

### WEBINAR MONITOREZ VOS APPLICATIONS



Lionel Porcheron CEO & co-fondateur de Bleemeo bleemeo.com



#### Who am I?

#### **Lionel Porcheron**, CEO & co-founder Bleemeo

- •Ops background, managing 500+ machines in classical DC and in the Cloud
- DevOps for +15 years (started my monitoring journey with nagios netsaint)
- Toulouse DevOps Meetup Leader, Capitole du Libre Leader, PyconFR 2017 Organizer

#### Bleemeo?

Observability & Monitoring as a service solution

Start monitoring your infrastructure in 30s

Prometheus, Graphite, StatsD, compatible

- 2 Open Source projects (<u>https://github.com/bleemeo</u>):
- **Glouton**, universal monitoring agent written in Go with Prometheus, Statsd, Graphite, Nagios compatibility
- **SquirrelDB**, a scalable Prometheus compatible storage backend based on Cassandra



#### From servers to the apps

•Server monitoring is now a standard

•Monitoring disk (space), memory, CPU, network, ...

•Services statistics became standard (with standard modules of monitoring software)

• Prometheus became de-facto standard for modern monitoring metric centric

•Application monitoring require a code effort... standards tools can help

#### **Teams Specialization**

•Team specialization allow to have a deeper knowledge of the stack/application

•Applications became more and more complex

•If you have application metrics and with application complexity (micro-services), a single sysadmin team can't manage all services

•Google introduced SRE (Site Reliability Engineer)

•Amazon introduced "you build it, you run it"

•Team specialization: running datacenter, running servers, running Cloud layer, running applications

#### Graphite... and Prometheus

•Graphite change the way we were doing monitoring: metrics became central

•Graphite appeared in 2008

•Big adoption in startup beginning in the 2010 years associated with StatsD

• Prometheus became de-facto standard for monitoring

•Prometheus was "initiated" in 2012 at Soundcloud and is now a (graduated) CNCF project





### Observability Key Metrics

The RED Method

- •(Request) Rate the number of requests, per second, you services are serving.
- •(Request) Errors the number of failed requests per second.
- •(Request) **D**uration distributions of the amount of time each request takes.

The USE Method

- •(Ressource) Utilization: as a percent over a time interval. eg, "one disk is running at 90% utilization".
- •(Ressource) Saturation: as a queue length. eg, "the CPUs have an average run queue length of four".
- •(Ressource) Errors: scalar counts. eg, "this network interface has had fifty late collisions".

#### StatsD

StatsD build passing Chat on gitter docker pulls 333k

A network daemon that runs on the Node.js platform and listens for statistics, like counters and timers, sent over UDP or TCP and sends aggregates to one or more pluggable backend services (e.g., Graphite).

•Created by Etsy (US online marketplace), inspired by Flickr (photo sharing service)

•Receive data from your application and send them to a backend (usually Graphite)

•StatsD + Graphite was a popular association in numerous startups

#### StatsD principles

- Applications push data to StatsD
- Metric value can be:
  - Counter: number sent by your application (e.g. number of active sessions)
  - Gauge: value with a state, can be increased/decreased (e.g. number of sales on your site)
  - Timing: time value



• You can find a library for your language

#### StatsD backends



• Prometheus propose a StatsD exporter

++		++			++			
StatsD	(UDP/TCP	repeater)>	statsd_exporter	<(scrape	/metrics)	Prometheus	1	
++		-+	++					

#### StatsD example in Python

Incrementing a counter

•Sending a timer

>>> import statsd
>>> c = statsd.StatsClient('localhost', 8125)
>>> c.incr('foo') # Increment the 'foo' counter.
>>> c.timing('stats.timed', 320) # Record a 320ms 'stats.timed'.

#### Push & Pull strategies

•With StatsD, application push custom metrics and data

•StatsD libraries are light

•You need to deploy a StatsD daemon... and a backend (can be a SaaS solution)

• Prometheus is working the other way: Prometheus scrape data from your application

#### Prometheus Overview

- •A Time Series Database where data is identified by metric name and labels (key/value pairs)
- •A powerfull PromQL query language
- •No complex storage: designed to store multiple days (not weeks) of data
- •Data are collected via a pull over HTTP
- •A rich ecosystem with exporters (to get metrics) and web panels (query & display)

#### Prometheus Python example

•Example of Django web framework

•For Django application: django-prometheus

Django Middleware for metrics

52	MIDDLEWARE = [
53	'django_prometheus.middleware.PrometheusBeforeMiddleware',
54	'django.contrib.sessions.middleware.SessionMiddleware',
55	'django.contrib.auth.middleware.AuthenticationMiddleware',
56	'django.contrib.messages.middleware.MessageMiddleware',
57	'django_prometheus.middleware.PrometheusAfterMiddleware',
58	]

korfuri / <b>django-prom</b> e	theus	TUsed by		Vatch - 21	★ Star	490 <b>% Fork</b> 127
<> Code ① Issues 24	Pull requests 8 III Project	s 0 💷 Wiki 🕕 Se	ecurity 🔟 Insi	ghts		
xport Django monitoring m	etrics for Prometheus.io					
prometheus django django	prometheus python monitoring	g exported-metrics m	etrics promethe	us-client		
⑦ 262 commits	ទ្រៃ 9 branches	♥ 27 releases	<u>\$</u> 27	contributors		彝 Apache-2.0
Branch: master - New pull re	quest		Create new file	Upload files	Find file	Clone or download <del>-</del>
asherf Merge pull request #13	from asherf/dups				Latest com	nit 016fa7f 6 days ago
django_prometheus	Merge pull request #133 fro	m asherf/dups				6 days ago
documentation	Use range instead of xrange	e for Python 3 compatibili	ty			24 days ago
examples	Change config.file declaration	on from .conf to .yml				last month
.gitignore	Added possibility to export r	netrics of several caches	backends (#69)			2 years ago
.travis.yml	use black					11 days ago
	Fix markdown lint issues.					last month
	Initial commit					5 years ago
MANIFEST.in	Include LICENSE file on pyp	bi.io				2 months ago
README.md	use black					11 days ago
requirements.txt	use black					11 days ago
setup.cfg	pep8 is deprecated, use py	codestyle				last month
setup.py	Add support for Python 3.8					15 days ago

#### Prometheus metrics

•Prometheus metrics endpoint is a plain text "web page"

•Human readable

- Scaped by a Prometheus server
- •Data can be queried with PromQL

•Can be used by Prometheus ecosystem: Alert Manager, Grafana...

$\leftarrow \rightarrow C \ c$
diango http requests latency seconds by view method bucket{le="+Inf",method="GET",view="prometheus-diango-metrics"} 1.0
django http requests latency seconds by view method count{method="GET",view="prometheus-django-metrics"} 1.0
django http requests latency seconds by view method sum{method="GET",view="prometheus-django-metrics"} 0.004497956004342996
django http requests latency seconds by view method bucket{le="0.01",method="GET",view=" <unnamed view="">"} 1.0</unnamed>
django http requests latency seconds by view method bucket{le="0.025",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django http requests latency seconds by view method bucket{le="0.05",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django http requests latency seconds by view method bucket{le="0.075",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django http requests latency seconds by view method bucket{le="0.1",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="0.25",method="GET",view=" <unnamed_view>"} 2.0</unnamed_view>
django_http_requests_latency_seconds_by_view_method_bucket{le="0.5",method="GET",view=" <unnamed_view>"} 2.0</unnamed_view>
django_http_requests_latency_seconds_by_view_method_bucket{le="0.75",method="GET",view=" <unnamed_view>"} 2.0</unnamed_view>
django_http_requests_latency_seconds_by_view_method_bucket{le="1.0",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="2.5",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="5.0",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="7.5",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="10.0",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="25.0",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="50.0",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="75.0",method="GET",view=" <unnamed view="">"} 2.0</unnamed>
django_http_requests_latency_seconds_by_view_method_bucket{le="+int",method="GEI",view=" <unnamed view="">"} 2.0</unnamed>
django_nttp_requests_tatency_seconds_by_view_method_count{method="GEL",view=" <unnamed_views"} 2.0<="" th=""></unnamed_views"}>
ajango http:requests latency seconds by view method sum{method="061",view=" <unnamed view="">"} 0.0159/734599/88459</unnamed>
django nttp requests latency seconds by View method bucket(le="0.01", method="GLI", View="Dieemed quote.views.index") 680.0
diango http:requests latency seconds by view method bucket[le="0.025",method="GET",view="bleemed quote.views.index"] 600.0
diango http://duests_latency_seconds_by_view_method_bucket{te= 0.05, method="CET", view= bitemeo_ducte.views.index 9 609.0
diange http://duests_tatency_seconds_by_view_method_bucket{te= 0.073, method= 0t1, view= btemeo_quote.views.index # 0.02.0
diango_nttp_requests_latency_seconds_by_new_method_bucket[te=0.1;method="GFT"_view_=bleemeo_quote_views_index / 002.0
diango http://equests.latency.seconds.by.view.method_bucketfle="0.5", method="(FT", view="hleemeo.gucte.views.index").602.0
diango http://ducests_latency_seconds_by_view_method_bucketfle="0.75" method="6FT" view="http://ducests.latency_seconds_by_view_method_bucketfle="0.75" method="6FT" view="http://ducests.latency_seconds_by_view_method_by_view="http://ducests.latency_seconds_by_view_method_by_view_method_by_view="http://ducests.latency_seconds_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_method_by_view_metho
diango http://equests.latency.seconds.by.view.method.bucket{le="1.0".method="GET",view="bleemeo.guote.views.index"} 692.0
diango http:requests latency seconds by view method bucket{le="2.5",method="GET",view="bleemeo guote.views.index"} 692.0
diango http requests latency seconds by view method bucket{le="5.0",method="GET",view="bleemeo guote.views.index"} 692.0
diango http requests latency seconds by view method bucket{le="7.5",method="GET",view="bleemeo guote.views.index"} 692.0
diango http requests latency seconds by view method bucket{le="10.0".method="GET".view="bleemeo guote.views.index"} 692.0
diango http requests latency seconds by view method bucket{le="25.0",method="GET",view="bleemeo quote.views.index"} 692.0
django http requests latency seconds by view method bucket{le="50.0",method="GET",view="bleemeo quote.views.index"} 692.0
django http requests latency seconds by view method bucket{le="75.0",method="GET",view="bleemeo quote.views.index"} 692.0
django http requests latency seconds by view method bucket{le="+Inf",method="GET",view="bleemeo quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_count{method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_sum{method="GET",view="bleemeo_quote.views.index"} 1.1842275859380607
<pre># TYPE django_http_requests_latency_seconds_by_view_method_created gauge</pre>
django_http_requests_latency_seconds_by_view_method_created{method="GET",view="prometheus-django-metrics"} 1.572701763289514e+09
django_http_requests_latency_seconds_by_view_method_created{method="GET",view=" <unnamed 1.5727017632967687e+09<="" th="" views"}=""></unnamed>
django_http_requests_latency_seconds_by_view_method_created{method="GET",view="bleemeo_quote.views.index"} 1.5727018193420057e+09
# HELP django_http_requests_unknown_latency_total Count of requests for which the latency was unknown.

#### Server to apps coverage

• Prometheus can easily scrape metrics from:

- Servers (node exporter, cAdvisor, Bleemeo Glouton)
- Services (services exporters)
- Kubernetes (kube state exporter)
- Application (instrumenting your code)

•Allow to correlate usage metrics and system metrics

•Allow to diagnose issue with business and technical data



#### Complete Example

Small Django application showing quote of the day



#### Monitoring your logs

•StatsD and Prometheus can be used to monitor logs

- •A deamon on your server can parse logs and generate metrics
- •Number of requests, error rate, request time are common metrics

•Promtail can be used to expose a Prometheus endpoint from your logs

#### Apps monitoring with Bleemeo

- •Can replace Prometheus server and Grafana
- Open Source agent "Glouton" scrape Prometheus endpoint
- •All network streams are authenticated (different for each server) and encrypted (TLS 1.3)
- •Metrics can be queried and dashboards configured using PromQL queries
- •Service autodiscovery is built-in in the agent
- •Coming soon: alerts can be configured using PromQL queries



#### Monitoring as a Service

•Setup and scale Prometheus infrastructure can be time consuming

- •Using a Monitoring as a Service platform may help
- Multiple vendors offer Prometheus as a Service backend
- •Bleemeo is one of them :)
- •Focus on application development and business

#### Conclusion

•StatsD and Prometheus are tools to easily start application monitoring

- StatsD is very basic but very simple to integrate
- Prometheus is much more powerfull but require more work

•Prometheus will allow you to mix system, services and applications metrics

• Prometheus is de-facto standard for monitoring in 2021

Prometheus data are easy to query

•OpenTelemetry is another framework you may want to have a look to go further

# Questions?

## Try Bleemeo for Free: https://bleemeo.com/trial