电子书

Red Hat

云原生遇到混合云：策略指南

构建应用环境，实现可靠性、生产力和变革

Red Hat

目录

红帽官方微博　红帽官方微信

简介

信息技术（IT）系统是将现代企业紧密联结在一起的粘合剂。它们通常是企业重要的市场优势，会为内部系统提供动力，并且要为客户、员工和合作伙伴提供不断改进的功能。同时，系统的复杂性和变革的速度也在逐渐提高：部署工作通常要跨越多个本地和云位置，使用一系列软件技术，而且这些系统背后的团队也存在很大的技能差异。人们越来越关注数字化转型，这无疑凸显出变革能够带来的商业优势，但光是转型，往往无法解决如何利用新技术发挥真正价值这一问题。

这些重重压力，造成了环境的可靠性与生产力之间的根本性矛盾。一方面，系统需要在运行中确保功能正常、安全和可预测。另一方面，开发人员和运维团队需要能够快速推动系统发展，以便为用户提供新的功能优势，让整个企业都能从系统进步中受益。

可靠性与生产力之间的这种主要矛盾会影响到许多选择，而在当今快速变革和改进的压力下，上述问题变得尤为棘手。IT 团队会需要不断平衡应用交付与开发方面的短期和长期变革。例如，他们要权衡是使用现有系统和基础架构来开发新功能，还是重新设计开发与交付功能。长期目标则是在整个公司应用环境中实现不断增值，同时改进敏捷开发与交付。

由于 IT 系统广泛分布在数据中心位置和公共云中，而大多数企业都采用了混合云 IT，因此应对好这些压力意义重大。此外，云原生及其他领域的无数新的开发技术也为开发团队提高生产力开辟了大量新的机会。

本指南可作为企业架构师和 IT 领导者的策略入门教程，帮助指导他们规划如何应对现代 IT 策略问题。本指南重点关注以下三种趋势的融合：

1. IT 系统对现代企业成功的重要性。
2. 混合云以及向多个数据中心、平台和云位置的转变。
3. 云原生应用开发技术及其如何与现有方法结合，从而打造出强大、高效的 IT 系统。

指南的内容涵盖了不同情境的注意事项、策略挑战、总体架构注意事项以及案例研究示例。

本电子书：

- 主要关注策略方面和总体架构，尤其是可以在各种情境中应用的总体概念。它讨论的并非具体详细的部署注意事项。本书通篇也提供了进行深入技术分析所需的资源。
- 不仅仅假定绿地开发。有些团队可能有条件从头开始构建新系统，但我们的大多数建议都适用于从现有的复杂系统演进。
- 具有普遍的技术适用性。本书探讨了不同领域的红帽® 客户和技术示例。尽管我们认为红帽工具无疑最适合相应的工作，但本电子书中讨论的大多数方法在其他技术环境（或者红帽技术与其他技术结合使用的环境）中也可以应用。

您也可以在以下电子书中找到其他辅助内容：《教大象跳舞》《云原生应用的构建之路》《基于容器的应用设计原则》，此外，redhat.com 的云原生资源页面也提供各种其他内容。

“像我们的许多客户一样，我们也面临过一些实际问题，比如‘我们如何才能做得更好，而不仅仅是跟上发展？’，以及‘我们该怎么做才能为一切处境做好充分准备？’”

Mike Kelly
红帽首席信息官

第一部分：现代化 IT 所面临的挑战

2017 年初夏，新任命的红帽首席信息官 (CIO) Mike Kelly 面临了一个艰难的局面。红帽拥有可靠的的 IT 基础架构，已经有了巨大的发展，但更大规模的增长正在逼近。红帽的快速增长意味着 IT 团队需要为增长形势做好准备，同时仍然要在有限的预算内保持正常运营。

Mike 和他的团队是如何应对这些挑战的？

红帽 IT 转型

IT 系统相当于企业的中枢神经系统，它要负责传递信息、协调行动，并且执行许多让组织保持正常运转的任务。近年来，对于大多数企业来说，这种应用系统的规模、密度和复杂性都大大的增加。红帽也不例外。

截止到 2016 年，红帽已运行了近 1,000 个独有的独立应用和服务，它们各司其职，分别用于业务的方方面面。这些应用由不同的团队运行，采用不同的技术堆栈，并跨越了多个冗余数据中心位置。团队的目标是解决三个部分的挑战：

1. 满足数字化业务对速度和适应性的需求。
2. 改进数字化系统的可用性、弹性和安全性。
3. 继续降低运维成本。

红帽创新实验室是一项沉浸式合作训练计划，旨在帮助客户加快实现其最具创新性的想法。[1]

这些目标展现了典型的两难境地：一方面是对速度、适应性、可用性和安全性的需求，但另一方面还要降低运维费用。

支持红帽日常业务的应用既有自定义 JavaTM 和其他应用，也有软件即服务 (SaaS) 和数据存储的集成，还有复杂的工作流程。

项目就这样开始了

首先，红帽 IT 与红帽开放创新实验室团队合作，对红帽客户进行了例行的全面分析。红帽开放创新实验室计划提供了沉浸式训练，可通过将技术和流程的变革结合起来帮助企业应对实际的业务挑战。这让红帽 IT 全面了解了他们所面临的选择以及一些最紧迫问题的可选解决方案。

最终的计划涵盖了红帽 IT 流程中各个领域的变革，既有即时性的，也有渐进式的：

- 从多位置、本地的故障转移式基础架构转变到混合云基础架构，包括首先采用一个公共云区域，然后采用来自不同提供商的两个公共云区域。
- 迁移至基于容器的云原生部署和应用开发模式，并以容器作为部署的主要单元。
- 在软件开发技术和方法进行重大转变，以便可根据每个应用的需求使用多种编程语言。
- 借助容器大幅节省成本，并能够从父代-子代故障转移设置迁移到三个活动站点之间的故障转移，从而提高可靠性并加快更新速度。
- 对一系列应用进行升级，并利用提高应用编程接口 (API) 的采用率和实现敏捷集成来对这些应用进行连接。

1 红帽开放创新实验室，2019 年。redhat.com/zh/services/consulting/open-innovation-labs。

尽管超过 50% 的企业已经实施了一定水平的 DevOps 实践[3]，但最坚定的（企业）采用者获得了高得多的绩效提升。DORA 的 2018 年 DevOps 状态调查显示在采用后通常会先导致绩效放缓，然后才能带来提升。

可观测、可衡量的成果

到 2019 年初，该计划已超出了其最初的转型目标（见图 1）。转型之旅尚未结束，而随着新挑战和新机遇的出现，团队仍不断发展着红帽的 IT 环境。而借助策略，这个团队得以为公司打造一个灵活、可靠的应用环境。他们使每个应用所占用的空间减少了 55%，安全性得到了提升，并且多站点混合云部署实现了零停机。流程交付的自动化使得功能交付的上市时间缩短了 40%。[2]

图 1. 红帽 IT 在云原生混合云转型中的收获（源自红帽 IT 内部数据）

现在，IT 团队成员（以及公司中的任何人）可以采用多种方式来部署新应用，并与他人共享访问权限。生产系统的基础架构不仅具有弹性，而且成本更低了。

此外，团队还学会了应对变革过程中难免的内部文化挑战。事实证明，不仅要重视热情的技术采用者的意见，还要重视团队中持怀疑态度者的意见，二者同等重要。

进一步了解红帽 IT 转型。

求快还是求稳？

正如红帽 IT 案例中所示，新技术可以为应对现代 IT 挑战带来切实的好处，但应用起来还需要经过三思和协调。在大型企业中，关于如何应对短期和长期变革的问题尤为突出，因为开发团队需要快速行动并采用效率更高的全新技术，而运维团队则要负责确保可靠性和稳定性。

DevOps 运动正是企业内部不同部门的这种冲突的体现。但是，即便在 DevOps 的采用上，每个企业通常也会有不同的做法。而技术选择多样化所带来的重压趋势，进一步加剧了这种挑战：

- 绝大多数企业（根据最近的 Flexera/Rightscale 报告，比例为 84%[4]）都在推行多云策略。很多情况下，企业已经本身就处于混合云状态，原因很简单：不同的部门采用不同的云提供商。
- 无论是在一般应用领域还是在企业中，使用的编程语言都在持续激增。尽管 Java 和 Javascript 在企业中仍占据主导地位，但在一些特定应用领域，其他语言却越来越受欢迎。（有关示例，请参阅 Red Monk 的 2019 年公共数据调查。[5]）

2 红帽 IT 内部数据

3 Stroud, Robert：“2018：企业 DevOps 年”，Forrester 博客，2017 年 10 月 17 日，https://go.forrester.com/blogs/2018-the-year-of-enterprise-devops/。

4 Flextera：“RightScale 2019 年度云现状报告”，2019 年，https://info.flexerasoftware.com/SLO-WP-State-of-the-Cloud-2019。

5 O'Grady, Stephen：“RedMonk 编程语言排名：2019 年 1 月”，Red Monk 博客，2019 年 3 月 20 日，https://redmonk.com/sogrady/2019/03/20/language-rankings-1-19/。

- 云原生技术的采用，尤其是向“容器即基础架构”和“容器管理即基础架构”的迁移已呈现加速态势。[6]

现在的挑战是如何平衡这些压力。

创建并发展有效的混合云应用环境

我们使用“应用环境”一词来指代企业用于进行应用交付和开发的一系列能力。我们有意采用了这一广义的视角，因为它是现代化 IT 挑战的整体基础。具体来说，传统应用需要与新应用协同工作，新技术需要在不破坏可靠性的情况下实现增值，同时 IT 系统可见于更多的位置。

随着新应用的部署、工具和流程的更新以及应用生成新的数据和事务，企业的应用环境会不断发生变化。最终，那些得到环境支持的生产应用将为企业的员工、合作伙伴和用户带来增值。

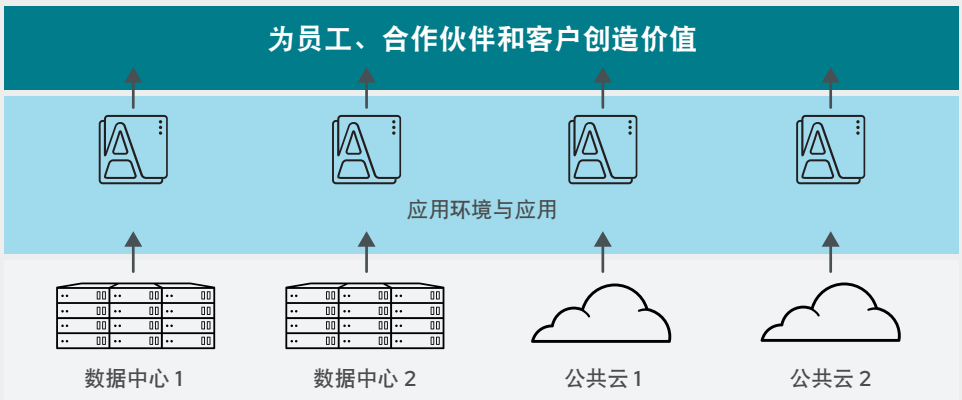


图 2. 应用环境支持应用交付价值

我们可以将企业的应用环境形象地比喻为整个企业范围内一个单一的、不断发展的生态系统，因为我们要聚焦于环境需要展现的特性与制定各个决策时所造成的紧张关系。

后面几节中所提到的许多折衷方案（包括可靠性与生产力、可预测性与变革能力、安全性与便利性、性能与成本以及刚性与自由度）都看似一种非此即彼的主张。做出这些选择似乎需要相互对立。尽管需要权衡取舍，但要想让企业脱颖而出，IT 部门必须做到二者兼顾：

- 可靠性与生产力。
- 可预测性与变革能力。
- 高性能和低成本。

凭借慎重的策略规划和重点明确的决策，这些需求中有很多都是可以满足的。

6 Hippold, Sarah：“Gartner 明确了 2019 年 PaaS 和平台架构的主要趋势”，Gartner 新闻稿，2019 年 4 月 29 日，https://www.gartner.com/en/newsroom/press-releases/2019-04-29-gartner-identifies-key-trends-in-paas-and-platform-ar

第二部分：策略入门

制定战略框架有许多不同的方法。在本文中，我们根据 Richard Rumelt 的《好战略，坏战略》一书，采用了一种简单的三步式框架。[7] 这三个步骤从总体上概述了构成长期策略的决策类型。许多红帽内部团队都使用这一框架的某个版本来制定规划。

基于此模型，策略讨论被分为三个方面：

1. **评估情况并设定目标。**掌握现状、挑战和出现的机遇以及对行动的重大制约。一般情况下，制定策略会同时包含一个长期议题（“我们希望在 5-10 年后变成什么样？”）和一个短期观点（“未来 12-18 个月会发生什么？”）。
2. **制定政策和准则。**掌握用于确定策略总体方向、适用范围的广义原则和参与规则，以及特定事务的一般行事惯例。政策和准则持续时间通常为中期：至少是一年，但理想是四到五年。
3. **确定下一步该做什么。**提出一系列要立即采取的具体措施，使企业更接近目标，并且更符合既定的政策和准则。

在第二部分中，我们将介绍这一结构的前两个方面。第三个方面将在第三部分（架构）和第四部分（从何处入手）中进行深入介绍。

正如后续章节所述，要想取得成功，一个反复出现的模式就是利用权力分散。因此，要想发挥最佳效果，可能任何策略都应该具有多层次结构，在企业层面上采用广义的政策和准则，同时还为各个部门留出余地，来根据需求添加自己的本地实例。

情况评估和目标

在获取对企业 IT 环境的情况评估时，需要考虑很多因素，包括现有政策、人员和预计的使用情况。尽管关键事项会因每个特定的企业而异，但在较高的抽象级别，有些关于整个应用环境的因素是必须要考虑的。

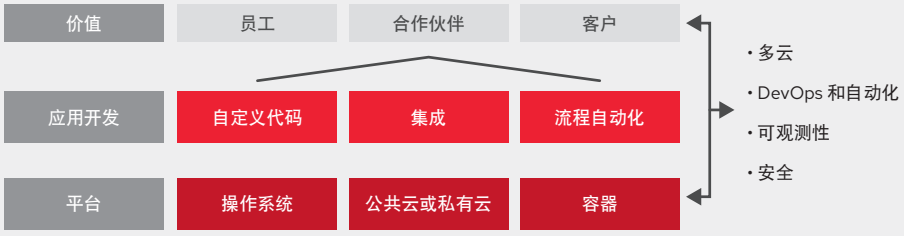


图 3. 简化的 IT 环境视图

具体来说，每个企业都可能具有：

- **一组内部数据中心——通过一个或多个私有托管服务托管的基础架构，以及公共云提供商。**这些平台通常分布在不同的地理区域，从而方便实现故障转移或数据保护。各区域通常包含一种或多种操作系统、虚拟化系统、私有云部署或容器化基础架构。

7 Rumelt, Richard P:《好战略，坏战略》，纽约：Crown Business，2011 年。

- **一组应用和应用开发解决方案。**从应用的角度看，该组合中很可能包含由企业或第三方顾问编写的自定义代码、商用现成（COTS）解决方案、SaaS 解决方案以及一系列其他应用。此外，企业可能还会使用应用服务器或其他技术来托管自定义代码、集成解决方案、消息传递和流程管理，以便将各个应用连接在一起。
- **开发人员工具、流程、自动化和管理功能**（用于管理基础架构的各个部分）。
- **众多利益相关者，**他们依赖于应用环境来提高生产力，包括构建与运维应用的开发人员和运维团队，以及使用相应应用的员工、客户和合作伙伴。

勾勒出这些关键要素非常有用，因为它可以让人了解整个应用环境的深度和广度。

在目标方面，我们将专注于两个方面的目标，它们几乎具有普遍意义：可靠性和生产力。

但您的企业可能还有其他一些特定的方面需要优先考虑。我们希望随后的一些指导也可以解决其他方面的常见目标，但我们首先还是将重点放在可靠性和生产力上，因为它们是企业中最普遍的课题。

我们所说的“生产力”是什么意思

在本文中，生产力指的是整个企业及其所有员工。整个企业能完成多少工作量？其实我们主要关注的是 IT、开发与运维团队的生产力，他们负责改进提供给最终用户的应用。但引申开来，这些应用的改进对于企业其他部门提高生产力也发挥着重要的作用。正如我们将在第三部分中看到的那样，开发人员与用户之间的界限在某些方面已变得模糊。

设定明确的目标

我们建议全面考虑以下三种目标：

1. **一般性结果。**我们希望应用环境的哪些方面获得改进？
2. **常见模式和反模式。**我们想复制哪些成功的应用或团队范例？有哪些失败的案例是要避免的？
3. **特定项目或计划中的收获。**哪些特定活动（例如新增功能、申请新技术以及淘汰旧功能）正在进行中，但调整方向也会带来好处？

政策和准则的考量

一旦设定了目标，下一步挑战就是建立能够帮助实现目标的流程。通常，要达到最佳效果，应建立一套清晰明确、人人尽知的政策和准则，并受到企业内不同部门的认同和遵守。

每个企业的确切政策组合会有所不同。在本节中，我们将介绍政策方面的多个注意事项，根据我们的观察，它们在许多大型 IT 环境中是行之有效的。这些内容既有来自我们与全球众多客户合作的经验总结，也是我们对在现场实践中证明有用的主流观点的一种改良汇编。

这些政策领域适用于组织层面。尽管它们借鉴了一些现有的云原生和混合云策略建议（有关示例，请参阅《教大象跳舞》）以及许多关于架构或微服务的 Martin Fowler 的博客文章，其内容会更加精简。后续章节还将提供有关实施的技术建议。

“如果一个大决定能做到，就不要做 1000 个小决定。好的‘指导政策’会往特定的方向引导行动，而不会确切定义应采取的措施。”

Richard Rumelt
作者，改编自《好战略，坏战略》[8]

权力分散

系统设计中的分散化趋势已经流行了很多年，就技术而言，它最主要体现在微服务等方法中。这种架构对许多类型的系统都有意义，也是云原生技术取得重大突破的领域之一。

在考虑实施分散化之前，还有一个层面也可以让它发挥重要作用。

分散式系统并不适用于 IT 系统的每个部分，但从组织结构角度讲，分散化几乎始终是有好处的。在团队之间划定责任界限，要比划定代码界限更为重要。这种方法似乎是对康威定律的直接运用；但是，由于企业太急于实施分散化应用开发模式（例如微服务），实践中却频频出错。

我们将这一政策原则表示为：**权力集中 -> 权力分散**。

赋能而非抓权

如果集中化 IT 能向整个企业中的其他团队进行授权并分摊开发和运维责任，其效果会比权力集中更好。

这并不意味着中央团队可以移交所有职责。他们必须要求其他团队遵循一些关键的流程和系统。但是，通过让其他人有机会承担责任，相应部门就可以有更多机会参与整个企业的运作。由此带来的好处包括：

“领导力是将成就伟大的能力融入到组织的人员和实践中，并将其与领导者的个性脱钩。”

David Marquet
美国海军潜艇司令 [9]

- 更高的可靠性和更好的最佳实践分享（较小的团队可能会先于中央团队发现问题）。
- 个人对营运成果投入更多。
- 对于哪些重要，哪些不重要，监督重点更明确。

生产力方面的提高包括：

- 当地团队能更好地根据其环境快速调整有关规定。
- 得益于自主权和主人翁意识的增强，大家对较小事务的决策速度更快。
- 更加以执行力和目标感为荣，从而进一步激发奉献精神，甚至提高生产力。

权力分散可以通过技术、远程工作、跟踪决策的高效方式和团队组织来实现，但它首先是个文化问题。指令和职责的微小变化都会导致结果的显著差异。有关处理文化问题的更多内容，请参阅第四部分；要想深入体验红帽创新文化，请参阅*《开放式组织》*。[10]

多面通用功能

公共云部署和数据中心数量的增加会导致企业中细分服务的激增：

- 有些服务仅在 Microsoft Azure 上可用，有些服务仅在 Amazon Web Services (AWS) 上可用，而其他一些服务则仅在公司数据中心内可用。
- 还有一些服务仅在由特定部门控制的托管系统的特定区域中可用。

---

8 Rumelt, Richard:《好战略，坏战略》，纽约：Crown Business，2011 年。

9 L. David Marquet:《你就是艇长：打造“全员领导者”的授权管理与激励日志》纽约：Portfolio，2013 年

10 Jim Whitehurst:《开放式组织》，马萨诸塞州波士顿：哈佛商业评论出版社，2015 年。

这种分而划之有些是出于充分理由的（例如安全防火墙或数据保护），但它通常需要通过一个非常复杂的过程来确定在何处托管新功能。紧随其后的，是关于如何在不同环境中完成同一任务，以及处理其他功能障碍的艰难的学习曲线。

这里要考虑的关键策略转变，是从在单个部门中解决挑战的典型极简主义方法："我们需要功能 X；我们在环境 A 中选择或构建了它；然后就把它放在原处，以备所需。" 转变为：

"我们需要功能 X，我们选择或构建了它，并使其在环境 A 中可用。如果需要，也可以在环境 B、C 和 D 中以完全相同的接口或配置部署它。"

换言之，即为：**单一用途功能 -> 多面通用功能**。

这个概念看似简单明了，但所需的筹划却只是做了简单的考虑，甚至根本没有。

作为一般原则，混合云环境中的功能应做到：

- 能根据需要，以相同的形式用于任意数量的数据中心和云位置。它们应尽可能使用相同的 API、端点和配置。
- 采用分布式功能（例如消息传递、集成、日志记录和跟踪及流程自动化），它们跨数据中心和云位置无缝运行，并且透明地连接这些位置。

注意：根据服务的不同，在另一个数据中心或云端 "可用" 可能意味着同一实例只是可作为 API 服务以远程方式从其他位置进行访问，或是有同一服务的本地实例可用。

由于坚持接口的统一性以及能够在各个位置以完全相同的方式重建功能，创建故障转移配置会变得容易得多。在规划和配置新功能时就进行这项可靠性准备工作，会在此流程的后期产生巨大影响。

与在不同位置使用不同的基础功能相比，这种方法还可以实现执行和重用的统一，从而带来更大的效益。从开发人员生产力的角度看，拥有同一套可用的服务可以减少所需的学习量并提高团队所学技能的适用性。

（部分）约束 = 自由

希望赋能给企业中的更多团队，使之自主做出贡献和运作，这一初衷是好的。但是在实践中，它会如何影响系统？对比控制与自由时，最重要的一点显而易见："您告诉我们来自主负责，但您却施加了大量限制和技术。"

- 太少的强制会导致缺乏规则和实验，从而让应用环境变得脆弱不堪。
- 而过多的强制则会导致人们不满、脱离或频繁采用创意性的变通方法，最终变得更加脆弱。尽管这些变通方法常常包含有用的创意，但最终却会阻碍长期进展。

要想取得成功，问题不在于 "我应该施加多少限制"，而是在于 "我应该在哪里施加限制，又应该在哪里放松限制？"

我们来以交通规则为例，说明这一方法。例如：

- 车辆在道路的哪一侧行驶通常有严格规定。
- 驾驶员几乎可以随意选择所用车辆的类型、颜色、马力和型号。

“Developers’ Exchange 消除了求助于私有领域的专家来解决政府业务问题的许多麻烦。任何开发人员都可以下载我们的源代码并开始使用，而不会遇到任何访问方面的棘手问题。”

Ian Bailey
加拿大不列颠哥伦比亚省政府，首席信息官办公室技术服务助理副部长[11]

因此，尽管从技术上讲，严格实行一条规则（在道路的某一侧行驶）是一种对自由的限制，但实际上也赋予了驾驶员相应的权力，因为它可以让驾驶员知道所有其他人在道路上的哪一侧行驶。实际上，车辆类型也会有一些限制，但其可选范围通常比较广，所以也会让驾驶员感觉并无约束。因此，许多严格的交通规则其实是释放了其他领域的选择空间和创新能力。

IT 系统中的挑战很大程度上在于确定哪里该严格，哪里又该容许自由。

我们将这一政策维度表示为：**监管障碍 -> 商定接口**。

努力确定每个人都应严格遵循的接口和核心流程，并通过以下几方面提高可靠性：

- 在关键领域实行建模规范。
- 强制部门之间就这些接口和技术选择进行沟通。
- 创建一组固定点来测量偏差、变化和性能下降。

令人意外的是，这种方法还可以提高生产力，例如：

- 边界处的接口促使域内部发生巨变，比如各种实施技术的采用。
- 如果能提前解决潜在的问题，团队浪费在与其他部门进行互动的时间就会相应减少。

自助服务配置

设定随处启用 IT 功能的目标，即启用位置不同于最初可能需要它们的云位置（政策领域 2），这一点可视为创建统一环境的一个横向维度。但是，该理念还有一个纵向维度，也就是着眼于如何提供功能。一种简单的办法就是将这种方法想成是一系列问题：

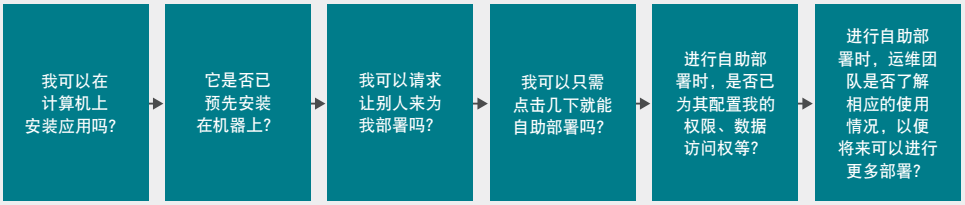


图 4. 您的 IT 系统的自助服务程度如何?

该置备范围要解决两件事：

1. 团队成员需要花费多少时间和精力来设置他们所需的资源。
2. 团队成员访问某些资源所需的潜在技能水平。

我们将这个连续过程表示为：**应用 -> 自助服务**。

11 红帽客户案例研究：“加拿大不列颠哥伦比亚省政府利用开源来改进服务”，https://www.redhat.com/zh/success-stories/government-of-british-columbia。

IT 环境中许多可用的功能都是可安装的应用，这些应用从测试到生产各不相同，仅在某些地方可用，并且可能需要丰富的技能才能正确设置。应用和应用开发工具的自助服务程度越高，企业的生产力也可能会越高。例如，企业可以：

- 减少开发人员团队调配资源、评估方案和提高工作效率所需的时间。
- 繁琐的系统设置通常需要专业知识，通过消除这一最初障碍，个人能够更快地贡献一己之力。

使用这种方法还具有明显的可靠性优势：

- 服务交付的自动化使得错误和调试工作减少，尤其是在初始设置和配置期间，错误可能会导致后期出现无法预料的问题，从而破坏生产的执行。
- 相比单个用户，在为较大共享资源库的多个用户置备资源时，服务的可用性更高。
- 更优的容量和预算规划让资源的使用更可靠、更高效。

采用产品思维方式

在设计微服务系统[12] 和基于 API 的架构[13] 时，一个人们经常提及的有力原则就是从“项目思维方式”过渡到“产品思维方式”。具体来说，面对 IT 系统持续运转的高压环境，完成一系列项目会让人感觉良好。这种方法对于一次性问题或任务尚有一定意义，但如果可交付成果是其他团队所依赖的持久系统元素，那就不太理想了。

我们可以把这概念比作猎人与农民之间的区别。对于猎人而言，捕捉到当天所需的猎物就算完成任务，万事大吉。相比之下，农民则需要定期劳作，并要随着时间的推移才会产生结果。从整个应用环境的角度看，真正的进步是核心资产的改进或新资产的创建，可以通过有意义的方式实现生产的长期升级和改善。

我们将这一政策原则表示为：**项目 -> 产品**。

把 IT 系统视为产品，可促使我们考虑持续生产、用户及用户需求。如果我们打算做更改，那么如何才能最好地提前与他们沟通？这些因素都将应用于客户所使用的产品。但是，这种想法放在影响任何用户组的任何系统也同样有价值（甚至很关键），其中包括：

- 合作伙伴。
- 员工。
- 开发人员。
- 运维团队。

没有产品思维，系统重用便无从谈起。而通过努力形成产品思维，我们可以：

- 通过为可重用服务和功能，确立更清晰的所有权来提高可靠性，创建更具可预测性的升级/降级/停用周期，并减少运维及其他团队所遭遇的意外。
- 通过强化团队所依赖的服务和功能来提高生产力，减少在本地重建或复制的需求，并促使内部服务将来可能成为有价值且经过测试的外部产品。

---

12 Martin Fowler 和 James Lewis：“微服务”，Martin Fowler 博客，2014 年 3 月 25 日，https://martinfowler.com/articles/microservices.html。

13 Manfred Bortenschlager 和 Steven Willmott：《API 使用手册》，https://www.redhat.com/zh/resources/3scale-api-owners-manual-ebook。

转向反脆弱性

“在设计时就为故障做好准备（Design for Failure）”的原则已随着云原生开发而得到普及。容器基础架构大大简化了在故障模式下进行上下扩展的过程，并且受 Netflix 启发的混沌工程学概念[14] 已被广泛采用，甚至在像银行这样历来较为保守的领域也不例外。[15] 这些方法可以说是在 IT 领域对更具普遍性的“反脆弱性”（这是作家 Nicolas Nassim Taleb 所创造的词）理念的概述。简而言之，Taleb 是这样描述这一概念的：“反脆弱性不仅仅是复原力或强韧性。复原力只是事物抵御冲击，并在重创后复原的能力；而反脆弱性则是让事物在压力下逆势生长、蒸蒸日上。”[16]

反脆弱性的核心原则是系统在受到压力时会变得更好。理想情况下，原则会自动发生（例如系统自我修复），但人工细心干预下的持续改进也很重要。

在发脆弱性的背后，更微妙但很重要的一点是，人类的本能以及 IT 领域的普遍本能是将关注点放在保护脆弱的系统上，而不是放在改进计划上。通常，我们的目标是首先防止问题，而不是着眼于当问题不可避免地发生时该如何从中恢复和获得发展。这一趋势是造成在实践中仍然很少采用混沌工程学方法的原因之一。大多数 IT 系统都是分隔的、脆弱的系统阵列，并通过适当的保护措施来防止不确定的因素造成干扰。

从 IT 角度看，反脆弱性实际上包含两个不同的功能：

1. 可观测性，或者说是检测故障或某些不良行为的能力。
2. 可治愈性，或者说是做出适当反应以减少损坏并在恢复后变得更强大的能力。

尽管自动化对于迅速做出反应变得越来越重要，但在这两个阶段中至少有一个可能涉及人工干预。

关键的理念转变是考虑我们目前可以容忍的故障模式范围。然后，我们如何检测它们？我们如何分析故障数据？我们该部署什么系统来应对发生故障时的扩展需要？如何使流程实现自动化？

我们将这一政策原则表示为：**脆弱的系统 -> 反脆弱性**。

对于 IT 企业而言，这条轴可能是最难推进的一条，但也可以说是最有价值的一条。哪怕是朝着这个目标逐步前行，也会有巨大的收获。从可靠性的角度看，通过研究机制，以系统化方式在每次故障后逐步改进，将使系统变得更加可靠。从生产力的角度看，由于减少了对中断的担心，因此可相应地缩短实验周期。

最后很重要的一点是，尽管最初我们可能会从系统和操作的角度来考虑脆弱性和反脆弱性，但这些概念对于开发流程、产品开发和团队协作同样重要。只要沿着反脆弱性轴取得一些进展，任何有助于向前推进的流程都可以从中受益。

---

14 Lori M Cameron, :“混沌工程学：听起来很吓人，但有意地损害系统可以发现更大的错误。如何进行文化转变，来自 Netflix 专家的经验之谈”，《计算机》杂志，2018 年 11 月 15 日，https://publications.computer.org/computer-magazine/2018/11/15/netflix-chaos-engineering/。

15 Paris Cowan：“NAB 部署 Chaos Monkey 来全天候攻击服务器”，IT News，2014 年 4 月 9 日，https://www.itnews.com.au/news/nab-deploys-chaos-monkey-to-kill-servers-24-7-382285。

16 Nassim Nicholas Taleb：《反脆弱：从不确定性中获益》，美国：兰登书屋，2012 年。

自动化即代码

自动化被视为是前两个政策领域的重要组成部分，但鉴于它非常重要，因此我们也希望能对它给予明确关注。

如果没有自动化，今天的大多数 IT 系统将无法正常运行。但在许多情况下，之前技术变革中设计的自动化也可能会成为问题的一部分。也就是：

- 只有一种功能的定制应用，而且是用一种在企业中或世界上不再常见的编程语言编写的。
- 错综复杂的脚本，没人愿意去碰，怕会让它失效。

好的自动化和不好的自动化之间有什么区别呢？尽管红帽的《自动化企业》电子书中详细描述了自动化最佳实践，但有个一般原则我们得了解，那就是：隐式和显式自动化之间的区别。

隐式自动化是原地执行的一次性代码或脚本，而显式自动化则被识别为代码，并作为系统配置的一部分明确地进行版本控制、测试、更新和管理。

我们将这一政策转型表示为：**手动配置 -> 自动化即代码**。

几乎每个企业在其系统的很多部分中都有手动配置。随着时间流逝，将其转换为显式管理的代码会创造很多价值。例如：

- 从可靠性的角度看，它减少了由于缺少依赖项而导致出错的机率，加快了变化能力，并在变更流程上建立了规范。
- 从生产力的角度看，它减少了缺乏了解的系统领域，并缩短了在难以诊断的故障上所花费的时间。
- 自动化即代码还提高了可重用性，缩短了学习曲线。

总体而言，向自动化即代码的转变是取得真正意义上的 DevOps 成功的前提，因为它将配置与代码分隔开来。

深度安全防护

长期以来，边界防护一直不足以解决 IT 安全问题，但无论对于整个基础架构还是在数据中心的子组内，许多企业仍然依赖于边界防护。在混合云方案中，不仅应用会分布在多个数据中心和云之间，而且一些位置之间的数据流还有集成和通道。这种配置意味着一旦一个位置的系统受到破坏，其他位置也同样可能会遭到不必要的访问。因此，至少应从三个维度来评估安全防护：

1. **纵向：**从操作系统到虚拟化、容器管理和代码执行运行时，再到应用管理，良好的安全防护要求整个堆栈中都使用一致且最新的软件。
2. **横向：**对于微服务模式、API 和应用间的通信，良好的安全防护要求对大多数甚至所有通信进行加密、跟踪和访问控制。
3. **团队和生命周期：**即便是研究并选择了正确的技术和流程，开发和运维团队也需要能够应用这些技术和流程。这些流程应尽可能实现自动化，并作为开发周期的一部分被强制执行。

这里要考虑的一个转变就是，是否要维持独立的安全和开发团队。开发团队名义上要遵守安全准则，但通常只需在生产代码部署之前的短时间内考虑安全问题。而另一种协作式、更加分散的方法可能会更好，它允许：

- 开发、运维和安全（DevSecOps）团队更加紧密地展开协作。
- 在一开始的原型阶段便内置安全保护功能。
- 将尽可能多的核心安全规程实现自动化并硬布线接入开发和操作环境，从而简化安全决策。

我们将这一政策原则表示为：**边界防护 -> 普适防护**。

显然，安全防护对 IT 系统的可靠性轴有着重要影响。通过保护流程与数据完整性，安全防护可减少停机时间和灾难性损失的风险。但是，仔细考虑安全防护模型，扩大团队责任并自动化许多流程（如预先配置并得到许可的容器镜像）就意味着要提前做出更多规划，从而进一步降低风险。

由于安全防护不像嵌入式密码、直接数据库访问或其他一些快捷功能，而是通常涉及多个复杂的开发步骤，所以可能会被视为对生产力的一种拖累。尽管短期内可能如此，但得益于充分的流程自动化和支持，安全防护可以减少未来的重构以及为响应事件而浪费的时间，最终提高生产力。拥有完善且可重用的安全机制也终究会让开发人员变得更加轻松。因此，花时间降低设置和配置的复杂度是非常值得的。

将政策领域集中到一起

并非所有的政策领域都对每个企业具有同等重要的作用，在某些领域，不同的选择其意义也各有不同。但是，当从整体上观察 IT 系统时，政策大致涵盖了可以释放显著优势的主要领域——参见表 1。

表 1. 各项建议的优势概览

建议	对可靠性的提高	对生产力的提高
权力分散	• 更易于应用最佳实践 • 对关键项目进行控制与监督 • 聚焦最重要的问题	• 灵活地利用本地环境和知识 • 更强的自主权和主人翁意识
多面通用功能	• 从多个位置提供服务——内置故障转移 • 执行的一致性 • 服务速度	• 约定优于配置——以相同的方式适用于任何地方 • 可转移的技能
约束 = 自由	• 建模规范 • 关于接口的明确沟通 • 使用固定参考点来测量偏差、变化和性能下降	• 边界处的固定接口促使域内部发生巨变，例如多语言 • 事先商定交互模式，减少部门间的时间浪费

建议	对可靠性的提高	对生产力的提高
自助服务	• 自动化服务交付和纵向扩缩容，以避免错误 • 更高的服务可用性 • 改进容量规划并提高效率	• 设置速度 • 使专业技术知识较少的用户也可以使用某些服务
产品思维	• 更大且更加明确的服务所有权 • 升级、降级、停用等的周期可预测	• 可供重用的组件更强大 • 无需本地重建 • 内部和外部使用同一产品会更加高效
转向反脆弱性	• 可靠性得到重大改进——从每一次故障和修复中学习 • 提高对中断范围的容忍度 • 分离依赖项，避免级联故障	• 减少对中断的担心 • 缩短实验周期
自动化即代码	• 通过消除所需的手动步骤来减少错误 • 需要部署修补程序时，加快更改速度 • 建立变更流程的设计规范	• 移除或加速单调任务 • 故障减少意味着有更多的时间用于创新
深度安全防护	• 安全事故的减少可增加正常运行时间 • 对多个层面的问题提前做出规划，让更多团队成员思考安全问题	• 从一开始就将安全防护纳入某些系统 • 共担责任可能在一开始时会让事情放缓，但最终会加快整体步伐

从政策到行动——一些建议

一旦建立了政策和准则，下一步便是确定相关的行动，以便将企业和所部署的系统推进到所需的状态。这些行动既可以是针对总体规划成果，也可以针对模式的强化或特定的项目。接下来的两节将提供有关这一关键实施阶段的注意事项，但我们会先从一些常规建议开始：

- **侧重于整个环境的演变，而不是颠覆性变革。**大多数企业已经为业务的不同部分制定了一系列复杂的计划、项目和活动，因此添加更多的内容可能会令人怯步。全环境的应用策略最有价值的贡献，就是根据整个企业的健康状况对变更做出评估。整体性的演变要比几个部门的颠覆性变革更能支持这一目标。这种方法并不表示英雄无用武之地。实际上，这些成就应作为整体改善的杰出典范来看待。
- **像思考目标一样，我们也要对系统和指标多加考虑。**设定改进目标通常相对直白。“我们需要在 X 上做得更好”，这句话说起来容易。但是，这类目标及其关联的行动也只能吸引一阵短期的活动。一种更有效的方法是选择一个或多个指标来获取此维度中的环境状态，并朝着一组可以加以度量并推动指标向前发展的行为和习惯不断努力。

表 2. 第二部分所讨论的政策领域的指标和流程示例

建议	指标示例	流程类型
权力分散	对变更或例外做出决策的时间（根据问题的大小评分）	• 企业和部门规划流程制定政策的常规节奏 • 部门间的变更请求或异常流程
从任何位置都能访问	每个区域、数据中心或云端的服务可用性及速度或服务级别协议（SLA）	• 跨属性逐级优先部署服务 • 对相似功能进行评估和连续测试
约束 = 自由	• 采用设计框架、模式或准则 • 解决方案的合规性 • 例外请求的数量	• 整个企业范围内接口、产品和内部服务的共享 • 对采用集中化准则的常规节奏认可
自助服务	使用某种指标（如所用 API 的“Time to first hello world”或所用 API 的“第一次出现第三方用户的时间”等）来访问关键服务或产品的速度或步骤数量	已更新的可用服务列表的常规节奏周期，以及用户访问它们所需的时间度量
产品思维	使用特定产品或服务的部门、服务和产品的数量	发布产品和服务的节奏，反馈机制，传达最新变更、共享使用指标的通信手段
转向反脆弱性	度量失败后复原的时间，以及再次发生类似故障的趋势（希望减少）	• 定期进行混沌测试，在整个系统中引发模拟或实际故障，以检测脆弱性 • 引发系统冲击的严重性范围的逐渐扩大
无处不在的自动化	测量手动部署或配置或其他步骤的数量，或实现完全重新部署所需的端到端总时间	定期审查自动化指标和现行计划，以减少最大的痛点
深度安全防护	• 检测和捕获到的安全事件数量 • 采用安全准则并受到监控的组件数量	• 定期对压力系统进行渗透及其他安全测试 • 事后回顾

“一旦有了好的主意，我们就可以立即着手开始构建相应的产品。这种敏捷性前所未有。”

Tobias Mohr
Lufthansa Technik 旗下 Aviatar
平台技术与基础架构主管 [17]

第三部分：通往成功所需的架构

上面几节中的策略原则比较抽象，因此可以应用于各种系统。接下来顺理成章的问题就是：

- 这些系统是什么样的？
- 需要考虑哪些组件？它们是如何组织的？

没有任何一种架构能适用于所有情况。每个企业对架构的选择始终各不相同，它们通常代表着可以依靠的核心优势。但是，如果有一份图谱，上面描绘了应用环境典型要素以及在应用云原生和混合云技术时如何连接这些要素，那将非常有用。

在本节中，我们将提供一个总定义，然后深入探讨对现代化 IT 至关重要的诸多方面。

统览全局

借用我们先前对应用环境的定义（即整个企业中用于进行应用交付和应用开发的一系列能力），图 5 展示了整个企业范围内大型应用环境的一个整体视图。

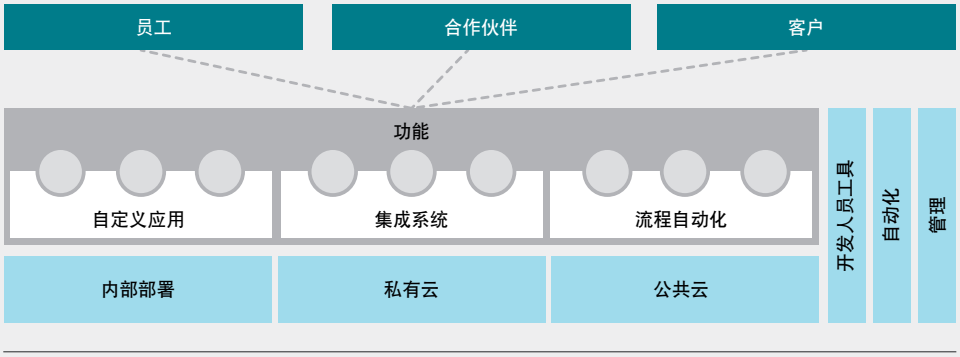


图 5. IT 应用环境概览

在本节中，我们重点介绍应用环境的组件及其连接方式。其中的每一层都可以视为与第二部分中讨论的策略原则相垂直。策略原则将横贯这些层，并且指导它们的设计。层的划分如下：

1. **平台与应用交付。**托管代码流程的基础架构，这些流程对应用进行实例化，以便在操作系统和数据中心层面提供价值。
2. **自定义代码应用。**支持执行在功能上具有创新性的、基于自定义代码，并使企业在竞争中脱颖而出的应用。
3. **集成应用。**支持应用之间的通信，从而直接创建用户可以体验的功能。
4. **流程自动化应用。**支持的应用不仅涉及执行代码，而且还涉及手动操作流程（工作流），以及由企业中非开发人员提供的逻辑规则。
5. **开发人员工具、DevOps 和管理。**围绕应用环境的功能，旨在保持团队成员的生产力和系统的正常运行。

---

17 红帽成功案例：“Lufthansa Technik 构建云平台，优化航空公司运营”，https://www.redhat.com/zh/success-stories/lufthansa-technik

“很明显，这个地方的地形已经改造得不成样了；没有其他办法再让大洲最终形成完美的正方形了。”

Rachel Bach
作者，*“天堂女王”* [19]

我们会在下面的相应章节分别对各个方面加以介绍。此外，要想成功实施这些策略和架构，其关键因素是解决手动流程和架构模式的问题。维持和发展系统的团队动力、行为和文化都是成功的重要因素。它们还记录了企业解决问题的方法。我们将在第四部分中讨论这些因素，您也可以在*《开放式组织》*找到更详细的分析。[18]

根据侧重点的不同，我们还有其他分割 IT 基础架构的方式，例如移动应用支持或物联网（IoT）。但是，这里的模型是有意构造的，旨在说明构成 IT 环境的全套连接元素的重要性。通常，大多数 IT 策略建议要么只专注于自定义代码开发，而忽略了大部分 IT 与集成和流程有关这一事实，要么就是假定采用特定的技术堆栈。

大多数大型企业的应用环境是多种多样的，自然涉及多种开发形式。典型的企业通常要运行数千个单独的应用、各种 SaaS 服务、现成的商业解决方案以及其他系统。我们所讨论的架构层提供了有关如何连接其中一些系统的一般概况。示例中引用的是红帽技术（在适用的情况下），但当然还有其他供应商的解决方案可供选择。任何足够复杂的系统都需要依靠多源技术。因此，此处所列的特定技术项目仅用于说明目的。

平台与交付

迄今为止，平台和应用交付技术的改进可以说是云原生和混合云功能最显而易见的驱动力。平台层提供了应用执行、集成和交付价值的基础。如今，从操作系统到虚拟化再到容器化部署的转变，让各种不同类型的部署和自动化成为可能，无论是在广度上还是深度上。这种技术上的转变还进一步推动了这样一个想法：即无论企业中的开发人员在何处部署（数据中心内，或诸多公共云或私有云中的任何一个），他们都可以使用完全相同的执行环境。

IT 地形分层的五个等级：

1) 平台 -> 土壤和养分

2) 应用 -> 原子生命、植物和小动物

3) 集成 / 消息传递 -> 社区

4) 流程 -> 人类文明

5) 开发人员工具 / DevOps 和管理 -> 生态系统流程及检查和平衡

在涵盖多个本地位置的混合云环境中，其更大的优势表现在：

- **跨位置体验的一致性**确保了在尽可能多的环境中使用相同的操作系统版本、补丁级别及其他系统元素。
- **统一管理**，统览本地部署、虚拟化、私有和公共云系统，便于以同一方式来跟踪整个 IT 基础架构。
- **对安全性和合规性的控制**仍然至关重要，尽管不同的云和数据中心部署量在激增。
- **提高了稳定性、敏捷性和可用性。**通常，基础架构扩展的目标是通过扩容实现故障转移和冗余。但是，对这些部署采用一致的方法至关重要，否则异常环境可能会损害可靠性。

要想深入了解红帽操作系统以及私有云和混合云方法（包括许多客户示例），请参阅红帽的混合云策略电子书。不过，近年来新增了一种在多个云环境中创建一致平台的方法：容器和容器平台的使用。

容器已成为现代化 IT 中首选的部署基础架构。这些架构对应用及其整个运行时环境（运行时所需的全部文件）一起进行打包和隔离。借助容器平台实现快速部署和轻松管理，意味着可以在容器平台可用的任何位置部署、更新和扩展应用实例。容器和容器平台真正实现了在混合云上进行云原生计算的愿景。

---

18 *Jim Whitehurst：《开放式组织》，马萨诸塞州波士顿：哈佛商业评论出版社，2015 年。*

19 *Rachel Bach：“天堂女王（悖论之书 3）”，Orbit，2014 年 4 月 22 日。*

在混合云环境中，利用容器平台可以在所有数据中心和云环境中创建真正相同的计算环境，方法如下：

- 在每个位置提供相同的计算基础架构。
- 允许打包的容器无需修改即可在混合云环境中的任何位置运行。
- 允许运维团队查看各个容器集群中的工作负载。
- 允许对每个位置的资源进行纵向扩缩容。

容器平台会在底层资源上创建一个抽象层，从而使它们能在任何地方重用并保持一致。基于 Kubernetes 的红帽 OpenShift® 容器平台便提供了所有这些功能，而且还不限于此。如今，在任何客户的混合云中，混合的位置都可以包含三个主要公共云（Amazon Web Services（AWS）、Google Cloud Platform 和 Microsoft Azure）各自的托管产品。有关在混合云中使用容器的更多信息，请参阅红帽云容器主题电子书。

平台策略的三个注意事项包括：

- **让平台尽可能地实现自助服务**，从而在采用严格一致的安全模式的同时，还允许不同的团队访问他们所需的容量和服务。允许开发团队自助置备虚拟资源对于提高速度至关重要。不过，自助置备需要进行后备控制，以保证系统安全性。
- **整个平台基础架构在所有位置的可靠性。**该注意事项不仅涉及到各个系统或数据中心，而且还涉及它们之间的相互依赖性，以及在发生故障时的更改策略：仅仅是“恢复”系统，还是对其进行更改以防止将来的故障？
- **在复杂的混合云环境中，自动化是关键。**建立自动化对保持混合云管理的可行性至关重要。同时，正如第三部分中所述，将自动化视为必须管理的代码至关重要。否则，自动化本身就会成为一个遗留问题。

自定义代码应用

应用代码本身不会自己执行。从早期的应用服务器到现代应用服务器、轻量级语言运行时的混合以及与容器管理的结合，这种缓慢演变的基础为源代码的执行提供了支持。

图 6. 在整个企业的不同环境中执行自定义代码

随着云原生技术开始进入主流，容器基础架构与语言运行时相结合为创新提供了最灵活的组合。随着企业对容器技术的采用，这种组合很可能会成为主要的部署模式，之前许多应用服务器的编排功能都在逐渐转移到容器管理层。

“现在，我们能够快速部署新应用，而真正的优势在于该方案所赋予我们的灵活性，让按需扩缩应用成为现实。轻点鼠标，只需一分钟，便可自动完成所有变更。无需再像过去一样，花费一天时间才能完成。”

Ivan Torreblanca
首席技术官，Leshop.ch[20]

对于大多数企业而言，这种转换将会花费一些时间，并且可能会混用应用托管方案，包括图 6 中所示的所有方法。

应用开发领域出现了以下的范例，并且正在为大多数大型企业所考虑：

- 在自定义代码的实现方面，尽管 Java 和 Javascript 在企业中仍占据主导地位，但其他语言数量却在不断增加。Cloud Foundry 最近的一份报告中称，业界使用的语言数量已达到了 25 种，其中许多语言在企业受访者的使用占比较低，百分比只有一位数。[21]
- 为了提高自定义代码项目的效率，紧密集成以及纳入用于消息传递和内存数据网格的辅助服务正成为环境中的重要服务内容。
- 随着对快速处理自定义应用的需求不断增加，响应式编程正获得极大的关注。对诸如 Eclipse Vert.x 等工具包的采用一路猛增。
- 功能即服务（FaaS）最初通过 AWS 的无服务器 Lambda 方法得以普及，现在又有了更通用的计算方法，让程序员不必为预先配置计算容量而担心。相反，代码（功能）的各个元素可以仅在发生相应事件的情况下再来部署和执行。FaaS 解决方案现已在主要公共云上提供，并可由内部运维团队为自己的开发人员团队置备（通常在容器平台上）。

这些趋势为企业 IT 开发自定义代码应用增加了重要的新功能。仔细选择要采用的技术非常重要，因为实现大量差异化功能的自定义代码系统会帮助企业在市场上赢得独特的竞争力。

红帽的应用开发产品组合既广泛又深入，涵盖了本节中所介绍的大多数领域。最近的一些创新包括：

- 广泛扩展了可用于支持自定义代码的企业级运行时。红帽的运行时全包套件现在包括 JBoss® 企业应用平台、Node.js、MicroProfile、Spring Boot 等。该套件非常灵活，开发人员可以从中选择他们喜欢的工具，而 CIO 则可以继续将整个企业自定义代码的风险降至最低。
- 纳入用于消息传递和内存数据网格的辅助服务，以及红帽 AMQ 代理和红帽数据网格。
- 与 Kubernetes 和红帽 OpenShift 容器平台紧密集成，藉此与传统上由应用服务器处理的低级别部署功能划清了界限。现在，我们可以通过运行时包中以代码为中心的运行时支持服务在容器平台中处理这些功能。
- 支持 Kubernetes 原生执行机制，在任何部署 Kubernetes 容器平台的地方实现 FaaS。与 Google、Pivotal 及其他云提供商的这种合作被称为 Knative，它为消息传递、事件处理和执行无服务器方法在任何地方运行奠定了基础。
- 最后一点，在云原生环境中，执行速度至关重要。团队要在云端、容器甚至是无服务器环境中使用的应用应在几微秒内启动和关闭：在不需要时速度可以趋同于零，但在被调用时又几乎立即可用。红帽的 Quarkus 技术现在就为基于 Java 虚拟机（JVM）的语言提供了这种速度和敏捷性。

---

20 *红帽成功案例：“LeShop.ch 采用敏捷、可扩展解决方案，支持创新文化”，https://www.redhat.com/zh/success-stories/leshop.ch*

21 *Cloud Foundry：“这些是企业应用开发的主要语言”，2018 年 8 月，www.cloudfoundry.org/wp-content/uploads/Developer-Language-Report_FINAL.pdf。*

“我们的红帽操作环境可以将我们解放出来，使我们无需过多关注基础架构，从而将更多的时间花在创新和交付可改善客户体验的新产品上。”

Michael Catuara
TransUnion 分布式系统主管 [22]

应用运行时策略的关键注意事项是：

- 自主权和一致性：允许在整个企业内使用不同的语言和方法。
- 与 DevOps 和开发人员工具的紧密集成，对于保障开发人员的生产力至关重要。实现标准流程、检查和平衡的自动化，可以提高编写、测试与部署更多代码的效率。
- 建立全面的安全策略可确保从操作系统到虚拟化，从容器层到代码运行时，所有系统都可以通过受信任供应商的自动部署来实现全面更新。

集成应用

由于 IT 系统首先连接到网络，而操作系统首先允许进程间通信，因此集成功能是必不可少的。软件系统的集成方式不断激增。集成已成为 IT 堆栈中最关键的部分之一。

对于现在跨多个数据中心和云的应用环境来说，集成至关重要，但它们也需要适应现代需求。尽管传统集成模式（例如企业服务总线（ESB）部署）为企业带来了长足的发展，但它们远远不足以应对混合云挑战。尤其是，集成技术还要面对许多挑战：

- **跨多个位置的无缝运维。**这种运维意味着消息传递、API 管理、转换、数据映射和其他功能都应在多个物理位置完全可用。这些功能也需要跨位置协同工作。例如，消息传递解决方案通常需要在多个物理位置上使用，而不仅仅是在一个位置内使用。IDC 在最近的报告中将该功能称为“可移植性”。[23]
- **集成即代码。**系统之间的集成可实时执行关键转换和数据映射。该功能需要对软件端点进行代码更改，提供可以表示对集成的同步更改的数据。集成应该像应用一样被视为代码，以便能以相同的方式进行版本控制。
- **极度扩展。**尽管集成技术始于主要连接后台应用的数据中心，但在许多情况下，面向客户的应用也需要访问这些后台系统。同时，流量也会显著增加。因此，对于使用最频繁的数据和交易系统来说，集成需要快速、低成本地扩展。
- **为更多利益相关者提供服务。**系统数量已迅猛增长，IT 几乎渗透到了每个业务功能中。企业中所需的集成数量和类型已显著增加。无论对于开发和运维团队，还是对于在业务线领域中技术培训不多的人员（称为普通集成人员），支持集成活动都变得越来越重要。
- **数据的重要性日益提高。**与集成在面向客户的应用中的重要性一样，集成在大规模、实时数据获取和分析趋势中也是重中之重。此外，由于物联网设备的激增，也需要新的方法来管理它们生成的大量信息。要让现代系统有效运行，我们需要及时处理、存储数据，并且经常准确复制。

为了应对这些变化，集成技术本身也在不断发展。在与云原生功能相结合的情况下，红帽产品组合中的集成技术：

- 可进行完全分布式部署，在需要的地方提供单独集成。
- 采用容器原生形式，允许自动扩展容量。
- 可通过基于容器的 Knative 技术按需使用，即时启动和关闭。

---

22 红帽成功案例：”TransUnion 对 IT 实施现代化改造，提供更出色的客户体验“，www.redhat.com/zh/success-stories/transunion

23 Fleming, Maureen：”2019–2023 年全球集成和 API 管理软件预测“，IDC，2019 年 6 月，https://www.idc.com/getdoc.jsp?containerId=US45126319。

“从感恩节到 12 月底，我们的业务会出现急剧增长。网上购物增多，而激增的退货量也延长了我们的工作时间。借助 OpenShift，我们可以在那些特定的高峰时间进行灵活扩展。如果需要，我们甚至还可以扩展到公共云。“

Carla Maier
UPS 云平台和技术部高级经理 [24]

- 所有集成都可以作为代码进行操作，这意味着可以应用常规 DevOps 实践，并可通过图形拖放界面进行访问。借助这一功能，无论是技能高超还是技能欠缺的个人都可以就相同的集成展开协作。
- 包含更广泛的消息传递功能，从传统的 AMQ 式消息传递到 Apache Kafka，不一而足。
- 消息传递、API 和集成技术可以跨不同的云进行部署，并提供运行最终用户应用所需的应用级通信。

对于企业而言，最重要的集成注意事项包括：

- 为哪些产品和服务可用以及其他软件团队如何使用这些产品和服务制定战略。通常，API 是扩大采用范围的一个重要组成部分。
- 针对谁来负责这些系统间的集成，确定一项策略。通过这种工作量分配，可以提高各部门的自主权并减少集中式整合能力中心 (ICC) 团队的工作量。
- 解决策略性问题，例如为变更数据的获取、联合数据的虚拟化、消息传递以及跨数据中心和云集成的数据转换确定最佳实践。这些功能将如何与平台层进行连接，并在任何地方提供无缝服务？

要想深入了解云原生和混合云环境中的集成策略，请参阅红帽的敏捷集成主题电子书。

流程自动化应用

企业执行的每个流程几乎都有软件参与。其中许多流程都涉及到人员 (员工、合作伙伴和客户)，以及基于数据和状态的复杂决策。

第三种在混合云时代发生显著改变的应用是流程驱动型应用。

显然，对业务流程管理 (BPM) 应用和自动化智能决策的需求都正在不断增长，其中涵盖了从基于规则的系统到机器人流程自动化的所有内容。随着基础架构分布在混合云之中，并且整个企业都广受 IT 方面的影响，流程驱动型应用表现出以下趋势：

- **微服务模式、API 和集成的采用**为整个企业中流程与新系统的连接创造了机会，它建立了更丰富的工作流，并将自动化应用于以前无法访问的数据集或系统。
- **采用配置、业务级逻辑和规则的解决方案变得越来越重要。**尽管环境中的新功能很大一部分都是自定义应用提供的，但能否根据不断变化的业务需求调用该自定义逻辑至关重要。可配置的规则能够在云和容器中将操作作为 FaaS 调用来执行。
- **流程分布程度增加**，因为企业中数据中心和云位置数量不断增加，同时流程还要跨越这些位置，这就需要采用云原生方法来进行流程管理。这种方法不仅仅是部署在一个位置，而且还可以配置成在任何需要的地方运行，并在所有位置创建一致的流。
- **BizDevOps 的兴起，**其目的是将业务需求与敏捷应用开发直接结合在一起。尤其是，工具链变得越来越集成，从而将代码概念与现金收益联系起来。
- **可扩展性。**流程驱动型应用现在已成为一些常用 IT 服务的基础，特别是在拥有大量员工或客户的企业中。因此，有必要按需纵向扩展关键流程的使用，以免出现故障和瓶颈。

***24** 红帽案例研究：”UPS 借助 DevOps 和红帽技术简化了包裹的跟踪和递送“，www.redhat.com/zh/resources/ups-customer-case-study*

“我们与整个企业中的多个 EMR、实验室及其他系统建立了数百个接口。这是一个巨大的挑战。健康信息交换团队在过去五到六年中一直在埋头苦干。但是，我们具有明显的优势，因为我们的大多数数据都集成在一个地方，可以轻松获得。这些数据可用于深入分析，或供决策引擎用来提供建议。”

Vipul Kashyap
Northwell Health 临床信息系统主任兼企业信息架构师 [25]

- **无服务器和 FaaS 平台的使用**，供业务规则和业务流程管理系统用于帮助提高可扩展性、改进性能并节省成本。决策及其操作可以作为 FaaS 执行。同样，业务流程中的步骤也可以作为 FaaS 执行。
- **后台系统与面向客户的系统之间的 IT 集成，以及由此带来的数据增长**为企业创造了一个良机来着手为客户部署智能化应用。制定基于规则的决策，将人为决策和自动化决策拼接在一起，同时辅以速度和规模，这都是利用这些机会的关键。

所有这些趋势都表明，流程驱动型应用的现代浪潮已成为 IT 策略的重要组成部分。

与集成技术一样，云原生开发也让流程自动化技术发生了重大变化。在红帽产品组合中，这代表着：

- **容器上的原生部署。**与容器和容器编排技术的紧密集成是红帽流程自动化平台的核心组成部分。流程驱动型应用受益于云原生架构所提供的可扩展性、弹性和位置独立性。
- **无服务器和 FaaS。**红帽的流程自动化解决方案 Kogito 通过为流程驱动型应用提供无服务器和 FaaS 模型支持来扩展云原生功能。Kogito 是利用 Quarkus 从头构建的，可提供按需服务所需的快速启动、扩展能力和内存效率。
- **认知计算。**云原生架构提供了支持资源密集型人工智能 (AI) 和机器学习 (ML) 工作负载所需的规模，这可以将决策服务扩展到更广泛的场景。通过将预测模型整合到用户编写的“决策模型和标记”(DMN) 决策模型中，红帽决策管理器帮助企业提高了自动化程度并减少了需要人工干预的异常情况。
- **智能化管理。**云原生开发可通过 Kubernetes Operator 自动执行应用生命周期管理和故障响应，进而减少运维工作量。红帽流程自动化解决方案中包含 Operator，便于流程驱动型应用轻松管理分布式混合云工作负载。

这方面相关的策略决策包括：

- **使越来越多的员工、合作伙伴和客户能够参与流程驱动型应用的创建和配置。**这种类型的应用可以借助技术专家之力提供一个安全的结构化环境，为权力分散创造了机会。之后，域专家可以利用该环境来控制应用以满足自己的需要。
- **确定协议和规程、角色和职责**，以便在实现手动业务流程的自动化时开展业务线与 IT 之间的协作。
- **确定使用 FaaS 和云进行决策和业务流程管理 (BPM) 的方法。**例如，是在云端、在本地，还是同时在两者中使用 BPM 即服务。
- **以服务的形式按需提供流程管理可能会带来最大的回报，**让以前没有流程专家的团队可以开始获取和编排自己的工作流。
- **转向反脆弱性。**通过用可配置的规则和标准格式来替换为流程定制、编译的代码，可以更好地验证复杂流程，并且在变更时无需涉及代码部署。这种方法降低了系统中断的风险，并有助于更多的团队成员了解重要流程中固有的依赖关系。

---

25 *RTInsights 分析师文章：“利用红帽决策管理器构建临床决策引擎平台”，2018 年，www.redhat.com/zh/resources/building-a-clinical-decision-engine-rtinsights-article。*

“OpenShift 和 OpenStack 使我们的团队能够快速构思想法，清除杂念，集中精力进行创新和发现。再也无需考虑是用大型服务器还是小型服务器来运行应用。”

José María Ruesta
BBVA 基础架构、服务和开放系统部门全球主管[26]

开发人员工具、DevOps 和应用管理

DevOps 和开发人员工具是云原生技术的基础。从本质上讲，许多云原生技术都是从对基础结构的核心需求演化而来，以便更有效地为开发人员提供支持。云原生计算基金会（CNCF）对云原生技术的定义开头写道：[27]

“云原生技术使企业能够在现代、动态环境（例如公共云、私有云和混合云）中构建和运行可扩展应用。”

CNCF 给出了一些示例，然后强调：

“这些技术使松散耦合的系统具有了弹性、可管理性和可观测性。结合强大的自动化功能，它们使工程师能够频繁且可预测地以最小的工作量进行高影响力的变更。”

以这种方式将弹性、可管理性和可观测性（从运维端）与工程影响（从开发人员和工程端）结合起来，促使许多技术汇集在云原生的周围。

正是这些特性让云原生技术成为一种引人注目的技术集，随着企业逐步采用多个云和数据中心作为基础架构布局，该技术可用来应对这一过程中出现的诸多挑战。

DevOps 实践、应用服务、中间件、开发人员工具和管理以及新的云原生变体共同构成了企业的 IT 环境，并为之提供支持。它们协同工作，创造了企业的应用环境。反过来，该应用环境所支持的功能也让员工、合作伙伴和客户获得了所需的价值。

关于 DevOps 技术在企业中的成功应用，有大量文献可供参考，其中红帽 DevOps 资源页面就是一个很好的起点。此外，红帽开放创新实验室及其他服务团队也提供了帮助入门的计划。本节无意重复这些一般性的指导，而是会重点介绍因采用混合云而加快的一些趋势：

- 随着基础架构和应用跨越多个数据中心和云，源自云原生 DevOps 的“基础架构即代码”理念变得至关重要。如果无法实现多个位置部署的完全自动化，混合云环境就会变得越发难以管理。
- 这种对自动化的需求也扩展到了应用代码身上。由于应用现在会跨越多个位置的众多服务和 API 进行紧密交织的集成，因此需要对部署进行精心编排，以免出现停机。
- 几乎所有的 IT 环境都是混合平台环境，其中可以包含运行各种操作系统的裸机、虚拟化服务器设施，以及私有云、公共云和容器。这些不同平台级别的混合意味着任何整体方法都需要跨平台来连接管理实践。
- 开发工具需要随着基础架构来变化和更新，以提供实时测试及其他云原生功能。为了满足这一需求，代码完全在浏览器中的云集成开发环境开始引起人们的关注。
- 在混合云环境中，可观测性和跟踪变得越来越困难，也越来越重要。代码执行通常涉及多个位置，必须在每次调用时进行跟踪。需要将系统不同部分的故障关联起来，以确定出了什么问题。

---

26 *红帽案例研究：“BBVA 借助云原生数字平台改变客户体验”，https://www.redhat.com/zh/resources/bbva-customer-case-study*

27 *摘自云原生计算基金会 (CNCF) 对云原生的官方定义 (v1.0) https://github.com/cncf/toc/blob/master/DEFINITION.md*

这些趋势都提出了新的挑战，而现今的工具也在迅速发展，以应对这些困难。每项挑战都源于基础架构在不同位置、不同应用类型和不同技术上越来越分散化。部分应对举措是实现部署元素的标准化。另外就是部署能明确解决系统分散性的工具和架构模式。要想取得成功，两者缺一不可。

红帽解决方案堆栈已经汇集了不少这方面的产品，其中包括：

- 红帽 OpenShift 下的多集群管理，以及各种工具，用于将持续集成和持续交付（CI/CD）流程与容器部署集成在一起，以创造一个强大的混合云环境。
- 红帽开发人员的 CodeReady Workspaces 提供了一个针对容器部署代码进行了优化的、完全基于 Web 浏览器的集成开发环境（IDE）。其功能包括基于容器的工作区、预建或自定义堆栈、开发人员配置的集中管理等等。
- 红帽 Ansible® 自动化平台可简化混合 IT 环境的管理并增强其安全性，它面向的是具有工作流功能和安全标准的混合云。[28]
- 红帽的预测性监控解决方案（即红帽智能分析）现在已包含在所有红帽企业 Linux® 订阅中。该平台可跨各种红帽平台提供集成安全性、性能、可用性和稳定性，以提高混合云的可见性。

从长期计划的角度看，以下策略领域是开发人员工具、DevOps 和应用管理的关键所在：

- **全面通用功能**确保了在每个位置以及针对应用交付基础架构（平台和自动化）和应用本身都可以使用相同的底层基础架构、服务和自动化。通常，部署工作需要在多个位置运行每个应用的最新版本。
- **开发人员工具方法。**举例来说，要在本地和云环境中提供一致的开发人员体验，最佳的选择可能是在容器上运行的、基于浏览器的 IDE。
- **开发和运维团队的产品思维。**创建可重用的应用和服务是一项艰巨的工作。但是，要建立对他人所运维的产品和服务的信任，更是难上加难。产品思维方式要求团队从长远的角度考虑他们所提供的系统。这会影响他们的规划流程，并且需要在代码更新、部署和停用所涉及的自动化、日常习惯与实践中加以体现。
- **自动化。**如果不对流程和自动化本身进行仔细管理，即使成功地实施了 DevOps，它也会很快变成一个僵化系统，回报率将不断下降。

---

28 *红帽新闻稿：“利用最新版的红帽 Ansible Tower，红帽统一了混合云管理的自动化”，2019 年 1 月 9 日，https://www.ansible.com/press-center/press-releases/red-hat-ansible-tower-3-4。*

第四部分：应用环境之旅

红帽 IT 团队评估了构成红帽业务的每个应用的价值及变革就绪状态（见图 7）。团队运用了相关的价值标准（例如应用的更改率及其寿命）及就绪状态标准（包括业务关键性、业务支持和架构差距）。

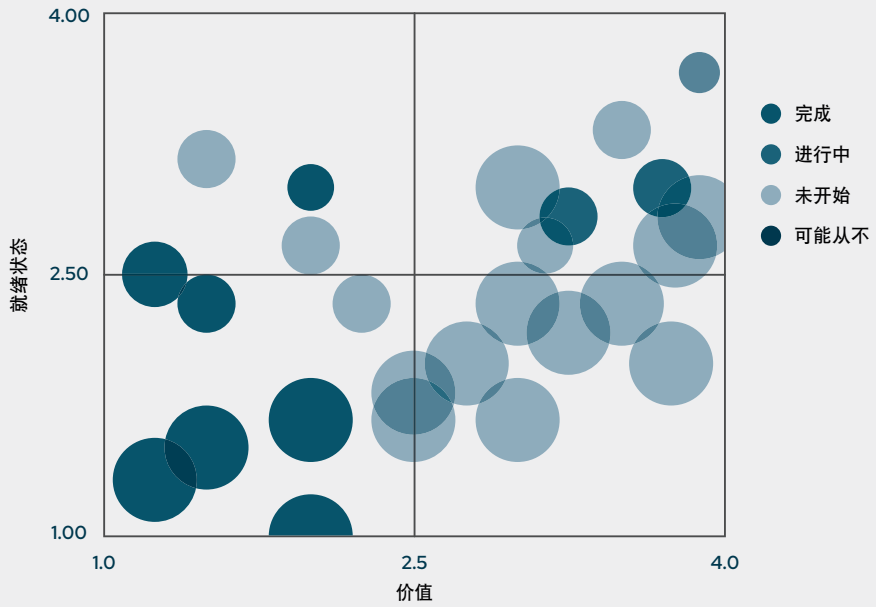


图 7. 针对现代化的应用评估（由红帽 IT 提供）

除了初创公司以外，很少有公司的 IT 策略纯粹是新的策略。通常，IT 策略研究的是如何发展整个功能系统，以提高可靠性和生产力。个例爆发快速生产力有时可能发挥作用，但如果不普遍采用最佳实践，这些实例最终只会耗费资源。

本节将第三部分中的策略区域在图 8 中进行了排序，旨在探讨云原生和混合云应用环境之旅。

图 8. 应用环境中弹性和生产力演变的线性表示

必须要注意的是，这些并不是技术采用阶段。有些企业的某些（或许多）部分可能从未采用有些技术。这些阶段反映的是政策、功能和决策。

期望复杂的大型 IT 企业立即或在整个组织内同步推进所有系统、流程和策略方向也是不现实的。

之所以将这些元素以可视化的形式显示为一个序列，目的是突出改进的逻辑顺序，更重要的是强调整个应用环境是一个实体。尽管可以对任何单元、任何阶段进行改进，但如果能考虑这些更改对整个环境的影响，那会很有意义：这种改进是否具有传播性？还是这种改进会保持分割状态并最终消失？

不成体系的行政体制

不成体系的行政体制听着对企业来说也许不太友好。但是，由于当今 IT 系统的复杂性，许多 IT 企业的现实就是如此。上一代构建者的规则、流程和技术选择，造就了系统和企业的现状。这些决策就反映在 IT 企业中。任何一系列变革，无论多么巨大或多么持久，也都不可能让企业及其系统变得完美无缺。下一次变革时，之前的完美就会变得不合时宜。正如 Gartner[29] 所述，管理技术负债实非易事。实际上，IT 企业不仅仅是要实施应用，他们始终要实施的是一个可能有数十年寿命的复杂生态系统。功能是暂时的，而结构是永久的。

重要的是，不仅不要被标签所困，还要推动企业和结构的不断改进。运作如此庞大、复杂的人员和技术系统组织，离不开相应的行政机构——有些不成体系始终在所难免，甚至是必要的。这里要考虑的问题是：

- 它为企业打造的现行结构和应用环境有多好？
- 要想使整体不断改进，最重要的方向和机制是什么？

云原生和混合云路径的步骤

对于许多 IT 企业而言，使用多个数据中心和多个公共云位置（经常来自多个云提供商）现在已经是家常便饭。从技术角度看，为了帮助提高这些环境的可管理性，有多种策略可以采纳。

改善为企业提供支持的应用环境可能需要一系列技术，但最重要的是需要就变革领域的优先顺序做出战略性的决策。从发展进程来看，以下四个方面就像是一步步的台阶：

- **产品思维。**针对任何要重用的系统采用并培养产品思维，将有助于团队更加自主地采取行动。这也是让企业中的团队依靠并信任他人所运行的系统的先决条件。实际上，这一关注点也是微服务架构的一个特征。微服务架构主张团队应在产品的整个生命周期（从开发到生产维护）全面掌控产品，从而拉近了与业务功能以及如何增强业务功能之间的联系。产品思维有助于引导关于哪些接口和流程真正重要的分析。域边界在哪里？哪些 API 和功能集需要切实的支持？
- **无处不在的服务。**十分有必要考虑超越初始部署的范围，并规划在多个位置提供无缝功能。这种规划（或部署）可能会有成本，但相比针对同一问题处理完全不同的解决方案，成本通常会低一些。这一方面还包括尽可能实现自助服务，以便能让开发人员和潜在用户自行配置资源。以这种方式运行基础架构可显著提高效率和生产力。随着容器和无服务器类型技术的日益普及，由此带来的好处会越来越重要。

29 Andy Kyte：“Gartner 主题演讲：关于技术债务，您始终想知道（但不敢问）的一切”，Gartner 应用策略与解决方案峰会，2018 年 11 月 27-29 日，https://emtevr.gcom.cloud/events/apn32/sessions/apn32%20-%20k4%20-%20gartner%20keynote%20everything%20you%20always%20wanted%20to%20k%20-%20373224_47073.pdf。

- **自动化即代码。**在整个 IT 环境中提高自动化水平可带来更高的投资回报率，但前提是必须以显式的方式维护这种自动化，就像维护代码和应用一样。没有显式管理，今天的自动化可能会在明天惨遭淘汰。类似的理念转变也适用于集成、决策规则和流程自动化代码。它们是应用功能的生成来源或支撑基础。在许多情况下，系统中存在的集成代码数量要多于自定义逻辑。如果将这些系统明确地视为代码，便可以更加轻松地管理跨系统部署。
- **反脆弱性。**转向反脆弱性转变意味着准备好强大的流程（自动或手动）来对故障做出反应并改进系统，使其超越以往。很少企业以任何有意义的规模采用了极端的、类似 Netflix 这样的混沌工程学理念，因此向反脆弱性推进的目标非常重要。同样，把安全防护纳入在这个步骤以及之前的所有步骤，也代表朝着主动性迈出的重要一步。

反馈循环和文化

采用新技术和新流程总是会带来痛苦和不确定性。在许多情况下，最好不要更改已经在正常工作的东西。聆听反对者和拥护者双方对变革的看法都至关重要，这是红帽 IT 团队所收获的一大宝贵经验。

通常，这两个观点相对的阵营都有值得借鉴之处。“为什么 A 已经正常运行，还要实施 B？”尽管这样的话一方面可能出于对 A 的感性依恋，但也可能表示 B 不太可能解决根本问题。

红帽 IT 团队进行了一项有益的实践，即根据应用的策略性质将其映射至相应的功能和组，如图 9 所示。

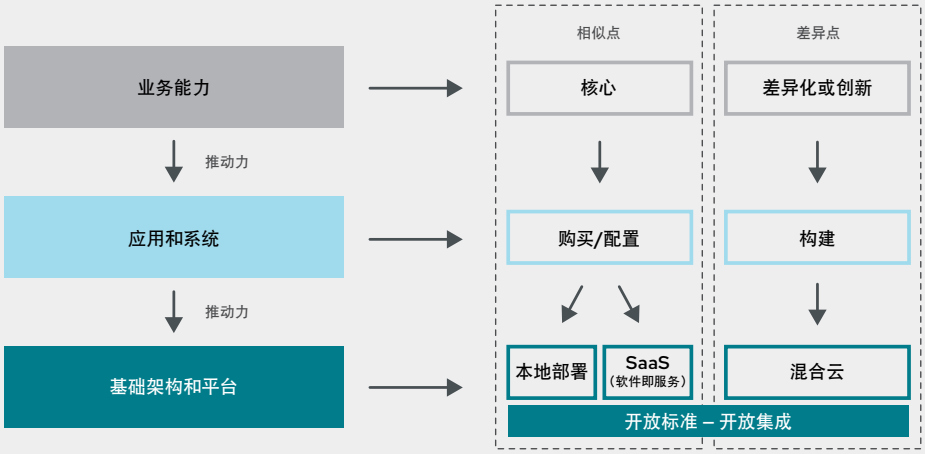


图 9. 按相似点和差异点划分应用可帮助团队确定最合理的努力方向

此外，他们还确定了哪些应用需要迁移以及何时建立起对新流程的信心。每项应用和流程更改可能会有所不同，但团队在这个过程中却收获到人们对其决策和执行力的信任。

图 10. 在红帽 IT 所采用的混合云架构中使用的并行平台

红帽采取的最终策略是针对不同目的在公司内部部署平台，而不仅仅是为红帽 IT 服务的单一平台。在红帽 IT 所采用的混合云中，部署了三种并行平台。除业务关键型托管平台外，UpShift 平台支持将红帽产品和技术用于其内部流程，而开放平台则允许整个公司中的任何员工构建新的应用。

通过部署在产品和技术部门内部使用的环境，公司可以先行采用自己的创新版本。通过赋予员工创建应用的权力，公司可以获得有关可用 IT 功能的可见性和反馈。

红帽的许多员工都是技术出身，但也有一些并没有相关背景。它的 IT 系统让非技术用户也可以在自己的域内进行创新。

有关红帽拥抱变革的组织框架的更多详细信息，请参阅*《开放式组织》*。

有可用的方法吗?

本电子书没有提供正式的方法，因为很难以这种方式讨论大型 IT 的主题。相反，本电子书的目标是列举与混合云和云原生技术对企业的影响方式相关的许多关键策略领域，并将它们汇总在一起。

把企业的 IT 系统视为一个单一应用环境，并牢记下面这些策略问题，就能让您重点关注这些注意事项:

- 哪些策略维度会带来最大收益?
- 如何在一个领域或在整个企业内传播好的做法?
- 如何将主要类型的平台（操作系统、虚拟化、容器）与应用（自定义代码、集成、流程驱动型应用）融合在一起?
- 如何将所有这些概念与开发人员生产力和运维稳定性相关联?

有关这些主题的详尽分析或对影响最大的领域的评估，请参阅下一节，了解红帽服务团队运作的各种计划。

第五部分：总结与后续步骤

如今，人们对 IT 系统的重视前所未见。但是，如果没有可同时解决可靠性和生产力问题的云原生敏捷策略，企业将越来越难以在市场上竞争。尽管之前的 IT 时代已经实现了巨大的创新，但在过去的几年间，企业前所未有地要依赖于 IT 来支撑几乎所有业务功能。后台和前台系统必须以空前的方式协同工作。

为了在这种环境中取得成功，企业需要一个健康、可靠且高效的应用环境来为创新奠定基础。该应用环境需要跨越多个数据中心和云端，同时还要采纳最新、最高效的云原生技术。

我们的目标是整合从客户身上和技术部署中获得的策略见解，以帮助其他人制定计划和策略。

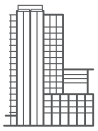
尽管并非所有见解都适用于每种情况，但希望其中的许多问题和注意事项能对您的 IT 之旅有所帮助。

进一步了解红帽如何在云原生和混合云应用之旅中为您提供支持：

- 了解红帽咨询可以提供哪些帮助：通过业务咨询探讨，获得最佳实践和规划指导。
- 查看我们的服务之声博客，获取相关的洞见信息、技术窍门等。
- 您的 DevOps 成熟度如何？您已为云原生之旅做好了哪些准备？完成 Ready to innovate 评估，明确自己的准备情况。

有关红帽技术产品的更多信息：

- 应用交付：红帽 OpenShift
- 应用开发：红帽中间件
- 开发人员工具：红帽开发人员工具集
- 管理：红帽智能管理



关于红帽

红帽是世界领先的企业开源软件解决方案供应商，依托强大的社区支持，为客户提供稳定可靠而且高性能的 Linux、混合云、容器和 Kubernetes 技术。红帽帮助客户集成现有和新的 IT 应用，开发云原生应用，在业界领先的操作系统上开展标准化作业，并实现复杂环境的自动化、安全防护和管理。凭借一流的支持、培训和咨询服务，红帽成为《财富》500 强公司备受信赖的顾问。作为众多云提供商、系统集成商、应用供应商、客户和开源社区的战略合作伙伴，红帽致力于帮助企业做好准备，拥抱数字化未来。

销售及技术支持

800 810 2100
400 890 2100

红帽北京办公地址

北京市朝阳区东大桥路 9 号侨福芳草地大厦 A 座 8 层　邮编：100020
8610 6533 9300

红帽官方微博　红帽官方微信

cn.redhat.com
#F19145_0919