

## 08 | 如何用 Nightingale 解决 Prometheus 的告警管理问题？

2023-01-25 秦晓辉 来自北京



天下无鱼

<https://shikey.com/>

《运维监控系统实战笔记》

[课程介绍 >](#)



讲述：秦晓辉

时长 12:11 大小 11.13M



你好，我是秦晓辉。上一讲我们聊到了 Prometheus 的存储问题，并提出了 3 种增强方案，这一讲我们继续关注 Prometheus 的另一个问题——告警管理。

Prometheus 的告警规则、记录规则都是采用配置文件的方式管理的，非常适合奉行 Infrastructure as Code 的公司或团队内部使用。但如果要把监控能力开放给全公司，就需要有较好的支持协同操作的 UI，让各个团队互不干扰的同时共享一些通用的成果。

解决这个需求的开源产品，有两款备选，一个是 Grafana，另一个是夜莺（Nightingale）。Grafana 擅长可视化，是监控绘图领域事实上的标准，而夜莺的侧重点是告警管理，所以这一讲我们重点来介绍一下夜莺，**我们可以通过夜莺搭建公司级的监控系统，把监控告警能力赋予公司所有的团队。**

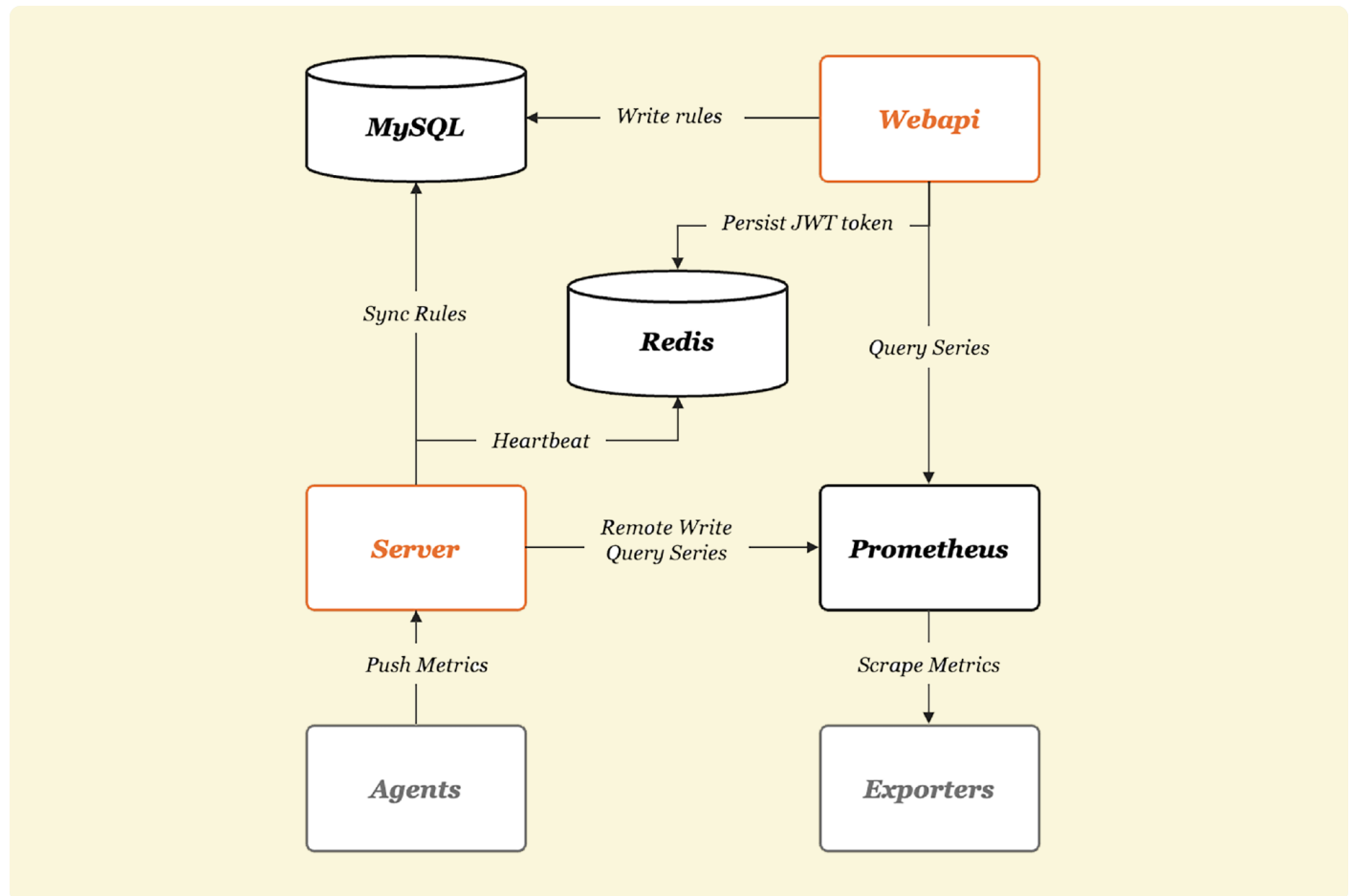
### 夜莺简介

夜莺最初是滴滴开源的，之后捐赠给了中国计算机学会开源发展委员会（CCF ODC），目标是整合云原生开源生态的众多能力，为用户提供开箱即用、一体化全方位的云原生监控解决方案。



注：点击查看 [夜莺的 GitHub 地址](#) 和 [文档地址](#)

我们先来看一下夜莺的架构，对夜莺的工作模式有个基本的认识。



夜莺的单机部署架构（图片源自Nightingale官网）

左下角 Agents 表示监控数据采集器，夜莺可以对接多种 Agent，比如 Categraf、Telegraf、Grafana-Agent、Datadog-Agent。这些 Agent 都是 PUSH 模型，周期性采集监控数据，然后推给 Server 的 HTTP 接口。

Server 接收到数据之后，会通过 Remote Write 协议把数据转发给时序库，这里时序库使用的是 Prometheus，Prometheus 要想接收 Remote Write 协议的数据，需要在启动参数中开启 `--enable-feature=remote-write-receiver`。除了 Prometheus，也可以使用 M3DB、VictoriaMetrics、Thanos 等作为时序库。

Server 的职能相当于一个 Pushgateway，同时也是一个告警引擎，它会周期性地从 MySQL 中同步告警规则，做规则判断生成告警事件并发送，对标 Prometheus 的告警引擎和 Alertmanager 模块。Server 还会往 Redis 发送心跳信息，不过后面的版本有计划下掉 Redis，直接使用 MySQL 处理心跳。

Webapi 模块提供 HTTP 接口，与前端 JavaScript 交互，主要有两个功能，一个是响应管理请求，比如告警规则、屏蔽规则、监控大盘的增删改查；一个是响应时序数据查询，作为一个 Proxy 把请求转发给后面的时序库，等时序库返回结果之后再转发给前端。

如果有些监控数据是使用 Exporter 采集的，就需要 PULL 模型的采集支持。通常有 3 种做法，一种是直接使用 Prometheus 本身，配置 Scrape 规则；也可以单独部署一个 agent mode 模式的 Prometheus 作为抓取器，和时序库的职能做进程级别的拆分；还可以使用其他支持 PULL 模式的抓取器，比如 Categraf、Grafana-Agent。

## 部署夜莺

了解了夜莺的架构之后，下面我们简单聊一下如何部署夜莺，最简单的方式是使用 Docker compose，一行命令就能搞定。

 复制代码

```
1 git clone https://github.com/ccfos/nightingale.git
2 cd docker
3 docker-compose up -d
```

浏览器访问 nwebapi 提供的 18000 端口就能看到登录页面，默认用户是 root，默认密码是 root.2020，如果你有 Docker 环境，可以使用这种方式快速部署体验。

如果你对 Kubernetes 和 Helm 比较熟悉，也可以采用 [🔗 Helm 的方式部署](#)。当然，最常用的实际是 [🔗 二进制的方式部署](#)，具体步骤你可以参考我给出的链接。

服务端部署完成之后，我们可以采集一些监控数据看看真实效果，推荐你 [🔗 安装 Categraf](#)。当然，因为夜莺支持多种采集器，你也可以使用其他自己熟悉的采集器。安装完成之后，我们来看一下夜莺具体是如何管理告警的。

## 告警管理

Prometheus 的告警管理是在 prometheus.yml 中配置告警规则，在 alertmanager.yml 中配置发送规则，都是需要修改配置文件的，上百人使用的话不太好协同管理。而夜莺提供了 UI 配置能力（当然也有 API），并且在一些方面做了增强，比如更丰富的告警规则配置、历史事件存档、活跃事件聚合查看、对接告警自愈等。下面我们一起来看一下夜莺告警管理的思路。

## 规则管理

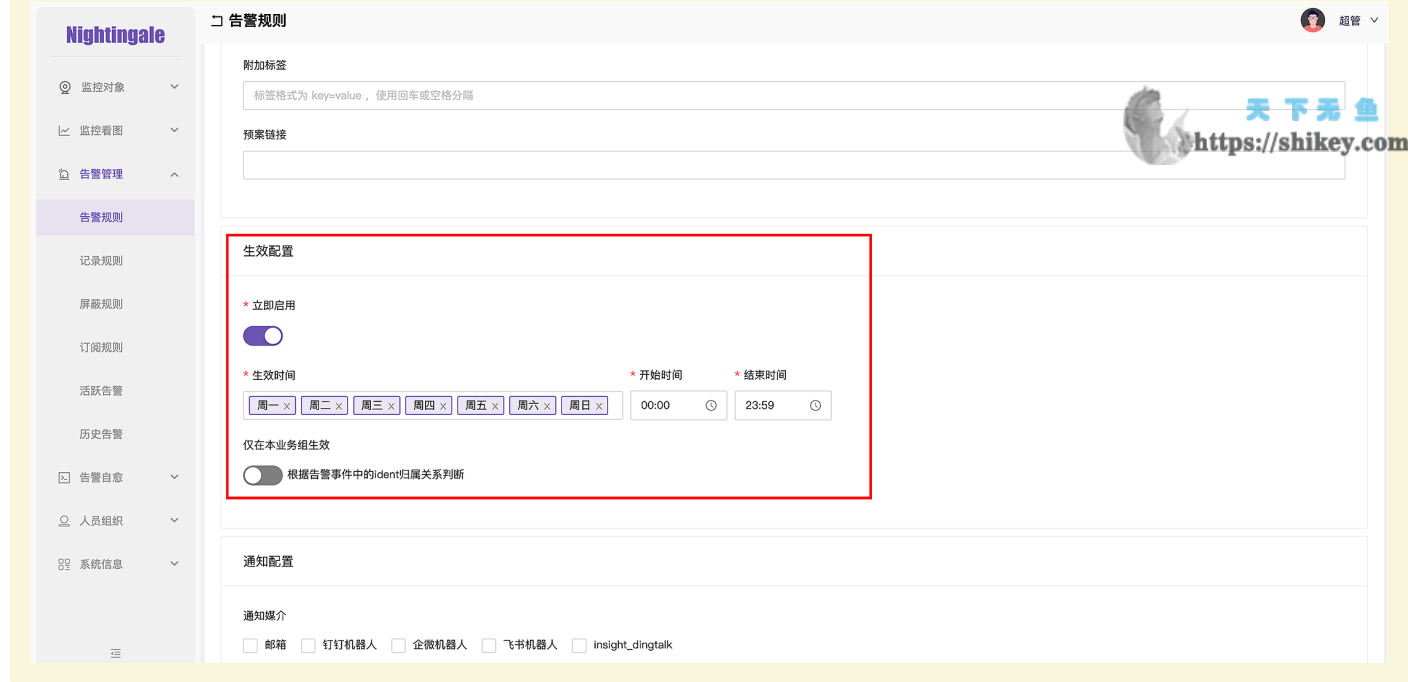
一个公司可能会有几十上百团队配置成千上万条告警规则，显然不能用一个扁平化的表格来罗列管理，夜莺引入了一个**业务组**的概念，每一条规则都要归属于某一个业务组，只有这个业务组的人可以管理组内的规则。



当然，业务组下面不仅有告警规则，还有监控大盘、监控对象、屏蔽规则、订阅规则、告警自愈脚本等等。业务组是夜莺里最重要的一个管理概念。

## 规则配置

告警规则的配置，核心还是 **PromQL 和持续时长**。当然夜莺会有一些额外的增强配置，比如规则的生效时间段、是否仅在本业务组生效、是否启用恢复通知、留观时长、最大发送次数等等。



夜莺的告警规则是把 PromQL 和发送方式整合到了一条规则中，这个做法和 Alertmanager 是不同的，Prometheus 的告警规则只有 PromQL、持续时长、附加标签、注解这些基本信息，至于发给谁、怎么发，都是在 Alertmanager 中配置的。如何评价这两种方式呢？

Prometheus+Alertmanager 的方式，可以看做是订阅式，告警规则中不指定接收者，在 Alertmanager 的配置中统一设置过滤条件和对应的接收者。这种方式非常灵活，但灵活的东西往往需要定规范，否则容易混乱，比如大家统一按照业务线的标签来做订阅，这就要求时序数据都要打上业务线的标签，或者把业务线的标签放到告警规则的附加标签中，需要付出一些额外的心力。

Alertmanager 的方式特别适合什么场景呢？就是所有的告警都统一由某一个团队来负责，在这种场景下，Prometheus+Alertmanager 不失为一种最佳实践。

### 「讨论」

据说 Borgmon 集群和 Borg 集群是一一对应的，那就是说，Borgmon 要处理对应 Borg 集群上的所有应用的告警，不知道在 Borgmon 中是怎么处理告警规则的，如果你了解的话可以留言分享一下。

夜莺的处理方式和 Datadog 很像，每个团队配置自己的告警规则，发给自己这个团队，即自己管自己的，不需要把告警规则和接收规则拆到两个地方分别配置，我个人觉得更加直观一些，而且夜莺也支持订阅方式，我们后面会介绍。

## 告警屏蔽

告警屏蔽这个功能比较简单，是指对一些告警事件做静默处理。对于那些预期内的告警，处理人不希望被打扰就会短时间做一下屏蔽，通常是根据标签对事件做过滤。



夜莺目前的版本，只能按照时间段屏蔽，比如屏蔽凌晨 0 点到早上 7 点的所有告警，不能做周期性屏蔽，后面的版本会考虑增加这个功能。

## 告警订阅

这个机制和 **Alertmanager** 有点儿像，可以根据告警规则或标签做事件订阅，类似于邮件的抄送功能。

比如我是业务方，我的业务跑在 **Kubernetes** 中，**Kubernetes** 平台如果发生重大故障，我是希望能及时知道的，所以我可以订阅 **Kubernetes** 的所有严重告警。另外，虽然我希望得知 **Kubernetes** 的严重告警，但我毕竟不是 **Kubernetes** 的运维人员，所以我在订阅这类事件的时候，不希望用电话这种方式接收告警（太重了），只希望用邮件之类的轻量级方式，所以订阅规则中通常可以重新定义发送媒介、重新定义事件级别。

## 历史告警存档

因为夜莺依赖 **MySQL** 做数据持久化，所以告警事件可以直接存入数据库，所有的历史告警都可以追踪查询。虽然这个功能很简单，但很多企业都需要，在自证清白的时候显得尤为重要。

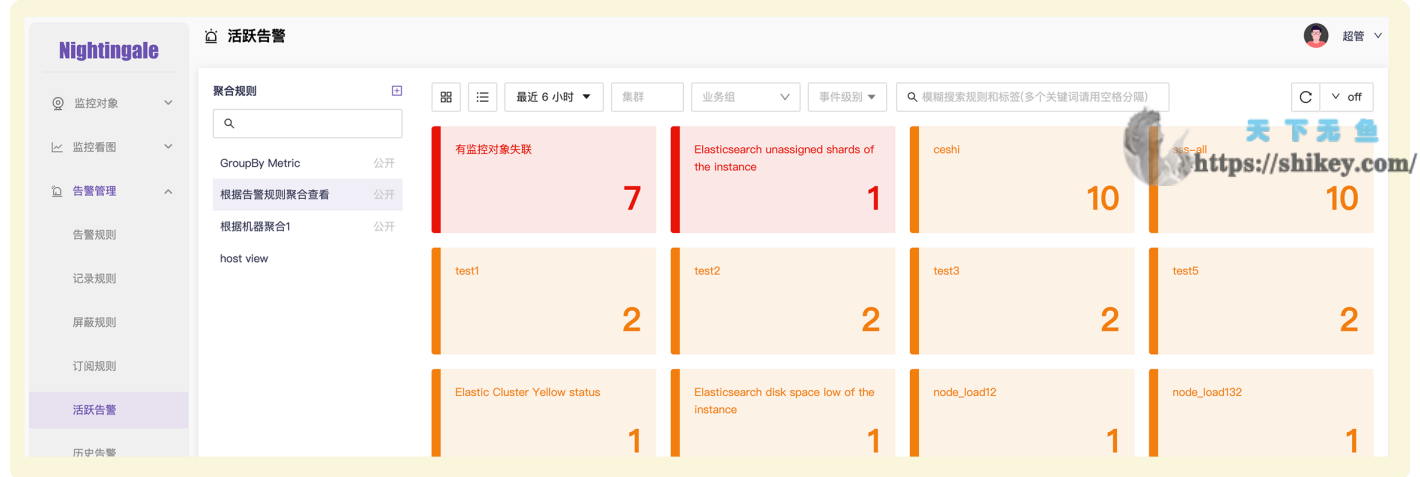
当然，所有历史告警存档，也可以用于后续分析，从这些数据中可以轻易得知哪些业务线的告警发送得最多，消耗电话、短信费用最多，哪些业务线的告警解决得最快，哪些人是接收告警的劳模等等。其实也能从侧面反映出这个团队的告警策略需要优化，或者业务稳定性需要优化。

## 活跃告警聚合

所谓活跃告警就是未恢复的告警，活跃告警功能很重要，应该作为日常巡检必须要关注的页面。

为了减少查看的心理负担，便于给告警事件分类，夜莺支持聚合卡片视图，而且聚合规则是可以自定义的，比如当前有 300 条未恢复的告警，我们可以根据告警规则聚合查看，也可以根据地域、业务线、服务、机器等维度聚合查看，一目了然非常方便。





## 告警自愈

当告警事件触发之后，能自动触发一个恢复动作来止损，这就是所谓的告警自愈。一般监控系统都会支持 **Webhook**，告警触发之后自动回调某个接口，我们就可以在这个接口里写一些自动化逻辑，但是这种方式还是要写个 **HTTP Server** 的，成本略高。夜莺除了支持 **Webhook** 之外，还可以在告警时自动执行某个脚本，用户就可以直接在脚本里写逻辑了。

告警自愈依赖 **ibex** 模块，这是一个批量执行脚本的小工具，你可以安装测试一下。不过有些公司会觉得有安全隐患，不敢开启这个功能，我觉得纯内网环境问题不大，如果开放到公网的话确实需要小心。

自愈脚本要能够在机器上运行，需要有较强的权限管控，这个权限也是依赖业务组的机制，只有这个业务组的管理人员，才能去这个组内的机器上跑脚本。夜莺里有个对象管理，主要就是管机器的，设计对象管理功能很重要的一个原因就是支持告警自愈。

## 失联告警

夜莺主要是用推模式来接收监控数据，所以如何感知监控对象失联是个比较麻烦的问题。**PromQL** 中有 **absent** 函数，但是这个函数使用起来非常麻烦，如果要为 100 台机器配置失联告警，就要配置 100 条告警规则，基本无法管理。

夜莺在服务端加了一个逻辑，接收到监控数据之后，会自动从数据中解析出 **ident** 标签当做机器标识，然后为这个机器生成 **target\_up** 指标。这个机器有监控数据上报，则 **target\_up** 的值设置为 1；如果长时间收不到机器的指标上报，则 **target\_up** 的值设置为 0。通过这种方式，只需要配置一条告警规则就可以覆盖所有的监控对象。

以上就是夜莺所有告警相关的功能，有些人了解了夜莺之后觉得不错，想要去尝试，但是之前很多规则已经用 Prometheus Yaml 文件管理了，感觉迁移起来比较麻烦，就来问我怎么搞比较合适。



我的观点是这样的：老的 Yaml 文件管理的规则其实可以不用动，甚至如果 Prometheus 只是给自己团队使用的话也不太需要引入夜莺，只有那些想要**把监控能力开放给全公司用的场景**才需要引入夜莺，而且新规则可以用夜莺管理，老的规则不迁移或者慢慢迁移都是可以的。还是那句话，**技术是为了解决现实问题，没有什么非黑即白。**

## 小结

这一讲我们要解决的问题是增强 Prometheus 的告警管理能力，因为 Prometheus 的 Yaml 文件管理方式不太方便做公司级协同管理。Grafana 和夜莺都可以解决这个问题，不过 Grafana 更擅长看图，夜莺更擅长告警管理，所以这一讲我们重点讲解了夜莺的告警功能。

夜莺告警管理能力分三类，一个是规则管理，包括告警规则、屏蔽规则、订阅规则，一个是事件管理，包括历史事件、活跃事件，最后一个告警自愈。为了方便你理解、记忆，我把这一讲的内容整理成了一张脑图，你可以参考。






## 互动时刻

不管是 Prometheus 还是 Nightingale，在告警恢复事件中都无法拿到当前值（告警触发事件中是可以拿到的），很多人觉得不理解，你知道为什么会这样吗？你有办法拿到恢复时候的最新值吗？欢迎你在评论区留言讨论。如果你觉得有收获，也欢迎你把今天的内容分享给你身边的朋友，邀他一起学习。我们下一讲见。

分享给需要的人，Ta 购买本课程，你将得 18 元

 生成海报并分享

上一篇 07 | 如何解决Prometheus的存储容量问题？

下一篇 09 | 监控概论（上）：有哪些方法可以指导监控数据采集？

## 精选留言 (9)

 写留言



**KEIO**

2023-01-28 来自重庆

老师 请教一下 可以对比一下Grafana和Altermanager的告警管理能力吗？

作者回复: Grafana是个单点，如果告警规则很多，可能会有瓶颈；其次是Grafana的告警规则配置我是感觉真难用，不知道其他人怎么看。alertmanager不负责告警规则管理，只负责告警事件的后续，对于告警事件的后续处理，包括分组、屏蔽、抑制，还是比较齐全的，不过alertmanager更多的是给一个团队使用，如果很多团队都使用一个alertmanager来管理告警，就容易混乱，需要制定良好的规范，需要制定良好的规范，重要的事情说三遍

共 2 条评论 >

 1



**Amos**

2023-01-25 来自江苏

原生k8s支持告警自愈吗？webhook的方式

作者回复: 这个问法欠妥，K8s本身是可以处理节点故障自动迁移pod的，从这个角度来讲，是有告警自愈的能力的。webhook是prometheus、nightingale等这种监控系统的职能，产生告警之后通过webhook调用第三方的系统，prometheus、nightingale、zabbix等都可以做



 1



**peter**

2023-01-25 来自北京

请教老师几个问题：

Q1: server与Redis之间的心跳有什么作用？

Q2: webapi没有界面吗？

“Webapi 模块提供 HTTP 接口，与前端 JavaScript 交互”，从这句话看，好像webapi没有界面。

“浏览器访问 nwebapi 提供的 18000 端口就能看到登录页面”，从这句话看，好像webapi有界面。

Q3: 架构图中, agents和exporter都采集数据, 有什么区别?

Q4: Prometheus和Nightingale都可以处理告警, 采用Nightingale后, 就禁掉Prometheus自身的告警功能, 是这样吗?



作者回复: 1, server使用redis心跳, 这样所有的server就都知道活着的server的列表, 然后就可以做分片逻辑, 每个server只处理一部分告警规则

2, 界面是js、css、html渲染的, 这些静态资源文件可以使用nginx来serve, 也可以使用webapi来serve, js拉取数据是走的webapi的接口

3, agent是推, exporter是拉

4, 是的



**SICUN**

2023-02-01 来自北京

老师能不能谈一下边沿触发告警和周期触发告警的适用场景?

作者回复: 没看懂



**隆哥**

2023-01-31 来自福建

快猫的采集器我觉得很好, 基本覆盖了常用服务的数据采集, 只需要修改配置一下就可以了。但是我有一个疑惑, 比如我监控几百台服务器, 每台服务器有可能有不同的服务需要被采集, 如果这样的话, 快猫服务采集的那些配置文件如何管理呢? 用表格来做扁平化管理嘛。

作者回复: 配置管理这块, 建议是ansible、puppet之类的工具, telegraf、datadog-agent这些目前也都是修改配置文件的方式, 后面也有考虑在Nightingale里开放agent的配置管理能力, 不过还没有时间搞



**Geek\_1a3949**

2023-01-29 来自上海

尝试回答下课后问题:

告警表达式是带判断的PromQL, 查询到值表示触发了阈值, 查询不到表示未触发; 而告警恢复的时候, PromQL表达式返回空值, 故没有\$value。

作者回复:



天下无鱼

<https://shikey.com/>



隆哥

2023-01-28 来自福建

请教一下老是，夜莺左侧的监控对象，我添加了快猫的采集，集群名称为啥都是Default，如何修改呢？

作者回复: 数据通过n9e-server上报，server.conf里有个配置叫ClusterName



lei

2023-01-26 来自河北

请教老师，我司也在打算研发一个成熟的监控平台。然而像关联分析，您在前面提过的：在虚拟机管理控制台可以查到宿主，宿主旁边加个看图的按钮，点击可以看宿主的监控数据，通过宿主还能看到虚机的列表，每个虚机旁边也有看图的按钮。

我能想到的就是基于api纯二次开发，可是想想工作量确实有些大。那基于夜莺或者Grafana，类似这样的功能，有什么好的建议吗？谢谢！

作者回复: 这些开源软件都是使用通用的解法来解决各个场景的需求。本质是要考虑数据如何组织，跟使用Grafana还是Nightingale没有关系。比如以Prometheus的数据结构为例，虚机的内存利用率监控指标，至少得有个标签标明是哪个虚机，如果能够再打上一个所属物理机的标签就更有价值，这个物理机的标签是你需要想办法打上去的，Grafana、Nightingale都不会管这些问题



无名无姓

2023-01-25 来自北京

是保存回复值么

