

Storing metrics at
scale with
Gnocchi



Hello!

I am Julien Danjou

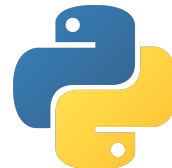
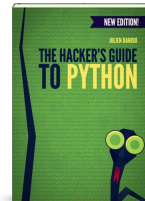
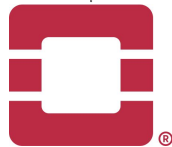
Principal Software Engineer at Red Hat



You can find me at julien@danjou.info and [@juldanjou](https://twitter.com/juldanjou)



I hack, create and contribute to FOSS projects:





What we'll talk about today

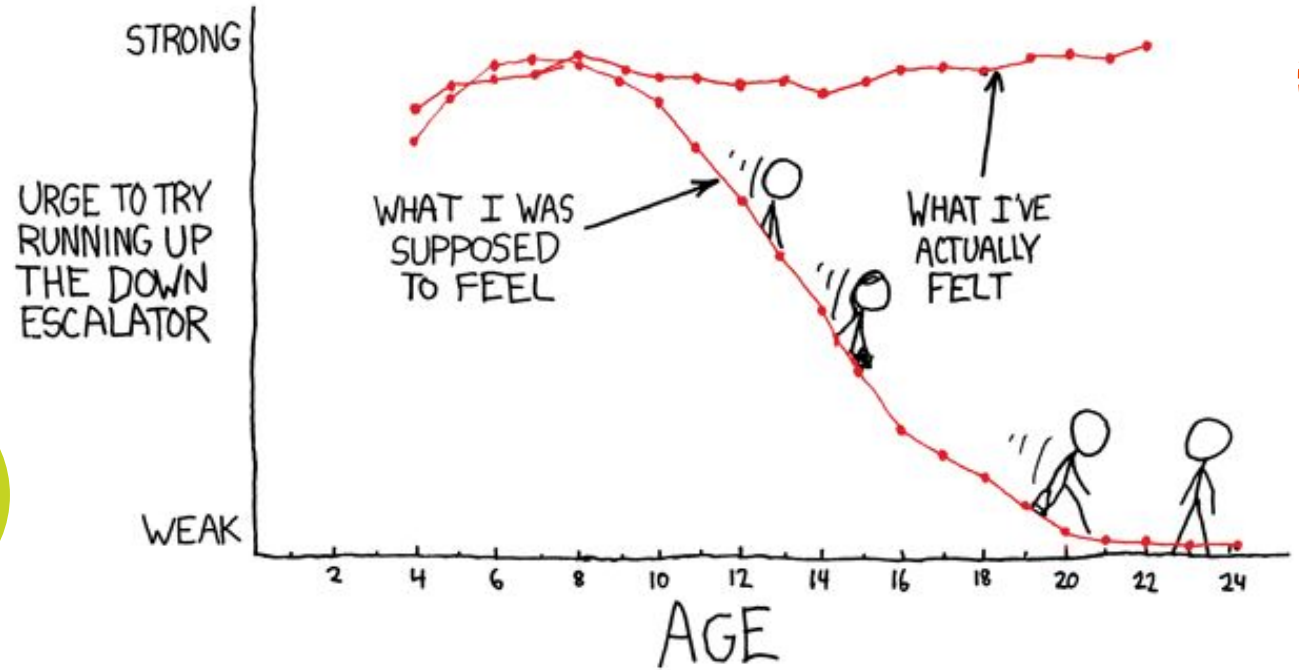
- ◎ **What is Gnocchi?**
(history, reasons, features)
- ◎ **How does Gnocchi work?**
(technology, principles)
- ◎ **How to use Gnocchi?**
(install, config, SDK)

The background is white with several colorful circles and dashed lines. In the top left, there is a large orange circle with a dashed white outline, overlapping a yellow circle. Below it is a small pink circle. In the top right, there is a green circle with a white dot inside, overlapping a yellow circle. Below it is a green circle with a dashed white outline. In the bottom left, there is a green circle with a dashed white outline, overlapping a large yellow circle. Below it is a small cyan circle. In the bottom right, there is a large cyan circle with a white dot inside, overlapping a cyan circle with a dashed white outline.

1

What's Gocchi?
Why is it there?

Time series





Perfect solution

Scalable

Targeting cloud platforms where thousands of instances and resources pop up every day.

Storing and retrieving data should be fast.

Easy to use

Provide an API that makes it easy to program against the solution. Build any kind of solution easily (billing, capacity planning, statistical analysis...)

Easy to operate

Installation and operation should be easy for administrators used to standard UNIX tools.



Existing solutions

- ◎ Graphite
 - ◎ Not scalable
 - ◎ Poor code base
 - ◎ Not modifiable
- ◎ InfluxDB
 - ◎ Does not work
 - ◎ Does not scale
- ◎ OpenTSDB
 - ◎ Need to set up Hadoop

...



Gnocchi – started in May 2014

Part of OpenStack Telemetry

Designed to solve Ceilometer storage issue back then.

But work stand-alone!

Free Software

Apache Licensed.

Easy to install

```
pip install gnocchi
```

Documented

Everything is documented.


“No doc: no merge” policy.

Written in Python

With some standard used libraries (SQLAlchemy, Pandas...)

Distributed & resilient

Design to run on cloud platforms. Native high-availability and workload distribution support.





Awesome features!

Search by metric value, compute aggregations

Look into metrics value and search for outliers.

Compute aggregation across several metrics.

Batching

Send batch of measures in one single HTTP call.

Trigger alarms

Using Aodh to evaluate your alarms.

Compression

Using LZ4 compression to compress data on the fly. Fast, reduce storage usage between x2-5.

Collectd, Nagios & statsd support

If you're already a Graphite user or you're polling tool support statsd, it's drop-in compatible.

Same for Nagios.

Grafana

Display charts, build dashboards.

Multi-tenant

ACL that guarantees your different tenants can't see each other resources. But the admin can see everything. Customizable.

HTTP REST API

That's what's used by the 'gnocchi' CLI. Add --debug to discover the HTTP requests, or read the API specs!

A decorative graphic featuring several overlapping circles in various colors (orange, yellow, pink, green, teal, blue) and dashed outlines. A large teal number '2' is centered within a large dashed circle. The text 'How does it work?' and 'How the magic happens.' is positioned below the number '2'.

2

How does it work?
How the magic happens.

Data model

Description of resources: attributes, like a class in Python

A time series

Instances of resource types

How to aggregate metrics

Archive policy

Archive policy

Metric
Metric
Metric
Metric
Metric
Metric

Resource

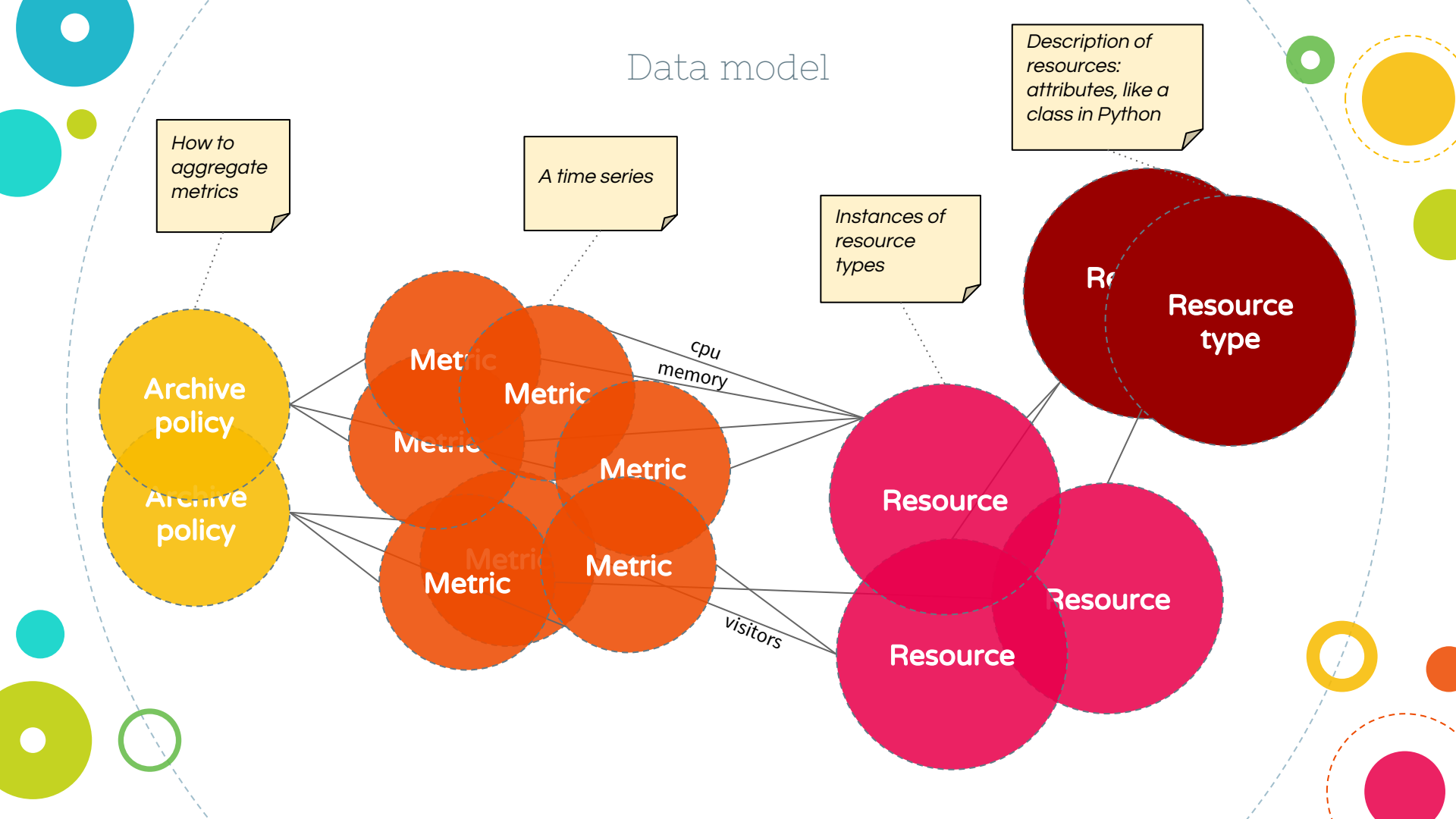
Resource

Resource type

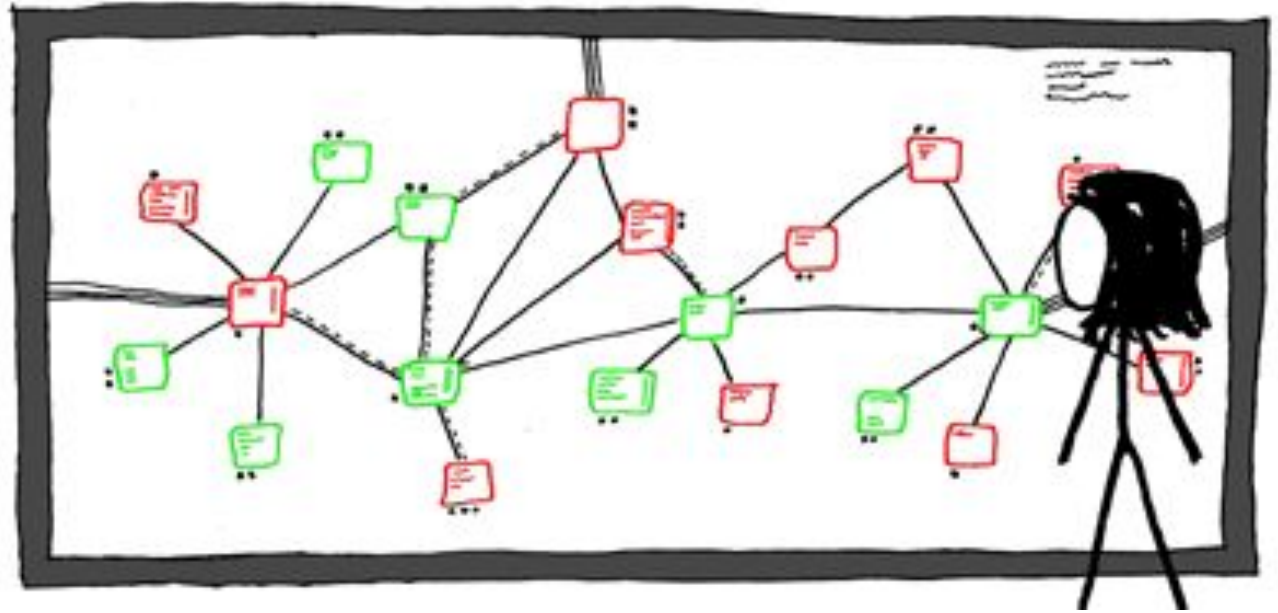
Resource

cpu
memory

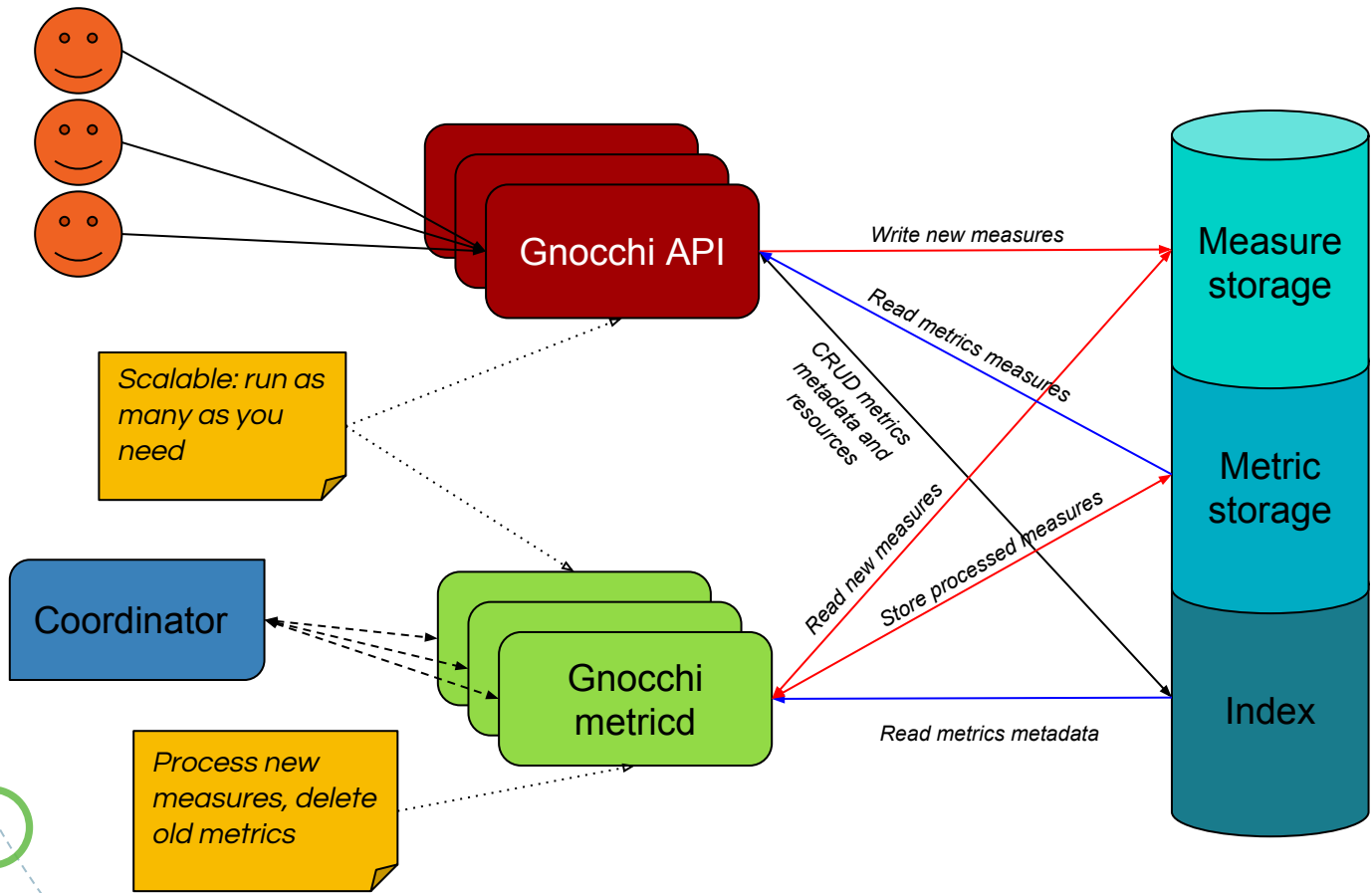
visitors



Index



Architecture





Backends

Index

Any RDBMS supported by SQLAlchemy.

Best choice:
PostgreSQL.

Though **MySQL** is also supported, sigh.



Storage

Simple deployment?
Plain files (with NFS if you want).

Scalable and robust?
Go for **Ceph.**

Got OpenStack?
Leverage **Swift.**

On AWS, use **S3.**



Magic libraries

Carbonara

Timeseries library
based on **Pandas** and
Numpy.

Not standalone yet.
Performance getting
good.



Tooz

Distributed Lock
Management

+

Group membership

Lots of drivers:

Memcached,
ZooKeeper,
PostgreSQL, MySQL,
Redis, etcd...



3

How to use it?
Basic things you need to know

Installation

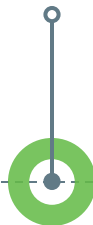
`pip install gnocchi[file,postgresql]
gnocchiclient`



`$EDITOR
/etc/gnocchi/gnocchi.conf`



Run:
`gnocchi-upgrade`



Run:
`gnocchi-api &
gnocchi-metricd &`



List archive policies and create a metric

→ `gnocchi archive-policy list`

name	back_window	definition	aggregation_methods
high	0	- points: 3600, granularity: 0:00:01, timespan: 1:00:00 - points: 10080, granularity: 0:01:00, timespan: 7 days, 0:00:00	std, count, 95pct, min, max, sum, median, mean
medium	0	- points: 8760, granularity: 1:00:00, timespan: 365 days, 0:00:00 - points: 1440, granularity: 0:01:00, timespan: 1 day, 0:00:00 - points: 168, granularity: 1:00:00, timespan: 7 days, 0:00:00	std, count, 95pct, min, max, sum, median, mean
low	0	- points: 365, granularity: 1 day, 0:00:00, timespan: 365 days, 0:00:00 - points: 12, granularity: 0:05:00, timespan: 1:00:00 - points: 24, granularity: 1:00:00, timespan: 1 day, 0:00:00 - points: 30, granularity: 1 day, 0:00:00, timespan: 30 days, 0:00:00	std, count, 95pct, min, max, sum, median, mean

→ `gnocchi metric create --archive-policy-name low`

Field	Value
archive_policy/aggregation_methods	std, count, 95pct, min, max, sum, median, mean
archive_policy/back_window	0
archive_policy/definition	- points: 12, granularity: 0:05:00, timespan: 1:00:00 - points: 24, granularity: 1:00:00, timespan: 1 day, 0:00:00 - points: 30, granularity: 1 day, 0:00:00, timespan: 30 days, 0:00:00
archive_policy/name	low
created_by_project_id	admin
created_by_user_id	admin
id	95fdc8ff-1aed-4dd3-b65b-bfb53f91081b
name	None
resource/id	None

Send & retrieve measures

```
→ gnocchi measures add -m 2016-05-16T12:00:00@42 -m 2016-05-16T12:01:03@45 -m 2016-05-16T12:06:07@22  
95fdc8ff-1aed-4dd3-b65b-bfb53f91081b
```

```
→ gnocchi measures show 95fdc8ff-1aed-4dd3-b65b-bfb53f91081b
```

```
+-----+-----+  
| timestamp                | granularity | value |  
+-----+-----+-----+  
| 2016-05-16T00:00:00+00:00 |      86400.0 | 36.3333333333 |  
| 2016-05-16T12:00:00+00:00 |      3600.0 | 36.3333333333 |  
| 2016-05-16T12:00:00+00:00 |       300.0 |    43.5 |  
| 2016-05-16T12:05:00+00:00 |       300.0 |    22.0 |  
+-----+-----+-----+
```

```
→ gnocchi measures show --aggregation min 95fdc8ff-1aed-4dd3-b65b-bfb53f91081b
```

```
+-----+-----+-----+  
| timestamp                | granularity | value |  
+-----+-----+-----+  
| 2016-05-16T00:00:00+00:00 |      86400.0 |    22.0 |  
| 2016-05-16T12:00:00+00:00 |      3600.0 |    22.0 |  
| 2016-05-16T12:00:00+00:00 |       300.0 |    42.0 |  
| 2016-05-16T12:05:00+00:00 |       300.0 |    22.0 |  
+-----+-----+-----+
```

```
→ gnocchi measures show --aggregation 95pct 95fdc8ff-1aed-4dd3-b65b-bfb53f91081b
```

```
+-----+-----+-----+  
| timestamp                | granularity | value |  
+-----+-----+-----+  
| 2016-05-16T00:00:00+00:00 |      86400.0 |    44.7 |  
| 2016-05-16T12:00:00+00:00 |      3600.0 |    44.7 |  
| 2016-05-16T12:00:00+00:00 |       300.0 |   44.85 |  
| 2016-05-16T12:05:00+00:00 |       300.0 |    22.0 |  
+-----+-----+-----+
```

Create a resource

```
➔ gnocchi resource-type create --attribute name:string --attribute host:string server
```

Field	Value
attributes/host	max_length=255, min_length=0, required=True, type=string
attributes/name	max_length=255, min_length=0, required=True, type=string
name	server

```
➔ gnocchi resource create --attribute name:www-42 --attribute host:compute1 --create-metric cpu:medium --create-metric memory:low --type server `uuidgen`
```

Field	Value
created_by_project_id	admin
created_by_user_id	admin
ended_at	None
host	compute1
id	e4c2eab7-52ed-4447-bbcb-48cb04f12015
metrics	cpu: d51d8ba3-ab06-4f0c-af6c-d88dbac8c2a8 memory: 0240ceb8-d1d6-435d-a37c-f7f3bf99a388
name	www-42
original_resource_id	E4C2EAB7-52ED-4447-BBCB-48CB04F12015
project_id	None
revision_end	None
revision_start	2016-05-16T13:35:43.985927+00:00
started_at	2016-05-16T13:35:43.985815+00:00
type	server
user_id	None

Update a resource

```
→ gnocchi resource update --attribute host:compute2 --type server e4c2eab7-52ed-4447-bbcb-48cb04f12015
```

Field	Value
created_by_project_id	admin
created_by_user_id	admin
ended_at	None
host	compute2
id	e4c2eab7-52ed-4447-bbcb-48cb04f12015
metrics	cpu: d51d8ba3-ab06-4f0c-af6c-d88dbac8c2a8 memory: 0240ceb8-d1d6-435d-a37c-f7f3bf99a388
name	www-42
original_resource_id	E4C2EAB7-52ED-4447-BBCB-48CB04F12015
project_id	None
revision_end	None
revision_start	2016-05-16T13:37:38.140460+00:00
started_at	2016-05-16T13:35:43.985815+00:00
type	server
user_id	None

See previous updates in JSON

```
→ gnocchi resource history --format json --details e4c2eab7-52ed-4447-bbcb-48cb04f12015
```

```
[  
  {  
    "created_by_user_id": "admin",  
    "started_at": "2016-05-16T13:35:43.985815+00:00",  
    "user_id": null,  
    "revision_end": "2016-05-16T13:37:38.140460+00:00",  
    "ended_at": null,  
    "created_by_project_id": "admin",  
    "metrics": "cpu: d51d8ba3-ab06-4f0c-af6c-d88dbac8c2a8\nmemory: 0240ceb8-d1d6-435d-a37c-f7f3bf99a388",  
    "host": "compute1",  
    "revision_start": "2016-05-16T13:35:43.985927+00:00",  
    "project_id": null,  
    "type": "server",  
    "id": "e4c2eab7-52ed-4447-bbcb-48cb04f12015",  
    "name": "www-42"  
  },  
  {  
    "created_by_user_id": "admin",  
    "started_at": "2016-05-16T13:35:43.985815+00:00",  
    "user_id": null,  
    "revision_end": null,  
    "ended_at": null,  
    "created_by_project_id": "admin",  
    "metrics": "cpu: d51d8ba3-ab06-4f0c-af6c-d88dbac8c2a8\nmemory: 0240ceb8-d1d6-435d-a37c-f7f3bf99a388",  
    "host": "compute2",  
    "revision_start": "2016-05-16T13:37:38.140460+00:00",  
    "project_id": null,  
    "type": "server",  
    "id": "e4c2eab7-52ed-4447-bbcb-48cb04f12015",  
    "name": "www-42"  
  }  
]
```

Send & get measures on a metric attached to a resource & search

```
→ gnocchi measures add -m 2016-05-16T12:00:00@42 -m 2016-05-16T12:01:03@45 -m 2016-05-16T12:06:07@22 --resource-id e4c2eab7-52ed-4447-bbcb-48cb04f12015 cpu
→ gnocchi measures show --resource-id e4c2eab7-52ed-4447-bbcb-48cb04f12015 cpu
```

timestamp	granularity	value
2016-05-16T00:00:00+00:00	86400.0	36.3333333333
2016-05-16T12:00:00+00:00	3600.0	36.3333333333
2016-05-16T12:00:00+00:00	60.0	42.0
2016-05-16T12:01:00+00:00	60.0	45.0
2016-05-16T12:06:00+00:00	60.0	22.0

```
→ gnocchi resource search --type server host=compute2
```

id	type	project_id	user_id	started_at	ended_at	revision_start	revision_end
e4c2eab7-52ed-4447-bbcb-48cb04f12015	server	None	None	2016-05-16T13:35:43.985815+00:00	None	2016-05-16T13:37:38.140460+00:00	None

Using in your Python app

```
from gnocchiclient import auth
from gnocchiclient import client

# Create connection to gnocchi
g = client.Client(version=1, auth=auth.GnocchiBasicPlugin("admin", "http://localhost:8041"))

# Create an orphan metric
metric = g.metric.create({"archive_policy_name": "low"})

# Send some measures to it
g.metric.add_measures(metric["id"], [{"timestamp": "2017-02-05 10:00:00", "value": 42}])

# Create a generic resource called myapp that have a visitor metrics
resource = g.resource.create("generic", {"id": "myapp", "metrics": {"visitors": {}}})

# Send the current number of visitors on our app
g.metric.add_measures("visitors", [{"timestamp": "2017-02-05 10:00:00", "value": 42}, {"timestamp": "2017-02-05 10:01:00", "value": 34}],
resource_id="myapp", )

# Get maximum of visitors over an hour, equivalent to:
# $ gnocchi measures show -f json --resource-id myapp --aggregation max --granularity=300 visitors
measures = g.metric.get_measures("visitors", resource_id="myapp", aggregation="max", granularity=300)
# [
# {
#   "timestamp": "2017-02-05T10:00:00+00:00",
#   "value": 42.0,
#   "granularity": 300.0
# }
# ]
```


Grafana support



Thanks!



<http://gnocchi.xyz>

#gnocchi @ Freenode

Any questions?

You can find me at @juldanjou and julien@danjou.info