Ansible 助力实现高效自动化

目录

红帽官方微博　红帽官方微信

免责声明

本指南以案例研究为基础，所有内容仅代表本指南作者的个人观点。文中信息均“按原样”提供，不作任何保证。这些信息在您阅读时可能已经过时。作者对与这些信息相关的任何内容概不负责。作者***认为***，这份指南在一般情况下会有所帮助。

版本历史记录

版本	日期	描述	作者和贡献者
1.0	2023 年 10 月 12 日	第一个公开版本	**作者：** Nicholas Wong、Gineesh Madapparambath、Michael Knyazev **贡献者：** Donna Benjamin、Jeremie Benazra、Denis Mikhalkin、Frederick Son、Valentine Schwartz

1. 简介

在这份基于案例研究的指南中，我们的目标是为最广泛的自动化专家群体和希望从自动化中真正受益的人员提供支持。本指南适用于 DevOps 工程师和架构师，以及产品负责人、IT 和运维经理。在我们的精心设计下，许多人在阅读本文档后的第二天，就可以开始使用所提及的资源和新知识。本指南的风格非常实用。对相应实践和重点建议的介绍与实际的自动化“示例项目”相交替，而且该项目从开始到最小可行产品（MVP）交付的整个过程贯穿各个章节。在以下根据 MIT 许可证发布的个人存储库中，可以找到一个可能的 MVP 实施方案：[1]https://github.com/mikhailknyazev/automation-samples

每个人都可以在这份指南中找到与他们的特定环境相关的独特内容。要说它有何独特之处，我们认为它展示了工程、产品和敏捷实践者如何以一种结构化的方式共同开展自动化项目，享受和谐的工作氛围，并为客户提供有价值的成果。

红帽提供的相关服务归类在最后的特殊部分：“红帽为客户提供哪些支持？”。

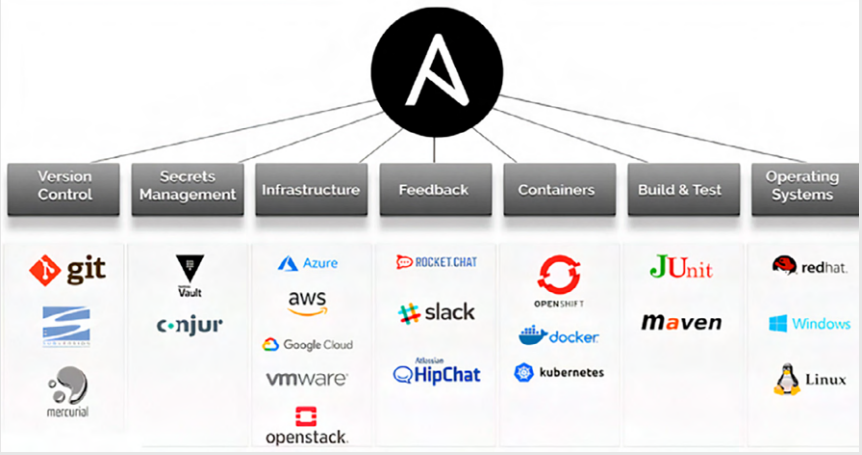
[1] 一个可能的 MVP 实施（在 MIT 许可证下发布的个人存储库）https://github.com/mikhailknyazev/automation-samples

2. Ansible，自动化的通用语言

复杂 IT 环境中的编排需求并不新鲜，而且您会发现许多生态系统已经有了自己的编排器。OpenStack 的 Heat、Amazon 的 CloudFormation、Azure Resource Manager、Jenkins 或 HashiCorp 的 Packer 以及 Terraform 等工具都是用于执行编排任务和实现“基础架构即代码”方法。

但是，客户将其编排工作限制在一个环境中的可能性有多大？这就是 Ansible® 的用武之地。凭借 Ansible 的模块库和易于扩展的特性，可以轻松编排不同环境中的不同“指挥者”，而且这一切都使用一种结构化的 Ansible 语法。关于该选择使用什么工具来实施新的脚本功能，Ansible 往往是替代“裸”Bash/PowerShell/Python 脚本编写的一个高效选择。

Ansible 可与客户现有的 SSH 和 WinRM 基础架构协同工作，因此无需打开额外的网络端口。Ansible 是企业 IT 工件的通用“粘合剂”。它可以与计算节点、存储设备、网络、负载平衡器、监控系统、网络服务和其他设备协同工作。例如，您可以向负载平衡池添加或从中删除服务器，并禁用正在更新的每台计算机的监控警报。



图片来自“使用 Ansible 实现自动化的行业用例”[2]

Ansible 在实际项目中的运用已屡见不鲜 – Ansible 驱动的基础架构置备、部署和测试，以及 Kubernetes 和红帽® OpenShift® 的自动化运维。您甚至可以将 Kubernetes 作为事件源添加到 Event-Driven Ansible（EDA）中，以便根据需要触发自动化。[3] 您可以在红帽 OpenShift 上部署 Ansible 自动化平台。[4]

以下是一些可能适用于客户企业的自动化想法示例：

- 通过确保一致的系统设置来减少偏移、中断和停机时间
- 通过自动执行运维任务来尽可能减少人为错误
- 降低跨多个领域的技能要求
- 为系统建立和维护所需的基线，以满足合规性要求
- 通过补丁管理降低安全漏洞和其他安全相关事件的风险

[2] 使用 Ansible 实现自动化的行业用例 https://www.linkedin.com/pulse/industry-use-cases-automation-using-ansible-anand-kumar/
[3] “结合使用 Kubernetes 与 Event-Driven Ansible” https://www.ansible.com/blog/kubernetes-meets-event-driven-ansible
[4] “全新参考架构：在红帽 OpenShift 上部署 Ansible 自动化平台 2” https://www.ansible.com/blog/new-reference-architecture-deploying-ansible-automation-platform-2-on-red-hat-openshift

谈及企业安全防护，特别需要指出的是它并非一个同质实体。通常情况下，它是多供应商解决方案的组合，由不同且往往孤立的团队运行。面对如此多的不同层级，自动化依然发挥着强大作用，经证明能有效帮助安全运维团队统筹兼顾而分工负责，并将多种安全防护技术整合在一起。

IT 团队成员经常发现自己日复一日地重复同样的任务。示例包括：

- 系统工程师负责构建服务器和虚拟机、安装软件包、修补旧系统、配置防火墙设备等。
- 每次发布新版本的编程语言或软件库时，开发人员都要费力地构建一个编码环境。他们还花费大量时间测试代码并等待测试结果。
- 数据库管理员花费宝贵的时间来置备磁盘空间和配置存储设备，在置备新数据库服务器时遇到延迟，或者面临网络或系统就绪问题。
- 运维团队面临着大量事件和警报的困扰，他们花费大量时间来过滤误报，从而导致工作效率降低。
- 安全防护团队努力纠正违规行为，确保系统符合相关的安全标准。

3. 规划基于自动化实现高效改进

我们的目标是实现事半功倍的效果。在开始实施自动化之前，不妨先停下来问几个问题：

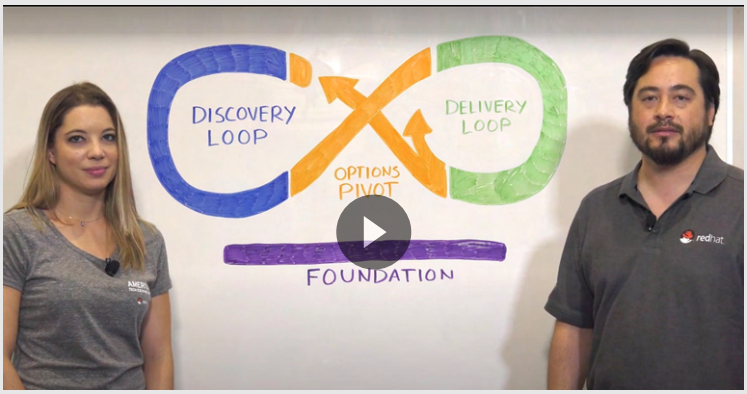
- 用例有多复杂?
- 能否减少人为错误?
- 能否缩短部署时间并加快任务执行速度?
- 我执行这项任务的频率如何?
- 自动执行这项任务可以为我节省多少时间?
- 投资回报率是多少？能否节省资金?

有这样一个常见误解，认为寻找用例并实施自动化完全是系统团队、平台团队或基础架构团队的责任。其实，当我们探索自己的工作环境和日常任务时，我们会发现有成千上万的任务可以使用 Ansible 实现自动化。比如，管理数据库服务器和实例的数据库团队、处理网络运维的网络团队，或者是希望更有效地部署应用更新的应用团队。在环境中实施自动化是一个协作之旅，需要来自不同团队的支持和指导。

有一些众所周知的开放实践可以促进采用结构化方法来实施自动化。本节所涵盖的实践和示例的逻辑顺序如下：

- “发现循环”
  - 动机映射
  - 全局事件风暴
  - 讲述领域故事
  - 基于指标的流程映射（MBPM）
  - 用户故事映射
  - 目标成果
- “方案转变”阶段
  - Kano 模型
  - 价值切片
  - MVP

下面的视频短片介绍了本指南所涉及的实践的大背景：



开放实践库的成果驱动型交付[5]

开放实践库可在此处查阅：[6] https://openpracticelibrary.com
您将在本指南中看到许多对其部分内容的引用，但不仅限于此。

下一节将介绍一种“结构化头脑风暴”方法，让不同团队中具有不同专业知识的成员相互交流知识，这就是“**事件风暴**”。这是了解自动化环境的全局所必需的。

我们还将介绍贯穿本指南的“**示例项目**”。

3.1. 让我们先探索一下我们所拥有的资源

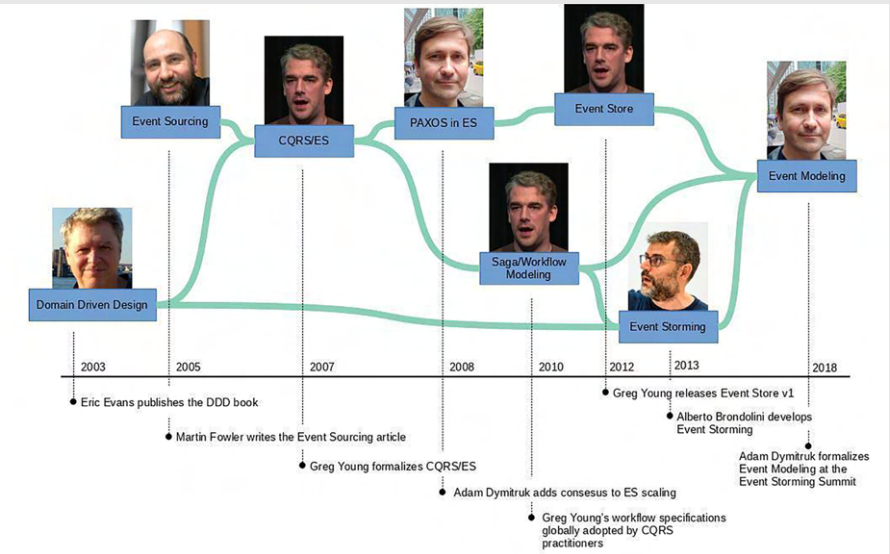
要展开一场由商务人士和技术专家共同参与的全面讨论，我们需要让他们之间的沟通尽可能顺畅。事实证明，**以事件为中心**有助于突破业务/技术的界限。与所讨论的领域相关的各种事件都可以被所有各方识别。与通常的头脑风暴一样，我们希望优先考虑推动前进而不是追求精确性。

3.1.1. 以事件为中心的常见实践

下一节将介绍“**事件风暴**”实践，其重点是发现问题领域。另一种包含以事件为中心的观点（而不是基于系统或时间的观点）的实践是“事件建模”。[7] 后者可基于“命令查询责任分离”和“事件溯源”（CQRS 和 ES 设计模式）的概念创建解决方案蓝图。换句话说，事件建模可以非常有效地构建某些类型的解决方案，但并不适合所有人。

以下是以事件为中心的实践演变摘要，右侧为“事件风暴”和“事件建模”：

[5] “开放实践库的成果驱动型交付” https://www.youtube.com/watch?v=N4mBIZg8MnQ
[6] “开放实践库” https://openpracticelibrary.com/
[7] “事件建模” https://openpracticelibrary.com/practice/event-modeling/



图片来自“关于事件建模”[8]

在本指南中，我们将重点介绍“事件风暴”实践。如果您想了解有关“事件建模”的更多信息，那么作者 Adam Dymitruk 的这篇“事件建模深度剖析”应该会有所帮助。[9] 您可能还会喜欢以下事件建模资源：

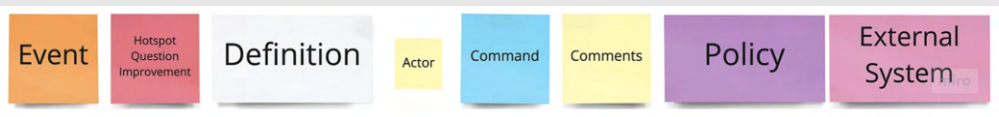
- 基于示例的精彩讲解[10]
- 在此处可以找到一些适用于 Miro 的资源[11]
- 如何从“全局事件风暴”中引导出“事件建模”过程[12]

3.1.1.1. 什么是“事件风暴”发现实践？

事件风暴研讨会有多种不同类型。“全局事件风暴”通常旨在发现业务领域和分享知识。“软件设计事件风暴”更侧重于系统设计。

“事件风暴”实践综合了来自 Gamestorming[13] 的促进小组学习实践和领域驱动设计（DDD）的原则。

知识以彩色便利贴的形式呈现：无论是使用 Miro[14] 等工具进行的在线会议中的虚拟便利贴，还是在一间拥有 8 到 10 米长墙壁（越大越好）的房间里进行现场记录。在下图中，您可以了解每张便利贴的颜色和用途。



[8] “关于事件建模” https://eventmodeling.org/about/
[9] “与 Vaughn Vernon 和 Adam Dymitruk 一起深入了解事件建模” https://www.youtube.com/watch?v=ufKgwjsD1l8
[10] “什么是事件建模？（附示例）” https://www.goeleven.com/blog/event-modeling/
[11] “事件建模备忘单” https://eventmodeling.org/posts/event-modeling-cheatsheet/
[12] “事件风暴与事件建模 | Rafal Maciag @DDD Istanbul” https://youtu.be/LDPlvmD9upk?t=2274
[13] Gamestorming | 维基百科 https://en.wikipedia.org/wiki/Gamestorming
[14] Miro https://miro.com/

在此，我们仅介绍核心概念，欢迎您访问本节下面的链接以了解更多详细内容。

事件

任何“事件风暴”都是从收集有关业务的知识开始的，这些知识以领域事件的形式表达，按时间顺序从左到右贴在墙（电子或实体）上。领域事件是指对您的业务至关重要并对其产生明确影响的任何事件。事件可能发生在业务系统内部或外部。按照惯例，每个事件用过去时态的动词来表达，写在橙色的便利贴上。
示例：“已请求预订”，“已接受预订”，“行程已开始”，“行程已完成”。

命令

在您觉得大部分领域知识都反映在墙上的事件中之后，就可以重点关注触发适当事件的命令。在浅蓝色的便利贴上写下对业务很重要的命令，并将其放置在它们产生的事件左侧。命令是代表用户意图的消息，可以表示为一个动作。
示例：“申请预订”、“取消预订”、“申请退款”。

策略

策略工件用于记录事件发生的条件和策略。因此，在风暴墙上，策略介于领域事件和命令之间。策略的表示形式为“每当...X，则...Y”或“如果...X，则...Y”。
示例：“每当创建预订时，则创建一份报价。”

另请参阅：

- 开放实践库中的事件风暴[15]
- 开放实践库中的全局[16]
- 全局事件风暴[17]

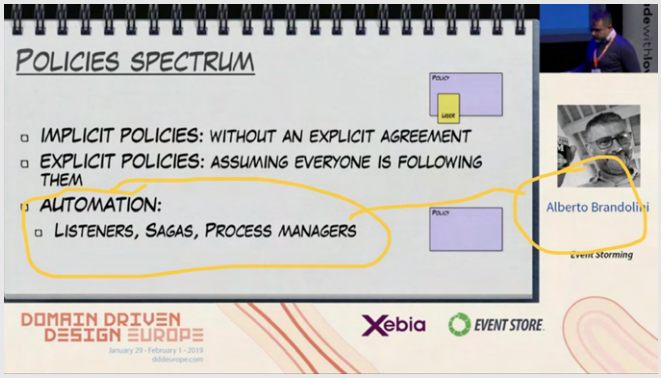
3.1.1.2. 事件从概念到技术“放大”的示例

如前所述，以事件为中心有助于在设计研讨会和随后的实施迭代期间突破业务/技术边界。在本节中，我们将展示用户/业务情境中的事件（即领域事件）如何帮助“放大”特定的技术实现，同时使用相同的“事件语言”。

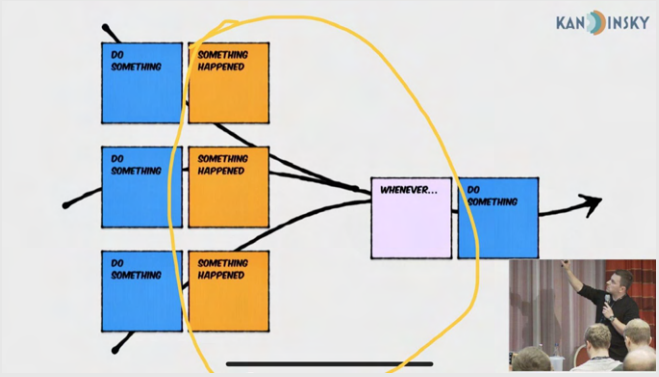
热门的“事件风暴”实践的原作者 Alberto Brandolini 提到了以下几种自动化“策略”（在“事件风暴”情境下）：

- 监听器
- 事务流
- 流程管理器

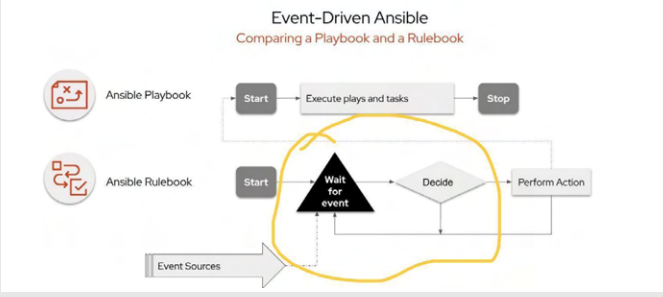
[15] 开放实践库中的事件风暴 https://openpracticelibrary.com/practice/event-storming/
[16] 开放实践库中的全局 https://openpracticelibrary.com/practice/the-big-picture/
[17] 全局事件风暴 https://medium.com/@chatuev/big-picture-event-storming-7a1fe18ffabb



另请参阅：“事件风暴 - Alberto Brandolini - DDD Europe 2019”[18]



该主题的另一位专家 Mariusz Gil 也在他的演讲“使用事务流/流程管理器模式对复杂流程和时间进行建模”中详细阐述了流程管理器模式。[19] Mariusz 还特别展示了“流程管理器”自动化模式的实施演示。通俗地说，这是关于将三个领域事件流合并成一个事件流的过程。谈到技术方面，在本指南中，EDA（Ansible Rulebook）非常适合“流程管理器”模式。下面的示意图展示了它的工作原理：



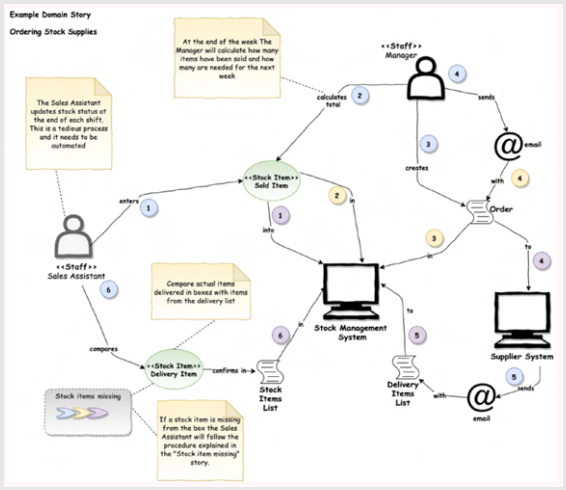
图片来自：“Ansible 自动化平台 2.3 新增功能”[20]

[18] 事件风暴 - Alberto Brandolini - DDD Europe 2019 https://www.youtube.com/watch?v=mLXQIYEwK24
[19] 使用事务流/流程管理器模式对复杂流程和时间进行建模 - Mariusz Gil https://www.youtube.com/watch?v=WvjTCmeGlGA
[20] Ansible 自动化平台 2.3 新增功能 https://www.ansible.com/blog/whats-new-in-red-hat-ansible-automation-platform-2.3

3.1.2. 领域发现的传统视角：“讲述领域故事”实践

描绘项目领域的一种方法可能是使用“讲述领域故事”实践。例如，您可能会遇到有多个人员或系统积极参与的流程部分。如果这些部分对您的领域分析至关重要，您可能希望从另一个角度了解流程，进一步将这些部分作为“领域故事”进行建模。

一图胜千言。下面是一个领域故事示例：“订购库存物资”。正如您所看到的，这个特定示例的图形符号类似于 UML 组件/部署图。因此，对于项目团队中的工程师来说，它可能是“友好的”。



图片来自：“领域故事示例”[21]

“讲述领域故事”的一个重要方面是，它是当前状态和预期状态之间的桥梁。因此，采用这一实践有助于进一步详细规划项目目标。以下是建议的步骤：

1. 让一位主题专家开始介绍业务流程（当前状态）
2. 主持人或支持团队在工作流程图中记录每个步骤
3. 对每个记录的步骤进行编号，并在相应编号旁添加详细说明
4. 重复该过程，直至达到预期状态

您可以在此处阅读有关“讲述领域故事”的更多信息。[22]

3.1.3.“示例企业”的示例项目：动机映射

正如我们所知，业务挑战往往是动态的，它们可能具有明显的季节性，而且会迅速、反复地发生变化。要及时掌握利益相关者需求，其中一种方法是深入了解他们的动机并运用“动机映射”实践确定映射关系。[23] 在本节中，我们将采用这一实践来构建本指南中示例项目的结构。在实际环境中，填好的“动机映射”视觉画布为项目团队和所有利益相关者呈现“同步动机”。这有助于每个人密切关注其实际活动是否符合计划的战略方向。

[21] 领域故事示例 https://medium.com/domain-driven-stories/example-of-a-domain-story-d20ade05831e
[22] 开放实践库中的讲述领域故事 https://openpracticelibrary.com/practice/domain-storytelling/
[23] 开放实践库中的动机映射 https://openpracticelibrary.com/practice/motivation-mapping/

总之，“动机映射”利用简单的画布来组织以下内容：

- **情境 –** 我们或我们客户周围的环境如何？我们在哪个平台上运行？有哪些因素会影响我们的环境？
- **目标 –** 我们或我们的客户有哪些目标？我们想要取得什么结果？
- **制定目标的原因 –** 明确我们制定这一目标或目的的原因。
- **障碍 –** 是什么阻碍了我们在实现目标方面取得进展？有哪些阻碍因素？
- **目标成果 –** 我们需要达到什么状态才说明我们已经实现目标？

在自动化环境下，我们建议检查以下优势是否与您的“动机映射”画布相关：

- 自动化可提高执行的一致性
- 缩短任务完成时间
- 提高团队效率，并增强执行更重要任务的能力
- 增强对漏洞管理流程的信心
- 简化跨环境部署
- 提高流程的可扩展性和可延展性
- 与 Ansible & co. 合作实现更多的“基础架构即代码”
- 改进功能测试
- 减少手动工作/开销，降低发生错误的可能性
- 某些自动化内容可在其他业务工作流中重复使用

本指南中的“示例项目”发生在一个名为“示例企业”的虚构企业中，该企业最近受到了扩展的法规和合规要求的约束。这是因为该公司正在将业务拓展到金融行业以及新的地理区域。例如，PCI-DSS 标准[24] 要求安装最新的补丁，同时规定了安装关键补丁的最长时限。本指南中的“应用”现在对“示例企业”的运营至关重要。这是用于在“示例企业”中请求访问权限的主要软件，因此其稳定性和可用性至关重要。

在项目的初始阶段，让我们也从“示例项目”团队的角度做一些评论。以下筹备活动将非常及时：

- 入职活动、测试所需的远程协作工具、安排一周会议、为白板会议准备墙壁（如果计划开展现场活动）
- 进行适当的介绍并确保每个人达成共识
- 讨论何为高绩效团队。它的优势和需要经历的阶段（Tuckman 的“形成–风暴–规范–执行”模式）
- 讨论并商定工作方式以及我们将如何作为一个团队开展工作（还可能制定并商定“社会契约”）。

以下是本指南中“示例项目”的“动机映射”画布的情况。它是由“示例项目”团队和为“示例企业”工作的知情利益相关者在一个小时的会议期间制作完成：

[24] PCI DSS | 维基百科 https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard

情境

类别“应用”（一个对业务至关重要的应用）

数据中心内的一台 Linux® 服务器运行着一个重要的业务应用，用于在整个企业内请求访问权限。最近，该应用的用户数量增加，并报告了不稳定行为。出于合规原因，该应用应该继续在 Linux 服务器上运行。目前，应用的部署由人工管理。

类别“修补机群”（Linux 机群的修补和安全修复）

公司正在向新市场扩张，现在他们需要更多的产能。剩余的手动步骤成为每天的瓶颈。预计在未来几个月内，企业内整个 Linux 机群的工作量和完成时限要求将迅速提升。

目标

应用

业务用户应能全天候稳定地访问应用。

修补机群

修补工作应完全自动进行。这种自动化将极大地提高向内部客户交付更新的速度和质量，同时使专家能够专注于更有意义的工作。

制定目标的原因

应用

随着进入不同时区的新市场，业务几乎全天候运营。尤其是在夜间和周末（原时区），更多用户需要使用该应用。越来越难找到适合进行应用升级/维护的“维护时段”。此外，新入职的高管对这款应用的缓慢响应速度也感到不满。

修补机群

随着越来越难满足对 Linux 服务器的监管合规要求，实现全面自动化势在必行。这一过程使员工无法从事更重要的活动。由于涉及大量的手动工作，出现人为错误的概率也很高。基础架构团队希望将工作方式转变为“站点可靠性工程师”模式。

障碍

“应用”和“修补机群”类别：

出于安全原因，为实施应用高可用性（HA）的工程师提供入职培训和改进修补流程将需要 2 到 4 周的时间。此外，项目团队可利用的区域数据中心的开发（非生产）环境计算能力目前也很稀缺。额外的计算能力可在 3 至 6 周内置备。

目标成果

应用

- 生产环境中新应用版本的更新，以及相应 Linux 系统的安全修补和维护，都应在不中断业务用户服务的情况下进行。
- 即使在使用高峰期，应用也应响应迅速。在极端情况下，三秒的响应时间是可以接受的。不过，95% 以上的用户请求应在一秒内完成。

修补机群

- 对于任务关键型 Linux 工作负载，自动化的修补和安全修复应该在不中断服务的情况下进行。对于该应用，这应该在六周内完成，其他几个任务关键型 Linux 工作负载应在两三个月内完成。
- 对于非关键型 Linux 工作负载，最多可在 24 小时内完成 3,000 台服务器的自动修补和安全修复（允许计划内停机）。这种“修补吞吐量”水平应在三个月内达到。

3.1.4. 示例：初始发现

对于事件风暴研讨会而言，有合适的人员参与是非常重要的。这包括知道要问什么问题的人员和知情的利益相关者（领域专家、产品负责人）。如果有一位出色的主持人和引导者（Scrum Master、敏捷教练）的帮助，研讨会就可以顺利进行。

在筹备“事件风暴”研讨会期间，“示例项目”团队决定重点关注以下知识领域：

- 使用中的应用 – 请求访问权限
- 部署新版本应用
- 修补应用的 Linux 服务器

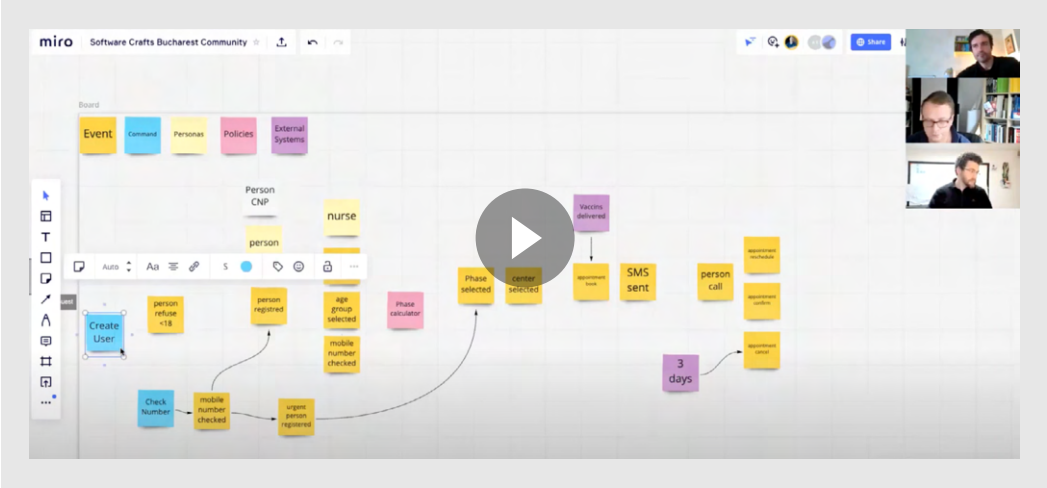
项目团队针对上述每个知识领域采取了以下步骤：

- 步骤 1 - 收集领域事件（全局）
- 步骤 2 - 完善领域事件（全局）
- 步骤 3 - 跟踪原因（流程建模）

在本指南中，最重要的知识领域是“部署新版本应用”。下一节列出了“示例项目”团队针对该领域按上述三个步骤进行事件风暴的结果。其他两个领域的结果可以分别在以下附录中找到：

- 附录 A. 针对“使用中的应用 – 请求访问权限”的事件风暴
- 附录 B. 针对“修补应用的 Linux 服务器”的事件风暴

如果您有兴趣了解从头开始的完整事件风暴会议示例，可以观看以下视频：“Bucharest 软件工艺社区的事件风暴研讨会”[25]

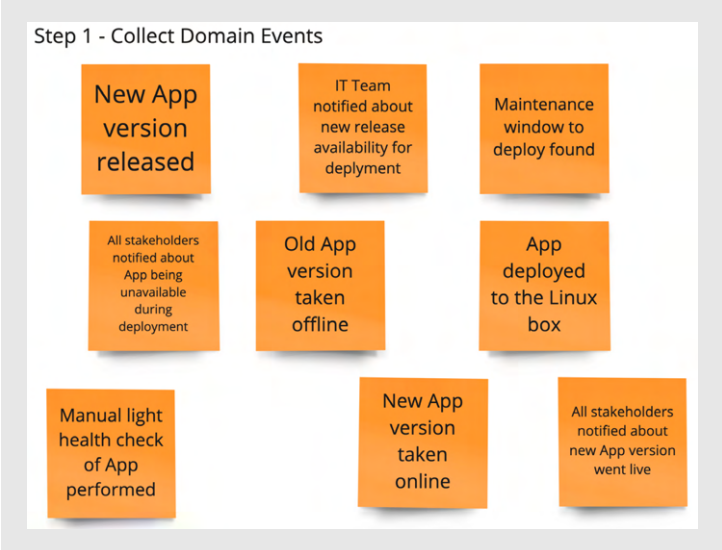


[25] Bucharest 软件工艺社区的事件风暴研讨会 https://www.youtube.com/watch?v=xVSaDdj3PVE

3.1.4.1. 示例：针对“部署新版本应用”的事件风暴

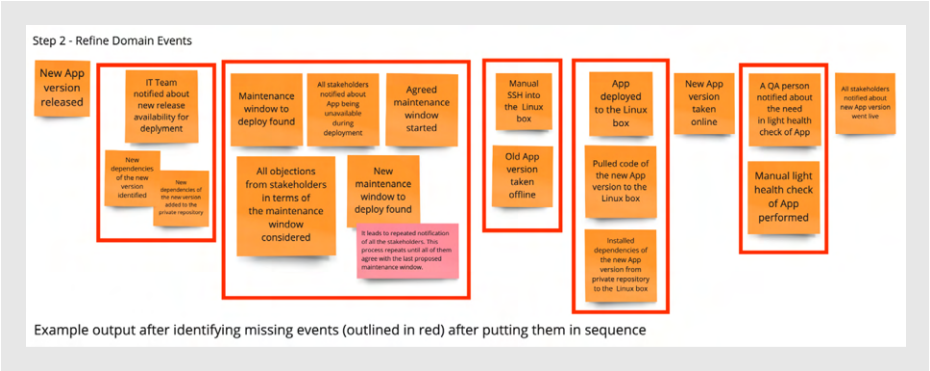
在“步骤 1 - 收集领域事件（全局）”中，每位参与者仅使用橙色便利贴。每张橙色便利贴代表一个专业事件。专业事件是指在业务过程中发生的与技术相关的事实。因此，便利贴上的动词必须是过去式。

第一轮是关于现有领域事件的纯头脑风暴过程。让参与者按事件发生的时间顺序将事件贴在墙上。



在“步骤 2 - 完善领域事件（全局）”中，与参与者一起查看领域事件便利贴。让参与者解释每个事件的含义。检查语法是否正确。

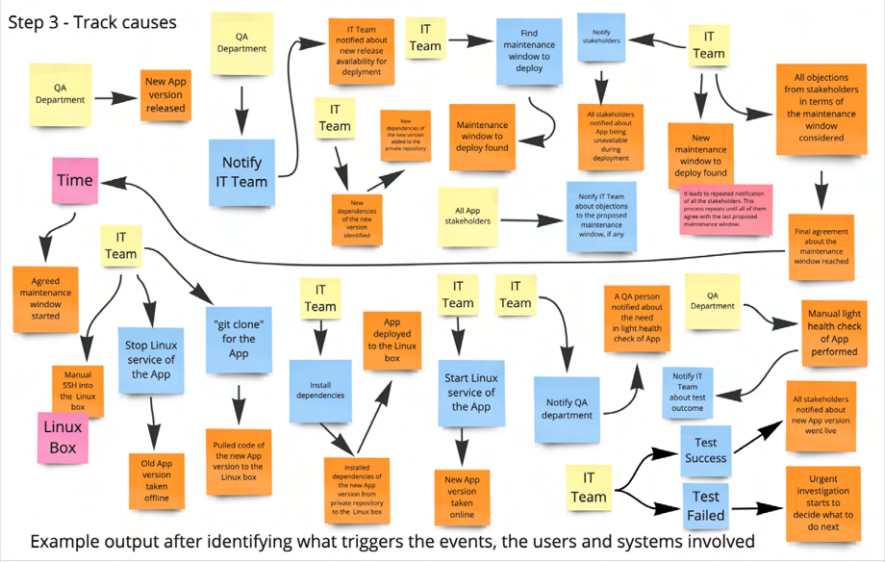
此外，还需再次讨论事件是否按时间顺序正确排列。统一出现的同义词（描述相同事物的不同术语）；如果同一术语用于描述不同的事物，则要突出差异。



在“步骤 3 - 跟踪原因（流程建模）”中，进行原因分析。领域事件从何而来？主要有四个原因：

- 用户操作（命令）
- 外部系统
- 时间（例如，预约已过）；业务流程
- 其他领域事件（通过自动反应；通过策略/业务规则）

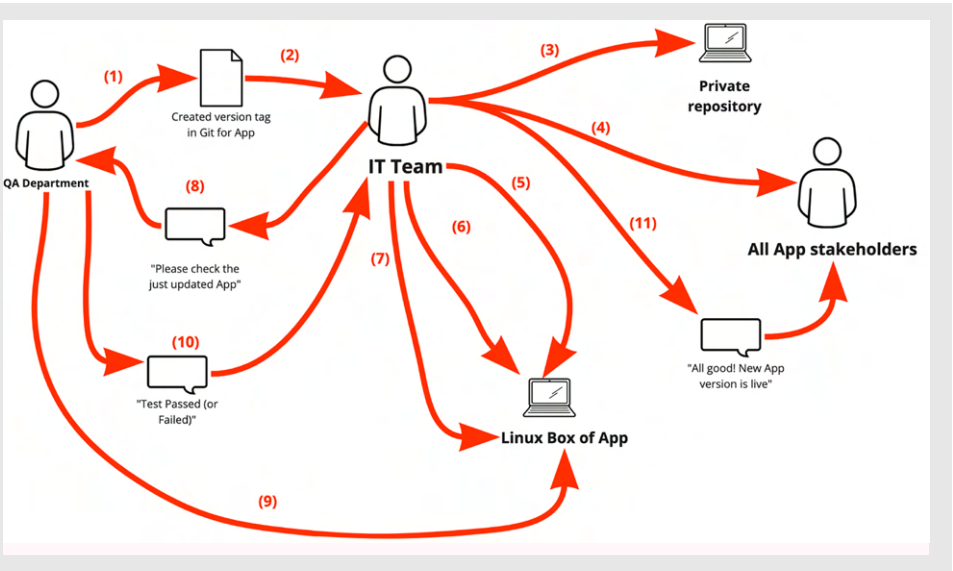
向参与者询问领域事件的触发因素。



3.1.4.2. 示例：“部署新版本应用”的领域故事（当前状态）

“示例项目”团队决定他们还希望多做一些努力，从另一个角度了解“部署新版本应用”的流程，他们另外将该过程中的合作部分建模为一个领域故事。该团队通过以下步骤绘制了下图及其说明：

- 让一位主题专家开始介绍业务流程当前状态
- 主持人或支持团队在工作流程图中记录每个步骤
- 对每个记录的步骤进行编号，并在相应编号旁添加详细说明



1. QA 部门在 Git 版本控制系统中标记新发布的应用
2. QA 部门通知 IT 团队有新版本可供部署
3. IT 团队识别新的依赖项，并将工件添加到私有存储库中
4. IT 团队与所有应用利益相关者就部署新版本应用的维护时段达成一致意见
5. 当商定的维护时段开始时，IT 团队手动进入 Linux 服务器，并停止应用（旧版本）的 Linux 服务
6. IT 团队将新版本的应用部署到 Linux 服务器中：（a）将新版本应用的代码提取到 Linux 服务器上；（b）将新版本应用的依赖项从私有存储库安装到 Linux 服务器中
7. IT 团队启动该应用的 Linux 服务，使其在线运行
8. IT 团队通知 QA 部门需要对刚更新的应用进行轻量级健康检查
9. QA 部门对刚刚部署的应用进行手动轻量级健康检查
10. QA 部门将测试结果通知 IT 团队
11. 如果新应用版本的轻量级测试结果报告为“通过”，那么 IT 团队会通知所有利益相关者新应用版本已上线

3.2. 我们面临哪些问题？我们的流程中存在哪些瓶颈？

现在，我们希望完善目前存在的问题、难题、低效和风险的理解。从当前情况和战略角度来看，哪些改进措施最具影响力？我们希望“项目团队”在流程改进工作上保持一致。在清楚了解我们的流程瓶颈之后，“项目团队”就可以集中精力，从战略角度出发，对当前最相关的流程进行改进。让我们在接下来的章节中看看 MBPM 实践如何帮助解决这个问题，并在此之后总结我们迄今为止收集到的所有相关信息。

3.2.1. 什么是“基于指标的流程映射”实践？

让我们来介绍一下 MBPM。MBPM 是一项精益流程改进技术。MBPM 工作有助于确定可能的流程改进及其影响。MBPM 实践会针对向客户/利益相关者提供结果/服务的流程绘制详细地图。该图显示流程步骤、责任人、交付周期指标和质量指标。它有助于确定下文所描述的对流程的潜在改进。

因此，在“执行 MBPM”时，对于**流程**中的每个**步骤**，项目参与者会识别或估计以下内容：

- 描述流程步骤操作的**动名词短语，例如“测试代码”**（为清晰起见，必要时可提供额外细节）
- 执行步骤的**团队**或**行动者**
- **流程时间（PT）**，即团队或行动者实际积极执行步骤中操作、从事相应有效工作的时间
- **交付周期（LT）**，或经过的时间：
  - **从**某项工作准备就绪，可以由团队/行动者在这一特定步骤中执行开始（换句话说，没有合理的依赖关系；例如，积压工作项满足“就绪的定义”）
  - **到**将这一特定步骤的输出/结果交付给下一个“工作步骤”（作为相应的输入）
- **“完整且准确百分比”（PCA** 或 **“%C&A”）**，即特定步骤的输出完整且准确的百分比。如果下游“工作中心”（团队/行动者）不需要任何进一步的更正、补充或澄清，就可以开始下游工作，那么输出就是完整且准确的。

例如，“映射”特定流程中的工作活动有助于确定：

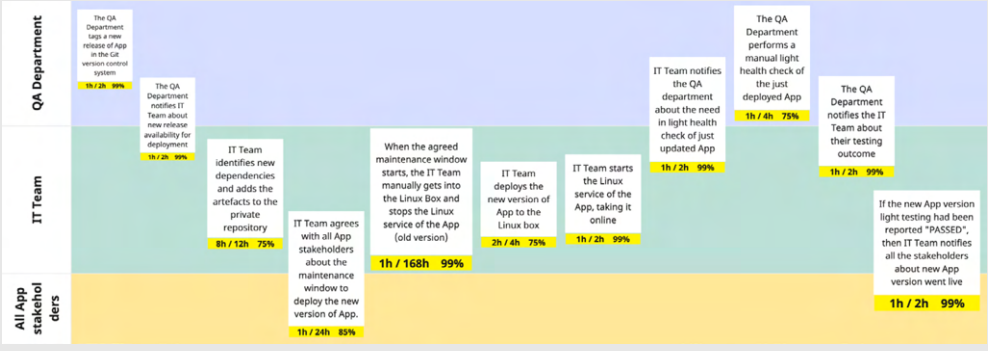
- 无缺陷交付成果的百分比，这是流程的产出。这些交付成果的 LT。
- 在某些中间流程步骤中需要修正或返工，从而导致流程缓慢的交付成果百分比。
- LT 较长而 PT 较短的步骤，这表明存在瓶颈。这些步骤可以通过一些具有更出色（更高）PT/LT 比值和更低 PT 的替代方案进行高效改进，甚至“重构”。

如果您对本指南范围之外的 MBPM 示例感兴趣，可观看以下视频：“使用基于指标的流程映射可视化、衡量和优化流程”。[26]

3.2.2. 示例：MBPM – 当前的流程、负责人和指标

下图展示了 MBPM 在“部署新版本应用”方面的示意图。每个流程步骤的底部都有如下数字：

流程时间（PT）/交付周期（LT）“完整且准确百分比”（PCA %）



[26] 利用基于指标的流程映射来可视化、衡量和优化流程 https://www.youtube.com/watch?v=g1XSbEwR3bU

让我们将收集/估算的 PT-LT-PCA 数据汇总成表，以便更简单地进行分析。它将展示 MBPM 实践如何以量化的方式提出流程改进建议。

步骤名称	PT（小时）	LT（小时）	PT/LT %	PCA %
1. 标记新版本	1	2	50%	99%
2. 通知 IT 团队有可用的新版本	1	2	50%	99%
3. 识别新的依赖项并更新私有存储库	8	12	67%	75%
4. 与每个人就用于部署的维护时段达成一致	1	24	4%	85%
5. 等待商定的维护时段开始并停止应用的 Linux 服务	1	168	0.5%	99%
6. 部署新版本应用	2	4	50%	75%
7. 启动应用的 Linux 服务	1	2	50%	99%
8. 通知 QA 部门需要进行轻量级健康检查	1	2	50%	99%
9. 对刚刚部署的应用进行手动轻量级健康检查	1	4	25%	75%
10. 通知 IT 团队相关测试结果	1	2	50%	99%
11. 通知所有利益相关者新版应用已上线	1	2	50%	99%
汇总	**PT 汇总：19 小时**	**LT 汇总：224 小时**	**PT/LT 汇总：8.5%** “PT 汇总”除以“LT 汇总”	**PCA 汇总：33%** 所有 PCA% 值相乘

上述示例流程图和表格显示了以下重要结果：

- 33% 的交付成果在流程中“流动”，并且完整、准确、无缺陷。这些交付成果的交付周期为 224 小时。
- 大多数交付成果（67%）需要在某些中间流程步骤中进行修复或返工，这会导致交付周期超过“预期”的 224 小时。
- 步骤 4 和 5 的交付周期较长，并且 PT/LT 比率较低，这表明存在瓶颈。改进这些步骤将最大限度地缩短整个流程的交付周期。
- 虽然步骤 9 的交付周期相对适中，但 PT/LT 比率仍然过低。此外，其 PCA 值为 75%，表明因不准确/不完整而导致 LT 增高的风险相对较高。

3.2.3. 示例：当前面临的问题和风险

到目前为止，“示例项目”团队已经完成了以下工作：

- “动机映射”：总体目标、情境、制定目标的原因、障碍和目标成果，并在“动机映射”定义的情境下进一步分析以下具体知识领域
- 使用中的应用 – 请求访问权限
  - 事件风暴
- 部署新版本应用
  - 事件风暴
  - 领域故事（当前状态）
  - MBPM
- 修补应用的 Linux 服务器
  - 事件风暴

之后，“示例项目”团队进行了几次会议，总结了所处情境下的问题、难题、低效和风险。以下是在那些分析会议期间收集的结果：

（a）问题分析：“对业务至关重要的应用”

- 在应用的内部运行方面没有发现任何逻辑问题。由于用户负载增加，单个 Linux 服务器上的应用有时会无响应。这是因为单个 Linux 服务器的处理能力不足（可能是 CPU 过度利用）
- 部署新版本并非易事
- 对于应用的运行状况没有适当的可见性，用户端错误在事后才能检测出
- 维护时段：需要花费大量时间协商，并等待开始执行
- 人工测试需要跨部门协调；准备时间远远长于流程时间
- 风险：手动步骤和手动测试容易出现人为错误

（b）问题分析：“Linux 机群的修补和安全修复”

修补工作仅部分实现自动化。缺点：

- 过于频繁地需要手动干预；这种手动参与的工作量往往与 Linux 服务器的数量成正比
- 完成速度太慢
- 有时可以避免停机
- 维护时段：需要花费大量时间协商，并等待开始执行
- 风险：手动步骤和手动测试容易出现人为错误

总之，上述问题和风险与流程自动化程度较低或分配用于执行流程的计算能力不足有关。

那么，改善这一切的策略是什么呢？在下一节中，我们将介绍一些有用的相关实践，并查看“示例项目”团队据此做出的决定。

3.3. 我们是否准备根据实际需求进行交付？

在第 3.1.3. 节“示例：动机映射……”中，我们介绍了示例项目的总体目标成果。当我们完成当前交互的发现实践后，我们希望将优先事项与实际的业务/客户需求联系起来，因为这些需求会随着时间的推移而变化，甚至会使刚刚优先考虑的项目变得多余。基本上，我们在每次迭代中都会检查以确保不会白白花费精力。“输出”不等于“结果”！也就是说，“实际付出努力的所有结果”与“仅对利益相关者有价值的结果/成就”是不一样的。

换言之，在“发现/开发”非初始迭代期间举行“目标成果”会议时，其主要关注点是刚刚完成迭代的结果/成果。我们希望进行交叉检查，并验证我们目前的工作方向是否最有效。

下文所述的“目标成果”实践有助于我们有条不紊地进行此类交叉检查。

3.3.1. 什么是“目标成果”发现实践？

“目标成果”实践是发现迭代的成果要素中的主要实践。它将从其他发现实践中获得的发现和经验总结成一套有限的陈述，而且这些陈述可以公开展示，供团队和利益相关者定期参考。另请参阅：目标成果。[27]

在这一点上，让我们从项目团队的角度进行评论。以下任何一种情况都可能表明，确立目标成果将使团队受益：

- 团队交付成果的范围发生意外变化
- 您的团队在客户与产品互动或提供产品反馈之前就庆祝成功交付
- 团队日常互动的重点是完成功能，而不是完成一个能达到预期效果的功能版本

3.3.2. 示例：首次交付迭代的策略

我们正在为“示例项目”的首次“交付迭代”做准备。在此阶段，专注于打造 MVP 并不罕见。什么是 MVP?

3.3.2.1. 什么是 MVP?

一般而言，MVP 旨在实现以下目标：

- 提供用户/利益相关者可以触摸/感受/互动的成果
- 在行动中展示产品理念，使利益相关者对项目更感兴趣，并吸引更多潜在用户
- 尽早验证产品创意是否值得投入更多精力/时间。否则，应明确指出在产品上花费更多资源是不合理的
- 收集真实用户/利益相关者的宝贵反馈意见

简单地说，这就是“完成整个旅程的最短路径”，而“整个旅程”则代表项目未来的总体成果。换言之，MVP 不关注“花里胡哨”的功能。

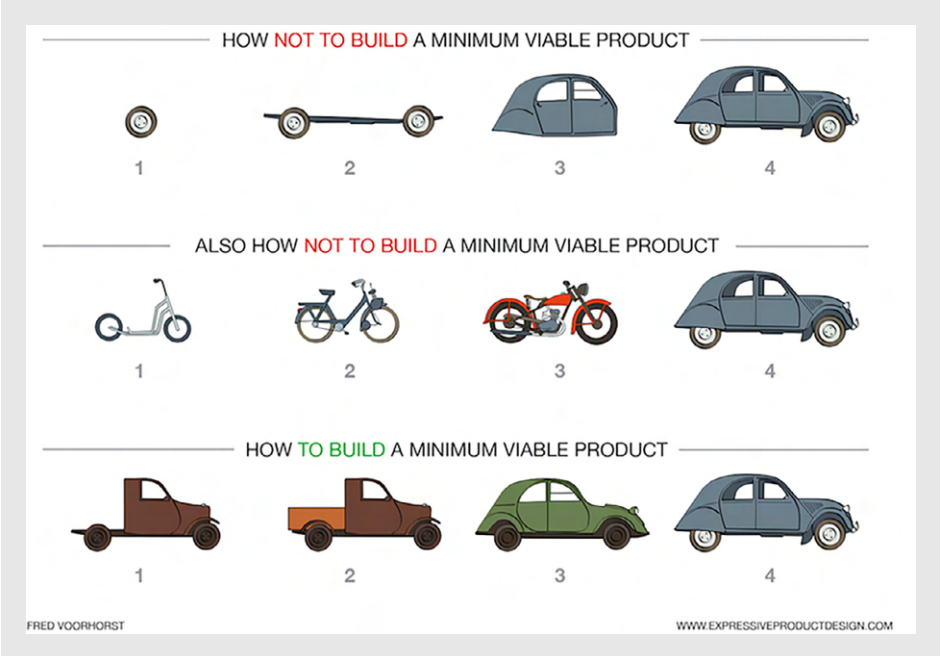
[27] 开放实践库中的目标成果 https://openpracticelibrary.com/practice/target-outcomes/

在 MVP 开发过程中，需要兼顾以下两方面：

A. 有效地推进整个项目计划的执行，使其更接近项目的最终总体理想成果

B. 在 MVP 发布之际，已经或很快会有一个对未来整个项目成果的最终用户非常有价值/有用的成果

在下图中，第三行描述了一个同时满足上述（A）和（B）两点的例子。至于图片中的另外两个示例，第二个示例不符合（A）项，而第一行不符合上面的（B）项。



在下一节中，我们将概述示例项目的 MVP。

3.3.2.2. 示例：MVP 想法和选择

总体而言，示例项目团队一致认为，以 MVP 的形式开始提供与示例项目目标相匹配的价值是明智之举。一旦 MVP 以合理的成本向利益相关者证明了它的所有价值，示例项目团队就可以在以后根据团队投票/优先级增加功能。

换句话说，无论团队/利益相关者现在提出什么功能，它们都可以分为两大类：

- MVP 范围内
- MVP 发布后

需要指出的是，所有预先计划用于某个“MVP 发布后”版本的功能/项目，最终可能会实施，也可能不会实施。MVP 发布后不久，将收集用户/利益相关者的反馈，所要求的改进措施可能会与已有的“候选方案”一起考虑纳入下一个版本。

在第 3.1.3. 节“示例：动机映射……”中，我们介绍了示例项目的总体目标以及其他背景细节。这是它们对应的领域：

（a）对业务至关重要的应用

业务用户应能全天候稳定地访问应用

（b）Linux 机群的修补和安全修复

修补工作应完全自动进行

基于这些目标，项目团队决定召开一次“让我们共同确立 MVP”会议。团队成员提出了一些想法，并将它们写在白板/Miro 板上。他们集体努力了大概 10 分钟才完成。会议主持人对这些想法进行了进一步的分组和细化。在进行分组活动的同时，还围绕这些要点进行了一些半非正式的讨论，目的是让所有参与者都能全面地了解这些要点。这又花了 25 分钟。

这里是针对示例项目在第一次交付迭代中要交付的 MVP 的最终改进想法：

A. 在停机情况下进行自动修补（适用于非关键 Linux 服务器）

B. 将应用更换到 Kubernetes/红帽 OpenShift 平台

C. 对应用部署、修补、可观测性和性能进行改进（我们将仅自动修补应用。稍后，我们将尝试将此想法扩展到更多具有 HA 需求的 Linux 服务器。）

在讨论上述事项时，团队强调了以下几点：

- **想法（A）**详细说明了“修补工作应完全自动进行”，但没有对应用给予合理的关注，例如在示例项目中对其可用性的期望。
- **想法（B）**有可能成为应用运维就绪方面的突破点，利用了最近在容器化应用编排、可观测性和可用性等方面的发展。但是，这里缺少“自动修补 Linux 机群”方面的内容，而企业自己的 Linux 机群是受益于示例项目成果的 IT 资产的一个自然而重要的组成部分。
- **想法（C）**涵盖了相关的应用改进以及 Linux 服务器的修补。该选项的一个缺点是，目前并不清楚究竟需要做些什么。不过，这个想法的原创者还是通过对技术设计的一些见解说服了团队，让大家相信它并没有那么糟糕

主持人决定举行一次投票会议，就最终的 MVP 创意（即团队决定交付的创意）达成一致。会议持续了 7 分钟。以下是投票结果：

- **第 1 名 – 想法（C）：**对应用部署、修补、可观测性和性能进行改进
- **第 2 名 – 想法（B）：**将应用更换到 Kubernetes/红帽 OpenShift 平台
- **第 3 名 – 想法（A）：**在停机情况下进行自动修补（适用于非关键 Linux 服务器）

正如之前提到的，目前获胜的想法（C）的一个缺点是，尚不完全明确需要采取哪些具体措施才能实现 MVP 的成果。在下一节中，我们将介绍几种有助于我们获得所需明确指示的实践。

3.4. 是否一切准备就绪，可以开始第一次交付迭代？

正如我们之前所讨论的，示例项目团队一致认为，以 MVP 的形式开始提供与示例项目目标相匹配的价值是明智之举。MVP 是：

对应用部署、修补、可观测性和性能进行改进
（我们将仅自动修补应用。稍后，我们将尝试将此想法扩展到更多具有 HA 需求的 Linux 服务器。）

这些改进必须与整体项目目标一起推进：

- 业务用户应能全天候稳定地访问应用
- 修补工作应完全自动进行

现在，应由示例项目团队和知情的利益相关者提出要在简单项目中交付的特定功能。然后将它们分为两大类：

- MVP 范围内
- MVP 发布后

“MVP 范围内”的功能将有效地构成 MVP 的工作范围（SoW）。

现在，团队对“示例企业”的领域已经非常了解。他们有全局事件风暴结果和“领域故事”图。他们通过应用 MBPM 实践找出了领域流程中的“瓶颈”。此外，他们总结了当前存在的问题，如“3.2.3. 示例：当前面临的问题和风险”一节所述。

这一切都是以团队/产品负责人希望交付的具体功能和非功能需求为依据。在下一节中，我们将快速修订一些相关术语和概念，然后重点确定“示例项目”首次交付迭代的功能。

3.4.1. 什么是“用户故事映射”和“价值切片”实践？

首先，我们需要确保在相关术语方面有一致的理解。

3.4.1.1. 术语

一般来说，“史诗”可以嵌套“功能”，而“功能”可以嵌套“用户故事”。它们都可以用来表示产品/系统的功能性和非功能性需求。“用户故事”只是“通过对话获取需求”的一种技术。在这里，非功能性需求与功能性需求一样，只是对话的一个类别。它们都是我们要交付的项目的一个方面。

史诗：

- 为利益相关者/客户提供新产品、解决方案和服务的大型计划
- 由大量功能组成

功能：

- 产品负责人感兴趣的功能
- 为用户/利益相关者提供价值
- 通过一定数量的用户故事来实现

用户故事：

- 代表用户/利益相关者的个体需求
- 描述对用户/利益相关者有价值的大量功能/非功能内容
- 作为项目规划的原子项
- 代表项目中最小的价值增量
- 便于作为项目团队针对性对话的关注点

3.4.1.2. “用户故事映射”和“价值切片”

“用户故事映射”是传统敏捷积压实践的演变。这是创建轻量级发布计划的有效实践，可以推动标准的敏捷交付实践。如果应用得当，它可以为您带来以下好处：

- 项目团队认为可在计划时段内交付的范围项目（以故事或简单的功能标题形式记录）的积压清单
- 将积压工作“切片”为大约三次“交付迭代”，使其成为最近交付计划的大纲
- 为计划的第一次“交付迭代”提供足够的细节，以便开始工作

另请参阅：用户故事映射和价值切片。[28]

下面是一个用户故事映射示例，其中通过“价值切片”确定了初始（MVP）交付迭代。**注意**：下图中的领域与指南中的“示例项目”不同，这个特定示例是针对一个虚构的现代出租车服务领域。我们将在接下来的章节中将这些实践应用到“示例项目”中。

下面的映射是由一个团队确定的，还有一些初步分配给“版本 1”和“版本 2”迭代的“价值切片”。

图片来源。[29]

上图中的绿色卡片代表产品的特性（产品功能）。蓝色卡片代表建议的用户故事（项目中的最小价值增量）。

一般来说，有两种截然不同的方法来对敏捷积压工作进行“切片”：

- **水平切片**，重点是逐个处理架构层
- **垂直切片**，通过端到端功能切分整个架构堆栈的工作

如果您的首要任务是**交付真正的价值**，那么在通常情况下，我们建议在形成“工作块”时**使用纵向切片而不是横向切片**。

[28] 开放实践库中的目标成果 https://openpracticelibrary.com/practice/target-outcomes/

[29] 故事图、MVP、优先级、估算 - 第 6/7 部分 | 敏捷产品之旅（从构思到开始）https://www.youtube.com/watch?v=D18HFmVLXQc

水平切片通常应被视为一种后备选择。例如，当在接下来的交付迭代中只有可以处理系统的单个层的团队成员可参与时。比如，仅对一个假想系统的业务逻辑层进行改进，这样系统的数据库层和表现层就无法在下一次迭代中进行物理改进，没有专家可以对它们进行处理。这意味着在下一次迭代中不能处理需要更改其他两层的“垂直切片”。在这种情况下，可以在迭代中加入与业务逻辑层相关的横向切片，以保持项目工作向前推进。

另请参阅：针对敏捷实践选择水平切片还是垂直切片？[30]

3.4.1.3. 您的“用户故事”是否已准备好开始实施?

我们怎样才能确信用户故事已经准备好实施了呢？在“支付出租车费用”功能下仅提供“现金”选项（用户故事的原型）似乎还不够。此时，下面的“就绪的定义”（“就绪”是指用户故事描述可用于开始实施）检查清单可以帮助项目团队详细说明用户故事：

- 团队已了解用户故事
- 用户故事具有明确的商业价值
- 对用户故事进行估算
- 确定用户故事的依赖关系
- 用户故事规模较小
- 定义了用户故事的验收标准

“良好用户故事”的另一组著名的“3 **C**”标准如下：

卡片（**C**ard）
- 写在便签卡上
- 可标注估算值、数值、注释等

对话（**C**onversation）
- 故事细节通过与客户的对话逐渐显现

确认（**C**onfirmation）
- 验收测试用于确认故事的实施是否完成

最后但并非最不重要的一点是，这里有几个合理的模板，可用于“良好用户故事”的主要文本：

- **作为一个** {角色}，**我可以** {做或拥有一些可衡量的品质} **来** {实现业务目标}
- **为了** {实现业务目标}，{角色} **可以** {做或拥有一些可衡量的品质}

3.4.2. 示例：史诗、功能和初始用户故事图

如前几节所述，“示例项目”团队决定构建以下 MVP：

对应用部署、修补、可观测性和性能进行改进

MVP 想法的原作者给了团队一些关于他们心中的技术设计的见解。已为示例项目的初始范围确定了以下两个史诗：

- **史诗：滚动更新应用**
  例如，用新的应用版本滚动更新 Linux 服务器的功能（集成到 Linux 修补流程中）
- **史诗：远程健康检查器**
  用于观察应用健康状况的健康检查器应用（类用户远程组件）

[30] 针对敏捷实践选择水平切片还是垂直切片？https://www.youtube.com/watch?v=jQg27pFGmWA

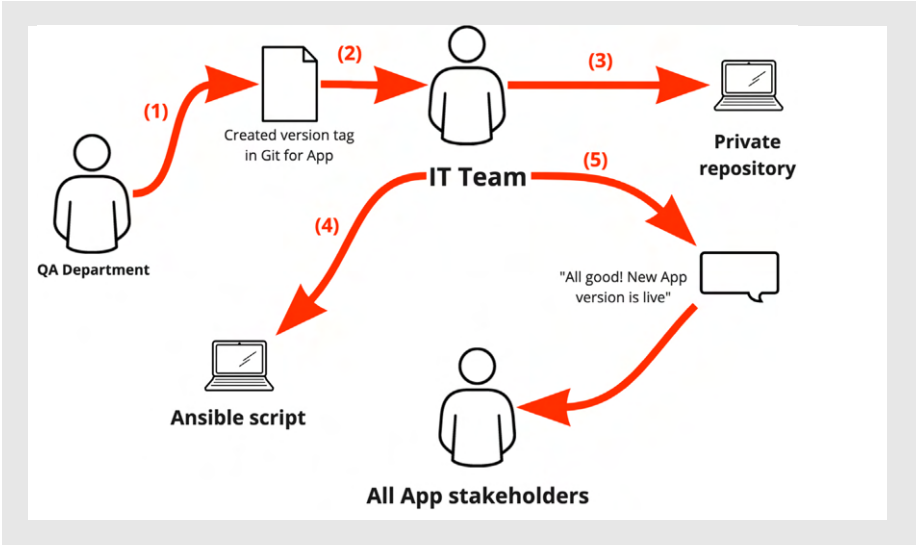
此时，“示例项目”团队决定，有必要开展“第二轮”领域故事会议，探讨“预期状态”。会议的目的是进一步明确哪些功能应该/可以作为 MVP 的一部分或稍后完成。

3.4.2.1. 示例：“部署新版本应用”的领域故事（预期状态）

如“3.2.2. 示例：MBPM - 当前的流程、负责人和指标”一节所述，“示例项目”团队在部署流程中发现了以下“瓶颈”：

- ...
- 与每个人就用于部署的维护时段达成一致
- 等待商定的维护时段开始并停止应用的 Linux 服务
- ...
- 对刚刚部署的应用进行手动轻量级健康检查
- ...

“示例项目”团队邀请了主题专家，即 MVP 想法的原始作者。首先，他们希望专注于与当前确定的“瓶颈”相关的改进。他们已经开展了一次会议，并提出了以下图表，描述了流程的期望状态。



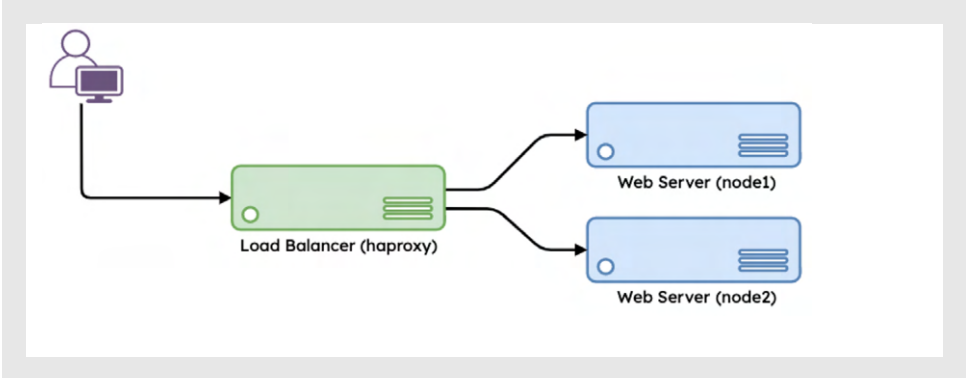
1. QA 部门在 Git 版本控制系统中标记新发布的应用
2. QA 部门通知 IT 团队有新版本可供部署
3. IT 团队识别新的依赖项，并将工件添加到私有存储库中
4. IT 团队执行自动化 Ansible 脚本以部署新的应用版本
5. 如果脚本报告新版应用自动测试“通过”，则 IT 团队通知所有利益相关者新版应用已上线

正如您所看到的，上面的新图摆脱了先前确定的“瓶颈”。这要归功于 Ansible 脚本和该想法的作者所描述的一些架构变化，如下所示：

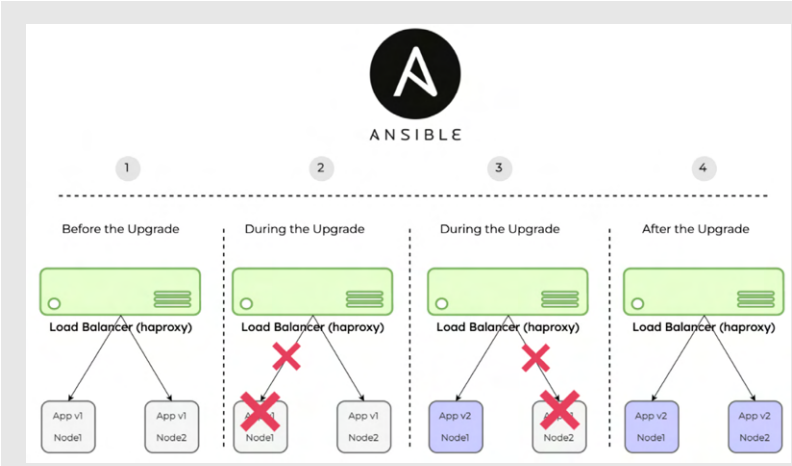
- 多个应用实例分布在相应的专用 Linux 服务器上，并由负载平衡器（例如 HAProxy）进行统一管理。[31]
  - 所有实例都用于处理用户请求，因此服务对用户具有更高的可用性。例如，如果其中一个实例出现故障，其他实例仍会处理请求;
  - 由于有更多的 CPU 可供使用，用户请求的处理速度更快
- 新版本应用实例的滚动部署在负载平衡器后进行。特别是，这些实例会逐个离线（然后恢复到“活动”状态），从而允许其他实例在整个滚动更新过程中为用户请求提供服务。

下面几张图直观呈现了专家向“示例项目”团队解释的内容（图片来自《Ansible 为现实生活提供自动化》）：[32]

应用托管在多个 Linux 服务器上，前面有一个负载平衡器



使用 Ansible 滚动更新应用

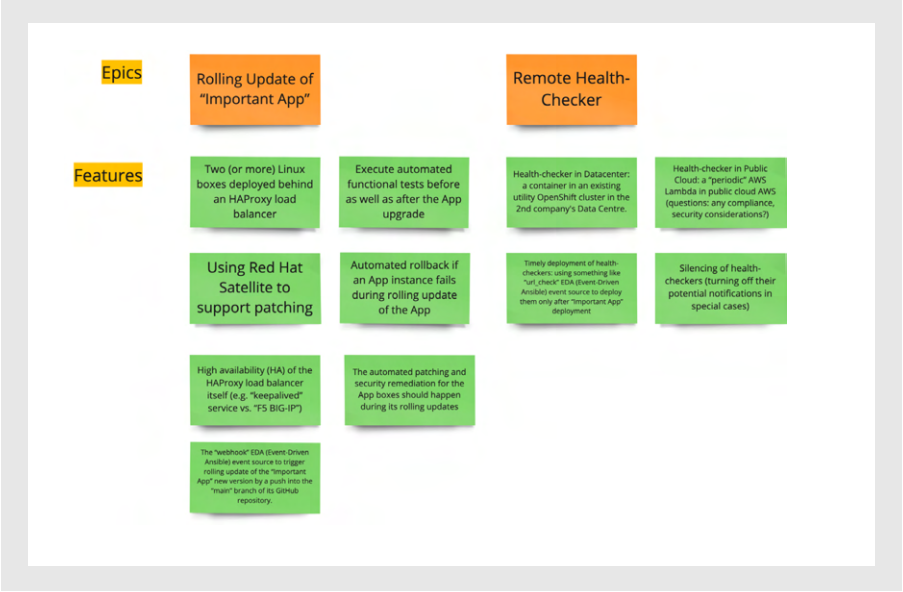


3.4.2.2. 示例：初始用户故事图

“示例项目”团队用一天时间进一步研究了上一节讨论的改进措施。之后，他们又召开了一次会议，开始绘制用户故事图，具体如下：

[31] HAProxy | 维基百科 https://en.wikipedia.org/wiki/HAProxy

[32] Gineesh Madapparambath 所著的《Ansible 为现实生活提供自动化》https://amzn.asia/d/flHSDjh



史诗：滚动更新应用
功能（利益相关者感兴趣的功能）：

- 两个（或更多）Linux 服务器部署在 HAProxy 负载平衡器之后
- 在应用升级之前和之后执行自动化功能测试
- 应在应用滚动更新期间进行应用服务器的自动化修补和安全修复
- 使用 Satellite 支持修补
- 如果应用实例在滚动更新过程中出现故障，可自动回滚
- “webhook”EDA（Event-Driven Ansible）事件源通过推送到 GitHub 存储库的“主”分支来触发应用新版本的滚动去更新。另请参阅：Event-Driven Ansible + Gitops。[33]
- HAProxy 负载平衡器本身的 HA（例如，“keepalived”服务与“F5 BIG-IP”）

史诗：远程健康检查器
功能（利益相关者感兴趣的功能）：

类用户（远程）健康检查器组件，有助于在真正的用户受到影响之前检测出面向用户的问题。它类似于 New Relic 的“合成”客户端或 AWS CloudWatch 的“金丝雀”。

- 数据中心的健康检查器：第二家公司数据中心现有的红帽 OpenShift 集群中的一个容器
- 公共云中的健康检查器：公共云 AWS 中的“定期”AWS Lambda（问题：是否有合规性、安全性方面的考虑？）
- 及时部署健康检查器：使用类似“url_check”的 EDA 事件源，仅在应用部署后才部署健康检查器
- 健康检查器的静默化（在特殊情况下关闭它们的潜在通知）

[33] Event-Driven Ansible + Gitops https://www.youtube.com/watch?v=Bb51DftLbPE

3.4.3. 第一次交付迭代会有哪些功能?

自动化几乎总是有帮助的，但其实施需要成本，因此我们需要对规划项目进行优先排序。这种优先排序是为以下目的所需：

- **充分提高所付出的努力的价值**
- **充分利用“未完成的工作”**（即，在没有意义时不要花费精力）
- **减少没有实际价值的“浪费”输出。**“输出”不等于“结果”！有些输出实际上是“浪费”，因为它们需要长期维护，而并没有给利益相关者/客户带来真正的价值。

在接下来的几节中，我们将介绍四种支持优先级排序的实践，并更多地关注“Kano 模型”实践。然后我们看到，“示例项目”团队决定将后者作为最注重客户的选择。

3.4.3.1. 确定优先级的实践

“影响和努力优先级排序（矩阵）”实践**根据实施任务的影响和所需的努力进行评估。**在满足团队需求的情况下，团队可以选择实现速赢，而不是影响更大、需要付出更多努力的改进。另请参阅：影响和努力优先级（矩阵）。[34]

“优先级滑块”实践是一种工具，有助于就**相对优先事项**进行对话，以确定即将开展的活动的重点。另请参阅：优先级滑块。[35]

基于**“How-Now-Wow”**矩阵的实践在图形方面与上面的实践相似。它对**创新想法的分类**尤其有价值。另请参阅：How-Now-Wow 优先级排序（矩阵）。[36]

作为一种更贴近实际的**“Kano 模型”**实践，它是这里提到的四种实践中**最注重客户需求**的。下面让我们来详细介绍一下 Kano 模型。

东京理科大学名誉教授 Noriaki Kano 发明了这一简单的排序方法，将产品的基本属性与客户/利益相关者的差异化属性区分开来。根据 Kano 教授的观点，消费者的偏好可分为以下五类：

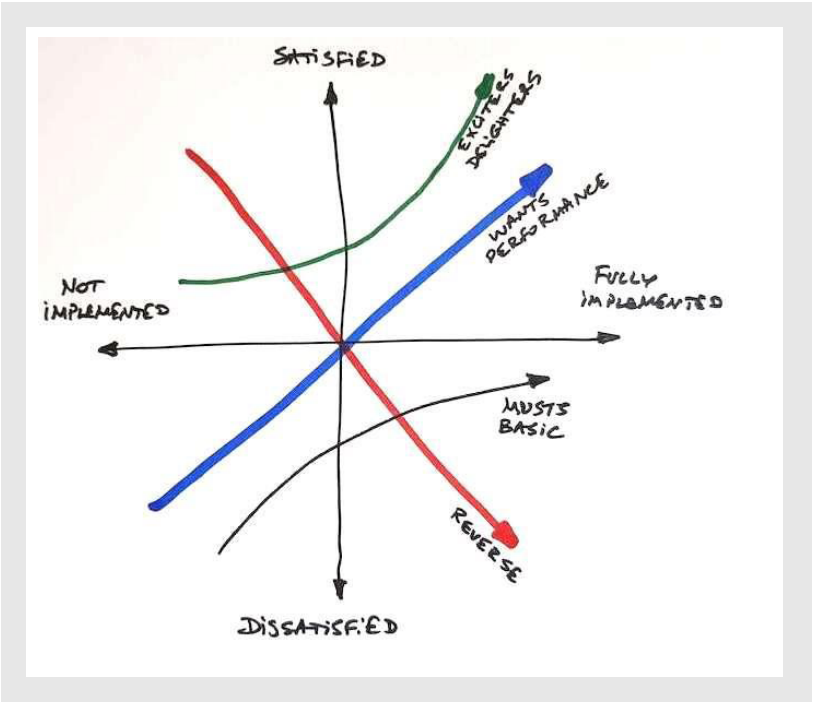
- **必备（又称必需品、基本需求，但缺失时会引起不满）** - 做得好时，客户持中立态度；不存在或做得不好时，客户会非常失望
- **一维质量（需求、满足因素、wow 因素）** - 存在则满意，不存在则不满意
- **魅力（又称取悦因素）** - 实现时令人满意，但不实现时不会引起不满
- **无差异（又称中性因素）** - 这些通常被忽视的功能需要花费精力去实现，但却白费力气；它们对客户来说并不重要。
- **反向** - 出现时实际上会导致一些客户感到不满，例如，不必要的过度复杂化。

[34] 开放实践库中的影响和努力优先级（矩阵）https://openpracticelibrary.com/practice/impact-effort-prioritization-matrix/
[35] 开放实践库中的优先级滑块 https://openpracticelibrary.com/practice/priority-sliders/
[36] 开放实践库中的 How-Now-Wow 优先级排序（矩阵）https://openpracticelibrary.com/practice/how-now-wow-prioritization-matrix/

因此，在规划产品时，以下是“基于 Kano”的一般建议：

- **包括所有必备**需求。这些是强制性的。
- **包含令人满意的“一维质量”（“需求”）和“魅力”（“取悦因素”）**功能，并将它们优化到刚刚好的程度。如果客户/利益相关者要求更多，也没关系；如果客户仍然认为产品有价值，您就不必立即优化这些功能。您可以在接下来的几次迭代中进行相应的优化/添加。

下面的“Kano 矩阵”可用于规划产品功能，并与项目团队、客户和利益相关者讨论功能的正确分类。然后，只需按照上述“基于 Kano”的建议，在下一次“交付迭代”中加入正确的功能即可。另请参阅：Kano 模型。[37]



3.4.3.2. 示例：定义的 MVP 范围，价值切片

“示例项目”团队决定采用“Kano 模型”实践，对迄今为止确定的初步功能进行优先排序/分类。他们喜欢这种真正以客户为中心的实践。哪些内容要纳入 MVP，哪些内容可以稍后实施？

3.4.3.2.1. 示例：应用 Kano 模型

特别是，研究小组决定采用 Kano 的标准化问卷，以隐性方式衡量参与者的意见。它应有助于以确定、“有限”的方式了解示例项目的利益相关者。因此，参与者需要针对每个产品功能回答两个问题，其中一个问题是以“积极”的方式提出的，另一个问题是以“消极”的方式提出的。

以下是针对“3.4.2.2. 示例：初始用户故事图”中概述的每个功能向每位参与者发放的调查问卷：

[37] 开放实践库中的 Kano 模型 https://openpracticelibrary.com/practice/kano-model/

	喜欢	理所当然	无所谓	勉强接受	不喜欢
“积极”问题					
如果产品有……，您会作何感想?					
如果有更多的……，您会作何感想?					
“消极”问题					
如果产品*没有*……，您会作何感想?					
如果……减少，您会作何感想?					

然后，主持人收集此类问卷，并根据 Kano 教授提出的下表计算参与者将该特征归入“Kano 类别”的“投票”：

“积极”问题		**“消极”问题**		**类别**
理所当然	+	不喜欢	→	必备
喜欢	+	不喜欢	→	一维质量（需求、满足因素）
喜欢	+	无所谓	→	有吸引力
无所谓	+	无所谓	→	无差异
不喜欢	+	理所当然	→	反向（引起不满）

例如，对于以下功能：

- “webhook”EDA 事件源通过推送到 GitHub 存储库的“主”分支来触发应用新版本的滚动去更新。

其中一位参与者填写的问卷如下：

	喜欢	理所当然	无所谓	勉强接受	不喜欢
“积极”问题					
如果产品有……，您会作何感想?	✓				
“消极”问题					
如果产品*没有*……，您会作何感想?					✓

因此，该参与者的投票计算如下：

“喜欢”（归入“积极”问题）**+（不喜欢）**（归入“消极”问题）**=> 一维质量（需求、满足因素）**– 存在则满意，不存在则不满意。

综上所述，问卷的相关参与者将这些特征分类如下：

史诗：滚动更新应用

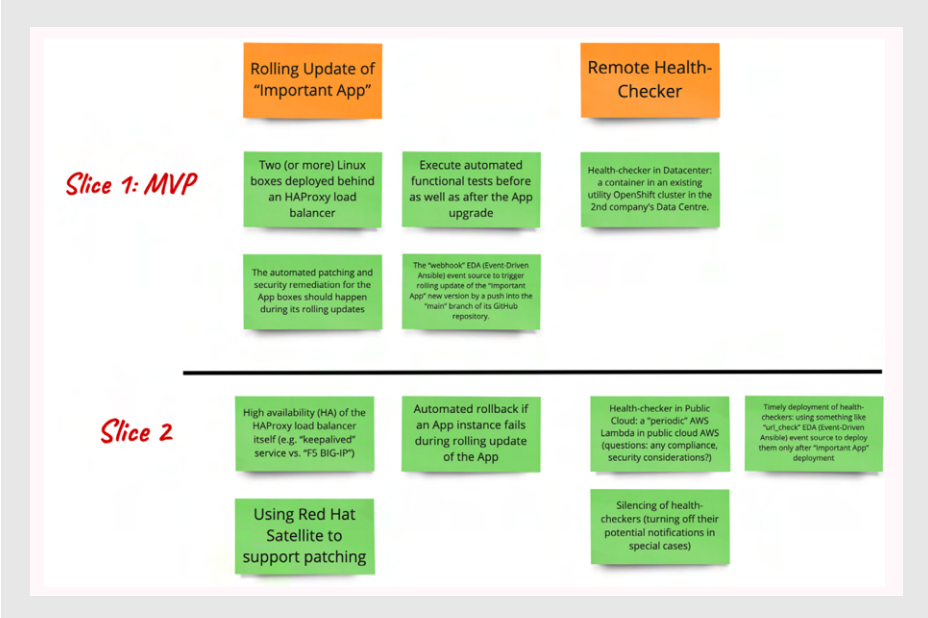
- **[必备]** 两个（或更多）Linux 服务器部署在 HAProxy 负载平衡器之后
- **[必备]** 在应用升级之前和之后执行自动化功能测试
- **[必备]** 应在应用滚动更新期间进行应用服务器的自动化修补和安全修复
- **[一维质量]** 使用 Satellite 支持修补
- **[一维质量]** 如果应用实例在滚动更新过程中出现故障，可自动回滚
- **[一维质量]** “webhook” EDA 事件源通过推送到 GitHub 存储库的“主”分支来触发应用新版本的滚动去更新
- **[魅力]** HAProxy 负载平衡器本身的 HA（例如，“keepalived”服务与“F5 BIG-IP”）

史诗：远程健康检查器

- **[必备]** 数据中心的健康检查器：第二家公司数据中心现有的红帽 OpenShift 集群中的一个容器
- **[无差异]** 公共云中的健康检查器：公共云 AWS 中的“定期”AWS Lambda（问题：是否有合规性、安全性方面的考虑？）
- **[魅力]** 及时部署健康检查器：使用类似“url_check”的 EDA 事件源，仅在应用部署后才部署健康检查器
- **[魅力]** 健康检查器的静默化（在特殊情况下关闭它们的潜在通知）

3.4.3.2.2. 示例：价值切片

下图展示了“示例项目”团队和相关的利益相关者如何切割出高价值，以形成增量发布策略。他们希望继续专注于交付有价值的成果。



4. 高效实现自动化

我们认为，有价值的工程解决方案应兼顾以下几个关键方面：

- 从业务角度看，解决方案对客户的价值
- 解决方案在生产环境中的运行状况
- 对解决方案的要求
- 解决方案的设计
- 开发和交付解决方案
- 自动化水平
- 安全防护和合规性
- 解决方案的成本

网站可靠性工程（SRE）等学科定义了一套原则和实践，有助于通过自动化管理 IT 解决方案，这比人工重复操作更具可扩展性和可持续性。SRE 特别适用于在 IT 解决方案的以下方面之间保持所需的平衡水平：

- 解决方案在生产环境中的运行状况
- 开发和交付解决方案

例如，SRE 建议如何管理系统可靠性风险。出现这种风险的原因可能是计划对 IT 解决方案本身进行修改，也可能是该解决方案的运行环境发生了某些变化。SRE 引入了一种特殊的“误差预算”指标，它消除了在决定允许多少风险时运维和开发人员之间的政治因素。另请参阅：拥抱风险。[38]

谈到风险，在原始项目计划可能突然改变或项目发生在高度创新或动态环境中的情况下，以下方面不平衡的风险相当高：

- 从业务角度看，解决方案对客户的价值
- 对解决方案的要求
- 解决方案的设计
- 开发和交付解决方案

遗憾的是，在这种情况下，从长远来看，以下交付方法可能并不奏效：“让我们按照原计划全面构建吧”。

对于项目团队来说，提高适应性和敏捷性对于此类项目的成功至关重要。根据我们的经验，大约 80% 的项目都是如此。因此，在下一节中，我们将在简要回顾本指南的背景之后，介绍一种“敏捷”的结构化方法。

4.1. “交付循环”和敏捷实践简介

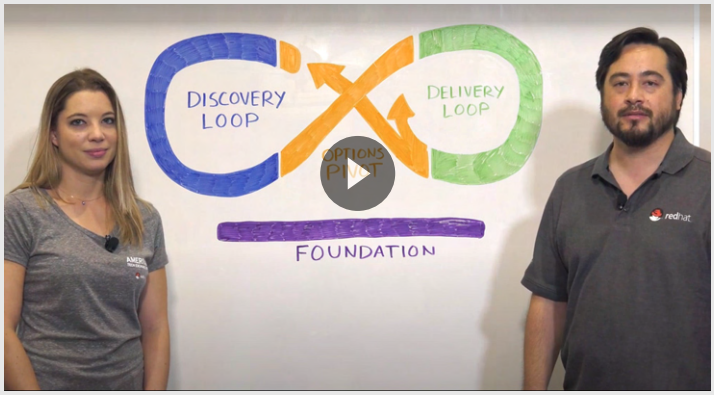
在前面内容较多的第 3 节中，我们介绍了以下实践：

- “发现循环”
  - 动机映射
  - 全局事件风暴
  - 讲述领域故事
  - MBPM
  - 用户故事映射
  - 目标成果

[38] 拥抱风险 https://sre.google/sre-book/embracing-risk/

- “方案转变”阶段
  - Kano 模型
  - 价值切片
  - MVP

在本节中，我们将重点介绍与“交付循环”相关的一些实践，这些实践在前面提到的同一视频的第二部分中进行了讲解，即“开放实践库的成果驱动型交付”视频：[39]



正如本第 4 大节的引言所述，我们通常希望项目团队保持“敏捷”，以取得项目的成功。随着时间的推移，敏捷实践者已经确定了几种常见的仪式/实践，可有效促进 IT 解决方案的敏捷开发和交付。以下是这些实践与“交付循环”的映射关系：

- 积压待办事项梳理（选项实践）
- “增量规划”或“冲刺规划”（交付实践）
- 每日站会（基础实践）
- 展示或演示（交付实践）
- 回顾（基础实践）
- “完成的定义”（基础实践）

有关这些实践的更多详情，请参阅以下章节。

4.1.1. 积压待办事项梳理（选项实践）

“积压待办事项梳理”是一项有时间限制的活动，在此期间，项目团队会与产品负责人/知情的利益相关者合作，审查当前增量（迭代）中的项目以及积压工作中的任何最优先项目。在积压待办事项梳理过程中，团队会明确每个交付项目的细节和验收标准。另请参阅：积压待办事项梳理。[40]

4.1.2. “增量规划”或“冲刺规划”（交付实践）

增量规划是一项有时间限制的活动，在此期间，项目团队与产品负责人/知情的利益相关者合作，承诺完成一系列已定义的工作，并为即将到来的增量（迭代）设定目标。另请参阅：迭代（冲刺）规划。[41]

[39] 开放实践库的成果驱动型交付 https://www.youtube.com/watch?v=N4mBIZg8MnQ
[40] 开放实践库中的积压待办事项梳理 https://openpracticelibrary.com/practice/backlog-refinement/
[41] 开放实践库中的迭代（冲刺）规划 https://openpracticelibrary.com/practice/iteration-planning/

4.1.3. 每日站会（基础实践）

每日站会是一项简短的活动，它为项目团队提供时间，使他们能够朝着商定的增量（迭代）目标同步努力。团队成员开会分享正在进行的工作的状态，并重点说明工作进展中的障碍。另请参阅：每日站会。[42]

4.1.4. 展示或演示（交付实践）

在展示活动中，项目利益相关者和有关各方将展示在增量（迭代）期间完成的近期工作，并收集反馈意见。展示（演示）也可以在产品/项目生命周期的关键里程碑上进行，或按需安排。另请参阅：展示。[43]

4.1.5. 回顾（基础实践）

在增量（迭代）结束时，项目团队使用“回顾”活动来反思、分析和调整他们的合作方式。回顾有助于揭示对项目团队绩效有影响的事实、观察和感受。这一切旨在提出、收集和商定相关项目团队的改进意见。另请参阅：回顾。[44]

4.1.6. “完成的定义”（基础实践）

“完成的定义”实践使整个项目团队和知情的利益相关者的理解和共同期望保持一致。例如，“完成的定义”可以商定如下：

- 根据验收标准测试用户故事
- 用户故事在交付时未发现任何错误
- 向利益相关者演示用户故事
- 产品负责人批准用户故事
- 已更新与用户故事相关的文档

另请参阅：“完成的定义”[45]

4.2. 示例：让我们根据首次迭代计划进行交付

为了进一步完善和规划活动，延续“方案转变”阶段的成果，“示例项目”团队和产品负责人决定相继采用以下两种实践：

- 积压待办事项梳理（选项实践）
- “增量规划”或“冲刺规划”（交付实践）

例如，他们注意到“示例项目”存在以下障碍（如第 3.1.3. 节“示例企业的示例项目：动机映射……”所述）

> 出于安全原因，为实施应用 HA 的工程师提供入职培训和改进修补流程将需要 2 到 4 周的时间。此外，项目团队可利用的区域数据中心的开发（非生产）环境计算能力目前也很稀缺。额外的计算能力可在 3 至 6 周内置备。

[42] 开放实践库中的“每日站会”https://openpracticelibrary.com/practice/daily-standup/
[43] 开放实践库中的“展示”https://openpracticelibrary.com/practice/showcase/
[44] 开放实践库中的“回顾”https://openpracticelibrary.com/practice/retrospectives/
[45] 开放实践库中的“完成的定义”https://openpracticelibrary.com/practice/definition-of-done/

因此，“示例项目”团队和产品负责人决定就以下额外行动项目达成一致：

> 在不影响数据中心内当前系统的情况下，找一个位置来置备支持 MVP 部署的环境。工程团队应能立即投入工作。

总体而言，他们已经完善了几项决定，并就第一次交付迭代的行动计划达成了一致。请在接下来的两节中查看。

4.2.1. 示例：迭代决策

MVP 商定如下：

- 在 HAProxy 负载平衡器后为应用安装两个 Linux 服务器
  - 这种计算资源的组织方式提高了计算能力
  - 它应该支持自动滚动更新新的应用版本
  - 它应该支持自动为两个 Linux 服务器进行修补和安全修复（在滚动更新期间）
  - 在 MVP 的范围内，我们将支持以 GitOps 风格启动滚动部署/更新 HAProxy 后的新版应用。特别是：
    - 我们将实施标准的“ansible.eda.webhook”事件源，通过 EDA 监听来自 GitHub 的应用存储库“推送”事件。一旦收到此类事件，EDA 应启动对新应用版本的自动化滚动更新。
    - 不支持回滚
- 在应用升级之前和之后执行自动化功能测试
- 类用户应用健康检查器：
  - 它是第二家公司的数据中心内现有实用程序红帽 OpenShift 集群中的一个容器。健康检查器应用将以 JavaScript/Typescript 实施。
  - 它应该及时部署，但目前还不能使用 EDA 提供的“url_check”事件源。

4.2.2. 示例：迭代行动计划

- 调查 HAProxy 负载平衡器配置
- 定义应用的功能测试。在 Ansible 中实施。
- 设计并实施 Ansible Playbook，以执行以下操作：
  - 自动滚动更新新的应用版本
  - 自动为两个 Linux 服务器进行修补和安全修复（在滚动更新期间）
  - 在应用升级之前和之后执行自动化功能测试
- 为 GitOps 式的应用滚动部署配置 EDA
  - 实施和部署 Rulebook。
    标准的“ansible.eda.webhook”事件源将通过 EDA 监听来自 GitHub 的应用存储库“推送”事件。一旦收到此类事件，EDA 应使用上述 Ansible Playbook 启动应用的自动滚动更新。
- 在不影响数据中心内当前系统的情况下，找一个位置来置备支持 MVP 部署的环境。工程团队应能立即投入工作。

  注： 为了满足本指南的需求，我们将在 AWS 公共云中部署用于 MVP 的 Linux 服务器。实际上，这也可能是“示例项目”团队做出的“现实生活中的决定”，当然，前提是要满足现实生活中的所有要求和规定。

  - 配置 AWS 以部署应用实例和 HAProxy 负载平衡器实例
  - 实施 Terraform 代码以部署类似数据中心的三个 EC2 Linux 实例

- 对于健康检查器应用：
  - 设计并实施健康检查器应用（使用 JavaScript/Typescript）
  - 研究与红帽 OpenShift 集成的 API

    注： “示例项目”团队计划部署到：
    “第二家公司的数据中心内现有一个实用程序红帽 OpenShift 集群”
    不过，该项目是针对一个假想的“示例企业”，但我们仍然想展示如何部署。让我们假设以下内容存在于第二家公司的数据中心：
    一个标准的“开发人员沙盒”红帽 OpenShift 环境
    另请参阅：开始免费探索开发人员沙盒。[46]
  - 在第二家公司的数据中心配置实用程序红帽 OpenShift 集群，用于部署健康检查器

    注： 在本指南中，实际上将：
    配置标准的“开发人员沙盒”红帽 OpenShift 环境

4.2.3. 示例：在迭代过程中

注： 在以下根据 MIT 许可证发布的个人存储库中，可以找到一个可能的 MVP 实施方案：
https://github.com/mikhailknyazev/automation-samples

在迭代过程中，“示例项目”团队按照以下行动计划进行了跟进。

- **调查 HAProxy 负载平衡器配置**
  有用：
  - Ansible 角色 - https://github.com/geerlingguy/ansible-role-haproxy
  - Packt 发布的《Ansible 为现实生活提供自动化》的代码存储库
    https://github.com/PacktPublishing/Ansible-for-Real-life-Automation
  - HAProxy 社区版 https://www.haproxy.org

```
---
- name: Deploy Load Balancer using HAProxy
  hosts: loadbalancer
  become: yes
  vars:
    haproxy_frontend_name: 'hafrontend'
    haproxy_backend_name: 'habackend'
    haproxy_backend_servers:
     - name: node1
       address: node1:80
     - name: node2
       address: node2:80
  tasks:

    - name: Install haproxy
      include_role:
        name: geerlingguy.haproxy
```

[46] 开始免费探索开发人员沙盒 https://developers.redhat.com/developer-sandbox

另请参阅：https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/haproxy-plus-health-checker.yaml

- **定义应用的功能测试。在 Ansible 中实施。**

```yaml
- name: Verify deployment
  hosts: "web"
  become: no
  tasks:
    - name: Verify application health
      delegate_to: localhost
      ansible.builtin.uri:
        url: http://{{ inventory_hostname }}
        status_code: 200
        return_content: true
      register: response
    - name: Check if 'Serving from...' is in the response
      delegate_to: localhost
      ansible.builtin.assert:
        that: "'Serving from {{ inventory_hostname }}' in response.content"
        fail_msg: "The phrase 'Serving from {{ inventory_hostname }}' was not found in the response"
```

另请参阅：https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/app-deploy.yaml

- **设计并实施 Ansible Playbook，以执行以下操作：**
  - **自动滚动更新新的应用版本**
  - **自动为两个 Linux 服务器进行修补和安全修复（在滚动更新期间）**
  - **在应用升级之前和之后执行自动化功能测试**

```yaml
---
- name: Rolling Update
  hosts: "web"
  become: yes
  serial: 1
  vars:
    haproxy_backend_name: 'habackend'
    application_repo: 'https://github.com/mikhailknyazev/automation-samples'
    application_branch: main
    subfolder_path: sample-app
    application_path: /var/www/html

  tasks:

    - name: Preparing rolling deployment of sample App
      ansible.builtin.debug:
        msg: >
          Branch: {{ application_branch }}
          Subfolder: {{ subfolder_path }}
          Repo: {{ application_repo }}
```

另请参阅：https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/rolling-update.yaml

- **为 GitOps 式的应用滚动部署配置 EDA**

```
yum update -y
yum install -y vim git wget java-17-amazon-corretto-devel sshpass

yum -y groupinstall "Development Tools"
yum -y install openssl-devel bzip2-devel libffi-devel

wget https://www.python.org/ftp/python/3.9.17/Python-3.9.17.tgz
tar xvf Python-3.9.17.tgz
cd Python-*/
./configure --enable-optimizations
make altinstall

cd ~
python3.9 --version
pip3.9 --version

/usr/local/bin/python3.9 -m pip install --upgrade pip
pip3.9 --version
pip3.9 install ansible-rulebook ansible ansible-runner wheel openshift

# Note: we are opening port 5000 for incoming webhook calls on port 5000
yum install -y firewalld
systemctl enable --now firewalld
firewall-cmd --add-port=5000/tcp --permanent
firewall-cmd --reload
firewall-cmd --list-ports

sudo -u devops /usr/local/bin/ansible-galaxy collection install community.general ansible.eda
```

另请参阅：https://github.com/mikhailknyazev/automation-samples/blob/main/sample-infra/user-data-ansible-engine-with-eda.sh

- **实施和部署 Rulebook。**
  标准的“ansible.eda.webhook”事件源将通过 EDA 监听来自 GitHub 的应用存储库“推送”事件。一旦收到此类事件，EDA 应使用上述 Ansible Playbook 启动应用的自动滚动更新。

```
---
- name: Listen for events on a webhook
  hosts: all

  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5000
      filters:
        - ansible.eda.dashes_to_underscores:

  rules:
    - name: Handle Webhook event
      condition: >
        event.payload is defined and event.meta.headers.X_GitHub_Event == "push"
        and event.payload.ref == vars.application_branch_in_webhook_event
        and event.payload.repository.url == vars.application_repo
#     condition: event.payload is defined

      action:
        run_playbook:
          name: rolling-update.yaml
          extra_vars:
            application_repo: "{{ application_repo }}"
            application_branch: "{{ application_branch }}"
            subfolder_path: "{{ subfolder_path }}"
```

另请参阅：https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/webhook-for-rolling-update.yaml

- **在不影响数据中心内当前系统的情况下，找一个位置来置备支持 MVP 部署的环境。工程团队应能立即投入工作。**
  - **配置 AWS 以部署应用实例和 HAProxy 负载平衡器实例**
  - **实施 Terraform 代码以部署类似数据中心的三个 EC2 Linux 实例**

有用：“使用 Terraform 在 AWS 中创建免费的 Ansible 实验室”https://www.techbeatly.com/use-terraform-to-create-a-free-ansible-lab-in-aws/

```
locals {
  connection = {
    type        = "ssh"
    user        = "ec2-user"
    private_key = file(pathexpand(var.ssh_pair_private_key))
  }
}

resource "aws_instance" "ansible-engine" {
  ami           = var.aws_ami_id
  instance_type = "t2.micro"
  key_name      = aws_key_pair.ec2loginkey.key_name
  security_groups = ["sample-mvp-sg", "sample-mvp-webhooks-sg"]
  user_data       = file("user-data-ansible-engine-with-eda.sh")

  // Copy the "get-automation-sample-playbooks.sh" script to the remote machine
  provisioner "file" {
    source      = "files-for-upload/get-automation-sample-playbooks.sh"
    destination = "/home/ec2-user/get-automation-sample-playbooks.sh"
    connection {
      type        = local.connection["type"]
      user        = local.connection["user"]
      private_key = local.connection["private_key"]
      host        = self.public_ip
    }
  }
```

另请参阅：https://github.com/mikhailknyazev/automation-samples/tree/main/sample-infra

- **对于健康检查器应用：**
  - **设计并实施健康检查器应用（使用 JavaScript/Typescript）**
  - **研究与红帽 OpenShift 集成的 API**
  - **配置标准的“开发人员沙盒”红帽 OpenShift 环境**

  有用：
  - 第 2 部分：将全栈 JavaScript 应用部署到红帽 OpenShift 的开发人员沙盒中 https://developers.redhat.com/developer-sandbox/activities/deploying-full-stack-javascript-applications-to-the-sandbox/part2
  - Gineesh Madapparambath 所著《Ansible 为现实生活提供自动化》中的“使用 Ansible 管理 Kubernetes”一章 https://amzn.asia/d/flHSDjh
  - “Ansible for Kubernetes 示例：使用 Ansible 实现 Kubernetes 集群自动化”，Luca Berton https://a.co/d/7tJWteG

```
- name: Display list of all Pods from the current Namespace (Project)
  kubernetes.core.k8s_info:
    kubeconfig: "{{ openshift.kubeconfig_file }}"
    kind: Pod
    namespace: "{{ openshift.namespace_name }}"
  register: pod_list
- name: Print pod_list
  debug:
    var: pod_list

- name: Deploy the Health-Checker with OpenShift feature "Source-to-Image" (S2I)
  ansible.builtin.command: >
    oc --kubeconfig={{ openshift.kubeconfig_file }} new-app
    https://github.com/mikhailknyazev/automation-samples --context-dir=sample-health-checker --name={{ health_checker_label }}
    -e HAPROXY_PUBLIC_IP={{ haproxy_public_ip }}
```

另请参阅：https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/health-checker.yaml

```
async function fetchData() : Promise<void>  {
  const haproxy_public_ip = document.getElementById('haproxy_public_ip').value;
  const response : Response  = await fetch(`http://${haproxy_public_ip}`, {
    method: 'GET',
    headers: {
      'Cache-Control': 'no-cache'
    }
  });
  const data : string  = await response.text();

  let lines : string[]  = data.split('\n');
  let resultIndex : number  = lines.findIndex(e : string  => e.includes("Serving from"));

  let str;
  if (resultIndex === -1) {
    str = "No matches found";
  } else {
    str = lines.slice(resultIndex).join('\n').replace(/<[^>]+>/g, ' ');
  }

  const listItem : HTMLLIElement  = document.createElement('li');
  listItem.textContent = str;
  document.getElementById('resultList').appendChild(listItem);
}

// Call fetchData every 5 seconds
setInterval(fetchData, 5000);
```

另请参阅：https://github.com/mikhailknyazev/automation-samples/blob/main/sample-health-checker/public/script.js

4.2.4. 示例：迭代的展示（演示）

注： 部分截图可在根据 MIT 许可证发布的个人存储库中找到：https://github.com/mikhailknyazev/automation-samples/blob/main/README.md

在第一次交付迭代结束时的展示（演示）中，“示例项目”的利益相关者和有关各方都观看了在增量（迭代）期间所开展的近期工作的演示，并给出了反馈意见。总体而言，他们对与 MVP 目标相关的结果印象深刻：“对应用部署、修补、可观测性和性能进行改进”。

特别是：

- 利益相关者估计，一些演示的自动化技术可以在整个企业范围内重复使用，并随着时间的推移将手动 IT 工作减少 50-80%
- 应用部署现已完全实现自动化。因此，基础架构团队可以切实计划将工作方式转变为高效的“站点可靠性工程师”模式。
- 利益相关者对于与负载平衡相关的架构变更表示赞赏，因为现在可以更轻松地根据需要增加应用的计算能力
- 演示的“GitOps 与 Ansible”技术应能提高应用的开发效率，并改善任务关键型 Linux 工作负载的变更可追溯性
- “健康检查器”远程组件提高了应用的可观测性，并最终提高了向用户提供服务的稳定性。利益相关者对于使用红帽 OpenShift 的源至镜像（S2I）功能来实现项目目标的简便性感到印象深刻。

谈到 MVP 可以做得更好的方面，利益相关者提到，如果有更完善的修补和安全修复演示，他们会受益匪浅。不过，他们很高兴地指出，通过展示的在应用滚动更新过程中更新 Linux 软件包，现在对其进行改进比以前容易得多。

5. 红帽为客户提供了哪些支持?

免责声明：本部分内容仅代表本指南作者的个人观点。文中信息均“按原样”提供，不作任何保证。这些信息在您阅读时可能已经过时。作者对与这些信息相关的任何内容概不负责。作者认为，它有助于支持使用其他相关的红帽资源。

本节介绍与本指南上下文相关的一些红帽商业产品和服务。

5.1. 红帽开放创新实验室助您更自信地展开自动化或容器采用之旅

红帽开放创新实验室可以帮助企业开启自动化或容器采用之旅！客户的员工将了解如何以开源方式打造出色产品。红帽的沉浸式驻留培训使工程师与开源专家共事，将客户的想法转化为业务成果。

在为期 4-12 周的开放创新实验室驻留培训期间，他们将了解如何使团队的想法与开源社区提供的最佳实践和工具联系起来。红帽专家在红帽技术、开源社区以及释放客户团队潜力所需的关键变革实践方面拥有丰富的经验。[47]

世界卫生组织（WHO）以虚拟方式举行了一次为期 8 周的开放创新实验室驻留培训，旨在创建一个 DevOps 平台。这里有一段简短的视频总结。[48] 此外，还值得查看以下引人入胜的演示，了解是哪些专业人员推动了参与：Donna Benjamin 讲解的“将故事转化为软件”[49]。

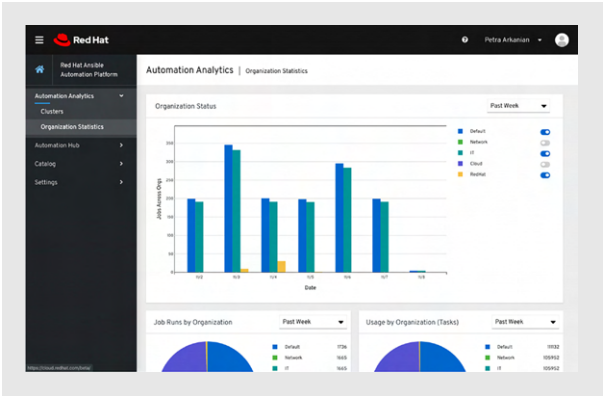
5.2. 有大量移动部件？考虑使用红帽 Ansible 自动化平台

红帽 Ansible 自动化平台[50] 是一款可帮助您管理多层架构、异构 IT 环境和用户权限的产品。它专为企业级 IT 管理而设计，提供了便捷的 UI 仪表板，其中显示所有托管主机的摘要，并可轻松导航到各种自动化功能。它满足了企业的灵活监管和安全防护需求。AAP 可高度嵌入到现有的标准方法中，例如使用 Active Directory 进行身份验证和授权。它为 IT 团队提供了对 Ansible Playbook 执行的最佳可观测性。

例如，凭借 Ansible 自动化平台，您可以：

- 通过基于角色的访问控制和变更请求管理来确保自动化安全。
- 通过整合到一个灵活的平台上，消除孤立的团队和流程
- 在任何位置扩展自动化 - 本地、云端或网络边缘

[47] 红帽开放创新实验室 https://www.redhat.com/zh/services/consulting/open-innovation-labs
[48] WHO 打造：共同创建一个创新型学习平台 https://www.youtube.com/watch?v=f69l6Vo9rGk
[49] 将故事转化为软件 https://www.youtube.com/watch?v=J3VaaPYshek
[50] 红帽 Ansible 自动化平台 https://www.redhat.com/zh/technologies/management/ansible



AAP 的工作流支持将任意数量的 playbook 关联起来，而无需考虑使用不同的库存、使用不同的凭据或在不同的用户下运行。另请参阅：第 23 章工作流。[51]

AAP 提供安全的 REST API 和相应的 CLI 工具。这使得与 ServiceNow 等 ITSM 工具和 Jenkins 等 CI 系统的集成成为一项简单的任务。例如，您可以通过 API 或 CLI 启动自动化作业（托管 playbook）。

AAP 增加了对机器和云系统所有凭证的安全存储，以及一个强大的基于角色的访问控制引擎，让您可以轻松设置有关谁可以在什么环境中运行什么自动化功能的策略，确保只有合适的人员才能访问机器和应用配置。AAP 本身是一个分布式系统，为了使其成为一个高可用性平台，在工程设计上花费了很多心思。

AAP 正在积极开发中，例如，根据“Ansible 自动化平台 2.4 新增功能”[52]，AAP 新增/修订了以下内容：

EDA；集合存储库管理；经验证的内容集成；Ansible Builder 3.0；对 ARM 的平台安装支持；搭载了 IBM Watson Code Assistant 的 Ansible Lightspeed 的技术预览现已推出。

6. 结论

感谢您对 Ansible 助力实现高效自动化的关注！

在这份基于案例研究的指南的开头，我们描述了为什么 Ansible 是 DevOps 工具的好搭档，并且可以作为 IT 工件的通用“粘合剂”。我们介绍了一些企业层面的自动化想法，以及自动化如何帮助完成日常重复性任务。

在指南的主要部分，我们解释并演示了“发现循环”，“方案转变”和“交付循环”等实践如何帮助自动化项目取得成功。最后，我们介绍了红帽提供的一些相关产品。

希望这对您有所帮助！

[51] 章节“23. 工作流” https://docs.ansible.com/automation-controller/latest/html/userguide/workflows.html
[52] Ansible 自动化平台 2.4 新增功能 https://www.ansible.com/blog/whats-new-in-ansible-automation-platform-2.4

7. 作者简介

Michael Knyazev
红帽

Michael Knyazev 是一名经验丰富的实践架构师/DevOps 顾问，擅长 OpenShift、Ansible、AWS、GCP 和 Kafka。

Michael 在为亚太地区的主要红帽客户提供有价值的成果方面发挥了核心作用。在加入红帽之前，他曾参与设计并构建了多个一流的解决方案。其中包括为 TPG Telecom 和 Insurance Australia Group 开发的基于 Kubernetes 的平台、澳大利亚国家冠状病毒帮助热线云解决方案、Morningstar Tick Data 和 Beam Wallet FinTech 平台。

Michael 乐于为社区做贡献！他是红帽公司《Ansible 助力实现高效自动化》电子书的主要撰稿人，开设了热门的 Udemy 课程《配置 Kubernetes 以提高可靠性》，并在 2006 年答辩的关于 IT 架构自动化开发的博士论文中取得了科研成果。

Michael 热衷于学习，他还拥有 MBA 以及超过 15 项专业认证。近年来，他曾在全球混沌工程大会和 JavaOne 大会上发言。

他与家人住在澳大利亚悉尼。

Gineesh Madapparambath
红帽

Gineesh Madapparambath 是红帽新加坡公司的平台与 DevOps 顾问。在他丰富的职业生涯中，他担任过从系统工程师到自动化专家和内容作者等各种角色，主要侧重于 Ansible 自动化、容器化（OpenShift 和 Kubernetes）和基础架构即代码（Terraform）。Gineesh 是《Ansible 为现实生活提供自动化》一书的作者。他在设计、开发和部署自动化解决方案方面拥有丰富的经验，主要使用 Ansible 和 Ansible 自动化平台（前身为 Ansible Tower）。这些解决方案包括各种任务，如构建裸机/虚拟服务器、修补、许可证管理、网络运维和自定义监控。此外，他还协调过世界各地数据中心服务器的设计和部署，在经典、私有云（OpenStack、VMware）、虚拟和公共云环境（AWS、Azure、Google Cloud）方面积累了宝贵的跨文化经验。在整个职业生涯中，Gineesh 曾担任过多种职务，包括系统工程师、自动化专家、基础架构设计师和内容作者。

Nicholas Wong
红帽

Nicholas Wong 拥有一股向善的力量，“促进社会进步”是他的人生使命。他认真履行职责，以指导者、教练和导师的身份与个人、团队和企业合作。他努力学习关键技能，提高自己与他人建立有意义联系的能力，从而随时随地营造美好的人际关系。

作为一名高级交付顾问、价值流专家、资深教练和多个领域的认证讲师，他的经验源自基层工作，曾担任质量评估顾问、系统管理员和技术交付负责人。

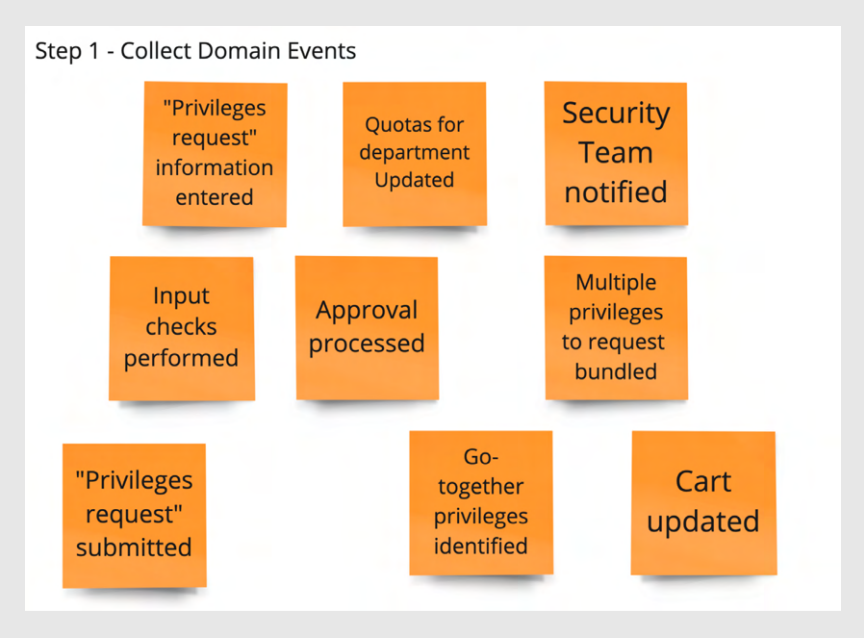
Nicholas 的大部分时间都在帮助他人实现最佳自我，面对新的/现有挑战，以及实现他们的目标。他非常了解学员的需求，并据此为他们量身定制课程。工作之余，他经常为高管和优秀员工提供一对一辅导。

Nicholas 还活跃于 DevOps、敏捷辅导和企业/个人辅导社区。他坚信，知识和专业技能应该惠及所有人，并喜欢与各行各业的领导者合作，带领高度参与、积极主动和自主的团队走向未来。

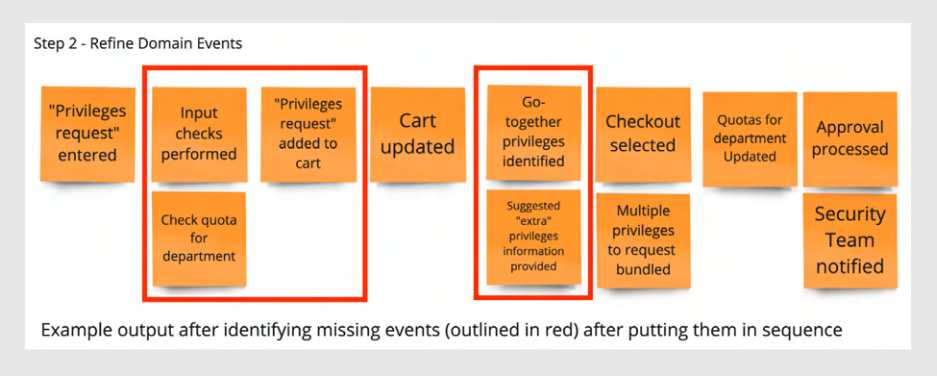
“以人为本，提升成果，改善生活”~ Nicholas Wong

附录 A. 针对“使用中的应用 – 请求访问权限”的事件风暴

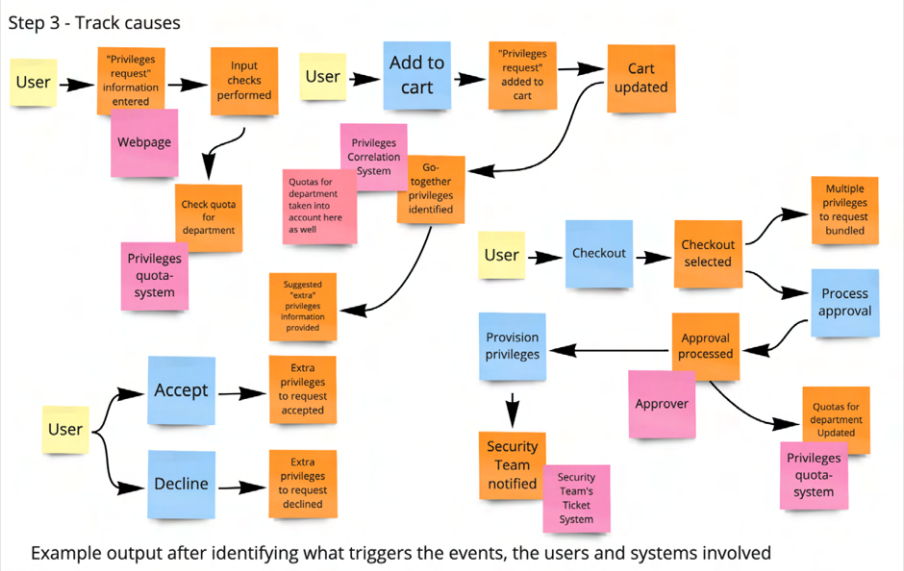
在“步骤 1 - 收集领域事件（全局）”中，每位参与者仅使用橙色便利贴。



在“步骤 2 - 完善领域事件（全局）”中，与参与者一起查看领域事件便利贴。让参与者解释每个事件的含义。检查语法是否正确。此外，还需再次讨论事件是否按时间顺序正确排列。统一出现的同义词（描述相同事物的不同术语）；如果同一术语用于描述不同的事物，则要突出差异。

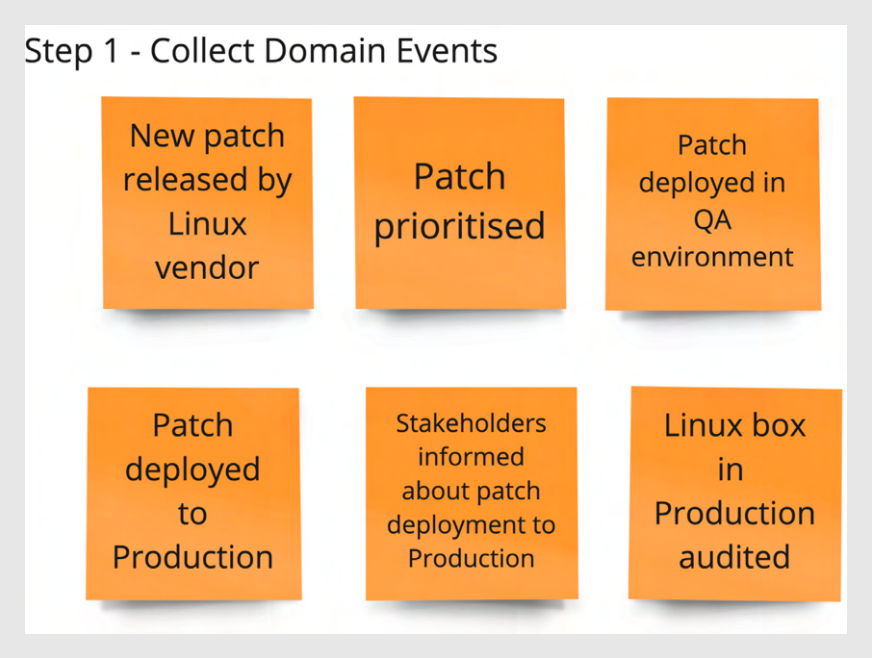


在“步骤 3 - 跟踪原因（流程建模）”中，进行原因分析。领域事件从何而来？

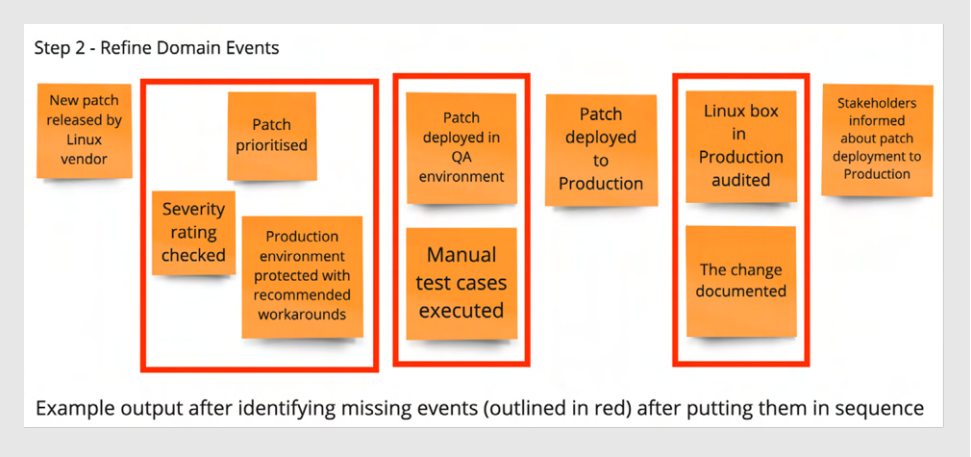

Step 3 - Track causes
User
"Privileges request" information entered
Input checks performed
Webpage
Check quota for department
Privileges quota-system
User
Add to cart
"Privileges request" added to cart
Cart updated
Privileges Correlation System
Quotas for department taken into account here as well
Go-together privileges identified
Suggested "extra" privileges information provided
User
Checkout
Checkout selected
Multiple privileges to request bundled
Process approval
Approval processed
Provision privileges
Approver
Security Team notified
Security Team's Ticket System
Quotas for department Updated
Privileges quota-system
User
Accept
Extra privileges to request accepted
Decline
Extra privileges to request declined
Example output after identifying what triggers the events, the users and systems involved

附录 B. 针对“修补应用的 Linux 服务器”的事件风暴

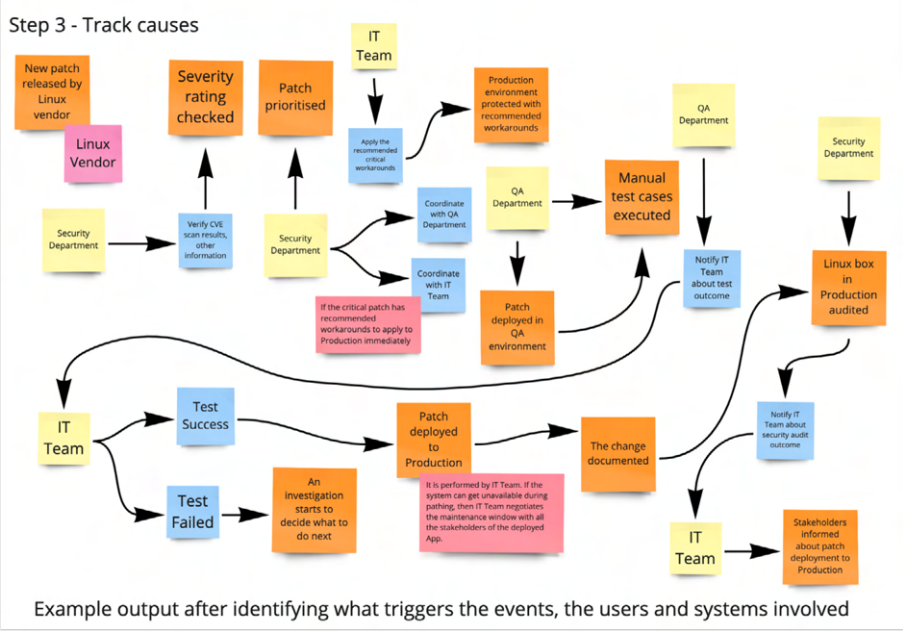
在“步骤 1 - 收集领域事件（全局）”中，每位参与者仅使用橙色便利贴。



在“步骤 2 - 完善领域事件（全局）”中，与参与者一起查看领域事件便利贴。让参与者解释每个事件的含义。检查语法是否正确。此外，还需再次讨论事件是否按时间顺序正确排列。统一出现的同义词（描述相同事物的不同术语）；如果同一术语用于描述不同的事物，则要突出差异。



在“步骤 3 - 跟踪原因（流程建模）”中，进行原因分析。领域事件从何而来？



Example output after identifying what triggers the events, the users and systems involved