AWS 白皮书

AWS 上的 DevOps 简介

AWS 上的 DevOps 简介: AWS 白皮书

Table of Contents

AWS 上的 DevOps 简介

发布日期：2020 年 10 月 16 日 (文档修订)

摘要

如今，企业比以往任何时候都热衷于踏上数字化转型之旅，与客户建立更深层次的联系，以实现可持续和持久的商业价值。如今各种形态和规模的组织都在通过更快、更高效地进行创新来颠覆竞争对手并进入新市场。对于这些组织来说，专注于创新和软件中断很重要，这使得简化软件交付意义重大。如果组织能缩短从构思到生产的时间，将速度和敏捷性作为优先事项，则可能会成为未来的颠覆者。

但是要成为下一个数字颠覆者，需要考虑若干因素，本白皮书重点介绍 DevOps 以及 AWS 平台中有助于提高组织高速交付应用程序和服务的能力的服务和功能。

引言

DevOps 是文化、工程实践以及模式和工具的组合，可以提高组织以更高质量高速交付应用程序和服务的能力。随着时间的推移，采用 DevOps 时出现了几种基本实践：持续集成、持续交付、基础设施即代码以及监控和日志记录。

本白皮书重点介绍了可帮助您加快 DevOps 之旅的 AWS 功能，以及 AWS 服务如何帮助消除与 DevOps 调整相关的无差别繁重任务。我们还将重点介绍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，以及如何利用基础设施即代码以一致且可重复的方式调配和管理云资源。

- 持续集成是一种软件开发实践经验，采用持续集成时，开发人员会定期将他们的代码变更合并到一个中央存储库中，之后系统会自动运行构建和测试操作。
- 持续交付是一种软件开发实践。通过持续交付，系统可以自动构建和测试代码更改，并为将其发布到生产环境做好准备。
- 基础设施即代码是采用版本控制和持续集成等代码和软件开发技术预配置和管理基础设施的规范。
- 监控和日志记录让组织能够了解应用程序和基础设施性能如何影响其产品的终端用户体验。
- 沟通与协作：通过构建工作流和分配 DevOps 的职责来建立实践以拉近团队的距离。
- 安全：应该是一个交叉问题。您的持续集成和持续交付（CI/CD）管道和相关服务应受到保护，并且应设置适当的访问控制权限。

对每一项原则的研究表明，它们与 Amazon Web Services（AWS）提供的产品有着密切的联系。

持续集成

持续集成（CI）是一种软件开发实践。采用持续集成时，开发人员会定期将代码更改合并到一个中央存储库中，之后系统会自动运行构建和测试操作。CI 有助于更快地发现并解决错误，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 提供以下服务以实现持续集成：

主题

- AWS CodeCommit
- AWS CodeBuild
- Amazon CodeArtifact

AWS CodeCommit

AWS CodeCommit 是一种用于托管私有 Git 存储库的安全、高度可扩展的托管源代码控制服务。CodeCommit 让您无需操作自己的源代码控制系统，无需预置和扩展硬件，也无需安装、配置和操作软件。您可以使用 CodeCommit 存储包括代码和二进制文件在内的任何内容，它支持 Git 的标准功能，使其能够与您基于 Git 的现有工具无缝协作。您的团队还可以使用 CodeCommit 的在线代码工具来浏览、编辑和协作处理项目。AWS CodeCommit 具有多项优势：

协作 – AWS CodeCommit 专门用于协作软件开发。您可以轻松提交、分化和合并代码，从而轻松掌控团队的项目。CodeCommit 还支持拉取请求，从而提供一种请求代码审查和与协作者讨论代码的机制。

加密 – 您可以使用 HTTPS 或 SSH（根据您的喜好）与 AWS CodeCommit 来回传输文件。您的存储库还会使用客户特定的密钥在休息时通过 AWS Key Management Service（AWS KMS）自动加密。

访问控制 – AWS CodeCommit 使用 AWS Identity and Access Management（IAM）控制和监控哪些人可以访问您的数据，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 CloudWatch 监控存储库。

高可用性和持久性 – AWS CodeCommit 会将您的存储库存储在 Amazon Simple Storage Service（Amazon S3）和 Amazon DynamoDB 中。您的加密数据会以冗余方式存储在多个设施上。该架构提高了存储库数据的可用性和耐用性。

通知和自定义脚本 – 您现在可以接收影响存储库的事件的通知。通知将以 Amazon Simple Notification Service（Amazon SNS）通知的形式出现。每个通知将包括状态消息以及指向其事件生成该通知的资

源的链接。此外，借助 AWS CodeCommit 存储库触发器，您可以使用 Amazon SNS 发送通知和创建 HTTP Webhook，或调用 AWS Lambda 功能以响应所选的存储库事件。

AWS CodeBuild

AWS CodeBuild 是一项完全托管式持续集成服务，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、管理和扩展自己的编译服务器。CodeBuild 可以使用 GitHub、GitHub Enterprise、BitBucket、AWS CodeCommit 或 Amazon S3 作为源服务器。

CodeBuild 可以持续扩展，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、功能或集成测试创建报告。这些报告提供了运行的测试用例数量以及通过或失败的测试用例数的直观视图。构建过程也可以在 Amazon Virtual Private Cloud（Amazon VPC）中运行，如果您的集成服务或数据库部署在 VPC 内，这将非常有用。

Amazon CodeArtifact

AWS CodeArtifact 是一项完全托管式构件存储库服务，组织可以使用它安全地存储、发布和共享其软件开发过程中所使用的软件包。可以将 CodeArtifact 配置为从公有构件存储库中自动获取软件包和依赖项，以便开发人员访问最新版本。

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。现在，对于软件开发团队来说，控制特定版本的开放源代码软件不含漏洞已变得至关重要。使用 CodeArtifact，您可以设置控件来强制执行上述操作。

CodeArtifact 可与常用的程序包管理器及构建工具（例如 Maven、Gradle、npm、yarn、twine 和 pip）配合使用，使其易于集成到现有开发工作流中。

持续交付

持续交付是一种软件开发实践，通过持续交付，系统可以自动为将代码更改发布到生产环境做好准备。现代应用程序开发的支柱，持续交付通过在构建阶段后将所有代码变更部署到测试环境和/或生产环境中，实现对持续集成的扩展。在正确实施时，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构件。

采用持续交付时，开发人员可以自动执行单元测试以外的测试，这样他们就可以在部署到客户环境前跨多个维度对应用程序更新进行验证。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。这有助于开发人员更全面地验证更新并抢先发现其中的问题。借助云，开发人员可轻松高效地自动创建和复制多个用于测试的环境，而这一点以前在本地很难实现。

AWS 提供以下服务以实现持续交付：

- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

主题

AWS CodeDeploy

AWS CodeDeploy 是一项完全托管式部署服务，可自动将软件部署到各种计算服务，例如 Amazon Elastic Compute Cloud（Amazon EC2）、AWS Fargate、AWS Lambda 和您的本地服务器。AWS CodeDeploy 使您能够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，并处理复杂的应用程序更新。您可以使用 CodeDeploy 自动执行软件部署，消除容易出错的手动操作。服务根据您的部署需求进行扩展。

CodeDeploy 具有几项与 DevOps 持续部署原则一致的好处：

自动化部署：CodeDeploy 可实现软件部署的完全自动化，使您能够快速可靠地部署。

集中控制：借助 CodeDeploy，您可以通过 AWS 管理控制台或 AWS CLI 轻松启动和跟踪应用程序部署的状态。CodeDeploy 会为您提供一份详细的报告，使您能够查看每个应用程序修订的部署时间和位置。您还可以创建推送通知以接收与您的部署有关的实时更新。

最大限度地减少停机时间：CodeDeploy 有助于最大限度地提高软件部署流程期间的应用程序可用性。它会逐步引入更改，并根据可配置规则跟踪应用程序运行状况。即使出错，您也可以轻松停止和回滚软件部署。

易于采用：CodeDeploy 可与任何应用程序配合使用，并在不同的平台和语言中提供相同的体验。您可以轻松重用现有的设置代码。CodeDeploy 还能与您现有的软件发布过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，请参阅部署策略。

AWS CodePipeline

AWS CodePipeline 是一种持续交付服务，它可以建模、可视化和自动执行发布软件所需的步骤。借助 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流程进行建模。然后，每当代码发生更改时，AWS CodePipeline 都会根据定义的工作流构建、测试和部署应用程序。您可以将 APN 合作伙伴工具和自己的自定义工具集成到发布流程的任何阶段，以形成一个端到端的持续交付解决方案。

AWS CodePipeline 具有几项符合 DevOps 持续部署原则的好处：

快速交付：AWS CodePipeline 可帮助您实现软件发布流程的自动化，从而使您能够向用户快速发布新功能。凭借 CodePipeline，您可以快速迭代反馈并将新功能更快地交付给客户。

提高了质量：通过实现构建、测试和发布流程的自动化，AWS CodePipeline 让您能够针对所有新更改运行一系列一致的质量检查，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。您可以在发布流程的任何阶段使用我们的预建插件或您自己的自定义插件。例如，您可以从 GitHub 拉取您的源代码，使用本地 Jenkins 构建服务器，使用第三方服务运行负载测试，或将部署信息传递到您的自定义操作控制面板。

可配置的工作流：借助 AWS CodePipeline，您可以使用控制台界面、AWS CLI、AWS CloudFormation 或 AWS SDK 为软件发布流程的不同阶段建模。您可以轻松指定要运行的测试，并自定义部署应用程序及其依赖项的步骤。

部署策略

部署策略定义了您希望如何交付软件。组织根据其业务模式遵循不同的部署策略。有些组织可能选择交付经过全面测试的软件，而另一些组织可能希望其用户提供反馈并让其用户评估开发中的功能（例如 beta 版本）。在下一节中，我们将讨论各种部署策略。

主题

- 就地部署
- 蓝/绿部署
- Canary 部署
- 线性部署
- 一次性部署

就地部署

在此策略中，按以下方法完成部署：停止部署组中每个实例上的应用程序，安装最新的应用程序修订版，然后启动并验证应用程序的新版本。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。就地部署可以是一次性的（假设服务中断），也可以滚动更新方式完成。AWS CodeDeploy 和 AWS Elastic Beanstalk 提供了一次部署一个、一半和全部的部署配置。这些适用于就地部署的部署策略在蓝/绿部署中同样可用。

蓝/绿部署

蓝/绿（有时称为红黑）部署是一项通过转移运行不同版本的应用程序的两个相同环境之间的流量来释放应用程序的技术。蓝/绿部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，从而降低停机和回滚功能带来的风险。蓝/绿部署使您能够与旧版本（蓝色）一起启动应用程序的新版本（绿色），并在将流量重新路由到新版本之前对其进行监控和测试，从而在检测到问题时回滚。

Canary 部署

流量在两次增量中转移。Canary 版本部署是一种更有效地规避风险的蓝/绿策略，其中使用了分阶段方法。这可以是两个步骤，也可以是线性的，即部署新的应用程序代码并公开以供试用，然后在接受后将其发布到环境的其余部分或以线性方式推出。

线性部署

线性部署意味着流量使用相等的增量转移，在每次增量之间的分钟数相同。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。

一次性部署

一次性部署意味着所有流量一次性地从原始环境转移到替换环境。

部署策略矩阵

以下矩阵列出了 Amazon Elastic Container Service（Amazon ECS）、AWS Lambda 和 Amazon EC2/本地部署支持的部署策略。

- Amazon ECS 是一项完全托管式编排服务。
- 利用 AWS Lambda，您可以运行代码而无需预配置或管理服务器。
- Amazon EC2 使您能够在云中运行安全、可调整规模的计算容量。

	A	B，	C	D
1	部署策略矩阵	Amazon ECS	AWS Lambda	Amazon EC2/本地部署
2	就地	✓	✓	✓
3	蓝/绿	✓	✓	✓*
4	Canary	✓	✓	X
5	线性	✓	✓	X
6	一次性	✓	✓	X

> **Note**
>
> 使用 EC2/本地部署的蓝/绿部署仅适用于 EC2 实例。

AWS Elastic Beanstalk 部署策略

AWS Elastic Beanstalk 支持以下类型的部署策略：

- 一次性：在所有实例上执行就地部署。
- 滚动：将实例拆分为多个批次，一次部署到一个批次。

- 附加批次滚动部署：将部署拆分为多个批次，但第一批将创建新的 EC2 实例，而不是部署在现有 EC2 实例上。
- 不可改变：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，然后在预先确定的时段内将一定百分比的流量转发到新实例。如果这些实例保持运行正常，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

DevOps 的基本原则是对待基础设施就像开发人员对待代码一样。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，该系统会记录代码开发、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后，我们期望创建一个一致的应用程序，并且构建是可重复且可靠的。

基础设施即代码 实践意味着将同样严格的应用程序代码开发应用于基础设施调配。所有配置都应以声明方式定义，并存储在源代码控制系统（例如 AWS CodeCommit）中，与应用程序代码相同。基础设施调配、编排和部署还应支持使用基础设施即代码。

以往，基础设施是使用脚本和手动过程的组合来调配的。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完成运行手册的那个人。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终结者。这会导致创建的新环境并非总是可重复、可靠或一致的。

与前面不同的是，AWS 提供了一种以 DevOps 为重点的创建和维护基础设施的方法。与软件开发人员编写应用程序代码的方式类似，AWS 提供允许以编程、描述和声明方式创建、部署和维护基础设施的服务。这些服务具有严谨性、明确性和可靠性。本白皮书中讨论的 AWS 服务是 DevOps 方法的核心，是众多更高级别的 AWS DevOps 原则和实践的基础。

AWS 提供以下服务来定义基础设施即代码。

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- 适用于 Kubernetes 的 AWS Cloud Development Kit

AWS CloudFormation

AWS CloudFormation 是一项服务，使开发人员能够以有序且可预测的方式创建 AWS 资源。资源是使用 JavaScript 对象表示法（JSON）或 Yet Another Markup Language（YAML）格式在文本文件中编写的。模板需要特定的语法和结构，具体取决于要创建和管理的资源类型。您可以使用任何代码编辑器（如 AWS Cloud9）在 JSON 或 YAML 中创作资源，将其签入版本控制系统，然后 CloudFormation 会以安全、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 命令行界面或 AWS CloudFormation API 管理堆栈。如果您需要更改堆栈中运行的资源，则可更新堆栈。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。利用更改集，您可以在实施更改之前，了解更改可能会对运行的资源（特别是关键资源）造成的影响。

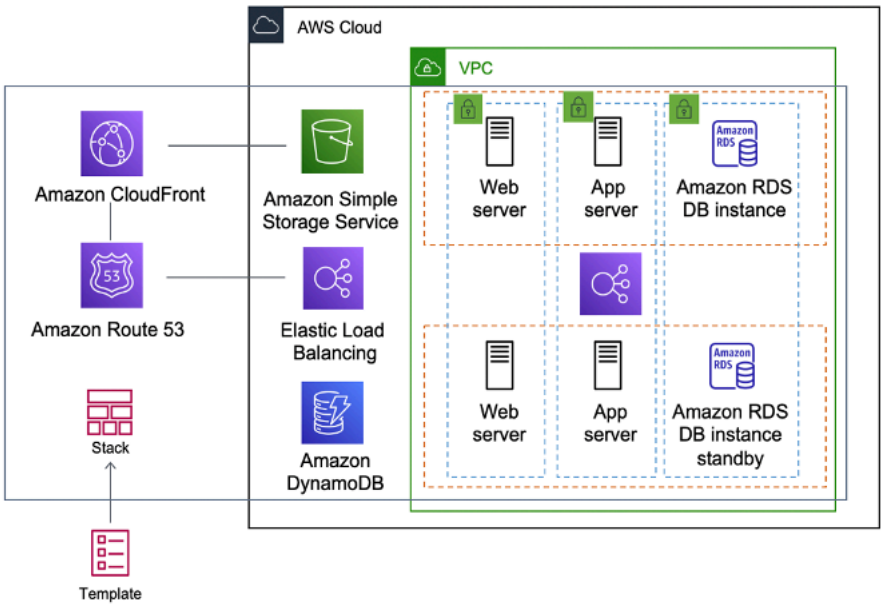

图 1 – AWS CloudFormation 通过一个模板工作流创建整个环境（堆栈）

您可以使用单个模板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层。这使模板可以模块化，并且还提供了对许多组织都很重要的治理层。

当您在控制台中创建或更新堆栈时，将会出现显示配置状态的事件。如果发生错误，默认情况下，堆栈将回滚到之前的状态。Amazon Simple Notification Service（Amazon SNS）提供事件通知。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，以及以编程方式和其他流程集成。

AWS CloudFormation 使组织和部署 AWS 资源集变得更为轻松，并让您能够描述任何依赖关系或配置堆栈时传入的特殊参数。

借助 CloudFormation 模板，您可以使用各种各样的 AWS 服务，例如 Amazon S3、Auto Scaling、Amazon CloudFront、Amazon DynamoDB、Amazon EC2、Amazon ElastiCache、AWS Elastic Beanstalk、Elastic Load Balancing、IAM、AWS OpsWorks 和 Amazon VPC。有关受支持资源的最新列表，请参阅 AWS 资源和属性类型参考。

AWS Cloud Development Kit

AWS Cloud Development Kit (AWS CDK) 是一种开源软件开发框架，用以使用熟悉的编程语言模拟和预置云应用程序资源。借助 AWS CDK，您可以使用 TypeScript、Python、Java 和 .NET 为应用程序基础设施建模。开发人员可以利用其现有的集成开发环境（IDE），利用自动完成和内联文档等工具来加快基础设施的开发。

AWS CDK 在后台利用 AWS CloudFormation 以安全、可重复的方式预置资源。结构是 CDK 代码的基本构建块。结构代表云组件，并封装 AWS CloudFormation 创建该组件所需的一切。AWS CDK 包含 AWS 构造库，其中包含代表许多 AWS 服务的结构。通过将结构组合在一起，您可以快速轻松地创建复杂的架构以在 AWS 中进行部署。

适用于 Kubernetes 的 AWS Cloud Development Kit

适用于 Kubernetes 的 AWS Cloud Development Kit（cdk8s）是一种开源软件开发框架，用于利用通用编程语言定义 Kubernetes 应用程序。

一旦您用编程语言定义了应用程序（截至发布之日，仅支持 Python 和 TypeScript），cdk8s 即会将您的应用程序描述转换为 Kubernetes 之前的 YML 版本。然后，在任何地方运行的任何 Kubernetes 集群都可使用这个 YML 文件。由于结构是用编程语言定义的，因此您可以使用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。

自动化

DevOps 的另一个核心理念和实践是自动化。自动化侧重于基础设施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，您可以以标准化和可重复的方式更快地设置环境。移除手动流程是 DevOps 策略取得成功的关键。过去，服务器配置和应用程序部署主要是手动过程。环境变得不合标准，难以再现出问题时的环境。

使用自动化对于实现云的全部优势至关重要。在内部，AWS 在很大程度上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，并且不足以支持敏捷的业务。通常，组织可能会占用技能精湛的资源来提供手动配置，而这些资源本可以将时间花在支持业务中其他更关键且价值更高的活动上。

现代运营环境通常依靠完全自动化来消除人工干预或对生产环境的访问。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。很多级别的自动化实践可以结合使用，以提供更高级别的端到端自动化流程。

自动化具有以下主要优势：

- 快速更改
- 工作效率提高
- 可重复的配置
- 可重现的环境
- 利用弹性
- 利用弹性伸缩
- 自动化测试

自动化是 AWS 服务的基石，在所有服务、功能和产品内部都受支持。

主题

- AWS OpsWorks
- AWS Elastic Beanstalk

AWS OpsWorks

AWS OpsWorks 比 AWS Elastic Beanstalk 更进一步地体现了 DevOps 的原则。可以将其视为应用程序管理服务，而不仅仅是应用程序容器。通过与配置管理软件（Chef）集成和应用程序生命周期管理

等附加功能，AWS OpsWorks 可实现更高级别的自动化。您可以使用应用程序生命周期管理来定义设置、配置、部署、取消部署或关闭资源的时间。

为了提高灵活度，AWS OpsWorks 让您在可配置堆栈中定义应用程序。您还可以选择预定义的应用程序堆栈。应用程序堆栈包含应用程序所需的所有 AWS 资源调配，包括应用程序服务器、Web 服务器、数据库和负载均衡器。

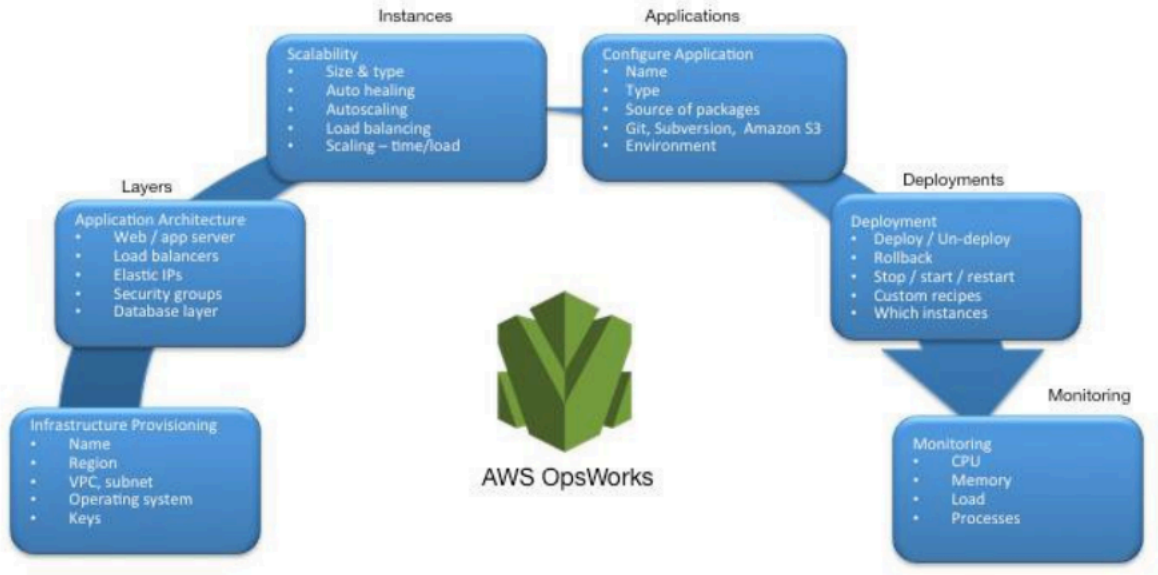

图 2 – 展示 DevOps 功能和架构的 AWS OpsWorks

应用程序堆栈被整理到架构层中，以便可以独立维护堆栈。示例层可能包括 Web 层、应用程序层和数据库层。开箱即用的 AWS OpsWorks 还简化了 Auto Scaling 组和 Elastic Load Balancing 负载均衡器的设置，进一步说明了 DevOps 自动化原则。就像 AWS Elastic Beanstalk 一样，AWS OpsWorks 支持应用程序版本控制、持续部署和基础设施配置管理。

AWS OpsWorks 还支持 DevOps 监控和日志记录实践（将在下一节中介绍）。监控支持由 Amazon CloudWatch 提供。所有生命周期事件都会被记录下来，一个单独的 Chef 日志会记录已运行的所有 Chef 配方以及所有异常。

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 Amazon EC2、Auto Scaling 之上的抽象概念，通过提供附加功能（例如克隆、蓝/绿部署、Elastic Beanstalk 命令行界面（eb cli），以及与 AWS Toolkit for Visual Studio、Visual Studio Code、Eclipse 和 IntelliJ 的集成）来简化部署，以提高开发人员的工作效率。

监控和日志记录

沟通和协作是 DevOps 理念的基础。要促进沟通和协作，反馈至关重要。在 AWS 中，反馈由两个核心服务提供：Amazon CloudWatch 和 AWS CloudTrail。它们共同提供了强大的监控、警报和审计基础设施，因此开发人员和运营团队可以紧密、透明地协同工作。

AWS 提供以下监控和日志记录服务：

主题

Amazon CloudWatch

Amazon CloudWatch 指标会自动从 Amazon EC2 实例、Amazon EBS 卷和 Amazon RDS 数据库实例等 AWS 服务中收集数据。然后，可以将这些指标整理为控制面板，就可以创建警报或事件来触发事件或执行 Auto Scaling 操作。

Amazon CloudWatch 警报

您可以根据 Amazon CloudWatch 指标收集的指标设置警报。然后，警报可以向 Amazon Simple Notification Service（Amazon SNS）主题发送通知或启动 Auto Scaling 操作。警报需要周期（评估指标的时间长度）、评估期（最近数据点的数量）和要报警的数据点（评估期内的数据点数）。

Amazon CloudWatch Logs

Amazon CloudWatch Logs 是一项日志聚合和监控服务。AWS CodeBuild、CodeCommit、CodeDeploy 和 CodePipeline 提供与 CloudWatch Logs 的集成，以便可以集中监控所有日志。此外，前面提到的各种其他 AWS 服务还提供与 CloudWatch 的直接集成。

使用 CloudWatch Logs，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
- 定义日志保留策略

Amazon CloudWatch Logs Insights

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 Logs 中的字段。

Amazon CloudWatch Events

Amazon CloudWatch Events 提供几乎实时的系统事件流，这些事件描述 AWS 资源的更改。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 Events 会在发生操作更改时感知到这些更改。CloudWatch Events 将响应这些操作更改并在必要时采取纠正措施，方式是发送消息以响应环境、激活函数、进行更改并捕获状态信息。

您可以在 CloudWatch Events 中配置规则以提醒您注意 AWS 服务的变化，并使用 Amazon EventBridge 将这些事件与其他第三方系统集成。以下是与 CloudWatch Events 集成的 AWS DevOps 相关服务。

- Application Auto Scaling Events
- CodeBuild Events
- CodeCommit Events
- CodeDeploy Events
- CodePipeline Events

Amazon EventBridge

Amazon CloudWatch Events 和 EventBridge 是相同的底层服务和 API，但 EventBridge 提供了更多功能。

Amazon EventBridge 是一个无服务器事件总线，它支持在 AWS 服务、软件即服务（SaaS）和您的应用程序之间进行集成。除了构建事件驱动的应用程序之外，EventBridge 还可用于通知来自 CodeBuild、CodeDeploy、CodePipeline 和 CodeCommit 等服务的事件。

AWS CloudTrail

为了接受 DevOps 协作、沟通和透明的原则，了解谁在修改您的基础设施非常重要。在 AWS 中，这种透明度是由 AWS CloudTrail 服务提供的。所有 AWS 交互都通过受 AWS CloudTrail 监控和记录的 AWS API 调用来处理。所有生成的日志文件都存储在您定义的 Amazon S3 存储桶中。日志文件将使用 Amazon S3 服务器端加密（SSE）进行加密。所有 API 调用都会被记录下来，无论它们是直接来自用户还是由 AWS 服务代表用户。许多团体都可以从 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以及负责计费的财务团队。

沟通与合作

无论您是在组织中采用 DevOps 文化，还是正在进行 DevOps 文化转型沟通，协作都是您方法的重要组成部分。在亚马逊，我们意识到有必要改变团队的思维方式，因此采用了双披萨团队 的概念。

主题

双披萨团队

Bezos 说：“我们试图在团队人数方面采用‘双披萨’原则。”“我们称之为双披萨团队规则。”

团队越小，协作越好。协作也非常重要，因为软件版本的推出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组织与竞争对手的区分因素。想象一下需要发布新产品功能或需要修复错误的情况，您会希望尽快发生这种情况，以便缩短上市时间。这也很重要，因为您不希望转型成为一个缓慢的过程，而不是一种敏捷的方法，在这种方法中，变革浪潮开始产生影响。

团队之间的沟通也很重要：我们朝着责任共担模式迈进，并开始摆脱孤立的开发方法。这在团队中引入了所有权的概念，并改变了他们的观点，将其视为端到端。您的团队不应该把您的生产环境看作是黑匣子，在那里他们没有可见性。

文化转型也很重要，因为您可能正在建立一个通用的 DevOps 团队，另一种方法是让您的团队中有一个或多个专注于 DevOps 的成员。这两种方法都在团队中引入了责任共担。

安全性

无论您是要进行 DevOps 转型还是第一次实施 DevOps 原则，都应该考虑将安全性集成到您的 DevOps 流程中。这应该是整个构建、测试部署阶段的交叉问题。

在讨论 AWS 上 DevOps 中的安全性之前，让我们先看一下 AWS 责任共担模式。

主题

- AWS 责任共担模式
- Identity and Access Management

AWS 责任共担模式

确保安全性是 AWS 和客户的共同责任。责任共担模式的不同部分解释如下：

- AWS 负责“云本身的安全”– AWS 负责保护运行所有 AWS 云服务的基础设施。该基础设施由运行 AWS 云服务的硬件、软件、网络和设施组成。
- 客户负责“云内部的安全”– 客户责任由客户所选的 AWS 云服务确定。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。

这种责任共担模式可以减轻客户的运营负担，因为 AWS 运行、管理和控制从主机操作系统和虚拟化层到服务运营所在设施的物理安全性的组件。当客户想要了解其构建环境的安全性时，这一点至关重要。

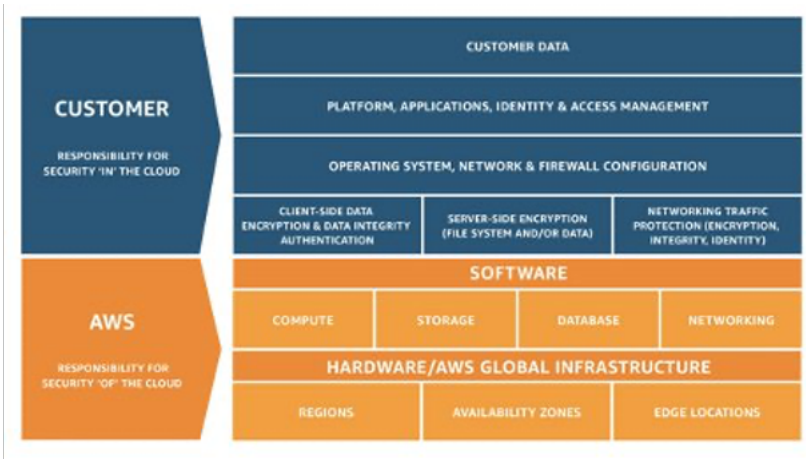

图 3 – AWS 责任共担模式

Identity and Access Management

AWS Identity and Access Management（IAM）定义了用于管理对 AWS 资源的访问的控制和策略。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在 Amazon Elastic Container Registry（Amazon ECR） 中存储 Docker 镜像，并且需要权限才能写入 Amazon ECR。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，您可以集中管理组、用户、服务角色和安全凭证（例如密码、访问密钥），以及对用户可以访问何种 AWS 服务和资源进行控制的许可策略。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 创建在所需的 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，他们将获得临时凭证来访问通常没有访问权限的服务。

总结

为了使云之旅顺畅、高效和有效，科技公司应采用 DevOps 原则和实践。这些原则已嵌入到 AWS 平台中。事实上，它们构成了众多 AWS 服务的基石，尤其是部署和监控服务方面。

首先，使用服务 AWS CloudFormation或 AWS Cloud Development Kit (AWS CDK) 定义基础设施即代码。接下来，借助 AWS CodeBuild、AWS CodeDeploy、AWS CodePipeline 和 AWS CodeCommit 等服务定义您的应用程序将以哪种方式使用持续部署。在应用程序级别，使用 AWS Elastic Beanstalk、Amazon Elastic Container Service（Amazon ECS）或 Amazon Elastic Kubernetes Service（Amazon EKS）之类的容器以及 AWS OpsWorks 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 DevOps 监控策略（例如 Amazon CloudWatch）和可靠的安全实践（例如 AWS IAM）。

将 AWS 作为您的合作伙伴，您的 DevOps 原则可为您的企业和 IT 组织带来敏捷性，并加快您的云之旅。

文档修订

要获得有关此白皮书的更新通知，请订阅 RSS 源。

更新-历史记录-更改	更新-历史记录-描述	更新-历史记录-日期
已恢复缺失的“贡献者”部分	已恢复缺失的“贡献者”部分和细微的文本更改	2020 年 11 月 21 日
更新了相关章节以包含新服务	更新了相关章节以包含新服务	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：

- Muhammad Mansoor，解决方案构架师
- Ajit Zadgaonkar，全球现代化技术负责人
- Juan Lamadrid – 解决方案构架师
- Darren Ball – 解决方案构架师
- Rajeswari Malladi – 解决方案构架师
- Pallavi Nargund – 解决方案构架师
- Bert Zahniser – 解决方案构架师
- Abdullahi Olaoye – 云科技解决方案构架师
- Mohamed Kiswani – 软件开发经理
- Tara McCann – 经理解决方案构架师

声明

客户负责对本文档中的信息进行独立评估判断。本文档：(a) 仅供参考；(b) 代表当前提供的 AWS 产品和实践，如有更改，恕不另行通知；并且 (c) AWS 及其附属机构、供应商或许可方不做任何承诺或保证。AWS 产品或服务“按原样”提供，不提供任何形式的保证、陈述或条件，无论是明示还是暗示。AWS 对其客户承担的责任和义务受 AWS 协议制约，本文档不是 AWS 与客户直接的协议的一部分，也不构成对该协议的修改。