Bleemeo

WEBINAR

Monitoring Security

23 Sept. 2022

11h00

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

Bleemeo?

Observability & Monitoring as a service solution

Monitor your Servers, Containers, and applications in 30s

Prometheus, Graphite, StatsD, Nagios, compatible

2 Open Source projects (https://github.com/bleemeo):

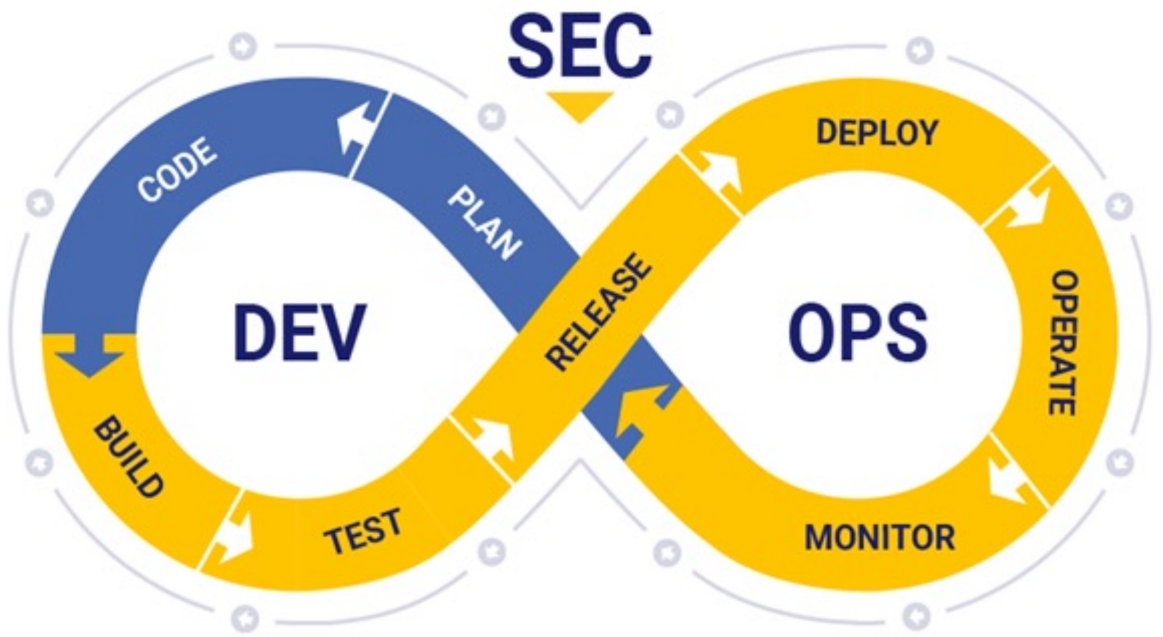
- **Glouton**, universal monitoring agent written in Go with Prometheus, StatsD, Graphite, Nagios compatibility
- **SquirrelDB**, a scalable Prometheus compatible storage backend based on Cassandra

Security Challenges

- Systems are more and more exposed to Internet
- Systems are more and more complex
- More and more security issues in software
- Everything is software: your phone, your laptop, VPN, AP wifi…
- An annual audit is no more enough. **Security evaluation must be performed continuously**
- You know people who have been hacked (at least ransomware)

DevSecOps

- Following DevOps principle, integrating Security concerns
- All teams need to integrate security in their design
- The ”corporate firewalls” are no more enough to warranty security



Security Information & Event Management

- Usually called ”SIEM”
- Splunk, Elastic have “SIEM” product offers
- Centralize all events from your infrastructure
- Detect security incidents that could be associated to events collected
- Detect unexpected behaviors
- Have dashboards and potentials alerts related to those events
- We are creating a simple SIEM in your monitoring tool

Security is not binary…
… it should be measurable

Monitoring Security

Metrics permit to measure infrastructure security issues

Metrics permit to identify what you should do first

Identify key metrics for monitoring your infrastructure security:

- Number of pending security patches to be applied
- Number of authentications failure
- Infrastructure key indicators: network bandwidth, CPU usage
- Applications errors rate

Key Metrics / “Golden Signals”

The RED Method

- (Request) **R**ate - the number of requests, per second, your services are serving.
- (Request) **E**rrors - the number of failed requests per second.
- (Request) **D**uration - distributions of the amount of time each request takes.

The USE Method

- (Resource) **U**tilization: as a percent over a time interval. e.g., "one disk is running at 90% utilization".
- (Resource) **S**aturation: as a queue length. e.g., "the CPUs have an average run queue length of four".
- (Resource) **E**rrors: scalar counts. e.g., "this network interface has had fifty late collisions".

Use Prometheus format!

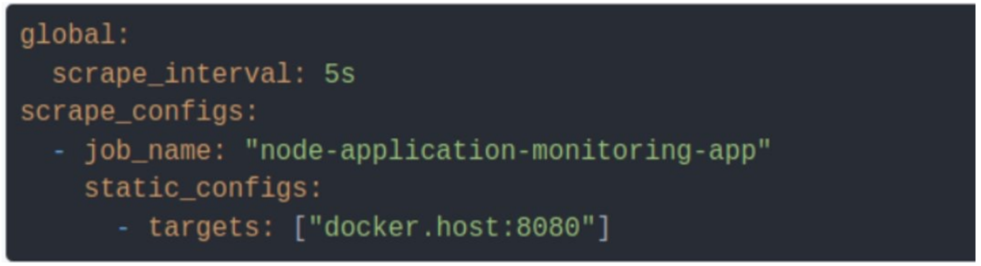
Prometheus in a nutshell

- Prometheus was "initiated" in 2012 at Soundcloud and is now a (graduated) CNCF project
- Prometheus became de-facto standard for monitoring
- A Time Series Database where data is identified by metric name and labels (key/value pairs)
- A powerful PromQL query language
- No complex storage: designed to store multiple days (not weeks) of data
- Data are collected via a poll over HTTP
- A rich ecosystem with exporters (to get metrics) and web panels (query & display)

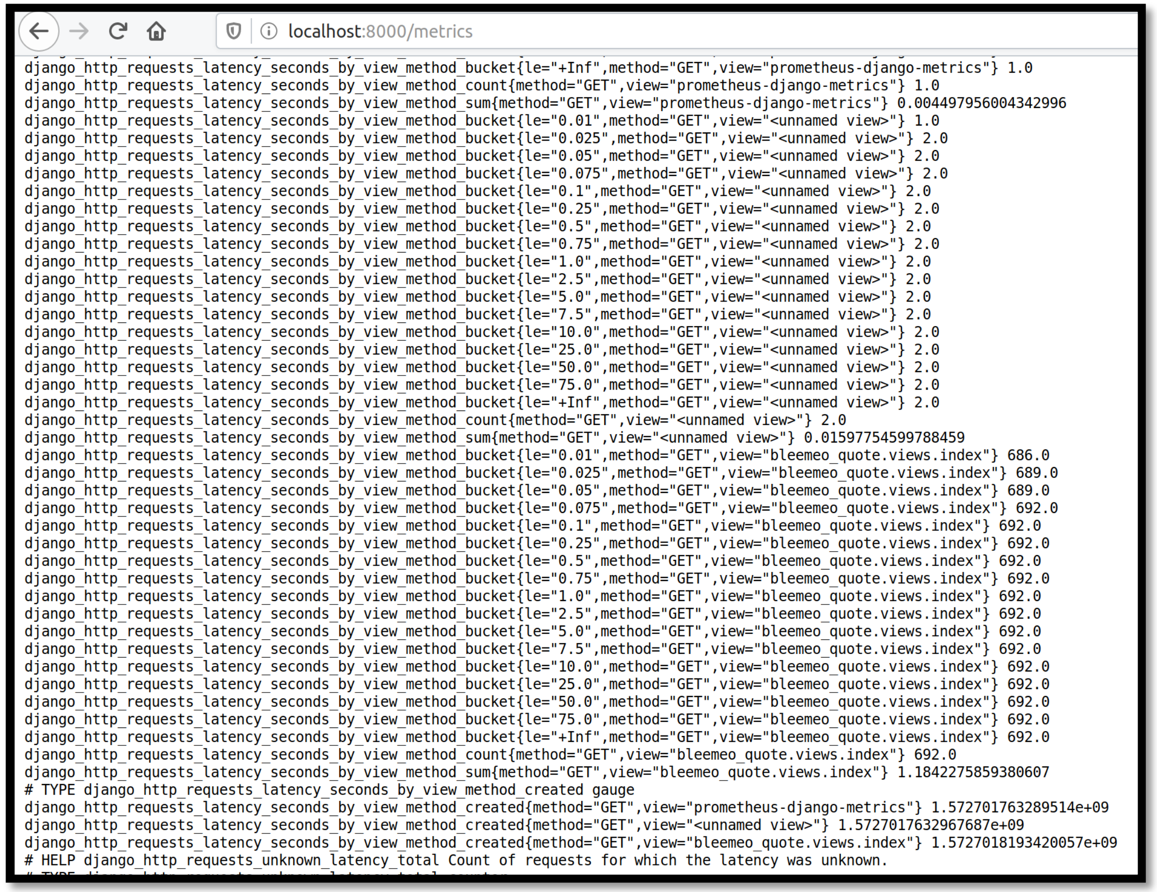
Prometheus

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...



```yaml
global:
  scrape_interval: 5s
scrape_configs:
  - job_name: "node-application-monitoring-app"
    static_configs:
      - targets: ["docker.host:8080"]
```



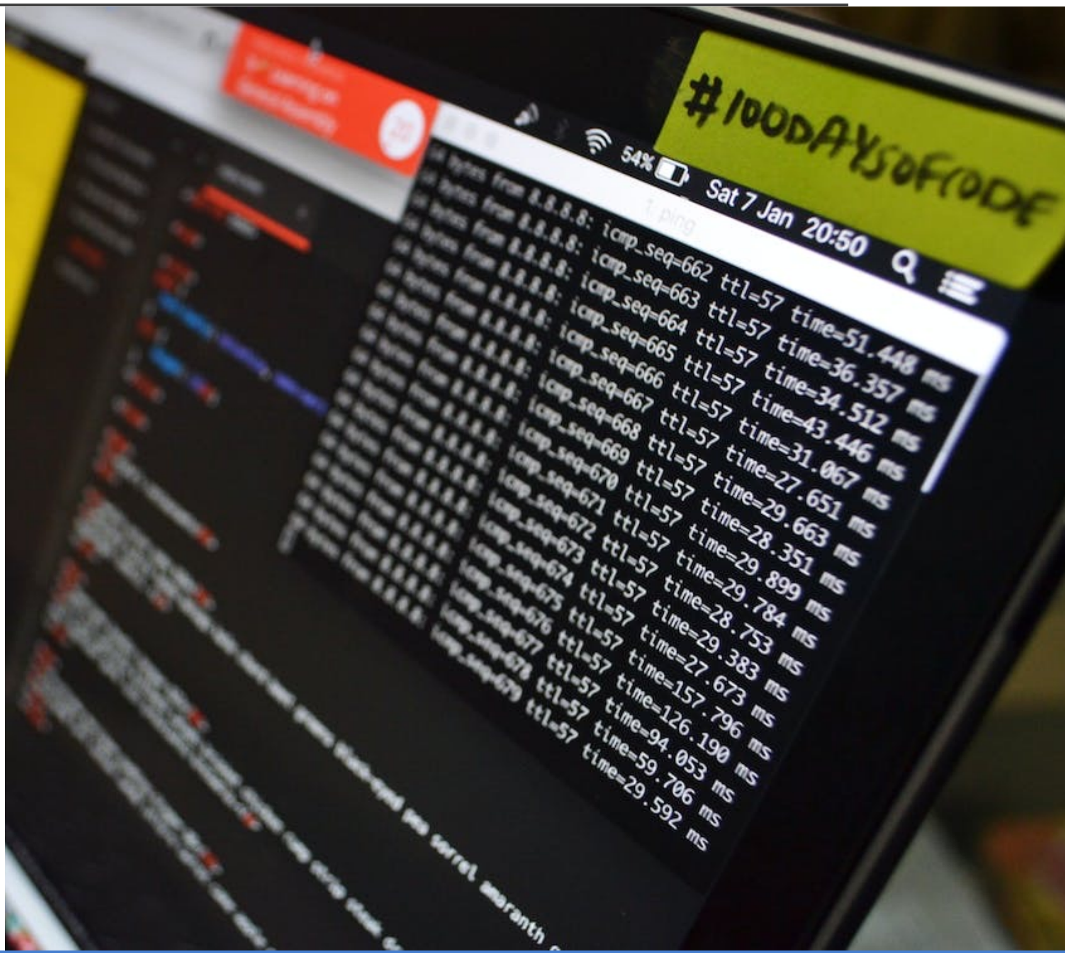
```
django_http_requests_latency_seconds_by_view_method_bucket{le="+Inf",method="GET",view="prometheus-django-metrics"} 1.0
django_http_requests_latency_seconds_by_view_method_count{method="GET",view="prometheus-django-metrics"} 1.0
django_http_requests_latency_seconds_by_view_method_sum{method="GET",view="prometheus-django-metrics"} 0.004497956004342996
django_http_requests_latency_seconds_by_view_method_bucket{le="0.01",method="GET",view="<unnamed view>"} 1.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.025",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.05",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.075",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.1",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.25",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.5",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.75",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="1.0",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="2.5",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="5.0",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="7.5",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="10.0",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="25.0",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="50.0",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="75.0",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_bucket{le="+Inf",method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_count{method="GET",view="<unnamed view>"} 2.0
django_http_requests_latency_seconds_by_view_method_sum{method="GET",view="<unnamed view>"} 0.01597754599788459
django_http_requests_latency_seconds_by_view_method_bucket{le="0.01",method="GET",view="bleemeo_quote.views.index"} 686.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.025",method="GET",view="bleemeo_quote.views.index"} 689.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.05",method="GET",view="bleemeo_quote.views.index"} 689.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.075",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.1",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.25",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.5",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="0.75",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="1.0",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="2.5",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="5.0",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="7.5",method="GET",view="bleemeo_quote.views.index"} 692.0
django_http_requests_latency_seconds_by_view_method_bucket{le="10.0",method="GET",view="bleemeo_quote.views.index"} 692.0
django_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
# TYPE django_http_requests_latency_seconds_by_view_method_created gauge
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 view>"} 1.5727017632967687e+09
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.
```

Centralize Security Health
In your existing Monitoring Solution

System monitoring

You should monitor on your systems:

- Resource utilization: CPU, memory, I/O, disk space
- Number of pending security updates
- SSL Certificates validity

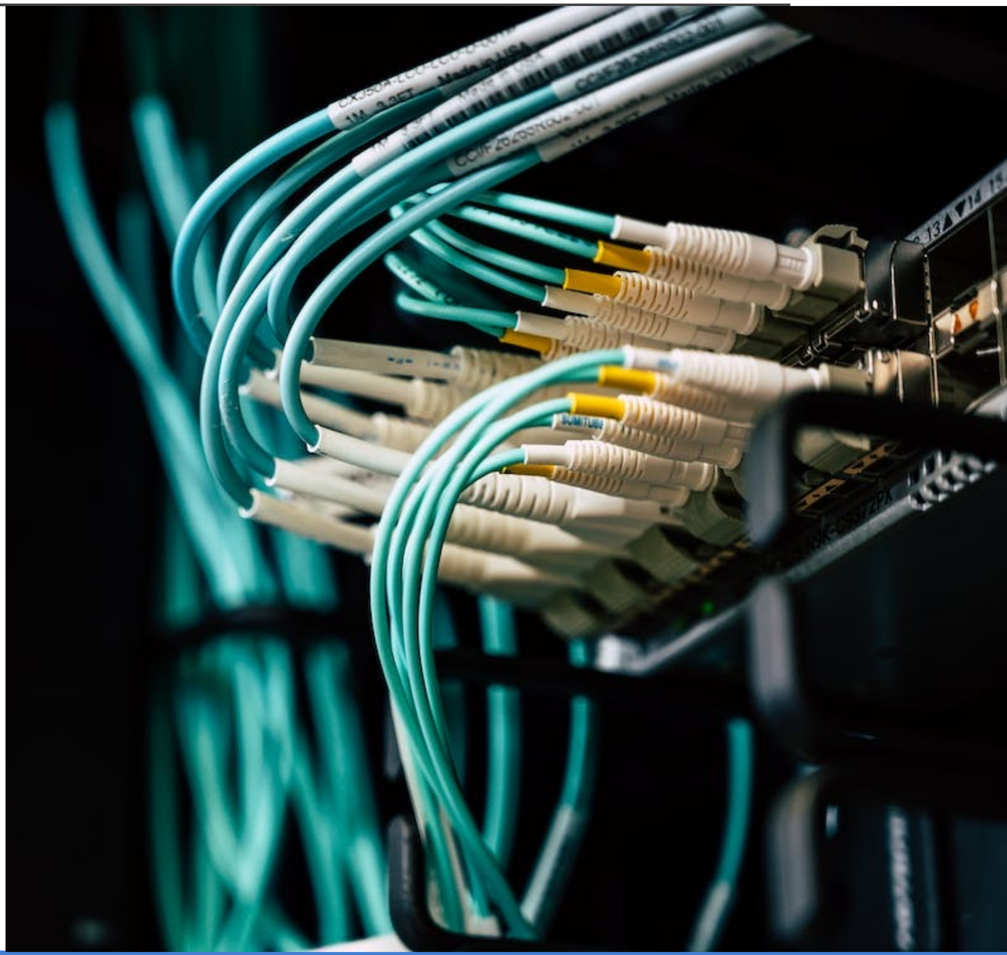


```
64 bytes from 8.8.8.8: icmp_seq=662 ttl=57 time=51.448 ms
64 bytes from 8.8.8.8: icmp_seq=663 ttl=57 time=36.357 ms
64 bytes from 8.8.8.8: icmp_seq=664 ttl=57 time=34.512 ms
64 bytes from 8.8.8.8: icmp_seq=665 ttl=57 time=43.446 ms
64 bytes from 8.8.8.8: icmp_seq=666 ttl=57 time=31.067 ms
64 bytes from 8.8.8.8: icmp_seq=667 ttl=57 time=27.651 ms
64 bytes from 8.8.8.8: icmp_seq=668 ttl=57 time=29.663 ms
64 bytes from 8.8.8.8: icmp_seq=669 ttl=57 time=28.351 ms
64 bytes from 8.8.8.8: icmp_seq=670 ttl=57 time=29.899 ms
64 bytes from 8.8.8.8: icmp_seq=671 ttl=57 time=29.784 ms
64 bytes from 8.8.8.8: icmp_seq=672 ttl=57 time=28.753 ms
64 bytes from 8.8.8.8: icmp_seq=673 ttl=57 time=29.383 ms
64 bytes from 8.8.8.8: icmp_seq=674 ttl=57 time=27.673 ms
64 bytes from 8.8.8.8: icmp_seq=675 ttl=57 time=157.796 ms
64 bytes from 8.8.8.8: icmp_seq=676 ttl=57 time=126.190 ms
64 bytes from 8.8.8.8: icmp_seq=677 ttl=57 time=94.053 ms
64 bytes from 8.8.8.8: icmp_seq=678 ttl=57 time=59.706 ms
64 bytes from 8.8.8.8: icmp_seq=679 ttl=57 time=29.592 ms
```

Network monitoring

You should monitor your network (even in Cloud):

- Bandwidth usage from equipment's
- Find unexpected network traffic
- CPU/memory usage of network gears if you have some
- Use SNMP to collect data



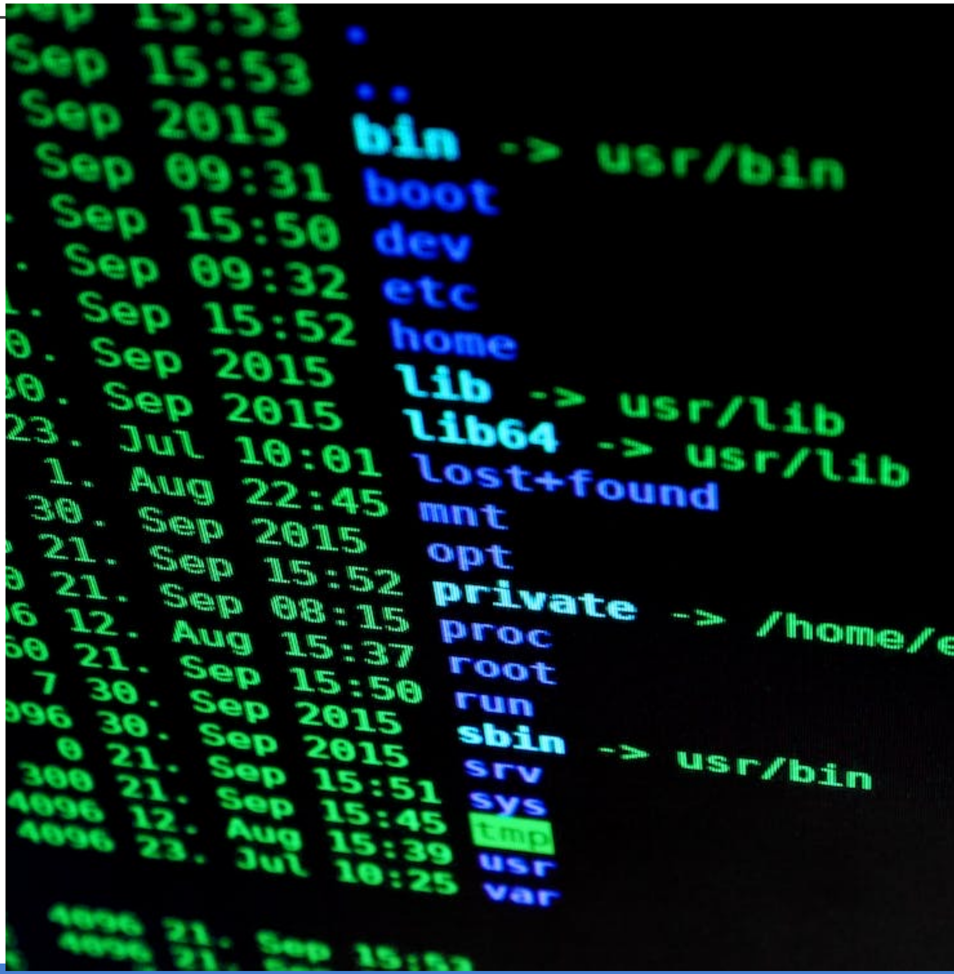
Logs monitoring

You should monitor your logs! Create metrics from logs:

- Errors rate
- Number of lines of logs
- Number of failed authentications
- Use regexp to identify interesting patterns in logs

👉 Check our blog post using mtail to create metrics:

blee.moe/metrics-logs



```
     Sep 15:53 .
     Sep 15:53 ..
     Sep 2015  bin -> usr/bin
     Sep 09:31 boot
   . Sep 15:50 dev
   . Sep 09:32 etc
  1. Sep 15:52 home
  0. Sep 2015  lib -> usr/lib
 30. Sep 2015  lib64 -> usr/lib
 23. Jul 10:01 lost+found
  1. Aug 22:45 mnt
 30. Sep 2015  opt
 21. Sep 15:52 private -> /home/e
 21. Sep 08:15 proc
6 12. Aug 15:37 root
60 21. Sep 15:50 run
7 30. Sep 2015 sbin -> usr/bin
96 30. Sep 2015 srv
0 21. Sep 15:51 sys
300 21. Sep 15:45 tmp
4096 12. Aug 15:39 usr
4096 23. Jul 10:25 var
```

Monitor external availability

- Monitor service availability from outside of your network
- Use external probes to monitor your service
- Monitor service from external point of view
- Use open source project: Blackbox exporter (for e.g.) or Cloud solutions: Uptime Robot, Bleemeo

Use Prometheus Exporters

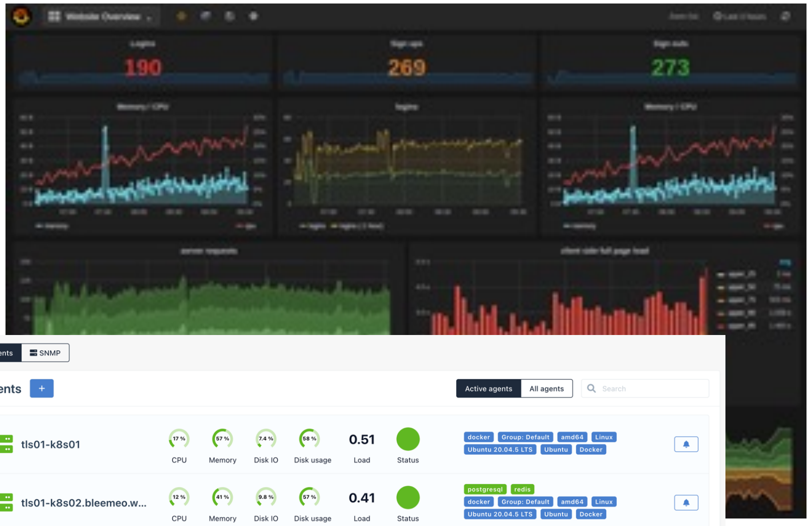
Prometheus Ecosystem has multiple exporters related to security:

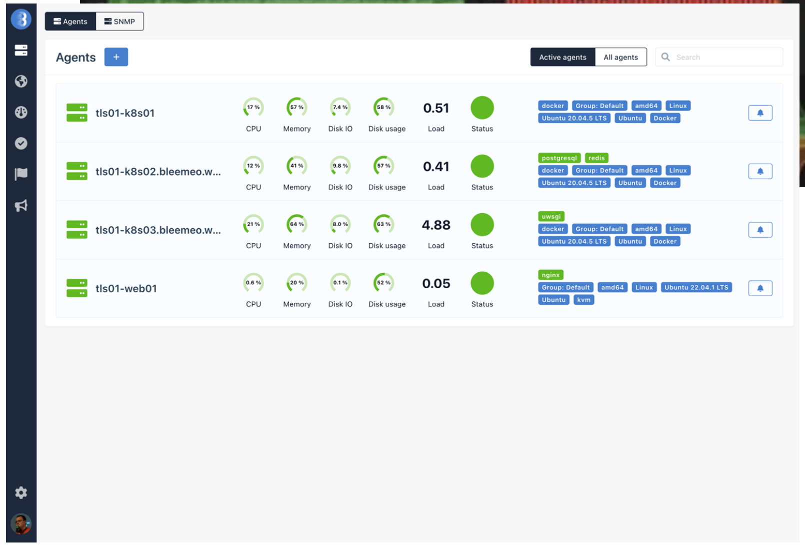
- Suricata/Snort exporter (NIDS)
- CrowdSec exporter (HIDS)
- pfSense/OpnSense exporter (firewall)
- SNMP exporter
- Create your own is very simple!

👉 exporters list: blee.moe/promexporters

Building custom dashboards

- If you use Prometheus, usual solution is Grafana
- You can find dashboards templates
- You can build your dashboards
- Prioritize golden signals of all cluster for your default dashboard
- Have detailed dashboards to go deeper for each node/Pod
- Cloud tools like Bleemeo offer automatic dashboarding





Agent	CPU	Memory	Disk IO	Disk usage	Load	Status	Tags
tls01-k8s01	17 %	57 %	7.4 %	58 %	0.51	●	docker, Group: Default, amd64, Linux, Ubuntu 20.04.5 LTS, Ubuntu, Docker
tls01-k8s02.bleemeo.w...	12 %	41 %	9.8 %	57 %	0.41	●	postgresql, redis, docker, Group: Default, amd64, Linux, Ubuntu 20.04.5 LTS, Ubuntu, Docker
tls01-k8s03.bleemeo.w...	21 %	64 %	8.0 %	63 %	4.88	●	uwsgi, docker, Group: Default, amd64, Linux, Ubuntu 20.04.5 LTS, Ubuntu, Docker
tls01-web01	0.6 %	20 %	0.1 %	52 %	0.05	●	nginx, Group: Default, amd64, Linux, Ubuntu 22.04.1 LTS, Ubuntu, kvm

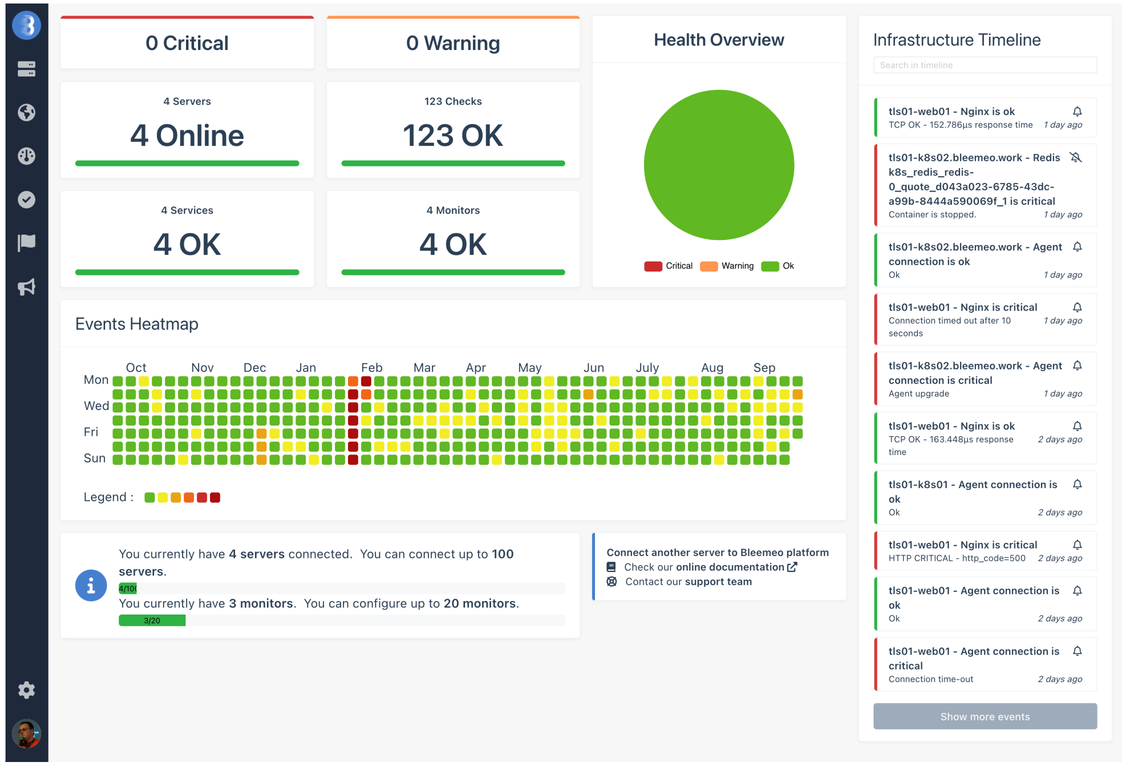
Reuse existing alerting!

- Monitoring comes with alerting
- Golden signals are a good source of alerts
- Alerting means immediate attention is required
- Notify only when human action is required
- Check your dashboards

Bleemeo simplifies and automates monitoring infrastructure security

Bleemeo Monitoring Solution

- Cloud based solution
- Compatible with Prometheus and market standards (StatsD, Nagios, …)
- Agent run on each server and discover services, containers and create dashboards for you
- Alerting with Slack, Teams, mail, SMS, …
- Query data (on dashboards and alerting) with standard PromQL
- Mobile application for iOS and Android



You currently have **4 servers** connected. You can connect up to **100 servers**.

You currently have **3 monitors**. You can configure up to **20 monitors**.

Connect another server to Bleemeo platform

Check our **online documentation**

Contact our **support team**

Infrastructure Timeline

tls01-web01 - Nginx is ok
TCP OK - 152.786µs response time *1 day ago*

tls01-k8s02.bleemeo.work - Redis k8s_redis_redis-0_quote_d043a023-6785-43dc-a99b-8444a590069f_1 is critical
Container is stopped. *1 day ago*

tls01-k8s02.bleemeo.work - Agent connection is ok
Ok *1 day ago*

tls01-web01 - Nginx is critical
Connection timed out after 10 seconds *1 day ago*

tls01-k8s02.bleemeo.work - Agent connection is critical
Agent upgrade *1 day ago*

tls01-web01 - Nginx is ok
TCP OK - 163.448µs response time *2 days ago*

tls01-k8s01 - Agent connection is ok
Ok *2 days ago*

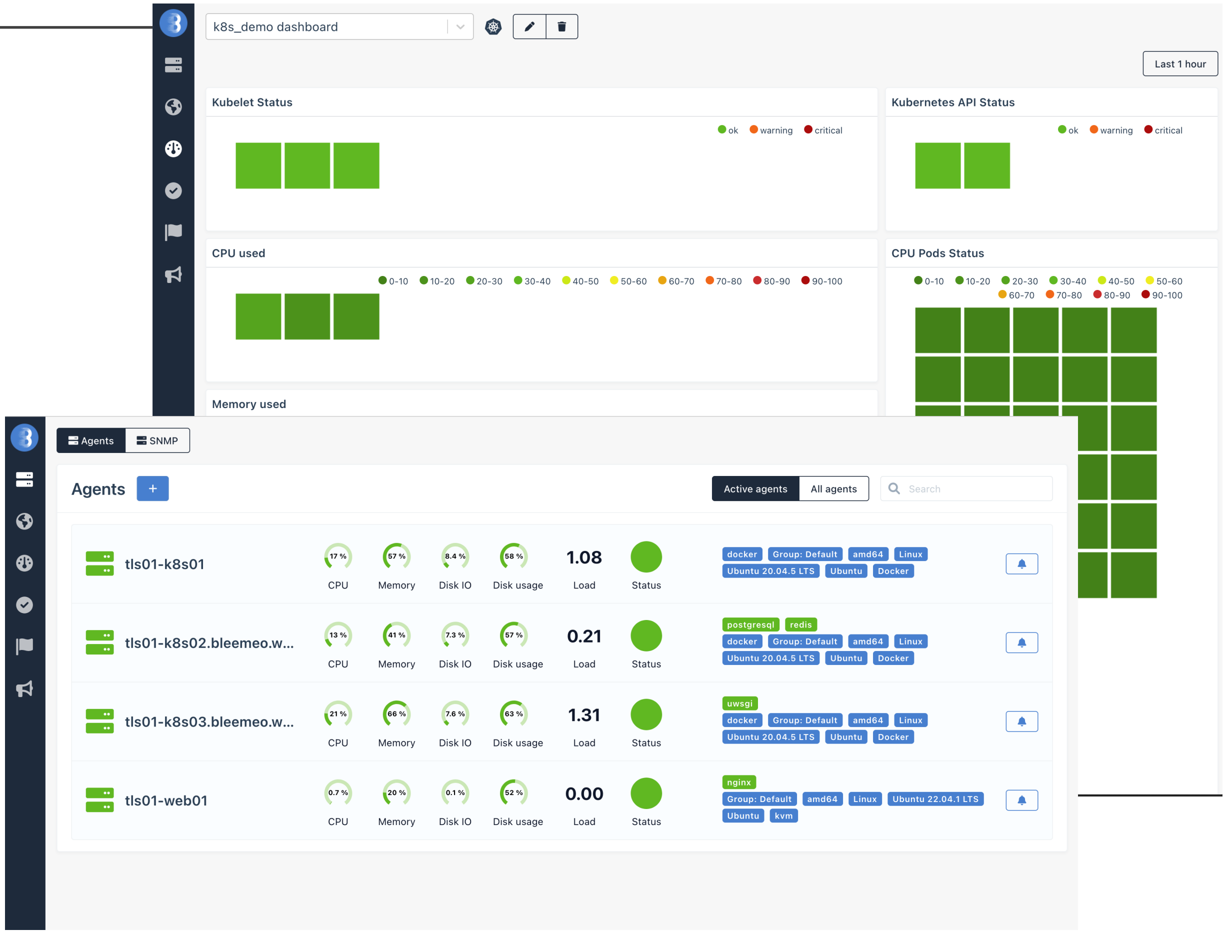
tls01-web01 - Nginx is critical
HTTP CRITICAL - http_code=500 *2 days ago*

tls01-web01 - Agent connection is ok
Ok *2 days ago*

tls01-web01 - Agent connection is critical
Connection time-out *2 days ago*

Security Monitoring with Bleemeo

- Pending security patches out of the box
- Connect to any Prometheus capable probe
- Monitor all network equipment's with SNMP
- Get a global health view of your infrastructure (security and all key indicators)



Agents

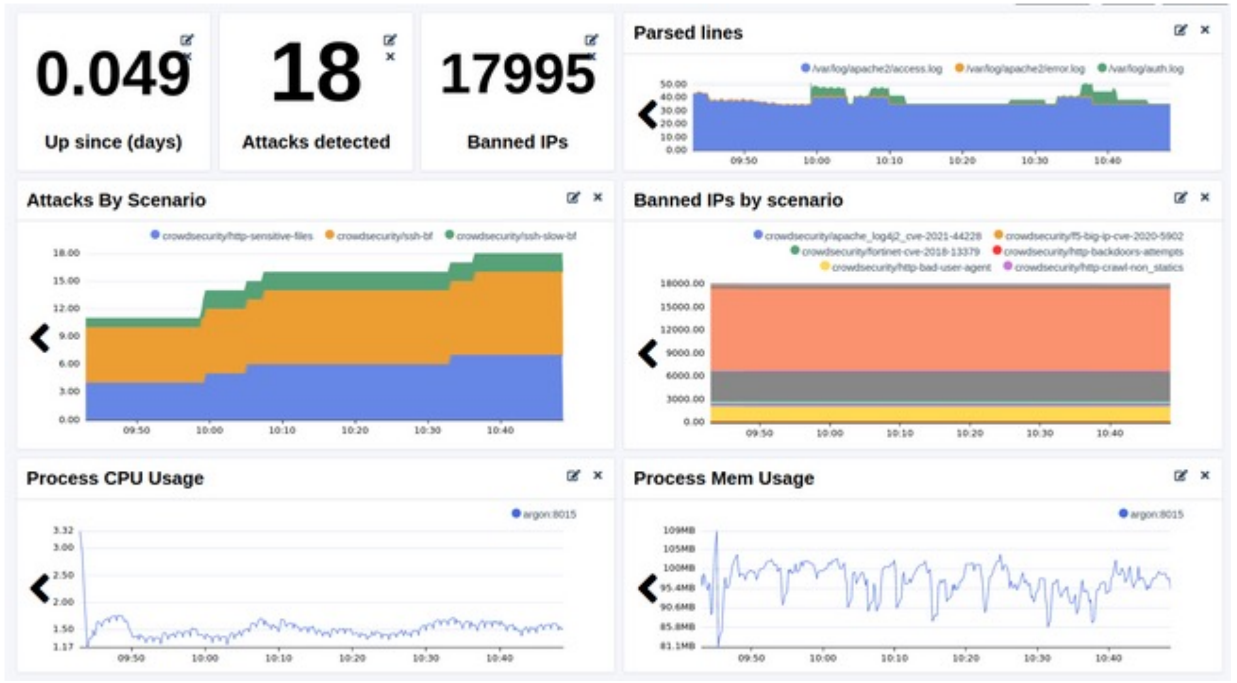
	CPU	Memory	Disk IO	Disk usage	Load	Status	
tls01-k8s01	17 %	57 %	8.4 %	58 %	1.08	●	docker, Group: Default, amd64, Linux, Ubuntu 20.04.5 LTS, Ubuntu, Docker
tls01-k8s02.bleemeo.w...	13 %	41 %	7.3 %	57 %	0.21	●	postgresql, redis, docker, Group: Default, amd64, Linux, Ubuntu 20.04.5 LTS, Ubuntu, Docker
tls01-k8s03.bleemeo.w...	21 %	66 %	7.6 %	63 %	1.31	●	uwsgi, docker, Group: Default, amd64, Linux, Ubuntu 20.04.5 LTS, Ubuntu, Docker
tls01-web01	0.7 %	20 %	0.1 %	52 %	0.00	●	nginx, Group: Default, amd64, Linux, Ubuntu 22.04.1 LTS, Ubuntu, kvm

Example of CrowdSec Integration

CrowdSec is an open-source software that allows you to detect peers with malicious behaviors and block them from accessing your systems. It benefits from a global community-wide IP reputation database.

- Expose Prometheus endpoint
- Metrics for number of scenario matched
- Metrics for number of IP banned

👉 our blog post on blee.moe/crowdsec


0.049
Up since (days)
18
Attacks detected
17995
Banned IPs
Parsed lines
/var/log/apache2/access.log
/var/log/apache2/error.log
/var/log/auth.log
Attacks By Scenario
crowdsecurity/http-sensitive-files
crowdsecurity/ssh-bf
crowdsecurity/ssh-slow-bf
Banned IPs by scenario
crowdsecurity/apache_log4j2_cve-2021-44228
crowdsecurity/f5-big-ip-cve-2020-5902
crowdsecurity/fortinet-cve-2018-13379
crowdsecurity/http-backdoors-attempts
crowdsecurity/http-bad-user-agent
crowdsecurity/http-crawl-non_statics
Process CPU Usage
argon:8015
Process Mem Usage
argon:8015

Conclusion

- Monitoring your Security is a must have nowadays
- Concentrating your security information in your Monitoring tool can be quick and efficient
- Use standards for getting metrics of your security software and equipment's
- Bleemeo automates all metrics collection and allows you to create custom dashboards

Questions?

👉 Try for Free Bleemeo

https://bleemeo.com/trial