All the things we don’t know

$ whoami

- Former network & infra eng. At Crédit Mutuel Arkéa
- Anti-DDoS Detection system at OVH
- Big Data Architect at OVH
- @OvhMetrics

What is Metrics?

Summary

- Make better decisions
- Code instrumentation is free
- A new take on Infrastructure monitoring
- Visualize with Grafana
- Prediction & Anomaly Detection
- Alerting

Make Better Decisions
By using Numbers

Code

Business value

Business value

- Examples
  - A new feature
  - Reducing bugs
  - Remove website slowness for users
  - Remove website ugliness
  - Improving experience
  - Making future changes easier
  - ...

Business value
is anything which
makes people more
likely to give us money

We want to generate more Business value

We need to make better decisions about our code

Our code

Generates **Business Value** when it ***runs***

Our code

Generates **Business Value** when it ***runs***

not when we ***write*** it

We need to know what our **code** does when it **runs**

We can’t do this unless we **measure** it

Why measure it?

Mental model

map ≠ territory

Mental model

perception ≠ reality

We have a **mental model**
of what our **code** does

This **representation**
can be **wrong**

```
items.sort_by { |i| i.name }

items.sort { |a, b| a.name <=> b.name }
```

Which implementation Is faster?

We don’t know

We **can’t** know until
we **measure** it

“This application is slow.
The page takes 500ms!
Fix it”

Find the bottleneck

SQL Query

Template Rendering

Session Storage

We don’t know

Find the bottleneck

with Observability

SQL Query.....53ms

Template Rendering.....1ms

Session Storage.....315ms

Find the bottleneck

with Observability

SQL Query.....53ms

Template Rendering.....1ms

Session Storage.....315ms

We made a better decision

We improve our **mental model** by **measuring** what our **code** does

We use our **mental model** to **decide** what to **do**

A better **mental model** makes us better at **deciding** what to **do**

Better **decisions** makes us
better at generating **Business Value**

Measuring make your **Business** better

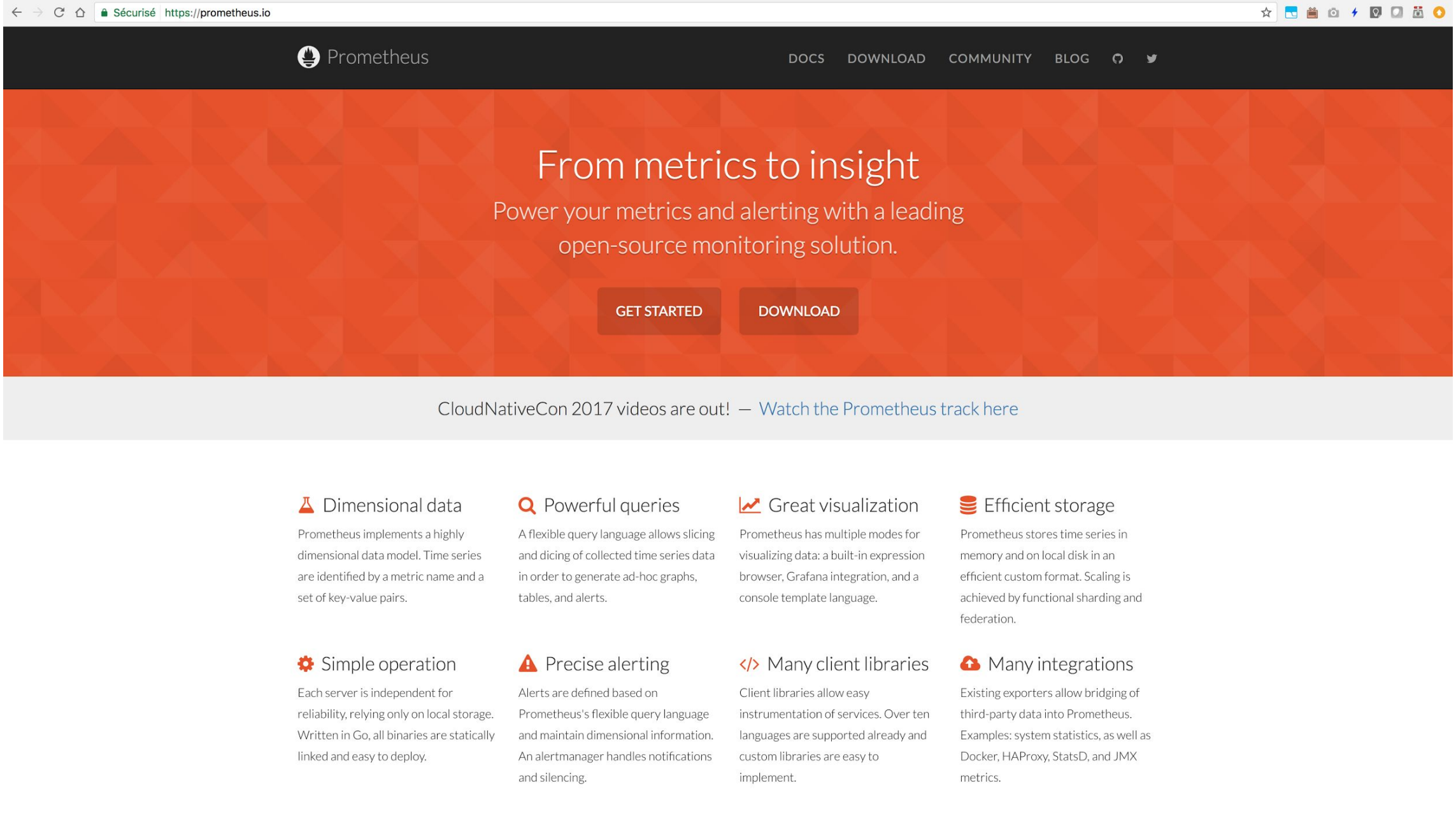
Instrumenting your app is free

Production

How to measure?

“Patron of mankind”

Prometheus


Sécurisé https://prometheus.io
Prometheus
DOCS DOWNLOAD COMMUNITY BLOG
From metrics to insight
Power your metrics and alerting with a leading open-source monitoring solution.
GET STARTED
DOWNLOAD
CloudNativeCon 2017 videos are out! — Watch the Prometheus track here
Dimensional data
Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.
Powerful queries
A flexible query language allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.
Great visualization
Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.
Efficient storage
Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.
Simple operation
Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically linked and easy to deploy.
Precise alerting
Alerts are defined based on Prometheus's flexible query language and maintain dimensional information. An alertmanager handles notifications and silencing.
Many client libraries
Client libraries allow easy instrumentation of services. Over ten languages are supported already and custom libraries are easy to implement.
Many integrations
Existing exporters allow bridging of third-party data into Prometheus. Examples: system statistics, as well as Docker, HAProxy, StatsD, and JMX metrics.

Client Libraries

and many more...

How to **measure** different kind of things

Measurements types

Counters
Gauge
Histograms
Summary

Gauges

an instantaneous value

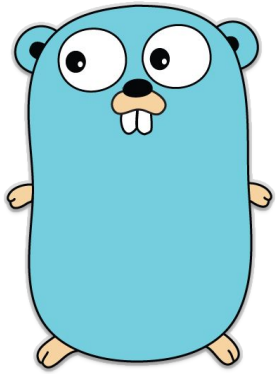
Gauges

an instantaneous value

Ex:

- # of calls in a queue
- # of registred users

Gauges



```
import "github.com/prometheus/client_golang/prometheus"

mailQueue := prometheus.NewGauge(prometheus.GaugeOpts{
    Name:      "mail_queued",
    Help:      "Number of mail waiting to be processed.",
})
prometheus.MustRegister(mailQueue)


// Queued 10 new mail.
mailQueue.Add(10)

// A worker has picked up a waiting mail.
mailQueue.Dec()

// And once more...
mailQueue.Dec()
```

Counters

an incrementing value

Counters

an incrementing value

Ex:

- # of connections
- # of orders

Counters

```python
from prometheus_client import Counter

shop_revenue = Counter('shop_revenue', 'Current shop revenue')

# Increment revenue by 1
shop_revenue.inc()

# Increment revenue by given value
shop_revenue.inc(2.0)
```

Histograms

a bucketized distribution of values

Histograms

a bucketized distribution of values

Ex:

- elapsed time per request
- # of products by cart

Histograms

```
const client = require('prom-client');

const cartProducts = new client.Histogram({
  name: 'cart_products',
  help: 'Number of product by cart',
  buckets: [ 1, 5, 10, 25, 50, 200, 500 ]
});


// Observe number of products by cart
cartProducts.observe(10);
```

Summary

a statistical distribution of values

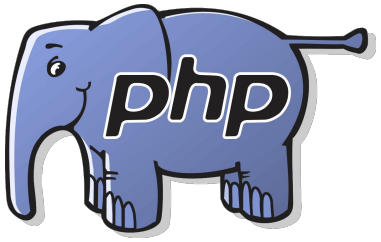
Summary

a statistical distribution of values

Ex:

- request latencies
- session time by user

Summary




```php
require_once dirname(__FILE__) . '/../src/Client.php';

$client = new Prometheus\Client([
  'base_uri' => 'http://localhost:9091/metrics/job/',
]);

$requestLatency = $client->newSummary([
  'name' => 'request_latency',
  'help' => 'Request latencies percentiles,
]);

// Observe request latency
$requestLatency>observe(['url' => 'home.php'], 253);

// Send metrics
$client->pushMetrics();
```

A new take on Infrastructure Monitoring

Classics are not enough anymore

#Nagios #Shinken #Centron #Zabbix

Old tools were fine while there were **few** changes

Today Business is

super charged accelerated *fast*

Track your **Code** in production

How do you **measure**?
How do you **analyse**?
How can you **debug**?

How do you **plan**?

Noderig

Beamium

Noderig

Collect and **Expose** host metrics

/ovh/noderig

Noderig

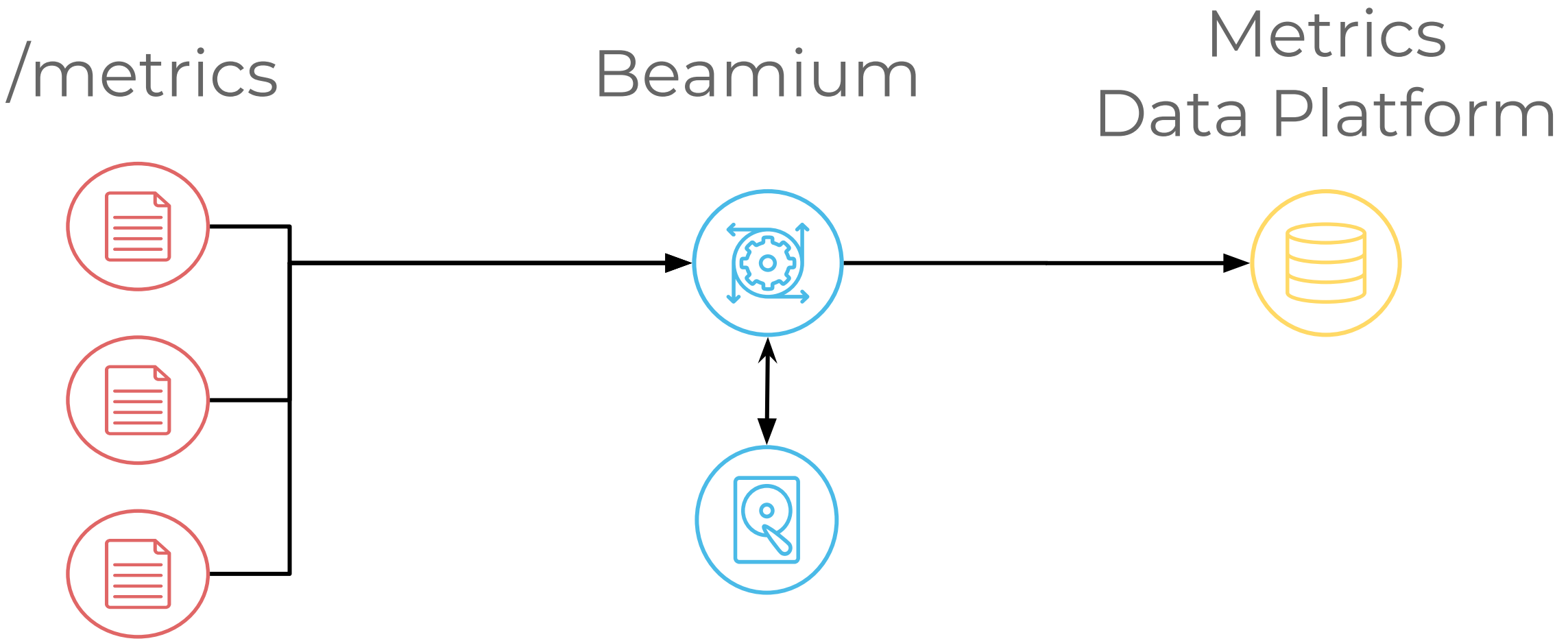
```
metrics@ovh:~# curl http://127.0.0.1:9100/metrics
1508147890273605// os.cpu{} 11.776711585191812
1508147890245599// os.mem{} 96.09097200951443
1508147890245599// os.swap{} 0
1508147890245685// os.load1{} 3.43
1508147890245685// os.load5{} 4.16
1508147890245685// os.load15{} 4.19
1508147890245912// os.net.bytes{iface=eth0,direction=in} 1220878638
1508147890245912// os.net.bytes{iface=eth0,direction=out}
230189460
1508147890245912// os.net.packets{iface=eth0,direction=in} 84172790
1508147890245912// os.net.packets{iface=eth0,direction=out} 807946
```

Beamium

Gather and **Send** metrics

/ovh/beamium

Beamium



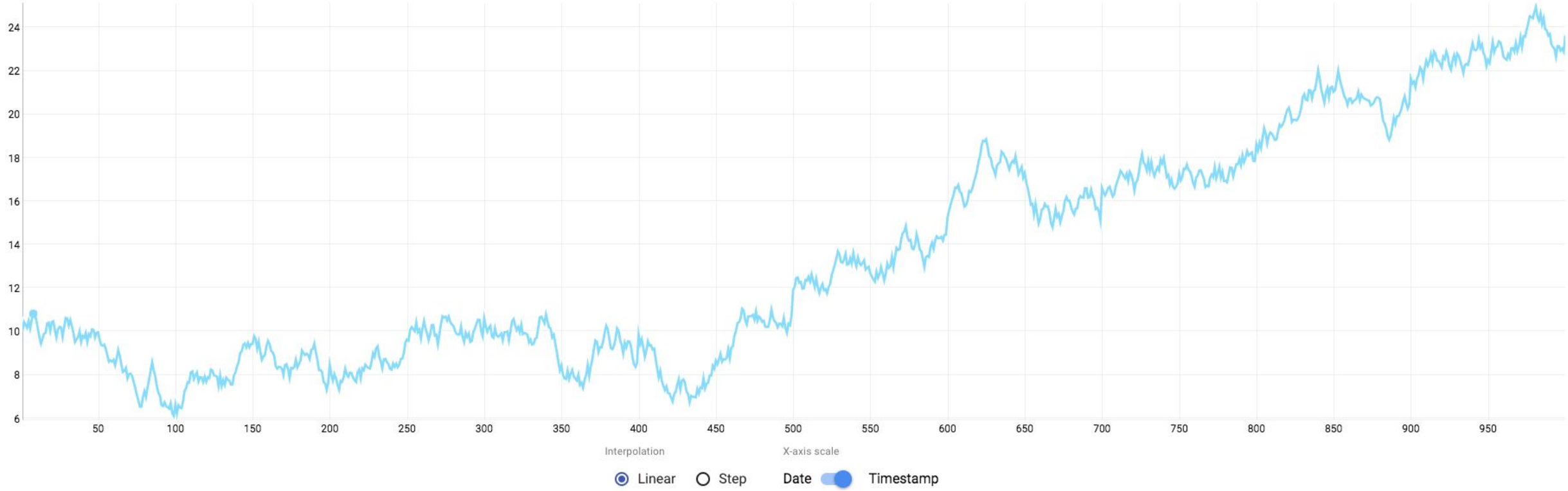
Visualize with Grafana


CPU usage
Load average
Memory & swap
Disks usage

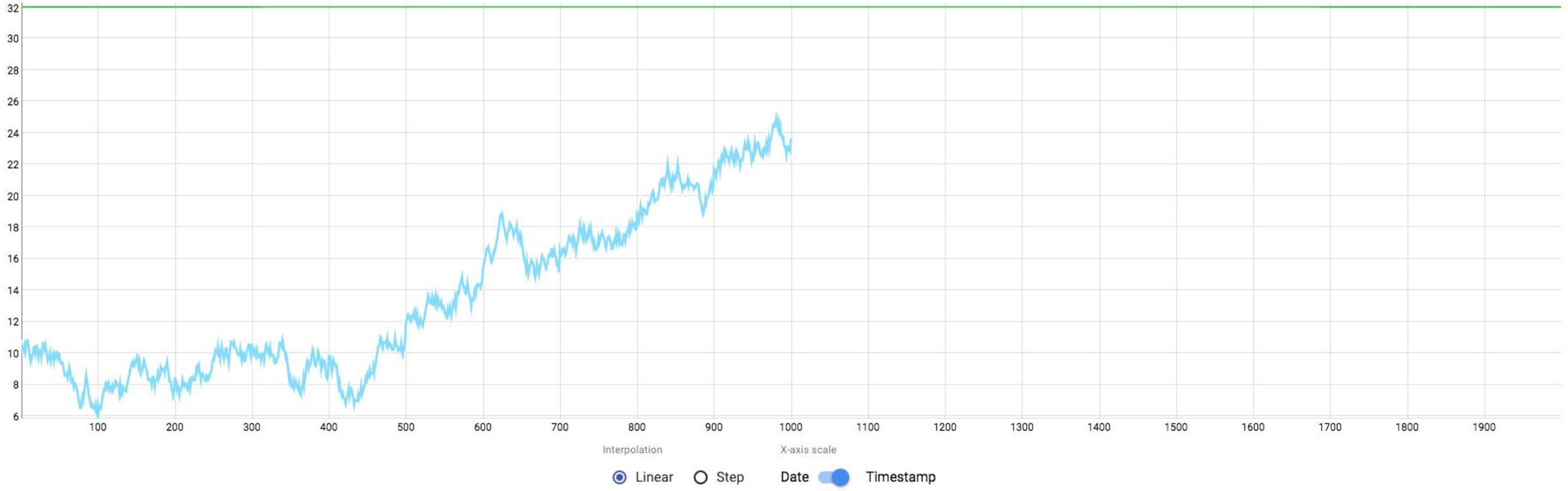
Prediction
&
Anomaly Detection

Predict your **Future**

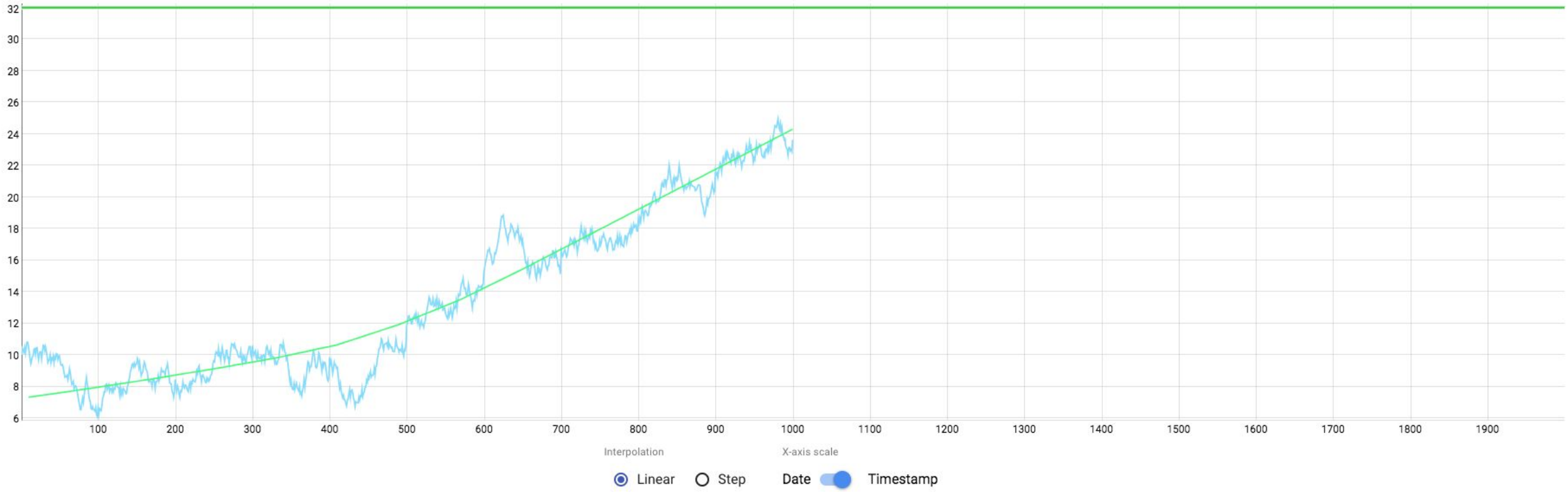
RAM consumption



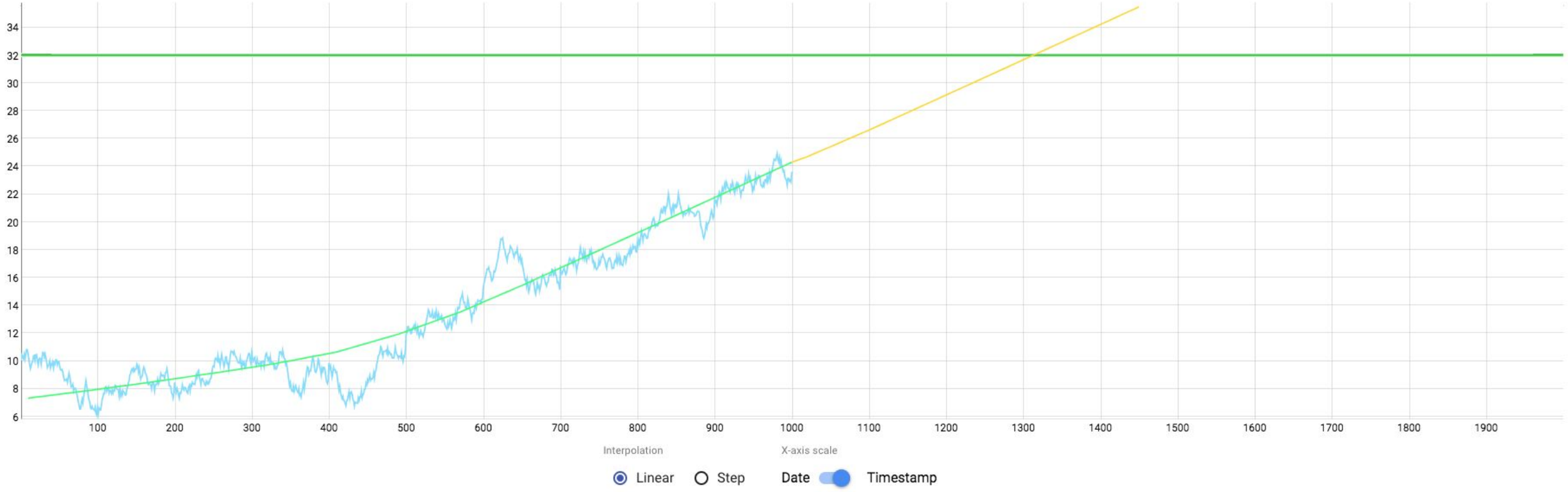
RAM consumption + limit



RAM consumption + limit + trend



RAM consumption + limit + trend + forecast

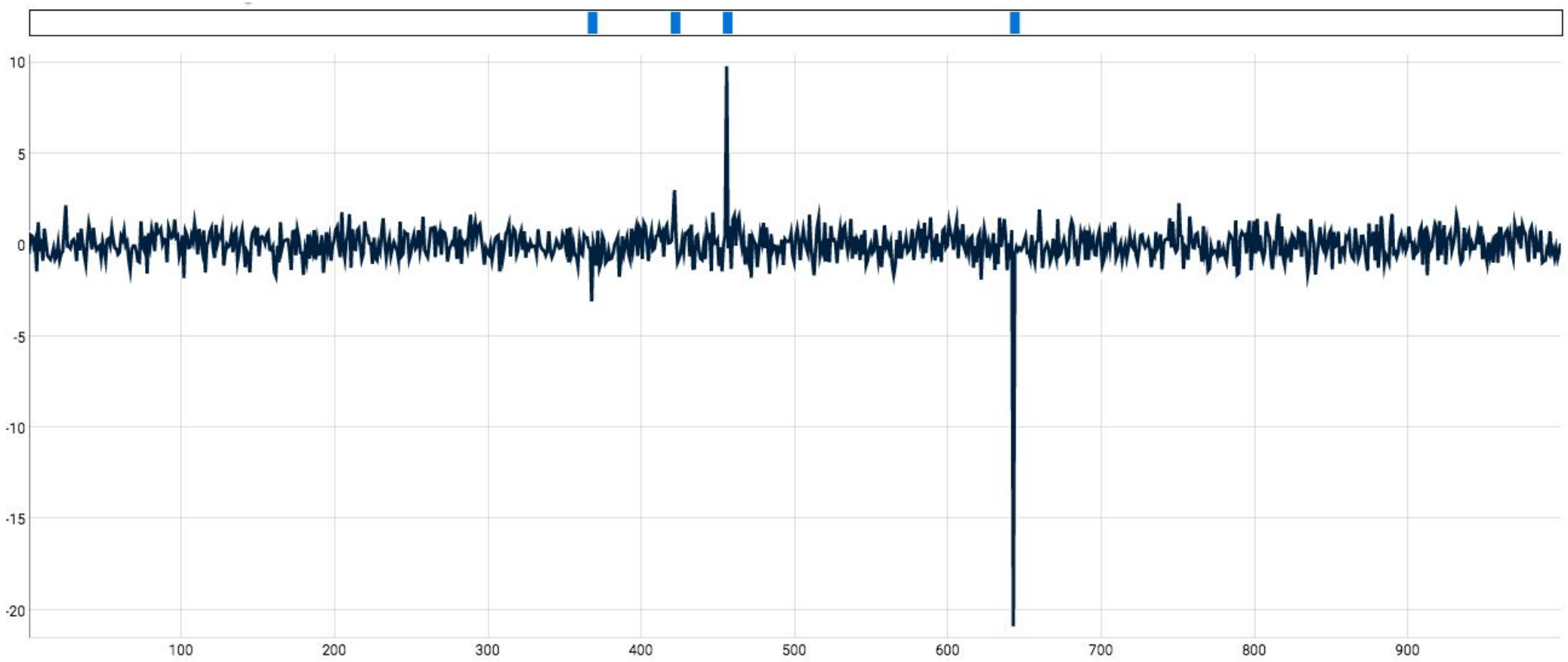


RAM consumption + limit + trend + forecast

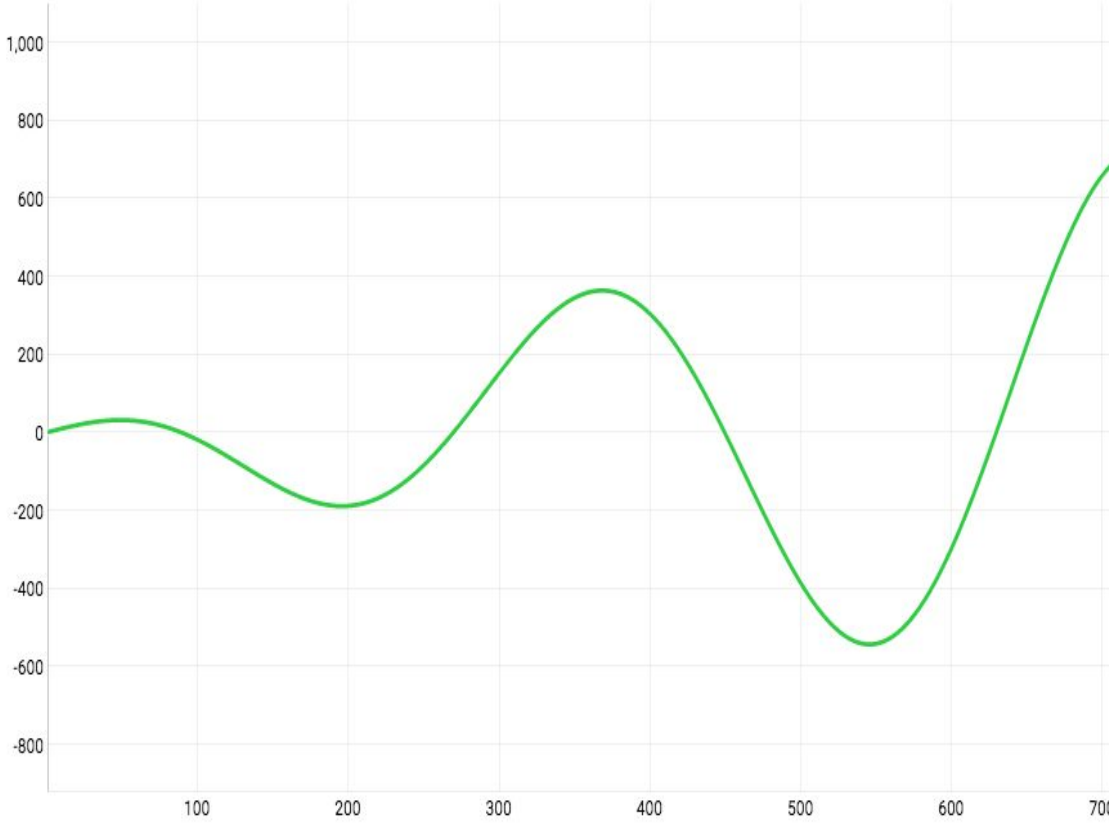
RAM consumption + limit + trend + forecast + alert or annotation

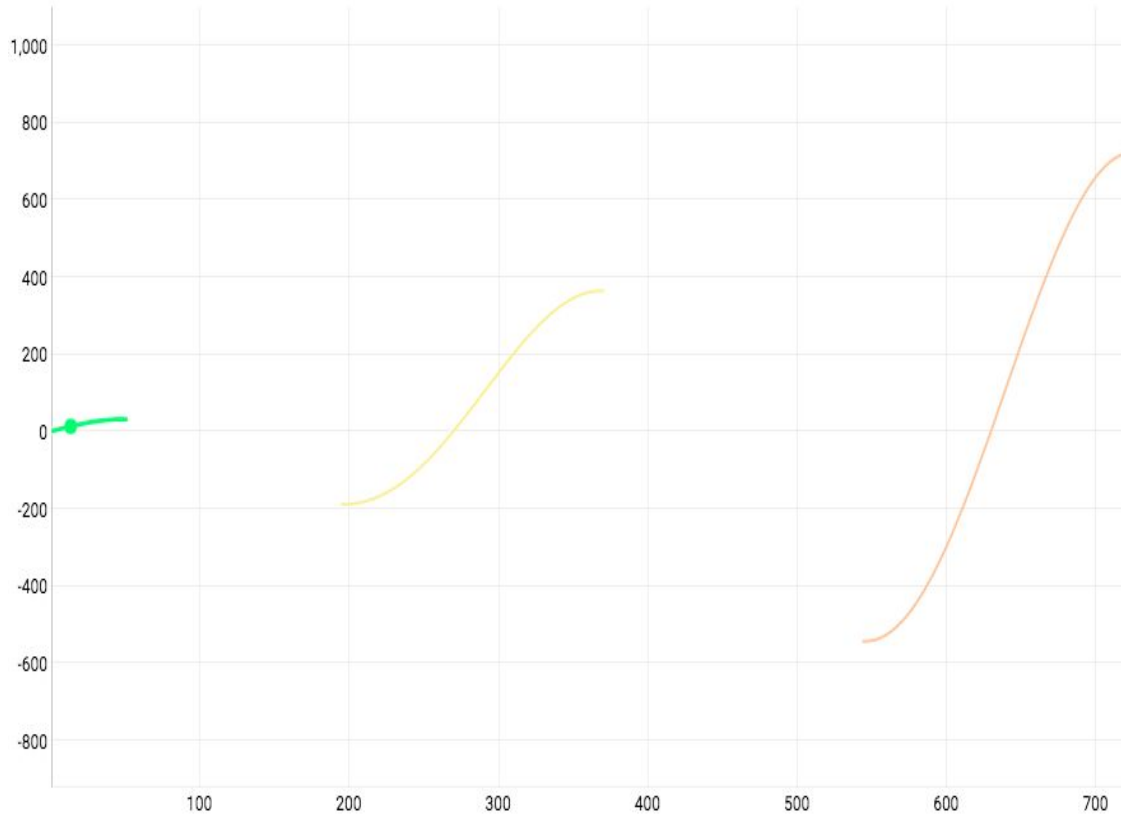
Detect Anomalies

Outliers



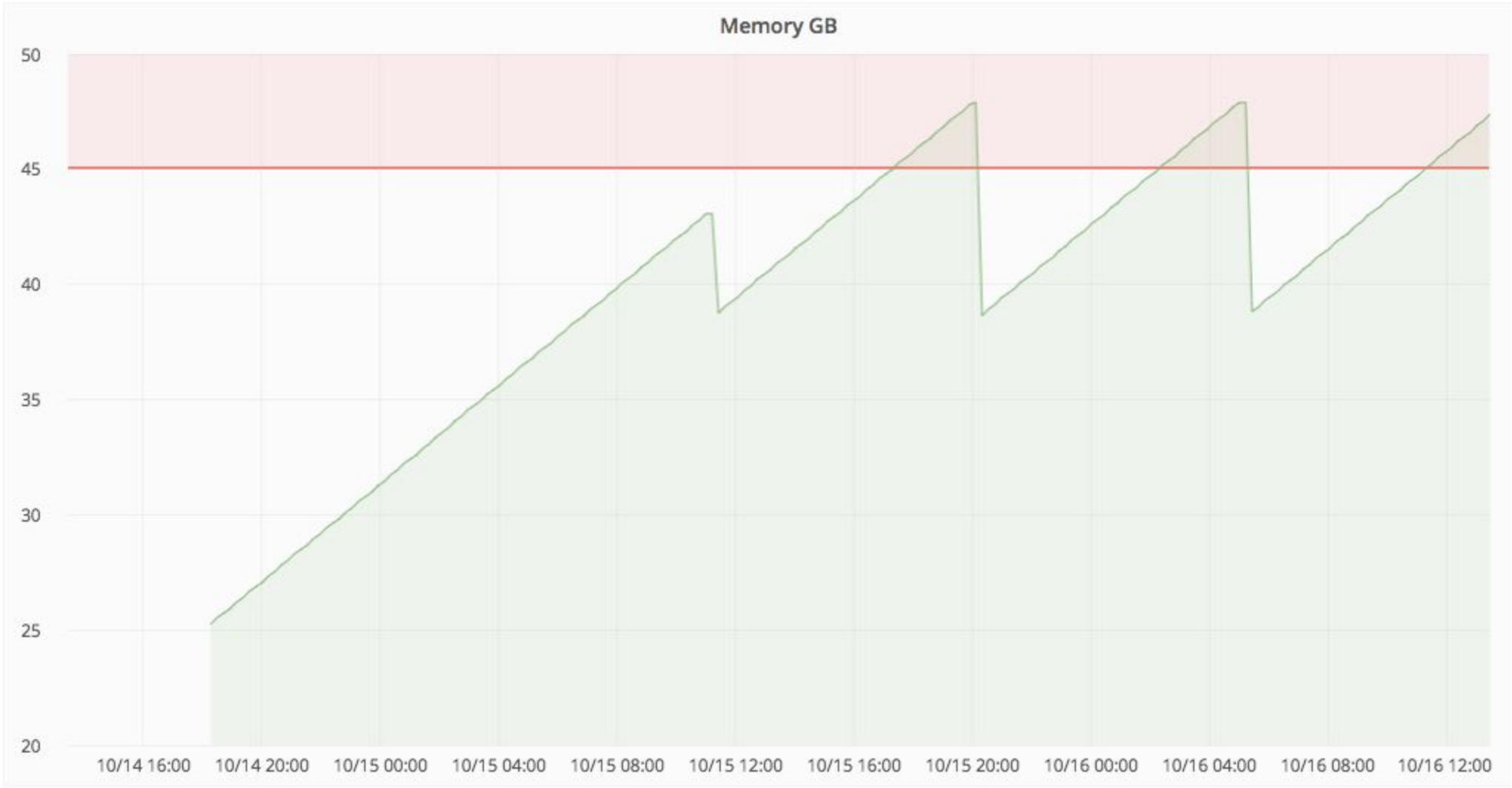
Pattern Matching





Series Decomposition

Metrics based Alerting


Memory GB
50
45
40
35
30
25
20
10/14 16:00
10/14 20:00
10/15 00:00
10/15 04:00
10/15 08:00
10/15 12:00
10/15 16:00
10/15 20:00
10/16 00:00
10/16 04:00
10/16 08:00
10/16 12:00

Thank you!

Alerting

Versatil

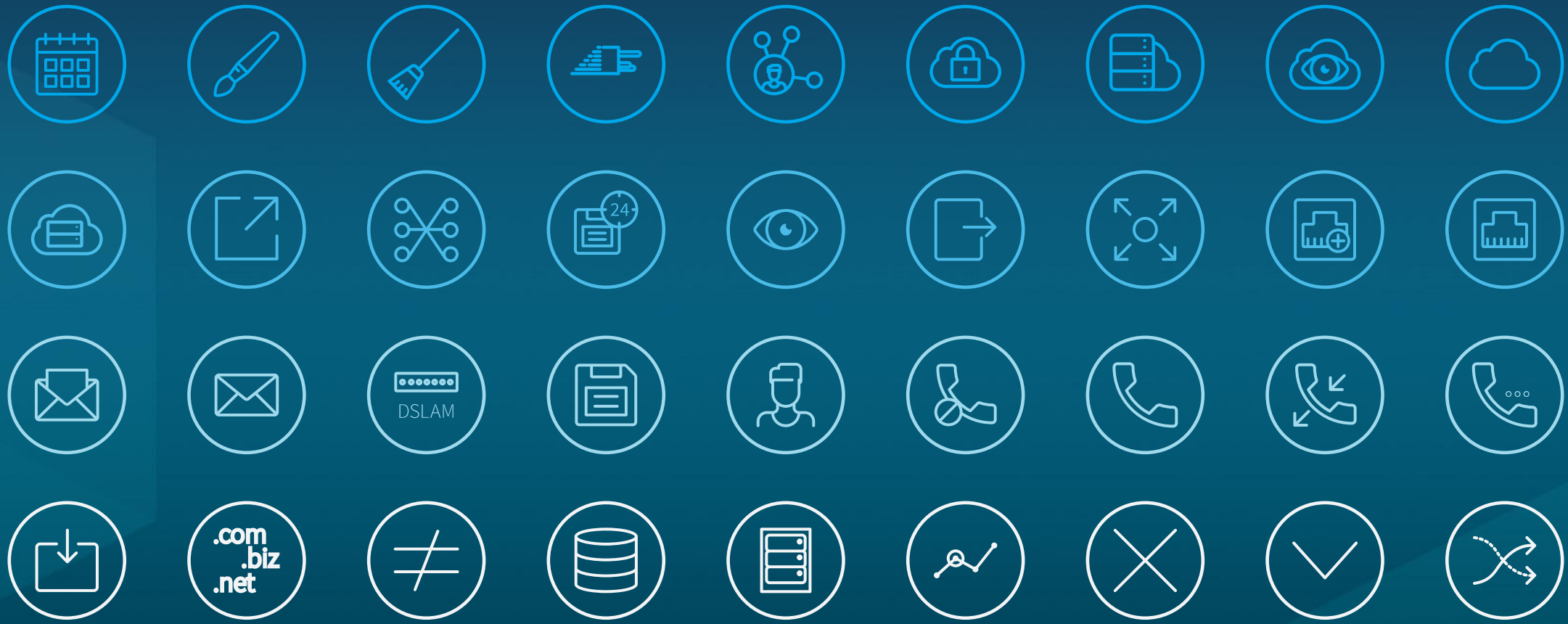
IoT

Analytics

WebUI

ICONS

ICONS



ICONS

ICONS

ICONS