

12 | 网络监控：如何监控网络链路和网络设备？

2023-02-03 秦晓辉 来自北京

《运维监控系统实战笔记》

课程介绍 >



讲述：秦晓辉

时长 12:53 大小 11.77M

你好，我是秦晓辉。

上一讲我们介绍了机器监控，机器属于基础设施。除了机器之外，还有一个常见的基础设施，就是网络。**网络监控主要包括网络链路监控和网络设备监控**，通常系统运维人员会比较关注。今天我们就来揭开网络监控的面纱，看看其中涉及了哪些关键技术和实践方法。

网络链路监控

网络链路监控主要包含三个部分，网络连通性、网络质量、网络流量。

连通性和质量的监控手段非常简单，就是在链路一侧部署探针，去探测链路另一侧的目标，通过 **ICMP、TCP、HTTP** 等协议发送探测数据包，分析回包的结果。典型的指标有丢包率、延迟、回包是否匹配预期条件等。

网络流量监控，则关注流量大小以及流量内容。流量大小广泛应用于水位管理，比如机器网卡、交换机的接口、外网出口、专线带宽等，及时发现网络瓶颈。分析流量内容，则可以识别过度耗用带宽的用户和应用程序，验证网络 QoS 策略等。

这一讲我们使用 **Catgraf** 来演示一下常用探针的配置方式，进行网络连通性和质量监控。网络流量大小，可以使用 **SNMP** 采集数据，相关方法我们会在后面介绍网络设备监控时讲解。流量内容监控我暂时没有找到开源方案，如果你知道的话，欢迎留言分享。

ICMP 探测

Catgraf 的 ICMP 探测使用 Ping 插件，相关配置在 `conf/input.ping/ping.toml`，主要是配置要探测的目标地址，你可以看一下我给出的样例。

 复制代码

```
1 [[instances]]
2 targets = [ "10.4.5.6", "10.4.5.7" ]
3 labels = { region="cloud", product="n9e" }
4
5 [[instances]]
6 targets = [ "10.4.5.8" ]
7 labels = { region="cloud", product="zbx" }
```

例子中是 Ping 3 个地址，其中有两个机器是 **Nightingale** 产品使用的，有一个机器是 **Zabbix** 产品使用的，都位于 **cloud** 这个区域（**region**）里，所以打了不同的标签来丰富其维度信息。

如果要同时 Ping 很多台机器，就会占用很多文件句柄，需要提前调大 **Catgraf** 的文件句柄限制。如果是用 **systemd** 来托管，可以在 `catgraf.service` 文件中添加 **LimitNOFILE** 配置。

 复制代码

```
1 [Service]
2 LimitNOFILE=8192
```


另一个坑是 **CAP** 问题，如果使用非 **root** 账号来运行 **Catgraf**，需要在 `catgraf.service` 文件中添加如下配置。

 复制代码

```
1 [Service]
```

```
2 CapabilityBoundingSet=CAP_NET_RAW
3 AmbientCapabilities=CAP_NET_RAW
```

Ping 插件可以采集到目标是否连通、延迟时间、丢包率等指标，可以据此做网络链路的监控。比如机房专线的探测，只需要在某个机房部署 **Catgraf**，来探测另一个机房的设备。

最后是  **仪表盘和告警规则**，`dashboard.json` 是仪表盘配置，`alerts.json` 是告警规则配置，导入夜莺就可以使用。如果是直接使用的 **Prometheus**，也可以参考 JSON 文件中的 PromQL，并将其改造成 **Prometheus yaml** 配置。

TCP 探测

很多时候机器是禁 Ping 的，此时 TCP 探测就派上用场了。TCP 探测用的是 **Catgraf** 的 `net_response` 插件，配置文件在 `conf/input.net_response/net_response.toml`。实际这个插件既可以探测 TCP 的响应，也可以探测 UDP 的响应。配置中最核心的是 `targets` 部分，指定探测的目标，我给出一个例子，你体会下。

 复制代码

```
1 [[instances]]
2 targets = [
3     "10.2.3.4:22",
4     "localhost:6379",
5     ":9090"
6 ]
```

- `10.2.3.4:22` 表示探测 `10.2.3.4` 这个机器的 `22` 端口是否可以连通。
- `localhost:6379` 表示探测本机的 `6379` 端口是否可以连通。
- `:9090` 表示探测本机的 `9090` 端口是否可以连通，没有写 IP 或主机名的就默认使用 `localhost`。

原理也很简单，就是 **Catgraf** 向目标地址发起网络连接。如果能连通，就认为是正常的，指标值上报为 `0`，如果失败就是非 `0` 的值。监控指标名字是 `net_response_result_code`。

如果是 **UDP** 的端口，是无法发起连接探测的。此时采用内容匹配探测，即通过 **UDP** 发个字符串给探测目标，理论上探测目标很快就会给出回复。我们来检查回复内容，如果回复内容包

含特定字符串，就表示探测目标活着。相关配置是 `send` 和 `expect` 字段，你可以看一下配置样例。

 复制代码

```
1 ## The following options are required for UDP checks. For TCP, they are
2 ## optional. The plugin will send the given string to the server and then
3 ## expect to receive the given 'expect' string back.
4 ## string sent to the server
5 send = "ping"
6 ## expected string in answer
7 expect = "pong"
```

`net_response` 插件也内置了仪表盘和告警规则，可以在 [🔗 代码目录](#) 中找到，导入夜莺就可以使用。

HTTP 探测

HTTP 探测和 TCP 的探测逻辑几乎完全一致，只不过 HTTP 是七层协议，Categraf 可以解析到 Status code、Response body 这些更细粒度的信息。相关配置在 `conf/input.http_response/http_response.toml`，我们看一个配置样例。

 复制代码

```
1 [[instances]]
2 targets = [
3     "http://localhost",
4     "https://www.baidu.com"
5 ]
```

很多公司都会把所有的机器上部署 Agent，Agent 会开一个 HTTP 端口，这样就可以通过探测这些 HTTP 端口，知道 Agent 是否存活，进而反推机器的存活性。

HTTP 插件可以对返回的 Response 做规则匹配，比如判断 Response body 中是否包含特定的字符串，或者 Status code 是否是指定的值等，你可以看下相关配置。

 复制代码

```
1 ## Optional substring match in body of the response (case sensitive)
2 expect_response_substring = "ok"
3
4 ## Optional expected response status code.
```



```
5 expect_response_status_code = 200
```

如果目标地址是 HTTPS 的，Categraf 也会自动检查证书过期时间，指标名是 `cert_expire_timestamp`，证书过期时间一定要记得加个告警规则，我见过很多公司因为证书过期而导致线上故障。

网络链路的几种探测方式，我们就讲到这里，下面我们再来看一下网络中的节点，网络设备的监控。

网络设备监控

网络设备监控的典型手段有三个，一个是 Ping 监控，探测是否存活，刚刚我们已经介绍过了。另一个是通过 SNMP 获取指标，比如各个网口的状态、流量、包量等。最后一个是 SNMP Trap，一般网络设备有问题，都会发出 Trap 消息，这些 Trap 消息很有价值，分析这些 Trap 消息是常用且有效的监控手段。我们先来看一下 SNMP 获取指标的方式。

SNMP 指标获取方式

要采集网络设备的监控指标，一定要了解 SNMP 协议。这个内容比较多，你可以结合 [《SNMP（简单网络管理协议）简介》](#) 来理解。简单来讲，就是交换机上有个组件叫 SNMP agent（即 `snmpd`），监听 UDP 161 端口，提供查询服务。SNMP manager，比如 Categraf，可以向 SNMP agent 发起查询请求，传入的参数是 OID，SNMP agent 返回 OID 对应的监控数据。

Categraf 提供了 SNMP 插件，配置文件在 `conf/input.snmp/snmp.toml`，核心配置就是 SNMP agent 的连接地址以及要采集的 OID 列表。

举个例子。

 复制代码

```
1 interval = 60
2
3 [[instances]]
4 agents = ["udp://172.30.15.189:161"]
5
6 timeout = "5s"
7 version = 2
8 community = "public"
```

```

9 agent_host_tag = "ident"
10 retries = 1
11
12 [[instances.field]]
13 oid = "RFC1213-MIB::sysUpTime.0"
14 name = "uptime"
15
16 [[instances.field]]
17 oid = "RFC1213-MIB::sysName.0"
18 name = "source"
19 is_tag = true
20
21 [[instances.table]]
22 oid = "IF-MIB::ifTable"
23 name = "interface"
24 inherit_tags = ["source"]
25
26 [[instances.table.field]]
27 oid = "IF-MIB::ifDescr"
28 name = "ifDescr"
29 is_tag = true

```

例子中，配置的交换机地址是 172.30.15.189，认证版本是 version 2，认证团体名是 public。如果是 v3 版本，可能和下面的认证配置差不多。

 复制代码

```

1 version = 3
2 sec_name = "managev3user"
3 auth_protocol = "SHA"
4 auth_password = "example.Demo.c0m"

```

要采集哪些指标，是通过 OID 字段来指定的，instances.field 有两个，一个是获取系统启动时间（OID 是 RFC1213-MIB::sysUpTime.0），一个是获取系统名称（OID 是 RFC1213-MIB::sysName.0），这两个 OID 都是 Scalar 类型，可以使用 snmpget 进行测试。

 复制代码

```

1 snmpget -v2c -c public 172.30.15.189 RFC1213-MIB::sysUpTime.0

```

因为 name = “uptime” 这个配置，指标会命名成 snmp_uptime，snmp 是插件前缀，uptime 就是 name 字段指定的。看起来挺顺畅，但是为什么 sysName 这个 field 多了一个 is_tag = true

的配置呢？因为 `sysName` 是个字符串，不是要上报的监控数据，它会作为一个标签 `source=$sysName` 附到所有时序数据上。

`instances.table` 部分采集的是 `Table` 类型的数据，`Table` 的 `OID` 是 `IF-MIB::ifTable`，`Table` 里有多列数据和索引，`Catgraf` 会自动获取 `Table` 中所有的索引字段，做成时序数据的标签，还会自动获取所有的数据列，作为指标上报。但是，有些列可能不是数值，比如 `ifDescr` 就是字符串，这个列无需作为指标上报，作为标签上报更合适，所以有个 `is_tag=true` 的配置。`inherit_tags` 表示继承下来的标签，`ifTable` 收集到的数据就会自动附上 `source` 标签。

`ifTable` 可以拿到各个网口的出入向流量、包量等，对我们做容量监控很有帮助，当然也可以拿到错包、丢包的情况，这些都是很重要的监控指标。这里我贴几条监控数据，让你有一个直观的认识。

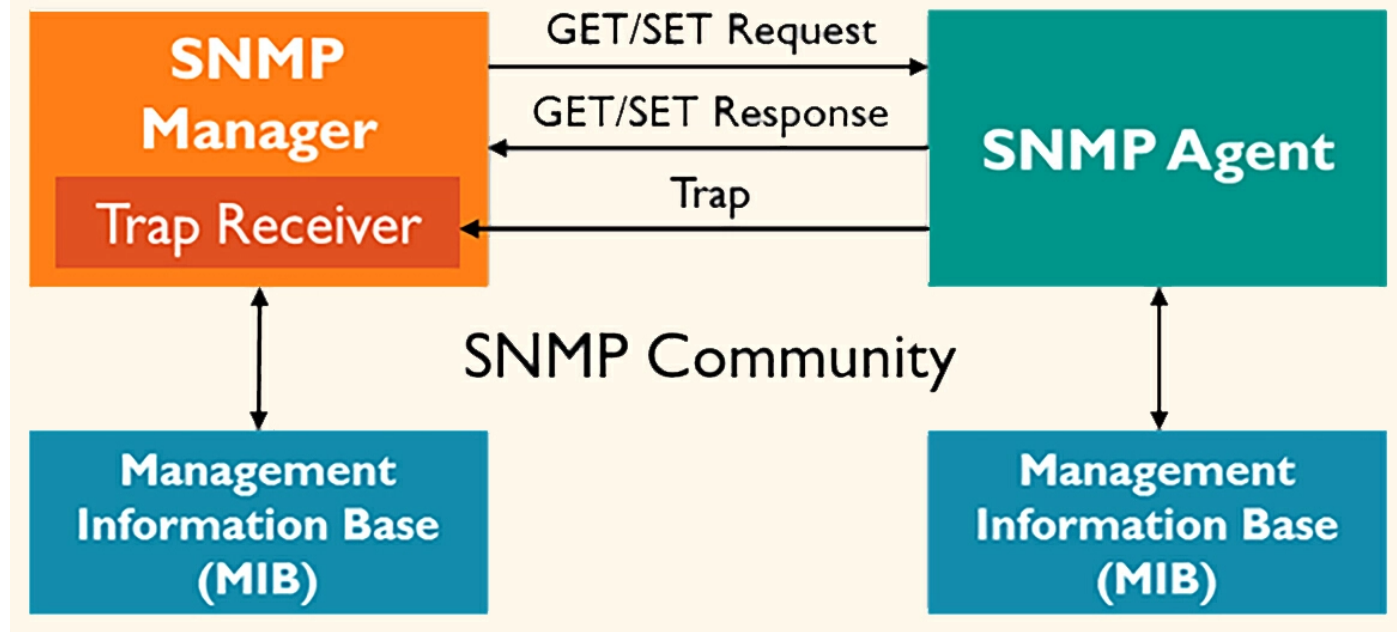
 复制代码

```
1 12:01:36 snmp_interface_ifInNUcastPkts agent_hostname=prober01 ident=172.30.15.
2 12:01:36 snmp_interface_ifInErrors agent_hostname=prober01 ident=172.30.15.189
3 12:01:36 snmp_interface_ifOutDiscards agent_hostname=prober01 ident=172.30.15.1
4 12:01:36 snmp_interface_ifOutErrors agent_hostname=prober01 ident=172.30.15.189
```

除了获取指标之外，`SNMP Trap` 也是很关键的监控手段，**SNMP 指标监控是轮询式，定期采集，而 Trap 是只要有消息就自动通知，即时性更好**。下面我们来看一下 `Trap` 的相关技术方案。

SNMP Trap

与 `SNMP` 采集指标的方式不同，`Trap` 消息是由交换机里的 `SNMP agent` 发消息给 `SNMP manager`（也是走的 `UDP` 协议），与指标采集的数据流向相反。目前 `Catgraf` 尚不支持 `Trap` 消息接收，你可以先用 `Telegraf` 的 `snmp_trap` 插件做测试。



SNMP Trap架构（图片来自网络）

用 Trap 机制做事件监控是比较便捷的方式，交换机出现关键问题的时候，都会立刻发出 Trap 消息。我们只要在 Trap Receiver 中配置消息匹配规则，指定什么样的消息应该产生告警即可。但是，匹配规则肯定是需要用人类易读的方式，这就需要借助 MIB 库，把 Trap 中的 OID 翻译成人类易读的字符串。不翻译的话，原始的 Trap 消息可能是这样的，很难读懂。

复制代码

```
1 {
2   "Version": 2,
3   "TrapType": 0,
4   "OID": null,
5   "Other": null,
6   "Community": "public",
7   "Username": "",
8   "Address": "172.16.1.64:49692",
9   "VarBinds": {
10    ".1.3.6.1.2.1.1.3.0": 79085276900000000,
11    ".1.3.6.1.2.1.2.2.1.1.18": 18,
12    ".1.3.6.1.2.1.2.2.1.2.18": "Vlanif103",
13    ".1.3.6.1.2.1.2.2.1.7.18": 2,
14    ".1.3.6.1.2.1.2.2.1.8.18": 2,
15    ".1.3.6.1.6.3.1.1.4.1.0": [1, 3, 6, 1, 6, 3, 1, 1, 5, 3]
16  },
17  "VarBindOIDs": [
18    ".1.3.6.1.2.1.1.3.0",
19    ".1.3.6.1.6.3.1.1.4.1.0",
20    ".1.3.6.1.2.1.2.2.1.1.18",
21    ".1.3.6.1.2.1.2.2.1.7.18",
22    ".1.3.6.1.2.1.2.2.1.8.18",
23    ".1.3.6.1.2.1.2.2.1.2.18"
24  ]
}
```


不过遗憾的是，我还没有在开源社区找到兼具翻译和规则匹配的软件，目前可能只有一些商业软件可以做到，比如卓豪。所以这里我通过讲解原理来让你了解整个过程，就不给你演示了。

小结

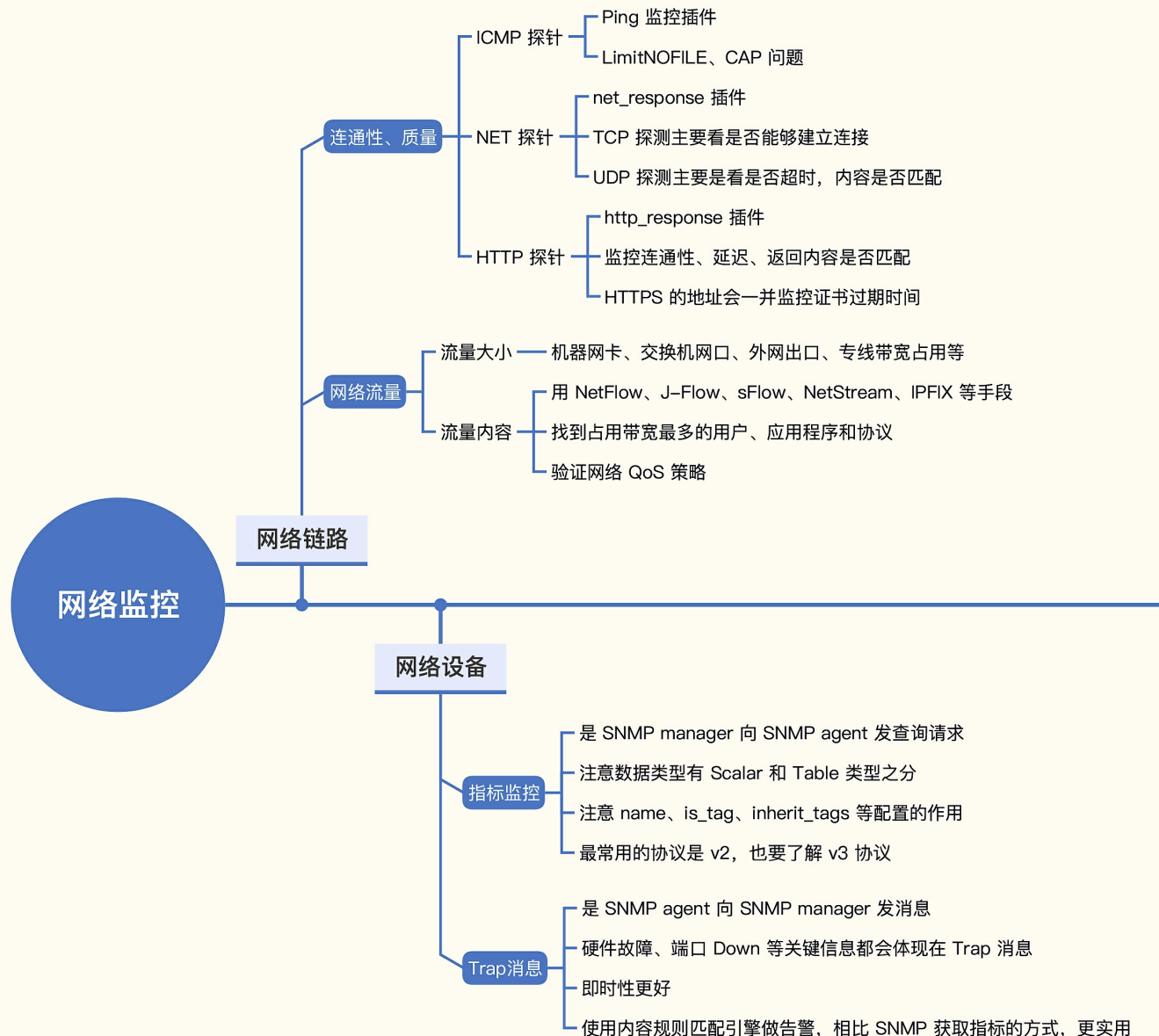
这一讲我从网络链路和网络设备两个方面，给你介绍了网络监控的方法。

网络链路监控，包括连通性和质量监控、流量和内容监控，典型的探测协议是 ICMP、TCP、HTTP。Categraf 有对应的采集插件可以使用，Prometheus 生态的 Blackbox-Exporter 也可以做这个事情。

网络流量包含多个方面，比如机器网卡是否跑满，我们可以用 Categraf 采集网卡流量数据，而交换机的网口流量需要用 SNMP 协议获取。网络内容监控，可以使用 NetFlow、J-Flow、sFlow、NetStream、IPFIX 等手段，找出占用带宽最多的用户、应用程序和协议。

网络设备监控，用途很广泛，除了交换机、路由器外，UPS、打印机、存储等都支持 SNMP 协议，我们可以通过 SNMP 获取到设备的各种监控指标。不过要注意，对于一些老式交换机，SNMP 采集不能太频繁，不然有可能影响交换机的性能。

另一个网络设备监控手段就是 Trap 了，通过 SNMP agent 主动发消息给 SNMP manager，即时性很好，我们只要写一个 Trap Receiver，做协议翻译和规则匹配，就是一个很好的监控手段。不过这个方向确实是非常细分的领域，目前我们还没有看到相关的开源产品。



互动时刻

交换机的监控，最常用的网口统计数据都是国际通用 **OID**，但也有很多常用监控指标是私有 **OID**，不同厂商不一样，比如 **CPU**、内存相关的指标都是私有 **OID**。不同型号常用的私有 **OID** 有哪些呢？欢迎留言分享，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。我们下一讲再见！

注：举两个例子，比如 HUAWEI ME60 的 CPU 利用率 **OID** 是 1.3.6.1.4.1.2011.6.3.4.1.2，Juniper 的 CPU 利用率 **OID** 是 1.3.6.1.4.1.2636.3.1.13.1.8。



生成海报并分享



赞 7



提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 11 | 机器监控：操作系统有哪些指标需要重点关注？

下一篇 13 | 组件监控：MySQL的关键指标及采集方法有哪些？

精选留言 (10)

写留言



dobby

2023-02-03 来自四川

snmp结果的解析太繁琐了，纯纯体力活，开源根本没什么好用的库

作者回复: 的确很恶心，不过，如果大家能一起贡献采集配置就好了，就能很快攒起来各种型号设备的采集能力

共 2 条评论 >



1



novoer

2023-04-09 来自福建

是否丢包，丢包率是怎么监控的



Mark.Q

2023-03-07 来自江苏

关于探测监控，比如说证明一个service或者设备是否活着
在老东家是这么做的，发送命令/字符串/等方式，获得一个期许的相应值，然后通过判断该响应值来证明是否存活。这个思路不仅仅局限于tcp icmp http, 很多服务都可以使用。
记得之前对于qemu alive/dead就干过类似的事。



Gong

2023-02-16 来自山东

老师好，请教一下我想监控各终端和服务器的交互流量，有什么办法吗？服务器接的终端数量一千台左右。

作者回复: 一般监控网卡流量就可以了



FORWARD

2023-02-10 来自北京

老师，catagraf采集的网卡流量是32位的，还是64位的？当流量超过1G时，32位的数据会不准确

作者回复: 采集的监控数据都是 float64



MiraClei

2023-02-08 来自北京

请教下载服务器离线状态下，categraf启动会频繁重启，报错信息是请求223.5.5.5，但服务器无法联网，这种情况下是如何解决？

作者回复: config.toml 里的 hostname 配置，不要写 \$ip，如果写 \$ip 就自动探测本机IP，自动探测的时候会请求223.5.5.5



戒贪嗔痴

2023-02-05 来自浙江

最后一张图，最常使用的协议是：v2c？还是V2

编辑回复: 感谢反馈，已经更正了



lei

2023-02-05 来自浙江

请教一下，平时的自定义服务数量非常多，每个服务又会对应多个进程或实例，每个进程会对应多线程，这种情况有什么好的方法监控服务吗？

作者回复: 监控服务, 就看服务对外提供的服务质量, 比如web服务就看可用性、延迟、错误率等, 是有方法论的, 可以参考第9讲。



peter

2023-02-03 来自北京

请问: ping一个机器会占用多少文件句柄?

作者回复: 1个

共 2 条评论 >



hshopeful

2023-02-03 来自湖北

老师有两个问题请教下:

- 1、telegraf 支持 snmp_trap, catagraf 没有支持的原因是什么呢? 难点主要是啥?
- 2、这节课介绍的插件, telegraf 中都有, 想请问下 catagraf 的优势是什么呢?

作者回复: 1, Categraf还没有精力做trap

2, 在前面agent选型的章节, 介绍过哈。除了前面章节介绍的, 另外就是categraf支持metrics、logs、traces三大支柱的数据采集, 集成了mtail, 改良了mysql采集, 改良了system采集, 增加了几个Telegraf不支持的plugin, 各有优劣吧

