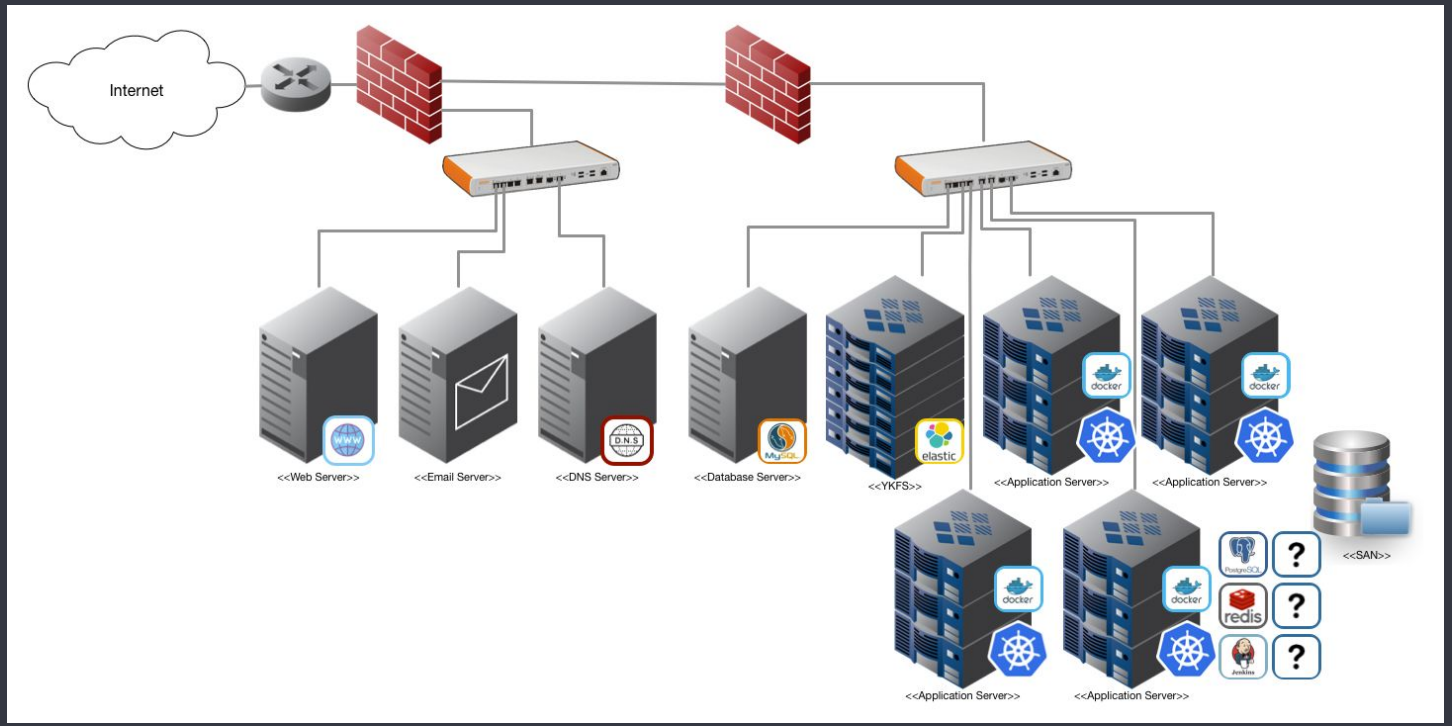
Elastic APM 介绍

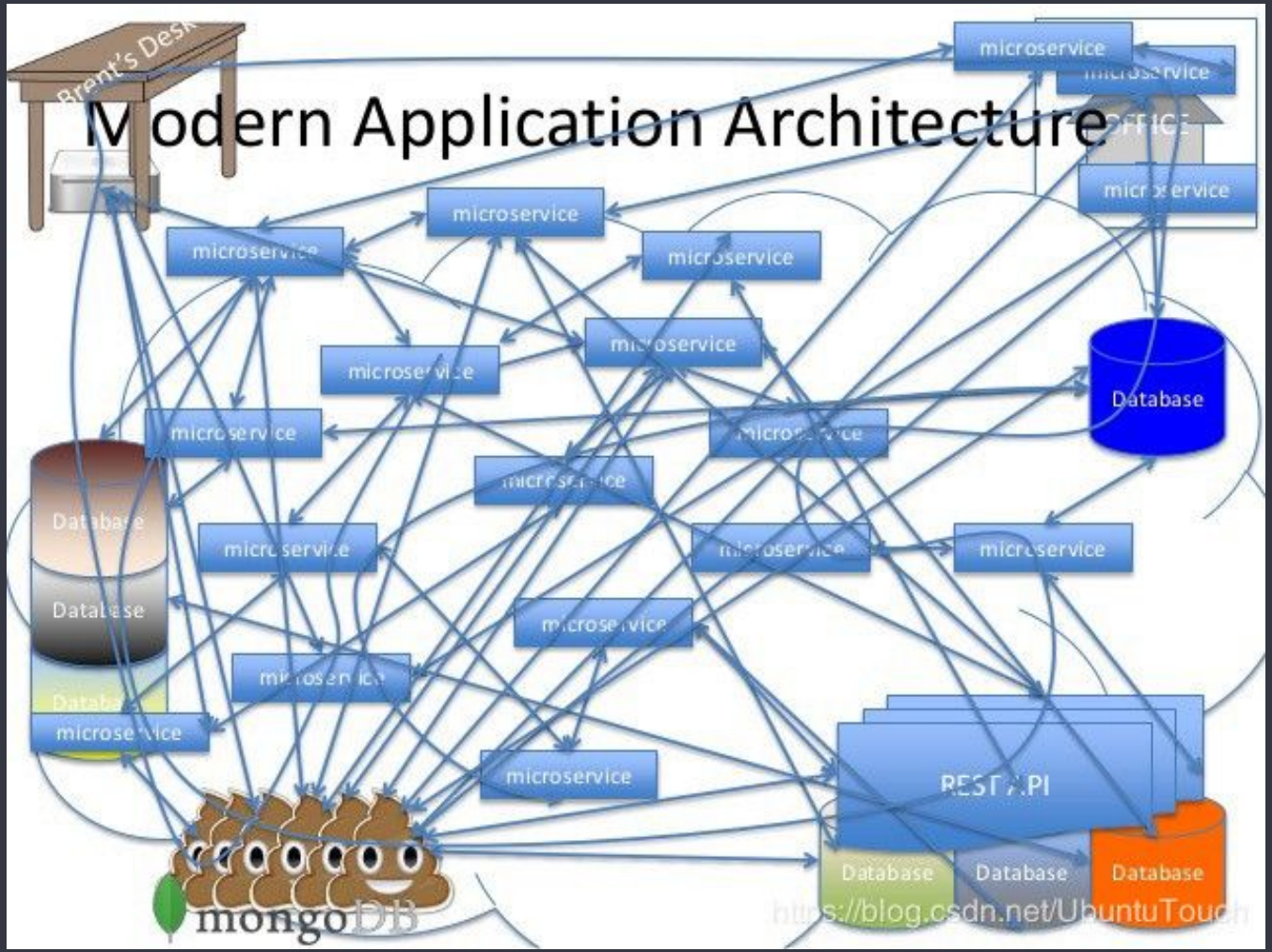
刘晓国
Elastic 社区布道师
2020年4月18日

elasticstack.blog.csdn.net

Elastic 产品生态

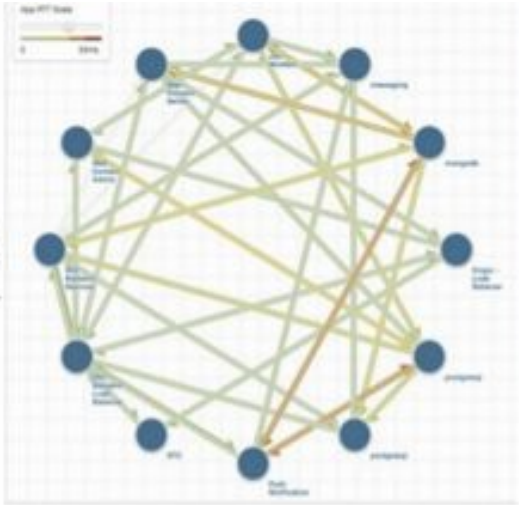
现代服务架构 ...


Internet
<<Web Server>>
<<Email Server>>
<<DNS Server>>
<<Database Server>>
<<YKFS>>
<<Application Server>>
<<Application Server>>
<<SAN>>
<<Application Server>>
<<Application Server>>
docker
elastic
MySQL
redis
?


Modern Application Architecture
microservice
Database
REST API
mongoDB
https://blog.csdn.net/UbuntuTouch

微服务可能非常复杂

Netflix



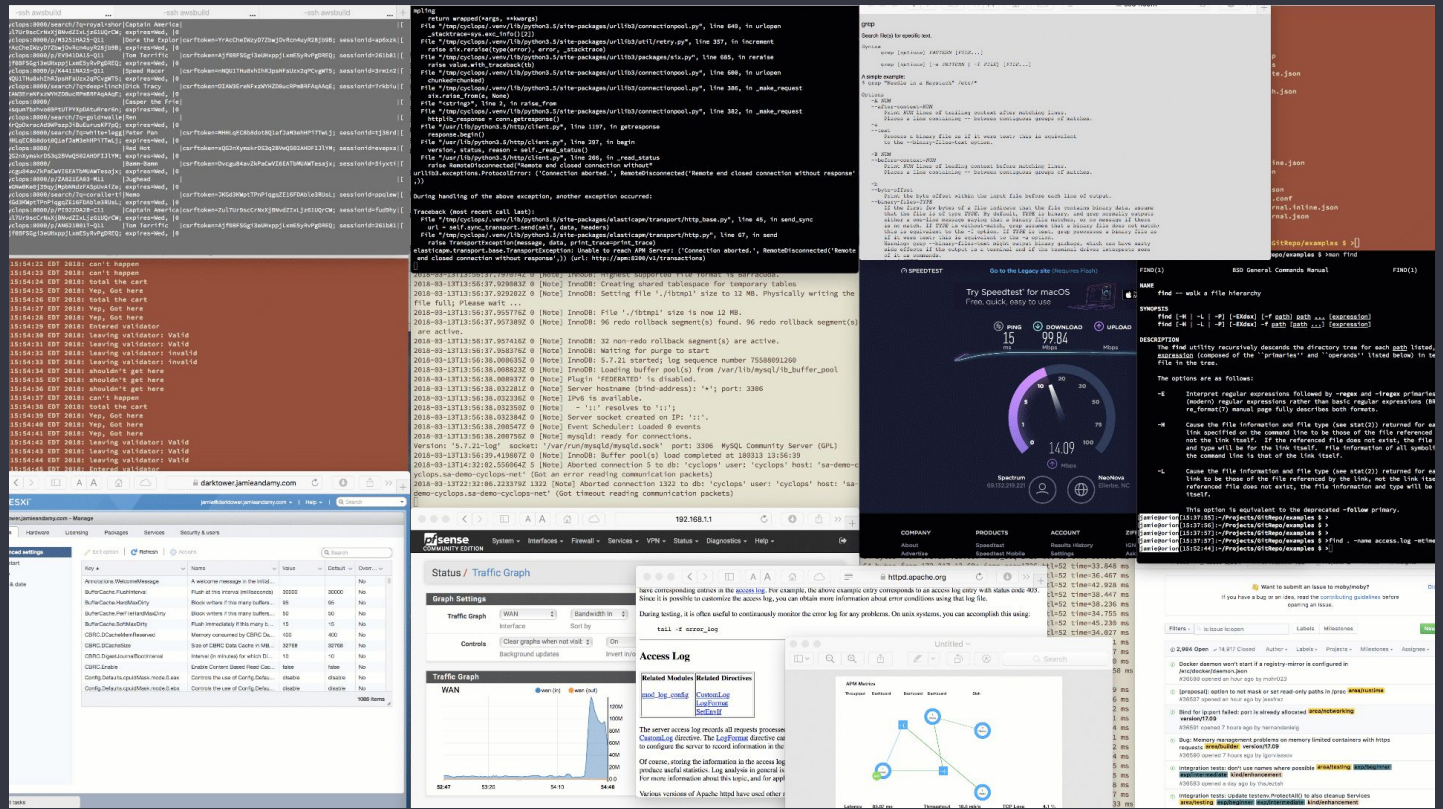
Gilt Groupe (12 of 450)

Twitter

elastic

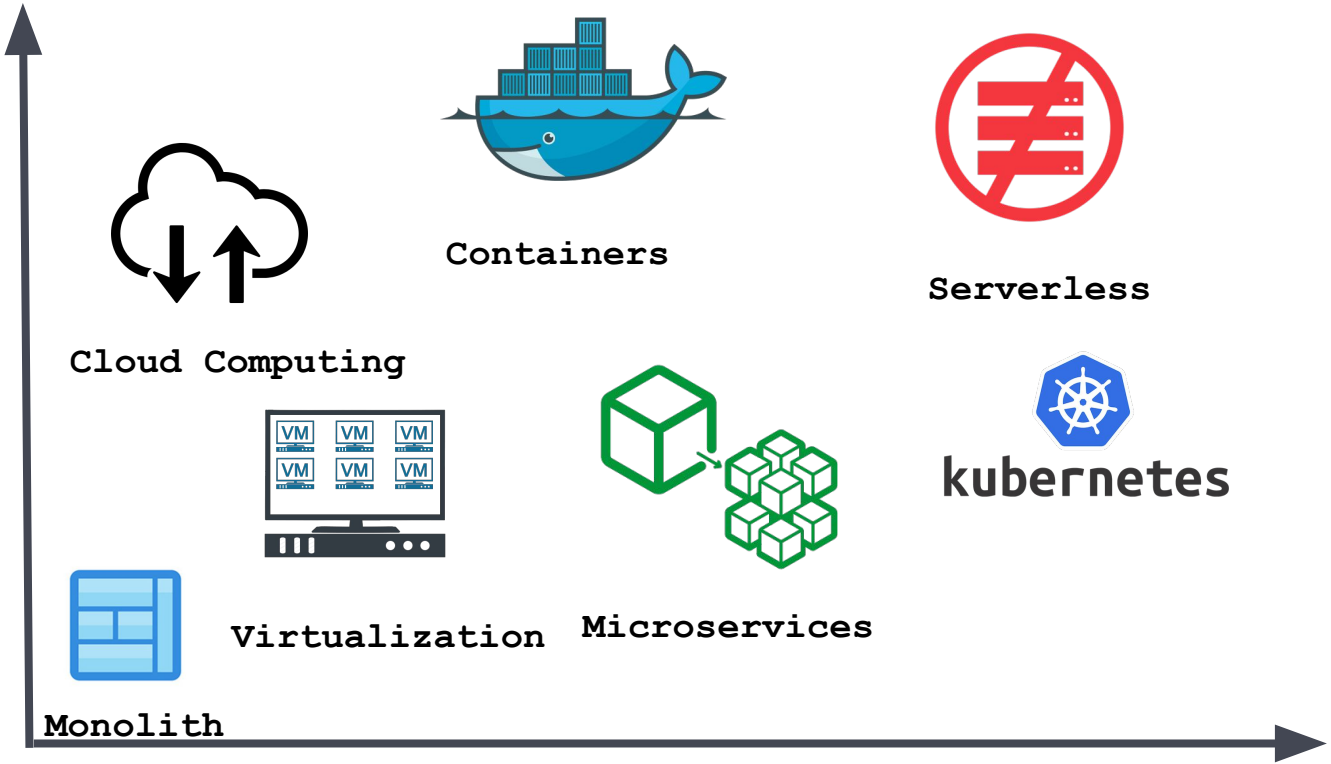
.. 并且当这个发生时...

你打开这些...



12 小时 (及 14 比萨) 以后...

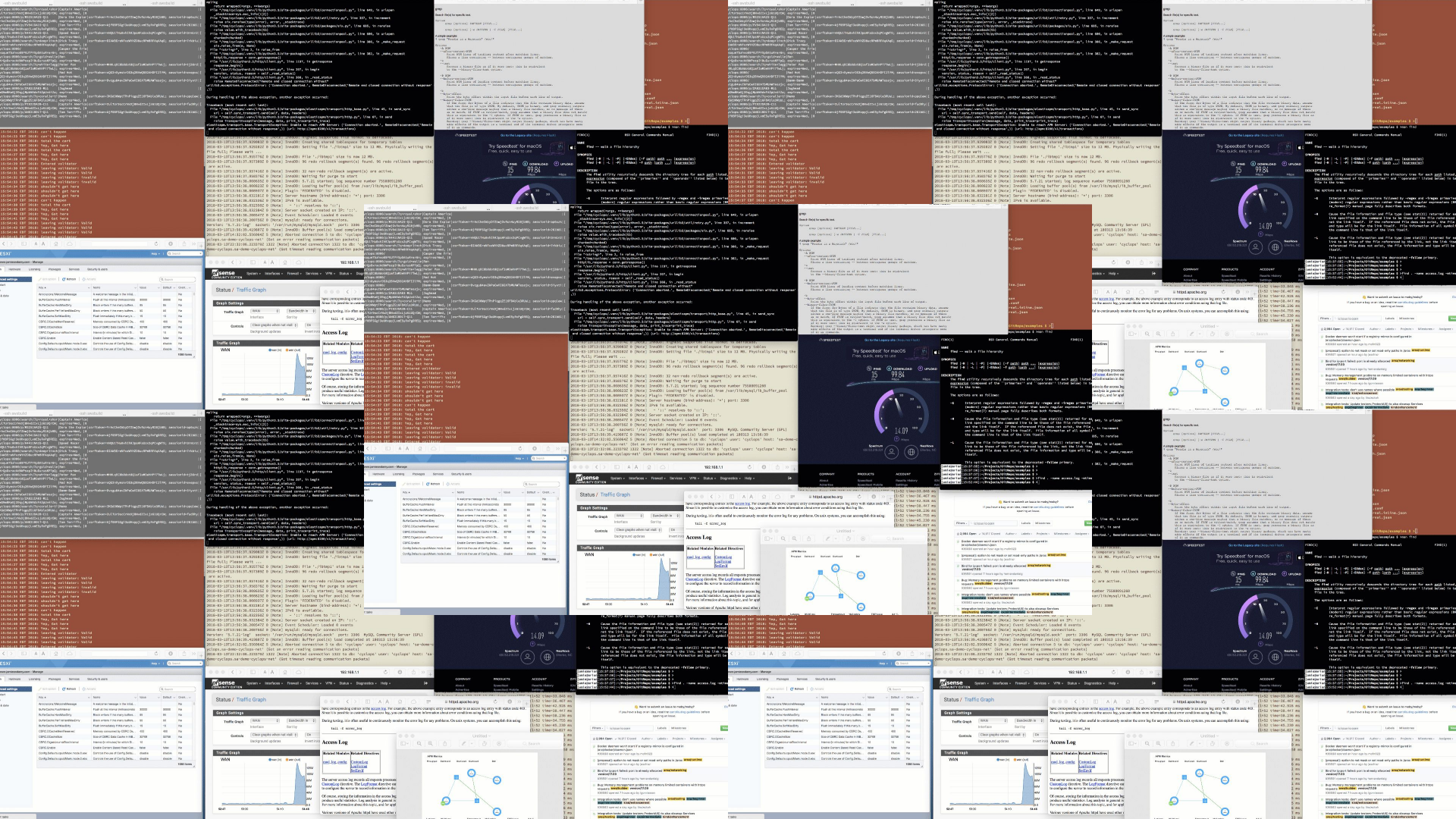
不断发展的架构 = ↑监视复杂性



扩展性

监控复杂性

elastic



Logs和Metrics比较: Logs

每次发生都会生成日志

```
64.242.88.10 - - [07/Mar/2017:16:10:02 -0800] "GET /mailman/listinfo/hsdivision HTTP/1.1" 200 6291
64.242.88.10 - - [07/Mar/2017:16:11:58 -0800] "POST /twiki/bin/view/TWiki/WikiSyntax HTTP/1.1" 404 7352
64.242.88.10 - - [07/Mar/2017:16:20:55 -0800] "GET /twiki/bin/view/Main/DCCAndPostFix HTTP/1.1" 200 5253
```

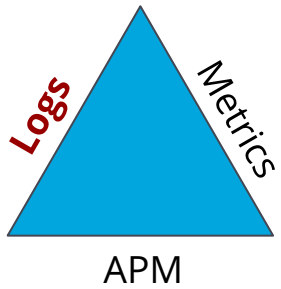
对于每个事件，打印出发生的情况。

优点:

- 通常在运行应用程序或服务的主机上可用
- 易于阅读和可自定义(适用于内部应用)

缺点:

- 日志记录在组件实例级别，而不是应用程序级别
- 如果您不编码输出，则不会打印
- 格式化很重要



我们擅长指标...

```
top - 14:47:00 up 9 days, 16:18,  2 users,  load average: 0.01, 0.02, 0.00
Tasks: 168 total,   1 running, 166 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.2 sy,  0.0 ni, 99.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32946088 total, 24675972 free,  3421124 used,  4848992 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 29010992 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1152 root      20   0 1377184  26688   7340 S   2.3  0.1  92:57.43 metricbeat
18790 ubuntu    20   0 8351988 2.758g  28292 S   2.0  8.8  59:38.09 java
 1099 root      20   0 1118692  51124  20592 S   0.3  0.2 165:35.92 dockerd
 1150 root      20   0  855392  57304  12040 S   0.3  0.2 269:27.91 packetbeat
 3529 elastic   20   0   40520   3604   2980 R   0.3  0.0   0:00.20 top
18960 ubuntu    20   0 1403904 248948  22252 S   0.3  0.8  11:44.81 node
    1 root      20   0  119796   5304   3364 S   0.0  0.0   0:06.90 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:02.30 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   2:38.79 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.35 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:02.21 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:01.92 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.36 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:02.41 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   16 root      rt   0       0      0      0 S   0.0  0.0   0:01.94 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:00.36 migration/2
```

Logs和Metrics对比: Metrics

指标是某些KPI的定期测量

```
07/Mar/2017 16:10:00  all 2.58  0.00  0.70  1.12  0.05  95.55  server1  containerX  regionA
07/Mar/2017 16:20:00  all 2.56  0.00  0.69  1.05  0.04  95.66  server2  containerY  regionB
07/Mar/2017 16:30:00  all 2.64  0.00  0.65  1.15  0.05  95.50  server2  containerZ  regionC
```

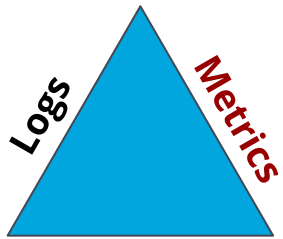
每X秒测量一次CPU负载并打印出来。

优点:

- 显示趋势和历史
- 可以与简单，可预测，可靠的规则一起使用，以捕获事件和异常

缺点:

- 大多数指标记录在组件实例级别，而不是应用程序级别
- 间隔内的异常值可能被“平均”掉了
- 通常基于主机或网络，但是容器可能会歪曲总数



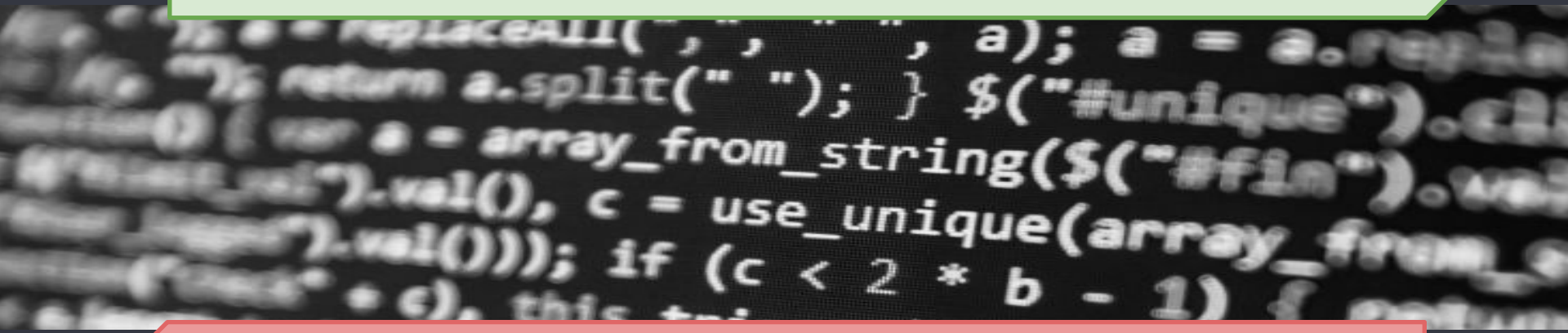
APM

elastic

日志

例子:错误及异常

03:43:59 Request "POST /api/checkout"



03:44:29 Response "/api/checkout 500 ERROR"

elastic

APM 概念类比

跳栏服务:

Transaction: 6.5 seconds

- fence 1: 1 second
- fence 2: 2 seconds
- fence 3: 2 seconds
- fence 4: 2.5 seconds

elastic

APM 概念类比

跳栏服务:

Transaction: 1 second

- fence 1 - 1 second
  错误被捕获

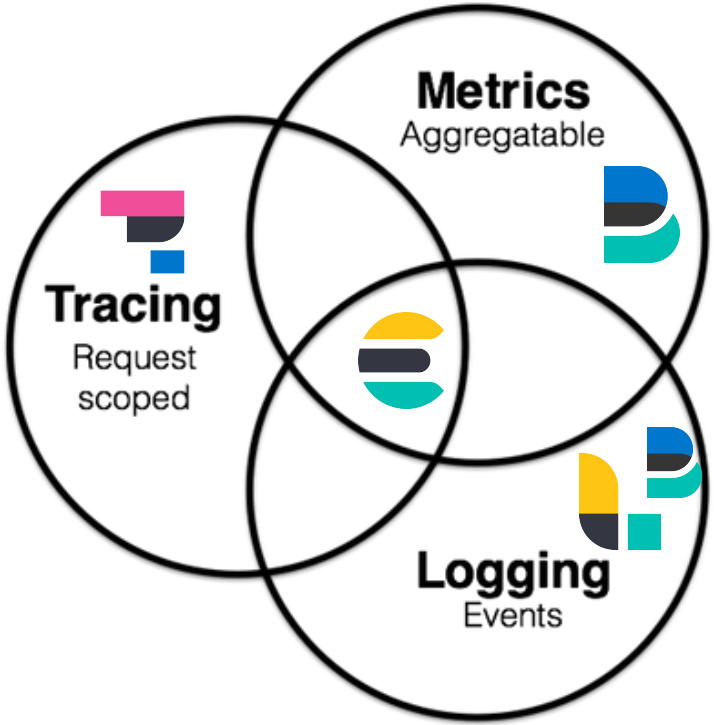
elastic


Elastic 带来了 APM...

Application **P**erformance **M**onitoring

监视和管理软件应用程序的性能和可用性

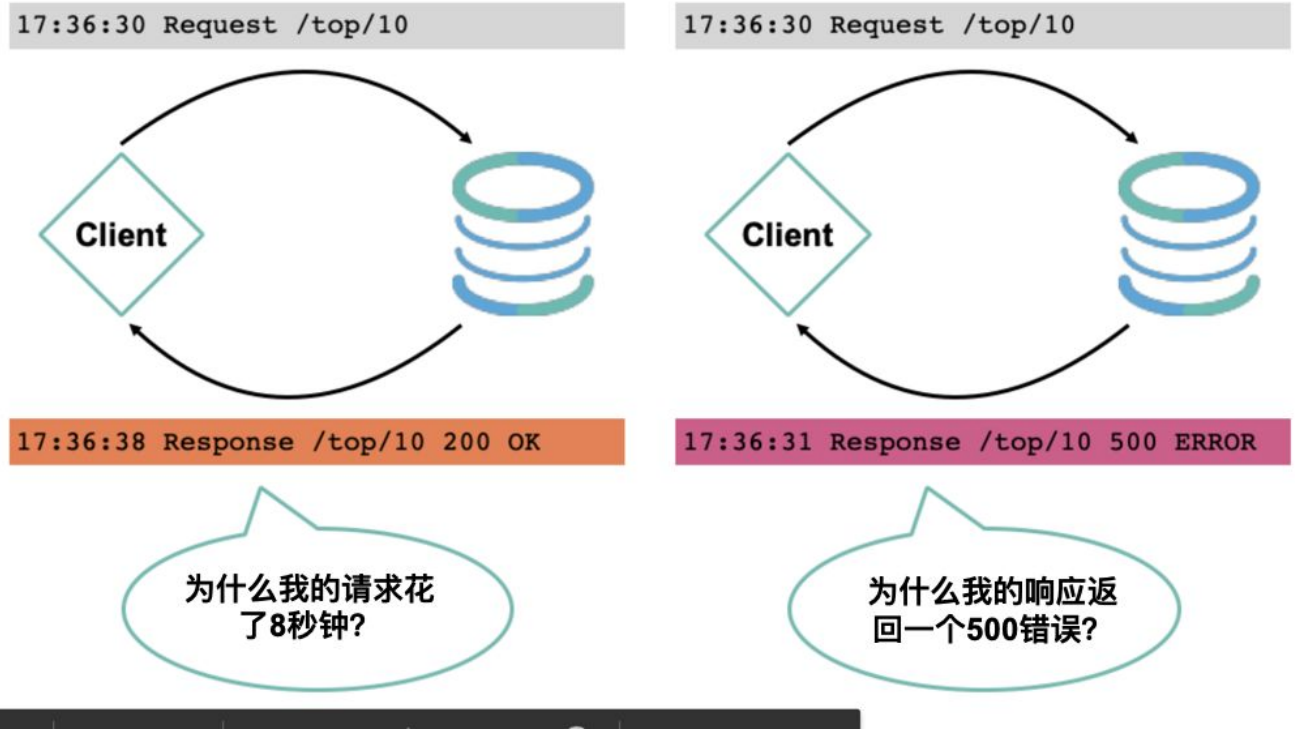
迈向全面可观察性



https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html

什么是APM?

Application Performance Monitoring



elastic

应用程序性能监控- APM

APM 数据: 应用程序分析

Name	Avg. resp. time	95th percentile	Req. per minute	Impact
GET cyclops.views.product_detail.ESProductDetail…	983 ms	1,331 ms	3.7 rpm	
GET cyclops.views.search.ElasticSearchView	775 ms	1,117 ms	1.3 rpm	
POST cyclops.shuup.front.views.basket.BasketView	1,696 ms	2,636 ms	0.6 rpm	

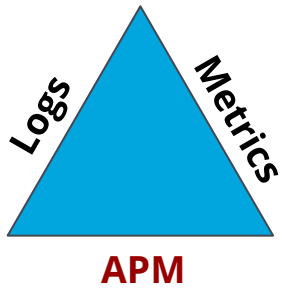
检测会自动告诉您应用程序内部发生了什么

优点:

- 自动了解您的应用程序在做什么
- 可扩展和可自定义，因此你可以定义自己的事务和跟踪

缺点:

- 可以产生很多数据
- 需要向您的应用程序添加库或代理



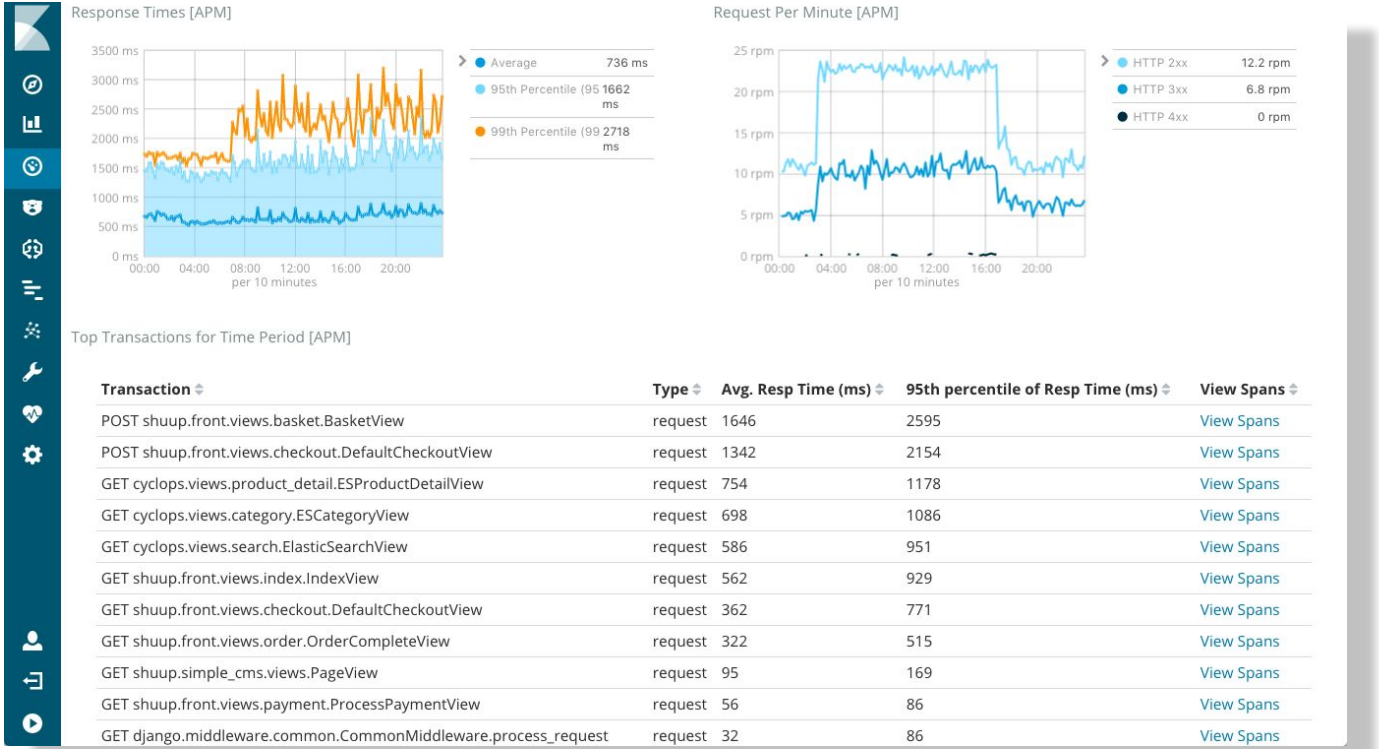
elastic

为什么需要APM?

- APM记录数据库查询，外部HTTP请求以及对应用程序的请求期间发生的其他缓慢操作的跟踪
  - 更容易了解您的应用程序在哪里花时间
- 它收集未处理的错误和异常
  - 更容易调试错误
- 在客户面对性能瓶颈和错误之前先找到它们
- 提高开发团队的生产力

第一个主要的开源APM解决方案

Agents, Server, Dashboards

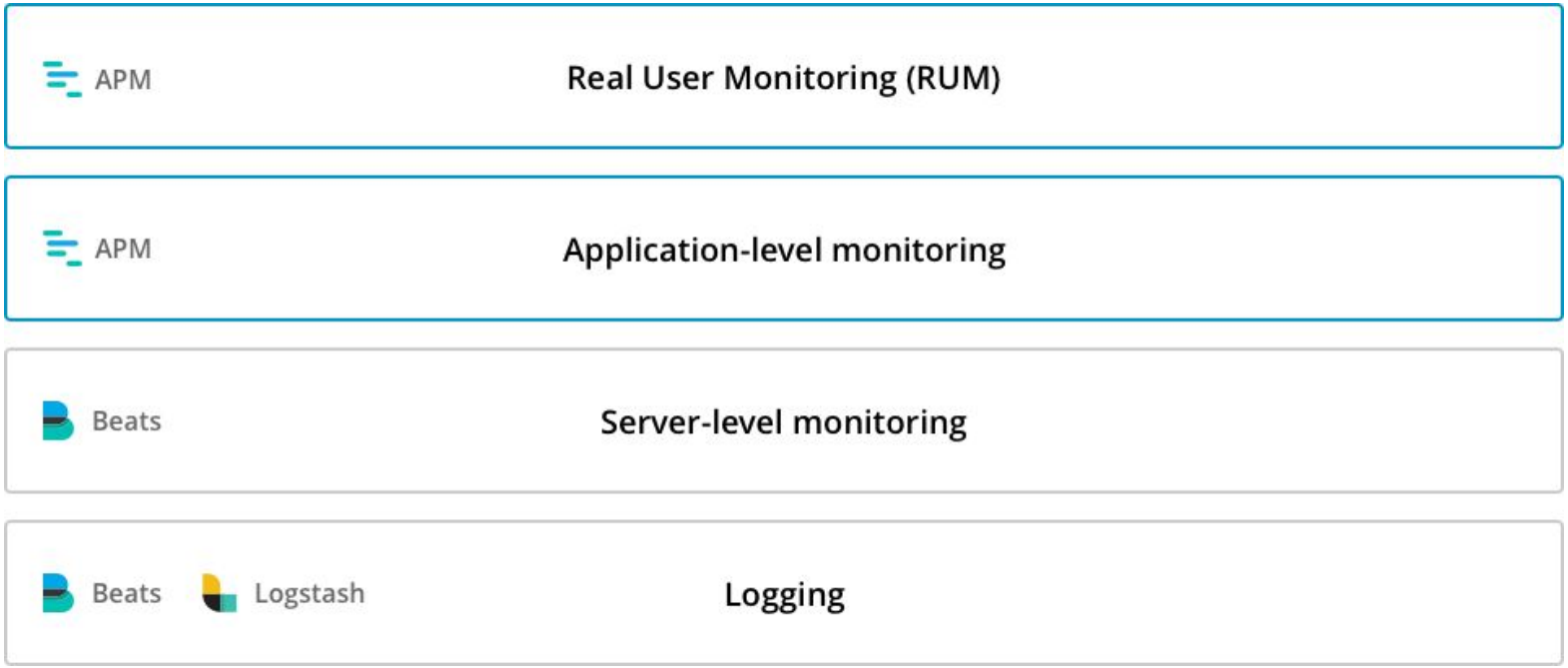


Transaction	Type	Avg. Resp Time (ms)	95th percentile of Resp Time (ms)	View Spans
POST shuup.front.views.basket.BasketView	request	1646	2595	View Spans
POST shuup.front.views.checkout.DefaultCheckoutView	request	1342	2154	View Spans
GET cyclops.views.product_detail.ESProductDetailView	request	754	1178	View Spans
GET cyclops.views.category.ESCategoryView	request	698	1086	View Spans
GET cyclops.views.search.ElasticSearchView	request	586	951	View Spans
GET shuup.front.views.index.IndexView	request	562	929	View Spans
GET shuup.front.views.checkout.DefaultCheckoutView	request	362	771	View Spans
GET shuup.front.views.order.OrderCompleteView	request	322	515	View Spans
GET shuup.simple_cms.views.PageView	request	95	169	View Spans
GET shuup.front.views.payment.ProcessPaymentView	request	56	86	View Spans
GET django.middleware.common.CommonMiddleware.process_request	request	32	86	View Spans

elastic

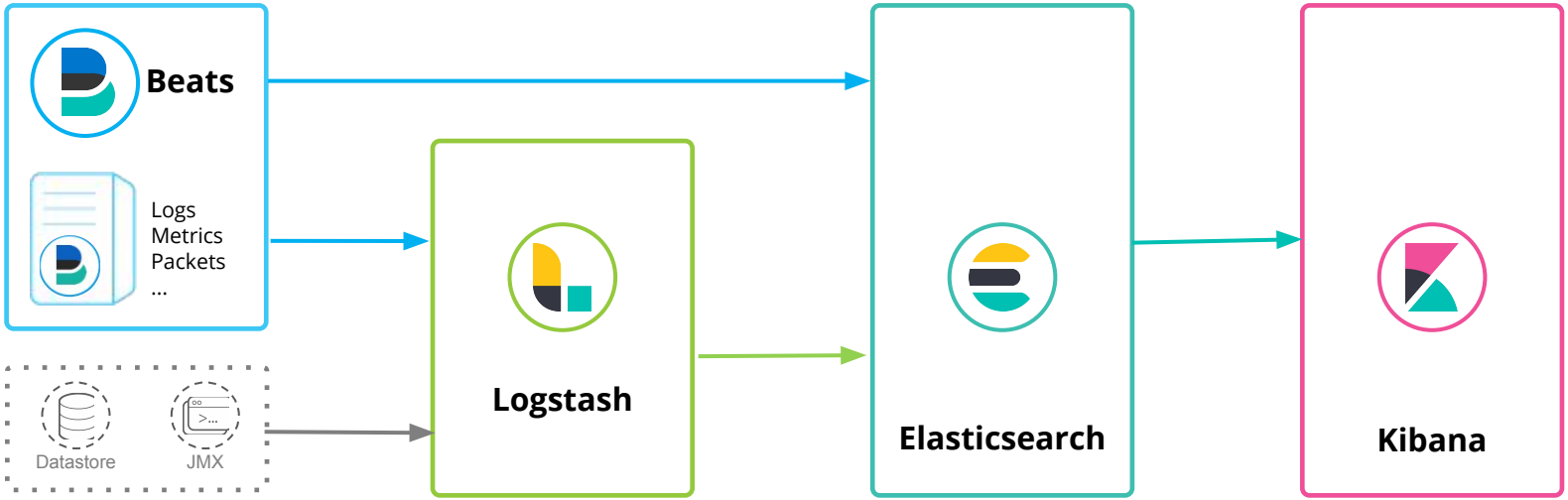
在一个地方进行全栈监控

向Elastic Stack添加最终用户体验和应用程序级监视



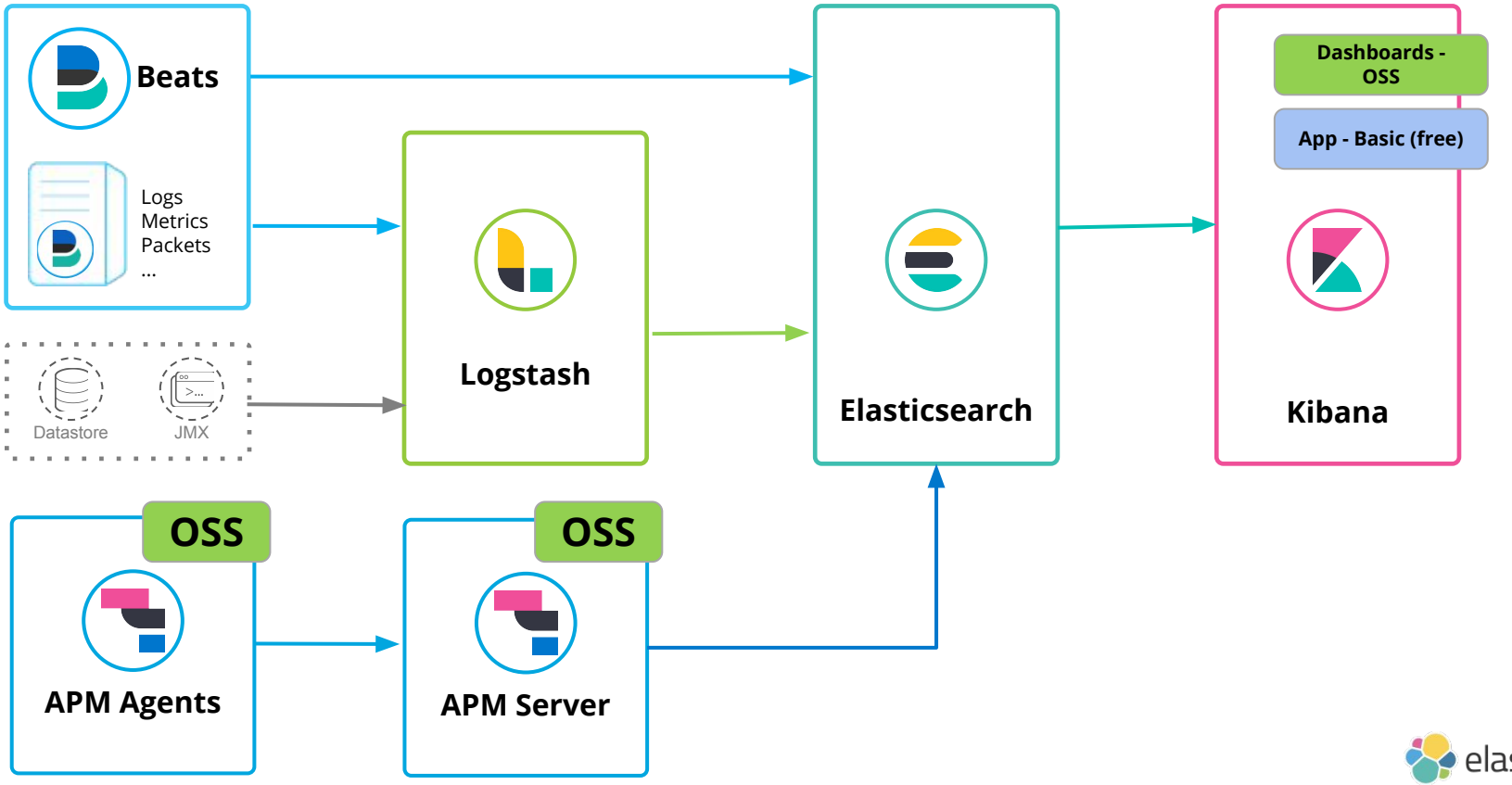
elastic

Where APM fits in the Elastic Stack



elastic

APM 处于Elastic Stack的什么位置？



elastic

支持的语言及框架

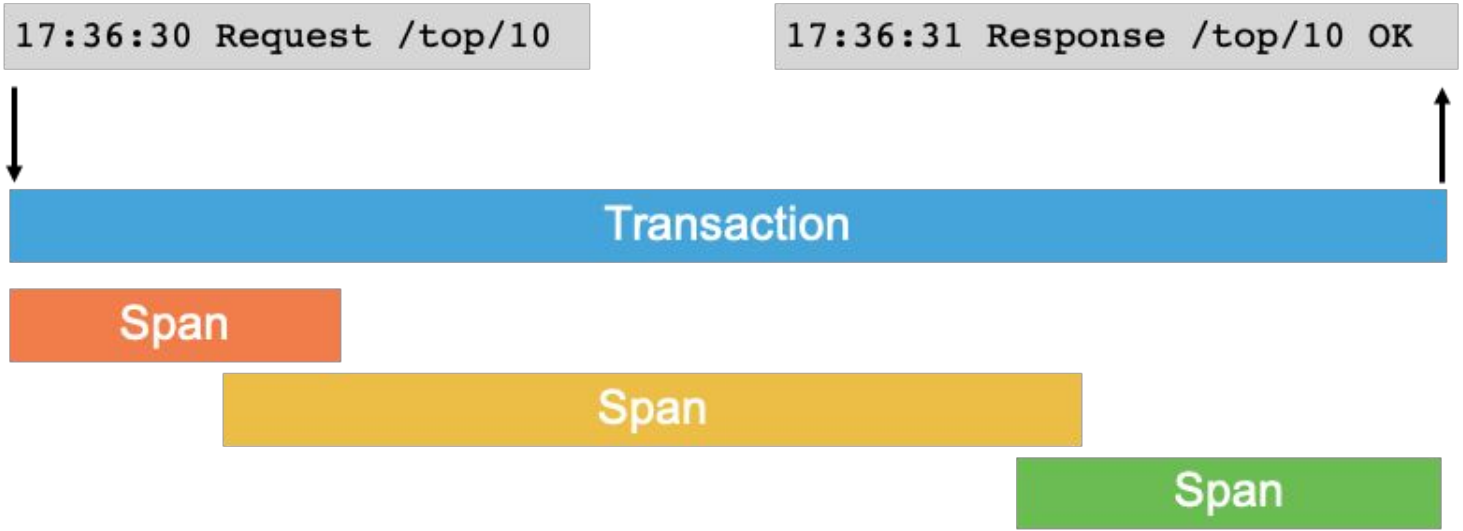
elastic

APM 术语

- ***Service***
  - 在APM代理配置中进行设置，以将特定的APM代理组标识为单个服务，这是一种逻辑上标识一组事务的方法。
- ***Transaction***
  - 组成一个服务的请求和响应，例如登录api调用，每个调用由单独的span组成。
- ***Span***
  - 事务中的单个事件，例如方法 调用，数据库查询或缓存插入或检索，即需要花 费时间才能完成的任何事件。
- ***Errors***
  - 具有匹配的异常或日志消息的异常 组
- ***Trace***
  - 代表请求的整个过程

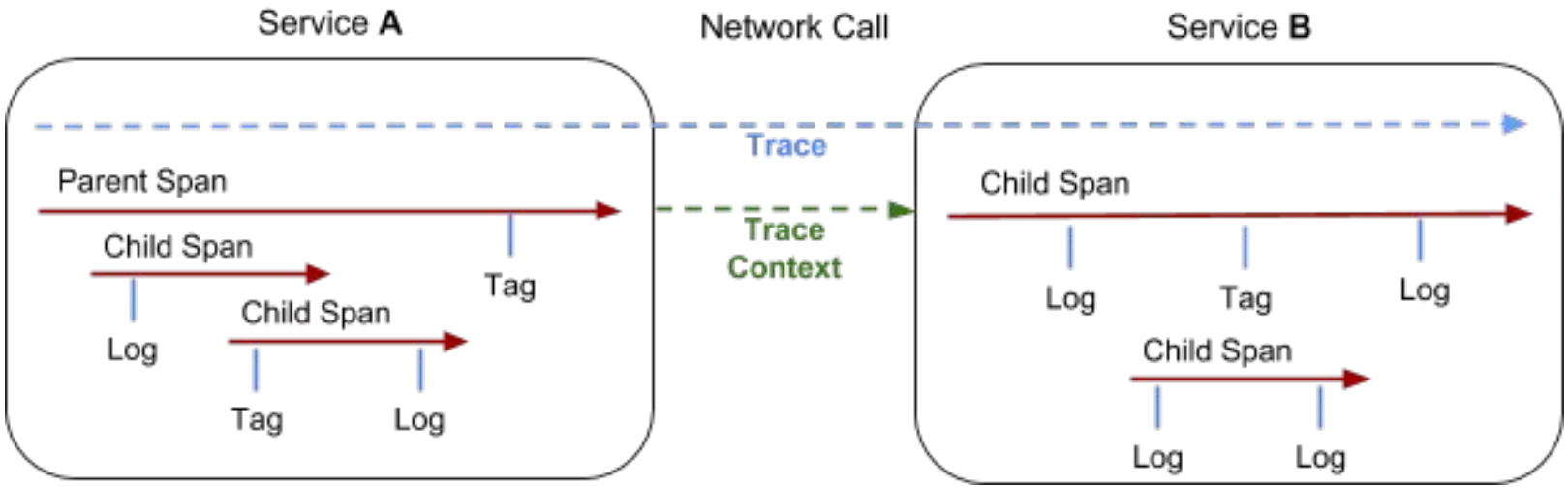
APM 术语

Transaction, trace & span



elastic

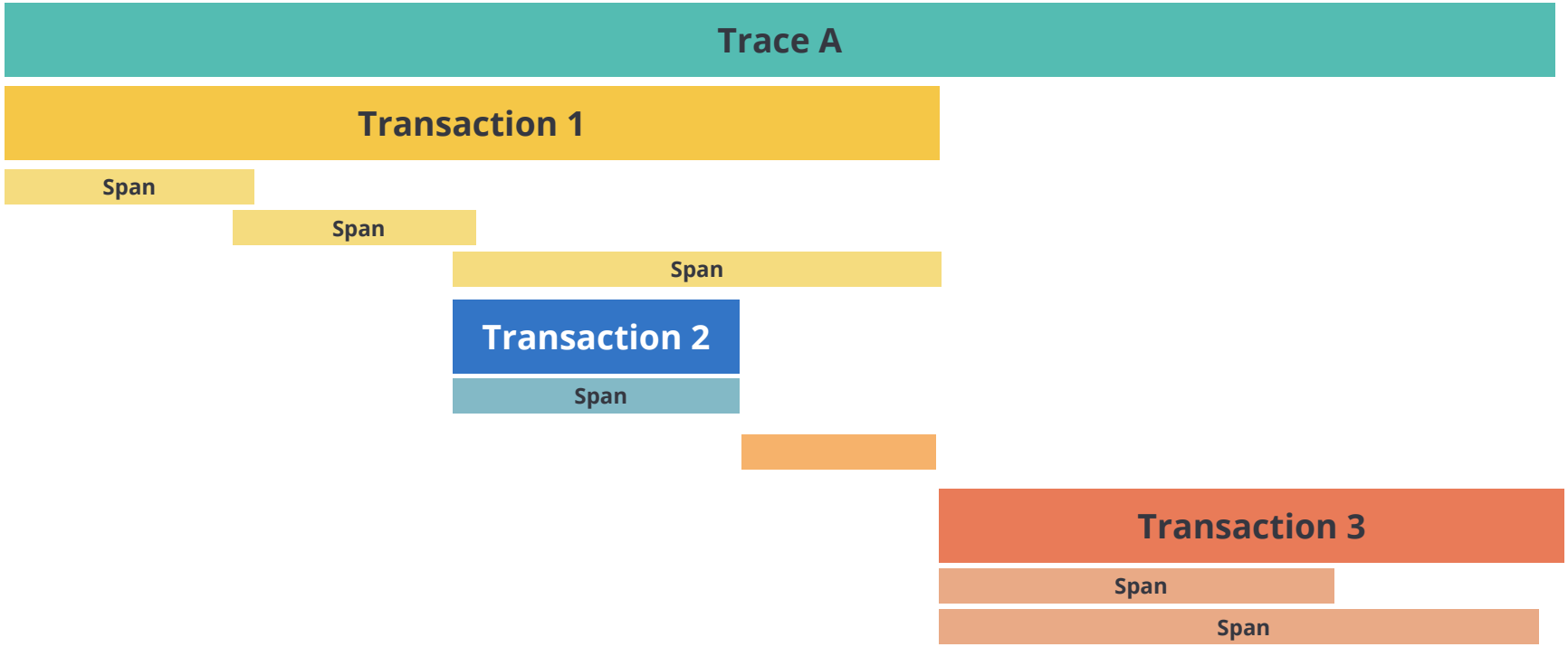
分布式跟踪



elastic

分布式跟踪

多个微服务


Trace A
Transaction 1
Span
Span
Span
Transaction 2
Span
Transaction 3
Span
Span

分布式跟踪如何工作?

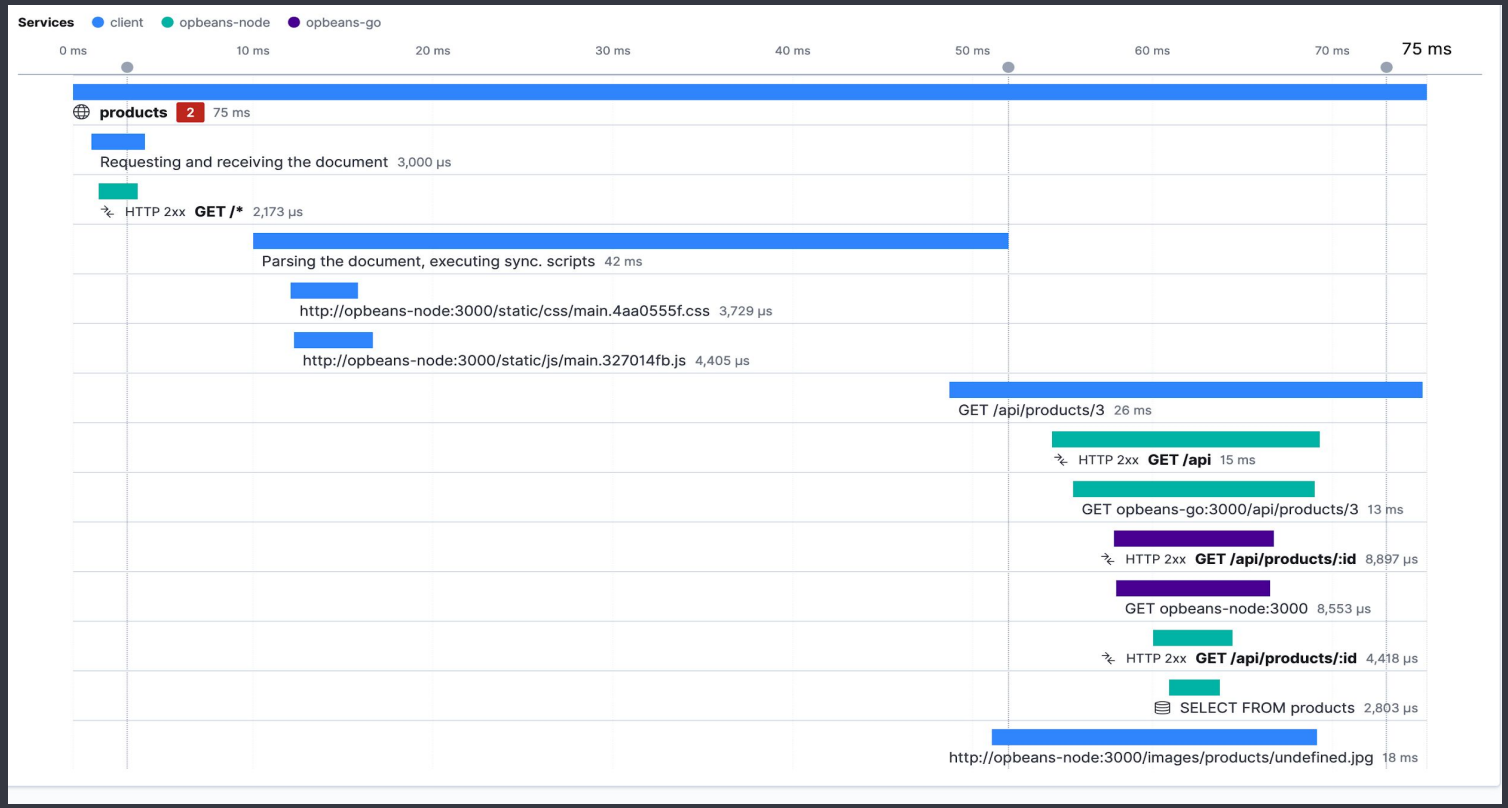
跟踪 id 传播

APM agent 写/读 唯一的跟踪 id

```
traceparent:
00-                                 // version
0af7651916cd43dd8448eb211c80319c-   // trace id
b7ad6b7169203331-                   // parent span id
01                                  // flags (sampling)
```

分布式跟踪

针对微服务



elastic

Span details

View span in Discover

Service
opbeans-python

Transaction
GET opbeans.views.product_type

Name
GET opbeans-ruby:3000

Duration
15 ms

% of transaction
77.8%

Type
external

Subtype
http

HTTP URL

```
http://opbeans-ruby:3000/api/types/1
```

Stack Trace | Labels

> 3 library frames

opbeans/views.py in **wrapped_view** at **line 57**

```
55.             logger.info("dt_ping_pong_proxy", url=url)
56.             try:
57.                 other_response = requests.get(url, timeout=15)
58.             except requests.exceptions.Timeout:
59.                 logger.error("dt_ping_pong_timeout", service=other_service)
```

> 39 library frames

bin/gunicorn in **<module>** at **line 10**

```
8.  if __name__ == '__main__':
9.      sys.argv[0] = re.sub(r'(-script\.pyw?|\.exe)?$', '', sys.argv[0])
10.     sys.exit(run())
```

让我们开始Demo吧