发布者：红帽开发人员

关于在 DevSecOps 中设置供应链安全防护的开发人员指南

在软件开发初期构建安全防护的 5 个步骤

Collin Chau、Dash Copeland 和 Markus Eisele

关于在 DevSecOps 中设置供应链安全防护的开发人员指南

在软件开发初期构建安全防护的 5 个步骤

DevSecOps 是软件开发领域一个相对较新的术语，随着越来越多的企业认识到将安全实践融入软件开发流程的重要性，这一术语正迅速流行起来。DevSecOps 将 DevOps 的原则（强调开发团队与运维团队之间的协作和自动化流程）与安全防护实践相结合，在软件开发生命周期内营造一种安全文化。

开发人员在实施 DevSecOps 实践方面发挥着至关重要的作用，因为他们负责编写构成软件的代码。不过，许多开发人员并不具备深厚的安全防护背景，而且有可能并不了解构建安全防护软件的最佳实践。

本指南为开发人员介绍了 DevSecOps，包括作为 DevSecOps 团队的一员构建安全防护软件所需了解的关键原则、工具和技术。

安全防护？真的吗？开发人员的观点

软件开发人员非常重视软件安全防护，并且认识到制定策略以确保其所开发软件的安全性的重要性。红帽在 2023 年开展的一项内部调研发现，软件开发人员未得到满足的首要需求是“最大限度地降低部署包含安全漏洞的应用的可能性”。

然而，确保软件安全并防范威胁会给开发过程增加额外的复杂性，这对已经不堪重负的开发人员来说可能是一种负担。作为开发人员，您必须时刻注意潜在的漏洞，并实施安全的编码实践，如输入验证、数字签名、数据加密和访问控制。这项工作通常非常耗时，可能会减慢开发进程，而且在面临快速交付软件的压力时，会带来沮丧感。

许多开发人员没有深厚的安全防护背景，甚至可能不熟悉最新的安全威胁和最佳实践。这可能会给实施有效的安全防护措施带来挑战，并增加出现安全漏洞的风险。

此外，软件安全防护是软件开发生命周期中的一个持续过程，需要定期更新和维护，以长期确保软件的安全。这种维护会给开发人员繁忙的日程增加更多的工作，从而导致倦怠和对工作的不满。

软件安全防护可能既复杂又耗时，而且导致无法集中精力创建有助于提升业务价值并让用户满意的功能软件。尽管如此，保护软件免受安全威胁并确保软件对用户安全可靠仍然至关重要。而在现代实践中，安全主题并不仅仅停留在源代码或非功能性需求上。它贯穿整个软件供应链。

应对软件供应链攻击

软件供应链（SSC）是指开发、使用和分发软件的过程。这可能涉及众多的参与方、软件组件和依赖项，从而使其容易受到安全风险的影响。

无论是使用第三方库还是分发自己的库，只要存在漏洞，就可能会带来深远的后果，包括经济损失、声誉损害以及将削弱客户信任的法律责任。仅仅实施安全开发实践并遵守既定的策略来防范这些风险是不够的，尤其是在我们无法直接控制、审核和保护项目开始时使用的源代码和传递性依赖项的情况下。

“从品牌角度来看，可能会造成巨大的损失，无论是人员信息泄露，还是失去大众信任。”

供应链攻击可能不被察觉，并对软件生产者和使用者造成不可估量的影响。这种缺乏可见性的情况可能会导致受损代码大量传播，并发生一系列的漏洞或滥用。因此，除了保护应用的独立组件之外，开发人员还应该锁定并保护其软件工厂的所有数字入口点。专注于软件供应链的某一方面的做法既不具备可扩展性，也不足以满足需求。

“它应该是全面的……除了应用和您正在使用的软件包，还要检查是否有任何不应该开启的端口，或者是否有一些本应为私有状态的内容被公开了。”

红帽的集中式平台整合了安全防护与合规性功能，可帮助 DevOps 和安全防护团队共同满足 DevSecOps 要求，其目标是帮助在开发和生产过程中确保云原生应用的安全。

凭借 30 年来在构建值得信赖的产品和软件包，并将其安全交付到企业所依赖的企业软件中的经验，红帽定能帮助开发人员了解并应对供应链安全防护。随着我们将用于构建和交付可信内容的软件供应链提供给开发人员和安全防护团队，我们认识到这是软件开发过程中所有相关方的共同责任。

为什么安全防护和安全供应链如今变得更加重要

数字化转型仍在以迅猛的速度进行，这使企业高管承担起更大的责任，以满足实现全面数字化客户体验的新需求。随着每家企业现在都成为软件驱动型公司，技术领导者被寄予厚望，期待他们能够通过迁移到云端来实现新的业务成果，如灵活性和大规模扩展。然而，许多企业在其复杂的混合 IT 环境中难以保持稳定的安全性和性能，导致软件工厂的转型工作停滞不前。

企业正面临着非常活跃的对手，他们资金充足，渴望利用不断扩大的威胁面。随着每一起安全事件的发生，网络攻击的收益也随之激增，这导致分布式拒绝服务攻击、勒索软件、零日漏洞等问题层出不穷。应对安全挑战比以往任何时候都更加困难，因为恶意行为者发起网络攻击的频率和复杂程度持续急剧增加。

孤立的通信、不兼容的接口以及太多不同产品之间缺乏标准化，显著降低了整个应用开发和部署系统的安全效率。企业需要支持 DevSecOps 实践的集成式安全工具和流程，部分原因如下：

- **安全开发的复杂性增加。**虽然开发人员了解编写安全、合规的代码的重要性，但在 DevSecOps 驱动的现代世界中，解决这些问题的复杂性正呈指数级增长。根据红帽内部的安全成果研究，开发人员对满足这些需求的能力越来越不满。

> “很难通过一个位置来了解全局，比如：是否一切都处于最新状态且没有漏洞？我的意思是，Google Cloud 本身就非常庞大。因此，使用这些不同的产品和类似的资源可能难以管理。我想，确保我们使用的每个产品都没有漏洞是很重要的。”

- **开源组件的使用日益增多。**许多企业使用开源组件来开发软件，而这些组件的安全性可能会被试图渗透供应链的攻击者破坏。实现企业级开源就绪至关重要。根据红帽的调研，三分之二的企业表示他们目前构建新的应用正使用开源软件（OSS）来增强内部开发，其余的企业也计划在未来采取这一做法。[1]

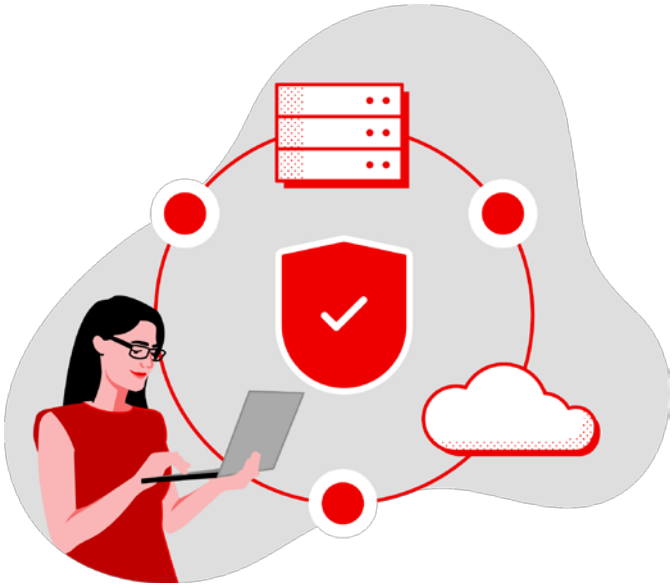
- **每天都会发现新的攻击途径。**企业越来越依赖第三方工具和服务依赖项。如果供应商没有在前期对其软件进行适当的安全防护，这必然会在开发生命周期的早期引入安全风险并影响应用的发布。最近的一项研究指出，在每月下载的超过 12 亿个依赖项中，每 7 个项目漏洞中就有 6 个来自传递性依赖项。[2]

> “所有软件都是基于许多不同的软件包进行组装和构建。这是另一个需要跟踪数百个不同软件包的地方，我们下载这些软件包并用于安装，一定希望有良好的保护措施。但现在的情况是，您无法手动检查每一个提供商，然后检查提供商的每一个依赖项。”

- **更严格的监管要求。**[3,4] 在许多受严格数据监管和合规要求约束的行业中，现在都要求企业在其软件供应链中实施安全措施，以实现更强大的网络弹性，包括跟踪软件中的所有代码依赖项。过去 3 年中，软件供应链攻击事件的年均增长率达到了惊人的 742%。[5] Gartner 预测，到 2025 年，全球将有 45% 的企业的软件供应链遭受攻击，比 2021 年增加三倍。[6]

1 “强强联合：DevOps 与开源齐头并进”，IDC Perspective，2022 年。
2 “第 8 次软件供应链状况年度报告”，Sonatype。
3 白宫关于改善国家网络安全的行政命令（EO 14028）。
4 欧盟委员会提出的《网络弹性法案》（CRA）。
5 Sonatype，“第 8 次软件供应链状况年度报告”。
6 Gartner，“2022 年网络安全七大趋势”。

通过确保软件供应链的安全，企业可以降低网络攻击、数据泄露和其他安全事件的风险，从而减少财务损失、声誉损害以及对最终用户的伤害。实施软件供应链安全防护最佳实践的企业可以提高其软件的整体质量和安全性，并有助于保护其系统和数据免受攻击。



DevSecOps 策略的重要性

对软件供应链安全的威胁迫使 DevOps 实践发生巨大变革，转而采用 DevSecOps 战略，将安全防护作为软件开发生命周期的一个基本且持续的方面。[7] 软件工程领导者在软件开发生命周期的早期就关注软件组件和依赖项的安全性，从而降低软件供应链的风险。他们在软件工厂的每个阶段都执行集成式安全网关，以实现一致、可重复且自动化的运维。

整个市场正朝着能够快速、安全、持续地部署出色软件体验的应用平台方向发展，而这正是企业竞争的优势所在。但现实情况是，企业在运行这些并行任务时往往会遇到困难。他们面临的挑战包括：

- 维护和改进传统应用及基础架构非常复杂，对本已有限的 IT 资源造成了压力。
- 使用现代框架和云原生应用架构构建和运行全新的应用会增加开发团队的认知负担。
- 安全防护往往是事后才想到的问题，由安全防护和 IT 运维团队在应用开发生命周期的最后阶段处理，而且几乎不与应用开发和其他团队协作。
- 不同的应用安全防护和 DevOps 工具、实践及脱节的流程导致工具泛滥；这阻碍了协作、可见性和生产力，并增加了出现人为错误的几率。

因此，企业往往无法及早发现安全问题，而此时修复这些问题更容易，成本也更低。这增加了安全漏洞的风险，并阻碍了应用开发和交付的速度和效率。

7 软件工程领导者如何降低软件供应链安全风险，Gartner，2021 年。

适用于开发人员的 DevSecOps 最佳实践

软件供应链安全防护和 DevSecOps 都是优先考虑安全性的重要软件开发方法，但它们侧重于软件开发流程的不同方面：

- **安全软件供应链**侧重于软件在整个供应链中的安全性。
- **DevSecOps** 强调通过安全编码实践以及对安全性的持续监控和改进来构建安全软件。

因此，两者缺一不可。对于开发人员来说，必须在软件开发过程中采用 DevSecOps 实践，并将这些实践建立在安全软件供应链的基础上。让我们简要了解一下对开发人员最重要的最佳实践。

尽早并经常实施安全防护

安全防护应从一开始就纳入软件开发流程，并且在开发生命周期的每个阶段都不容忽略。这包括安全测试、漏洞扫描、代码分析以及安全编码实践。

“在安全方面没有太多的自动化功能和检查。我感兴趣的是自动化，这样我们就可以进行手动检查，但任何能帮助我们尽早发现问题的方法都是更可取的。”

尽可能实现安全自动化

自动化可以简化安全测试和审查流程，使其更加高效和有效。例如，自动漏洞扫描、自动测试以及持续集成和部署管道。

强调开发、安全防护和运维团队之间的协作

DevSecOps 基于企业内不同部门之间的协作和团队合作原则。通过合作，团队可以更快、更有效地识别和解决安全问题。

使用安全编码实践

输入验证、数据加密和访问控制等安全编码实践有助于防止软件漏洞。开发人员应接受安全编码实践方面的培训，并遵循既定的编码标准，以确保所编写代码的安全性。

定期进行安全评估

定期安全评估有助于发现软件中的潜在漏洞和安全问题。这可能包括渗透测试、代码审查和漏洞扫描等技术。

持续监控和提高安全性

安全防护是一个持续过程，必须定期监控和更新软件，以确保其始终安全。这包括监控新的安全威胁，根据需要更新软件，以及在出现安全问题时及时处理。通过持续监控和提高安全性，企业可以降低安全漏洞的风险，并保护其软件和数据。

在 DevSecOps 中设置供应链安全防护：应许之地

软件供应链安全防护是一个复杂而多方面的领域，需要采用全面的 DevSecOps 方法才能取得成效。成功的 DevSecOps 实践的核心源于，企业如何确保软件供应链中开源软件的安全。

由于缺乏对开源代码及其第三方依赖项就漏洞进行持续监控和测试，在早期就引入了安全风险。开发和安全团队无法跟踪和执行急需的安全维护活动，以确定服务事件的优先级并加以预防。

红帽提高了供应链的弹性，与企业的创新周期保持同步，从而在满足安全要求的同时更快地发布应用。作为一项云服务，红帽可信软件供应链提供了一个 DevSecOps 框架，可据此持续编码、构建和监控可信软件供应链。只需点击几下，我们就可以在软件开发生命周期的每个阶段集成安全防护措施，确保在应用中安全使用开源软件和第三方依赖项，从而更快地实现价值。

开发人员非常重视用户的信任，[8] 我们希望让他们在没有任何额外开销的情况下取得成功。红帽的解决方案可以帮助开发人员在整个软件开发生命周期中实现安全左移（图 1）。

图 1：在从代码到生产的软件开发生命周期的早期阶段，确保软件组件和依赖项的安全。

8 开发人员安全成果发现报告（红帽，2023 年 1 月）。

实现成功的 DevSecOps 实践

要成功实施 DevSecOps，早在进入应用管道之前就应开始相关工作。首先，企业需要确保其底层基础架构和应用服务在企业级开源基础上运行，并预先通过内置的安全工具和功能加以强化。红帽的平台由专门的产品安全团队管理，该团队负责监控、识别并快速解决漏洞。持续的安全更新会交付到安全增强型 Linux 容器，并通过安全渠道分发。

开发人员需要对基于云的应用的各个方面进行安全扫描和指导。除了软件包，他们还需要对工具、应用配置和整个解决方案架构（包括基础架构）进行安全检查。

开发人员还需要能够灵活地将工作负载转移到与使用选项最契合的任何空间，以满足企业对开放混合云的需求。基于值得信赖、经过行业验证的容器编排平台而构建可以增加标准化和一致性方面的优势，从而继续投资于 Kubernetes 原生 Java 框架（如 Quarkus）等方面。

您可以通过工具、库和扩展的组合，为现代云原生世界安全地大规模创建应用，并获得顺畅的开发体验。基于流行的集成开发环境（IDE）工具，团队可以通过使用本地机器上的集中式开发人员工作区，在几分钟内在 Kubernetes 上贡献代码。在整个软件工厂使用相同的单点登录（SSO）工具，以确保开发工作区和源代码访问的安全。

1. **获取开箱即用的镜像和库** 通过使用流行应用框架（包括 Java、Node.js、Python、Go）和红帽企业 Linux（RHEL）软件包中库形式的可信内容，随时掌握最新的漏洞和安全风险。根据最佳实践实现内容标准化，以减少团队的认知负担，从而促进相互关联的软件工厂之间的共同协调和更轻松的协作。在软件开发生命周期（SDLC）中使用已根据红帽最佳实践扫描和强化的 OSS 内容，最大限度地缓解风险状况。

在安全性方面，开发人员希望做正确的事情。他们希望有卓越的专业表现，但更注重赢得并保持用户的信任。开发人员可以访问大量的第三方软件库、框架、API 和工具（这些资源都得到了积极维护，符合企业的安全标准），从而专注于构建新功能，而无需对每个依赖项和工具进行评估并确保其安全。[9]

2 维护一个高度可用的容器镜像仓库，以便从中安全地访问和整合经过验证且精心策划的软件包

使用基于角色的精细访问权限控制（RBAC）限制对容器镜像仓库和其中存储的镜像的访问，以降低未经授权进行访问的风险。安全存储和管理用于部署应用和服务的镜像，确保生产中仅使用可信的镜像。运行无根容器镜像，在容器内安全地安装软件包和运行服务，而不会影响主机。

提高软件工厂的透明度和可见性，以在安全团队和 DevOps 团队之间建立信任。允许通过镜像签名进行验证和身份验证，这有助于防止恶意代码添加到镜像仓库中。验证软件物料清单的真实性并防止篡改，以确保代码的完整性。支持使用数字签名和证书，以证明软件组件来自可信来源。使用软件构件的供应链级别（SLSA）标准来显示构建和管道的软件出处，以减少伪造和恶意修改的风险。

务必要认识到，开发人员有时需要访问系统来解决安全问题。在规模较大的企业中，必须建立 RBAC 和 GitOps 等信任隔离机制，以执行特权访问。此外，他们还需要所有特权会话的可审核记录。[10]

9 开发人员安全成果发现报告（红帽，2023 年 1 月）。
10 开发人员安全成果发现报告（红帽，2023 年 1 月）。

3 使用安全防护最佳实践保护代码管理中的源代码和依赖项

分析并检测潜在的漏洞、恶意软件或其他恶意代码，以免用于软件工厂。在将镜像提交到代码存储库之前，利用自动代码分析功能扫描其中的潜在安全漏洞和其他安全问题。您需要仔细管理依赖项，并定期审核构建过程中使用的任何库或组件是否存在漏洞。组件分析可帮助企业识别和评估软件供应链中第三方组件的风险。

许多开发者认为，他们不应部署存在严重漏洞的任何内容，因此及早发现严重漏洞以确定是否能够部署到生产环境是非常重要的。开发人员希望更主动地检测并尽早发现漏洞，但也需要专注于编写代码并快速交付。随着安全检查和修复流程实现自动化，您无需手动跟踪漏洞，并能轻松遵循安全防护最佳实践。[11]

通过无处不在的加密签名来防止篡改代码，并通过所有活动的公开、不可变的开源日志自动记录每次提交。确保代码和构件安全地存储在由强大的身份和访问管理（IAM）策略保护的版本控制系统中。通过实施加密和安全备份系统，将源代码安全地保存在一个定期审核漏洞并防止外部威胁的存储库中。

11 开发人员安全成果发现报告（红帽，2023 年 1 月）。

4 通过自动化信任链和审批网关来加强 CI/CD 管道控制软件依赖项流，确保在构建和部署过程中仅使用受信任的软件包，以防止在软件工厂中执行遭病毒入侵的管道。首先自动生成软件物料清单（SBOM），并提供有关每个构件的构建方式的元数据，以管理和保护构建过程中使用的各种软件组件。通过对开发过程中使用的所有软件组件进行版本控制、审核和追溯，按照行业标准验证出处。

实现 CI/CD 管道自动化，并在整个构建过程中集成定期安全检查，以确保团队在编译代码、构建镜像和运行测试时所有输入和输出的安全性。建立强有力的保护措施，防止通过交叉构建污染进行篡改。立即检测并警告对源代码和 OSS 依赖项的任何更改或未经授权的修改，这些更改或修改会影响存储库中的构件。确定在任何特定应用中使用的组件和版本，并了解该变更的影响，以降低 SDLC 中的风险。

允许企业定义和执行企业合同，规定软件供应链中软件组件的可接受使用和行为。将发布策略作为代码持续部署到声明性状态，以确保自动检测错误配置，并暂停和回滚部署。持续实施安全扫描，包括二进制分析，以检查和识别编译代码和镜像中的常见漏洞和暴露（CVE）以及病毒。触发自动修复措施以防止利用这些安全漏洞。

开发人员需要一个能自动跟踪和维护依赖项的全面而准确的清单，以确保应用的安全。依赖项的管理规模庞大，因此手动操作并不可行。一个应用由数百个不同的软件包组成，而每个软件包都可能通过其依赖项而引入安全漏洞。这些依赖项往往会阻碍开发人员安装必要的软件包更新。因此，开发人员需要帮助跟踪依赖项并保持不断更新。[12]

12 开发人员安全成果发现报告（红帽，2023 年 1 月）。

5 在运行时监控应用程序，深入了解已部署工作负载的漏洞和威胁

通过实施适当的访问控制、威胁预防和异常检测、网络分段以及运行时漏洞检测，确保部署环境在运行时的安全性。提供对所有组件及其各自来源的完整端到端可见性，以持续监控并主动识别恶意组件导致的风险状况变化。实施监控和日志记录系统，即时检测、警报和指导处理潜在的安全事件。这包括详细的报告和分析，以帮助企业了解其风险态势并做出明智的决策。

开发人员需要根据受影响软件包的使用方式来确定安全漏洞的影响。仅凭严重程度不足以做出决定。他们需要有关软件包使用方式和应用部署环境的上下文信息，以便确定其真正的重要性。开发人员应该可以通过一个位置来轻松跟踪和管理整个应用架构中的漏洞。[13]

支持展开合规性检查和审查工作，以确保符合针对软件工厂的安全评估和联邦政府网络安全指令。在跨可用区连接服务和向正确的用户传输数据时，采取措施以提高 API 安全性。同时，持续保护数据在静态、传输或运行时的完整性和保密性。使用 API 管理、加密通信和安全认证等安全技术，防止在部署期间未经授权访问敏感数据和系统。

13 开发人员安全成果发现报告（红帽，2023 年 1 月）。

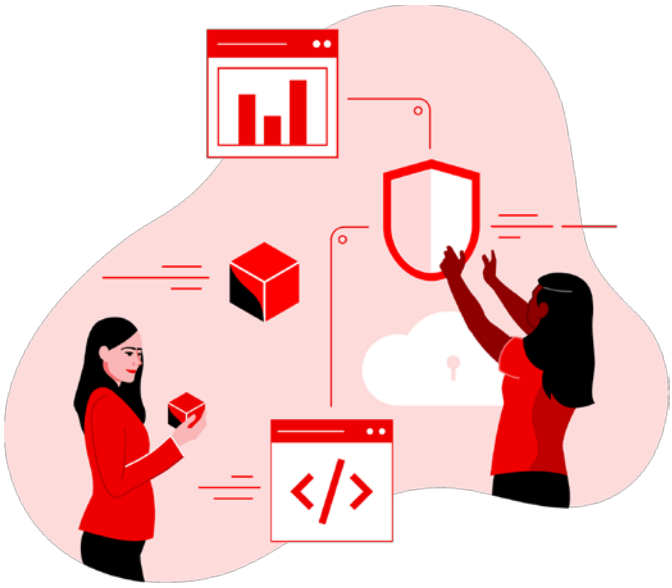
红帽如何支持以安全防护为重点的云原生开发

使用支持软件供应链安全防护策略的开源软件，无论是在本地部署环境还是跨混合云和多云环境。红帽的针对软件供应链安全策略采取的 DevSecOps 方法提供了以下优势。

- **利用集成的安全工具生态系统来识别和评估与软件供应链相关的潜在风险。**整合标准化的精选服务，并通过一个集成式自助服务门户网站进行使用。红帽基于其 30 年的企业 Linux 经验和 7 年的企业 Kubernetes 经验提供了安全防护措施，并提供了一种主动方法来通过 DevSecOps 工具确保软件供应链的安全，帮助您在满足运维安全要求的同时快速创新。
- **有了安全的软件供应链，客户和用户就可以更加信任他们正在使用的软件。**降低将漏洞和威胁引入软件的风险可以增强客户忠诚度和品牌声誉。您还可以更快地发布新的软件功能和更新，以跟上不断变化的客户偏好。
- **通过实施软件供应链安全防护解决方案，更好地遵循行业法规和标准。**红帽可以帮助企业避免因不合规而遭受高昂的罚款和处罚，同时提高软件的整体质量。这样，软件就会更加稳定可靠，我们也能提前发现影响用户的安全问题。
- **开发人员需要对基于云的应用的各个方面进行安全扫描和指导。**除了软件包，您还需要对工具、应用配置和整个解决方案架构（包括基础架构）进行全面的安全检查。此外，您还需要高度信任数据的准确性。它应该与多个安全数据源进行交叉引用，以实现全面的安全策略。[14]

14 开发人员安全成果发现报告（红帽，2023 年 1 月）。

红帽与其合作伙伴一起提供工具和服务，以构建一个全面的 DevSecOps 生态系统，并提供专业知识和能力，为在开放的混合云上构建、部署和运行以安全为重点的应用程序提供强大的产品组合。这样就能改进流程，在不牺牲安全性的情况下加快应用开发，营造协作文化，并降低风险。



了解更多信息

探索红帽开发人员提供的教程、电子书和其他学习资源，为您的软件供应链安全防护策略提供支持：

- DevOps
- DevSecOps
- GitOps
- 安全编码
- 软件供应链安全防护
- 红帽 OpenShift 开发人员沙盒

作者简介

Collin Chau 热衷于帮助敏捷开发人员规划、编码、测试、构建和部署优质的数字体验。在那段时间里，他扩大了 DevOps 团队的规模，以进行持续测试和自动化应用发布，同时监控生产环境中 SRE 的 IT 服务运行状况并排除故障，并使 ITOps 团队能够为混合云服务提供桥接和代理。

Dash Copeland 是一名用户体验设计师和研究员，热衷于为技术人员打造出色的体验。他在企业产品领域拥有 10 多年的经验，曾为使用 Linux、容器和 Kubernetes 的合作伙伴和开发人员设计数字体验。

Markus Eisele 是一位 Java 大师、O'Reilly 作家、德国 JavaLand 会议的创始人、世界各地 Java 会议的知名演讲者，而且在企业 Java 领域也享有很高的知名度。

凭借超过 16 年的行业专业经验，他为《财富》500 强公司设计和开发了大型企业级应用。作为一名经验丰富的团队领导者和架构师，他帮助汽车、金融和保险公司实施了一些超大规模的集成项目。