



DoD Enterprise DevSecOps Reference Design

国防部企业 **DevSecOps**

参考设计

Version 1.0 12
August 2019

Department of Defense (DoD)
Chief Information Officer

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖

（前言，附录，全文校对）、周景川（4.3-7 章）、赵

庆安（3 章）排名不分先后

Trademark Information

商标信息

Names, products, and services referenced within this document may be the trade names, trademarks, or service marks of their respective owners. References to commercial vendors and their products or services are provided strictly as a convenience to our readers, and do not constitute or imply endorsement by the Department of any non-Federal entity, event, product, service, or enterprise.

本文档中引用的名称，产品和服务可能是其各自所有者的商标名称，商标或服务标记。对商业供应商及其产品或服务的引用，仅限于方便读者阅读，并不构成或暗示美国商务部对任何非联邦实体，事件，产品，服务或企业的认可。

Executive Summary 文档摘要

Legacy software acquisition and development practices in the DoD do not provide the agility to deploy new software “at the speed of operations”. In addition, security is often an afterthought, not built in from the beginning of the lifecycle of the application and underlying infrastructure. DevSecOps is the industry best practice for rapid, secure software development.

国防部传统的软件采购和开发实践无法为部署新软件提供“以运营速度”敏捷性。此外，安全通常是后置的，而不是从应用和基础结构生命周期的开始就内置。DevSecOps 是用于快速，安全的软件开发的行业最佳实践。

DevSecOps is an organizational software engineering culture and practice that aims at unifying software development (Dev), security (Sec) and operations (Ops). The main characteristic of DevSecOps is to automate, monitor, and apply security at all phases of the software lifecycle: plan, develop, build, test, release, deliver, deploy, operate, and monitor. In DevSecOps, testing and security are shifted to the left through automated unit, functional, integration, and security testing - this is a key DevSecOps differentiator since security and functional capabilities are tested and built simultaneously.

DevSecOps 是一种组织化的软件工厂文化和实践，旨在统一软件开发（Dev），安全性（Sec）和运维（Ops）。DevSecOps 的主要特征是在软件生命周期的所有阶段自动执行，监视和应用安全性，包括规划，开发，构建，测试，发布，交付，部署，操作和监控。在 DevSecOps 中，通过自动化的单元、功能、集成和安全性测试将测试和安全性进行左移，这是 DevSecOps 的主要区别，因为安全性和功能性是同时进行测试和构建的。

The benefits of adopting DevSecOps include:

采用 DevSecOps 的收益包括：

- Reduced mean-time to production: the average time it takes from when new software features are required until they are running in production;
缩短平均投产时间：从软件新功能被确认到投入生产所需的平均时间；
- Increased deployment frequency: how often a new release can be deployed into the production environment;
加快部署频率：多长时间一个新的版本可以被部署到生产环境
- Fully automated risk characterization, monitoring, and mitigation across the application lifecycle;
在整个应用程序生命周期中可进行全自动的风险识别，监控和缓解；
- Software updates and patching at “the speed of operations”.
更快的执行软件更新和打补丁。

This DoD Enterprise DevSecOps Reference Design describes the DevSecOps lifecycle, supporting pillars, and DevSecOps ecosystem; lists the tools and activities for DevSecOps software factory and ecosystem; introduces the DoD enterprise DevSecOps container service that provides hardened DevSecOps tools and deployment templates to the program application DevSecOps teams to select; and showcases a sampling of software factory reference designs and application security operations. This DoD Enterprise DevSecOps Reference Design provides implementation and operational guidance to Information Technology (IT) capability providers, IT capability consumers, application teams, and Authorizing Officials.

此美国国防部-企业 DevSecOps 参考设计描述了 DevSecOps 的生命周期，支撑支柱和 DevSecOps 生态系统。列出了 DevSecOps 软件工厂和生态系统所需的工具和活动；介绍了美国国防部-企业 DevSecOps 容器服务，该服务向应用程序 DevSecOps 团队提供了加固的 DevSecOps 工具和部署模板，并展示了软件工厂参考设计和应用程序安全运维的样本。此参考设计为信息技术（IT）能力提供商，IT 能力使用者，应用团队和授权官员提供了实施和操作指南。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

目录

1	Introduction 导论	7
1.1	Background 背景	7
1.2	Purpose 目的	8
1.3	Scope 范围	9
1.4	Document Overview 文档总览	9
2	Assumptions and Principles 假设和原则	11
2.1	Assumptions 假设	11
2.2	Principles 原则	12
3	DevSecOps Concepts DevSecOps 概念	13
3.1	Key Terms 关键术语	13
3.1.1	Conceptual Model 概念模型	16
3.2	DevSecOps Lifecycle DevSecOps 生命周期	17
3.3	DevSecOps Pillars DevSecOps 的重要支柱	18
3.3.1	Organization 组织	19
3.3.2	Process 流程	20
3.3.3	Technology 技术	22
3.3.4	Governance 治理	23
3.4	DevSecOps Ecosystem DEVSECOPS 生态系统	26
3.4.1	Planning 计划	27
3.4.2	Software Factory 软件工厂	28
3.4.3	Operations 运营	29
3.4.4	External Systems 外部系统	30
4	DevSecOps Tools and Activities DevSecOps 工具和活动	31
4.1	Planning Tools and Activities 规划工具和活动	31
4.2	Software Factory Tools and Activities 软件工厂工具和活动	37
4.2.1	CI/CD Orchestrator CI/CD 编排器	37
4.2.2	Develop 开发	38
4.2.3	Build 构建	42
4.2.4	Test 测试	46
4.2.5	Release and Deliver	54
4.3	Production Operation Tools and Activities 生产运营工具与活动	55
4.3.1	Deploy 部署	56
4.3.2	Operate 运行	59
4.3.3	Monitor 监控	60
4.4	Security Tools and Activities Summary 安全工具与活动概述	64
4.5	Configuration Management Tools and Activities Summary 配置管理工具与活动概述	64
4.6	Database Management Tools and Activities Summary 数据库管理工具与活动概述	66
5	DoD Enterprise DevSecOps Container Service DoD 企业 DevSecOps 容器服务	68
5.1	DoD Enterprise DevSecOps Container Factory DoD 企业 DevSecOps 容器工厂	69
5.1.1	DoD Hardened Containers DoD 已加固容器	69
5.1.2	Container Hardening Process 容器加固流程	70

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

5.2	DoD Centralized Artifact Repository DoD 中心化工件仓库	72
6	DevSecOps Ecosystem Reference Designs DevSecOps 生态系统参考设计	73
6.1	Containerized Software Factory 容器化软件工厂	73
6.1.1	Hosting Environment 托管环境	74
6.1.2	Container Orchestration 容器编排	75
6.1.3	Software Factory Using Hardened Containers 使用已加固容器的软件工厂	76
6.1.4	DoD Applications DoD 应用程序	77
6.2	Software Factory using Cloud DevSecOps Services 使用云 DevSecOps 服务的软件工厂	78
6.3	Serverless Support 无服务器支持	79
6.4	Application Security Operations 应用安全运行	81
6.4.1	Continuous Deployment 持续部署	81
6.4.2	Continuous Operation 持续运行	82
6.4.3	Continuous Monitoring 持续监控	82
6.4.4	Sidecar Container Security Stack 边车容器安全堆栈	84
7	Conclusion	89
	Appendix A Acronym Table 字母缩写表	90
	Appendix B Glossary of Key Terms 附录 B: 关键术语词汇表	92
	Appendix C References 附录 C 参考文档	99

图 1	Containers 容器	15
图 2	Conceptual Model 概念模型	17
图 3	DevSecOps Software Lifecycle 软件生命周期	18
图 4	DevSecOps Pillars 重要支柱	19
图 5	Application DevSecOps Processes 应用 DEVSECOPS 流程	21
图 6	Five Principles of Next Generation Governance 下一代治理的五项原则	25
图 7	Assessment and Authorization Inheritance 评估和授权继承	26
图 8	DevSecOps Ecosystem 生态系统	27
图 9	DevSecOps Software Factory 软件工厂	28
图 10	DoD 企业 DevSecOps 容器服务架构	68
图 11	容器加固流程的主要步骤	70
图 12	中心化软件工厂参考设计	74
图 13	DevSecOps 平台选项	75
图 14	应用程序生命周期中的软件工厂阶段	77
图 15	软件工厂使用云 DevSecOps 服务	79
图 16	Operational Efficiency 运营效率	80
图 17	日志记录与分析流程	84
图 18	边车模式	85
图 19	边车组件	87
图 20	边车容器安全堆栈交互	88
图 21	Hypervisor 和虚拟机	96
表 1	Key Terms 关键术语	13
表 2	Roles of Authorizing Officials in DevSecOps 授权角色	26
表 3	Plan Phase Tools 计划阶段工具	31
表 4	Plan Phase Activities 计划阶段活动	34

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

表 5CI/CD Orchestrator CI/CD 编排器37

表 6 Develop Phase Tools 开发阶段工具38

表 7Develop Phase Activities 开发阶段活动40

表 8 Build Phase Tools 构建阶段工具43

表 9 Build Phase Activities 构建阶段活动44

表 10Test Phase Tools 测试阶段工具46

表 11Test Phase Activities 测试阶段活动50

表 12Release and Deliver Phase Tools 发布和交付阶段工具54

表 13Release and Deliver Phase Activities 发布和交付阶段活动55

表 14 Deploy Phase Tools 部署阶段工具57

表 15 Deploy Phase Activities 部署阶段活动58

表 16Operate Phase Tools 运行阶段工具60

表 17 Operate Phase Activities 运行阶段活动60

表 18 Monitor Phase Tools 监控阶段工具61

表 19 Monitor Phase Activities 监控阶段活动63

表 20 Security Activities Summary 安全活动概述64

表 21 Configuration Management Activities Summary65

表 22 Database Management Activities Summary 数据库管理活动概述67

表 23 Sidecar Container Security Stack Components 边车容器安全堆栈组件87

1 Introduction 导论

1.1 Background 背景

DevSecOps is an organizational software engineering culture and practice that aims at unifying software development (Dev), security (Sec) and operations (Ops). The main characteristic of DevSecOps is to improve customer outcomes and mission value by automating, monitoring, and applying security at all phases of the software lifecycle: plan, develop, build, test, release, deliver, deploy, operate, and monitor. Practicing DevSecOps provides demonstrable quality and security improvements over the traditional software lifecycle, which can be measured with these metrics:

DevSecOps 是一种组织软件工程文化和实践，旨在统一软件开发（Dev），安全（Sec）和运营（Ops）。**DevSecOps** 的主要特征是在软件生命周期的各个阶段通过自动化，监视和应用安全化来提高客户成果和使命价值：包括计划，开发，构建，测试，发布，交付，部署，操作和监视。实践 **DevSecOps** 能够提供更好的质量和安全性改进相比传统软件开发周期，可以通过以下指标进行衡量：

- Mean-time to production: the average time it takes from when new software features are required until they are running in production.
生产间隔时间：从需要新的软件功能到开始生产所需的平均时间。
- Average lead-time: how long it takes for a new requirement to be delivered and deployed.
平均交货时间：新需求交付和部署需要多长时间。
- Deployment speed: how fast a new version of the application can be deployed into the production environment.
部署速度：将新版本的应用程序部署到生产环境中的速度。
- Deployment frequency: how often a new release can be deployed into the production environment.
部署频率：新版本可以多久部署到生产环境中一次。
- Production failure rate: how often software fails during production.
生产失败率：生产期间软件发生故障的频率。
- Mean-time to recovery: how long it takes applications in the production stage to recover from failure.
平均恢复间隔时间：生产阶段的应用程序从故障中恢复需要多长时间。

In addition, DevSecOps practice enables:

此外，**DevSecOps** 实践还可以：

- Fully automated risk characterization, monitoring, and mitigation across the application lifecycle.
在整个应用程序生命周期中进行全自动风险特征描述，监控和缓解。
- Software updates and patching at a pace that allows the addressing of security vulnerabilities and code weaknesses.
软件更新和修补以允许解决安全漏洞和代码漏洞的步伐进行。

DevSecOps practice enables application security, secure deployment, and secure operations in close alignment with mission objectives. In DevSecOps, testing and security are shifted to the left through automated unit, functional, integration, and security testing - this is a key DevSecOps differentiator since security and functional capabilities are tested and built simultaneously. In

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

addition, some security features are automatically injected into the application without developer intervention via a sidecar container.

DevSecOps 实践可实现与任务目标紧密一致的应用程序安全性，安全部署和安全操作。在 **DevSecOps** 中，通过自动化的单元，功能，集成和安全性测试将测试和安全性进行左移-这是 **DevSecOps** 的关键区别，因为安全性和功能性是同时进行测试和构建的。此外，一些安全功能会自动注入到应用程序中，而无需开发人员通过边车类型的容器进行干预。

1.2 Purpose 目的

The main purpose of this document is to provide a logical description of the key design components and processes to provide a repeatable reference design that can be used to instantiate a DoD DevSecOps software factory.

本文档的主要目的是提供关键设计组件和流程的逻辑描述，以提供可重复使用的参考设计，该参考设计可用于实例化 **DoD DevSecOps** 软件工厂。

The target audiences for this document include:

本文档的目标读者包括：

- DoD Enterprise DevSecOps capability providers who build DoD Enterprise DevSecOps hardened containers and provide a DevSecOps hardened container access service
国防部企业 **DevSecOps** 功能提供商，这些提供商构建 **DevSecOps** 加固的容器和提供 **DevSecOps** 更坚强的容器访问服务
- DoD organization DevSecOps teams who manage (instantiate and maintain) DevSecOps software factories and associated pipelines for its programs
国防部组织 **DevSecOps** 团队，负责管理（实例化和维护）**DevSecOps** 软件工厂及其程序的相关流水线
- DoD program application teams who use DevSecOps software factories to develop, secure, and operate mission applications
使用 **DevSecOps** 软件工厂开发，安全和运营任务应用程序的国防部应用系统团队
- Authorizing Officials (AOs)
授权的官员

The DoD Enterprise DevSecOps reference design leverages a set of hardened DevSecOps tools and deployment templates that enable DevSecOps teams to select the appropriate template for the program application capability to be developed. For example, these templates will be specialized around a specific programming language or around different types of capabilities such as web application, transactional, big data, or artificial intelligence (AI) capabilities. A program selects a DevSecOps template and toolset; the program then uses these to instantiate a DevSecOps software factory and the associated pipelines that enable Continuous Integration and Continuous Delivery (CI/CD) of the mission application.

国防部企业 **DevSecOps** 参考设计了一组经过强化的 **DevSecOps** 工具和部署模板，这些工具和部署模板使 **DevSecOps** 团队可以为要开发的程序应用程序能力选择适当的模板。例如，这些模板将围绕特定的编程语言或不同能力类型，例如 **Web** 应用程序，事务性，大数据或人工智能（**AI**）能力。程序选择一个 **DevSecOps** 模板和工具集。然后，程序使用它们来实例化 **DevSecOps** 软件工厂和关联的流水线，以启用任务应用程序的持续集成和持续交付（**CI / CD**）。

This reference design aligns with these reference documents:

该参考设计与这些参考文档一致：

- DoD Cloud Computing Strategy [1]
国防部云计算战略[1]

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

- DoD Cloud Computing Security Requirements Guide [2]
[DoD 云计算安全要求指南 \[2\]](#)
- DoD Secure Cloud Computing Architecture (SCCA) [3]
[国防部安全云计算架构（SCCA）\[3\]](#)
- Presidential Executive Order on Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure (Executive Order (EO) 1380) [4]
[总统行政命令，关于加强联邦网络和关键基础设施的网络安全（执行命令（EO）1380）\[4\]](#)
- National Institute of Standards and Technology (NIST) Cybersecurity Framework [5]
[美国国家标准技术研究院（NIST）网络安全框架\[5\]](#)
- DoD Container Hardening Security Requirements Guide [6].
[DoD 容器加固安全要求指南\[6\]](#)。

1.3 Scope 范围

This document describes the reference design to enable DevSecOps to scale across the department. DevSecOps is an established mature capability in industry, and it is already used within some pockets of the Government; this reference design formalizes its usage across the DoD. This design is product agnostic and provides execution guidance for use by software teams. It is applicable to developing new capabilities and to sustaining existing capabilities both business and weapons systems software, including business transactions, C2, embedded systems, big data, and Artificial Intelligence (AI).

[本文档描述了 DevSecOps 的参考设计相关内容以便能够在整个部门范围内扩展。](#)

[DevSecOps 是行业中已建立的成熟功能，已经使用在政府的一些地方；该参考设计将其在国防部的使用正式文档化。此设计与产品无关，仅提供执行指导以供软件团队使用。它适用于开发新的能力并维持商业和武器系统软件的现有能力，包括商业交易，C2，嵌入式系统，大数据和人工智能（AI）。](#)

This document does not address policy or acquisition.
[本文档不涉及政策或采购。](#)

1.4 Document Overview 文档总览

The documentation is organized as follows.
[该文档的内容如下。](#)

- Section 1 describes the background, purpose and scope of this document.
[第 1 节介绍了本文档的背景，目的和范围。](#)
- Section 2 describes the assumptions made in developing this reference design, as well as stating foundational principles.
[第 2 节描述了在开发此参考设计时所做的假设，并阐述了基本原理。](#)
- Section 3 describes the DevSecOps lifecycle, the four pillars to assist DevSecOps adoption, and a technical architecture of the DevSecOps software factory and its ecosystem.
[第 3 节描述了 DevSecOps 的生命周期，协助采用 DevSecOps 的四个支柱以及 DevSecOps 软件工厂及其生态系统的技术架构。](#)
- Section 4 describes the DevSecOps ecosystem tools and the activities along software lifecycle phases.
[第 4 节介绍了 DevSecOps 生态系统工具以及软件生命周期各个阶段的活动。](#)
- Section 5 describes the DoD Enterprise DevSecOps Service. The target audience of this

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

section is DoD Enterprise DevSecOps capability providers.

第 5 节介绍了 **DoD Enterprise DevSecOps** 服务。本部分的目标受众是 **DoD Enterprise DevSecOps** 功能提供商。

- Section 6 describes the reference design for DoD programs to build their DevSecOps software factory and ecosystem

第 6 节介绍了 **DoD** 程序的参考设计，以构建其 **DevSecOps** 软件工厂和生态系统。

2 Assumptions and Principles 假设和原则

2.1 Assumptions 假设

This document makes the following assumptions:

本文档进行以下假设：

- For most organizations, deploying to a certified and monitored cloud environment will become their preferred solution technically and culturally.
大多数组织，从技术和文化角度而言，部署一个经过认证和受监控的云环境将成为他们的首选解决方案。
- Rapidly changing technology dictates designing the DevSecOps pipelines and patterns for flexibility as new development capabilities enter/exit the commercial product market.
迅速变化的技术要求随着新的开发能力进入/退出商业产品市场而设计 **DevSecOps** 流水线和模式以实现灵活性。
- The DoD Enterprise DevSecOps software factory is designed to avoid vendor lock-in and leverage Open Container Initiative (OCI) compliant containers and Cloud Native Computing Foundation (CNCF) certified Kubernetes to orchestrate and manage the containers.
国防部企业 **DevSecOps** 软件工厂被设计出来，旨在避免供应商锁定，利用符合开放容器倡议（**OCI**）的容器，以及云原生计算基金会（**CNCF**）认证的 **Kubernetes** 来编排和管理容器。
- The government must balance open source integration risks vs. using pre-integrated Commercial Off-The-Shelf (COTS) products that have vendor “cost of exit” and vendor insider risks.
政府必须平衡开源集成风险与使用具有卖方“退出成本”和卖方内部风险的预集成商业现货（**COTS**）产品。
- It must be possible to host a DevSecOps software factory in any DoD general-purpose cloud environment, as well as in disconnected and classified environments.
必须能将一个 **Devsecops** 软件工厂托管在任何 **DOD** 通用云环境，断开网络连接的和机密性的环境。
- The DevSecOps architecture must have the capability to scale to any type of operational requirement needing a software solution, including:
DevSecOps 架构必须具有扩展能力，以适应需要软件解决方案的任何类型的运营需求，包括：
 - Business systems
业务系统
 - Command and Control systems
指挥与控制系统
 - Embedded and Weapon systems
嵌入式和武器系统
 - Intelligence analysis systems
情报分析系统
 - Autonomous systems
自治系统

- Assisted human operations
辅助人类运营

2.2 Principles 原则

There are several key principles to implementing a successful DevSecOps approach:
实施成功的 DevSecOps 的方法有几个关键原则：

- Remove bottlenecks (including human ones) and manual actions.
消除瓶颈（包括人为的瓶颈）和人工操作。
- Automate as much of the development and deployment activities as possible.
使尽可能多的开发和部署活动自动化。
- Adopt common tools from planning and requirements through deployment and operations.
采用从规划，需求到部署，运营的通用工具。
- Leverage agile software principles and favor small, incremental, frequent updates over larger, more sporadic releases.
充分利用敏捷软件的原理，优先选择较小的，增量的，频繁的更新，而不是较大的，更零散的版本。
- Apply the cross-functional skill sets of Development, Cybersecurity, and Operations throughout the software lifecycle, embracing a continuous monitoring approach in parallel instead of waiting to apply each skill set sequentially.
在整个软件生命周期中应用开发，网络安全和运营等跨职能技能，同时采用连续不断的监视方法，而不必等待依次应用每个技能。
- Security risks of the underlying infrastructure must be measured and quantified, so that the total risks and impacts to software applications are understood.
必须测量和量化基础架构的安全风险，以便了解对软件应用程序的总体风险和影响。
- Deploy immutable infrastructure, such as containers. The concept of immutable infrastructure is an IT strategy in which deployed components are replaced in their entirety, rather than being updated in place. Deploying immutable infrastructure requires standardization and emulation of common infrastructure components to achieve consistent and predictable results.
部署不可变的基础架构，例如容器。不可变基础架构的概念是一种 IT 战略，其中已部署的组件将全部替换，而不是就地更新。部署不可变的基础架构需要对通用基础架构组件进行标准化和仿真，以实现一致且可预测的结果。

3 DevSecOps Concepts DevSecOps 概念

DevSecOps describes an organization^ culture and practices enabling organizations to bridge the gap between developers, security team, and operations team; improve processes through collaborative and agile workflows; drive for faster and more secure software delivery via technology; and achieve consistent governance and control. There is no uniform DevSecOps practice. Each DoD organization needs to tailor its culture and its DevSecOps practices to its own unique processes, products, security requirements, and operational procedures. Embracing DevSecOps requires organizations to shift their culture, evolve existing processes, adopt new technologies, and strengthen governance.

DevSecOps 描述了一个组织的文化和实践，它使组织能够弥合开发人员、安全团队和操作团队之间的分歧；并通过协作和敏捷 workflows 改进流程；通过技术驱动实现更快和更安全的软件交付；并做到一致的治理和控制。目前没有统一的 DevSec 操作实践。每个 DoD 组织需要根据自己独特的流程、产品、安全要求和操作程序调整其文化和 DevSecOps 实践。接受 DevSecOps 则要求组织转变文化，提升现有流程，采用新技术，并加强治理。

This section will briefly discuss the DevSecOps lifecycle, supporting pillars, and the DevSecOps ecosystem.

本节将简要讨论 DevSecOps 生命周期、重要支柱和 DevSecOps 生态系统。

3.1 Key Terms 关键术语

Here are some key terms used throughout the document. Refer to the glossary in Appendix B for the full list.

这里包括了贯穿本文所使用的关键术语

表 1 Key Terms 关键术语

Term 术语	Definition 定义
DevSecOps Ecosystem DevSecOps 生态系统	<p>A collection of tools and process workflows created and executed on the tools to support all the activities throughout the full DevSecOps lifecycle.</p> <p>The process workflows may be fully automated, semi- automated, or manual.</p> <p>基于工具而创建和执行的工具及工作流程的集合，以支持整个 DevSecOps 生命周期中的所有活动。</p> <p>工作流程可以是完全自动化、半自动化或手动化</p>
Software Factory 软件工厂	<p>A software assembly plant that contains multiple pipelines, which are equipped with a set of tools, process workflows, scripts, and environments, to produce a set of software deployable artifacts with minimal human intervention. It automates the activities in the develop, build, test, release, and deliver phases. The software factory supports multitenancy.</p> <p>一个包含多个流水线的软件组装工厂，它配备了一组工具、流程工</p>

	<p>作流、脚本和环境，以在最小的人为干预下产生一组软件可部署工件。 它使开发、构建、测试、发布和交付阶段的活动自动化。 软件工厂支持多租户。</p>
CI/CD Pipeline CI/CD 流水线	<p>The set of tools and the associated process workflows to achieve continuous integration and continuous delivery with build, test, security, and release delivery activities, which are steered by a CI/CD orchestrator and automated as much as practice allows.</p> <p>一组工具和相关的工作流程的集合，以实现完成构建、测试、安全和发布交付活动的持续集成和持续交付，这些活动由 CI/CD 编排器指导，并尽可能实现自动化。</p>
CI/CD Pipeline Instance CI/CD 流水线实例	<p>A single process workflow and the tools to execute the workflow for a specific software language and application type for a project. As much of the pipeline process is automated as is practicable.</p> <p>单个的工作流和工具，用于执行项目中特定软件语言和应用程序类型的工作流。 在实践中，尽可能多的流程是自动化实现的。</p>
Environment 环境	<p>Sets a runtime boundary for the software component to be deployed and executed. Typical environments include development, integration, test, pre-production, and production.</p> <p>设置要部署和执行的软件组件的运行状态边界。典型的环境包括开发、集成、测试、预生产和生产。</p>
Software Factory Artifact Repository 软件工厂工件仓库	<p>A local repository tied to the software factory. It stores artifacts pulled from DoD Centralized Artifact Repository (DCAR) as well as locally developed artifacts to be used in DevSecOps processes. The artifacts include, but are not limited to, virtual machine (VM) images, container images, binary executables, archives, and documentation. It supports multi-tenancy.</p> <p>Note that programs may have a single artifact repository and use tags to distinguish the content types. It is also possible to have separate artifact repositories to store local artifacts and released artifacts.</p> <p>绑定到软件工厂的本地仓库。 它存储了从 DoD 中心工件存储库 (DCAR) 提取的工件以及本地开发的工件，用于 DevSecOps 进程。 工件包括但不限于虚拟机 (VM) 图像、容器镜像、二进制可执行文件、档案和文档。 它支持多租户。</p> <p>注意，多个大型项目可能有一个工件仓库，并使用标签来区分内容</p>

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	类型。 也可以用单独的工件仓库来存储本地工件和发布的工件。
Code 代码	<p>Software instructions for a computer, written in a programming language. These instructions may be in the form of either human-readable source code, or machine code, which is source code that has been compiled into machine executable instructions.</p> <p>计算机的软件说明，用编程语言编写。 这些指令可以是人类可读的源代码，也可以是机器代码，这是已经编译成机器可执行指令的源代码。</p>
Containers 容器	<p>A standard unit of software that packages up code and all its dependencies, down to, but not including the Operating System (OS)_ It is a lightweight, standalone, executable package of software that includes everything needed to run an application except the OS: code, runtime, system tools, system libraries and settings.</p> <p>Several containers can run in the same OS without conflicting with one another.</p> <p>一个标准的软件单元，它将代码及其所有依赖项打包，但不包括操作系统(OS)。它是一个轻量级的，独立的、可执行的软件包，包括运行应用程序所需的一切，除了 OS：代码，运行态，系统工具，系统库和设置。</p> <p>多个容器可以在同一个操作系统中运行，而不会相互冲突。</p>  <p>图 1 Containers 容器</p> <p>Containers run on the OS, so no hypervisor (virtualization) is necessary (though the OS itself may be running on a hypervisor).</p> <p>容器运行在操作系统上。因此，没有必要使用管理程序（虚拟化）(尽管操作系统本身可能正在管理程序上运行)。</p>

	<p>Containers are much smaller than a VM, typically by a factor of 1,000 (MB vs GB), partly because they don't need to include the OS. Using containers allows denser packing of applications than VMs.</p> <p>容器比 VM 小得多，通常是 1000 倍 (MB vs GB)，部分原因是它们不需要包括操作系统。使用容器可以比 VMs 更密集地打包应用程序。</p> <p>Unlike VMs, containers are portable between clouds or between clouds and on-premise servers. This helps alleviate Cloud Service Provider (CSP) lock-in, though an application may still be locked-in to a CSP, if it uses CSP-specific services.</p> <p>不像 VM，容器可在云之间或云与服务器之间移动。这有助于缓解云服务提供商 (CSP) 的锁定情况，尽管如果应用程序使用特定于 CSP 的服务，它仍然可能被锁定到 CSP。</p> <p>Containers also start much faster than a VM (seconds vs. minutes), partly because the OS doesn't need to boot.</p> <p>容器的启动速度也比 VM 快得多 (秒 vs 分钟)，部分原因是操作系统不需要启动。</p>
--	---

3.1.1 Conceptual Model 概念模型

The following conceptual model shows some of the most important concepts described in this paper along with their relationships. It should help to clarify these relationships. When reading text along an arrow, follow the direction of the arrow. So, a DevSecOps Ecosystem contains one or more software factories, and each software factory contains one or more pipelines. The diagram also shows that each software factory contains only one CI/CD Orchestrator, and that many software factories use DCAR

下面的概念模型显示了本文描述的一些最重要的概念及其关系。它有助于澄清这些关系。当沿着箭头阅读文本时，遵循箭头的方向。所以，一个 DevSecOps 生态系统包含一个或多个软件工厂，每个软件工厂包含一个或多个流水线。该图还显示，每个软件工厂只包含一个 CI/CD 编排器，以及众多软件工厂都使用 DCAR。

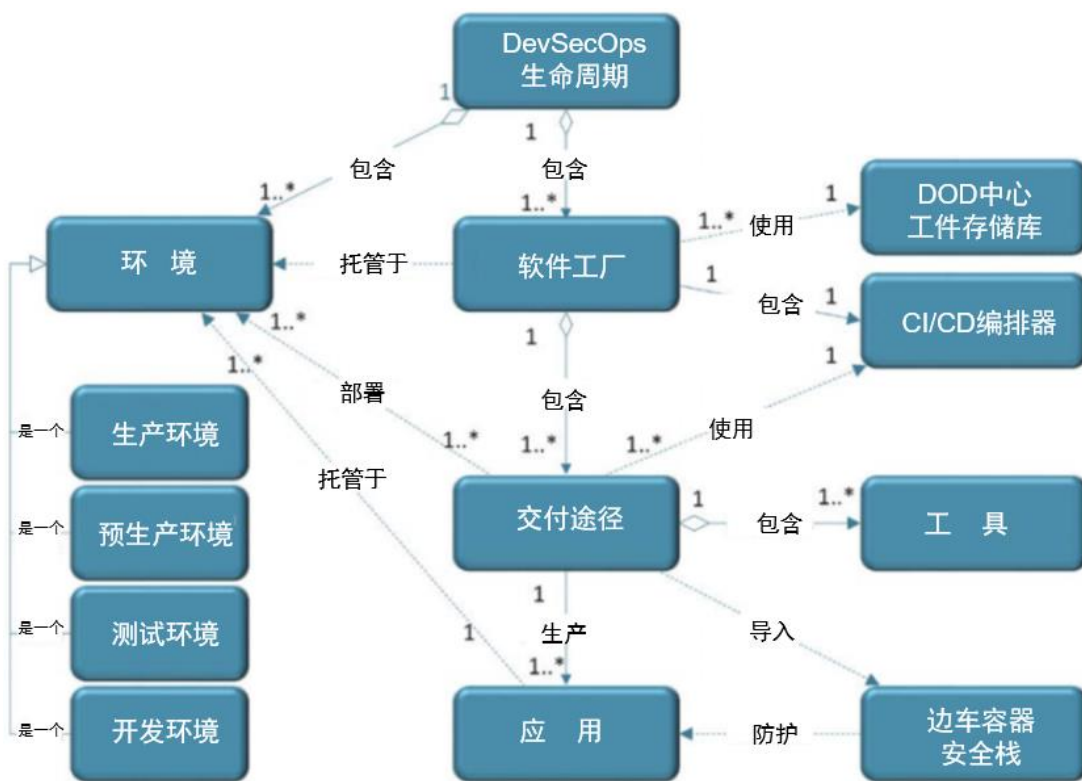


图 2 Conceptual Model 概念模型

3.2 DevSecOps Lifecycle DevSecOps 生命周期

The DevSecOps software lifecycle phases are illustrated in Figure 3. There are nine phases: plan, develop, build, test, release, deliver, deploy, operate, and monitor. Security is embedded within each phase.

DevSecOps 软件生命周期阶段如图 3 所示。有九个阶段：计划，开发，构建，测试，发布，交付，部署，运行，和监测。内部每个阶段都嵌入了安全。



图 3 DevSecOps Software Lifecycle 软件生命周期

With DevSecOps, The software development lifecycle is not a monolithic linear process. The “big bang” style delivery of the Waterfall process is replaced with small but more frequent deliveries, so that it is easier to change course as necessary. Each small delivery is accomplished through a fully automated process or semi-automated process with minimal human intervention to accelerate continuous integration and delivery. The DevSecOps lifecycle is adaptable and has many feedback loops for continuous improvement.

根据 DevSecOps 模型，软件开发生命周期不是一个单一的线性流程。以往“一锤定音”交付风格的瀑布流程的被小的但更频繁的交付所取代，必要时更容易改变过程。每一个小的交付都是通过一个最小人工干预、完全自动化或半自动的流程来完成，以加速持续集成和交付。DevSecOps 生命周期是可自适应的，并有许多反馈循环用于持续改进。

3.3 DevSecOps Pillars DevSecOps 的重要支柱

DevSecOps are supported by four pillars: organization, process, technology, and governance, as illustrated in Figure 4.

DevSecOps 由四个支柱支持：组织、流程、技术和治理，如图 4 所示。



图 4 DevSecOps Pillars 重要支柱

For each DoD organization, the practice of DevSecOps starts with buy-in of the DevSecOps philosophy by senior leaders within the organization. This leads to a change to the organizational culture, along with the development of new collaborative processes, technologies and tools to automate the process and to apply consistent governance. A project must advance in all four areas to be successful.

对于每个 DoD 组织，DevSecOps 的实践始于组织内高层领导对 DevSecOps 理念的认可。这导致组织文化的改变，同时开发新的协作流程、技术和工具，使流程自动化并应用一致的治理。一个项目必须在所有四个领域取得成功。

3.3.1 Organization 组织

The organization should embrace the following philosophies and ideas and incorporate them into their daily activities and software lifecycle management processes.

组织应接受以下理念和思想，并将其纳入其日常活动和软件生命周期管理流程。

- Change the organizational culture to take a holistic view and share the responsibility of software development, security and operations. Train staff with DevSecOps concepts and new technologies. Gradually gain buy-in from all stakeholders.
改变组织文化，以整体的观点，共担软件开发、安全和运维的责任。对员工进行 DevSecOps 概念和新技术的培训。逐渐获得所有利益相关者的认可。
- Break down organizational silos. Increase the team communication and collaboration in all phases of the software lifecycle.
打破组织的隔阂。在软件生命周期的所有阶段增加团队沟通和协作。
- Actionable security and quality assurance (QA) information, such as security alerts or QA reports, must be automatically available to the teams at each software lifecycle phase to make collaborative actions possible.
可操作的安全和质量保证(QA)信息，如安全告警或 QA 报告，必须在每个软件生命周期阶段自动提供给团队，以使协作行动成为可能。
- Build a culture of safety by sharing after-action reports on both positive and negative

events across the entire organization. Teams should use both success and failure as learning opportunities to improve the system design, harden the implementation, and enhance the incident response capability as part of the DevSecOps practice.

通过在整个组织分享关于积极和消极事件的事后报告，建立一种安全文化。团队应将成功和失败作为学习机会，以改进系统设计，加固部署，并将提高事件响应能力作为 DevSecOps 实践的一部分。

- Make many small, incremental changes instead of fewer large changes. The scope of smaller changes is more limited and thus easier to manage.

做许多小的，增量的改变而不是更少的大的改变。较小变化的范围更有限，因此更容易管理。

- Embrace feedback and user driven change to respond to new, emerging, and unforeseen requirements.

拥抱反馈和用户驱动的变化，以响应新的、正在兴起的、和不可预见的需求。

- Plan and budget for continuous code refactoring to ensure constant buy down of accumulated technical debt.

做好持续的代码重构的计划和预算，以确保不断减少累积的技术债务。

3.3.2 Process 流程

Depending on the mission environment, system complexity, system architecture, software design choices, risk tolerance level, and system maturity level, each program's software lifecycle has its own unique management processes.

根据任务环境，系统复杂性，系统体系结构，软件设计选择，风险承受能力水平，系统成熟度水平，每个项目的软件生命周期都有自己独特的管理流程。

For example, suppose a mature web application software system has adopted a microservices design. Its development, pre-production, and production environment are on the same cloud. The test procedures are fully automated. This system could have a process flow to automate the develop, build, test, secure, and delivery tasks to push updates into production quickly without human intervention. On the other hand, a complex mission critical embedded system, such as a weapons system, may have a different process that requires some tests that cannot be fully automated. The software lifecycle process for that system will be significantly different from the process for the web application system.

例如，假设一个成熟的 Web 应用软件系统采用了微服务设计。它的开发、预生产和生产环境都在同一个云上。测试流程完全自动化。这个系统可以有一个流程流来自动化开发、构建、测试、安全和交付任务，这样在没有人干预的情况下可以快速地将更新投产。另一方面，复杂的关键嵌入式系统，如武器系统，可能有一个不同的流程，需要一些无法完全自动化的测试。该系统的软件生命周期流程将与 Web 应用系统的流程显著不同。

To adopt a DevSecOps process successfully, implement it in multiple, iterative phases. Start small with some tasks that are easy to automate, then gradually build up the DevSecOps capability and adjust the processes to match. Figure 5 illustrates this concept; it shows that a software system can start with a Continuous Build pipeline, which only automates the build process after the developer commits code. Over time, it can then progress to Continuous Integration, Continuous Delivery, Continuous Deployment, Continuous Operation, and finally Continuous Monitoring, to achieve the full closed loop of DevSecOps. A program could start

with a suitable process and then grow progressively from there. The process improvement is frequent, and it responds to feedback to improve both the application and the process itself.

要成功地采用 DevSecOps 流程，请在多个迭代阶段实现它。从一些易于自动化的任务开始，然后逐步建立 DevSecOps 功能，并调整流程以匹配。图 5 说明了这个概念；它显示软件系统可以从一个持续构建流水线开始，它只在开发人员提交代码后自动生成构建流程。随着时间的推移，它可以进步到持续集成，持续交付，持续部署，持续运行，最后持续监控，达到 DevSecOps 的全闭环。一个程序可以从一个合适的流程开始，然后从那里逐步成长。流程改进是频繁的，它是对反馈的响应，用来改进应用和流程本身。

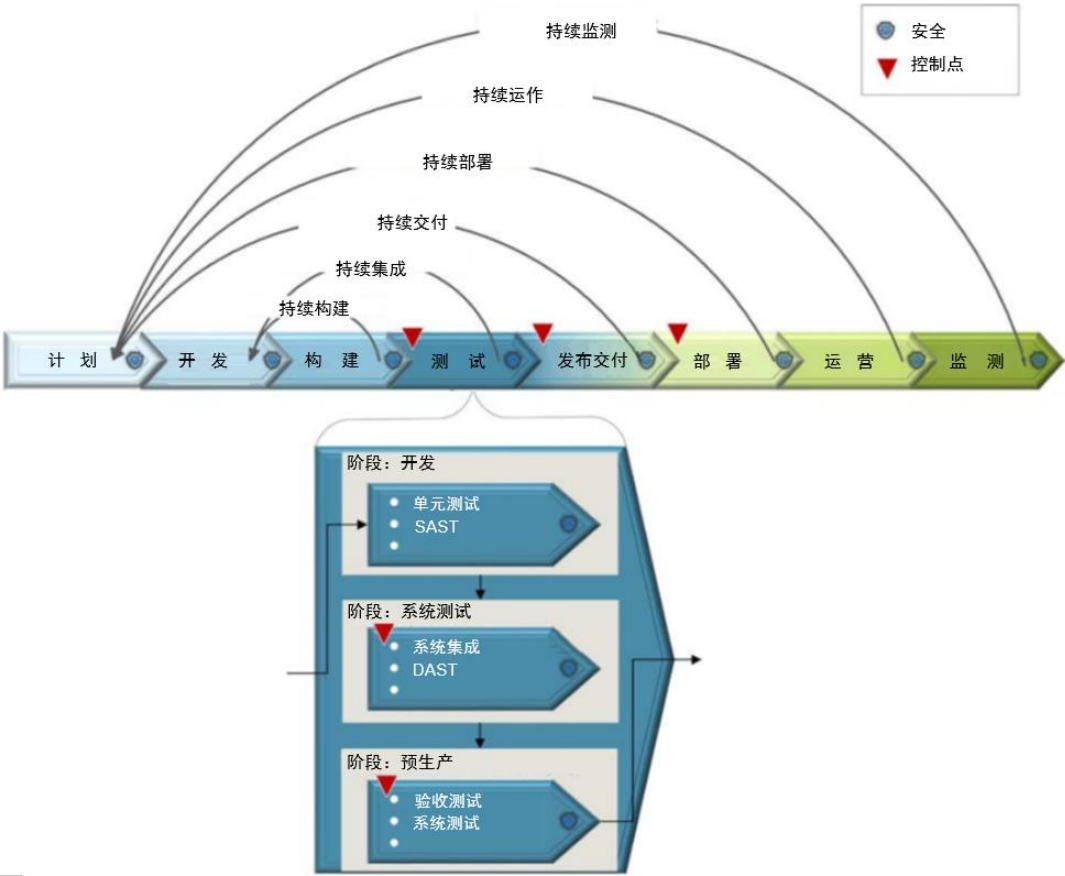


图 5 Application DevSecOps Processes 应用 DEVSECOPS 流程

There is no “one size fits all” solution for process design. Each software team has its own unique requirements and constraints. Below is a list of some best practices to guide the process design:

流程设计没有普适的解决方案。每个软件团队都有自己独特的需求和约束。下面列出一些指导流程设计的最佳做法：

1. The process design is a collective effort from multidisciplinary teams.
流程设计是多学科团队的集体努力。
2. Most of the processes should be automatable via tools and technologies.
大多数流程应该通过工具和技术实现自动化。
3. The DevSecOps lifecycle is an iterative closed loop. Start small and build it up progressively to strive for continuous improvement. Set up human intervention at the control gates when necessary, depending on the maturity level of the process and the

team^ confidence level in the automation. Start with more human intervention and gradually decrease it as possible.

DevSecOps 生命周期是一个迭代闭环。从小做起，逐步建立起来，争取持续改进。必要时在控制门处设置人为干预，这取决于流程的成熟度水平和团队对自动化的信心水平。从更多的人类干预开始，并逐渐减少它。

4. AO should consider automating the Authority to Operate (ATO) process as much as possible.

AO 应考虑使运行授权 (ATO) 尽可能自动化的实现。

To help organizations evolve their DevSecOps capabilities and processes, the DoD has developed a DevSecOps Maturity Model. This model details many steps that organizations can take to move incrementally towards a higher DevSecOps maturity level. That maturity model is presented in the DoD DevSecOps Playbook [7].

为了帮助组织发展他们的 DevSecOps 能力和流程，DoD 开发了一个 DevSecOps 成熟度模型。该模型详细说明了组织可以采取的许多步骤，以便逐步向更高的 DevSecOps 成熟度级别移动。该成熟度模型在 DoD DevSecOps Playbook[7]中提出。

3.3.3 Technology 技术

Many DevSecOps tools automate many tasks in the software lifecycle without human involvement. Other DevSecOps tools, such as collaboration and communication tools, facilitate and stimulate human interaction to improve productivity. Some DevSecOps tools aim to help an activity at a specific lifecycle phase. For example, an Integrated Development Environment (IDE) DevSecOps plug-in for develop phase, or a static application security test tool for the build phase. Most tools assist a particular set of activities. The tags added to artifacts in the artifact repository help guarantee that the same set of artifacts move together along a pipeline. Section 4 will introduce a variety of DevSecOps tools.

许多 DevSecOps 工具在软件生命周期中自动化了许多任务，而不需要人为参与。其他 DevSecOps 工具，如协作和通信工具，促进和刺激人际交往以提高生产力。一些 DevSecOps 工具旨在帮助特定生命周期阶段的活动。例如，用于开发阶段的 DevSecOps 的集成开发环境 (IDE) 插件，或用于构建阶段的静态应用程序安全测试工具。大多数工具都有助于一组特定的活动。在工件仓库中添加到工件的标签有助于保证同一组工件沿着流水线一起移动。第 4 节将介绍各种 DevSecOps 工具。

The instantiation of the DevSecOps environments can be orchestrated from configuration files instead of setting up one component at a time manually. The infrastructure configuration files, the DevSecOps tool configuration scripts, and the application run-time configuration scripts are referred to as Infrastructure as Code (IaC). Taking the same approach as IaC, security teams program security policies directly into configuration code, as well as implement security compliance checking and auditing as code, which are referred as Security as Code (SaC). Both IaC and SaC are treated as software and go through the rigorous software development processes including design, development, version control, peer review, static analysis, and test.

DevSecOps 环境的实例化可以从配置文件中编排，而不是一次手动设置一个组件。基础设施配置文件、DevSecOps 工具配置脚本和应用程序运行时配置脚本被称为基础设施即代码 (IaC)。安全团队采用与 IaC 相同的方法，将安全策略直接编码到配置代码中，并将安全遵从性检查和审计作为代码实现，这些代码被称为安全即代码 (SaC)。IaC 和 SaC 都被视为软件，并经历了严格的软件开发过程，包括设计、开发、版本控制、同行评审、静态分析和测试。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Technologies and tools play a key role in DevSecOps practice to shorten the software lifecycle and increase efficiency. They not only enable software production automation as part of a software factory, but also allow operations and security process orchestration.

技术和工具在 DevSecOps 实践中起着关键作用，以缩短软件生命周期和提高效率。它们不仅使软件生产自动化成为软件工厂的一部分，而且还允许操作和安全过程编排。

3.3.4 Governance 治理

Governance actively assesses and manages the risks associated with the mission program throughout the lifecycle. Governance activities do not stop after ATO but continue throughout the software lifecycle, including operations and monitoring. DevSecOps can facilitate and automate many governance activities.

治理在整个生命周期中主动地评估和管理与任务程序相关的风险。治理活动不会在 ATO 之后停止，而是在整个软件生命周期中继续进行，包括运行和监控。DevSecOps 可以促进和自动化许多治理活动。

3.3.4.1 Management Structure 管理结构

The management objective of DevSecOps must be both “top-down” and “bottom-up” to balance larger strategic goals. Studies (e.g., [8]) have shown that senior leader buy-in is crucial for success. But buy-in at the staff level is also important to engender a sense of ownership, to encourage the appropriate implementation of processes related to governance, and to enable team members to support continuous process improvement. Continuous process improvement - seeking opportunities to simplify and automate whenever and wherever possible - is essential for governance to keep pace with a rapidly changing world.

Dev SecOps 的管理目标必须是“自上而下”和“自下而上”，以平衡更大的战略目标。研究（例如，[8]）表明，高级领导人的支持对成功至关重要。但是，工作人员一级的认同对于产生主人翁意识、鼓励适当的与治理有关的流程实施，以及使团队成员能够支持持续的流程改进也很重要。持续的流程改进——寻找尽可能简化和自动化的机会——对于治理与迅速变化的世界保持同步至关重要。

Early DevSecOps efforts in the DoD, such as [9] have leveraged and adopted commercial best practices. That document identifies Five Fundamental Principles of Next Generation Governance (NGG):

早期 DevSecOps 在 DoD 中的工作，如[9]已经利用和采用了商业化最佳实践。该文件确定了下一代治理的五项基本原则 (NGG)：

1. Run IT with Mission Discipline: Tie requirements back to your organization's mission. Every action should be aligned to the mission. If they are not, then an evaluation should be performed with Continuous Process Improvement to address how to tie actions to missions.
与任务纪律一起运行 IT：将需求与组织的任务联系起来。每项行动都应 与任务目标保持一致。如果没有，则应通过持续流程改进进行评价，以解决如何将行动与任务目标挂钩的问题。
2. Invest in Automation: Automate everything possible, including actions, business processes, decisions, approvals, documentation, and more. Automation, including designs, interfaces, functional and security tests, and their related documentation, create the Artifacts of Record that provide the body of evidence required by the Risk Management Framework (RMF) and for historical audits when needed.

投资自动化：自动化一切可能的事情，包括行动，业务流程，决策，审批，文档等等。自动化，包括设计、接口、功能和安全测试及其相关文档，创建记录的工件，提供风险管理框架(RMF)所需的证据，并在需要进行历史审计。

3. Embrace Adaptability: Accept that change can be required at any time, and all options are available to achieve it. Fail fast, fail small, and fail forward. An example of failing forward is when a developer finds that a release does not work. Then instead of restoring the server to its pre-deployment state with the previous software, the developer's change should be discrete enough that they can fix it and address the issue through a newer release.

拥抱适应性：接受可以在任何时候要求更改，并且所有选项都可以实现它。快速失败，失败小型化，前进性失败。前进性失败的一个例子是开发人员发现一个版本不起作用。然后，然后，开发人员所做的更改应该足够离散，以便他们可以修复它并通过新的版本解决问题，而不是使用以前的软件将服务器恢复到部署前的状态。

4. Promote Transparency: Offer open access across the organization to view the activities occurring within the automated process and to view the auto-generated Artifacts of Record. Transparency generates an environment for sharing ideas and developing solutions comprised of Subject Matter Experts (SMEs) or leads from across the enterprise in the form of cross-functional teams to avoid the “silo effect.” When composed of all representative stakeholders, the team possesses the skills needed to build a mission system and the collective ingenuity necessary to overcome all encountered challenges.

促进透明度：提供跨组织的开放访问，以查看自动化过程中发生的活动，并查看自动生成的记录工件。透明度为分享想法和制定解决方案创造了一个环境，由主题专家（中小企业）组成，或以跨职能团队的形式从整个企业领导，以避免“孤岛效应”。当由所有具有代表性的利益方组成时，团队就拥有了建立任务系统所需的技能和克服所有遇到的挑战所需的集体智慧。

5. Inherent Accountability: Push down or delegate responsibility to the lowest level:

固有责任心：责任下放或下放到最低级别：

- Strategic: This is related to the Change Control Board (CCB) or Technical Review Board (TRB); it involves “Big Change” unstructured decisions. These infrequent and high-risk decisions have the potential to shape the strategy and mission of an organization.

战略：这与变更控制委员会(CCB)或技术审查委员会(TRB)有关；它涉及“大变化”的非结构化决策。这些不频繁的和高风险的决定有可能塑造一个组织的战略和使命。

- Operational: (Various Scrum) Cross-cutting, semi-structured decisions. In these frequent and high-risk decisions, a series of small, interconnected decisions are made by different groups as part of a collaborative, end-to-end decision process.

运行：(各种 Scrum) 横切，半结构化决策。在这些频繁和高风险的决策中，一系列小的、相互关联的决策是由不同的群体作为协作、端到端决策过程的一部分作出的。

- Tactical: (Global Enterprise Partners (GEP)/Product Owner/Developers Activities) Delegated, structured decisions. These frequent and low-risk decisions are effectively handled by an individual or working team, with limited input from others.

战术：(全球企业伙伴(GEP)/产品负责人/开发人员活动)委托、结构化决策。这些频繁和低风险的决定只需要其他人的有限输入，就可以由个人或工作组有效地处理。

These 5 principles are summarized in Figure 6, which is from [9].

图 6 总结了这 5 个原则，它来自[9]。

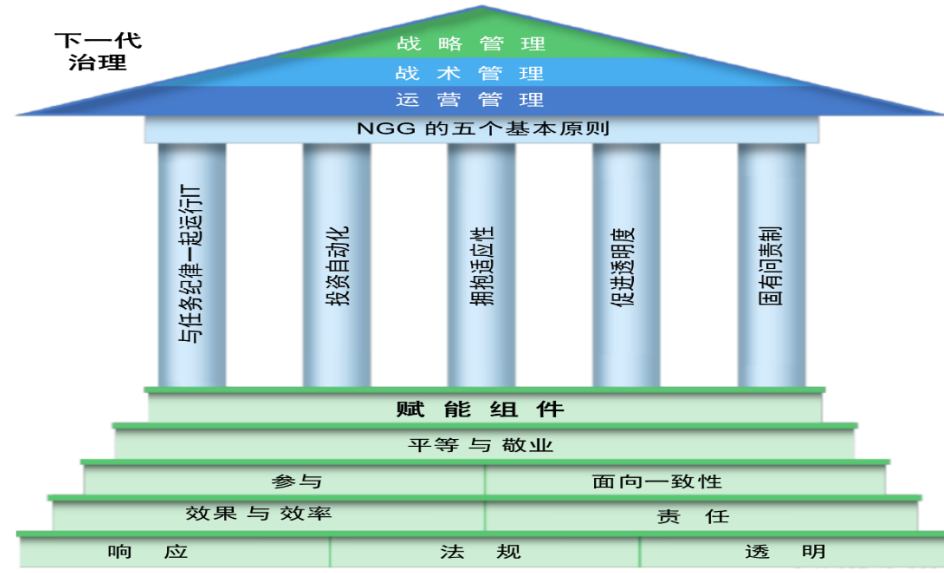


图 6Five Principles of Next Generation Governance 下一代治理的五项原则

3.3.4.2 Authorizing Official 授权官员

DoD Instruction (DoDI) 8510.01 [10] is the ultimate governance policy and states processes that all DoD information system and platform information technology system must follow. It is under revision and the following DevSecOps related governance information will be incorporated in the future release.

DoD 指示 (DoDI) 8510.01 [10] 是最终的治理策略，并声明所有 DoD 信息系统和平台信息技术系统必须遵循的过程。它正在修订，以下 DevSecOps 相关的治理信息将纳入未来的版本。

For initial standup of a new DevSecOps software factory instance and a production operations environment, the RMF process follows an enterprise level process. The assessment and authorization (A&A) should inherit the certifications and authorizations of the underlying infrastructure (e.g., a DoD cloud provisional authorization) and of the DoD Enterprise Hardened Containers, without having to re-certify them. The program should have a formal Service Level Agreement (SLA) with the underlying infrastructure provider about what services are included and what authorizations can be inherited. This affects the status of applicable assessment procedures and prepares the stage for inheritance into the operations environment and application. Once the instance is authorized and operational, the specialty AO for the functional area or the local AO for the program has cognizance for Continuous Authorization of the environment. Figure 7 illustrates the A&A inheritance.

对于初始设立新的 DevSecOps 软件工厂实例和生产运行环境，RMF 过程应遵循企业级过程。评估和授权 (A&A) 应继承基础设施的认证和授权 (例如，一个 DoD 云临时授权) 和 DoD 企业加固容器，而无需重新认证。该程序应该与底层基础设施提供者有一个正式的服务级别协议 (SLA)，说明包含哪些服务以及可以继承哪些授权。这影响到适用的评估程序的状态，并为继承到运行环境和应用程序做好准备。一旦实例被授权和运行，功能区域的专用 AO 或该程序的本地 AO 就具有对环境进行连续授权的权限。图 7 显示了 A&A 继承。

The specialty AO for the functional area or the local AO for the program has cognizance for Continuous Authorization of the mission applications.

功能区域的特定 AO 或程序的本地 AO 具有持续授权任务应用程序的权限。

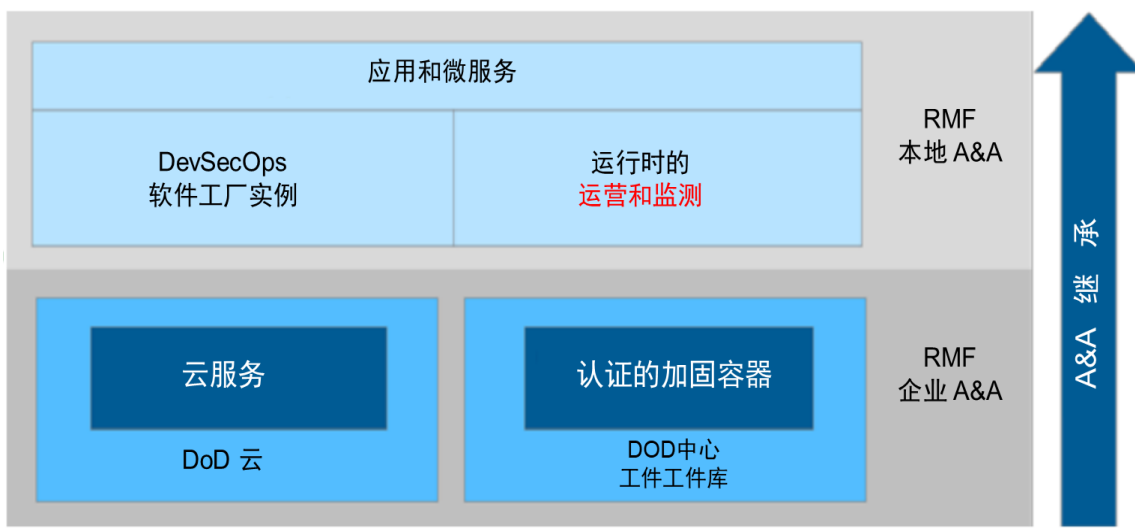


图 7 Assessment and Authorization Inheritance 评估和授权继承

表 2 Roles of Authorizing Officials in DevSecOps 授权角色

Capability 能力	Authorizing Official 授权官员
DoD Enterprise Hardened Containers DOD 企业加固容器	Enterprise AO (e.g., Defense Information Systems Agency (DISA)) 企业 AO (例如, 国防信息系统局 (DISA))
DevSecOps software factory instances DevSecOps 软件工厂实例	Enterprise AO (e.g., DISA, Military Department (MilDep) CIO) 企业 AO (例如 DISA、军事部门 (MilDep) 首席信息官 (CIO))
Continuous Process Improvement / Continuous Authorization of DevSecOps software factory instances 持续过程改进/持续授权 DevSecOps 软件工厂实例	Specialty or Local AO (e.g., Program executive Officer (PEO)) 专业或本地 AO (如项目主管高级职员 (PEO))
Specialty or Local AO (e.g., Program executive Officer (PEO)) 专业或本地 AO (如项目主管高级职员 (PEO))	Specialty or Local AO (e.g., PEO) 专业或本地 AO (如 PEO)

3.4 DevSecOps Ecosystem DEVSECOPS 生态系统

The DevSecOps ecosystem is a collection of tools and the process workflows created and executed on the tools to support all the activities throughout the full DevSecOps lifecycle. As illustrated in Figure 8, the DevSecOps ecosystem is composed of three subsystems: planning, a software factory, and production operations. The DevSecOps ecosystem interacts with external enterprise services to get all dependency support, and with enterprise and local AO to gain operation authorization.

DevSecOps 生态系统是一个工具集合和在工具上创建和执行的工作流，以支持整个 DevSecOps 生命周期中的所有活动。如图 8 所示，DevSecOps 生态系统由三个子系统组成：规划、软件工厂和生产运行。DevSecOps 生态系统与外部企业服务交互以获得所有依赖支持，并与企业和本地 AO 交互以获得运行授权。

The DevSecOps administration team is responsible for administrating the ecosystem tools and automating the process workflows. The mission application team focuses on the development, testing, security and operations tasks using the ecosystem.

DevSecOps 管理团队负责管理生态系统工具和自动化流程工作流程。任务应用团队专注于使用生态系统的开发、测试、安全和运行任务。

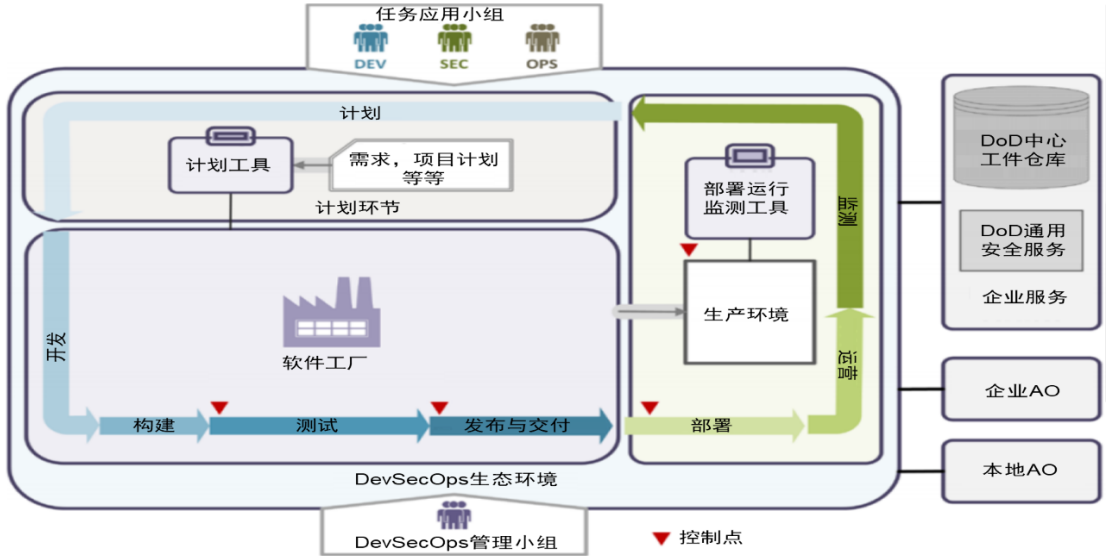


图 8 DevSecOps Ecosystem 生态系统

3.4.1 Planning 计划

The plan phase involves activities that help the project manage time, cost, quality, risk and issues. These activities include business-need assessment, project plan creation, feasibility analysis, risk analysis, business requirements gathering, business process creation, system design, DevSecOps design and ecosystem instantiation, etc. The plan phase repeats when DevSecOps the lifecycle recycles. It is a best practice to develop a minimum viable product (MVP) for critical business needs as the first thing to develop. Then get into the feedback loop process as quickly as possible; this is recommended in the Lean Startup methodology [11]. The DevSecOps design creates the DevSecOps processes and control gates, which will guide the automation throughout the lifecycle. DevSecOps ecosystem tools will facilitate process automation and consistent process execution.

计划阶段涉及帮助项目管理时间、成本、质量、风险和问题的活动。这些活动包括业务需求评估、项目计划创建、可行性分析、风险分析、业务需求收集、业务流程创建、系统设计、DevSecOps 设计和生态系统实例化等。当 DevSecOps 生命周期循环时，计划阶段会不断重复。这是一个最佳实践，开发一个最小化可行的产品 (MVP)，以满足关键的业务需求，作为开发的第一件事。然后尽快进入反馈循环过程；这是在精益启动方法论[11]中推荐的。DevSecOps 设计创建了 DevSecOps 流程和控制门，这将指导整个生命周期的自动化。DevSecOps 生态系统工具将促进流程自动化和一致的流程执行。

The DevSecOps planning subsystem supports the activities in the plan phase using a set of communication, collaboration, project management, and change management tools. In this

phase, the process workflows are not fully automated. The planning tools assist human interaction and increase team productivity.

DevSecOps 计划子系统使用一组通信、协作、项目管理和变更管理工具支持计划阶段的活动。在这一阶段，流程工作流没有完全自动化。规划工具帮助团队沟通，提高团队生产力。

3.4.2 Software Factory 软件工厂

A software factory, illustrated in Figure 9, contains multiple pipelines, which are equipped with a set of tools, process workflows, scripts, and environments, to produce a set of software deployable artifacts with minimal human intervention. It automates the activities in the develop, build, test, release, and deliver phases. The environments that are set up in the software factory should be orchestrated with scripts that include IaC and SaC and which run on various tools. A software factory must be designed for multi-tenancy and automate software production for multiple projects. A DoD organization may need multiple pipelines for different types of software systems, such as web applications or embedded systems.

一个软件工厂，如图 9 所示，包含多个流水线，它们配备了一组工具、流程工作流、脚本和环境，以产生一组具有最小化人工干预的软件可部署工件。它使开发、构建、测试、发布和交付阶段的活动自动化。在软件工厂中设置的环境应该使用包含 IaC 和 SaC 的脚本进行编排，这些脚本在各种工具上运行。软件工厂必须为多租户设计，并为多个项目进行自动化软件生产。对于不同类型的软件系统，如 Web 应用程序或嵌入式系统，DoD 组织可能需要多条流水线。

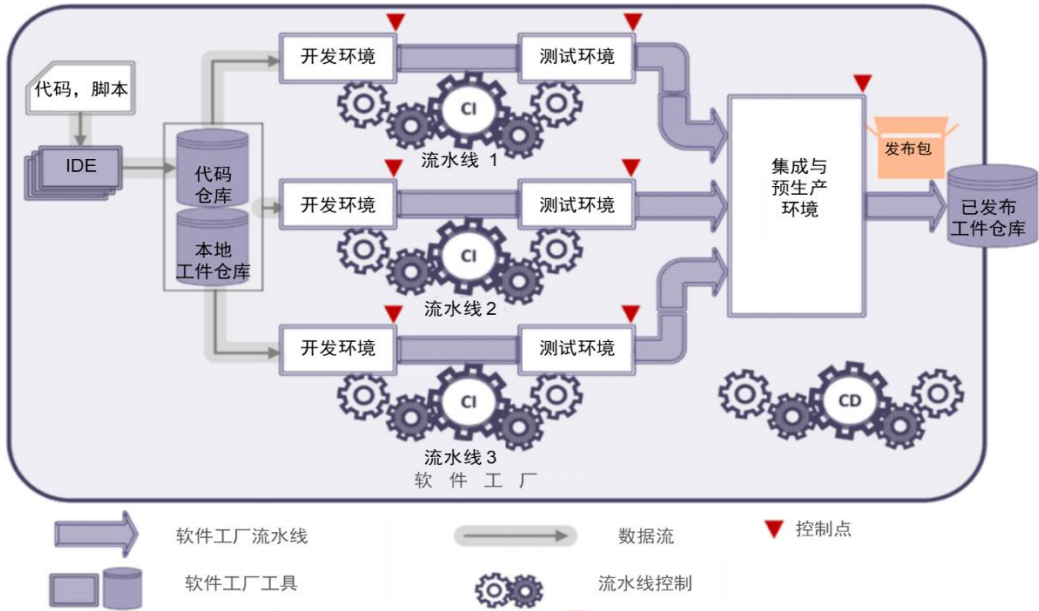


图 9DevSecOps Software Factory 软件工厂

The factory starts with the development team developing application code and IaC, QA developing test scripts, and the security team developing SaC in their suitable IDEs. The entire Software Factory should leverage OCI compliant containers and DoD Hardened Containers whenever available. Once COTS, Government off the Shelf (GOTS), or newly developed code and scripts are committed into the Software Factory's code repository, the assembly line

automation kicks in. There could be multiple CI pipeline instances as assembly lines. Each is for a specific application subsystem, such as a JavaScript assembly line for a web front-end, a Python or R assembly line for data analytics, or a GoLang assembly line for a backend application. The CI assembly line glides the subsystem through continuous integration by building the code and incorporating dependencies (such as libraries) from the local artifact repository. In addition, it performs tests in the development and test environments, such as unit tests, static code analysis, functional tests, interface tests, dynamic code analysis, etc. The subsystems that pass CI assembly line control gate policies will move into the pre-production environment for systems integration. The CD assembly line takes over control from this point. More tests and security scans are performed in this environment, such as performance tests, acceptance test, security compliance scan, etc. The CD assembly line releases and delivers the final product package to the released artifact repository if the control gate policies are met.

工厂从开发团队从开发团队编写应用程序代码和 IaC， QA 部门开发测试脚本开始，安全团队在合适的 IDE 中开发 SaC。 只要有可能，整个软件工厂应该利用符合 OCI 的容器和 DoD 加固容器。 一旦使用 COTS (Government off the Shelf)， 或者新开发的代码和脚本被提交到软件工厂的代码仓库中，流水线自动化就开始了。 可能有多个持续集成流水线实例作为整体流水线。 每个应用程序子系统用于特定的应用程序子系统，例如用于 Web 前端的 Java 脚本流水线、用于数据分析的 Python 或 R 流水线或用于后端应用程序的 GoLang 流水线。 通过构建代码并合并本地工件仓库中的依赖项（如：库），CI 流水线通过连续集成来滑动子系统储存库。 此外，它还在开发和测试环境中执行测试，如单元测试、静态代码分析、功能测试、接口测试、动态代码分析等。 通过 CI 装配线控制门策略的子系统将进入系统集成的预生产环境。 光盘装配线从这一点接管控制。 在此环境中执行更多的测试和安全扫描，如性能测试，验收测试，安全符合性扫描等。如果满足了控制门策略，CD 装配线将发布并将最终产品包交付给发布的工件存储库。

Developing applications using a DevSecOps software factory provides many benefits:

使用 DevSecOps 软件工厂开发应用程序提供了许多好处：

- Improved software product consistency and quality
提高了软件产品的一致性和质量
- Shortened time to market and increased productivity
缩短了上市时间，提高了生产率
- Simplified governance
简化治理

3.4.3 Operations 运营

In the production environment, the released software is pulled from the released artifact repository and deployed. Operations, operation monitoring, and security monitoring are performed. Production operation tools aim to streamline and automate the deployment, operations, and monitoring activities. Tools should be selected based on system functional requirements and their suitability for the production environment infrastructure.

在生产环境中，已发布工件从已发布工件仓库中提取并进行部署。，并执行运行，运行监控，安全监控。 生产环境的运行工具旨在简化和自动化部署、运行和监控活动。 工具的选择应基于系统功能需求及其对生产环境基础设施的适用性。

3.4.4 External Systems 外部系统

The DevSecOps ecosystem itself and program applications depend on some DoD enterprise services to acquire the necessary baseline tools, application dependencies, and security services to operate.

DevSecOps 生态系统本身和程序应用程序依赖于一些 DoD 企业服务来获取必要的基线工具、应用程序依赖项和安全服务来运行。

- DoD Centralized Artifact Repository (DCAR) holds the hardened VM images and hardened OCI compliant container images of: DevSecOps tools, container security tools, and common program platform components (e.g. COTS or open source products) that DoD program software teams can utilize as a baseline to facilitate the authorization process.

DoD 集中式工件仓库(DCAR)保存已加固的 VM 镜像和加固的符合 OCI 标准的容器镜像, 包括 DevSecOps 工具、容器安全工具和公共程序平台组件(例如。 COTS 或开源产品), DoD 程序软件团队可以利用这些作为基线, 以加快授权流程。

- DoD Common Security Services are DoD enterprise-level common services that facilitate cybersecurity enforcement and IT management. One security service will perform traffic inspection and filtering to protect the mission enclave and mission applications. Some security service examples include firewalls; Intrusion Detection System (IDS)/Intrusion Prevention System (IPS); malware detection; data loss prevention; host-based security; log/telemetry aggregation and analysis; and Identity, Credential, and Access Management (ICAM). A Cybersecurity Service Provider (CSSP) will provide additional services, including Attack Sensing and Warning (ASW), Forensic Media Analysis (FMA), Assurance Vulnerability Management (AVM), Incident Reporting (IR), Incident Handling Response (IHR), Information Operations Condition (INFOCON), Cyber Protection Condition (CPCON), Malware Notification Protection (MNP), and Network Security Monitoring (NSM).

DoD 通用安全服务是 DoD 企业级的通用服务, 以促进网络安全的执行和 IT 管理。 一个安全部门将进行流量检查和过滤, 以保护特定飞地和特定应用程序。 一些安全服务示例包括防火墙、入侵检测系统(IDS)/入侵防御系统(IPS)、恶意软件检测、数据泄露防护; 基于主机的安全、日志/遥测聚合和分析、身份、证书和访问管理(ICAM)。 网络安全服务提供商(CSSP)将提供更多的服务, 包括攻击感知和警告(ASW)、取证分析(FMA)、漏洞管理(AVM)、事件报告(IR)、事件处理响应(IHR)、信息操作条件(INFOCON)、网络保护条件(CPCON)、恶意软件通知保护(MNP)和网络安全监控(NSM)。

The DevSecOps ecosystem interacts with the enterprise AO for the initial software factory ATO and initial application ATO, as well as the local AO for continuous ATO for the application.

DevSecOps 生态系统与企业 AO 交互, 用于初始软件工厂 ATO 和初始应用 ATO, 以及本地 AO 用于应用程序的连续 ATO。

4 DevSecOps Tools and Activities DevSecOps 工具和活动

This section describes both tools for the DevSecOps ecosystem and DevSecOps activities for each phase. Activities and tools are listed in table format. The Baseline column in the tool tables has two values: Minimal Viable Product (MVP) and objective. They indicate whether the tool must be available in the DevSecOps ecosystem MVP as threshold or if the tool is an objective to be reached as the ecosystem matures. Activity tables list a wide range of activities for DevSecOps practice. DoD organizations should define their own processes, choose proper activities, and then select tools suitable for their systems to build software factories and DevSecOps ecosystems. With the DevSecOps maturity progression, the level of activity automation will increase.

本节介绍了每个阶段的 DevSecOps 生态系统工具和 DevSecOps 活动。活动和工具以表格形式列出。工具表中的“基线”列具有两个值：最小化可行产品（MVP）和目标。它们指出该工具是否必须在 DevSecOps 生态系统 MVP 中作为阈值使用，或者该工具是否可作为 DevSecOps 生态系统成熟的目标。活动表列出了 DevSecOps 实践的各种活动。DoD 内的各组织应定义自己的流程，选择适当的活动，然后选择适合其系统的工具来构建软件工厂和 DevSecOps 生态系统。随着 DevSecOps 成熟度的提高，活动自动化的水平将随之提高。

4.1 Planning Tools and Activities 规划工具和活动

The Planning tools support software development planning, which includes configuration management planning, change management planning, project management planning, system design, software design, test planning, and security planning. Table 3 lists some tools that can assist the planning process. Some tools will be used throughout the software lifecycle, such as a team collaboration tool, an issue tracking system, and a project management system. Some tools are shared at the enterprise level across programs. Policy and enforcement strategy should be established for access controls on various tools.

规划工具支持软件开发规划，其中包括配置管理规划，变更管理规划，项目管理规划，系统设计，软件设计，测试规划和安全规划。表 3 列出了一些有助于规划流程的工具。一些工具可以在整个软件生命周期中使用，例如团队协作工具，问题跟踪系统和项目管理系统。一些工具在企业级别之间跨项目进行共享。政策和执行策略应该为不同类型的工具上的权限控制而进行建立。

表 3 Plan Phase Tools 计划阶段工具

Tool 工具	Features 特性	Benefits 收益	Inputs 输入	Outputs 输出	Baseline 基线
Team collaboration system 团队协作系统	Audio/video conferencing; chat/messaging; brainstorming discussion board; group calendars; file sharing; Wiki website 音频/视频会议；聊天/消息；头脑风暴	Simplify communication and boost team efficiency 简化沟通并提高团队效率	Team meetings; Design notes; Documentation 团队会议；设计说明；文献资料	Organized teamwork; Version controlled documents 有组织的团队合作；版本控制文件	MVP 最小化可行产品

	讨论板； 团体日历； 文件共享； Wiki 网站				
Issue tracking system 问题跟踪系统	Bugs and defect management; Feature and change management; Prioritization management; Assignment management; Escalation management; Knowledge base Management 漏洞和缺陷管理； 特性和变更管理； 优先级管理； 作业管理； 问题升级管理； 知识库	Easy to detect defect trends Improve software product quality Reduce cost and improve Return on Investment (ROI) 易于发现缺陷趋势提高软件产品质量；降低成本并提高投资回报率 (ROI)	Bug report; Feature/change request; Root cause analysis 错误报告； 功能/更改请求； 根本原因分析； 解决方案	Issues feature/change tickets Issue resolution tracking history 发行功能/变更票。问题解决跟踪历史记录	MVP 最小化可行产品
Asset inventory management 资产库存管理	Collect information about all IT assets; Maintain a “real-time” inventory of all applications, software licenses, libraries, operating systems, and versioning information 收集有关所有 IT 资产的信息； 维护所有应用程序，软件许可证，库，操作系统和版本信息的“实时”清单	Increase situation awareness 增强态势感知	IT assets (applications, software licenses, libraries, operating systems, and versioning information) IT 资产（应用程序，软件许可，库，操作系统和版本控制信息）	Asset Inventory 资产库存清单	Objective 目标
Configuration management database(CMDB) 配置管理数据库	Auto-discovery; Dependency mapping; Integration with other tools; Configuration auditing 自动发现； 依赖关系映射； 与其他工具集成； 配置审计	Centralized database used by many systems (such as asset management, configuration management, incident management, etc.) during development and operations phases. 集中在开发和运营阶段，许多系统（例如资产管理，配置管理，事件管理等）使	IT hardware and software components information IT 硬件和软件组件信息	Configuration Items 配置项	Objective 目标

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

		用的数据库。			
Project management system 项目管理系统	Task management Scheduling and time management Resource management Budget management Risk management 任务管理计划和时间管理；资源管理；预算管理；风险管理	Assist project progress tracking Optimize resource allocation 协助项目进度跟踪；优化资源分配	Tasks, scheduling, resource allocation, etc. 任务，计划，资源分配等	Project plan 项目计划	MVP 最小化可行产品
Software system design tool 软件系统设计工具	Assist requirement gathering, system architecture design, components design, and interface design 协助需求收集，系统架构设计，组件设计和界面设计	Independent of programming languages Helps visualize the software system design 独立于编程语言，有助于可视化软件系统设计	User requirements Design ideas 用户需求设计思路	System design documents, Function design document, Test plan, System deployment environment configuration plan 系统设计文件，功能设计文件，测试计划，系统部署环境配置计划	Objective 目标
Threat modeling tool 威胁建模工具	Document system security design; Analyze the design for potential security issues; Review and analysis against common attack patterns; Suggest and manage mitigation 文件系统设计；分析设计中潜在的安全问题；审查和分析常见的攻击模式；建议和管理缓解措施	Allows software architects to identify and mitigate potential security issues early. 允许软件架构师及早发现并缓解潜在的安全问题。	System design 系统设计	Potential threats and mitigation plan 潜在的威胁和缓解计划	Objective 目标
Data modeling tool 数据建模工具	Model the interrelationship and flows between different data elements 建模不同数据元素之	Ensure the required data objects by the system are accurately represented 确保系统正确表示所需的数据对	System requirement; Business logic 系统要求；业务逻辑	Data model 数据模型	Objective if using a database 使用数据库时的目标

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	间的相互关系和流程	象			
--	-----------	---	--	--	--

The activities supported by the plan phase are listed in Table 4. Some activities are suitable at enterprise or program level, such as DevSecOps ecosystem design, project team onboarding planning, and change management planning. Others fit at the project level and are considered continuous in the DevSecOps lifecycle.

计划阶段支持的活动在表 4 中列出。某些活动适用于企业或程序级别，例如 DevSecOps 生态系统设计，项目团队的正式开展计划和变更管理计划。其他项目则适合项目级别，在 DevSecOps 生命周期中被认为是持续的。

表 4Plan Phase Activities 计划阶段活动

Tool 工具	Features 特征	Benefits 收益	Inputs 输入	Outputs 产出	Tool Dependencies 工具依赖性
DevSecOps ecosystem design 生态系统设计	Design the DevSecOps process workflows that are specific to this project 设计特定于此项目的 DevSecOps 流程 工作流	-Change Management process -System design; -Release plan & schedule. 变更管理流程； 系统设计；发布计划和日程安排	DevSecOps process flow chart; DevSecOps ecosystem tool selection; Deployment platform selection DevSecOps 工作流图； DevSecOp 生态系统工具选择；部署平台选择	Team Collaboration system 团队协作系统	
Project team onboarding planning 项目团队入场计划	Plan the project team onboarding process, interface, access control policy 计划项目组入场流程，接口，访问控制	Organization policy 组织策略	Onboarding plan 入场计划	Team collaboration system 团队协作系统	
Change management planning 变更管理计划	Plan the change control process 规划变更控制流程	-Organizational policy; -Software development best practice. 组织策略；软件开发最佳实践	Change control procedures; Review procedures; Control review board; change management plan 变更控制程序； 评审程序；控制评审委员会；变更管理计划	Team collaboration system; Issue tracking system 团队协作系统；问题跟踪系统	
Configuration management (CM) planning 配置管理计划	Plan the configuration control process; Identify configuration items 计划配置控制流	-Software development, security and operations best practice; -IT	CM processes and plan; CM tool selection; Responsible configuration	Team collaboration system; Issue tracking system、 团队协作系统；	

	程；识别配置项	infrastructure asset; -Software system components. 软件开发、安全和操作的最佳实践；IT 基础设施资产； 软件系统组件。	items; Tagging strategy 配置管理流程和计划；配置管理工具选择；配置项职责；标签策略	问题跟踪系统	
Software requirement analysis 软件需求分析	Gather the requirements from all stakeholders 收集所有利益攸关方的需求	-Stakeholder inputs or feedback; -Operation monitoring feedback; -Test feedback. -利益相关方的投入或反馈； -运行监控反馈； -测试反馈。	-Feature requirements - Performance requirements - Privacy requirements - Security requirements 特性需求； 性能需求； 隐私需求； 安全需求	Team collaboration system; Issue tracking system 团队协作系统； 问题跟踪系统	
System design 系统设计	Design the system based the requirements 根据需求设计系统	Requirements Documents 需求文档	文件： 系统架构；功能设计；数据流程图；测试计划；基础设施配置计划；工具选择；开发工具；测试工具；部署平台	Team collaboration system; Issue tracking system Software system design tools 团队协作系统； 问题跟踪系统 软件系统设计工具	
Project planning 项目计划	Project task management Release planning 项目任务管理发布计划		Task plan & schedule; Release plan & schedule. 任务计划和时间表；发布计划和时间表。	Team collaboration system; Project management system 团队协作系统； 项目管理系统	
Risk management 风险管理	Risk assessment 风险评估	-System architecture; -Supply chain information; -Security risks. -系统架构； -供应链信息； -安全风险。	Risk management plan 风险管理计划	Team collaboration system; 团队协作系统	
Configuration	Discover or	-IT	Configuration	CMDB;	

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Identification 配置识别	manual input configuration items into CMDB; Establish system baselines 发现或手动将配置项输入 CMDB; 建立系统基线	infrastructure asset; -Software system components (include DevSecOps tools); -code baselines -document baselines. -IT 基础设施资产; 软件系统组件(包括 DevSecOps 工具); -编码基线 -文档基线。	items 配置项	Source code repository; Artifact repository; Team collaboration system cmdb; 源代码储存库; 工件库; 团队协作系统	
Threat modeling 威胁模型	Identify potential threats, weaknesses and vulnerabilities. Define the mitigation plan 识别潜在的威胁, 弱点和脆弱点。 明确缓解计划	System design 系统设计	Potential threats and mitigation plan 潜在威胁和缓解计划	Threat modeling tool 威胁建模工具	
Database design 数据库设计	Data modeling; database selection; Database deployment topology 数据建模; 数据库选择; 数据库部署拓扑	System requirement; System design 系统需求; 系统设计	Database design document 数据库设计文档	Data modeling tool; Team collaboration system 数据建模工具; 团队协作系统	
Design review 设计评审	Review and approve plans and documents 评审合批准计划和文档	Plans and design documents; 计划和设计文档	Review comments; Action items 评审意见, 行动项	Team collaboration system 团队协作系统	
Documentation version control 文档版本控制	Track design changes 跟踪设计变更	Plans and design documents; 计划和设计文档	Version controlled documents 版本控制文档	Team collaboration system 团队协作系统	

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

4.2 Software Factory Tools and Activities 软件工厂工具和活动

Software factory tools include a CI/CD orchestrator, a set of development tools, and a group of tools in the build, test, release, and deliver phases that are pluggable to the CI/CD orchestrator.

软件工厂工具包括 CI/CD 编排器、一组开发工具以及构建、测试、发布和交付阶段中可插入到 CI/CD 编排器的一组工具。

4.2.1 CI/CD Orchestrator CI/CD 编排器

The CI/CD Orchestrator is the central automation engine of the CI/CD pipeline. It manages pipeline creation, modification, execution, and termination.

CI/CD 编排器是 CI/CD 流水线的中央自动化引擎。它管理流水线的创建、修改、执行和终止。

The DevSecOps team creates a pipeline workflow in the Orchestrator by specifying a set of stages, stage conditions, stage entrance and exit control rules, and stage activities. The Orchestrator automates the pipeline workflow by validating the stage control rules. If all the entrance rules of a stage are met, the Orchestrator will transition the pipeline into that stage and perform the defined activities by coordinating the tools via plugins. If all the exit rules of the current stage are met, the pipeline exits out the current stage and starts to validate the entrance rules of the next stage.

DevSecOps 团队通过指定一组阶段，阶段条件，阶段入口和出口控制规则以及阶段活动在 Orchestrator 中创建流水线工作流。编排器通过验证阶段控制规则来自动化流水线工作流程。如果满足了某个阶段的所有进入规则，那么编排器将通过流水线将流水线转换到该阶段并通过插件协调工具来执行定义的活动。如果满足当前阶段的所有退出规则，则流水线退出当前阶段并开始验证下一阶段的进入规则。

Table 5 shows the features, benefits, and inputs and outputs of the CI/CD Orchestrator.
表 5 显示了 CI/CD 编排器的特性，收益以及输入和输出。

表 5CI/CD Orchestrator CI/CD 编排器

Tool 工具	Features 特征	Benefits 收益	Inputs 输入	Outputs 产出	Baseline 基线
CI/CD Orchestrator CI/CD 编排器	Create pipeline workflow 创建流水线工作流	Customizable pipeline solution 个性化流水线解决方案	Human input about: A set of stages A set of event triggers Each stage entrance and exit control gate Activities in each stage 人工输入： 一组阶段 一组事件触发器 每个阶段的进出控制门 每个阶段的活动	Pipeline workflow configuration 流水线工作流配置	MVP 最小可用产品

	Orchestrate pipeline workflow execution by coordinating other plugin tools or scripts. 通过协调其他插件工具或脚本来协调流水线工作流程的执行。	Automate the CI/CD tasks; Auditable trail of activities 自动执行 CI / CD 任务； 可审计的活动轨迹	Event triggers (such as code commit, test results, human input, etc.); Artifacts from the artifact repository 事件触发器（例如代码提交，测试结果，人工输入等）；来自工件库的工件	Pipeline workflow execution results (such as control gate validation, stage transition, activity execution, etc.); Event and activity audit logs 流水线工作流执行结果（例如控制门验证，阶段过渡，活动执行等）；事件和活动审核日志	
--	---	--	--	--	--

4.2.2 Develop 开发

The Develop phase uses tools to support the development activities that convert requirements into source code, the source code includes application code, test scripts, Infrastructure as Code, Security as Code, DevSecOps workflow scripts, etc. The development team may rely on a single modern integrated development environment (IDE) for multiple programming language support. The IDE code assistance feature aids developers with code completion, semantic coloring, and library management to improve coding speed and quality. The integrated compiler, interpreter, lint tools, and static code analysis plugins can catch code mistakes and suggest fixes before developers check code into the source code repository. Source code peer review or pair programming are other ways to ensure code quality control. All the code generated during development must be committed to the source code repository and thus version controlled. Committed code that breaks the build should be checked in on a branch and not merged into the trunk until it is fixed.

开发阶段使用工具来支持将需求转换为源代码的开发活动，源代码包括应用程序代码，测试脚本，基础架构即代码，安全性即代码，DevSecOps 工作流脚本等。开发团队可能依赖于一个单一的现代版本。集成开发环境（IDE），支持多种编程语言。IDE 代码辅助功能可帮助开发人员进行代码完成，语义着色和库管理，以提高编码速度和质量。集成的编译器，解释器，lint 工具和静态代码分析插件可以在开发人员将代码检入源代码存储库之前捕获代码错误并提出修复建议。源代码同行评审或结对编程是确保代码质量控制的其他方法。开发期间生成的所有代码都必须提交给源代码存储库，并由此进行版本控制。中断构建的提交代码应在分支上检查，直到没问题修复后才能够合并到主干中。

The following tables list the components that facilitate code development, along with their inputs and outputs.

下表列出了有助于代码开发的组件以及其输入和输出。

表 6 Develop Phase Tools 开发阶段工具

Tool 工具	Features 特征	Benefits 收益	Inputs 输入	Outputs 产出	Baseline 基线
Integrated development	Source code editor Intelligent code completion	Visual representation Increase	Developer coding input 开发代码输入	Source code 源代码	MVP 最小化可用产品

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

environment (IDE) 集成开发环境 (IDE)	Compiler or interpreter Debugger Build automation (integration with a build tool) 源代码编辑器 智能代码完成 编译器或解释器 调试器 构建自动化 (与构建工具集成)	efficiency Faster coding with less effort Improved bug fixing speed Reproducible builds via scripts 可视化 提高效率 更快的编码工作量与最小的努力 错误修复速度提高 通过脚本可复制的构建	入 开发代码输入		
Integrated development environment (IDE) security plugins 集成开发环境 (IDE) 安全插件	Scan and analyze the code as the developer writes it, notify developer of potential code weakness and may suggest remediation 在开发人员编写代码时对其进行扫描和分析, 通知开发人员潜在的代码漏洞, 并可能建议补救措施	Address source code weaknesses and aid developers to improve secure coding skills 解决源代码缺陷并帮助开发人员提高安全编码技能	Source code; known weaknesses 源代码; 已知的弱点	source code weakness findings 源代码弱点发现	Objective 目的
Source code repository 源代码库	Source code version control Branching and merging Collaborative code review 源代码版本控制 分支和合并协作 代码审查	Compare files, identify differences, and merge the changes if needed before committing. Keep track of application builds 在提交之前, 比较文件, 识别差异并合并更改 (如果需要)。跟踪应用程序构建	Source code Infrastructure as code 源代码 基础架构即代码	Version controlled source code 源代码版本控制	MVP 最小可用产品
Source code repository security plugin 源代码库安全性插件	Check the changes for suspicious content such as Secure Shell (SSH) keys, authorization tokens, passwords and other sensitive information before pushing the changes to the main repository. If it finds suspicious content, it notifies the	Helps prevent passwords and other sensitive data from being committed into a version control repository 帮助防止将密码和其他敏感数据提交到版本控制存储库中	Locally committed source code 本地提交的源代码	Security findings and warnings 安全发现和警告	Objective 目的

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

	<p>developer and blocks the commit.</p> <p>源代码存储库安全插件在将变更推送到主存储库之前，请检查更改是否包含可疑内容，例如 Secure Shell (SSH) 密钥，授权令牌，密码和其他敏感信息。如果发现可疑内容，它将通知开发人员并阻止提交。</p>				
<p>Code quality review tool</p> <p>代码质量审查工具</p>	<p>View code changes, identify defects, reject or approve the changes, and make comments on specific lines. Sets review rules and automatic notifications to ensure that reviews are completed on time.</p> <p>查看代码变更，发现缺陷，拒绝或批准更改以及在特定行上发表评论。设置审阅规则和自动通知，以确保按时完成审阅。</p>	<p>Automates the review process which in turn minimizes the task of reviewing the code.</p> <p>使检查流程自动化，从而使检查代码的任务最小化。</p>	<p>Source code</p> <p>源代码</p>	<p>Review results (reject or accept), code Comments</p> <p>查看结果（拒绝或接受），代码注释</p>	<p>Objective</p> <p>目的</p>

The activities supported by the develop phase are listed in Table 7.

表 7 中列出了开发阶段支持的活动。

表 7Develop Phase Activities 开发阶段活动

Activities 活动	Description 描述	Inputs 输入	Outputss 输出	Tool Dependencies 工具依赖
<p>Application code development</p> <p>应用代码开发</p>	<p>Application coding</p> <p>应用编码</p>	<p>Developer coding input</p> <p>开发人员编码输入</p>	<p>Source code</p> <p>源代码</p>	<p>IDE</p>
<p>Infrastructure code development</p> <p>基础架构代码开发</p>	<p>-System components and infrastructure orchestration coding</p> <p>-Individual component configuration script coding</p> <p>-系统组件和基础架构流程编码</p>	<p>Developer coding input</p> <p>开发人员编码输入</p>	<p>Source code</p> <p>源代码</p>	<p>IDE</p>

	-单个组件配置脚本编码			
Security code development 安全代码开发	Security policy enforcement script coding 安全策略实施脚本编码	Developer coding input 开发人员编码输入	Source code 源代码	IDE
Test development 测试开发	Develop detailed test procedures, test data, test scripts, test scenario configuration on the specific test tool 在特定的测试工具上开发详细的测试程序, 测试数据, 测试脚本, 测试方案配置	Test plan 测试计划	Test procedure document; Test data file; Test scripts 测试程序文件; 测试数据文件; 测试脚本	IDE; Specific test tool IDE; 专用测试工具
Database Development 数据库开发	Implement the data model using data definition language or data structure supported by the database; Implement triggers, views or applicable scripts; Implement test scripts, test data generation scripts. 使用数据库支持的数据定义语言或数据结构实现数据模型; 实现触发器, 视图或适用的脚本; 实施测试脚本, 测试数据生成脚本。	Data model 数据模式	Database artifacts (including data definition, triggers, view definitions, test data, test data generation scripts, test scripts, etc.) 数据库工件 (包括数据定义, 触发器, 视图定义, 测试数据, 测试数据生成脚本, 测试脚本等)	IDE or tools come with the database software 数据库软件随附的 IDE 或工具
Code commit 代码提交	Commit source code into version control system 将源代码提交到版本控制系统中	Source code 源代码	Version controlled source Code 版本控制的源代码	Source code repository 源代码库
Code commit scan 代码提交扫描	Check the changes for sensitive information before pushing the changes to the main repository. If it finds suspicious content, it notifies the developer and blocks the commit. 在将变更推送到主存储库之前, 检查变更以获取敏感信息。 如果发现可疑内容, 它将通知开发人员并阻止提交。	Locally committed source code 本地提交的源代码	Security findings and warnings 安全发现和警告	Source code repository security plugin 源代码库安全性插件
Code review 代码审查	Perform code review to all source code.	Source code 源代码	Review comments 回顾评论	Code quality review tool 代码质量检测工具

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

	Note that pair programming counts. 对所有源代码执行代码审查。 请注意，配对编程很重要。			
Documentation 文档	Detailed implementation documentation 详细的实施文档	User input; Source code 用户输入; 源代码	Documentation; Auto generated Application Programming Interface (API) documentation 文件; 自动生成的应用程序编程接口 (API) 文档	c
Static code scanbefore commit 提交前进行静态代码扫描	Scan and analyze the code as the developer writes it. Notify developers of potential code weakness and suggest remediation. 在开发人员编写代码时对其进行扫描和分析。 通知开发人员潜在的代码弱点并建议补救措施。	Source code; known weaknesses 源代码\已知弱点	source code\ weakness findings 源代码\弱点识别	IDE security Plugins IDE 安全插件
Container or VM hardening 容器或虚拟机加固	Harden the deliverable for production deployment 加固交付产品以进行生产部署	Running VM or container 运行虚拟机或容器	Vulnerability report and recommended mitigation 漏洞报告和建议的缓解措施	Container security tool Security compliance tool 容器安全工具 安全合规工具

4.2.3 Build 构建

The build tools perform the tasks of building and packaging applications, services, and microservices into artifacts. For languages like C++, building starts with compiling and linking. The former is the act of turning source code into object code and the latter is the act of combining object code with libraries to create an executable file. For Java Virtual Machine (JVM) based languages, building starts with compiling to class files, then building file such as a jar, war or ear file, which includes some metadata, and may include other files such as icon images. For interpreted languages, such as Python or JavaScript, there is no need to compile, but lint tools help to check for some potential errors such as syntax errors. Building should also include generating documentation, such as Javadoc, copying files like libraries or icons to appropriate locations, and creating a distributable file such as a tar or zip file. The build script should also include targets for running automated unit tests.

构建工具执行将应用程序，服务和微服务构建和打包到工件中的任务。对于 C ++之类的语言，构建始于编译和链接。前者是将源代码转换为目标代码的行为，后者是将目标代码与库组合以创建可执行文件的行为。对于基于 Java 虚拟机 (JVM) 的语言，构建首先要编译 class 文件，然后构建诸如 jar, war 或 ear 文件之类的文件，其中包含一些元数据，并且可能包含其他文件（如图标图像）。对于诸如 Python 或 JavaScript 之类的解释型语言，无需进行编译，但是 lint 工具可帮助检查某些潜在错误，例如语法错误。构建还应包括生成文档（例如 Javadoc），将文件（例如库或图标）复制到适当的位置，以及创建可分发文件（例如 tar 或 zip 文件）。构建脚本还应包括用于运行自动化单元测试的目标。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Modern build tools can also be integrated into both an IDE and a source code repository to enable building both during development and after committing. For those applications that use containers, the build stage also includes a containerization tool.

现代构建工具也可以集成到 IDE 和源代码存储库中，以支持在开发期间和提交后进行构建。对于那些使用容器的应用程序，构建阶段还包括一个容器化工具。

The following tables list build-related tools along with their inputs and outputs.

下表列出了与构建相关的工具及其输入和输出。

表 8 Build Phase Tools 构建阶段工具

Tool 工具	Features 特征	Benefits 收益	Inputs 输入	Outputs 产出	Baseline 基线
Build tool 构建工具	Dependency Management Compile Link (if appropriate) Built-in lint stylistic checking Integration with IDE -依赖关系 -管理 -编译 -链接（如果适用） -内置的 lint 样式检查 -与 IDE 集成	Reduces human mistakes Saves time 减少人为错误，节省时间	Source code under version control Artifacts 版本控制工件库下的源代码	Binary artifacts stored in the Artifact repository 存储在工件库中的二进制工件	MVP 最小化可用产品
Lint tool Lint 工具	Analyzes source code to flag programming errors, bugs, stylistic errors, and suspicious constructs. Applicable to both compiled or interpreted languages 分析源代码以标记编程错误，bug，风格错误和可疑结构。适用于编译或解释语言	Improve code readability; Pre-code review; Finding (syntax) errors before execution for interpreted languages 提高代码可读性；预编码审查；在执行解释后的语言之前发现（语法）错误	Source code or scripts 源代码或脚本	Analyze Results 分析结果	Objective 目的
Container Builder 容器构建器	Build a container image based on a build instruction file 根据构建命令文件构建容器镜像	Container image build automation 自动构建容器镜像	Container base image; Container build file 容器基本镜像；容器构建文件	OCI compliant container image 符合 OCI 的容器镜像	MVP 最小化可用产品
Artifact Repository 工件仓库	Binary artifact version control; Container registry 符合 OCI 的容器镜像 二进制工件版本控制；	Separate binary control from source control to avoid external access to source control system. Improved build stability by reducing reliance on external	Artifacts 工件	Version controlled artifacts 版本控制的工件	MVP 最小化可用产品

	容器注册	repositories. Better quality software by avoiding outdated artifacts with known issues. 将二进制控制与源代码控制分开，以避免外部访问源代码控制系统。 通过减少对外部存储库的依赖来提高构建稳定性。 避免出现已知问题的过时工件，从而提高软件质量。			
Static Application Security Test (SAST) tool 静态应用程序安全测试 (SAST) 工具	SAST analyzes application static codes, such as source code, byte code, binary code, while they are in a nonrunning state to detect the conditions that indicate code weaknesses. SAST 分析应用程序静态代码（例如源代码，字节代码，二进制代码），它们在系统非运行状态时扫描代码弱的情况。	Catch code weaknesses at an early stage. Continuous assessment during development. 尽早发现代码弱点。在开发流程中不断评估。	Source code; known vulnerabilities and weaknesses 源代码；已知的漏洞和弱点	Static code scan report and recommended mitigation. 静态代码扫描报告和建议的缓解措施。	MVP
Dependency checking /Bill of Materials (BOM) checking tool 依赖性检查/资产清单 (BOM) 检查工具	Identify vulnerabilities in the dependent components based on publicly disclosed open source vulnerabilities 根据公开披露的开源漏洞识别相关组件中的漏洞	Secure the overall application; Manage the supply chain risk 保护整个应用程序；管理供应链风险	Dependency list or BOM list 依赖关系列表或 BOM 清单	Vulnerability report 漏洞报告	Objective 目的

The activities supported by the build phase are listed in Table 9.

表 9 中列出了构建阶段所需的活动。

表 9 Build Phase Activities 构建阶段活动

Activities 活动	Description 描述	Inputs 输入	Outputs 输出	Tool Dependencies 工具依赖
---------------	----------------	-----------	------------	---------------------------

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Build 构建	Compile and link 编译和链接	Source code; dependencies 源码; 依赖关系	Binary artifacts 二进制工件	Build tool; Lint tool; Artifact repository 构建工具; LINT 工具; 工件库
Static application security test and scan 静态应用安全测试和扫描	Perform SAST to the software system 执行 SAST 到软件系统	Source code; known vulnerabilities and weaknesses 源代码; 已知的漏洞和 弱点	Static code scan report and recommended mitigation. 静态代码扫描报告和建 议的缓解措施。	SAST tool SAST 工具
Dependency vulnerability Checking 依赖脆弱性检查	Identify vulnerabilities in the open source dependent components 识别开源相关组件中的 漏洞	Dependency list or BOM list 依赖关系列表或 BOM 表	Vulnerability report 漏洞报告	Dependency checking / BOM checking tool 依赖性检查/BOM 检查工 具
Containerize 集成	Packages all required components OS, developed code, libraries, etc.) into a hardened container 将所有必需的组件 OS, 开发的代码, 库等打包 到一个加固的容器中	Container base image; Container build file 容器基本镜像; 容器构建文件	Container Image 容器镜像	Container builder 容器镜像
Release Packaging 发布打包	Package binary artifacts, container or VM images, infrastructure configuration scripts, proper test scripts, documentation, checksum, digital signatures, and release notes as a package. 将二进制工件, 容器或 VM 镜像, 基础结构配置 脚本, 适当的测试脚 本, 文档, 校验和, 数 字签名以及发行说明打 包为一个软件包。	Binary artifacts; Scripts; Documentation; Release notes 将二进制工件, 容器 或 VM 镜像, 基础结构配 置脚本, 适当的测试脚 本, 文档, 校验和, 数 字签名以及发行说明打 包为一个软件包。	Released package with checksum and digital signature 发布带有校验和和数 字签名的软件包	Release packaging tool 发布包工具
Store artifacts 工件仓库	Store artifacts to the artifact repository 将工件存储到工件库	Binary artifacts; Database artifacts; Scripts; Documentation; Container images 二进制工件; 数据库工 件; 脚本; 文档; 容 器镜像	Versioned controlled artifacts 版本受控的工件	Artifact Repository 工件仓库
Build configuration control and audit 建立配置控制和审核	Track build results, SAST and dependency checking report; Generate action items; Make go/no-go decision to the next phase 跟踪构建结果, SAST 和 依赖性检查报告; 产 生行动项目; 决定是 否进入下一个阶段	Build results; SAST report; Dependency checking report 构建结果; SAST 报 告; 依赖性检查报告	Version controlled build report; Action items; Go/no-go decision 版本控制的构建报告; 行动项目; 通过/不通 过的决定	Team collaboration system; Issue tracking system; CI/CD orchestrator 团队协作系统; 事件追踪系统; CI / CD 编排器

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

4.2.4 Test 测试

Test tools support continuous testing across the software development lifecycle. Test activities may include, but are not limited to, unit test, functional test, integration test, system test, regression test, acceptance test, performance test, and variety of security tests. . Mission programs can select their own test activities and merge several tests together based on the nature of their software and environment. All tests start with test development, which develops detailed test procedures, test scenarios, test scripts, and test data. Automated test can be executed by running a set of test scripts or running a set of test scenarios on the specific test tool without human intervention. If full automation is not possible, the highest percentage of automation is desired. It is highly recommended to leverage emulation and simulation to test proper integration between components such as microservices and various sensors/systems, so integration testing can be automated as much as possible. Automation will help achieve high test coverage and make continuous ATO practicable, as well as significantly increase the quality of delivered software.

测试工具支持整个软件开发生命周期中的持续测试。测试活动可以包括但不限于单元测试，功能测试，集成测试，系统测试，回归测试，验收测试，性能测试以及各种安全测试。任务程序可以选择自己的测试活动，并根据其软件 and 环境的性质将多个测试合并在一起。所有测试均始于测试开发，后者会开发详细的测试流程，测试方案，测试脚本和测试数据。可以通过在特定的测试工具上运行一组测试脚本或运行一组测试场景来执行自动化测试，而无需人工干预。如果无法实现完全自动化，则需要最高的自动化百分比。强烈建议利用模拟和仿真来测试组件之间的正确集成，例如微服务和各种传感器/系统，因此集成测试可以尽可能地自动化。自动化将有助于实现较高的测试覆盖率，并使连续的 ATO 切实可行，并显着提高所交付软件的质量。

The components involved with the test phase are listed in the following table.
下表列出了测试阶段涉及的组件。

表 10Test Phase Tools 测试阶段工具

Tool 工具	Features 特征	Benefits 收益	Inputs 输入	Outputs 产出	Baseline 基线
Test development tool 测试开发工具	Assists test scenario, test script, and test data development. The specific tool varies, depending on the test activity (such as unit test, penetration test) and the application type (e.g., web application, or Hadoop data analytics) 协助测试方案，测试脚本和测试数据开发。具体工具会有所不同，具体取决于测试活动（例如单元测试，渗透测试）和应用程序类型（例如	Increase the automation and rate of testing 提高自动化程度和测试速度	Test plan 测试计划	test scenarios, test scripts, test data 测试场景，测试脚本，测试数据	MVP

	Web 应用程序或 Hadoop 数据分析)				
Test data generator 测试数据生成器	Generates test data for the system (such as network traffic generator, web request generator) 生成系统的测试数据（例如网络流量生成器，Web 请求生成器）	Increase test fidelity 提升测试真实度	Test scenario, test data 测试场景，测试数据	Input data for the system under test 输入被测系统的数据	Objective 目的
Test tool suite 测试工具套件	A set of test tools to perform unit test, interface test, system test, integration test, performance test and acceptance test of the software system. Generate test report Specific tool varies depending on the type of tests, software application, and programming language 一套测试工具，用于执行软件系统的单元测试，接口测试，系统测试，集成测试，性能测试和验收测试。生成测试报告特定工具因测试类型，软件应用程序和编程语言而异	Increase test automation, speed 提高测试自动化速度	Test scenario, test scripts, test data 测试场景，测试脚本，测试数据	Test results, test report 测试结果，测试报告	MVP
Test coverage tool 测试覆盖工具	Measures how much code is exercised while the automated tests are running 衡量自动化测试运行时执行了多少代码	Shows the fidelity of the test results 显示测试结果的真实性	Application code, automated tests 应用程序代码，自动化测试	The percentage of code that is exercised by the tests. 测试执行的代码百分比。	MVP
Test Management Tool 测试管理工具	Manages requirements, streamlines test case design from requirements, plans test activities, manages test environment, tracks test status	Increases QA team collaboration and streamlines test processes. 加强质量检查团队的协作并简化测试流程。	Requirements ,test cases, test results 需求，测试用例，测试结果	Test progress, test results statistics 测试进度，测试结果统计	Objective

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	and results. 管理需求，根据需求简化测试用例设计，计划测试活动，管理测试环境，跟踪测试状态和结果。				
Non-security compliance scan 非安全合规性扫描	Such as Section 508 accessibility compliance 如第 508 条可访问性合规	Ensures Compliance 确保合规	Artifacts 工件	Compliance Report 合规性报告	Objective
Software license compliance checker 软件许可证合规性检查器	Inventory software license; Audit the compliance. 库存软件许可证； 审计合规性。	Software license compliance and software asset management 软件许可证合规性和软件资产管理	Purchased license info; Software instances 购买的许可证信息； 软件实例	Compliance Report 合规报告	Objective
Dynamic Application Security Test (DAST) tool 动态应用程序安全测试 (DAST) 工具	DAST tools analyze a running application dynamically and can identify runtime vulnerabilities and environment related issues. DAST 工具可以动态分析正在运行的应用程序，并且可以识别运行时漏洞和与环境相关的问题。	Catch the dynamic code weakness in runtime and under certain environment setting. Identify and fix issues during continuous integration. 在运行时和特定环境设置下捕获动态代码弱点。识别并修复持续集成流程中的问题。	Running software application; fuzz inputs 运行软件应用程序； 模糊输入	dynamic code scan report and recommended mitigation. 动态代码扫描报告和建议的缓解措施。	Objective
Interactive Application Security Test (IAST) tool 交互式应用程序安全测试 (IAST) 工具	Analyze code for security vulnerabilities while the application is run by an auto-test, human tester, or any activity "interacting" with the application functionality 在应用运行时进行自动测试、人工测试或者任何与应用程序进行交互的活动时，分析代码中的安全漏洞	Provide accurate results for fast triage; pinpoint the source of vulnerabilities 提供准确的结果以进行快速分类； 查明漏洞来源	Running application, and operating systems; Fuzz inputs 运行应用程序和操作系统； 模糊输入	Analysis report and recommended mitigation. 分析报告和建议的缓解措施。	Objective
Container security tool 容器安全工具	Container image scan OS check 容器镜像扫描 OS 检查	Ease the container hardening process	Container images or running containers 容器镜像或运行中的容器	Vulnerability report and recommended mitigation. 漏洞报告和建议	MVP

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

		简化容器加固流程		的缓解措施。	
Container policy enforcement 容器策略执行	Support for Security Content Automation Protocol (SCAP) and Container configuration policies. These policies can be defined as needed. 支持安全内容自动化协议 (SCAP) 和容器配置策略。可以根据需要定义这些策略。	Automated policy enforcement 自动化政策执行	Policies in SCAP form. SCAP 形式的策略。	Compliance Report 合规报告	MVP
Security compliance tool 安全合规工具	Scan and report for compliance regulations, such as DISA Security Technical Implementation Guides (STIGs), NIST 800-53. 扫描并报告合规性法规, 例如 DISA 安全技术实施指南 (STIG), NIST 800-53。	Speed up ATO process. 加快 ATO 流程。	Container images. 容器镜像	Vulnerability report and recommended mitigation. 漏洞报告和建议的缓解措施。	Objective
Network security test tool 网络安全测试工具	Simulate real-world legitimate traffic, distributed denial of service (DDOS), exploits, malware, and fuzzing. 模拟现实世界中的合法流量, 分布式拒绝服务 (DDOS), 漏洞利用, 恶意软件和模糊测试。	Validate system security; increase attack readiness; reduce the risk of system degradation. 验证系统安全性; 增加攻击准备; 降低系统降级的风险。	Test Configuration 测试配置	Test traffic 测试流量	Objective
Database test tool suite 数据库测试工具套件	Tools that facilitate database test; It includes test data generator, database functional test tool, database load test tool; 促进数据库测试的工具; 它包括测试数据生成器, 数据库功能测试工具, 数据库负载测试	Automate or semiautomate the database tests 自动化或半自动化数据库测试	Test data; Test scenario 测试数据, 测试场景	Test results 测试结果	Objective if using a database 目标如果使用数据库

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

	试工具;				
Database security scan and test tool 数据库安全扫描和测试工具	Find the database common security vulnerabilities, such as weak password, known configuration risks, missing patches; Structured Query Language (SQL) injection test tool; Data access control test; User access control test; Denial of service test 查找数据库常见的安全漏洞, 如弱密码、已知配置风险、缺少补丁等; 结构化查询语言注入测试工具; 数据访问控制测试; 用户访问控制测试; 拒绝服务测试	Reduce the security risks 降低安全风险	Test data; Test scenarios 试验数据; 测试场景	Vulnerability findings; Recommended mitigation actions 脆弱性调查结果; 建议的缓解措施	Objective if using a database 目标如果使用数据库

The activities supported by the test phase are listed in Table 11. These activities happen at different test stages.

- Development stage: unit test, SAST discussed in the build phase
- System test stage: DAST or IAST, integration test, system test
- Pre-production stage: manual security test, performance test, regression test, acceptance test, container policy enforcement, and compliance scan

Test audit, test deployment, and configuration audit happen at all stages.

表 11 中列出了测试阶段支持的活动。这些活动发生在不同的测试阶段。

- 开发阶段: 单元测试, 在构建阶段讨论了 SAST
- 系统测试阶段: DAST 或 IAST, 集成测试, 系统测试
- 生产前阶段: 手动安全性测试, 性能测试, 回归测试, 验收测试, 容器策略执行和合规性扫描

测试审核, 测试部署和配置审核在所有阶段进行。

表 11 Test Phase Activities 测试阶段活动

Activities 活动	Description 描述	Inputs 输入	Outputs 输出	Tool Dependencies 工具依赖
Unit test 单元测试	Assist unit test script development and unit test execution. It is typically language specific. 协助单元测试脚本开发和单元测试执行。它通常是特定于语言的。	Unit test script, individual software unit under test (a function, method or an interface), test input data, and expected output data 单元测试脚本、被测的单个软件单元 (函数、方法或接口)、测	Test report to determine whether the individual software unit performs as designed. 确定单个软件单元是否按设计执行的测试报告。	Test tool suite, Test coverage tool 测试工具套件, 测试覆盖工具

		试输入数据和预期输出数据		
Dynamic application security test and scan 动态应用安全测试与扫描	Perform DAST or IAST testing to the software system 进行 DAST 或者 IAST 测试到软件系统	Running application and underlying OS; fuzz inputs 运行应用程序和底层操作系统；模糊输入	Vulnerability, static code weakness and/or dynamic code weakness report and recommended mitigation 漏洞、静态代码弱点和/或动态代码弱点报告和 建议的缓解措施	DAST tool or IAST tool DAST 工具或者 IAST 工具
Integration test 集成测试	Develops the integration test scripts and execute the scripts to test several software units as a group with the interaction between the units as the focus. 开发集成测试脚本并执行脚本，将多个软件单元作为一个整体进行测试，以单元之间的交互为重点。	Integration test scripts, the software units under test, test input data, and expected output data 集成测试脚本、被测软件单元、测试输入数据和预期输出数据	Test report about whether the integrated units performed as designed. 集成单元是否按设计要求运行的试验报告。	Test tool suite 测试工具套件
System test 系统测试	System test uses a set of tools to test the complete software system and its interaction with users or other external systems. 系统测试使用一套工具来测试整个软件系统及其与用户或其他外部系统的交互。	System test scripts, the software system and external dependencies, test input data and expected output data 系统测试脚本、软件系统和外部依赖关系、测试输入数据和预期输出数据	Test result about if the system performs as designed. 关于系统是否按设计执行的测试结果。	Test tool suite 测试工具套件
Manual security test 手动安全测试	Such as penetration test, which uses a set of tools and procedures to evaluate the security of the system by injecting authorized simulated cyber-attacks to the system. CI/CD orchestrator does not automate the test, but the test results can be a control point in the pipeline. 例如渗透测试，它通过向系统注入授权的模拟网络攻击，使用一套工具和程序来评估系统的安全性。 CI/CD 编排器不会自动执行测试，但测试结果可以是流水线中的控制点。	Running application, underlying OS, and hosting environment 运行应用程序、底层操作系统和宿主环境	Vulnerability report and recommended mitigation 脆弱性报告和 建议的缓解措施	Varies tools and scripts (may include network security test tool) 各种工具和脚本（可能包括网络安全测试工具）

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Performance test 性能试验	Ensure applications will perform well under the expected workload. The test focus is on application response time, reliability, resource usage and scalability. 确保应用程序在预期的工作负载下运行良好。测试的重点是应用程序响应时间、可靠性、资源使用率和可伸缩性。	Test case, test data, and the software system 测试用例、测试数据和软件系统	Performance Metrics 绩效指标	Test tool suite, Test data generator 测试工具套件，测试数据生成器
Regression test 回归试验	A type of software testing to confirm that a recent program or code change has not adversely affected existing features. 一种软件测试，用于确认最近的程序或代码更改没有对现有功能产生不利影响。	Functional and nonfunctional regression test cases; the software system 功能和非功能回归测试用例；软件系统	Test report 测试报告	Test tool suite 测试工具套件
Acceptance test 验收试验	Conduct operational readiness test of the system. It generally includes: Accessibility and usability test failover and recovery test performance, stress and volume test security and penetration test interoperability test compatibility test supportability and maintainability 对系统进行战备测试。一般包括： 可访问性和可用性测试故障转移和恢复测试性能、压力和容量测试安全和渗透测试互操作性测试兼容性测试可支持性和可维护性	The tested system Supporting system Test data 被测系统支持系统测试数据	Test report 试验报告	Test tool suite, Non-security compliance scan 测试工具套件，非安全合规性扫描
Container policy Enforcement 容器强制策略	Check developed containers to be sure they meet container policies 检查已开发的容器，确保它们符合容器策略	Container, Policies in SCAP form 容器，SCAP 形式的策略	Container compliance report 容器合规报告	Container policy enforcement 容器强制策略
Compliance Scan 容器扫描	Compliance audit 容器审计	Artifacts; Software instances; System components	Compliance Reports 合规性报告	Non-security compliance scan; Software license

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

		工件，软件实例，系统组件		Compliance checker; Security compliance tool 非安全合规扫描； 文件许可证符合性检查； 安全合规工具
Test audit 测试审计	Test audit keeps who performs what test at what time and test results in records 测试审计将在什么时间执行什么测试以及测试结果存档	Test activity and test results 测试活动和测试结果	Test audit log 测试审计日志	Test management tool 测试管理工具
Test deployment 测试部署	Deploy application and set up testing environment using Infrastructure as Code 使用基础架构作为代码部署应用程序并设置测试环境	Artifacts (application artifacts, test code) Infrastructure as Code 工件（应用程序工件、测试代码）作为代码的基础设施	The environment ready to run tests 环境已准备好运行测试	Configuration automation tool; IaC 配置自动化工具
Database functional test 数据库功能测试	Perform unit test and functional test to database to verify the data definition, triggers, constraints are implemented as expected 对数据库进行单元测试和功能测试，验证数据定义、触发器、约束是否按预期实现	Test data 测试数据	Test results 测试结果	Database test tools 数据库测试工具
Database nonfunctional test 数据库非功能测试	Conduct performance test, load test, and stress test; Conduct failover test 进行性能测试、负载测试和压力测试；进行故障转移测试	Test data; Test scenarios 试验数据； 测试场景	Test results 测试结果	Database test tools 数据库测试工具
Database security test 数据库安全测试	Perform security scan; Security test 执行安全扫描；安全测试	Test data; Test scenarios 试验数据； 测试场景	Test results 测试结果	Vulnerability findings; Recommended mitigation actions 脆弱性调查结果； 建议的缓解措施
Test Configuration control and audit 测试配置控制和审计	Track test and security scan results; Generate action items; Make go/no-go decision to the next phase. (There may be several iterations for several tests across stages) 跟踪测试和安全扫描结果；生成操作项；对下一阶段做出通过/不通过的决定。 (跨阶段的多个测试可能有多个迭代)	Security scan and compliance scan report 安全扫描和符合性扫描报告	Version controlled test results; Action items; Go/no-go decision 版本控制测试结果；行动项目；通过/不通过决定	Team collaboration system; Issue tracking system; CI/CD orchestrator 团队协作系统；问题跟踪系统；CI/CD 编排器

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

4.2.5 Release and Deliver

In the release and deliver phase, the software artifacts are digitally signed to verify that they have passed build, all tests, and security scans. They are then delivered to the artifact repository. The content of the artifacts depends on the application. It may include, but is not limited to, container images, VM images, binary executables (such as jar, war, ear files), test results, security scan results, and Infrastructure as Code deployment scripts. Artifacts will be tagged with the release tag if GO release decision is made based on the configuration audit results. The artifacts with the release tag are delivered to production.

在发布和交付阶段，对软件工件制品进行数字签名，以验证它们是否通过了构建、所有测试和安全扫描。然后将它们交付到工件存储库。工件的内容取决于应用程序。它可以包括但不限于容器镜像、VM 镜像、二进制可执行文件（例如 jar、war、ear 文件）、测试结果、安全扫描结果以及作为代码部署脚本的基础结构。如果根据配置审计结果做出 GO 发布决策，那么工件将被标记为 release 标签。带有 release 标签的工件被交付到生产环境中。

表 12Release and Deliver Phase Tools 发布和交付阶段工具

Tool 工具	Features 特征	Benefits 收益	Inputs 输入	Outputs 产出	Baseline 基线
Release packaging tool 发布打包工具	Package binary artifacts, container or VM images, infrastructure configuration scripts, proper test scripts, documentation, release notes as a package; generate checksum and digital signature for the package. The package may be prepared for a specific installer or it is a self-extracting installer itself. 将二进制工件、容器或 VM 镜像、基础设施配置脚本、适当的测试脚本、文档、发行说明打包为一个包；为包生成校验和数字签名。 包可以是为特定的安装程序准备的，也可以是自解压安装程序本身。	Release package (such as a bundle of artifacts, self-extracting software installer, software tar file, etc.) 发布包（如一个工件制品包、自解压软件安装程序、软件 tar 文件等）	Binary artifacts, base containers or VM images, infrastructure configuration scripts, proper test scripts, documentation, release notes 二进制工件，基本容器或虚拟机镜像、基础架构配置脚本等 测试脚本、文档、发行说明	Release package with checksum and digital signature (a bundle of artifacts, such as a self-extracting software 带有校验和数字签名的发布包（一组工件，如自解压软件）	

The mission program could have more than one artifact repository, though more likely there is a centralized one and tags separate artifact types. One artifact repository (or set of tags) is used in the build stage to store build results. The test deployment activity can fetch the artifacts from the build stage artifact repository to deploy the application into various environments (development, test, or pre-production). Another artifact repository (or set of tags)

may be used by the production environment, which is the one that the store artifacts stage uses to push the final deliverables to production. The production deployment will get all the artifacts from the production artifact repository to deploy the application.

任务程序可能有不止一个工件库，不过更有可能有一个集中的工件库，并标记不同的工件类型。在构建阶段使用一个工件存储库（或一组标记）来存储构建结果。测试部署活动可以从构建阶段工件库获取工件，以便将应用程序部署到各种环境（开发、测试或预生产）中。生产环境可以使用另一个工件存储库（或一组标记），存储工件阶段使用该存储库将最终可交付成果推送到生产环境。生产部署将从生产工件库获取所有工件，以部署应用程序。

Some mission program application systems have geographically distributed operational regions across the country or even overseas. In order to increase deployment velocity, a remote operational region may have its own local artifact repository that replicates the artifact repository completely or partially. During release, a new artifact is pushed into the artifact repository and then replicated to other regional artifact repositories.

The activities supported by the release and deliver phase are listed below.

一些任务程序应用系统在全国甚至海外都有地理分布的作战区域。为了提高部署速度，远程操作区域可以有自己的本地工件存储库，它可以完全或部分复制工件存储库。在发布期间，一个新的工件被推入工件库，然后复制到其他区域工件库。

表 13Release and Deliver Phase Activities 发布和交付阶段活动

Tool 工具	Features 特征	Benefits 收益	Inputs 输入	Outputs 产出	Baseline 基线
Release go / no-go decision 放行/不放行决定	This is part of configuration audit; Decision on whether to release artifacts to the artifact repository for the production environment. 这是配置审计的一部分；决定是否将工件发布到生产环境的工件仓库。	Design documentation; Test reports; Security test and scan reports; Artifacts 设计文档；测试报告；安全测试和扫描报告；工件	go / no-go decision; Artifacts are tagged with release tag if go decision is made 执行/不执行决策；如果执行决策，工件将打上发布标签	CI/CD Orchestrator CD/CD 编排器	
Deliver released artifacts 交付发布的工件	Push released artifacts to the artifact repository 将发布的工件推送到工件仓库	The release package 发布包	New release in the artifact repository 工件仓库中的新版本	Artifacts Repository 工件仓库	
Artifacts Replication 工件构建	Replicate newly release artifacts to all regional artifact repositories 将新发布的工件复制到所有区域工件仓库	Artifacts 工件	Artifacts in all regional artifact repositories 所有区域工件仓库中的工件	Artifacts repositories (release, regional) 工件仓库（区域、发布）	

4.3 Production Operation Tools and Activities 生产运营工具与活动

The production operations tools provide the capability to deploy artifacts, including containers and VM images, to the production environment, and to monitor their operations. For Cloud-native applications, it is recommended to use containers whenever possible over virtual machines due to the baked-in security provided by the Sidecar Container Security Stack.

生产运营工具提供了将工件（包括容器和 VM 镜像）部署到生产环境并监视其运行的功能。对于云原生应用程序，由于边车容器安全堆栈（SCSS）提供了内置安全性，建议尽可能在虚拟

机上使用容器。

4.3.1 Deploy 部署

The tools used in deploy phase are environment and deployment mode dependent.

While it is highly recommended to leverage containers for new system design and development, if the application is deployed as a VM, the virtualization manager in the hosting environment is the key component with which IaC will interface to deploy and configure the application system. The virtualization manager manages the virtual compute, storage and network resources. In some hosting environments, such as a general-purpose cloud, the virtualization manager also provides some security capabilities, such as micro-segmentation, which creates security zones to isolate VMs from one another and secure them individually. Several capabilities of the virtualization manager are keys to the success of mission application runtime operation and security, such as health checking, virtual resource monitoring, and scaling. The application production environment infrastructure has to leverage these capabilities in its architecture and configuration.

部署阶段使用的工具依赖于环境和部署模式。

虽然强烈建议利用容器进行新系统设计和开发，但如果将应用程序部署为 VM，则宿主环境中的虚拟化管理器是 IaC 部署和配置应用程序系统的关键组件。虚拟化管理器管理虚拟计算、存储和网络资源。在一些托管环境中，例如通用云，虚拟化管理器还提供了一些安全功能，例如微隔离，它创建了安全区域，将 vm 彼此隔离开，并进行单独保护。虚拟化管理器的几个功能是任务应用程序运行时操作和安全的成功关键，例如健康检查、虚拟资源监视和扩展。应用程序生产环境基础架构必须在其体系结构和配置中利用这些功能。

The use of “clones” from a master image library enables VMs to be created quickly. A clone is made from a snapshot of the master image. The use of clones also enables the concept of immutable infrastructure by pushing updated, clean images to the VM each time it is started. Only the master image needs to be patched or updated with the latest developed code; each running image is restarted to pick up these changes.

使用主镜像库中的“克隆”可以快速创建虚拟机。从主镜像的快照进行克隆。克隆还使用了不可变基础设施的概念，每次启动时都将更新的、干净的镜像推送到 VM。只有主镜像需要用最新开发的代码修补或更新；每个正在运行的镜像都会重新启动以获取这些更改。

4.3.1.2 Container deployment 容器部署

A container manager provides capabilities that check for new versions of containers, deploys the containers to the production environment, and performs post-deployment checkout.

容器管理器提供检查新版本容器的能力，部署容器到生产环境，并且执行部署后校验。

The container manager consists of an OCI-compliant container runtime and a CNCF-certified Kubernetes, which is an orchestration tool for managing microservices or containerized applications across a cluster of nodes. The nodes could be bare metal servers or VMs. The container manager may be owned by a mission program or provided by the cloud hosting environment. It simplifies container management tasks, such as instantiation, configuration, scaling, monitoring, and rolling updates. The CNCF-certified Kubernetes interacts with the underlying virtualization manager in the cloud environment to ensure each node's health and performance, and scale it as needed. This scaling includes container scaling within the CNCF-certified Kubernetes cluster, but when running in a cloud, it also includes the ability to auto-scale number of nodes in a cluster by adding or deleting VMs.

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

容器管理器由一个符合 OCI 的运行时容器和一个 CNCF 认证的 Kubernetes 组成，Kubernetes 是一个编排工具，用于跨节点集群管理微服务或容器化应用程序。

节点也可以是裸金属服务器或虚拟机。容器管理器可以由任务程序拥有，也可以由云托管环境提供。它简化了容器管理任务，例如实例化、配置、缩放、监视和滚动更新。CNCF 认证的 Kubernetes 与云环境中的底层虚拟化管理器进行交互，以确保每个节点的健康和性能，并根据需要进行扩展。这种扩展包括 CNCF 认证的 Kubernetes 在集群内的容器扩展，但是当在云中运行时，它还包括通过添加或删除 vm 来自动扩展集群中节点数量的能力。

The following tables list deployment-related tools and their inputs and outputs.

下表列举了部署相关工具以及输入输出。

表 14 Deploy Phase Tools 部署阶段工具

工具	功能	收益	输入	输出	基线
Virtualization Manager 虚拟化管理器	VM instance management VM resource monitoring (provided on hosting environment) VM 实例管理 VM 资源监控（宿主环境提供）	Centralized VM instantiation, scaling, and monitoring 集中化 VM 实例化，扩展与监控	VM instance specification and monitoring policy VM 实例规范与监控策略	Running VM 运行 VM	MVP if using VMs MVP 如使用 VM
CNCF-certified Kubernetes CNCF 认证的 Kubernetes	Container grouping using pods Health checks and self-healing Horizontal infrastructure scaling Container auto-scalability Domain Name Service (DNS) management Load balancing Rolling update or rollback Resource monitoring and logging 使用 pods 的容器分组 健康检查和自我康复 横向基础设施扩展 容器自动可扩展性 域名服务（DNS）管理 负载均衡 滚动更新或回滚 资源监视和日志记录	Simplify operations by deployment and update automation Scale resources and applications in real time Cost savings by optimizing infrastructure resources 通过部署和更新简化操作 自动化实时扩展资源 and 应用程序 通过优化基础架构资源节约成本	Container instance specification and monitoring policy 容器实例规范与监控策略	Running container 运行容器	MVP
Data masking tool 数据脱敏工具	Shield personally identifiable information or other confidential data 防护个人身份信息或其他机密数据	Provide data privacy; Reduce the risk of data loss during data breach 提供数据隐私 减少数据违规中的数据泄露风险	Original data 原始数据	Masked data 脱敏数据	Objective if database contains sensitive data 目标数据库是否包含敏感数据
Database encryption tool 数据库加密工具	Encrypt data at rest and in transit 动态与静态数据加密	Provide data privacy and security; Prevent data loss 提供数据隐私与安全 数据方泄露	Original data 原始数据	Encrypted data 脱敏数据	MVP if database contains highly sensitive data MVP 如包含高敏感数据

Database automation tool 数据库自动化工具	Automate database tasks, such as deployments, upgrades, discovering and troubleshooting anomalies, recovering from failures, topology changes, running backups, verifying data integrity, and scaling. 自动化数据库任务, 例如部署, 升级, 发现异常并进行故障排除, 从故障中恢复, 拓扑更改, 运行备份, 验证数据完整性和扩展性。	Simplify database operations and reduce human errors 简化数据库操作, 减少人为错误	Database artifacts; Data; Running status and events 数据库工件数据; 运行状态和事态	Status report; Warnings; Alerts 状态报告; 警告与告警	Objective if using a database 目标是否使用数据库
Configuration automation tool 配置自动化工具	Execute the configuration scripts to provision the infrastructure, security policy, environment, and the application system components. 执行配置脚本以提供基础设施, 安全策略, 环境与信息系统组件	Configuration automation Consistent provisioning 配置自动化持续提供	Infrastructure configuration scripts Infrastructure configuration data 基础设施配置脚本 基础设施配置数据	Provisioned deployment infrastructure 已提供的部署架构	MVP
Service mesh 服务网格	Ability to create a network of deployed services with load balancing, service-to-service authentication, and monitoring. 能够通过负载均衡, 服务到服务的身份验证和监控来创建已部署服务的网络。	Support for microservice interactions. 支持微服务交互	Control plane: service communication routing policies, authentication certificates. Data plane: service communication data 控制平面: 服务沟通路由策略, 认证证书 数据平面: 服务沟通数据	Control plane: service status reports Data plane: routed service communication data 控制平面: 服务状态报告 数据平面: 路由数据沟通数据	MVP

The activities supported by the deploy phase are listed in Table 15.

部署阶段所需的活动如表 15 所示。

表 15 Deploy Phase Activities 部署阶段活动

活动	描述	输入	输出	工具依赖关系
Artifact download 工件下载	Download newly release artifacts from the artifact repository 从工件库下载新的工件	Artifact download request 工件库下载需求	Requested artifacts 需要的工件	Artifact repository 工件仓库
Infrastructure provisioning automation 基础设施提供自动化	Infrastructure systems auto provisioning (such as software defined networking, firewalls, DNS, auditing and logging system, user/group permissions, etc.) 基础设施系统自动配置 (例如软件定义的网络, 防火墙, DNS, 审核和日志记录系统, 用户/组权限等)	Infrastructure configuration scripts / recipes / manifests / playbooks 基础设施配置脚本/方法/清单/剧本	Provisioned and configured infrastructure 已提供已配置的基础架构	Configuration automation tools; IaC 配置自动化工具; IaC
Create linked clone of VM master image 创建 VM 主镜像的链接克隆	Instantiate VM by creating a link clone of parent VM with master image 通过创建父 VM 的主镜像的链接克隆来实例化 VM	VM parent New VM instance parameters 新的父 VM 实例参数	New VM instance 新的 VM 实例	Virtualization Manager 虚拟化控制器
Deliver container to container registry 交付容器到容器注册表	Upload the hardened container and associated artifacts to the container registry 上传加固容器与关联工件到容器注册表	Hardened container 已加固容器	New container instance 新的容器实例	CNCF-certified Kubernetes; Artifact repository container registry

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

				CNCF 认证的 Kubernetess; 工件库容器注册表
Post-deployment security scan 部署后安全扫描	System and infrastructure security scan 系统和基础架构安全扫描	Access to system components and infrastructure components 访问系统组件与基 础设施组件	Security vulnerability findings 安全漏洞发现	Security compliance tool 安全合规工具
Post-deployment checkout 部署后校验	Run automated test to make sure the important functions of system are working 运行自动化测试以确认重要系统功 能正常	Smoke test scenarios and test scripts 冒烟测试场景与测 试脚本	Test results 解释结论	Test scripts 测试脚本
Database installation and database artifact deployment 数据库安装与数据 库工件部署	Database software installation; Cluster or high availability setup; Database artifacts deployment and data loading 数据库软件安装，集群与高可用安 装，数据库工件部署与数据读取	Artifacts in the repository; Data 在库工件； 数据	Running database system 运行数据库系 统	Artifact repository; Database automation tool; Data masking or encryption tool if needed 工件仓库； 数据库自动化工具； 所需到数据脱敏或加密 工具

4.3.2 Operate 运行

The Operate phase uses tools for system scaling, load balancing, and backup.

Load balancing monitors resource consumption and demand, and then distributes the workloads across the system resources. Scaling helps dynamic resource allocation based on demand. Both virtualization manager and CNCF-certified Kubernetes support load balancing and scaling capabilities. CNCF-certified Kubernetes handles the load balancing and scaling at the container level. The virtualization manager works at the VM level.

Application deployment must have proper load balancing and scaling policies configured with the virtualization manager or the CNCF-certified Kubernetes based on VM deployment or container deployment respectively. During runtime, the management layer will continuously monitor the resources. If the configured threshold is met (for example if memory or Central Processing Unit (CPU) usage meets a pre-set threshold), then the system triggers the load balancing or scaling action automatically. Auto-scaling must be able to scale both up and down.

操作阶段使用工具进行系统扩展、负载均衡和备份。负载均衡监视资源消耗和需求，然后跨系统资源分配工作负载。扩展有助于根据需求动态分配资源。虚拟化管理器和 CNCF 认证的 Kubernetes 都支持负载均衡与扩展功能。CNCF 认证的 Kubernetes 在容器级别处理负载均衡和扩展，虚拟化管理器则在虚拟机层。

应用程序部署必须分别基于虚拟机部署或容器部署，使用虚拟化或 CNCF 认证的 Kubernetes 配置适当的负载均衡和扩展策略。在运行期间，管理层将持续监控资源。如果达到配置的阈值（例如，如果内存或中央处理器（CPU）使用率满足预先设置的阈值），则系统会自动触发负载均衡或扩展。自动扩展必须既可以向上也可以向下。

The activities supported by the operate phase are listed in the table below.

运行阶段支持的活动列举如下表。

表 16Operate Phase Tools 运行阶段工具

工具	功能	收益	输入	输出	基线
Backup management 备份管理	Data backup System components (VM or container) snapshot 数据备份系统组件（VM 或组件）快照	Improve failure recovery 提升失败恢复	Access to the backup source 访问备份资源	Backup data System VM or container snapshot 备份数据系统 VM 或容器	MVP
Operations dashboard 运营仪表盘	Provide operators a visual view of operations status, alerts, and actions. 为操作员提供操作状态的直观视图，警报和操作。	Improve operations management 提升运营管理	All operational monitoring status, alerts, and recommended actions 全部运营监控、状态、告警与推荐措施	Dashboard display 仪表盘展示	Objective 目标

表 17 Operate Phase Activities 运行阶段活动

活动	描述	输入	输出	工具依赖关系
Backup 备份	Data backup; System backup 数据备份 系统备份	Access to backup system 访问备份系统	Backup data or image 备份数据或镜像	Backup management; Database automation tool 备份管理 数据库自动化工具
Scale 扩展	Scale manages VMs/containers as a group. The number of VMs/containers in the group can be dynamically changed based on the demand and policy. 扩展将虚拟化/容器作为一个组进行管理。租种虚拟机/容器的数量可以根据需求和政策动态修改	Real-time demand and VM/container performance measures Scale policy (demand or Key Performance Indicator (KPI)threshold; minimum, desired, and maximum number of VMs/containers) 实时需求和虚拟机/容器性能度量 扩展策略（需求或关键绩效指标（KPI）阈值；最小，所需和最大数量的虚拟机/容器）	Optimized resource allocation 优化资源分配	VM management capability on the hosting environment; Container management on the hosting environment 宿主环境的 VM 管理能力； 宿主环境的容器管理
Load balancing 负载均衡	Load balancing equalizes the resource utilization 负载均衡可均衡资源利用率	Load balance policy Real time traffic load and VM/container performance measures 负载均衡策略实时流量负载与 VM /容器性能指标	Balanced resource utilization 平衡资源利用率	VM management capability on the hosting environment; Container management on the hosting environment 宿主环境中虚拟机管理能力 宿主环境中容器管理能力

4.3.3 Monitor 监控

In the monitor phase, tools are utilized to collect and assess key information about the use of the application to discover trends and identify problem areas. Monitoring spans the underlying hardware resources, network transport, applications / microservices, containers, interfaces, normal and anomalous endpoint behavior, and security event log analysis.

在监视阶段，使用工具来收集和评估有关应用程序使用的关键信息，以发现趋势和确定问题区域。监控涵盖底层硬件资源、网络传输、应用程序/微服务、容器、接口、正常和异常终端行为以及安全事件日志分析。

It also includes behavior and signature-based detection in the runtime environment. All these security capabilities are mapped against the NIST controls and follow NIST Special Publication 800-190 Application Container Security Guide [12] for continuous compliance.

它还包括运行时环境中基于行为和签名的检测。所有这些安全功能都对应于 NIST 控制，并遵循 NIST 专门出版物 800-190 《应用程序容器安全指南》[12]，以实现持续合规性。

表 18 Monitor Phase Tools 监控阶段工具

工具	功能	收益	输入	输出	基线
Logging 日志记录	Logging events for all user, network, application, and data activities 记录所有用户、网络应用与数据行为的日志	Assist troubleshooting the issues. Assist detection of advanced persistent threats and forensics. 协助解决问题。 协助检测高级持续性威胁和取证。	All user, network, application, and data activities 所有用户、网络、应用与数据行为	Event logs 事态日志	MVP
Log aggregator 日志汇聚	Filter log files for events of interest (e.g., security), and transform into canonical format 过滤日志文件中感兴趣的事件（例如安全性），并转换为规范格式		Logs 日志	Aggregated, filtered, formatted event log 汇聚的、过滤的、格式化的事态日志	MVP
Log analysis & auditing 日志分析与审计	Analyze and audit to detect malicious threats / activity; Automated alerting and workflows for response Forensics for damage assessment 分析和审计用于检测恶意威胁/活动； 自动化告警与工作流用于响应		Logs 日志	Alert messages, emails, etc. Remediation report and log 告警信息、邮件等 修复报告和日志	MVP
Operations monitoring 运营监控	Report various performance metrics such as resource utilization rates, number of concurrent user sessions, and Input/Output (IO) rates; Provide dashboards to display performance; Alert performance issues Establish a baseline for comparison 报告各种性能指标，例如资源利用率，并发用户会话数和输入/输出（IO）率； 提供显示性能仪表板；告警性能问题建立比较基准	Improve operations continuity Identify the area to improve Better end-user experience 提升运营持续性，识别最终用户更优体验的改进领域	Performance KPI and Service Level Agreement (SLA) 性能 KPI 与 SLA	Performance statistics Performance alerts 性能统计性能告警	MVP

Information Security Continuous Monitoring (ISCM) 信息安全持续监控 (ISCM)	<p>Monitor network security Monitor personnel activity Monitor configuration changes Perform periodical security scan to all system components Monitor the IT assets and detect deviations from security, fault tolerance, performance best practices. Monitor and analyze log files Audit IT asset's configuration compliance Detect and block malicious code Continuous security vulnerability assessments and scans Provide browse, filter, search, visualize, analysis capabilities Generate findings, assessments and recommendations. Provide recommendations and/or tools for remediating any non-compliant IT asset and/or IT workload.</p> <p>监控网络安全 监督人员活动 监控配置更改 定期对所有系统组件进行安全扫描 监控 IT 资产并检测与安全性、容错性的偏差 性能最佳实践 监控和分析日志文件 审核 IT 资产的配置符合性 检测和阻止恶意代码 持续的安全漏洞评估和扫描 提供浏览、过滤、搜索、可视化、分析能力 提出调查结果、评估和建议。 提供建议和/或工具，以纠正任何不合规的 IT 资产和/或 IT 工作负载。</p>	<p>Detect unauthorized personnel, connections, devices, and software Identify cybersecurity vulnerability Detect security and compliance violation Verify the effectiveness of protective measures 检测未经授权的人员、连接、设备和软件 识别网络安全漏洞 检测安全性和合规性违规 验证防护措施的有效性</p>	<p>IT asset Network Personnel activities Known vulnerabilities IT 资产 网络 个人行为 已知漏洞</p>	<p>Vulnerabilities Incompliance Findings, assessments and recommendations 漏洞，不合规发现 评估与推荐</p>	MVP
Alerting and notification 告警与通知	<p>Notify security teams and/or administrators about detected events. Support automatic remediation of high-priority time-critical events. 将检测到的事件通知安全团队和/或管理员。 支持高优先级关键时间事件的自动修正。</p>	<p>Improve visibility of system events Reduce system downtime Improve customer service 提高系统事件的可见性 减少系统停机时间 改善客户</p>	<p>Logs, monitoring data. Support automatic remediation of high-priority time-critical events. 日志 监控数据 高优先级关键时间的事件</p>	<p>Alert messages, emails, etc. Remediation report Issue ticket 告警信息，邮件等； 修正报告； 事件工单；</p>	MVP
Database monitoring tool 数据库监控工具	<p>Baseline database performance and database traffic; Detect anomalies 基线数据库 性能与数据库流量 基线数据库性能和数据库流量； 检测异常</p>	<p>Improve database operations continuity 改进数据库运营持续性</p>	<p>Running database 运行数据库</p>	<p>Logs; Warnings and alerts 日志； 警告与告警</p>	Objective if using a database 如使用数据库的目标
Database security audit tool 数据库安全审计工具	<p>Perform user access and data access audit; Detect anomalies from events correlation; Detect SQL injection; Generate alert 执行用户访问与数据访问审计； 从事件关联中检测到异常； 检测 SQL 注入； 产生告警</p>	<p>Enhance database security 提升数据库安全</p>	<p>Running database 运行数据库</p>	<p>Audit logs; Warnings and alerts 审计日志； 警告与告警</p>	如使用数据库的 MVP

The activities supported by the monitor phase are listed in Table 19.

监控阶段所需的活动如表 19 所示。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

表 19 Monitor Phase Activities 监控阶段活动

活动	描述	输入	输出	工具依赖关系
Logging 记录日志	Log system events 记录系统时间	All user, network, application, and data activities 所有用户 网络 应用 与数据活动	Logs 日志	Logging 记录日志
Log analysis & auditing 日志分析	Filter or aggregate logs; Analyze and correlate logs 过滤或汇聚日志; 分析与纠正日志	Logs 日志	Alerts and remediation report 告警与修正报告	Log aggregator Log analysis& auditing 汇聚日志 日志分析与审计
System performance monitoring 系统性能监控	Monitor system hardware, software, database, and network performance; Baselining system performance; Detect anomalies 监控系统硬件, 软件, 数据库, 与网络性能; 设定系统性能基线 检测异常	Running system 运行系统	Performance KPI measures; Recommended actions; Warnings or alerts 测量性能 KPI; 建议活动; 警告或告警	Operation monitoring Issue tracking system; Alerting and notification; Operations dashboard 运行监控; 事件追踪系统; 告警与通知; 运行仪表盘
System Security monitoring 系统安全监控	Monitor security of all system components Security vulnerability assessment System security compliance scan 监控所有系统组件安全; 安全漏洞评估 系统合规性扫描	Running system 运行系统	Vulnerabilities; Incompliance Findings; assessments and recommendations; Warnings and alerts. 漏洞; 不合规发现; 评估与建议 警告与告警	ISCM; Issue tracking system; Alerting and notification; Operations dashboard ISCM: 事件追踪系统; 告警与通知; 运行仪表盘
Asset Inventory 资产清单	Inventory system IT assets IT 资产清单系统	IT assets IT 资产	Asset inventory 资产清单	Inventory Management; 清单管理
System configuration monitoring 系统配置监控	System configuration (infrastructure components and software) compliance checking, analysis, and reporting 系统配置 (基础架构组件与软件) 合规检查, 分析与报告	Running system configuration; Configuration baseline 运行系统配置; 配置基线	Compliance report; Recommended actions; Warnings and alerts 合规性报告; 推荐的行动 警告与告警	ISCM; Issue tracking system; Alerting and notification; Operations dashboard ISCM: 事件追踪系统; 告警与通知; 运行仪表盘
Database monitoring and security auditing 数据库监控与安全审计	Database performance and activities monitoring and auditing 数据库性能与行为监控与审计	Database traffic, event, and activities 数据库流量, 事态, 与活动	Logs; Warnings and alerts 日志; 警告与告警	Database monitoring tool; Database security audit tool; Issue tracking system; Alerting and notification; Operations dashboard 数据库监控工具; 数据库安全审计工具

4.4 Security Tools and Activities Summary 安全工具与活动概述

Security is not a separate phase of the DevSecOps lifecycle; rather security activities occur in all phases. This DevSecOps security practice facilitates automated risk characterization, monitoring, and mitigation across the application lifecycle. Table 20 summarizes the security activities of all phases.

安全不是 DevSecOps 生命周期的单独阶段，而是安全活动发生在全部阶段。这个 DevSecOps 安全实践有助于在整个应用程序生命周期中自动进行风险特征描述，监控和缓解。表 20 汇总了全部阶段的安全活动。

表 20 Security Activities Summary 安全活动概述

活动	阶段	活动表参考	工具依赖关系	工具表参考
Threat modeling 风险建模	Plan 规划	Table 4	Threat modeling tool 威胁建模工具	Table 3
Security code development 安全代码开发	Develop 开发	Table 7	IDE	Table 6
Static code scan before commit 提交前静态代码扫描	Develop 开发	Table 8	IDE security plugins IDE 安全插件	Table 6
Code commit scan 代码提交扫描	Develop 开发	Table 9	Source code repository security plugin 源代码仓库安全插件	Table 6
Container or virtual machine hardening 容器或虚拟机加固	Develop 开发	Table 7	Container security tool Security compliance tool 容器安全工具； 容器合规插件	Table 10
Static application security test and scan 静态应用安全测试和扫描	Build 构建	Table 9	SAST tool SAST 工具	Table 8
Dependency vulnerability checking 依赖关系漏洞检查	Build 构建	Table 9	Dependency checking / BOM checking tool 依赖关系检查/BOM 检查工具	Table 8
Dynamic application security test and scan 动态应用安全测试和扫描	Test 测试	Table 11	DAST tool or IAST tool DAST 工具或 IAST 工具	Table 10
Manual security testing (such as penetration test) 手动安全测试（如渗透测试）	Test 测试	Table 11	Varies tools and scripts (may include network security test tool) 各类工具与脚本（可能包括网络安全测试工具）	Table 10
Container policy enforcement 容器策略执行	Test 测试	Table 11	Container policy enforcement 容器策略执行	Table 10
Post-deployment security scan 部署后安全扫描	Deploy 部署	Table 15	Security compliance tool 安全合规工具	Table 10
System Security monitoring 系统安全监控	Monitor 监控	Table 19	Information Security Continuous Monitoring (ISCM) 信息安全持续监控（ISCM）	Table 18

4.5 Configuration Management Tools and Activities Summary 配置管理工具与活动概述

Configuration management plays a key role in DevSecOps practice. It ensures the configuration of a software system's infrastructure, software components, and functionalities are not only known initially but also knowable and well controlled throughout the DevSecOps lifecycle.

配置管理在 DevSecOps 实践中起到关键作用。它确保软件系统中基础架构、软件组件和功能的配置不仅最初是已知的，而且在 DevSecOps 的整个生命周期中也是已知的和良好受控的。

Configuration management consists of three sets of activities:

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

配置管理包含三组活动

- Configuration identification: identify the configuration items. This can be done manually or with assistance from a discovery tool. The configuration items include infrastructure components, COTS or open source software components used in the system, documented software design, features, software code or scripts, artifacts, etc.
配置识别：识别配置项。 这可以手动完成，也可以在发现工具的帮助下完成。 配置项包括基础架构组件，系统中使用的 COTS 或开源软件组件，文档化的软件设计，功能，软件代码或脚本，工件等。
- Configuration control: control the changes of the configuration items. Each configuration item has its own attributes, such as model number, version, configuration setup, license, etc. The CMDB, source code repository, and artifact repository are tools to track and control the changes. The source code repository is used primarily during development. The other two are used in both development and operations.
配置控制：配置项的变更控制。 每个配置项都有其自己的属性，例如型号，版本，配置设置，许可证等。CMDB、源代码仓库和工件仓库是跟踪和控制变更的工具。 源代码仓库主要在开发期间使用。 另外两个用于开发和运营中。
- Configuration verification and audit: verify that the configuration items meet the documented requirements and design. Configuration verification and audit are control gates along a pipeline to control the go/no-go decision to the next phase.
配置验证和审计：验证配置项目是否满足记录的要求和设计。 配置验证和审计是一道控制门，用于控制在流水线中通过/不通过进入下一阶段的决策。

表 21 Configuration Management Activities Summary

活动	阶段	活动表参考	工具依赖关系	工具表参考
Configuration management planning 配置管理规划	Plan 规划	Table 4	Team collaboration system; Issue tracking system 团队协作系统; 事件追踪系统;	Table 3
Configuration identification 配置项识别	Plan 规划	Table 4	CMDB	Table 3
Design review 设计评审	Plan 规划	Table 4	Team collaboration system 团队协作系统	Table 3
Documentation version control 文件版本控制	Plan 规划	Table 4	Team collaboration system 团队协作系统	Table 3
Code review 代码评审	Develop 开发	Table 7	Code quality review tool 代码质量评审工具	Table 6
Code Commit 代码提交	Develop 开发	Table 7	Source code repository 源代码仓库	Table 6
Store artifacts 存储工件	Build 构建	Table 9	Artifact repository 工件仓库	Table 8
Build phase configuration control and audit 构建阶段配置控制和审计	Build 构建	Table 9	Team collaboration system; Issue tracking system CI/CD orchestrator 团队协作系统; 事件追踪系统; CI/CD 编排器	Table 3 Table 5

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Test phase configuration control and audit 测试阶段配置控制和审计	Test 测试	Table 11	Team collaboration system; Issue tracking system CI/CD orchestrator 团队协作系统; 事件追踪系统; CI/CD 编排器	Table 3 Table 5
Release go / no-go decision 发布 通过/不通过决策	Release 发布	Table 13	CI/CD orchestrator CI/CD 编排器	Table 5
Infrastructure provisioning automation 架构提供自动化	Deploy 部署	Table 15	Configuration automation tool 配置自动化工具	Table 14
Post-deployment security scan 部署后安全扫描	Deploy 部署	Table 15	Security compliance tool 安全合规工具	Table 19
Post-deployment checkout 部署后校验	Deploy 部署	Table 15	Test scripts 测试脚本	
Asset inventory 资产清单	Monitor 监控	Table 19	Asset inventory tool 资产清单工具	Table 18
System performance monitoring 系统性能监控	Monitor 监控	Table 19	Operation monitoring Issue tracking system; Alerting and notification; Operations dashboard 运行监控; 事件追踪系统; 告警与通知; 运营仪表板	Table 3 Table 16 Table 18
System configuration monitoring 系统配置监控	Monitor 监控	Table 19	ISCM; Issue tracking system; Alerting and notification; Operations dashboard ISCM; 事件追踪系统; 告警与通知; 运营仪表板	Table 3 Table 16 Table 18

4.6 Database Management Tools and Activities Summary 数据库管理工具与活动概述

Databases are commonly used in the DoD software systems. They hold some of the most critical information of an enterprise or a mission and are typically the center piece of the software system. Data security and privacy protection are paramount to enterprises and missions. Relational databases continue to be a prime target for data thieves, and security vulnerabilities are compound by adoption of big data platforms, such as Hadoop, NoSQL databases, and Database as a Service (DBaaS) in the cloud. Here we discuss some database activities throughout the DevSecOps lifecycle to improve database security and operations.

数据库在 DoD 软件系统中普遍使用。它们保存着企业或任务中一些最关键的信息，通常是软件系统的核心部分。数据安全和隐私保护对企业 and 任务至关重要。关系数据库仍然是数据窃取者的主要目标，而安全漏洞由于采用大数据平台（例如 Hadoop，NoSQL 数据库和云中的数据库即服务（DBaaS））而变得更加复杂。这里我们将讨论整个 DevSecOps 生命周期中的一些数据库活动，以提高数据库的安全性和操作性。

In development phases, database design, development, and testing activities generate database artifacts, which are data models, database schema files, trigger definitions, view definition, test data, test data generation scripts, test scripts, etc. These database artifacts must be under configuration management control. During test phase, database functional test is like application code unit test and functional test to validate the schema, triggers, and data compliance. The non-functional test includes load testing, stress test, and performance test. The security test focuses on vulnerability scan, user authentication and authorization, unauthorized access to data, data encryption, privilege elevation, SQL injection, and denial of service.

在开发阶段，数据库设计，开发和测试活动会生成数据库工件，这些工件是数据模型，数据库模式文件，触发器定义，视图定义，测试数据，测试数据生成脚本，测试脚本等。这些数据库工件必须受配置管理控制。在测试阶段，数据库功能测试就像应用程序代码单元测试和功能测试一样，用于验证架构，触发器和数据符合性。非功能测试包括负载测试，压力测试和性能测试。安全测试的重点是漏洞扫描，用户身份验证和授权，对数据的未经授权的访问，数据加密，特权提升，SQL 注入和拒绝服务。

During operations, the deployment and operational activities can be automated via database automation tools. The continuous monitoring is achieved using database monitoring tool and security audit tool.

在运行流程中，可以通过数据库自动化工具来自动化部署和操作活动。使用数据库监视工具和安全审核工具可以实现持续监控。

Table 22 summarizes the database activities of all phases.

表 22 总结了所有阶段的数据库活动

表 22 Database Management Activities Summary 数据库管理活动概述

Activities	阶段	活动表参考	工具依赖关系	工具表参考
Database design 数据库设计	Plan 计划	Table 4	Data modeling tool 数据建模工具	Table 3
Database development 数据库开发	Develop 开发	Table 7	IDE or tools come with the database software 数据库软件附带的 IDE 或工具	Table 6
Code review (database schemas, codes) 代码评审 (数据库 schema、代码)	Develop 开发	Table 7	Code quality review tool 代码质量评审工具	Table 6
Code Commit (database schemas, codes) 代码提交 (数据库 schema、代码)	Build 构建	Table 7	Source code repository 源代码仓库	Table 6
Store artifacts (database artifacts) 存储工件 (数据库工件)	Test 测试	Table 9	Artifact repository 工件仓库	Table 8
Database functional test 数据库功能测试	Test 测试	Table 11	Database test tool 数据库测试工具	Table 10
Database non-functional test 数据库非功能测试	Test 测试	Table 11	Database test tool 数据库测试工具	Table 10
Database security test 数据库安全测试	Test 测试	Table 11	Database security test tool 数据库安全测试工具	Table 10

Database installation and database artifact deployment 数据库安装和数据库工件部署	Test and Deploy 测试和部署	Table 11 Table 15	Artifact repository; Database automation tool; Data masking or encryption tool if needed 工件仓库; 数据库自动化工具; 所需的数据脱敏或加密工具	Table 8 Table 14
Backup 备份	Operate 运行	Table 17	Database automation tool 数据库自动化工具;	Table 14
Database monitoring and security auditing 数据库监控和安全审计	Monitor 监控	Table 19	Database monitoring tool; Database security audit tool 数据库监控工具; 数据库安全审计工具	Table 18

5 DoD Enterprise DevSecOps Container Service DoD 企业 DevSecOps 容器服务

The DoD Enterprise DevSecOps Container Service creates DevSecOps hardened containers and provides hardened container access service to DoD programs to instantiate their own DevSecOps ecosystem.

国防部企业 DevSecOps 容器服务创建 DevSecOps 加固容器，并向国防部项目提供加固容器访问服务，以实例化其自身的 DevSecOps 生态系统。

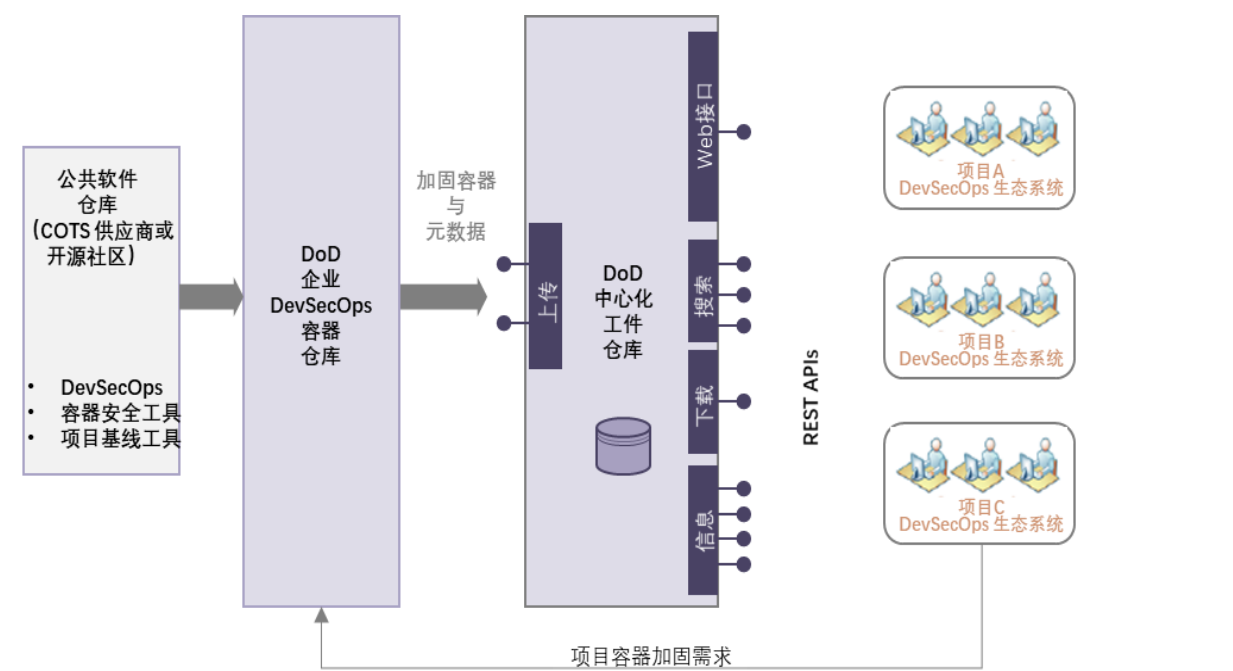


图 10 DoD 企业 DevSecOps 容器服务架构

Figure 10 illustrates the DoD Enterprise DevSecOps Container Service architecture. It contains a DoD Enterprise DevSecOps Container Factory and a DoD Centralized Artifact Repository (DCAR). The Container Factory takes public container images as input and automates the container hardening process to produce the hardened container images. The DCAR stores the hardened container images and allows DoD programs access these images.

图 10 描述了国防部企业 DevSecOps 容器服务体系结构。它包含一个 DoD 企业 DevSecOps 容

器工厂和一个国防部中心化工件仓库（DCAR）。容器工厂将公用容器镜像作为输入，并自动用容器加固流程生成加固的容器镜像。DCAR 存储加固的容器镜像，并允许 DoD 项目访问这些镜像。

5.1 DoD Enterprise DevSecOps Container Factory DoD 企业 DevSecOps 容器工厂

The Container Factory produces the hardened containers of DevSecOps tools. It is imperative for the Container Factory to automate its hardening process as much as possible. It does this by leveraging CI/CD pipelines in an instance of the software factory specifically configured for hardening of DevSecOps tool containers.

容器工厂使用 DevSecOps 工具生成加固的容器。至关重要的是，容器工厂必须尽可能自动化其加固流程。它通过在专门配置用于加固 DevSecOps 工具容器的软件工厂实例中利用 CI / CD 流水线来实现此目的。它通过利用软件工厂一个实例中的 CI/CD 流水线来实现此目的，该实例专门配置用于加固 DevSecOps 工具容器。

5.1.1 DoD Hardened Containers DoD 已加固容器

A DoD hardened container is an Open Container Image (OCI) compliant image that is secured and made compliant with the DoD Container Hardening Security Requirements Guide [6].

国防部加固容器是符合开放式容器镜像（OCI）的镜像，它是安全的且符合 DoD 容器加固安全要求指南[6]。

Container images should adhere to the OCI Image Format Specification to ensure portability. Each hardened container includes the “global configuration”, which includes all security hardening configuration. The packaged container is tagged with integrity metadata, such as a digital signature or a digital hash. Tags may be implemented as metadata bound to the object, or as attributes in a file associated with the object. Some configuration values can be changed for local use, such as the DNS location. These “local configuration” values are outside the scope of the integrity tag, and do not “break” the chain of trust for the hardened container.

容器镜像应遵循 OCI 镜像格式规范，以确保可移植性。每个加固容器都包含“全局配置”，其中包含所有安全强化配置。打包的容器使用完整性元数据（如数字签名或数字哈希）进行标记。标签可作为元数据绑定到客体上，或作为与客体关联的文件属性。一些配置值可以修改后供本地使用，如 DNS 位置。这些“本地配置”值不在完整性标签的范围内，并且不会“破坏”加固容器的信任链。

Artifacts related to the hardened container should include the Information Assurance (IA) controls that the hardened container has successfully addressed, so that users of the container know which controls they can inherit, versus which controls they must address. This capability may not exist in the MVP, but it is an objective that enables reciprocity.

与加固容器相关的工件应包括加固容器已成功解决的信息保证（IA）控制项，以便容器的用户知道他们可以继承哪些控制，以及他们还须完成哪些控制。

这种能力可能不存在于 MVP 中，但它是一个实现互惠的目标。

The Hardened Container Factory produces following types of containers:

- Hardened containers of DevSecOps CI/CD pipeline tools
- Sidecar Container Security Stack (see details in Section 6.4.4) containers to be used in runtime environments for container security
- Common containers (such as OS, database, web servers, etc.) to be used as a program development baseline

加固容器工厂生成以下类型容器：

- DevSecOps CI / CD 流水线工具的加固容器
- 边车容器安全堆栈（请参阅第 6.4.4 节中的详细信息）容器在运行时环境中用于容器安全
- 用作程序开发基线的通用容器（例如 OS，数据库，Web 服务器等）

5.1.2 Container Hardening Process 容器加固流程

The Container hardening process, including required documentation, sustainment of the hardened containers, and cybersecurity requirements, is fully described in the DoD Enterprise DevSecOps Initiative Hardening Containers [13]. The basic process is depicted in Figure 11.

容器加固流程，包括所需的文件，加固容器的维护和网络安全要求，在 DoD 企业 DevSecOps 计划加固容器[13]中有详细描述。基本流程如图 11 所示。



图 11 容器加固流程的主要步骤

5.1.2.1 Select the Container Base Image 基于镜像选择容器

A base image is a container image that comes from a vendor or an open source community; it is used as the starting point to create a hardened container image. Use the base image without creating forks to enable direct coupling with its updates. The container should be built starting with the respective DoD hardened base OS STIG image.

容器镜像的基本镜像来自供应商或开放源代码社区，用于创建加固容器镜像的起点。在不创建分支的情况下使用基本镜像以启用带更新的直接耦合。容器应该从相应的 DoD 加固的基本操作系统 STIG 镜像开始构建。

5.1.2.2 Harden the Container 容器加固

The Container Factory uses a set of instructions given in [13] to harden the container to mitigate findings and ensure proper DoD compliance. Reuse the instructions as much as possible between versions so that the hardening will be consistent across versions. It is possible that new versions bring new features, which may require additional hardening.

容器工厂使用[13]中给出的一组说明来加固容器，以缓解问题并确保符合 DoD 的要求。在加固时不同版本之间尽可能多的重用说明，这样有助于帮助不同版本之间保持一致。新版本可能引入新特性，因此可能需要进行额外加固。

The DoD Centralized Container Source Code Repository (DCCSCR) is used to store instruction files to build container images, associated checksums, and various documentation. The source code repository is centrally hosted so hardeners can store their code and leverage a CI/CD pipeline (Container Factory). This is what feeds the container hardening process and DCAR repository.

DoD 集中式容器源代码库（DCCSCR）用于存储说明文件，用于构建容器镜像、相关校验值以及各种文档。源代码仓库集中托管，因此加固人员可以存储其代码和使用 CI/CD 流水线（容器工厂）。这些都是提供给容器加固流程和 DCAR 仓库的信息。

Use the CI/CD orchestration tool; download the DCCSCR folder content into the pipeline and use the Instructions file to build the container.

使用 CI/CD 编排工具；将 DCCSCR 文件夹的内容下载到流水线中，并使用说明文件构建容器。

The CI/CD pipeline will then run the required Container Hardening Scanners, scanning the container image. Based on the findings of the Container Hardening Scanners, add instructions to mitigate the findings as needed. Rebuild and rescan until findings are mitigated or accepted.

然后，CI/CD 流水线将运行所需的容器加固扫描器，扫描容器图像。根据容器加固扫描器的发现，按需添加减轻问题的说明。重新构建和扫描，直到问题被缓解或接受。

5.1.2.3 Store the Hardened Container 存储已加固镜像

The DCAR is used to store the hardened containers, associated checksums, and various documentation. This repository will be centrally hosted. Each container will have its own folder in the DCAR. Subfolders should be used for versioning.

DCAR 用于存储已加固的容器、相关校验值以及各种文档。DCAR 仓库将集中托管，每个容器在 DCAR 中都有自己的文件夹。子文件夹应用于版本控制。

Store the hardened container and checksum inside the DCAR. It will be tagged as “pre-production” as well until the artifact receives an ATO, in which case it will then be tagged for “production”.

将已加固的容器和校验值存储在 DCAR 中。容器在工件收到 ATO 之前被标记为“准生产”，收到 ATO 后，它将被标记为“生产”。

5.1.2.4 Documentation 文件

Content and documentation provided for the hardened container will include:

为已加固容器提供的内容和文档包括：

- A description of the container, how to deploy it, the functional capabilities it provides, and its interfaces.
描述：如何部署容器、容器提供的功能和接口的描述。
- Scripts, including the instructions file for building the container, and related configuration files for deploying and scaling the hardened image or container
脚本：包括用于构建容器的说明文件，以及用于部署和缩放强化镜像或容器的相关配置文件
- Security test results including findings, false positives, and a recommended mitigation plan.
安全测试结果：包括发现、误报和缓解建议计划。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

- Accepted risks, including a Plan of Action and Milestones (POA&M) for critical and high findings that are not yet resolved.
可接受的风险：包括尚未解决的关键和重大发现的行动计划和里程碑（POA&M）。
- Change log of significant changes since the last version.
自上一版本以来重大变更的变更日志。
- Human-readable licenses for all products are included within the container. COTS license keys will not be included, and they will need to be acquired separately.
容器中包含所有产品的可读许可证。COTS许可密钥将不包括在内，需要单独获取。

5.1.2.5 Continuous Engineering 持续工程化

The hardened container factory CI/CD pipeline searches for and downloads new base images that are posted by the vendor or in an open source community repository and runs the steps in the Section 5.1.2.2. These steps are triggered automatically, as soon as a new image is released into the open source repository. If the build passes all scans, it should automatically store the new container into DCAR, where it will be tagged as “pre-production”. It also automatically notifies the team if a build fails to pass any of the scans.

加固容器工厂 CI/CD 流水线搜索并下载由供应商或开源社区仓库中发布新的基础镜像，并运行第 5.1.2.2 节中的步骤。一旦新镜像发布到开源仓库中，这些步骤立即自动触发。如果构建通过了所有扫描，它会自动将新容器存储到 DCAR 中，在那里镜像将被标记为“准生产”。如果构建未通过扫描，将会自动通知团队。

5.1.2.6 Cybersecurity 网络安全

The hardened containers produced by the factory should meet the related cybersecurity requirements, which include NIST Special Publication (SP) 800-53 [14], NIST SP 800-37 [15], the DoD DISA Security Technical Implementation Guides (STIGs) and Security Requirements Guides (SRGs), and industry best practices.

加固容器应符合相关网络安全需求，其中包括 NIST 特别出版物（SP）800-53[14]、NIST SP 800-37[15]、国防部 DISA 安全技术实施指南（STIG）和安全要求指南（SRG）以及行业最佳实践。

5.2 DoD Centralized Artifact Repository DoD 中心化工件仓库

The DCAR holds the DoD hardened container images that the DoD Enterprise DevSecOps Container Factory produces. DoD program DevSecOps teams can utilize these to instantiate their own DevSecOps ecosystem and software factory. The DCAR also holds the DoD hardened containers for base operating systems, web servers, application servers, databases, API gateways, message buses, and additional enterprise capabilities for use by DoD program software teams as a program system deployment baseline. Separate hardened container images are created for different versions of base images. DCAR hardened container images are version controlled. These hardened containers, along with security accreditation reciprocity, greatly simplify and speed the process of obtaining an Approval to Connect (ATC) or Authority to Operate (ATO).

DCAR 保存 DoD 企业级 DevSecOps 容器工厂生产的 DoD 加固容器镜像。DoD 项目 DevSecOps 团队可以利用这些来实例化他们自己的 DevSecOps 生态系统和软件工厂。DCAR 还为基本操作系统、web 服务器、应用服务器、数据库、API 网关、消息总线和其他企业能力提供 DoD 加固容器，供 DoD 项目软件团队用作项目系统部署基线。为基础镜像的不同版本创建独立的加固容器镜像。DCAR 加固容器镜像由版本控制。这些加固的容器，与安全鉴证互认一起，极大成都简化并加快了获得连接批准（ATC）或运行授权（ATO）的流程。

The DCAR provides the capability to allow DoD programs (including approved DoD contractors) to search, list information about, and download artifacts from the repository for DoD software development on the approved environment.

DCAR 提供能力允许 DoD 项目组（包括获批准的 DoD 承包商）搜索，列举相关信息，以及在获批准的环境中从仓库下载工件进行 DoD 软件开发。

6 DevSecOps Ecosystem Reference Designs DevSecOps 生态系统参考设计

This section will discuss two software factory reference designs. One is based on the DoD Enterprise DevSecOps Container Service offering to create a software factory using DevSecOps tool hardened containers from DCAR. The other is based on DoD authorized cloud DevSecOps service offerings as provided by a CSP.

本节将讨论两个软件工厂的参考设计。一个是基于国防部企业 DevSecOps 容器服务，使用 DCAR 的 DevSecOps 工具加固容器创建软件工厂。另一种是基于 DoD 授权的云 DevSecOps 服务，由 CSP 提供。

This section also discusses secure operations for containerized applications in a production environment by leveraging container security tools in the DCAR.

本节还将讨论如何利用 DCAR 中的容器安全工具在生产环境中对容器化应用程序进行安全运行。

6.1 Containerized Software Factory 容器化软件工厂

A containerized software factory can be instantiated using a set of DevSecOps hardened containers that are offered in the DCAR. These enterprise containers are preconfigured and secured to reduce the certification and accreditation burden and are often available as a predetermined pattern or pipeline that will need limited or no configuration. Figure 12 illustrates a containerized software factory reference design. The software factory is built on an underlying container orchestration layer and a host environment. It produces DoD applications as the product. These applications use different sets of hardened containers from the DCAR than the ones used to create the software factory.

容器化软件工厂可以通过 DCAR 中提供的一组 DevSecOps 加固容器来实例化。这些企业容器是预先配置的，并且为减少认证和认可负担采取了安全措施，通常作为需要限定配置或缺省配置的预定模式或流水线提供。图 12 展示了一个容器化软件工厂参考设计。软件工厂构建在底层容器编排层和主机环境上。它生成 DoD 的应用程序作为产品。这些应用程序使用不同于用于创建软件工厂的 DCAR 加固容器集。

DoD programs may have already implemented a DevSecOps platform. One of the pain points is sustaining that platform. It is highly recommended that, as incremental updates are made to the existing platform, the program migrates capabilities to the DoD Enterprise DevSecOps hardened containers. For those cases where a DoD Enterprise hardened container is not available, or requires a custom policy, the program in conjunction with the DoD Enterprise DevSecOps program office is encouraged to create, sustain, and deliver the hardened container to the DCAR.

国防部项目可能已经实施了 DevSecOps 平台。其中一个痛点是维护这个平台。强烈建议在

平台进行增量更新时，项目将能力迁移到 DoD 企业级 DevSecOps 加固容器。对于 DoD 企业加固容器不可用或需要自定义策略的情况，鼓励项目与 DoD 企业 DevSecOps 项目办公室共同创建、维护，并交付已加固容器给 DCAR。

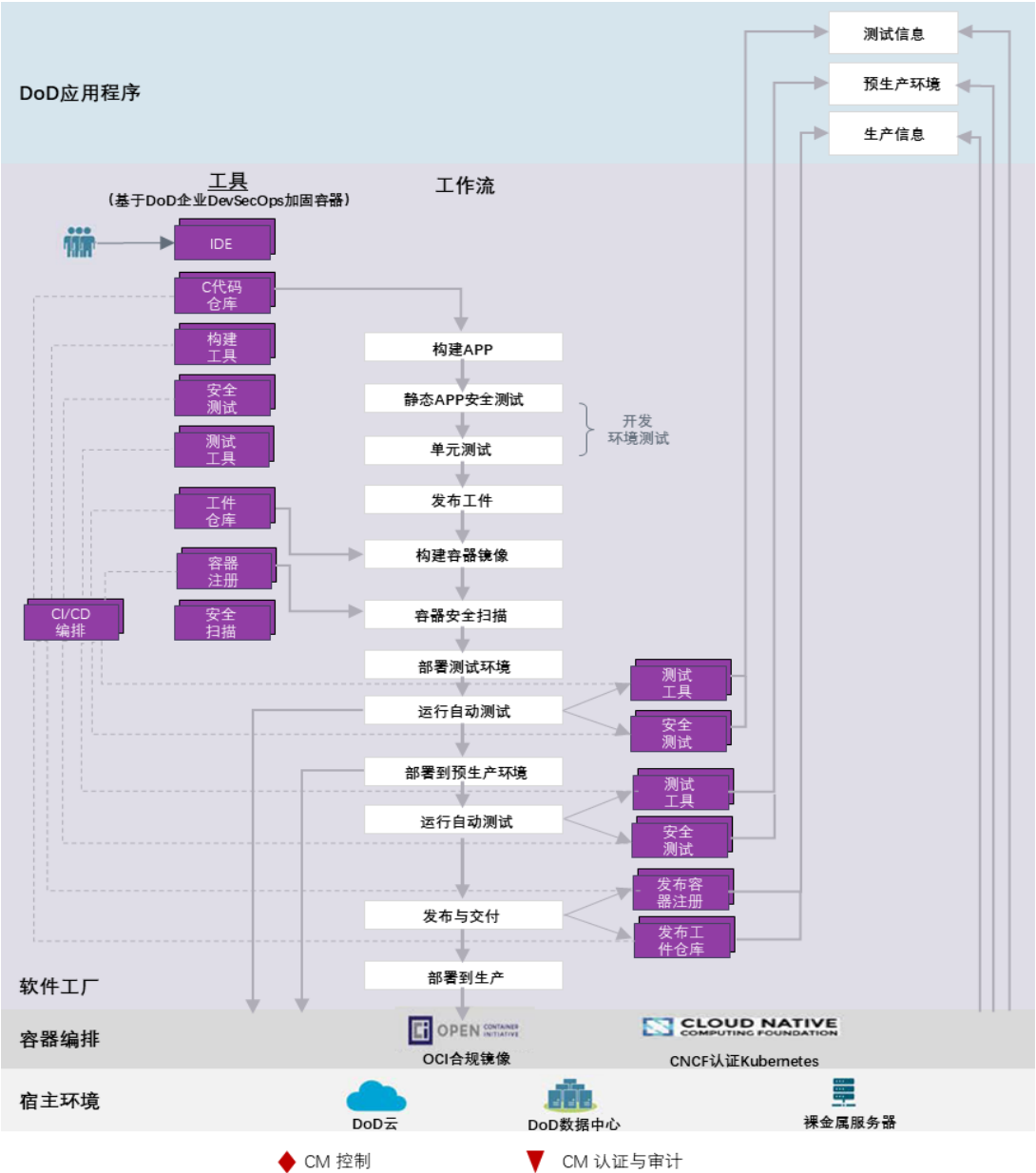


图 12 中心化软件工厂参考设计

6.1.1 Hosting Environment 托管环境

The reference design does not restrict the software factory hosting environment, which could be DoD-approved Cloud Service Providers, DoD data centers or even on-premises servers. The hosting environment provides compute, storage, and network resources in either

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

physical or virtual form.

本参考设计不限制软件工厂的托管环境，可以是 DoD 授权的云服务提供商，DoD 数据中心甚至是本地数据中心。托管环境提供物理和虚拟化的计算，存储和网络资源。

6.1.2 Container Orchestration 容器编排

In order to support containerized software factory tools, the underlying container orchestration must use CNCF certified Kubernetes and support OCI compliant containers. CNCF-certified Kubernetes orchestrates containers, interacts with underlying hosting environment resources, and coordinates clusters of nodes at scale in development, testing and pre-production in an efficient manner. There are two options for the container orchestration layer as illustrated in Figure 13.

为支持容器化软件工厂工具，底层容器编排必须使用 CNCF 认证的 Kubernetes 和支持符合 OCI 的容器。CNCF 认证的 Kubernetes 以高效的方式编排容器、与底层宿主环境资源交互，并在开发，测试和预生产环境中大规模协调节点集群。容器编排层有两个选项，如图 13 所示。

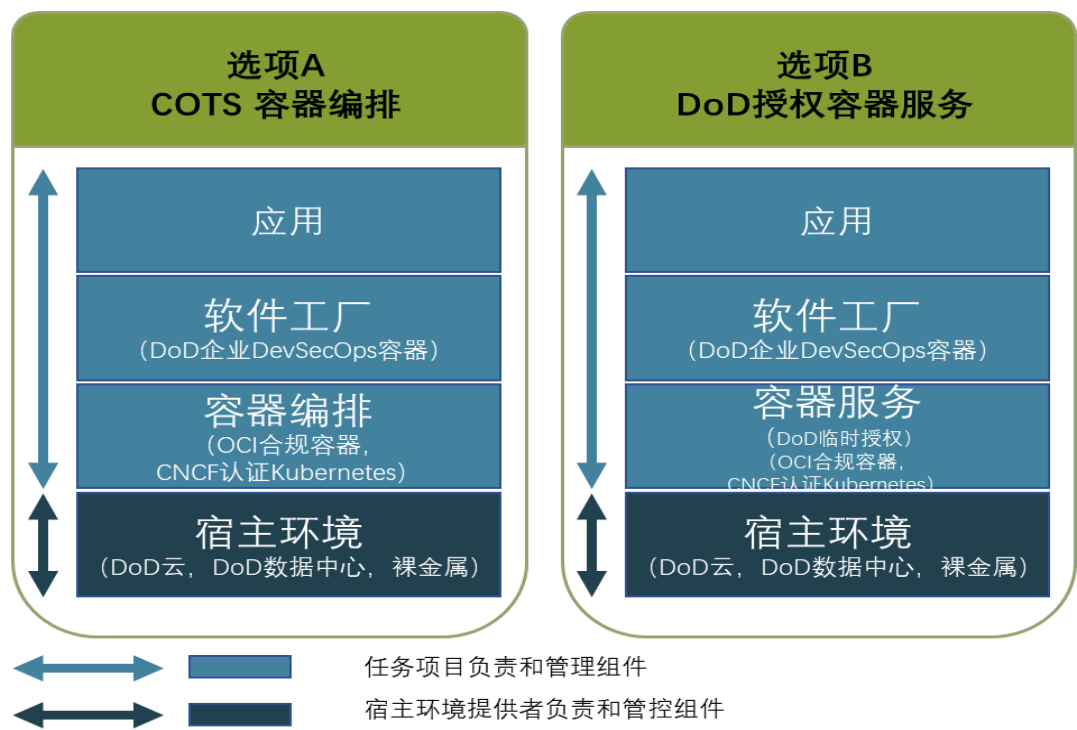


图 13 DevSecOps 平台选项

In Option A, it is the mission program's responsibility to build and maintain the container orchestration layer (CNCF-certified Kubernetes) using COTS solutions. The container orchestration layer can be deployed on top of a DoD authorized cloud environment, a DoD data center, or on bare metal servers. The container orchestration system components are subject to monitoring and security control under the DoD policy in that hosting environment, such as the DoD Cloud Computing Security Requirements Guide (SRG) [2] and DISA's Secure

Cloud Computing Architecture (SCCA) [3] for the cloud environment.

在方案 A 中，任务项目负责使用 COTS 解决方案构建和维护容器编排层（CNCf 认证的 Kubernetes）。容器编排层可以部署在 DoD 授权的云环境、DoD 数据中心或裸金属服务器上。容器编排系统组建作为客体在宿主环境中受 DoD 策略的监控和安全管控，如用于云环境的 DoD 云计算安全需求指南（SRG）[2]和 DISA 的安全云计算架构（SCCA）[3]。

In Option B, the mission program uses a CSP container service, which must have a DoD provisional authorization and must be based on CNCf-certified Kubernetes.

在方案 B 中，任务项目使用 CSP 容器服务，该服务必须具有国防部授权，并且必须基于 CNCf 认证的 Kubernetes。

6.1.3 Software Factory Using Hardened Containers 使用已加固容器的软件工厂

The software factory leverages technologies and tools to automate the CI/CD pipeline processes defined in the DevSecOps lifecycle plan phase. There are no “one size fits all” or hard rules about what CI/CD processes should look like and what tools must be used. Each software team needs to embrace the DevSecOps culture and define its processes that suit its software system architectural choices. The tool chain selection is specific to the software programming language choices, application type, tasks in each software lifecycle phase, and the system deployment platform.

软件工厂利用技术和工具自动化 DevSecOps 生命周期计划阶段中定义的 CI/CD 流水线流程。没有普适的或硬性规则说明 CI/CD 流程应该是什么样子以及必须使用什么工具，每个软件团队都需要接受 DevSecOps 文化，并定义适合自身软件系统体系架构选择的流程。工具链选择则具体视软件编程语言选择、应用程序类型、每个软件生命周期阶段中的任务以及系统部署平台的情况而定。

Software factory building itself follows the DevSecOps philosophy and goes through its own design, instantiate, verify, operate and monitor phases. It evolves through the application lifecycle iteration. Figure 14 illustrates the software factory phases, activities, and the relationship with the application lifecycle. Security must be applied across the software factory phases. Sidecar Container Security Stack (SCSS) discussed in 6.4.4 can be used for software factory runtime cybersecurity monitoring.

软件工厂建设本身遵循 DevSecOps 的理念，并经历了自身的设计、实例化、验证、运行和监控阶段。它在应用程序生命周期迭代中不断发展。图 14 说明了软件工厂阶段、活动以及与应用程序生命周期的关系。软件工厂的各个阶段必须保障安全性。6.4.4 中讨论的边车式容器安全堆栈（SCSS）可用于软件工厂运行时的网络安全监控。

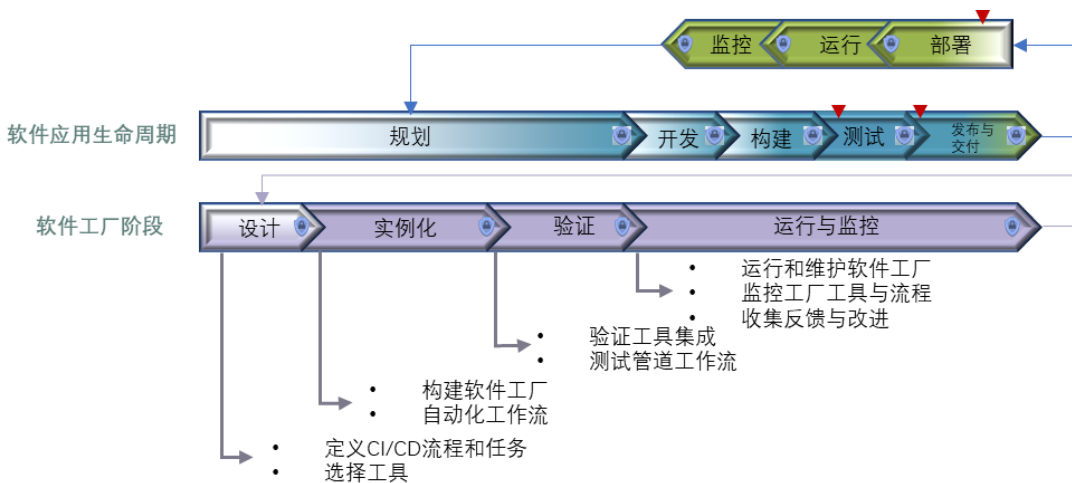


图 14 应用程序生命周期中的软件工厂阶段

Figure 12 is a software factory reference design. It includes the tools and process workflows to develop, build, test, secure, release, and deliver software application for production deployment. All the tools are based on the DoD enterprise DevSecOps hardened containers. Committing code into the code repository kicks off the automated factory CI/CD pipeline workflow. The CI/CD orchestrator executes the workflow by coordinating different tools to perform various tasks. Some tasks are completed by a set of DevSecOps tools, such as build, static code analysis, unit test, publish artifacts, build container image, etc. Other tasks may need assistance from underlying container orchestration layer, such as deploy application to test, pre-production, and final production environments. Some test and security tasks may need human involvement.

图 12 是一个软件工厂参考设计。它包括一组工具和工作流程，用于生产部署中软件应用程序的开发、构建、测试、保护、发布和交付。所有工具均基于 DoD 企业 DevSecOps 已加固容器。将代码提交到代码仓库后将触发工厂 CI/CD 流水线的自动工作流程。CI/CD 编排器通过调用不同的工具来运行各种任务来执行工作流程。一些任务由一组 DevSecOps 工具完成，例如构建，静态代码分析，单元测试，发布工件，构建容器镜像等。其他任务可能需要底层容器编排层的帮助，例如将应用程序部署到测试，准生产环境和生产环境。一些测试和安全工作可能需要人工参与。

6.1.4 DoD Applications DoD 应用程序

The term “DoD Application” refers to a DoD software program hosted by an information system [16], which spreads widely from legacy monolithic infrastructure-dependent applications to modern modular infrastructure-agnostic applications. Most systems are in brownfield with legacy applications or mixed legacy and modern applications. Programs should consider the nature of their application and the deployment environment when designing their software factory. It is recommended to leverage the Strangler Pattern [17] [18] to refactor legacy applications to modern microservices/containerized applications.

术语“DoD 应用程序”是指由信息系统[16]托管的 DoD 软件程序，该程序从传统的依赖于基础架构的整体应用程序广泛传播到与现代模块化基础设施无关的应用程序。大多数系统处于传统应用程序或传统与现代应用程序混合的灰色地带。项目在设计软件工厂时应考虑其应用程

序和部署环境的性质。建议利用 Strangler 模式[17] [18]将传统应用程序重构为现代微服务/容器化应用程序。

- DoD application environments at all phases (development, test, pre-production, production) are subject to security control from DoD common security services.
所有阶段（开发，测试，准生产，生产）的 DoD 应用程序环境均受 DoD 通用安全服务的安全管控。
- While refactoring legacy code or writing new code, it is highly recommended to leverage the DoD hardened containers or hardening scripts to facilitate the application's ATO.
重构旧代码或编写新代码时，强烈建议利用 DoD 加固的容器或加固脚本来简化应用程序的 ATO。

6.2 Software Factory using Cloud DevSecOps Services 使用云 DevSecOps 服务的软件工厂

The DoD authorized cloud may already or will soon offer DevSecOps services, such as code repository, artifact repository, build service, code deploy service, etc. Programs should consider using these native managed services as alternatives to self-built and self-maintained DevSecOps tool sets, but they should understand that using them may lead to vendor lock-in with the CSP. The CSP is responsible for maintaining the service offering and the DoD Provisional Authorization (PA) for each service. The program still needs to follow the software factory lifecycle and performs the design, DevSecOps service selection instead of tool selection, CI/CD pipeline process workflow automation, verify the workflows, operates and monitors the workflows.

DoD 授权的云可能已经或即将提供 DevSecOps 服务，例如代码仓库，工件仓库，构建服务，代码部署服务等。项目应考虑使用这些原生管理服务替代自建和自维护的 DevSecOps 工具集，但他们应该意识到，使用它们可能会导致与 CSP 供应商绑定。CSP 负责维护服务产品和每项服务的 DoD 临时授权（PA）。该程序仍然需要遵循软件工厂的生命周期并执行设计，DevSecOps 用服务选择代替工具选择，CI / CD 流水线流程工作流程自动化，验证工作流程，运行和监控工作流。

Another consideration is that the CSP may not offer a full solution set. For the capability that a DevSecOps service with a DoD PA is not available, the corresponding DoD Enterprise hardened container that provides the proper capability should be used. A CNCF-certified Kubernetes container orchestration service is required for container runtime. Figure 15: Software Factory illustrates a software factory using both cloud DevSecOps services and self-maintained security tools.

另一个考虑是 CSP 可能无法提供完整的解决方案集。当带有 DoD PA 的 DevSecOps 服务的功能无法使用时，应使用提供适当功能的相应 DoD Enterprise 加固容器。容器运行时需要 CNCF 认证的 Kubernetes 容器编排服务。图 15: Software Factory 展示了同时使用云 DevSecOps 服务和自维的安全工具的软件工厂。

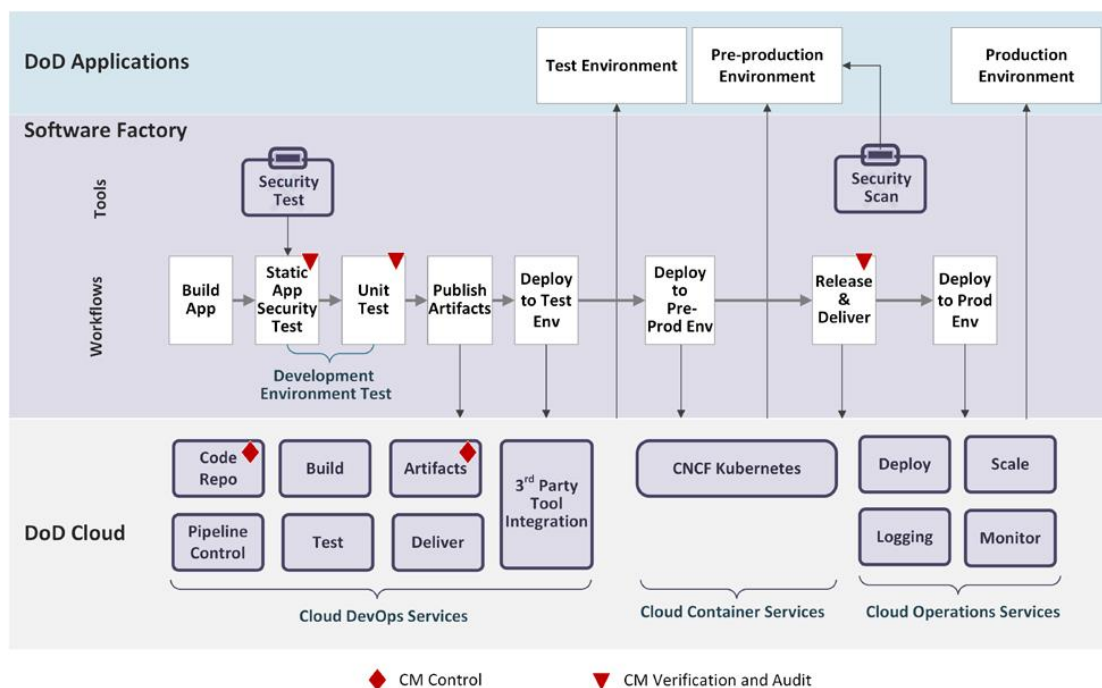


图 15 软件工厂使用云 DevSecOps 服务

6.3 Serverless Support 无服务器支持

So-called “serverless computing” is becoming more popular in the DoD, and it is being used extensively in industry. This is a kind of Platform as a Service (PaaS) that is sometimes called Function as a Service (FaaS). Despite the “serverless” moniker, a FaaS still needs servers, but developers don’t have to worry about the servers, but rather how to deploy the code to them, how to set up autoscaling, and other deployment tasks. This frees the developers to focus on the code.

所谓的“无服务器计算”在 DoD 变得越来越流行，并且在行业中得到了广泛应用。这是一种平台即服务（PaaS），有时也称为功能即服务（FaaS）。尽管有“无服务器”的绰号，但 FaaS 仍然需要服务器，但是开发人员不必担心服务器，而只需担心如何向其部署代码，如何设置自动扩展以及其他部署任务。这使开发人员可以将精力集中在代码上。

Figure 16 illustrates that a FaaS can reduce development complexity and increase efficiency over an Infrastructure as a Service (IaaS), a Containers as a Service (CaaS), or some Platform as a Service (PaaS) offerings. Although a Software as a Service (SaaS) offering is even more efficient, and good to use when it meets requirements, a SaaS is typically focused on only a few capabilities, and cannot provide the flexibility the DoD needs to develop custom applications. A good FaaS, on the other hand, does provide that flexibility, although some applications will need the even greater flexibility of an IaaS.

图 16 展示了 FaaS 降低开发复杂性和提效方面优于基础架构即服务（IaaS），容器即服务（CaaS）或某些平台即服务（PaaS）。尽管软件即服务（SaaS）产品更加高效，并且在满足要求时易于使用，但 SaaS 通常只专注于少数功能，且不能提供 DoD 开发定制应用程序所需的灵活性。另一方面，好的 FaaS 确实提供了这种灵活性，尽管某些应用程序需要 IaaS 层面提供更高的灵活性。



图 16 Operational Efficiency 运营效率

The FaaS concept has made its way into the Kubernetes environment. The basic concept is to hand the FaaS some code, then the FaaS will build an image from the code and start it running on Kubernetes.

FaaS 概念已经进入了 Kubernetes 环境。基本概念是提交一些代码给 FaaS，然后 FaaS 将从这些代码构建一个镜像，并在 Kubernetes 上启动运行。

The FaaS must have these features:

FaaS 必须有以下功能：

- 1) **Build** – builds containers from source code
 - a) Uses container images as the deployment unit
 - b) Given source code, build the code and create a container to house it
- 1) 构建 – 从源代码生成容器
 - a) 使用容器镜像作为部署单元
 - b) 给定源代码，构建代码并创建一个容器来容纳它
- 2) **Serving** – runs the containers created by Build and automatically scales them up or down as necessary
 - a) Uses CNCF Kubernetes as the underlying container orchestration layer
 - b) Auto-scale up (given that the code is written to allow this)
 - c) Auto-scale down all the way to zero
 - d) Gradual rollouts of new versions
 - e) Network routing within the cluster, and ingress connections into the cluster
- 2) 服务 – 运行由Build创建的容器，并根据需要自动按比例放大或缩小
 - a) 使用CNCF Kubernetes作为底层容器编排层
 - b) 自动放大（假设编写了代码以允许这样做）
 - c) 自动缩小到零
 - d) 逐步推出新版本
 - e) 群集内的网络路由，以及到群集的入口连接
- 3) **Eventing** – allows functions/applications to publish and subscribe to event streams, to enable loosely-coupled, event-driven systems

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

- a) Universal subscription, delivery, and management of events
 - b) Bind events to functions or containers
 - c) Trigger functions when called via and Hypertext Transfer Protocol (HTTP) requests
 - d) Automatically scale from a few events per day to live streams
- One popular open source product that implements FaaS for Kubernetes is Knative. Another open source product is Kubeless. The DevSecOps Software Factories must offer Knative support. They may also support Kubeless or another FaaS for Kubernetes.
- 3) 事件处理 - 允许函数/应用程序发布和订阅事件流，以启用松耦合的事件驱动系统
- a) 通用的事件订阅，交付和管理
 - b) 将事件绑定到函数或容器
 - c) 通过和超文本传输协议（HTTP）请求调用时的触发函数
 - d) 自动从每天的几个事件扩展到实时事件流

Knative是为Kubernetes实现FaaS的一种流行的开源产品。另一个开源产品是Kubeless。DevSecOps软件工厂必须提供本地支持。他们还可能为Kubernetes支持Kubeless或其他FaaS。

6.4 Application Security Operations 应用安全运行

This section focuses on the software application lifecycle in the production environment. Continuous Deployment, Continuous Operation, and Continuous Monitoring are keys to streamlined and secure operations.

本节重点介绍生产环境中的软件应用程序生命周期。持续部署、持续运行和持续监控是简化运行和安全运行的关键。

6.4.1 Continuous Deployment 持续部署

Continuous deployment is triggered by the successful delivery of released artifacts to the artifact repository and may be subject to control with human intervention according to the nature of the program application. The typical activities for continuous deployment include, but are not limited to, deploying a new software release to the production environment, applying necessary infrastructure and security configuration changes, running a smoke test to make sure essential functionality is working, and performing security scans. Each activity is completed by specific tools or configuration/orchestration scripts. The selection of tools and the configuration/orchestration system depends on the application and the production platform. For example, if the application is containerized and the production platform uses Kubernetes, the orchestration is done by Kubernetes. In this case, the configuration scripts would be Kubernetes Operators or Helm charts. On the other hand, if the application is VM based, the configuration/orchestration tool could be Chef, Puppet or Ansible.

持续部署是由已发布的工件成功交付到工件仓库触发的，并且可以根据程序应用程序的性质通过人工干预进行控制。持续部署的典型活动包括但不限于：将新的软件版本部署到生产环境、必要的基础架构和安全配置变更生效、运行冒烟测试以确保基本功能正常工作，以及执行安全扫描。每个活动都由特定的工具或配置/编排脚本完成，工具和配置/编排系统的选择取决于应用程序和生产平台。例如，如果应用程序是容器化的，而生产平台使用 Kubernetes，则编排由 Kubernetes 完成。在这种情况下，配置脚本将是 Kubernetes 操作符或 Helm 图表。另一方

面，如果应用程序是基于 VM 的，那么配置/编排工具可以是 Chef、Puppet 或 Ansible。

Continuous deployment interacts with other DevSecOps components, such as the artifact repository for retrieving new releases, the log storage and retrieval service for logging deployment events, and the issue tracking system for recording any deployment issues. The first-time deployment may involve heavy infrastructure provisioning, dependency system configuration (such as monitoring tools, logging tools, scanning tools, backup tools, etc.), and external system connectivity (such as DoD common security services, etc.).

持续部署与其他 DevSecOps 组件进行交互，例如用于检索新版本的工件仓库，用于记录部署事件的日志存储和检索服务以及用于记录所有部署问题的问题跟踪系统。首次部署可能涉及繁重的基础架构配置，依赖系统配置（例如监视工具，日志记录工具，扫描工具，备份工具等）以及外部系统连接性（例如 DoD 通用安全服务等）。

6.4.2 Continuous Operation 持续运行

Continuous operation is an extension of continuous deployment. It is triggered by a successful deployment to the production environment, so that it operates the latest stable software release. The activities of continuous operation include, but are not limited to, system patching, compliance scanning, data backup, system recovery if failure happens, and resource optimization with load balancing and scaling. The selection of the tools that facilitate the activities are application and environment dependent. The resource optimization heavily depends on the underlying platform. A containerized application can rely on Kubernetes to automatically scale containers across cluster nodes. On the other hand, a VM-based application with no containers can rely on the underlying CSP's scaling service.

持续运行是持续部署的扩展。它是由生产环境的成功部署触发，因此可以运行最新的稳定软件版本。持续运行的活动包括但不限于：系统补丁，合规性扫描，数据备份，发生系统故障时进行恢复，以及负载平衡和弹性扩展的资源优化。改善这些活动的工具选择取决于应用程序和环境。资源优化在很大程度上取决于基础平台。容器化的应用程序可以依靠 Kubernetes 在集群节点之间自动扩展容器。另一方面，没有容器而基于 VM 的应用程序可以依赖底层云服务提供商的扩容服务。

Continuous operation interacts with the logging system, issue tracking system, and the underlying infrastructure platform.

持续运行与日志记录系统，问题跟踪系统以及基础架构平台交互。

6.4.3 Continuous Monitoring 持续监控

Continuous monitoring is an extension to continuous operation. It continuously inventories all system components, monitors the performance and security of all components, and logs application and system events. Figure 20 in Section 6.4.4 illustrates the components for monitoring a containerized application deployed on Kubernetes. Figure 17 illustrates a simplified sample process of monitoring, logging, and log analysis and alerting, which also applies to deployments to non-containerized environments.

持续监控是对持续运营的扩展。它持续盘点所有系统组件，监视所有组件的性能和安全

性，并记录应用程序和系统事件。6.4.4 节中的图 20 说明了用于监视 Kubernetes 上部署的容器化应用程序的组件。图 17 展示了监视，日志记录，日志分析和警报的简化示例流程，该流程也适用于非容器化环境的部署。

The process starts with application logging, compute resource monitoring, storage monitoring, network monitoring, security monitoring, and data monitoring at the Kubernetes pod level in the case of containerized deployment or individual subsystem level in the case of VM deployment. Each application will need to determine how it is divided into subsystems, the number of subsystems, and the specific monitoring mechanisms within the subsystems. The security tools within each subsystem (e.g., the Sidecar Container Security Stack) will aggregate and forward the event logs gathered from monitoring to a locally centralized aggregated logs database on the mission program platform. This should be automated within the Kubernetes cluster. The aggregated logs will be further forwarded to the Logs/Telemetry Analysis in the DoD Common Security Services after passing the program application configured log filter. The program's local log analysis capability will analyze the aggregated logs and generate incident alerts and reports. Incidents will be forwarded to the mission program incident management system to facilitate change request generation for incident resolution. The mission program incident management should alert or notify the responsible personnel about the incidents. The change request may be created to address the incident. These actions make the DevSecOps pipeline a full closed loop from secure operations back to planning.

该流程在容器化部署情况下，从 Kubernetes pod 级别进行应用程序日志记录，计算资源监控，存储监控，网络监控，安全监控和数据监视，而在 VM 部署的情况下则从单个子系统级别开始。每个应用程序都需要确定如何将其划分为子系统，子系统的数量以及子系统内的特定监控机制。每个子系统中的安全工具（例如，边车容器安全堆栈）将汇总从监控收集的事件日志并将其转发到任务程序平台上的本地集中的聚合日志数据库。这应该在 Kubernetes 集群中实现自动化。聚合的日志通流程序应用程序配置的日志过滤器后，进一步转发到 DoD 通用安全服务中的日志/遥测分析。该程序的本地日志分析功能将分析聚合日志并生成事件警报和报告。事件将被转发到任务程序事件管理系统，以促进解决事件的变更请求生成。任务程序事件管理应发送事件警告或通知责任人，以便于创建变更请求来解决事件。这些活动使 DevSecOps 流水线形成从安全运营到计划的完整闭环。

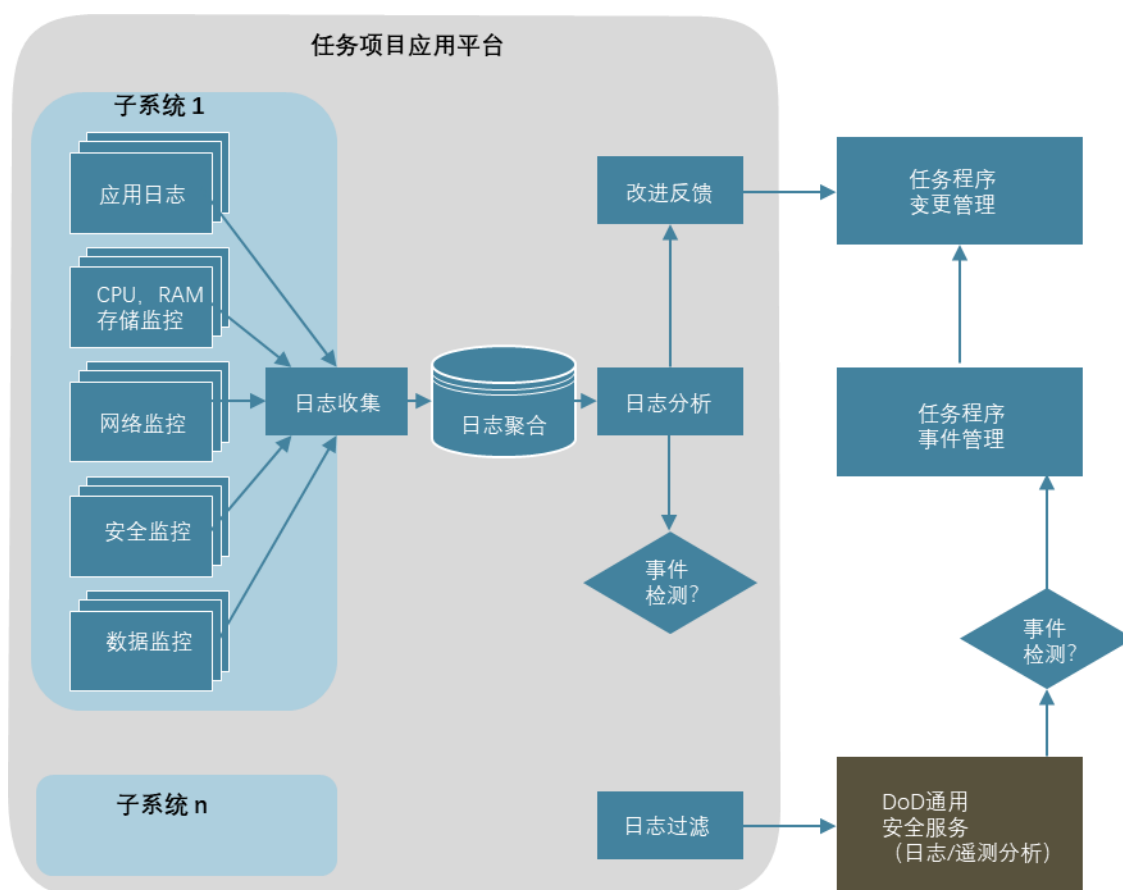


图 17 日志记录与分析流程

6.4.4 Sidecar Container Security Stack 边车容器安全堆栈

A new service that is enabled by DevSecOps and the container-based Kubernetes runtime environment is the Sidecar Container Security Stack (SCSS). This security stack enables: correlated and centralized logs, container security, east/west traffic management, a zero-trust model, a whitelist, Role-Based Access Control (RBAC), continuous monitoring, signature-based continuous scanning using Common Vulnerabilities and Exposures (CVEs), runtime behavior analysis, and container policy enforcement.

边车容器安全堆栈（SCSS）是 DevSecOps 和基于容器的 Kubernetes 运行时环境启用的一项新服务。该安全堆栈可实现：关联和集中的日志，容器安全，东西向流量管理，零信任模型，白名单，基于角色的访问控制（RBAC），连续监视，使用基于 CVE（常见漏洞与泄露）签名的连续扫描，运行时行为分析和容器策略执行。

One advantage of using the SCSS is that Kubernetes can inject the sidecar automatically, without the team having to do anything, once it's configured.

The sidecar pattern is depicted in Figure 18. A container group or pod is a set of containers that are deployed together. A sidecar is a container running inside a pod alongside an application container. If there is one application container and one sidecar container in the container group, then we have the sidecar pattern as depicted in Figure 18.

使用 SCSS 的优势之一是，一旦配置好 Kubernetes，便可以自动连接边车，而团队则无需执

行任何操作。

边车模式如图 18 所示。容器组或 pod 是一组部署在一起的容器。边车是在 pod 中运行的容器，与应用程序容器并排。如果容器组中有一个应用程序容器和一个边车容器，则我们具有如图 18 所示的边车模式。

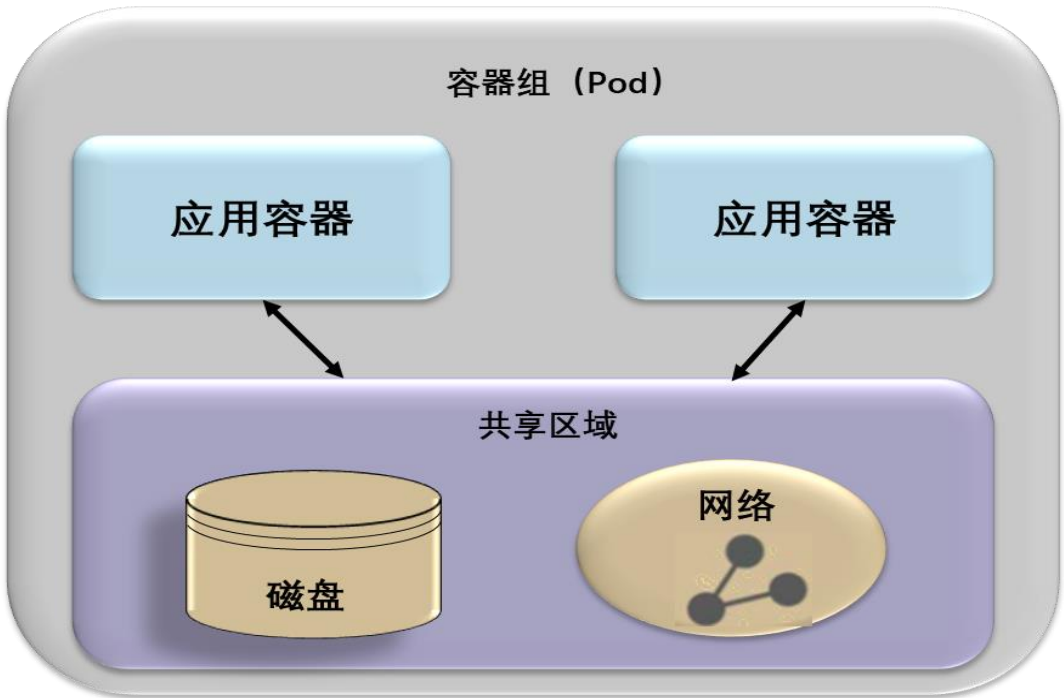


图 18 边车模式

The sidecar can share state with the application container. In particular, the two containers can share disk and network resources while their running components are isolated from one another.

边车可以与应用程序容器共享状态。特别是，两个容器可以共享磁盘和网络资源，而它们的运行组件彼此隔离。

A sidecar is a general container pattern. The Sidecar Container Security Stack has a sidecar container that contains a security stack, along with some supporting services that run in the hosting environment, such as a logging service. The security stack in the security sidecar container will include:

边车是一种通用的容器模式。边车容器安全性堆栈具有一个边车容器，其中包含一个安全性堆栈以及在宿主环境中运行的一些支持服务，例如日志记录服务。安全边车容器中的安全堆栈将包括：

1. A logging agent to push logs to a platform centralized logging service.
2. Container policy enforcement. This includes ensuring container hardening from DCAR containers are preserved and complies with the NIST 800-190 requirements [12].
3. Runtime Defense, this can perform both signature-based and behavior-based detection. This can also be used to send notifications when there is anomalous behavior.
4. Vulnerability Management

5. A service mesh proxy to connect to the service mesh
6. Zero Trust down to the container level. Zero trust requires strict controls, never trust anything by default and always verify. Key aspects of zero trust at the container level include mutual Transport Layer Security authentication (mTLS), an encrypted communication tunnel between containers, strong identities per Pod using certificates, and whitelisting rather than blacklisting.

1. 日志记录代理，用于将日志推送到平台集中式日志记录服务；
2. 容器策略执行，包括确保 DCAR 容器加固的容器已预留备用，并符合 NIST 800-190 的要求[12]；
3. 运行时防御，可以执行基于签名的检测和基于行为的检测，用于出现异常行为时发送通知；
4. 漏洞管理；
5. 连接到服务网格的服务网格代理；
6. 零信任下沉至容器级。零信任需要严格的控制，默认情况下从不信任任何内容并始终进行验证。容器层面零信任的关键方面包括双向传输层安全协议（mTLS），一种容器之间的加密通信隧道；使用证书为每个 pod 提供强健的身份，以及使用白名单而非黑名单。

In addition to the components in the sidecar, there are a few services that support the security sidecar. These include:

另外，边车中的组件有些服务支持安全边车。包括：

1. Program-specific Log Storage and Retrieval Service
 2. Service Mesh
 3. Program-specific artifact repository
 4. Runtime Behavior Analysis Artificial Intelligence (AI) service
 5. DCAR for the hardened containers
 6. Common Vulnerabilities and Exposures (CVE)Service / host-based security to provide CVEs for the security sidecar container
1. 项目专用的日志存储和检索服务；
 2. 服务网格；
 3. 项目专用的工件仓库；
 4. 运行时人工智能（AI）行为分析服务；
 5. 存储已加固容器的 DCAR；
 6. CVE（常见漏洞和披露）服务/主机安全为边车安全容器提供 CVE。

The interaction of these services with the sidecar components is depicted in Figure 19. The arrows show the direction of the data flow. The items in purple are services provided by the DoD, while blue indicates components that are provided by the DoD but instantiated and operated by the program.

这些服务与边车组件的交互如图 19 所示。箭头表示数据流的方向。紫色项是由 DoD 提供的服务，而蓝色项则表示由 DoD 提供但由程序实例化和操作的组件。

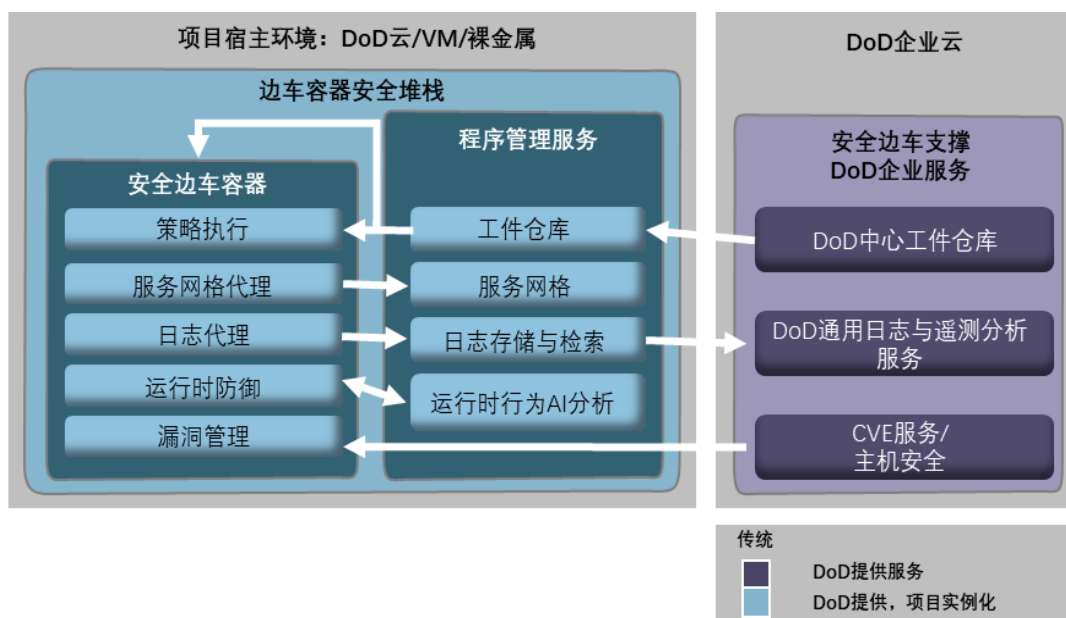


图 19 边车组件

表 23 Sidcar Container Security Stack Components 边车容器安全堆栈组件

Tool 工具	Features 特性	Benefit 收益	Baseline 基线
Logging agent 日志代理	Send logs to a logging service 发送日志到日志服务	Standardize log collection to a central location. This can also be used to send notifications when there is anomalous behavior. 标准化收集到的日志到集中位置。也可以用于出现异常行为时发送通知。	MVP
Logging Storage and Retrieval Service 日志存储和检索服务	Stores logs and allows searching logs 存储日志并允许搜索日志	Place to store logs 提供存储日志的位置	MVP
Log visualization and analysis 日志可视化与分析	Ability to visualize log data in various ways and perform basic log analysis. 能够用各种方法实现日志数据可视化和执行基本日志分析。	Helps to find anomalous patterns 帮助发现异常模式	Objective
Container policy enforcement 容器策略执行	Support for Security Content Automation Protocol (SCAP) and container configuration policies. These policies can be defined as needed. 支持 SCAP (安全内容自动化协议) 和容器配置策略, 这些策略可以按需自定义。	Automated policy enforcement 自动策略执行	MVP
Runtime Defense 运行时防御	Creates runtime behavior models, including whitelist and least privilege 创建运行时行为模型, 包括白名单和最小权限	Dynamic, adaptive cybersecurity 动态, 自适应网络安全	MVP
Service Mesh proxy 服务网格代理	Ties to the Service Mesh. Used with a microservices architecture. 绑定到服务网格。用于微服务架构	Enables use of the service mesh. 启用服务网格	Objective
Service Mesh 服务网格	Used for a microservices architecture 用于微服务架构	Better microservice management 更好的微服务管理	Objective
Vulnerability Management 漏洞管理	Provides vulnerability management 提供漏洞管理	Makes sure everything is properly patched to avoid known vulnerabilities 确保补丁正常更新以避免已知漏洞	MVP

CVE Service / Host Based Security CVE 服务/主机安全	Provides CVEs. Used by the vulnerability management agent in the security sidecar container. 提供 CVE. 用于安全边车容器漏洞管理。	Makes sure the system is aware of known vulnerabilities in components. 确保关注系统组件的已知漏洞	MVP
Artifact Repository 工件仓库	Storage and retrieval for artifacts such as containers. 存储和检索工件，例如容器。	One location to obtain hardened artifacts such as containers 一个获取加固工件的位置，例如容器	MVP
Zero Trust model down to the container level 零信任模型下沉至容器层	Provides strong identities per Pod with certificates, mTLS tunneling and whitelisting of East-West traffic down to the Pod level. 使用证书为每个 Pod 提供更强健的身份，mTLS 隧道和 Pod 级别的东西向流量白名单。	Reduces attack surface and improves baked-in security 减少并改善攻击面 嵌入式安全	MVP

Figure 20 depicts another view of the sidecar, along with some of the other DevSecOps components. Again, the arrows show the direction of the data flow, and not all interactions between the depicted components are indicated. The program dashboard displays information about the application. The dashboard is built partly using visualizations from the log visualization service. The items in purple and blue are either services or components that are provide by the DoD. But those in blue will be stood up by the program. For example, the Service Mesh is provided as a hardened container. Similarly, the Sidecar Container Security Stack is provided as a hardened container that the program installs in each pod (container group); this is injected by Kubernetes automatically without application developer involvement.

图 20 描绘了边车的另一视图，以及其他一些 DevSecOps 组件。同样，箭头表示数据流的方向，但并未表示出图中组件之间的所有交互。程序仪表板显示有关应用程序的信息，仪表板部分是使用日志可视化服务的可视化功能构建的。紫色和蓝色项是 DoD 提供的服务或组件，但蓝色项则由项目负责支持，例如，服务网格是作为已加固容器来提供的。同样，边车容器安全堆栈作为项目安装在每个 Pod（容器组）中的加固容器被提供，这由 Kubernetes 自动注入，无需应用开发人员参与。

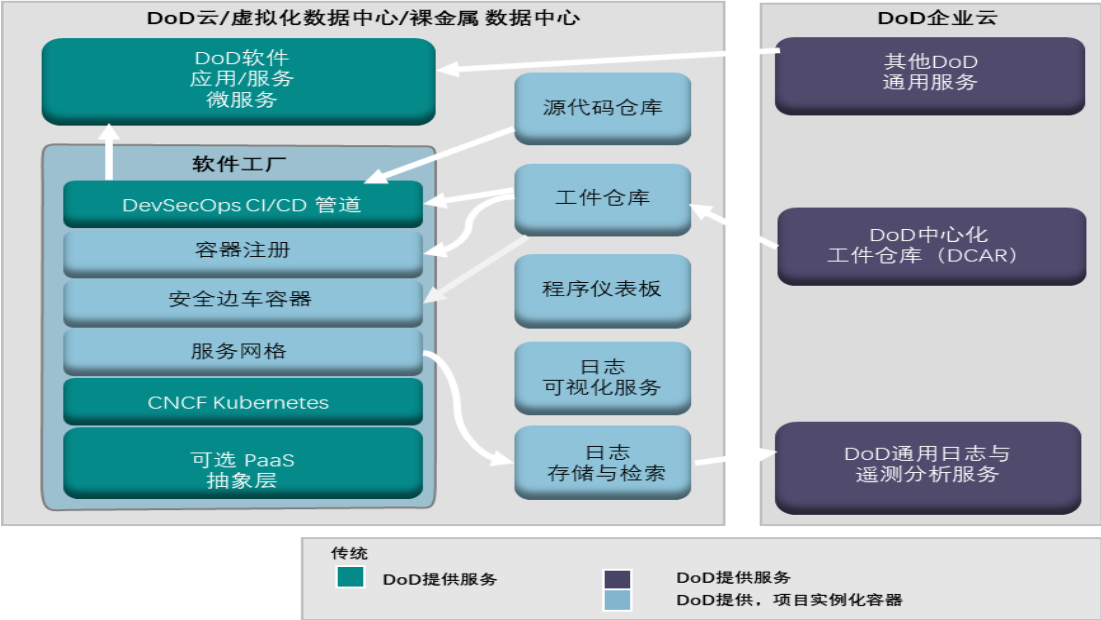


图 20 边车容器安全堆栈交互

7 Conclusion

We have introduced key DevSecOps concepts, described the DevSecOps Ecosystem, including the Software Factory and the Sidecar Container Security Stack, and indicated how the ecosystem should be set up and used. More detail on the components described here, such as DCAR, can be found in other DoD CIO documents.

我们介绍了 DevSecOps 的几个关键概念，描述了 DevSecOps 生态系统，包括软件工厂和边车容器安全堆栈，并指出了如何建立和使用它们。更多相关组件描述的详细信息可以在其他 DoD CIO 文档中找到（例如 DCAR）

Moving to DevSecOps improves agility and speeds new capabilities into the field. But it also requires new policies, processes and culture change. More information on DevSecOps culture, metrics, and the maturity model can be found in the DoD DevSecOps Playbook [7].

转向 DevSecOps 可以提高敏捷性，并加快将新能力投入战场的速度。但这也需要新的策略、流程和文化变革。有关 DevSecOps 文化，衡量指标和成熟度模型的更多信息，请参见 DoD DevSecOps Playbook [7]。

Appendix A Acronym Table 字母缩写表

Acronym 首字母缩写	Definition 定义	中文翻译
A&A	Assessment and Authorization	评估与授权
A&S	Acquisition and Sustainment	采购与维护
AI	Artificial Intelligence	人工智能
AO	Authorizing Official	授权官员
API	Application Programming Interface	应用程序接口
AQ	Acquisition	获取
ASW	Attack Sensing and Warning	攻击感知和警告
ATC	Approval to Connect (ATC)	批准连接 (ATC)
ATO	Authority to Operate (ATO)	操作权限 (ATO)
AVM	Assurance Vulnerability Management	保证漏洞管理
BOM	Bill of Materials	材料清单
CaaS	Containers as a Service	容器服务
CCB	Change Control Board	变更控制管理
CD	Continuous Delivery	持续交付
CI	Continuous Integration	持续集成
CIO	Chief Information Officer	首席信息官
CM	Configuration Management	配置管理
CMDB	Configuration Management Data Base	配置管理数据库
CNCF	Cloud Native Computing Foundation	云原生计算基金会
CNSS	Committee on National Security Systems	国家安全系统委员会
CNSSI	Committee on National Security Systems Instruction	国家安全系统指导委员会
COTS	Commercial Off The Shelf	商用现成品或技术或商用货架产品
CPCON	Cyber Protection Condition	网络保护条件
CPU	Central Processing Unit	中央处理单元
CSP	Cloud Service Provider	云服务提供商
CSSP	Cybersecurity Service Provider	网络安全服务提供商
CVE	Common Vulnerabilities and Exposures	通用漏洞和披露
DAST	Dynamic Application Security Test	动态应用程序安全性测试
DCAR	DoD Centralized Artifact Repository	国防部 集中工件仓库
DCCSCR	DoD Centralized Container Source Code Repository	国防部 集中式容器源代码存储库
DCIO	Deputy Chief Information Officer	副首席信息官
DBaaS	Database as a Service	数据库即服务
DDOS	Distributed Denial of Service	分布式拒绝服务
DevSecOps	Development, Security, and Operations	开发, 安全和运营
DISA	Defense Information Systems Agency	国防信息系统局
DNS	Domain Name Service	域名服务
DoD	Department of Defense	国防部
DoDI	DoD Instruction	国防部指令
DTRA	Defense Threat Reduction Agency	国防威胁降低局
EO	Executive Order	总统行政令
FaaS	Function as a Service	功能即服务
FMA	Forensic Media Analysis	失效媒体分析
GB	Gigabyte	千兆字节
GEP	Global Enterprise Partners	全球企业合作伙伴
GOTS	Government Off The Shelf	政府现货供应
HTTP	Hypertext Transfer Protocol	超文本传输协议
IA	Information Assurance	信息保障
IaaS	Infrastructure as a Service	基础架构即服务
IaC	Infrastructure as Code	基础架构即代码
IAST	Interactive Application Security Test	交互式应用程序安全性测试
ICAM	Identity, Credential, and Access Management	身份, 凭证和访问管理
IDE	Integrated Development Environment	集成开发环境
IDS	Intrusion Detection System	入侵侦测系统
IE	Information Enterprise	信息企业
IHR	Incident Handling Response	事件处理响应
INFOCON	Information Operations Condition	信息运行条件
IO	Input/Output	输入/输出
IPS	Intrusion Prevention System	入侵防御系统
IR	Incident Reporting	事故报告
ISCM	Information Security Continuous Monitoring	信息安全持续监控
IT	Information Technology	信息技术

主编: 唐龙 译者: 唐龙 (1-2 章, 4.1-4.2)、张晨晖 (前言, 附录, 全文校对)、周景川 (4.3-7 章)、赵庆安 (3 章) 排名不分先后

JVM	Java Virtual Machine	Java 虚拟机
KPI	Key Performance Indicator	关键绩效指标
MB	Megabyte	兆字节
MilDep	Military Department	军事部
MNP	Malware Notification Protection	恶意软件通知保护
mTLS	mutual Transport Layer Security autiientication	相互传输层安全认证
MVP	Minimum Viable Product	最低可行产品
NGG	Next Generation Governance	下一代管治
NIST	National Institute of Standards and Technology	国立标准技术研究所
NoSQL	Non SQL	非关系 SQL
NSM	Network Security Monitoring	网络安全监控
OCI	Open Container Initiative	开放容器倡议
OS	Operating System	操作系统
PA	Provisional Authorization	临时授权
PaaS	Platform as a Service	平台即服务
PEO	Program executive Officer	计划执行官
POA&M	Plan of Action and Milestones	行动计划和里程碑
QA	Quality Assurance	质量保证
RBAC	Role-Based Access Control	基于角色的访问控制
RMF	Risk Management Framework	风险管理框架
ROI	Return on Investment	投资回报
SaaS	Software as a Service	软件作为服务
SaC	Security as Code	安全即代码
SAF	Secretary of the Air Force	空军部长
SAST	Static Application Security Test	静态应用程序安全性测试
SCAP	Security Content Automation Protocol	安全内容自动化协议
SCCA	Secure Cloud Computing Architecture	安全云计算架构
scss	Sidecar Container Security Stack	Sidecar 集装箱安全栈
SLA	Service Level Agreement	服务水平协议
SRG	Security Requirements Guide	安全要求指南
SQL	Structured Query Language	结构化查询语言
SSH	Secure Shell	安全壳
STIG	Security Technical Implementation Guide	安全技术实施指南
TRB	Technical Review Board	技术审查委员会
VM	Virtual Machine	虚拟机

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Appendix B Glossary of Key Terms 附录 B：关键术语词汇表

Following are the key terms used in describing the reference design in this document.

下述是被用于在本文中描述参考设计的关键术语

Term (术语)	Definition (定义)
Artifact 工件 Software Artifact 软件工件	<p>An artifact is a consumable piece of software produced during the software development process. Except for interpreted languages, the artifact is or contains compiled software. Important examples of artifacts include container images, virtual machine images, binary executables, jar files, test scripts, test results, security scan results, configuration scripts, Infrastructure as a Code, documentation, etc. Artifacts are usually accompanied by metadata, such as an id, version, name, license, dependencies, build date and time, etc. Note that items such as source code, test scripts, configuration scripts, build scripts, and Infrastructure as Code are checked into the source code repository, not the artifact repository, and are not considered artifacts.</p> <p>工件是在软件开发流程中生产的可消费的软件组件。除解释语言外，工件是以及或者包含了被编译的软件。工件的重要示例包括容器镜像，虚拟机镜像，二进制可执行文件，jar 文件，测试脚本，测试结果，安全扫描结果，配置脚本，“基础架构即代码”，文档等。工件通常伴随有元数据，例如 id，版本，名称，许可证，依赖，构建日期和时间等。需要注意的是，诸如源代码，测试脚本，配置脚本，构建脚本和“基础架构即代码”之类的项目已签入源代码仓库，而不在工件仓库中，不被视为工件。</p>
Artifact Repository 工件仓库	<p>An artifact repository is a system for storage, retrieval, and management of artifacts and their associated metadata.</p> <p>Note that programs may have separate artifact repositories to store local artifacts and released artifacts. It is also possible to have a single artifact repository and use tags to distinguish the content types.</p> <p>工件仓库是一个用于存储，检索和管理工件及其关联的元数据的系统。</p> <p>请注意，程序可能使用分离的工件仓库，以存储本地工件和已发布的工件。也可能使用一个工件仓库，并使用标签来区分内容类型。</p>
Bare Metal 裸金属 Bare Metal Server 裸金属服务器	<p>A bare metal or bare metal server refers to a traditional physical computer server that is dedicated to a single tenant and which does not run a hypervisor. This term is used to distinguish physical compute resources from modern forms of virtualization and cloud hosting.</p> <p>裸金属或裸金属服务器是指传统的物理服务器专用于单个租户且不运行 hypervisor。该术语用于将物理计算资源与虚拟化和云托管的现代形式区分开。</p>
Binary 二进制 Binary File 二进制文件	<p>Binary refers to a data file or computer executable file that is stored in binary format (as opposed to text), which is computer-readable, but not human-readable. Examples include images, audio/video files, exe files, and jar/war/ear files.</p> <p>二进制是指数据文件或计算机可执行文件以二进制格式（相对于文本）存储，该格式是计算机可读的，但不是人类可读的。示例包括图像，音频/视频文件，exe 文件和 jar/war/ear 文件。</p>
Build 构建 Software Build 软件构建	<p>The process of creating a set of executable code that is produced by compiling source code and linking binary code.</p> <p>创建一组通过编译源代码和链接二进制代码产生的可执行代码的流程。</p>
Build tools 构建工具 Software Build Tools 软件构建工具	<p>Used to retrieve software source code, build software, and generate artifacts.</p> <p>用于检索软件源代码，构建软件和生成构件。</p>
CI/CD Orchestrator CI/CD 编排器	<p>CI/CD orchestrator is a tool that enables fully or semi-automated short duration software development cycles through integration of build, test, secure, store artifacts tools.</p> <p>CI/CD orchestrator is the central automation engine of the CI/CD pipeline</p> <p>CI/CD 编排器是一种可通过集成构建，测试，安全，存储工件工具来实现完全或半自动化的短期软件开发周期的工具。</p>

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	<p>CI/CD 编排器是 CI/CD 流水线的中央自动化引擎。</p>
<p>CI/CD Pipeline CI/CD 流水线</p>	<p>CI/CD pipeline is the set of tools and the associated process workflows to achieve continuous integration and continuous delivery with build, test, security, and release delivery activities, which are steered by a CI/CD orchestrator and automated as much as practice allows.</p> <p>CI/CD 流水线是用于通过构建，测试，安全性和发布交付活动实现持续集成和连续交付的一组工具和相关的流程工作流，这些活动由 CI/CD 编排器操纵并在实践允许的范围内尽可能地自动化。</p>
<p>CI/CD Pipeline Instance CI/CD 流水线实例</p>	<p>CI/CD pipeline instance is a single process workflow and the tools to execute the workflow for a specific software language and application type for a project. As much of the pipeline process is automated as is practicable.</p> <p>CI/CD 流水线实例是一个单流程工作流，以及为项目的特定软件语言和应用程序类型执行工作流的工具。大多数的流水线流程都是自动化的。</p>
<p>Cloud Native Computing Foundation (CNCF) 云原生计算基金会</p>	<p>UCNCF is an open source software foundation dedicated to making cloud native computing universal and sustainable. Cloud native computing uses an open source software stack to deploy applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization. Cloud native technologies enable software developers to build great products faster. - https://www.cncf.io/</p> <p>UCNCF 是一个开源软件基金会，致力于使云原生计算具备通用性和可持续性。云原生计算使用开源软件堆栈将应用程序部署为微服务，将每个部分打包到其自己的容器中，并动态编排这些容器以优化资源利用率。云原生技术使软件开发人员能够更快地构建出色的产品。 - https://www.cncf.io/</p>
<p>CNCF-certified Kubernetes CNCF 认证的 Kubernetes</p>	<p>CNCF has created a Certified Kubernetes Conformance Program. Software conformance ensures that every vendor's version of Kubernetes supports the required APIs. Conformance guarantees interoperability between Kubernetes from different vendors. Most of the worlds leading vendors and cloud computing providers have CNCF certified Kubernetes offerings.</p> <p>CNCF 已创建了 Kubernetes 认证合格计划。软件一致性确保每个供应商的 Kubernetes 版本都支持所需的 API。一致性保证了来自不同供应商的 Kubernetes 之间的互操作性。世界上大多数领先的供应商和云计算提供商都拥有 CNCF 认证的 Kubernetes 产品。</p>
<p>Code 代码</p>	<p>Software instructions for a computer, written in a programming language. These instructions may be in the form of either human- readable source code, or machine code, which is source code that has been compiled into machine executable instructions.</p> <p>用编程语言编写的计算机软件说明。这些指令可以是人类可读源代码或机器代码的形式，机器代码是已经被编译成机器可执行指令的源代码。</p>
<p>Configuration Management 配置管理</p>	<p>Capability to establish and maintain a specific configuration within operating systems and applications.</p> <p>在操作系统和应用程序中建立和维护特定配置的能力。</p>
<p>Container 容器</p>	<p>A standard unit of software that packages up code and all its dependencies, down to, but not including the OS. It is a lightweight, standalone, executable package of software that includes everything needed to run an application except the OS: code, runtime, system tools, system libraries and settings.</p> <p>打包代码及其所有依存关系的标准软件单元，包括但不限于 OS。它是一个轻量的，独立的，可执行的软件软件包，其中包括运行应用程序所需的一切（操作系统除外）：代码，运行时，系统工具，系统库和设置。</p>
<p>Continuous Build 持续构建</p>	<p>Continuous build is an automated process to compile and build software source code into artifacts. The common activities in the continuous build process include compiling code, running static code analysis such as code style checking, binary linking (in the case of languages such as C++), and executing unit tests. The outputs from continuous build process are build results, build reports (e.g., the unit test report, and a static code analysis report), and artifacts stored into Artifact Repository. The trigger to this process could be a developer code commit or a code merge of a branch into the main trunk.</p> <p>持续构建是将软件源代码编译和构建为工件的自动化流程。连续构建流程中的常见活动包括编译代码，运行静态代码分析（如代码风格检查），二进制链接（对于 C 等语言）以及执行单元测试。连续构建流程的输出是构建结果，构建报告（例如，单元测试报告和静态代码分析报告）以及存储在 Artifact 存储库中的</p>

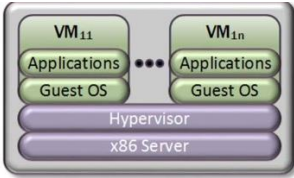
主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	工件。该流程的触发条件可能是开发人员代码提交或分支到主干的代码合并。
Continuous Delivery 持续交付	<p>Continuous delivery is an extension of continuous integration to ensure that a team can release the software changes to production quickly and in a sustainable way.</p> <p>The additional activities involved in continuous integration include release control gate validation and storing the artifacts in the artifact repository, which may be different than the build artifact repository.</p> <p>The trigger to these additional activities is successful integration, which means all automation tests and security scans have been passed.</p> <p>The human input from the manual test and security activities should be included in the release control gate.</p> <p>The outputs of continuous delivery are a release go/no-go decision and released artifacts, if the decision is to release.</p> <p>持续交付是持续集成的扩展，以确保团队能以可持续的方式将软件变更快速发布到生产环境中。</p> <p>持续集成中涉及的其他活动包括发布控制门验证以及将工件存储在工件仓库中，这可能与构建工件仓库不同。</p> <p>成功进行集成是这些活动的触发条件，这意味着所有自动化测试和安全扫描都已通过。</p> <p>人工测试和安全活动的人工输入应包括在发布控制门中。</p> <p>如果决定要发布，则连续交付的输出是发布通过/不通过决定和被发布的工件。</p>
Continuous Deployment 持续部署	<p>Continuous deployment is an extension of continuous delivery. It is triggered by a successful delivery of released artifacts to the artifact repository.</p> <p>The additional activities for continuous deployment include, but are not limited to, deploying a new release to the production environment, running a smoke test to make sure essential functionality is working, and a security scan.</p> <p>The output of continuous deployment includes the deployment status. In the case of a successful deployment, it also provides a new software release running in production. On the other hand, a failed deployment causes a rollback to the previous release.</p> <p>持续部署是持续交付的扩展。它由成功交付已发布的工件到工件仓库触发。</p> <p>持续部署的其他活动包括但不限于将新版本部署到生产环境，运行冒烟测试以确保基本功能正常运行以及安全扫描。</p> <p>持续部署的输出包括部署状态。在成功部署的情况下，它还提供了在生产环境中运行的新软件版本。另一方面，部署失败会导致回滚到以前的版本。</p>
Continuous Integration 持续集成	<p>Continuous integration goes one step further than continuous build. It extends continuous build with more automated tests and security scans. Any test or security activities that require human intervention can be managed by separate process flows.</p> <p>The automated tests include, but are not limited to, integration tests, a system test, and regression tests. The security scans include, but are not limited to, dynamic code analysis, test coverage, dependency/BOM checking, and compliance checking.</p> <p>The outputs from continuous integration include the continuous build outputs, plus automation test results and security scan results.</p> <p>The trigger to the automated tests and security scan is a successful build.</p> <p>持续集成比持续构建更进了一步，它通过更多的自动化测试和安全扫描对持续构建进行扩展。任何需要人工干预的测试或安全活动都可以通过单独的流程进行管理。</p> <p>自动化测试包括但不限于集成测试，系统测试和回归测试。安全扫描包括但不限于动态代码分析、测试覆盖率、依赖项/ BOM 检查和合规性检查。</p> <p>持续集成的输出包括持续构建的输出、以及自动化测试结果和安全扫描结果。</p> <p>自动化测试和安全扫描的触发条件是一次成功的构建。</p>
Continuous monitoring 持续监控	<p>Continuous monitoring is an extension to continuous operation. It continuously monitors and inventories all system components, monitors the performance and security of all the components, and audits & logs the system events.</p> <p>持续监控是对持续运营的扩展。它持续监视和盘点所有系统组件，监视所有组件的性能和安全性，以及审</p>

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	核和记录系统事件。
Continuous Operation 持续运营	<p>Continuous operation is an extension to continuous deployment. It is triggered by a successful deployment. The production environment operates continuously with the latest stable software release.</p> <p>The activities of continuous operation include, but are not limited to: system patching, compliance scanning, data backup, and resource optimization with load balancing and scaling (both horizontal and vertical).</p> <p>持续运营是对持续部署的扩展。它由一次成功的部署触发。生产环境通过使用最新的、稳定的软件版本持续运行。</p> <p>持续运营的活动包括但不限于：系统修补，法规遵从性扫描，数据备份以及具有负载均衡和扩展（水平和垂直）的资源优化。</p>
Cybersecurity, Software Cybersecurity 网络安全，软件安全	<p>The preventative methods used to protect software from threats, weaknesses and vulnerabilities.</p> <p>用于保护软件免受威胁、弱点和漏洞影响的预防方法。</p>
DoD Centralized Artifact Repository (DCAR) 国防部集中化工件仓库	<p>Holds the hardened container images of DevSecOps components that DoD mission software teams can utilize to instantiate their own DevSecOps pipeline. It also holds the hardened containers for base operating systems, web servers, application servers, databases, API gateways, message busses for use by DoD mission software teams as a mission system deployment baseline. These hardened containers, along with security accreditation reciprocity, greatly simplifies and speeds the process of obtaining an Approval to Connect (ATC) or Authority to Operate (ATO).</p> <p>用于保存国防部任务软件团队可用来实例化其自己的 DevSecOps 流水线的 DevSecOps 组件的加固容器镜像。它还包含用于基础操作系统，Web 服务器，应用程序服务器，数据库，API 网关，消息总线的加固容器，供 DoD 任务软件团队用作任务系统部署基准。这些经过加固的容器以及安全互信，大大简化并加快了获得连接许可（ATC）或运行授权（ATO）的流程。</p>
Delivery 交付	<p>The process by which a released software is placed into a artifact repository that operational environment can download.</p> <p>将发布的软件放置到操作环境可以下载工件仓库中的流程。</p>
Deployment 部署	<p>The process by which the released software is downloaded and deployed to the production environment.</p> <p>下载已发布软件并将其部署到生产环境的流程。</p>
DevSecOps	<p>DevSecOps is a software engineering culture and practice that aims at unifying software development (Dev), security (Sec) and operations (Ops). The main characteristic of DevSecOps is to automate, monitor, and apply security at all phases of software development: plan, develop, build, test, release, deliver, deploy, operate, and monitor.</p> <p>DevSecOps 是一种软件工厂文化和实践，旨在统一软件开发（Dev），安全性（Sec）和运营（Ops）。DevSecOps 的主要特征是在软件开发的各个阶段，包括计划，开发，构建，测试，发布，交付，部署，操作和监视，实现自动化、监视和应用安全性。</p>
DevSecOps Ecosystem DevSecOps 生态系统	<p>A collection of tools and process workflows created and executed on the tools to support all the activities throughout the full DevSecOps lifecycle.</p> <p>The process workflows may be fully automated, semi-automated, or manual.</p> <p>一系列工具、以及在工具上创建和执行的工作流程的集合，以支持整个 DevSecOps 整个生命周期中的所有活动。</p> <p>流程工作流可以是全自动，半自动化或手动的。</p>
DevSecOps Pipeline DevSecOps 流水线	<p>DevSecOps pipeline is a collection of DevSecOps tools, upon which the DevSecOps process workflows can be created and executed.</p> <p>DevSecOps 流水线是 DevSecOps 工具的集合，可以在其上创建和执行 DevSecOps 工作流。</p>
DevSecOps phase DevSecOps 阶段	<p>The software development, security, and operation activities in the software lifecycle are divided into phases. Each phase completes a part of related activities using tools.</p> <p>软件生命周期中的软件开发，安全性和操作活动分为多个阶段。每个阶段都使用工具完成一部分相关活动。</p>

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Environment 环境	<p>Sets a runtime boundary for the software component to be deployed and executed. Typical environments include development, integration, test, pre-production, and production.</p> <p>为要部署和执行的软件组件设置运行时边界。典型的环境包括开发，集成，测试，预生产和生产。</p>
Factory, Software Factory 工厂，软件工厂	<p>A software assembly plant that contains multiple pipelines, which are equipped with a set of tools, process workflows, scripts, and environments, to produce a set of software deployable artifacts with minimal human intervention. It automates the activities in the develop, builds, test, release, and deliver phases. The software factory supports multi-tenancy.</p> <p>一个软件组装厂，包含多个流水线，这些流水线配备了一组工具，流程工作流，脚本和环境，以最少的人工干预即可生成一组可软件化部署的工件。它使开发、构建、测试，发布和交付阶段中的活动自动化。该软件工厂支持多租户。</p>
Software Factory Artifact Repository 软件工厂工件仓库	<p>Stores artifacts pulled from DCAR as well as locally developed artifacts to be used in DevSecOps processes. The artifacts include, but are not limited to, VM images, container images, binary executables, archives, and documentation. It supports multi-tenancy.</p> <p>Note that program could have separate artifact repositories to store local artifacts and released artifacts. It is also possible to have a single artifact repository and use tags to distinguish the contents.</p> <p>存储从 DCAR 提取的工件以及将在 DevSecOps 流程中使用的本地开发的工件。工件包括但不限于 VM 镜像，容器镜像，二进制可执行文件，归档和文档。它支持多租户。</p> <p>请注意，程序可能具有独立的工件仓库，以存储本地工件和已发布的工件。也可以使用一个工件库并使用标签来区分内容。</p>
Hypervisor	<p>A hypervisor is a kind of low-level software that creates and runs virtual machines (VMs). Each VM has its own Operating System (OS). Several VMs can run on one physical machine.</p> <p>系统管理程序是一种创建和运行虚拟机（VM）的低级软件。每个 VM 都有自己的操作系统（OS）。一台物理机上可以运行多个 VM。</p>  <p>图 21 Hypervisor 和虚拟机</p>
Image Management, Software Image Management, Binary Image Management, Container Image Management, VM Image Management 镜像管理，软件镜像管理，二进制镜像管理，容器镜像管理，虚拟机镜像管理	<p>The process of centralizing, organizing, distributing, and tracking of software artifacts.</p> <p>集中、整理、分发、跟踪软件工件的流程</p>
Immutable infrastructure 不可变基础设施	<p>An infrastructure paradigm in which servers are never modified after they're deployed. If something needs to be updated, fixed, or modified in any way, new servers built from a common image with the appropriate changes are provisioned to replace the old ones. After they're validated, they're put into use and the old ones are decommissioned.</p> <p>The benefits of an immutable infrastructure include more consistency and reliability in your infrastructure and a simpler, more predictable deployment process. (from: https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure)</p> <p>一种基础架构范例，在该范例中，服务器在部署后再也不会修改。如果需要以任何方式更新，修复或修改某些内容，则可以使用具有适当更改的通用镜像构建新服务器，以替换旧服务器。经过验证后，它们便会</p>

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	<p>投入使用，旧的将退役。</p> <p>不变的基础架构的好处包括基础架构具有更高的一致性和可靠性，以及更简单，更可预测的部署流程。</p> <p>（摘自：https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure）</p>
Infrastructure as Code 基础架构即代码	<p>The management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning that the DevSecOps team uses for source code. Infrastructure as Code evolved to solve the problem of environment drift in the release pipeline.</p> <p>使用与 DevSecOps 团队用于管理源代码的相同的版本控制方法，以描述性模型管理基础架构（网络，虚拟机，负载均衡和连接拓扑）。随着 IaaS 的发展，它解决了发布流水线中的环境漂移问题。</p>
Kubernetes	<p>An open-source system for automating deployment, scaling, and management of containerized applications. It was originally designed by Google and is now maintained by the CNCF. Many vendors also provide their own branded Kubernetes. It works with a range of container runtimes. Many cloud services offer a Kubernetes-based platform as a service.</p> <p>一个开源系统，用于自动化容器化应用程序的部署，扩展和管理。它最初由 Google 设计，现在由 CNCF 维护。许多供应商还提供了自己的品牌 Kubernetes。它适用于一系列容器运行时。许多云服务都提供基于 Kubernetes 的平台即服务。</p>
Lockdown 封锁	<p>The closing or removal of weaknesses and vulnerabilities from software.</p> <p>关闭或消除软件中的弱点和漏洞。</p>
Microservices 微服务	<p>Microservices are both an architecture and an approach to software development in which a monolith application is broken down into a suite of loosely coupled independent services that can be altered, updated, or taken down without affecting the rest of the application.</p> <p>微服务既是软件开发的体系结构，又是软件开发的一种方法，其中，将整体应用程序分解为一组松散耦合的独立服务，可以对其进行更改，更新或删除，而不会影响应用程序的其余部分。</p>
Mission Application Platform 任务应用平台	<p>The mission application platform is the underlying hosting environment resources and capabilities, plus any mission program enhanced capabilities that form the base upon which the mission software application operates.</p> <p>任务应用程序平台是基础托管环境的资源和能力，以及构成任务软件应用程序运行基础的任何任务程序增强功能。</p>
Monitoring Security Monitoring 监控 安全监控	<p>The regular observation, recording, and presentation of activities.</p> <p>定期观察，记录和展现活动。</p>
Node 节点 Cluster node 集群节点	<p>A node is a worker machine in CNCF Kubernetes. A node may be a VM or physical machine, depending on the cluster. Each node contains the services necessary to run pods and is managed by the master components, including the node controller.</p> <p>节点是 CNCF Kubernetes 中的工作机。节点可以是 VM 或物理机，具体取决于集群。每个节点都包含运行 Pod 所需的服务，并由主控组件（包括节点控制器）进行管理。</p>
OCI	<p>"An open governance structure for the express purpose of creating open industry standards around container formats and runtime" - https://www.opencontainers.org/</p> <p>“明确表达围绕容器格式和运行时创建开放行业标准的开放治理结构” - https://www.opencontainers.org/</p>
OCI Compliant Container 符合 OCI 规范的容器	<p>The image of the OCI compliant container must conform with the OCI Image Specification.</p> <p>符合 OCI 规范的容器镜像必须符合 OCI 镜像规范。</p>
OCI Compliant Container Runtime 符合 OCI 规范的容器运行时	<p>A container runtime is software that executes containers and manages container images on a node. OCI compliant container runtime must conform with the OCI Runtime Specification.</p> <p>容器运行时是一种在节点上执行容器并管理容器镜像的软件。符合 OCI 的容器运行时必须符合 OCI 运行时规范。</p>
Orchestration 编排器	<p>Automated configuration, coordination, and management.</p>

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

	自动化的配置，协调和管理。
Platform 平台	A platform is a group of resources and capabilities that form a base upon which other capabilities or services are built and operated. 平台是一组资源和能力，构成了其他能力或服务被构建和运行的基础。
Pod	A group of containers that run on the same CNCF Kubernetes worker node and share libraries and OS resources. 在同一 CNCF Kubernetes 工作者节点上运行，并共享库和操作系统资源的一组容器。
Provisioning 供应	Instantiation, configuration, and management of software or the environments that host or contain software. 实例化，配置和管理软件或托管或包含软件的环境。
Reporting 报告	An account or statement describing an event. 用于描述事件的解释或报表。
Repository 仓库	A central place in which data is aggregated and maintained in an organized way. 数据以一种有组织的方式集中和维护的中心位置。
Resource 资源	CPU, Memory, Disk, Networking CPU,内存，磁盘，网络
Scanning, Security Scanning 扫描，安全扫描	The evaluation of software for cybersecurity weaknesses and vulnerabilities. 评估软件的网络安全弱点和漏洞。
Sidecar Container Security Stack 边车容器安全堆栈	A sidecar container security stack is a stack of sidecar containers aimed to enhance the security capabilities of the main containers in the same Pod. 边车容器安全堆栈是指旨在提高同一 Pod 中主要容器的安全能力的边车容器堆栈。
Sidecar 边车	A sidecar is a container used to extend or enhance the functionality of an application container without strong coupling between two. When using CNCF Kubernetes, a pod is composed of one or more containers. A sidecar is a utility container in the pod and its purpose is to support the main application container or containers inside the same pod. For more information, see Section 6.4.4 and Kubernetes documentation. 边车是用于扩展或增强应用程序容器功能的容器，而两者之间没有强烈的耦合。使用 CNCF Kubernetes 时，一个 Pod 由一个或多个容器组成。边车是 Pod 中的工具容器，其目的是在同一容器中支撑一个或多个主应用程序容器。有关更多信息，请参见第 6.4.4 节和 Kubernetes 文档。
Telemetry 遥测	Capability to take measurements and collect and distribute the data. 能够进行测量、收集和分发数据的能力。
Test Coverage, Code Coverage 测试覆盖率，代码覆盖率	Test coverage is a measure used to describe what percentage of application code is exercised when a test suite runs. A higher percentage indicates more source code executed during testing, which suggests a lower chance of containing undetected bugs. 测试覆盖率是一种度量指标，用于描述测试套件运行时所执行的应用程序代码所占的百分比。较高的百分比表示在测试流程中执行的源代码更多，这表明包含未检测到的错误的可能性更低。
Virtual Machine (VM) 虚拟机	Emulates a physical computer in software. Several VMs can run on the same physical device. 在软件中模拟物理计算机。多个 VM 可以在同一物理设备上运行。
Virtual Network 虚拟网络	Networks constructed of software-defined devices. 由软件定义设备构成的网络。
Virtual Storage 虚拟存储	Storage constructed of software-defined devices. 由软件定义设备构成的存储。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

Appendix C References 附录 C 参考文档

The following references are utilized in the development of the DevSecOps reference design document:

开发 DevSecOps 参考设计文件时使用了以下参考文件：

- [1] Department of Defense, "DoD Cloud Computing Strategy," December 2018.

国防部，DoD 云计算战略，2018 年 12 月。

- [2] DISA, "Department Of Defense Cloud Computing Security Requirements Guide, V1R2," 18 March, 2016.

DISA，国防部云计算安全需求指南，V1R2，2016 年 3 月 18 日。

- [3] DISA, "DoD Secure Cloud Computing Architecture (SCCA) Functional Requirements," January 31, 2017.

DISA，"DoD 安全云计算架构（SCCA）功能需求”，2017 年 1 月 31 日。

- [4] White House, "Presidential Executive Order on Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure (EO 1380)," May 11, 2017.

白宫，"关于加强联邦网络和关键基础设施网络安全的总统行政命令（EO 1380）”，2017 年 5 月 11 日。

- [5] National Institute of Standards and Technology, Framework for Improving Critical Infrastructure Cybersecurity, 2018.

国家标准与技术研究所，"改善关键基础设施网络安全框架”，2018 年。

- [6] DISA, "Department Of Defense Container Hardening Security Requirements Guide (Draft)," 2019.

DISA，“国防部容器加固安全要求指南（草案）”，2019 年。

- [7] Office of the DoD CIO, "DoD DevSecOps Playbook", 2019. [Online]. Available: <https://www.milsuite.mil/book/groups/dod-enterprise-devsecops>.

国防部 CIO 办公室，DoD DevSecOps 实战手册，2019。[在线]。可用：
<https://www.milsuite.mil/book/groups/dod-enterprise-devsecops>。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后

[8] Puppet Labs, "Puppet State of DevOps Report 2018", Puppet Labs, 2018.

Puppet Labs, “2018 年 DevOps 态势报告”, Puppet Labs, 2018 年。

[9] Defense Threat Reduction Agency (DTRA), "Next-Generation Technology Governance, " 2018.

国防威胁降低局（DTRA），“下一代技术治理，”2018 年。

[10] DoD, "DoDI 8510.01: Risk Management Framework for DoD Information Technology," 24 May 2016.

国防部，“DoDI 8510.01：国防部信息技术风险管理框架，”2016 年 5 月 24 日。

[11] E. Ries, "The Lean Startup," [Online]. Available:
<http://theleanstartup.com/principles>. [Accessed 15 July 2019].

**E.Ries, “The Lean Startup”, [在线]。可用：
<http://theleanstartup.com/principles>。【访问日期：2019 年 7 月 15 日】。**

[12] NIST, "NIST Special Publication 800-190, Application Container Security Guide, " September 2017.

NIST, “NIST 特别出版物 800-190，应用容器安全指南”，2017 年 9 月。

主编：唐龙 译者：唐龙（1-2 章，4.1-4.2）、张晨晖（前言，附录，全文校对）、周景川（4.3-7 章）、赵庆安（3 章）排名不分先后