

毛台

整齐胜于凌乱, 简单胜于复杂. 果断胜于凌乱, 依赖胜于独立, 强大胜于彷徨, 自信胜于

博客园

首页

订阅

管理

公告

昵称: \_毛台  
园龄: 3年2个月  
粉丝: 19  
关注: 4  
[+加关注](#)

搜索

找找看

常用链接

[我的随笔](#)  
[我的评论](#)  
[我的参与](#)  
[最新评论](#)  
[我的标签](#)

友链

[51cto](#)  
[52devops](#)  
[yangrong](#)  
[youku](#)  
[卑鄙的我](#)  
[惨绿少年](#)  
[陈明乾](#)  
[陈思齐](#)  
[骏马金龙\(总结的文档很全面\)](#)  
[七夜的故事\(Python爬虫开发与项目实战作者\)](#)

随笔-393 文章-4 评论-9

# [svc]influxdb最佳实战-监控对比

目录(?)

[+]

最近在搞容器的监控,遇到influxdb这个库,搞了两天,些许明白了些套路,做个记录,备忘....

小结如下:

influxdb go语言编写

默认情况influxdb创建的库关联autogen的RP(存储策略),即数据会保留永久

## 监控和日志的区别

最近搞监控,所谓监控就是监控服务肉体是否健康(还活着/生病? 各项指标是否正常?)

区分日志搜集: 分析服务的精神状态是健康(服务的一个履历/日记)

## 如何做一个监控

参考: <https://segmentfault.com/a/1190000011082379>

回想到如果是你自己去做一个监控,能够做到记录每分钟 CPU 的空闲率是多少,要怎么做?

搞一个数据库, 用来放数据的  
写一个脚本, 用来获取 CPU 的相关数据, 加上时间戳, 然后  
创建一个定时任务, 一分钟运行一次脚本  
写一个简单的程序, 从数据库查到数据, 然后根据时间戳, 生

[telegraf搜集器](#) + influxdb(存储) + grafana(展示)

grafana 的套路基本上跟 kibana 差不多, 都是根据查询条件设置聚合规则, 在合适的图表上进行展示, 多个图表共同组建成一个 dashboard, 熟悉 kibana 的用户应该可以非常容易上手。另外 grafana 的可视化功能比 kibana 强得多, 而且 4 以上版本将集成报警功能。

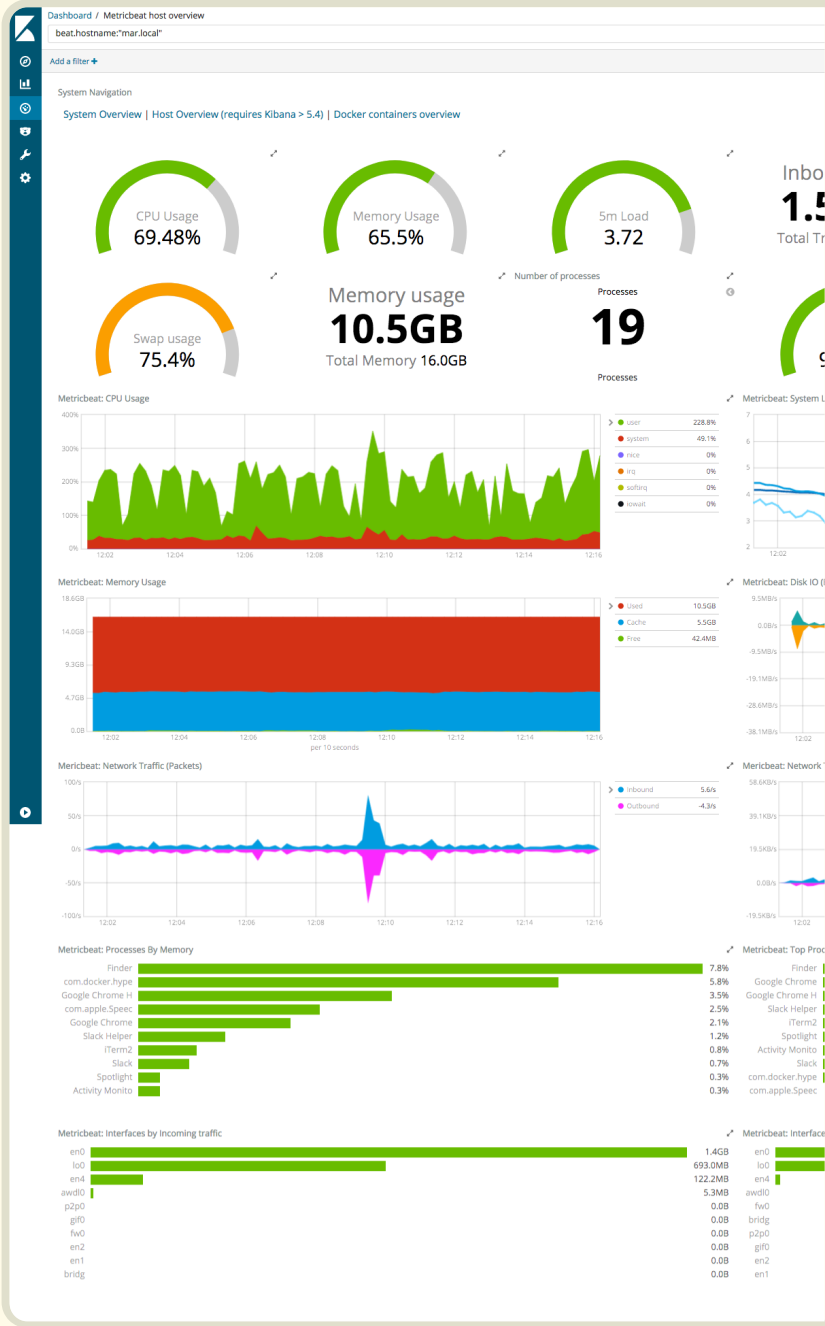
grafana主机监控效果图:

http://www.cnblogs.com/iiiiher/p/8046600.html

1/18



之前用metricbeat做的主机监控效果图-进程级别



监控的对比influxdb vs 普罗

特性对比

Name	Graphite X	InfluxDB X
Description	Data logging and graphing tool for time series data	DBMS for storing time series data
Primary database model	Time Series DBMS	Time Series DBMS
DB-Engines Ranking	Score 2.65 Rank #86 Overall #4 Time Series DBMS	Score 9.87 Rank #40 Overall #1 Time Series DBMS
Trend Chart		
Website	<a href="https://github.com/graphite-project/graphite-web">github.com/graphite-project/graphite-web</a>	<a href="https://www.influxdata.com/time-series/">www.influxdata.com/time-series/</a>
Technical documentation	<a href="https://graphite.readthedocs.io">graphite.readthedocs.io</a>	<a href="https://docs.influxdata.com/influxdb/">docs.influxdata.com/influxdb/</a>
Developer	Chris Davis	

Initial release	2006	2013
Current release		v1.3, July 2017
License	Open Source	Open Source
Cloud-based	no	no
Implementation language	Python	Go
Server operating systems	Linux Unix	Linux OS X
Data scheme	yes	schema-free
Typing	Numeric data only	Numeric data and Strings
XML support	no	no
Secondary indexes	no	no
SQL	no	SQL-like query language
APIs and other access methods	HTTP API Sockets	HTTP API JSON over UDP
Supported programming languages	JavaScript (Node.js) Python	.Net Clojure Erlang Go Haskell Java JavaScript JavaScript (Node.js) Lisp Perl PHP Python R Ruby Rust Scala
Server-side scripts	no	no
Triggers	no	no
Partitioning methods	none	Sharding
Replication methods	none	selectable replication fact
MapReduce	no	no
Consistency concepts	none	
Foreign keys	no	no
Transaction concepts	no	no
Concurrency	yes	yes
Durability	yes	yes
In-memory capabilities		yes
User concepts	no	simple rights managemen

参考: <http://gitbook.cn/books/59395d3d5863cf478e6b50ba/index.html>

influxdb集成已有的概念, 比如查询语法类似sql, 引擎从LSM优  
influxdb支持的类型有float, integers, strings, boolea

influxdb的时间精度是纳秒，prometheus的则是毫秒。  
 influxdb仅仅是个数据库，而prometheus提供的是整套监控解决方案。  
 influxdb支持的math **function**比较少，prometheus相对来说支持更多。  
 influxdb支持event **log**，prometheus不支持。

注: 已上对比的是普罗v1,现在普罗有v2版本了,听说比influxdb更强悍了。  
 而且influxdb集群方案已闭源。

## influxdb的特性和特点

influxdb中文翻译官方的文档,感觉很棒

<https://jasper-zhang1.gitbooks.io/influxdb/content/>

[https://jasper-](https://jasper-zhang1.gitbooks.io/influxdb/content/Concepts/key_concepts.html)

[zhang1.gitbooks.io/influxdb/content/Concepts/key\\_concepts.html](https://jasper-zhang1.gitbooks.io/influxdb/content/Concepts/key_concepts.html)

参考: <http://www.ttlsa.com/monitor-safe/monitor/distributed-time-series-database-influxdb/>

- influxdb 它的特性  
它有三大特性:

1. Time Series (时间序列): 你可以使用与时间有关的相关函数。
2. Metrics (度量): 你可以实时对大量数据进行计算。
3. Events (事件): 它支持任意的事件数据。

时序性 (Time Series): 与时间相关的函数的灵活使用 (例如最值函数)。

度量 (Metrics): 对实时大量数据进行计算;

事件 (Event): 支持任意的事件数据, 换句话说, 任意事件的数据。

- influxdb 它的特点  
参考: <http://dbaplus.cn/news-73-1291-1.html>

schemaless(无结构), 可以是任意数量的列。

无特殊依赖, 几乎开箱即用 (如ElasticSearch需要Java)。

自带数据过期功能;

自带权限管理, 精细到“表”级别;

原生的HTTP支持, 内置HTTP API。

强大的类SQL语法, 支持min, max, sum, count, mean, median等函数。

## influxdb最佳实践

### 1. 登录 建库 查询

参考: [https://jasper-](https://jasper-zhang1.gitbooks.io/influxdb/content/Introduction/getting_start.html)

[zhang1.gitbooks.io/influxdb/content/Introduction/getting\\_start.html](https://jasper-zhang1.gitbooks.io/influxdb/content/Introduction/getting_start.html)

```
influx -precision rfc3339 # -precision参数表明了1
CREATE DATABASE mydb
SHOW DATABASES
USE mydb
INSERT cpu,host=serverA,region=us_west value=0.64
SELECT "host", "region", "value" FROM "cpu"

INSERT temperature,machine=unit42,type=assembly e
SELECT * FROM "temperature"

> SELECT * FROM /.*/ LIMIT 1
> SELECT * FROM "cpu_load_short"
> SELECT * FROM "cpu_load_short" WHERE "value" >
```

2.了解influxdb基本概念

参考: <http://dbaplus.cn/news-73-1291-1.html>

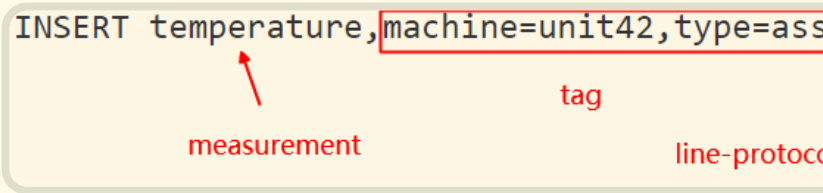
InfluxDB中的名词	传统数据库中的概念
database	数据库
measurement	数据库中的表
points	表里面的一行数据

# InfluxDB中特有的概念

Point相当于传统数据库里的一行数据，如下表所示：  
Point由时间戳（time）、数据（field）、标签（tags）组成。

## line-protocol格式

```
<measurement>[,<tag-key>=<tag-value>...] <field-k
INSERT temperature,machine=unit42,type=assembly e
```



更多如:

```
cpu,host=serverA,region=us_west value=0.64
payment,device=mobile,product=Notepad,method=credit
stock,symbol=AAPL bid=127.46,ask=127.48
temperature,machine=unit42,type=assembly external
```

**Tag: 被索引**  
上面的location和server就是tag key, us和host1是tag value

**Field: value**支持的类型floats, integers, strings, booleans  
上面的temperature是field key, 82是field value。field value支持浮点、整数、字符串、布尔、JSON、二进制、数组

**Timestamp**  
格式是: RFC3339 UTC。默认精确到纳秒, 可选。

**Series:**  
measurement, tag set, retention policy相同的数据集合

**Retention Policy:**  
保留策略包括设置数据保存的时间以及在集群中的副本个数。默认策略是默认的数据库策略

**Continuous Query:**  
CQ是预先配置好的一些查询命令, 定期自动执行这些命令并将查询结果写入新的数据库

**Shard:**  
存储一定时间间隔的数据, 每个目录对应一个shard, 目录的名字是时间戳

## 2.实操如下: 理解 point&measurement&series(field set) (被索引的tag set)

向库中插入如下数据:

属性	值
库名	my_database
measurement	census
field key	butterflies和honeybees
tag key	location和scientist

```
name: census
--
```

time		butterf
2015-08-18T00:00:00Z	12	23
2015-08-18T00:00:00Z	1	3
2015-08-18T00:06:00Z	11	28
2015-08-18T00:06:00Z	3	28
2015-08-18T05:54:00Z	2	1
2015-08-18T06:00:00Z	1	1
2015-08-18T06:06:00Z	8	2
2015-08-18T06:12:00Z	7	2

sql语句如下

```
'INSERT census,location=1,scientist=langstroth bu
'INSERT census,location=1,scientist=perpetua butt
'INSERT census,location=1,scientist=langstroth bu
'INSERT census,location=1,scientist=perpetua butt
'INSERT census,location=2,scientist=langstroth bu
'INSERT census,location=2,scientist=langstroth bu
'INSERT census,location=2,scientist=perpetua butt
'INSERT census,location=2,scientist=perpetua butt
```

- 造数据用到的2个脚本  
为了模拟隔多久插入数据  
模拟插入数据时,随机赋值

```
$ cat fake_data.sh
arr=(
'INSERT orders,website=30 phone=10'
'INSERT orders,website=39 phone=12'
'INSERT orders,website=56 phone=11'
)

#while ;;do
for((i=0;i<${#arr[*]};i++));do
    /usr/bin/influx -database 'my_food' -execute
    sleep 10
#    echo "${arr[i]}"
done
#done
```

```
$ cat data.sh
#!/bin/bash
```



```
function rand(){
    min=$1
    max=$(( $2-$min+1 ))
    num=$((date +%s%N))
    echo $(( $num%$max+$min ))
}

while :;do
    /usr/bin/influx -database 'my_database' -exec
    sleep 2;
#     echo "INSERT orders,website=$(rand 1 50) pho
#     break
done
```

field value就是你的数据，它们可以是字符串、浮点数、整数、布尔值，因为InfluxDB是时间序列数据库，所以field value总是和时间戳相关联。在示例中，field value如下：

```
12    23
1      30
11     28
3      28
2      11
1      10
8      23
7      22
```

在上面的数据中，每组field key和field value的集合组成了field set，在示例数据中，有八个field set：

```
butterflies = 12 honeybees = 23
butterflies = 1 honeybees = 30
butterflies = 11 honeybees = 28
butterflies = 3 honeybees = 28
butterflies = 2 honeybees = 11
butterflies = 1 honeybees = 10
butterflies = 8 honeybees = 23
butterflies = 7 honeybees = 22
```

注意，field是没有索引的。如果使用field value作为过滤条件来查询，则必须扫描其他条件匹配后的所有值。因此，这些查询相对于tag上的查询（下文会介绍tag的查询）性能会低很多。

在上面的数据中，tag set是不同的每组tag key和tag value的集合，示例数据里有四个tag set：

```
location = 1, scientist = langstroth
location = 2, scientist = langstroth
location = 1, scientist = perpetua
location = 2, scientist = perpetua
```

现在你已经熟悉了measurement，tag set和retention policy，那么现在是讨论series的时候了。在InfluxDB中，series是共同retention policy，measurement和tag set的集合。以上数据由四个series组成：

任意series编号	retention policy	measurement
series 1	autogen	census
series 2	autogen	census
series 3	autogen	census
series 4	autogen	census

理解series对于设计数据schema以及对于处理InfluxDB里面的数据都是很有必要的。  
最后，point就是具有相同timestamp的相同series的field集合。例如，这就是一个point：

```
name: census
-----
time                butterflies      hc
2015-08-18T00:00:00Z      1              30
```

例子中的series的retention policy为autogen，measurement为census，tag set为location = 1, scientist = perpetua。point的timestamp为2015-08-18T00:00:00Z。

## wal(Write Ahead Log)

参考: <https://jasper-zhang1.gitbooks.io/influxdb/content/Concepts/glossary.html>

最近写的点数的临时缓存。为了减少访问永久存储文件的频率，InfluxDB将最新的数据点缓冲进WAL中，直到其总大小或时间触发然后flush到长久的存储空间。这样可以有效地将写入batch处理到TSM中。  
可以查询WAL中的点，并且系统重启后仍然保留。在进程开始时，在系统接受新的写入之前，WAL中的所有点都必须flushed。

## 目录结构

参考: <http://gitbook.cn/books/59395d3d5863cf478e6b50ba/index.html>

InfluxDB的数据存储有三个目录, 分别是meta、wal、data。meta用于存储数据库的一些元数据, meta目录下有一个meta.db文件。wal目录存放预写日志文件, 以.wal结尾。data目录存放实际存储的数据文件, 以.tsm结尾。基本结构如下:

```
-- wal
-- test
-- autogen
-- 1
-- _00001.wal
-- 2
-- _00002.wal
-- data
-- test
-- autogen
-- 1
-- 0000000001-0000000001.tsm
-- 2
-- 0000000001-0000000010.tsm
-- meta
-- meta.db
```

## 数据采集--> 理解cq和rp

Continuous Query (CQ)是在数据库内部自动周期性跑着的一个InfluxQL的查询, CQs需要在SELECT语句中使用一个函数, 并且一定包括一个GROUP BY time()语句。+

Retention Policy (RP)是InfluxDB数据架构的一部分, 它描述了InfluxDB保存数据的时间。InfluxDB会比较服务器本地的时间戳和你数据的时间戳, 并删除比你在RPs里面用DURATION设置的更老的数据。单个数据库中可以有多个RPs但是每个数据的RPs是唯一的。

实例数据:

db: food\_data

measurement: orders

name: orders

-----

time	phone	website
2016-05-10T23:18:00Z	10	30
2016-05-10T23:18:10Z	12	39
2016-05-10T23:18:20Z	11	56

目标:

自动删除1h以上的原始2秒间隔数据 --> rp实现  
 自动删除超过5min的30s间隔数据 --> rp实现  
 自动将2秒间隔数据聚合到30s的间隔数据 ----> cq实现

2s中插入一次数据:(脚本参考上面fake数据)

```
create databaes food_data
CREATE RETENTION POLICY "a_hour" ON "food_data" D
CREATE RETENTION POLICY "a_week" ON "food_data" D
CREATE CONTINUOUS QUERY "cq_10s" ON "food_data" E
```

在步骤1里面创建数据库时，InfluxDB会自动生成一个叫做autogen的RP，并作为数据库的默认RP，autogen这个RP会永远保留数据。在输入上面的命令之后，a\_hours会取代autogen作为food\_data的默认RP。

验证:

```
select * from "a_week"."downsampled_orders";
select * from "orders";
```

## influxdb数据聚合

### 参考

表名都可以正则

```
select * from /.*/ limit 1
```

查询一个表里面的所有数据

```
select * from cpu_idle
```

查询数据大于200的。

```
select * from response_times where value > 200
```

查询数据里面含有下面字符串的。

```
select * from user_events where url_base = 'frien
```

约等于

```
select line from log_lines where line =~ /paul@in
```

按照30m分钟进行聚合，时间范围是大于昨天的 主机名是serve

```
select mean(value) from cpu_idle group by time(30
```

```
select column_one from foo where time > now() -
```

```
select reqtime, url from web9999.httpd where reqtime > 1000000000
select reqtime, url from web9999.httpd where time > 1000000000
```

url搜索里面含有login的字眼，还以login开头

```
select reqtime, url from web9999.httpd where url like 'login'
```

还可以做数据的merge

```
select reqtime, url from web9999.httpd merge web0000.httpd
```

## influxdb备份恢复

### 参考

参考: <http://stedolan.github.io/jq/>

```
#!/bin/bash

function parse_options {
    function usage() {
        echo -e >&2 "Usage: $0 dump DATABASE [options]
\t-u USERNAME\t(default: root)
\t-p PASSWORD\t(default: root)
\t-h HOST\t\t(default: localhost:8086)
\t-s\t\t\t(use HTTPS)"
    }

    if [ "$#" -lt 2 ]; then
        usage; exit 1;
    fi

    username=root
    password=root
    host=localhost:8086
    https=0
    shift
    database=$1
    shift

    while getopts u:p:h:s opts
    do case "${opts}" in
        u) username="${OPTARG}";;
        p) password="${OPTARG}";;
        h) host="${OPTARG}";;
        s) https=1;;
        ?) usage; exit 1;;
    esac
    done
```

```

if [ "${https}" -eq 1 ]; then
    scheme="https"
else
    scheme="http"
fi
}

function dump {
    parse_options $@

    curl -s -k -G "${scheme}://${host}/db/${databas}
        | jq . -c -M
    exit
}

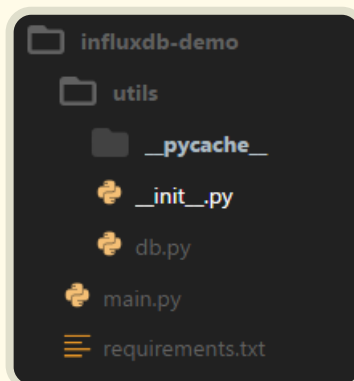
function restore {
    parse_options $@

    while read -r line
    do
        echo >&2 "Writing..."
        curl -X POST -d "[${line}]" "${scheme}://${ho
    done
    exit
}

case "$1" in
    dump)      dump $@;;
    restore)   restore $@;;
    *)         echo >&2 "Usage: $0 [dump|restore] ..."
               exit 1;;
esac

```

## python调用influxdb实现数据增删



utils/db.py

```
# - * - coding: utf-8 - * -

from influxdb import InfluxDBClient

def get_db_connection():
    db_conn = InfluxDBClient(host="192.168.x.x",
    return db_conn
```

main.py

```
#!/home/ansible/.venv/bin/python
# - * - coding: utf-8 - * -

from influxdb.exceptions import InfluxDBClientError
from utils import db

def insert_success_point_2db():
    db_conn = db.get_db_connection()
    # 写入成功记录, success字段值约定为1
    success_point = [{
        "measurement": "wake",
        "tags": {
            "isp": "mobile",
            "region": "上海",
        },
        "fields": {
            "mobile": 159123456xx,
            "success": 1,
        }
    }]

    try:
        db_conn.write_points(success_point)
    except InfluxDBClientError as e:
        print("influxdb db client error: {0}".format(e))
    except InfluxDBServerError as e:
        print("influxdb db server error: {0}".format(e))
    except Exception as e:
        print("influxdb error: {0}".format(e))
    finally:
        if db_conn is not None:
            db_conn.close()
```

```
def insert_fail_point_2db():
    db_conn = db.get_db_connection()
    # 写入失败记录, fail字段值约定为0
    fail_point = [{
        "measurement": "wake",
        "tags": {
            "isp": "mobile",
            "region": "上海",
        },
        "fields": {
            "mobile": 1591234xxxx,
            "fail": 0,
        }
    }]
    try:
        db_conn.write_points(fail_point)
    except InfluxDBClientError as e:
        print("influxdb db client error: {0}".format(e))
    except InfluxDBServerError as e:
        print("influxdb db server error: {0}".format(e))
    except Exception as e:
        print("influxdb error: {0}".format(e))
    finally:
        if db_conn is not None:
            db_conn.close()

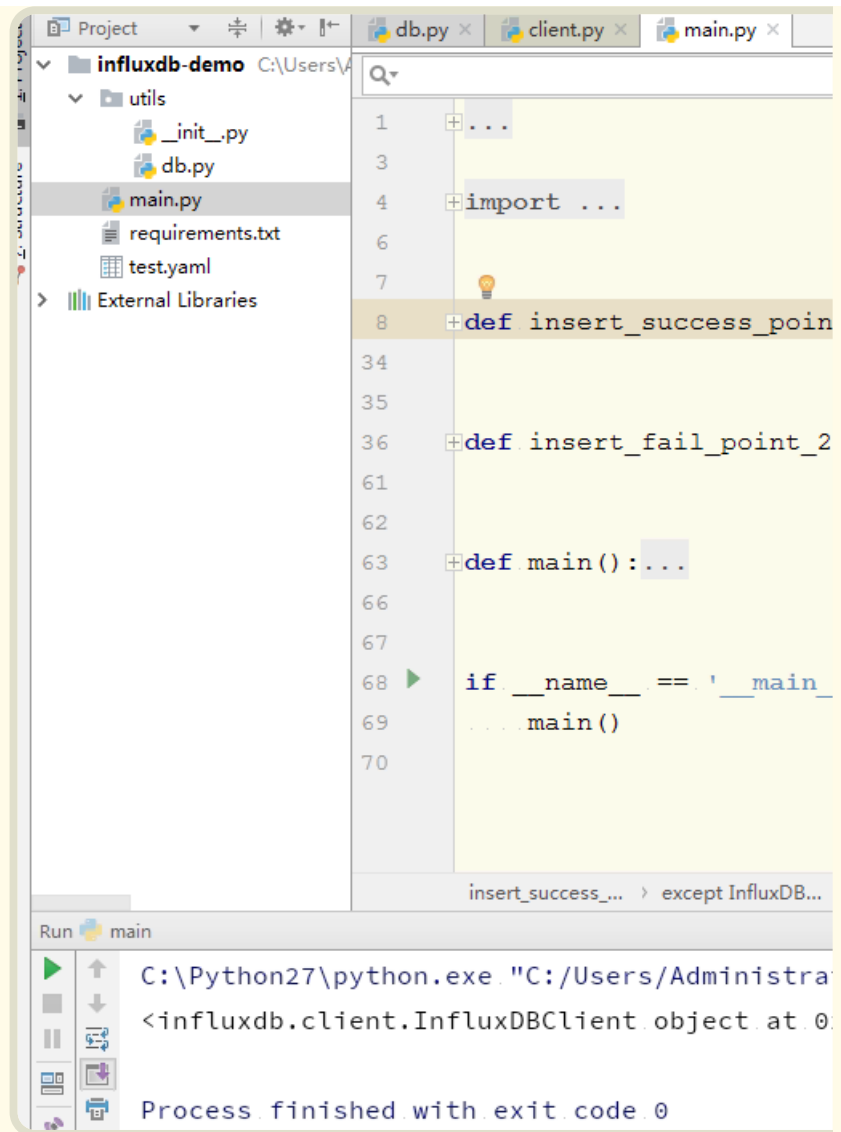
def main():
    insert_success_point_2db()
    insert_fail_point_2db()

if __name__ == '__main__':
    main()
```

requirements.txt

```
certifi==2017.11.5
influxdb==5.0.0
```





好文要顶

关注我

收藏该文



毛台

关注 - 4

.....  
粉丝 - 19

+加关注

0

0

👍 推荐


 反对

« 上一篇: [\[k8s\]k8s的控制层kubelet+docker配合调度机制\(k8架构\)](#)

» 下一篇: [svc]influxdb+grafana实战-各省份api访问成功率统计

posted @ 2017-12-16 15:03 [毛台](#) 阅读(573) 评论(0) [编辑](#) [收藏](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

**【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库!**

【活动】华为云普惠季 1折秒杀 狂欢继续

【工具】SpreadJS纯前端表格控件，可嵌入应用开发的在线Excel

【腾讯云】拼团福利，AMD云服务器8元/月



高性能云服务器 首购**1核1G75元**/年

**100%** 基准CPU性能

推荐好友可享受高达**45%** 返现奖励

立即购买

相关博文：

- 如何实现Docker应用的自定义弹性伸缩
- collectd+influxDB+grafana搭建性能监控平台
- 基于jmxtrans+influxdb+grafana实现对canal监控
- Jmeter + Grafana + InfluxDB 性能测试监控
- InfluxDB部署





血拼风暴 一促即发

4核8G云主机 直降**2811元/年**

立即抢购



新至强，创“芯” 构建高效云

最新新闻：

- 对话万达加速器高管：万达物联网创新背后的逻辑
  - 阿里健康公布半年报：营收18.79亿元 净利1050万元
  - 天猫双11组委会：退货率仅6% 远低于行业平均水平
  - 京东自研无人货机首飞成功：起飞重量超过吨级
  - 小米CFO回应与美图手机合作：让小米多了一张牌
- » 更多新闻...