性能指标数据收集(metric)

hydra 提供了 QPS、并发数、响应时间、响应状态码的统计数据，只需简单配置即可将这些数据保存到 influxdb，通过 grafana 配置为动态图表进行实时监控。或使用 `convoy` 配置报警，通过短信、微信发送到指定的用户组。

1. metric 配置

设置服务的子节点配置，配置名为 `metric` 。 `metric` 配置实际是 `influxdb` 服务器连接信息:

参数名	必须	说明
host	是	influxdb 服务器地址
dataBase	是	数据库名称
cron	是	保存到 influxdb 的间隔时长
userName	否	用户名
password	否	密码

```
package main

import (
        "github.com/micro-plat/hydra/context"

        "github.com/micro-plat/hydra/hydra"
)

type apiserver struct {
        *hydra.MicroApp
}

func main() {
        app := &apiserver{
                hydra.NewApp(
                        hydra.WithPlatName("mall"),
                        hydra.WithSystemName("apiserver"),
                        hydra.WithServerTypes("api")),
        }

        app.API("/hello", hello)
        app.Conf.API.SetSubConf("metric", `{
                "host":"http://192.168.106.219:8086",
        "dataBase":"mall_apiserver",
        "cron":"@every 10s",
        "userName":"",
        "password":""
    }   `)

        app.Start()

}

func hello(ctx *context.Context) (r interface{}) {
        return "hello world"
}
```

2.查看 influxdb 中的配置信息

请求服务

运行后通过 `ab` 发送请求

```
ab -c 1 -n 100000 http://localhost:8090/hello
```

查看 influxdb 中表和数据

- 登录 influxdb http://host:8083,切换到当前使用的数据库
- 执行命令 `SHOW MEASUREMENTS`

  measurements

name	说明
api.server.request.qps	qps 数据
api.server.request.timer	时长数据
api.server.request.working	处理中数据
api.server.response.meter	响应数据

- 查询表数据
  - 执行命令 `select * from "api.server.request.qps"`

api.server.request.qps

time	host	m1	m15	m5	name	url
2019-07-03T01:27:34.571054403Z	"192.168.4.121"	360	360	360	"mall.apiserver.yl"	"/hello"
2019-07-03T01:27:44.570693377Z	"192.168.4.121"	28044	28044	28044	"mall.apiserver.yl"	"/hello"
2019-07-03T01:27:54.570326513Z	"192.168.4.121"	54963	54963	54963	"mall.apiserver.yl"	"/hello"

> 数据为每隔 10 秒保存 1 次， `name` 是 `平台名称` ， `系统名称` ， `集名名称` 的组合

- 执行命令 `select * from "api.server.request.timer"`

api.server.request.timer

time	count	host	m1	m15	m5	max	
2019-07-03T01:27:34.571061828Z	360	"192.168.4.121"	0	0	0	390112	1189
2019-07-03T01:27:44.570700223Z	28044	"192.168.4.121"	385.03330790386286	27.699062827315974	82.1883343275836	840981	1183
2019-07-03T01:27:54.570333895Z	54962	"192.168.4.121"	739.9682623368426	57.20104574471909	167.93066219699298	804686	1139

- 执行命令 `select * from "api.server.request.working"`

api.server.request.working

time	host	name	url	value
2019-07-03T01:27:34.571057319Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0
2019-07-03T01:27:44.570697624Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0
2019-07-03T01:27:54.570331476Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0
2019-07-03T01:28:04.570261436Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	1
2019-07-03T01:28:14.570386432Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0
2019-07-03T01:28:24.570468063Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0
2019-07-03T01:28:34.5708024Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0
2019-07-03T01:28:44.571053764Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0
2019-07-03T01:28:54.570372963Z	"192.168.4.121"	"mall.apiserver.yl"	"/hello"	0

- 执行命令 `select * from "api.server.response.meter"`

api.server.response.meter

time	count	host	m1	m15	m5	mean
2019-07-03T01:27:34.571046412Z	359	"192.168.4.121"	0	0	0	3064.33685334
2019-07-03T01:27:44.570900407Z	28045	"192.168.4.121"	2479.2290952832423	2474.36232594048	2475.080966920066	2771.95955048
2019-07-03T01:27:54.570519887Z	54963	"192.168.4.121"	2512.6667246963975	2476.8296322716205	2482.3749431652154	2732.16510766
2019-07-03T01:28:04.57114069Z	81256	"192.168.4.121"	2533.481301839644	2478.7314953310274	2487.832644814569	2697.97361676
2019-07-03T01:28:14.570873075Z	100000	"192.168.4.121"	2471.434900021882	2475.0488338914784	2476.532333046619	2560.76833217