

THE NEW STACK

Cloud Native Observability for DevOps Teams

新堆栈

DevOps 团队的云原生可观察性

Alex Williams,创始人兼发行人

电子书团队和客座作者：

凯瑟琳·帕格尼尼,作者

丹耶尔费舍尔,作者

弗朗西斯·埃斯佩尼多,作者

Gabriel H. Dinh,执行制片人

希瑟·乔斯林 (Heather Joslyn),电子书编辑

杰森·摩根,作者

Joab Jackson,主编

朱迪·威廉姆斯,文案编辑

Libby Clark,战略发展副总裁

彼得·普茨,作者

史蒂夫蒂德威尔,作者

苏珊厅,文案编辑

支持团队：

Benjamin Ball,销售和客户管理总监

米歇尔马赫,助理编辑

© 2021 新堆栈。版权所有。

20220612

目录

赞助商	4
介绍.....	5
可观察性在云原生应用程序中的作用	
了解云原生可观察性堆栈	11
超越指标、日志和跟踪的可观察性	19
可观察性如何支持 DevOps	28
实践中的 KUBERNETES 可观察性	
什么以及如何登录 Kubernetes	37
Kubernetes 可观察性策略	45
为什么开发人员应该学习 Kubernetes	49
总结一下.....	53
披露	54

赞助

我们感谢电子书赞助商的支持：



Mezmo,以前称为 LogDNA,是一个可观察性平台 ,用于管理您的数据并对其采取行动。它摄取、处理和路由日志数据以推动企业级应用程序开发和交付、安全性和合规性用例。

介绍

如果您正在阅读本文,您可能已经在使用云原生应用程序和架构,或者您的组织正在踏上云之旅。如果是这样,您已经熟悉云原生环境提供的压倒性选择。如此多的工具和如此多的机会做出错误的决定。

一句话,太复杂了。

即使您的团队只监督一个或几个微服务集群,这些集群也可能部署在多个公共云或多个公共云的组合中

云和本地服务器。更复杂。

当出现异常情况时 延迟、应用程序编程接口 (API) 调用激增、基本服务突然中断 你怎么知道是什么原因造成的?你怎么知道它是一个孤立的事件还是一个会导致一切崩溃的小故障?

问题是,除非您的整个团队都具有完全的可观察性,否则您无法知道。

对于一个组织的能力而言,没有什么比可观察性更重要的了。也许没有比这更广泛的了

误解。

可观察性意味着从系统的外部输出推断系统的内部状态。但这不仅仅是查看系统中正在发生的事情的能力。它是理解这一切的能力,收集和分析您需要的信息以防止事件发生,并在事件发生时追踪其路径,尽管采取了各种保护措施,以确保它们不会再次发生。

传统上,可观察性一直是运维工程师的职责。但随着 DevOps 团队的出现,以及更多责任“向左转移”给开发人员,这已成为每个团队成员的工作。如果您正在构建应用程序,

可观察性不能在应用程序生命周期的后期降级为“附加”状态。不这么想就像造一辆汽车,把速度表、里程表和仪表灯留给经销商安装。

在[一项调查](#)中VMware 于 2021 年 1 月对 300 多名 IT 专业人员进行了调查
Tanzu,84% 的参与者表示他们的云应用程序会更好
如果包括开发人员在内的更多利益相关者拥有可用性和性能
了解其系统的整体基础架构和性能指标。

当前关于可观察性的讨论始于 Kubernetes 等改变游戏规则的云原生技术引入之前。在[2013 年一篇](#)被广泛引用的博文中由 Cory Watson 撰写,科技界了解了 Twitter 的工程师如何设法保持

在公司从单一系统转变为分布式系统时跟踪他们的系统
建筑学。在这个时候,正如 Watson 所描述的那样,Twitter 专注于其可观察性
收集和监控指标以及生成的可视化的努力
从它收集的数据点:

“图表通常是临时创建的,以便快速共享信息
在部署或事件期间在团队中,但也可以在仪表板中创建和保存它们。
一个命令行工具
工程师可以使用仪表板创建、通用指标的可重用组件库和自动化 API。

日志记录和跟踪在标题下的单个段落中进行了处理
“相关系统。”

Watson 写道,Twitter 创建了一个命令行工具来帮助其工程师创建他们自己的仪表板来跟踪
他们的指标生成的图表:

“Twitter 的平均仪表板包含 47 个图表。如果您在办公室闲逛,通常会在大屏幕或工程师的显示器上看到这些仪表板。 Twitter 的工程师生活在这些仪表盘!

随着云计算十年的推进,越来越多的工程师开始使用他们的仪表板。他们了解到,仅仅监控数据点是不够的。因此,这种观念传播开来,即可观察性不仅仅意味着监控,而是基于

三大支柱,后来被称为:

1. Metrics:衡量系统中各种活动的指标。
- 2.跟踪:请求通过分布式系统时所采用的路径。
- 3.日志:系统内活动的记录。

越来越多地,围绕可观察性的讨论正在超越三大支柱,采取更加微妙的观点。人们对这三个支柱如何组合在一起有了更多的认识,并且更加强调分析。 DevOps 团队越来越认识到衡量满足服务真正重要的因素的重要性

水平目标 (SLO) 。

管理人员在高流动率和相对较少的人才库中苦苦挣扎,正试图找出如何减轻“传呼疲劳”的人力成本。运营工程师需要随时响应警报。白天或晚上,这可能预示着业务关键事件,也可能不预示着。

SLO 是 Google 站点可靠性工程团队在其[SRE书中记录的一个概念](#),根据每个组织甚至团队的目的,可能会有很大差异,例如为一定数量的请求实现特定延迟或确定有多少客户可以同时在线购物车应用程序中进行购买。服务水平指标 (SLI) 是照亮强大的信号

可观察性过程,并且可以显示团队是否正在按计划满足其 SLO,或者是否一个问题正在酝酿。

而且,如前所述,分布式系统和云原生技术增加了可观察性的额外复杂层。毕竟,Kubernetes 运行无处不在,而且“无处不在”可能很难追踪。

在 VMware Tanzu 调查中,90% 的参与 IT 专业人士表示

分布式应用程序带来的监控挑战比其他应用程序大一个数量级。

超过 80% 的调查参与者表示,传统的监控工具不是足以跟踪现代云应用程序。只有 8% 的受访者表示他们对其组织当前的监控工具“非常满意”,并且过程。

云技术并不总是易于观察。例如,普通的 Kubernetes 仅通过 kubectl 提供非常基本的功能,用于检查集群中对象的状态,并且没有成熟的本地日志记录解决方案,正如 Mezmo 的高级技术合作伙伴项目经理 Francis Espenido 在他的章节中所写的那样这本电子书。

但克服这些挑战可以为企业带来回报。

在 VMware Tanzu 调查中,92% 的受访者表示可观察性驱动更好商业决策。可观察性如何嵌入其中的一个例子企业经营的方式涉及运动服装零售商阿迪达斯。

阿迪达斯发现,随着规模的扩大,它需要让可观察性变得更容易,根据[拉斯特科·武卡西诺维奇的说法](#),该公司的解决方案架构总监。所以它建立了自己的整体监控系统这使它不仅可以收集和看技术指标,也要看商业数据。

其全球 DevOps 团队现在每天编译代码超过 10,000 次。和阿迪达斯全面数字化转型助力其[电商收入从 2012 年为 4700 万美元,2020 年为 47 亿美元。](#)

对于开发人员而言,拥有更多关于可观察性的知识 并构建安全、可观察的应用程序,这些应用程序可以轻松满足 SLO 意味着为总体业务目标做出更多贡献。根据[451 Research 2019 年的一项调查](#),开发人员有 55% 的时间花在维护和管理满足当前业务需求的自定义应用程序上;只花费了 45% 的时间

构建新的应用程序以帮助企业从中脱颖而出
公司的竞争。

根据 451 Research 的[2020 年可观测性报告](#),受相扑委托
逻辑,更加关注 SLO 可以帮助开发人员将更多时间花在应用程序上
推动新业务:

“通过了解描述绩效的关键目标
这对最终用户很重要,开发人员可以优先考虑他们在现有应用程序上
所做的工作以解决最重要的性能问题,而不是例如对用户没有负面影响的
基础设施或应用程序异常。

云原生可观测性的故事还在不断增加新的篇章。科技界正在关注[OpenTelemetry](#) 一个旨在创建标准化的开源项
目
一组工具、API 和软件开发工具包 (SDK) 现在是一个沙盒
[云原生计算基金会](#)的项目。

在这本电子书中,The New Stack 收集了一些关于当前的最佳文章
可观察性状态,来自 Mezmo、Buoyant 的专家的贡献,
Dynatrace 和蜂巢。它针对 DevOps 团队的每个成员,为全栈参与确保云原生应用程序和系统平稳运行并让客户
满意提供了理由。将应用程序代码 “翻墙”并让运维工程师处理问题的日子

后果结束了。

希瑟·乔斯林,The New Stack

[希瑟·乔斯林](#)是 The New Stack 的功能编辑器。此前,她曾担任云原生咨询公司 Container Solutions 的主
编,以及 The Chronicle of Philanthropy 和 Baltimore City Paper 的编辑/记者。在 Twitter 上,她是
@ha_joslyn。

可观察性在云原生中的作用应用

第一章

认识云原生 可观察性堆栈

[来源文章](#)由 Buoyant 的 Catherine Paginini 和 Jason Morgan 于 2021 年 5 月 17 日出版。

编者按:要事第一。有哪些工具可以帮助 DevOps 团队提高他们对云原生应用程序的可观察性?他们解决了哪些问题,又是如何解决的?在这里,来自 Buoyant 的专家提供了[云原生计算基金会云原生景观](#)的鸟瞰图,专注于可以实现可观察性的工具。

让我们从定义可观察性和分析开始。可观察性是一个系统

大号 描述系统可以被理解的程度的特征

从其外部输出。以中央处理器 (CPU) 时间衡量,内存、磁盘空间、延迟、错误等,计算机系统可能或多或少可观察的。另一方面,分析是一种活动,您可以在其中查看可观察的数据并理解它。

为确保没有服务中断,您需要观察和分析每一个

应用程序的一个方面,以便立即检测和纠正任何异常。

这就是云原生计算基金会 (CNCf) 的这个类别

景观就是一切。它贯穿并观察所有层,这就是它开启的原因

侧面而不是嵌入特定层。

此类别中的工具分为日志记录、监控、跟踪和混沌工程。请注意,虽然在这里列出,但混沌工程是

可靠性高于可观察性或分析工具。

日志记录

这是什么

应用程序发出源源不断的日志消息流,描述它们在任何给定时间所做的事情。这些日志消息捕获发生在

系统 例如失败或成功的操作、审计信息或健康事件。

日志工具收集、存储和分析这些消息以跟踪错误报告和相关数据。与指标和跟踪一起,日志记录是可观察性。

它解决的问题

收集、存储和分析日志是构建现代平台的重要组成部分。日志记录工具可以帮助或执行其中一项或全部任务。一些

工具处理从收集到分析的各个方面,而其他工具则专注于

收集之类的单一任务。所有日志记录工具都旨在帮助组织控制他们的日志消息。

它如何帮助

在收集、存储和分析应用程序日志消息时,您将了解应用程序在任何给定时间正在通信的内容。但请注意:日志代表应用程序可以故意发出的消息。他们不一定会查明给定问题的根本原因。

也就是说,随着时间的推移收集和保留日志消息是一项非常强大的功能,将帮助团队诊断问题并满足监管和合规要求。

技术101

虽然收集、存储和处理日志消息绝不是一个新问题,但[云原生模式](#)和 Kubernetes 使我们处理日志的方式发生了重大变化。适用于虚拟和

日志记录

第 01 章:了解云原生可观测性堆栈



来源:<https://landscape.cncf.io>

© 2021 THE NEW STACK

图 1.1:目前包含在 CNCF 格局中的日志记录工具。

物理机器,比如将日志写入文件,不适合容器化
文件系统不会比应用程序更持久的应用程序。在云原生中
Fluentd 等日志收集工具与应用程序容器一起运行,并直接从应用程序收集消息。然后将消息转发到中央日志存储以进行聚合和分析。在撰写本文时,Fluentd 是

该领域唯一的 CNCF 项目。

- 流行语:日志记录
- 热门项目: Fluentd 和 Fluentbit、Elastic Logstash

监控

这是什么

监控是指检测应用程序以收集、汇总和分析日志和指标,以提高我们对其行为的理解。虽然日志描述了特定事件,但指标是在给定时间点对系统的测量。

它们是两种不同的东西,但两者都是全面了解系统健康状况所必需的。监控包括从观察单个节点上的磁盘空间、CPU 使用率和内存消耗到执行详细的综合事务以查看系统或应用程序是否正确及时地响应的所有内容。监控系统有许多不同的方法

和应用程序。














































它解决的问题

运行应用程序或平台时,您希望它按设计完成特定任务,并确保它只能由授权用户访问。监控可让您了解其是否正常、安全且经济高效地工作;仅由授权用户访问;和/或您可能正在跟踪的任何其他特征。

它如何帮助

良好的监控使操作员能够在事件发生时快速并可能自动做出响应。它提供了对系统当前健康状况的洞察和

图 1.2:目前 CNCF 环境中包含的监控工具。

监控									
 Prometheus		 cortex		 Thanos		 Amazon CloudWatch	 APPDYNAMICS	 Application High Availability Service	 ManageEngine® Applications Manager
CNCF 毕业		CNCF 孵化		CNCF 孵化		 AppNeta	 appoptics	 Aternity	 Azure Monitor
 beats	 bluematador	 catchpoint	 centreon	 checkmk	 chronosphere	 CloudHealth	 DATADOG	 dynatrace	 epsagon
 falcon	 FLOWMILL	 Google Stackdriver	 Gradle	 Grafana	 graphite	 Honeybadger	 ICINGA	 influxdata	 INSTANA
 IRONdb	 kiali	 kuberhealthy	 LeanIX	 LogicMonitor	 logz.io	 M3	 mackerel	 Nagios	 NETDATA
 New Relic	 NexClipper	 Nightingale	 NODESOURCE	 OPENMETRICS	 OPENTSOB	 opstrace	 OverOps	 replex	 ROOKOUT
 Sensu	 SENTRY	 SignalFx	 Skooner	 SOSIVIO	 StackState	 StormForge	 sysdig	 听云 TINGYUN	 trickster
 turbonomic	 VECTOR By Timber.io	 VICTORIA METRICS	 WAVEFRON by vmware	 weave cloud	 weave scope	 WhaTap	 ZABBIX		

观察变化。监控跟踪从应用程序运行状况到用户行为的所有内容,是有效运行应用程序的重要组成部分。

技术101

云原生环境中的监控通常类似于监控传统应用程序。您需要跟踪指标、日志和事件以了解应用程序的健康状况。主要区别在于一些托管对象是短暂的,这意味着它们可能不会持久,因此将您的监控与自动生成的资源名称联系起来并不是一个好的长期策略。这个领域有许多 CNCF 项目,主要围绕 CNCF 毕业项目 Prometheus 展开。

- 流行语:监控、时间序列、警报、指标
- 热门项目/产品: Prometheus、Cortex、Thanos、Grafana

追踪

这是什么

在微服务世界中,服务不断地通过网络相互通信。跟踪是日志记录的一种特殊用途,它允许您跟踪请求在分布式系统中移动时的路径。

它解决的问题

了解微服务应用程序在任何给定时间点的行为是一项极具挑战性的任务。虽然许多工具提供了对服务的深入洞察行为,可能很难将单个服务的操作与更广泛的服务联系起来了解整个应用程序的行为方式。

它如何帮助

跟踪通过向发送的消息添加唯一标识符来解决此问题应用。该唯一标识符允许您跟踪或追踪个人

追踪

第 01 章:了解云原生可观察性堆栈



JAEGER

CNCF 毕业 CNCF 孵化



OPENTRACING



Aspecto



elastic apm



honeycomb.io



Lightstep



OpenTelemetry





Skywalking



SOFATracer



Spring Cloud Sleuth



ZIPKIN

这些公司的产品/服务是开源的

不是开源的

来源:<https://landscape.cncf.io>

© 2021 THE NEW STACK

图 1.3:目前包含在 CNCF 格局中的追踪工具。

交易通过您的系统。您可以使用此信息来查看应用程序的运行状况并调试有问题的微服务

或活动。

技术101

Tracing 是一种强大的调试工具,可让您进行故障排除和微调

分布式应用程序的行为。这种力量确实是有代价的。

需要修改应用程序代码以发出跟踪数据,并且任何跨度都需要由应用程序数据路径中的基础设施组件传播,特

别是服务网格及其代理。 Jaeger 和 Open Tracing 都是 CNCF

这个空间的项目。

- 流行语： Span、Tracing
- 热门项目： Jaeger、OpenTracing

混沌工程

这是什么

混沌工程是指故意将故障引入系统以创建更具弹性的应用程序和工程团队的做法。混沌工程工具将提供一种可控的方式来引入故障并针对应用程序的特定实例运行特定的实验。

它解决的问题

复杂的系统会失败。他们失败的原因有很多,而后果是

分布式系统通常很难理解。混沌工程被那些接受失败会发生的组织所接受,而不是试图防止失败,而是练习从失败中恢复。这被称为优化[均值](#)

[恢复时间](#),或平均修复时间。

旁注:维护应用程序高可用性的传统方法称为优化[平均故障间隔时间](#),或 MTBF。您可以在组织中观察到这种做法,这些组织使用变更审查委员会和长期变更冻结等方式通过限制变更来保持应用程序环境的稳定。 《[加速](#)》一书的作者建议高性能 IT 组织通过优化 MTTR 来实现高可用性。

它如何帮助

在云原生世界中,应用程序必须动态调整以适应故障,这是一个相对

新概念。这意味着,当出现故障时,系统不会完全崩溃,而是会优雅地降级或恢复。混沌工程工具使您能够在生产环境中对软件系统进行试验,以确保它们正常降级

或者在真正发生故障时恢复。

简而言之,你试验一个系统是因为你想确信它

可以承受动荡和意外的情况。您不必等待事情发生并了解您的申请情况如何,而是通过它

在受控条件下进行胁迫,以找出弱点并在之前解决它们
机会揭开他们。

技术101

混沌工程工具和实践对于实现高可用性至关重要

你的应用程序。分布式系统通常过于复杂而无法被完全理解

由任何一位工程师,没有任何变更过程可以完全预先确定影响

混沌工程

第 01 章:了解云原生可观察性堆栈



这些公司的产品/服务是开源的 不是开源的

来源:https://landscape.cncf.io

© 2021 THE NEW STACK

图 1.4:目前包含在 CNCF 版图中的混沌工程工具。

环境的变化。通过引入深思熟虑的混沌工程实践,团队能够练习并自动从故障中恢复。混沌网络和 Litmus Chaos 是这个领域的 CNCF 工具,但有许多开源和专有选项可用。

- 流行语:混沌工程
- 热门项目: Chaos Mesh、Litmus Chaos

结论

正如我们所看到的,可观察性和分析层都是关于了解系统的健康状况并确保它即使在恶劣的条件下也能保持运行。日志记录工具捕获应用程序发出的事件消息,监视日志和指标,并跟踪单个请求的路径。结合使用时,这些工具可以理想地提供 360 度的系统视图,了解系统中正在发生的事情。混沌工程有点不同。它提供了一种安全的方法来验证系统是否可以承受意外事件,基本上确保它保持不变

健康。

[杰森·摩根](#)是 Buoyant Linkerd 的技术布道者。他负责帮助培训工程师使用最初的服务网格 Linkerd。

[凯瑟琳帕吉尼](#)是 Linkerd 的创建者 Buoyant 的营销主管。从营销领导者转变为云原生传播者,她热衷于就新堆栈及其提供的关键灵活性对业务领导者进行教育。

第02章

可观察性超越 指标、日志和跟踪

[来源文章](#) Honeycomb 的 Danyel Fisher 于 2020 年 8 月 6 日出版。

编者注:可观察性的三大支柱 指标、跟踪和日志 本身会生成数据,但如果不进行分析,这些数据的价值有限。在这里,作为 Honeycomb 首席设计研究员的作者向我们介绍了使用每个支柱提供的视角来全面了解系统中正在发生的事情意味着什么。

系统。为了解释它的含义以及如何实现它，
可观察性 ~~有些人认为它是生产软件的相对较新的概念~~ [三大支柱](#) 将其拆分为三个部分
[可观察性](#)”将其拆分为指标、跟踪和日志。但我认为这错过了一个
观察系统意味着什么的临界点。

可观察性衡量您对系统内部状态的理解程度
基于信号和外部可见输出。该术语描述了一种内聚能力。可观察性的目标是帮助你清楚地看到你整个的状态
系统。

当您将这三个“支柱”统一为一个有凝聚力的方法时,一种新的能力
还出现了几种了解系统完整状态的新方法。支柱本身并不存在这种新的理解;当您采用统一方法时,它是各个部分的总和。简单地通过其各个组成部分来定义可观察性会错过大局。

考虑自己的能力

描述该系统如何组合在一起的一个很好的类比是使用滤色镜。每个镜头都会去除一些信息波长,以换取强调其他信息。当您需要专注于特定的红色阴影时,使用红色镜片会有所帮助,这是您应该使用的工具。但是要看到全局,您需要能够实时看到所有鲜艳的颜色。

例如,假设您注意到生产服务似乎运行不正常。
让我们看看不同的镜头如何看待这个问题。

监控和指标

已触发警报,通知您传入连接数高于指定阈值。这是否意味着用户在生产中的体验很差?这还不清楚。但是第一个镜头,监控工具 (即指标)可以告诉我们一些我们可能需要知道的非常重要的事情。

指标镜头将系统状态反映为数字的时间序列,这些数字是
主要用作量规。在任何给定时间,每项措施的表现是否超过或低于我们关心的阈值?这当然是重要的信息。指标
当我们可以缩减您的时间序列并使用
每台机器或每个端点都不同。

“简单地通过其各个组成部分来定义可观察性会错过大局。”

丹耶尔·费舍尔,蜂巢

当指标在预定义的时间长度内超过指定阈值时,可以使用诸如 Prometheus 之类的指标监控工具来触发警报。大多数指标工具都可以让您汇总少量标签的性能。对于您触发的警报,这可能会帮助您简单地了解哪个服务遇到问题或问题发生在哪些机器上。

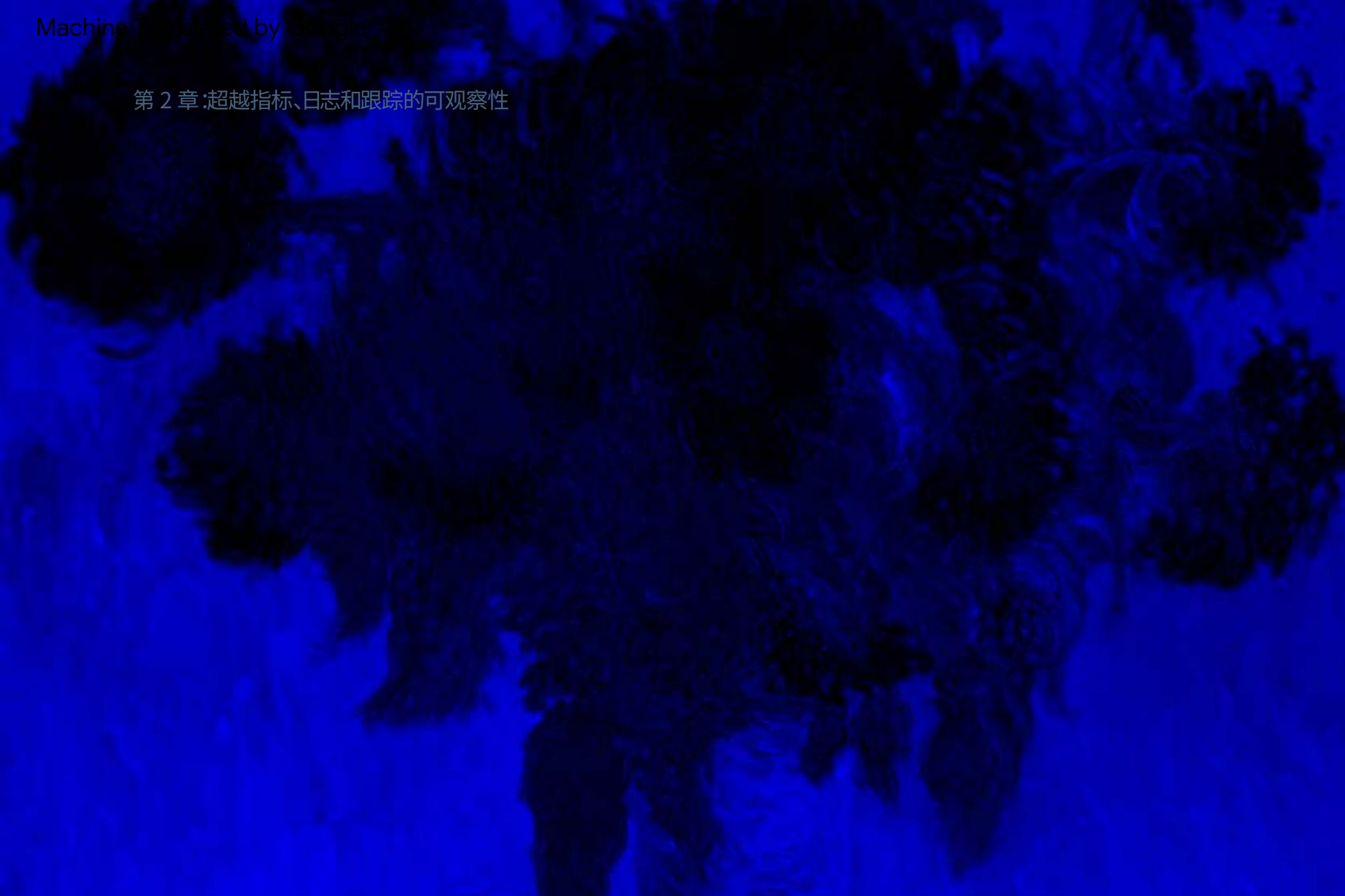


图 2.1:蓝色镜片的视图。

但是,要更精确地缩小范围,您需要以[高基数](#)跟踪数百或数千个不同的值。这可以让你创建比较足够时间序列的视图,您可以推断出更精细的见解。您可以确定哪些特定服务端点遇到错误,或者确定有关哪些用户正在积极使用它们的信息。

对于类比,指标可以表示为数据上带有蓝色透镜的放大镜。在这种情况下,您对生产中发生的事情的看法看起来有点像上图。

追踪

我们在生产中仍然有一个有问题的服务。通过高基数指标,我们能够确定失败的原因以及受影响的用户,但原因尚不清楚。下一个镜头,追踪,可以告诉我们一些非常重要的事情

我们可能需要知道才能找出答案。[痕迹](#)帮助您查看各个系统调用和

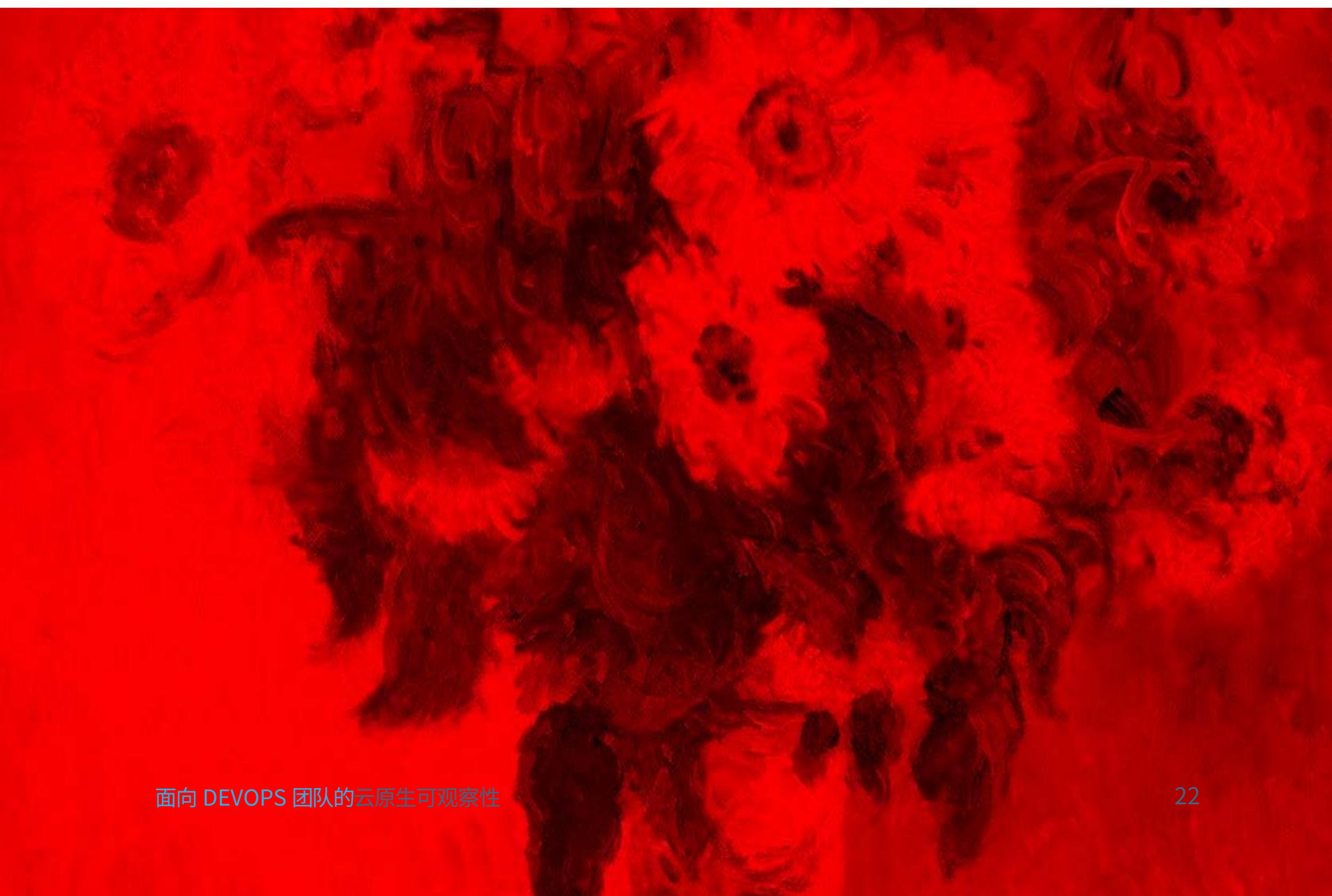
了解为返回结果所采取的各个基本步骤。

跟踪工具,例如 Jaeger,可以很好地了解底层组件正在发生的事情。通过跟踪,您可以查看哪个组件执行时间最长或最短,或者特定的底层函数是否导致了错误。跟踪可以成为更深入调查指标系统触发的警报的有用方法。

例如,您可能能够使用从指标仪表板中找到的信息来查找与遇到问题的端点命中相同端点的跟踪。跟踪可能会直观地向我们展示发出的有问题的请求中最慢的部分,或单个跟踪跨度,是对特定数据库的系统调用。这个跨度比平时花费更多的时间。

但是,我们已经在连续性上遇到了障碍。从度量工具到跟踪工具的过程是坎坷的。它们是两种不同类型的收集工具

图 2.2:红色镜片的视图。



不同类型的数据。要了解跨度需要更长的时间

与往常相比,您需要以某种方式关联两个工具之间的信息。

该过程既耗时又容易出错,通常无法提供调试生产问题所需的快速准确度。您需要在两者之间关联的关键数据甚至可能在两个系统中都不可用。

例如,您的指标可能表明某些用户在特定页面上的页面加载时间正在飙升。但是,除非我们的指标系统和跟踪系统共享相同的基础数据,否则我们可能无法找到反映加载时间特定变化的跟踪。我们可能不得不搜索一个完全不同的系统来找到可以说明我们正在努力寻找的问题的样本踪迹。

在我们的类比中,追踪可以表示为带有红色镜片的放大镜。

从这个镜头来看,画面看起来很不一样,但有足够的共同点

我们可以成功识别某些共同部分的属性,并且通常可以

在图像之间保持定向。在这个镜头下,有些部分比

其他人,而细节的某些方面完全消失了。

日志记录

我们的跟踪工作向我们展示了根本问题发生在哪里:在对某个数据库的一次特定调用中。但是为什么数据库调用变慢了呢?至

继续找到问题的根源,您将需要检查系统或软件级别发生的情况。第三个镜头,日志,提供了对原始系统信息的必要洞察,可以帮助我们弄清楚你发生了什么

数据库。

例如,在该数据库的日志中滚动,您可能会发现数据库发出的一些警告表明它当时过载或者您

可能会识别出特别慢的查询,或者您可能会发现事件队列

变得太长了。一旦您知道在哪里查看,日志有助于阐明问题。

但这种微观层面的观点是极其有限的。例如,如果你想知道



图 2.3:通过绿色镜片的视图。

这个特定问题发生的频率,你需要回到度量工具来检查那个特定数据库队列的历史,假设你有

指标。

与其他镜头一样,在工具之间切换的过程（在这种情况下,从跟踪到日志记录)需要
一组新的搜索、一组新的交互 并且，
当然,更多的时间。

打个比方,我们可以把原木想象成一个绿色的镜片。

将镜头放在一起

不幸的是,一些可观察性解决方案只是将这些观点混为一谈
镜头在一起。他们将其作为单独的功能来执行,这取决于观察者
确定差异。



图 2.4:每个镜头显示的内容的比较。

这不是一个糟糕的开始。正如我们所看到的,一些属性在一个文件中是完全不可见的看,但容易在别人身上看到。并排比较可以帮助缓解这些差距。对于这个例子,每张图片都更清楚地展示了不同的方面:蓝色图片最能显示花朵的轮廓;红色显示小花的细节;绿色似乎最能突出阴影和深度。

但这三个独立的镜头有其固有的局限性。可观察性不仅仅是一次看每一件的能力,这也是理解更广泛的能力
图片并查看这些部分如何组合以向您展示系统状态。

各部分的大和

你的系统的真相是,每个镜头所突出的方面并不是独立存在于真空中的。只有一个底层系统在运行,具有所有丰富的属性。如果我们将这些维度分开 如果我们从日志和跟踪中单独收集指标监控 那么我们可能会忽视这样一个事实,即这些数据反映了一个单一底层系统的状态。

一种统一的方法可以收集并保存丰富性和维度。查看

总的来说,我们需要平稳、准确、高效和连贯地处理数据。当这些镜头被统一到一个工具中时,我们可以做一些事情,比如快速发现一条轨迹中哪里包含在其他轨迹中重复出现的异常,包括找出位置和频率。

可观察性工具不仅需要这些基本镜头,还需要维护

一组遥测数据和存储,保留足够丰富的上下文,我们可以同时从指标、跟踪和日志的角度查看系统事件。

监控、跟踪和日志不应该是不同的数据集。相反,他们应该

是同一张连贯图片的不同视图。这些镜头不适合观看单独的图片。每一个都只是将某些方面带入更清晰的焦点。可观察性工具应该使度量线图上的任何点或热图连接到相应的轨迹;任何跟踪跨度都应该能够查询日志数据的视图。

可观察性的力量不仅仅来自于不必切换上下文。拥有单一数据存储还解锁了执行视觉切片等操作的能力在您的数据中。换句话说,您可以查看两组事件的不同之处彼此跨越所有不同的维度(即领域)。就是这样的分析孤立的度量系统根本无法显示,因为它们不存储个别事件或者对于日志系统来说会非常耗尽。

手动连接这些镜头意味着需要自己进行关联以发现不明显的问题。对于我们的类比,这就像看到一个区域

图 2.5:克劳德·莫奈, [“向日葵束”](#), 1881年



在绿色和红色镜头中看起来都是浅色的,直觉上常见的颜色实际上必须是黄色。一些艺术家可能会推断出,系统可以为你做数学运算,或者你可以在图像之间来回翻转,盯着比特对比。

可观察性让您看到您的作品美丽而完整的画面软件系统。您不需要技能和经验即可将这些镜头组合在一起你的头脑去判断哪种黄色阴影可能是个问题。你只需要不小于整体之和的部分。

[丹尼尔费舍尔](#)是 Honeycomb 的首席设计研究员。他的工作以数据分析和信息可视化为中心。他专注于用户如何更好地理解他们的数据。他为数据分析师、最终用户和那些恰好有大量信息的人提供支持。他的研究视角从人机交互的背景出发,将前沿的数据管理技术与新的可视化算法相结合。

第03章

可观察性如何实现 开发运维

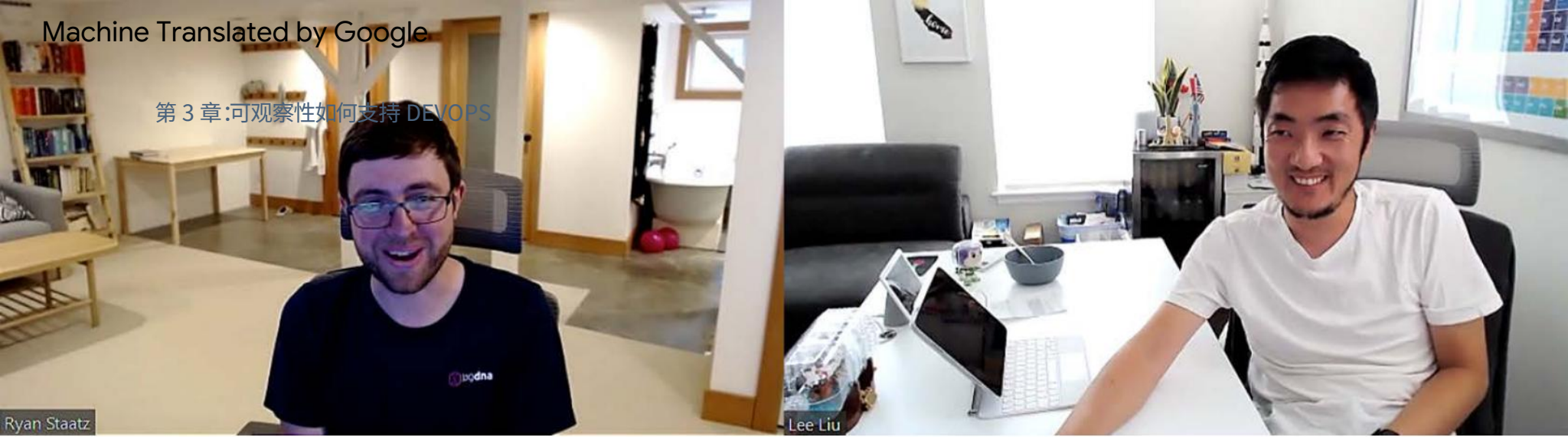
[来源文章](#) 2021 年 7 月 15 日,由 The New Stack 的 Joab Jackson 出版。

编者按:如果团队中的每个人都可以充分利用可观察性,这在实践中意味着什么?在这个问答环节中, Mezmo 的两位专家 公司的联合创始人 Lee Lui 和系统架构师 Ryan Staatz 与 The New Stack 的主编 Joab Jackson 讨论了正确的工具和组织文化如何帮助 DevOps团队不仅可以更高效地交付,而且可以更好地了解彼此的工作和挑战。

在 Mezmo 的人们,DevOps 就是同理心,超越 “烫手山芋”心态,因为这不是你的问题而不必担心,而是让全体工作人员一起努力解决问题 共同挑战。 Mezmo 自己的根源在于解决传统日志管理软件无法解决的问题:摄取非结构化日志并对其进行结构化 这样开发人员或管理员就不必手动操作了。

该公司对自己的产品采用了这种整体方法,构建了一个平台,为开发人员和站点可靠性工程师 (SRE) 提供访问权限 到他们构建和维护软件所需的信息。

我们采访了长期的 Mezmo 系统架构师 Ryan Staatz 和公司 联合创始人 Lee Liu 关于公司如何拥抱 DevOps 最佳实践 它的大小显着增加,以及可观察性的重要性 DevOps 以及如何为多种类型的用户制作最好的可观察性工具。



为了简洁明了,对采访进行了编辑。

你对 DevOps 的定义是什么?

Ryan Staats: DevOps 是一个模糊的术语。它介于开发人员和操作。如果你查看维基百科,它会说这是一套非常具体的方法和实践,这是正式的定义。但就像大多数事情一样,一旦行业掌握了它,它就会有自己的定义。

而且,至少对我而言,DevOps 正在弄清楚如何为应用程序的生命周期构建具有凝聚力的生命周期,从代码开发到测试 [持续集成/持续交付],再到将其推向生产,并获得反馈,整个循环。我觉得很多循环都有差距,这取决于您的组织处于哪个成熟阶段。

当你真的很小,而且你只有几个人时,拥有一个或一个人并不难两个人就拥有整个生命周期。沟通真的很快。随着你的成长,你有更多的团队和更多的人可以让你跑得更快。但是你有这些差距现在开始出现在您的生命周期中,应用程序可能在开发人员将其移交之后,或者可能在发布之间,或者您可能没有发布团队。

DevOps 是您拥有这些差距的想法,您希望整个生命周期都能正常工作。正是这种想法,虽然开发和运营在很大程度上被视为做不同事情的独立实体,但它们具有一些相同的目标。我们见过

许多安全和运营责任在生命周期中左移。现在,我们看到传统上由开发人员和产品负责的职责,例如

调整业务目标,右移。最终,我们都必须保持同步交付很棒的产品。

在许多组织中,Dev 和 Ops 之间存在哪些差距?

Staatz:我听到的关于 Dev 和 Ops 团队的一个古老的抱怨是,这个烫手山芋不断被抛到一边。这是你想要避免的不健康的互动。在 DevOps 中,您有一个正在努力实现共同目标。

你会在某些地方看到的态度是,“我怎样才能让这不是我的问题?”

我认为,这只是在一家可能没有预留所有资源或可能没有所有正确流程的公司的不幸部分。和不

公司是完美的。

其中一些只是与这样一个事实有关,即随着公司的发展,您拥有更严格定义的角色。当你到达一些共享的区域时,基本上很难说,“好吧,谁拥有它?”嗯,答案是没有。

没有神奇的修复,你知道吗?通常更多的是鼓励人们思考其他团队在做什么,想想他们为什么这样做。

了解他们的目标,也许看看他们的工具。所以,需要一点努力,诚实地。

Lee Liu:当 Dev 和 Ops 团队不一致时,我们看到的挑战之一是风险逆境。在航空业,你可以看到这个问题在过去几十年里不断恶化。你看看驾驶舱,你会看到一些非常古老的东西,比如那台看起来像是属于我爸爸办公桌的电脑,对吧?

飞机的其他一切都非常现代,比如椅背电视等。

但实际的技术似乎真的很老了。

我认为这可能是风险逆境带来的结果之一,在这种情况下,Ops 团队被烧毁了太多次。所以现在有多年来发展起来的创伤后应激障碍。运维团队就像是,“只是不要做太多大事

变化,然后事情会更稳定。”

航空业面临的是飞机失事,人员死亡。所以,他们不是

急于更改软件。软件可能从 1990 年就开始使用,但不需要更改。它会驾驶飞机,它会降落飞机。仅此而

已

需要做的。它不需要现代化。

最终发生的是现在 [you re] 使用 20 年前的技术,我认为至少对于软件初创公司来说这不是正确的方法。

当然,部署新代码总是存在风险。但这取决于我们使用这些工具,尤其是

可观察性工具,以帮助促进 [so] 当问题发生时,我们知道如何

修复它们。

可观察性可以为 DevOps 带来什么?

Staatz:整个生命周期的一部分就是弄清楚哪里出了问题,对吧?相反,在这个生命周期中,我可以改进什么东

西,所以东西不

分解?

归根结底,可观察性就是获取有关事物的 [the] 详细信息

继续。

有很多不同的工具可以帮助做不同的事情,而且经常在

该生命周期的不同部分。它可以特定于您的某些部分

基础设施。它可能来自您的应用程序代码中正在某处发送数据的内容。你可以拥有可以记录的东西。它可能是

指标。它可能会密切关注您的环境状态。

于是,就出现了这种有趣的交集,就像我们谈到的

早些时候,不同的团队在某些空间重叠,并且不清楚

谁拥有这些空间中的东西。如果你有可以帮助两者的可观察性工具

这些团队实现了他们的目标,希望他们的目标在某种程度上是一致的,那么这将是一个

巨大的帮助。

“你能做的最好的事情之一就是对其他正在开发代码或维护你的代码的人做一个好心人。”

Ryan Staatz , [Mezmo](#)

我们开发了一项名为[Kubernetes Enrichment](#)的产品功能。它显示有关与该应用程序中该日志行相关的 Kubernetes 集群状态的信息,让您了解此时环境中是否出现问题。看到这些信息对两个人真的很有帮助

不同的团队,这种开发和基础架构人员的重叠。所以它有助于澄清一个有点模糊的领域:那时可能会发生什么及时。

如果没有 Kubernetes Enrichment,有人会怎么做?他们如何获得 Kubernetes 日志?他们甚至可以访问这些信息吗?

Staatz:有些复杂情况因我认为是的差距而加剧前面讲过,这里有很多环境,尤其是生产环境,可能不允许许多开发人员访问,因此获取有关这些 pod 的信息可能涉及登录另一个工具,假设您已经设置了该工具。在我们的例子中,这类似于 Sysdig。但它的种类有限

对开发人员有用的指标。

你通常在没有任何工具的情况下做的是,你基本上只是说,“嘿,SRE 朋友,你能帮我看看环境吗?”到那时,想弄清楚发生了什么可能为时已晚。所以需要执行很多步骤

跨团队,这通常很困难。

如果您有其他工具,开发人员是否知道如何使用它,就会涉及到整个问题。如果它不被开发人员普遍使用,这会返回

对于同样的问题,其中有多少是个人动机 “我真的需要有点深入挖掘并做更多” 这些组织流程有多少,其中有多少只是选择正确的工具?

开发人员的可观察性需求与系统管理员或站点可靠性工程师 (SRE) 的需求有何不同?

刘:我会说,至少从日志记录的角度来看,我们尝试制作一个工具,双方都可以使用。它从来都不是完美的。所以,你只能迎合这么一个观众与另一个观众,因为他们从根本上寻找不同的东西,并且他们理解不同的背景。

开发人员并不真正关心其他人的应用程序。他们关心那些他们写过的和他们需要调试的。他们对他们的应用程序的了解比他们对系统上运行的其他东西以及困扰他们的应用程序运行的系统的了解要深入得多。

系统管理员和 SRE 正在寻求确保整个系统稳定。它更像是一个宏观与微观的层面。一切都需要表现凝聚力,以使其发挥作用。

我们可以构建的工具可以尝试拥有不同的东西,以便每个人都有他们正在寻找的数据。但是拥有这样的工具绝对是具有挑战性的为双方工作。



概括

开发、运维和安全团队对可观察性有不同的目标。集中式日志管理等工具可以帮助他们调整这些目标,并为每个团队的工作如何影响其他团队提供背景信息。

重点产品

Mezmo 是一个 DevOps 日志管理解决方案,让团队能够控制他们的数据,并允许他们从日志中获得有价值的见解。

主要合作伙伴

IBM Cloud、Heroku、Render、寻呼机、NGINX

GitHub

<https://github.com/logdna>

Kubernetes 在可观察性方面有哪些挑战？

Staatz：Kubernetes 做了很多很酷的事情,很多内置工具 [和] CLI 都很棒,甚至还有一个你可以使用的仪表板。话虽如此,跟踪运行在数百个节点上的微服务的指标和日志可能真的很难,就像,即使你是 Kubernetes 专家,并且你知道所有标签选择器和日志。

Kubernetes 相当新,它会随着时间的推移而成熟,但很难跟踪

一切都下来了。并不是每个人都想经常使用命令行界面工具,即使它们是技术性的。Kubectrl 很棒,但在某些时候,您想要使用对用户更友好的工具。

Lee,促使您和 Chris Nguyen 将 Mezmo 专注于日志记录的技术问题是什么?其他日志记录工具让您感到沮丧的是什么?

Liu:我们在 Elasticsearch 上构建了整个后端。我不喜欢 Elasticsearch,虽然它实际上不是 Elasticsearch 本身,但它是摄取部分。Elastic 使用 Logstash 来摄取日志。Logstash 的问题是您需要告诉它您正在摄取什么类型的日志,以便它可以进行正则表达式过滤

并丰富数据和所有这些东西。

如果你有一个应用程序,那并不难,对吧?但如果你同时使用 Mongo 或 Redis,Logstash 变得有点难以维护,因为你有不同的日志源。因此,我们编写了自己的摄取器,它基本上会获取日志传入的数据,并自动检测它是什么类型的文件并自动解析它。

不管我发送什么样的日志,它都会处理一些一般的事情,而一些事情是非常具体的。如果它检测到网络日志,它会为网络日志做这些事情。如果它看起来像一个 MongoDB 日志,它会做这些与 MongoDB 相关的事情。

我们就 Elasticsearch 谈过的其他一些公司,那是他们的

斗争,缺乏自动检测。他们必须手动完成所有这些事情。和他们不想手动完成。

当公司没有专门的 DevOps 人员或 DevOps 团队时,为什么跨团队同理心如此重要?日志如何帮助解决这个问题?

Staatz:我现在面临的一些挑战与其说是解决技术问题,不如说是必须有组织地解决这个技术问题。这是一组不同的问题,因为我不能简单地进入并修复产品。不是这样的。所以,很多人都在弄清楚是哪个团队在做这件事。

当你开始做这件事时,你会习惯性地问,“嘿,你们在做什么?”当您尝试完成工作时,您会不经意地对这些不同的团队产生同理心,因为,您知道,也许

在这一点上,工作不仅仅包括一个团队。

生产是一个大而可怕的地方,你只是把你的应用程序运送到那里,对吧?你就像,“哦,我的应用程序在那里。我希望它有效。魔术,对吧?嗯,这不是魔法。

如果有问题,另一端有人必须清理你的烂摊子或让你清理你的烂摊子。我知道,这是一种消极的思考方式。但你能做的最好的事情之一就是对其他正在开发代码或维护你的代码的人做一个好心人。

当你作为一名开发人员变得更有经验时,你开始思考基础设施每天处理的一些相同的事情。因此,拥有涵盖这两件事的日志和可观察性,并且可以由两者协作,可以走很长的路。任何可以帮助建立这些桥梁并可以建立信任并使人们保持一致的东西都很棒。

[乔布·杰克逊](#)是 The New Stack 的主编。他在基础设施 IT 新闻领域拥有超过 25 年的经验,包括在 IDG 和 Government Computer News 任职。他于 1995 年开始担任美国报纸巴尔的摩城市报的第一位互联网专栏作家,后来为美国国防部承包商工作,帮助解释最初为美国导弹防御系统开发的商业化技术的优点。

Kubernetes 可观察性 实践

第四章

什么以及如何登录 Kubernetes

[来源文章](#)由 Mezmo 的弗朗西斯·埃斯佩尼多 (Francis Espenido) 于 2020 年 8 月 26 日出版。

编者按： Kubernetes 日志如何提高应用程序的可观察性?在本章中,我们将深入了解 Kubernetes 在日志记录功能方面提供的功能及其局限性。如果您使用的是 Kubernetes,本实用指南可以帮助您确定在日志记录解决方案中需要什么。

监控是您开始 Kubernetes 之旅时需要解决的首要挑战之一。是否有单
无论您的 Kubernetes 环境是什么样的,日志记录和

在 PC 上本地运行、托管开发环境的小节点集合或托管生产应用程序的大型多主
机集群,访问监控数据和日志以进行故障排除至关重要

问题和优化性能。

鉴于 Kubernetes 由许多不同的部分组成,知道在哪里查找这些数据以及如何解释它可能会很棘手。事实上,Kubernetes 日志记录和监控需要使用多种数据源和多种工具,因为 Kubernetes 以多种方式生成日志。

本章概述了 Kubernetes 的日志记录,包括 Kubernetes 中可用的日志数据类型以及如何访问这些数据。大多数类型的 Kubernetes 日志数据都有多种访问方法。出色地

解释如何评估您的日志记录需求并设计适合的日志记录策略他们。

“鉴于 Kubernetes 由许多不同的部分组成，知道在哪里查找数据以及如何解释数据可能会很棘手。”

弗朗西斯·埃斯佩尼多, [梅兹莫](#)

以下大部分信息适用于任何类型的 Kubernetes 环境。

但是,为了讨论的基础,我们将参考[IBM Cloud Kubernetes Service](#) (IKS) 在相关的地方解释我们讨论的工具和实践如何适用于真实世界的生产 Kubernetes 服务。

什么是 Kubernetes?

如果您正在阅读本章,您可能已经知道 Kubernetes 是什么。

然而,有必要简要说明它的作用,因为了解 Kubernetes 能做什么和不能做什么是将日志策略扩展到支持它。

Kubernetes 提供了几种关键类型的功能：

- 应用托管： Kubernetes 的首要特性是托管应用程序。通常,这些应用程序托管在容器中,尽管它可以运行其他类型的工作负载,例如虚拟机，Kubernetes。
- 负载均衡： Kubernetes 自动在不同的系统之间分配流量应用程序实例以优化性能和可用性。
- 存储管理： Kubernetes 可以管理对存储池的访问应用程序用于存储有状态数据。
- 自愈:当出现问题时,例如应用程序故障，

Kubernetes 会尝试自动修复它。然而,它并不总是成功,
这就是 Kubernetes 日志记录如此重要的原因之一。

Kubernetes 也做其他事情,但这些是它的核心功能领域
提供。

Kubernetes 和日志记录:很复杂

您会注意到日志记录和监控不在核心 Kubernetes 功能列表中。这并不是因为 Kubernetes 不提供
任何类型的日志记录和监控功能。可以,但是很复杂。

一方面,Kubernetes 通过 kubectl 提供了一些非常基本的功能来检查集群中对象的状态,我们将在下面讨
论。它还为某些类型的数据创建日志,并以可通过第三方日志工具收集的方式公开其他类型的数据。

另一方面,Kubernetes 没有提供成熟的本地日志记录解决方案。

与 Amazon Web Services 不同,Amazon Web Services 具有 CloudWatch 形式的内置日志记录解决方
案,或 OpenStack 具有自己的综合日志记录解决方案,stock Kubernetes 没有完整的本机日志记录服务,甚
至没有首选的第三方-方记录方法。相反,它希望您使用外部工具来收集和解释日志数据。

也就是说,某些 Kubernetes 发行版确实带有基于第三方工具的内置日志扩展,或者至少是他们首选的日志记
录方法

支持。例如,正如我们将在下面看到的,IBM Cloud Kubernetes Service (IKS) 和
IBM Log Analysis 与 Mezmo 集成以收集 Kubernetes 日志数据并使用 Mezmo 启用实时分析和日志管
理。

在大多数情况下,可以使用替代日志记录方法,即使在具有首选或本机集成日志记录解决方案的
Kubernetes 发行版上也是如此。

但是,供应商支持的方法通常更易于实施。

在 Kubernetes 中登录什么

无论您选择哪种日志记录选项,您都可以选择多种日志数据类型可以在 Kubernetes 中收集。

应用程序日志

首先是在 Kubernetes 上运行的应用程序的日志。这些日志中存储的数据包含您的应用程序输出的信息

他们跑。通常,此数据被写入到内部的 stdout (标准输出)

应用程序运行的容器。

我们将在“查看应用程序日志”部分了解如何访问此数据如下。

Kubernetes 集群日志

构成 Kubernetes 本身的几个组件会生成自己的日志：

- Kube-apiserver。
- Kube 调度程序。
- 等等。
- Kube 代理。
- Kubelet。

这些日志通常存储在服务器上的 /var/log 目录下的文件中

服务运行。对于大多数服务,该服务器是 Kubernetes 主节点。

然而,Kubelet 在工作节点上运行。

如果您遇到集群级别的问题,而不是仅影响某个容器或 pod 的问题,这些日志是寻找洞察力的好地方。例如,如果您的应用程序在访问配置数据时遇到问题,您可以查看 Etcd 日志以查看问题是否出在 Etcd 上。如果工作节点未能按预期上线,则其 Kubelet 日志可以提供见解。

Kubernetes 事件

Kubernetes 会跟踪它所谓的“事件”,这些事件可以是集群中对象状态的正常变化,例如正在创建或启动的容器,或者资源耗尽等错误。

事件仅提供有限的上下文和可见性。他们告诉你发生了什么事,但并没有告诉你为什么会发生。它们仍然是一种有用的获取方式有关集群中各种对象状态的快速信息。

Kubernetes 审计日志

Kubernetes 可以配置为将请求记录到 Kube-apiserver。这些包括人类发出的请求,例如请求正在运行的 pod 的列表,以及 Kubernetes 资源,例如请求访问存储的容器。

审核日志记录发出请求的人或内容、请求的目的和结果。如果您需要对与 API 请求相关的问题进行故障排除,审计日志可提供大量可见性。它们对于通过查找异常请求来检测异常行为也很有用,例如用户重复尝试访问集群中不同资源的失败尝试,这可能表明正在寻找不正确保护资源的人企图滥用。（它也可能反映出您的身份验证配置或证书存在问题。）

如何访问 Kubernetes 日志数据

可以通过不同的方式访问上述各种类型的日志数据。

查看应用程序日志

与应用程序日志数据交互的主要方式有两种。首先是运行一个命令之类的

```
kubectl 记录 pod 名称
```

其中“pod-name”是托管应用程序的 pod 的名称,该应用程序记录了您的日志

想访问。

kubectl 方法对于快速查看日志数据很有用。假设您想持久存储日志并系统地分析它们。在这种情况下,您最好使用 Mezmo 等外部日志记录工具来收集和解释日志。最简单的方法是运行一个所谓的 sidecar 容器,它与应用程序一起运行,收集其日志并将它们提供给外部日志记录工具。

在 IKS 上,您可以设置一个 Mezmo 实例来为应用程序执行此功能
日志以及与 Kubernetes 本身关联的日志,从命令行或在 IKS Web 控制台中执行几个步骤。有关完整说明,请查看

[IBM 云文档。](#)

查看集群日志

有多种查看集群日志的方法。您可以简单地登录到托管您要查看的日志的服务器（如前所述,在大多数情况下是 Kubernetes 主节点服务器)并直接在less、cat中打开各个日志文件

最喜欢的文本编辑器 或者您喜欢的任何命令行工具。或者,您可以使用 journalctl 为您检索和显示给定类型的日志。

对用户最友好的解决方案还是使用外部日志记录工具,如 Mezmo。
如本章前面所述,IBM Cloud与 [Mezmo 的集成](#)可以轻松收集 Kubernetes 集群和应用程序日志并通过集中式界面进行分析,而不必担心通过命令行从每个节点收集单独日志的繁琐过程。

查看事件

您可以使用如下命令通过 kubectl 查看 Kubernetes 事件数据

```
kubectl 获取事件 -n 默认
```

其中 “-n”标志指定您要查看其事件的命名空间（默认

在上面的例子中) 。命令

```
kubectl 描述我的 pod
```

将向您显示特定 pod 的事件数据。

由于事件数据的上下文是有限的,您可能会发现记录所有事件并不是很有用。但是,您始终可以将 kubectl 的 CLI 输出重定向到日志文件中,然后使用日志分析工具对其进行分析。

查看审计日志

与其他类型的 Kubernetes 日志数据相比,您查看和管理审计日志的方式在很大程度上取决于您使用的 Kubernetes 发行版和日志收集器来收集这些日志。没有通用且直接的方法可以直接从 kubectl 收集审计日志。

在 IKS 上,审计事件被路由到一个 webhook URL。从那里,你可以收集

按照这些[说明](#)使用 IKS 的本地 Mezmo 集成记录数据。

如何构建 Kubernetes 日志记录解决方案

正如我们在上面看到的,Kubernetes 中有多种类型的日志数据可用,但也有多种访问方式。设计最适合您的 Kubernetes 日志记录策略和工具集需要权衡几个因素:

- 需要收集哪些类型的Kubernetes日志?某些类型的数据对你来说可能或多或少重要。例如,如果你运行简单对于不生成有意义的监控数据的应用程序,应用程序日志可能不如 Kubernetes 集群日志重要。
- 您的日志记录目标是什么?如果你只想快速查看日志文件,kubectl (或 journalctl,在某些情况下)可以完成这项工作。但是,如果您需要长期的日志管理,您将需要一个外部日志收集器。
- 您需要可视化吗?当您访问 Kubernetes 日志文件时,您是否

只是寻找特定的信息,比如 API 请求的来源?

或者您是否希望能够可视化数据以识别趋势或比较日志?

在后一种情况下,提供日志的外部工具或工具组合

需要收集和可视化。

- 是否需要聚合日志?您的目标是访问单个日志文件,还是想将多个日志聚合在一起并集中分析?你

需要外部工具来完成后者。

- 您需要将日志保留多长时间?大多数日志数据存储在 Kubernetes 会在一段时间后删除,这取决于日志类型和您的日志记录配置。因此,如果您想长期保留历史日志数据,则需要将其导出到外部日志记录平台。

结论

Kubernetes 中的日志记录有很多细微差别。尽管 Kubernetes 提供了一些基本的内置日志记录和监控功能,但它与成熟的日志记录解决方案相去甚远。要充分利用 Kubernetes 日志记录,您需要一个外部日志收集、分析和管理工作具。

[弗朗西斯·埃斯佩尼多](#)是 Mezmo 的高级技术合作伙伴项目经理,专注于 IBM Cloud 上可观察性产品的技术支持。他之前曾担任该公司的技术支持工程师,在那里他对现代日志记录实践和开发人员工作流程中的挑战有了深刻的理解。

第 05 章

Kubernetes策略_可观察性

[来源文章](#)由 Dynatrace 的 Peter Putz 于 2021 年 5 月 6 日发布。

编者注:分布式系统,尤其是那些应用程序部署到多个云和环境的系统,使可观察性变得更加复杂。以下是当您的基础架构在 Kubernetes 上运行时保持可观察性的一些最佳策略。

允许团队更快地构建新的数字服务和功能,以便他们能够适应快速变化的业务需求并继续推动客户的成功。然而,保持对这些云原生环境的可见性可能是一个真正的挑战。

Kubernetes 是流行的应用程序和创新平台,非常擅长自动化和管理容器化的工作负载和应用程序。然而,动态使它在环境中如此灵活和可移植的抽象层可以导致难以发现、排除故障和预防的新型错误。

将监控工具生成的复杂数据网络连接回业务结果更难。[研究](#)显示超过三分之二的首席

信息官 (CIO) 认为 Kubernetes 的兴起导致太多 IT 团队管理的移动部件,他们需要一种截然不同的 IT 和云运营管理方法。

以下是为什么在 Kubernetes 环境中获得可观察性如此困难的三个主要原因,以及组织可以克服这些挑战的一些方法。

1. Kubernetes 是高度动态的,所以 AIOps 和自动化是必需的

虽然像 Kubernetes 这样的分布式平台能够实现更快的创新和更好的可扩展性,它们也是高度动态和复杂的。集群、节点和 Pod 不断变化,因此没有时间手动配置和检测监控功能。IT 团队正忙于深入了解健康状况

他们的应用程序并跟上他们的 Kubernetes 的速度
环境正在发生变化,可用于启动新服务的时间
推动业务成功。

保持对这种动态环境的可见性的唯一方法是让团队
能够在新服务上线时自动发现服务,并且
现有的规模,并在飞行中对它们进行检测。利用连续
AIOps 辅助的自动化使平台和应用程序团队能够操作
具有数百万实时变化的大规模环境,并持续监控整个堆栈的系统退化和性能异常。

这不仅可以让团队全面了解他们的 Kubernetes 环境,还可以让他们通过确定哪些技术变更将产生最大的业务影响来更好地确定任务的优先级。有了这种洞察力,团队可以在影响用户体验的问题发生之前预防它们,并重新专注于持续优化

为企业及其客户提供最佳成果的服务。

2. Kubernetes 在许多地方运行,因此全栈方法是关键

除了跟踪不断变化的微服务和工作负载之外,当您考虑到组织经常跨平台部署 Kubernetes 时,保持可观察性的挑战变得更加复杂

多个环境。

这是因为 Kubernetes 可以在任何云上运行,使组织能够灵活地跨多个平台部署他们的微服务,并通过托管

服务,例如 Amazon Elastic Kubernetes Service (Amazon EKS)、Azure

Kubernetes Service (AKS) 和 Google Kubernetes Engine (GKE),以及它们自己的本地服务器。因此,许多组织使用不同的监控工具和云平台指标来管理他们的 Kubernetes 环境。

然而,手动收集和关联来自所有这些来源的可观察性数据,以获得更大的图景和完整的上下文,是非常耗时的。具有点监控解决方案的孤立团队进一步阻碍了这一点,并可能崩溃

跨团队协作。

一种有效的可观察性方法应该促进跨部门的合作

通过帮助打破团队之间的孤岛来组织。因此,它需要将所有 Kubernetes 指标、日志和跟踪统一到一个具有通用的平台中

数据模型。它还需要包括来自传统服务和

与 Kubernetes 部署一起运行的技术堆栈,以确保平台

和应用程序团队在整个环境中拥有统一的视图。这个

端到端的可观察性方法为这些团队提供了更大的背景

用于更成功地优化 Kubernetes 工作负载和应用程序。

3. 在用户上下文中查看数据至关重要

同样重要的是要记住,可观察性不仅与访问更多数据有关,还与组织如何使用这些数据来识别其技术堆栈中需要改进的领域有关。指标、日志和跟踪很重要,但

它们并不能说明全部情况,而且确实经常限制开发人员和应用程序

所有者只允许他们获得后端视角。

要了解 Kubernetes 性能对业务成果的影响,

组织需要能够将他们推入的代码之间的点连接起来

生产,后端的底层云平台和前端的用户体验。这意味着他们需要将 Kubernetes 监控数据与实时业务指标（例如用户体验洞察力和转化率)结合起来。

团队可以通过应用程序拓扑映射更轻松地获得这些见解

自动可视化所有关系和依赖关系的功能

Kubernetes 环境和更广泛的云技术堆栈,包括实时的用户体验数据。之间垂直映射依赖关系

集群、主机、Pod 和工作负载 以及数据中心之间的水平,

应用程序和服务 允许 IT 团队确定哪些问题对业务的总体影响最大。以这种方式将用户体验与后端性能相关联,

可为业务领导者和数字团队提供信息

他们需要就如何优化系统以及在哪里优化做出更好的决策

进一步投资于他们的数字基础设施,以改善服务并提供更好的服务

用户体验。

最终,解决出现的可观察性挑战的最有效方法

使用 Kubernetes 架构是为了拥抱自动化的好处,

人工智能 (AI) 。将可观察性与 AIOps 和自动化相结合,使团队能够将洞察力扩展到指标、日志和跟踪之外,

以及

合并其他有价值的数据,例如用户行为和业务关键绩效

指标 (KPI) 。通过以这种方式重新考虑他们的 Kubernetes 监控方法,组织可以消

除孤岛,这样他们的团队就可以花更少的时间进行故障排除,而将更多的时间用于优

化服务以推动更好的业务

结果。

[彼得·普茨](#)是 Dynatrace 的技术倡导者。他在领导由科学家和工程师组成的国际软件项目团队以及管理复杂、创新的 IT 解决方案的整个生命周期方面拥有超过 15 年的经验。在加入 Dynatrace 之前,他是 NASA 艾姆斯研究中心的资深科学家。

第六章

为什么开发人员应该学习Kubernetes

[来源文章](#)由 Mezmo 的 Steve Tidwell 于 2021 年 7 月 14 日发布。

编者注:我们在整本电子书中都提到了软件开发中的“左移”运动。这个想法是,开发人员在确保他们的应用程序投入生产后表现良好方面发挥更大的作用。但承担更大的责任意味着什么?在本章中,作者详细说明了开发人员作为 DevOps 团队成员更充分地参与其中特别需要学习的内容。

随着基于容器的软件开发工作流程的兴起,
近年来,Kubernetes 作为
驱动 选择部署这些容器。

自 2016 年起,[云原生计算基金会](#)进行了一项年度调查,以评估各种工程组织对容器和云原生技术的采用情况。根据[CNCF 2020 调查](#)受访者中,92% 的公司正在生产中运行容器,其中 83% 正在使用

Kubernetes 作为他们的编排工具。

与此同时,许多组织也在采用 DevOps 和站点可靠性工程 (SRE) 最佳实践来提高其应用程序的可靠性和交付新应用程序功能所需的时间。

由于这种采用,工程团队看到了合并运营和开发最佳实践的好处。运营团队越来越多

面向服务和软件,开发团队正在学习
他们部署应用程序的平台和环境。

近年来的大部分焦点都集中在软件开发上
运营最佳实践,从而显着提高了交付和可靠性。但有时,开发团队存在差距

并不总是具备使用其应用程序所需的操作技能
有效地在开发环境之外。

虽然构建和运输容器化应用程序可能无需
维护独特的开发环境,开发人员至关重要
了解容器在运行时是如何工作的,尤其是在使用协调器时
像 Kubernetes,在生产中更是如此。可能没有必要
开发人员拥有完整的操作技能,但他们需要学习足够的知识
关于 Kubernetes 和生产环境成为可靠的参与者
DevOps 团队。

培养技能

了解您的应用程序所在的平台至关重要。成功的工程组织努力确保开发和运营团队避免各自为政。相反,他们的目标是在软件开发生命周期的早期进行协作,以便编码、构建、测试和部署都很好

被参与该过程的所有团队所理解。

开发人员不需要是 Kubernetes 专家,但应该精通
影响其应用程序性能的技能。技能如
持续集成/持续交付、生产部署、监控和了解 CPU、内存以及集群和 Pod 健康状况是

应用难题。

了解有关组织使用的应用程序平台和工具的一些基础知识对于进行开发和运营大有帮助

更高效。拥有这些技能可以帮助开发人员快速、更有效地响应事件,从而在出现问题时解决问题而无需升级到另一个团队。

开发人员和运营团队需要了解的内容

开发人员和运维工程师都需要了解一些关于他们的同龄人知道什么:

- 他们需要了解各种服务的特点和特点
他们选择的云提供商与其他提供商的比较。这些知识应该适用于云是公共云、私有云还是混合云。
- 他们需要认识到为其提供资源的财务影响
应用程序并了解如何降低成本和消除浪费
从开发人员的角度应用。启动一个新的很简单
云中的环境和基础设施,这也意味着它很容易
忘记如果我们对资源管理不善,这些成本会以多快的速度增加。例如,考虑自动缩放策略及其对
这些政策设置不正确时的成本。
- 他们需要了解应用程序性能管理,尤其是工具
以及用于分析和改进应用程序性能的技术。
- 他们需要知道正确的事件响应技术来处理
事件发生时,并在适当的时候升级它们。中的一个
DevOps 的基本原则是接受并找到减轻故障的方法,因此在事件出现时高效且有效地处理它们至关重要。
- 他们需要在开发和开发的双方建立反馈循环
操作围栏,以便所有团队都知道他们的工具或工具中的任何缺陷
应用程序以及开发人员如何纠正它们。工具和环境的共享所有权是鼓励这种做法的绝佳方式。

那么开发人员应该具体了解 Kubernetes 的哪些方面呢？

- 他们组织的 CI/CD 系统是如何工作的,从概念到生产,从代码签入构建、测试和部署。
- Kubernetes [pod](#)以及它们与[容器的关系](#)。
- 应用程序如何与 Kubernetes[网络交互](#),包括[服务](#),[域名系统\(DNS\)](#)和[负载均衡](#)。
- 了解用于对其部署进行本地测试和对应用程序部署方式进行建模的常用工具的知识,例如 [minikube](#)、[kubectl](#),[舵,种类](#)和 Kubernetes[仪表盘](#)。
- [监控、日志记录和调试](#)事情发生时的集群和容器错误的。

当然,运营和开发团队可以做的还有很多了解他们如何一起工作,但这是一个很好的起点。

[史蒂夫蒂德威尔](#)在科技行业工作了 20 多年,从最终用户支持到扩展全球数据摄取和分析平台以处理网络上一些最大的流媒体事件的数据分析,无所不包。他曾为多家公司工作,帮助改善他们的运营并使他们的基础设施自动化。

目前,史蒂夫正计划在他办公室的一角用基于云的技术接管世界。

把它包起来

在我们最新的电子书中,我们提出了使可观察性成为一个完整堆栈的案例责任。我们概述了可用于帮助的云原生工具提高指标、跟踪和日志的可见性,还提供了一些实用的策略和策略来提高 Kubernetes 环境中的可观察性。

关键要点是可观察性在任何 DevOps 团队的重要组成部分工作 帮助提高应用程序性能和弹性,保持系统平稳运行,最重要的是,为您组织的用户和客户提供优质、可靠的服务。

云原生应用程序和架构仍然很复杂,随着新工具的出现和系统变得越来越分散,工程师和开发人员必须面对的挑战可能不会变得更容易。但是更大的可观察性,即开发人员和运维人员之间的共同责任,可以防止这种复杂性妨碍稳健性

服务。

希瑟·乔斯林

披露

以下公司是 The New Stack 的赞助商：

Accurics、any9ines、Armory、Aspen Mesh、Amazon Web Services (AWS)、Bridgecrew、CAST AI、Check Point、CircleCI、Circonus、Citrix、Cloud Foundry Foundation、CloudBees、云原生计算基金会 (CNCF)、Confluent、Cox、Cribl、DataStax、Dell Technologies、Dynatrace、Equinix、Fairwinds、GitLab、Harness、HashiCorp、Honeycomb、惠普企业 (HPE)、IBM Cloud、InfluxData、Infoblox、JFrog、Kasten、LaunchDarkly、Lightstep、MayaData、Mezmo、MinIO、Mirantis、MongoDB、New Relic、NGINX、Okta、Oracle、PagerDuty、Palo Alto Networks、Pure Storage、Puppet、Red Hat、Redis Labs、Rezilion、Rookout、Sentry、Shipa、Snowflake、Solo.io、Sonatype、StepZen、StorageOS、Storj、StormForge、Styra、Synopsys、Teleport、Tetrade、Linux Foundation、Thundra、Tigera、Torq、Tricentis、TriggerMesh、Vates、VMware、Weaveworks 和 WSO2。

The New Stack 是 Insight Partners 的全资子公司。TNS 所有者 Insight Partners 是以下公司的投资者,这些公司也是 The New Stack 的赞助商:Armory、Jfrog、Kasten、MayaData、Mirantis、StormForge、Tigera 和 Tricentis。

