为您的组织选择基础设施即代码工具

AWS 规范性指导

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 规范性指导: 为您的组织选择基础设施即代码工具

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

为您的组织选择基础设施即代码工具

Amazon Web Services（贡献者）

2026 年 2 月（文档历史记录）

基础设施即代码 (IaC) 是通过一组配置文件配置和管理应用程序基础架构的过程。IaC 旨在帮助您集中管理基础设施、实现资源标准化和快速扩展，使新环境具有可重复性、可靠性和一致性。它是敏捷和 DevOps 实践的关键组成部分，例如版本控制、持续集成和持续部署。

选择基础设施即代码 (IaC) 工具被视为组织的战略决策。这一决定会影响为公司构建基础架构、应用程序和服务的所有团队。每种工具都有优缺点；因此，没有 one-size-fits-all模型。

过去，管理和配置基础设施是一个充满错误的手动过程。IaC 通过代码简化了这些任务，并已成为部署基础设施的可靠解决方案。IaC 工具使开发人员能够使用编程语言定义和部署基础架构。这不仅可以提高业务灵活性，还可以加速增长和创新速度。此外，IaC 还能显著提高安全性，因为 IaC 允许您的组织在部署之前扫描代码，从而验证基础设施的可靠性和安全性。归根结底，正确的IaC工具不仅仅是一个技术决策，而是一个直接影响业务整体成功的战略决策。

本指南探讨了可用于配置 AWS 资源的五种不同的 IaC 工具： AWS CloudFormation、 AWS Serverless Application Model (AWS SAM) AWS Cloud Development Kit (AWS CDK)、 HashiCorp Terraform 和 Pulumi。它比较了这些工具，并指导您完成选择满足团队、组织和云人才需求的工具的过程。关键是要使所选的 IaC 工具与您的组织目标和开发人员的技能组合保持一致。例如，如果您的团队精通 JavaScript，则可以选择 AWS CDK 使用 TypeScript 作为主要的 IaC 工具，因为它可以优化您的开发工作流程。

实施 IaC 的好处

通过为您的组织选择合适的 IaC 工具，您可以获得以下好处：

- 速度 — IaC 的目标之一是通过消除手动流程来加快工作速度。基于代码的方法可以更轻松地在更短的时间内完成更多工作。手动设置每个 IT 环境非常昂贵，需要专门的工程师和架构师来设置硬件和软件。通过提高速度，组织能够更好地快速适应不断变化的客户需求和市场状况。它还允许您编写 IaC 代码一次，然后将相同的代码部署到数百个环境，所用时间仅为手动创建它们所需时间的一小部分。
- 可扩展性 — IaC 可以更轻松地向现有基础设施添加资源。升级配置速度很快，因此您可以在使用高峰期快速扩展。例如，在黑色星期五等需求旺盛时期，运营在线服务的组织可以扩大规模以满足用户需求。
- 增强安全性 — IaC 擅长为组织实现安全性和合规性。Organizations 能够设置基准安全策略和配置，并通过对您的 IaC 运行自动测试，通过管道强制执行这些策略和配置。如果 IaC 违反合规性规则，管道就会失败并自动向开发者提供反馈。这可以防止创建不安全的基础架构。
- 一致性 — 一致性是 IaC 的另一个重要优势。当多名工程师手动部署配置时，不一致是不可避免的。随着时间的推移，跟踪和重现相同的环境变得越来越困难。这些不一致之处通常会导致开发、质量保证和生产环境之间的关键差异。
- 效率 — IaC 提高了整个开发生命周期的效率和生产力。程序员创建沙盒环境以进行单独开发。运营部门可以快速配置用于安全测试的基础架构。QA 工程师在测试期间拥有生产环境的精确副本。当他们准备好部署时，开发人员只需一步即可将基础架构和代码推送到生产环境。它还可以实现更高水平的协作和团队合作。团队可以为应用程序创建安全的模式，然后与其他团队共享这些模板或结构，从而减少重复工作。
- 降低成本 — IaC 降低了开发软件的成本。无需花费资源手动设置环境。您只需为自己正在使用的资源付费，因此不会产生不必要的开销。
- 提高可靠性-如果您的基础架构很大，则很容易错误地配置资源或按错误的顺序配置服务。使用 IaC，资源的配置和配置始终完全按照声明的方式进行。由于手动创建资源很容易出错，因此在使用 IaC 时，您可以确信您的基础架构将按预期创建。
- 配置偏差检测-当配置您的环境的配置不再与实际环境匹配时，就会发生配置偏差。许多 IaC 工具可帮助您检测漂移并进行补救。例如，如果有人错误地手动修改了您的资源，则可以使用 IaC 工具来检测已更改的内容并将其恢复到预期状态。
- 实验 — 由于 IaC 可以更快、更轻松地配置新基础架构，因此您无需投入大量时间和资源即可进行和测试实验性更改。如果您喜欢结果，可以快速扩展新的生产基础架构。

查看 IaC 的工具 AWS Cloud

本指南探讨了可用于配置 AWS 资源的五种不同的 IaC 工具。它比较了这些工具，并指导您完成选择满足团队、组织和云人才需求的工具的过程。关键是要使所选的 IaC 工具与您的组织目标和开发人员的技能组合保持一致。

在本节中，您将详细了解每种工具的工作原理及其优缺点。例如，有些工具只能在中使用 AWS Cloud，它们不太适合多云或混合云部署。其他人则需要专业编程语言的知识，而其他人则支持常见的编程语言。评估每种工具的优缺点，并考虑哪种工具最适合您的组织。

本节讨论以下 IaC 工具：

- AWS CloudFormation
- AWS Serverless Application Model (AWS SAM)
- AWS Cloud Development Kit (AWS CDK)
- HashiCorp Terraform
- Pulumi

用 AWS CloudFormation 作 IaC 工具

AWS CloudFormation AWS 服务 是一种使用模板文件自动配置 AWS 资源的工具。您可以创建一个描述要部署的所有 AWS 资源的模板，并为您预 CloudFormation 置和配置这些资源。

CloudFormation 模板是使用 JSON 或 YAML 编写的。堆 CloudFormation栈是模板中定义的资源的实现。您可以通过、通过 CloudFormation SDK 以编程方式管理 CloudFormation 堆栈 AWS 管理控制台，也可以通过 AWS Command Line Interface ()AWS CLI来管理堆栈。有关工作 CloudFormation 原理的更多信息，请参阅 CloudFormation 文档中的AWS CloudFormation 概念和AWS CloudFormation 工作原理。

使用的优点 CloudFormation：

- CloudFormation 更改集允许您在部署正在运行的堆栈之前预览这些更改。变更集汇总了对现有堆栈中正在运行的资源的拟议更改。这可以帮助您在部署之前识别冲突或意外后果。例如，如果您更改了 Amazon Relational Database Service (Amazon RDS) 数据库实例的名称，则 CloudFormation 会创建一个新数据库并删除旧数据库。除非您已经备份了旧数据库中的数据，否则您将丢失这些数据。如果您生成更改集，则会看到您的更改将导致数据库被替换，并且您将能够在更新堆栈之前进行相应的计划。

- 如果在部署更改集的过程中出现错误，则会自动回 CloudFormation 滚到上次已知的工作状态。
- 您可以使用 CloudFormation 堆栈集跨多个 AWS 账户 和部署资源 AWS 区域。
- 使用以下命名空间中的资源 CloudFormation 提供者无需支付额外费用：AWS:: *、Alexa:: * 和自定义:: *。在这些情况下，您只需为预置的 AWS 资源付费，就像手动配置资源一样。
- CloudFormation 为你管理状态。这意味着 CloudFormation 可以调用底层服务 AWS 来配置和配置 CloudFormation 模板中定义的资源。
- CloudFormation 提供用于检测和修复配置偏差的工具。有关更多信息，请参阅文档中的检测堆栈和资源的非托管配置更改。 CloudFormation
- 您可以使用 CloudFormation 创建自定义资源。您可以在模板中编写自定义配置逻辑，这些逻辑 CloudFormation 可在您创建、更新或删除堆栈时运行。
- CloudFormation 支持使用CloudFormation 注册表对第三方应用程序资源进行建模、配置和管理。
- CloudFormation 支持将现有资源导入 CloudFormation 管理。

使用的缺点 CloudFormation：

- 如果你不熟悉 JSON 或 YAML 语法，可能需要一些时间来适应。JSON 不是为人类可读而设计的，它不允许你进行内联注释。YAML 允许您发表评论并且更易于阅读。但是，它的语法基于制表符和空格，因此很容易犯缩进错误。
- CloudFormation 不支持多云部署。
- 必须使用更高级别的实现（例如）来创建可重复使用的构造和其他模块化代码。 AWS Cloud Development Kit (AWS CDK)

使用 AWS Serverless Application Model (AWS SAM) 作为 IaC 工具

AWS Serverless Application Model (AWS SAM) 是一个可扩展的工具包 AWS CloudFormation。它包括旨在帮助您更快地创建无服务器应用程序的其他功能。部署 AWS SAM 模板时，会将其转换为 CloudFormation 以创建定义的资源。 AWS SAM 由两部分组成，AWS SAM 模板规范和AWS SAM 命令行界面 (AWS SAM CLI)。尽管您可以直接在 AWS SAM 模板中使用 CloudFormation语法，但它 AWS SAM 提供了自己独特的语法，专门用于加快无服务器开发。这种简短语法允许对无服务器资源（例如 Amazon API Gateway）和资源的 IaC 进行优化。 AWS Lambda AWS Step Functions AWS SAM CLI 是一种开发者工具，其中的功能可帮助您在本地测试 AWS Lambda 功能、创建持续集成和持续交付 (CI/CD) 管道，以及运行命令来部署无服务器应用程序。

使用的优点 AWS SAM：

- AWS SAM 具有与 CloudFormation。相同的优点。
- 相比之下 CloudFormation，您可以更轻松地使用它 AWS SAM 来创建无服务器应用程序和资源，例如由 AWS Lambda支持的 Amazon API Gateway。
- 使用 C AWS SAM LI，您可以在本地测试 AWS Lambda 函数。当您在调试模式下本地调用 Lambda 函数时，可以将调试器附加到该函数。借助调试程序，您可以逐行分步调试代码，查看各种变量的值，并像处理任何其他应用程序一样修正问题。

使用的缺点 AWS SAM：

- AWS SAM 有同样的缺点 CloudFormation。
- AWS SAM 不能在外面使用 AWS。

将作 AWS CDK 为 IaC 工具使用

AWS Cloud Development Kit (AWS CDK)是一个开源软件开发框架，允许您使用熟悉的编程语言来定义云应用程序资源。 AWS CDK 支持 JavaScript、 TypeScript、Python、Java、C# 和 Go。它们 AWS CDK 以安全、可重复的方式 AWS CloudFormation配置您的资源。当你合成 AWS CDK 代码时，结果是一个 CloudFormation 模板。 AWS CDK 提供了简化 AWS 资源定义过程的高级抽象。

AWS CDK 使用构造的概念。 结构是应用程序中的一个组件，它代表一个或多个 CloudFormation资源及其配置，例如亚马逊简单存储服务 (Amazon S3) Service 存储桶。可以对构造进行组合和定制，以创建更复杂的基础架构。有关更多信息，请参阅 AWS CDK 文档中的构造关卡。 根据开发人员编写的代码 AWS CDK 生成 CloudFormation 模板。这样就无需手动创建 CloudFormation 模板。许多组织在社区中自定义、共享和重复使用构造，就像任何其他软件库一样。共享构造可以帮助开发人员更快地编写代码，并在默认情况下采用最佳实践。

AWS CDK 方面可以帮助组织将标准应用于给定范围内的所有结构。该方面可以修改结构，例如通过添加标签。或者它可以验证一些关于构造状态的信息。

AWS CDK 允许开发人员利用他们现有的编程技能和知识来定义云基础架构。通过使用熟悉的编程语言，开发人员可以运用他们的专业知识来描述 AWS 资源，从而更轻松地从应用程序开发过渡到基础设施配置。此外， AWS CDK 还可以加快 AWS 基础设施的创建。与手动编写 CloudFormation 模板相比，这加快了开发生命周期。

使用以下的优点 AWS CDK：

- AWS CDK 支持众所周知的编程语言。

- 通用语言允许使用逻辑结构，例如 for 循环、对象、强类型和其他编程技术。这可以帮助开发人员以简洁无误的方式声明基础架构。这种方法还使得使用集成开发环境 (IDE) 和相关工具来帮助管理大量资源声明的复杂性成为可能。
- AWS CDK 结构是可共享的，可帮助您满足治理和合规要求。
- 这些 AWS CDK 构造可以减少开发的时间和精力。如需了解更多信息，请参阅构造库 API 参考。
- AWS CDK 是基于 CloudFormation。如果你熟悉 CloudFormation 其概念，那么 AWS CDK 概念就更容易理解。
- AWS CDK 可以帮助您执行单元测试和快照测试。
- 如果原生不支持某项功能 AWS CDK，则可以使用 1 级构造和原始覆盖。或者，您可以使用直接调用 API 的CloudFormation 自定义资源。
- 您可以通过删除 CloudFormation 堆栈来高效地清理资源。

使用以下方法的缺点 AWS CDK：

- AWS CDK 需要在每个环境中都有一个引导环境。 AWS 账户 Bootstrapping 是一次性操作，您必须对部署资源的每个环境执行此操作。
- AWS CDK 只能用于在中部署 IaC。 AWS Cloud

使用 Terraform 作为 IaC 的工具 AWS Cloud

HashiCorp Terraform 是一款基础设施即代码 (IaC) 工具，可帮助您管理云基础架构。使用 Terraform，您可以在配置文件中定义云和本地资源，您可以对其进行版本控制、重复使用和共享。然后，您可以使用一致的工作流程在整个生命周期中配置和管理所有基础架构。

开发人员使用一种名为 Terraform 语言的高级配置语言。Terraform 语言的原生低级语法是HashiCorp 配置语言 (HCL)。Terraform 语言旨在便于人类阅读和写作。您可以使用 Terraform 语言来描述云或本地基础设施的所需终端状态。然后，Terraform 会生成达到该最终状态的计划，然后您执行该计划来配置基础架构。

使用 Terraform 的优点：

- Terraform 与平台无关。您可以将其与任何云服务提供商一起使用。您可以跨云提供商和许多其他云提供商配置、测试 AWS 和部署基础架构。如果您的组织使用多个云提供商，Terraform 可以成为管理云基础架构的单一、统一、一致的解决方案。有关多云支持的更多信息，请参阅 Terraform 网站上的多云配置。

- Terraform 是无代理的。它不需要在托管基础架构上安装任何软件。
- Terraform 模块是重用代码并坚持自己不要重复 (DRY) 原则的强大方法。例如，您可能为包含亚马逊弹性计算云 (Amazon EC2) 实例、亚马逊弹性块存储 (Amazon EBS) 卷和其他按逻辑分组的资源的应用程序进行了特定的配置。如果您需要创建此配置或应用程序的多个副本，则可以将资源打包到 Terraform 模块中并创建该模块的多个实例，而不必多次复制整个代码。这些模块可以帮助您组织、封装和重用配置。它们还提供一致性并确保最佳实践。
- Terraform 能够检测和管理基础架构中的偏差（Terraform 博客文章）。例如，如果在 Terraform 之外修改了 Terraform 管理的资源，则可以使用 Terraform CLI 检测偏差并将其恢复到所需的状态。

使用 Terraform 的缺点：

- 可能不支持与任何云提供商相关的新功能或新资源。
- Terraform 不会像那样自动管理你的状态。 AWS CloudFormation默认情况下，它存储在本地文件中，但您也可以将其远程存储在 Amazon S3 存储桶中或通过 Terraform Enter prise。
- Terraform 状态可能包含敏感数据，例如数据库密码，这可能会带来安全问题。最佳做法是加密状态文件，将其远程存储，对其启用文件版本控制，并使用最低权限对其进行读写操作。有关更多信息，请参阅使用 AWS Secrets Manager 和 HashiCorp Terraform 保护敏感数据。
- 2023年8月，Hashicorp宣布将不再根据 Mozilla 公共许可证获得开源许可。相反，它现在是根据商业来源许可证获得许可的。

使用 Pulumi 作为 IaC 的工具 AWS Cloud

Pulumi 是一个开源基础设施即代码工具。它支持现有的常见编程语言，例如、、Python TypeScript JavaScript、Go、.NET、Java 和 YAML。它还使用其原生生态系统通过Pulumi SDK与云资源进行交互。可下载的 CLI、运行时、库和托管服务共同用于配置、更新和管理云基础架构。

您可以使用 Pulumi SDK 在任何云上创建和部署使用容器、无服务器函数、托管服务和基础设施的云软件。

你可以选择将 Pulumi 与 Pulumi Cloud 配对。Pulumi Cloud 是一项托管服务，用于存储您的状态和机密，并管理您的 Pulumi 基础设施的部署。

使用 Pulumi 的优点：

- Pulumi 提供来自五十多家云和软件即服务 (SaaS) 提供商的基础设施。
- 它提供了一个完整且一致的界面，旨在降低云的复杂性。

使用 Pulumi 的缺点：

- 虽然Pulumi有一个活跃的社区可以提供支持，但它比Terraform社区要小。
- Pulumi 的学习曲线更陡峭。

选择 IaC 工具

那么，你应该选择哪种工具呢？

由于有这么多不同的工具选项和不同的业务需求，因此没有 one-size-fits-all 一种方法。除了本指南中讨论的每种工具的优缺点外，还要考虑以下针对您的业务需求和运营模式的建议：

- 如果您正在管理或部署依赖关系最少的无服务器 AWS 解决方案， AWS Serverless Application Model (AWS SAM) 可能是一个不错的选择。它具有与之相同的所有功能 AWS CloudFormation。它还简化了对无服务器应用程序的测试和部署。 AWS Cloud
- 如果您完全在上面管理基础架构 AWS AWS CloudFormation ，那么 AWS Cloud Development Kit (AWS CDK) 这些都是不错的选择。它们提供 out-of-the-box 状态管理，您也可以在本地使用新功能或 AWS 资源。
- 如果你想要一个多提供商实用程序，特别是用于管理多云或混合云基础架构，那么Terraform可能是一个不错的选择，因为它与平台无关。借助Terraform，您还可以使用各种插件，并且它拥有一个提供企业支持选项的庞大社区。
- 如果你有一个包含最佳实践的自上而下的发行版，并且你有使用常用编程语言创建、发布和分发可重复使用的模块的编排，那么 AWS CDK 这可能是一个不错的选择。
- 如果您的组织可以承受高风险并需要支持多云或混合云环境，请考虑使用 Pulumi。

后续步骤

本指南讨论了几种基础设施即代码 (IaC) 工具的优缺点，您可以使用这些工具来构建、部署和管理 AWS 资源和基础架构。根据您选择的工具，我们建议您访问以下资源以开始使用。

AWS Cloud Development Kit (AWS CDK)

- AWS CDK 入门研讨会

AWS CloudFormation

- AWS CloudFormation 实验室
- AWS CloudFormation 研讨会

HashiCorp Terraform

- AWS Terraform 工作坊
- 适用于 T@@ erraform 存储库的 CDK () GitHub

Pulumi

- Pulumi 存储库 () GitHub
- AWS 使用 Pulumi 车间实现现代化

资源

AWS 文档

- 使用 in 创建 IaC 项目的最佳实践（AWS 规范性指南） AWS CDK TypeScript
- 使用 cdk-nag 规则包（AWS 规范性指南）查看 AWS CDK 应用程序或 CloudFormation 模板以获取最佳实践
- 简介 AWS： DevOps 基础架构即代码（AWS 白皮书）
- AWS CloudFormation 文档

其他文档

- 基础架构即代码入门（HashiCorp网站）
- HashiCorpTerraform 文档
- Pulumi 文档
- Pulumi：什么是基础设施即代码（Pulumi 网站）

工具

- cdk-nag () GitHub
- cfn-nag () GitHub
- Terratest

贡献者

编写

- Siamak Heshmati，高级建筑师， DevOps AWS
- 梅森·卡希尔，高级 DevOps 顾问， AWS
- Sandip Gangapadhyay，高级建筑师， DevOps AWS
- Sandeep Gawande，高级首席顾问， AWS
- 首席顾问 Rajneesh Tyagi AWS

正在审阅

- 大卫·豪顿，高级参与经理， AWS
- Steven Guggenheimer，高级云应用程序架构师， AWS

技术写作

- Lilly AbouHarb，高级技术撰稿人， AWS

文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 RSS 源。

变更	说明	日期
Pulumi 最新消息	我们更新了 Pulumi 章节。	2026年2月17日
初次发布	—	2024 年 2 月 23 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。