AWS 白皮书

AWS DevOps 上的简介

AWS DevOps 上的简介: AWS 白皮书

Table of Contents

AWS DevOps 上的简介

发布日期：2023 年 4 月 7 日 (文档修订)

如今，企业比以往任何时候都更加踏上数字化转型之旅，与客户建立更深层次的联系，实现可持续和持久的商业价值。各种形式和规模的组织都在通过比以往任何时候都更快地进行创新，从而颠覆竞争对手并进入新的市场。对于这些组织来说，重要的是要专注于创新和软件颠覆，这使得简化其软件交付至关重要。那些缩短从构思到生产的时间，将速度和敏捷性作为优先事项的组织可能会成为未来的颠覆者。

尽管要成为下一个数字颠覆者，需要考虑几个因素，但本白皮书重点 DevOps介绍了 Amazon Web Services (AWS) 平台中的服务和功能，这些服务和功能将有助于提高组织高速交付应用程序和服务的能力。

简介

DevOps 是文化理念、工程实践和工具的结合，可提高组织以更高的速度和更高质量地交付应用程序和服务的能力。随着时间的推移，在采用时出现了几种基本实践 DevOps：持续集成 (CI)、持续交付 (CD)、基础设施即代码 (IaC) 以及监控和记录。

这篇论文重点介绍了可帮助您加快 DevOps 旅程的 AWS 功能，以及 AWS 服务如何帮助消除与 DevOps适应相关的无差别繁重的工作。它还描述了如何在不管理服务器或构建节点的情况下构建持续集成和交付能力，以及如何使用 IaC 以一致且可重复的方式配置和管理您的云资源。

- 持续集成：一种软件开发实践，开发人员定期将其代码更改合并到中央存储库中，然后运行自动构建和测试。
- 持续交付：一种软件开发实践，在这种实践中，可以自动构建、测试代码更改，并为发布到生产环境做好准备。
- 基础设施即代码：一种使用代码和软件开发技术（例如版本控制和持续集成）来配置和管理基础架构的实践。
- 监控和记录：使组织能够了解应用程序和基础架构性能如何影响其产品最终用户的体验。
- 沟通与协作：建立实践是为了拉近团队的距离，并通过建立工作流程和分配职责 DevOps。
- 安全：应该是一个跨领域的问题。应保护您的持续集成和持续交付 (CI/CD) 管道和相关服务，并应设置适当的访问控制权限。

对这些原则中的每一项的研究表明，这些原则与提供的产品有着密切的联系 AWS。

您的架构是否良好？

AWS Well-Architected Fram ework 可帮助您了解在云中构建系统时所做决策的利弊。利用此框架的六个支柱，您可以了解到设计和运行可靠、安全、高效、经济有效且可持续的系统的架构最佳实践。使用 AWS 管理控制台中免费提供的 AWS Well-Architected 工具，您可以针对每个支柱回答一系列问题，根据这些最佳实践来审查您的工作负载。

持续集成

持续集成 (CI) 是一种软件开发实践，开发人员定期将其代码更改合并到中央代码存储库中，然后运行自动构建和测试。CI 有助于更快地发现和解决错误，提高软件质量，并缩短验证和发布新软件更新所需的时间。

AWS 为持续集成提供以下服务：

主题

- AWS CodeCommit
- AWS CodeBuild
- AWS CodeArtifact

AWS CodeCommit

AWS CodeCommit是一项安全、高度可扩展的托管源代码控制服务，用于托管私有 git 存储库。CodeCommit 减少了您操作自己的源代码控制系统的需求，无需配置和扩展硬件，也无需安装、配置和操作软件。你可以用它 CodeCommit 来存储从代码到二进制文件的所有内容，而且它支持标准功能 GitHub，可以与现有的基于 Git 的工具无缝协作。您的团队还可以使用 CodeCommit在线代码工具来浏览、编辑和协作处理项目。 AWS CodeCommit 有几个好处：

- 协作-专 AWS CodeCommit 为协作软件开发而设计。您可以轻松提交、分支和合并代码，这有助于您轻松控制团队的项目。 CodeCommit 还支持拉取请求，它提供了一种请求代码审查和与协作者讨论代码的机制。
- 加密 — 您可以根据需要 AWS CodeCommit 使用 HTTPS 或 SSH 来回传输文件。还会使用客户特定的密钥通过 AWS Key Management Service(AWS KMS) 自动对存储库进行静态加密。
- 访问控制 — AWS CodeCommit 使用 AWS Identity and Access Management(IAM) 来控制和监控谁可以访问您的数据，以及他们访问数据的方式、时间和地点。 CodeCommit 还可以帮助您通过AWS CloudTrail和 Amazon 监控您的存储库 CloudWatch。

  高可用性和耐久性 — 将您的 AWS CodeCommit 存储库存储在亚马逊简单存储服务 (Amazon S3) 和亚马逊 DynamoDB 中。您的加密数据以冗余方式存储在多个设施中。这种架构提高了存储库数据的可用性和持久性。
- 通知和自定义脚本-现在，您可以接收影响仓库的事件的通知。通知将以亚马逊简单通知服务 (Amazon SNS) 通知的形式发送。每份通知都将包含一条状态消息以及一个指向其事件生成该通知的

资源的链接。此外，使用 AWS CodeCommit 存储库提示，您可以发送通知并使用 Amazon SNS 创建 HTTP 网络挂钩，或者AWS Lambda调用函数来响应您选择的存储库事件。

AWS CodeBuild

AWS CodeBuild 是一项完全托管式连续集成服务，可编译源代码、运行测试以及生成可供部署的软件包。您无需预置、管理和扩展自己的构建服务器。 CodeBuild 可以使用 GitHub 企业版 GitHub、BitBucket AWS CodeCommit、或 Amazon S3 中的任何一个作为源提供商。

CodeBuild 可以连续扩展，并且可以同时处理多个构建。 CodeBuild 为各种版本的微软 Windows 和 Linux 提供了各种预配置的环境。客户还可以将其自定义的构建环境作为 Docker 容器使用。CodeBuild 还与 Jenkins 和 Spinnaker 等开源工具集成。

CodeBuild 还可以为单元测试、功能测试或集成测试创建报告。这些报告提供了运行了多少测试用例以及有多少通过或失败的测试用例的直观视图。构建过程也可以在亚马逊虚拟私有云（Amazon VPC）内运行，如果您的集成服务或数据库部署在 VPC 内，这将非常有用。

AWS CodeArtifact

AWS CodeArtifact是一项完全托管的工件存储库服务，组织可以使用它来安全地存储、发布和共享其软件开发过程中使用的软件包。 CodeArtifact 可以配置为自动从公共工件存储库中获取软件包和依赖项，以便开发人员可以访问最新版本。

软件开发团队越来越依赖开源软件包来执行其应用程序包中的常见任务。对于软件开发团队来说，保持对特定版本的开源软件的控制以确保软件没有漏洞已变得至关重要。使用 CodeArtifact，您可以设置控件来强制执行此操作。

CodeArtifact 可与常用的包管理器和构建工具（例如 Maven、Gradle、npm、yarn、twine 和 pip）配合使用，因此可以轻松集成到现有的开发工作流程中。

持续交付

持续交付 (CD) 是一种软件开发实践，在这种实践中，代码更改会自动为发布到生产环境做好准备。作为现代应用程序开发的支柱，持续交付通过在构建阶段之后将所有代码更改部署到测试环境和/或生产环境来扩展持续集成。如果实施得当，开发人员将始终拥有已通过标准化测试流程的部署就绪构建工件。

持续交付可以让开发人员自动执行测试，而不仅仅是单元测试，这样他们就可以在部署给客户之前跨多个维度验证应用程序更新。

这些测试可能包括 UI 测试、负载测试、集成测试、API 可靠性测试等。这可以帮助开发人员更全面地验证更新并抢先发现问题。使用云可以轻松且经济高效地自动创建和复制用于测试的多个环境，而以前在本地很难做到这一点。

AWS 为持续交付提供以下服务：

- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

主题

- AWS CodeDeploy
- AWS CodePipeline

AWS CodeDeploy

AWS CodeDeploy是一项完全托管的部署服务，可自动将软件部署到各种计算服务，例如亚马逊弹性计算云 (Amazon EC2) AWS Fargate、 AWS Lambda、和您的本地服务器。 AWS CodeDeploy 使您可以更轻松地快速发布新功能，帮助您避免应用程序部署期间的停机，并处理更新应用程序的复杂性。您可以使用 CodeDeploy 来自动化软件部署，从而减少对容易出错的手动操作的需求。该服务可根据您的部署需求进行扩展。

CodeDeploy 有几个与持续部署 DevOps 原则一致的好处：

- 自动部署- CodeDeploy 完全自动化软件部署，使您能够可靠、快速地进行部署。

- 集中控制 — CodeDeploy 使您能够通过 AWS 管理控制台 或轻松启动和跟踪应用程序部署的状态 AWS CLI。 CodeDeploy为您提供详细的报告，使您能够查看每个应用程序修订的部署时间和部署地点。您还可以创建推送通知以接收有关您的部署的实时更新。
- 最大限度地减少停机时间 — CodeDeploy 有助于在软件部署过程中最大限度地提高应用程序的可用性。它以增量方式引入更改，并根据可配置的规则跟踪应用程序的运行状况。如果出现错误，可以很容易地停止和回滚软件部署。
- 易于采用 — CodeDeploy 适用于任何应用程序，并在不同的平台和语言中提供相同的体验。您可以轻松地重复使用现有的设置代码。 CodeDeploy 还可以与您现有的软件发布流程或持续交付工具链（例如，、 AWS CodePipeline GitHub、Jenkins）集成。

AWS CodeDeploy 支持多种部署选项。有关更多信息，请参阅本文档的部署策略部分。

AWS CodePipeline

AWS CodePipeline是一项持续交付服务，可用于对发布软件所需的步骤进行建模、可视化和自动化。使用 AWS CodePipeline，您可以对构建代码、部署到预生产环境、测试应用程序以及将其发布到生产环境的完整发布过程进行建模。 AWS CodePipeline 然后，每次发生代码更改时，都会根据定义的工作流程构建、测试和部署您的应用程序。您可以将合作伙伴工具和您自己的自定义工具集成到发布过程的任何阶段，以形成 end-to-end持续交付解决方案。

AWS CodePipeline 有几个与持续部署 DevOps 原则一致的好处：

- 快速交付 — AWS CodePipeline 自动执行软件发布流程，使您能够快速向用户发布新功能。借CodePipeline助，您可以快速迭代反馈，更快地为用户提供新功能。
- 提高质量-通过自动执行构建、测试和发布流程， AWS CodePipeline 您可以通过一组一致的质量检查来运行所有新更改，从而提高软件更新的速度和质量。
- 易于集成 — AWS CodePipeline 可以轻松扩展以适应您的特定需求。您可以在发布过程的任何步骤中使用预先构建的插件或您自己的自定义插件。例如，您可以从中提取源代码 GitHub，使用本地 Jenkins 构建服务器，使用第三方服务运行负载测试，或者将部署信息传递到自定义操作控制面板。
- 可配置的工作流程 — AWS CodePipeline 允许您使用控制台界面、或 AWS 对软件发布过程的 AWS CLI不同阶段进行建模 SDKs。CloudFormation您可以轻松地指定要运行的测试并自定义部署应用程序及其依赖项的步骤。

部署策略数

部署策略定义了您想要如何交付软件。Organizations 根据其业务模式遵循不同的部署策略。有些人选择提供经过全面测试的软件，而另一些人可能希望用户提供反馈并让用户评估正在开发的功能（例如测试版）。下一节讨论了各种部署策略。

就地部署

在此策略中，停止每个计算资源上先前版本的应用程序，安装最新的应用程序，并启动和验证应用程序的新版本。这使应用程序部署能够在对底层基础设施的干扰降至最低。通过就地部署，您无需创建新的基础设施即可部署应用程序；但是，在这些部署期间，应用程序的可用性可能会受到影响。这种方法还可以最大限度地减少与创建新资源相关的基础架构成本和管理开销。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。就地部署可以是 all-at-once假设服务中断，也可以是滚动更新。 AWS CodeDeploy 而且 AWS Elast ic Beanstalk 为、和提供部署配置 one-at-a-time。 half-at-a-time all-at-once

蓝绿部署

蓝/绿部署（有时也称为部red/black deployment, is a technique for releasing applications by shifting traffic between two identical environments running differing versions of the application. Blue/green 署）可帮助您最大限度地减少应用程序更新期间的停机时间，降低与停机和回滚功能相关的风险。

蓝/绿部署使您可以启动应用程序的新版本（绿色）和旧版本（蓝色），并在将流量重新路由到新版本之前对其进行监控和测试，然后在问题检测到时回滚。

金丝雀部署

金丝雀部署的目的是降低部署影响工作负载的新版本的风险。该方法将逐步部署新版本，使新用户能够以较慢的方式看到新版本。当你对部署充满信心时，你需要部署它来完全取代当前版本。

线性部署

线性部署意味着流量以相等的增量移动，每次增量之间的分钟数相等。您可以从预定义的线性选项中进行选择，这些选项指定在每次增量中转移的流量百分比以及每次增量之间的分钟数。

All-at-once 部署

All-at-once部署意味着所有流量将同时从原始环境转移到替代环境。

部署策略矩阵

以下矩阵列出了亚马逊弹性容器服务 (Amazon ECS) 和 Amazon EC2 /on-prem AWS Lambda ise 支持的部署策略。

- Amazon ECS 是一项完全托管的编排服务。
- AWS Lambda 允许您在不预置或管理服务器的情况下运行代码。
- Amazon EC2 使您能够在云中运行安全、可调整大小的计算容量。

部署策略	Amazon ECS	AWS Lambda	亚马逊 EC2 /本地	
就地	✓	✓	✓	
蓝/绿	✓	✓	✓*	
金丝雀	✓	✓	X 形	
线性	✓	✓	X 形	
A ll-at-once	✓	✓	X 形	

> **Note**
>
> Blue/green deployment with EC2/on-prodise 仅适用于实 EC2 例。

AWS Elastic Beanstalk 部署策略

AWS Elastic Beanstalk支持以下类型的部署策略：

- A 在所有实例上ll-at-once执行原地部署。
- Rol ling 将实例拆分成批次，一次部署到一个批次。
- 滚动其他批次会将部署拆分为多个批次，但是对于第一批次会创建新 EC2 实例，而不是在现有 EC2 实例上部署。

- 不可变如果您需要使用新实例而不是使用现有实例进行部署。
- 流量拆分执行不可变部署，然后在预先确定的持续时间内将一定比例的流量转发到新实例。如果实例保持运行正常，则将所有流量转发到新实例并关闭旧实例。

基础设施即代码

的一个基本原则 DevOps 是以开发者对待代码的方式对待基础架构。应用程序代码具有定义的格式和语法。如果代码不是按照编程语言的规则编写的，则无法创建应用程序。代码存储在版本管理或源代码控制系统中，该系统记录了代码开发、更改和错误修复的历史记录。当代码被编译或内置到应用程序中时，我们希望创建一个一致的应用程序，并且构建是可重复且可靠的。

实践基础架构即代码意味着将同样严格的应用程序代码开发应用于基础设施配置。所有配置都应以声明方式定义，并存储在源代码控制系统中，例如 AWS CodeCommit，与应用程序代码相同。基础设施配置、协调和部署还应支持使用基础架构即代码。

传统上，基础架构是使用脚本和手动流程的组合来配置的。有时，这些脚本存储在版本控制系统中，或者逐步记录在文本文件或运行手册中。通常，编写运行手册的人不是执行这些脚本或关注运行手册的人。如果这些脚本或运行手册不经常更新，它们可能会成为部署中的佼佼者。这导致创建的新环境并不总是可重复、可靠或一致的。

相比之下， AWS 它提供了一种以创建和维护基础设施为 DevOps中心的方式。与软件开发人员编写应用程序代码的方式类似， AWS 它提供的服务能够以编程、描述性和声明性的方式创建、部署和维护基础架构。这些服务提供了严谨性、清晰度和可靠性。本 paper 中讨论的 AWS 服务是 DevOps 方法论的核心，构成了许多更高层次 AWS DevOps 的原则和实践的基础。

AWS 提供以下服务来将基础架构定义为代码。

Services

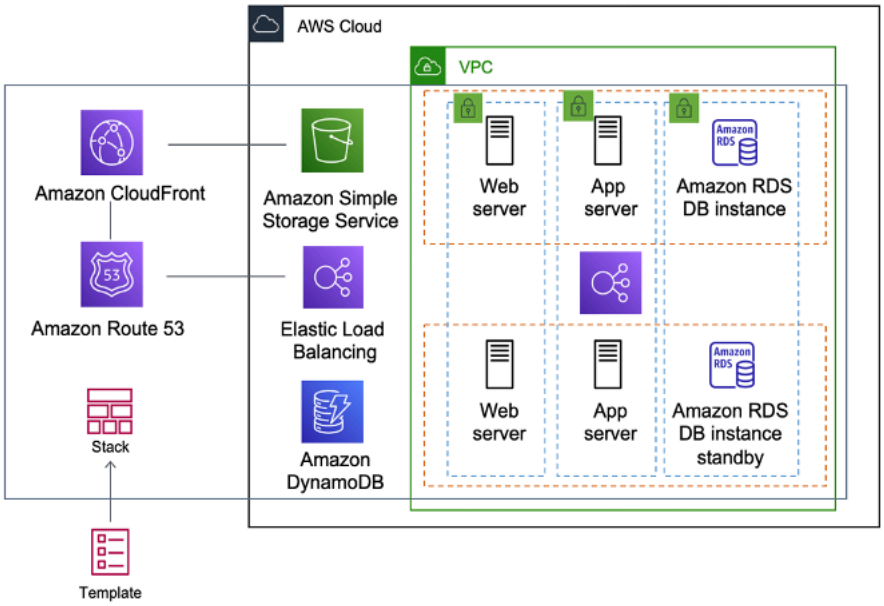
- CloudFormation
- AWS Serverless Application Model
- AWS Cloud Development Kit (AWS CDK)
- 适用于 Kubernetes 的 AWS 云开发套件
- 适用于 Terraform 的 AWS Cloud 开发套件
- AWS 云端控制 API

CloudFormation

AWS CloudFormation 是一项使开发人员能够以有序且可预测的方式创建 AWS 资源的服务。资源使用 JSON 或 YAML 格式写入文本文件。模板需要特定的语法和结构，具体取决于创建和管理的资源类

型。您可以使用任何代码编辑器（例如）以 JSON 或 YAML 格式创作资源 AWS Cloud9，将其签入版本控制系统，然后 CloudFormation 以安全、可重复的方式构建指定的服务。

CloudFormation 模板作为堆栈部署到 AWS 环境中。您可以通过 AWS 管理控制台 AWS Command Line Interface、或 CloudFormation APIs管理堆栈。如果您需要更改堆栈中正在运行的资源，则需要更新堆栈。在更改资源之前，您可以生成一个更改集，这是建议进行的更改的摘要。变更集使您能够在实施更改之前了解更改会如何影响正在运行的资源，尤其是关键资源。



AWS CloudFormation 使用一个模板创建整个环境（堆栈）

您可以使用单个模板来创建和更新整个环境，也可以使用单独的模板来管理环境中的多个图层。这使模板可以模块化，还提供了对许多组织都很重要的治理层。

当您在 CloudFormation 控制台中创建或更新堆栈时，会显示事件，显示配置的状态。如果发生错误，则默认情况下，堆栈会回滚到之前的状态。亚马逊 SNS 提供有关事件的通知。例如，您可以使用 Amazon SNS 通过电子邮件跟踪堆栈的创建和删除进度，并以编程方式与其他流程集成。

AWS CloudFormation 可以轻松组织和部署 AWS 资源集合，并允许您在配置堆栈时描述任何依赖关系或传入特殊参数。

使用 CloudFormation 模板，您可以使用各种 AWS 服务，例如亚马逊 S3、Auto Scaling、亚马逊、亚马逊 DynamoDB、亚马逊、亚马逊 CloudFront、亚马逊、Elastic Load Balancing、IAM、A OpsWorks WS 和亚马逊 VPC。 EC2 ElastiCache AWS Elastic Beanstalk有关最新支持的资源列表，请参阅AWS 资源和属性类型参考。

AWS Serverless Application Model

AWS Serverless Application Model(AWS SAM) 是一个开源框架，可用于在其上 AWS构建无服务器应用程序。

AWS SAM 与其他 AWS 服务集成，因此使用创建无服务器应用程序 AWS SAM 具有以下好处：

- 单一部署配置 — AWS SAM 便于组织相关组件和资源，并在单个堆栈上运行。您可以使用 AWS SAM 在资源之间共享配置（例如内存和超时），并将所有相关资源作为单个版本控制实体一起部署。
- 扩展 CloudFormation — 因为 AWS SAM 是的扩展 CloudFormation，所以您可以获得的可靠部署功能 CloudFormation。您可以通过在 AWS SAM 模板 CloudFormation 中使用来定义资源。
- 内置最佳实践-您可以使用 AWS SAM 来定义和部署 IaC。这使您能够使用和实施最佳实践，例如代码审查。

AWS Cloud Development Kit (AWS CDK)

AWS Cloud Development Kit (AWS CDK)是一个开源软件开发框架，用于使用熟悉的编程语言对您的云应用程序资源进行建模和配置。 AWS CDK 使您能够使用 Python TypeScript、Java 和.NET 对应用程序基础架构进行建模。开发人员可以利用其现有的集成开发环境 (IDE)，使用自动完成和内联文档等工具来加快基础架构的开发。

AWS CDK CloudFormation 在后台使用以安全、可重复的方式配置资源。构造是 CDK 代码的基本组成部分。构造代表云组件，它封装了创建该组件 CloudFormation 所需的一切。 AWS CDK 包括 A WS 构造库，其中包含代表许多 AWS 服务的构造。通过将构造组合在一起，您可以快速轻松地创建用于部署的 AWS复杂架构。

适用于 Kubernetes 的 AWS 云开发套件

适用于 Kubernetes 的 AWS Cloud Development Kit 是一个开源软件开发框架，用于使用通用编程语言定义 Kubernetes 应用程序。

使用编程语言定义应用程序后（截至本出版物发布之日，仅支持 Python 和 TypeScript ），cdk8s 会将您的应用程序描述转换为 Kubernetes 之前的 YAML。然后，在任何地方运行的任何 Kubernetes 集群都可以使用这个 YAML 文件。由于结构是用编程语言定义的，因此您可以使用该编程语言提供的丰富功能。您可以使用编程语言的抽象功能来创建自己的样板代码，并在所有部署中重复使用它。

适用于 Terraform 的 AWS Cloud 开发套件

CDK for Terraform (CD KTF) 建立在开源 JSII 库之上，允许你使用自己选择的 C#、Python、Java 或 Go 编写 Terraform 配置，同时仍然可以从 Terraform 提供程序和模块的完整生态系统中受益。TypeScript您可以将任何现有的提供程序或模块从 Terraform Registry 导入到您的应用程序中，CDKTF 将生成资源类供您使用目标编程语言进行交互。

借助 CDKTF，开发人员无需从他们熟悉的编程语言切换上下文，即可使用与应用程序业务逻辑相似的相同工具和语法来配置基础设施资源。团队可以使用熟悉的语法进行协作，同时仍然使用Terraform生态系统的强大功能，并通过已建立的Terraform部署管道部署其基础设施配置。

AWS 云端控制 API

AWS 云端控制 API是一项新 AWS 功能，它引入了一组常用的创建、读取、更新、删除和列出 (CRUDL)，可帮助开发人员 APIs 以简单一致的方式管理其云基础架构。通用云控制 API APIs 允许开发人员统一管理 AWS 和第三方服务的生命周期。

作为开发人员，您可能更愿意简化所有资源生命周期的管理方式。您可以使用具有预定义格式的 Cloud Control API 的统一资源配置模型来标准化您的云资源配置。此外，在管理资源时，您还将受益于统一的 API 行为（响应元素和错误）。

例如，您会发现，通过 Cloud Control API 显示的独立于您操作的资源的统一错误代码，可以很容易地在 CRUDL 操作期间调试错误。使用 Cloud Control API，您还会发现配置跨资源依赖关系非常简单。您也将不再需要跨多个供应商工具编写和维护自定义代码，也不 APIs 需要同时使用 AWS 和第三方资源。

自动化和工具

另一个核心理念和实践 DevOps 是自动化。自动化侧重于基础架构及其上运行的应用程序的设置、配置、部署和支持。通过使用自动化，您可以更快地以标准化和可重复的方式设置环境。取消手动流程是成功 DevOps 策略的关键。过去，服务器配置和应用程序部署主要是一个手动过程。环境变得非标准化，在出现问题时很难再现环境。

自动化的使用对于实现云的全部优势至关重要。在内部，AWS 严重依赖自动化来提供弹性和可扩展性的核心功能。

手动流程容易出错、不可靠，不足以支持敏捷业务。通常，组织可能会占用高技能的资源来提供手动配置，这样可以更好地将时间花在支持业务中其他更重要、更高价值的活动上。

现代操作环境通常依靠完全自动化来消除手动干预或访问生产环境。这包括所有软件发布、计算机配置、操作系统修补、故障排除或错误修复。许多级别的自动化实践可以一起使用，以提供更高级别的 end-to-end自动化流程。

自动化具有以下主要优点：

- 快速变化
- 提高生产力
- 可重复的配置
- 可重现的环境
- 弹性
- 自动扩缩
- 自动测试

自动化是 AWS 服务的基石，所有服务、功能和产品都得到内部支持。

主题

- AWS OpsWorks
- AWS Elastic Beanstalk
- EC2 Image Builder
- AWS Proton
- AWS Service Catalog
- AWS Cloud9

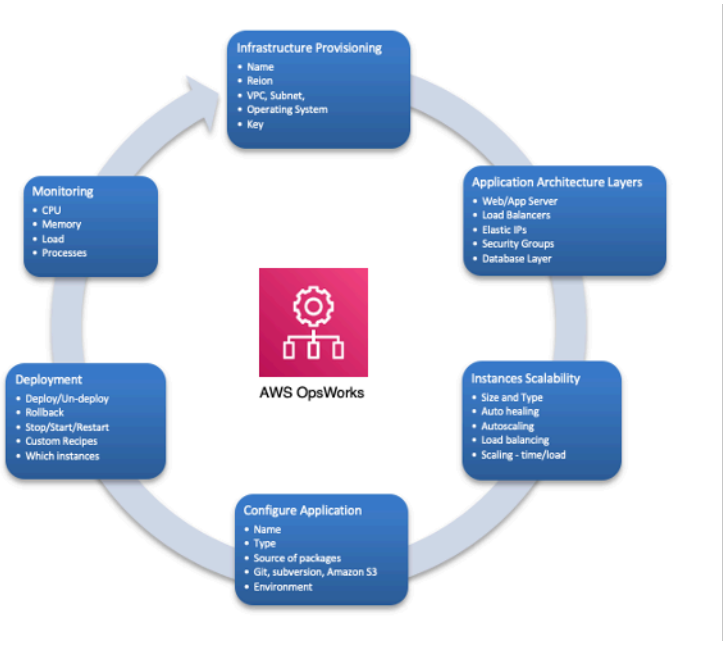
- AWS CloudShell
- 亚马逊 CodeGuru

AWS OpsWorks

AWS OpsWorks所遵循的原则 DevOps甚至远不止于 AWS Elastic Beanstalk。它可以被视为应用程序管理服务，而不仅仅是一个应用程序容器。 OpsWorks 提供更高级别的自动化，并提供其他功能，例如与配置管理软件 (Chef) 的集成和应用程序生命周期管理。您可以使用应用程序生命周期管理来定义何时设置、配置、部署、取消部署或终止资源。

为了增加灵活性 AWS OpsWorks ，您可以在可配置堆栈中定义应用程序。您也可以选择预定义的应用程序堆栈。应用程序堆栈包含您的应用程序所需的 AWS 资源的所有预配置，包括应用程序服务器、Web 服务器、数据库和负载均衡器。

应用程序堆栈被组织成架构层，因此可以独立维护堆栈。示例图层可能包括 Web 层、应用程序层和数据库层。开箱即用，AWS OpsWorks 还简化了 AWS A uto Scaling 组和 El astic L oad Balancing (ELB) 负载均衡器的设置，进一步说明了自动化 DevOps 原理。就像 AWS Elastic Beanstalk 一样 OpsWorks ，AWS 支持应用程序版本控制、持续部署和基础设施配置管理



OpsWorks 显示 DevOps 功能和架构

AWS OpsWorks 还支持监控和日志记录的 DevOps 做法（将在下一节中介绍）。监控支持由 Amazon 提供 CloudWatch。所有生命周期事件都会被记录下来，并且单独的 Chef 日志记录了所有正在运行的 Chef 配方以及任何异常。

AWS Elastic Beanstalk

AWS Elastic Beanstalk 是一项服务，用于在熟悉的服务器（例如 Apache、NGINX、Passenger 和 IIS）上部署和扩展使用 Java、.NET、PHP、Node.js、Python、Ruby、GO 和 Docker 开发的 Web 应用程序。

Elastic Beanstalk 是亚马逊 Auto Scaling 之上的一个抽象概念 EC2，它提供了其他功能，例如克隆、部署、Elasti c Beanstalk 命令行界面 (EB CLI) 以及与适用于 Visual Studio 的 AWS Toolkit、Visual Studio Code、E clipse 和 IntelliJ 集成，从而提高开发人员的工作效率，从而简化了 blue/green 部署。

EC2 Image Builder

EC2 Image Builder 是一项完全托管的 AWS 服务，可帮助您自动创建、维护、验证、共享和部署自定义、安全、 up-to-date Linux 或 Windows 自定义 AMI。 EC2 Image Builder 还可用于创建容器镜像。您可以使用 AWS 管理控制台 AWS CLI、或 APIs 在您的 AWS 账户中创建自定义图片。

EC2 Image Builder 通过提供简单的图形界面、内置的自动化功能 up-to-date和 AWS提供的安全设置，显著减少了保护图像和安全的工作量。使用 EC2 Image Builder，无需手动步骤来更新图像，也不必构建自己的自动化管道。

AWS Proton

AWS Proton使平台团队能够连接和协调开发团队在基础设施配置、代码部署、监控和更新方面所需的所有不同工具。 AWS Proton 支持自动化基础设施即代码配置和部署无服务器和基于容器的应用程序。

AWS Proton 使平台团队能够定义其基础架构和部署工具，同时为开发人员提供获取基础架构和部署代码的自助服务体验。平台团队通过 AWS Proton提供共享资源并定义应用程序堆栈，包括 CI/CD 管道和可观察性工具。然后，您可以管理哪些基础架构和部署功能可供开发人员使用。

AWS Service Catalog

AWS Service Catalog使组织能够创建和管理经批准的 IT 服务目录。 AWS这些 IT 服务可以包括从虚拟机映像、服务器、软件、数据库等到完整的多层应用程序架构的所有内容。 AWS Service Catalog 允许您集中管理已部署的 IT 服务、应用程序、资源和元数据，以实现对 IaC 模板的一致管理。

借 AWS Service Catalog助，您可以满足合规性要求，同时确保您的客户可以快速部署他们所需的经批准的 IT 服务。最终用户可在遵循组织设定约束的情况下快速部署他们所需已获得批准的 IT 服务。

AWS Cloud9

AWS Cloud9是一个基于云的 IDE，您只需使用浏览器即可编写、运行和调试代码。它包括代码编辑器、调试器和终端。 AWS Cloud9 预先打包了适用于流行编程语言（包括 Python JavaScript、PHP 等）的基本工具，因此您无需安装文件或配置开发计算机即可启动新项目。由于您的 AWS Cloud9 IDE 是基于云的，因此您可以在办公室、家中或任何地方使用联网的计算机处理项目。

AWS CloudShell

AWS CloudShell是一个基于浏览器的外壳，可以更轻松地安全地管理、浏览资源并与之交互。 AWS AWS CloudShell 已使用您的控制台凭据进行预身份验证。预先安装了常用的开发和操作工具，因此无需在本地计算机上安装或配置软件。

亚马逊 CodeGuru

Amazon CodeGuru 是一款开发者工具，可提供智能建议，以提高代码质量并识别应用程序中最昂贵的代码行。 CodeGuru 集成到您现有的软件开发工作流程中，在应用程序开发期间自动进行代码审查，持续监控应用程序在生产中的性能，并就如何提高代码质量、应用程序性能和降低总体成本提供建议和可视化线索。 CodeGuru 有两个组成部分：

- Amazon CodeGuru Reviewer — Amazon CodeGuru Reviewer 是一项自动代码审查服务，可识别关键缺陷以及与 Java 和 Python 代码编码最佳实践的偏差。它会扫描拉取请求中的代码行，并根据从主要开源项目和 Amazon 代码库中学到的标准提供智能建议。
- Amazon CodeGuru P CodeGuru rofiler — Amazon Profiler 分析应用程序运行时配置文件，并提供智能建议和可视化效果，指导开发人员如何提高代码中最相关部分的性能。

监控和可观测性

沟通和协作是 DevOps 哲学的基础。为此，反馈至关重要。此反馈由我们的监控和可观测性服务套件提供。

AWS 提供以下监控和记录服务：

主题

亚马逊 CloudWatch 指标

亚马逊 CloudWatch 指标会自动从亚马逊 EC2实例、亚马逊 EBS 卷和亚马逊 RDS 数据库 (DB) 实例等 AWS 服务收集数据。然后可以将这些指标组织为仪表板，并可以创建警报或事件来触发事件或执行 Auto Scaling 操作。

亚马逊 CloudWatch 警报

您可以根据亚马逊指标收集的指标，使用亚马逊 CloudWatch 警报设置警报。 CloudWatch 然后，警报可以向 Amazon SNS 主题发送通知，或者启动 Auto Scaling 操作。警报需要周期（评估指标的时间长度）、评估周期（最新数据点的数量）和需要警报的数据点（评估周期内的数据点数）。

Amazon CloudWatch 日志

Amazon CloudWatch Logs 是一项日志聚合和监控服务。 AWS CodeBuild CodeCommit、 ， CodeDeploy 并 CodePipeline提供与 CloudWatch 日志的集成，以便可以集中监控所有日志。此外，前面提到的服务各种其他 AWS 服务可直接与之集成 CloudWatch。

使用 CloudWatch 日志，您可以：

- 查询您的日志数据
- 监控来自 Amazon EC2 实例的日志
- 监控 AWS CloudTrail 记录的事件
- 定义日志保留政策

Amazon CloudWatch 日志见解

Amazon CloudWatch Logs Insights 会扫描您的日志，使您能够执行交互式查询和可视化。它可以理解各种日志格式并自动发现 JSON 日志中的字段。

亚马逊 CloudWatch 活动

Amazon CloudWatch Events 提供近乎实时的系统事件流，这些事件描述了 AWS 资源的变化。通过使用可快速设置的简单规则，您可以匹配事件并将事件路由到一个或多个目标函数或流。

CloudWatch 事件在发生时就会意识到操作变化。 CloudWatch 事件通过发送消息以响应环境、激活功能、进行更改和捕获状态信息，对这些操作更改做出响应，并在必要时采取纠正措施。

您可以在 Amazon CloudWatch Events 中配置规则，提醒您注意 AWS 服务变化，并将这些事件与使用 Amazon 的其他第三方系统集成 EventBridge。以下是与 CloudWatch 事件集成的 AWS DevOps 相关服务。

- App Auto Scaling 事件
- CodeBuild 事件
- CodeCommit 事件
- CodeDeploy Events
- CodePipeline Events

Amazon EventBridge

> **Note**
>
> Amazon Ev CloudWatch ents 和 Amazon Events EventBridge 是相同的底层服务，但是 API EventBridge 提供了更多功能。

Amazon EventBridge 是一种无服务器事件总线，可实现 AWS 服务、软件即服务 (SaaS) 和您的应用程序之间的集成。除了构建事件驱动的应用程序外，还 EventBridge 可用于通知来自服务的事件 CodeBuild，例如、 CodeDeploy CodePipeline、和 CodeCommit。

AWS CloudTrail

要遵循协作、沟通和透明 DevOps 的原则，重要的是要了解谁在修改您的基础架构。在中 AWS，这种透明度由提供AWS CloudTrail。所有 AWS 交互均通过 AWS API 调用进行处理，API 调用由其监控和记录 AWS CloudTrail。所有生成的日志文件都存储在您定义的 Amazon S3 存储桶中。日志文件使用 Amazon S3 服务器端加密 (SSE) 进行加密。无论是直接来自用户还是由 AWS 服务代表用户发出的，所有 API 调用都会被记录下来。许多群体都可以从 CloudTrail 日志中受益，包括负责支持的运营团队、负责治理的安全团队和负责计费的财务团队。

Amazon DevOps Guru

Amazon DevOps Guru 是一项由机器学习 (ML) 提供支持的服务，旨在轻松提高应用程序的运行性能和可用性。 DevOps Guru 可以帮助检测偏离正常运营模式的行为，因此您可以在运营问题影响客户之前很早就将其识别出来。

DevOps Guru 使用以 Amazon.com 多年的经验和卓越 AWS 运营为基础的机器学习模型来帮助识别异常应用程序行为（例如，延迟增加、错误率、资源限制等），并揭示可能导致潜在中断或服务中断的关键问题。

当 DevOps Guru 发现关键问题时，它会从大量数据源中获取相关和具体的信息，从而节省调试时间，并自动发送警报，提供相关异常的摘要以及问题发生的时间和地点的背景信息。

AWS X-Ray

AWS X-Ray帮助开发人员分析和调试生产型分布式应用程序，例如使用微服务架构构建的应用程序。借助 X-Ray，您可以了解您的应用程序及其底层服务的性能，从而识别和排除性能问题和错误的根本

原因。X-Ray 提供请求在应用程序中传输时的 end-to-end视图，并显示应用程序底层组件的地图。X-Ray 让您可以轻松地执行以下操作：

- 创建服务地图-通过跟踪向您的应用程序发出的请求，X-Ray 可以创建您的应用程序使用的服务地图。这为您提供了应用程序中服务之间的连接视图，并使您能够创建依赖关系树，在跨 AWS 可用区或区域工作时检测延迟或错误，将注意力集中在未按预期运行的服务上，等等。
- 识别错误和错误 — X-Ray 可以通过分析向应用程序提出的每个请求的响应代码，自动突出显示应用程序代码中的错误或错误。这样可以轻松调试应用程序代码，而无需重现错误或错误。
- 构建您自己的分析和可视化应用程序 — X-Ray 提供了一组查询， APIs 您可以使用这些查询来构建自己的分析和可视化应用程序，这些应用程序使用 X-Ray 记录的数据。

Amazon Managed Service for Prometheus

适用于 Prometheus 的亚马逊托管服务是一项针对与开源 Prometheus 兼容的指标的无服务器监控服务，可让您更轻松地安全地监控容器环境并发出警报。适用于 Prometheus 的亚马逊托管服务减少了开始监控亚马逊 Elastic Kubernetes Service、亚马逊弹性容器服务以及自我管理的 Kubernetes 集群中的应用程序所需的繁重工作。 AWS Fargate

Amazon Managed Grafana

Amazon Man aged Grafana 是一项完全托管的服务，具有丰富的交互式数据可视化效果，可帮助客户分析、监控多个数据源的指标、日志和跟踪并发出警报。您可以创建交互式仪表板，并通过自动扩展、高度可用和企业安全的服务与组织中的任何人共享。

沟通与协作

无论您是在组织中采用 DevOps 文化，还是正在经历 DevOps文化转型，沟通和协作都是您方法的重要组成部分。在亚马逊，我们意识到需要改变我们团队的思维方式，因此采用了 Two-Pi zza Teams 的概念。

贝索斯说：“我们努力组建规模不超过两个披萨可以喂饱的球队。” “我们称之为双披萨团队规则。”

团队越小，协作效果越好。协作非常重要，因为软件发布的速度比以往任何时候都快。而且，团队交付软件的能力可以成为您的组织在竞争中脱颖而出的一个因素。想象一下需要发布新产品功能或需要修复错误的情况。你希望这种情况尽快发生，这样你就可以缩短 go-to-market时间。你不希望转型是一个缓慢发展的过程；你想要一种敏捷的方法，让一波又一波的变革开始产生影响。

当你转向责任共担模式并开始摆脱孤立的开发方法时，团队之间的沟通也很重要。这为团队带来了所有权的概念，并改变了他们的视角，将整个过程视为一项 end-to-end冒险。您的团队不应将您的生产环境视为看不见的黑匣子。

文化转型也很重要，因为你可能正在组建一个共同的 DevOps团队，或者在你的团队中有一个 DevOps专心致志的成员。这两种方法都将责任共担引入团队。

安全性

无论您是第一次经历 DevOps 转型还是实施 DevOps 原则，都应该考虑将安全性整合到您的 DevOps 流程中。这应该是整个构建、测试部署阶段的交叉问题。

在探讨安全之前 AWS， DevOps 本 paper 着眼于 AWS 分担责任模型。

主题

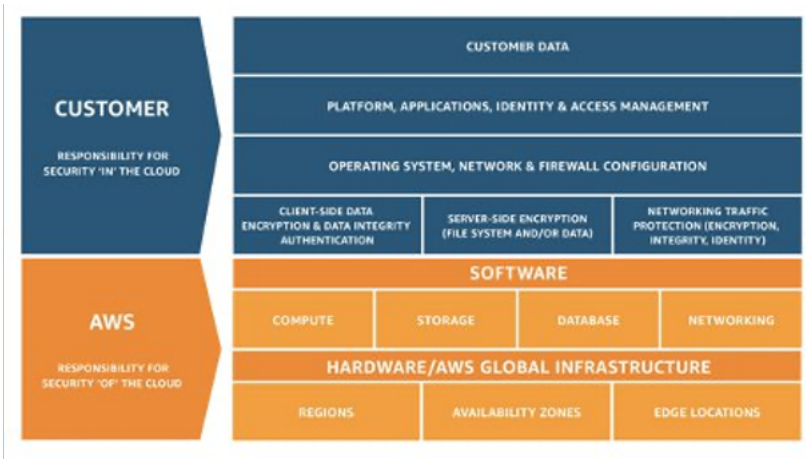
- AWS 责任共担模型
- 身份和访问管理

AWS 责任共担模型

安全是客户 AWS 和客户共同承担的责任。责任共担模型的不同部分是：

- AWS 责任 “云安全”- AWS 负责保护运行中提供的所有服务的基础设施 AWS Cloud。该基础架构由运行 AWS Cloud 服务的硬件、软件、网络和设施组成。
- 客户责任 “云端安全” — 客户责任由客户选择的 AWS Cloud 服务决定。这决定了客户必须执行的作为其安全责任一部分的配置工作量。

这种共享模式可以帮助减轻客户的运营负担，因为他们可以 AWS 操作、管理和控制从主机操作系统和虚拟化层到服务运行设施的物理安全的组件。在客户想要了解其构建环境的安全性的情况下，这一点至关重要。



AWS 责任共担模型

对于 DevOps，根据最低权限权限模型分配权限。该模型指出，“用户（或服务）应拥有完成其角色职责所需的确切访问权限——不多、不少。”

权限在 IAM 中维护。您可以使用 IAM 控制谁经过身份验证（登录）和授权（拥有权限）使用资源。

身份和访问管理

AWS Identity and Access Management(IAM) 定义了用于管理 AWS 资源访问的控制和策略。使用 IAM，您可以创建用户和群组并定义对各种 DevOps 服务的权限。

除了用户之外，各种服务可能还需要访问 AWS 资源。例如，您的 CodeBuild 项目可能需要访问权限才能将 Docker 镜像存储在亚马逊弹性容器注册表 (Amazon ECR) 中，并且需要写入亚马逊 ECR 的权限。这些类型的权限由称为服务角色的特殊类型角色定义。

IAM 是 AWS 安全基础设施的一个组成部分。借助 IAM，您可以集中管理群组、用户、服务角色和安全证书，例如密码、访问密钥和控制用户可以访问哪些 AWS 服务和资源的权限策略。IAM 策略允许您定义权限集。然后，可以将此策略附加到角色、用户或服务，以定义其权限。

您还可以使用 IAM 创建在所需 DevOps策略中广泛使用的角色。在某些情况下，以编程方式AssumeRole而不是直接获取权限是完全合理的。当服务或用户担任角色时，他们将获得临时证书，以访问他们通常无权访问的服务。

结论

为了使云之旅顺利、高效、有效，科技公司应 DevOps 遵循原则和实践。这些原则嵌入在众多 AWS 服务中 AWS，并构成了这些服务的基石，尤其是部署和监控产品中的服务。

首先，使用服务 AWS CloudFormation 或将您的基础架构定义为代码 AWS CDK。接下来，定义应用程序在诸如、 AWS CodeBuild、 AWS CodeDeploy和之类的服务的帮助下使用持续部署的方式 AWS CodeCommit。 AWS CodePipeline在应用程序级别，使用诸如 AWS Elastic Beanstalk Amazon ECS 或 Amazon Elastic Kubernetes Service（Amazon EK S）之类的容器。 OpsWorks 用于简化常见架构的配置。使用这些服务还可以轻松包含其他重要服务，例如 Auto Scaling 和 Elastic Load Balancing。

最后，使用诸如Amazon CloudWatch 之类的监控 DevOps 策略以及诸如IAM之类的可靠安全实践。

AWS 作为您的合作伙伴，您的 DevOps 原则可以为您的业务和 IT 组织带来灵活性，并加快您的云之旅。

文档修订

如需获取有关该白皮书更新的通知，请订阅 RSS 信息源。

变更	说明	日期
已更新	已更新	2023 年 4 月 7 日
更新了部分以包含新服务	更新了部分以包含新服务	2020 年 10 月 16 日
初次发布	白皮书首次发布	2014年12月1日

贡献者

本文档的贡献者包括：

- Abhra Sinha，解决方案架构师
- Anil Nadiminti，解决方案架构师
- 穆罕默德·曼苏尔，解决方案架构师
- Ajit Zadgaonkar，全球现代化技术负责人
- 胡安·拉马德里德，解决方案架构师
- 达伦·鲍尔，解决方案架构师
- Rajeswari Malladi，解决方案架构师
- Pallavi Nargund，解决方案架构师
- Bert Zahniser，解决方案架构师
- Abdullahi Olaoye，云解决方案架构师
- 穆罕默德·基斯瓦尼，软件开发经理
- Tara McCann，经理，解决方案架构师

版权声明

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。