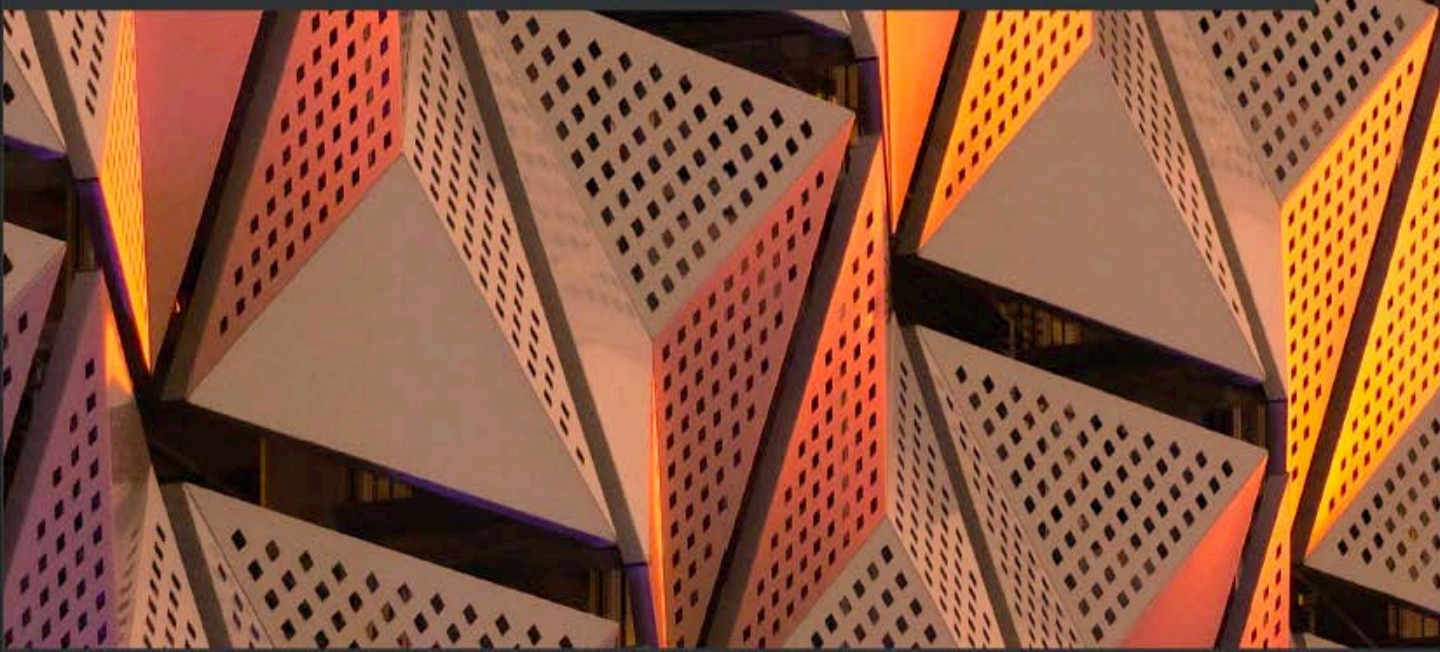
Enterprise DevOps for Architects

Leverage AIOps and DevSecOps for secure digital transformation



Jeroen Mulder

企业DevOps架构指南

安全篇

基于AIOps和DevSecOps进行安全的数字化转型

【拆书版：DevSecOps节选】

（非官方不专业中文学习版）

Jeroen Mulder

翻译：庄飞（OXFE1FE1）

2022 年 2 月

译者说明

本文翻译工作为公益性质的学习目的。翻译本身基于水平与目标不同，并不能与专业翻译水平相提并论，在此基础之上，为维护原作者知识产权保护问题，本翻译内容任何人不得以任何商业形式产生盈利行为。因本版本在翻译发布前市场未出现相同书目的中文版本，故本翻译版不产生在今后可能出现的商业翻译版本的冲突，但建议阅读者能够购买后期可能出现的商业翻译版本，以便能够通过阅读更高水平的翻译版本弥补本版本的缺失和遗漏。欢迎广大读者能够对书中翻译的缺漏进行补充和探讨。

本文由微信公众号"DevSecOps联盟"运营者OXFE1FE1独立翻译。DevSecOps联盟初衷是基于开源精神，向业界分享DevSecOps安全实践。后续稿件问题请联系DevSecOps联盟(idevsecops@outlook.com)进行不定期订正，力求保留书籍原意。

OXFE1FE1

2022-02-10

拆书版说明

《企业DevOps架构师指南》前2个章节主要介绍DevOps及AIOps，目录如下：

- **第 1 节：企业DevOps架构**
  - 1 定义企业DevOps参考架构
  - 2 从架构上管理DevOps
  - 3 DevOps质量架构
  - 4 规模化DevOps
  - 5 基于SRE的下一代DevOps架构
- **第 2 节：AIOps驱动左移**
  - 6 定义运维架构
  - 7 AI对DevOps的影响
  - 8 AIOps架构
  - 9 DevOps中集成AIOps
  - 10 向NoOps迈出最后一步

本拆书版略过前2节，摘录自原书 "**第 3 节： DevSecOps安全赋能**"。

第3节：DevSecOps安全赋能

在前面的章节中，我们实现了开发和运营的自动化。现在，最后一步是将安全也一起自动化。这就是DevSecOps的起点：在开发和发布到运营的每一步中集成安全。DevSecOps包括测试、部署和最终交付。在这一部分中，你将深入了解DevSecOps，并学习其架构是什么样的，以及安全为何必须与DevOps集成。

本节将包括如下章节：

- 第11章，了解DevOps安全
- 第12章，DevSecOps架构
- 第13章，行业安全框架与DevSecOps
- 第14章，DevSecOps与DevOps集成
- 第15章，实现零信任架构

11
了解DevOps安全

谈到云计算、现代应用，以及数字化转型，就不能不谈安全。一个流行的术语是设计安全（security by design）。但即使是设计安全也需要嵌入到企业架构中。这也适用于DevOps生命周期：DevOps需要设计安全。在我们讨论这个问题和零信任等原则之前，我们首先需要对安全有一个很好的理解，以及它是如何影响DevOps实践的。本章对DevOps中的安全进行了介绍。

阅读本章后，你将了解到为什么将安全纳入企业架构是很重要的，以及架构师如何收集和评估风险，并能够识别DevOps中具体有哪些风险。你还将了解如何设立安全控制，以及DevSecOps中需要关注的重点是什么。

在这一章中，我们将涵盖以下主题：

- 在企业架构中安全内嵌
- 了解DevOps中的安全风险
- 掌握DevSecOps的技巧
- 定义需求和度量指标

在企业架构中嵌入安全

这是一个你几乎每天都能读到的话题：受到某种黑客或攻击的企业。系统中最小的漏洞都会被犯罪分子发现并利用。目前（2021年），最流行的攻击有以下几种：

- **勒索软件**
- **网络钓鱼**
- **拒绝服务**

前两个，勒索软件和网络钓鱼，真正利用了企业防御层的漏洞。最后一种基本上是对一个系统进行大量的流量轰炸，使系统最终崩溃。所有这三种都是相当容易执行的。事实上，你可以购买软件甚至服务，对目标地址发起攻击。你甚至不需要到暗网中去找这些东西。它就在那里，在公开的地方在，正常的互联网上。

一个企业如何保护自己免受这些攻击？首先，必须认识到任何企业的IT已经变得更加复杂。IT系统不再只在一个私有的数据中心，而是变成了一个生态系统，使用公共云、私有的数据中心或同城的私有技术栈、**平台即服务**（PaaS）和**软件即服务**（SaaS）。安全必须是企业使用的每一项服务的内在因素。

企业安全的四个传统原则如下。

- **预防**
- **检测**
- **纠正**

- **指导**

前三个是关于检测安全问题，用缓解行动来纠正它们，但显然，如果问题能够被预防，那就更好了--因此，预防是第一优先。

指导是关于指导方针和护栏：企业定义政策和安全标准以保持所有系统的安全。这就是第五项原则的作用：一致性。在一个有不同部门、群体和团队的企业中，需要确保安全在该企业的每个角落都得到实施。你不能让一个部门或团队不遵守企业的安全。换句话说：链条的强度是由最薄弱的环节决定的。

那么，我们该从哪里开始建设企业安全呢？企业架构框架可以帮助你开始。Sherwood Applied Business Security Architecture（SABSA）就是一个例子，它提供了一种定义安全架构的方法。它由六个层次组成，如下图所示。

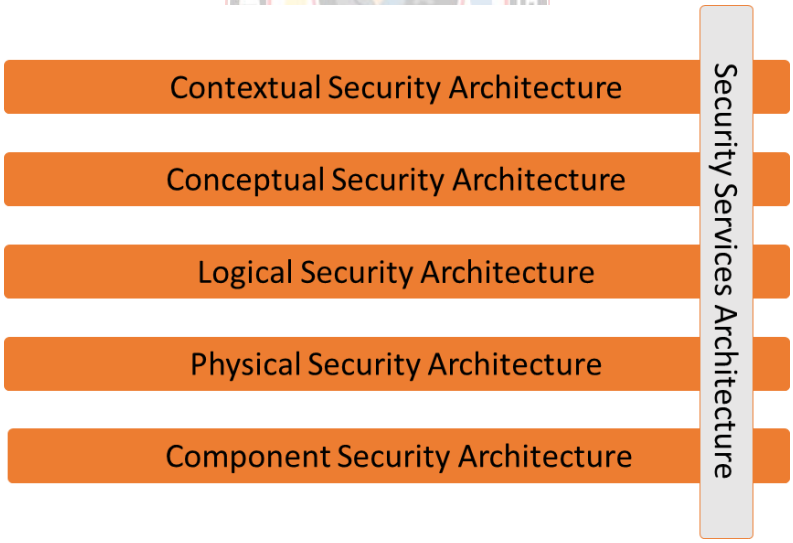


图11.1-安全架构的SABSA模型

信息系统审计与控制协会（ISACA）将SABSA与较为知名的**信息及相关技术控制**

目标（COBIT）的五项原则相结合。这五项原则如下：

- 满足企业利益相关者的需求
- 覆盖企业端到端
- 应用单一的综合框架
- 促成一个整体的方法
- 将治理与管理分开

SABSA和COBIT的结合带来了一种自上而下的方法来定义整个企业的架构。企业架构师必须与首席安全架构师合作，至少要执行以下步骤：

1. 确定业务目标。这是企业架构的第一个阶段。企业架构总是从业务战略、目标和目的开始。
2. 识别商业风险。
3. 确定管理风险所需的控制措施。
4. 设计并实施这些控制措施，例如：

   - 安全治理流程
   - 访问控制
   - 事件管理流程
   - 证书管理流程

5. 为所用平台、网络、操作系统和数据存储等设计物理架构。物理架构还应该包括云平台和服务，如PaaS和SaaS以及DevOps实践，如CI/CD。
6. 根据企业必须遵守的合规性和安全标准及协议，验证业务和物理架构。
7. 定义并实施运营架构，例如：

   - 配置管理

   - 监控

   - 日志

   - 变更管理

注意事项

前面提供的清单并不意味着是详尽的。在进一步阅读部分，我们包括了一个链接到ISACA关于企业安全架构的期刊。在该杂志中，你会发现更多关于所述步骤的细节，使用SABSA和COBIT。

定义安全架构的一个重点是了解风险。风险是什么，哪些系统有风险，对企业及其业务的影响是什么？DevOps带来了它自己的风险。我们将在下一节讨论这个问题。

了解DevOps中的安全风险

互联网上有一个经典的卡通片。它展示了一个拳击场。演讲者在拳击台的左角宣布了一套巨大的安全工具和规则。然后，在右角，他宣布了戴夫：一个看起来像

书呆子的家伙，穿着一件写着"人为错误"的衬衫。意思是：你可以拥有世界上所有的安全系统，但它不能阻止人为错误。而开发仍然主要是由人完成的工作，人会犯错误，这是DevOps中最大的风险，或者还有其他需要注意的特别风险？我们将在本节中讨论这个问题。

要回答DevOps是否含有特定风险的问题，答案为是的。实施DevOps而不关注安全，仅仅通过提高系统的攻击面，会明确地增加攻击的风险。有三个重点需要解决。

- **访问管理。**DevOps团队可能使用代码仓库，由开发人员或工具手动访问。代码需要被保护，即使在开放源码模式下运行，或者代码被共享，以便更多的开发者可以对代码做出贡献。即使在这种情况下，公司也会希望规范对代码的访问，以避免代码被公开，甚至更糟的是，恶意代码被注入。

  你需要一个基于角色的代码仓库访问模型：谁有只读权和写入权，谁有完全访问权--出于什么原因？

  追踪账户的情况。例如，GitHub 公司可以有内部仓库，只有指定的员工或公司的工具可以访问。在这个内部仓库中，管理员会分配角色。凭证是根据公司的安全策略来设置的。一个推荐的保持访问控制的方法是实施特权访问管理（PAM）。

  因为DevOps，团队将不得不创建更多的特权账户，在开发人员和工具之间手动或自动共享。这些账户还涉及服务账户、加密密钥、SSH密钥和API证书，这些都保存在仓库中。未经授权访问这些仓库是灾难性的。

  PAM解决方案提供了一种授权和审计DevOps生命周期中任何活动的方

法。这些解决方案大多使用密钥保险库来保证访问细节的安全，并在用户实际访问存库之前对其进行认证和授权。

- **缺少DevOps工作方式和工具的护栏和指南。**访问代码是一回事；接下来的事情是：我们该如何处理这些代码？企业不太可能允许DevOps团队直接提交和推送新代码到生产中。首先，企业或首席架构师--在大多数大型企业中，会有一组领先的架构师--真的需要考虑首选工具集。

  重要的是，每个DevOps团队都使用该工具集，而不是实施他们自己选择的工具，不管它多么诱人。问题是，DevOps工具集没有一个共同的安全策略标准，例如访问控制。这是企业本身需要建立和实施的东西，如果你使用一个工具集，这就更容易了。

  这同样适用于工作方式：这需要在整个企业中保持一致；我们怎么强调都不为过。所有这些都必须在首选技术清单和DevOps护栏和原则中定义。通常情况下，会有一个主分支。新的代码将首先被推送到一个单独的分支--通常是称为特性分支，并在此进行测试。经过验证的、积极的测试结果后，代码被合并到主干。主干代码或主分支再次被存储在仓库中。这一原则如下图所示：

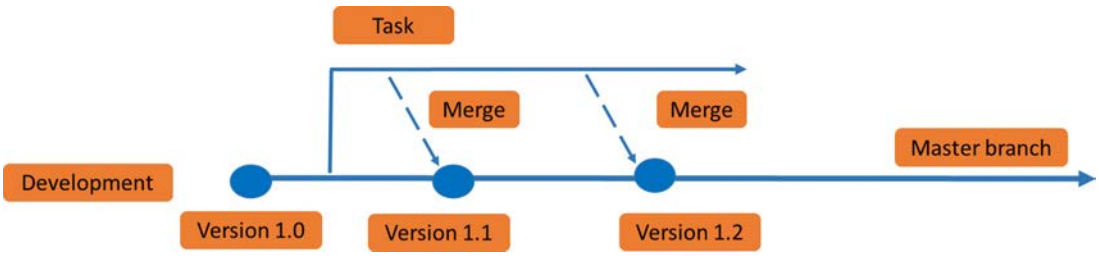


图 11.2-合并新代码的原则

护栏定义了代码何时从仓库中拉取，如何提交到特性分支，以及为了与主干合并，如何进行测试并最终发布。

- **注重开发过程和速度，忽视安全。**DevOps主要关注开发速度。但这绝不应该成为忽视安全的借口。在下面的章节中，我们将用一个真实的例子来说明这一点。

总而言之，DevOps安全主要关注三个方面：

- **可追溯性。**追踪DevOps生命周期和流水线中的每个行动。
- **可审计性。**确保在DevOps中开发的系统符合企业的安全标准和企业所认可的行业框架。第13章，使用行业安全框架开展DevSecOps工作，更详细地谈到了这一点。
- **可见性**：要有可靠的监控系统。

现在我们对DevOps中的安全风险有了充分的了解。在下一节，我们将讨论如何开始开展DevSecOps。

掌握DevSecOps的技巧

安全始于对仓库的访问，即DevOps生命周期开始的代码源。正如我们到目前为止所了解的，我们希望在开发、测试和部署中尽可能多地实现自动化。接下来，通过采用DevOps，企业希望加快新应用和功能的开发和部署。速度和敏捷性可能会导致安全风险，因为代码没有得到充分的测试，或者更糟糕的是，为了争取时间，代码没

有应用适当的安全策略就被推到生产中去。让我们用一个现实生活中的例子来说明。

开发人员从仓库中fork代码，并开始在该代码上工作。在某个阶段，它需要被推送到指定的基础设施来运行该代码。在开发阶段，代码运行良好，因为它还没有与生产系统对接。一旦当代码准备在生产中发布时，它将需要建立这些连接。通常，在企业中，特定的路由和防火墙端口需要被打开，以允许这种连接和流量的传输。

防火墙规则，更具体地说，开放端口和分配防火墙规则通常不是自动完成的；安全工程师希望在批准实施设置之前对请求进行评估。在很多情况下，这确实拖累了整个DevOps和敏捷的进程。但几乎每个企业都是这样的做法：安全是最终部署前的最后一站，没有嵌入到DevOps生命周期中。然而，绕过这一点是个坏主意。它将增加代码和系统的攻击面。

要点：安全必须嵌入DevOps中--它使开发和部署成为开发人员、运维人员和安全工程师的共同责任。安全不仅仅是安全团队的责任，而是整个DevOps团队的责任。

DevSecOps入手点

安全扫描从代码被拉出仓库时就开始了。首先要做的是让**基于角色的访问控制**（RBAC）模型应用到仓库中。RBAC定义了谁有访问权，访问到什么级别。一个身份是只允许查看代码，还是被授予完全的访问权，可以拉取代码、修改代码、添加代码到版本库？请注意，这不一定是人。DevOps工具也是身份，你需要考虑到对这些工具的必要访问。

在整个DevOpss生命周期中，代码被不断地审计、扫描和测试以发现漏洞。一

个非常简单的例子是，如果一个应用程序禁止连接到互联网，而代码中发现一处指向端口80，那么它可能被标记为一个风险。如果一个应用程序依赖一个通过互联网连接的服务，它可能被允许连接互联网，但这需要根据安全策略进行验证。扫描、测试和验证需要在开发过程中尽可能早地完成。另外，左移原则也适用于这里。

在DevOps中嵌入安全的最大好处是，我们也可以通过自动化来做检查，一旦发现或确认有漏洞，甚至可以立即打补丁和修复。例如，如果一个新发布的代码库包含**通用漏洞披露（CVE）**的补丁，这可以被注入安全基线并与安全测试程序集成。代码会根据这个新的基线自动检查。软件可以执行测试，验证代码是否符合要求，如果代码没有通过测试，在出现问题的情况下进行标记，或者通过安装补丁触发自动化补救。这不仅适用于应用程序代码，当然也适用于使用的基础设施、操作系统和中间件。

使用容器的DevSecOps

另一个例子：越来越多的企业使用容器进行代码分发工作。就像虚拟机一样，容器也需要符合安全策略。企业可能会使用加固的容器，如下设置策略：

- **加固的主机操作系统**。通常，这些是Linux操作系统，可以保护主机不被受感染的容器破坏。例如，应用于Linux主机的安全软件包，如SELinux和seccomp；这些Linux发行版允许对内核设置进行**强制访问控制（MAC）**。企业可以选择开发自己的**Linux安全模块（LSM）**。
- **容器运行时安全策略**。为挂载容器、特权容器设置特定权限，在容器中禁用SSH，设置绑定端口和暴露端口。

下图显示了基于CVE的容器安全扫描的原理。

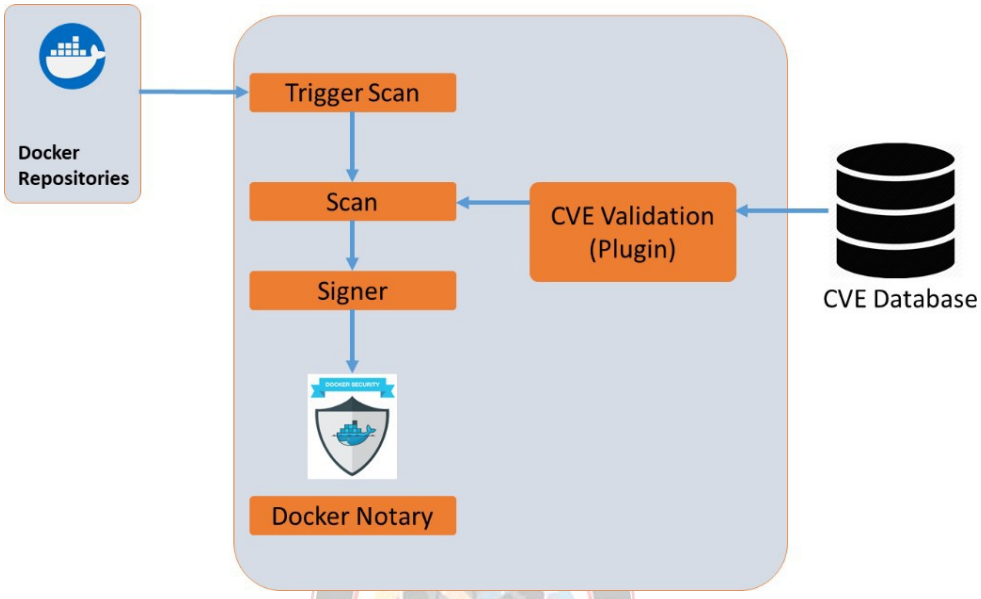


图 11.3-Docker安全扫描的概念

扫描是基于CVE数据库的数据完成的。这可以是来自**国家标准技术研究所**（NIST）、MITRE，或微软等供应商的数据（他们也为自己的产品和服务发布CVE通告）。扫描之后，Docker会对从仓库中提取的镜像进行签名。为此，它使用了Docker Notary，它验证了镜像，并防止开发者使用没有签名的镜像。

我们现在有了经过加固、验证的容器，并设置了特定的权限。现在，下一个最重要的事情是控制设置；一旦容器的权限被设置，就不应该有办法改变这些权限。容器不应该能够获得新的权限；否则，它们就没有被加固。在Docker中，有一个简单的方法来检查和设置这个。使用下面的命令，你可以列出容器的所有安全设置。

它将返回关于加固容器的信息，为容器创建的独立分区，以及对Docker文件

和目录的审计配置。接下来，设置无新权限选项。

```
docker run <run-options> --security-opt=no-new-
privileges <image> <cmd>
```

很明显，这些设置需要持续地评估。DevSecOps和DevOps一样，是一个可重复、持续的过程，目的是为了持续改进。这意味着，在企业采用DevSecOps的过程中，需要对目标、需求、新识别的风险以及缓解这些风险的控制措施进行评估和处理。我们将在本章的最后一节讨论这个问题。

定义需求和度量指标

在本章的第一节中，我们讨论了架构师为定义企业安全必须采取的步骤。在本节中，我们将解释如何收集、验证需求和指标，并转化为控制和KPI。

业务目标

我们在第一章 "定义企业DevOps的参考架构 "中已经谈到了这一点，但显然，了解企业想要实现的目标很重要。他们所处的市场是什么，他们如何为这些市场的客户提供服务，以及产品组合是什么？如果一个企业经营的是金融产品或医疗保健，这确实有很大的区别。他们的市场决定了风险水平。银行或投资公司的风险可 能主要是金融方面的，而对于医疗保健，最大的风险可能是涉及病人的生命。目标也会不同：投资公司的目标可能是用资金支持尽可能多的企业，而医疗保健公司的目标是用具体的解决方案来治愈人们。所以，企业和目标将决定企业的属性。

业务属性

属性可以是系统的可用性、数据的准确性，以及客户的隐私。在业务类型和目标的旁边，法规将设定这些属性的水平。例如，金融机构将不得不遵守国家和国际金融法规，如**萨班斯-奥克斯利法案**（SOx），而医疗机构将遵守**健康保险可携性和责任法案**（HIPAA）。请注意，这些规定是经过审计的。我们将在*第13章 行业安全框架与DevSecOps*中谈及这个问题。

风险

基于业务目标和属性，我们可以定义风险。如果某个事件发生，什么将是对企业的主要威胁？如果你考虑到可用性，可能是企业没有办法在关键系统因任何原因停止时执行故障转移的情况。那么企业如何从重大故障中恢复呢？没有冗余的系统可能是一种风险，就像可被犯罪分子利用的应用程序漏洞一样。MITRE ATT&CK框架可以帮助识别风险；它也将在*"第13章 行业安全框架与DevSecOps"*中讨论。

控制措施

下一步是定义风险控制措施。如下示例：

- 创建一个适应灾难恢复的业务连续性计划。
- 实施带有身份存储和保险库的**公钥基础设施**（PKI），以确保用户的隐私。
- 实施具有特定防火墙规则的应用防火墙，以保护关键系统。

- 设置控制措施来管理、更新和升级前面所有的内容：维护业务连续性计划，管理安全策略，并定期评估防火墙设置。

验证这些控制措施和审计的属性和应用的安全框架相关联。对于每一个控制措施，都必须有一个可以在审计中验证的理由。

好消息是，企业不必自己去考虑这些控制措施。**互联网安全中心（CIS）**已经为很多IT领域定义了控制措施：CIS控制框架。基本的CIS控制措施包括特权访问的控制使用，所有IT系统资产（包括容器）的安全配置，以及网络端口、协议和服务的控制措施。对于Azure、AWS、GCP以及Kubernetes等容器平台，CIS已经定义了具体的控制措施。

CIS、MITRE和其他框架，以及它们对DevOps的影响将在后面的章节中进一步讨论，但首先，我们将了解DevSecOps架构应该包括什么。这是下一章的主题。

摘要

本章介绍了将安全融入DevOps的情况，讨论了DevSecOps的概念。我们已经讨论了安全在企业架构中的重要性，以及如何推动企业DevOps的安全。我们已经了解了采用DevOps所涉及的主要安全风险，并对作为DevOps实践中使用最多的技术之一的容器的安全进行了仔细的研究。这样，我们确定了采用DevSecOps的一些关键入手点。

在最后一节，我们学习了如何从业务目标和业务属性中收集和评估风险，介绍了常用的安全控制框架，如CIS框架。通过一些例子，我们探讨了为了实现一个可以应用于DevOps的安全标准，架构师需要采取的各种步骤。

在下一章中，我们将更详细地探讨DevSecOps的架构，然后开始将安全策略和行业框架与DevOps的工作方式相结合。

问题

1. 说出企业安全的四个传统原则。
2. Docker用什么来验证签名的容器？
3. 真或假。安全不应妨碍DevOps的速度和敏捷性。

进一步阅读

4. ISACA杂志关于使用SABSA和COBIT的企业安全架构：

https://www.isaca.org/resources/isaca-journal/issues/2017/ volume-4/enterpris-security-architecturea-top-down-approach

5. CIS网站：cissecurity.org

12
DevSecOps架构

与企业IT领域的一样，DevSecOps需要一个架构基础。在本章中，你将学习如何为DevSecOps实践组成参考架构，并设计DevSecOps的流水线。我们还将讨论主要公有云供应商的最佳DevSecOps实践，即AWS、Azure和GCP。为此，我们将详细介绍市场上的一些领先工具。在最后一节，你将了解企业应该采取哪些步 骤来实施DevSecOps。

学完本章后，你将能够说出DevSecOps架构中的不同组件，以及如何将这些组件纳入DevSecOps流水线中。你还将学会如何保护容器，以及在各种公有云中的最佳做法是什么。最重要的是，你将能够解释为什么将安全纳入DevOps对企业来说是至关重要的。

在这一章中，我们将涵盖以下主要议题：

- DevSecOps 生态系统
- 创建参考架构

- 构建DevSecOps流水线
- 将DevSecOps应用于AWS、Azure和GCP
- 规划部署

DevSecOps生态系统

在上一章中，我们讨论了安全原则以及对DevOps工作方式的影响。我们得出的结论是，安全必须是开发和部署周期中每一步的核心，从代码从仓库中提取到实际的代码提交并推送到生产中。在本章中，我们将探讨DevSecOps的基础，即**嵌入安全的DevOps**。

DevSecOps由三层组成。

- **文化**。这不是一个技术层面，但人们常常忘记，DevOps不仅仅是应用工具和 创建CI/CD流水线。很明显，这也适用于DevSecOps。在DevSecOps中，每个团队成员都觉得自己对安全有责任，并采取相应的行动，承担起安全的责任。但这并不意味着安全专家已经过时了。在团队中配备一名安全工程师或专业人员，有时被称为安全卫士（security champion），这是一个很好的做法。这个人必须在应用安全标准和政策方面领导所有流程，确保合规。
- **设计安全**。安全被嵌入到系统的每一层。这通常意味着企业有一个定义好的架构，涵盖了安全的每一个方面，并对系统实施安全措施：认证、授权、

保密性、数据完整性、隐私、问责制和可用性，包括系统受到攻击时的补救和纠正措施。

软件开发人员不需要在每次设计和构建新的应用程序或功能时都考虑到安全问题—只需要在开发开始时就采用这些措施。安全架构、框架、规则和标准是集中管理的。

- **自动化。**在DevSecOps中，我们希望尽可能地实现自动化，这包括安全。安全自动化的理由是，我们可以防止人为错误，也可以有自动化的关卡，进行代码扫描以发现可能的漏洞或不合规的组件，如未经许可的代码。安全负责人也要承担起安全自动化的责任，但要和团队一起进行。自动化还意味着在发生攻击或违规时，进行自动化审计和收集证据。接下来，自动化过程确保在DevSecOps实践中，安全指标被收集并送回进行反馈。例如，如果在扫描时，发现了代码中的漏洞或许可证违规，证据将被收集并发送便于反馈。

为了管理这些层面，DevSecOps依赖于以下组件：

- **加固仓库**
- **应用（代码）安全**
- **云平台安全**
- **漏洞评估和测试**

DevSecOps不应该与安全即服务（SECaaS）混为一谈。SECaaS可以为DevSecOps实践中的一个组成部分，但SECaaS的概念主要是指将安全作为一种责

任转移给服务提供商。这是一种采购模式，允许企业以订阅方式从服务提供商那里获得网络安全。实施SECaaS有很好的理由，其中之一是基于最新的情报，供应商负责所有的安全更新。企业可以为事件响应时间和安全实践的及时应用定义服务级别协议。正如我们之前提到的，它可以被整合到DevSecOps中，但SECaaS也意味着企业必须依靠第三方来实施和管理安全基线。

在下一节中，我们将讨论DevSecOps的组成部分并定义参考架构。

创建参考架构

在讨论DevSecOps的参考架构之前，我们需要了解DevOps的作用是什么以及安全如何融入其中。DevOps是关于软件开发生命周期的。我们必须注意的一个重要的事实是，开发人员越来越多地使用开源组件。这是有道理的，因为这在开发新代码时提供了极大的灵活性。

开源是社区驱动的，所以开发者可以互相贡献代码，加快进程。项目可以在开放的Git和GitHub仓库中共享，也可以在企业内部共享。内部开源（InnerSource)类型的项目就是一个很好的例子。内部开源，在组织的范围内使用开源的最佳实践进行软件开发。通常情况下，内部开源项目会使用GitHub或类似的封闭的、访问受限的仓库。

然而，从安全的角度来看，开源也有一些需要解决的风险。由于它的开放性、社区性、（开放源码的优势），代码库引入漏洞的风险增加了。第二个风险是许可证合规性。许可证在开源中并不是每个人都会想到的，但要知道即使是开源软件和工具也需要许可证。

我们先来看看这个过程。软件开发生命周期是一个重复的过程。开发人员从仓库中拉出源代码，然后触发构建。在代码编写完成后，代码被打包并启用，以便部署到推广路径的下一个阶段；也就是测试、验收，最后是生产。整个过程是通过CI/CD流水线来促进的，并进行监控。正如我们在前几章中得出的结论，在整个过程中代码测试是至关重要的。我们也要扫描代码的安全性和合规性。这应该在过程中的每一个步骤都进行。

事实上，我们从DevOps过程的一开始就需要安全。在实践中，这意味着我们从代码从仓库中拉取的那一刻起就开始扫描以发现安全问题。代码仓库确实也是软件开发生命周期的一部分，所以必须保护这些仓库不被非法访问。这就要求对仓库实施**基于角色的访问控制**（RBAC）以及**身份和访问管理**（IAM）。

考虑到这一点，我们可以用以下组件创建DevSecOps的参考架构：

- 用RBAC模式访问仓库
- **静态应用安全测试**（SAST）。这将检测源代码中的错误
- **软件组成分析**（SCA）。这将检测代码中的依赖关系
- **动态应用安全测试**（DAST）。这将动态地扫描代码

这些组件被嵌入到DevSecOps流水线中，我们将在下一节讨论。

构建DevSecOps流水线

让我们先看看一个常见的DevOps流水线。基本流水线如下图所示。

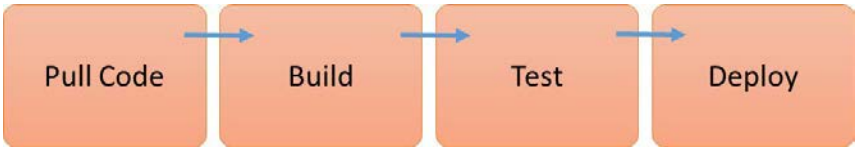


图 12.1-DevOps流水线

流水线中的基本步骤如下：

- 从仓库中拉取代码
- 构建
- 测试
- 部署

在DevSecOps中，我们将安全嵌入到流水线中，使安全标准和策略成为流水线的一个组成部分。安全是一个应用于流水线中每一步的层级，但它确实包括几个步骤。如下图所示：

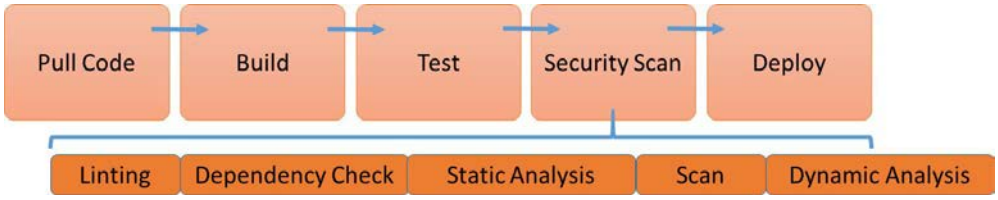


图12.2-DevSecOps流水线

这些步骤如下：

1. **依赖性检查**：首先，任何可使代码暴露在可利用的风险中的漏洞都应该被移除。这包括依赖其他代码运行的代码。代码的依赖性是有区别的：开发

人员可以有受控的和不受控的依赖性。作为一种常见的做法，我们不希望在代码中存在依赖性。其风险在于，如果依赖关系所依赖的代码被攻破或该代码的功能被停止，整个应用程序可能会失败。另外，在某段代码中注入恶意软件， 但其与其他代码有依赖关系，则会感染整个代码栈。因此，基本原则是消除依赖关系--这是零信任的基本原则之一，我们将在 *"第15章 实施零信任架构"* 中进一步讨论。

软件包管理器将检查代码。例如，Python代码的pipenv和Node.js的npm。这里用于检查的命令分别是 pipenv check 和 npm audit。

TIPS

查看pipenv网站上的脚本和教程 https://pipenv.pypa.io/en/latest/。看看https://docs.npmjs.com/cli/v6/commands/npm-audit，了解npm代码检查。

2. **静态分析**: 这可以检查不良的编码实践，如不良配置。几乎每一种编码语言都有开源的工具。一些工具的例子如下。
   - ArchUnitNet和Puma Scan for C#
   - Go Vet for Go
   - Checkstyle和Owasp dependency checks for Java
   - Flow for JavaScript

- Parse for PHP
- Bandit for Python

> TIPS
>
> 这个列表不是详尽的。在https://github.com/analysis-tools-dev/static-analysis，你会发现一个当前最常用的工具的列表。

3. **扫描**：开发人员可能会使用容器，从而使用容器镜像来构建和打包他们的应用程序。这些镜像需要扫描所使用的二进制文件和库中的漏洞。这种扫描是通过已知漏洞的基础列表完成的；这些列表由**国家标准技术研究所**（NIST）等机构提供，但也有软件供应商以**通用漏洞和暴露**（CVE）通知的形式提供。一旦有新的CVE报告，名单就会被更新，扫描工具也会自动更新并被触发重新进行扫描。Clair（https://github.com/quay/clair）是一个执行这些扫描的开源工具，也适用于Docker镜像。扫描涉及到linting，我们将在下一节谈到加固容器时更详细地解释它。
4. **动态分析**：在web应用程序的情况下，开发人员可以运行自动web应用程序扫描，检查坏头或缺失token，以发现**跨站请求伪造**（CSRF或XSRF）漏洞。这些token可以防止利用来自受信任用户的未经授权的命令--这也可以是不同网站上的一个功能。这些自动化的动态扫描可以被集成到流水线中。OWASP Zed Attack Proxy是一个免费的web安全工具（https://owasp.org/www-project-zap/）。

现在，我们有了一个内嵌安全的CI/CD流水线，它将自动覆盖代码中最常见的漏洞。不过，有一个具体的项目我们在本节中没有触及，那就是容器的使用以及我们如何保护这些容器。我们将在下一节中研究安全容器的构建。

在流水线中使用安全容器

大多数开发人员会使用容器来打包和部署他们的代码，通常是Docker容器。在使用和保护容器方面，有一些最佳实践。为了保持容器的一致性和安全性，即使应用程序已经达到稳定状态、更新的频率较低，或者主动开发已经停止，也应该定期对其进行扫描。如果应用程序仍然在其托管不同的应用程序组件的底层容器中运行，这些容器必须被扫描，因为依赖性总是有可能产生新的漏洞。

由容器组成的应用程序是由Dockerfiles定义的。Linting(分析代码中使用的错误或不良语法)可以用来对Dockerfiles进行**静态代码分析**（SCA），确保这些文件安全。做到这一点的一个流行的linting工具是Haskell Dockerfile Linter（Hadolint）。它以Docker镜像的形式提供，可以通过以下命令轻松执行。

```
docker run --rm -i hadolint/hadolint
```

Hadolint扫描代码，如果一切正常，它将返回一个0退出码。当它发现错误或不良做法时，它将抛出一个Hadolint错误（DL）或**SellCheck错误(SC)**。

> TIPS
>
> 常见错误的概述收集在
> https://github.com/hadolint/hadolint#rules。

除了linting，Docker还推荐了一些保持容器安全的最佳实践。Docker已经考虑到了命名空间和网络堆栈，以提供隔离，除非在配置中特别指定，否则容器不能获得对其他容器的特权访问。接下来，有一些重要的事情需要考虑。

- Docker使用Docker守护程序。这个守护程序需要root权限，这导致潜在的安全风险。首先，应该只允许受信任的用户为守护进程设置控制。接下来，你需要采取行动，通过设置Docker主机和客户容器的访问权限来限制守护进程的攻击面，特别是当容器可以通过Web服务器的API进行配置时。
- 强烈建议使用Docker内容信任签名验证。这是一个dockerd二进制文件中存在的功能，允许你将Docker引擎设置为只运行已签名的镜像。对于签名本身，你可以使用Notary。
- 对Linux主机系统使用加固的模板，如AppArmor和SELinux。

如果我们遵循Docker的所有建议，我们将拥有经过测试的、不可变的镜像，例如，我们可以用来在Kubernetes上部署容器。Kubernetes将使用可信的镜像库，并负责容器的配置、扩展和负载均衡。Kubernetes的安全特性之一是它对滚动更新的支持：如果镜像库被更新了补丁或增强功能。

Kubernetes将部署新的版本并销毁以前的版本。有了这个功能，开发人员将始终确保只使用最新的、经过加固的镜像版本。

密钥管理

数据库凭证、API密钥、证书和访问令牌必须始终保存在安全的地方。CI/CD和容器的使用并没有改变这一点。强烈建议在流水线访问CI/CD的仓库之外使用一个保管库。密钥管理的最佳实践如下：

- 静态存储及传输过程进行加密。建议使用AES-256加密密钥。
- 密钥，如key，决不能存储在Git/GitHub的仓库中。
- 建议通过安全字符串作为环境变量将密钥注入到应用程序中。

Hashicorp（Terraform）提供的Vault是一个安全访问密钥的开源解决方案。该服务允许我们在整个生命周期内轻松地轮转、管理和检索数据库凭证、API密钥和其他密钥。

CyberArk提供了一个更强大的解决方案。CyberArk Conjur是一个独立于平台的密钥管理解决方案，专门为保护容器和微服务而设计。该解决方案是平台无关的，这意味着它可以被部署到任何云或企业内部系统中。

这两种工具都与Azure和AWS的密钥管理的原生环境集成，例如，Azure和AWS分别使用Azure Key Vault和AWS Secrets Manager。

AWS、Azure和GCP DevSecOps实践

在前面的章节中，我们讨论了DevSecOps的原则，以及如何用嵌入式安全构建流水线。在本节中，我们将探讨将DevSecOps应用于主要公有云平台的最佳实

践，即AWS、Azure和**谷歌云平台（GCP）**。

在AWS CodePipeline中采用DevSecOps

在我们开始探索AWS中的DevSecOps之前，我们需要了解在AWS中的部署应该基于**云采用框架（CAF）**的原则。该框架涵盖了具体的安全任务和责任，分为企业安全的四个类别或原则，我们在*"第11章 了解DevOps安全"*中讨论过：

- 预防
- 检测
- 纠正
- 指导

> **注意事项**
>
> AWS用不同的术语提到了这些原则，即纠正和指导。在CAF中，这些随后被称为检测和响应。

为了管理CI/CD流水线中的安全策略，AWS提供了本地解决方案：Amazon CloudWatch Alarms、AWS CloudTrail、Amazon CloudWatch Events、AWS Lambda和AWS Config。下图显示了使用这些解决方案的DevSecOps的CI/CD流水线：

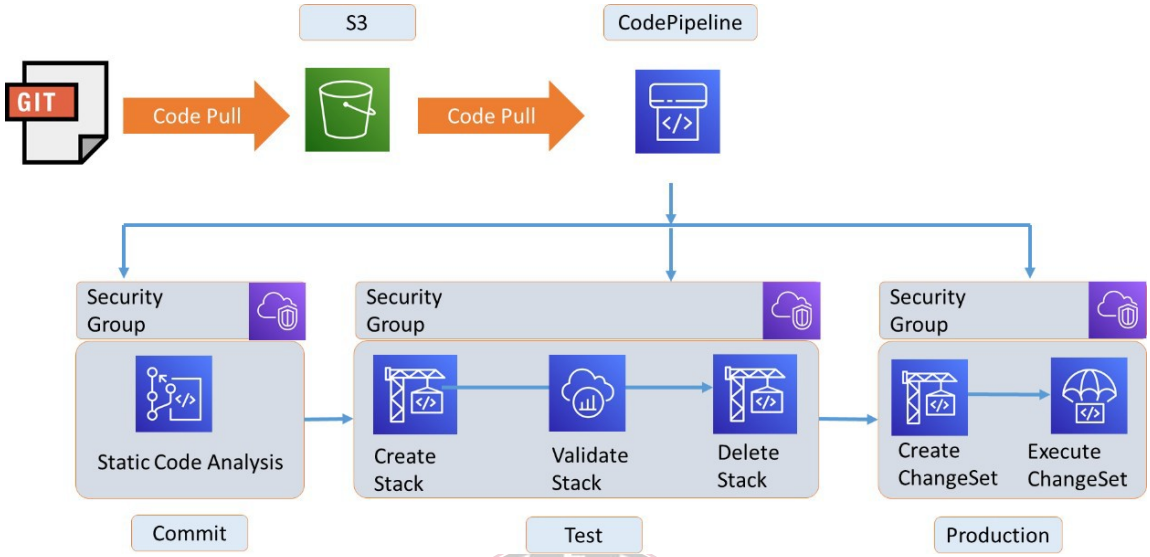


图 12.3-在AWS中使用CodePipeline和安全组

AWS CodePipeline被用来编排流水线中的不同步骤。一个重要的工件是安全组：这些是定义流水线中开发和部署的所有组件的安全策略的容器。它包含必须应用于这些件的模板、护栏和策略。我们可以在流水线中定义三个阶段。

1. **源或提交**。静态代码分析是对从S3桶中拉取的代码进行的。在安全组被破坏的情况下，构建将被停止。
2. **测试**：在这个阶段，CloudFormation被用来创建一个包含AWS中的**虚拟私有云（VPC）**的栈，以运行测试。接下来，AWS Lambda被用来运行堆栈中的代码并验证构建。AWS称此为堆栈验证。Lambda函数将根据安全组验证堆栈。如果检测到漏洞，Lambda函数将删除堆栈并发出错误信息。这是为了防止堆栈和代码进入下一个阶段。

3. **生产**。堆栈验证成功后，会触发一个Lambda函数，使用CloudFormation模板为生产准备堆栈。这一变更集合，用生产模板将测试堆栈转换到生产中，然后被执行。

> **TIPS**
> AWS在https://github.com/awslabs/automating-governance-sample/tree/master/DevSecOps-Blog-Code中提供了CloudFormation模板和流水线的样本。

根据安全组检查的项目的例子可以是验证用户访问和权限、对S3桶的访问控制、以及创建使用实例的策略，例如EC2计算资源。CloudWatch和CloudTrail用于监控组件、访问级别及其使用情况，并将收集流水线中各个步骤执行过程中触发的事件的日志。

使用GitHub和Azure服务实现DevSecOps

微软Azure对DevSecOps采用了不同的方法：它利用GitHub的扫描能力和Azure Kubernetes Services（AKS）的功能，旁边是集成到Azure DevOps和Azure Security Center的Azure Pipelines，用于储存安全策略。下图显示了一个使用GitHub和Azure服务的安全嵌入式CI/CD流水线的高阶架构：

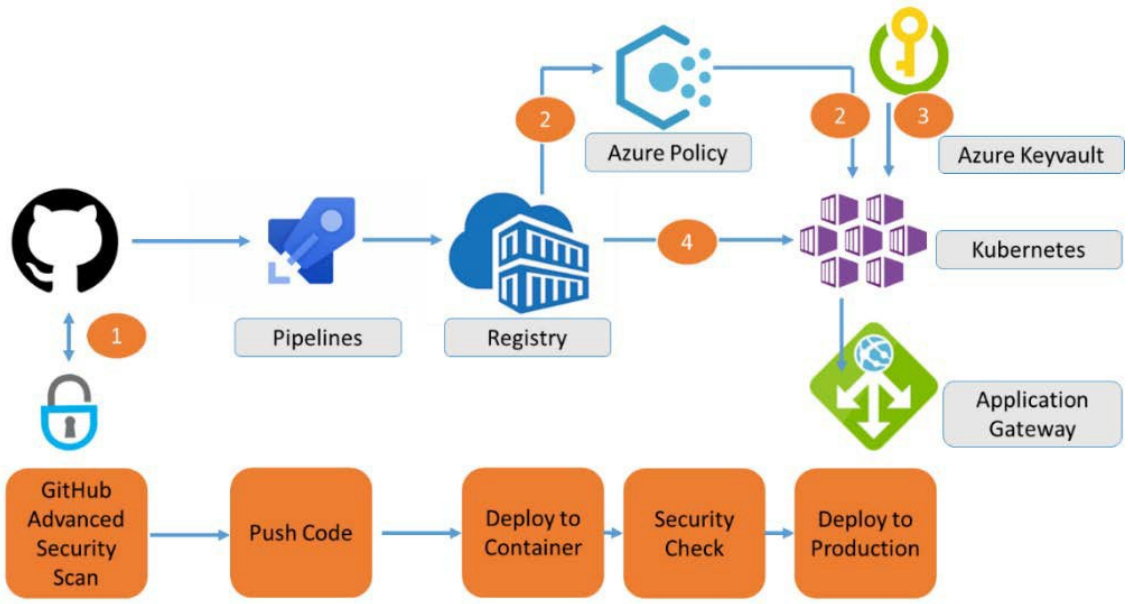


图 12.4-使用GitHub和Azure服务的DevSecOps

前面图表中的数字代表采取的步骤的顺序。一旦容器被推送到Azure容器registry（ACR），就会根据存储在Azure策略中的策略对它们进行扫描。接下来，会获取适当的安全密钥，并向Azure Kubernetes Services（AKS）验证容器。只有当所有的检查都通过后，代码才会被推送到应用网关。

让我们更详细地看一下：

1. **源**：该解决方案从GitHub中的代码分析开始，包括使用CodeQL和Dependabot来分别检测源代码和依赖项中的漏洞。
2. **测试。**一旦代码得到验证，它就会被打包到Docker容器中，并使用Azure Dev Spaces部署到测试环境中。这种编排是通过Azure Pipelines完成的。

Azure Dev Spaces将使用AKS建立一个隔离的测试环境。这与AWS的CloudFormation构建堆栈的方式相当。

3. **扫描**。容器存储在ACR中，在那里根据安全策略对它们进行扫描。为此，Azure使用Azure安全中心，这是一个巨大的库，保存着在Azure注册的环境的所有安全策略。
4. **生产**。使用AKS将扫描的容器推送到Kubernetes集群中。Azure策略用于验证配置集群和容器的合规性。

就像AWS一样，Azure使用几种不同的解决方案来提供一个端到端的解决方案，在整个CI/CD过程中嵌入安全规则和策略。然而，所有这些解决方案都始于一个存储和管理这些安全护栏和准则的仓库：通过AWS Security Hub管理的安全组，或者在Azure中的Azure安全中心。

在谷歌云中使用Anthos和JFrog实施DevSecOps

GCP为使用Anthos和JFrog实现DevSecOps流水线提供了一个有趣的最佳实践方案。它不仅提供了一个云原生流水线，而且还为使用GCP和本地系统的混合环境提供了一个开发和部署的解决方案。

这对企业来说很有吸引力，因为很多企业不会把他们的IT系统完全转移到公共云上。大多数企业预计会将越来越多的系统转移到云端，但他们的一些系统仍将保留在私有环境上。同时满足云和企业内部解决方案的CI/CD流水线是有利的，有了Kubernetes，它们就相对容易建立。

该架构如下图所示：

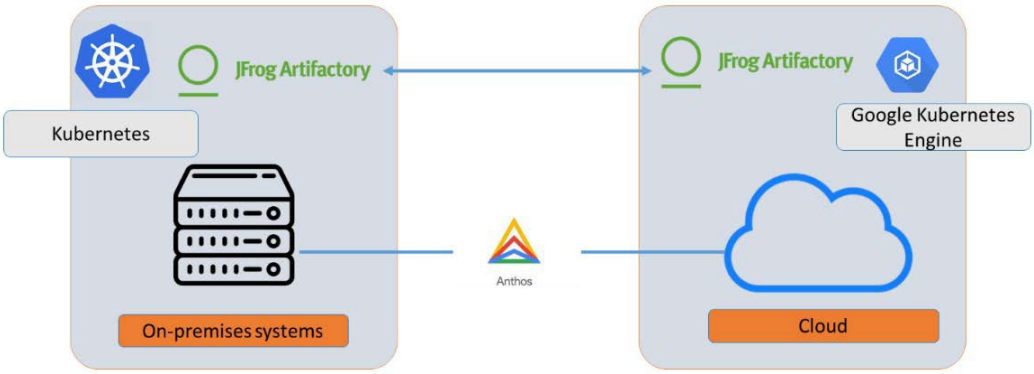


图 12.5-使用JFrog Artifactory和Google Anthos的高层架构

GCP提倡使用JFrog Artifactory和JFrog Xray。

- JFrog Artifactory负责存储构建应用程序时使用的工件。在这一章中，我们看到一个流水线是从源码库中拉取代码开始的。开发人员需要能够全面而安全地依赖于存储和订购工件的工具--代码构建块，以便软件可以自动化交付到流水线。
- JFrog XRay通过Artifactory扫描工件（代码构建块），以发现已知漏洞和许可证合规性。通过扫描源工件，Xray倡导左移思维。

该解决方案如下图所示：

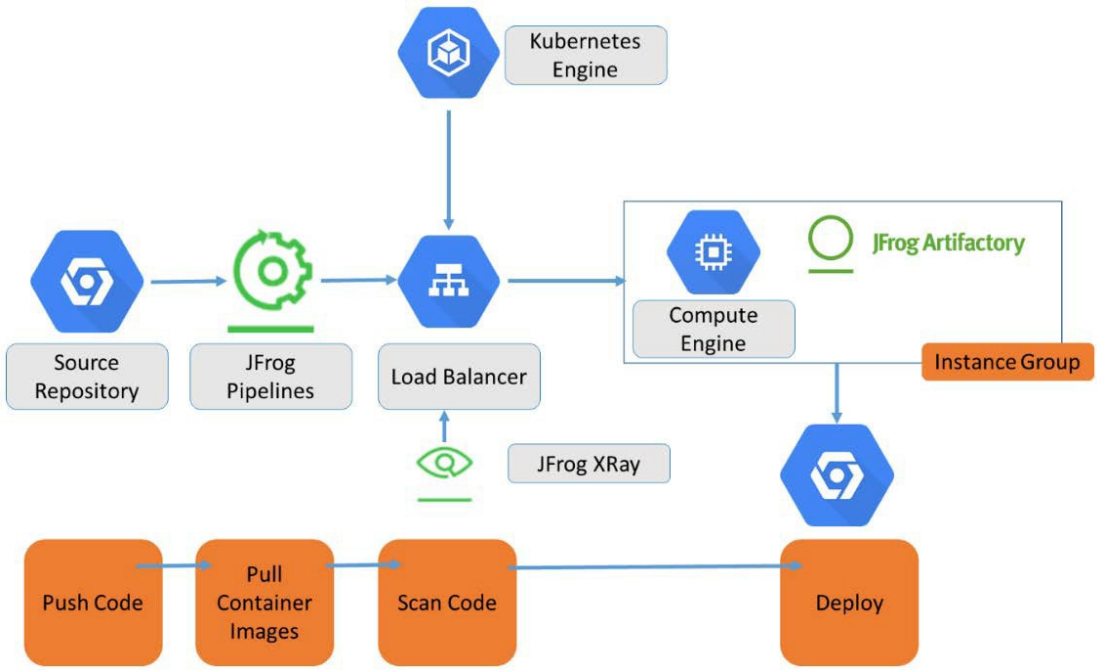


图12.6-使用JFrog XRay的谷歌云的DevSecOps

在这个解决方案中，JFrog XRay是嵌入流水线的安全解决方案。然后，使用GCP中的Kubernetes，并在Anthos上将构建物推送到生产中。然而，Anthos确保了在使用**谷歌Kubernetes引擎**（GKE）的原生云和本地部署及管理Kubernetes集群的一致层。这个解决方案不仅在GCP上可行，而且可以在本地的VMWare堆栈之上使用，也可以在AWS上使用。

规划部署

到目前为止，我们已经讨论了DevSecOps流水线的参考架构以及AWS、Azure和GCP的最佳实践。如果我们有了这个架构，下一步就是计划在企业中部署DevSecOps 和流水线。这就是这最后一节的主题。

企业需要遵循三个主要步骤来实施DevSecOps：

1. **评估企业的安全性**。企业很可能已经采用了安全政策并采取了措施来保护他们的系统。由于政府或行业的规定，他们还需要遵守安全标准和框架。安全专家将进行风险评估并分析可能的威胁。这些专家了解并管理安全控制。默认情况下，这是将安全合并到DevOps实践中的起点。一个非常强烈的建议是，DevOps团队在开发和部署新代码时，不应该一开始就不包括安全政策和标准，甚至在试点项目或POC中也是如此。安全问题必须从第一天起就成为重中之重。
2. **将安全嵌入到DevOps中**。安全政策和标准被集成到开发过程中。DevOps的工作流程与安全准则和护栏相匹配。这包括漏洞测试和代码扫描，我们在本章中广泛讨论过。如果没有适当的流程和工具，DevOps团队就不能开始开发新代码。增加系统的攻击面并最终对企业造成巨大损失的风险太大。无论是大公司还是小公司，都处于黑客和安全威胁的持续威胁之下。这就把我们带到了第三步。
3. **训练，训练，训练**。DevOps和DevSecOps不仅仅是关于技术，它是一种工作方式，甚至是一种思维方式。也许更好的表述是：它是一种文化，人

们需要接受培训来采用这种文化。这种培训不是一次性的。员工、开发人员和运维人员需要不断地、持续地接受培训。开发人员、运维人员和安全工程师需要完全致力于在整个工作中实施安全控制，这意味着他们需要始终意识到企业在安全漏洞和黑客方面所面临的风险。

当然，适当的工具是必不可少的。建议企业至少具有以下工具：

- **测试**。这是DevSecOps的关键因素。市场上提供了大量的工具来进行测试。例如，Chef Inspec、Haikiri和Infer。
- **告警**。当检测到安全威胁时，需要告警并发送出去。Elastalert是一个告警工具的例子。
- **自动修复**。StackStorm等工具可以帮助在发现安全问题时立即提供修复措施。
- **可视化**。开发人员和运维员需要能够看到系统中正在发生的事情。Grafana 和Kibana是流行的工具，有助于可视化和共享安全信息。

这份清单绝非详尽无遗。所提到的工具是第三方工具，它们与AWS、Azure和Google Cloud的DevOps工具和本地工具整合得很好。当然，公有云平台本身也提供广泛的安全工具。例如，Azure中的Sentinel和Azure Security Center，AWS中的Security Hub，以及GCP中的Security Command Center。

读完这一章后，DevSecOps的好处应该很清楚了，但我们要用一个结论来总结：通过DevSecOps，企业可以实现开发人员、运维人员和安全工程师之间更好的合作，并以此来确保在开发的早期阶段发现安全威胁和漏洞，从而将企业的风险

降到最低。

我们将在 *"第14章 DevSecOps与DevOps集成"* 中详细阐述在DevOps中实施安全的问题，在那里我们还将讨论DevSecOps治理。但首先，我们将在下一章中学习如何在DevOps中配合和集成行业安全标准。

摘要

在本章中，我们研究了DevSecOps的不同组成部分。我们了解到，DevSecOps不仅是关于工具和自动化，而且在很大程度上也是关于文化。DevOps团队必须与企业中的安全专家合作，他们必须完全致力于接受并将和安全准则嵌入到开发和部署新的代码中。工具当然可以帮助在DevOps中实现最大的安全性。本章的大部分内容是关于构建DevSecOps实践。

然后，我们讨论了主要公有云供应商的DevSecOps的最佳实践，也就是AWS、Azure和Google Cloud。这些实践通常包括使用Docker容器和Kubernetes作为容器编排平台。我们还学习了在将容器部署到生产平台之前如何扫描代码和保护容器。重要的活动包括静态代码分析和动态扫描。

在本章的最后一节，我们讨论了企业实施DevSecOps实践必须采取的步骤，并提供了一些必要工具的建议。

企业通常必须遵守政府和行业的安全标准和框架。下一章是关于DevSecOps中与这些标准的合作。

问题

1. 软件组成分析（SCA）的功能是什么？
2. 使用什么技术来保持容器的安全？
3. AWS中用于创建堆栈的本地工具是什么？
4. AWS、Azure和GCP公有云供应商提供他们自己的Kubernetes服务来运行容器。他们各自的服务的名字是什么？

进一步阅读

- 关于在DevSecOps中使用AWS CodePipeline的博客： https://aws.amazon.com/blogs/devops/implementing- devsecops-using-aws-codepipeline/#:~:text=%20Implementing%20DevSecOps%20Using%20AWS%20CodePipeline%20%201，%206%20Create%20change%20set%3A.%20%20More%20
- 关于在Azure中应用DevSecOps实践的文档：https://azure.microsoft.com/en-us/solutions/devsecops/
- 关于使用GCP、Anthos和JFrog的DevSecOps CI/CD的文件：https://cloud.google.com/architecture/partners/a-hybrid-cloud- native-devsecops-pipeline with-jfrog-artifactory-and-gke on-prem#best_practices

5. 关于Docker安全的文档：https://docs.docker.com/engine/ security/trust/

13
行业安全框架与DevSecOps

安全以及DevSecOps中的一个重要因素是安全框架。有一些通用的框架，如**互联网安全中心**（CIS），但通常情况下，各行业必须遵从特定的行业安全标准并报告合规情况。这些对企业内部处理安全的方式有影响，因此对DevSecOps的实施也有影响。

本章将解释框架的功能和影响，以及如何将其纳入DevSecOps。本章包括一个关于MITRE ATT&CK框架的使用和价值的单独段落，因为它作为一个基础框架正在变得更加知名和被广泛接受。

完成本章后，你将对最常用的安全框架以及这些框架的控制措施如何应用于DevOps有很好的了解。

在这一章中，我们将涵盖以下主要议题：

- 了解行业安全框架
- 使用MITRE ATT&CK框架
- 将框架应用于DevSecOps
- 创建合规报告并指导审计工作

了解行业安全框架

多年来，IT已经变得更加复杂。这同样适用于IT安全。这两者之间存在着一种关联性。企业的IT环境不再是单一的了。坐落在公司地下室的系统，作为企业的数据中心发挥作用。今天，IT环境共享不同的组件，并通过互联网连接与外部世界相通。在这种观念下，系统默认是可以通过互联网访问的。然而，只有经过授权的用户才能访问这些系统。因此，我们需要一些强有力的防御措施来保护系统免受安全漏洞的影响。

每个行业所需的安全水平会有所不同。首先，金融机构要确保银行账户不能被泄露，资金不被非法转移。医疗机构需要保护他们病人的个人和健康数据。制造商希望保护他们的知识产权和专利。最重要的是，在安全方面有几个总体原则，保护数据、身份和加固系统免受外部攻击。追踪所有这些几乎是不可能的，这就是安全框架的作用：它们为在企业中实施正确的一套安全政策提供指导。

在我们了解安全框架如何影响CI/CD和DevOps之前，我们需要了解这些框架

是什么。简而言之，一个框架是一套政策和关于实施和管理这些政策的文件化指南。这些政策本身的重点是识别风险、减轻风险、减少系统和程序的攻击面，以防止发现漏洞。这是一种通用的方法，但行业框架根据行业的具体情况需要对这种方法进行调整。

通用的IT安全框架包括ISO IEC 27001/ISO 2700212，**美国国家标准技术研究所（NIST）**网络安全框架、**互联网安全中心（CIS）**和**信息及相关技术控制目标（COBIT）**。让我们先更详细地探讨一下：

- **ISO IEC 27001/ISO 2700212/27017。**ISO 27001正被设定为系统安全控制的国际标准。重点是检测将对系统的可用性和完整性产生严重影响的威胁的控制。ISO 27002规定了管理控制措施本身的额外标准，如用户访问管理和维护资产清单。ISO 27017特别针对云计算。例如，它涉及到平台即服务（PaaS）和软件即服务（SaaS）环境中的共同责任，确保部署和删除云系统的安全，并监控云服务。
- **NIST。**NIST网络安全框架没有规定控制措施，但它确实提供了五个功能来加强安全：识别、保护、检测、响应和恢复。这些功能允许组织设置控制措施，以管理数据泄露风险。控制措施必须包括访问控制、保护数据的措施，还有工作人员的意识。响应控制的要求必须描述组织应该如何对威胁和攻击做出反应，包括缓解和沟通准则。恢复是最后的手段：组织需要有一个明确的战略，关于如何从攻击中恢复，如系统和数据恢复。NIST的五个领域在下图中显示：

图13.1-NIST的网络安全框架

- CIS：CIS提供广泛的框架，对平台、操作系统、数据库和容器进行具体控制。一些CIS框架被嵌入到平台中，如CIS for Azure和AWS。在这些情况下，可以从Azure Security Center和AWS Security Hub访问CIS基准。CIS基准确保使用的组件是经过加固的。与NIST相比，最大的区别是NIST专注于评估风险的准则， 而CIS提供了一长串的安全控制和最佳实践。
- COBIT：COBIT是由**信息系统审计与控制协会（ISACA）**推出的，该协会是IT安全和审计的国际组织。最初，COBIT是关于识别和减轻IT系统的技术风险，但随着最近COBIT 5框架的发布，它也涵盖了与IT有关的商业风险。COBIT的实施和管理很复杂，因为它涵盖了整个企业，包括所有的IT管理流程，如事件、问题、配置和变更管理。

这些都是控制框架。所有这些框架都可能有涵盖特定行业要求的版本，但通常情况下，各行业必须遵守自己的标准，同时也要完全合规。这在行业被审计时是很 重要的。在本章的最后一节，我们将更详细地讨论审计问题。

主要的行业框架（实际上，这些是合规证明）是针对医疗保健的**《健康保险可携性和责任法案》（HIPAA）**，针对美国政府组织的**《联邦风险和授权管理计划》（FedRAMP）**，欧盟的**《通用数据保护条例》（GDPR）**，以及针对金融机构的**《支付卡行业数据安全标准》（PCI-DSS）**。

所有这些都是全球性的或至少是区域性的实施，但也可能有公司需要适用的特定国家安全法规。一个例子是**纽约金融服务部（NYDFS）**的网络安全条例。这个框架于2017年发布，对美国的所有金融机构制定了安全法规。然而，该框架中的规则与NIST保持一致，并适用ISO 27001标准。然而， NYDSF确实有一些规则取代了这些通用框架。根据NYDSF，数据加密和增强的多因素认证是所有入站连接的强制性安全控制。

有一个框架我们还没有讨论，那就是MITRE ATT&CK。MITRE ATT&CK不是一个真正的框架，比如我们在本节中讨论的那些框架。它是一个知识库，涵盖了关于系统可能被攻击和破坏的战术。然而，它可以作为输入来定义风险策略和威胁模型以保护系统。在下一节，我们将学习如何使用MITRE ATT&CK。

MITRE ATT&CK框架

也许这不是一个完全公平的说法，但无论如何我们都会在这里发布。MITRE ATT&CK让你在安全问题上从攻击者的角度思考。这个框架的优势在于，任何人都可以为它做出贡献。它并不真正描述系统中的实际漏洞，而是描述攻击者可以用来利用这些漏洞的技术。MITRE ATT&CK使用一个带有14种攻击战术的矩阵。

接下来，它将这些战术划分为主要的平台或技术，包括云和容器。在云方面，有一个针对Azure、AWS和GCP的细分。

TIPS

完整的MITRE ATT&CK框架可以在https://attack. mitre.org/上找到。不过，建议在Twitter上也关注MITRE，网址是@MITREattack。该矩阵是开源的，因此一个活跃的社区正在为框架中收集的战术和技术做出贡献。MITRE邀请人们加入这个社区，并积极贡献他们的发现。

在本节中，我们将简要介绍这14种战术，然后专门讨论针对容器的战术，因为这些战术被广泛用于**持续集成/持续部署**（CI/CD）。

- **侦查**。这些技术收集尽可能多的信息以准备攻击。这包括扫描系统，但也包括社会工程，即利用组织的工作人员来获取信息。
- **资源开发**。这些技术涉及创建、购买或窃取黑客可以用来执行攻击的资源。
- **初始访问**。这是对系统进行访问的第一次尝试。这包括滥用有效（服务）账户和网络钓鱼。
- **执行**。这种技术是指运行恶意代码。
- **持久化**。这种技术利用后门获得访问权。它使用容器来注入恶意代码、启动，并登录初始化脚本。
- **权限提升**。这种技术涉及使用漏洞来利用系统上的特权，最终获得更多的控制权。当涉及到使用容器时，这是一种常用的策略。容器应该始终被加固，以防止它们获得额外的权限。
- **防御绕过**。这涉及到代码被用来绕过入侵检测、记录和其他预防措施的战术。

在云环境中，这种战术被用来操纵云（编码）防火墙，例如，通过不同地区的未使用环境或未受保护的沙盒环境进入。

- **凭证访问**。通常情况下，这涉及到暴力攻击以获得用户名和密码。
- **发现**。这种策略用于寻找用户数据、设备、应用程序、数据和服务，以获得尽可能多的关于可用系统的信息。
- **横向移动**。这种策略被用来将系统和数据从一台主机转移到另一台，有时是在一个不受企业控制的不同环境中。哈希传递攻击和远程管理访问是常用的技术。
- **收集**。这种策略被用来收集键盘击打或屏幕捕捉的数据。在云中，收集API密钥以访问存储和密钥库是流行的技术。
- **命令和控制**。当使用这种技术时，黑客试图与系统沟通，试图获得对它们的控制。
- **数据窃取**。这种策略涉及获得对数据和数据流的控制，例如，将数据发送到不同的存储环境并对数据进行加密。
- **危害**。这是一个广泛的类别，包括拒绝服务技术和资源劫持。

MITRE ATT&CK不是一个魔杖：它不能解决所有的安全问题。它应该被视为另一个来源，你可以通过提供不同的洞察力来开始以更好的方式保护IT环境。它显示了潜在的攻击模式和路径，安全工程师可以将其纳入安全策略。MITRE ATT&CK 提供了来自特定平台和技术的洞察力，这使得它相当独特。常见的攻击策略在不同的平台

上可能有不同的路径和模式，而这正是MITRE ATT&CK的一个很好的护栏。

注意事项

在DevOps中，扫描代码是至关重要的。静态和动态扫描必须是CI/CD的默认做法。扫描通常是根据基线进行的。在DevOps中常用的一个基线OWASP，即开放式web应用程序安全项目。OWASP是开源的，每年都会列出应用程序中的十大漏洞。我们将在"*第14章 DevSecOps与DevOps集成*"中更详细地讨论。

在下一节，我们将向你展示如何利用矩阵来更好地保护容器。

容器中使用MITRE ATT&CK战术

那么，MITRE ATT&CK在实践中是如何工作的？让我们以容器的矩阵为例，因为它们在CI/CD流水线中经常被使用。首先，我们必须去看关于https://attack.mitre.org/matrices/enterprise/containers/的具体矩阵。你会认识到我们在上一节中讨论的14种战术中的一些。并非所有的都适用；对于容器来说，有八个战术被证明是相关的。

- 初始访问
- 执行
- 持续化
- 权限提升
- 防御绕过

- **凭证访问**
- **发现**
- **危害**

接下来，我们可以看看这些战术中的一种。我们将以 "**执行**"为例，如以下截图所示：

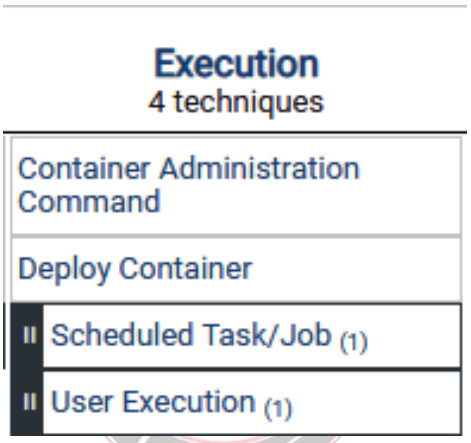


图13.2-MITRE ATT&CK容器的标签

执行排名前两位的如下：

- **容器管理命令**
- **部署容器**

如果我们点击第一个，**容器管理命令**，矩阵为我们提供了关于容器管理中的特定漏洞是如何被利用的信息。漏洞本身可能发生在使用Docker守护程序或Kubernetes API管理容器的时候。这些可能允许在容器启动时使用远程访问和管理。MITRE给出了两个被用于此技术的例子。第一种技术是Hildegard，它使得使用kubelet API运行命令

在运行的容器上执行命令成为可能。Kinsing是MITRE提到的第二种技术，它利用Ubuntu的一个入口点来运行shell脚本来接管容器管理进程。

接下来，矩阵提供了缓解措施。在给定的例子中，缓解动作是使用只读容器和通过SSH远程访问限制与容器服务、守护程序或Kubernetes的通信。

你会发现Kinsing也出现在部署容器标签下，旁边是Doki漏洞，这是一个在2020年春天发现的恶意软件，目标是Docker容器。

该矩阵将引导你了解各种漏洞，并帮助你缓解这些漏洞。

至此，我们已经讨论了各种框架的内容。在下一节中，我们将学习如何在DevSecOps中使用它们，以及如何创建合规报告以显示框架已被应用。

将框架应用于DevSecOps

在本节中，我们将学习如何在DevOps中包含框架的控制措施，并将其作为DevSecOps嵌入。好消息是：这并不像听起来那么难。下图显示了这个过程：

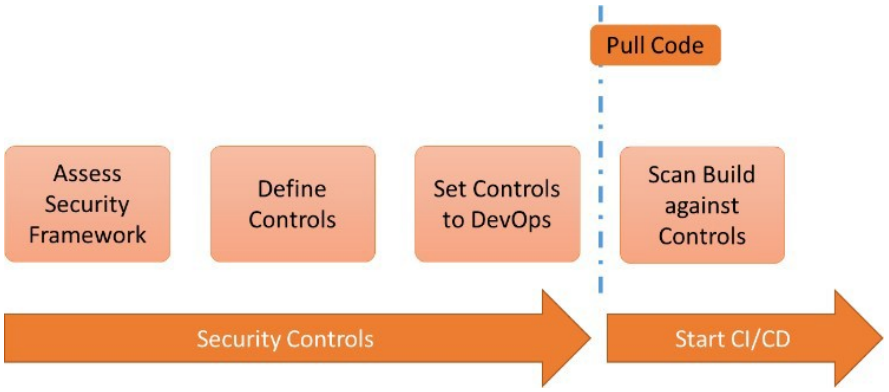


图13.3-将安全框架的控制措施应用于DevOps的过程

一般来说，我们首先评估企业需要应用于其IT环境的框架。从该评估中得出不同的控制措施，并将其设置到应用程序和基础设施的开发和部署周期中。一旦代码从仓库中取出，就开始针对这些控制措施进行扫描。

我们在这里使用CIS基准作为例子，因为CIS是设置安全控制措施的最常用框架。应用控制措施首先要认识到，在DevOps中，IT环境默认是高度动态的。包括基础设施在内的一切都变成了代码，因此应用程序将在容器或无服务器模式下运行。这就需要一些特定的控制措施。

必须采用一些通用控制措施。这些包括以下内容：

- **漏洞管理**：这必须在代码被推送到生产之前作为一项控制措施来实施，但考虑到左移的原则，漏洞扫描应该从代码被拉出库的那一刻就开始。
- **访问控制**：通过这个控制措施，你可以限制和管理所有资源的权限，包括容器。
- **日志**：这包括构建和测试代码时的日志。有时，只收集生产环境中的日志，但如果你想控制DevOps生命周期，这还不够。

这些都是通用的。CIS已经开发了一个特定的框架来确保容器的安全，如下图所示：

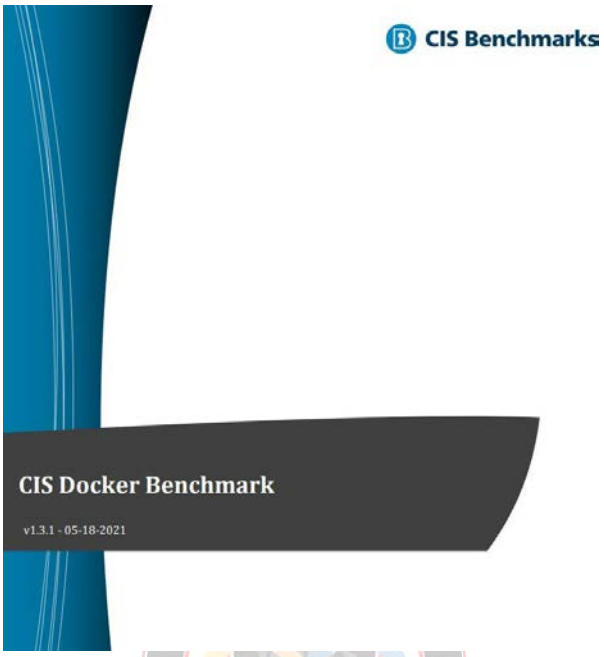


图 13.4-CIS Docker基准测试

CIS所说的这一基准，包含了以下方面的控制措施：

- **Linux主机配置**；例如，确保容器有一个单独的分区，并确保只有受信任的用户可以控制Docker守护程序。
- **Docker守护进程配置**；例如，如果可能的话，以非root用户身份运行守护进程，并确保容器在获得新权限时受到限制。
- **容器镜像和构建文件配置**；例如，确保容器只使用受信任的基础镜像。
- **容器运行时配置**；例如，Linux内核能力在容器中受到限制，确保特权端口在容器中不被映射。

CIS也有一些通用的建议，比如确保容器是加固的，Docker版本是最新的。

> TIPS
>
> 所有CIS基准可在https://www. cisecurity.org/免费下载。

该基准不仅告诉你应该有哪些控制，而且还给出了如何实施这些控制的建议，以及为什么应该有这些控制的理由。那么，请看这个例子，CIS建议有容器的独立分区（在CIS v1.3.1（2021）for Docker 的control 1.1.1）。

这要从配置文件的适用性说起。在control 1.1.1中，这项被设置为1级-Linux主机。这意 味着这些设置只适用于Linux主机，在不妨碍组件（在这种情况下，Linux主机）的预期功能情况下，提供了明显的安全收益。

接下来，描述了control本身，以及其背后的原理。在这个例子中，它描述了Docker如何默认使用/var/lib/docker来存储它的所有组件。该目录是与Linux主机共享的，这意味着它很容易被完全填满，使Docker和主机都无法使用。因此，建议使用一个单独的分区。最后，CIS提供了一份手册，说明如何通过为/var/lib/docker挂载点创建一个单独的分区来做到这一点。

你必须遵从所有这些建议吗？CIS已经明确区分了关键控制和重要控制。显然，关键的应该在所有情况下实施，无论在你的DevOps实践中实施它们是否有意义，你都需要评估每一个控制。这里的黄金法则是，如果你实施了一个控制措施，你需要持续地遵守它，并报告它的合规性。企业将就他们实施的规则和政策接受审计。在本章的最后一节，我们将讨论报告和审计。

创建合规报告并指导审计工作

DevOps正在企业中大行其道。将安全嵌入到DevOps中是合乎逻辑的下一步。但是，企业如何才能确保他们的DevOps和DevSecOps符合我们在本章中讨论的框架？这个问题的答案是：通过审计。IT系统会被定期审计，DevOps实践也应该如此。尽管如此，审计DevOps仍然是一个未知的领域，尽管毕马威和德勤等主要会计师事务所已经发布了关于这个问题的白皮书。

DevOps审计应至少包括以下方面：

- **评估DevSecOps战略**。战略是否清晰？治理如何安排？DevOps战略可以按业务单元设定，也可以在企业范围内设定。两者都可以，只要该战略被持续贯彻执行。目标应该是明确的，并被每个团队所采纳。这同样适用于团队中所有学科的工作方式。测试程序和验收标准等流程必须是透明的，并且毫无例外地被遵守。
- **评估DevSecOps培训的水平**。培训不仅仅是简单地创建一个关于**规模化敏捷框架（SAFe）**的单页幻灯片，并展示DevOps的生命周期。DevOps在很大程度上与文化有关，但有时，组织只是被一种突然出现的新工作方式所淹没。例如，实施DevOps也意味着创建具有正确技能的团队。这需要组织起来，而且要比在组织中发布**Spotify模式**更进一步。员工不会因为告诉他们必须这样做就把自己组织成行会和小队。一个企业需要对员工进行DevOps的培训，并确保团队拥有正确的技能组合。培训还包括管理组织。

注意事项

Spotify模式已经在企业中大受欢迎，成为扩展敏捷工作方式的一种方法，而DevOps是其中的一部分。Spotify模式是以音频流媒体服务所实施的敏捷工作方式命名的，它提倡团队的自主性，组织成小队。每个小组都被允许选择自己的工具集和敏捷框架，如Scrum或Kanban。

- **审查DevSecOps工具链**。是否有一个指定DevOps工具的架构，并且它是连贯的？它是否为战略服务，是否与企业的IT战略保持一致？例如，如果企业有一个开源战略，那么这些工具就必须遵守这个战略。最后，就像企业中使用的任何工具一样，它需要被置于架构变更控制之下。
- **审查DevSecOps流程**。DevOps并不意味着流程不再有效。企业仍然需要有基本的IT流程，如事件管理、问题管理和变更管理。这些流程必须被记录下来，包括其升级级别。另外，在实施DevSecOps时，必须对这些流程中的角色进行明确的描述，并加以贯彻。安全管理在这里占据了一个特殊的位置，它必须描述如何定义安全策略，如何在企业中实施和管理这些策略，以及如何将其嵌入到DevOps流程中。

就这样，我们已经研究了DevOps中安全的基本原则和安全的行业框架。现在，我们需要将安全与我们的DevOps实践相融合（或者说集成）。这就是 *"第14章 DevSecOps与DevOps集成"* 的主题。

摘要

在本章中，我们讨论了各种安全框架。这些框架是为企业的IT环境设置安全控制措施的准则。这些控制措施适用于系统和应用程序，同时也适用于DevOps实践。从开发人员从仓库中拉取代码并开始构建，直到部署和生产，IT环境，包括CI/CD流水线，都需要遵从安全控制措施。有很多不同的框架。其中一些被企业广泛接受，如NIST、CIS和COBIT。

我们还讨论了MITRE ATT&CK框架，从不同的角度出发，与其他安全控制框架进行了比较。MITRE ATT&CK列出了黑客可能使用或已经使用的漏洞利用的战术和技术。就像CIS一样，MITRE ATT&CK列出了各种平台和技术的具体情况，包括CI/CD中常用的容器。

在上一节中，我们研究了对DevSecOps的审计。建议审查工具、流程和DevOps团队的技能的一致使用等主题。

在下一章中，我们将把安全实践整合到DevOps中，并了解企业如何采用真正的DevSecOps战略。

问题

1. 什么ISO标准是专门针对云计算的？
2. MITRE ATT&CK在执行战术下为容器提到了哪两种技术？
3. 真或假：CIS没有提到Docker的版本控制。

进一步阅读

- 毕马威，2020年1月： https://advisory.kpmg.us/articles/2020/ role-of-internal-audit-devops.html
- 发现Spotify模式，Mark Cruth在Atlassian上的博文：https://www.atlassian.com/agile/agile-at-scale/spotify#:~:text=It%20is%20now%20known%20as%20the%20Spotify%20model.,by%20focusing%20on%20autonomy%2C%20communication%2C%20accountability%2C%20and%20quality
- CIS的网站：https://www.cisecurity.org/
- 可以找到COBIT框架5的ISACA网站：https://www. isaca.org/
- NIST的网站: https://www.nist.gov/
- 国际标准化组织的网站：https://www.iso.org/standards.html

14
DevSecOps与DevOps集成

本章的标题可能听起来有点奇怪，但DevSecOps和DevOps并不是独立的。它应该是一种工作方式：安全应该与DevOps实践相结合，而不是在DevOps之上添加安全原则。这意味着架构师必须定义一个总体的治理模式，将威胁建模整合到DevOps中，并以集成工具集为目标。最后，监控集成需要覆盖DevSecOps生命周期的每个方面。我们将了解到，监控集成接近于我们在本书前面讨论的东西。AIOps。在这一章中，我们将把所有的东西拉到一起。

完成本章后，你将学会如何实施治理，了解威胁建模，并理解它在**安全软件开发生命周期**（SDLC）中的重要性。你还将了解到安全是如何嵌入到持续集成中的，以及如何对其进行监控，并了解这一领域的一些主要工具。

在这一章中，我们将涵盖以下主要议题：

- 在DevSecOps中定义治理
- 了解并使用威胁建模

- 工具集成和自动化
- 实施监控

在DevSecOps中定义治理

到目前为止，我们已经起草了一个DevSecOps架构，确定了流程，然后将这些与企业的业务目标对齐。下一步是管理这一切，这就是治理的主题。DevSecOps不仅仅是一个PowerPoint演示文稿和显示CI/CD流水线的Visio图。一个企业需要熟练的工作人员与之合作，并需要一个描述安全的数字运营模式的治理模型。在本节中，我们将以The Open Group的IT4IT框架作为最佳实践来讨论这个问题。

在"第6章 定义运维架构"中，我们介绍了产品的价值流，并描述了IT如何创造价值。该模型可以在下图中看到。

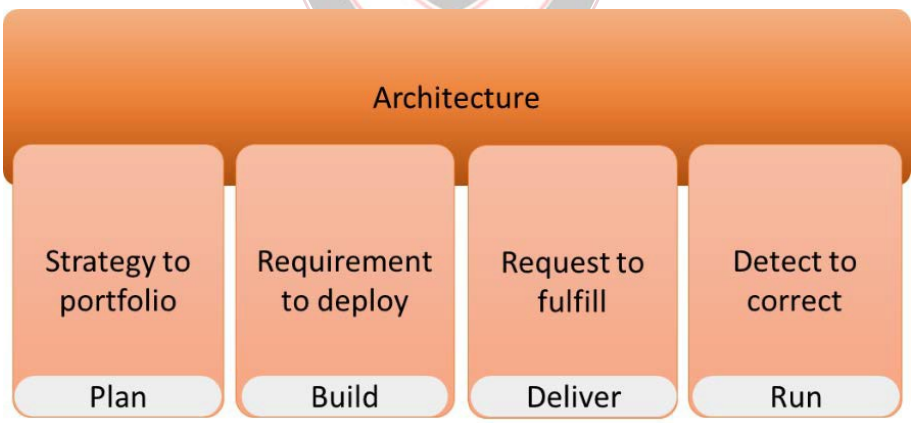


图14. 1-IT4IT价值流

在IT4IT中，**治理、风险和合规（GRC）**是四个价值流的支持活动。这意味着GRC

被完全嵌入到每个价值流中。GRC是做什么的？

简单地说，GRC就是通过应用商定和接受的行业商业政策，在管理不确定性的同时实现目标。每个企业都需要遵守某些政策。这些政策可以是国际贸易法规、国家法律，也可以是行业特定标准。这包括安全和数据隐私法规。实施GRC不是一次性的工作：企业将需要定期调整。这就是GRC能力模型的作用。这个模型如下图所示：

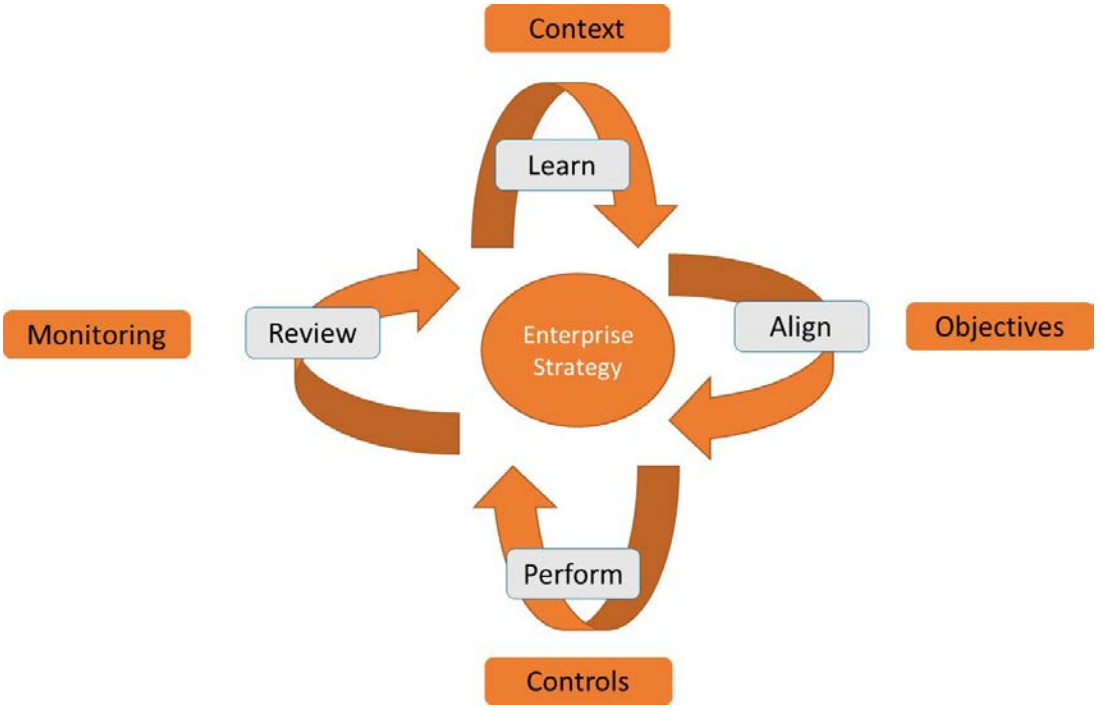


图 14.2-治理、风险和合规的模型

这个模型由四个要素组成：

- **学习**。企业的目标必须明确界定，但这些目标是在企业运作的背景下确定的。例如，医院的目标是让人们变得更好；背景则更广泛，包括例如，与

制药公司和医疗保险公司的关系。目标和背景决定了一个公司的战略。

- **统一**：企业的目标和危及企业战略的潜在威胁被评估。这带来了需求和应对措施，使企业能够实现其目标，并在减轻威胁的同时抓住新的机会。
- **执行**。这是检测不希望发生的事件并采取行动的阶段。
- **审查**。企业需要不断审查威胁是否被及时发现，是否被适当地评级，以及减轻威胁的行动是否成功。这可以反馈到学习中，因为环境可能会改变，战略也需要调整。

实施和管理这些能力需要治理。企业将不得不指派人员来控制这些能力，以及企业中不同层次对这些能力的采用程度。这包括将GRC能力整合到DevSecOps中。我们如何安排一个最佳实践的治理模式？

我们将需要研究企业中的不同层次，而这正是IT4IT所做的。下图显示了该模型的高层原则：

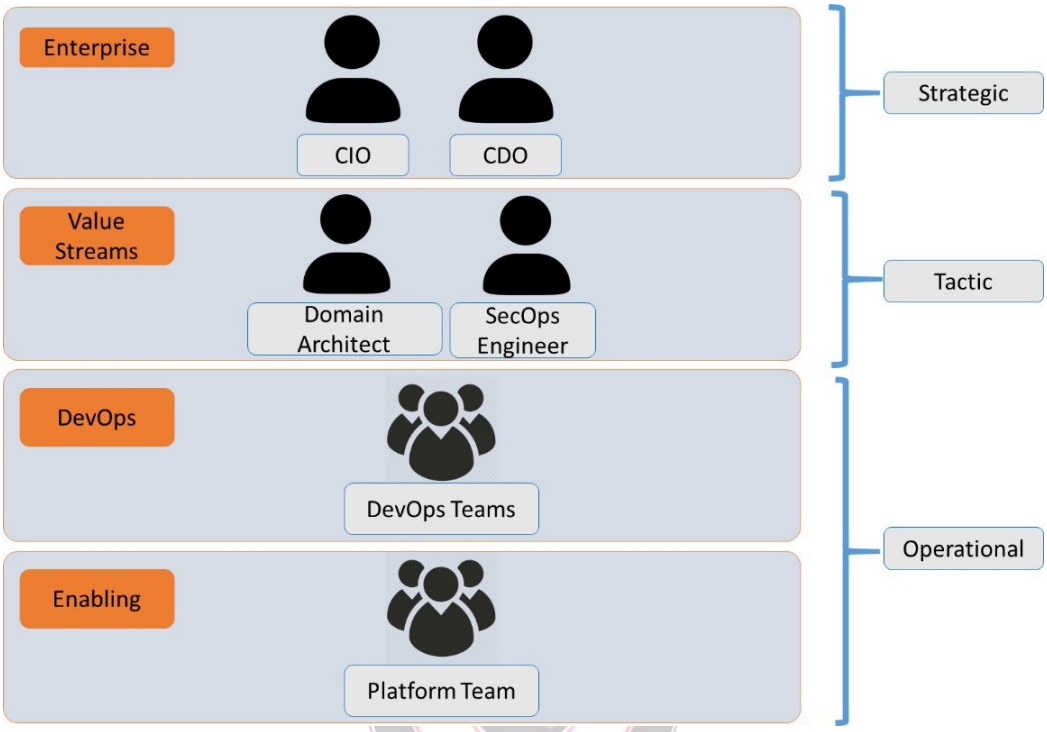


图14. 3-企业中不同安全治理水平的层级

让我们更详细地看一下这些层级：

- **企业级**。企业级的关键角色是企业架构师，他负责企业内的架构。在这个层面上，企业架构师将与**首席信息官**（CIO）一起工作。在现代企业中，我们看到首席信息官的角色在变化，其他角色也在增加。**首席数字官**（CDO）和首席数据或首席隐私官越来越多地出现在组织架构图上。CDO是实施数字化转型战略的一个重要职位，包括采用DevSecOps。请注意，DevSecOps本身绝不是一个战略目标：它是一种指导数字化转型的方法论。

- **价值流**。企业层面是战略层面；价值流是战术层面。在这个层面上的关键角色是领域架构师和SecOps工程师。他们需要监督总体的DevOps架构和安全在各个DevOps团队中的实施。
- **DevOps团队**。DevOps的黄金法则是：你建立它，你运行它，你破坏它， 你修复它。这并不意味着DevOps中没有其他的规则（或者更好的是指南）。一个企业在DevOps中寻求一致性，跨越所有不同的团队。这些团队将自主地运作，但他们仍然需要遵守在企业层面制定的、在价值流层面实施和管理的集中指南。为什么这很重要？一个企业的服务或产品很可能是由不同的DevOps团队负责的构件构建的。如果工作方式、指南和安全护栏不一致，最终产品或服务很可能不符合企业的质量标准。
- **赋能团队**。这些是负责基础工作的团队；例如，托管平台和用于存储工件的仓库。DevOps工程师和开发人员将不需要担心平台上的网络设置，这些都是由赋能团队安排的。

注意事项

这是对完整模型的一个介绍。请参考企业架构师Rob Akershoek（ @robakershoek）的工作，他对IT4IT框架做出了广泛的贡献。他的一本书列在进一步阅读部分。

这一节是关于治理和控制DevSecOps的过程。在下一节，我们将讨论安全团队和专家如何识别可能影响代码开发和部署的事件。对威胁建模的充分理解和了解是一项要求。

了解并使用威胁建模

在上一节中，我们讨论了企业中的安全治理以及如何作为DevSecOps进行集成。在本节中，我们将学习安全问题如何影响SDLC。谈到在DevOps中集成安全，你需要对威胁建模有一个很好的理解，它为我们提供了安全威胁可能影响软件代码的开发和部署的信息。我们先看一下**开放式web应用程序安全项目（OWASP）**的定义，来解释什么是威胁建模。OWASP是一个在线社区，提供对安全威胁、工具和技术的洞察力。

从本质上讲，威胁建模显示了安全威胁如何影响应用程序的完整性。该模型集合和分析安全数据，并帮助做出如何保护应用程序的决定，从而通过评估需求、重新审视设计和实施改进的安全策略，提高代码和托管环境的安全性。

什么是威胁？这是任何对应用程序产生负面影响并导致故障或不必要的事件，如数据泄漏。威胁建模确定了可能的漏洞，然后定义了缓解的行动。对于识别，它可以使用
MITRE ATT&CK框架，我们在上一章讨论过。威胁建模不仅仅是检测安全问题，甚至是防止或修复这些问题。

建模是一种结构化的、有计划的、重复的活动，用来持续评估环境和可能的漏洞，然后帮助实施结构化的方法来减轻这些漏洞。因此，威胁建模是需要在整个SDL过程中进行的，在这个过程中，模型和由模型触发的后续行动会被持续审查和完善。其原因是，在开发过程中，新的功能被添加，甚至可能是新的技术。有了这些，就有可能引入新的漏洞（威胁），所以模型需要不断修订。

威胁建模通常包括以下步骤：

1. **范围分析**。我们的威胁建模和分析的范围是什么？想想应用程序代码、编程语言（例如C#、Python或Java）、API、虚拟机、平台（想想AWS或Azure等云平台）、数据库引擎（例如SQL、Postgres或Cassandra）和数据库。
2. **识别威胁代理**。在MITRE ATT&CK中，我们专注于漏洞和这些漏洞被利用的技术。在威胁建模中，我们还确定谁可能对攻击感兴趣。OWASP称这些为威胁代理：这些可以是内部和外部的。之所以做出这种明确的区分，是为了确定环境对内部人员来说是容错的，还是容易从外部突破。
3. **评估缓解措施**。OWASP将此称为对策。已知的缓解措施，例如加入补丁的可能性，必须包括在模型中。
4. **评估漏洞**。如果你知道范围、可能的攻击者，并且已经确定了可能的对策，那么我们就可以开始分析漏洞。这些是否在范围内，它们能被谁利用， 以及可以采取什么具体的缓解措施来防止或尽量减少影响？
5. **风险优先级排序**。分析漏洞被利用的可能性，以及实际损害会是什么。这就确定了威胁的优先级。
6. **执行缓解措施**。根据优先次序，必须在行动中设置缓解行动，以减少已经确定的风险。

> TIPS
>
> OWASP社区为威胁建模管理着自己的GitHub页面。这些页面可以在https://github.com/OWASP/www-community/blob/master/pages/Threat_Modeling.md找到。

有更多的模型定义了威胁建模，但在本质上，它们都归结为相同的原则：评估、识别威胁、评估威胁，并确定降低风险的行动。

我们有治理下的安全，我们知道如何在可能的威胁方面评估我们的环境。下一步是将其自动化并集成到DevOps工具中。如果我们做到了这一点，那么我们就可以说，我们已经实施了DevSecOps。

工具集成和自动化

在本书中，我们已经讨论过几次测试的重要性。DevOps提倡在生命周期的每一个阶段进行测试，从开发到部署。这包括安全测试。但是，我们如何才能实现这种持续集成呢？我们的目标是让测试在开发人员check-in时运行，当他们从仓库拉取代码时，在构建过程中，以及在实际部署过程中，包括模拟环境。

让我们先来看看**持续集成（CI）**。开发人员会经常对代码进行检查；在某些情况下，这可能是每天的几次构建。这就是CI和DevOps中的敏捷工作方式的目的：开发人员不再在巨大的程序上工作；相反，他们应用小的代码迭代构建，一次增加一个功能。这样一来，就更容易跟踪代码的变化，重要的是，如果增加的功能导致失败，就可以回滚。

CI是将这些变更和添加的内容集成到代码中。开发者check in，修改代码，并将其发布到特性分支。从那里，它被推送到生产中，并返回到仓库，在那里，源代码被更新为新的小块代码，所有这些都在很短的时间内完成。由于这些都是小规模的代码迭代，在生产中集成和实施这些代码要比在大版本中这样做更容易。然而，这只有在代码被沿途测试的情况下才能很好地发挥作用。所有的测试都必须应用于每一个新的构

建和对构建的每一次修改。

在DevOps流水线中，我们以自动化的方式运行这些测试。这需要集成的工具来打包代码，运行脚本来启动适当的基础设施，应用安全策略，启动监控，并执行测试。

总之，测试是贯穿整个开发和部署生命周期的。下图显示了我们的意思，同时也应用了左移的原则：

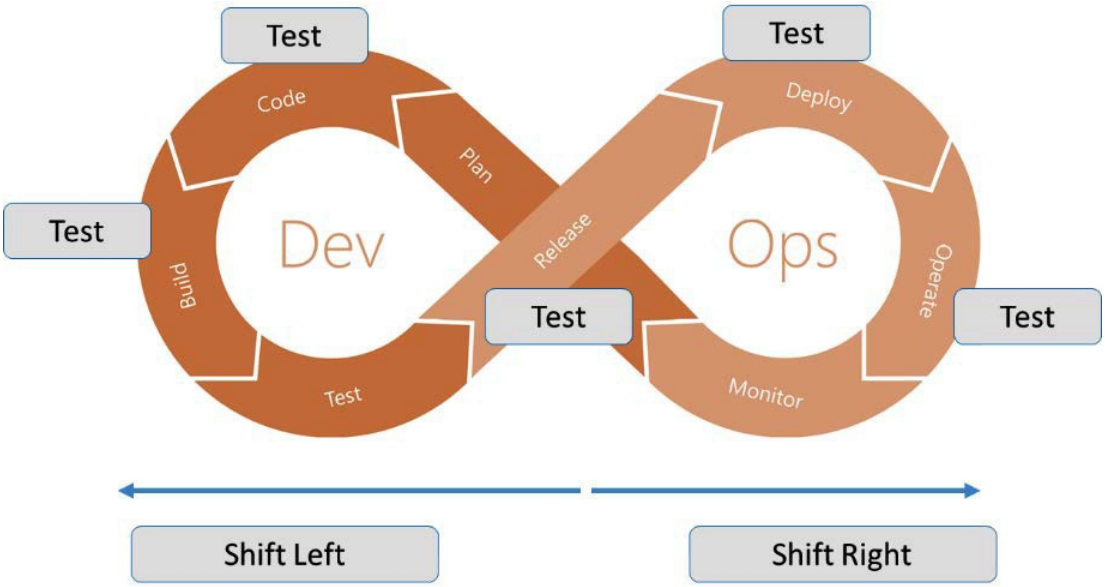


图 14.4-在安全测试中应用左移

这包括安全测试。毕竟，目标是将其与DevOps工具集成，并使CI/CD流水线自动化。我们在以下实例中运行安全测试：

- **代码库**。签入和拉取源代码
- **构建**。编写、修改和编译代码

- **预发布或模拟**。在实际部署到生产和主分支之前进行类似生产的测试

在 "*第12章 DevSecOps架构*" 中，我们讨论了可以应用于流水线的各种扫描类型，如**静态应用安全测试**（SAST）和**动态应用安全测试**（DAST）。

所有这些测试都可以而且应该被自动化。对于很多企业来说，这是一个真正的范式转变。许多安全工程师和经理坚持手工测试，而不是将安全测试自动化。问题是，安全测试会成为DevOps的一个瓶颈：部署被停止了，或者就是被安全测试拖慢了。这就是为什么我们要将安全测试整合到流水线中，并在SDLC的每个阶段实现自动化的原因。

接下来，通过自动化和完全集成的安全测试，我们确保开发人员收到关于代码中漏洞的即时反馈。这将明确地改进代码。

静态应用安全测试集成

SAST是至关重要的。我们在 "*第12章 DevSecOps架构*" 中简要地讨论了这个问 题，所以现在，我们将深入探讨一下。

首先，我们需要了解，有两种类型的SAST：

- 扫描原始源代码的SAST工具。
- 扫描来自库的反编译源代码的SAST工具，如**动态链接库**（DLL）或Java Archive（JAR）。

SAST工具逐行扫描代码。它们会报告在代码中发现的潜在漏洞，以便开发人员确

切地知道该在哪里寻找。大多数工具还对漏洞进行评级，甚至提供修复该问题的建议。请注意，SAST工具需要具有语言识别能力。它们扫描源代码，所以它们需要理解代码所使用的语言。如果使用多种语言，可能需要多个SAST工具。

SAST工具被集成到CI/CD流水线中，以实现在整个开发过程中的扫描。大多数工具 都能识别通常被报告和列出的安全问题，如OWASP的安全问题。每年，OWASP都会发布十大web应用程序安全风险。目前，前10名列出了代码注入、敏感数据暴露和监控不足等问题。

> TIPS
>
> OWASP top 10可在https://owasp.org/www- project-top-ten/找到。

我们将在本章的最后一节再谈监控。

动态应用安全测试集成

DAST并不扫描代码。简单地说，DAST工具是模拟攻击的。在 "*第13章 使用行业安全框架实施DevSecOps*" ，我们讨论了MITRE ATT&CK框架，其中列出了有助于利用漏洞的技术。DAST工具通过注入恶意代码串或通过暴力手段执行这些技术。通过这样做，DAST试图识别应用程序功能中的漏洞，而不是源代码中的漏洞。

DAST是复杂的，这意味着它的成本很高。它通过应用程序运行事务，与此同时，它也依赖于与应用程序交互的组件。例如，你可以想到前端的应用程序和后端的数据

库。DAST工具需要配置好才能有效。

在最后一点上，DAST接近渗透测试。大多数安全人员和工程师将依靠手动渗透测试来检测不同应用程序堆栈和服务之间的集成漏洞，特别是当这些堆栈和服务由不同的平台和供应商使用时。在现代IT中，系统由**基础设施即服务（IaaS）**、**平台即服务（PaaS）**和**软件即服务（SaaS）**组成，托管在各种平台上，如AWS、Azure或私有数据中心，集成和测试集成点变得越来越重要。

请注意，公有云平台对渗透测试有严格的政策。例如，AWS只支持数量非常有限的服务，如他们的计算平台EC2。违反这些政策可能会受到惩罚，或者在最坏的情况下，禁止使用服务。

使用CircleCI orbs进行集成

在DevSecOps集成的世界里，一个相当新的现象，但在这里值得一提的是Orbs--可重复使用的YAML代码片段，它负责可重复的操作，如安全扫描。Orbs允许与流行的安全扫描工具直接集成。这是CircleCI的一个概念，这是一家位于旧金山的公司，提供自动化CI的工具。

它声称这些工具在CI/CD流水线中都有开箱即用的集成。例如，orbs for Probely（web漏洞扫描器）和SonarCloud（代码分析）是可用的，但开发人员也可以创建自己的orbs，并将其推送到开源的Orb Registry。

现在，我们已经涵盖了集成的安全框架和工具，我们必须确保对结果进行跟踪。在 最后一节，我们将更详细地讨论监控。

实施监控

安全的一个关键因素是确保必要的安全政策到位，并确保环境确实得到保护。这听起来很简单，但它需要对监控进行适当的配置。开发人员需要信息来帮助他们在第一时间修复错误，但也需要改进代码，并随之改进应用程序。这适用于客户体验和性能，但也适用于确保应用程序受到保护。黑客们不会坐以待毙：他们会不断找出攻击系统的新方法。因此，我们需要不断监控应用程序内部发生的事情。

安全监控不仅是检测意外行为。它要分析所有的行为。这有助于开发人员更深入了解如何改进他们的代码。为此，监控需要提供三个主要服务：

- **收集**
- **分析**
- **告警**

有时，存储和可视化被添加到这些服务中。然而，存储监控数据更多的是关于日志，而可视化是关于全面呈现监控数据。这些都是重要的服务，但它们不是监控本身的核心。当然，你需要方法论来接收和查看告警。举个例子，Grafana是一个流行的工具，它提供了跨平台的仪表盘，允许我们查询和可视化存储的监控数据。

在我们实施监控之前，架构师需要评估以下四个W问题：

- （what）我们在监控什么？
- （why）我们为什么要监控这个（为什么不是其他东西）？
- （when）我们什么时候监测？

- （who）谁在关注（谁需要被告知）？

（what）什么是指收集数据，（why）为什么是指触发分析，而（who） 谁是指向企业内正确的人或实体发送告警。正如我们之前提到的，正确的监控对于我们在DevOps中的反馈循环至关重要，包括可能影响系统的安全事件的告警。有个很大的误解是，认为只有代码和托管在云平台上的系统是被监控和默认安全的。其实不然，像AWS、Azure和Google Cloud这样的平台只是提供了巨大的工具箱，你可以用它来保护你在平台上运行的代码。工程师将需要自己配置监控。这包括监控基础设施，如在云中使用的**虚拟机（VM）**的健康状态。虚拟机使用量的某种增加（虚拟机的CPU或内存出现无法解释的峰值）表明可能出现了问题。

接下来，为了对DevOps有用，监控必须从应用程序本身收集指标。这样一来，监控就提供了关于应用程序运行状态的信息。应用程序必须被启用以向监控系统提供这些信息。这些信息通常在应用程序的代码中标明。例子包括**调试**debug和**警告**warning等标签。监控系统会接收这些信息，并确保这些信息以可理解的格式传递给工程师。

监控工具如何做到这一点？有几种方法可以做到这一点，但在大多数情况下，工具要么使用收集系统数据的代理，要么模拟（无代理的）事务。通过事务，工具将向应用程序发送一个事务，并在事务被返回时分析其响应。

注意事项

基于事务的监控并不意味着这些工具不能或不会使用代理。存在有代理和无代理的系统。一些企业的政策规定，由于系统的开销，或者由于企业担心代理的侵入性，他们的系统上的代理是被禁止的。

在监控代理的情况下，当应用程序被部署到平台上时，这些可以而且应该成为应用程序的理想状态配置的一部分。记住左移（shift-left）：这应该被集成到整个DevOps链中，以便监控从代码被推送到开发、测试和模拟系统的那一刻开始。换句话说，监控不仅仅是为了生产。

由于监控工具必须具备的能力，这些工具会变得非常庞大和复杂。代理可能会导致一些所谓的系统开销，也就是说，代理会增加对系统中资源的使用，或者引发更多的网络流量。这些方面应该在架构上加以考虑。再次，从四个W问题开始。

最后，企业最终可能不会只使用一种监控工具，而是使用一连串的工具。他们可能会使用AWS CloudWatch或Azure Monitor来监控云资源，使用Prometheus来收集容器平台等高动态系统的实时数据，并使用Sysdig来监控云原生应用程序。具体到安全监控方面，企业可以使用Splunk Security Essentials或云原生的Azure Sentinel。要注意的是，很多安全工具的重点是网络和身份以及对系统的访问，而不太关注应用程序或应用程序代码。这就是为什么企业最终要使用监控工具链：这是有原因的。架构师对什么工具能满足企业的需求有很大的发言权。

注意事项

我们在这里将只提到几个工具。这决不是为了详尽无遗，也不是为了推广特定的工具而不是其他工具。这里提到的这些工具是在企业中广泛使用的工具，但还有许多其他伟大的工具可用。

工程师们可能不希望使用几个控制台来观察这些不同工具的输出。诸如ServiceNow和BMC Helix这样的总体企业套件提供了实现单一视图的中介平台：各种

监控工具和数据收集过程可以在这一套件中汇总。这些都是复杂的系统，需要高度熟练的专业人员来实施和配置，但在一个IT日趋复杂的世界中，这种投资是值得的。关于DevOps团队，请记住，他们完全负责开发、部署和运维他们的代码。你可能不想依赖这些 "单体"类型的整体管理系统，但在拥有各种产品和服务的企业中，对所有的资产、开发和这些部署的状态有一个完整的视图是非常必要的。

因此，我们已经遵从了安全框架，定义了政策，并在我们的DevOps和CI/CD生命周期中整合了工具，但我们真的可以开始了吗？IT正在迅速变化并且每天都在变得更具挑战性。攻击和违规行为每天都在新闻中出现。这对我们如何保护我们的系统有很大影响。我们不能再相信我们企业网络中的任何人。另一方面，我们希望给开发人员尽可能多的自由，以便在编码方面获得最佳结果。毕竟，DevOps的第一句话是关于信任的：你建立，你运行，因为你可以。但我们如何处理信任问题？这就是本书的最后一个主题：零信任架构以及它们对DevOps的影响。

摘要

DevOps和DevSecOps并不是独立的东西：安全必须与DevOps完全集成。在这一章中，我们讨论了如何在DevOps中集成安全，不仅关注扫描工具，而且主要关注治理，应用威胁建模，以及监控我们的DevOps环境。对于治理，我们研究了GRC原则，使企业能够管理不确定因素（如安全风险），同时确定战略以实现其业务目标。这是将安全融入企业所有层面的基础步骤，随之而来的是产品和服务的开发。

为了检测、识别和抵御攻击，我们需要与威胁建模合作。在这一章中，我们讨论了OWASP，它提供了对安全事件如何影响企业的洞察力。接下来，我们以更详细的方式讨论了安全扫描。SAST和DAST是DevSecOps的必要条件。

在上一节中，我们了解了架构师在实施监控时需要采取的各种步骤。他们需要问自己四个基本问题：我们要监控什么，为什么要监控，何时监控，以及谁需要被告知？我们还研究了监控工具的特点。

安全是关于信任的，在现代信息技术中，随着攻击数量和种类的增加，基本规则是企业不能再相信任何人或任何东西。这就导致了架构中的一个特定领域：**零信任**。这就是本书最后一章的主题。

问题

1. 真或假：在OWASP中，威胁代理可以是内部和外部的。
2. 说出两种类型的SAST工具。
3. 监控的三个主要功能是什么？

进一步阅读

- 用于管理IT业务的IT4IT，Rob Akershoek等人，The Open Group。
- Mike Vizard在DevOps.com上发表的关于引入私人轨道的博文，2021年：https://devops.com/circleci-adds-private-orbs-to-devops-toolchain/#:~:text=Constructing%20an%20orb%20gives%20DevOps%20team%20relatively,of%20the%20DevOps%20tea-m%20can%20more%20easily%20consume。
- ITSecuroty.org上的文章，对安全测试和整合有深入的见解，作者Adrian Lane

，2019年：https://itsecurity.org/enterprise- devsecops- security-test-integration-and-tooling/。

- DevOps中流行的安全工具概述：https://dzone.com/ articles/an- overview-of-security-testing-tools in-devops。
- Hands-on Security in DevOps，Tony Hsiang-Chih Hsu，Packt Publishing，2018。

15

实施零信任架构

数字化转型是企业的新范式。企业正在采用数据驱动的架构，在云中使用越来越多的本地服务，并通过这种方式加速其产品和服务的发展。在这种压力下，安全必须跟上，并确保环境（在很多情况下甚至是关键任务环境）保持弹性。这就是零信任的领域。

本章解释了什么是零信任以及为什么它对DevOps很重要。零信任假定企业网络内的一切都很安全，这包括DevOps流水线。零信任环境中使用的一些技术是服务网格和微服务，我们将在本章的最后一节讨论这个话题。

完成本章后，你将了解到零信任的含义以及它对DevOps的影响。你将了解到微服务和安全服务网格如何推动安全的数字化转型。在最后一节，我们将简要地讨论云平台提供的一些解决方案。

在这一章中，我们将涵盖以下主要议题：

- 了解零信任原则
- 零信任安全的架构
- 在架构中包含微服务
- 在流水线中集成零信任

了解零信任原则

首先，零信任真的意味着零信任。在过去的几年里，零信任的原则在IT安全领域已经有了很大的发展，这是有原因的。攻击不只是来自外部，也来自企业的内部网络。零信任主张任何用户，或者说每一个身份，都要经过认证，无论该用户是在企业的网络内部还是外部。当认证通过后，用户必须根据安全策略进行验证，并在访问应用程序之前获得授权。数据访问应该只通过验证过的应用程序来授予，而用户是通过验证和授权的。

在我们了解这将如何在DevSecOps中工作之前，特别是在**持续集成/连续部署**（CI/CD）流水线中，我们需要更深入地了解零信任的原则。

零信任从知道谁在企业的网络中开始。在这一点上，有一件重要的事情需要注意：在云中，一切是一个身份。它可以是一个真实的用户，一个人，但也可能是一个被触发用来执行特定动作的服务。另外，服务有一定的权利：它们被允许执行特定的动作或获取定义的数据集，并被禁止采取其他动作。因此，所有的身份，或者更准确地说是账户，都必须是已知的，除此之外，还必须清楚他们有什么权利。这意味着，企业必须不断监控和验证所有的账户，以及他们的证书和权利。这必须是实时进行的。

现在，你可能认为零信任主要是关于监控账户，但还有更多，零信任还意味着，企业已经采取了措施，防止认证用户做超过他们授权的事情。你可能在考虑给账户设置最低权限，但你也需要考虑网络分段和限制网络上的特定协议。基本上，你需要考虑包括账户的一切，以便它只能在账户被授权的地方执行它被授权的任务。这必须通过强大的**身份和访问管理**（IAM）策略、网络分段、外部和内部防火墙、网关和严格的路由策略（如拒绝所有和只允许白名单地址）来执行。

零信任中必须包含的原则如下：

- **帐户类型和凭证总是基于最小权限。**
- **指定的特权权利和这些权利的适用规则。**
- **定义了服务和应用的端点。**
- **认证协议。**
- **安全监控包括入侵检测、入侵预防和异常检测。**
- **用最新的版本和最新的补丁对操作系统进行加固。**
- **软件的生命周期，有最新的版本和最新的补丁。**

这对DevOps有什么影响？这个问题的答案是：零信任对DevOps和敏捷工作方式有巨大影响。DevOps是关于在开发和部署应用程序代码方面追求速度。这就要求DevOps团队具有灵活性并承担很大的责任。诚然，非常严格的安全规则会阻碍开发和部署的快速进程。然而，没有其他方法可以保护企业的资产。DevOps团队也有责任保护这些资产。

其后果是，DevOps团队也必须坚持零信任。团队只能使用被允许进入代码库的账户，在云中企业网络特定分段中进行构建，只使用被批准的操作系统、软件和工具，并应用由路由和防火墙规则执行的安全策略。

不过，零信任并不意味着DevOps的进程会默认放慢。只有当零信任的责任被置于团队之外时才会发生这种情况。例如：团队已经准备好部署的代码，但现在必须等待

一个特定的防火墙端口被打开。如果该端口已经被列入白名单，并且自动安全扫描已经验证了代码符合防火墙规则，那么就可以迅速完成。如果批准必须通过安全部门，需要手动验证一切，那么它将大大减慢这一过程。

因此，我们需要在DevOps中包括零信任。我们将在接下来的章节中讨论这个问题。

零信任安全架构

有了对零信任概念的充分理解，我们就可以定义应用零信任原则的架构。下面的指南将有助于定义架构。其中一些原则可能是显而易见的，而其他原则可能会导致开发人员开发和部署应用程序方式的限制。但是，在一天结束时，我们需要确保企业资产的安全。

- **评估和分析所有的访问控制**。对于IAM的严格策略必须执行到位。最低权限必须是这些策略的一部分。根据**美国国家标准技术研究所**（NIST）的说法，这是零信任的骨干。他们定义了一套零信任架构的原则，全部涉及企业处理IAM的方式。关键的原则是要有一个单一的身份来源。在大多数情况下，企业将使用**活动目录**（AD）来实现这一点。简而言之，任何用户或身份都必须被AD所了解。

- **接下来，必须有强有力的认证**。该身份真的是它所声称的人吗？**多因素认证**（MFA）是强烈建议的。NIST还强调，需要验证和确认用户被授权和认证的环境。例如：从哪台机器访问仓库，该设备是否符合企业的标准？很多开发者都有自己的机器，有自己喜欢的工具。这必须被评估，以明确这是否符合安全策略。

- **必须定义和控制对应用程序的具体访问策略**。一个在营销网站上工作的开发人员

可能不需要访问控制企业供应链的应用程序。在这种情况下，对该应用程序的访问应该受到限制。因此，零信任意味着每个应用程序都有自己的一套策略：谁有资格访问它，访问到什么程度，以及该应用程序的权利是什么？

- **数据分类和数据安全是零信任架构的下一个构建模块。**数据必须得到保护。现代的、基于云的IT所面临的挑战是，数据可以在任何地方，并且在不同的平台、应用程序和服务中共享。企业需要确切地知道他们的数据在哪里，是什么类型的数据，以及在严格的条件下谁或什么被允许访问它。数据必须被识别和分类：例如，它是保密的，还是可以被公开访问？严格的隐私法规，如欧盟的**《通用数据保护条例》**（GDPR），是对数据进行分类的准则；应用这些准则是企业的责任。
- **NIST和国家网络安全卓越中心（NCCoE）也将可信云定义为一个构建模块。**这是因为云在默认情况下所具有的动态特性。现在我们真正处于DevOps 的核心，我们按照自主工作团队的规则，可以在他们需要的瞬间轮转环境，修改环境，甚至删除环境。这些环境将使用数据，但其中一些环境可能只是短暂的，而其他环境最终会被推到生产中。一切都是代码的云技术为这种情况提供了便利。这对安全来说是一个巨大的挑战，特别是在保持环境与安全策略的一致性方面。因此，安全必须被嵌入到DevOps中。监控应该是实时的，并控制能够识别任何违反安全策略的行为，即使这确实意味着开发会因此而停止。

综上所述，我们可以说，零信任主要是指尽可能地分离网段分段、应用、数据和服务，只允许经过认证和授权的用户以最少的权限访问这些不同的组件。微服务将帮助架构师们实现这一目标。然而，微服务确实带来了挑战。这些挑战可以用服务网格来克服。

微服务架构

DevOps是指通过更快的代码发布获得更高的生产力。DevOps团队可以专注于特定的任务和只为执行该任务而设计的代码。他们开发的代码独立于其他服务，以提高重点、交付速度和客户体验。安全原则被应用于这些服务，并通过自动扫描的方式不断验证。DevOps默认为分布式架构，与系统作为一个整体来设计和构建的单体架构相反。在DevOps中，架构将由微服务驱动：一个应用程序被定义为独立服务的集合，它们将通过指定的协议相互通信。下图展示了微服务的原理：

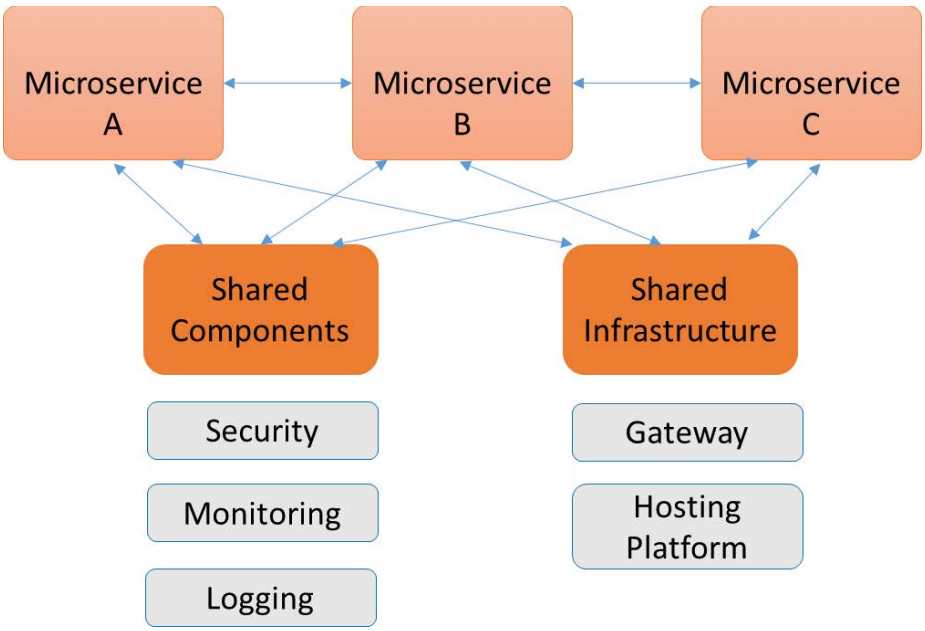


图15.1-微服务的原理

在安全方面，我们可以认为微服务架构比单体系统更安全。如果其中一个服务被攻破，并不自动意味着整个应用栈被攻破，只要受影响的服务被控制得足够好。不幸的是，事情并没有那么简单。原因是，微服务确实需要能够相互通信。下一个问题是：

我们怎样才能以安全的方式实现这一点？答案是服务网格。

首先，让我们看看微服务架构的最佳实践：

- **防御策略**。微服务允许各种防御层或安全层。作为一个例子，一个门户网站需要公开访问，但应用程序和数据应该受到保护。一个很好的例子是移动银行应用程序。该应用可以在任何智能手机上访问：用户可以从应用商店下载并在手机上安装它。要访问检索和展示账户信息的应用程序，用户将需要拥有几样东西：一个在该特定银行的账户和一个允许他们使用移动应用程序的账户。这是两件独立的事情。很明显，账户数据也应该受到保护，例如，通过加密。
- DevSecOps：正如我们在前几章中所看到的，这都是关于将安全实践嵌入到DevOps中。在整个构建过程中，代码会根据策略和行业安全与合规框架进行自动化扫描。但是，这不仅仅是在构建过程中；在部署后，应用程序和代码应持续监控漏洞。
- MFA：每个应用程序都应该只用MFA（多因素认证）来访问。一个用户名和密码是不够的；应该使用第二因素进行认证，例如，在某人经常登录的设备以外的不同的设备上使用认证应用程序。即使MFA已经被用于访问一个应用程序，当特定的、高度机密的数据从该应用程序被访问时，可能需要进行重新认证。对一个应用程序具有访问权限，并不意味着默认情况下，用户应该可以访问该应用程序所能检索到的所有数据。应用程序和数据是不同的层级或层次。
- **依赖性**。在云环境中，我们将可能使用云服务，如**平台即服务**（PaaS）和**软件即服务**（SaaS）。我们将需要应用编程接口（API）来实现这些服务之间的互动。这些都是依赖关系，如果没有得到很好的验证和配置，它们可能会导致

漏洞和安全威胁。必须对源代码进行扫描，以发现脆弱的依赖关系。

依赖性可能是安全方面的最大挑战。在现代架构中，我们如何使用微服务来处理这个问题？

了解和使用服务网格

微服务很好的服务了DevOps。这是开发和部署新功能到代码中而不影响其他正在运行的服务的完美方式。由于微服务的颗粒性，开发和部署也可以在低水平上得到保障，从而使用户的服务被中断的风险很低。使用微服务意味着错误的配置或糟糕的编程实现被最小化到只有正在开发和部署的特定服务，同时也最小化了整个应用堆栈的攻击面。为了实现这种工作方式，容器发挥了重要作用。服务和功能被包装并作为容器部署。

下一个挑战是如何让这些容器化的服务和功能安全地相互作用。这就是服务网格的意义所在。为了建立交互，开发人员需要在应用程序代码中配置这些。他们将整合能够与应用程序外的服务进行通信的库，如服务发现、负载平衡，以及设置内部**传输层安全**（TLS）流量到其他服务。首先，配置字符串和它们从应用程序代码中调用的服务需要共享一种通用语言。但更重要的是，当一个服务发生变化时，它也需要在应用代码中进行调整。这使得应用程序代码变得复杂。

服务网格解决了这个问题，它从应用程序中移除复杂性，并将其转移到服务代理上。这个代理现在负责处理应用程序用来与其他功能组件交互的大量第三方服务。比如流量管理，包括负载均衡、认证，当然还有安全和监控。这些服务现在被从应用程序代码中抽象出来，作为一个单独的组成部分。

开发人员只需担心应用程序代码，因为所有其他服务都由服务代理负责。有了这个，我们就有了严格的责任分工。

这听起来是一个很好的解决方案，但它在实践中是如何运作的呢？我们将在本章的最后一节中了解。

在流水线中集成零信任

在前面的章节中，我们讨论了零信任架构的原则以及微服务如何帮助我们实现零信任。接下来，我们学习了如何让微服务通过安全服务网格进行交互。在本节中，我们将学习如何通过容器化应用程序和使用CI/CD流水线的云服务来实现这一点。AWS和Azure等平台提供了这方面的解决方案，我们将讨论这些解决方案。

首先，我们需要了解我们如何为服务网格添加安全性。做到这一点的一个方法是使用sidecar。以一种非常简单的方式来解释，sidecar是容器集群中插入安全策略的一个点。你可以把它设想成一条汽车行驶的主干道，一辆载有特定安全策略的汽车从侧路驶来，插入主路上的汽车队伍中。然而，发生这种情况的点是固定的。

有各种工具可以提供sidecar服务网格。流行的有Istio、Linkerd和Envoy。这些工具的共同点是，它们将所需的功能放在一个单独的容器中，并将其插入到应用程序容器附近，就像我们的描述的插入的汽车。由于大多数使用容器的开发者都是用Kubernetes工作的，所以必须知道，sidecar容器必须和应用容器放在同一个Kubernetes pod中。这是因为pod的命名空间需要相同。应用容器和sidecar可以在CI/CD流水线中集成。

服务网格和sidecar代理的整体原理如下图所示：

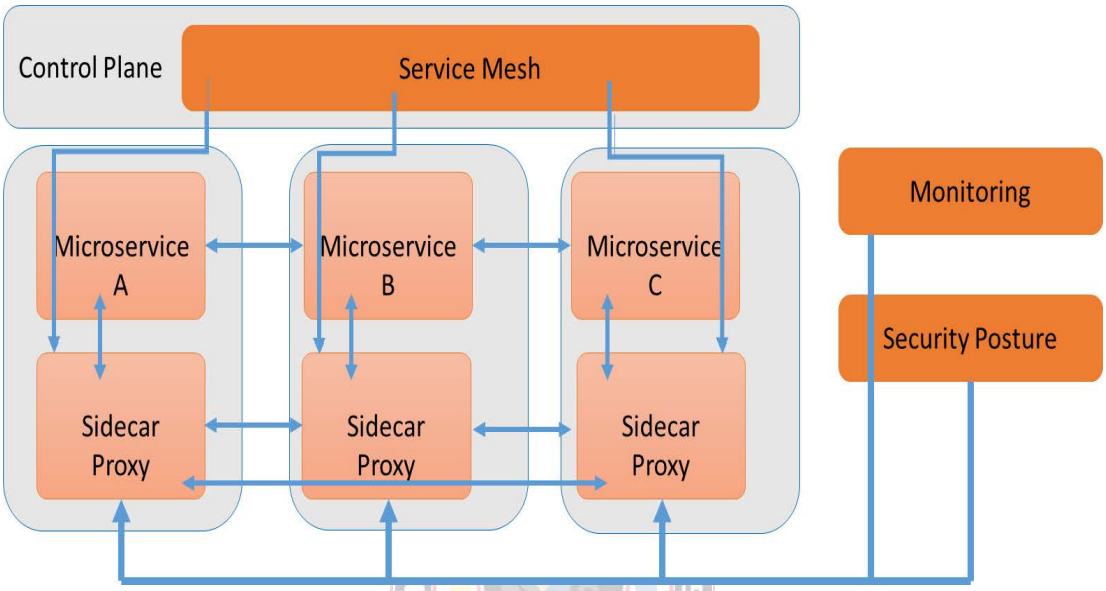


图 15.2-服务网格和sidecar原理

如前所述，云平台也提供服务网格。AWS有AWS App Mesh，当它使用Envoy sidecar代理时，它允许服务之间的交互，而不受底层基础设施的影响。Native App Mesh与AWS Fargate的无服务器基础设施服务、计算引擎EC2，以及Elastic Container Services（ECS）和Elastic Kubernetes Services（EKS）的容器编排服务一起工作。AWS App Mesh的高层架构如下图所示：

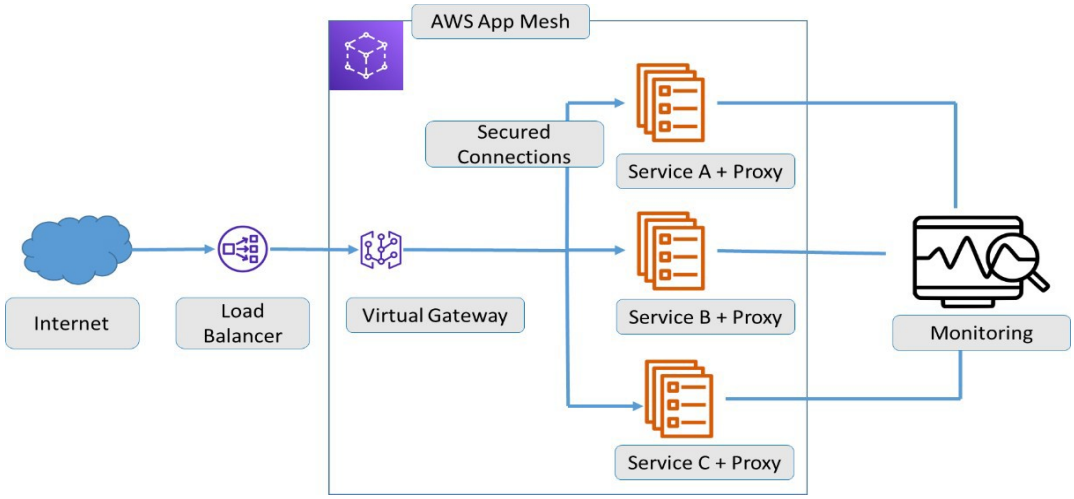


图 15.3-AWS APP Mesh架构

在Azure中，我们使用Azure Service Fabric，这是微软用于部署和管理微服务的容器编排器。2018年推出的完全管理化的网格服务Azure Service Fabric Mesh，从2021年4月开始已经被微软退役。使用Azure的公司可以使用Azure Container Service、Azure Kubernetes Services（AKS）或Azure Service Fabric管理的集群来创建网格功能。Azure Service Fabric的原理如下图所示：

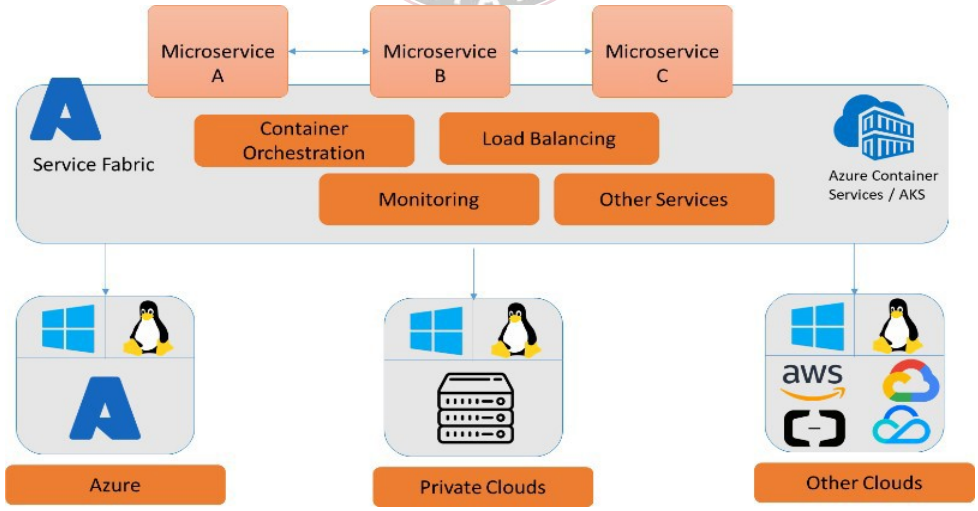


图15.4-Azure Service Fabric的高阶架构

我们的企业DevOps、AIOps和DevSecOps之旅到此结束。在这个数字化转型的时代，架构师们面临着一项艰巨的任务，那就是了解这些方法论如何帮助企业实现IT环境的现代化、在软件开发中变得更加敏捷，同时在开发和部署过程中确保最大的安全性。本书只是一个起点。*The proof of the pudding is in the eating, so go out and try to make it work*。

摘要

在本章中，我们首先学习了零信任架构的原则，我们了解到DevOps团队也需要遵守这些原则。零信任首先要清楚地知道谁可以访问代码库，并知道构建的代码只能部署到严格控制的网段，以便其他服务不受影响。下一步。我们了解到，微服务架构可以很好地服务于DevOps。它们允许独立开发和部署代码中的功能，而不影响其他服务。

我们了解到，微服务是一种安全的架构类型。然而，挑战在于如何在这些微服务之间建立交互。我们研究了服务网格作为一个解决方案，并学会了如何使用sidecar代理技术，将安全策略整合为一个容器化的微服务。我们了解到，sidecar可以用来在我们的微服务旁边插入安全服务和监控。

在最后一节，我们介绍了一些由云供应商Azure和AWS提供的网格服务。至此，我们结束了针对DevOps、DevSecOps和AIOps的企业架构之旅，所有这些都变得越来越重要，需要理解和实施，最终成功推动企业的数字化转型。

问题

1. 在零信任环境下，我们对账户的特权适用什么基本规则？
2. 我们用什么类型的服务来在应用容器旁边插入具有安全策略的独立容器？
3. AWS提供什么来实现服务网格？

进一步阅读

- 国家网络安全卓越中心（NCCoE）关于零信任架构的网站：https://www.nccoe.nist.gov/projects/building- blocks/zero-trust-architecture
- Hands-On Microservices with Kubernetes，作者Gigi Sayfan，Packt出版社。2019
- 关于Microsoft Azure Service Fabric的文档：https://docs. microsoft.com/en-us/azure/service-fabric/service-fabric-overview#:~:text=%20Overview%20of%20Azure%20Service Fabric%2020%201,application%20lifecycle...%204%20Next%20 steps.%20%20More%20
- 在AWS APP Mesh的博客文章：https://aws.amazon.com/app- mesh/?aws-app-mesh-logs.sort- by=item.additionalFields.createdDate&aws-app-mesh- logs.sort-order=desc&whats-new- cards.sort-by=item.additionalFields.postDateTime&whats- new- card.sort-order=desc