THE ITALIAN RESEARCH & EDUCATION NETWORK

Consortium GARR

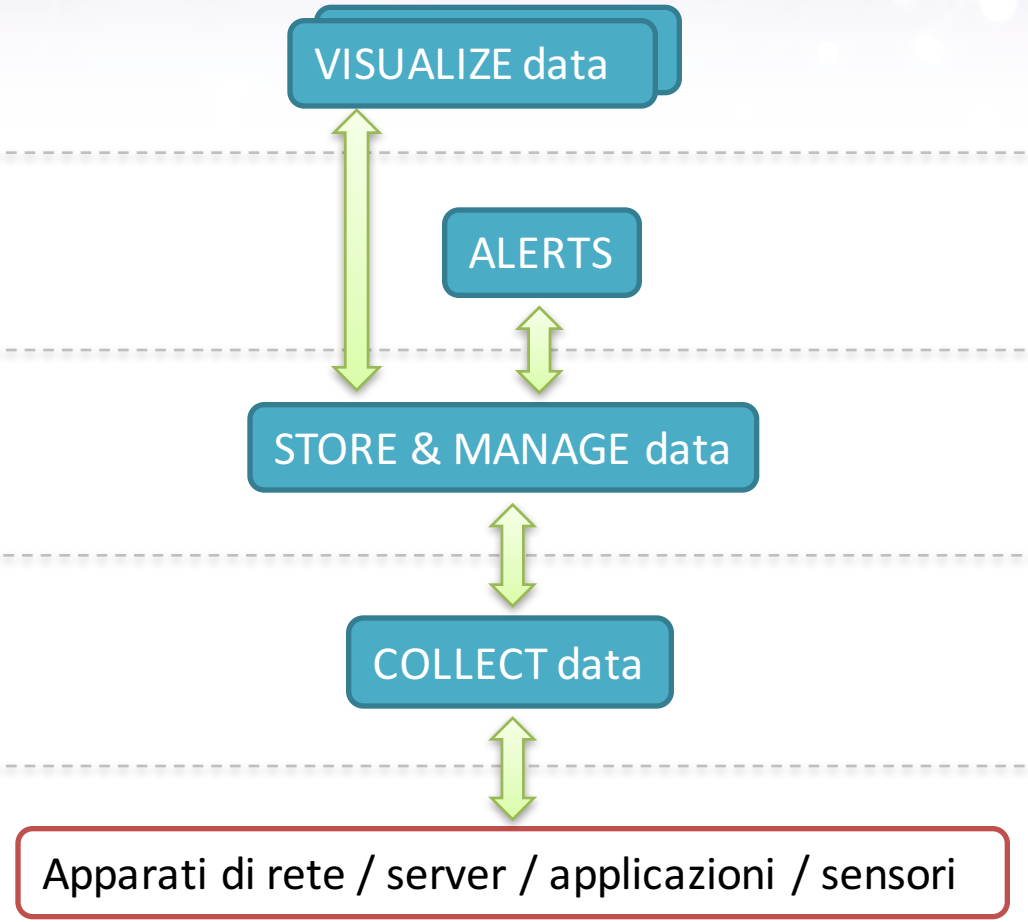
Monitoring 2016

InfluxDB, Telegraf, Kapacitor, Grafana

Giovanni Cesaroni (GARR)

Workshop GARR 2016, Roma, 18-21/04/2016

Architettura di un sistema di monitoring



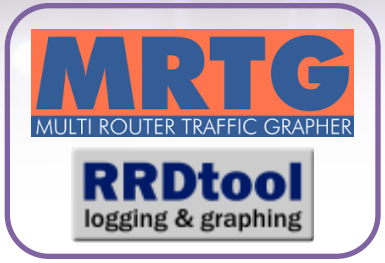
Come scegliere il sistema giusto?

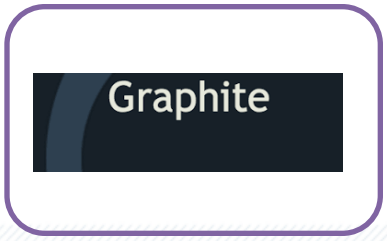
Dipende:

- Cosa dobbiamo monitorare?
- Quali sono i nostri numeri?

- Competenze, tempo, risorse

Il mondo open





other

Survey


MRTG
MULTI ROUTER TRAFFIC GRAPHER
RRDtool
logging & graphing
5
Nagios®
10
smokeping
1
kibana
4
ZABBIX
10
ICINGA
0
cacti
5
logstash
2
openNMS®
0
MUNIN
0
Zenoss
Open Source IT Monitoring
1
elastic
4
Graphite
1
Grafana
1
influxdata
1
Nedi
1

Survey results:

ZABBIX

MRTG
MULTI ROUTER TRAFFIC GRAPHER

RRDtool
logging & graphing

Nagios®

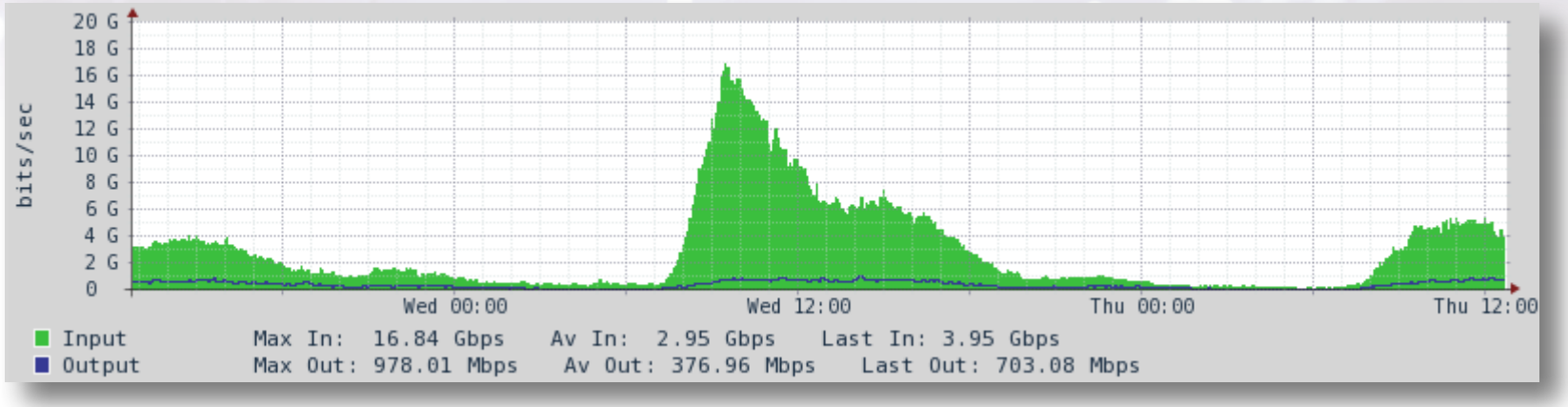
kibana

cacti

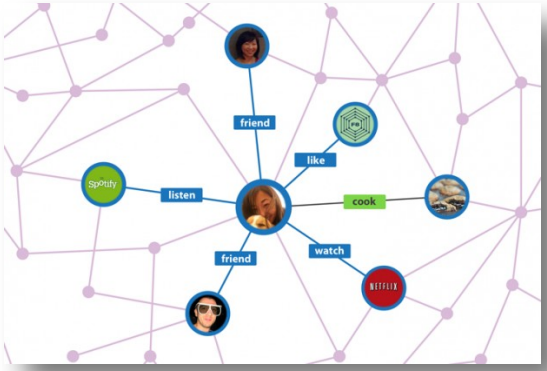
elastic

Data: time series

metrics



time



?

Time Series DBMS

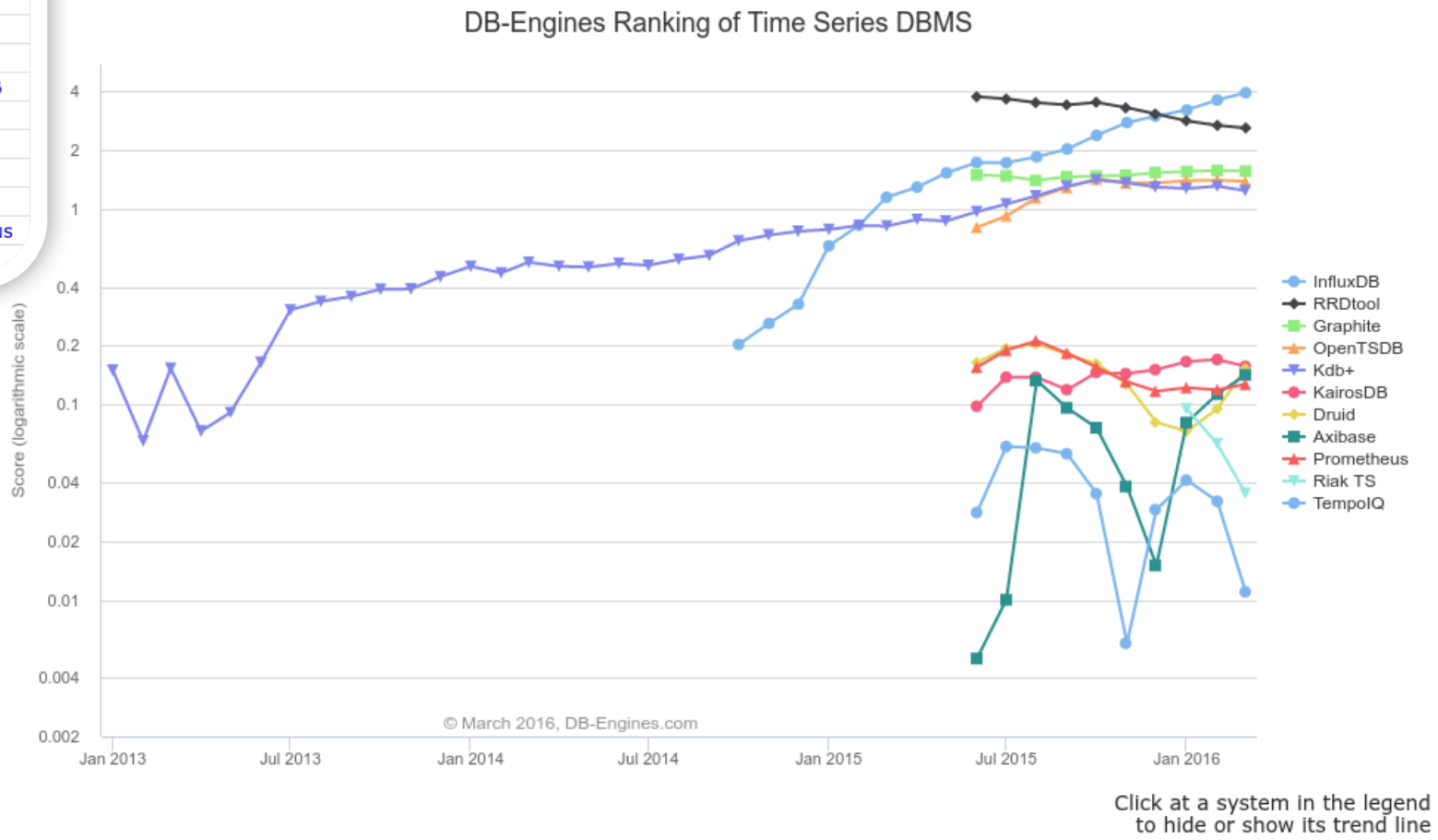
A time series database (TSDB) is a software system that is optimized for handling time series data, arrays of numbers indexed by time (a datetime or a datetime range).

Time Series DBMS

- Efficiente scrittura: massiva, molto frequente
- Efficiente lettura (time-related): veloce
- Downsampling

DB-Engines Ranking of Time Series DBMS

Rank Mar 2016	Rank Feb 2016	Rank Mar 2015	DBMS
1.	1.	1.	InfluxDB
2.	2.		RRDtool
3.	3.		Graphite
4.	4.		OpenTSDB
5.	5.	↓ 2.	Kdb+
6.	6.		KairosDB
7.	↑ 9.		Druid
8.	8.		Axibase
9.	↓ 7.		Prometheus
10.	10.		Riak TS



Click at a system in the legend to hide or show its trend line

Le ragioni della scelta di oggi

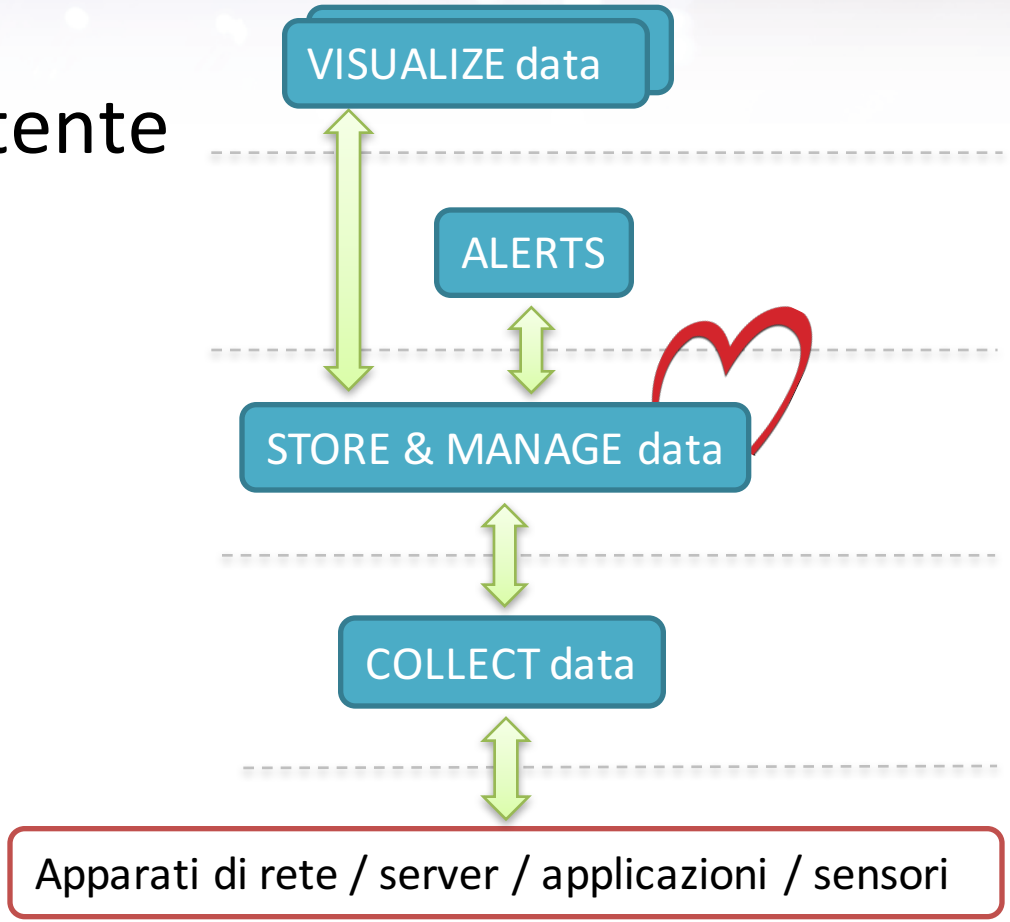
SI

- Moderno, promettente
- Espressivo
- Utile

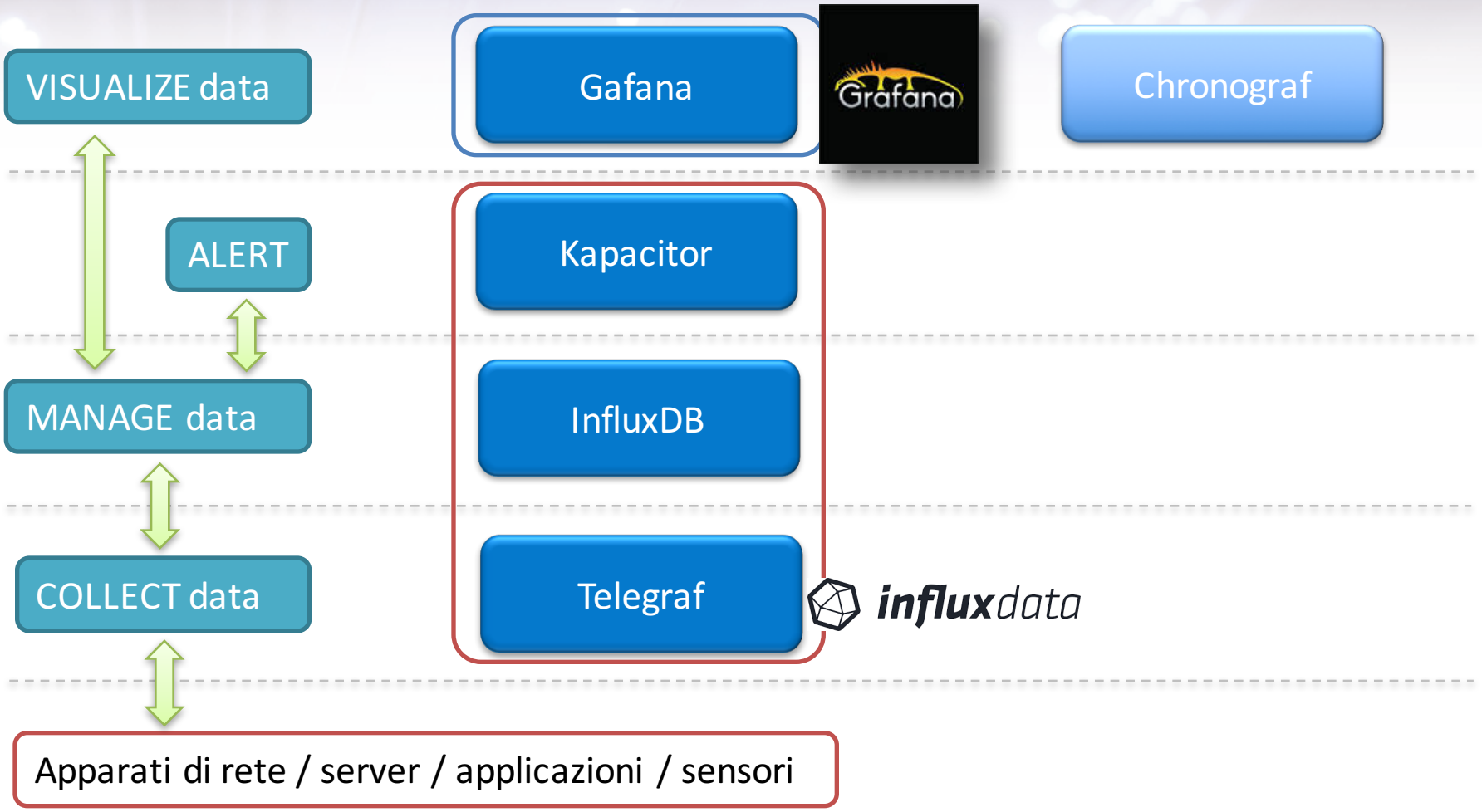
NO

- Stabile

UPCOMING RELEASES: 0.*,1.*

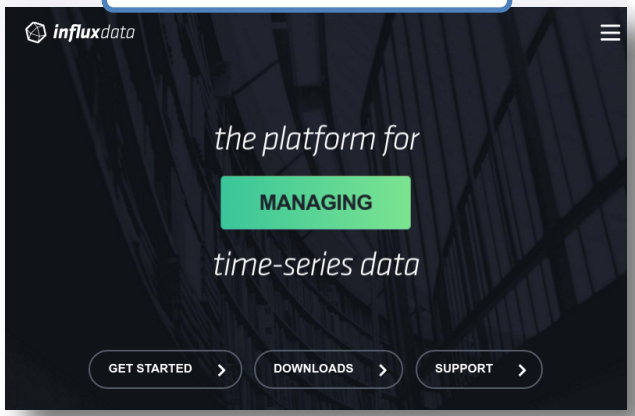


La scelta di oggi

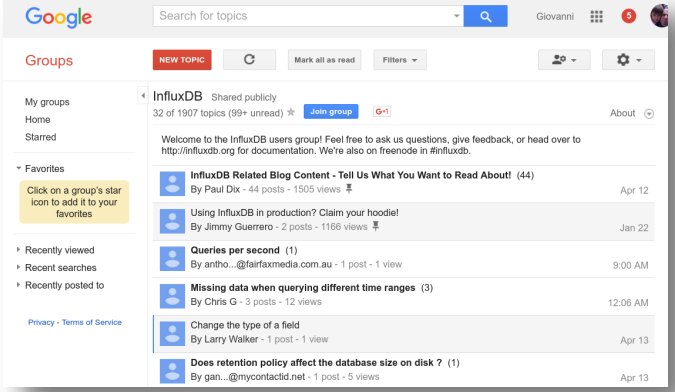


Canali di documentazione

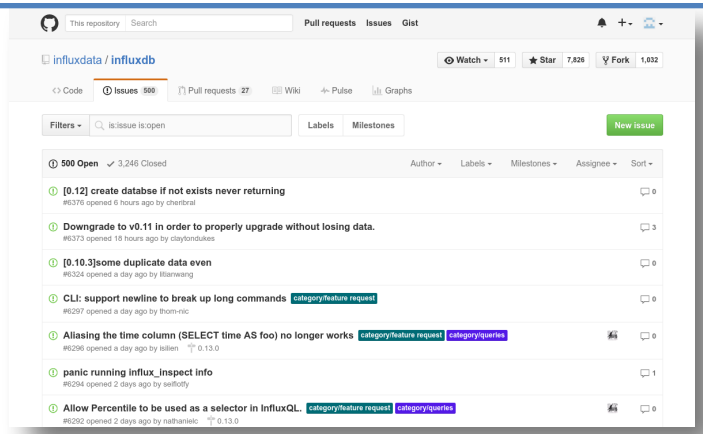
https://influxdata.com/



https://groups.google.com/forum/#!forum/influxdb



https://github.com/influxdata/influxdb/issues



InfluxDB

- Time series database, no external dependencies

Features

- Built-in HTTP API
- Data can be tagged, allowing very flexible querying.
- SQL-like query language.
- Clustering is supported out of the box.
- Simple to install and manage, and fast to get data in and out.
- It aims to answer queries in real-time. That means every data point is indexed as it comes in and is immediately available in queries that should return in < 100ms.

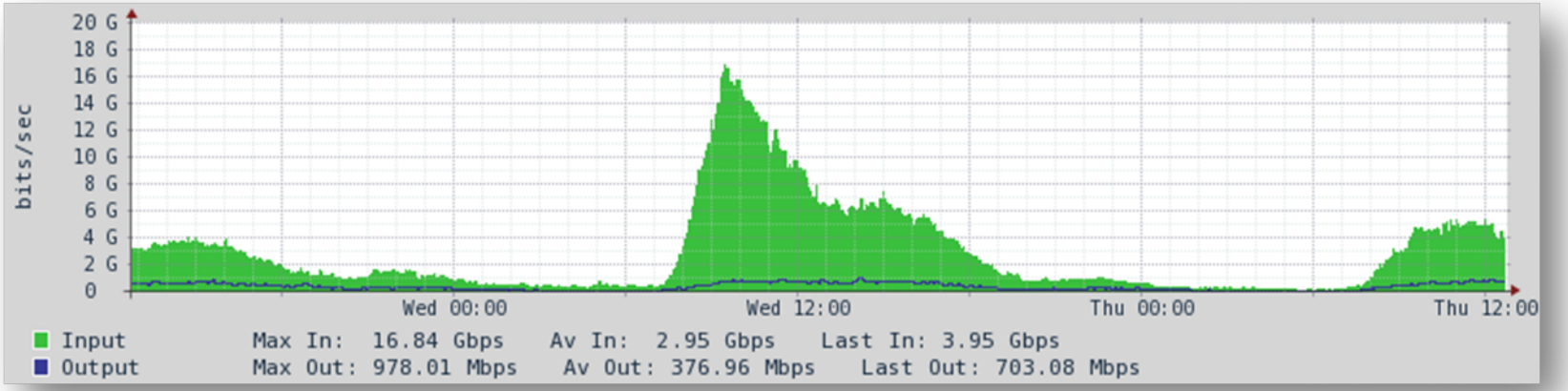
InfluxDB: key concepts & data model

- Databases
  - A logical container for users, retention policies, continuous queries, and time series data.

InfluxDB: key concepts & data model

Traffic load = ***Measurement***

Hostname = ***TAG (host='myhost')*** Interface = ***TAG (ifName='myifName')***



Bitrate Input = ***FIELD (ifHCInOctets='value')***

Bitrate Output = ***FIELD (ifHCOutOctets='value')***

InfluxDB: key concepts & data model

- Databases
- Measurements
  - traffic load
- Tags
  - host, interfaccia *indexed*
- Series
  - measurement + unique tagset: traffic load di host su interfaccia
- Fields
  - traffic load IN, traffic load Out
- Points
  - Time
  - Fields (key,value) = metrics
  - Tags (key,value) = metadata

InfluxDB: key concepts & data model

- Measurements
  - CPU
- Tags
  - host, cpu
- Series
  - measurement + unique tagset: CPU di host
- Fields
  - usage_idle, usage_user

```
> SELECT cpu, host, edificio, usage_idle, usage_user FROM cpu  limit 6
name: cpu
---------
time                 cpu       host  usage_idle          usage_user
1459510950000000000  cpu-total  pc    94.18721691004838   3.472571716168557
1459510950000000000  cpu3       pc    97.47729566094942   2.1190716448028493
1459510950000000000  cpu1       pc    90.35175879403482   2.2110552763883495
1459510950000000000  cpu0       pc    96.36730575189748   2.5227043390579826
1459510950000000000  cpu2       pc    92.2922922921211    7.207207207195479
1459510960000000000  cpu0       pc    90.44715447151766   7.723577235777317
```

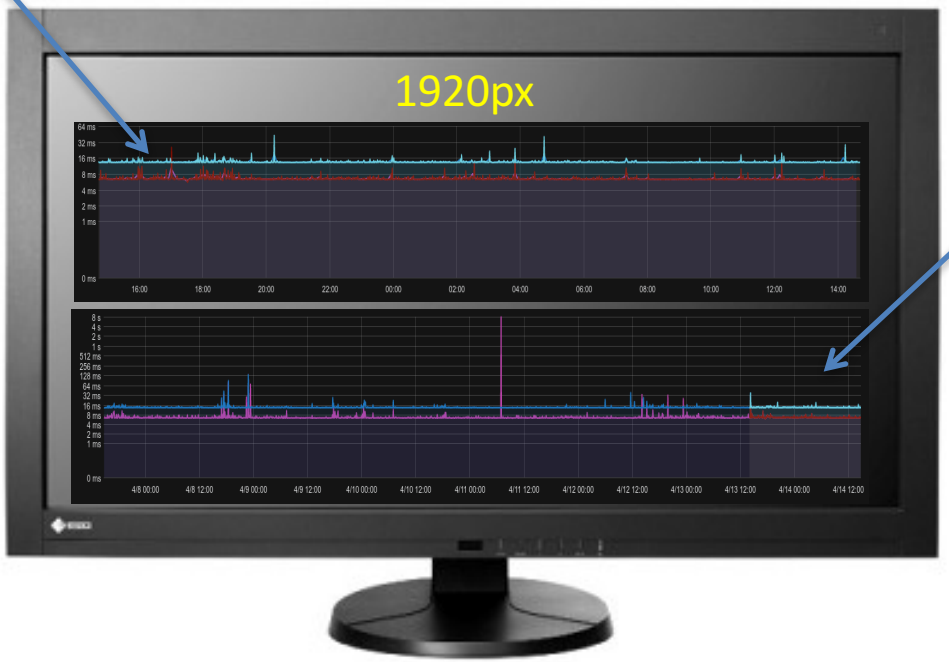
Downsampling, esempio

Misure ogni 1m
60 punti in 1h
60*24 = 1440 punti in 1d

1 settimana
60*24*7 = 10080

Aggregazione temporale
In intervalli di 7m

1440

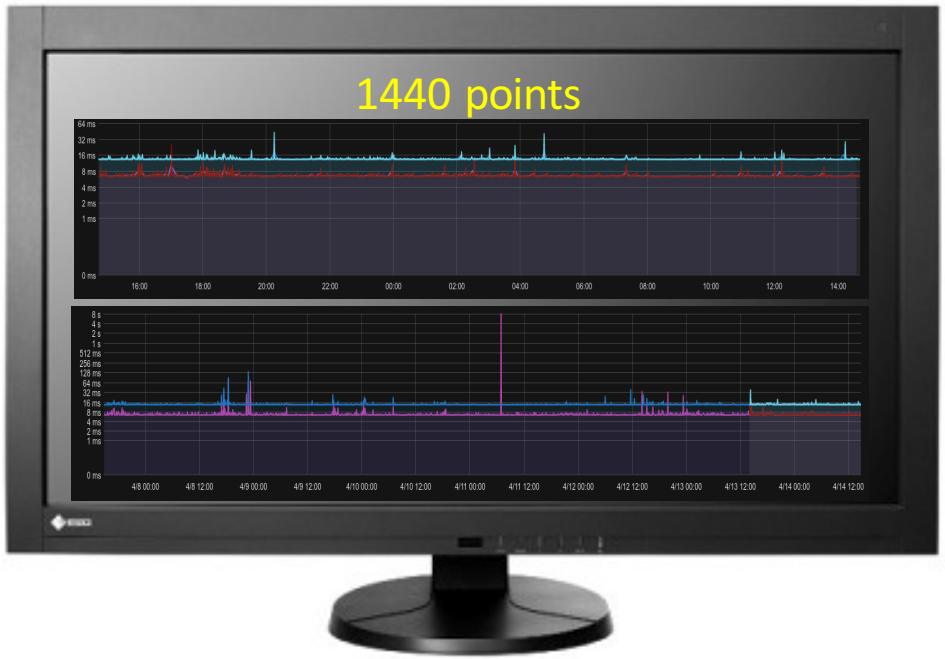


Downsampling, esempio

Aggregazione temporale
in intervalli di 7m

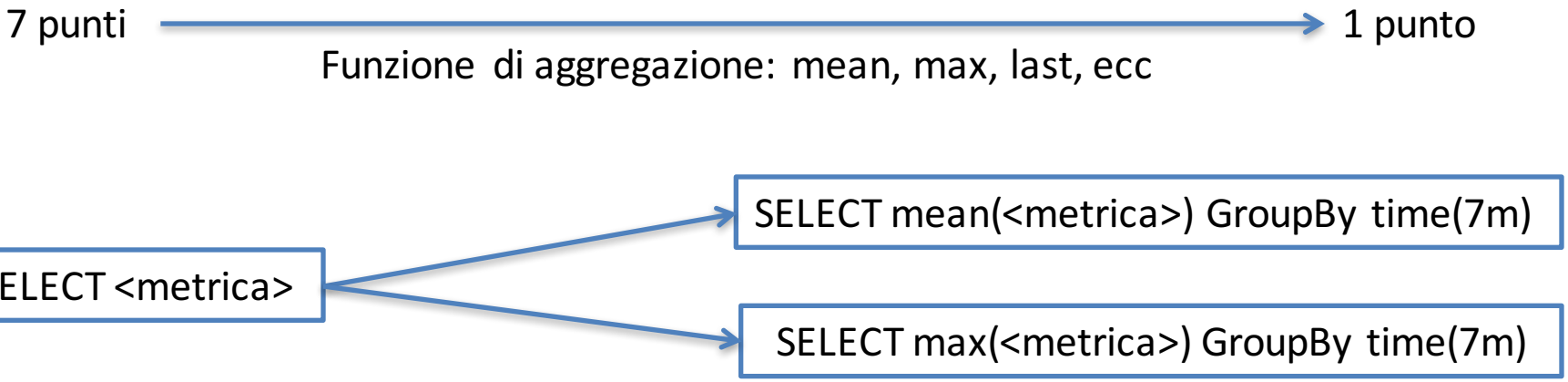
7 punti → 1 punto

Funzione di aggregazione: mean, max, last, ecc



Downsampling, esempio

Aggregazione temporale
in intervalli di 7m



InfluxDB: installazione

Installiamo in pochi secondi…. Da https://influxdata.com/downloads/#influxdb

```
$ wget https://s3.amazonaws.com/influxdb/influxdb_0.12.1-1_amd64.deb
$ sudo dpkg -i influxdb_0.12.1-1_amd64.deb
$ service influxd start

$ influx
Connected to http://localhost:8086 version 0.12.0~n201603311344
InfluxDB shell 0.12.1
>

> CREATE DATABASE mydb
```

InfluxDB

```
> SHOW DATABASES
name: databases
---------------
name
_internal
mydb

> USE mydb
Using database mydb

> SHOW MEASUREMENTS

> SHOW SERIES

> SELECT
```

InfluxDB: line protocol

measurement[,**tag_key1**=tag_value1...] **field_key**=field_value[,field_key2=field_value2] [**timestamp**]

measurement,**tagset** **fielset** **timestamp**

CPU,**host=<myhost>**,**cpu=<which_cpu>** | **usage_idle=<value>, usage_user =<value>** | **timestamp**

ifHCInOctets,**host=<myhost>**,**instance=<which_interface>**,**units=octets** | **ifHCInOctets=<value>** | **timestamp**

InfluxDB: line protocol, insert

```
> INSERT cpu,host=serverA,edificio=A,piano=4 value=0.10
> select * from cpu
name: cpu
---------
time                    edificio  host     piano  value
1458738048007774121     A         serverA  4      0.1


> INSERT cpu,host=serverA,edificio=A,piano=4 value=0.15 1458725894404765956
> select * from cpu
name: cpu
---------
time                    edificio  host     piano  value
1458725894404765956     A         serverA  4      0.15
1458738048007774121     A         serverA  4      0.1
```

InfluxDB: line protocol, insert

```
> use mydb

> INSERT cpu,host=serverA,edificio=A,piano=4 value=0.15
> show measurements
name: measurements
------------------
name
cpu

> show series
key
cpu,host=serverA,edificio=A,piano=4

> select * from cpu
name: cpu
---------
time                edificio  host    piano  value
1460971739223669018 A         serverA 4      0.15
```

InfluxDB: line protocol, multi insert

```
curl -i -XPOST 'http://localhost:8086/write?db=mydb'  --data-binary
'cpu,host=serverA,edificio=A  value=0.10 1434055562000000000'

curl -i -XPOST 'http://localhost:8086/write?db=mydb'
      --data-binary
            'cpu,host=serverB  value=0.30
             cpu,host=serverB,edificio=A  value=0.55 14225685437029002 57
             cpu,sala=2,host=serverA,edificio=A  value=2.0 14225685437029002 57'

curl -i -XPOST 'http://localhost:8086/write?db=mydb'  --data-binary @cpu_data.txt
```

InfluxDB is a schemaless database. You can add new measurements, tags, and fields at any time. Note that if you attempt to write data with a different type than previously used (for example, writing a string to a field that previously accepted integers), InfluxDB will reject those data.

Note: If your data file has more than 5,000 points, it may be necessary to split that file into several files in order to write your data in batches to InfluxDB.

InfluxDB: line protocol, multi insert

```
$ curl -G 'http://localhost:8086/query?pretty=true'  --data-urlencode "db=mydb"  --data-urlencode
"q=SELECT value FROM cpu WHERE edificio='A';"
{
   "results": [
      {
         "series": [
            {
               "name": "cpu",
               "columns": [
                  "time",
                  "value"
               ],
               "values": [
                  [
                     "2016-03-23T09:38:14.404765956Z",
                     0.15
                  ],
                  [
                     "2016-03-23T13:00:48.007774121Z",
                     0.1
                  ]
               ]
            }
         ]
   }
```

For large queries, results are returned in batches of 10,000 points unless you use the query string parameter chunk_size to explicitly set the batch size. Esempio con 20000 punti: --data-urlencode "chunk_size=20000"

InfluxDB: *Downsampling* and *Data Retention*

- Gestire centinaia di migliaia di punti al secondo
- Manteniamo i dati per lungo tempo
  - Downsample e' la soluzione naturale
  - Alta precisione per un tempo limitato
  - Dati aggregati per tempi lunghi

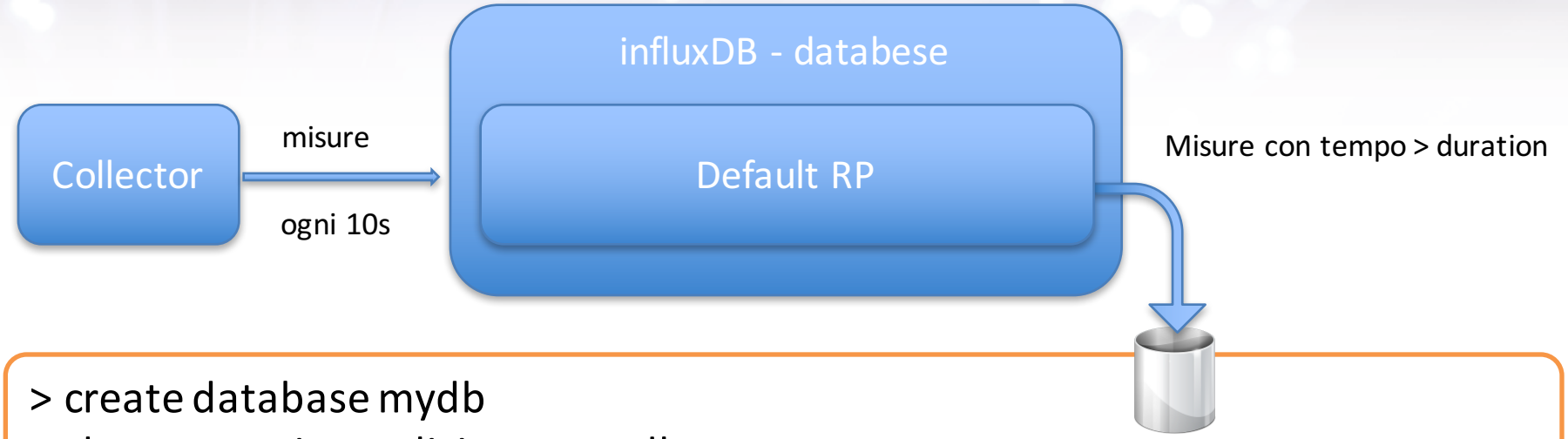
Retention Policies (RP)

- Per quanto tempo manteniamo i dati
- Quante repliche scriviamo

Continuous Queries(CQ)

- Query automatica e periodica
- Estrae i dati relativi ad un intervallo
- Li conserva come se fossero misure

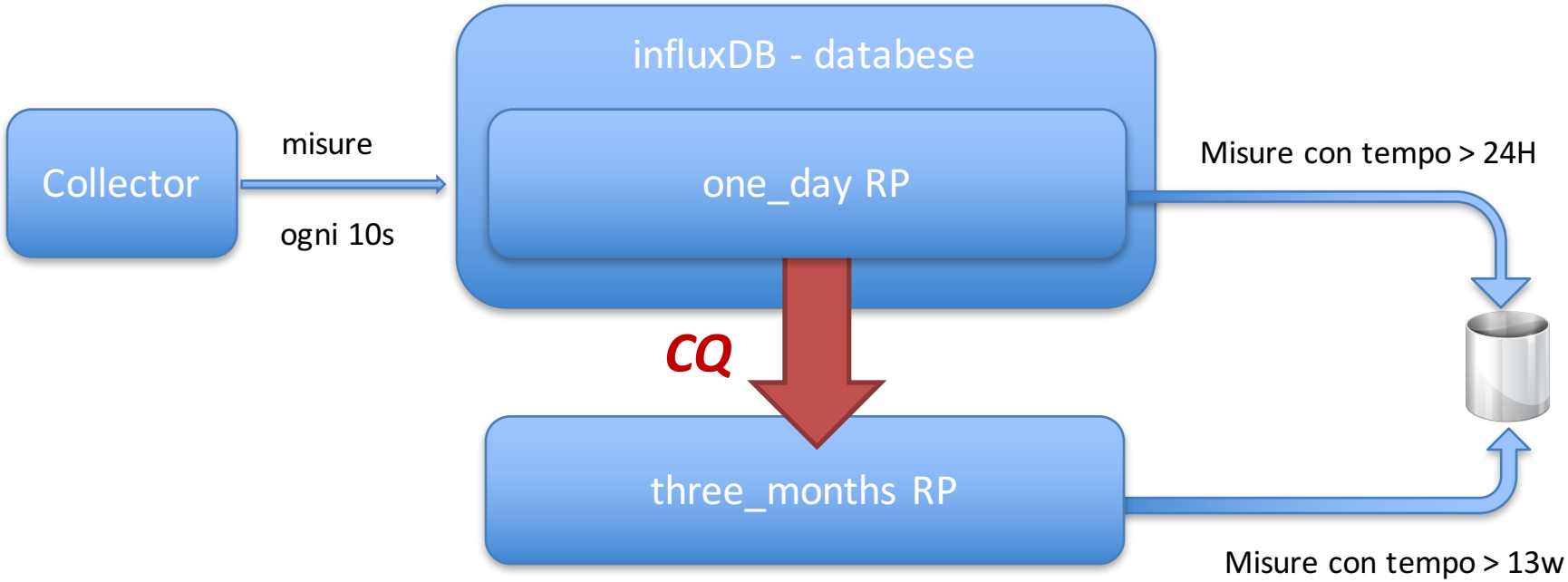
InfluxDB: RP



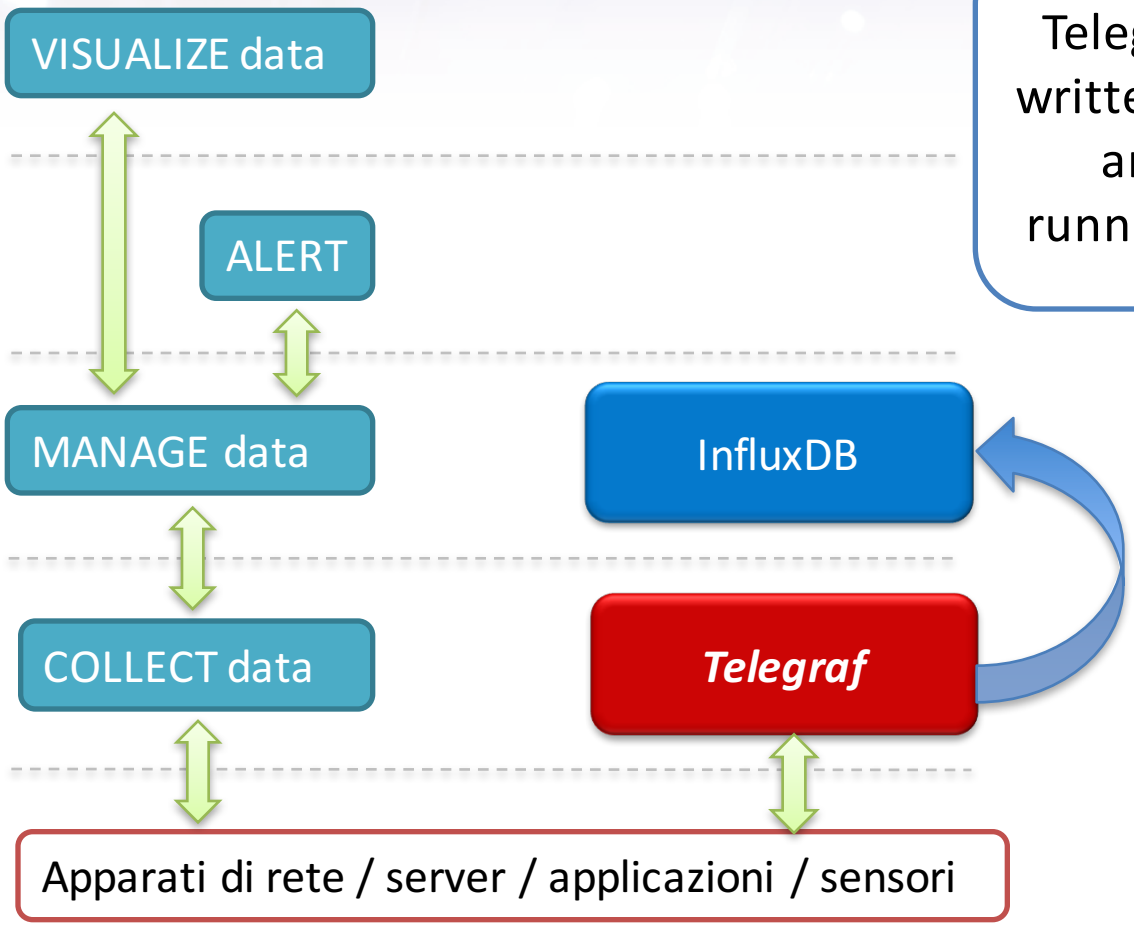

```
> create database mydb
> show retention policies on mydb
name      duration      shardGroupDuration    replicaN      default
default             0          168h0m0s             1              true

> ALTER RETENTION POLICY default ON mydb DURATION 1h DEFAULT
> show retention policies on mydb
name      duration      replicaN      default
default   1h0m0s        1             true
```

InfluxDB: RP & CQ

> CREATE RETENTION POLICY three_months ON mydb DURATION 13w REPLICATION 1

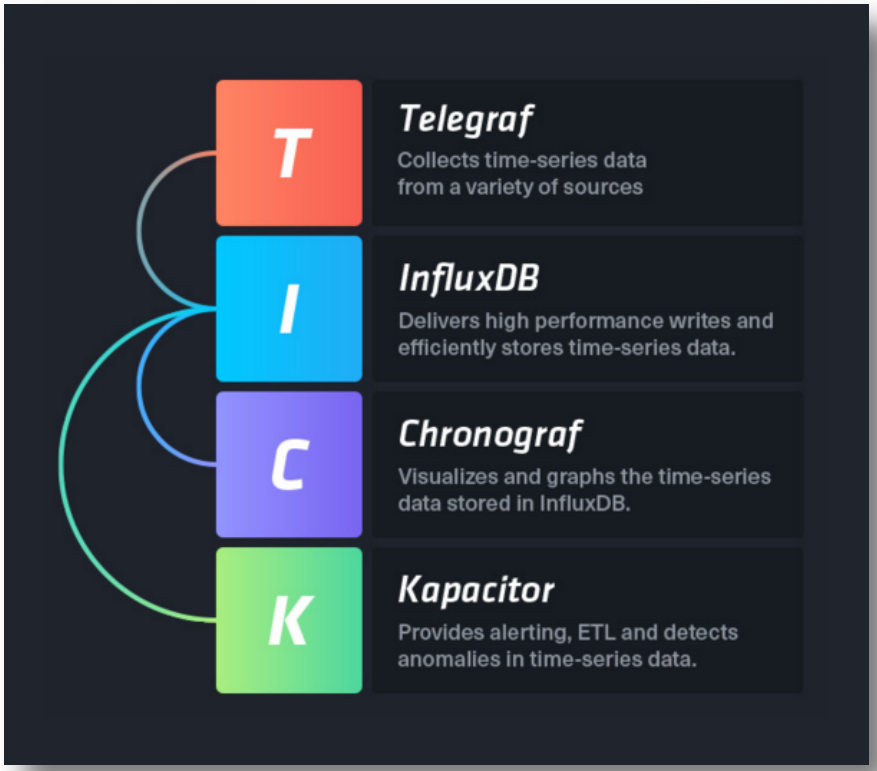


Telegraf



Telegraf is an open source agent written in Go for collecting metrics and data on the system it's running on or from other services

Telegraf

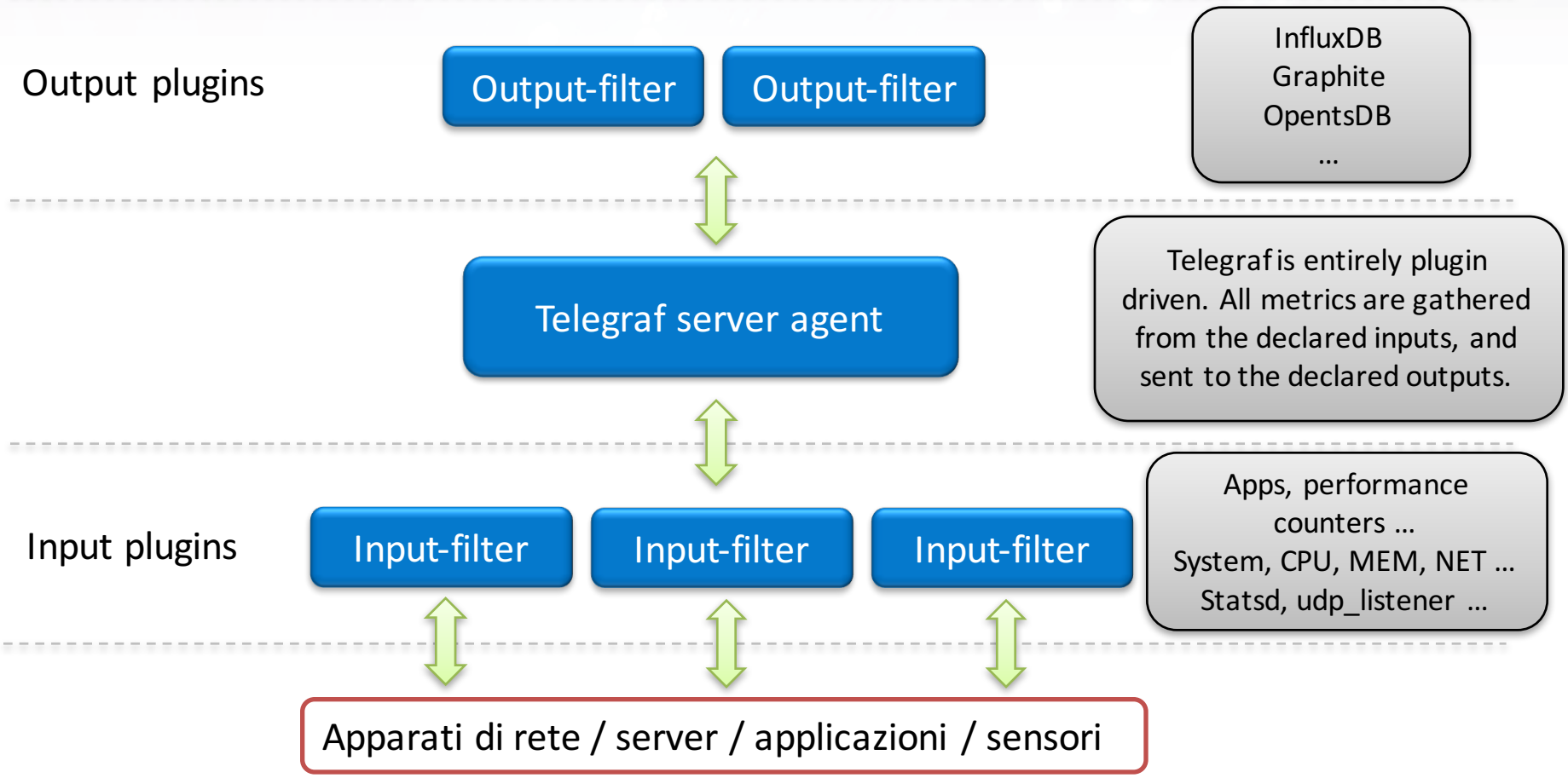


UPCOMING RELEASES

The cadence for Telegraf releases is approximately every 3 weeks.

0.11	0.12	1.0
March 2016	April 2016	May 2016

Telegraf: architettura



Cosa facciamo ora?

1. *Come funziona Telegraf*
2. Acquisizione alcune metriche di un host
3. Misure di latenza verso 2 router
4. SNMP input plugin
5. Acquisizione delle risorse CPU/MEM di piu' di un router Juniper
6. Acquisizione traffico delle interfacce di un router

Telegraf: start

```
$ telegraf -help
The flags are:

  -config <file>         configuration file to load
  -test                  gather metrics once, print them to stdout, and exit
  -sample-config         print out full sample configuration to stdout
  -config-directory      directory containing additional *.conf files
  -input-filter          filter the input plugins to enable, separator is :
  -input-list            print all the plugins inputs
  -output-filter         filter the output plugins to enable, separator is :
  -output-list           print all the available outputs
  -usage                 print usage for a  plugin, ie, 'telegraf -usage mysql'
  -debug                 print metrics as they're generated to stdout
  -quiet                 run in quiet mode
  -version               print the version to stdout
```

Telegraf: configuration

- creo la configurazione

$ telegraf -sample-config -input-filter ***cpu*** -output-filter ***influxdb*** > **test.conf**

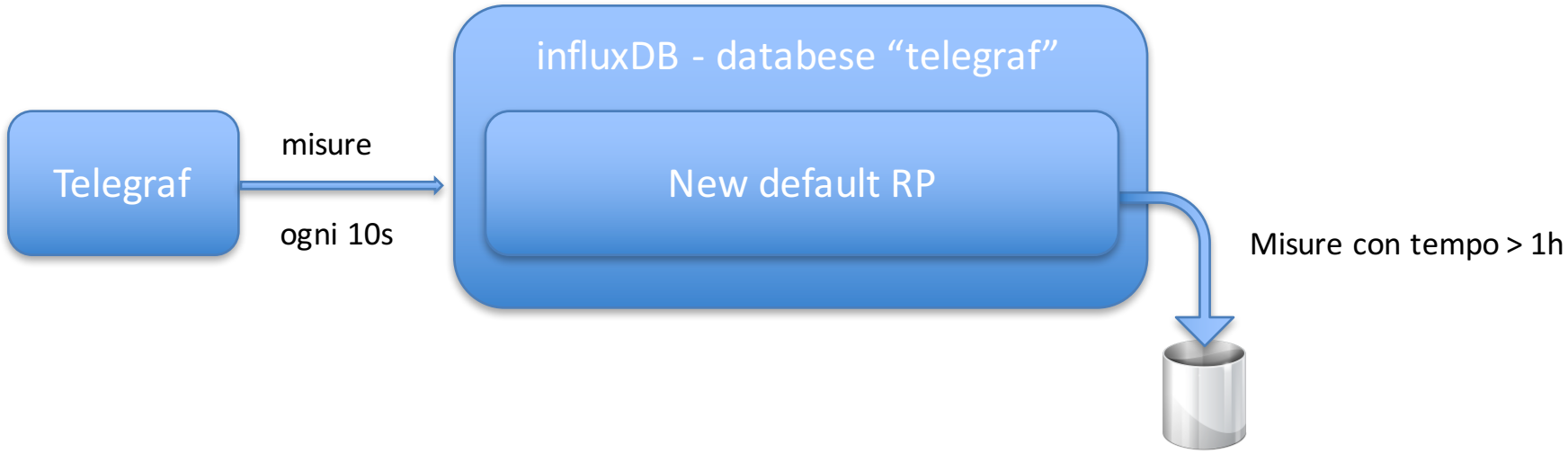
- test acquisizione metriche

$ telegraf -config test.conf –test

```
$ telegraf -config test.conf -test
* Plugin: cpu, Collection 1
* Plugin: cpu, Collection 2
> cpu,cpu=cpu0
usage_guest=0,usage_guest_nice=0,usage_idle=90.3846153866604,usage_iowait=0,usage_irq=0,usage_nice=0
,usage_softirq=0,usage_steal=0,usage_system=1.9230769231139218,usage_user=7.692307692499413
1458816766029872362
> cpu,cpu=cpu1
usage_guest=0,usage_guest_nice=0,usage_idle=88.23529411798275,usage_iowait=0,usage_irq=0,usage_nice=0,
usage_softirq=0,usage_steal=0,usage_system=0,usage_user=11.76470588250767  1458816766029872362
> cpu,cpu=cpu-total
usage_guest=0,usage_guest_nice=0,usage_idle=87.68472906420891,usage_iowait=4.926108374311956,usage_i
rq=0,usage_nice=0,usage_softirq=0,usage_steal=0,usage_system=0.9852216748623912,usage_user=6.40394088
6616744 1458816766029872362
```

Configuriamo RP e CQ per telegraf

```
> create database telegraf
> ALTER RETENTION POLICY default ON telegraf DURATION 1h DEFAULT
> show retention policies on telegraf
name      duration      replicaN      default
default   1h0m0s        1             true
```



Telegraf: run

$ telegraf -config test.conf

2016/03/24 11:55:27 Starting Telegraf (version 0.11.1)
2016/03/24 11:55:27 Loaded outputs: **influxdb**
2016/03/24 11:55:27 Loaded inputs: **cpu**
2016/03/24 11:55:27 Tags enabled: host=pcgarr9
2016/03/24 11:55:27 Agent Config: **Interval:10s,** Debug:false, Quiet:false, Hostname:"pcgarr9", Flush Interval:10s
2016/03/24 11:55:30 Gathered metrics, (10s interval), from 1 inputs in 1.20362ms
2016/03/24 11:55:40 **Gathered metrics, (10s interval), from 1 inputs** in 1.922114ms
2016/03/24 11:55:40 Wrote 5 metrics to output influxdb in 377.002809ms
2016/03/24 11:55:50 Gathered metrics, (10s interval), from 1 inputs in 868.333µs
2016/03/24 11:55:50 **Wrote 5 metrics to output influxdb** in 140.186328ms
2016/03/24 11:56:00 Gathered metrics, (10s interval), from 1 inputs in 1.275503ms
2016/03/24 11:56:00 Wrote 5 metrics to output influxdb in 42.728255ms
2016/03/24 11:56:10 Gathered metrics, (10s interval), from 1 inputs in 4.368422ms
2016/03/24 11:56:10 Wrote 5 metrics to output influxdb in 60.675082ms
2016/03/24 11:56:20 Gathered metrics, (10s interval), from 1 inputs in 822.709µs
2016/03/24 11:56:20 Wrote 5 metrics to output influxdb in 140.398613ms
2016/03/24 11:56:30 Gathered metrics, (10s interval), from 1 inputs in 1.408604ms
2016/03/24 11:56:30 Wrote 5 metrics to output influxdb in 204.426939ms

Telegraf: input-filters

$ telegraf -sample-config -input-filter **cpu:mem:net:disk:netstat** -output-filter influxdb > test.conf

$ telegraf -config test.conf

2016/03/24 12:08:59 Starting Telegraf (version 0.11.1)
2016/03/24 12:08:59 Loaded outputs: influxdb
2016/03/24 12:08:59 **Loaded inputs: cpu disk mem net netstat**
2016/03/24 12:08:59 Tags enabled: host=pcgarr9
2016/03/24 12:08:59 Agent Config: **Interval:10s**, Debug:false, Quiet:false, Hostname:"pcgarr9", Flush Interval:10s
2016/03/24 12:09:00 Gathered metrics, (10s interval), from **5 inputs** in 67.758351ms
2016/03/24 12:09:10 Gathered metrics, (10s interval), from 5 inputs in 63.491281ms
2016/03/24 12:09:10 **Wrote 19 metrics to output influxdb** in 117.278484ms
2016/03/24 12:09:20 Gathered metrics, (10s interval), from 5 inputs in 60.254643ms
2016/03/24 12:09:20 Wrote 12 metrics to output influxdb in 109.474876ms
2016/03/24 12:09:30 Gathered metrics, (10s interval), from 5 inputs in 64.679953ms

Telegraf: run

- configurazioni

```
$ ls  /etc/telegraf/telegraf.d/
re1.aq1.conf
telegraf_ping.conf
telegraf_snmp_router_resources.conf
telegraf_snmp_traffic.conf
```

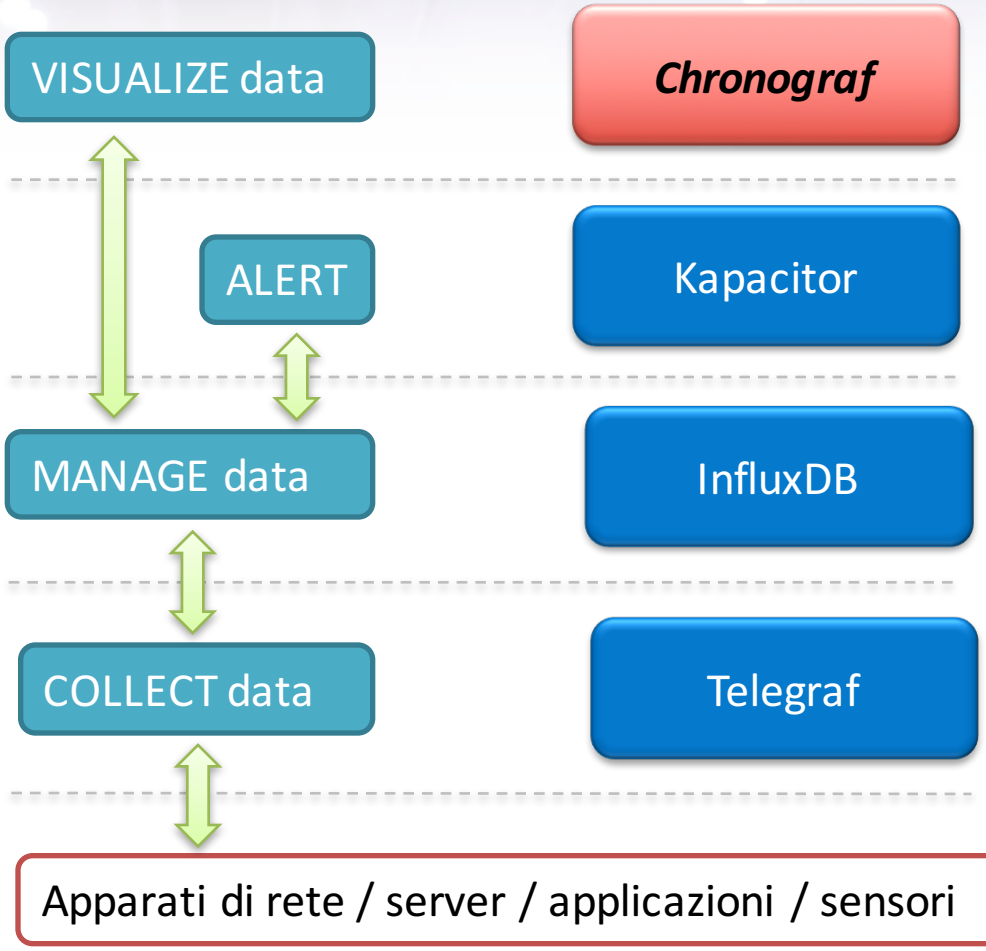
- restart

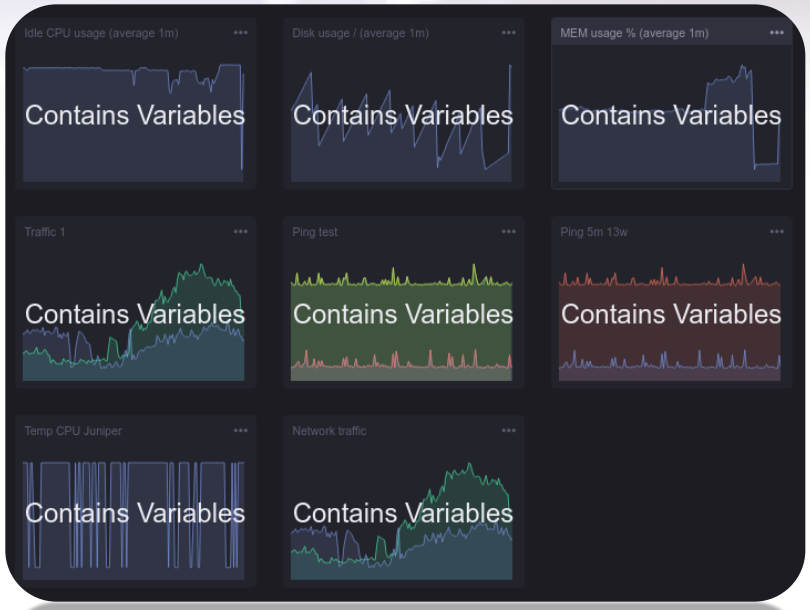
```
$ sudo service telegraf restart
telegraf process was stopped [ OK ]
Starting the process telegraf [ OK ]
telegraf process was started [ OK ]
```

Cosa facciamo ora?

1. Come funziona Telegraf
2. Acquisizione alcune metriche di un host
    1. ***Chronograf***
    2. ***Grafana***
3. Misure di latenza verso 2 router
4. SNMP input plugin
5. Acquisizione delle risorse CPU/MEM di piu' di un router Juniper
6. Acquisizione traffico delle interfacce di un router

Chronograf



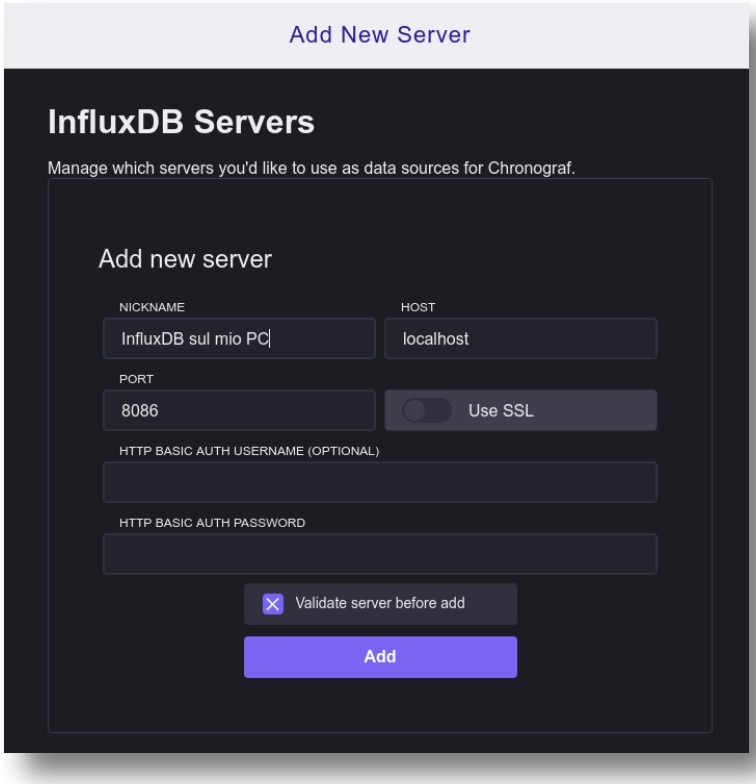


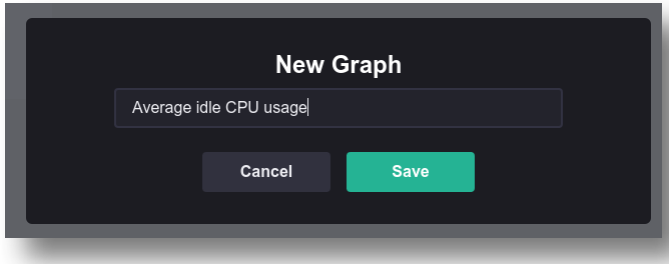
- **Embrione di WEB UI**
  - **Visualization = graph**
  - **Dashboard = graph collection**

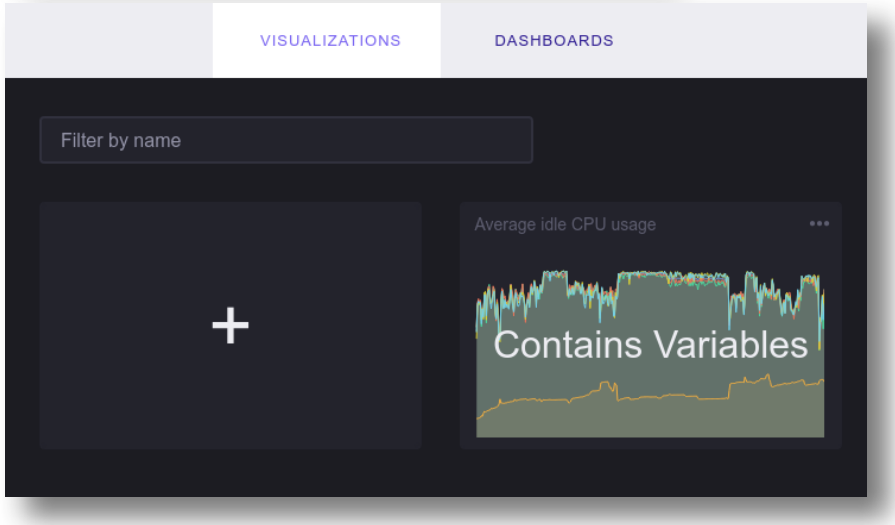
Chronograf: start

```
$ wget https://s3.amazonaws.com/get.influxdb.org/chronograf/chronograf_0.11.0_amd64.deb
$ sudo dpkg -i chronograf_0.11.0_amd64.deb
$ sudo service chronograf start
```

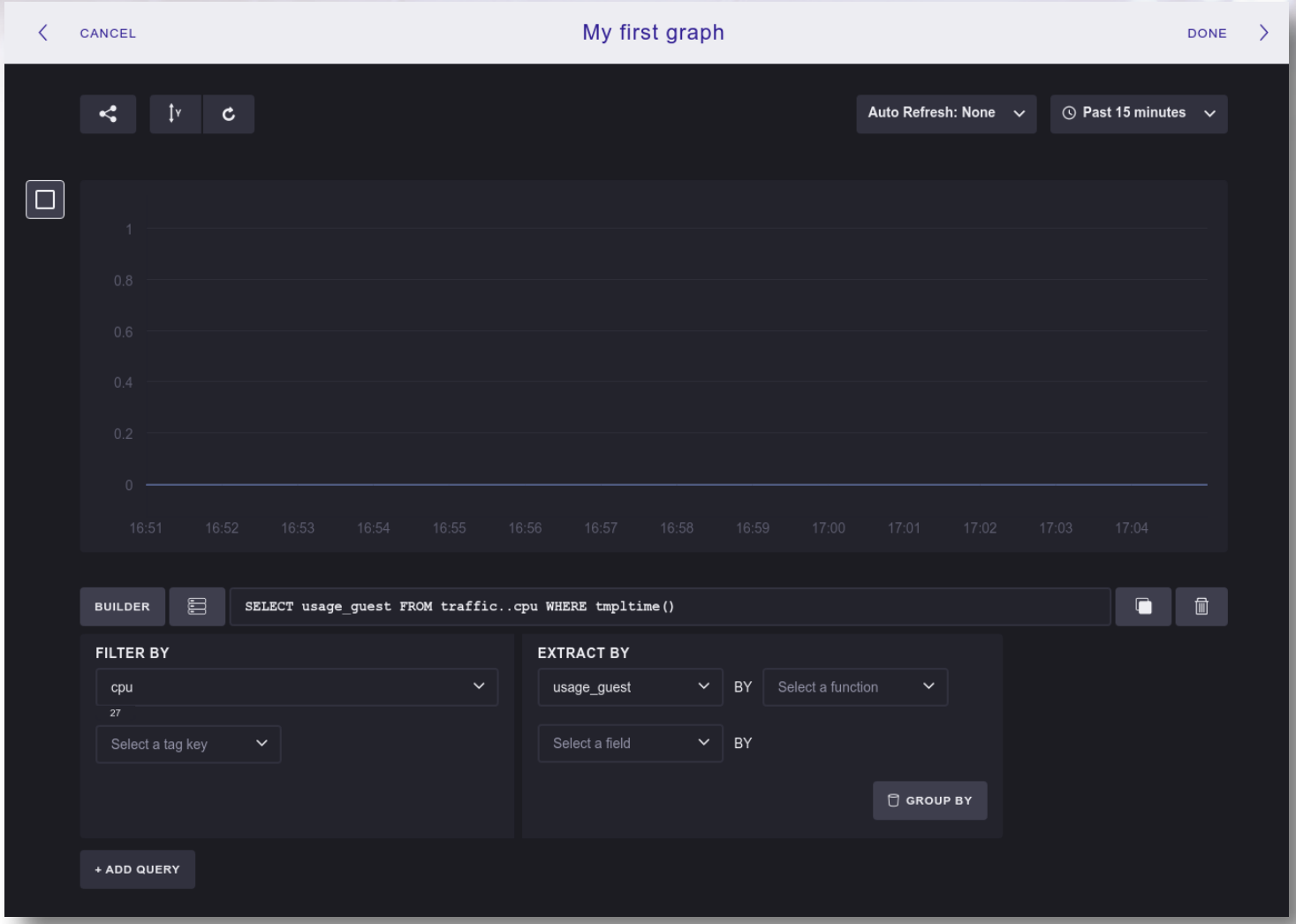
http://127.0.0.1:10000/settings/servers



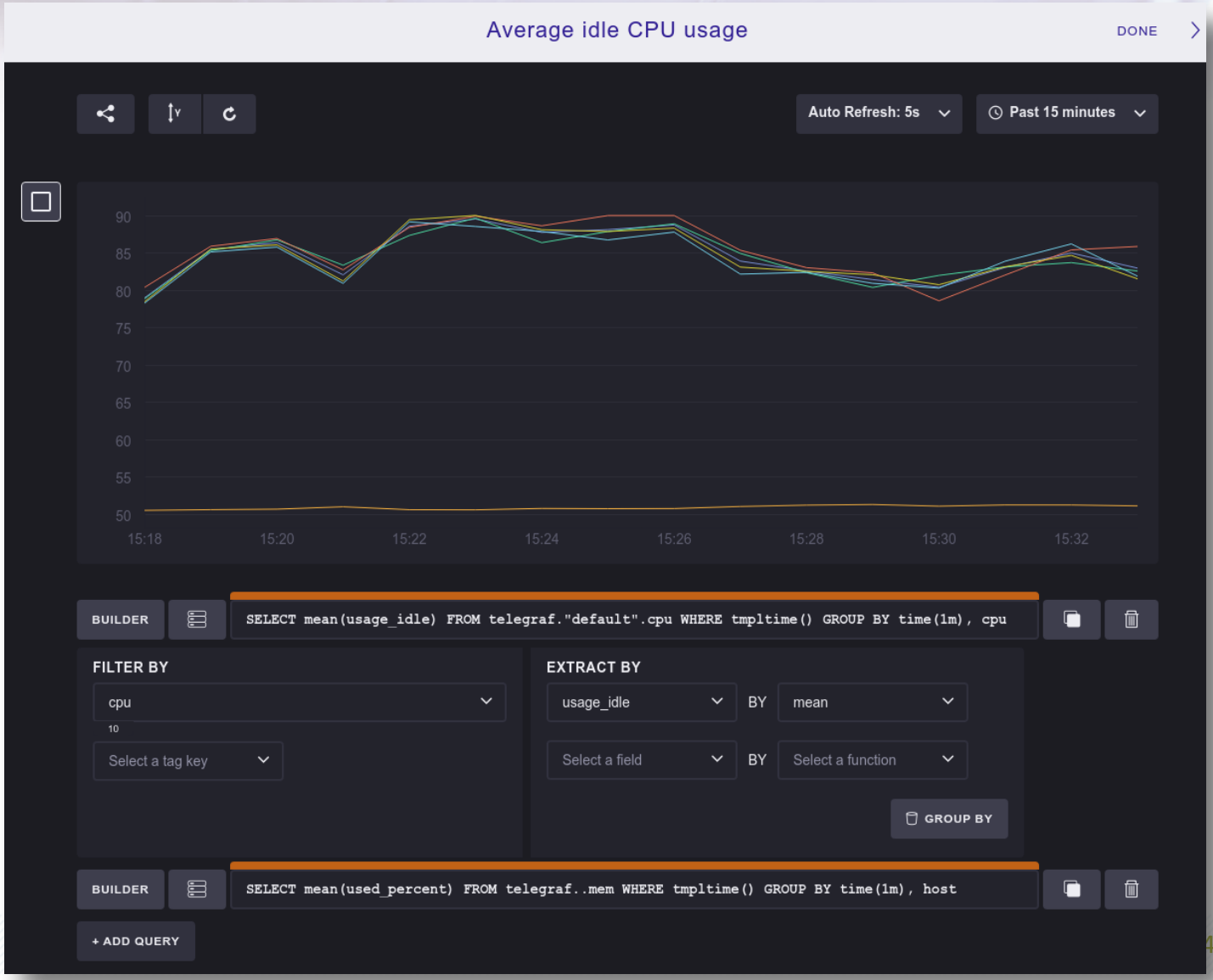




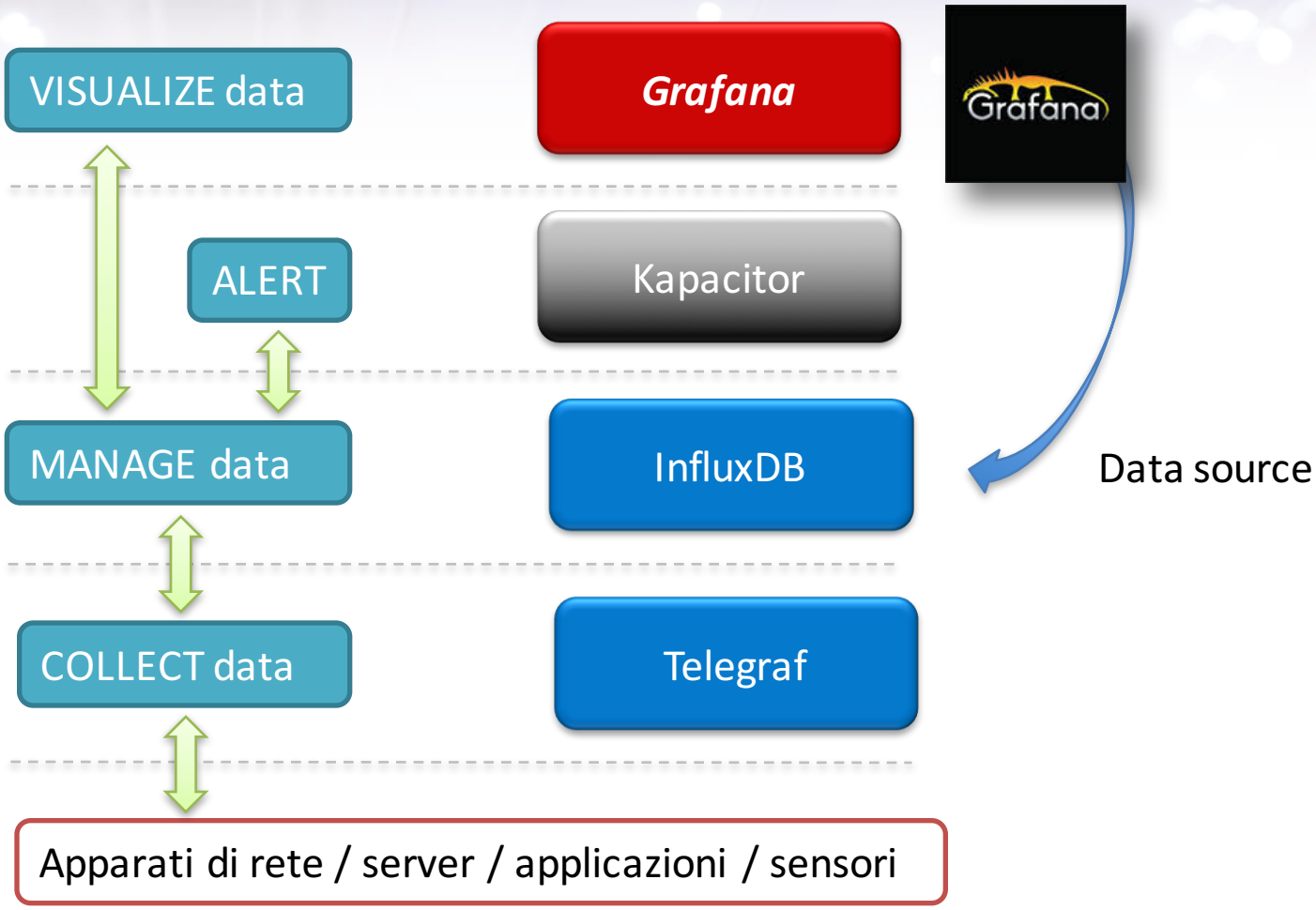
Chronograf: visualization



Chronograf: visualization



Grafana


VISUALIZE data
Grafana
Grafana
ALERT
Kapacitor
MANAGE data
InfluxDB
Data source
COLLECT data
Telegraf
Apparati di rete / server / applicazioni / sensori

Grafana

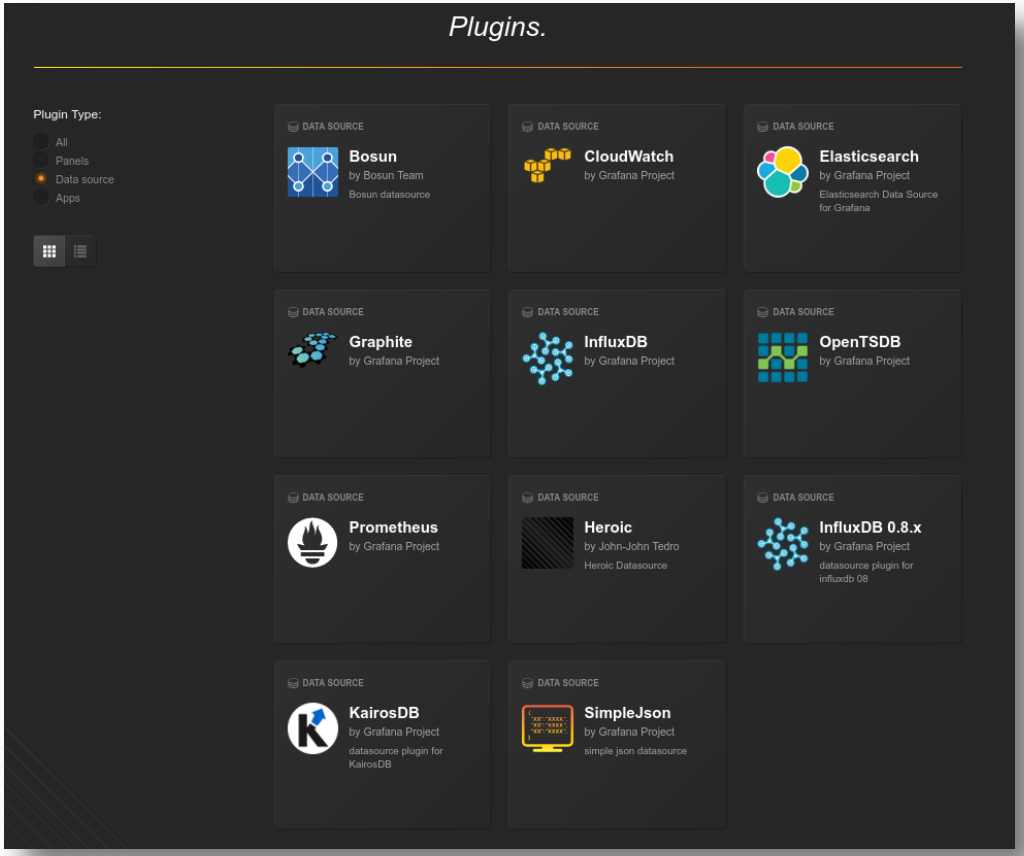
Grafana is a
"…graph and dashboard builder for visualizing time series metrics."

It makes it easy to create dashboards for displaying time-series data.

It works with several different data sources such as Graphite, Elasticsearch, InfluxDB, and OpenTSDB.

Grafana

Grafana 3.0 beta now available for download



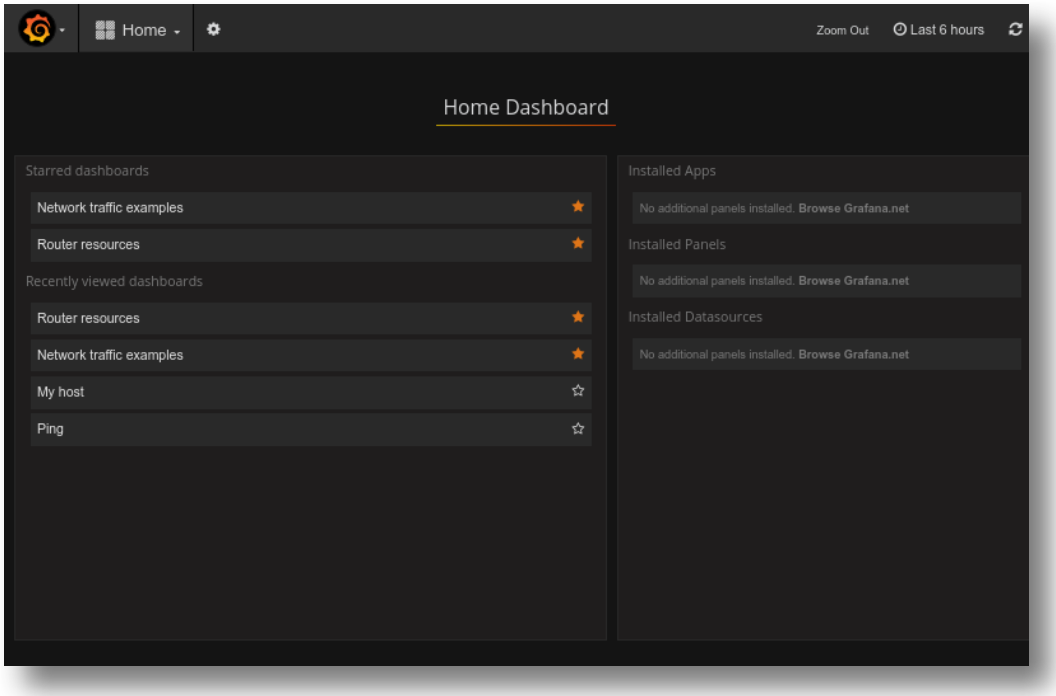
Grafana

Grafana v3.0.0-beta4 (2016-04-13)

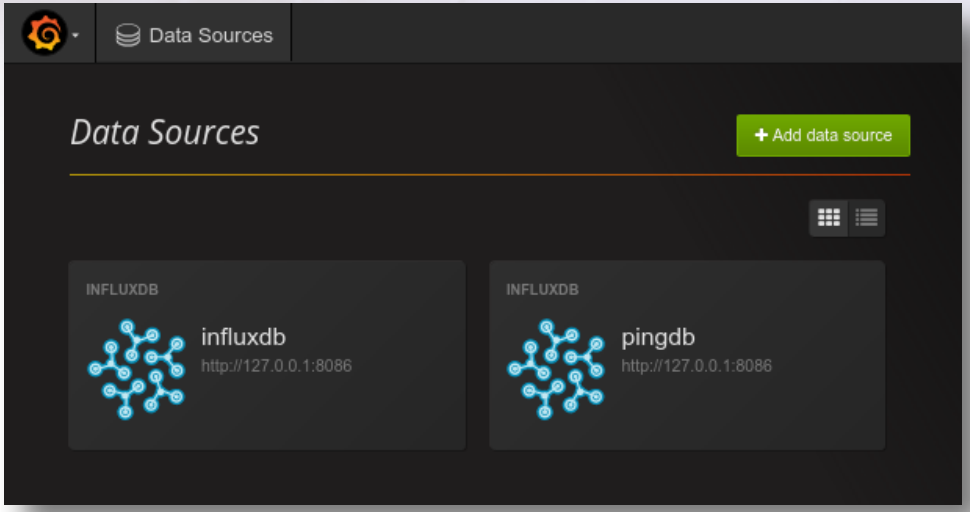
1. http://grafana.org/download/
2. Download and install

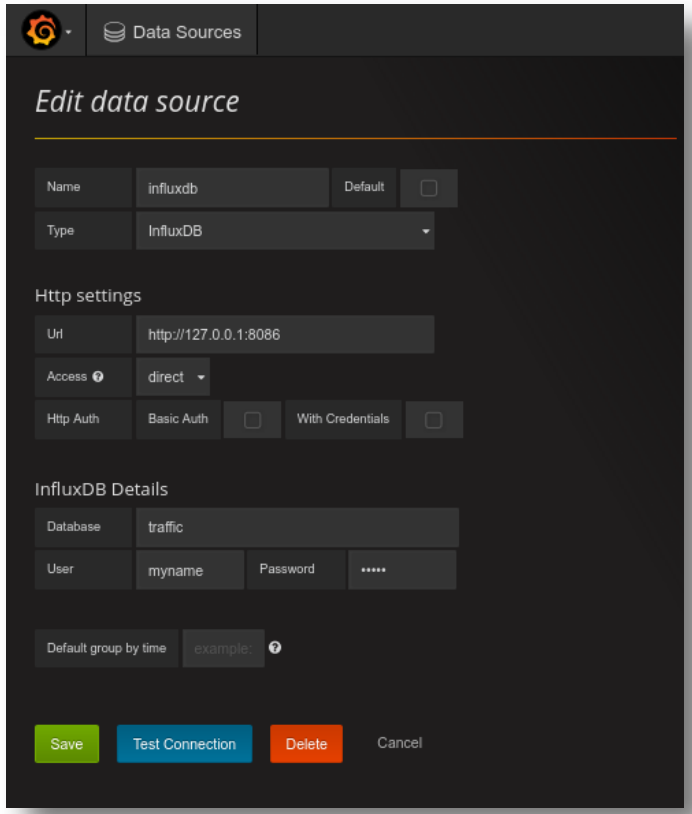
$ sudo service grafana-server start

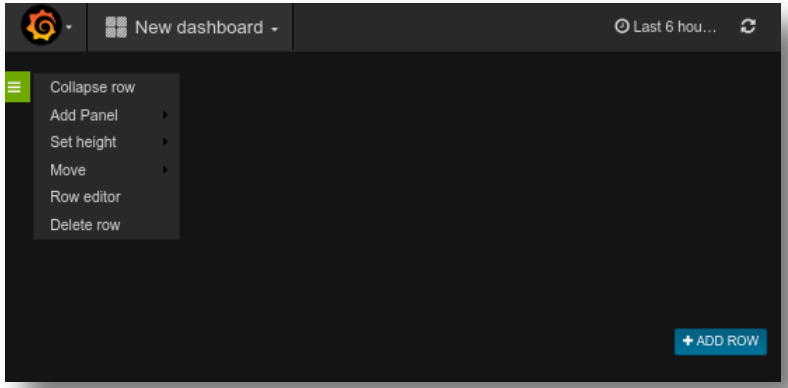
http://localhost:**3000**/



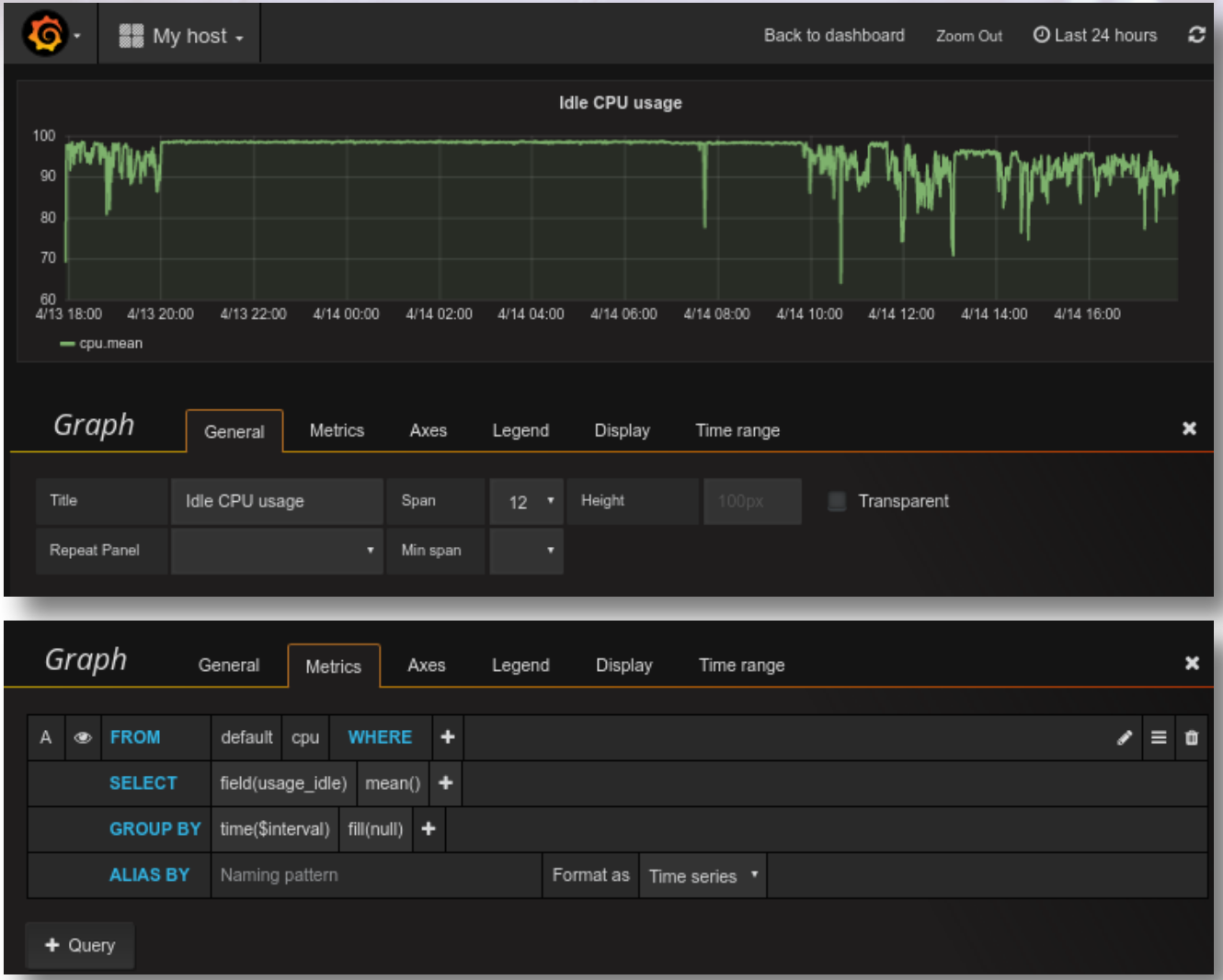
Grafana: data source





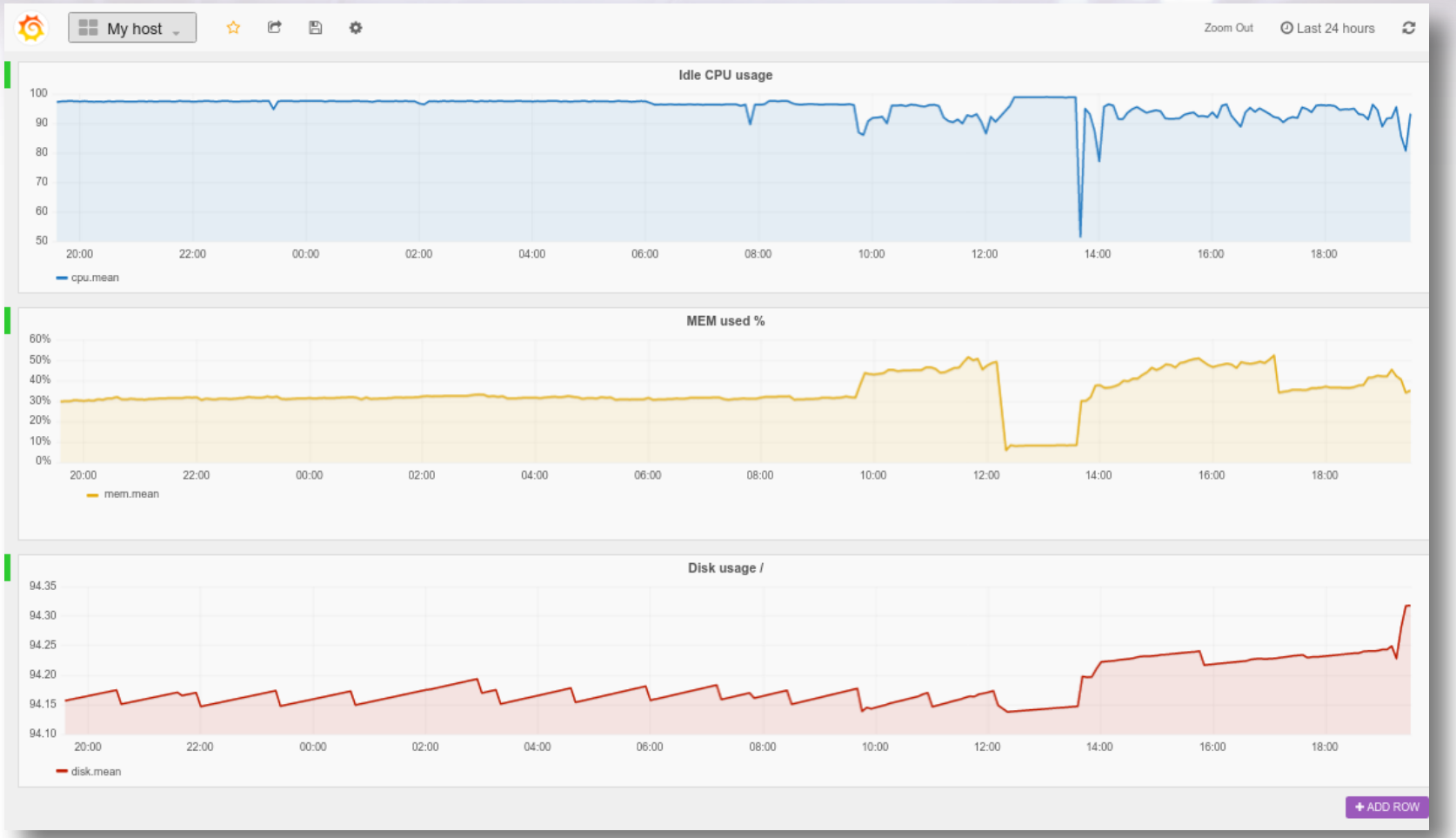


Grafana: graph



Grafana: dashboard

Grafana: dashboard



Cosa facciamo ora?

1. Come funziona Telegraf

2. Acquisizione alcune metriche di un host
   1. Chronograf
   2. Grafana

3. ***Misure di latenza verso 2 router***

4. SNMP input plugin

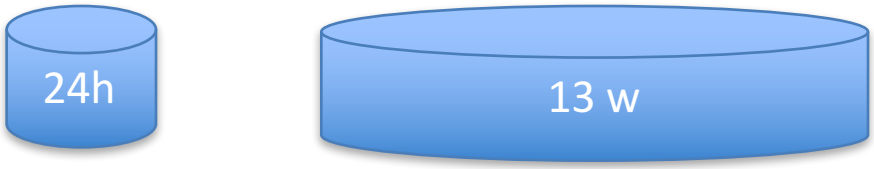
5. Acquisizione delle risorse CPU/MEM di piu' di un router Juniper

6. Acquisizione traffico delle interfacce di un router

Ping system: database

1- Creo il database su InfluxDB e le retention policies

```
> CREATE DATABASE pingdb
> CREATE RETENTION POLICY one_day ON pingdb DURATION 1d REPLICATION 1 DEFAULT
> CREATE RETENTION POLICY three_months ON pingdb DURATION 13w REPLICATION 1
```



Definisce la lunghezza temporale, non la risoluzione

Ping system: Telegraf ping plugin

2- Creo la configurazione di Telegraf

Telegraf: ping input-plugin

```
$ cd /etc/telegraf/telegraf.d/
$ telegraf -sample-config -input-filter ping -output-filter influxdb > telegraf_ping.conf

$ vi telegraf_ping.conf
```

```
[agent]
  interval = "10s"
  flush_interval = "20s"

[[outputs.influxdb]]
  urls = ["http://localhost:8086"]
  database = "pingdb"
  retention_policy = "one_day"
```

```
[[inputs.ping]]
  urls = [
  "host 1",
  "host 2",
   "host n"
  ]

  count = 1                ## ping -c <COUNT>
  ping_interval = 0.0      ## ping -i <PING_INTERVAL>
  timeout = 0.0            ## ping -t <TIMEOUT>
  interface = ""           ## ping -I <INTERFACE>
```

Ping system: ping test

3- Test della configurazione di Telegraf

```
$ telegraf -config telegraf_ping.conf -test
* Plugin: ping, Collection 1
> ping,url=host1 average_response_ms=6.487,packets_received=1i,packets_transmitted=1i,percent_packet_loss=0
1459326997105326804
> ping,url=host2 average_response_ms=13.516,packets_received=1i,packets_transmitted=1i,percent_packet_loss=0
1459326997112279558
```

Line protocol: measurement,tagset fielset timestamp

```
ping,url=host1
average_response_ms=6.487,packets_received=1i,packets_transmitted=1i,percent_packet_loss=0
1459326997105326804
```

Ping system: ping run

4- Telegraf run

```
$ telegraf -config telegraf_ping.conf

2016/03/30 10:50:10 Starting Telegraf (version 0.11.1)
2016/03/30 10:50:10 Loaded outputs: influxdb
2016/03/30 10:50:10 Loaded inputs: ping
2016/03/30 10:50:10 Tags enabled: host=pcgarr9
2016/03/30 10:50:10 Agent Config: Interval:10s, Debug:false, Quiet:false,
Hostname:"pcgarr9", Flush Interval:10s
2016/03/30 10:50:20 Gathered metrics, (10s interval), from 1 inputs in 21.353768ms
2016/03/30 10:50:30 Gathered metrics, (10s interval), from 1 inputs in 18.718892ms
2016/03/30 10:50:30 Wrote 4 metrics to output influxdb in 204.189576ms
2016/03/30 10:50:40 Gathered metrics, (10s interval), from 1 inputs in 19.214861ms
2016/03/30 10:50:40 Wrote 2 metrics to output influxdb in 60.271142ms
```

Ping system: ping run

```
$ telegraf -config telegraf_ping.conf

2016/04/15 10:48:04 Starting Telegraf (version 0.12.0)
2016/04/15 10:48:04 Loaded outputs: influxdb
2016/04/15 10:48:04 Loaded inputs: ping
2016/04/15 10:48:04 Tags enabled: host=pcgarr9
2016/04/15 10:48:04 Agent Config: Interval:10s, Debug:false, Quiet:false,
Hostname:"pcgarr9", Flush Interval:20s
2016/04/15 10:48:10 Gathered metrics, (10s interval), from 1 inputs in 18.263965ms
2016/04/15 10:48:20 Gathered metrics, (10s interval), from 1 inputs in 17.167947ms
2016/04/15 10:48:30 Gathered metrics, (10s interval), from 1 inputs in 17.492848ms
2016/04/15 10:48:30 Wrote 6 metrics to output influxdb in 106.649905ms
2016/04/15 10:48:40 Gathered metrics, (10s interval), from 1 inputs in 17.476315ms
2016/04/15 10:48:50 Gathered metrics, (10s interval), from 1 inputs in 16.750948ms
2016/04/15 10:48:50 Wrote 4 metrics to output influxdb in 197.693432ms
```

Ping system: Influxdb

InfluxDB: schema exploration

> **USE pingdb**

> **SHOW MEASUREMENTS**

name: measurements

---

name

ping

Line protocol: measurement,tagset fielset timestamp

> **SHOW SERIES**

key

ping,host=pcgarr9,url=rc-av-ru-ipseoa-rdoria-av.av.garr.net

ping,host=pcgarr9,url=rc-ba1-re-marcopolo-ba.ba1.garr.net

Ping system: Influxdb

InfluxDB: schema exploration

> SHOW TAG KEYS FROM ping

name: ping

---

tagKey

host

url

> SHOW FIELD KEYS

name: ping

---

fieldKey

average_response_ms

packets_received

packets_transmitted

percent_packet_loss

Line protocol: measurement,tagset fielset timestamp

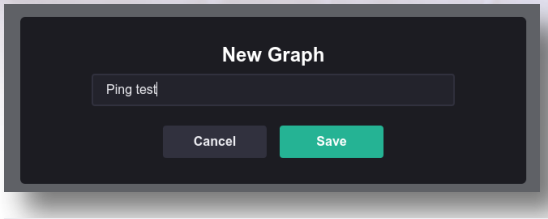
Ping system: Influxdb

InfluxDB: data exploration

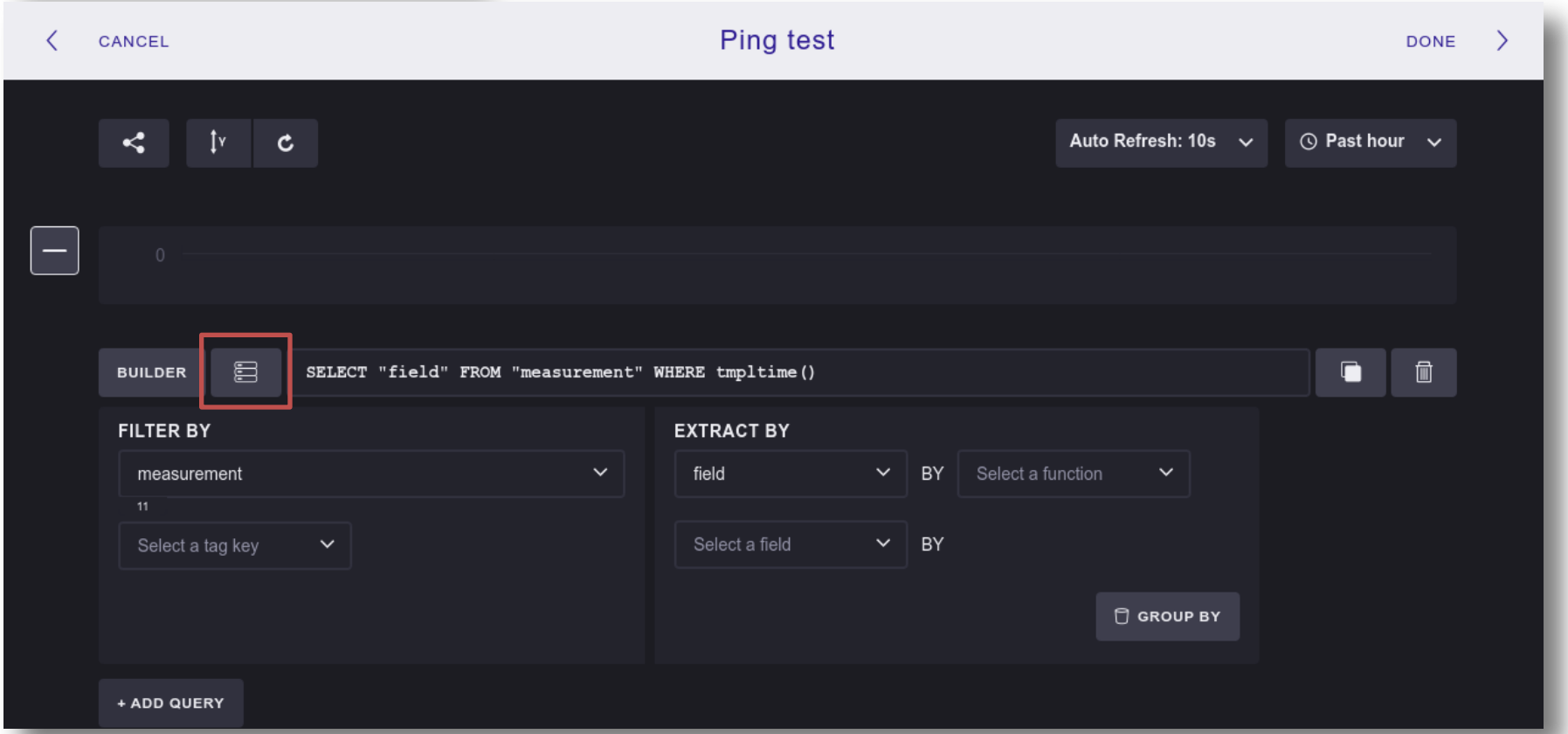
```
> SELECT * FROM ping LIMIT 5

name: ping
----------
```

time	average_response_ms	host	packets_received	packets_transmitted	percent_packet_loss	url
1459327820000000000	6.532	pcgarr9	1	1	0	rc-av-ru-ipseoa-rdoria-av.av.garr.net
1459327820000000000	13.447	pcgarr9	1	1	0	rc-ba1-re-marcopolo-ba.ba1.garr.net
1459327830000000000	6.516	pcgarr9	1	1	0	rc-av-ru-ipseoa-rdoria-av.av.garr.net
1459327830000000000	13.554	pcgarr9	1	1	0	rc-ba1-re-marcopolo-ba.ba1.garr.net
1459327840000000000	*6.508*	pcgarr9	1	1	0	rc-av-ru-ipseoa-rdoria-av.av.garr.net
1459327840000000000	13.589	pcgarr9	1	1	0	rc-ba1-re-marcopolo-ba.ba1.garr.net
1459327850000000000	6.728	pcgarr9	1	1	0	rc-av-ru-ipseoa-rdoria-av.av.garr.net
1459327850000000000	13.485	pcgarr9	1	1	0	rc-ba1-re-marcopolo-ba.ba1.garr.net
1459327860000000000	6.427	pcgarr9	1	1	0	rc-av-ru-ipseoa-rdoria-av.av.garr.net
1459327860000000000	13.427	pcgarr9	1	1	0	rc-ba1-re-marcopolo-ba.ba1.garr.net

Ping system: Chronograf

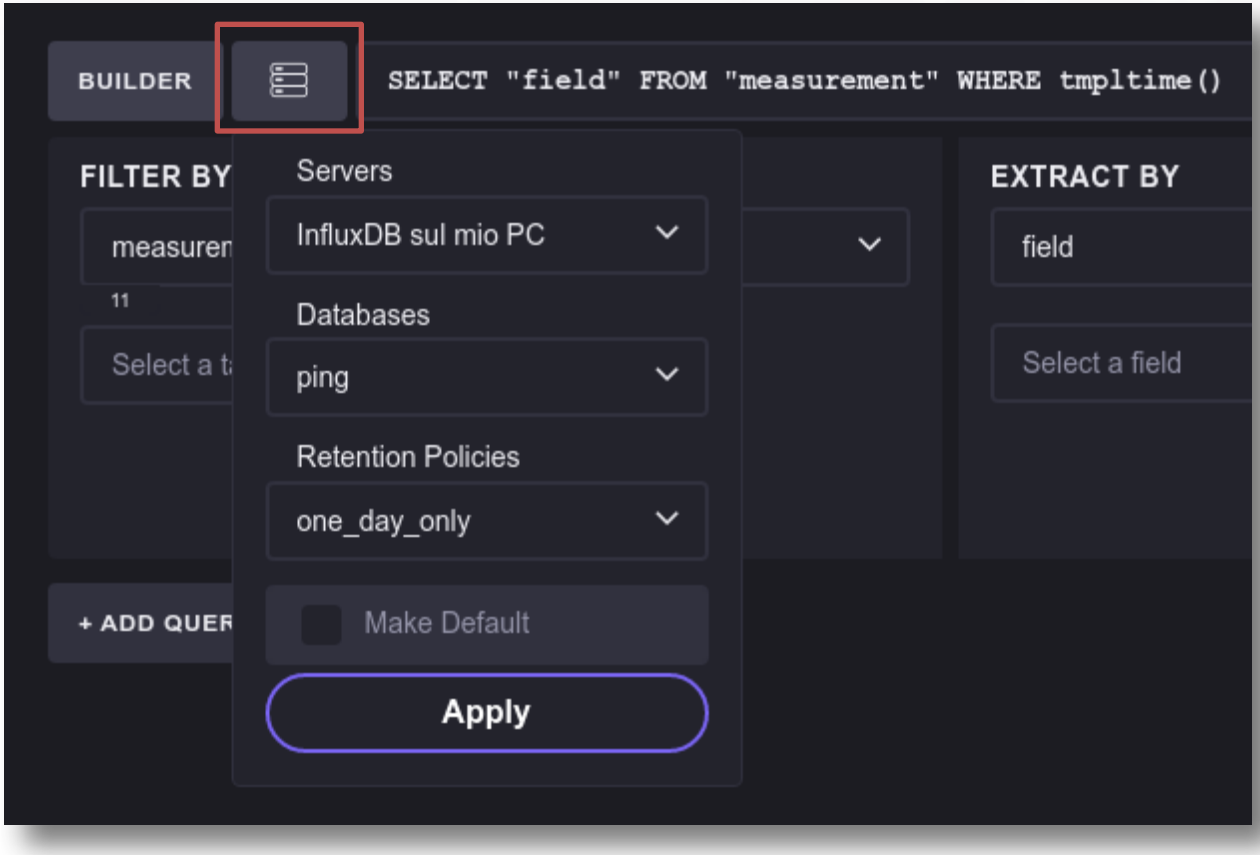


Create Visualization



Ping system: Chronograf

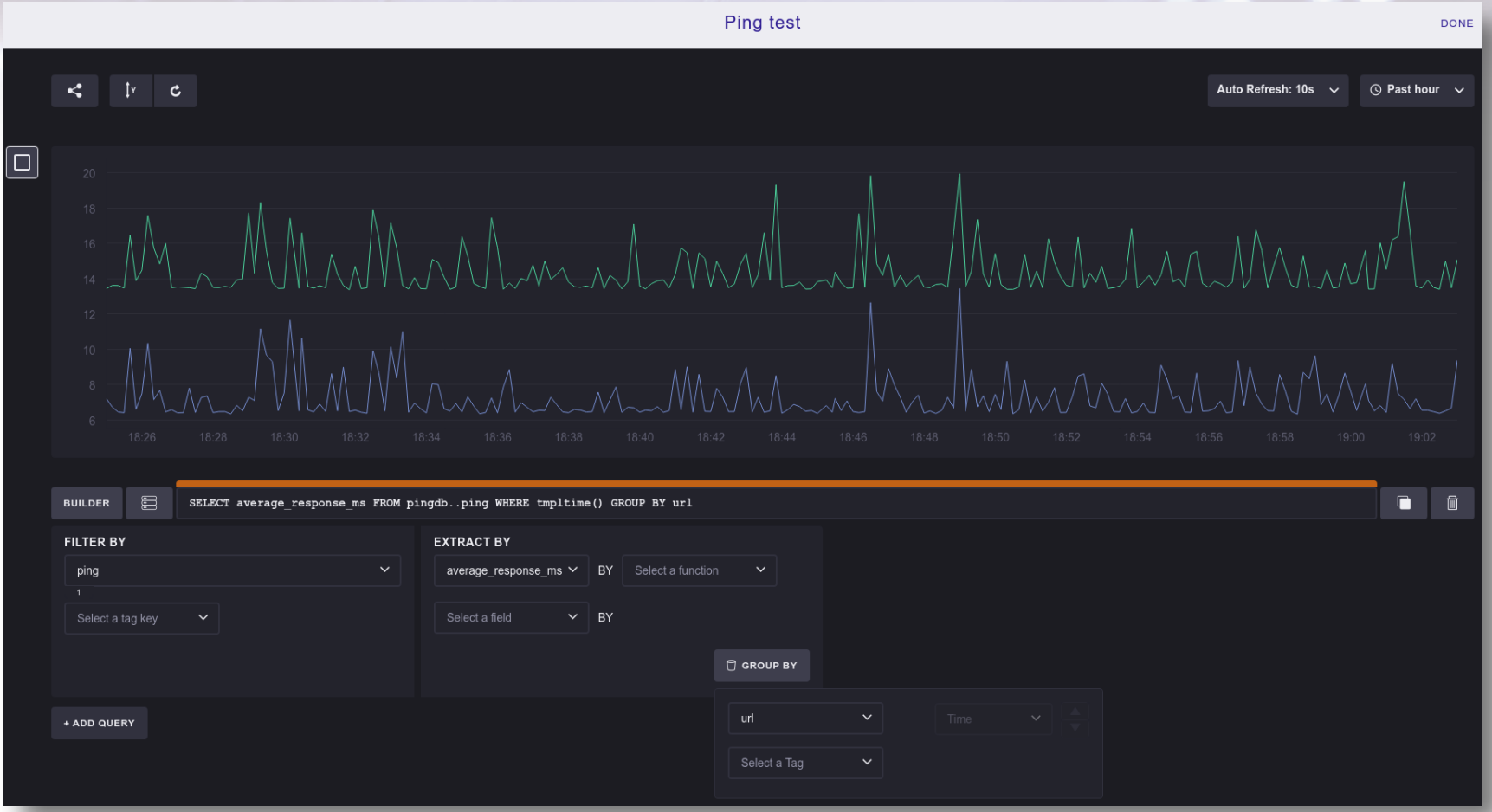
Edit Visualization



Ping system: Chronograf

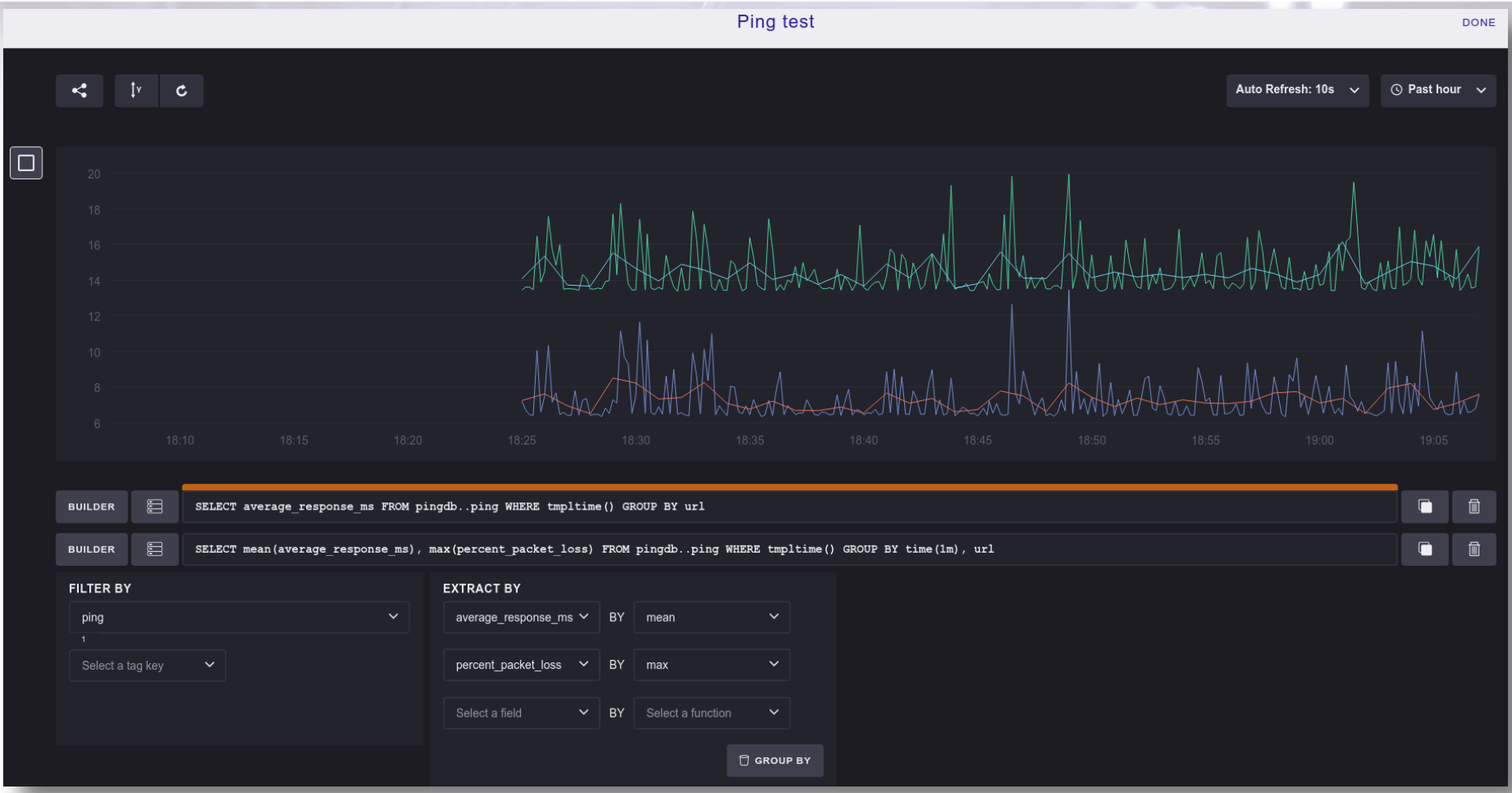
Singola query per un target (url)

Ping system: Chronograf



Singola query per tutti i target (url) con group by

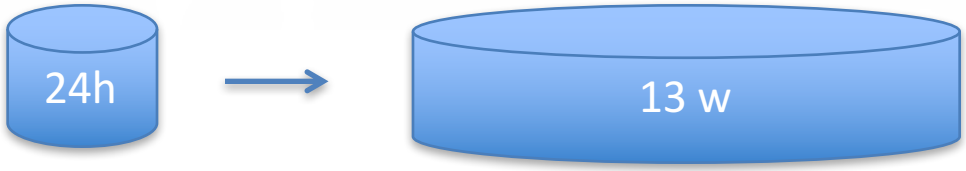
Ping system: Chronograf



3 metriche: RTT, RTT medio su 1m, LOSS % massimo su 1m

Ping system: downsampling & CQ

```
> CREATE RETENTION POLICY one_day ON pingdb DURATION 1d REPLICATION 1 DEFAULT
> CREATE RETENTION POLICY three_months ON pingdb DURATION 13w REPLICATION 1
```



Resolution:	***10 s***	→	***5 m***
Measurement:	ping	→	ping_5m

Creo la ***continuous query*** (CQ) su Influxdb
che popola la nuova measurement nella RP a 13w

mean(average_response_ms) → **mean_ms**

max(percent_packet_loss) → **max_loss**

Ping system: downsampling & CQ

```
> SELECT mean(average_response_ms) AS mean_ms,
          max(percent_packet_loss) AS max_loss
   FROM ping
   WHERE time > now() - 5m
   GROUP BY url,time(5m)

name: ping
tags: url=host1
time                 mean_ms           max_loss
----                 -------           --------
1459341300000000000  6.647312499999999 0
1459341600000000000  6.653428571428571 0


name: ping
tags: url=host2
time                 mean_ms           max_loss
----                 -------           --------
1459341300000000000  13.586562500000001 0
1459341600000000000  13.670714285714284 0
```

Definiamo la query che useremo per il downsampling

Intervalli 10 s => 5 m

Durata 24 h => 1 w

Ping system: downsampling & CQ

```
SELECT mean(average_response_ms) AS mean_ms
FROM ping
WHERE time > now() - 5m
GROUP BY url,time(5m)
```

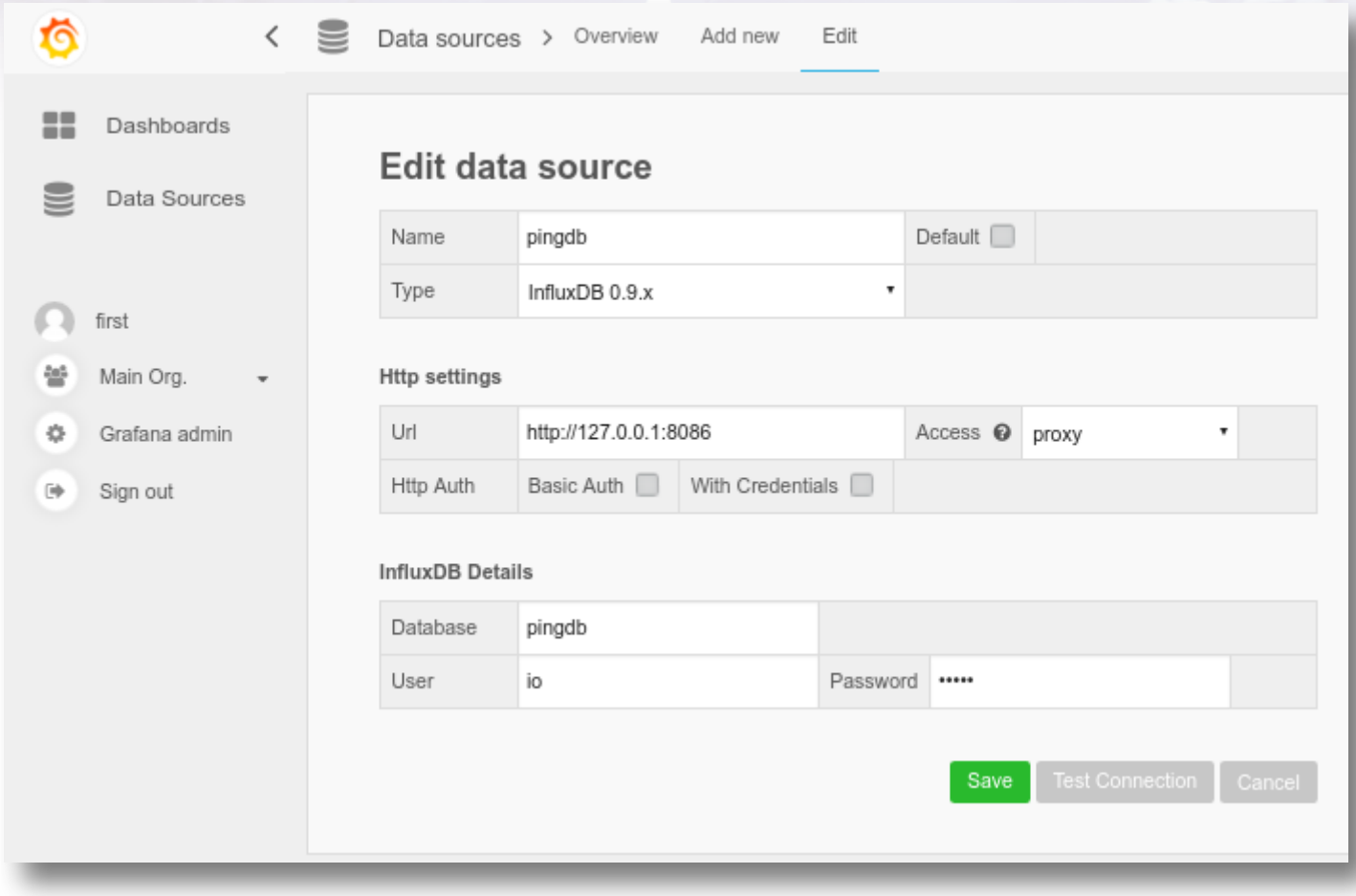
Definiamo la CONTINUOUS QUERY

```
CREATE CONTINUOUS QUERY cq_ping_5m ON pingdb BEGIN
SELECT
      mean(average_response_ms) AS mean_ms,
      max(percent_packet_loss) AS max_loss
INTO pingdb."three_months"."ping_5m"
FROM ping GROUP BY url,time(5m)
END
```

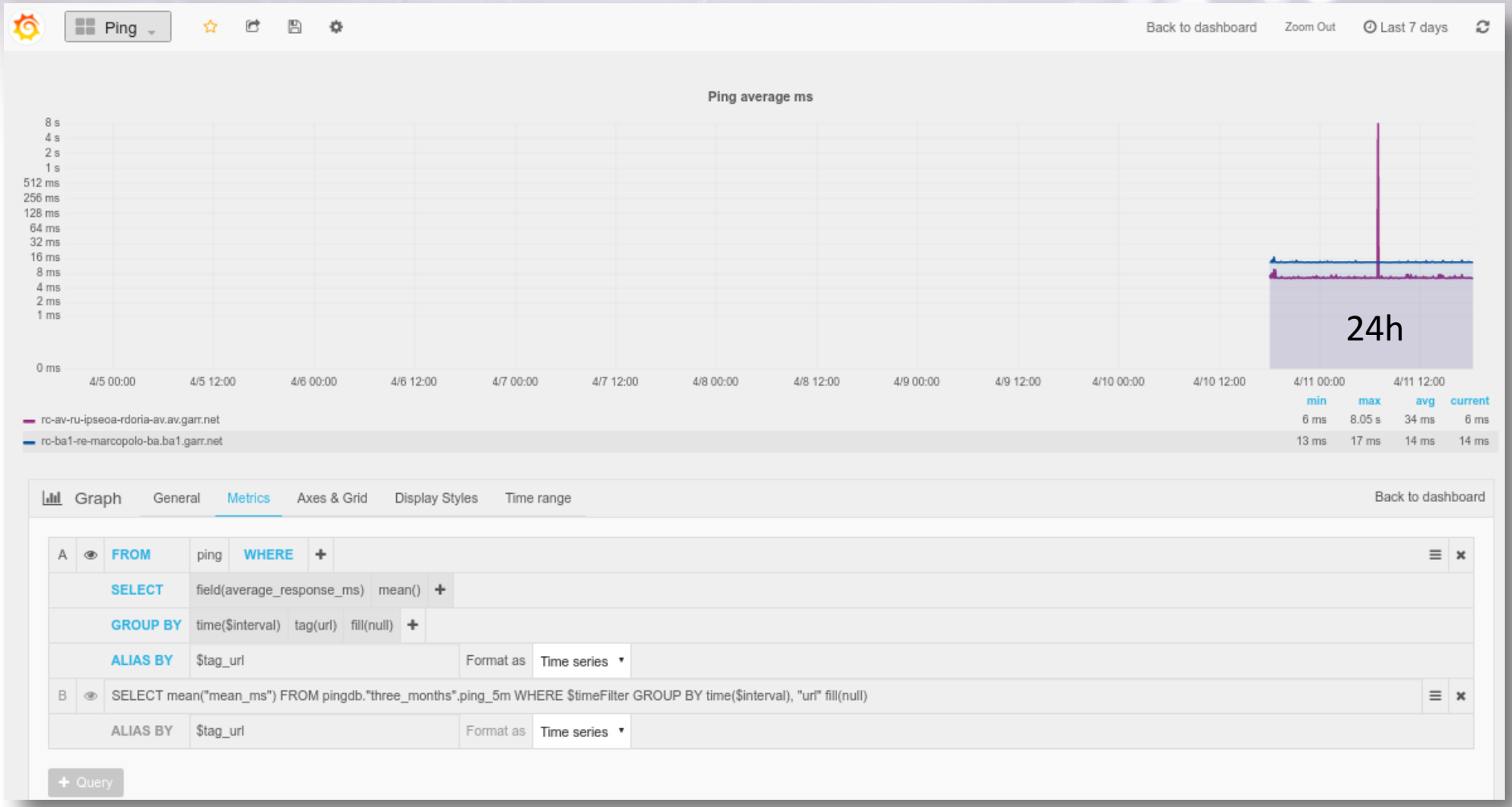
Intervalli: 10 s => 5 m
Durata: 24 h => 13 w
Measure: ping => ping_5m
RP: one_day => three_months

```
> select * from three_months.ping_5m
name: ping_5m
------------
time                  max_loss  mean_ms            url
1459515900000000000   0         6.831433333333336  host1
1459515900000000000   0         13.76773333333333  host2
```

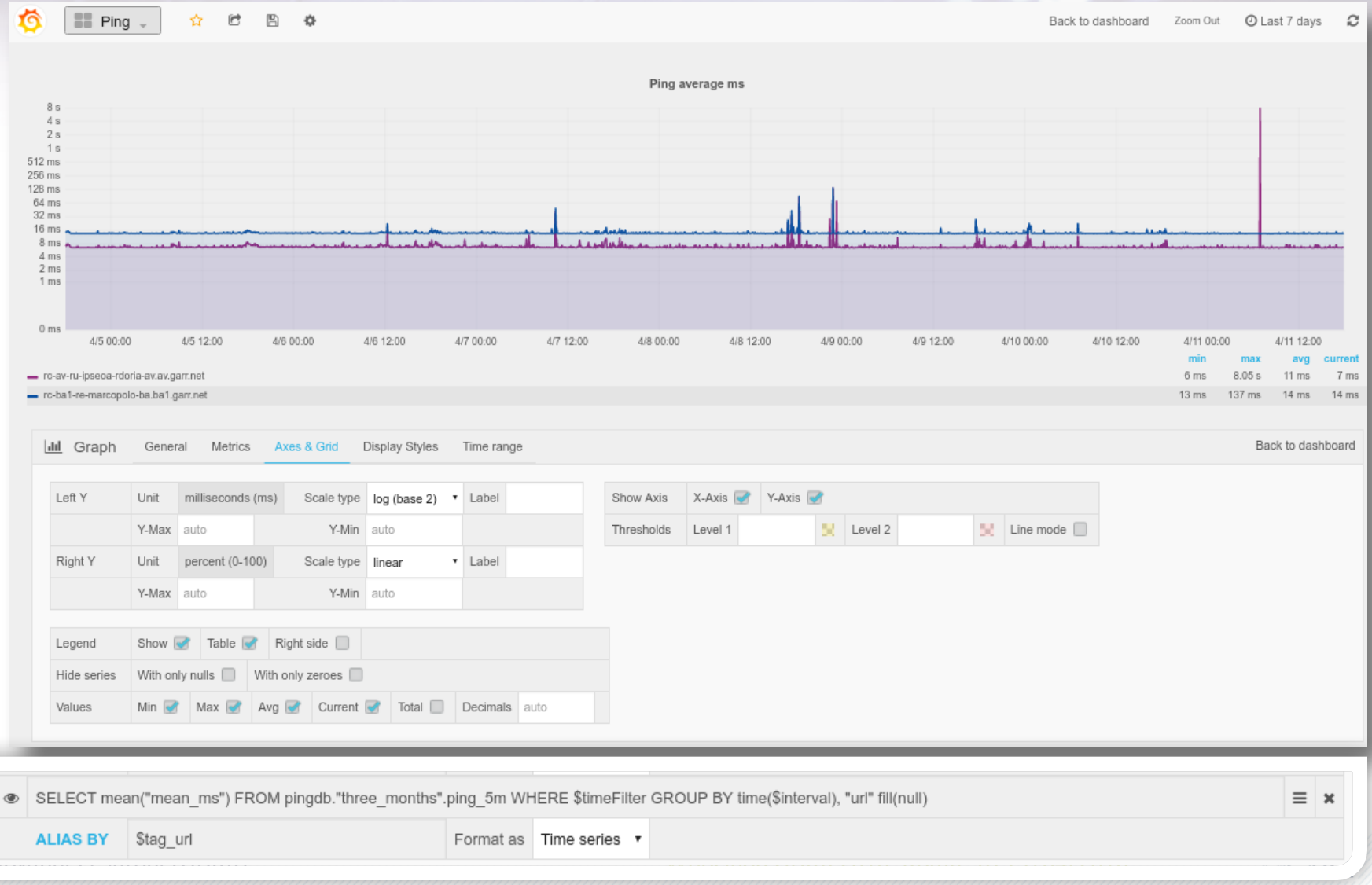
Ping system: Grafana



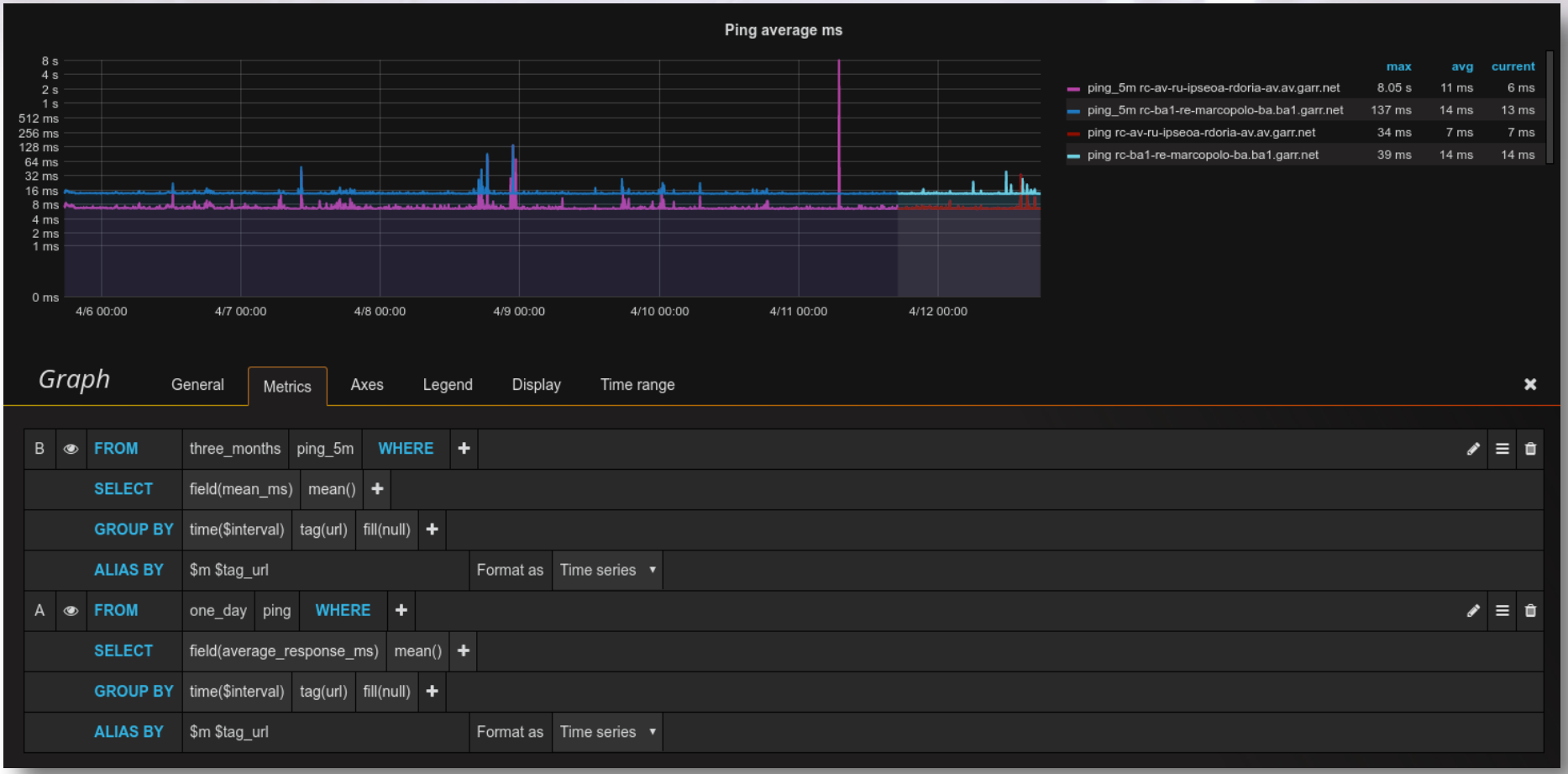
Ping system: Grafana



Ping system: Grafana



Ping system: Grafana



Cosa facciamo ora?

1. Come funziona Telegraf

2. Acquisizione alcune metriche di un host
   1. Chronograf
   2. Grafana

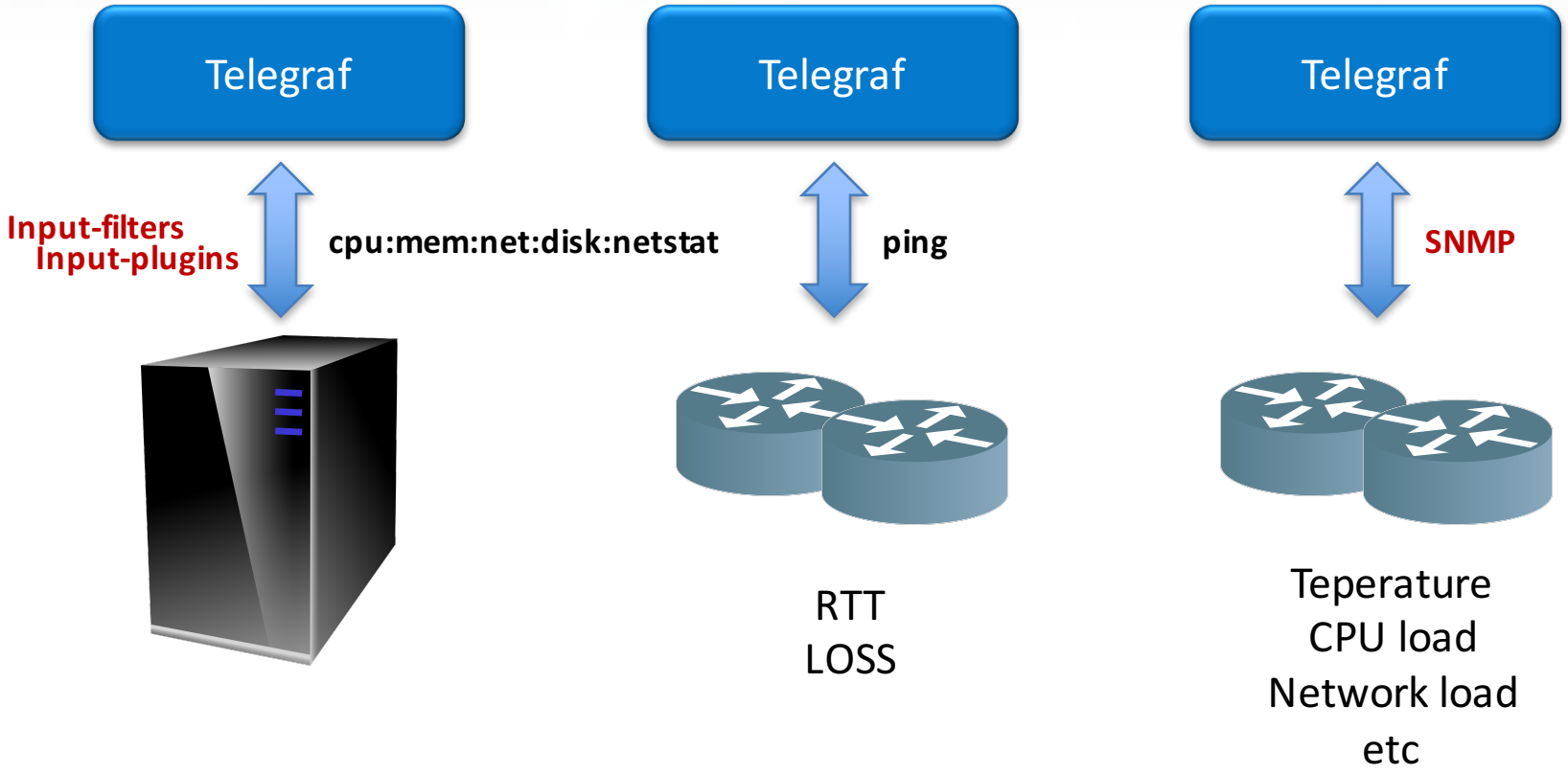
3. Misure di latenza verso 2 router

4. **SNMP input plugin**

5. Acquisizione delle risorse CPU/MEM di piu' di un router Juniper

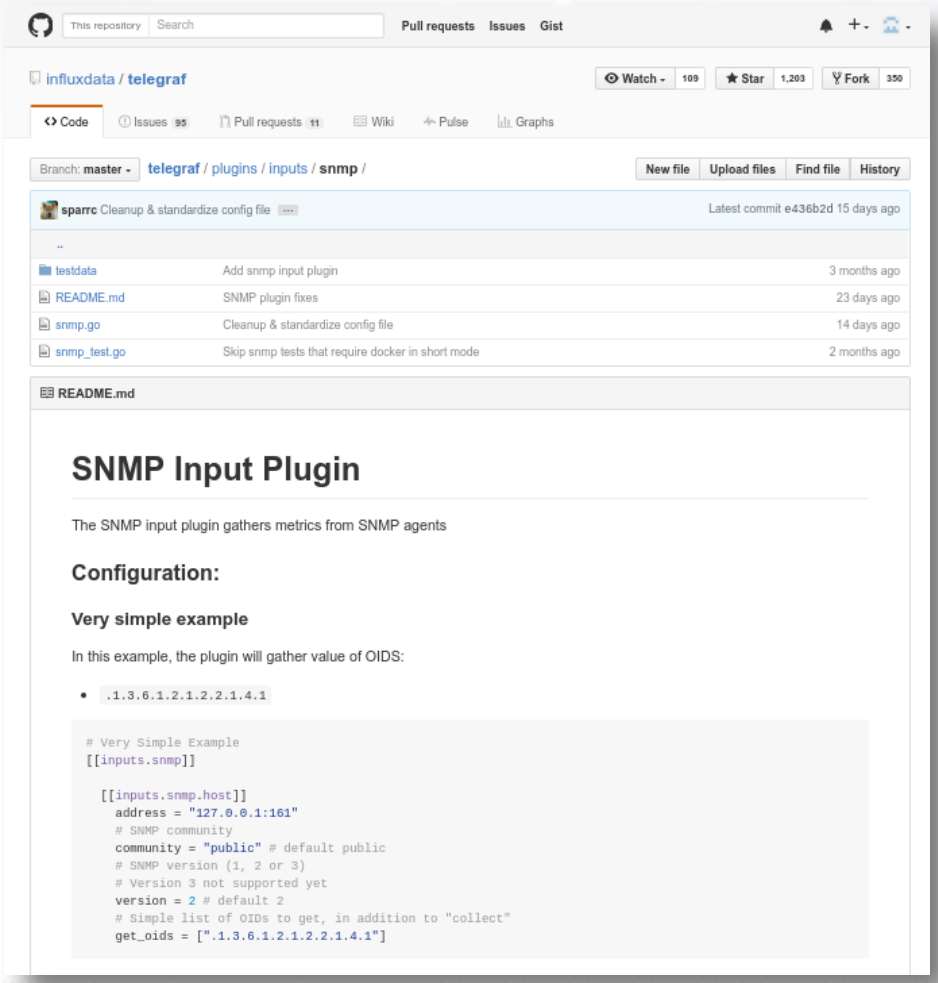
6. Acquisizione traffico delle interfacce di un router

Telegraf: input-filters



Telegraf & SNMP input plugin

https://github.com/influxdata/telegraf/tree/master/plugins/inputs/snmp



This repository Search Pull requests Issues Gist

influxdata / telegraf

Watch 109 Star 1,203 Fork 350

Code Issues 95 Pull requests 11 Wiki Pulse Graphs

Branch: master telegraf / plugins / inputs / snmp / New file Upload files Find file History

sparrc Cleanup & standardize config file Latest commit e436b2d 15 days ago

testdata	Add snmp input plugin	3 months ago
README.md	SNMP plugin fixes	23 days ago
snmp.go	Cleanup & standardize config file	14 days ago
snmp_test.go	Skip snmp tests that require docker in short mode	2 months ago

README.md

SNMP Input Plugin

The SNMP input plugin gathers metrics from SNMP agents

Configuration:

Very simple example

In this example, the plugin will gather value of OIDS:

- `.1.3.6.1.2.1.2.2.1.4.1`

```
# Very Simple Example
[[inputs.snmp]]

  [[inputs.snmp.host]]
    address = "127.0.0.1:161"
    # SNMP community
    community = "public" # default public
    # SNMP version (1, 2 or 3)
    # Version 3 not supported yet
    version = 2 # default 2
    # Simple list of OIDs to get, in addition to "collect"
    get_oids = [".1.3.6.1.2.1.2.2.1.4.1"]
```

Telegraf: snmp plugin, configurazione

Example 1

```
[[inputs.snmp]]
  [[inputs.snmp.host]]
    address = “<host>:161”
    community = "public"
    version = 2
    get_oids = [“oid1“,”oid2”]
```

host

lista OIDs

```
[[inputs.snmp]]
  snmptranslate_file = “/tmp/oids.txt”
```

```
$  snmptranslate -m all -Tz -On | sed -e 's/"//g' > /tmp/oids.txt
$  snmptranslate -M /mycustommibfolder -Tz -On -m all | sed -e 's/"//g' > oids.txt
```

Telegraf: snmp plugin, configurazione

```
[[inputs.snmp]]
  snmptranslate_file = “/tmp/oids.txt”

  [[inputs.snmp.host]]
    address = “<host-1>:161”
    collect = [“ifnumber”,  “interface_speed”, “if_out_octets”]

  [[inputs.snmp.get]]
    name = "ifnumber"
    oid = ".1.3.6.1.2.1.2.1.0"

  [[inputs.snmp.get]]
    name = "interface_speed"
    oid = "ifSpeed"
    instance = “1”

  [[inputs.snmp.bulk]]
     name = "if_out_octets"
     oid = "ifOutOctets"
```

Example 2

associazione metriche per host

Get OID

Get OID of index

Walk OID

Telegraf: snmp plugin, configurazione

Example 3:
Bulk request table by oid

```
[[inputs.snmp]]
  snmptranslate_file = "/tmp/oids.txt"
  [[inputs.snmp.host]]
    address = “<host-1>:161”

    [[inputs.snmp.host.table]]
       name = "iftable1"

    [[inputs.snmp.table]]
         name = "iftable1"
         oid = “<oid>”
```

Example 4:
Bulk request table with subtables

```
[[inputs.snmp]]
    [[inputs.snmp.host]]
    address = " <host-1>: 161"
    [[inputs.snmp.host.table]]
      name = "iftable2"

  [[inputs.snmp.table]]
    name = "iftable2"
    sub_tables = [“<oid-1>”,“<oid-2>”]
```

Telegraf: snmp plugin, mapping

```
$ snmptranslate -On -m ALL IF-MIB::ifHCInOctets
.1.3.6.1.2.1.31.1.1.1.6
```

```
$ snmpget -v2c -c public <host> .1.3.6.1.2.1.31.1.1.1.6.579
.1.3.6.1.2.1.31.1.1.1.6.579 = Counter64: 486962474
```

Line protocol: measurement,tagset fielset timestamp

Field: ifHCInOctets=486962474 TAG: instance=579

```
$ snmpget -v2c -c public <host> .1.3.6.1.2.1.31.1.1.1.1.579
.1.3.6.1.2.1.31.1.1.1.1.579 = STRING: "xe-0/0/0"
```

Mapping Table

Field: ifHCInOctets=486962474 TAG: instance=xe-0/0/0

Telegraf: snmp plugin, configurazione

```
[[inputs.snmp.host]]
  address = “<host-1>:161"
    [[inputs.snmp.host.table]]
      name = "iftable4"
      include_instances = [“eth0”, “eth1”]

 [[inputs.snmp.table]]
   name = "iftable4"
   mapping_table = ".1.3.6.1.2.1.31.1.1.1.1"
   sub_tables=["bytes_in", "bytes_out"]

 # SNMP SUBTABLES
 [[inputs.snmp.subtable]]
   name = "bytes_in"
   oid = ".1.3.6.1.2.1.31.1.1.1.6"
   unit = "octets"

 [[inputs.snmp.subtable]]
   name = "bytes_out"
   oid = ".1.3.6.1.2.1.31.1.1.1.10"
   unit = "octets"
```

Example 5:
Bulk request table with subtables and MAPPING

$ snmptranslate -On IF-MIB::ifName
.1.3.6.1.2.1.31.1.1.1.1

$ snmptranslate -m all .1.3.6.1.2.1.31.1.1.1.1
IF-MIB::ifName

IF-MIB::ifHCInOctets

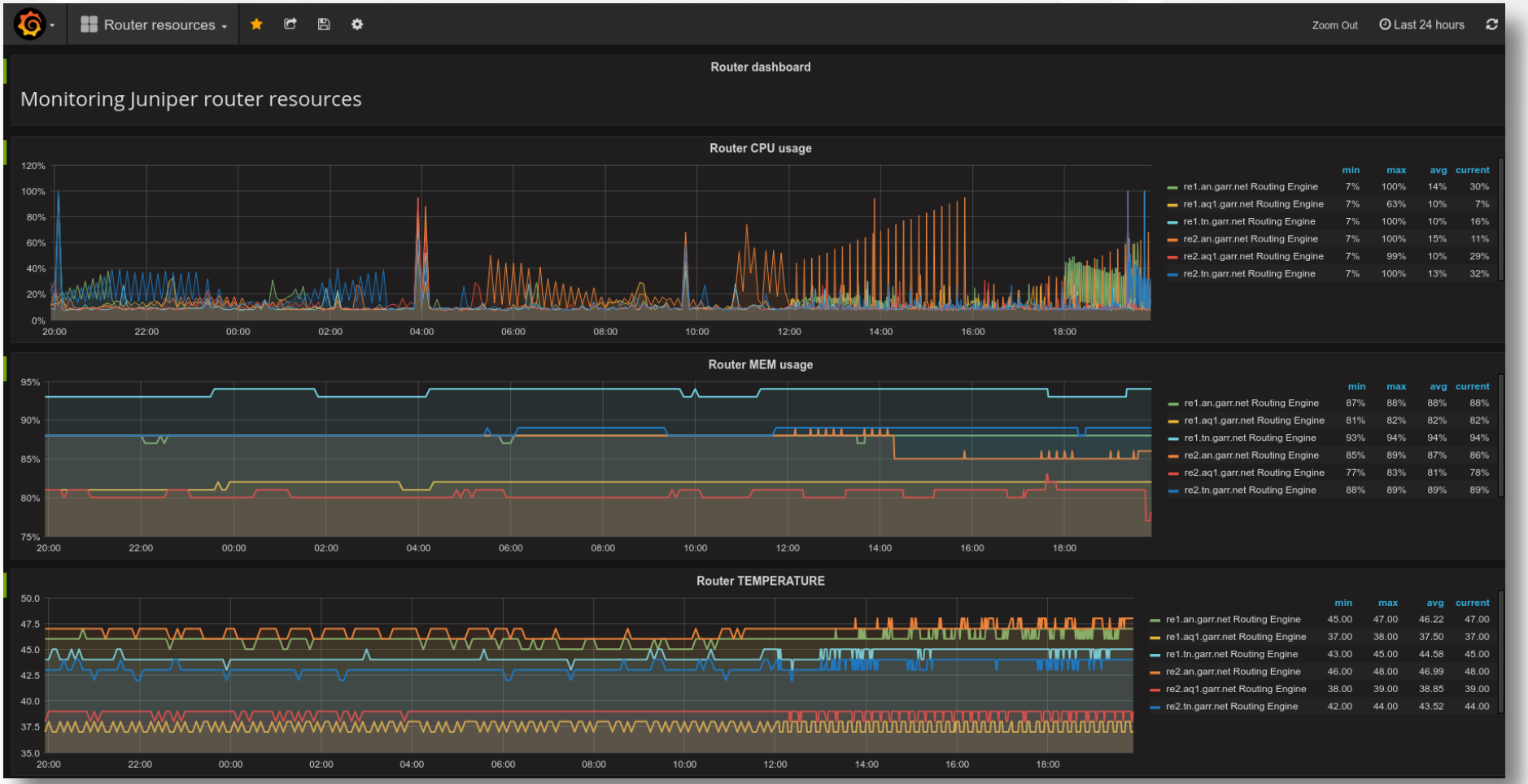
IF-MIB::ifHCOutOctets

Cosa facciamo ora?

1. Come funziona Telegraf
2. Acquisizione alcune metriche di un host
    1. Chronograf
    2. Grafana
3. Misure di latenza verso 2 router
4. SNMP input plugin
5. **Acquisizione delle risorse CPU/MEM di un router Juniper**
6. Acquisizione traffico delle interfacce di un router

Grafana: router resources

6 router, last 24h



Telegraf: snmp plugin, use case

Use case: CPU load e temperatura su Juniper

jnxOperatingTemp 1.3.6.1.4.1.2636.3.1.13.1.7
jnxOperatingCPU 1.3.6.1.4.1.2636.3.1.13.1.8

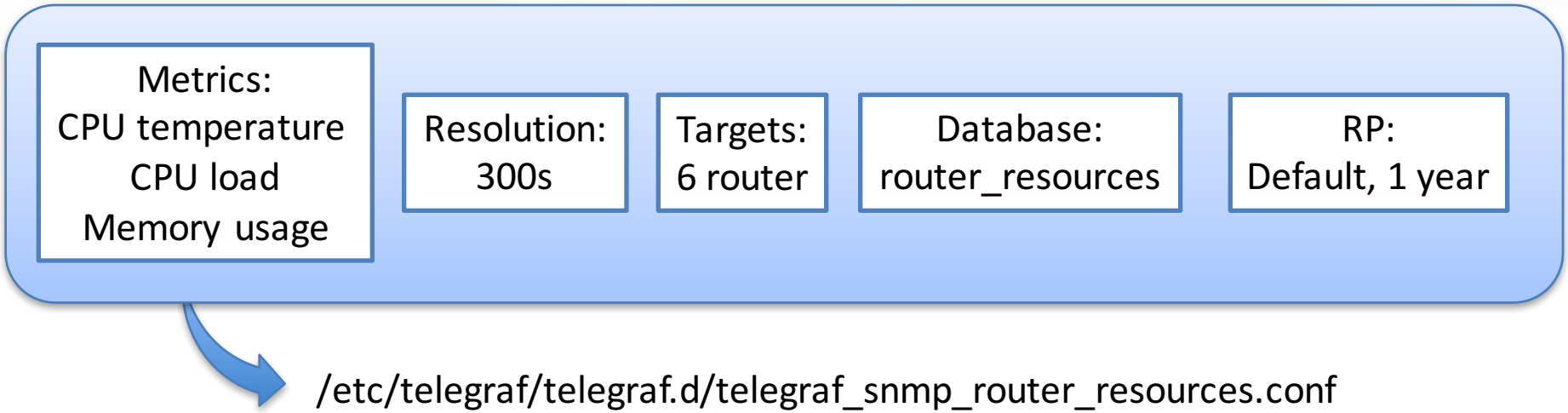
jnxOperatingTemp.<oid_of_entity>

jnxOperatingDescr 1.3.6.1.4.1.2636.3.1.13.1.5
jnxOperatingDescr.<oid_of_entity>

jnxOperatingTemp.1.1.0.0	0
jnxOperatingTemp.2.1.0.0	40
jnxOperatingTemp.2.2.0.0	45
jnxOperatingTemp.2.3.0.0	45
jnxOperatingTemp.2.4.0.0	45
jnxOperatingTemp.4.1.0.0	39
jnxOperatingTemp.4.1.1.0	39
jnxOperatingTemp.4.1.2.0	39
jnxOperatingTemp.4.1.3.0	39
jnxOperatingTemp.4.1.4.0	39
jnxOperatingTemp.4.1.5.0	39
jnxOperatingTemp.4.1.6.0	39
jnxOperatingTemp.4.1.7.0	39
jnxOperatingTemp.4.1.8.0	39
jnxOperatingTemp.4.1.9.0	39
jnxOperatingTemp.4.1.10.0	39
jnxOperatingTemp.4.1.11.0	39
jnxOperatingTemp.4.1.12.0	39
jnxOperatingTemp.4.2.0.0	30
jnxOperatingTemp.4.2.1.0	30
jnxOperatingTemp.4.2.2.0	30
jnxOperatingTemp.4.2.3.0	30
jnxOperatingTemp.4.2.4.0	30
jnxOperatingTemp.4.2.5.0	30
jnxOperatingTemp.4.2.6.0	30
jnxOperatingTemp.4.2.7.0	30
jnxOperatingTemp.4.2.8.0	30
jnxOperatingTemp.4.2.9.0	30
jnxOperatingTemp.4.2.10.0	30
jnxOperatingTemp.4.2.11.0	30
jnxOperatingTemp.4.2.12.0	30
jnxOperatingTemp.7.1.0.0	32
jnxOperatingTemp.7.2.0.0	29
jnxOperatingTemp.7.4.0.0	31
jnxOperatingTemp.7.5.0.0	31
jnxOperatingTemp.7.8.0.0	31
jnxOperatingTemp.7.9.0.0	27
jnxOperatingTemp.7.10.0.0	30
jnxOperatingTemp.7.11.0.0	31
jnxOperatingTemp.7.12.0.0	31

jnxOperatingDescr.7.1.0.0	FPC: MPC4E 3D 2CGE+8XGE @ 0/*/*
jnxOperatingDescr.7.2.0.0	FPC: MPC4E 3D 2CGE+8XGE @ 1/*/*
jnxOperatingDescr.7.4.0.0	FPC: MPCE Type 3 3D @ 3/*/*
jnxOperatingDescr.7.5.0.0	FPC: MPCE Type 3 3D @ 4/*/*
jnxOperatingDescr.7.8.0.0	FPC: MPCE Type 3 3D @ 7/*/*
jnxOperatingDescr.7.9.0.0	FPC: MS-MPC @ 8/*/*
jnxOperatingDescr.7.10.0.0	FPC: MPC4E 3D 2CGE+8XGE @ 9/*/*
jnxOperatingDescr.7.11.0.0	FPC: MPC4E 3D 2CGE+8XGE @ 10/*/*
jnxOperatingDescr.7.12.0.0	FPC: MPC4E 3D 2CGE+8XGE @ 11/*/*
jnxOperatingDescr.8.1.1.0	PIC: 4x10GE SFPP @ 0/0/*
jnxOperatingDescr.8.1.2.0	PIC: 1X100GE CFP @ 0/1/*
jnxOperatingDescr.8.1.3.0	PIC: 4x10GE SFPP @ 0/2/*
jnxOperatingDescr.8.1.4.0	PIC: 1X100GE CFP @ 0/3/*
jnxOperatingDescr.8.2.1.0	PIC: 4x10GE SFPP @ 1/0/*
jnxOperatingDescr.8.2.2.0	PIC: 1X100GE CFP @ 1/1/*
jnxOperatingDescr.8.2.3.0	PIC: 4x10GE SFPP @ 1/2/*
jnxOperatingDescr.8.2.4.0	PIC: 1X100GE CFP @ 1/3/*
jnxOperatingDescr.8.4.1.0	PIC: 2X40GE QSFPP @ 3/0/*
jnxOperatingDescr.8.4.3.0	PIC: 10x 1GE(LAN) SFP @ 3/2/*
jnxOperatingDescr.8.4.4.0	PIC: 10x 1GE(LAN) SFP @ 3/3/*
jnxOperatingDescr.8.5.1.0	PIC: 2X40GE QSFPP @ 4/0/*
jnxOperatingDescr.8.8.1.0	PIC: 2X40GE QSFPP @ 7/0/*
jnxOperatingDescr.8.8.3.0	PIC: 10x 1GE(LAN) SFP @ 7/2/*
jnxOperatingDescr.8.8.4.0	PIC: 10x 1GE(LAN) SFP @ 7/3/*
jnxOperatingDescr.8.9.1.0	PIC: MS-MPC-PIC @ 8/0/*
jnxOperatingDescr.8.9.2.0	PIC: MS-MPC-PIC @ 8/1/*
jnxOperatingDescr.8.9.3.0	PIC: MS-MPC-PIC @ 8/2/*
jnxOperatingDescr.8.9.4.0	PIC: MS-MPC-PIC @ 8/3/*
jnxOperatingDescr.8.10.1.0	PIC: 4x10GE SFPP @ 9/0/*
jnxOperatingDescr.8.10.2.0	PIC: 1X100GE CFP @ 9/1/*
jnxOperatingDescr.8.10.3.0	PIC: 4x10GE SFPP @ 9/2/*
jnxOperatingDescr.8.10.4.0	PIC: 1X100GE CFP @ 9/3/*
jnxOperatingDescr.8.11.1.0	PIC: 4x10GE SFPP @ 10/0/*
jnxOperatingDescr.8.11.2.0	PIC: 1X100GE CFP @ 10/1/*
jnxOperatingDescr.8.11.3.0	PIC: 4x10GE SFPP @ 10/2/*
jnxOperatingDescr.8.11.4.0	PIC: 1X100GE CFP @ 10/3/*
jnxOperatingDescr.8.12.1.0	PIC: 4x10GE SFPP @ 11/0/*
jnxOperatingDescr.8.12.2.0	PIC: 1X100GE CFP @ 11/1/*
jnxOperatingDescr.8.12.3.0	PIC: 4x10GE SFPP @ 11/2/*
jnxOperatingDescr.8.12.4.0	PIC: 1X100GE CFP @ 11/3/*
jnxOperatingDescr.9.1.0.0	Routing Engine 0
jnxOperatingDescr.9.2.0.0	Routing Engine 1
jnxOperatingDescr.10.1.1.0	FPM Board

Use case: router resources



Use case: router resources

Metrics:
CPU temperature
CPU load
Memory usage

Resolution: 300s

Targets: 6 router

Database: router_resources

RP: Default, 1 year

```
[agent]
  interval = “300s”

[[outputs.influxdb]]
  urls = ["http://localhost:8086"]
  database = “router_resources”
  retention_policy = "default“

[[inputs.snmp]]
 [[inputs.snmp.host]]
  address = “host1:161"
  [[inputs.snmp.host.table]]
   name = "router_metrics"
   include_instances=["Routing Engine"]

 [[inputs.snmp.host]]
  address = “host2:161"
  [[inputs.snmp.host.table]]
   name = "router_metrics"
   include_instances=["Routing Engine"]
```

```
[[inputs.snmp.table]]
        name = "router_metrics"
        mapping_table = ".1.3.6.1.4.1.2636.3.1.13.1.5"
        sub_tables=["mem_perc","cpu_temp","cpu_load"]

        [[inputs.snmp.subtable]]
            name = "mem_perc"
            oid = ".1.3.6.1.4.1.2636.3.1.13.1.11"
            unit = "%"

        [[inputs.snmp.subtable]]
            name = "cpu_temp"
            oid = ".1.3.6.1.4.1.2636.3.1.13.1.7"
            unit = "C"

        [[inputs.snmp.subtable]]
            name = "cpu_load"
            oid = ".1.3.6.1.4.1.2636.3.1.13.1.8"
            unit = "%"
```

Use case: router resources, influxdb

```
> CREATE DATABASE router_resources
> ALTER RETENTION POLICY default ON router_resources DURATION 366d DEFAULT
> SHOW RETENTION POLICIES ON router_resources
name    duration         shardGroupDuration    replicaN    default
default 8784h0m0s        168h0m0s              1           true
```

Use case: router resources

Telegraf: test della configurazione

```
$ telegraf -config telegraf_snmp_router_resources.conf -test
* Plugin: snmp, Collection 1
> jnxOperatingTemp,host=re1.aq1.garr.net,instance=Routing\ Engine,unit=C jnxOperatingTemp=37i 1460731566292176208
> jnxOperatingCPU,host=re1.aq1.garr.net,instance=Routing\ Engine,unit=% jnxOperatingCPU=11i 1460731566292248166
> jnxOperatingBuffer,host=re1.aq1.garr.net,instance=Routing\ Engine,unit=% jnxOperatingBuffer=81i 1460731566292281664
> jnxOperatingBuffer,host=re2.aq1.garr.net,instance=Routing\ Engine,unit=% jnxOperatingBuffer=81i 1460731566351188759
> jnxOperatingTemp,host=re2.aq1.garr.net,instance=Routing\ Engine,unit=C jnxOperatingTemp=38i 1460731566351254025
> jnxOperatingCPU,host=re2.aq1.garr.net,instance=Routing\ Engine,unit=% jnxOperatingCPU=14i 1460731566351295602
> jnxOperatingCPU,host=re1.an.garr.net,instance=Routing\ Engine,unit=% jnxOperatingCPU=11i 1460731566417831668
> jnxOperatingBuffer,host=re1.an.garr.net,instance=Routing\ Engine,unit=% jnxOperatingBuffer=88i 1460731566417891704
> jnxOperatingTemp,host=re1.an.garr.net,instance=Routing\ Engine,unit=C jnxOperatingTemp=47i 1460731566417936529
> jnxOperatingBuffer,host=re2.an.garr.net,instance=Routing\ Engine,unit=% jnxOperatingBuffer=87i 1460731566468463369
> jnxOperatingTemp,host=re2.an.garr.net,instance=Routing\ Engine,unit=C jnxOperatingTemp=47i 1460731566468524464
> jnxOperatingCPU,host=re2.an.garr.net,instance=Routing\ Engine,unit=% jnxOperatingCPU=23i 1460731566468559297
> jnxOperatingBuffer,host=re1.tn.garr.net,instance=Routing\ Engine,unit=% jnxOperatingBuffer=82i 1460731566547304602
> jnxOperatingTemp,host=re1.tn.garr.net,instance=Routing\ Engine,unit=C jnxOperatingTemp=43i 1460731566547369192
> jnxOperatingCPU,host=re1.tn.garr.net,instance=Routing\ Engine,unit=% jnxOperatingCPU=15i 1460731566547409448
> jnxOperatingBuffer,host=re2.tn.garr.net,instance=Routing\ Engine,unit=% jnxOperatingBuffer=82i 1460731566624360940
> jnxOperatingTemp,host=re2.tn.garr.net,instance=Routing\ Engine,unit=C jnxOperatingTemp=43i 1460731566624419069
> jnxOperatingCPU,host=re2.tn.garr.net,instance=Routing\ Engine,unit=% jnxOperatingCPU=22i 1460731566624458327
```

Use case: router resources, influxdb

Risultati

```
> use router_resources
Using database router_resources
> select * from jnxOperatingCPU limit 3
name: jnxOperatingCPU
---------------------
time                 host            instance        jnxOperatingCPU unit
1460727781000000000  re1.aq1.garr.net  Routing Engine  8               %
1460727781000000000  re2.an.garr.net   Routing Engine  9               %
1460727781000000000  re1.tn.garr.net   Routing Engine  65              %

> select * from jnxOperatingTemp limit 6
name: jnxOperatingTemp
----------------------
time                 host            instance        jnxOperatingTemp unit
1460727781000000000  re1.aq1.garr.net  Routing Engine  38               C
1460727781000000000  re1.tn.garr.net   Routing Engine  44               C
1460727781000000000  re1.an.garr.net   Routing Engine  46               C
```

Use case: router resources, influxdb

Risultati

> **select * from jnxOperatingCPU limit 6**

name: jnxOperatingCPU

time	host	instance	jnxOperatingCPU	unit
1459956725000000000	<host-1>	Routing Engine 0	12	%
1459956725000000000	<host-1>	Routing Engine 1	6	%
1459956780000000000	<host-1>	Routing Engine 0	8	%
1459956780000000000	<host-1>	Routing Engine 1	1	%
1459956840000000000	<host-1>	Routing Engine 0	10	%
1459956840000000000	<host-1>	Routing Engine 1	1	%

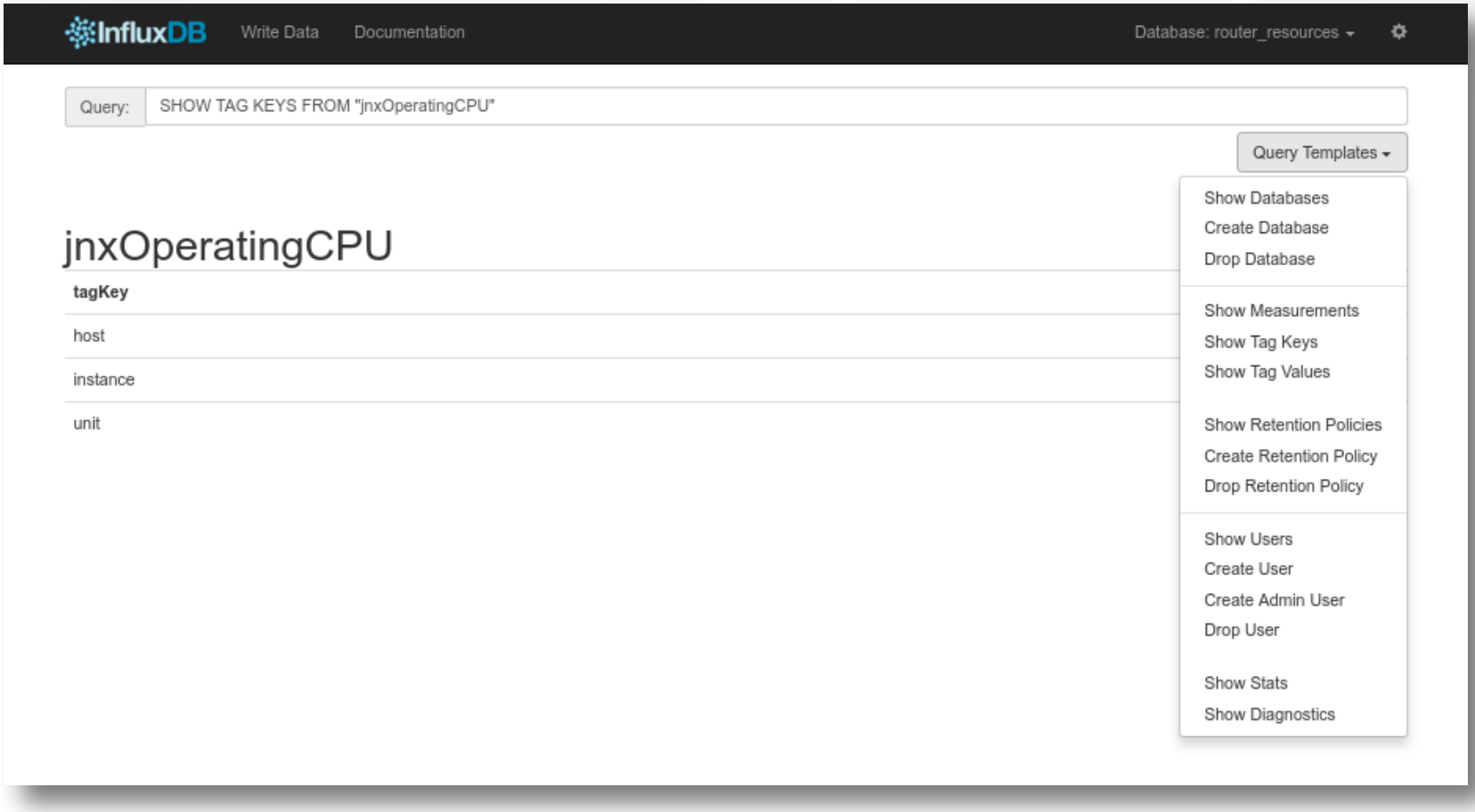
> **select * from jnxOperatingTemp limit 6**

name: jnxOperatingTemp

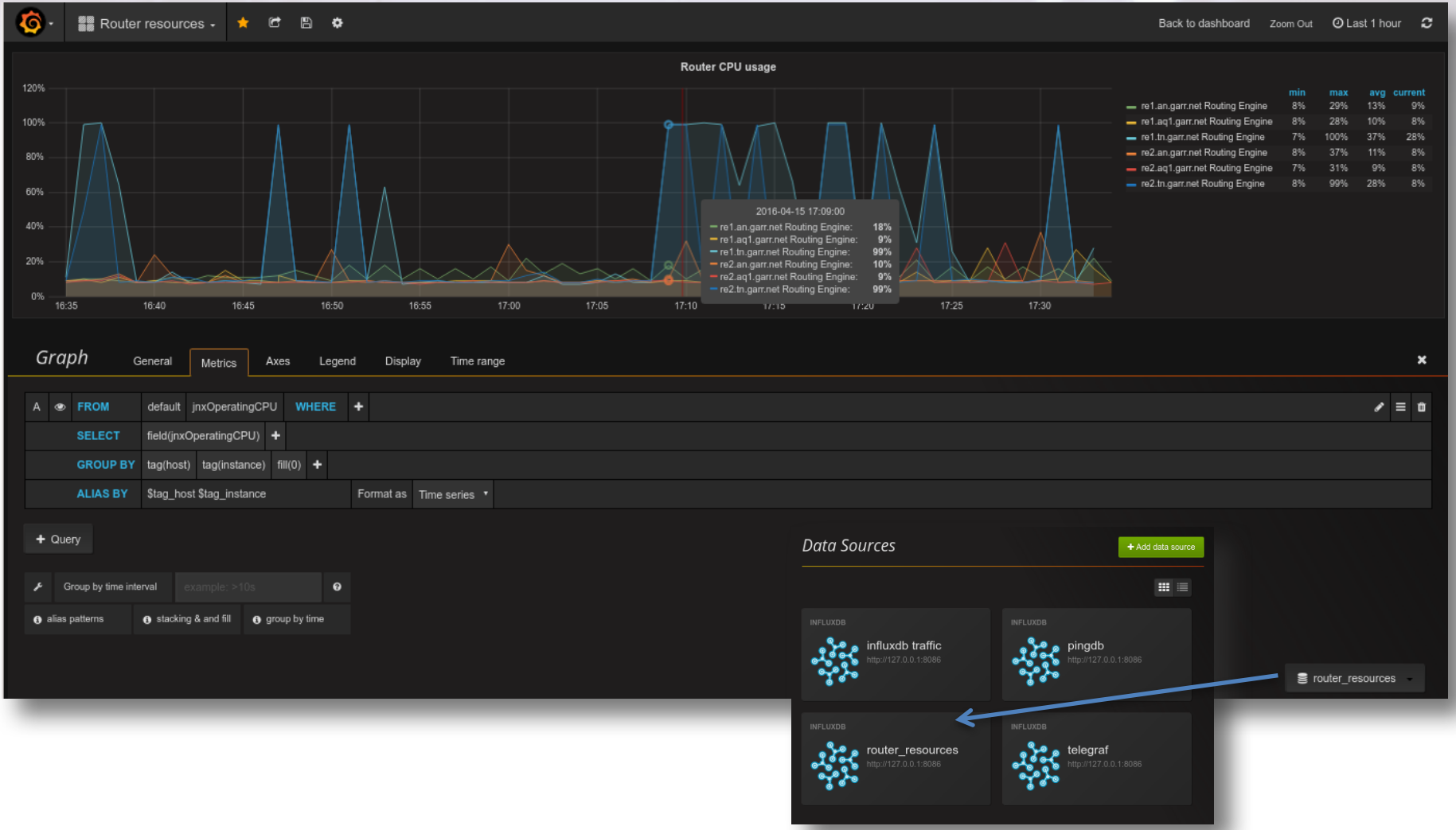
time	host	instance	jnxOperatingTemp	unit
1459956725000000000	<host-1>	Routing Engine 1	30	C
1459956725000000000	<host-1>	Routing Engine 0	30	C
1459956780000000000	<host-1>	Routing Engine 1	30	C
1459956780000000000	<host-1>	Routing Engine 0	30	C
1459956840000000000	<host-1>	Routing Engine 1	30	C
1459956840000000000	<host-1>	Routing Engine 0	30	C

InfluxDB: WEB UI

http://localhost:8083/



Grafana


Router resources
Back to dashboard
Zoom Out
Last 1 hour
Router CPU usage
120%
100%
80%
60%
40%
20%
0%
16:35
16:40
16:45
16:50
16:55
17:00
17:05
17:10
17:25
17:30
min max avg current
re1.an.garr.net Routing Engine 8% 29% 13% 9%
re1.aq1.garr.net Routing Engine 8% 28% 10% 8%
re1.tn.garr.net Routing Engine 7% 100% 37% 28%
re2.an.garr.net Routing Engine 8% 37% 11% 8%
re2.aq1.garr.net Routing Engine 7% 31% 9% 8%
re2.tn.garr.net Routing Engine 8% 99% 28% 8%
2016-04-15 17:09:00
re1.an.garr.net Routing Engine: 18%
re1.aq1.garr.net Routing Engine: 9%
re1.tn.garr.net Routing Engine: 99%
re2.an.garr.net Routing Engine: 10%
re2.aq1.garr.net Routing Engine: 9%
re2.tn.garr.net Routing Engine: 99%
Graph
General
Metrics
Axes
Legend
Display
Time range
A
FROM default jnxOperatingCPU WHERE
SELECT field(jnxOperatingCPU)
GROUP BY tag(host) tag(instance) fill(0)
ALIAS BY $tag_host $tag_instance
Format as Time series
+ Query
Group by time interval
example: >10s
alias patterns
stacking & and fill
group by time
Data Sources
+ Add data source
INFLUXDB
influxdb traffic
http://127.0.0.1:8086
INFLUXDB
pingdb
http://127.0.0.1:8086
router_resources
INFLUXDB
router_resources
http://127.0.0.1:8086
INFLUXDB
telegraf
http://127.0.0.1:8086

Cosa facciamo ora?

1. Come funziona Telegraf
2. Acquisizione alcune metriche di un host
    1. Chronograf
    2. Grafana
3. Misure di latenza verso 2 router
4. SNMP input plugin
5. Acquisizione delle risorse CPU/MEM di un router Juniper
6. **Acquisizione traffico delle interfacce di un router**

Network traffic

Influxdb: creo il DB per il traffico e le retention policy

```
> create database traffic
> CREATE RETENTION POLICY one_day        ON traffic DURATION 1d REPLICATION 1 DEFAULT
> CREATE RETENTION POLICY one_year       ON traffic DURATION 366d REPLICATION 1

> show retention policies on traffic
name        duration         shardGroupDuration    replicaN    default
default     0                168h0m0s              1           false
one_day     24h0m0s          1h0m0s                1           true
one_year    8784h0m0s        168h0m0s              1           false
```

Telegraf: snmp plugin, traffic load

```
[agent]
        interval = “300s“
....
[[outputs.influxdb]]
        urls = ["http://localhost:8086"]
        database = "traffic"
        retention_policy = "one_day“
....
[[inputs.snmp]]
 snmptranslate_file = “/<>/oids.txt”
 [[inputs.snmp.host]]
  address = “<host>:161”
  community = "public"
  version = 2
  [[inputs.snmp.host.table]]
   name = "traffic"
   include_instances = ["ae3.1","ae13.1"]
                                ->
```

```
->
  [[inputs.snmp.table]]
   name = "traffic"
   mapping_table = ".1.3.6.1.2.1.31.1.1.1.1"
   sub_tables=[ "bytes_in", "bytes_out"]

    [[inputs.snmp.subtable]]
      name = "bytes_in"
     oid = ".1.3.6.1.2.1.31.1.1.1.6"
     unit = "octets"

    [[inputs.snmp.subtable]]
     name = "bytes_out"
     oid = ".1.3.6.1.2.1.31.1.1.1.10"
     unit = "octets"
```

Telegraf: snmp plugin, traffic load

> use traffic

> select * from ifHCInOctets limit 6

name: ifHCInOctets

time	host	ifHCInOctets	instance	unit
1459956726000000000	<host-1>	4351606994801	ae13.1	octets
1459956726000000000	<host-1>	1993476041393054	ae3.1	octets
1459956780000000000	<host-1>	4351619427675	ae13.1	octets
1459956780000000000	<host-1>	1993481753563463	ae3.1	octets
1459956840000000000	<host-1>	4351635233203	ae13.1	octets
1459956840000000000	<host-1>	1993489848850885	ae3.1	octets

> select * from ifHCOutOctets limit 6

name: ifHCOutOctets

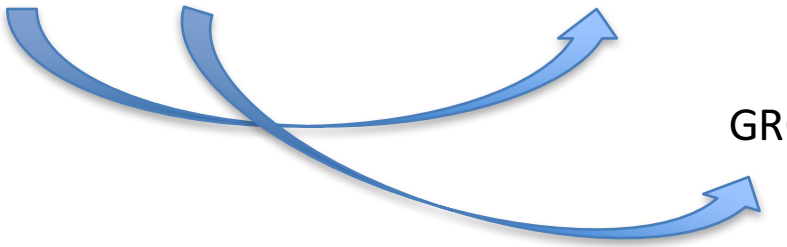
time	host	ifHCOutOctets	instance	unit
1459956726000000000	<host-1>	4046441953597389	ae3.1	octets
1459956726000000000	<host-1>	74488515522382	ae13.1	octets
1459956780000000000	<host-1>	4046456941803575	ae3.1	octets
1459956780000000000	<host-1>	74489082969856	ae13.1	octets
1459956840000000000	<host-1>	4046472739030133	ae3.1	octets
1459956840000000000	<host-1>	74489723483331	ae13.1	octets

InfluxDB bitrate

```
SELECT  8*derivative(mean(ifHCInOctets),1s)
FROM "traffic". "one_day"."ifHCInOctets"
GROUP BY  time(<time_interval>), instance, host
```

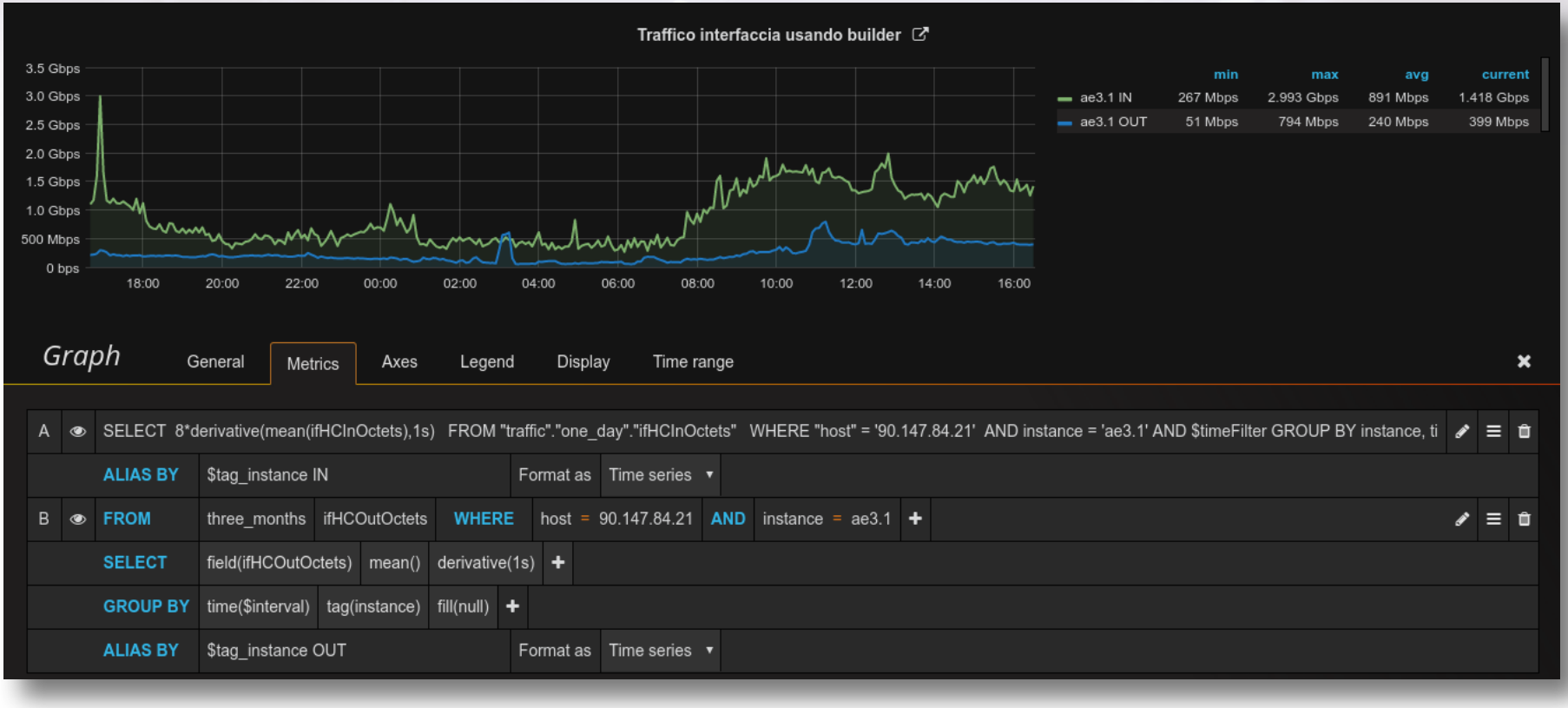
ifHCInOctets: contatore incrementale dei Bytes a 64bit

8*derivative(mean(ifHCInOctets),1s) : bps



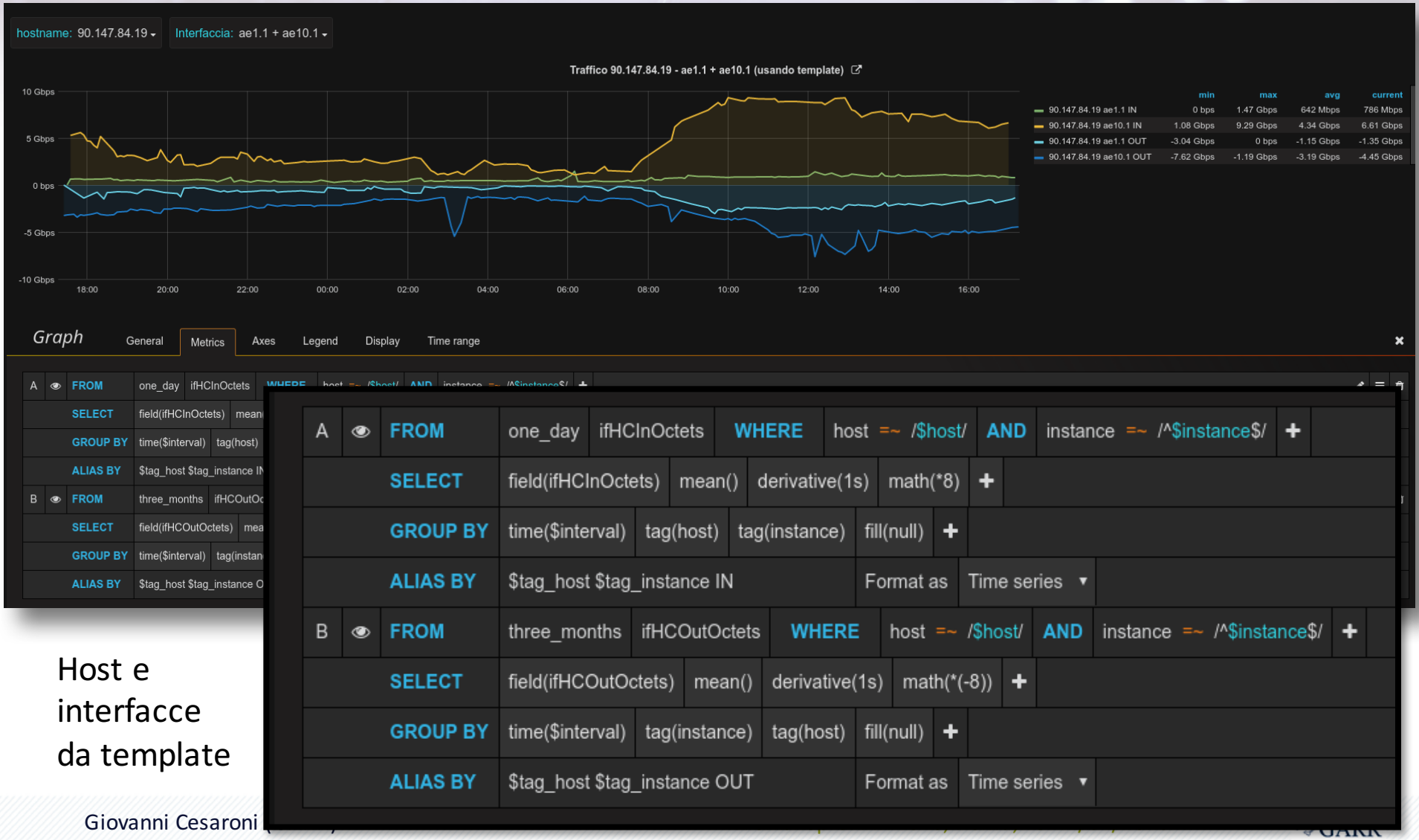
GROUP BY time(<time_interval>)

Grafana:



IN da query
OUT da builder

Grafana: templating

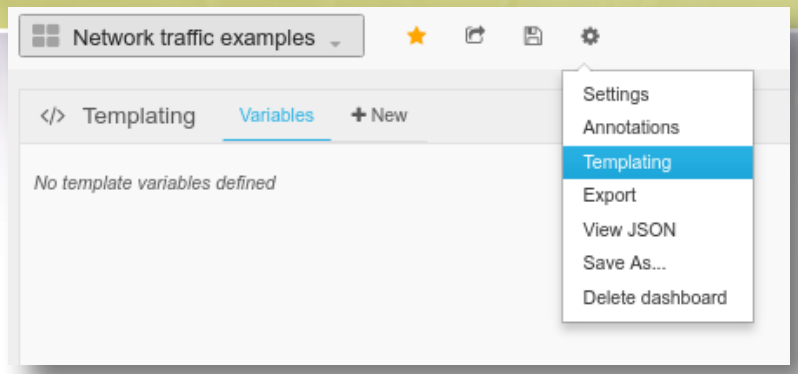


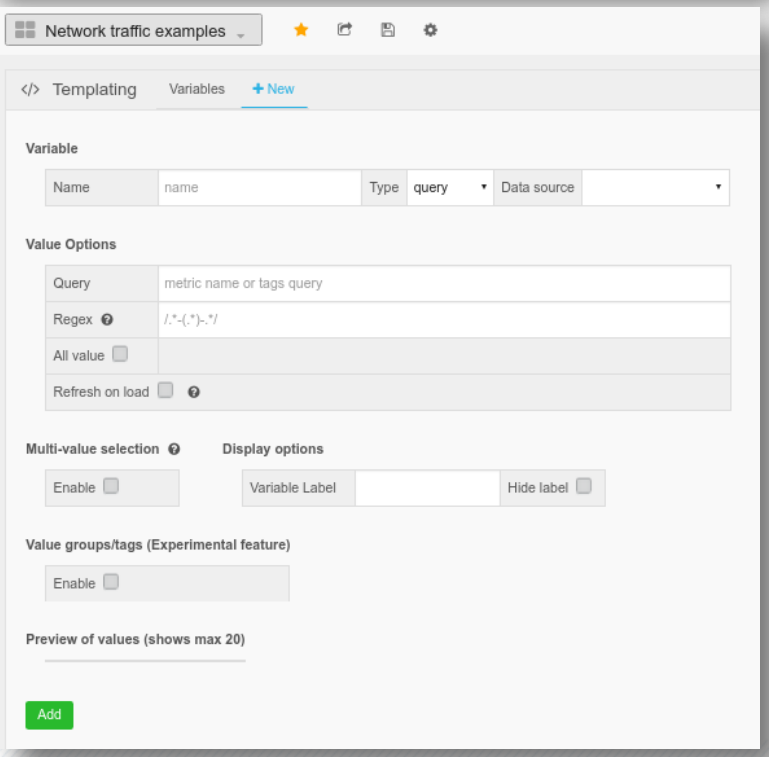
Host e interfacce da template

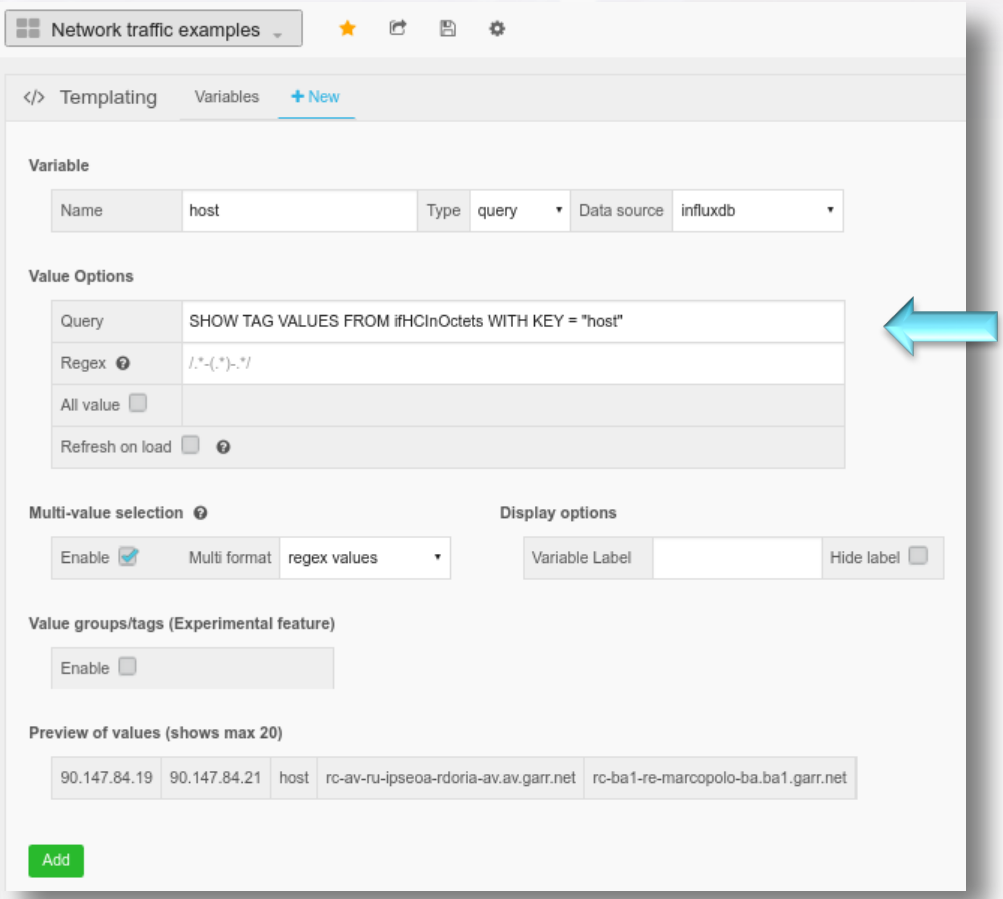
Grafana:


Network traffic examples
Back to dashboard
Zoom Out
Last 24 hours
hostname: re1.aq1.garr.net
Interfaccia: All
All logical interfaces of re1.aq1.garr.net
1.0 Gbps
750 Mbps
500 Mbps
250 Mbps
0 bps
-250 Mbps
-500 Mbps
-750 Mbps
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
min max avg current
re1.aq1.garr.net ge-1/0/0.0 IN 0 bps 18 Mbps 4 Mbps 447 kbps
re1.aq1.garr.net ge-1/0/1.1 IN 0 bps 775 Mbps 233 Mbps 75 Mbps
re1.aq1.garr.net ge-1/0/2.0 IN 0 bps 54 bps 31 bps 49 bps
re1.aq1.garr.net ge-1/1/0.2 IN 0 bps 1 kbps 648 bps 392 bps
re1.aq1.garr.net ge-1/1/1.0 IN 0 bps 117 Mbps 22 Mbps 7 Mbps
re1.aq1.garr.net ge-1/1/2.0 IN 0 bps 427 kbps 17 kbps 474 bps
re1.aq1.garr.net xe-0/0/0.1 IN 0 bps 103 kbps 5 kbps 4 kbps
re1.aq1.garr.net xe-1/2/0.1 IN 0 bps 29 kbps 341 bps 0 bps
re1.aq1.garr.net ge-1/0/0.0 OUT -362 Mbps 0 bps -113 Mbps -72 Mbps
re1.aq1.garr.net ge-1/0/1.1 OUT -162 Mbps 0 bps -26 Mbps -15 Mbps
re1.aq1.garr.net ge-1/0/2.0 OUT -137 bps 0 bps -82 bps -93 bps
re1.aq1.garr.net ge-1/1/0.2 OUT -1 kbps 0 bps -685 bps -19 bps
re1.aq1.garr.net ge-1/1/1.0 OUT -655 Mbps 0 bps -122 Mbps -50 Mbps
re1.aq1.garr.net ge-1/1/2.0 OUT -22 Mbps 0 bps -505 bps -2 kbps
re1.aq1.garr.net xe-0/0/0.1 OUT -122 kbps 0 bps -6 kbps -4 kbps
re1.aq1.garr.net xe-1/2/0.1 OUT -128 kbps -7 bps -1 kbps -18 bps
Graph
General
Metrics
Axes
Legend
Display
Time range
A SELECT 8*derivative(mean(ifHCInOctets),1s) FROM "traffic"."one_day"."ifHCInOctets" WHERE $timeFilter AND instance =~ /.*\..*/ AND instance !~ /lo.*|fxp.*|em.*|lc.*|lsi.*|me.*|pf.*|ae.*|^+.*32767$/ AND host =~ /$host/ GROUP BY time
ALIAS BY $tag_host $tag_instance IN
Format as Time series
B SELECT (-8)*derivative(mean(ifHCOutOctets),1s) FROM "traffic"."one_day"."ifHCOutOctets" WHERE $timeFilter AND instance =~ /.*\..*/ AND instance !~ /lo.*|fxp.*|em.*|lc.*|lsi.*|me.*|pf.*|ae.*|^+.*32767$/ AND host =~ /$host/ GROUP BY
ALIAS BY $tag_host $tag_instance OUT
Format as Time series

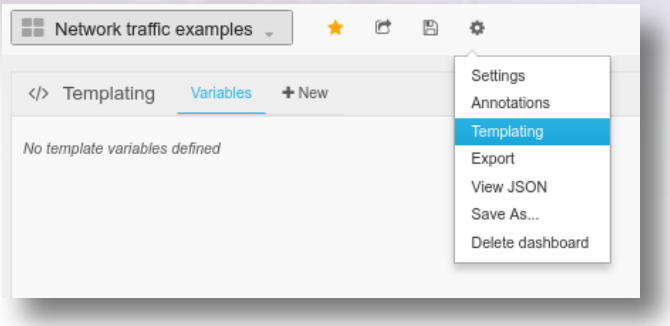
Grafana: templating

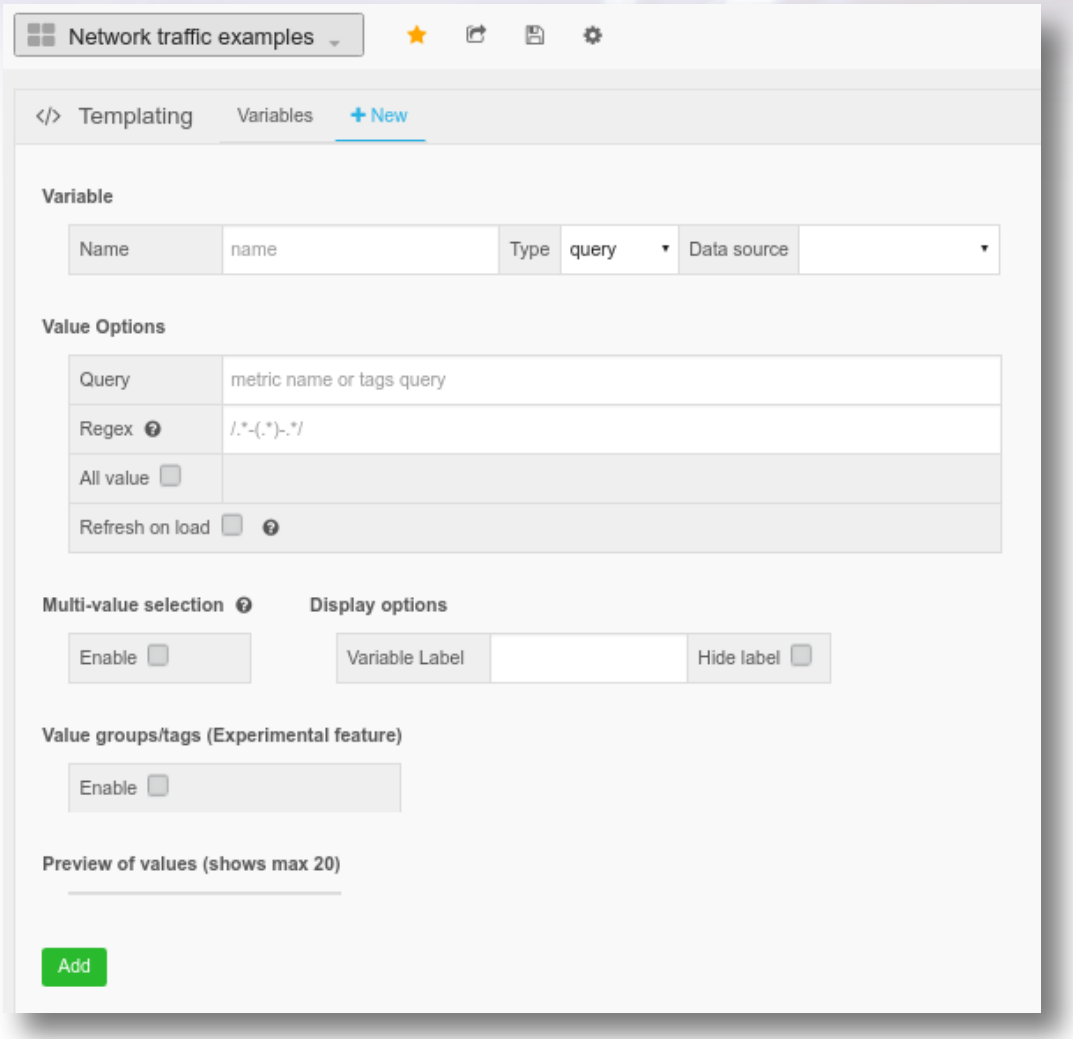




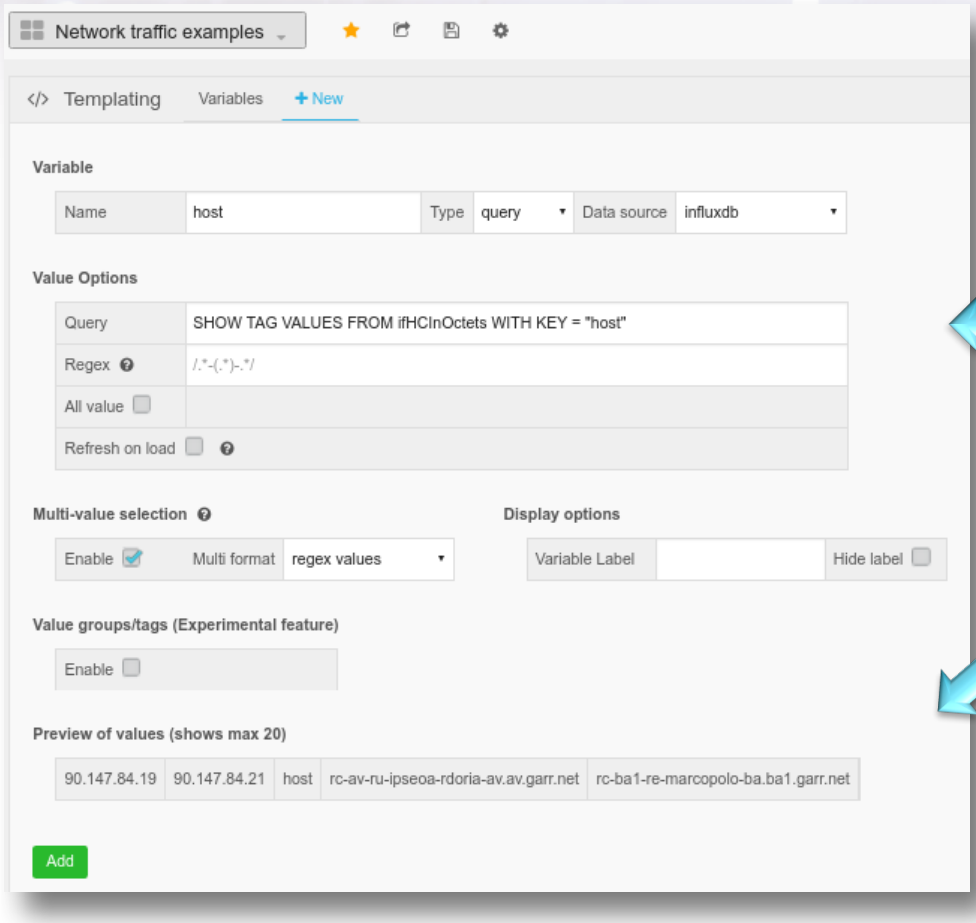


Grafana: templating





Grafana: templating



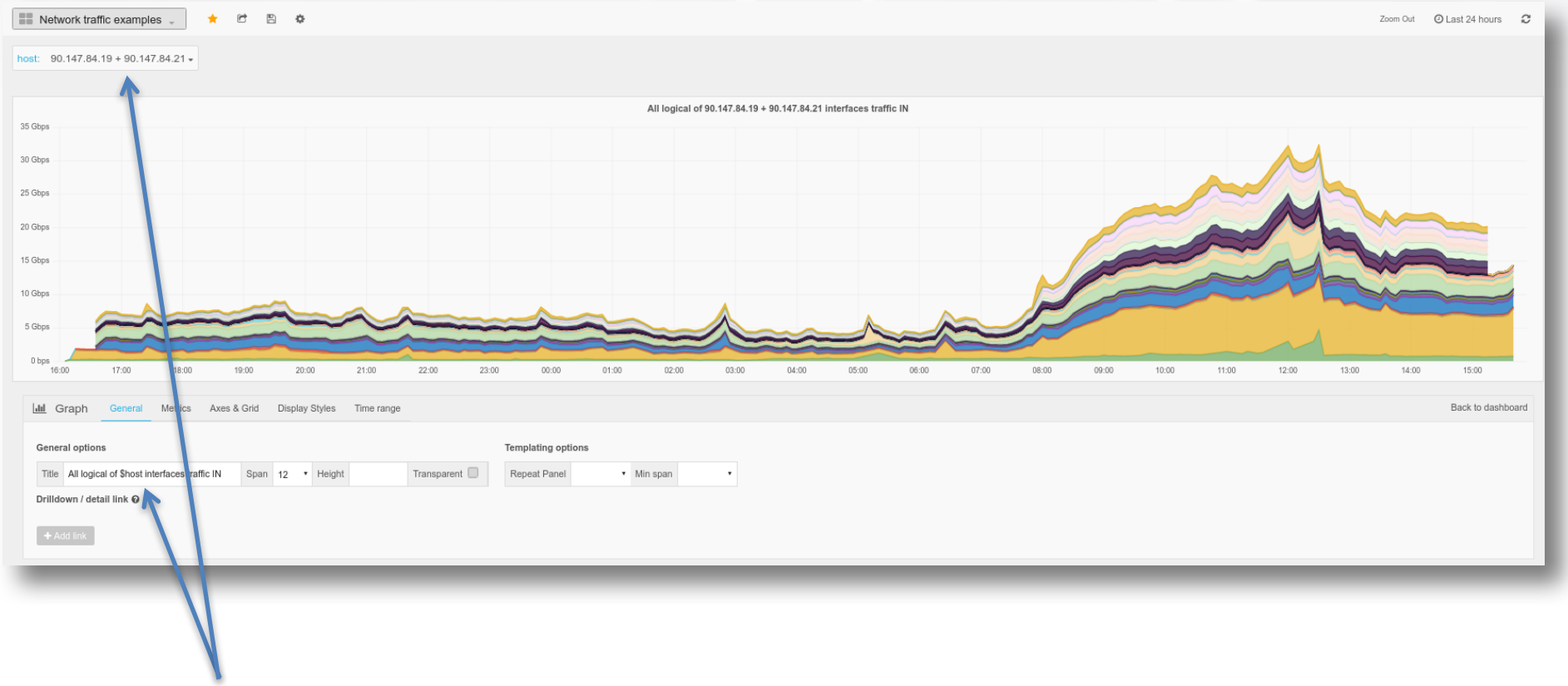

```
> SHOW TAG VALUES FROM ifHCInOctets
WITH KEY = "host"
name: ifHCInOctets
------------------
key     value
host    host1.garr.net
host    host2.garr.net
```

Grafana: templating

```
SELECT  8*derivative(mean(ifHCInOctets),1s)
FROM "traffic".."ifHCInOctets"
WHERE $timeFilter   AND
      instance !~ /tap|lo0|dsc|gre|fxp0|pf.*|ms.*|lsi.*|pc.*|.*32767/ AND
      instance =~ /.*\..*/   AND
      host =~ /$host/
GROUP BY  time(300s), instance, host
```

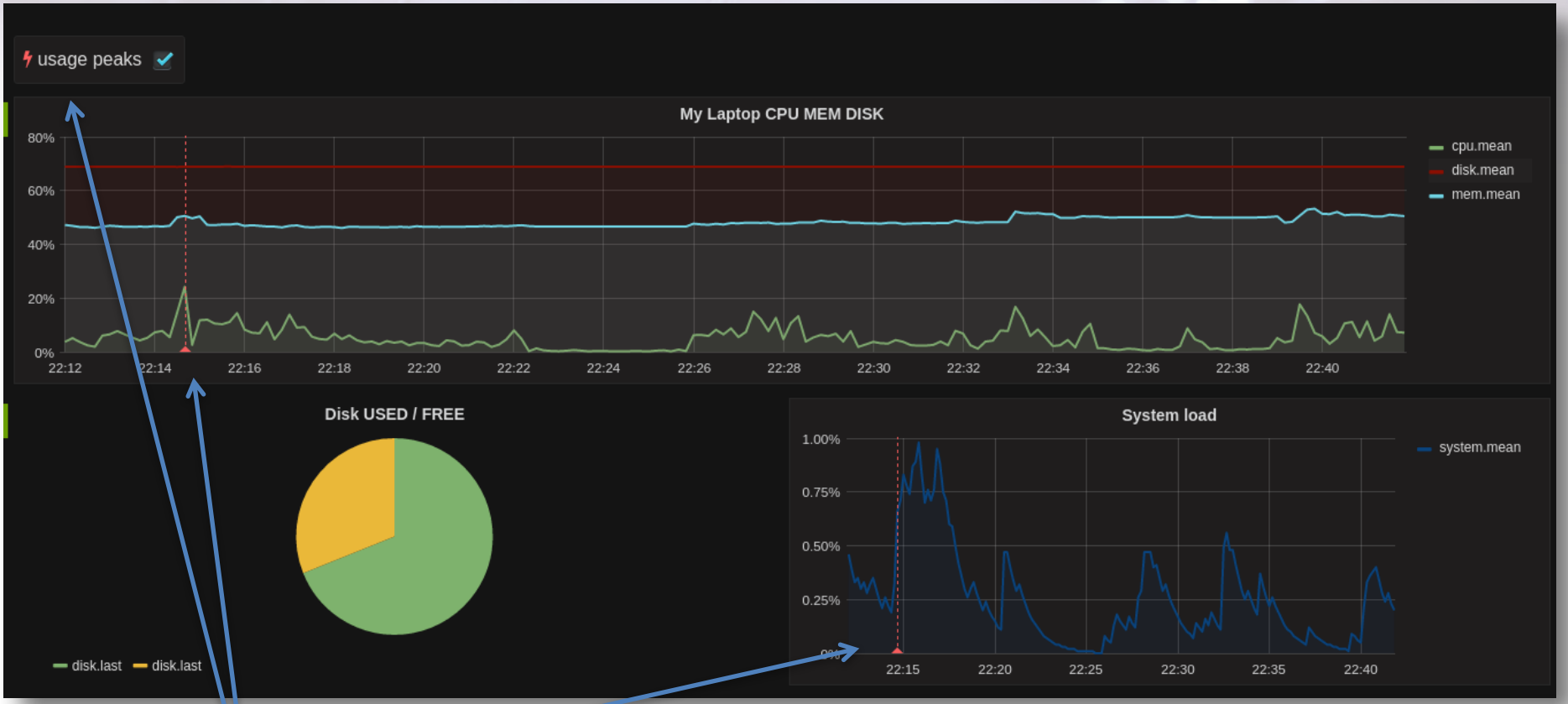
$host : SHOW TAG VALUES FROM ifHCInOctets WITH KEY = "host"

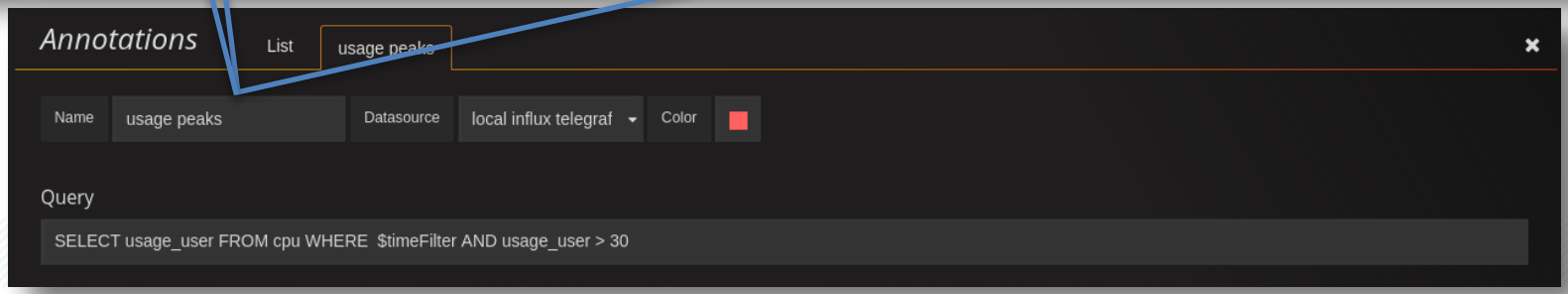
Grafana: templating



$host from template

Grafana: annotations

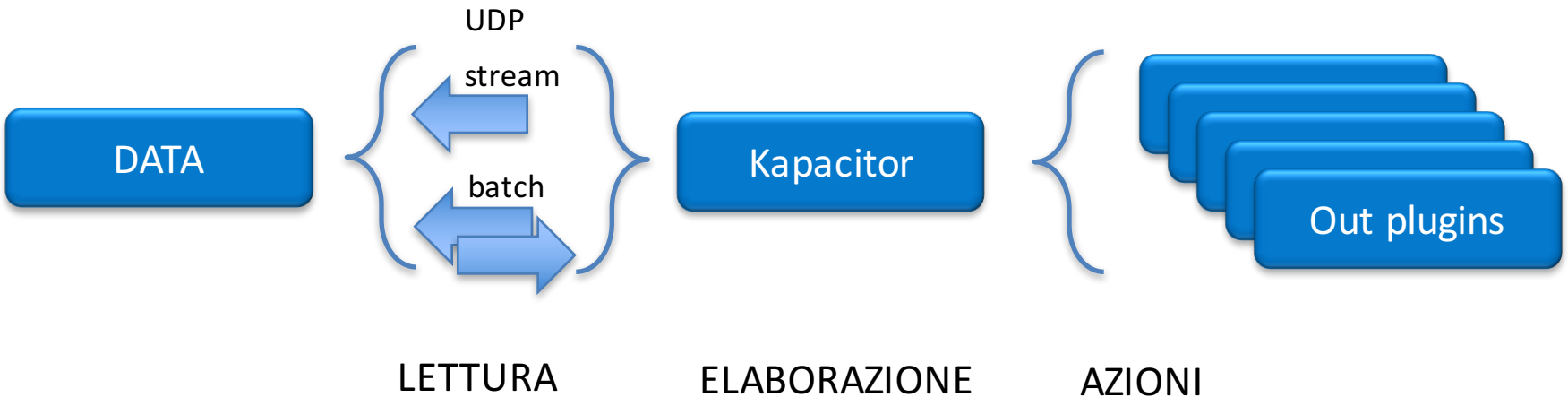




Consortium GARR

Kapacitor

Kapacitor = data processing engine.



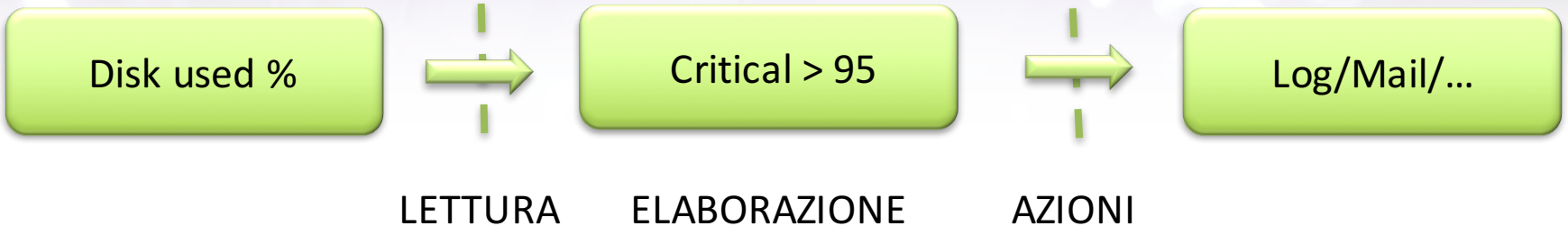
Kapacitor

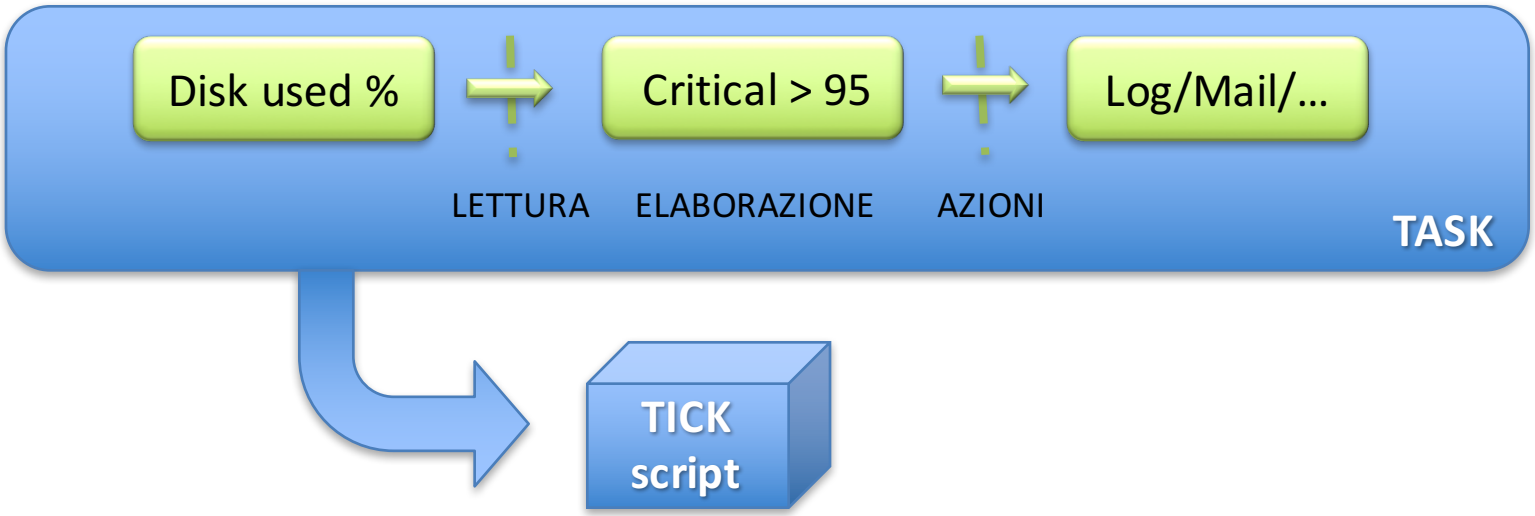
Kapacitor = data processing engine.

start

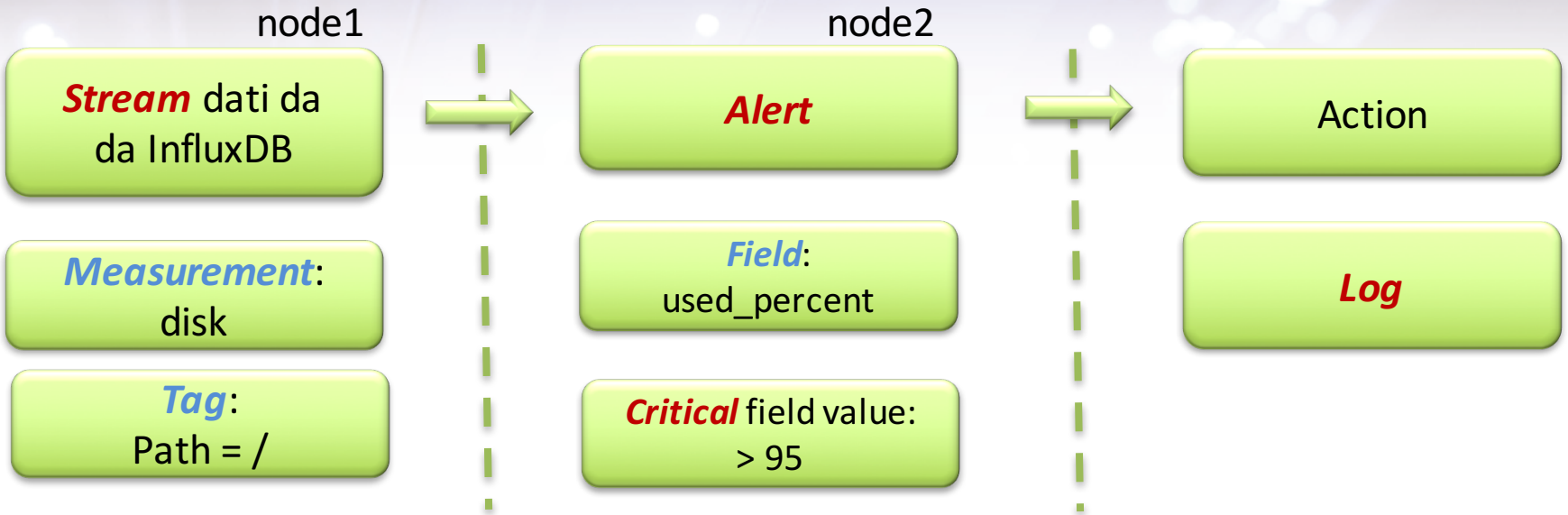
```
$ kapacitord config > kapacitor.conf
$ kapacitord -config kapacitor.conf
```

Kapacitor: use case





Kapacitor: task




```
stream
    |from()
        .measurement('disk')
        .where(lambda:"path"=='/')
    |alert()
        .crit(lambda: "used_percent" > 95)
        .log('/tmp/alerts.log')
```

Tick script

Kapacitor: quale metrica

```
> use <db>
> SHOW TAG VALUES  FROM disk WITH key = path
name: disk
----------
key     value
path    /
path    /boot
```

```
> SHOW FIELD KEYS FROM disk
name: disk
----------
fieldKey
free
inodes_free
inodes_total
inodes_used
total
used
used_percent
```

```
> SELECT used_percent FROM disk WHERE path = '/' limit  1
name: disk
----------
time                    used_percent
1459510950000000000     95.69366163268641
```

Kapacitor: tick script

disk_alert.tick

```
stream
    |from()
        .measurement('disk')
        .where(lambda:"path"=='/')
    |alert()
        .crit(lambda: "used_percent" > 95)
        .log('/tmp/alerts.log')
```

Definisce:
- metrica
- condizione critica
- azione

```
$ kapacitor define
        -name disk_alert
        -type stream
        -tick disk.tick
        -dbrp telegraf.default
```

Definisce:
- dove e come leggere i dati

Abbiamo definito il task ***disk_alert***

Kapacitor: test task, run task

Registriamo uno stream di dati su cui provare il task ***disk_alert***

```
$ kapacitor record stream -name disk_alert -duration 20s
034e8591-d4a2-48b5-b015-7d93dbd4a5bc
```

```
$ rid=034e8591-d4a2-48b5-b015-7d93dbd4a5bc
$ kapacitor list recordings $rid
ID                                    Type    Size     Created
034e8591-d4a2-48b5-b015-7d93dbd4a5bc  stream  2.6 kB   05 Apr 16 14:28 CEST
```

Eseguiamo il test del tick sui dati registrati

```
$ kapacitor replay -id $rid -name disk_alert -fast
```

/tmp/alerts.log

```
$ kapacitor enable disk_alert
```

Kapacitor: cosa abbiamo fatto?

1. Definire la logica del task e scrivere il tick script
2. Definire su quali dati il task lavora (***define***)
   - kapacitor define -name <name> -type stream -tick <file>.tick -dbrp <db>.<rp>
3. Registrare un sample di dati per fare un test del task (***record***)
   - kapacitor record stream -name <name> -duration <duration>
   - -> <record_id>
4. Eseguire il test (***replay***)
   - kapacitor replay -id <record_id> -name <name> -fast
5. Eseguire/abilitare il task (***enable***)
   - kapacitor enable <name>

Kapacitor: out plugins

Kapacitor.conf

[smtp]

[opsgenie]

[victorops]

[pagerduty]

[sensu]

[slack]

[hipchat]

[alerta]

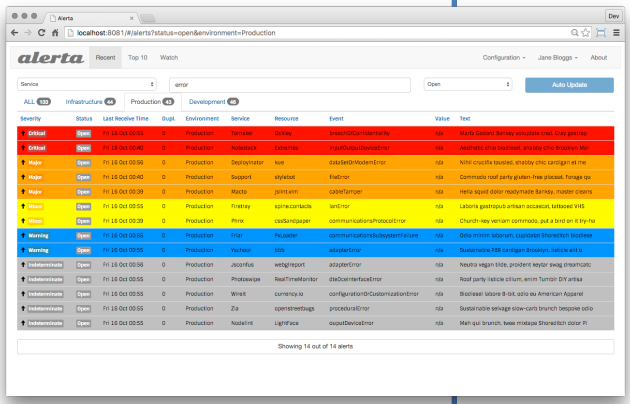
[reporting]

[stats]

[udf]

[deadman]

[talk]



- Alert Node
- Derivative Node
- Eval Node
- Group By Node
- HTTP Output Node
- InfluxDB Output Node
- InfluxQL Node
- Join Node
- Log Node
- NoOp Node
- Sample Node
- Shift Node
- Source Batch Node
- Source Stream Node
- Stats Node
- UDF (User Defined Function) Node
- Union Node
- Where Node
- Window Node

Kapacitor: mail

Kapacitor.conf

[smtp]

Kapacitor.conf

```
[smtp]
  enabled = true
  host = "localhost"
  port = 25
  username = ""
  password = ""
  no-verify = true
  global = false
  state-changes-only = false
  from = “sender@mydomain.it"
  idle-timeout = "30s"
  to = [“mymail@mydomain.com"]
```

```
stream
    |from()
        .measurement('disk')
        .where(lambda:"path"=='/')
    |alert()
          .id('Kapacitor/{{ index .Tags "host" }}')
          .message('{{ .ID }} is {{ .Level }} value:{{ index .Fields "used_percent" }}')
          .info(lambda: "used_percent" > 80)
          .warn(lambda: "used_percent" > 90)
          .crit(lambda: "used_percent" > 95)
          .email()
```

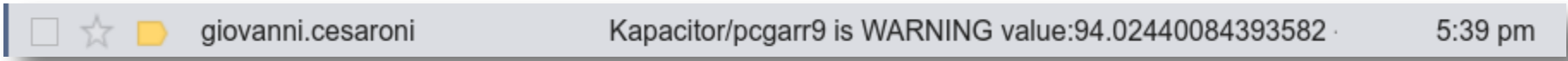
Kapacitor: mail

Kapacitor.conf

```
[smtp]
```

```
stream
    |from()
      .measurement('disk')
      .where(lambda:"path"=='/')
    |alert()
        .id('Kapacitor/{{ index .Tags "host" }}')
        .message('{{ .ID }} is {{ .Level }} value:{{ index .Fields "used_percent" }}')
        .info(lambda: "used_percent" > 80)
        .warn(lambda: "used_percent" > 90)
        .crit(lambda: "used_percent" > 95)
        .email()
```

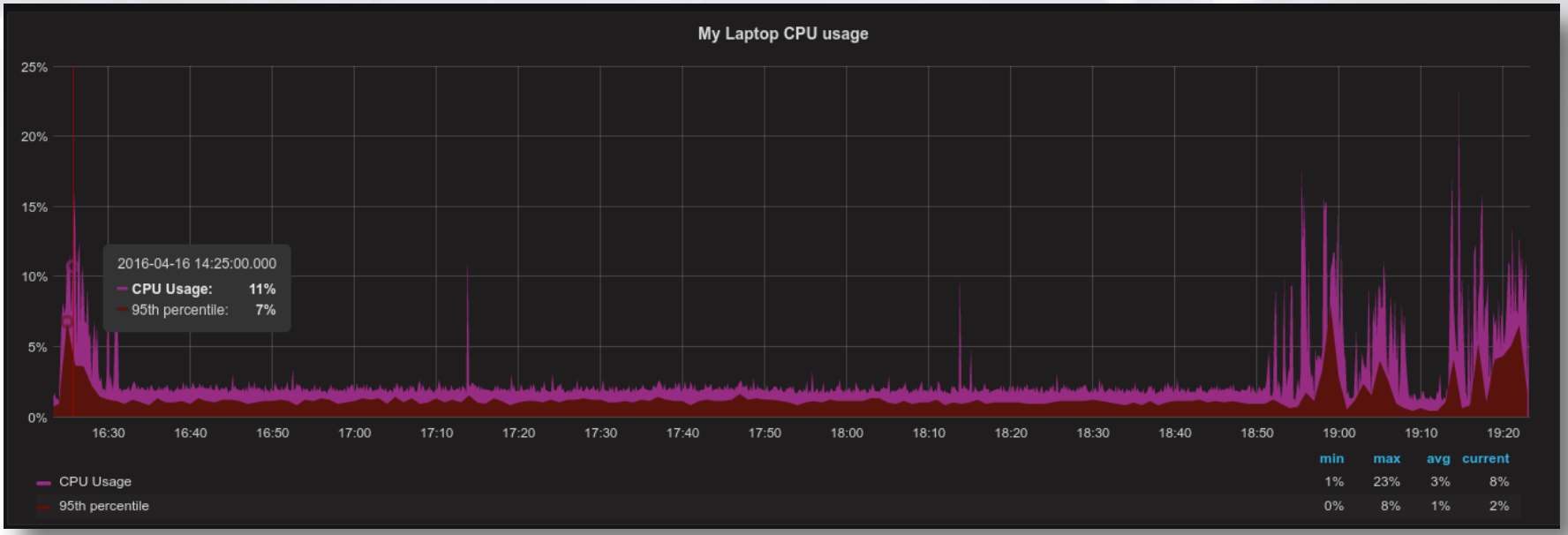
```
$ kapacitor replay -id $rid -name disk_alert -fast
```



Kapacitor: tick, eval, CPU usage

```
stream
    |from()
        .measurement('cpu')
        .where(lambda: "cpu" == 'cpu-total')
    |eval(lambda: 100.0 - "usage_idle")
        .as('usage')
    |alert()
        .warn(lambda: "usage" > 70)
        .crit(lambda: "usage" > 80)
        .log('/tmp/alerts.log')
```

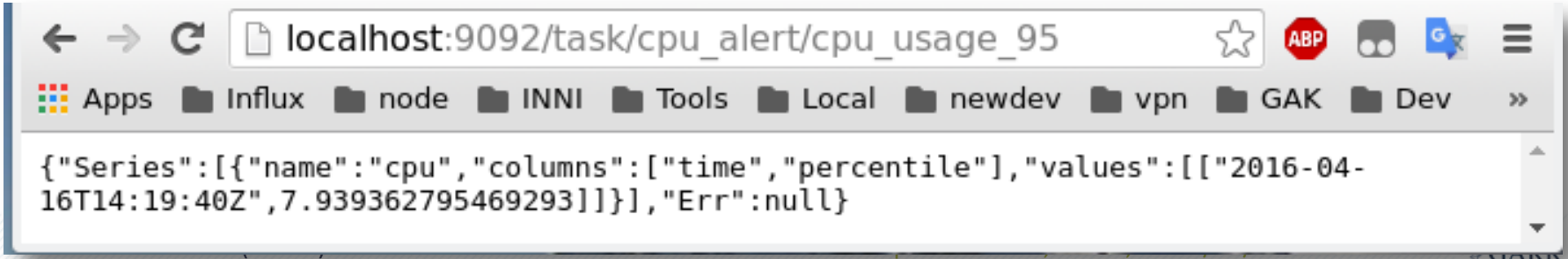
Percentile



A		SELECT 100 - mean("usage_idle") FROM "cpu" WHERE $timeFilter GROUP BY time($interval) fill(null)			
	ALIAS BY	CPU Usage	Format as	Time series	
D		SELECT 100 - percentile("usage_idle",95) FROM "cpu" WHERE $timeFilter GROUP BY time(60s) fill(null)			
	ALIAS BY	95th percentile	Format as	Time series	

Kapacitor: tick, CPU usage,httpOut

```
stream
    |from()
        .measurement('cpu')
        .where(lambda: "cpu" == 'cpu-total')
    |eval(lambda: 100.0 - "usage_idle")
        .as('usage')
    |window()
        .period(1m)
        .every(1m)
    |percentile('usage', 95.0)
    |httpOut('cpu_usage_95')
```



Kapacitor: CPU usage, alerts

```
stream
    |from()
      .measurement('cpu')
    |eval(lambda: 100.0 - "usage_idle")
      .as('used')
    |groupBy('host','cpu')
    |window()
      .period(1m)
      .every(1m)
    |percentile('used', 95.0)
    |eval(lambda: sigma("percentile"))
      .as('sigma')
      .keep('percentile', 'sigma')
     |alert()
      .id('Task:{{ .Name }} Host:{{ index .Tags "host" }} CPU:{{ index .Tags "cpu" }}')
      .message('{{ .ID }} is {{ .Level }} cpu-95th:{{ index .Fields "percentile" }}')
      .warn(lambda: "sigma" > 2.5)
      .crit(lambda: "sigma" > 3.0)
      .log('/tmp/alerts.log')
```

{"id":"Task:cpu Host:folio CPU:cpu3","message":"Task:cpu Host:folio CPU:cpu3 is WARNING cpu-95th:9.847715736036818","details":...

Sigma > 0.5

Kapacitor: batch

```
batch
    |query('''
      SELECT used_percent
      FROM "telegraf"."default"."disk"
      WHERE path = "/"
    ''')
      .period(5m)
      .every(5m)
      .groupBy(time(1m))
    |alert()
      .crit(lambda: "value"  > 95)
```

```
kapacitor define -name batch_disk_alert -type batch -tick batch_disk_alert.tick -dbrp telegraf.default
```

Kapacitor: batch

```
var bps = batch
    |query(' select 8*derivative(mean("bytes_recv"),1s)
          FROM "telegraf"."default"."net"
          WHERE time > now() - 1d  GROUP BY time(1m)  ')
        .period(1m)
        .every(1m)
         .groupBy(time(1m), *)
        .fill(0)

bps
        |influxDBOut()
        .database('bitrate')
        .measurement('bps_in')
```

```
> select * from bps_in
name: bps_in
------------
time                    derivative          host   interface
1460841420000000000     158.4               folio  wlo1
1460841480000000000     274.93333333333334  folio  wlo1
```

Fine

GRAZIE per la lunga attenzione

giovanni.cesaroni@garr.it