# 快速入门 | Open-Falcon

https://book.open-falcon.org/zh/usage/getting-started.html

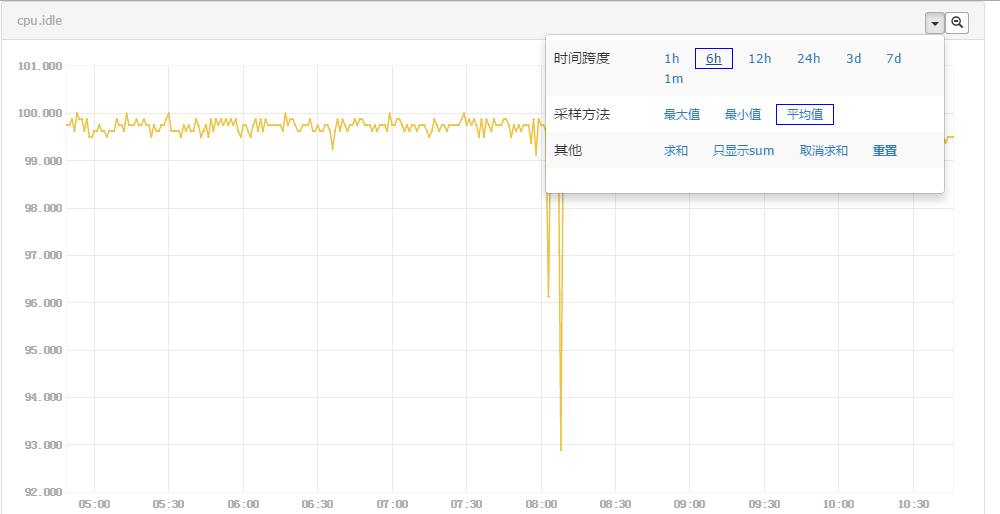
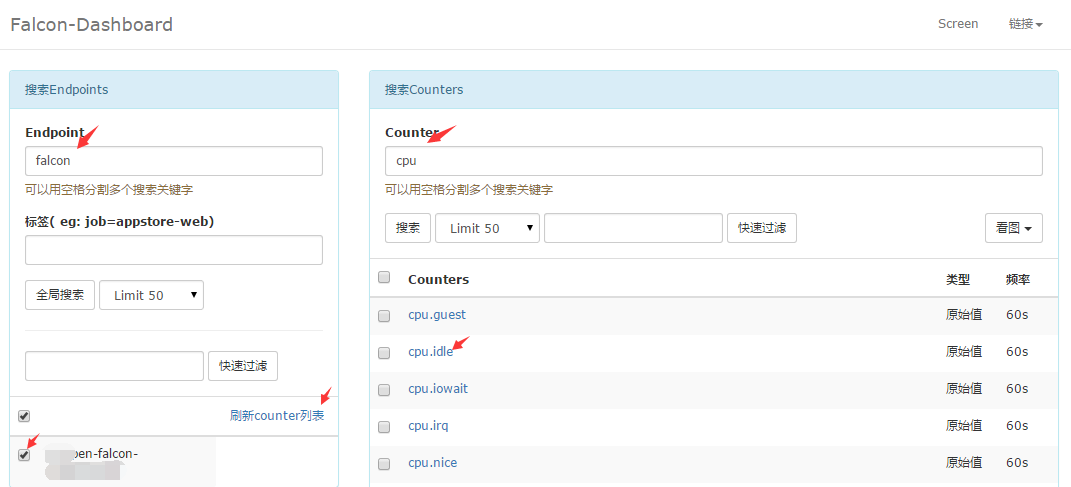
系统搭建好了，应该如何用起来，这节给大家逐步介绍一下

查看监控数据

我们说agent只要部署到机器上，并且配置好了heartbeat和transfer就自动采集数据了，我们就可以去dashboard上面搜索监控数据查看了。dashboard是个web项目，浏览器访问之。左侧输入endpoint搜索，endpoint是什么？应该用什么搜索？对于agent采集的数据，endpoint都是机器名，去目标机器上执行hostname，看到的输出就是endpoint，拿着hostname去搜索。

搜索到了吧？嗯，选中前面的复选框，点击“查看counter列表”，可以列出隶属于这个endpoint的counter，counter是什么？counter=${metric}/sorted(${tags})

假如我们要查看cpu.idle，在counter搜索框中输入cpu并回车。看到cpu.idle了吧，点击，会看到一个新页面，图表中就是这个机器的cpu.idle的近一小时数据了，想看更长时间的？右上角有个小三角，展开菜单，可以选择更长的时间跨度



如何配置报警策略

上节我们已经了解到如何查看监控数据了，如果数据达到阈值，比如cpu.idle太小的时候，我们应该如何配置告警呢？

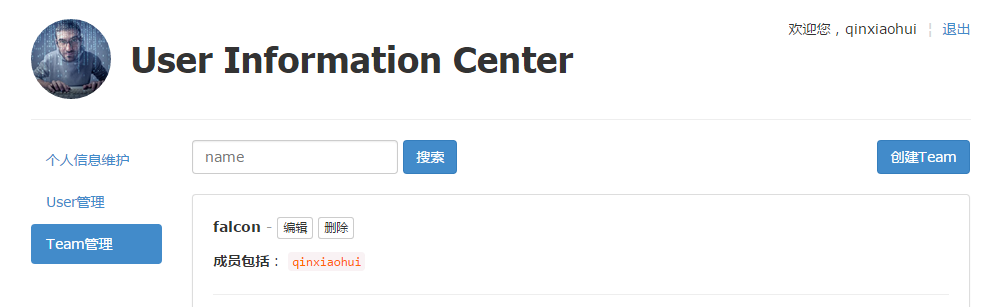
配置报警接收人

falcon的报警接收人不是一个具体的手机号，也不是一个具体的邮箱，因为手机号、邮箱都是容易发生变化的，如果变化了去修改所有相关配置那就太麻烦了。我们把用户的联系信息维护在一个叫UIC(新用户推荐使用Go版本的UIC，即：falcon-fe项目)的系统里，以后如果要修改手机号、邮箱，只要在UIC中修改一次即可。报警接收人也不是单个的人，而是一个组（UIC中称为Team），比如falcon这个系统的任何组件出问题了，都应该发报警给falcon的运维和开发人员，发给falcon这个团队，这样一来，新员工入职只要加入falcon这个Team即可；员工离职，只要从falcon这个Team删掉即可。

1

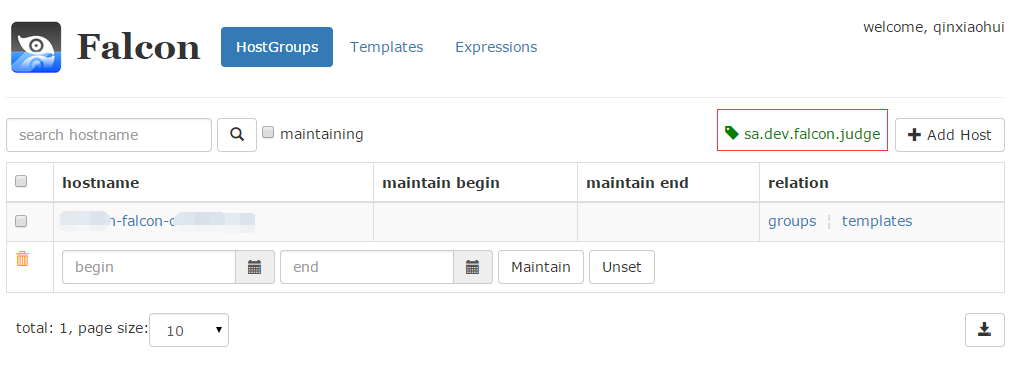
浏览器访问UIC，如果启用了LDAP，那就用LDAP账号登陆，如果没有启用，那就注册一个或者找管理员帮忙开通。创建一个Team，名称姑且叫falcon，把自己加进去，待会用来做测试。

2



创建HostGroup

比如我们要对falcon-judge这个组件做端口监控，那首先创建一个HostGroup，把所有部署了falcon-judge这个模块的机器都塞进去，以后要扩容或下线机器的时候直接从这个HostGroup增删机器即可，报警策略会自动生效、失效。咱们为这个HostGroup取名为：sa.dev.falcon.judge，这个名称有讲究，sa是我们部门，dev是我们组，falcon是项目名，judge是组件名，传达出了很多信息，这样命名比较容易管理，推荐大家这么做。

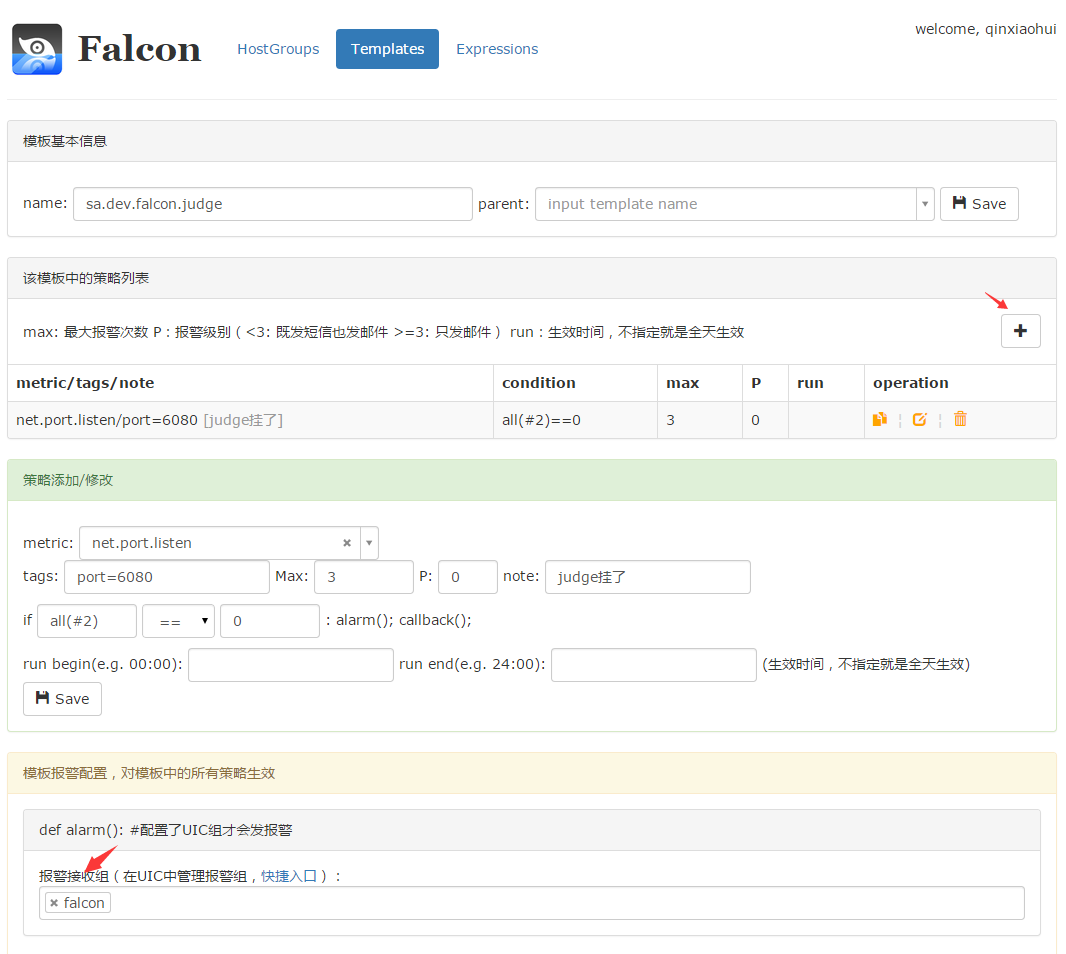


在往组里加机器的时候如果报错，需要检查portal的数据库中host表，看里边是否有相关机器。那host表中的机器从哪里来呢？agent有个heartbeat(hbs)的配置，agent每分钟会发心跳给hbs，把自己的ip、hostname、agent version等信息告诉hbs，hbs负责写入host表。如果host表中没数据，需要检查这条链路是否通畅。

1

创建策略模板

portal最上面有个Templates链接，这就是策略模板管理的入口。我们进去之后创建一个模板，名称姑且也叫：sa.dev.falcon.judge，与HostGroup名称相同，在里边配置一个端口监控，通常进程监控有两种手段，一个是进程本身是否存活，一个是端口是否在监听，此处我们使用端口监控。

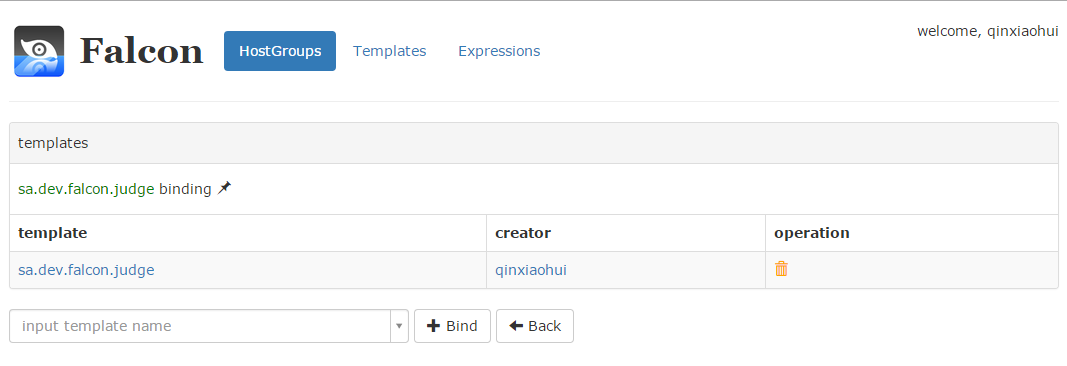


右上角那个加号按钮是用于增加策略的，一个模板中可以有多个策略，此处我们只添加了一个。下面可以配置报警接收人，此处填写的是falcon，这是之前在UIC中创建的Team。

1

将HostGroup与模板绑定

一个模板是可以绑定到多个HostGroup的，现在我们重新回到HostGroups页面，找到sa.dev.falcon.judge这个HostGroup，右侧有几个超链接，点击【templates】进入一个新页面，输入模板名称，绑定一下就行了。



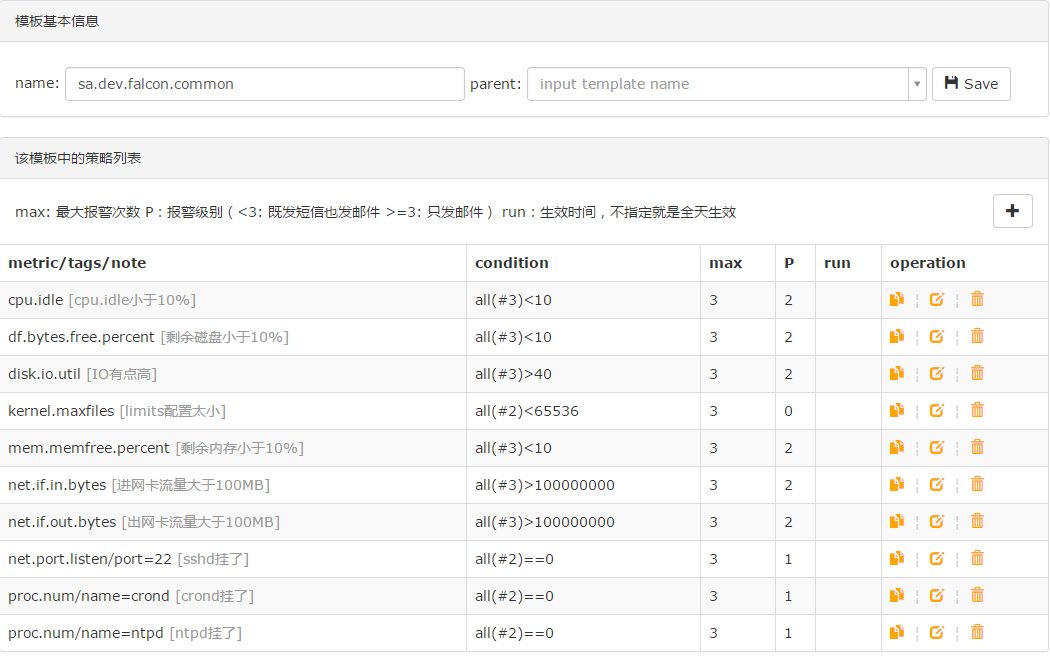
补充

上面步骤做完了，也就配置完了。如果judge组件宕机，端口不再监听了，就会报警。不过大家不要为了测试报警效果，直接把judge组件给干掉了，因为judge本身就是负责判断报警的，把它干掉了，那就没法判断了……所以说falcon现在并不完善，没法用来监控本身的组件。为了测试，大家可以修改一下端口监控的策略配置，改成一个没有在监听的端口，这样就触发报警了。

上面的策略只是对falcon-judge做了端口监控，那如果我们要对falcon这个项目的所有机器加一些负载监控，应该如何做呢？

1. 创建一个HostGroup：sa.dev.falcon，把所有falcon的机器都塞进去
2. 创建一个模板：sa.dev.falcon.common，添加一些像cpu.idle,load.1min等策略
3. 将sa.dev.falcon.common绑定到sa.dev.falcon这个HostGroup

附：sa.dev.falcon.common的配置样例



大家可能不知道各个指标分别叫什么，自己push的数据肯定知道自己的metric了，agent push的数据可以参考：<https://github.com/open-falcon/agent/tree/master/funcs>

## 如何配置策略表达式

策略表达式，即expression，具体可以参考[HostGroup与Tags设计理念](https://book.open-falcon.org/zh/philosophy/tags-and-hostgroup.html)，这里只是举个例子：



上例中的配置传达出的意思是：falcon-judge这个模块的所有实例，如果qps连续3次大于1000，就报警给falcon这个报警组。

expression无需绑定到HostGroup，enjoy it

# Tag和HostGroup | Open-Falcon

https://book.open-falcon.org/zh/philosophy/tags-and-hostgroup.html

这个话题很难阐述，大家慢慢读：）

咱们从数据push说起，监控系统有个agent部署在所有机器上采集负载信息，比如cpu、内存、磁盘、io、网络等等，但是对于业务监控数据，比如某个接口调用的cps、latency，是没法由agent采集的，需要业务方自行push，其他监控数据，比如MySQL相关监控指标，HBase相关监控指标，Redis相关监控指标，agent也是无能为力，需要业务方自行采集并push给监控。

于是，监控Server端需要制定一个接口规范，大家调用这个接口即可完成数据push，拿agent汇报的cpu.idle数据举个例子：

{

"metric": "cpu.idle",

"endpoint": `hostname`,

"tags": "",

"value": 12.34,

"timestamp": `date +%s`,

"step": 60,

"counterType": "GAUGE"

}

metric即监控指标的名称，比如disk.io.util、load.1min、df.bytes.free.percent；endpoint是这个监控指标所依附的实体，比如cpu.idle，是谁的cpu.idle？显然是所在机器的cpu.idle，所以对于agent采集的机器负载信息而言，endpoint就是机器名；tags待会再说；value就是这个监控指标的值喽；timestamp是时间戳，单位是秒；step是rrdtool中的概念，Falcon后端存储用的rrdtool，我们需要告诉rrdtool这个数据多长时间push上来一次，step就是干这个事的；counterType也是rrdtool中的概念，目前Falcon支持两种类型：GAUGE、COUNTER。

tags没有给大家解释，这是重点，为了便于理解，我再举个例子，某个方法调用的latency：

{

"metric": "latency",

"endpoint": "11.11.11.11:8080",

"tags": "project=falcon,module=judge,method=QueryHistory",

"value": 12.34,

"timestamp": `date +%s`,

"step": 60,

"counterType": "GAUGE"

}

tags可以对push上来的这条数据打一些tag，就像写篇blog，可以打上几个tag便于归类。上例中我们push了一条latency数据，打了三个tag，传达的意思是：这个latency是调用QueryHistory这个方法的延迟，模块是judge，项目是falcon。

对于push上来的这种数据，我们配置监控就很方便了，比如：对于falcon-judge这个模块的所有方法调用的latency只要大于0.5就报警。我们可以配置一个expression（在portal中）：

each(metric=latency project=falcon module=judge)

阈值触发条件是：all(#1)>0.5

我们没有写method，所以对于所有method都生效，一条配置搞定！这就是tag的强大之处。

--重头戏分割线--

对于业务方自己push的数据，可以加各种tag来区分，传达更多信息。但是，对于agent采集的数据呢？cpu.idle、load.1min这种数据应该加什么tag呢？这种数据没有任何tag……那我们应该如何配置报警呢？总不至于一台机器一台机器配置吧……

each(metric=latency endpoint=host1)

公司才一万台机器，嗯，这酸爽……

tag，实际是一种分组方式，如果数据push的时候无法声明自己的分类，那我们就要**在上层手工对数据做分组**了。这，也就是HostGroup的设计出发点。

比如我们把falcon这个项目用到的所有机器放到一个HostGroup（名称姑且叫sa.dev.falcon，名称中包含部门、团队信息，这么命名不错吧）中，然后为sa.dev.falcon绑定一个策略模板，下面的所有机器就都生效了，扩容的时候增加的机器塞到sa.dev.falcon，也就自动生效了监控，机器下线直接从该组删掉即可。

总结，HostGroup实际是对endpoint的一种分组。

想象一下一条数据push上来了，我们应该怎么判断这条数据是否应该报警呢？那我们要做的就是找到这条数据关联的Expression和Strategy，Expression比较好说，无非就是看配置的expression中的tag是否是当前push上来的数据的子集；Strategy呢？push上来的数据有endpoint，我们可以看这个endpoint是否属于某几个HostGroup，HostGroup又是跟Template绑定的，Template下就是一堆Strategy，顺藤摸瓜：）

--忧伤的分割线--

但是，HostGroup是一个扁平结构，使用起来可能不是那么方便，举个例子。

* 我们可以把公司所有机器放到一个分组，配置硬件监控，比如磁盘坏，收到报警之后直接发送到系统组
* A同学和B同学共同管理了3个项目（包括falcon项目），可以把这三个项目的机器放到一个分组，配置机器负载监控，模板名称姑且叫sa.dev.common，配置一下cpu.idle、df.bytes.free.percent之类的，报警发给A、B两个同学
* falcon这个项目的磁盘io压力比较高，比其他两个项目io压力高，这是正常情况，比如平时我们配置disk.io.util大于40就报警，但是falcon的机器io压力大于80才需要报警，于是，我们需要创建一个新模板继承自sa.dev.common，然后绑定到falcon上，以此覆盖父模板的配置，提供不同的报警阈值
* falcon中有个judge组件，监听在6080端口，我们需要把judge的所有机器放到一个HostGroup，为其配置端口监控

OK，问题来了，如果judge扩容5台机器，这5台机器就要分别加到上面四个分组里！

略麻烦哈~所以我们内部使用falcon实际是配合机器管理系统的，机器管理对机器的管理组织方式是一棵树，机器加到叶子节点，上层节点也就同样拥有了这个机器。

所以，大家在用falcon的时候，规模如果比较小，就用扁平化的HostGroup即可。如果规模比较大，可能就需要做个二次开发，与内部机器管理系统结合了。如果你们内部没有机器管理，我们过段时间可能会提供一个树状机器管理系统，敬请期待。

# Expression例子

<Endpoint:192.168.0.193, Status:PROBLEM, Strategy:<Id:1657, Metric:cpu.busy, Tags:map[], avg(#3)>=0.01 MaxStep:3, P6, , <Id:87, Name:访问策略, ParentId:0, ActionId:51, Creator:181f1ab4e4a6baac5f9158b265767ebc>>, Expression:<nil>, LeftValue:0.33333, CurrentStep:1, PushedTags:map[], TS:2016-09-23 14:15:00>

<Endpoint:192.168.0.188, Status:OK, Strategy:<nil>, Expression:<Id:1, Metric:container.cpu.usage.busy, Tags:map[podname:dengxiaoqian24lrfq95o-zqqlu], all(#3)>=0.01 MaxStep:3, P0 ActionId:51>, LeftValue:0, CurrentStep:1, PushedTags:map[id:f2edefd1ee3e627f7596314dd8d6b04ffa78d81ee7666410c2b4301b9c39194d podname:dengxiaoqian24lrfq95o-zqqlu], TS:2016-09-23 13:39:00>

<Tos:13432342123, Content:[P0][PROBLEM][192.168.0.188][][ all(#3) container.cpu.usage.busy id=57d0417d33a6e96ea194814a5abe2a743713f2413fecb16d533dff1575547055,podname=houweitao-14lrfqcsi-zf07r 0.05517>=0.01][O2 2016-09-23 14:12:00]>

<Tos:cavan.wang@xx.com, Subject:[P0][PROBLEM][192.168.0.188][][ all(#3) container.cpu.usage.busy id=57d0417d33a6e96ea194814a5abe2a743713f2413fecb16d533dff1575547055,podname=houweitao-14lrfqcsi-zf07r 0.05248>=0.01][O2 2016-09-

23 17:31:00], Content:PROBLEM

P0

Endpoint:192.168.0.188

Metric:container.cpu.usage.busy

Tags:id=57d0417d33a6e96ea194814a5abe2a743713f2413fecb16d533dff1575547055,podname=houweitao-14lrfqcsi-zf07r

all(#3): 0.05248>=0.01

Note:

Max:3, Current:2

Timestamp:2016-09-23 17:31:00

https://demo.dev.yihecloud.com/monitor/expression/view/2

>

{

id: "e\_3\_3161a56380c2aa85945608ff8322864b",

endpoint: "192.168.0.188",

metric: "container.cpu.usage.busy",

counter: "192.168.0.188/container.cpu.usage.busy id=57d0417d33a6e96ea194814a5abe2a743713f2413fecb16d533dff1575547055,podname=houweitao-14lrfqcsi-zf07r",

func: "all(#3)",

leftValue: "0.07101",

operator: ">=",

rightValue: "0.01",

note: "测试podname告警",

maxStep: 3,

currentStep: 1,

priority: 0,

status: "PROBLEM",

timestamp: 1474612680,

expressionId: 3,

strategyId: 0,

templateId: 0,

link: "https://demo.dev.yihecloud.com/monitor/expression/view/3"

},

# 触发expression邮件队列

## Expression 2017-1-13

lpush /mail "{\"tos\":\"liyang@yihecloud.com\",\"subject\":\"[P6][PROBLEM][192.168.1.73][][ avg(#3) container.mem.usage.percent deploy\_id=726nac2hltxqhsjsfddd4us7xsgjshy,id=d00c4be8eaae1a10fe9bdad66a2d66322f6ecb6bc4bc98776974d41e4d3fb334 63.67874\\u003e=40][O1 2017-01-09 10:19:00]\",\"content\":\"{\\\"id\\\":\\\"e\_21\_9f9a01c0c723eaa134cd5d2319fdfd78\\\",\\\"strategy\\\":null,\\\"expression\\\":{\\\"id\\\":666,\\\"metric\\\":\\\"container.mem.usage.percent\\\",\\\"tags\\\":{\\\"deploy\_id\\\":\\\"726nac2hltxqhsjsfddd4us7xsgjshy\\\"},\\\"func\\\":\\\"avg(#3)\\\",\\\"operator\\\":\\\"\\\\u003e=\\\",\\\"rightValue\\\":40,\\\"maxStep\\\":3,\\\"priority\\\":6,\\\"note\\\":\\\"\\\",\\\"actionId\\\":716,\\\"tplAlarmLevel\\\":1},\\\"status\\\":\\\"PROBLEM\\\",\\\"endpoint\\\":\\\"192.168.1.73\\\",\\\"leftValue\\\":63.678741455078125,\\\"currentStep\\\":1,\\\"eventTime\\\":1483957140,\\\"pushedTags\\\":{\\\"deploy\_id\\\":\\\"726nac2hltxqhsjsfddd4us7xsgjshy\\\",\\\"id\\\":\\\"d00c4be8eaae1a10fe9bdad66a2d66322f6ecb6bc4bc98776974d41e4d3fb334\\\"}}\"}"

## 触发expression邮件队列/mail

lpush /mail "{\"tos\":\"demo@xx.com\",\"subject\":\"[P6][PROBLEM][192.168.0.184][][ avg(#3) container.mem.usage.percent deploy\_id=71feoin9orpbqvkt5uhutvaeorcjw8e,id=273a9d4093375a2ed92414c1b5163c1ba32ef33e32fdf76a5a383b9555f47d61 34.73587\\u003e=20][O3 2016-09-28 16:15:00]\",\"content\":\"PROBLEM\\r\\nP6\\r\\nEndpoint:192.168.0.184\\r\\nMetric:container.mem.usage.percent\\r\\nTags:deploy\_id=71feoin9orpbqvkt5uhutvaeorcjw8e,id=273a9d4093375a2ed92414c1b5163c1ba32ef33e32fdf76a5a383b9555f47d61\\r\\navg(#3): 34.73587\\u003e=20\\r\\nNote:\\r\\nMax:3, Current:3\\r\\nTimestamp:2016-09-28 16:15:00\\r\\nhttps://demo.dev.yihecloud.com/monitor/expression/view/49\\r\\n\"}"