

An INFSCI 2711 Lab
by
Yichi Zhang
Quan Zhou

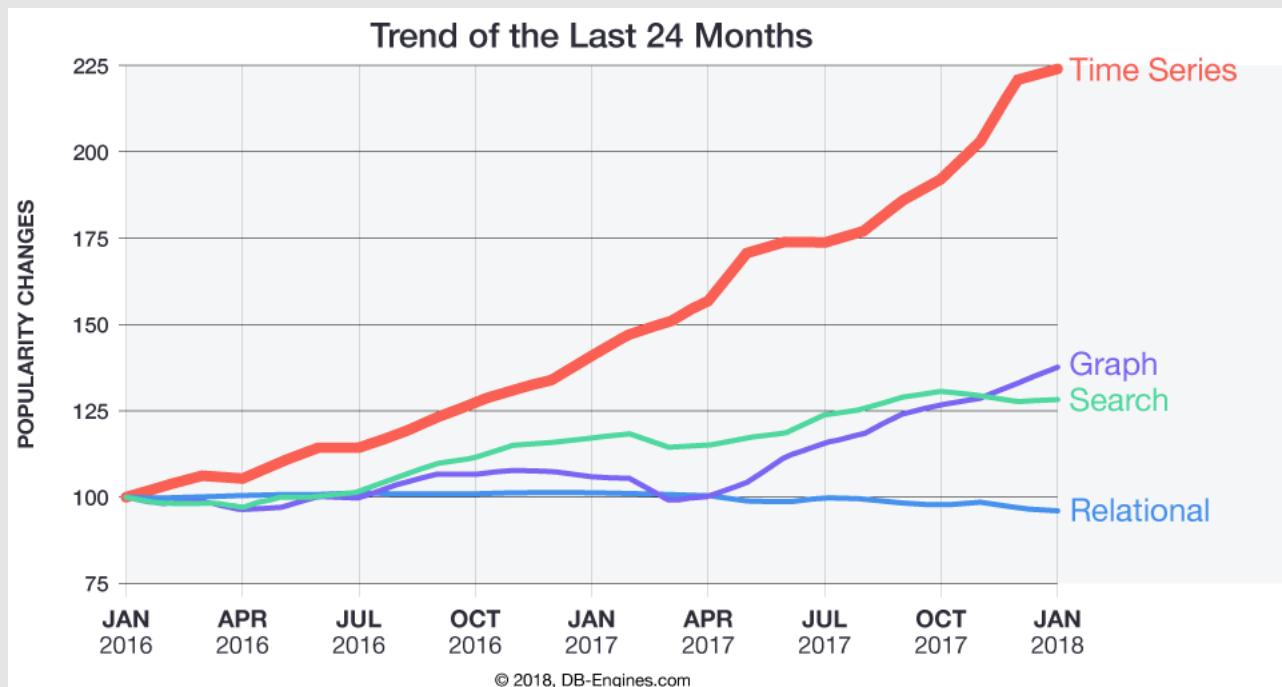
InfluxDB

A State of the Art
Time Series Database

Outline

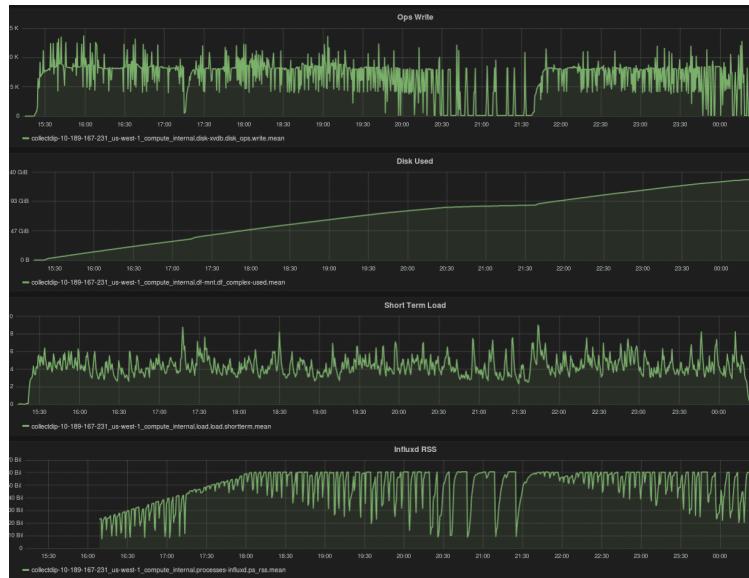
- Intro to Time Series Database
- Intro to InfluxDB
- Installation
- Basic Operations
- Data Analysis with Grafana

Trending in the Database Industry



Time Series Database (TSDB)

- A database optimized for time-stamped or time series data
 - Server metrics
 - Financial data
 - Weather data
 - Industrial machine data
 - Auto-piloting cars

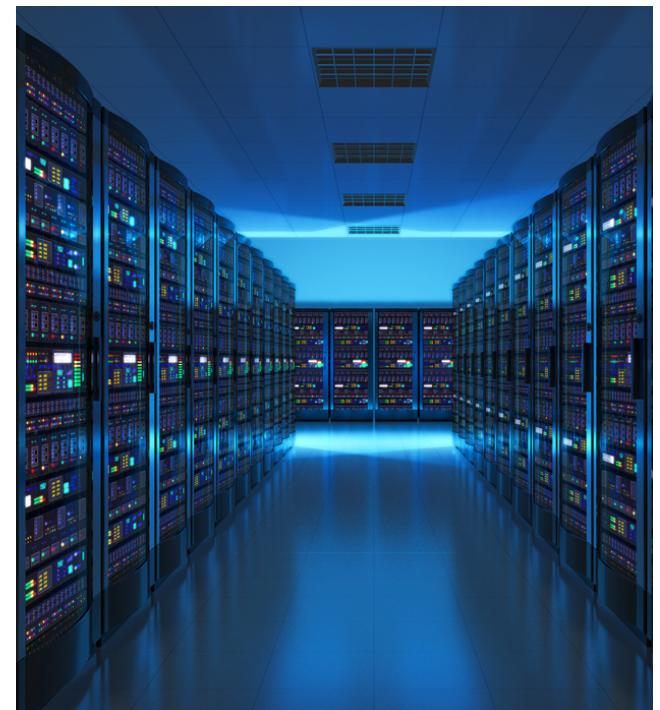


Why TSDB?

- Relational databases are originally designed for OLTP (e.g. bank transactions)
- Strong and useful features of RDBMS
 - Secondary index support
 - Rich query language
 - Multi-table JOINs
- The problems of RDBMS
 - **SCALING**
 - **PERFORMANCE**

Why TSDB?

- Time-series data accumulates very quickly
 - A auto-piloting car collects 25GB of data per hour
 - A 10,000 machine datacenter easily generates 1,000,000,000 system status data points per day



RDBMS vs. TSDB

- **OLTP Writes**

- Primarily UPDATES
- Randomly distributed (over the set of primary keys)
- Often transactions across multiple primary keys

- **Time-series Writes**

- Primarily INSERTS
- Primarily to a recent time interval
- Primarily associated with both a timestamp and a separate primary key (e.g., server ID, device ID, security/account ID, vehicle/asset ID, etc.)

Popular TSDBs

- **InfluxDB**
 - Written in Go
- **OpenTSDB**
 - Built on top of HBase
- **KDB+**
 - Column-based relational TSDB with in-memory capabilities
- **Graphite**
 - Data logging and graphing tool for time series data, numeric data only

Why InfluxDB

- **Most popular Time-series database**

Rank			DBMS
Mar 2018	Feb 2018	Mar 2017	
1.	1.	1.	InfluxDB 
2.	2.	↑ 5.	Kdb+ 
3.	3.	↓ 2.	RRDtool
4.	4.	↓ 3.	Graphite
5.	5.	↓ 4.	OpenTSDB
6.	↑ 7.	↑ 7.	Prometheus
7.	↓ 6.	↓ 6.	Druid
8.	8.	8.	KairosDB
9.	9.	9.	eXtremeDB 
10.	10.	↑ 12.	Riak TS

- **Widely deployed in the industry**
 - <https://www.influxdata.com/customers/testimonials/>
- **Excellent performance**
 - Top 3 in all categories
 - Write performance
 - Query speed
 - Data compression (Storage)
- **Easiness of use**
 - SQL like queries
 - Best functionality
- **Best Overall Score**

Installation & Starting InfluxDB

- Mac OS
 - Install Homebrew (if you don't already have it)
 - `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
 - Update Homebrew repo and install InfluxDB
 - `brew update`
 - `brew install influxdb`
- Windows
 - Download and unzip the package from
 - https://dl.influxdata.com/influxdb/releases/influxdb-1.5.0_windows_amd64.zip
- Getting Started
 - From terminal / cmd, start the service with
 - `influxd`
 - Open another window and connect to the shell with
 - `influx -precision rfc3339`

Install on Windows OS

- Unzip 'influxdb-1.5.0_windows_amd64.zip'
- Change 'influxdb.conf' if you would like to (e.g. Database location)
- Start 'influxd.exe'

```
24 [meta]
25   # Where the metadata/raft database is stored
26   dir = "/var/lib/influxdb/meta"
27
28   # Automatically create a default retention database
29   # retention-autocreate = true
30
31   # If log messages are printed for the store
32   # logging-enabled = true
33
34 #####
35 #### [data]
36 #####
37 #### Controls where the actual shard data is stored
38 #### flushed from the WAL. "dir" may need to be
39 #### for your system, but the WAL setting should work for most systems
40 #### defaults should work for most systems
41 #####
42
43 [data]
44   # The directory where the TSM storage is located
45   dir = "/var/lib/influxdb/data"
46
47   # The directory where the TSM storage is located
48   wal-dir = "/var/lib/influxdb/wal"
```

influx.exe	3/6/2018 5:52 PM	Application
influx_inspect.exe	3/6/2018 5:52 PM	Application
influx_stress.exe	3/6/2018 5:52 PM	Application
influx_tsm.exe	3/6/2018 5:52 PM	Application
influxd.exe	3/6/2018 5:52 PM	Application
influxdb.conf	3/6/2018 5:52 PM	CONF File

Start this

```
E:\MyDesktop\influxdb-1.5.0-1\influxd.exe
```

You are good to go!

```
8888888 .d888 888 8888888b. 888888b.
888 d88P' 888 888 "88b 888 "88b
888 888 888 888 888 888 888 888 888
888 88888b. 888888 888 888 888 888 888 888888K.
888 888 "88b 888 888 888 888 Y8bdP' 888 888 888 "88b
888 888 888 888 888 888 X88K 888 888 888 888
888 888 888 888 888 Y88b 888 .d88P 888 d88P
8888888 888 888 888 888 "Y88888 888 888 8888888P" 8888888P"
```

```
2018-03-21T06:21:06.05866Z info InfluxDB starting {"log_id": "06yoLoeG000", "version": "1.5.0", "branch": "1.5", "commit": "6ac835404e7e04ea7299a0eebcce1ablef15fe3c"}
2018-03-21T06:21:06.087842Z info Go runtime {"log_id": "06yoLoeG000", "version": "go1.9.2", "maxprocs": 4}
2018-03-21T06:21:06.837125Z info Using data dir {"log_id": "06yoLoeG000", "service": "store", "path": "C:\Users\CptTony\influxdb\data"}
2018-03-21T06:21:06.838126Z info Open store (start) {"log_id": "06yoLoeG000", "service": "store", "trace_id": "06yoL3aW000", "op_name": "tsdb_open", "op_event": "start"}
2018-03-21T06:21:06.839129Z info Open store (end) {"log_id": "06yoLoeG000", "service": "store", "trace_id": "06yoL3aW000", "op_name": "tsdb_open", "op_event": "end", "op_elapsed": "1.003ms"}
2018-03-21T06:21:06.839129Z info Opened service {"log_id": "06yoLoeG000", "service": "subscriber"}
2018-03-21T06:21:06.840135Z info Starting monitor service {"log_id": "06yoLoeG000", "service": "monitor"}
2018-03-21T06:21:06.840135Z info Registered diagnostics client {"log_id": "06yoLoeG000", "service": "monitor", "name": "build"}
2018-03-21T06:21:06.840135Z info Registered diagnostics client {"log_id": "06yoLoeG000", "service": "monitor", "name": "runtime"}
2018-03-21T06:21:06.840135Z info Registered diagnostics client {"log_id": "06yoLoeG000", "service": "monitor", "name": "network"}
```

Basic Terminologies of InfluxDB

MySQL	InfluxDB
database	database
table	measurement
row	point / series
index	tag (string)
column	field (string boolean integer float)

Basic Operations

- Creating a database
 - `CREATE DATABASE Test`
 - `SHOW DATABASES`
- Specifying a database for operations
 - `USE Test`
- InfluxDB has no strict schema, so there is no **CREATE MESUREMENT** statement

```
[> CREATE DATABASE Test
[> SHOW DATABASES
  name: databases
    name
    ----
    _internal
    Test
  >
```

Data Insertion

- Insert data

- INSERT server,region=us_east,host=server_A cpu=0.65,mem=2335
- INSERT server,region=us_east,host=server_A cpu=0.53 15210288000000000000
- INSERT server,region=us_east cpu=0.43,mem=3532 15210288000000000000
- INSERT server,region=us_east,host=server_B cpu=0.33,mem=3323 15210285000000000000

measurement

tags

fields

timestamp (in nanoseconds)

- If no timestamp is specified, the current system time will be used.
- All the syntax (the commas and whitespaces indicating measurement, tags and fields) in the INSERT query should be strictly followed!

Data Insertion

```
> CREATE DATABASE Test
> USE Test
Using database Test
[> INSERT server,region=us_east,host=server_A cpu=0.65,mem=2335
> INSERT server,region=us_east,host=server_A cpu=0.53 15210288000000000000
> INSERT server,region=us_east cpu=0.43,mem=3532 15210288000000000000
> INSERT server,region=us_east,host=server_B cpu=0.33,mem=3323 15210285000000000000
> SELECT * FROM server
name: server
      time          cpu   host     mem  region
-----  ---  ----  ---  -----
2018-03-14T11:55:00Z        0.33 server_B 3323 us_east
2018-03-14T12:00:00Z        0.43           3532 us_east
2018-03-14T12:00:00Z        0.53 server_A    us_east
2018-03-22T01:12:12.136057115Z 0.65 server_A 2335 us_east
>
```

Data Exploration

- Import sample data
 - The air quality measure of 5 Chinese cities from 2010 to 2015 (<https://archive.ics.uci.edu/ml/datasets/PM2.5+Data+of+Five+Chinese+Cities>)
 - Run the shell script provided in terminal: `sh import.sh`
- Data schema exploration
 - `USE air`
 - `SHOW MEASUREMENTS`
 - `SHOW TAG KEYS`
 - `SHOW FIELD KEYS`

```
> USE air
Using database air
>
[> SHOW MEASUREMENTS
name: measurements
name
-----
pm
[> SHOW TAG KEYS
name: pm
tagKey
-----
city
[>
[> SHOW FIELD KEYS
name: pm
fieldKey          fieldType
-----          -----
cumulative_precipitation float
humidity          float
pm                float
precipitation    float
pressure          float
temperature      float
wind_direction   string
wind_speed        float
> |
```

Data Exploration

- Simple SELECT Queries

- Select the first 5 records of Beijing
- `SELECT * FROM pm WHERE city = 'Beijing' LIMIT 5`

```
> SELECT * FROM pm WHERE city='Beijing' LIMIT 5
name: pm
time          city    cumulative_precipitation humidity pm precipitation pressure temperature wind_direction wind_speed
---          ---
2009-12-31T16:00:00Z Beijing 0                  43      0        1021     -11      NW       1.79
2009-12-31T17:00:00Z Beijing 0                  47      0        1020     -12      NW       4.92
2009-12-31T18:00:00Z Beijing 0                  43      0        1019     -11      NW       6.71
2009-12-31T19:00:00Z Beijing 0                  55      0        1019     -14      NW       9.84
2009-12-31T20:00:00Z Beijing 0                  51      0        1018     -12      NW      12.97
>
```

- Select all the records of Shanghai where PM2.5 exceeds 600
- `SELECT pm FROM pm WHERE city = 'Shanghai' AND pm > 600`

```
> SELECT pm FROM pm WHERE city='Shanghai' AND pm>600
name: pm
time          pm
---          --
2011-12-31T07:00:00Z 730
2012-02-19T22:00:00Z 650
> █
```

Data Exploration

- Simple SELECT Queries

- Select the first 5 PM2.5 and humidity of Guangzhou starting from 01/01/2015
 - `SELECT pm, humidity FROM pm WHERE city = 'Guangzhou' AND time >= '2015-01-01' LIMIT 5`

```
[> SELECT pm, humidity FROM pm WHERE city = 'Guangzhou' AND time >= '2015-01-01' LIMIT 5
  name: pm
    time          pm          humidity
    ----          --          -----
2015-01-01T00:00:00Z 51          68
2015-01-01T01:00:00Z 48          53
2015-01-01T02:00:00Z 49          42
2015-01-01T03:00:00Z 45.666666666666664 32
2015-01-01T04:00:00Z 42.333333333333336 31
>
```

- **Important:** Tag values and field string values in WHERE clause should be wrapped by SINGLE QUOTES

Data Exploration

- OLAP queries
 - Return the mean and max of PM2.5 by each city from all time
 - `SELECT max(pm), mean(pm) FROM pm GROUP BY city`

```
> SELECT max(pm), mean(pm) FROM pm GROUP BY city
name: pm
tags: city=Beijing
time          max  mean
----  -----
1970-01-01T00:00:00Z 994 95.18355527059914

name: pm
tags: city=Chengdu
time          max          mean
----  ---  -----
1970-01-01T00:00:00Z 600.6666666666666 79.8612003584463

name: pm
tags: city=Guangzhou
time          max  mean
----  -----  -----
1970-01-01T00:00:00Z 387 51.13099411700067

name: pm
tags: city=Shanghai
time          max  mean
----  -----  -----
1970-01-01T00:00:00Z 730 54.838865737414494

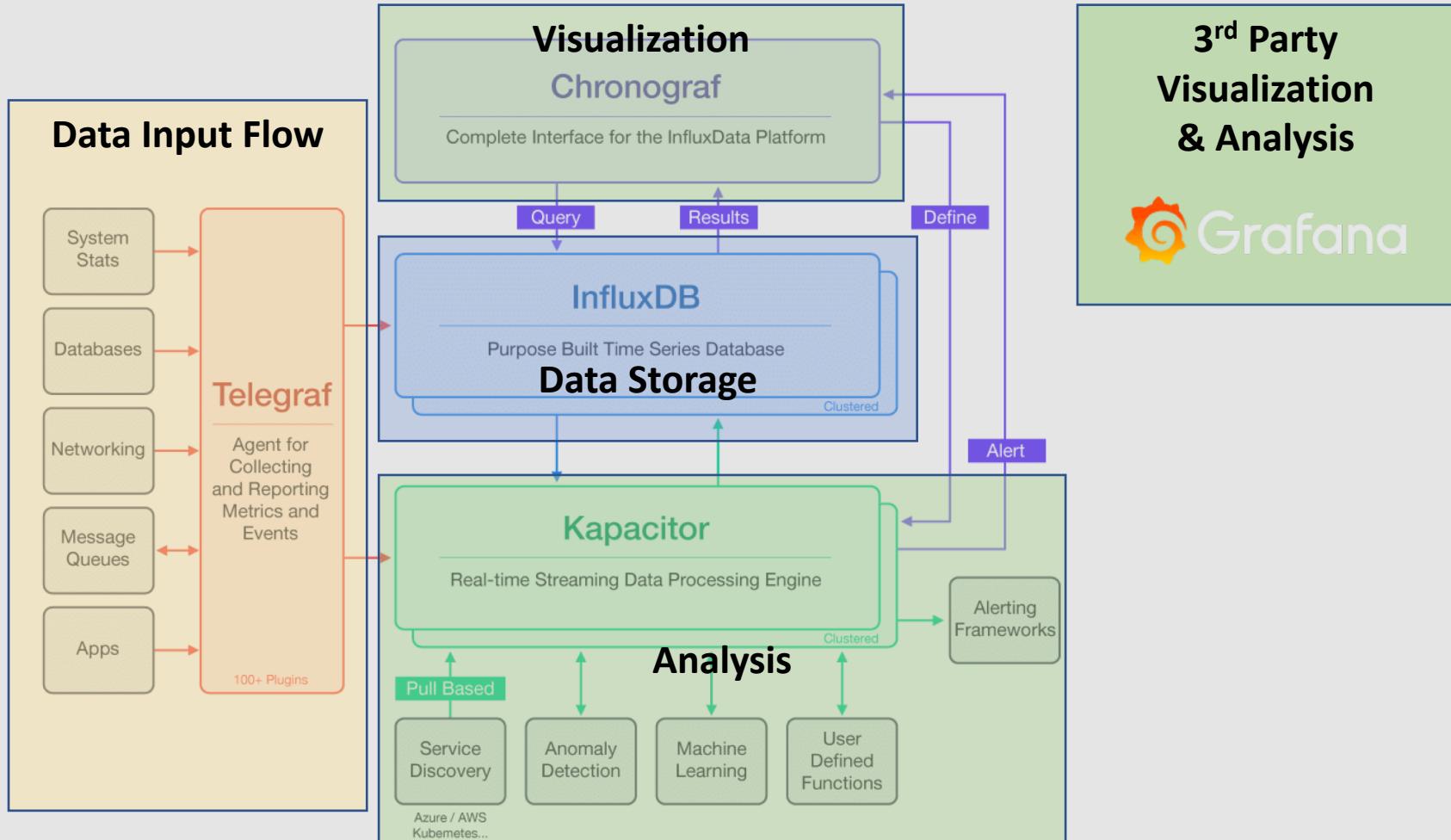
name: pm
tags: city=Shenyang
time          max          mean
----  ---  -----
1970-01-01T00:00:00Z 1134.6666666666667 78.30155817397484
> █
```

Data Exploration

- OLAP queries (GROUP BY time interval)
 - Return the mean of PM2.5 and precipitation in Beijing for every 2 weeks in the year of 2014
 - `SELECT mean(pm), mean(precipitation) FROM pm WHERE time >= '2014-01-01' AND time < '2015-01-01' AND city='Beijing' GROUP BY time(7d)`

```
> SELECT mean(pm), mean(precipitation) FROM pm WHERE time >= '2014-01-01'  
  AND time < '2015-01-01' AND city='Beijing' GROUP BY time(7d)  
name: pm  
time          mean        mean_1  
----          ----        -----  
2013-12-26T00:00:00Z 82.625      0  
2014-01-02T00:00:00Z 93.55357142857143  0  
2014-01-09T00:00:00Z 124.92063492063492  0  
2014-01-16T00:00:00Z 121.3640873015873  0  
2014-01-23T00:00:00Z 95.90178571428572  0  
2014-01-30T00:00:00Z 81.47321428571429  0  
2014-02-06T00:00:00Z 79.21974206349208  0.03392857142857143  
2014-02-13T00:00:00Z 196.41964285714286  0  
2014-02-20T00:00:00Z 296.9831349206349  0.006547619047619048  
2014-02-27T00:00:00Z 88.57837301587301  0  
2014-03-06T00:00:00Z 90.47172619047619  0  
2014-03-13T00:00:00Z 63.40029761904762  0  
2014-03-20T00:00:00Z 158.99255952380952 0  
2014-03-27T00:00:00Z 105.42063492063491 0.0035714285714285713  
2014-04-03T00:00:00Z 80.670138888888889  0
```

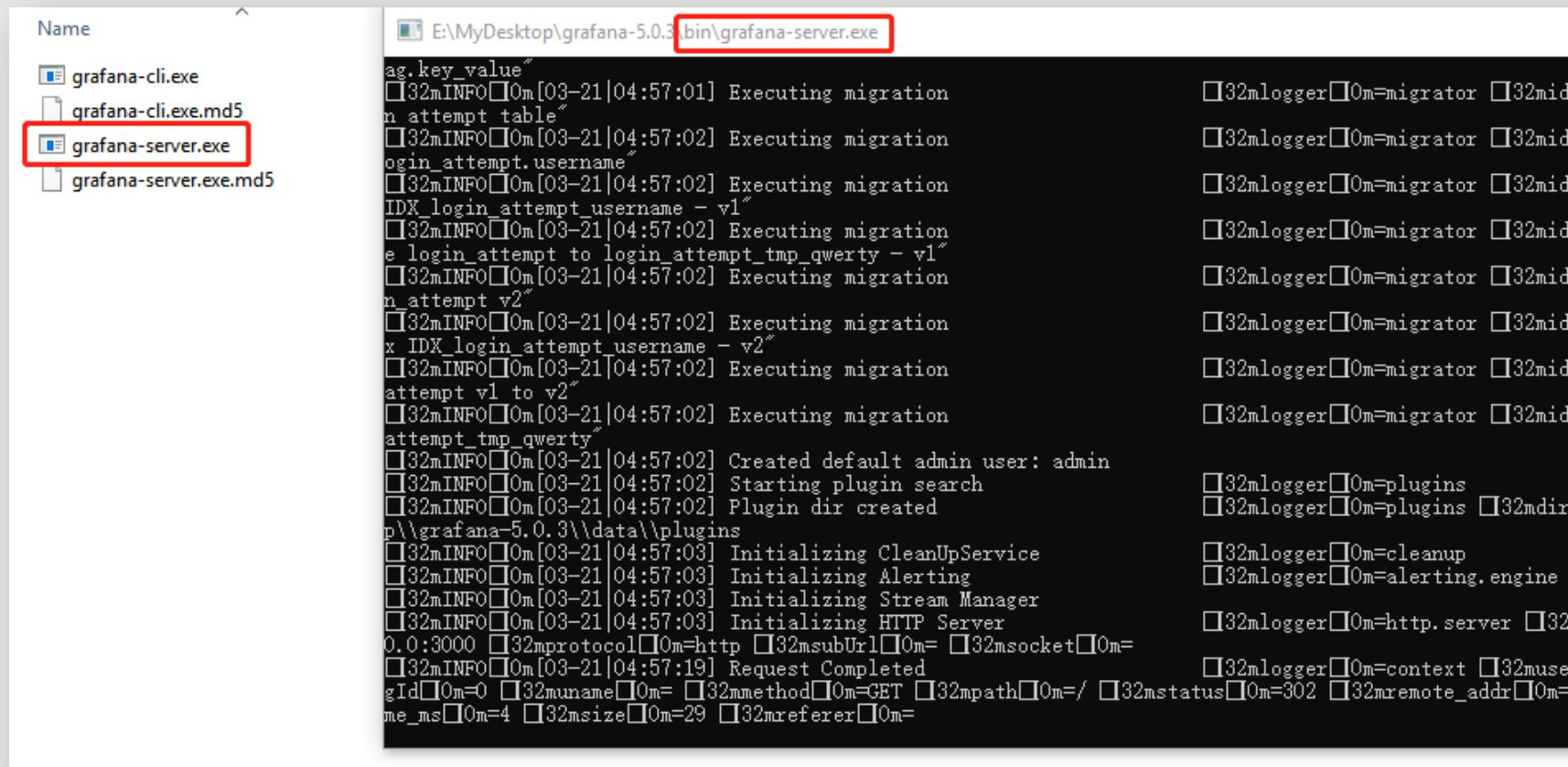
Influx Data PLATFORM



Grafana Installation

- Mac OS
 - Installation
 - `brew update`
 - `brew install Grafana`
 - Startup
 - `brew services start Grafana`
- Window
 - Download Package
 - <http://docs.grafana.org/installation/windows/>
 - Unzip & Run
 - `bin/grafana-server.exe`

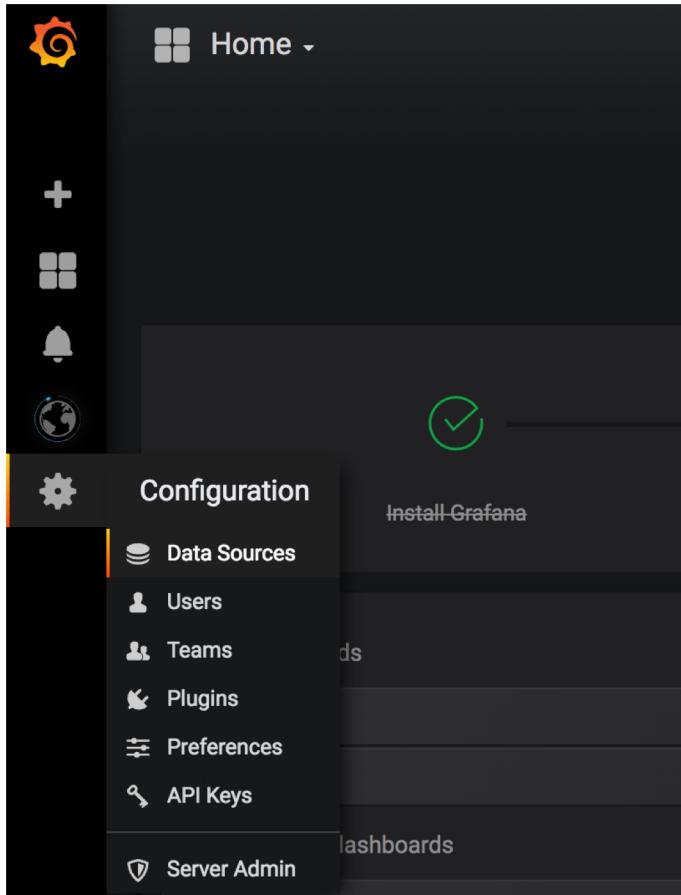
Grafana on Windows



The screenshot shows a Windows File Explorer window on the left and a terminal window on the right. The File Explorer window displays files in the directory `E:\MyDesktop\grafana-5.0.3\bin`. The file `grafana-server.exe` is selected and highlighted with a red box. The terminal window shows the command-line output of the application, starting with migration logs and followed by various initialization and service logs.

```
E:\MyDesktop\grafana-5.0.3\bin\grafana-server.exe

ag.key_value"
[32mINFO[0m[03-21|04:57:01] Executing migration
n_attempt table"
[32mINFO[0m[03-21|04:57:02] Executing migration
login_attempt.username"
[32mINFO[0m[03-21|04:57:02] Executing migration
IDX_login_attempt_username - v1"
[32mINFO[0m[03-21|04:57:02] Executing migration
e_login_attempt to login_attempt_tmp_qwerty - v1"
[32mINFO[0m[03-21|04:57:02] Executing migration
n_attempt v2"
[32mINFO[0m[03-21|04:57:02] Executing migration
x_IDX_login_attempt_username - v2"
[32mINFO[0m[03-21|04:57:02] Executing migration
attempt v1 to v2"
[32mINFO[0m[03-21|04:57:02] Executing migration
attempt_tmp_qwerty"
[32mINFO[0m[03-21|04:57:02] Created default admin user: admin
[32mINFO[0m[03-21|04:57:02] Starting plugin search
[32mINFO[0m[03-21|04:57:02] Plugin dir created
p:\\grafana-5.0.3\\data\\plugins
[32mINFO[0m[03-21|04:57:03] Initializing CleanUpService
[32mINFO[0m[03-21|04:57:03] Initializing Alerting
[32mINFO[0m[03-21|04:57:03] Initializing Stream Manager
[32mINFO[0m[03-21|04:57:03] Initializing HTTP Server
0.0.3000 [32mprotocol[0m=http [32msubUrl[0m= [32msocket[0m=
[32mINFO[0m[03-21|04:57:19] Request Completed
gId[0m=0 [32muname[0m= [32mmethod[0m=GET [32mpath[0m=/ [32mstatus[0m=302 [32mremote_addr[0m=
me_ms[0m=4 [32msize[0m=29 [32mreferer[0m=
```



Visualize Data with Grafana

- Access the web interface
 - localhost:3000
 - Login with admin/admin

 **Data Sources / air**
Type: InfluxDB

Settings

Name	air	i	Default	<input type="checkbox"/>
Type	InfluxDB	▼		

HTTP

URL	http://localhost:8086	i
Access	direct	▼ i

Auth

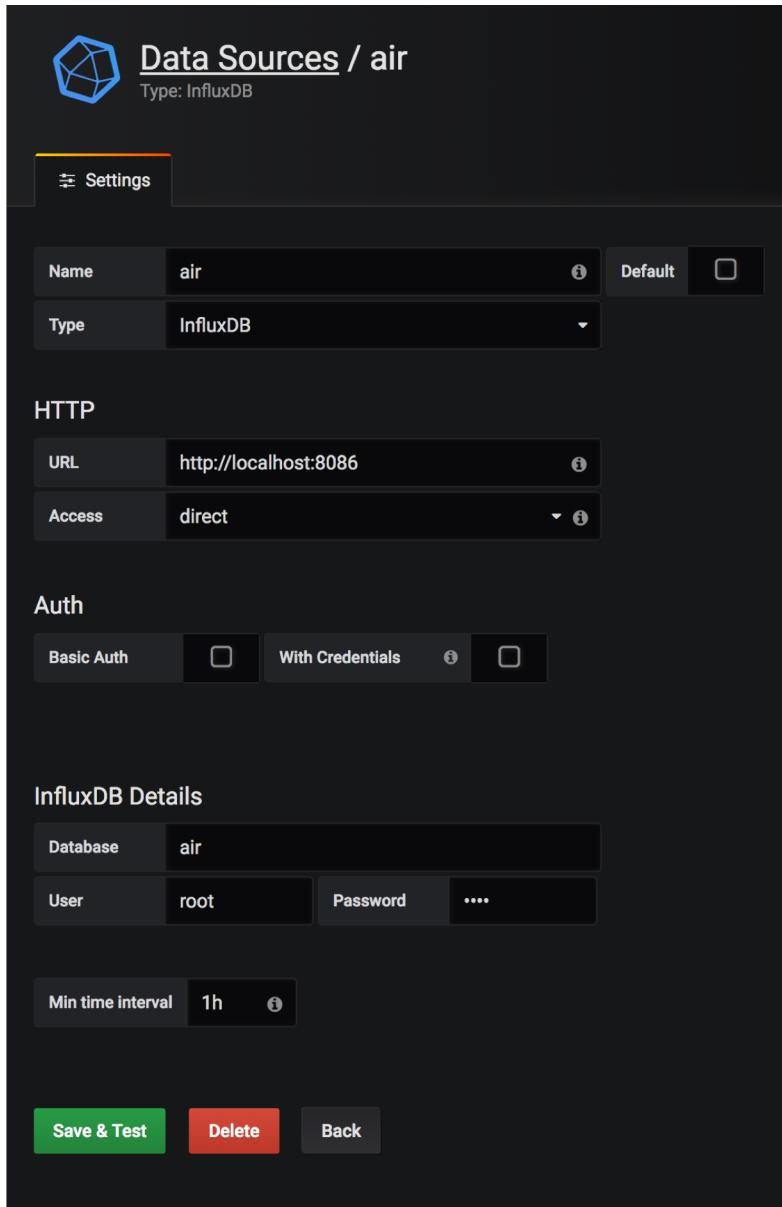
Basic Auth	<input type="checkbox"/>
With Credentials	i <input type="checkbox"/>

InfluxDB Details

Database	air		
User	root	Password	****

Min time interval [i](#)

Save & Test **Delete** **Back**

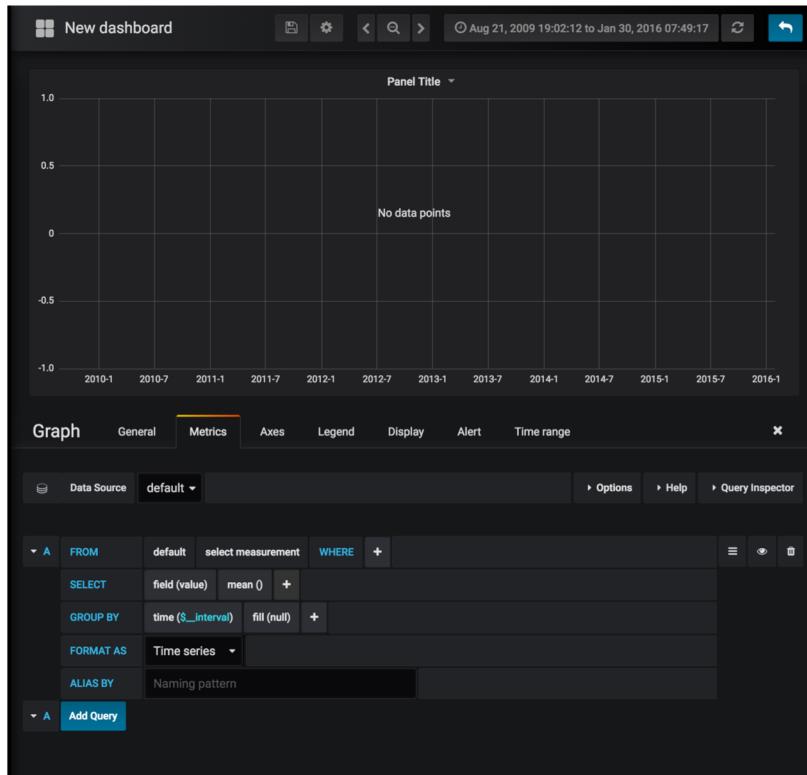
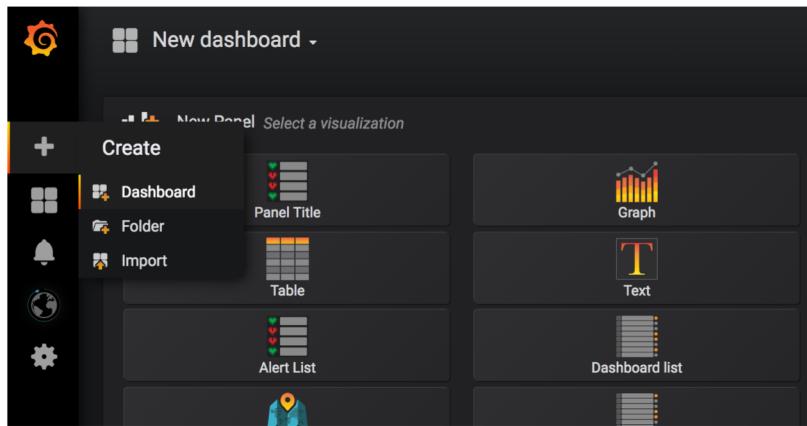


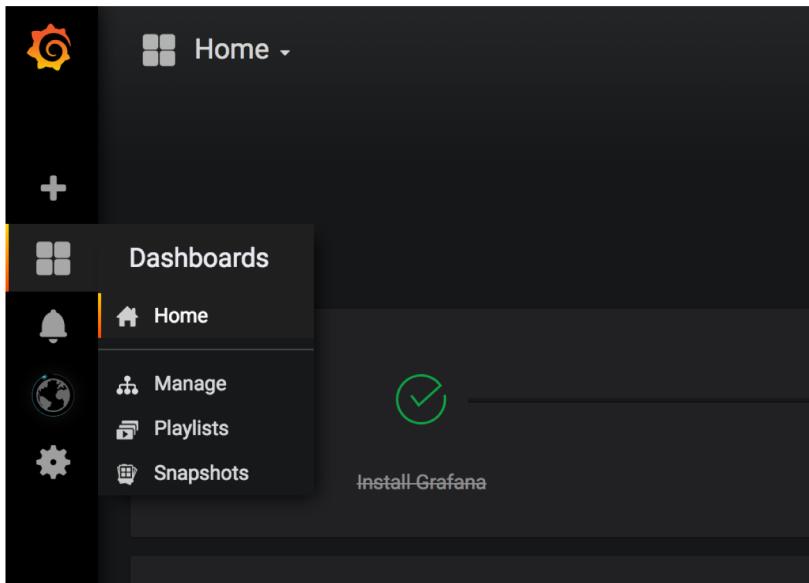
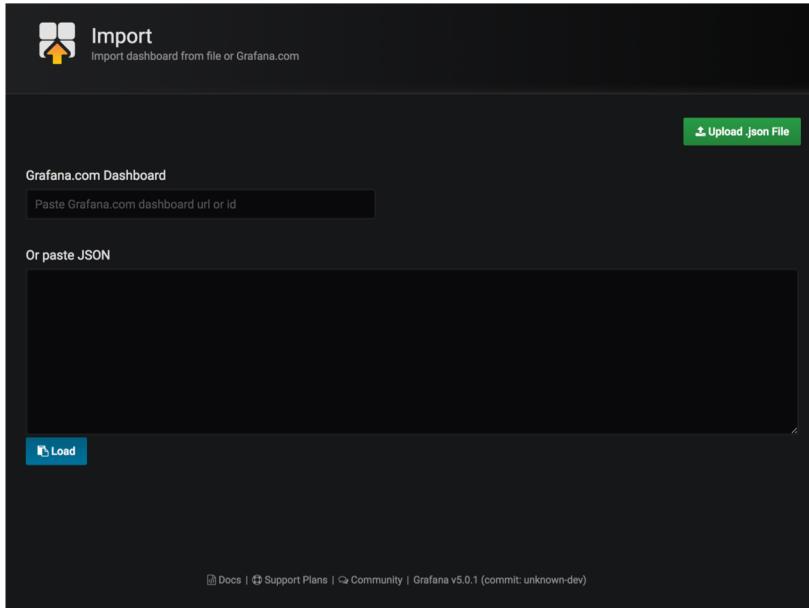
Visualize Data with Grafana

- Access the web interface
 - localhost:3000
 - Login with admin/admin
- Set up data source
 - User: [root](#)
 - Password: [root](#)
 - Database: [air](#)

Visualize Data with Grafana

- Access the web interface
 - localhost:3000
 - Login with admin/admin
- Set up data source
 - User: [root](#)
 - Password: [root](#)
 - Database: [air](#)
- Create a new dashboard
 - Add a new graph panel
 - Start editing panel

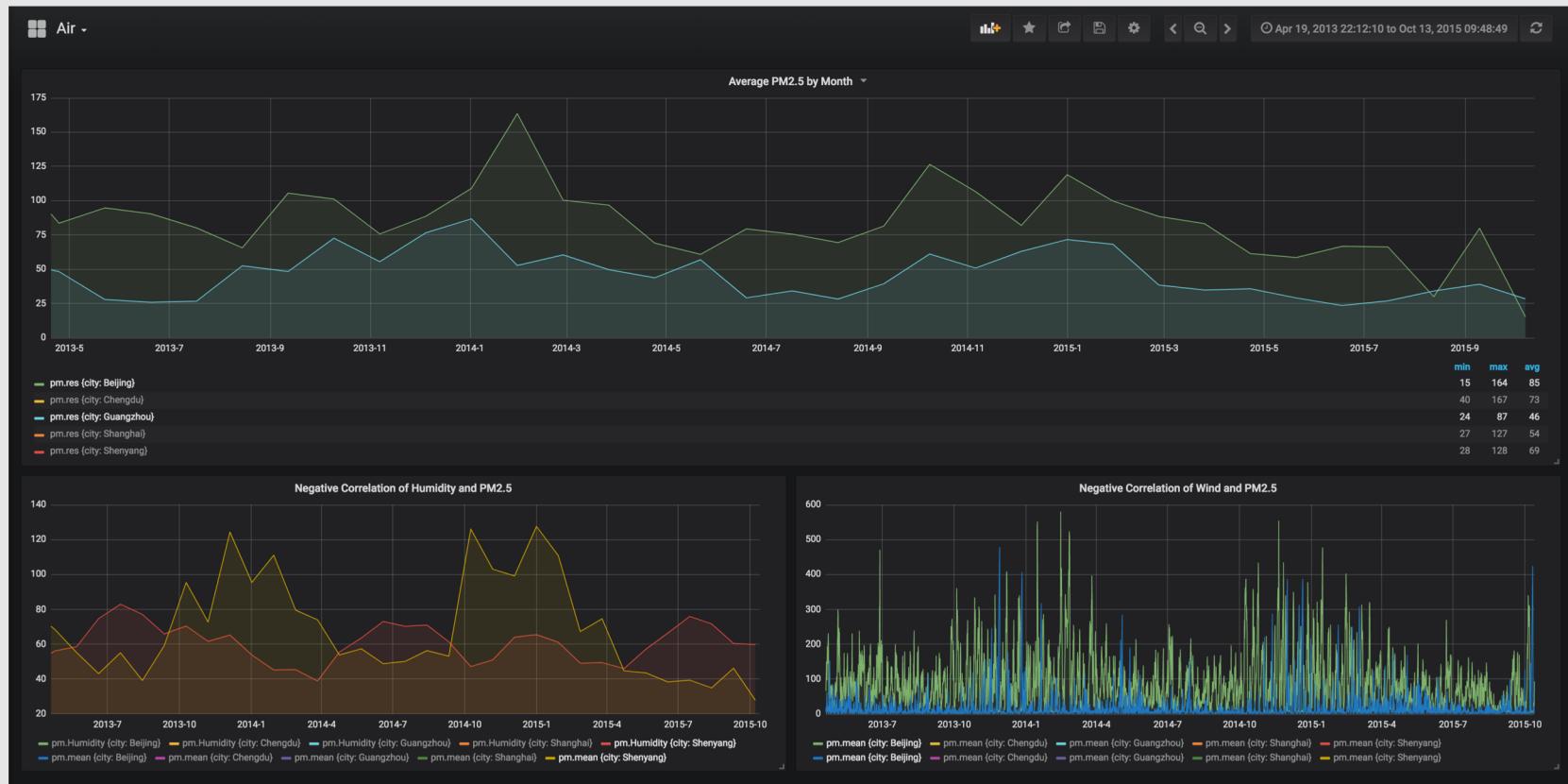




Visualize Data with Grafana

- Access the web interface
 - localhost:3000
 - Login with admin/admin
- Set up data source
 - User: [root](#)
 - Password: [root](#)
 - Database: [air](#)
- Create a new dashboard
 - Add a new graph panel
 - Start editing panel
- Import dashboard from a JSON file
 - Import [air_dashboard.json](#)

Visualize Data with Grafana



Advanced Functionality - Retention Policy (RP)

- Manages how long InfluxDB keeps the data on certain measurement.
- After data expires duration set in retention policy, InfluxDB automatically drops the data.
- Every time a database is created, InfluxDB generates a default RP called *autogen*, which has no duration time (data never dropped)
- Create a user defined RP
 - [CREATE DATABASE realtime](#)
 - [USE realtime](#)
 - [CREATE RETENTION POLICY "one_hour" ON "realtime" DURATION 1h REPLICATION 1](#)
 - [SHOW RETENTION POLICIES](#)

```
> CREATE DATABASE realtime
> USE realtime
Using database realtime
> CREATE RETENTION POLICY "one_hour" ON "realtime" DURATION 1h REPLICATION 1
> SHOW RETENTION POLICIES
name      duration shardGroupDuration replicaN default
-----
autogen    0s        168h0m0s          1       true
one_hour  1h0m0s   1h0m0s            1       false
> █
```

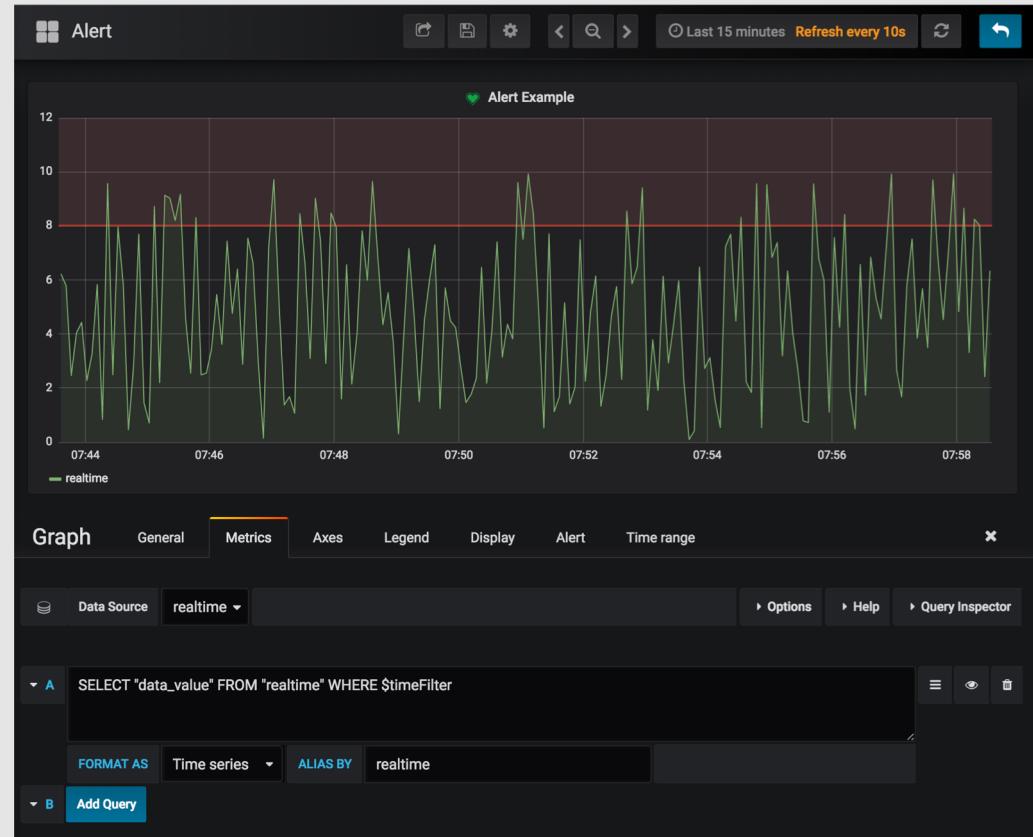
Advanced Functionality - Continuous Query (CQ)

- A query that runs automatically and periodically within a database.
- Normally used for automatically down sampling data.
- Continuous queries require a function in the SELECT clause and must include a GROUP BY time() clause.
- Create a CQ
 - `CREATE CONTINUOUS QUERY "cq_one_min" ON "realtime" BEGIN SELECT mean(data_value) INTO realtime.autogen.realtime_mean FROM realtime.one_hour.realtime GROUP BY time(1m) END`
 - `SHOW CONTINUOUS QUERIES`

```
CREATE CONTINUOUS QUERY "cq_one_min" ON "realtime"
BEGIN
    SELECT mean(data_value)
        INTO realtime.autogen.realtime_mean
        FROM realtime.one_hour.realtime
        GROUP BY time(1m)
END
```

Monitor Realtime Data with Grafana

- Run realtime data insert script
 - pip install influxdb
 - python realtime.py
- Create a new data source that links to the database `realtime`
- Create a new dashboard
- Create panel with query
 - `SELECT "data_value" FROM "realtime" WHERE $timeFilter`



Alerting with Grafana

- Edit the Alert tab and be notified by mail if certain criteria has been met

Graph General Metrics Axes Legend Display Alert Time range

Alert Config

Notifications (1) Name alert Evaluate every 30s

State history Conditions

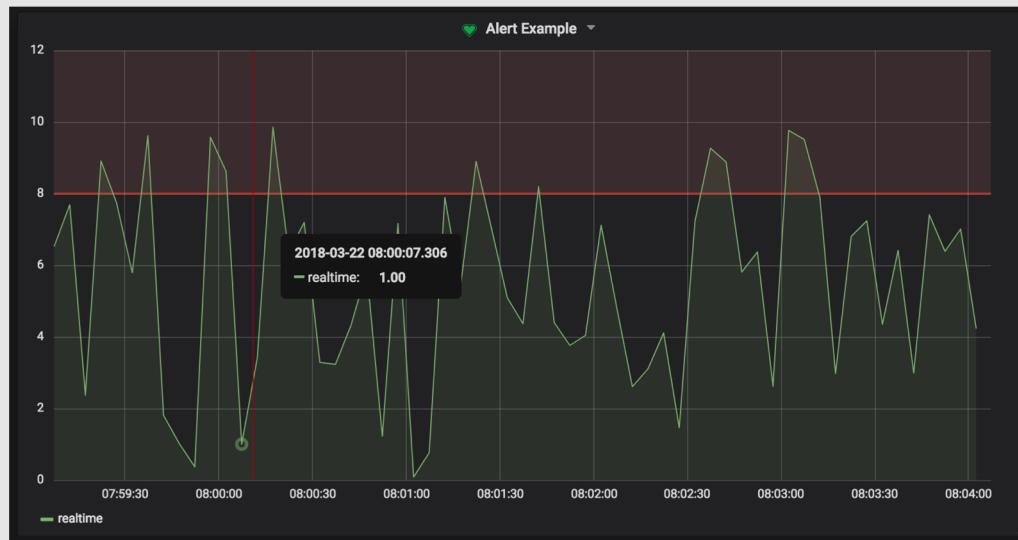
Delete WHEN avg () OF query (A, 30s, now) IS ABOVE 8

If no data or all values are null SET STATE TO No Data

If execution error or timeout SET STATE TO Alerting

Test Rule

The screenshot shows the 'Alert' tab in Grafana's configuration interface. It displays an alert rule named 'alert'. The 'Conditions' section contains a query 'avg () OF query (A, 30s, now)' set to 'IS ABOVE' a value of 8. There are two state transition rules: one for 'No Data' and another for 'Alerting' in case of an execution error or timeout. A 'Test Rule' button is also present.



References

- https://docs.influxdata.com/influxdb/v1.5/introduction/getting_started/
- <http://docs.grafana.org/>
- <https://blog.timescale.com/time-series-data-why-and-how-to-use-a-relational-database-instead-of-nosql-d0cd6975e87c>
- <https://blog.timescale.com/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563>