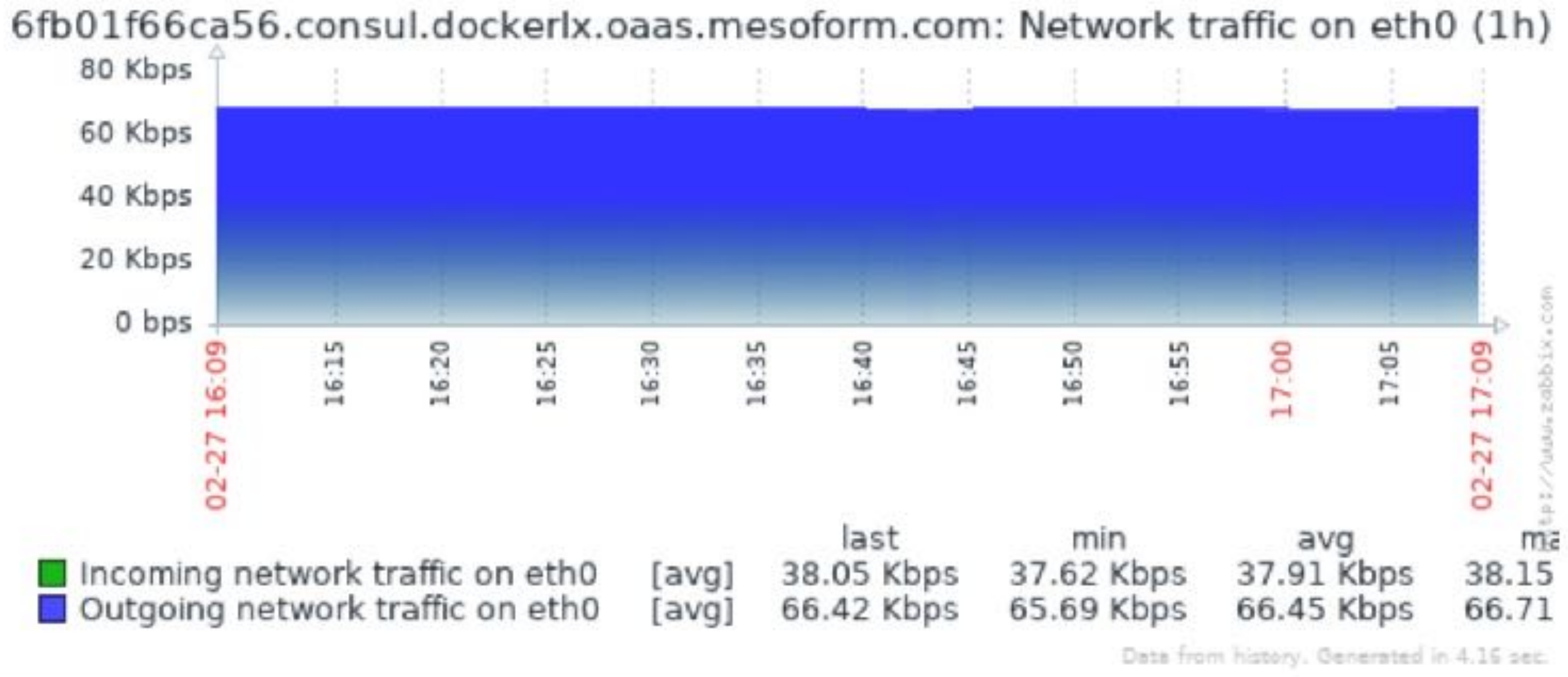
		last	min	avg	max
🟩 Available memory	[avg]	87.55 MB	86.2 MB	87.41 MB	87.59 MB

Data from history. Generated in 3.35 sec.



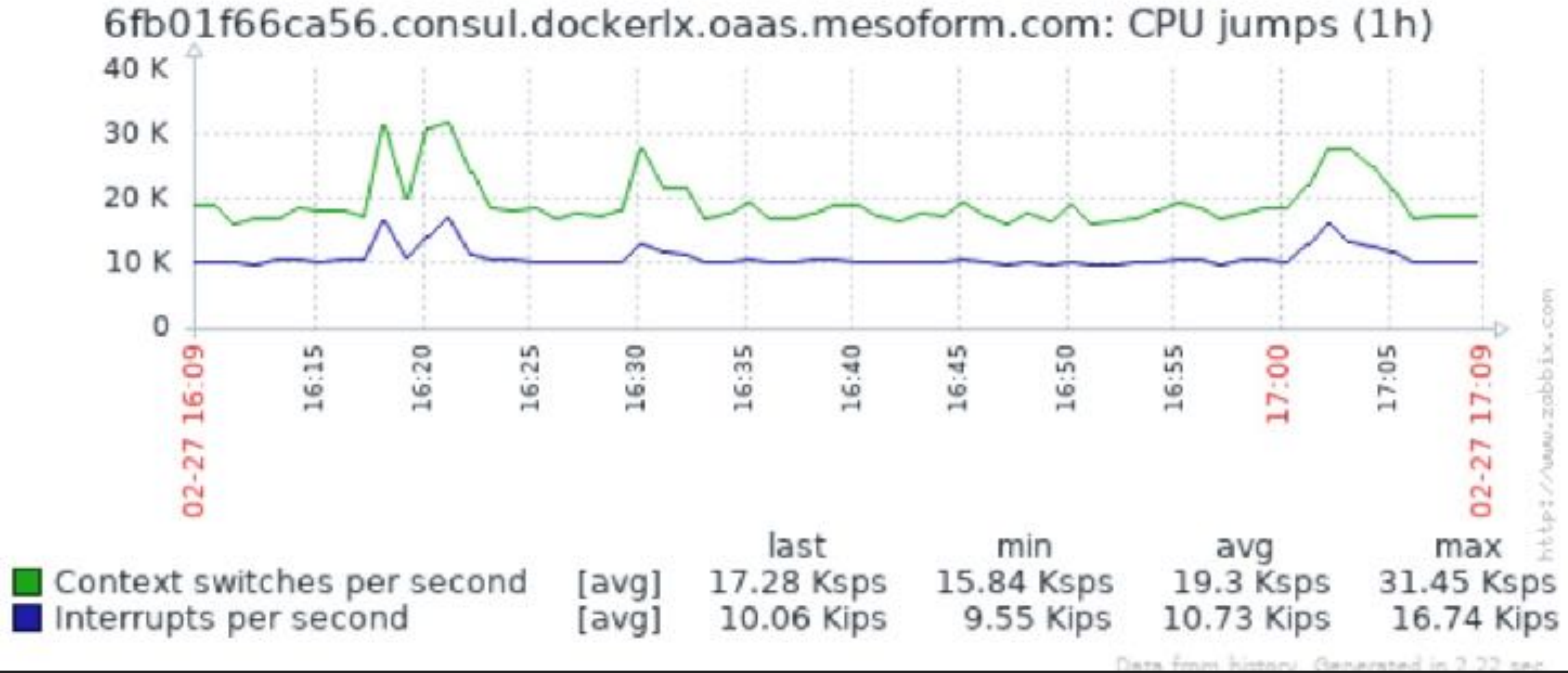
		last	min	avg	max
🟩 CPU idle time	[avg]	65.58 %	37.31 %	64.13 %	68.86 %
🟦 CPU user time	[avg]	12.99 %	11.59 %	14.06 %	27.7 %
🟥 CPU system time	[avg]	19.84 %	16.51 %	20.16 %	32.81 %
🟨 CPU iowait time	[avg]	0 %	0 %	0 %	0 %
🟪 CPU nice time	[avg]	0 %	0 %	0 %	0 %
🟦 CPU interrupt time	[avg]	1.59 %	1.51 %	1.65 %	2.18 %
🟩 CPU softirq time	[avg]	0 %	0 %	0 %	0 %
🟥 CPU steal time	[avg]	0 %	0 %	0 %	0 %

Data from history. Generated in 6.73 sec.

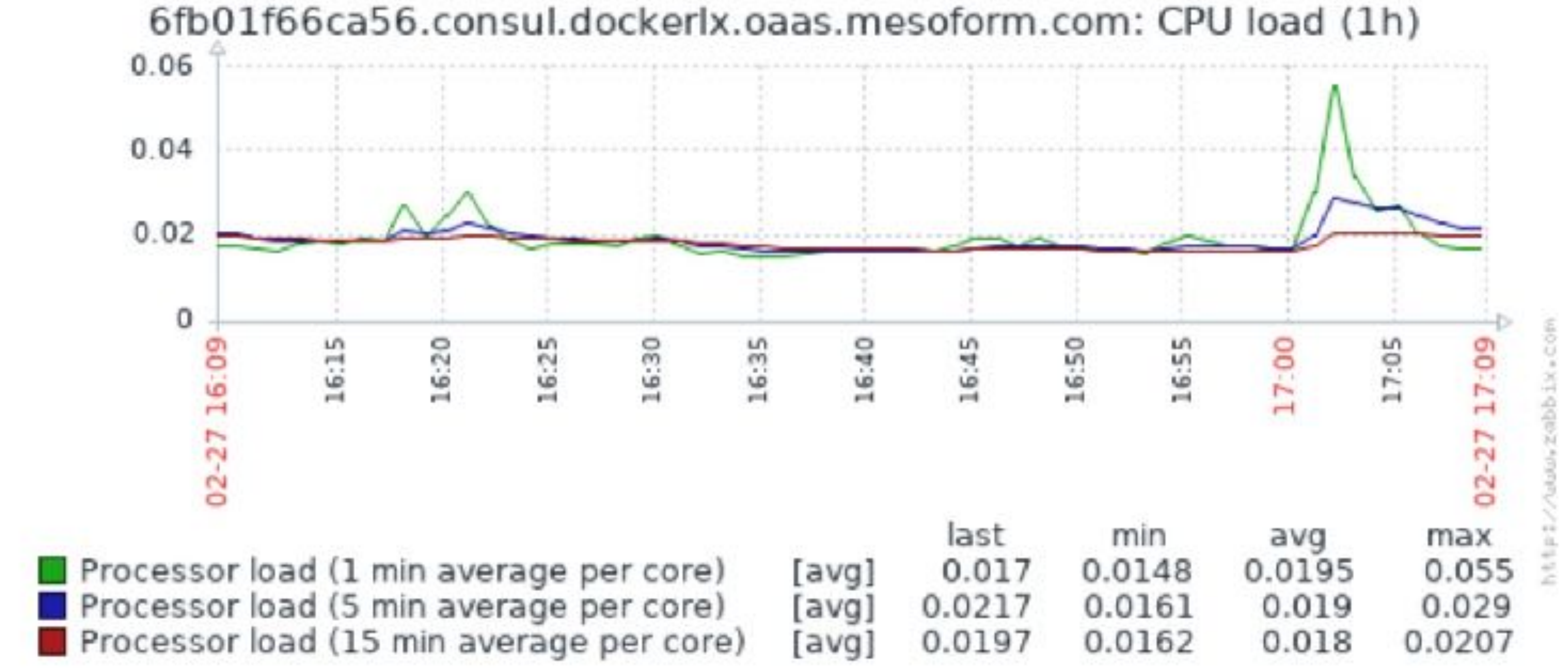


		last	min	avg	ma
🟩 Incoming network traffic on eth0	[avg]	38.05 Kbps	37.62 Kbps	37.91 Kbps	38.15
🟦 Outgoing network traffic on eth0	[avg]	66.42 Kbps	65.69 Kbps	66.45 Kbps	66.71

Data from history. Generated in 4.16 sec.

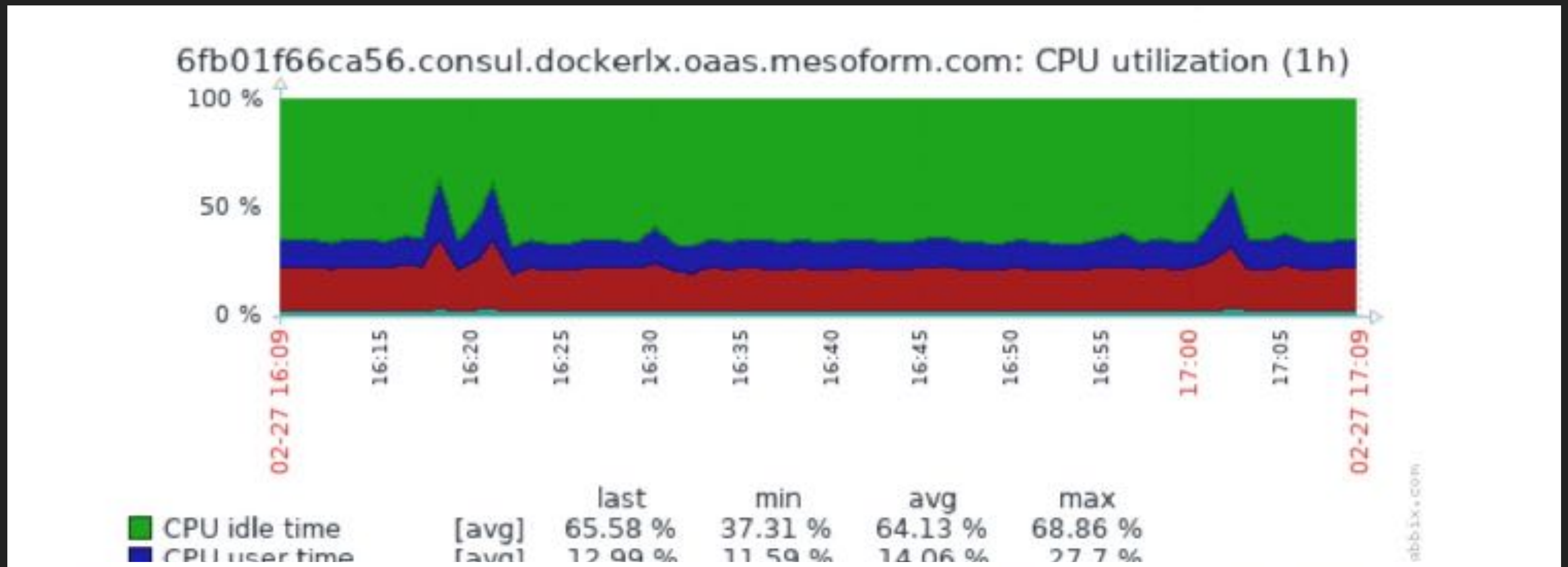


		last	min	avg	max
🟩 Context switches per second	[avg]	17.28 Ksps	15.84 Ksps	19.3 Ksps	31.45 Ksps
🟦 Interrupts per second	[avg]	10.06 Kips	9.55 Kips	10.73 Kips	16.74 Kips

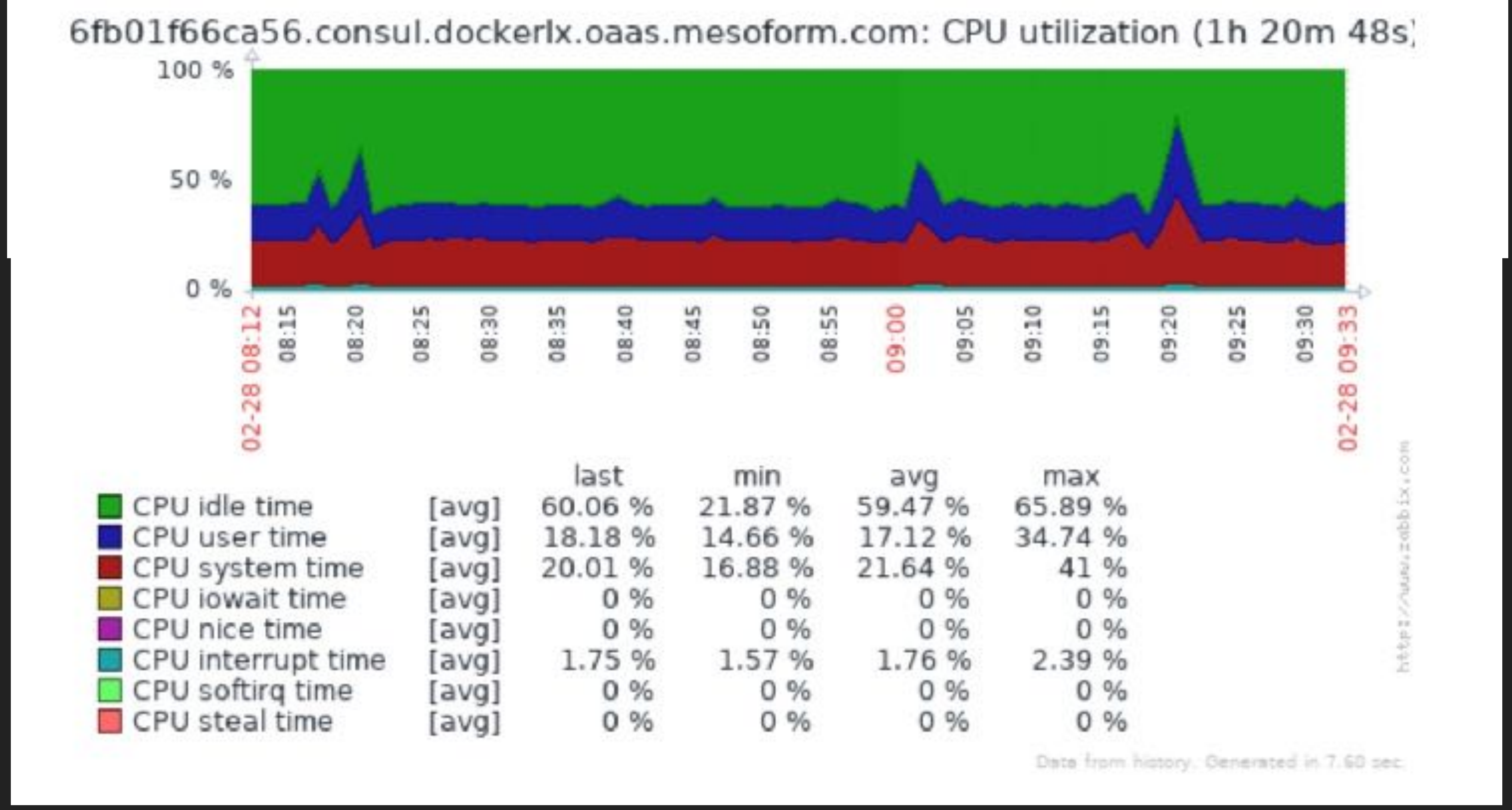


		last	min	avg	max
🟩 Processor load (1 min average per core)	[avg]	0.017	0.0148	0.0195	0.055
🟦 Processor load (5 min average per core)	[avg]	0.0217	0.0161	0.019	0.029
🟥 Processor load (15 min average per core)	[avg]	0.0197	0.0162	0.018	0.0207

CPU UTILISATION



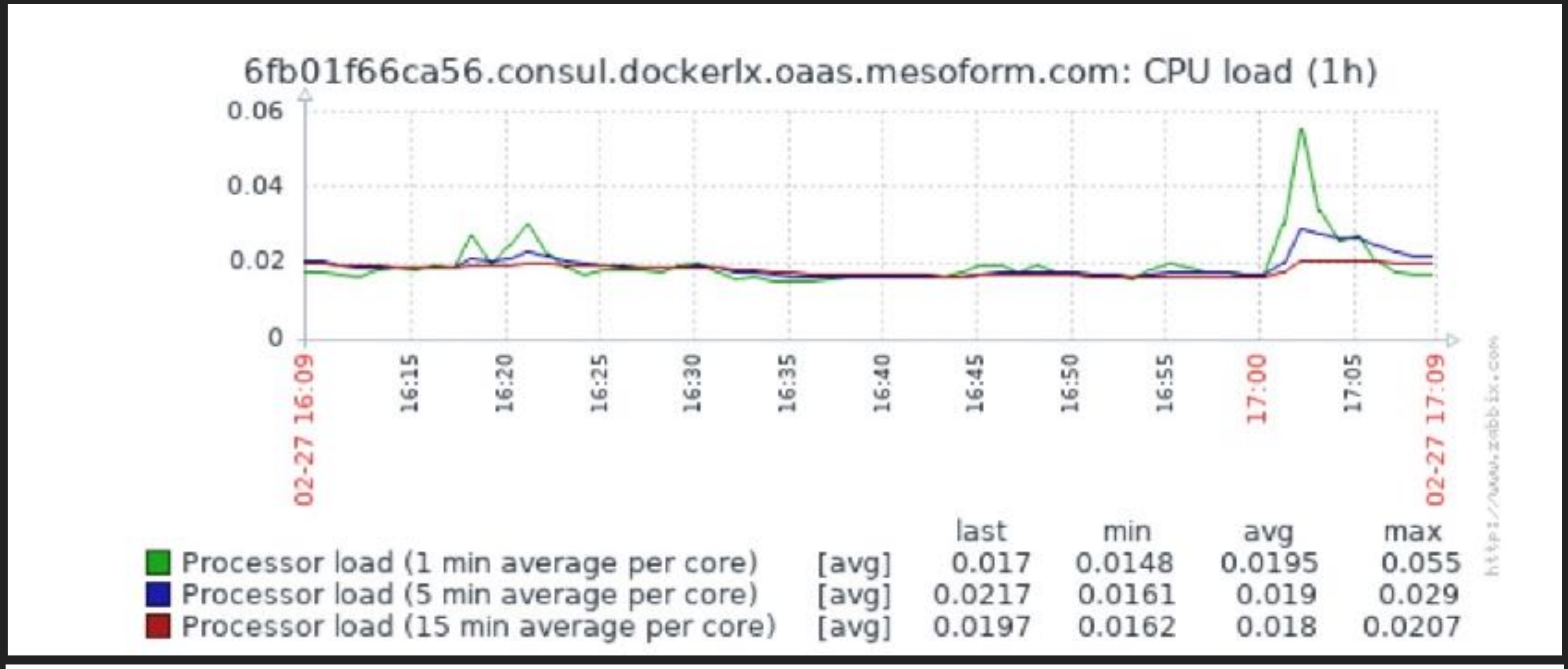
		last	min	avg	max
🟩 CPU idle time	[avg]	65.58 %	37.31 %	64.13 %	68.86 %
🟦 CPU user time	[avg]	12.99 %	11.59 %	14.06 %	27.7 %



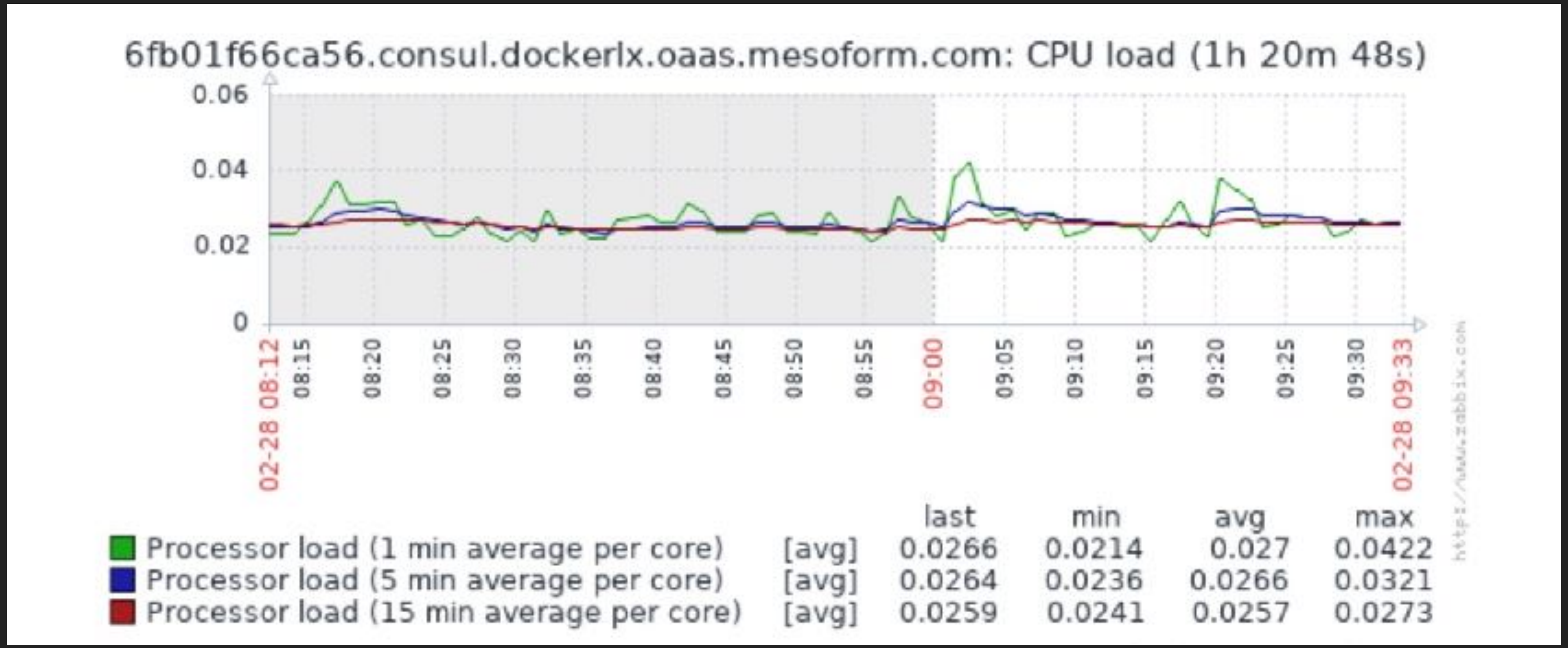
		last	min	avg	max
🟩 CPU idle time	[avg]	60.06 %	21.87 %	59.47 %	65.89 %
🟦 CPU user time	[avg]	18.18 %	14.66 %	17.12 %	34.74 %
🟥 CPU system time	[avg]	20.01 %	16.88 %	21.64 %	41 %
🟨 CPU iowait time	[avg]	0 %	0 %	0 %	0 %
🟪 CPU nice time	[avg]	0 %	0 %	0 %	0 %
🟦 CPU interrupt time	[avg]	1.75 %	1.57 %	1.76 %	2.39 %
🟩 CPU softirq time	[avg]	0 %	0 %	0 %	0 %
🟥 CPU steal time	[avg]	0 %	0 %	0 %	0 %

Data from history. Generated in 7.60 sec.

LOAD

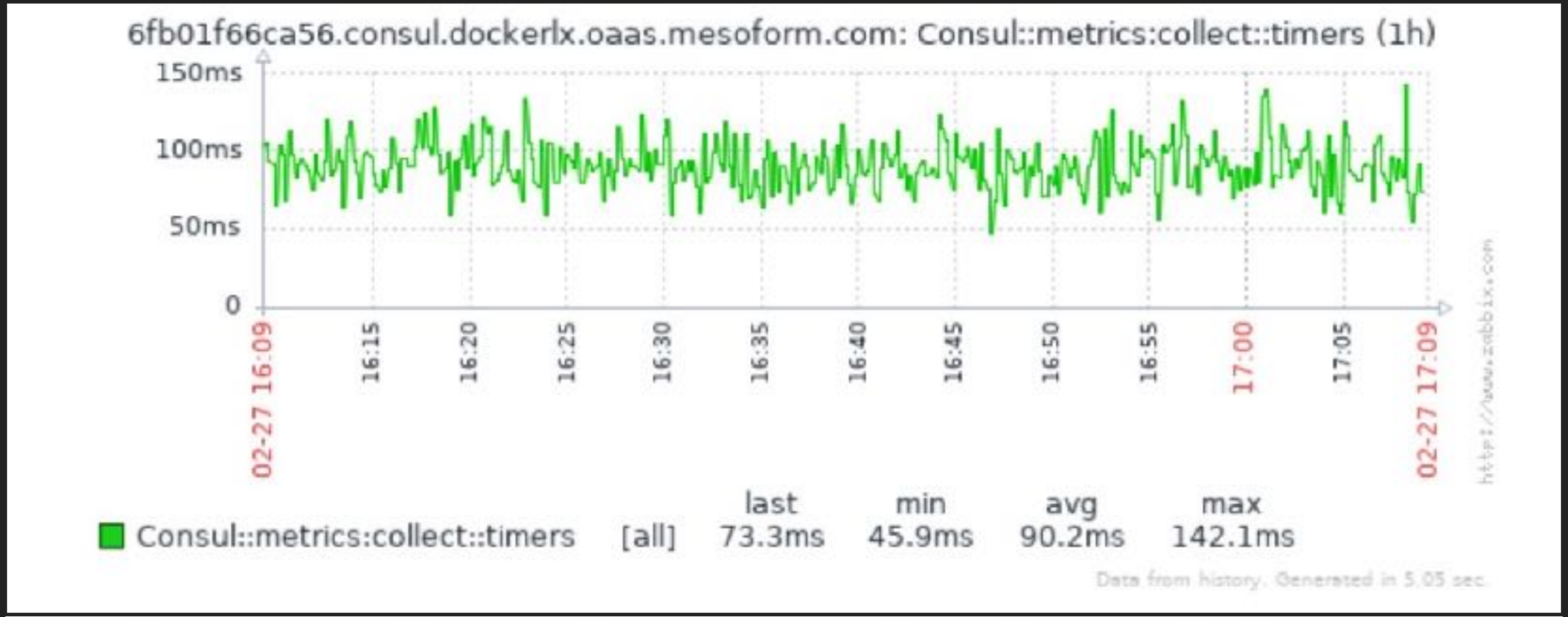


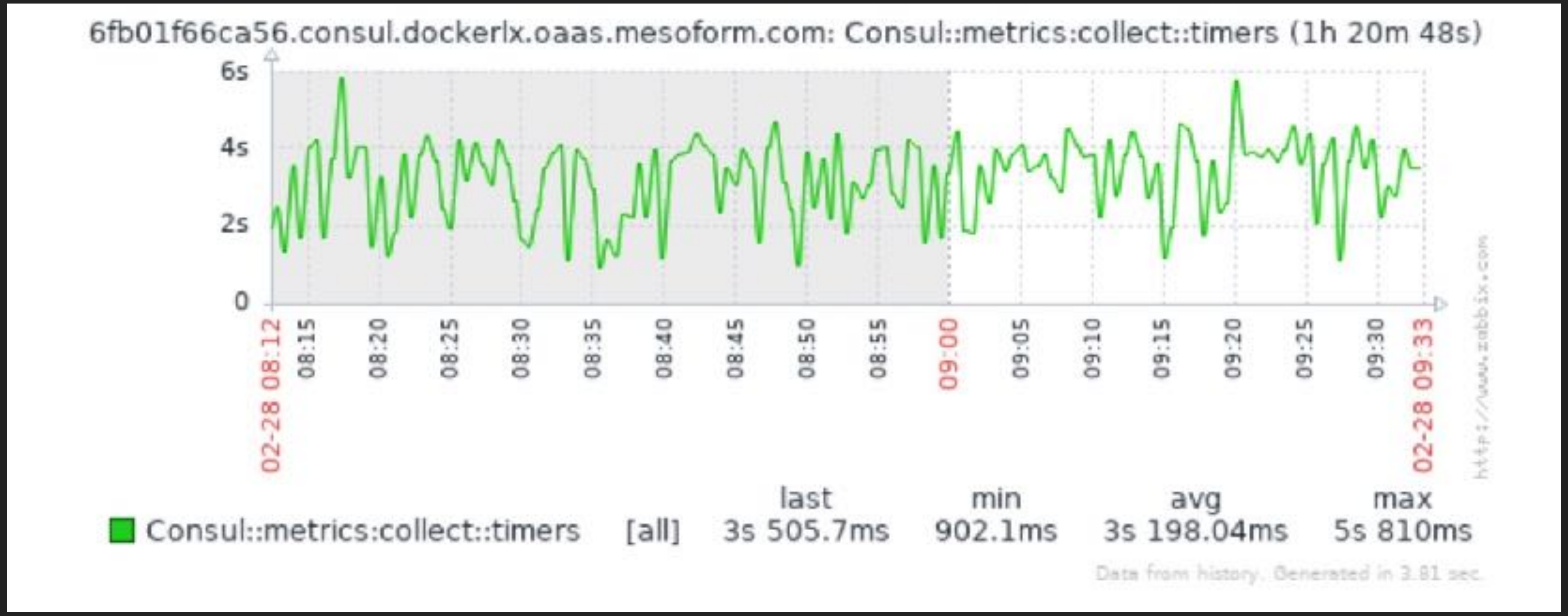
		last	min	avg	max
🟩 Processor load (1 min average per core)	[avg]	0.017	0.0148	0.0195	0.055
🟦 Processor load (5 min average per core)	[avg]	0.0217	0.0161	0.019	0.029
🟥 Processor load (15 min average per core)	[avg]	0.0197	0.0162	0.018	0.0207

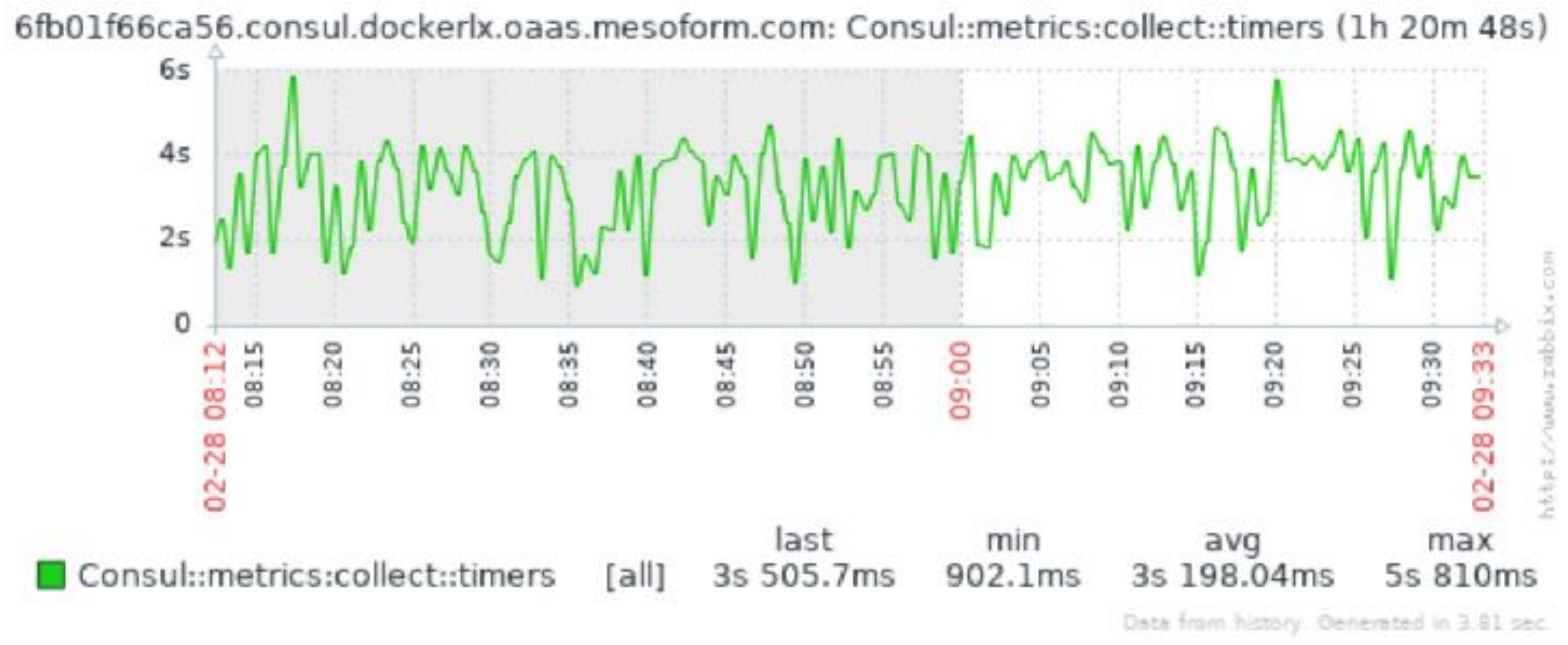


		last	min	avg	max
🟩 Processor load (1 min average per core)	[avg]	0.0266	0.0214	0.027	0.0422
🟦 Processor load (5 min average per core)	[avg]	0.0264	0.0236	0.0266	0.0321
🟥 Processor load (15 min average per core)	[avg]	0.0259	0.0241	0.0257	0.0273

TIMING

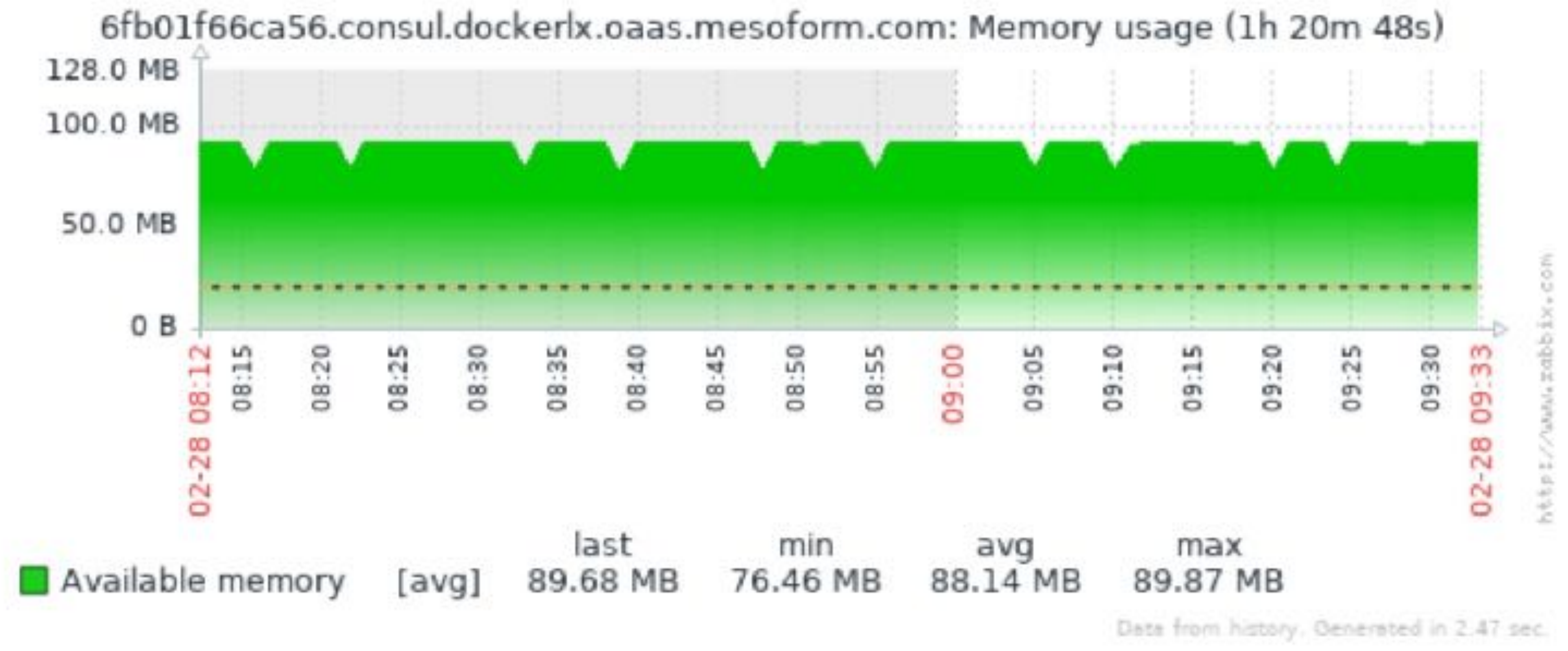

6fb01f66ca56.consul.dockerlx.oaas.mesoform.com: Consul::metrics:collect::timers (1h)
150ms
100ms
50ms
0
02-27 16:09
16:15
16:20
16:25
16:30
16:35
16:40
16:45
16:50
16:55
17:00
17:05
02-27 17:09
http://www.zabbix.com
last
min
avg
max
🟩 Consul::metrics:collect::timers
[all]
73.3ms
45.9ms
90.2ms
142.1ms
Data from history. Generated in 5.05 sec.


6fb01f66ca56.consul.dockerlx.oaas.mesoform.com: Consul::metrics:collect::timers (1h 20m 48s)
6s
4s
2s
0
02-28 08:12
08:15
08:20
08:25
08:30
08:35
08:40
08:45
08:50
08:55
09:00
09:05
09:10
09:15
09:20
09:25
09:30
02-28 09:33
http://www.zabbix.com
last
min
avg
max
🟩 Consul::metrics:collect::timers
[all]
3s 505.7ms
902.1ms
3s 198.04ms
5s 810ms
Data from history. Generated in 3.81 sec.



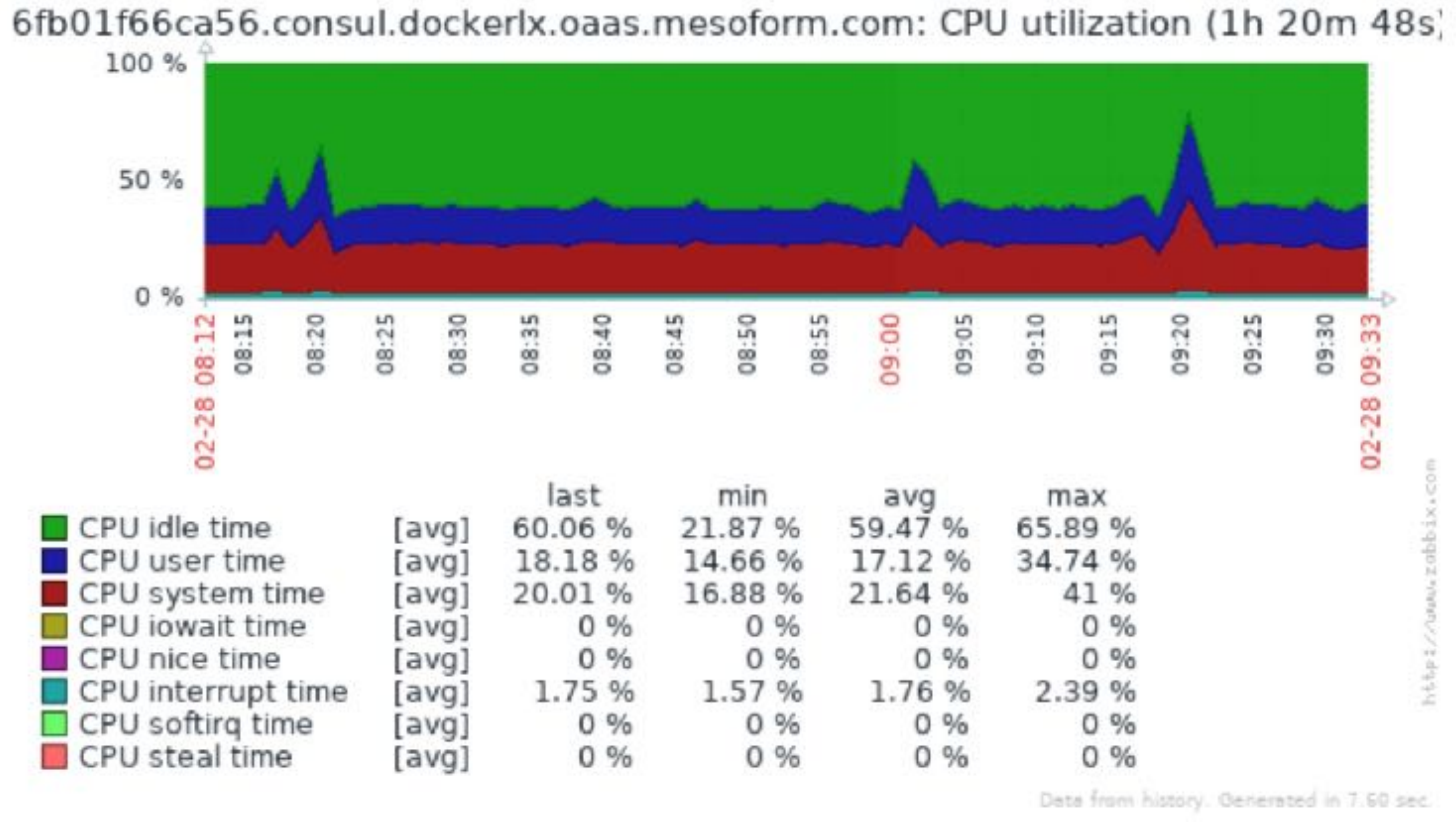
		last	min	avg	max
🟩 Consul::metrics:collect::timers	[all]	3s 505.7ms	902.1ms	3s 198.04ms	5s 810ms

Data from history. Generated in 3.81 sec.



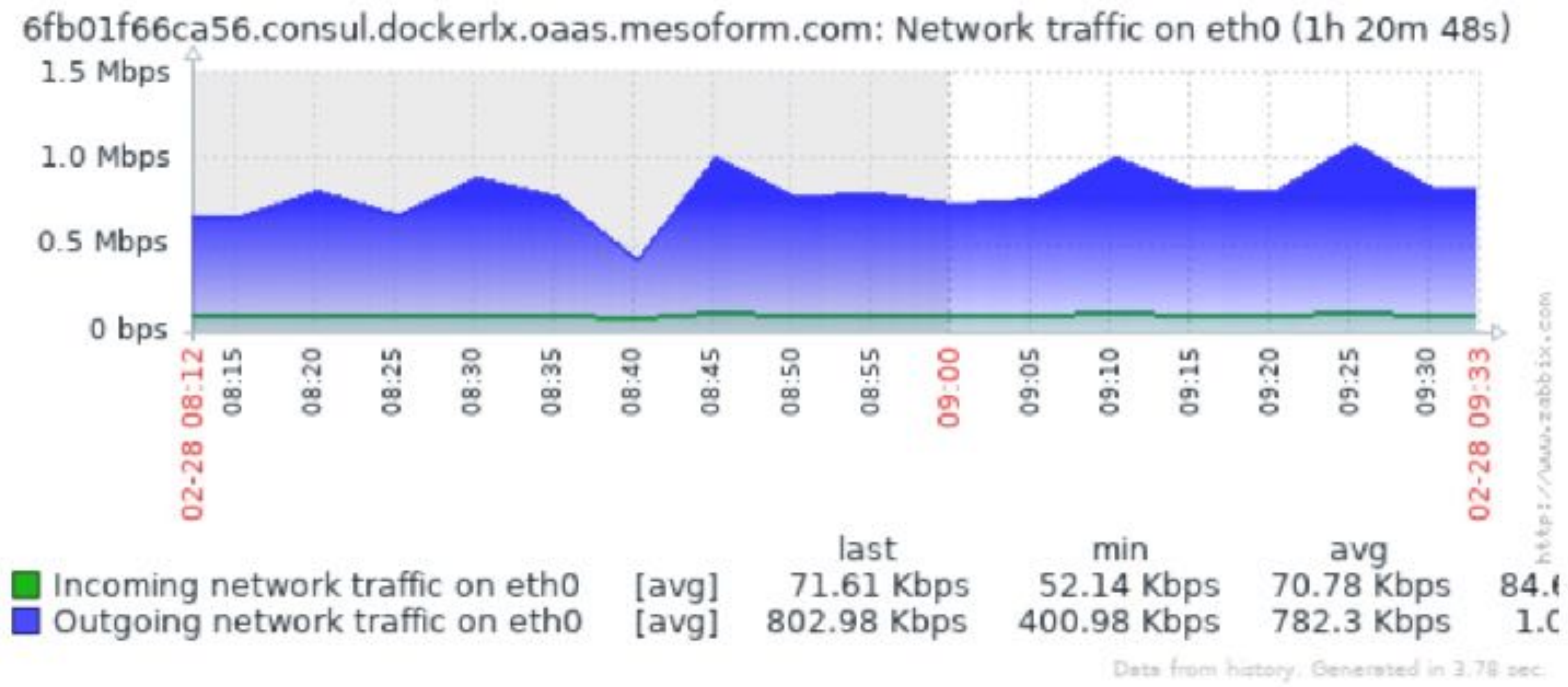
		last	min	avg	max
🟩 Available memory	[avg]	89.68 MB	76.46 MB	88.14 MB	89.87 MB

Data from history. Generated in 2.47 sec.



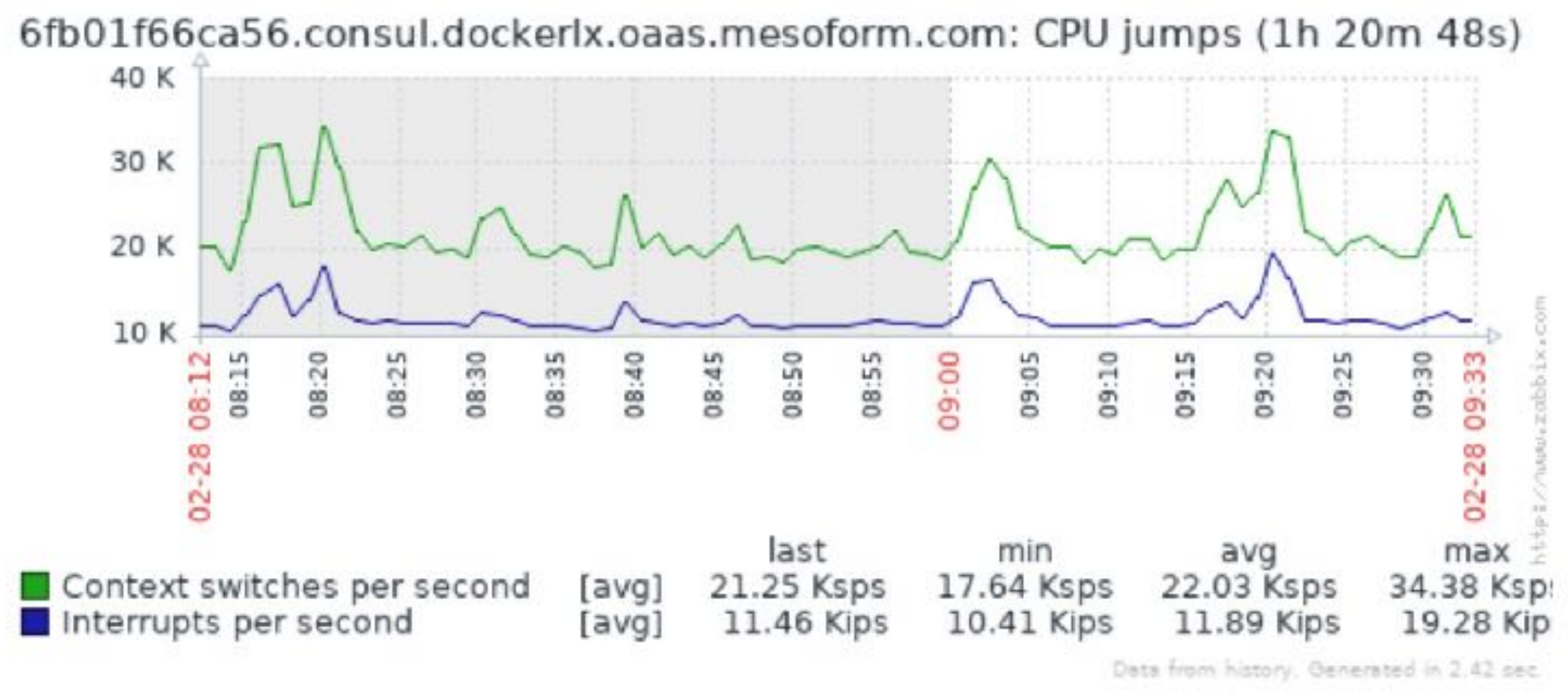
		last	min	avg	max
🟩 CPU idle time	[avg]	60.06 %	21.87 %	59.47 %	65.89 %
🟦 CPU user time	[avg]	18.18 %	14.66 %	17.12 %	34.74 %
🟥 CPU system time	[avg]	20.01 %	16.88 %	21.64 %	41 %
🟨 CPU iowait time	[avg]	0 %	0 %	0 %	0 %
🟪 CPU nice time	[avg]	0 %	0 %	0 %	0 %
🟦 CPU interrupt time	[avg]	1.75 %	1.57 %	1.76 %	2.39 %
🟩 CPU softirq time	[avg]	0 %	0 %	0 %	0 %
🟥 CPU steal time	[avg]	0 %	0 %	0 %	0 %

Data from history. Generated in 7.60 sec.



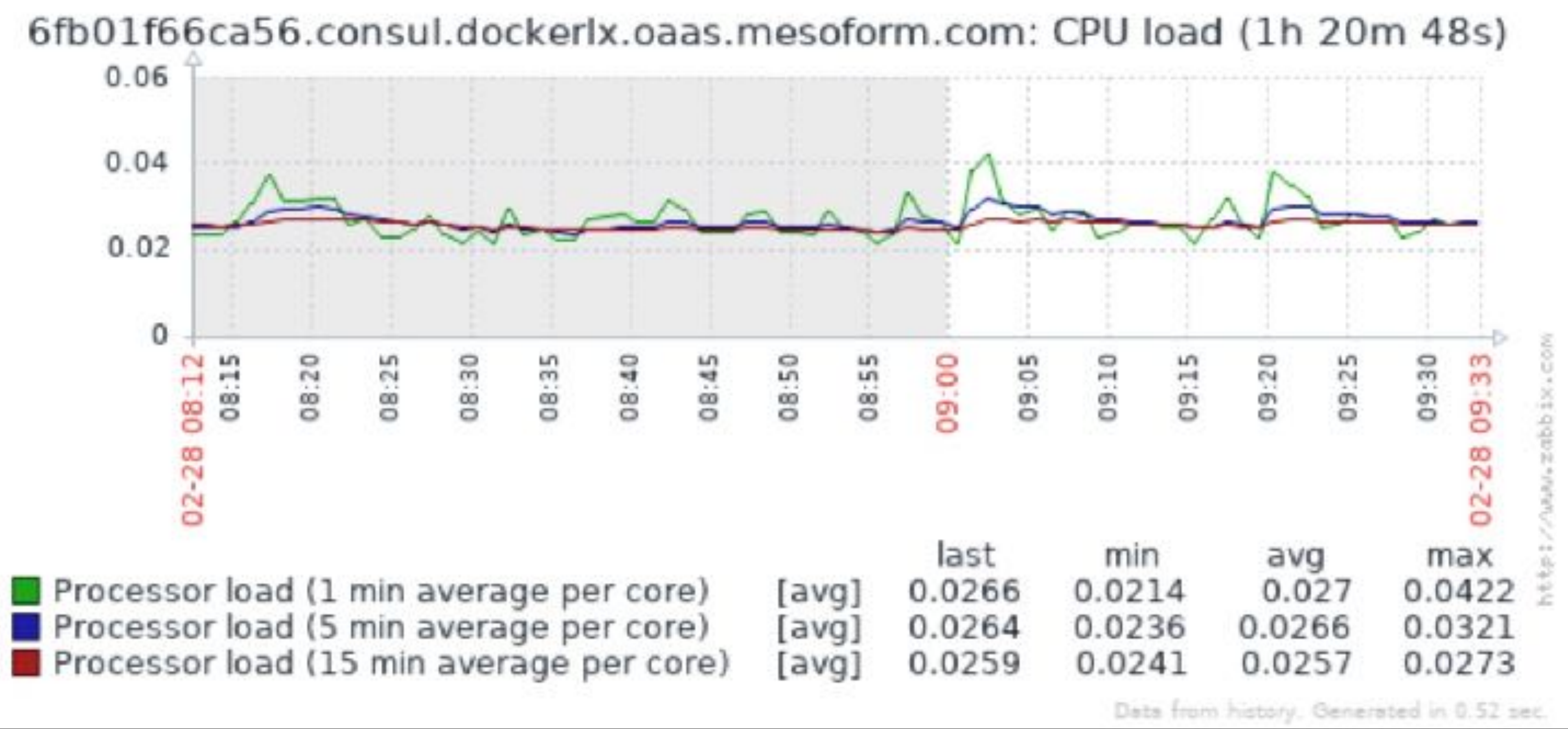
		last	min	avg	
🟩 Incoming network traffic on eth0	[avg]	71.61 Kbps	52.14 Kbps	70.78 Kbps	84.(
🟦 Outgoing network traffic on eth0	[avg]	802.98 Kbps	400.98 Kbps	782.3 Kbps	1.(

Data from history. Generated in 3.78 sec.



		last	min	avg	max
🟩 Context switches per second	[avg]	21.25 Ksps	17.64 Ksps	22.03 Ksps	34.38 Ksp:
🟦 Interrupts per second	[avg]	11.46 Kips	10.41 Kips	11.89 Kips	19.28 Kip

Data from history. Generated in 2.42 sec.



		last	min	avg	max
🟩 Processor load (1 min average per core)	[avg]	0.0266	0.0214	0.027	0.0422
🟦 Processor load (5 min average per core)	[avg]	0.0264	0.0236	0.0266	0.0321
🟥 Processor load (15 min average per core)	[avg]	0.0259	0.0241	0.0257	0.0273

Data from history. Generated in 0.52 sec.

THE CONCIERGE COURIER

- 3rd party features
- No windowing
- High Performance
- Send to anywhere
- Pull from anywhere
- Agnostic

CONCIERGE_COURIER.PY

```
def discover_timers():
    """
    Output Zabbix formatted JSON of keys
    """
    # just for testing purposes, simply open a file with metrics
    with open("/tmp/metrics.json", "r") as metrics_file:
        keys = metrics_file.read()
        keys_json = json.loads(keys)

        discovery_data_dict = \
            {'data': [{"{#TIMER}": key} for key in keys_json['timers']]}
        print(json.dumps(discovery_data_dict))
```

CONCIERGE_COURIER.PY

```
def get_timers():
    with open("/tmp/metrics.json", "r") as metrics_file:
        keys = metrics_file.read()
        keys = json.loads(keys)
        with open("/tmp/timer_metrics_zabbix.sender", "w") as sender_file:
            for timer_name, metrics in keys['timers'].items():
                for metric_name, metric_value in metrics.items():
                    sender_file.write("- timer[{0}.{1}] {2}\n"
                        .format(timer_name, metric_name, metric_value))
    send_metrics("timer")


def send_metrics(metric_type):
    filename = "/tmp/" + metric_type + "_metrics_zabbix.sender"
    call("zabbix_sender -c /etc/coprocesses/zabbix/zabbix_agentd.conf -i "
       + filename + " >/dev/null", shell=True)
    print time.time() - startTime
```

THE ENFIELD METHOD

- Accurate, single-shot, immediate feedback
- Like the rifle
- Backoff under network issues
- Greater confidence in container state
- Greater confidence in state of whole system
- More frequent updates

STATE TO STATE

- State in service discovery
- State in event management
- End-to-end view of whole system
- State history
- Dev/Ops on the same page
- State manipulation!

STATE CONTROL

- Consul keeps configuration state
- Monitoring performance and availability state
- Dynamic Asset database
- Automate scheduling, scaling, archiving

WHAT IF I TOLD YOU
SIRI IS A PERSONAL ASSISTANT, NOT A TALK-BUDDY
quickmeme.com

THE CONCIERGE SCHEDULER

- Containers Auto-register
- Push & pull state
- Optimised over many years
- Grouping containers by service
- Data about whole system
- Basically just runs *docker-compose scale*

SCALING

- Complex trigger profiles
- Pre-scaling using a predictive trigger
- Compare upstream service performance as well
- Vertical scaling
- Escalation steps
- Scaling events and problem events in one system

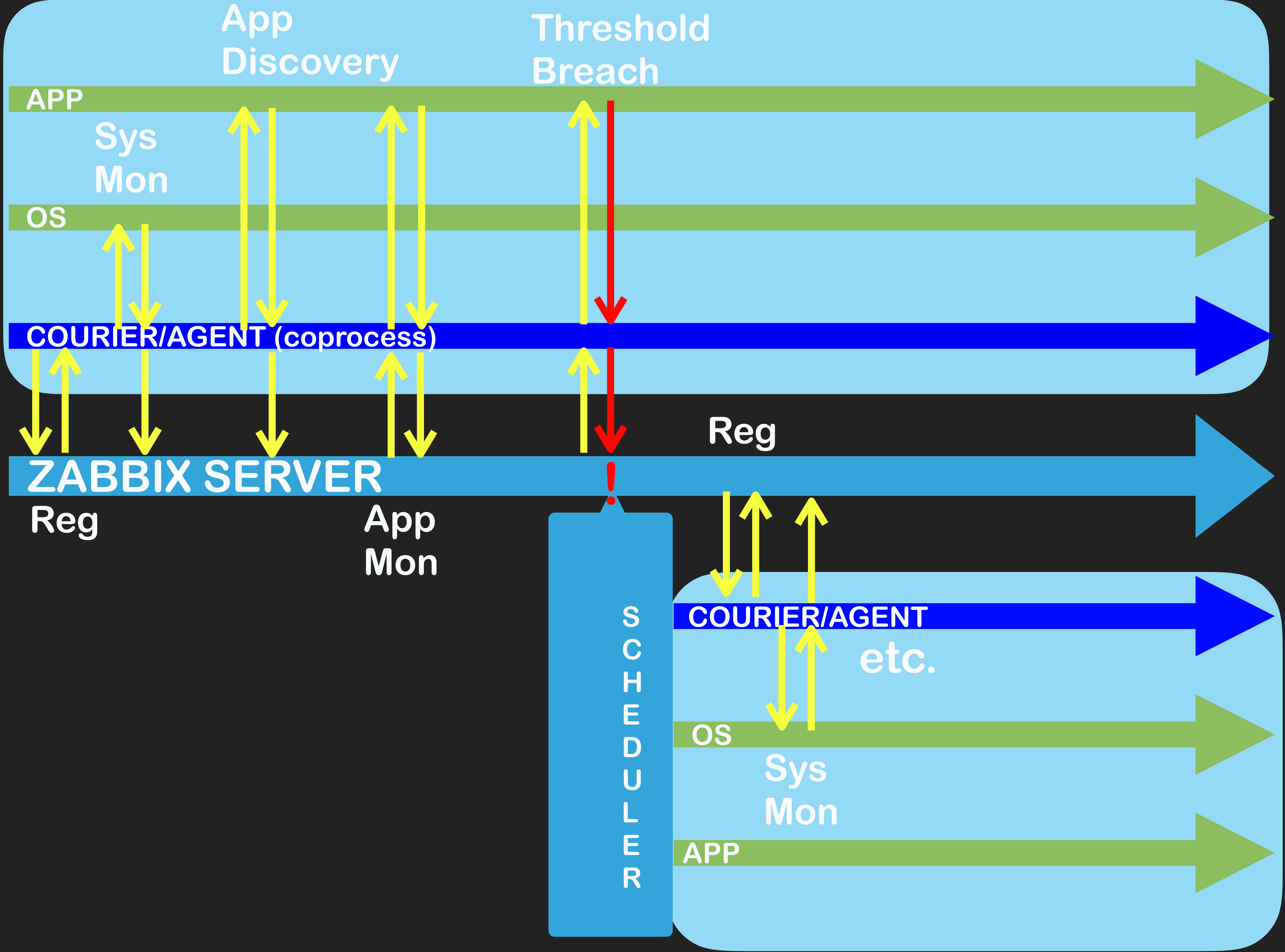
CONCIERGE_SCHEDULER.SH

```
# Variable assignment
action=$1; service_name=$2; current_scale=$3; increment=$4

scale_service(){
    /usr/bin/docker-compose --tlsverify --tlscert=${DOCKER_CERT_PATH}cert.pem \
       --tlscacert=${DOCKER_CERT_PATH}ca.pem \
       --tlskey=${DOCKER_CERT_PATH}key.pem --project-name dockerlx \
       --host tcp://dockerapi-private-lab1.mesoform.com:2376 --file /tmp/docker-compose.yml \
       scale ${service_name}=$1
    echo "$(date): Scaled ${service_name} from ${current_scale} to $1" \
       >> /tmp/app_scheduler_output
    exit 0
}

scale_up(){
    desired_scale=$((current_scale + increment))
    scale_service ${desired_scale}
}

scale_down(){
    desired_scale=$((current_scale - increment))
    scale_service ${desired_scale}
}
```


App
Discovery
Threshold
Breach
APP
Sys
Mon
OS
COURIER/AGENT (coprocess)
Reg
ZABBIX SERVER
!
Reg
App
Mon
SCHEDULER
COURIER/AGENT
etc.
OS
Sys
Mon
APP

KEEPING ACTIVE

- Works for bare metal, VMs and containers
- application-level guarantees
- Active versus passive
- Troubleshooting directly connected services
- Performance and Reliability

CONCLUSION

- Autopilot Pattern and Enfield Method
- We're already: doing event management, auto-registering, aggregating metrics, performing actions on triggers, maintaining system state, highly optimised, self-healing,
- Controlling the state
- Accuracy and performance
- Short lead time

WHATS NEXT

- Load testing
- Use Zabbix Python interpreter module
- Key management with vault
- Swarm/Nomad not docker compose
- DevOps everything!

SO LONG AND THANKS FOR ALL THE FISH

- Read the full article at http://www.mesoform.com/blog-listing/info/the-concierge-paradigm
- Search: "concierge paradigm"
- @MesoformLtd
- /mesoform
- /mesoform
- http://www.mesoform.com/contact-us