aws

为您的 Amazon EKS 集群选择合适的 GitOps 工具

AWS 规范性指导

AWS 规范性指导: 为您的 Amazon EKS 集群选择合适的 GitOps 工具

Table of Contents

为您的 Amazon EKS 集群选择合适的 GitOps 工具

Pradip Kumar Pandey 和 Pratap Kumar Nanda，Amazon Web Services ()AWS

2025 年 4 月（文档历史记录）

在快速发展的云原生技术格局中， GitOps 它已成为管理和部署应用程序和基础设施的强大方法。如果您使用的是 Amazon Elastic Kubernetes Service（Amazon EKS） GitOps ，则实施原则可以显著增强您的部署流程，提高可靠性并简化操作。有各种各样的 GitOps 工具可供选择，为你的 EKS 集群选择合适的工具是一个关键的决定，它可能会影响团队的效率和 DevOps实践的整体成功。

为您的 Amazon EKS 环境选择合适的 GitOps 工具需要仔细考虑各种因素，包括您的具体要求、团队专业知识、可扩展性需求以及与现有工具的集成能力 AWS 服务。每种工具都有自己的功能、优势和潜在局限性，因此必须使您的选择与组织的目标和运营环境保持一致。

本指南探讨了为Amazon EKS选择 GitOps 工具时的关键注意事项，比较了常用的选项，并提供了一些见解来帮助您做出明智的决定。它涵盖了九种流行的 GitOps 工具：

- Argo CD
- 通量
- 编织 GitOps
- Jenkins X
- GitLab CI/CD
- Spinnaker
- 牧场主舰队
- Codefresh
- Pulumi

目标业务成果

以下列表讨论了选择在开发和运营流程中实施 GitOps 原则的工具时的潜在目标和成果。

与 Amazon EKS 无缝集成

您的 GitOps 工具应与 Amazon EKS 顺利集成，并与 Amazon EKS 特有的功能和优化兼容。

- Amazon EKS 原生支持：寻找能够为 Amazon EKS 提供内置支持的工具，包括简单的集群连接和管理。
- AWS 服务 集成：确保该工具可以与其他工具进行交互， AWS 服务 例如 AWS Identity and Access Management (IAM)、亚马逊弹性容器注册表 (Amazon ECR)Container Registry 和亚马逊。CloudWatch
- Amazon EKS 附加组件兼容性：确认该工具支持 Amazon EKS 插件并且可以有效地对其进行管理。

可扩展性和性能

您的 GitOps 工具应该能够处理从小型集群到大型多集群环境的 Amazon EKS 操作规模。

- 资源效率：评估工具的资源消耗及其对集群性能的影响。
- 大规模运营：评估该工具同时管理多个应用程序和集群的能力。
- 负载下的性能：考虑该工具在高频更新和大规模部署期间的性能。

安全与合规

安全功能和合规能力至关重要，尤其是在受监管的行业或处理敏感数据时。

- 访问控制：寻找与 IAM 集成的强大基于角色的访问控制 (RBAC) 功能。
- 密钥管理：评估该工具如何处理敏感信息以及如何与AWS Secrets Manager或其他解决方案集成。
- 审计跟踪：确保该工具提供全面的日志记录和审计功能，以实现合规性和故障排除。
- 安全扫描：考虑为部署中的漏洞提供内置安全扫描的工具。

易用性和学习曲线

该工具应易于使用，并与团队的技能保持一致，以确保快速采用和高效使用。

- 用户界面：评估命令行界面 (CLI) 和图形用户界面 (GUI) 功能的直观性。
- 文档质量：查找全面的 up-to-date文档和教程。
- 学习资源：考虑培训材料、课程和社区资源的可用性。

社区和网络支持

强大的社区和网络可以提供宝贵的资源、插件和长期可持续性。

- 积极开发：检查更新频率和维护者的响应能力。
- 社区规模：考虑用户社区的规模和活动，以获得支持和知识共享。
- 第三方集成：评估插件的可用性以及与堆栈中其他工具的集成。

多集群管理功能

如果您有多个 EKS 集群，那么高效管理它们的能力至关重要。

- 集中管理：寻找允许从单个控制平面管理多个集群的功能。
- 集群联合：考虑支持多集群应用程序的 Kubernetes 联合的工具。
- 环境平等：评估该工具在开发、暂存和生产等不同环境中保持一致性的程度。

可观察性和监测

该工具应提供对部署状态和集群运行状况的清晰见解。

- 部署可见性：寻找能够清晰查看部署状态和历史记录的功能。
- 与监控工具集成：考虑一下该工具与 Prometheus 和 Grafana 等常用监控解决方案的集成程度。
- 警报功能：评估该工具设置和管理部署问题或偏差警报的能力。

灵活性和定制

能够根据您的特定工作流程和要求调整工具对于长期满意度非常重要。

- 可扩展性：寻找插件架构或 APIs 使您能够扩展工具功能的架构。
- 自定义资源支持：确认该工具可以有效地处理自定义 Kubernetes 资源。
- 工作流程自定义：评估根据团队需求定制 GitOps工作流程的难易程度。

持续交付和渐进式部署支持

高级部署策略通常对于最大限度地降低风险和确保顺利更新至关重要。

- Canary 部署：寻找对 Canary 版本的内置支持。
- Blue/green deployments: Assess the tool's capabilities for blue/green部署策略。
- 回滚机制：确保强大的 easy-to-use回滚功能，以便从失败的部署中快速恢复。

成本效益和资源利用率

考虑采用和维护该工具的总体成本，包括直接成本和间接成本。

- 许可成本：将开源选项与商业解决方案进行比较，并考虑支持和企业功能。
- 运营开销：评估管理和维护方面的额外运营成本。
- 资源消耗：根据所需的计算和存储资源评估工具的效率。

通过仔细考虑这些结果及其各个方面，您可以就最适合您的 EKS 集群的 GitOps 工具做出明智的决定，并确保该工具与组织的需求、能力和长期战略保持一致。

GitOps 适用于 EKS 集群的工具

目前市场上有几种适用于 Kubernetes 的 GitOps 工具。以下是一些最广泛使用的选项的列表：

- Argo CD
- 通量
- 编织 GitOps
- Jenkins X
- GitLab CI/CD
- Spinnaker
- 牧场主舰队
- Codefresh
- Pulumi

点击链接查看有关这些工具如何实施 GitOps实践的详细信息。每种工具都有其优势和用例。选择取决于诸如您的具体要求、现有基础架构、团队专业知识和所需功能等因素。根据组织的需求和 Kubernetes 环境的复杂性来评估这些工具非常重要。

Argo CD

Argo CD 是一款广泛使用的 Kubernetes GitOps 持续交付 (CD) 工具，它符合多个关键原则。 GitOps

GitOps 支持

区域图	工具功能
声明式配置	Argo CD 使用存储在 Git 存储库中的声明性配置。应用程序和基础架构的所需状态在 YAML 文件中定义。这些配置描述的是应该部署的内容，而不是如何部署它们。
版本控制系统作为单一事实来源	Git 存储库是整个系统的单一事实来源。对应用程序和基础架构的所有更改都是通过 Git 进行的。这样可以确保完整的审计跟踪，并能够回滚到之前的任何状态。

区域图	工具功能
自动同步	Argo CD 会持续监控 Git 存储库中的更改。当检测到更改时，它会自动将集群的实际状态与 Git 中定义的所需状态同步。这样可以确保集群始终反映存储库中描述的状态。
Kubernetes 原生	Argo CD 专为 Kubernetes 环境而设计。它利用 Kubernetes 中的声明性质和自定义资源来管理应用程序。
自我修复和漂移检测	Argo CD 定期将集群的实时状态与 Git 中的所需状态进行比较。如果它检测到任何偏差（实际状态和所需状态之间的差异），它可以自动纠正这些差异。
多集群和多租户支持	Argo CD 可以从一个实例管理多个 Kubernetes 集群。它支持多租户，因此不同的团队可以独立管理他们的应用程序。
应用程序定义	Argo CD 中的应用程序是使用应用程序 CRD（自定义资源定义）定义的。这允许使用 Kubernetes 原生的方式来定义应该部署什么以及如何部署。
部署和发布分离	Argo CD 将代码部署与向用户发布代码区分开来。这是通过各种部署策略实现的，例如 blue/green 或金丝雀部署。
可观察性和可审计性	Argo CD 提供了用于观察应用程序和集群状态的 Web 用户界面和 CLI。所有操作都记录在案，以提供对更改和部署的清晰审计跟踪。
安全和 RBAC	Argo CD 与 Kubernetes 基于角色的访问控制 (RBAC) 集成。它支持用于身份验证和授权的单点登录集成。

区域图	工具功能
可插拔架构	Argo CD 支持各种源代码控制管理系统、Helm 图表、Kustomize 和其他 Kubernetes 清单格式。这种灵活性使其能够适应不同的环境和工作流程。
持续交付 (CD)	尽管 Argo CD 专注于持续交付，但它可以与持续集成 (CI) 工具集成以创建完整的 CI/CD 管道。

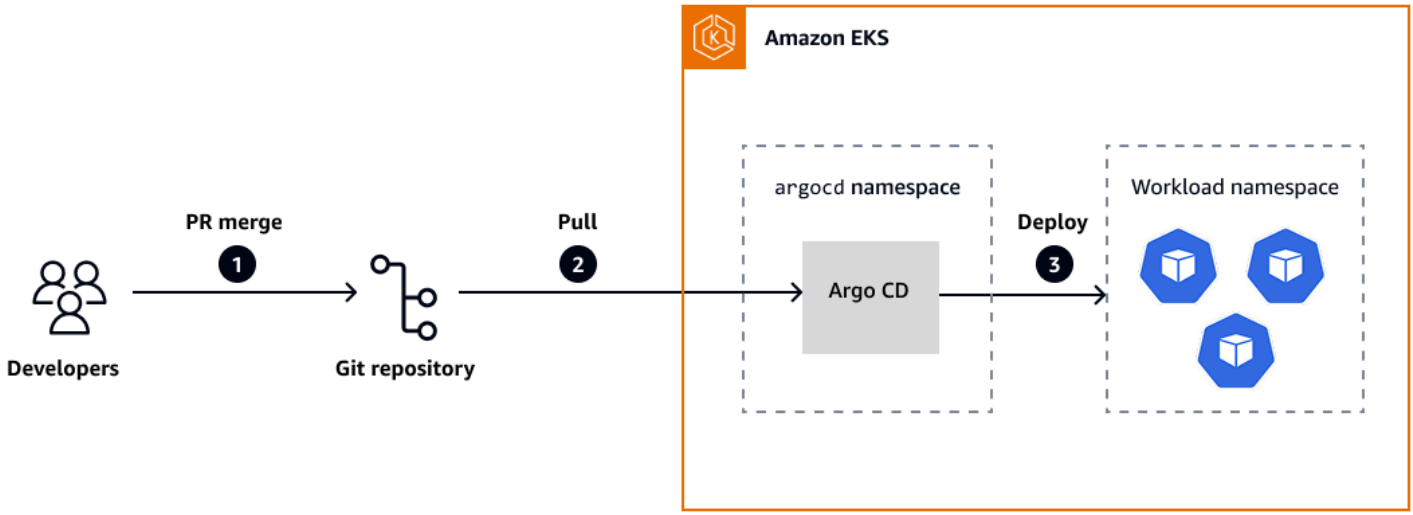
通过遵守这些 GitOps 原则，Argo CD 提供了一种强大、可扩展且安全的方法来管理 Kubernetes 部署。它可确保系统的运行状态始终与 Git 存储库中定义的所需状态保持同步，并提高复杂 Kubernetes 环境中的一致性、可靠性和易管理性。

有关 Argo CD 可以满足的场景和要求，请参阅本指南后面的 Argo CD 用例。有关 Argo CD 和 Flux 之间的比较，请参阅本指南后面的功能比较。

有关更多信息，请参阅 Argo CD 文档。

架构

下图说明了在 GitOps EKS 集群中使用 Argo CD 的驱动型 CD 工作流程。有关详细信息，请参阅 Argo CD 文档。



其中：

- 第 1 步：合并拉取请求 (PR)。开发者提交对存储在 Git 存储库中的 Kubernetes 清单或 Helm 图表的更改。审核 PR 并将其合并到主分支后，将在源代码管理中更新应用程序的所需状态。
- 步骤 2：存储库同步。Argo CD 在 EKS 集群的专用命名空间 (`argocd`) 中运行，并持续监控配置的 Git 存储库。当它检测到更改时，它会提取最新的更新以协调声明的状态。
- 步骤 3：部署到目标命名空间。Argo CD 将来自 Git 的所需状态与集群中的实时状态进行比较。然后，它会对目标工作负载命名空间进行必要的更改，以便相应地部署或更新应用程序。这包括管理部署、服务和密钥等 Kubernetes 资源 ConfigMaps，以保持集群与 Git 真实来源的一致性。

通量

Flux 是 Kubernetes 的另一种工具，它以独特 GitOps 的方式实现原理。

GitOps 支持

区域图	工具功能
Git 是唯一的真相来源	Flux 使用 Git 存储库作为定义系统所需状态的权威来源。应用程序和基础设施的所有配置都存储在 Git 中。
声明式配置	Flux 使用对集群所需状态的声明性描述。这些描述通常是 Kubernetes 清单、Helm 图表或 Kustomize 叠加层。
自动同步	Flux 会持续监控 Git 存储库中的更改。当它检测到更改时，它会自动将其应用于集群。
Kubernetes 原生	Flux 是作为一组 Kubernetes 控制器和自定义资源构建的。它使用 Kubernetes 中的扩展机制来提供功能。 GitOps
基于拉取的部署模型	与传统的基于推送的 CI/CD 系统不同，Flux 使用基于拉动的模型。集群从 Git 中提取所需的状态，而不是使用外部系统来推送更改。

区域图	工具功能
持续对账	Flux 会不断将集群的实际状态与 Git 中的所需状态进行比较。它会自动校正这些状态之间检测到的任何偏差。
多租户	Flux通过其自定义概念支持多租户和. HelmReleases 不同的团队可以独立管理自己的配置部分。
渐进式交付	Flux 通过其 Flagger 组件支持高级部署策略，例如金丝雀版本和 A/B 测试。
头盔集成	Flux 包括对 Helm 的原生支持，因此你可以通过它轻松管理 Helm 版本 GitOps。
图像更新自动化	当容器注册表中有新版本可用时，Flux 可以自动更新 Git 中的容器镜像。
自定义支持	你可以使用 Flux 为 Kustomize 提供的原生支持来自定义和修补 Kubernetes 清单。
安全和 RBAC	Flux 与 Kubernetes RBAC 集成以实现访问控制。它支持通过各种后端管理机密。
可观测性	Flux 提供有关协调和操作的状态信息和指标。它与监控工具集成，增强了可观察性。
事件驱动型架构	Flux 使用事件驱动的方法来实现对账和更新。
可扩展性	该工具旨在实现可扩展，因此您可以添加自定义控制器和资源。
跨集群同步	Flux 支持管理来自一组存储库的多个集群。
依赖关系管理	它允许定义系统不同部分之间的依赖关系，并确保操作顺序正确。
Webhook 接收器	您可以将 Flux 配置为接收来自 Git 提供商或其他系统的 webhook，从而立即开始对账。

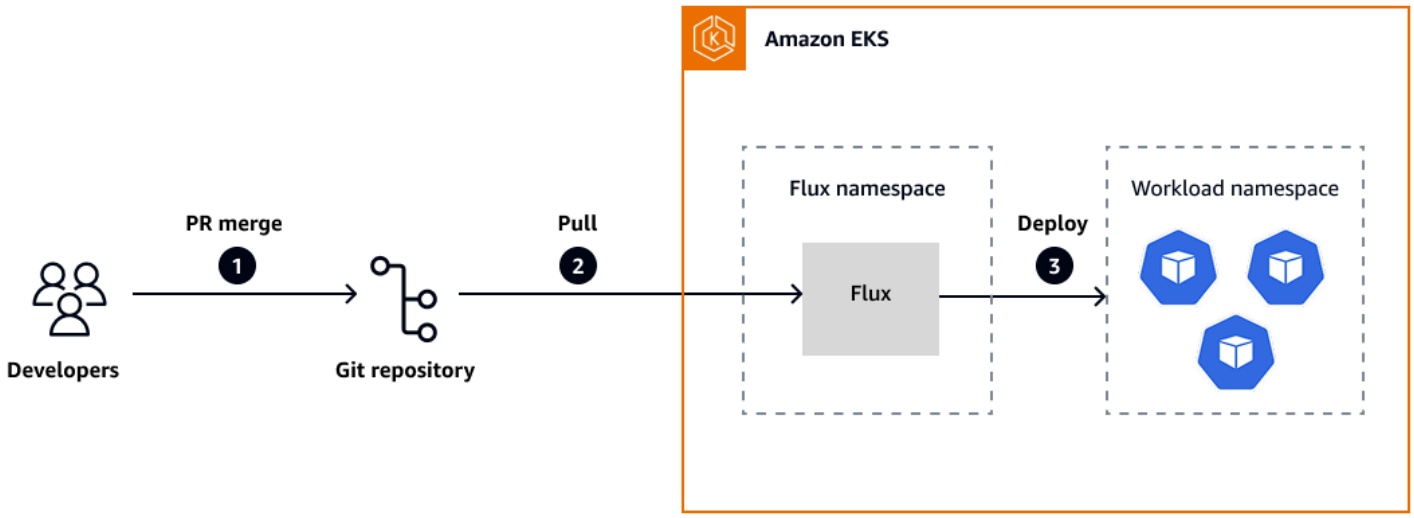
通过实现这些 GitOps 原则，Flux 为管理 Kubernetes 集群和应用程序提供了一个强大而灵活的系统。它可确保您的基础架构和应用程序始终与 Git 存储库同步，并在复杂的 Kubernetes 环境中提供一致性、可靠性和易管理性。该工具的 Kubernetes 原生方法和对自动化的关注使其特别适合云原生环境。

有关 Flux 可以解决的场景和要求，请参阅本指南后面的 Flux 用例。有关 Argo CD 和 Flux 之间的比较，请参阅本指南后面的功能比较。

有关更多信息，请参阅 Flux 文档。

架构

下图说明了在 GitOps EKS 集群中使用 Flux 的驱动型 CD 工作流程。有关详细信息，请参阅 Flux 文档。



其中：

- 第 1 步：合并拉取请求 (PR)。开发者提交对存储在 Git 存储库中的 Kubernetes 清单或 Helm 图表的更改。审核 PR 并将其合并到主分支后，将在源代码管理中更新应用程序的所需状态。
- 步骤 2：存储库同步。Flux 在 EKS 集群的专用命名空间中运行，并持续监控配置的 Git 存储库。当它检测到更改时，它会提取最新的更新以协调声明的状态。
- 步骤 3：部署到目标命名空间。Flux 将来自 Git 的所需状态与集群中的实时状态进行比较。然后，它会对目标工作负载命名空间进行必要的更改，以便相应地部署或更新应用程序。

编织 GitOps

Weave GitOps 由 Weaveworks 开发，该公司是引入该术语的公司。GitOps该工具提供了基于核心 GitOps 原则的全面 GitOps 解决方案。

GitOps 支持

区域图	工具功能
Git 是唯一的真相来源	Weave GitOps 使用 Git 存储库作为定义系统所需状态的权威来源。所有配置，包括应用程序清单、基础架构定义和策略，都存储在 Git 中。
声明式配置	系统依赖于对整个系统状态的声明性描述。这些描述通常是 Kubernetes 清单、Helm 图表或其他声明格式。
自动同步	Weave 会 GitOps 持续监控 Git 存储库的变化。当它检测到更改时，它会自动将其应用于目标环境。
Kubernetes 原生架构	Weave GitOps 是作为一组 Kubernetes 控制器和自定义资源构建的。它使用 Kubernetes 中的扩展机制来提供功能。 GitOps
持续对账	该工具不断将集群的实际状态与 Git 中定义的所需状态进行比较。它会自动校正这些状态之间检测到的任何偏差。
多集群管理	Weave GitOps 支持通过单个控制平面管理多个 Kubernetes 集群。它可以在不同的环境中实现一致的应用程序部署。
政策即代码	Weave GitOps 将政策概念作为执行安全与合规规则的代码相结合。策略与应用程序代码和基础架构定义一起受版本控制。
渐进式交付	此工具支持高级部署策略，例如金丝雀版本和 blue/green 部署。它与 Flagger 集成，可实现自动化、渐进式交付。

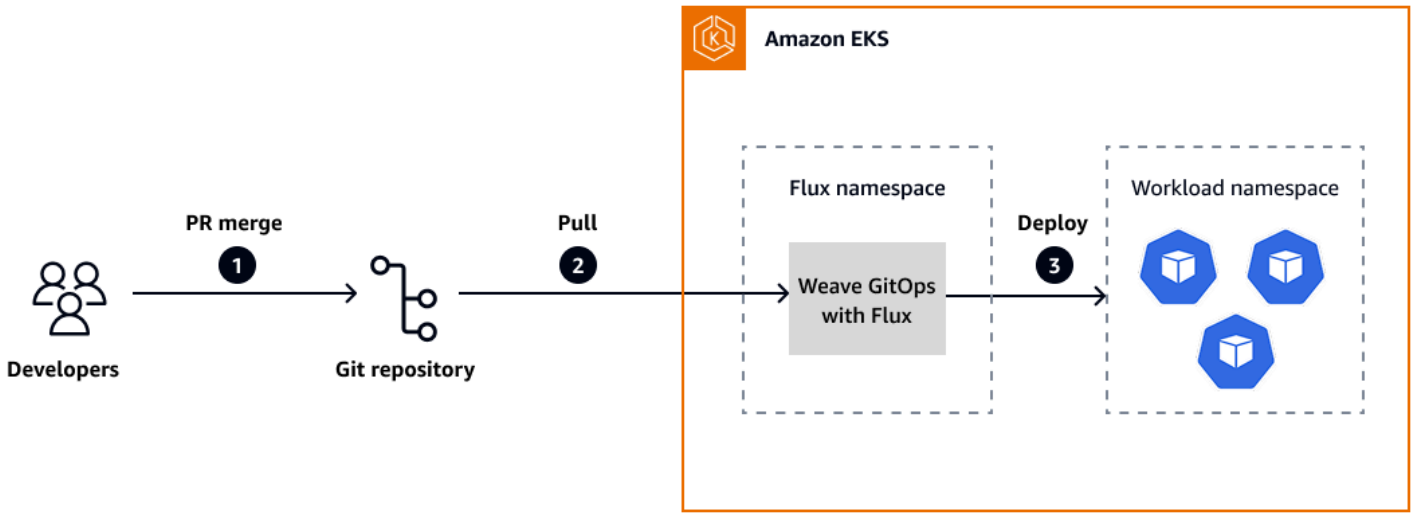
区域图	工具功能
可观察性和仪表板	Weave GitOps 提供了用于监控应用程序和集群状态的内置仪表板。它提供了对账流程和集群运行状况的见解。
通过设计确保安全	该工具实现了安全最佳实践，包括 RBAC 集成和机密管理。它支持多种身份验证方法，并与企业身份提供商集成。
可扩展性和集成性	该工具旨在与各种云原生工具配合使用。它支持 Flux、Helm 和 Kustomize 等流行工具。
自助开发者平台	Weave GitOps 可以为开发人员创建自助服务平台。它为应用程序部署提供了模板和护栏。
GitOps 自动化	该工具可自动执行 GitOps 工作流程的许多方面，包括为更新生成拉取请求。
持续交付渠道	它与 CI/CD 系统集成以创建 end-to-end交付管道。
审计与合规性	Weave DevOps 提供了所有更改和操作的完整审计跟踪。它通过版本控制和自动化流程帮助您满足合规性要求。
可扩展性	该工具旨在从小型项目扩展到大型企业级部署。
团队协作	Weave 通过基于 Git 的工作流程 GitOps 促进开发和运营团队之间的协作。
GitOps 作为一项服务	该工具 GitOps 以托管服务形式提供，可简化采用和管理。
混合云和多云支持	Weave GitOps 支持跨不同的云提供商和本地环境进行一致的管理。
持续安全	该工具在整个部署过程中集成了安全扫描和策略实施。

Weave GitOps 实施这些原则是为了提供超越基本部署自动化的全面 GitOps解决方案。它旨在为云原生应用程序创建完整的运营模型，该模型侧重于安全性、可扩展性和易用性。通过遵守这些 GitOps 原则，Weave GitOps 可以帮助组织跨多个集群和云提供商实现对其 Kubernetes 环境的一致、可审计和高效的管理。

有关更多信息，请参阅 Weave GitOps 文档。

架构

下图说明了在 EKS GitOps 集群 GitOps 中使用 Weave 的驱动型 CD 工作流程。有关详细信息，请参阅 Weave GitOps 存储库。



其中：

- 第 1 步：合并拉取请求 (PR)。开发者提交对存储在 Git 存储库中的 Kubernetes 清单或 Helm 图表的更改。审核 PR 并将其合并到主分支后，将在源代码管理中更新应用程序的所需状态。
- 步骤 2：存储库同步。Weave 在 EKS 集群的 Flux 命名空间中 GitOps运行，并持续监控配置的 Git 存储库。当它检测到更改时，它会提取最新的更新以协调声明的状态。
- 步骤 3：部署到目标命名空间。Weave GitOps 将来自 Git 的所需状态与集群中的实时状态进行比较。然后，它会对目标工作负载命名空间进行必要的更改，以便相应地部署或更新应用程序。

Jenkins X

Jenkins X 是一个云原生开源 CI/CD 平台，它实现了 Kubernetes 环境 GitOps的原理。尽管 Jenkins X 不仅仅是像 Argo CD 或 Flux 这样的 GitOps工具，但它在工作流程中融入了 GitOps 实践。

GitOps 支持

区域图	工具功能
以 Git 为中心的工作流程	Jenkins X 使用 Git 存储库作为应用程序代码和配置的主要真实来源。对应用程序和基础架构的所有更改都是通过 Git 进行的。
环境即代码 (eaC)	环境（例如暂存环境和生产环境）定义为 Git 存储库中的代码。这允许对环境配置进行版本控制和审查。
自动化 CI/CD 管道	Jenkins X 会自动为项目设置 CI/CD 管道。这些管道被定义为代码（管道即代码）并存储在 Git 中。
Kubernetes 原生	Jenkins X 是专门为 Kubernetes 环境构建的。它使用 Kubernetes 资源和自定义资源定义 ()。CRDs
预览环境	Jenkins X 会自动为拉取请求创建临时环境。它允许在合并之前轻松查看和测试更改。
在不同环境之间进行推广	Jenkins X 使用一种 GitOps 方法在环境之间推广应用程序（例如，从暂存到生产）。促销活动通过拉取请求来处理，以确保适当的审查和批准流程。
Helm 图表管理	Jenkins X 使用 Helm 图表来打包和部署应用程序。图表在 Git 存储库中受版本控制。
自动版本控制	Jenkins X 会自动管理应用程序和版本的版本控制。它使用语义版本控制并生成发行说明。
ChatOps 整合	Jenkins X ChatOps 支持常见操作。这符合自动化和协作 GitOps 的原则。
可扩展性	该工具提供了一个用于扩展功能的插件系统。它允许与各种云原生工具集成。

区域图	工具功能
基础设施即代码（IaC）	Jenkins X 支持 Terraform、 CloudFormation AWS Cloud Development Kit (AWS CDK)、和其他 IaC 工具来定义和管理基础架构。基础架构定义与应用程序代码一起受版本控制。
自动回滚	如果在部署后检测到问题，Jenkins X 支持自动回滚。
密钥管理	该工具与外部机密管理解决方案集成，可安全地处理敏感信息。
可观测性	Jenkins X 提供了与监控和日志工具的集成，以实现可观察性。
多云支持	Jenkins X 旨在跨不同的云提供商和本地环境运行。
团队协作	该工具鼓励通过基于 Git 的工作流程和拉取请求进行协作。
持续反馈	该工具通过自动测试和预览环境提供有关更改的快速反馈。
DevOps 最佳实践	Jenkins X 默认会实现 DevOps 最佳实践，包括 GitOps 原则。
声明式配置	该工具使用声明性配置来定义应用程序和环境。
自动升级	Jenkins X 提供了自动升级 Jenkins X 平台本身的工具。

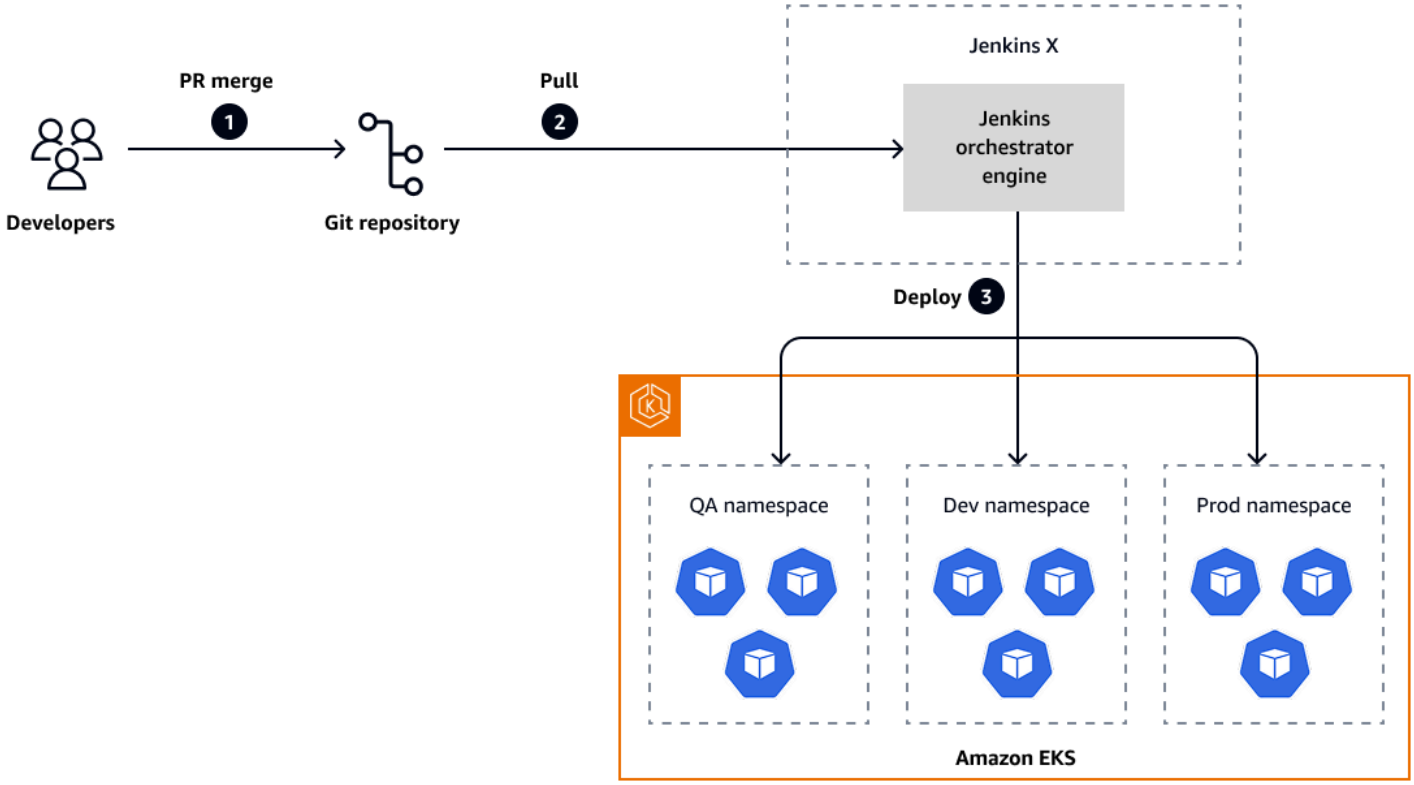
Jenkins X 实现了这些 GitOps 原则，为 Kubernetes 创建了全面的 CI/CD 解决方案。它旨在自动化和简化从代码提交到生产部署的整个软件交付流程，同时遵守 GitOps 实践。通过这样做，它可以帮助团队在云原生环境中实现更快、更可靠、更一致的部署。

Jenkins X 与 Argo CD 或 Flux 等工具之间的主要区别在于，Jenkins X 提供了更全面的 CI/CD 解决方案，包括构建自动化和管道管理，同时仍包含部署和环境管理 GitOps 原则。这使得它特别适合需要在单个 GitOps框架内同时涵盖 CI 和 CD 方面的 all-in-one解决方案的团队。

有关更多信息，请参阅 Jenkins X 文档。

架构

下图说明了使用 Jenkins X 的 GitOps驱动型 CD 工作流程。有关详细信息，请参阅 Jenkins X 文档。



其中：

- 第 1 步：合并拉取请求 (PR)。开发者创建一个拉取请求，其中包含对 Kubernetes 清单、Helm 图表或存储在 Git 存储库中的应用程序代码的更改。审核和批准后，PR 将合并到主分支中，并在源代码管理中更新所需的状态。
- 步骤 2：存储库同步。Jenkins X 在检测到更改时会自动触发 CI/CD 管道。该管道使用 GitOps原理在不同的环境（例如暂存和生产）中构建、测试和推广应用程序。
- 步骤 3：部署到目标命名空间。Jenkins X 使用新的应用程序版本更新环境存储库（暂存和生产存储库）。集群通过从 Git 中提取最新清单并将应用程序部署到相应的命名空间来自动协调更改。

GitLab CI/CD

GitLab CI/CD is an integrated part of the GitLab platform that provides continuous integration, delivery, and deployment capabilities. Although GitLab CI/CD不仅仅是一个 GitOps 工具，你可以对其进行配置以实现 GitOps 原则，尤其是在你将其用于 Kubernetes 部署时。

GitOps 支持

区域图	工具功能
Git 是唯一的真相来源	GitLab CI/CD 使用 Git 存储库来存储应用程序代码和基础架构配置。对系统的所有更改均通过 Git 进行，这可确保完整的历史记录和审计跟踪。
声明式配置	GitLab CI/CD 管道是在.gitlab-ci.yml 文件中定义的，该文件是存储在 Git 存储库中的声明性配置。Kubernetes 清单、Helm 图表或其他基础设施即代码 (IaC) 文件可以存储在同一个存储库中，以定义基础设施的所需状态。
自动化管道	GitLab 当更改推送到存储库时，CI/CD 会自动触发管道。这些管道可以包括构建、测试和部署应用程序的阶段。
Kubernetes 集成	GitLab CI/CD 提供原生 Kubernetes 集成，并支持对 Kubernetes 集群进行 GitOps样式部署。它可以根据 Git 中的配置自动创建和管理 Kubernetes 资源。
环境管理	GitLab CI/CD 支持将多个环境（例如暂存环境和生产环境）定义为代码。根据 GitOps惯例，可以自动部署到这些环境或可能需要手动批准。
查看申请	GitLab 可以自动为合并请求创建临时环境，类似于其他 GitOps工具中的预览环境。这支持在合并之前轻松查看和测试更改。

区域图	工具功能
持续部署	GitLab 可以将 CI/CD 配置为在更改合并到特定分支时自动将更改部署到 Kubernetes 集群。
IaC	GitLab CI/CD 支持与 Terraform 等工具集成，并支持将基础设施作为代码 CloudFormation 进行管理。基础设施定义可以与应用程序代码一起进行版本控制。
可观察性和监测	GitLab CI/CD 提供内置的监控和可观察性功能，包括与 Prometheus 和 Grafana 的集成。
安全扫描	GitLab CI/CD includes built-in security scanning tools that can be integrated into the CI/CDpipeline，作为 GitOps 工作流程的一部分来强制执行安全性。
容器注册表	GitLab CI/CD 包括一个内置的容器注册表，用于将容器镜像管理无缝集成到工作流程中。GitOps
自动 DevOps	GitLab CI/CD can automatically configure CI/CD管道中的自动 DevOps 功能遵循 Kubernetes 部署 GitOps 原则。
批准工作流程	GitLab CI/CD 支持部署批准流程，该流程可在环境之间提供受控的促销。
密钥管理	GitLab CI/CD provides features to securely manage and use secrets within CI/CD管道。
版本控制和发布	GitLab CI/CD supports automatic versioning and release management as part of the CI/CD进程。
回滚	GitLab 如果在部署后检测到问题，CI/CD 可以轻松回滚到以前的版本。

区域图	工具功能
审核日志	GitLab CI/CD 为所有操作提供全面的审计日志，以支持可追溯性方面。 GitOps
多项目管道	GitLab CI/CD 支持跨多个项目或存储库的复杂 GitOps 工作流程。
ChatOps	GitLab CI/CD 支持 ChatOps 集成，通过聊天界面提供协作和操作。
Kubernetes 集群管理	GitLab CI/CD 提供了直接从界面管理 Kubernetes 集群的功能。 GitLab

但是 GitLab CI/CD is not exclusively designed for GitOps, it can be used effectively to implement GitOps practices, especially for teams that already use GitLab as their primary development platform. Its integrated approach, which combines source control, CI/CD，再加上 Kubernetes 的管理，使其成为实现工作流程的强大工具。 GitOps

GitLab CI/CD and dedicated GitOps tools such as Argo CD or Flux is that GitLab provides a more comprehensive platform that includes source control management, issue tracking, and other development tools along with its CI/CD功能之间的关键区别。这使得它特别适合需要能够在更广泛的开发系统中实施 GitOps 实践的 all-in-one解决方案的团队。

有关 C GitLab I/CD 及其架构的更多信息，请参阅 CI/ GitLab C D 文档。

三角帆

尽管 Spinnaker 并不是专门设计为一种 GitOps 工具，但你可以对其进行配置以实现 GitOps 原则，尤其是在将其用于云原生和 Kubernetes 部署时。

GitOps 支持

区域图	工具功能
声明式配置	Spinnaker 使用声明式管道定义，这些定义通常存储为 JSON 或 YAML 文件。根据惯例，这些

区域图	工具功能
	管道定义可以在 Git 存储库中进行 GitOps 版本控制。
IaC	Spinnaker 支持将基础架构和部署配置定义为代码。这些定义可以存储在 Git 存储库中，并且可以作为单一事实来源。
多云部署	Spinnaker 旨在跨多个云提供商和 Kubernetes 集群运行。它可以在不同的环境中实现一致的 GitOps实践。
管道即代码	Spinnaker 管道可以定义为代码并存储在 Git 存储库中。这允许进行版本控制和审查部署过程。
自动部署	您可以将 Spinnaker 配置为根据 Git 存储库中的更改自动启动部署。该工具支持持续部署实践，这些实践是其中的核心 GitOps。
不可变的基础架构	Spinnaker 提倡使用不可变基础架构，这是中的一个关键概念。 GitOps它鼓励部署新实例，而不是修改现有实例。
回滚和版本控制	Spinnaker 提供了强大的回滚功能，可以快速恢复到之前已知的良好状态。它支持根据可追溯性 GitOps 原则对部署进行版本控制。
批准工作流程	Spinnaker 在管道中包括手动判断阶段，以支持环境之间的受控推广。这支持将部署和版本分开的 GitOps 做法。
Canary 和 blue/green 部署	Spinnaker 支持高级部署策略，这些策略与安全和受控版本的 GitOps 实践保持一致。
与版本控制系统集成	Spinnaker 可以与各种 Git 提供程序集成，根据仓库事件启动管道。

区域图	工具功能
Kubernetes 集成	Spinnaker 为 Kubernetes 提供原生支持，并支持 Kubernetes 资源的 GitOps样式管理。
Artical 管理	Spinnaker 支持工件管理和版本控制，这对于维护工作流程至关重要。 GitOps
可观察性和监测	Spinnaker 提供与监控工具的集成，以支持可观察性方面。 GitOps
审计跟踪	Spinnaker 提供了详细的部署日志和历史记录，这些日志和历史记录支持的可审计性原则。 GitOps
基于角色的访问控制 (RBAC)	该工具实现了 RBAC，可根据安全惯例对谁可以执行哪些操作进行精细控制。 GitOps
模板化和参数化	Spinnaker 支持在管道定义中进行模板化，以实现可重复使用和参数化的部署。
环境促进	Spinnaker 以受控的方式促进了不同环境（例如，从暂存到生产）之间的应用程序推广。
与 CI 工具集成	Spinnaker 可以与各种持续集成 (CI) 工具集成，以提供符合原则的完整 CI/CD 管道。 GitOps
自定义舞台和扩展	该工具支持自定义阶段和扩展，因此团队可以实施根据其需求量身定制 GitOps 的工作流程。
集中管理	Spinnaker 提供了一个集中式平台，用于管理跨多个环境和云提供商的部署。

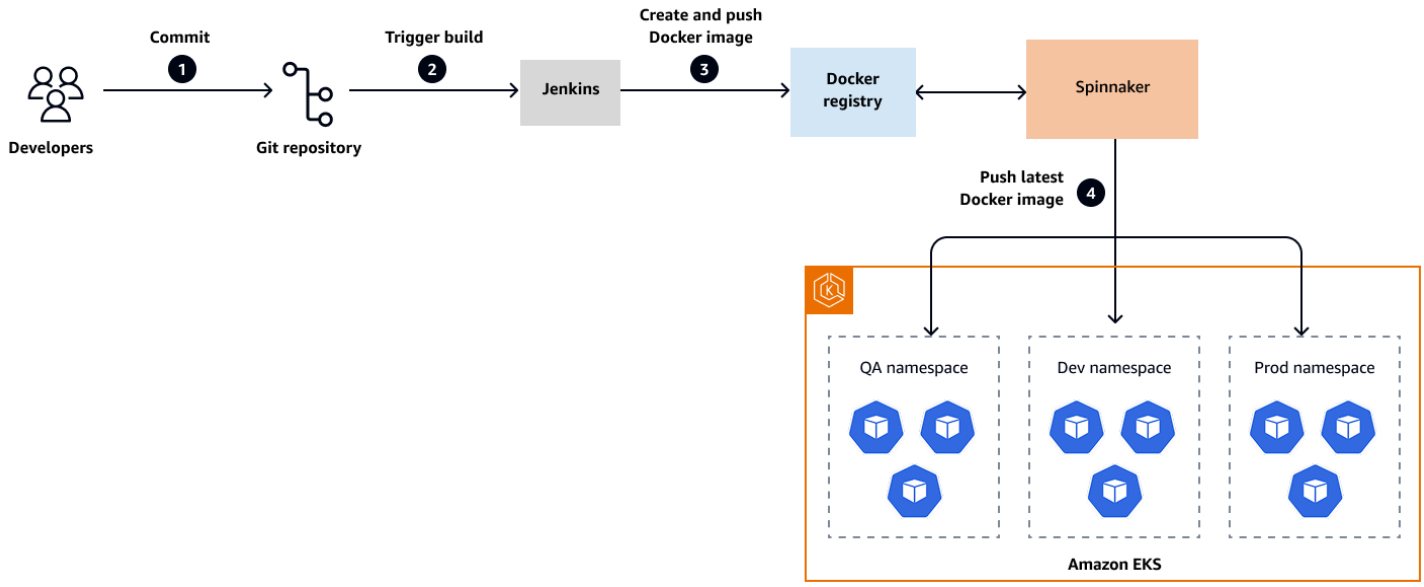
尽管Spinnaker主要不是作为一种 GitOps 工具销售的，但其灵活性和强大的功能集使其能够实现 GitOps 工作流程，尤其是在复杂的多云环境中。Spinnaker与Argo CD或Flux等专用 GitOps 工具的主要区别在于，Spinnaker提供了更全面的持续交付平台，具有高级部署策略和多云支持。

Spinnaker的优势在于它能够处理各种云提供商的复杂部署场景以及对高级部署策略的支持。当 Spinnaker 配置得当时，它可以有效地实现 GitOps 原理。这使其成为想要在多样和复杂环境中采用 GitOps 实践的组织的强大工具。

有关更多信息，请参阅 Spinnaker 文档。

架构

下图说明了使用 Spinnaker 和 Jenkins X 的 GitOps驱动型 CD 工作流程。有关详细信息，请参阅 Spinnaker 文档。



其中：

- 步骤 1：提交代码。开发人员将应用程序代码更改提交到 Git 存储库。这些更改可能包括对应用程序本身、Dockerfiles 或 Kubernetes 清单的更新。
- 第 2 步：Jenkins 构建和镜像创建。Jenkins 由 Git 存储库通过 webhook 或轮询自动触发。Jenkins 构建应用程序，创建 Docker 镜像，然后将构建的映像推送到已配置的 Docker 注册表（例如 Amazon ECR 或 Docker Hub）。
- 第 3 步：三角帆图像监控和管道触发。Spinnaker 会持续监控 Docker 注册表中是否有新镜像。当检测到新的映像版本时，Spinnaker 会自动触发管道以启动部署过程。
- 步骤 4：部署到目标命名空间。Spinnaker 将新的 Docker 镜像部署到亚马逊 EKS。根据管道配置，映像将部署到集群中的目标命名空间。Spinnaker 确保部署最新的应用程序版本，同时遵守已定义的部署策略，例如 blue/green 或 canary 部署。

牧场主舰队

Rancher Fleet 是一款专为管理多个 Kubernetes 集群而设计的 GitOps-at-scale解决方案。它严格遵守 GitOps 原则，同时注重可扩展性和多集群管理。

GitOps 支持

区域图	工具功能
Git 是唯一的真相来源	Fleet 使用 Git 存储库作为权威来源，用于定义跨多个集群的应用程序和资源的所需状态。所有配置，包括 Kubernetes 清单、Helm 图表和自定义资源，都存储在 Git 中。
声明式配置	Fleet 使用对应用程序和资源所需状态的声明性描述。这些资源可以是原始的 Kubernetes YAML、Helm 图表、Kustomize 文件或特定于舰队的自定义资源。
自动同步	Fleet 会持续监控 Git 存储库是否有更改。当它检测到 Git 状态和集群状态之间的差异时，它会自动将更改应用于目标集群。
多集群管理	队列专为管理多个 Kubernetes 集群间的部署而设计。它可以处理来自单个控制平面的成千上万个集群。
Kubernetes 原生架构	舰队是作为一组 Kubernetes 自定义资源和控制器构建的。它使用 Kubernetes 中的扩展机制进行操作。 GitOps
持续对账	Fleet 会不断将集群的实际状态与 Git 中定义的所需状态进行比较。它会自动校正这些状态之间检测到的任何偏差。
集群分组和定位	Fleet 允许您对集群进行分组，并将部署定位到特定组或单个集群。它支持跨不同环境和群集类型进行一致的应用程序部署。

区域图	工具功能
分层配置	Fleet 支持分层配置，分层配置为基础配置提供环境特定的叠加层。这符合高效管理多个环境的 GitOps 做法。
头盔集成	Fleet 为 Helm 图表提供原生支持，并可轻松管理复杂的应用程序。它可以通过 GitOps 工作流程对Helm版本进行版本控制和管理。
自定义资源定义 (CRDs)	Fleet 使用自定义资源（例如 GitRepo 和 Bundle）来定义部署。它们 CRDs 提供了一种基于 Kubernetes 的原生方式来定义工作流程。GitOps
安全和 RBAC	Fleet 与 Kubernetes RBAC 集成以实现访问控制。它支持敏感信息和凭证的安全管理。
可观测性	Fleet 提供有关集群和应用程序同步状态的状态信息。它提供了对整个集群中 GitOps 流程的见解。
可扩展性	Fleet 旨在扩展以高效管理数千个集群。它支持企业环境中的大规模 GitOps 运营。
依赖关系管理	您可以定义不同资源和应用程序之间的依赖关系。Fleet 可确保在复杂的部署中遵循正确的操作顺序。
自定义和可扩展性	Fleet 支持自定义脚本和生命周期挂钩，用于对部署进行高级自定义。它允许与现有工具和工作流程集成。
离线和隔空支持	Fleet 可以在互联网连接有限或没有互联网连接的环境中运行。它支持高安全性或受监管环境中的 GitOps 工作流程。
渐进式推出	Fleet 支持跨集群分阶段部署，这允许采用受控和渐进的部署策略。

区域图	工具功能
统一的管理界面	Fleet 提供了一个用于管理所有集群 GitOps工作流程的单一界面。它简化了复杂的多集群环境中的操作。
与其他 Rancher 工具集成	Fleet 可与其他 Rancher 工具集成，提供全面的 Kubernetes 管理解决方案。
审计跟踪和合规性	Fleet 会对所有变更和部署进行清晰的审计跟踪。它通过版本控制、基于 Git 的操作帮助您满足合规性要求。

Rancher Fleet 在实施这些 GitOps 原则时非常注重可扩展性和多集群管理。它的设计特别适合管理跨不同环境、数据中心或云提供商的大量 Kubernetes 集群的组织。

Fleet 的主要区别在于其大规模处理 GitOps 能力。此功能对于管理大量集群的大型企业或托管服务提供商来说特别有价值。诸如 Argo CD 或 Flux 之类的工具通常用于单个集群管理，而 Fleet GitOps 则用于管理大型集群。

通过遵守这些 GitOps 原则，Rancher Fleet 为希望在多样化和大规模的 Kubernetes 环境中对应用程序和资源实施一致、可扩展和自动管理的组织提供了解决方案。

有关更多信息，请参阅 Fleet 文档。

架构

有关架构和工作流程的信息，请参阅 Fleet 存储库。

Codefresh

Codefresh 是一个支持 GitOps 原则的现代 CI/CD 平台，特别适用于 Kubernetes 部署。Codefresh 提供了一组全面的 CI/CD 功能，其 GitOps 功能非常引人注目。

GitOps 支持

区域图	工具功能
Git 是唯一的真相来源	Codefresh 使用 Git 存储库作为应用程序代码、基础架构定义和管道配置的权威来源。对系统的所有更改均通过 Git 进行，这可确保完整的历史记录和审计跟踪。
声明式配置	Codefresh 通过使用存储在 Git 中的 YAML 文件来支持声明式管道定义。Kubernetes 清单、Helm 图表、 CloudFormation 模板和其他 IaC 文件可以在同一个存储库中进行版本控制。
GitOps 仪表板	Codefresh 提供了一个用于可视化和管理工作流程的专用 GitOps 仪表板。 GitOps 它提供了 Git 和集群状态之间同步状态的清晰视图。
自动同步	Codefresh 会持续监控 Git 存储库的变化。当它检测到差异时，它会自动启动管道以将更改应用于目标环境。
Kubernetes 集成	Codefresh 提供了与 Kubernetes 的深度集成，以支持跨多个集群的 GitOps样式部署。它支持各种 Kubernetes 资源和自定义资源定义 ()。CRDs
环境管理	您可以将多个环境（例如开发、暂存和生产）定义和管理为代码。Codefresh 通过使用 GitOps 实践来支持在环境之间进行推广。
Argo CD 集成	Codefresh 与 Argo CD 集成以增强功能。GitOps它将 CI 功能与 Argo CD 的 CD 优势相结合，提供了完整的 GitOps解决方案。
头盔支持	Codefresh 支持 Helm 图表，可通过它轻松管理复杂的应用程序。 GitOps它还提供 Helm 图表版本控制和推广。

区域图	工具功能
渐进式交付	Codefresh 支持高级部署策略，例如金丝雀和 blue/green 部署。您可以通过 GitOps 工作流程实施和管理这些策略。
回滚和版本控制	如果在部署后检测到问题，Codefresh 可以轻松回滚到以前的版本。它维护部署版本控制以实现可追溯性。
批准工作流程	Codefresh 支持手动和自动的部署批准流程。它允许在环境之间进行有控制的推广，符合GitOps惯例。
IaC	Codefresh 支持与 IaC 工具集成，例如 CloudFormation 和 Terraform。它支持对基础架构定义和应用程序代码进行版本控制。
可观察性和监测	Codefresh 提供了内置的监控和可观察性功能。它还提供与外部监控工具的集成，以增强可见性。
安全扫描	Codefresh 包括可以集成到 GitOps 工作流程中的安全扫描功能。安全检查是自动部署过程的一部分。
审核跟踪	Codefresh 会为所有操作和更改维护全面的审计日志。它支持的可追溯性和合规性方面 GitOps。
RBAC 和访问控制	Codefresh 实现了基于角色的访问控制 (RBAC)，以实现细粒度的权限管理。这有助于确保跨团队和环境的安全 GitOps 运营。
GitOps 自动化	Codefresh 提供了自动执行 GitOps 工作流程各个方面的功能，包括拉取请求 (PR) 的创建和合并。

区域图	工具功能
多云和混合部署	Codefresh 支持跨多个云提供商和本地环境 GitOps 的工作流程。
模板化和参数化	Codefresh 支持管道和部署配置中的模板。这样可以实现可重复使用和参数化的工作流程。GitOps
集成的图像管理	Codefresh 提供了内置的容器镜像管理功能。它将映像构建和部署集成到 GitOps 工作流程中。
GitOps 用于机密管理	Codefresh 提供了在工作流程中 GitOps 管理密钥的安全方法。它与外部机密管理解决方案集成。
协作功能	Codefresh 为流程内的 GitOps 团队协作提供了功能。这些功能包括评论、通知和共享仪表板。

Codefresh 的方法以 GitOps 将 CI/CD 功能与实践集成而著称。 GitOps 它旨在提供一个全面的平台，在遵守 GitOps原则的同时，涵盖整个软件交付生命周期。

Codefresh 在该领域的关键 GitOps 区别在于其统一的平台方法，该方法将 CI 功能与 CD 和功能相结合。 GitOps 这使得它特别适合那些想要在实施 GitOps 实践的同时能够处理复杂 CI/CD 场景的 all-in-one解决方案的团队。

Codefresh 为希望在更广泛的 CI/CD 背景下采用 GitOps方法的组织提供了一个平台，尤其是在使用 Kubernetes 和云原生技术时。

有关更多信息，请参阅 Codefresh 文档。

Pulumi

Pulumi 是一个 IaC 平台，并不是专门为之设计的。 GitOps但是，它可以有效地用于实施 GitOps 原则，特别是对于云基础设施和 Kubernetes 部署。

GitOps 支持

区域图	工具功能
IaC	Pulumi 允许你使用通用编程语言（例如 Python 和 Go）来定义基础架构。 TypeScript这种基于代码的方法符合对版本化、声明性配置的 GitOps 重视。
Git 是唯一的真相来源	Pulumi 中的基础设施代码可以存储在 Git 存储库中并进行版本控制。这可确保 Git 充当基础架构定义的单一事实来源。
声明式期望状态	尽管 Pulumi 使用编程语言，但它仍然以声明方式描述了所需的基础设施状态。该代码定义了基础架构应该是什么样子，而不是创建它的 step-by-step过程。
自动同步	Pulumi 可以与 CI/CD 管道集成，以便在 Git 中更新代码时自动应用更改。这样可以持续部署基础架构变更，这是一项关键 GitOps 原则。
多云和 Kubernetes 支持	Pulumi 支持各种云提供商和 Kubernetes，因此您可以在不同的环境中遵循 GitOps 实践。该工具支持跨不同平台对资源进行一致的管理。
状态管理	Pulumi 管理基础设施的状态，可以远程安全地存储。这种状态管理对于 GitOps 实践至关重要，它可以确保基础架构的定义状态与实际状态之间的一致性。
漂移检测和对账	Pulumi 可以检测所需状态（在代码中）和基础设施的实际状态之间的差异。它按照持续和解 GitOps 的原则调和这些分歧。
政策即代码	您可以使用 Pulumi CrossGuard 将策略定义为代码并强制执行。这样可以对合规性和安全策略进行版本控制 GitOps式的管理。

区域图	工具功能
密钥管理	Pulumi 提供了安全的方式来管理基础设施代码中的敏感信息。它支持与外部机密管理系统的集成，这对于 GitOps安全实践至关重要。
模块化和可重复使用的组件	Pulumi 支持创建可重复使用的组件和模块。这种模块化符合管理复杂的多环境部署的 GitOps 实践。
预览和计划	Pulumi 提供了在应用更改之前对其进行预览的功能。这支持对基础设施进行安全、可预测的变更 GitOps 的原则。
回滚和历史记录	Pulumi 保留了部署历史并支持回滚到以前的状态。这符合可追溯性和可逆性 GitOps 原则。
基础设施的持续交付	Pulumi 可以集成到 CI/CD 管道中，以持续交付基础设施变更。它支持基础设施代码的自动测试和验证。
RBAC 和访问控制	Pulumi 提供基于角色的访问控制，用于管理谁可以更改基础架构。这支持 GitOps安全和治理实践。
可观察性和日志	Pulumi 为基础设施变更提供日志和监控功能。这些功能支持 GitOps 实践的可观察性方面。
与其他工具集成	Pulumi 可以与云端的各种工具集成。这种灵活性可以实现全面 GitOps的工作流程。
环境管理	Pulumi 支持使用具有不同配置的相同代码库来管理多个环境（开发、暂存、生产）。这符合一致的多环境管理 GitOps 实践。
依赖关系管理	Pulumi 处理资源之间的依赖关系，并确保操作顺序正确。这对于涉及相互依赖组件的复杂 GitOps 部署至关重要。

区域图	工具功能
自定义资源提供者	Pulumi 允许您创建自定义提供商来管理任何 API 驱动的服务。这将 GitOps 实践扩展到标准云产品之外的各种资源。
协作功能	Pulumi 通过共享状态和访问控制支持团队协作。这简化了团队环境中的 GitOps 工作流程。

通过使用这些 Pulumi 功能，组织可以对其基础设施实施 GitOps 实践，尤其是在他们需要精细控制或复杂逻辑，或者想要在单一、一致的框架内管理各种云和本地资源的场景中。

Pulumi 的方法 GitOps 是独一无二的，因为它在遵守原则的同时，将通用编程语言的强大功能和灵活性带到了基础设施管理中。 GitOps 对于喜欢使用熟悉的编程语言并希望将软件工程最佳实践应用于基础设施管理的团队来说，这可能特别有利。

Pulumi的主要区别 GitOps 在于它使用标准编程语言来定义基础架构。传统 GitOps 工具通常使用 YAML 或特定领域的语言，而 Pulumi 允许更复杂的逻辑、更好的代码重用以及更轻松地与现有开发工作流程集成。

有关更多信息，请参阅 Pulumi 文档。

GitOps 工具比较

以下是前几节中讨论的九种 GitOps 工具的比较。在选择工具时，请考虑您的具体要求、现有基础架构、团队专业知识以及所需的控制和定制级别。

易于使用

- Argo CD、Flux 和 Rancher Fleet 通常更容易设置。
- Spinnaker 和 Jenkins X 的学习曲线更陡峭。
- Weave GitOps 可能需要更多设置才能使用高级功能。
- GitLab CI/CD 和 Codefresh 提供集成体验。

Kubernetes 集成

- Argo CD、Flux 和 Rancher Fleet 非常以 Kubernetes 为中心。
- Jenkins X 和 Weave GitOps 提供了更广泛的 DevOps 功能。
- 其他工具支持 Kubernetes，但没有专门关注它。

CI/CD 功能

- Jenkins X， GitLab CI/CD, and Codefresh offer complete CI/CD解决方案。
- Argo CD、Flux 和 Weave 更多地 GitOps 关注工作流程的 CD 方面，通常需要与单独的 CI 工具集成。

GitOps 纯度

- Argo CD 和 Flux 是专门关注的 GitOps工具。
- 其他工具在不同程度上纳入了 GitOps 原理。

多云支持

- Spinnaker 和 Pulumi 在多云场景中表现出色。

- 其他工具可以跨云运行，但可能需要额外的设置。

多集群支持

- 所有工具都支持多集群部署。
- Argo CD 和 Weave GitOps 具有更高级的多集群管理功能。

集成

- Flux 拥有强大的云原生计算基金会 (CNCF) 支持。
- Argo CD 拥有一个庞大而活跃的社区。
- Argo CD 和 Flux 具有很强的 Kubernetes 集成。
- Jenkins X 使用更广泛的 Jenkins 系统。
- Weave虽然 GitOps 较新，但在强大的商业支持下不断发展。
- GitLab CI/CD 与紧密集成。 GitLab
- Rancher Fleet 在 Rancher 系统中运行良好。

社区和支持

- Flux 有 CNCF 的强大支持。
- Argo CD 和 Spinnaker 拥有庞大的社区。 GitLab
- 大多数工具都提供商业支持。

企业功能

- 默认情况下，Weave GitOps 和 Jenkins X 提供了更多以企业为中心的功能。
- Argo CD 和 Flux 有企业级产品，也可以扩展以供企业使用。

灵活性和可扩展性

- Flux 高度模块化且可扩展。
- Argo CD 提供了不错的自定义选项。

- Jenkins X 具有很强的可扩展性，但可能需要付出更多的努力。
- Weave GitOps 旨在提供一个对可扩展性需求较少的完整解决方案。

可扩展性

- Spinnaker 和 GitLab CI/CD 以企业级可扩展性而闻名。
- Argo CD 和 Flux 可以很好地处理大规模 Kubernetes 部署。

基础设施管理

- Pulumi 专注于基础设施管理。
- Weave GitOps 和 Flux 提供了不错的 IaC 功能。

编程模型和语言支持

- 在 Pulumi 中，你可以使用通用编程语言（例如 Python、Go TypeScript、C# 和 Java）来定义基础架构。Pulumi 使用标准语言，可以将基础设施代码与熟悉的开发工作流程、测试实践和复杂逻辑集成。
- Terraform 使用 HashiCorp 配置语言 (HCL)。
- CloudFormation 使用 JSON 和 YAML 模板。
- Argo CD、Flux、Rancher Fleet、Weave GitOps、Spinnaker 和 C GitLab I/CD 主要管理 YAML 或声明性配置文件。
- Jenkins X 管理 YAML 和基于脚本的管道，但本机不为 IaC 提供通用编程。

Argo CD 和 Flux 用例

本节重点介绍两种工具，Argo CD和Flux，它们提供了纯粹的 GitOps功能。在这种情况下，p ure GitOps 是指一种模型，其中 Git 存储库充当应用程序和基础架构所需状态的单一事实来源。所有更改均通过 Git 提交完成，系统会自动同步实时环境以匹配存储库中定义的状态。除了 Git 操作之外，无需手动干预。

一般注意事项

- 在视觉管理和以应用程序为中心的工作流程非常重要的环境中，您可能更喜欢使用 Argo CD。
- 如果您需要轻量级解决方案、强大的多租户或与更广泛的云原生计算基础 (CNCF) 网络的深度集成，则可以选择 Flux。
- GitOps 由于其直观的用户界面，Argo CD 经常吸引那些正在从传统 CI/CD 过渡到的团队。
- 在已经建立基于 CLI 的工作流程和 IaC 实践的云原生环境中，Flux 通常受到青睐。

归根结底，Argo CD 和 Flux 之间的选择通常取决于您的特定组织需求、现有工具和团队偏好。这两个工具都能够处理大多数 GitOps 场景，因此我们建议您根据自己的具体用例和要求对其进行评估。

Argo CD 用例

可视化管理：

- 当你需要一个用户友好的用户界面来管理部署和可视化应用程序状态时。
- 适用于喜欢使用图形界面进行监控和故障排除的团队。

以应用程序为中心的方法：

- 当您想在应用程序级别管理部署而不是管理单个资源时。
- 适用于围绕应用程序概念构建部署的组织。

多集群管理：

- 当管理跨多个集群的部署是主要要求时。
- 适用于具有许多群集的复杂分布式环境。

回滚和同步波浪：

- 当您需要对部署过程进行精细控制时，包括同步波和手动干预。
- 适用于需要复杂回滚策略的场景。

与现有工具集成：

- 当你已经在 Argo 项目中使用其他工具，例如 Argo Workflows 和 Argo Events 时。

企业环境：

- 适用于默认情况下需要强大的 RBAC 和单点登录集成的大型企业。

Flux 用例

轻量级部署：

- 当您需要更轻量级、资源密集 GitOps度更低的解决方案时。
- 适用于资源可能受限的边缘计算或物联网场景。

自动图像更新：

- 当自动检测和部署新的容器镜像是关键要求时。
- 适用于专注于持续部署并频繁更新映像的团队。

多租户：

- 当需要强大的多租户支持时，尤其是在共享集群环境中。
- 适用于在团队或项目之间严格区分的服务提供商或大型组织。

IaC：

- 当通过相同 GitOps的工作流程管理应用程序和基础架构很重要时。
- 适用于在 IaC 模式上投入大量资金的团队。

头盔集成：

- 当广泛使用 Helm 图表成为部署策略的一部分时。
- 适用于基于 Helm 的复杂部署的环境。

CNCF 项目整合：

- 当与其他CNCF项目的紧密整合很重要时。
- 适用于符合 CNCF 技术和原则的组织。

模块化架构：

- 当您需要灵活地仅使用 GitOps工具包的特定组件时。
- 适用于想要使用模块化组件构建自定义 GitOps 工作流程的团队。

渐进式交付：

- 当诸如金丝雀版本或 A/B 测试之类的高级部署策略是核心要求时。

功能对比

区域图	Argo CD	通量
Support 对核心 GitOps 原则的支持	✓是	✓是
架构	End-to-end 用于实现 Kubernet GitOps es 工作流程的应用程序	为 Kubernetes CRDs 和控制器提供 GitOps
设置	简便	复杂
头盔支持	✓是	✓是
自定义支持	✓是	✓是

区域图	Argo CD	通量
集成图形用户界面	CLI 和功能齐全的网页用户界面	CLI 和可选的轻量级 Web 界面
RBAC 支持	精细控制	Kubernetes 原生 RBAC
多租户和多集群支持	对多集群的出色支持	对多租户的出色支持
单点登录身份验证	✓是	✓是
同步自动化	能够同步窗口	能够设置协调间隔
部分同步	✓是	✗否
对账流程	支持手动和自动同步。有几种不同的策略可供选择。	支持手动和自动同步。
可扩展性	支持自定义插件。自定义选项有限。	支持自定义控制器。良好的可扩展性和第三方集成。
社区支持	庞大而活跃的社区。	规模较小但不断增长的社区。
可扩展性	可扩展性良好，但受到 Web UI 数据获取速率的限制。社区分析表明，支持成千上万个应用程序。	清晰的横向和纵向可扩展性指南，最多可容纳成千上万个应用程序。

选择 GitOps 工具的最佳实践

本节提供为你的 EKS 集群选择 GitOps 工具的注意事项、提示和最佳实践。正确的选择取决于您的具体背景、要求和长期战略。在做出最终决定之前，对您的首选进行概念验证通常是有益的。

评估贵组织的需求和能力：

- 考虑一下你的团队当前的技能组合和学习新工具的意愿。
- 评估您的 Amazon EKS 环境的复杂性。（例如，您使用的是单个集群还是多个集群？）
- 确定您对合规性、安全性和可扩展性的具体要求。

> **最佳实践**
>
> 创建详细的需求文档，概述必需的功能和有用但不是必需的功能。

评估工具的成熟度和采用率：

- 研究潜在 GitOps 工具的成熟度及其在行业中的采用率。
- 寻找在 Amazon EKS 环境中具有良好记录的工具。

> **最佳实践**
>
> 优先考虑已被广泛采用并在云原生计算基金会 (CNCF) 网络中占有重要地位的工具。

考虑与您现有的工具链集成：

- 评估该 GitOps 工具与您当前的 CI/CD 管道、监控解决方案和其他运营工具的集成程度。
- 寻找原生集成， AWS 服务 例如 IAM、Amazon ECR 和。 CloudWatch

> **最佳实践**
>
> 在做出最终决定之前，先创建概念验证以测试集成功能。

评估安全功能：

- 优先考虑具有强大的基于角色的访问控制 (RBAC) 功能并与 IAM 很好地集成的工具。
- 寻找支持安全机密管理和策略执行的功能。

> **最佳实践**
>
> 选择支持 GitOps基于安全实践的工具，包括策略即代码和自动合规性检查。

评估可扩展性和性能：

- 考虑一下该工具在处理大量应用程序和集群时表现如何。
- 评估其对群集性能和资源消耗的影响。

> **最佳实践**
>
> 使用与您的生产环境相似的工作负载进行性能测试，以确保该工具能够处理您的规模。

考虑多集群和多环境支持：

- 如果您拥有或计划拥有多个 EKS 集群，请优先考虑具有强大多集群管理功能的工具。
- 寻找支持跨不同环境（例如开发、暂存和生产）一致部署的功能。

> **最佳实践**
>
> 选择一款既能集中管理多个集群，又能维护特定环境配置的工具。

评估可观测性和监控能力：

- 寻找能够清晰了解部署状态和集群运行状况的工具。
- 考虑一下该工具与您现有的监控和日志解决方案的集成程度。

> **最佳实践**
>
> 优先考虑提供可自定义仪表板和警报机制的工具，以实现主动问题检测。

评估学习曲线和文档：

- 评估该工具文档的质量和全面性。
- 考虑培训资源和社区支持的可用性。

> **最佳实践**
>
> 选择具有维护良好的文档、活跃的社区论坛以及官方培训计划或认证的工具。

考虑成本和资源利用率：

- 评估采用该工具的直接成本（例如许可和支持）和间接成本（例如运营管理费用和培训成本）。
- 评估该工具在计算和存储资源消耗方面的效率。

> **最佳实践**
>
> 执行包括短期和长期成本在内的总拥有成本 (TCO) 分析。

评估灵活性和自定义选项：

- 寻找可让您自定义工作流程以满足特定需求的工具。
- 考虑该工具通过插件或 APIs的可扩展性。

> **最佳实践**
>
> 选择一款既能平衡默认功能又能根据您的独特要求进行自定义的工具。

评估持续交付和渐进式部署能力：

- 寻找支持高级部署策略（例如 Canary 版本和 blue/green 部署）的工具。
- 评估实施和管理这些策略的难易程度。

> **最佳实践**
>
> 优先考虑为渐进式交付模式提供内置支持的工具，以最大限度地降低部署风险。

考虑供应商锁定和可移植性：

- 评估该工具对特定云提供商或技术的依赖性。
- 如果需要，可以考虑将来是否可以轻松迁移到其他工具。

> **最佳实践**
>
> 优先选择使用开放标准并为您的 GitOps 配置提供导出功能的工具。

评估社区支持和扩展：

- 看看用户社区的规模和活动。
- 评估第三方集成和插件的可用性。

> **最佳实践**
>
> 在做出决定之前，加入社区论坛或用户群组，从其他用户那里获得第一手体验。

考虑合规和审计要求：

- 评估该工具在多大程度上支持您的合规需求，包括审计跟踪和报告。
- 寻找有助于维护和证明合规性的功能。

> **最佳实践**
>
> 选择能够提供全面审计日志并支持生成合规报告的工具。

评估回滚和灾难恢复能力：

- 评估回滚机制的简便性和可靠性。
- 考虑一下该工具如何支持灾难恢复方案。

> **最佳实践**
>
> 作为评估的一部分，对回滚和恢复过程进行全面测试。

常见问题解答

问：Amazon EKS 最受欢迎的 GitOps 工具有哪些？

答：亚马逊 EKS 最受欢迎的 GitOps 工具包括 A rgo CD、Flux、Jenkins X 和 GitLab CI/CD。每种工具都有其优势，但是 Argo CD 和 Flux 因其 Kubernetes 原生方法和强大的社区支持而备受推崇。

问：如何 GitOps 改进 EKS 集群管理？

答：通过为基础设施提供版本控制、自动化部署、通过声明式配置 GitOps 提高安全性、更轻松的回滚以及更好的可审计性来改进 EKS 集群管理。它还可以增强协作并减少部署中的人为错误。

问：我应该在 Amazon EKS 的 GitOps 工具中寻找哪些关键功能？

答：需要寻找的主要功能包括：无缝的 Amazon EKS 集成、强大的 RBAC、多集群支持、可观察性功能、对渐进式交付策略的支持、可扩展性以及与 IAM 和 Amazon ECR AWS 服务 等的集成。

问： GitOps 在 Amazon EKS 中实施时，如何确保安全？

答：为确保安全，请选择一种与 IAM 具有强大的 RBAC 集成、安全机密管理、支持加密 Git 存储库以及能够将安全策略作为代码实施的工具。此外，请验证该工具是否提供了全面的审核日志。

问： GitOps 工具能否处理多集群 Amazon EKS 环境？

答：是的，诸如 Argo CD 和 Flux 之类的 GitOps 工具具有强大的多集群管理功能。它们允许您从单个控制平面管理多个 EKS 集群，从而确保跨环境的一致性。

问： GitOps 工具如何与现有的 CI/CD 管道集成？

答： GitOps 工具通常通过充当管道的部署阶段来与现有的 CI/CD 管道集成。当更改推送到 Git 存储库时，它们可以由 CI 工具触发，它们可以自动部署到 EKS 集群。

问： GitOps 在 Amazon EKS 中实施会面临哪些挑战？

答：常见的挑战包括安全地管理机密、确保适当的访问控制、处理有状态的应用程序、管理 Git 和集群状态之间的偏差，以及根据 GitOps 模型调整团队工作流程。

问：在 Amazon EKS 中， GitOps 工具是如何处理回滚的？

答： GitOps 工具通常通过恢复到 Git 存储库中的先前提交来处理回滚。这会自动触发先前已知良好状态的部署，从而实现快速可靠的回滚。

问： GitOps 工具能否管理 Amazon EKS 插件和其他 AWS 资源？

答：许多 GitOps 工具都可以管理 Amazon EKS 插件和一些 AWS 资源，尤其是当它们与 IaC 工具（例如 Terraform 或）结合使用时。 CloudFormation但是，此功能的范围可能会有所不同；有关每种GitOps工具的具体信息，请参阅工具部分。

问： GitOps 工具如何支持 Amazon EKS 中的合规要求？

答： GitOps 工具通过提供所有变更的清晰审计跟踪、强制执行批准流程、实施自动合规性检查的政策即代码以及提供详细的日志和报告功能来支持合规性。

问：在 Amazon EKS GitOps 中实施的学习曲线是怎样的？

答：学习曲线可能因工具和团队的现有知识而异。通常，熟悉 Git、Kubernetes 和 Amazon EKS 的团队会比其他团队更快地适应。大多数流行的工具都提供大量的文档和培训资源，便于采用。

问：在 Amazon EKS 中， GitOps 工具是如何处理机密管理的？

答： GitOps 工具通常与外部机密管理解决方案（例如 AWS Secrets Manager 或 HashiCorp Vault）集成。有些工具还为存储在 Git 存储库中的机密提供内置加密功能。

问：在 Amazon EKS 中， GitOps 工具能否同时使用无状态和有状态应用程序？

答：是的， GitOps 工具可以与无状态应用程序和有状态应用程序一起使用。但是，管理有状态的应用程序通常需要额外的注意事项，例如处理永久卷和确保更新期间的数据一致性。

问： GitOps 工具如何支持 Canary 或 Amazon EKS 中的 blue/green 部署？

答：许多 GitOps 工具都为高级部署策略提供内置支持。他们可以管理新版本的逐步推出，监控问题，并在检测到问题时自动回滚。所有这些操作都被定义为 Git 存储库中的代码。

问：使用 GitOps 工具和使用**`kubectl apply`** CI/CD 管道有什么区别？

答：与简单`kubectl apply`命令相比， GitOps 工具具有优势，包括自动偏差检测和对账、通过基于拉取的部署提高安全性、更好的可审计性以及更复杂的部署策略。它们还提供了一种更全面的方法来管理整个集群状态。

资源

以下资源提供了官方文档、实用指南、案例研究和深入分析，可帮助您在为 EKS 集群选择 GitOps工具时做出明智的决定。它们涵盖的各个方面 GitOps，包括实施策略、最佳实践、不同工具之间的比较以及现实世界的经验。

AWS 资源：

- Amazon EKS 文档
- 使用 GitOps（AWS 博客文章）自动化 Amazon EKS
- 使用 Weaveworks 介绍 EKS（工作坊） GitOps AWS
- Flux 实验室（亚马逊 EKS 研讨会）
- Argo CD 实验室（亚马逊 EKS 研讨会）

GitOps 和工具文档：

- GitOps 持续部署和渐进式安全的最佳实践（ DevOps.com 网络研讨会点播）
- Kubernetes 文档
- Argo CD 文档
- Flux 文档
- 编织文档 GitOps
- Jenkins X 文档
- GitLab CI/CD 文档
- Spinnaker 文档
- 牧场舰队文档
- CodeFresh 文档

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
初次发布	—	2025 年 4 月 30 日

AWS 规范性指导词汇表

以下是 AWS 规范性指导提供的策略、指南和模式中的常用术语。若要推荐词条，请使用术语表末尾的提供反馈链接。

数字

7 R

将应用程序迁移到云中的 7 种常见迁移策略。这些策略以 Gartner 于 2011 年确定的 5 R 为基础，包括以下内容：

- Refactor/re-architect — 充分利用云原生功能来提高敏捷性、性能和可扩展性，从而移动应用程序并修改其架构。这通常涉及到移植操作系统和数据库。示例：将您的本地 Oracle 数据库迁移到亚马逊 Aurora PostgreSQL-Compatible 版。
- 更换平台：将应用程序迁移到云中，并进行一定程度的优化，以利用云功能。示例：将本地 Oracle 数据库迁移到 AWS Cloud中的 Amazon Relational Database Service（Amazon RDS）for Oracle。
- 重新购买：转换到其他产品，通常是从传统许可转向 SaaS 模式。示例：将您的客户关系管理 (CRM) 系统迁移到 Salesforce.com。
- 重新托管（直接迁移）：将应用程序迁移到云，无需进行任何更改即可利用云功能。示例：将本地 Oracle 数据库迁移到 AWS Cloud中 EC2 实例上的 Oracle。
- 重新放置（虚拟机监控器级直接迁移）：将基础设施迁移到云中，无需购买新硬件、重写应用程序或修改现有操作。您将服务器从本地平台迁移到同一平台的云服务中。示例：将Microsoft Hyper-V应用程序迁移到 AWS。
- 保留（重访）：将应用程序保留在源环境中。其中可能包括需要进行重大重构的应用程序，并且您希望将工作推迟到以后，以及您希望保留的遗留应用程序，因为迁移它们没有商业上的理由。
- 停用：停用或删除源环境中不再需要的应用程序。

A

A2A () Agent-to-Agent

一种支持任务委托和状态转移的代理到代理协作的状态协议。

ABAC

请参阅基于属性的访问控制。

抽象服务

请参阅托管服务。

ACID

请参阅原子性、一致性、隔离性、持久性。

主动-主动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步（通过使用双向复制工具或双写操作），两个数据库都在迁移期间处理来自连接应用程序的事务。这种方法支持小批量、可控的迁移，而不需要一次性割接。它比主动-被动迁移更灵活，但工作量更大。

主动-被动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步，但在将数据复制到目标数据库时，只有源数据库处理来自连接应用程序的事务。目标数据库在迁移期间不接受任何事务。

座席

一种能够使用工具自主推理、计划和采取行动来实现目标的人工智能系统。

特工行动

在生产环境中大规模构建、测试、部署和运行 AI 代理的操作实践。

聚合函数

一种 SQL 函数，它对一组行进行操作并计算该组的单个返回值。聚合函数的示例包括 SUM 和 MAX。

AI

请参阅人工智能。

AIOps

请参阅人工智能运营。

匿名化

永久删除数据集中个人信息的过程。匿名化可以帮助保护个人隐私。匿名化数据不再被视为个人数据。

反模式

一种用于解决反复出现的问题的常用解决方案，而在这类问题中，此解决方案适得其反、无效或不如替代方案有效。

应用程序控制

一种安全方法，仅允许使用经批准的应用程序，以帮助保护系统免受恶意软件的侵害。

应用程序组合

有关组织使用的每个应用程序的详细信息的集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是产品组合发现和分析过程的关键，有助于识别需要进行迁移、现代化和优化的应用程序并确定其优先级。

人工智能（AI）

计算机科学领域致力于使用计算技术执行通常与人类相关的认知功能，例如学习、解决问题和识别模式。有关更多信息，请参阅什么是人工智能？

人工智能运营（AIOps）

使用机器学习技术解决运营问题、减少运营事故和人为干预以及提高服务质量的过程。有关如何在 AWS 迁移策略中使用 AIOps 的更多信息，请参阅运营集成指南。

非对称加密

一种加密算法，使用一对密钥，一个公钥用于加密，一个私钥用于解密。您可以共享公钥，因为它不用于解密，但对私钥的访问应受到严格限制。

原子性、一致性、隔离性、持久性（ACID）

一组软件属性，即使在出现错误、电源故障或其他问题的情况下，也能保证数据库的数据有效性和操作可靠性。

基于属性的访问权限控制（ABAC）

根据用户属性（如部门、工作角色和团队名称）创建精细访问权限的做法。有关更多信息，请参阅 AWS Identity and Access Management (I AM) 文档 AWS中的 AB AC。

权威数据来源

存储主要数据版本的位置，被认为是最可靠的信息源。您可以将数据从权威数据来源复制到其他位置，以便处理或修改数据，例如对数据进行匿名化、编辑或假名化。

可用区

中的一个不同位置 AWS 区域 ，不受其他可用区域故障的影响，并向同一区域中的其他可用区提供低成本、低延迟的网络连接。

AWS 云采用框架 (AWS CAF)

该框架包含指导方针和最佳实践 AWS ，可帮助组织制定高效且有效的计划，以成功迁移到云端。AWS CAF将指导分为六个重点领域，称为视角：业务、人员、治理、平台、安全和运营。业务、人员和治理角度侧重于业务技能和流程；平台、安全和运营角度侧重于技术技能和流程。例如，人员角度针对的是负责人力资源（HR）、人员配置职能和人员管理的利益相关者。从这个角度来看，AWS CAF 为人员发展、培训和沟通提供了指导，以帮助组织为成功采用云做好准备。有关更多信息，请参阅 AWS CAF 网站和 AWS CAF 白皮书。

AWS 工作负载资格框架 (AWS WQF)

一种评估数据库迁移工作负载、推荐迁移策略和提供工作估算的工具。 AWS WQF 包含在 AWS Schema Conversion Tool (AWS SCT) 中。它用来分析数据库架构和代码对象、应用程序代码、依赖关系和性能特征，并提供评测报告。

B

恶意机器人

一种旨在扰乱或伤害个人或组织的机器人。

BCP

请参阅业务连续性计划。

行为图

一段时间内资源行为和交互的统一交互式视图。您可以使用 Amazon Detective 的行为图来检查失败的登录尝试、可疑的 API 调用和类似的操作。有关更多信息，请参阅 Detective 文档中的行为图中的数据。

大端序系统

一个先存储最高有效字节的系统。另请参阅字节顺序。

二进制分类

一种预测二进制结果（两个可能的类别之一）的过程。例如，您的 ML 模型可能需要预测诸如“该电子邮件是否为垃圾邮件？”或“这个产品是书还是汽车？”之类的问题

bloom 筛选条件

一种概率性、内存高效的数据结构，用于测试元素是否为集合的成员。

blue/green 部署

一种部署策略，您可以创建两个独立但完全相同的环境。在一个环境中运行当前应用程序版本（蓝色），在另一个环境中运行新应用程序版本（绿色）。此策略可帮助您在影响最小的情况下快速回滚。

自动程序

一种通过互联网运行自动任务并模拟人类活动或交互的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的 Web 爬网程序。还有一些被称为恶意机器人的机器人，其目的是扰乱或伤害个人或组织。

僵尸网络

被恶意软件感染并受单方（称为僵尸网络控制者或僵尸网络操作者）控制的僵尸网络。僵尸网络是最著名的扩展机器人及其影响力的机制。

分支

代码存储库的一个包含区域。在存储库中创建的第一个分支是主分支。您可以从现有分支创建新分支，然后在新分支中开发功能或修复错误。为构建功能而创建的分支通常称为功能分支。当功能可以发布时，将功能分支合并回主分支。有关更多信息，请参阅关于分支（GitHub 文档）。

紧急（break-glass）访问

在特殊情况下，通过批准的流程，用户 AWS 账户 可以快速访问他们通常没有访问权限的内容。有关更多信息，请参阅指南中的 “实施破碎玻璃程序” AWS Well-Architected 指示器。

棕地策略

您环境中的现有基础设施。在为系统架构采用棕地策略时，您需要围绕当前系统和基础设施的限制来设计架构。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

缓冲区缓存

存储最常访问的数据的内存区域。

业务能力

企业如何创造价值（例如，销售、客户服务或营销）。微服务架构和开发决策可以由业务能力驱动。有关更多信息，请参阅在 AWS上运行容器化微服务白皮书中的围绕业务能力进行组织部分。

业务连续性计划（BCP）

一项计划，旨在应对大规模迁移等破坏性事件对运营的潜在影响，并使企业能够快速恢复运营。

C

CAF

请参阅 AWS 云采用框架。

金丝雀部署

缓慢而渐进地向最终用户发布版本。当您确信无误后，即可部署新版本，并完全替换当前版本。

CCoE

请参阅云卓越中心。

CDC

请参阅更改数据捕获。

更改数据捕获（CDC）

跟踪数据来源（如数据库表）的更改并记录有关更改的元数据的过程。您可以将 CDC 用于各种目的，例如审计或复制目标系统中的更改以保持同步。

混沌工程

故意引入故障或破坏性事件来测试系统的韧性。您可以使用 AWS Fault Injection Service (AWS FIS) 来执行实验，对您的 AWS 工作负载施加压力并评估其响应。

CI/CD

请参阅持续集成和持续交付。

分类

一种有助于生成预测的分类流程。分类问题的 ML 模型预测离散值。离散值始终彼此不同。例如，一个模型可能需要评估图像中是否有汽车。

公民开发者

使用无code/low代码平台创建 AI 应用程序但没有专业技术技能的企业用户。

客户端加密

在目标 AWS 服务 收到数据之前，对数据进行本地加密。

云卓越中心（CCoE）

一个多学科团队，负责推动整个组织的云采用工作，包括开发云最佳实践、调动资源、制定迁移时间表、领导组织完成大规模转型。有关更多信息，请参阅 AWS Cloud 企业战略博客上的 CCoE 帖子。

云计算

通常用于远程数据存储和 IoT 设备管理的云技术。云计算通常连接到边缘计算技术。

云运营模型

在 IT 组织中，一种用于构建、完善和优化一个或多个云环境的运营模型。有关更多信息，请参阅构建您的云运营模型。

云采用阶段

组织迁移到 AWS Cloud中时通常会经历四个阶段：

- 项目 - 出于概念验证和学习目的，开展一些与云相关的项目
- 基础 - 进行基础投资以扩大云采用率（例如，创建登录区、定义 CCoE、建立运营模型）
- 迁移 - 迁移单个应用程序
- Re-invention — 优化产品和服务，在云端进行创新

Stephen Orban在 AWS Cloud 企业战略博客的博客文章《走向之旅 Cloud-First 和采用阶段》中定义了这些阶段。有关它们与 AWS 迁移策略的关系的信息，请参阅迁移准备指南。

CMDB

请参阅配置管理数据库。

代码存储库

通过版本控制过程存储和更新源代码和其他资产（如文档、示例和脚本）的位置。常见的云存储库包括 GitHub 或 Bitbucket Cloud。每个版本的代码都称为一个分支。在微服务结构中，每个存储库都专门用于一个功能。单个 CI/CD管道可以使用多个存储库。

冷缓存

一种空的、填充不足或包含过时或不相关数据的缓冲区缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取，这比从缓冲区缓存读取要慢。

冷数据

很少访问的数据，且通常是历史数据。查询此类数据时，通常可以接受慢速查询。将这些数据转移到性能较低且成本更低的存储层或类别可以降低成本。

计算机视觉（CV）

一种 AI 领域，它使用机器学习来分析和提取数字图像和视频等视觉格式中的信息。例如，Amazon SageMaker AI 为 CV 提供了图像处理算法。

配置偏移

对于工作负载而言，一种偏离预期状态的配置更改。这可能会导致工作负载变得不合规，且通常是渐进的，不是故意的。

配置管理数据库（CMDB）

一种存储库，用于存储和管理有关数据库及其 IT 环境的信息，包括硬件和软件组件及其配置。您通常在迁移的产品组合发现和分析阶段使用来自 CMDB 的数据。

合规性包

一系列 AWS Config 规则和补救措施，您可以汇编这些规则和补救措施，以自定义合规性和安全性检查。您可以使用 YAML 模板将一致性包作为单个实体部署在 AWS 账户 和区域或整个组织中。有关更多信息，请参阅 AWS Config 文档中的一致性包。

持续集成和持续交付 (CI/CD)

自动执行软件发布过程的源代码、构建、测试、暂存和生产阶段的过程。 CI/CD 通常被描述为管道。 CI/CD 可以帮助您实现流程自动化、提高生产力、提高代码质量和更快地交付。有关更多信息，请参阅持续交付的优势。CD 也可以表示持续部署。有关更多信息，请参阅持续交付与持续部署。

CV

请参阅计算机视觉。

D

静态数据

网络中静止的数据，例如存储中的数据。

数据分类

根据网络中数据的关键性和敏感性对其进行识别和分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它可以帮助您确定对数据的适当保护和保留控制。数据分类是《 AWS Well-Architected 框架》中安全支柱的组成部分。有关详细信息，请参阅数据分类。

数据漂移

生产数据与用来训练机器学习模型的数据之间的有意义差异，或者输入数据随时间推移的有意义变化。数据漂移可能降低机器学习模型预测的整体质量、准确性和公平性。

传输中数据

在网络中主动移动的数据，例如在网络资源之间移动的数据。

数据网格

一种架构框架，可提供分布式、去中心化的数据所有权以及集中式管理和治理。

数据最少化

仅收集并处理绝对必要数据的原则。在中进行数据最小化 AWS Cloud 可以降低隐私风险、成本和分析碳足迹。

数据边界

AWS 环境中的一组预防性防护措施，可帮助确保只有可信身份才能访问来自预期网络的可信资源。有关更多信息，请参阅在上构建数据边界。 AWS

数据预处理

将原始数据转换为 ML 模型易于解析的格式。预处理数据可能意味着删除某些列或行，并处理缺失、不一致或重复的值。

数据溯源

在数据的整个生命周期跟踪其来源和历史的过程，例如数据如何生成、传输和存储。

数据主体

正在收集和处理其数据的个人。

数据仓库

一种支持商业智能（例如分析）的数据管理系统。数据仓库通常包含大量历史数据，通常用于查询和分析。

数据库定义语言（DDL）

在数据库中创建或修改表和对象结构的语句或命令。

数据库操作语言（DML）

在数据库中修改（插入、更新和删除）信息的语句或命令。

DDL

请参阅数据库定义语言。

深度融合

组合多个深度学习模型进行预测。您可以使用深度融合来获得更准确的预测或估算预测中的不确定性。

深度学习

一个 ML 子字段使用多层人工神经网络来识别输入数据和感兴趣的目标变量之间的映射。

深度防御

一种信息安全方法，经过深思熟虑，在整个计算机网络中分层实施一系列安全机制和控制措施，以保护网络及其中数据的机密性、完整性和可用性。当你采用这种策略时 AWS，你会在 AWS Organizations 结构的不同层面添加多个控件来帮助保护资源。例如，深度防御方法可能将多因素身份验证、网络分段和加密结合起来。

委派管理员

在中 AWS Organizations，兼容的服务可以注册 AWS 成员帐户来管理组织的帐户并管理该服务的权限。此账户被称为该服务的委托管理员。有关更多信息和兼容服务列表，请参阅 AWS Organizations 文档中使用 AWS Organizations的服务。

部署

使应用程序、新功能或代码修复在目标环境中可用的过程。部署涉及在代码库中实现更改，然后在应用程序的环境中构建和运行该代码库。

开发环境

请参阅环境。

侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出提醒。这些控制是第二道防线，提醒您注意绕过现有预防性控制的安全事件。有关更多信息，请参阅在 AWS上实施安全控制中的侦测性控制。

开发价值流映射（DVSM）

用于识别对软件开发生命周期中的速度和质量产生不利影响的限制因素并确定其优先级的流程。DVSM 扩展了最初为精益生产实践设计的价值流映射流程。其重点关注在软件开发过程中创造和转移价值所需的步骤和团队。

数字孪生

真实世界系统的虚拟再现，如建筑物、工厂、工业设备或生产线。数字孪生支持预测性维护、远程监控和生产优化。

维度表

星型架构中的一种较小的表，其中包含事实表中定量数据的数据属性。维度表属性通常是文本字段或行为类似于文本的离散数字。这些属性通常用于查询约束、筛选和结果集标注。

灾难

阻止工作负载或系统在其主要部署位置实现其业务目标的事件。这些事件可能是自然灾害、技术故障或人为操作的结果，例如无意的配置错误或恶意软件攻击。

灾难恢复（DR）

您用来最大程度地减少由灾难造成的停机时间和数据丢失的策略和流程。有关更多信息，请参阅《工作负载灾难恢复 AWS： AWS Well-Architected 框架中的云端恢复》。

DML

请参阅数据库操作语言。

领域驱动设计

一种开发复杂软件系统的方法，通过将其组件连接到每个组件所服务的不断发展的领域或核心业务目标。埃里克·埃文斯（Eric Evans）在他的《Domain-Driven 设计：解决软件核心的复杂性》（波士顿： Addison-Wesley 专业版，2003年）一书中介绍了这个概念。有关如何使用带有 strangler fig 模式的域驱动设计的信息，请参阅使用容器和 Amazon API Gateway 逐步实现传统微软 ASP.NET (ASMX) 网络服务的现代化。

DR

请参阅灾难恢复。

偏差检测

跟踪与基准配置的偏差。例如，您可以使用 AWS CloudFormation 来检测系统资源中的偏差，也可以使用 AWS Control Tower 来检测着陆区中可能影响监管要求合规性的变化。

DVSM

请参阅开发价值流映射。

E

EDA

请参阅探索性数据分析。

EDI

请参阅电子数据交换。

边缘计算

该技术可提高位于 IoT 网络边缘的智能设备的计算能力。与云计算比较时，边缘计算可以减少通信延迟并缩短响应时间。

电子数据交换（EDI）

组织之间业务文件的自动交换。有关更多信息，请参阅什么是电子数据交换。

加密

一种将人类可读的纯文本数据转换为加密文字的计算流程。

加密密钥

由加密算法生成的随机位的加密字符串。密钥的长度可能有所不同，而且每个密钥都设计为不可预测且唯一。

字节顺序

字节在计算机内存中的存储顺序。 Big-endian 系统首先存储最重要的字节。 Little-endian 系统首先存储最低有效字节。

端点

请参阅服务端点。

端点服务

一种可以在虚拟私有云（VPC）中托管，与其他用户共享的服务。您可以使用其他 AWS 账户 或 AWS Identity and Access Management (IAM) 委托人创建终端节点服务， AWS PrivateLink 并向其授予权限。这些账户或主体可通过创建接口 VPC 端点来私密地连接到您的端点服务。有关更多信息，请参阅 Amazon Virtual Private Cloud（Amazon VPC）文档中的创建端点服务。

企业资源规划（ERP）

一种自动化和管理企业关键业务流程（例如会计、MES 和项目管理）的系统。

信封加密

用另一个加密密钥对加密密钥进行加密的过程。有关更多信息，请参阅 AWS Key Management Service (AWS KMS) 文档中的信封加密。

环境

正在运行的应用程序的实例。以下是云计算中常见的环境类型：

- 开发环境 — 正在运行的应用程序的实例，只有负责维护应用程序的核心团队才能使用。开发环境用于测试更改，然后再将其提升到上层环境。这类环境有时称为测试环境。
- 下层环境 — 应用程序的所有开发环境，比如用于初始构建和测试的环境。
- 生产环境 — 最终用户可以访问的正在运行的应用程序的实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上层环境 — 除核心开发团队以外的用户可以访问的所有环境。这可能包括生产环境、预生产环境和用户验收测试环境。

epic

在敏捷方法学中，有助于组织工作和确定优先级的功能类别。epics 提供了对需求和实施任务的总体描述。例如， AWS CAF 安全史诗包括身份和访问管理、侦探控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中 epics 的更多信息，请参阅计划实施指南。

ERP

请参阅企业资源规划。

探索性数据分析（EDA）

分析数据集以了解其主要特征的过程。您收集或汇总数据，并进行初步调查，以发现模式、检测异常并检查假定情况。EDA 通过计算汇总统计数据和创建数据可视化得以执行。

F

事实表

星型架构中的中心表。它存储有关业务运营的定量数据。通常，事实表包含两种类型的列：包含度量的列和包含维度表外键的列。

快速失效机制

一种使用频繁且增量式的测试来缩短开发生命周期的理念。这是敏捷方法的关键部分。

故障隔离边界

在中 AWS Cloud，诸如可用区 AWS 区域、控制平面或数据平面之类的边界，它限制了故障的影响并有助于提高工作负载的弹性。有关更多信息，请参阅 AWS 故障隔离边界。

功能分支

请参阅分支。

特征

您用来进行预测的输入数据。例如，在制造环境中，特征可能是定期从生产线捕获的图像。

特征重要性

特征对于模型预测的重要性。这通常表示为数值分数，可以通过各种技术进行计算，例如 Shapley 加法解释（SHAP）和积分梯度。有关更多信息，请参阅机器学习模型的可解释性 AWS。

功能转换

为 ML 流程优化数据，包括使用其他来源丰富数据、扩展值或从单个数据字段中提取多组信息。这使得 ML 模型能从数据中获益。例如，如果您将“2021-05-27 00:15:37”日期分解为“2021”、“五月”、“星期四”和“15”，则可以帮助学习与不同数据成分相关的算法学习精细模式。

少样本提示

在要求 LLM 执行类似任务之前，先向其提供少量示例，以演示任务和预期输出。这种技术是情境学习的应用，模型可以从提示中嵌入的示例（镜头）中学习。 Few-shot 对于需要特定格式、推理或领域知识的任务，提示可能非常有效。另请参阅零样本提示。

FGAC

请参阅精细访问控制。

精细访问控制（FGAC）

使用多个条件允许或拒绝访问请求。

快闪迁移

一种数据库迁移方法，通过更改数据捕获使用连续数据复制，在极短的时间内迁移数据，而非使用分阶段方法。目标是将停机时间降至最低。

FM

请参阅基础模型。

基础模型（FM）

一个大型深度学习神经网络，它已使用海量的通用和未标注数据集进行训练。FM 能够执行各种常规任务，例如理解语言、生成文本和图像以及使用自然语言进行对话。有关更多信息，请参阅什么是基础模型。

FM 网关

一种集中式中介，用于控制和规范对基础模型的访问。也称为 LLM 网关。

G

生成式人工智能

AI 模型的一个子集，这些模型已经过大量数据训练，可以使用简单的文本提示来创建新的内容和构件，例如图像、视频、文本和音频。有关更多信息，请参阅什么是生成式人工智能。

地理阻止

请参阅地理限制。

地理限制（地理阻止）

在 Amazon 中 CloudFront，一种阻止特定国家/地区的用户访问内容分发的选项。您可以使用允许列表或阻止列表来指定已批准和已禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档中的限制内容的地理分布。

GitFlow 工作流程

一种方法，在这种方法中，下层和上层环境在源代码存储库中使用不同的分支。Gitflow 工作流程被认为是传统的工作流程，而基于中继的工作流程则是现代的、首选的方法。

黄金映像

系统或软件的快照，用作部署该系统或软件的新实例的模板。例如，在制造业中，黄金映像可用于在多个设备上预调配软件，并有助于提高设备制造操作的速度、可扩展性和生产效率。

全新策略

在新环境中缺少现有基础设施。在对系统架构采用全新策略时，您可以选择所有新技术，而不受对现有基础设施（也称为棕地）兼容性的限制。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

防护机制

一种高级规则，用于跨组织单位（OU）管理资源、策略和合规性。预防性防护机制会执行策略以确保符合合规性标准。它们是使用服务控制策略和 IAM 权限边界实现的。侦测性护栏会检测策略违规和合规性问题，并生成提醒以进行修复。它们通过使用 AWS Config、、Amazon、 AWS Security Hub CSPM GuardDuty AWS Trusted Advisor、Amazon Inspector 和自定义 AWS Lambda 支票来实现。

护栏 (AI)

用于过滤、验证和限制代理输入和输出的安全机制，有助于确保负责任和安全的 AI 行为。

H

HA

请参阅高可用性。

异构数据库迁移

将源数据库迁移到使用不同数据库引擎的目标数据库（例如，从 Oracle 迁移到 Amazon Aurora）。异构迁移通常是重新架构工作的一部分，而转换架构可能是一项复杂的任务。AWS 提供了 AWS SCT 来帮助实现架构转换。

高可用性（HA）

在遇到挑战或灾难时，工作负载无需干预即可连续运行的能力。HA 系统旨在自动进行故障转移、持续提供良好性能，并以最小的性能影响处理不同负载和故障。

历史数据库现代化

一种用于实现运营技术（OT）系统现代化和升级以更好满足制造业需求的方法。历史数据库是一种用于收集和存储工厂中各种来源数据的数据库。

保留数据

从用于训练机器学习模型的数据集中保留的一部分标注的历史数据。通过将模型预测与保留数据进行比较，您可以使用保留数据来评估模型性能。

人机在圈 (HitL)

一种工作流程模式，其中代理执行在关键决策点暂停以供人工审查和批准。

同构数据库迁移

将源数据库迁移到共享同一数据库引擎的目标数据库（例如，从 Microsoft SQL Server 迁移到 Amazon RDS for SQL Server）。同构迁移通常是更换主机或更换平台工作的一部分。您可以使用本机数据库实用程序来迁移架构。

热数据

经常访问的数据，例如实时数据或近期的转化数据。这些数据通常需要高性能存储层或存储类别才能提供快速的查询响应。

修补程序

针对生产环境中关键问题的紧急修复。由于其紧迫性，修补程序通常是在典型的 DevOps 发布工作流程之外进行的。

hypercare 周期

割接之后，迁移团队立即管理和监控云中迁移的应用程序以解决任何问题的时间段。通常，这个周期持续 1-4 天。在 hypercare 周期结束时，迁移团队通常会将应用程序的责任移交给云运营团队。

我

IaC

请参阅基础设施即代码。

基于身份的策略

附加到一个或多个 IAM 委托人的策略，用于定义他们在 AWS Cloud 环境中的权限。

空闲应用程序

90 天内平均 CPU 和内存使用率在 5% 到 20% 之间的应用程序。在迁移项目中，通常会停用这些应用程序或将其保留在本地。

IIoT

请参阅工业物联网。

不可变基础设施

一种模型，可为生产工作负载部署新的基础设施，而不是更新、修补或修改现有基础设施。不可变基础设施本质上比可变基础设施更一致、更可靠、更可预测。有关更多信息，请参阅框架中的使用不可变基础架构部署最佳实践。 AWS Well-Architected

入站（入口）VPC

在 AWS 多账户架构中，一种接受、检查和路由来自应用程序外部的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

增量迁移

一种割接策略，在这种策略中，您可以将应用程序分成小部分进行迁移，而不是一次性完整割接。例如，您最初可能只将几个微服务或用户迁移到新系统。在确认一切正常后，您可以逐步迁移其他微服务或用户，直到停用遗留系统。这种策略降低了大规模迁移带来的风险。

工业 4.0

该术语由克劳斯·施瓦布（Klaus Schwab）在2016年推出，指的是通过连接性、实时数据、自动化、分析和的进步实现制造流程的现代化。 AI/ML

基础设施

应用程序环境中包含的所有资源和资产。

基础设施即代码（IaC）

通过一组配置文件预调配和管理应用程序基础设施的过程。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。

工业物联网（IIoT）

在工业领域使用联网的传感器和设备，例如制造业、能源、汽车、医疗保健、生命科学和农业。有关更多信息，请参阅制定工业物联网（IIoT）数字化转型策略。

检查 VPC

在 AWS 多账户架构中，一种集中式 VPC，用于管理 VPC（相同或不同 AWS 区域）、互联网和本地网络之间的网络流量检查。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

物联网（IoT）

由带有嵌入式传感器或处理器的连接物理对象组成的网络，这些传感器或处理器通过互联网或本地通信网络与其他设备和系统进行通信。有关更多信息，请参阅什么是 IoT？

可解释性

它是机器学习模型的一种特征，描述了人类可以理解模型的预测如何取决于其输入的程度。有关更多信息，请参阅机器学习模型的可解释性 AWS。

物联网

请参阅物联网。

IT 信息库（ITIL）

提供 IT 服务并使这些服务符合业务要求的一套最佳实践。ITIL 是 ITSM 的基础。

IT 服务管理（ITSM）

为组织设计、实施、管理和支持 IT 服务的相关活动。有关将云运营与 ITSM 工具集成的信息，请参阅运营集成指南。

ITIL

请参阅 IT 信息库。

ITSM

请参阅 IT 服务管理。

L

基于标签的访问控制（LBAC）

强制访问控制（MAC）的一种实施方式，其中明确为用户和数据本身分配了安全标签值。用户安全标签和数据安全标签之间的交集决定了用户可以看到哪些行和列。

登录区

landing zone 是一个架构精良的多账户 AWS 环境，具有可扩展性和安全性。这是一个起点，您的组织可以从这里放心地在安全和基础设施环境中快速启动和部署工作负载和应用程序。有关登录区的更多信息，请参阅设置安全且可扩展的多账户 AWS 环境。

大语言模型（LLM）

一种基于大量数据进行预训练的深度学习 AI 模型。LLM 可以执行多项任务，例如回答问题、总结文档、将文本翻译成其他语言以及完成句子。有关更多信息，请参阅什么是 LLM。

大规模迁移

迁移 300 台或更多服务器。

LBAC

请参阅基于标签的访问控制。

最低权限

授予执行任务所需的最低权限的最佳安全实践。有关更多信息，请参阅 IAM 文档中的应用最低权限许可。

直接迁移

请参阅 7 R。

小端序系统

一个先存储最低有效字节的系统。另请参阅字节顺序。

LLM

请参阅大型语言模型。

下层环境

请参阅环境。

M

机器学习（ML）

一种使用算法和技术进行模式识别和学习的人工智能。ML 对记录的数据（例如物联网（IoT）数据）进行分析和学习，以生成基于模式的统计模型。有关更多信息，请参阅机器学习。

主分支

请参阅分支。

恶意软件

旨在危害计算机安全或隐私的软件。恶意软件可能会破坏计算机系统、泄露敏感信息或获得未经授权的访问权限。恶意软件的示例包括病毒、蠕虫、勒索软件、木马、间谍软件和键盘记录器。

托管式服务

AWS 服务 它 AWS 运行基础设施层、操作系统和平台，您可以访问端点来存储和检索数据。Amazon Simple Storage Service（Amazon S3）和 Amazon DynamoDB 就是托管服务的示例。这些服务也称为抽象服务。

制造执行系统（MES）

一种软件系统，用于跟踪、监控、记录和控制在车间将原材料转化为成品的生产过程。

MAP

请参阅迁移加速计划。

MCP

参见模型上下文协议。

模型上下文协议（MCP）

一种用于代理与工具通信的无状态协议。

MCP 服务器

一种通过模型上下文协议公开一个或多个工具的服务。

机制

一个完整的过程，您可以在其中创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运作过程中自我强化和改善的循环。有关更多信息，请参阅在 AWS Well-Architected 框架中构建机制。

成员账户

AWS 账户 除属于组织中的管理账户之外的所有账户 AWS Organizations。一个账户一次只能是一个组织的成员。

MES

请参阅制造执行系统。

消息队列遥测传输（MQTT）

一种基于publish/subscribe模式的轻量级机器对机器 (M2M) 通信协议，适用于资源受限的物联网设备。

微服务

一种小型独立服务，通过明确定义的 API 进行通信，通常由小型独立团队拥有。例如，保险系统可能包括映射到业务能力（如销售或营销）或子域（如购买、理赔或分析）的微服务。微服务的好处包括敏捷、灵活扩展、易于部署、可重复使用的代码和恢复能力。有关更多信息，请参阅使用 AWS 无服务器服务集成微服务。

微服务架构

一种使用独立组件构建应用程序的方法，这些组件将每个应用程序进程作为微服务运行。这些微服务使用轻量级 API 通过明确定义的接口进行通信。该架构中的每个微服务都可以更新、部署和扩展，以满足对应用程序特定功能的需求。有关更多信息，请参阅在上实现微服务。 AWS

迁移加速计划（MAP）

AWS 该计划提供咨询支持、培训和服务，以帮助组织为迁移到云奠定坚实的运营基础，并帮助抵消迁移的初始成本。MAP 提供了一种以系统的方式执行遗留迁移的迁移方法，以及一套用于自动执行和加速常见迁移场景的工具。

大规模迁移

将大部分应用程序组合分波迁移到云中的过程，在每一波中以更快的速度迁移更多应用程序。本阶段使用从早期阶段获得的最佳实践和经验教训，实施由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付简化工作负载的迁移。这是 AWS 迁移策略的第三阶段。

迁移工厂

Cross-functional 通过自动化、敏捷的方法简化工作负载迁移的团队。迁移工厂团队通常包括运营、业务分析师和所有者、迁移工程师、开发 DevOps 人员和冲刺专业人员。20% 到 50% 的企业应用程序组合由可通过工厂方法优化的重复模式组成。有关更多信息，请参阅本内容集中有关迁移工厂的讨论和云迁移工厂指南。

迁移元数据

有关完成迁移所需的应用程序和服务器的信息。每种迁移模式都需要一套不同的迁移元数据。迁移元数据的示例包括目标子网、安全组和 AWS 账户。

迁移模式

一种可重复的迁移任务，详细列出了迁移策略、迁移目标以及所使用的迁移应用程序或服务。示例：使用 AWS 应用程序迁移服务重新托管向 Amazon EC2 的迁移。

迁移组合评测（MPA）

一种在线工具，提供了用于验证迁移到 AWS Cloud的业务案例的信息。MPA 提供了详细的组合评测（服务器规模调整、定价、TCO 比较、迁移成本分析）以及迁移计划（应用程序数据分析和数据收集、应用程序分组、迁移优先级排序和波次规划）。所有 AWS 顾问和 APN 合作伙伴顾问均可免费使用 MPA 工具（需要登录）。

迁移准备情况评测（MRA）

使用 AWS CAF 深入了解组织的云就绪状态、确定优势和劣势以及制定行动计划以缩小已发现差距的过程。有关更多信息，请参阅迁移准备指南。MRA 是 AWS 迁移策略的第一阶段。

迁移策略

将工作负载迁移到 AWS Cloud的方法。有关更多信息，请参见术语表中的 7 R 词条，以及动员您的组织以加快大规模迁移。

ML

请参阅机器学习。

现代化

将过时的（原有的或单体）应用程序及其基础设施转变为云中敏捷、弹性和高度可用的系统，以降低成本、提高效率和利用创新。有关更多信息，请参阅在 AWS Cloud中实现应用程序现代化的策略。

现代化准备情况评估

一种评估方式，有助于确定组织应用程序的现代化准备情况；确定收益、风险和依赖关系；确定组织能够在多大程度上支持这些应用程序的未来状态。评估结果是目标架构的蓝图、详细说明现代化进程发展阶段和里程碑的路线图以及解决已发现差距的行动计划。有关更多信息，请参阅在 AWS Cloud中评估应用程序的现代化准备情况。

单体应用程序（单体式）

作为具有紧密耦合进程的单个服务运行的应用程序。单体应用程序有几个缺点。如果某个应用程序功能的需求激增，则必须扩展整个架构。随着代码库的增长，添加或改进单体应用程序的功能也会变得更加复杂。若要解决这些问题，可以使用微服务架构。有关更多信息，请参阅将单体分解为微服务。

MPA

请参阅迁移组合评测。

MQTT

请参阅消息队列遥测传输。

多分类器

一种帮助为多个类别生成预测（预测两个以上结果之一）的过程。例如，ML 模型可能会询问“这个产品是书、汽车还是手机？”或“此客户最感兴趣什么类别的产品？”

可变基础设施

一种用于更新和修改生产工作负载的现有基础设施的模型。为了提高一致性、可靠性和可预测性，该 AWS Well-Architected 框架建议使用不可变基础设施作为最佳实践。

O

OAC

请参阅来源访问控制。

OAI

请参阅来源访问身份。

OCM

请参阅组织变革管理。

离线迁移

一种迁移方法，在这种方法中，源工作负载会在迁移过程中停止运行。这种方法会延长停机时间，通常用于小型非关键工作负载。

OI

请参阅运营集成。

OLA

请参阅运营级别协议。

在线迁移

一种迁移方法，在这种方法中，源工作负载无需离线即可复制到目标系统。在迁移过程中，连接工作负载的应用程序可以继续运行。这种方法的停机时间为零或最短，通常用于关键生产工作负载。

OPC-UA

请参阅开放流程通信 – 统一架构。

开放流程通信-统一架构 (OPC-UA)

一种用于工业自动化的机器对机器 (M2M) 通信协议。 OPC-UA 提供了数据加密、身份验证和授权方案的互操作性标准。

运营级别协议（OLA）

一项协议，阐明了 IT 职能部门承诺相互交付的内容，以支持服务水平协议（SLA）。

运营准备情况审查（ORR）

一份问题核对清单和关联的最佳实践，可帮助您了解、评估、预防或缩小事件和可能的故障的范围。有关更多信息，请参阅 AWS Well-Architected 框架中的运营准备情况审查 (ORR)。

运营技术（OT）

与物理环境配合使用以控制工业运营、设备和基础设施的硬件和软件系统。在制造业中，OT 和信息技术（IT）系统的集成是工业 4.0 转型的关键重点。

运营整合（OI）

在云中实现运营现代化的过程，包括就绪计划、自动化和集成。有关更多信息，请参阅运营整合指南。

组织跟踪

由 AWS CloudTrail 此创建的跟踪记录组织 AWS 账户 中所有人的所有事件 AWS Organizations。该跟踪是在每个 AWS 账户 中创建的，属于组织的一部分，并跟踪每个账户的活动。有关更多信息，请参阅 CloudTrail文档中的为组织创建跟踪。

组织变革管理（OCM）

一个从人员、文化和领导力角度管理重大、颠覆性业务转型的框架。OCM 通过加快变革采用、解决过渡问题以及推动文化和组织变革，帮助组织为新系统和战略做好准备和过渡。在 AWS 迁移策略中，该框架被称为人员加速，因为云采用项目需要变更的速度。有关更多信息，请参阅 OCM 指南。

来源访问控制（OAC）

在中 CloudFront，一个增强的选项，用于限制访问以保护您的亚马逊简单存储服务 (Amazon S3) 内容。OAC 全部支持所有 S3 存储桶 AWS 区域、使用 AWS KMS (SSE-KMS) 进行服务器端加密，以及对 S3 存储桶的动态PUT和DELETE请求。

来源访问身份（OAI）

在中 CloudFront，一个用于限制访问权限以保护您的 Amazon S3 内容的选项。当您使用 OAI 时，CloudFront 会创建一个 Amazon S3 可以对其进行身份验证的委托人。经过身份验证的委托人只能通过特定 CloudFront 分配访问 S3 存储桶中的内容。另请参阅 OAC，其中提供了更精细和增强的访问控制。

ORR

请参阅运营准备情况审查。

OT

请参阅运营技术。

出站（出口）VPC

在 AWS 多账户架构中，一种处理从应用程序内部启动的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

P

权限边界

附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可以拥有的最大权限。有关更多信息，请参阅 IAM 文档中的权限边界。

个人身份信息（PII）

直接查看其他相关数据或与之配对时可用于合理推断个人身份的信息。PII 的示例包括姓名、地址和联系信息。

PII

请参阅个人身份信息。

playbook

一套预定义的步骤，用于捕获与迁移相关的工作，例如在云中交付核心运营功能。playbook 可以采用脚本、自动化运行手册的形式，也可以是操作现代化环境所需的流程或步骤的摘要。

PLC

请参阅可编程逻辑控制器。

PLM

请参阅产品生命周期管理。

policy

一个对象，可以定义权限（请参阅基于身份的策略）、指定访问条件（请参阅基于资源的策略）或定义 AWS Organizations 的组织中所有账户的最大权限（请参阅服务控制策略）。

多语言持久性

根据数据访问模式和其他要求，独立选择微服务的数据存储技术。如果您的微服务采用相同的数据存储技术，它们可能会遇到实现难题或性能不佳。如果微服务使用最适合其需求的数据存储，则可以更轻松地实现微服务，并获得更好的性能和可扩展性。

组合评测

一个发现、分析和确定应用程序组合优先级以规划迁移的过程。有关更多信息，请参阅评估迁移准备情况。

谓词

返回 `true` 或 `false` 的查询条件，通常位于 `WHERE` 子句中。

谓词下推

一种数据库查询优化技术，可在传输之前筛选查询中的数据。这将减少从关系数据库检索和处理的数据量，并提高查询性能。

预防性控制

一种安全控制，旨在防止事件发生。这些控制是第一道防线，帮助防止未经授权的访问或对网络的意外更改。有关更多信息，请参阅在 AWS上实施安全控制中的预防性控制。

主体

中 AWS 可以执行操作和访问资源的实体。此实体通常是 IAM 角色的根用户或用户。 AWS 账户有关更多信息，请参阅 IAM 文档中角色术语和概念中的主体。

隐私设计

一种在整个开发过程中都考虑隐私的系统工程方法。

私有托管区

私有托管区就是一个容器，其中包含的信息说明您希望 Amazon Route 53 如何响应一个或多个 VPC 中的某个域及其子域的 DNS 查询。有关更多信息，请参阅 Route 53 文档中的私有托管区的使用。

主动控制

一种安全控制，旨在防止部署不合规资源。这些控制会在资源预置之前对其进行扫描。如果资源与控制不兼容，则不会预置它。有关更多信息，请参阅 AWS Control Tower 文档中的控制参考指南，并参见在上实施安全控制中的主动控制 AWS。

产品生命周期管理（PLM）

对产品在其整个生命周期内的数据和流程的管理，从设计、开发和发布，到增长和成熟，再到衰退和淘汰。

生产环境

请参阅环境。

可编程逻辑控制器（PLC）

在制造业中，一种高度可靠、适应性强的计算机，用于监控机器并实现制造过程自动化。

提示串接

使用一个 LLM 提示的输出作为下一个提示的输入，以生成更好的响应。该技术用于将复杂的任务分解为子任务，或者迭代地完善或扩展初步响应。它有助于提高模型响应的准确性和相关性，并允许获得更精细的个性化结果。

假名化

用占位符值替换数据集中个人标识符的过程。假名化可以帮助保护个人隐私。假名化数据仍被视为个人数据。

publish/subscribe (pub/sub)

一种支持微服务间异步通信的模式，可提高可扩展性和响应能力。例如，在基于微服务的 MES 中，微服务可以将事件消息发布到其他微服务可以订阅的频道。系统可以在不更改发布服务的情况下添加新的微服务。

Q

查询计划

一系列用于访问 SQL 关系数据库系统中的数据的步骤，类似于指令。

查询计划回归

当数据库服务优化程序选择的最佳计划不如数据库环境发生特定变化之前时。这可能是由统计数据、约束、环境设置、查询参数绑定更改和数据库引擎更新造成的。

R

RACI 矩阵

请参阅责任、问责、咨询和知情（RACI）。

RAG

请参阅检索增强生成。

勒索软件

一种恶意软件，旨在阻止对计算机系统或数据的访问，直到付款为止。

RASCI 矩阵

请参阅责任、问责、咨询和知情（RACI）。

RCAC

请参阅行列访问控制。

只读副本

用于只读目的的数据库副本。您可以将查询路由到只读副本，以减轻主数据库的负载。

重新架构

请参阅 7 R。

恢复点目标（RPO）

自上一个数据恢复点以来可接受的最长时间。这决定了从上一个恢复点到服务中断之间可接受的数据丢失情况。

恢复时间目标（RTO）

服务中断和服务恢复之间可接受的最大延迟。

重构

请参阅 7 R。

Region

地理区域内的 AWS 资源集合。每一个 AWS 区域 都相互隔离，彼此独立，以提供容错、稳定性和弹性。有关更多信息，请参阅指定您的账户可以使用的 AWS 区域。

回归

一种预测数值的 ML 技术。例如，要解决“这套房子的售价是多少？”的问题 ML 模型可以使用线性回归模型，根据房屋的已知事实（如建筑面积）来预测房屋的销售价格。

重新托管

请参阅 7 R。

版本

在部署过程中，推动生产环境变更的行为。

重新放置

请参阅 7 R。

更换平台

请参阅 7 R。

重新购买

请参阅 7 R。

韧性

应用程序抵御中断或从中断中恢复的能力。在 AWS Cloud中规划韧性时，高可用性和灾难恢复是常见的考虑因素。有关更多信息，请参阅 AWS Cloud 韧性。

基于资源的策略

一种附加到资源的策略，例如 AmazonS3 存储桶、端点或加密密钥。此类策略指定了允许哪些主体访问、支持的操作以及必须满足的任何其他条件。

责任、问责、咨询和知情（RACI）矩阵

定义参与迁移活动和云运营的所有各方的角色和责任的矩阵。矩阵名称源自矩阵中定义的责任类型：负责（R）、问责（A）、咨询（C）和知情（I）。支持（S）类型是可选的。如果包括支持，则该矩阵称为 RASCI 矩阵，如果将其排除在外，则称为 RACI 矩阵。

响应性控制

一种安全控制，旨在推动对不良事件或偏离安全基线的情况进行修复。有关更多信息，请参阅在AWS上实施安全控制中的响应性控制。

保留

请参阅 7 R。

停用

请参阅 7 R。

检索增强生成（RAG）

一种生成式人工智能技术，其中 LLM 在生成响应之前引用其训练数据来源之外的权威数据来源。例如，RAG 模型可以对组织的知识库或自定义数据执行语义搜索。有关更多信息，请参阅什么是 RAG。

轮换

定期更新密钥以使攻击者更难访问凭证的过程。

行列访问控制（RCAC）

使用已定义访问规则的基本、灵活的 SQL 表达式。RCAC 由行权限和列掩码组成。

RPO

请参阅恢复点目标。

RTO

请参阅恢复时间目标。

运行手册

执行特定任务所需的一套手动或自动程序。它们通常是为了简化重复性操作或高错误率的程序而设计的。

S

SAML 2.0

许多身份提供商 (IdPs) 使用的开放标准。此功能支持联合单点登录 (SSO)，因此用户无需在 IAM 中为组织中的所有人创建用户即可登录 AWS 管理控制台 或调用 AWS API 操作。有关基于 SAML 2.0 的联合身份验证的更多信息，请参阅 IAM 文档中的关于基于 SAML 2.0 的联合身份验证。

SCADA

请参阅监督控制和数据采集。

SCP

请参阅服务控制策略。

机密密钥

在中 AWS Secrets Manager，您以加密形式存储的机密或受限信息，例如密码或用户凭证。它由密钥值及其元数据组成。密钥值可以是二进制、单个字符串或多个字符串。有关更多信息，请参阅 Secrets Manager 文档中的什么是 Amazon Secrets Manager 密钥？。

安全设计

一种在整个开发过程中都考虑安全的系统工程方法。

安全控制

一种技术或管理防护机制，可防止、检测或降低威胁行为体利用安全漏洞的能力。安全控制有以下四种类型：预防性、检测性、响应性和主动性。

安全固化

缩小攻击面，使其更能抵御攻击的过程。这可能包括删除不再需要的资源、实施授予最低权限的最佳安全实践或停用配置文件中不必要的功能等操作。

安全信息和事件管理（SIEM）系统

结合了安全信息管理（SIM）和安全事件管理（SEM）系统的工具和服务。SIEM 系统会收集、监控和分析来自服务器、网络、设备和其他来源的数据，以检测威胁和安全漏洞，并生成警报。

安全响应自动化

一种预定义的程序化操作，旨在自动响应或修复安全事件。这些自动化可作为侦探或响应式安全控制措施，帮助您实施 AWS 安全最佳实践。自动响应操作的示例包括修改 VPC 安全组、修补 Amazon EC2 实例或轮换凭证。

服务器端加密

由接收数据的人在目的地对数据 AWS 服务 进行加密。

服务控制策略（SCP）

一种策略，用于集中控制 AWS Organizations的组织中所有账户的权限。SCP 为管理员可以委托给用户或角色的操作定义了防护机制或设定了限制。您可以将 SCP 用作允许列表或拒绝列表，指定允许或禁止哪些服务或操作。有关更多信息，请参阅 AWS Organizations 文档中的服务控制策略。

服务端点

的入口点的 URL AWS 服务。您可以使用端点，通过编程方式连接到目标服务。有关更多信息，请参阅 AWS 一般参考 中的 AWS 服务 端点。

服务水平协议（SLA）

一份协议，阐明了 IT 团队承诺向客户交付的内容，比如服务正常运行时间和性能。

服务水平指示器（SLI）

对服务性能方面的衡量，例如错误率、可用性或吞吐量。

服务水平目标（SLO）

代表服务运行状况的目标指标，由服务水平指示器衡量。

责任共担模式

描述您在云安全与合规方面共同承担 AWS 的责任的模型。 AWS 负责云的安全，而您则负责云中的安全。有关更多信息，请参阅责任共担模式。

暗影人工智能

在组织内受管控渠道之外构建或使用的未经授权的 AI 应用程序。

SIEM

请参阅安全信息和事件管理系统。

单点故障（SPOF）

应用程序的单个关键组件出现故障，可能会中断系统。

SLA

请参阅服务水平协议。

SLI

请参阅服务水平指示器。

SLO

请参阅服务水平目标。

split-and-seed 模式

一种扩展和加速现代化项目的模式。随着新功能和产品发布的定义，核心团队会拆分以创建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的工作效率，支持快速创新。有关更多信息，请参阅在 AWS Cloud中实现应用程序现代化的分阶段方法。

SPOF

请参阅单点故障。

星型架构

一种数据库组织结构，它使用一个大型事实表来存储事务数据或测量数据，并使用一个或多个较小的维度表来存储数据属性。此结构专为在数据仓库中使用或用于商业智能目的而设计。

strangler fig 模式

一种通过逐步重写和替换系统功能直至可以停用原有的系统来实现单体系统现代化的方法。这种模式用无花果藤作为类比，这种藤蔓成长为一棵树，最终战胜并取代了宿主。该模式是由 Martin

Fowler 提出的，作为重写单体系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅使用容器和 Amazon API Gateway 逐步实现传统微软 ASP.NET (ASMX) 网络服务的现代化。

子网

您的 VPC 内的一个 IP 地址范围。子网必须位于单个可用区中。

监督控制和数据采集（SCADA）

在制造业中，一种使用硬件和软件来监控实物资产和生产操作的系统。

对称加密

一种加密算法，它使用相同的密钥来加密和解密数据。

综合测试

以模拟用户交互的方式测试系统，以检测潜在问题或监控性能。你可以使用 Amazon S CloudWatch ynthetics 来创建这些测试。

系统提示

一种为 LLM 提供上下文、说明或准则以指导其行为的技术。系统提示有助于设置上下文并制定与用户交互的规则。

T

标签

Key-value 对充当用于组织 AWS 资源的元数据。标签有助于您管理、识别、组织、搜索和筛选 资源。有关更多信息，请参阅标记您的 AWS 资源。

目标变量

您在监督式 ML 中尝试预测的值。这也被称为结果变量。例如，在制造环境中，目标变量可能是产品缺陷。

任务列表

一种通过运行手册用于跟踪进度的工具。任务列表包含运行手册的概述和要完成的常规任务列表。对于每项常规任务，它包括预计所需时间、所有者和进度。

测试环境

请参阅环境。

训练

为您的 ML 模型提供学习数据。训练数据必须包含正确答案。学习算法在训练数据中查找将输入数据属性映射到目标（您希望预测的答案）的模式。然后输出捕获这些模式的 ML 模型。然后，您可以使用 ML 模型对不知道目标的新数据进行预测。

工具

代理可以调用以在外部系统中执行操作的函数或 API。

中转网关

中转网关是网络中转中心，您可用它来互连 VPC 和本地网络。有关更多信息，请参阅 AWS Transit Gateway 文档中的什么是公交网关。

基于中继的工作流程

一种方法，开发人员在功能分支中本地构建和测试功能，然后将这些更改合并到主分支中。然后，按顺序将主分支构建到开发、预生产和生产环境。

可信访问权限

向您指定的服务授予权限，该服务可以代表您在其账户中执行任务。 AWS Organizations 当需要服务相关的角色时，受信任的服务会在每个账户中创建一个角色，为您执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的AWS Organizations 与其他 AWS 服务一起使用。

优化

更改训练过程的各个方面，以提高 ML 模型的准确性。例如，您可以通过生成标签集、添加标签，并在不同的设置下多次重复这些步骤来优化模型，从而训练 ML 模型。

双披萨团队

一个小 DevOps 团队，你可以用两个披萨来喂食。双披萨团队的规模可确保在软件开发过程中充分协作。

U

不确定性

这一概念指的是不精确、不完整或未知的信息，这些信息可能会破坏预测式 ML 模型的可靠性。不确定性有两种类型：认知不确定性是由有限的、不完整的数据造成的，而偶然不确定性是由数据中固有的噪声和随机性导致的。

无差别任务

也称为繁重工作，即创建和运行应用程序所必需的工作，但不能为最终用户提供直接价值或竞争优势。无差别任务的示例包括采购、维护和容量规划。

上层环境

请参阅环境。

V

vacuum 操作

一种数据库维护操作，包括在增量更新后进行清理，以回收存储空间并提高性能。

版本控制

跟踪更改的过程和工具，例如存储库中源代码的更改。

VPC 对等连接

两个 VPC 之间的连接，允许您使用私有 IP 地址路由流量。有关更多信息，请参阅 Amazon VPC 文档中的什么是 VPC 对等连接。

漏洞

损害系统安全的软件缺陷或硬件缺陷。

W

热缓存

一种包含经常访问的当前相关数据的缓冲区缓存。数据库实例可以从缓冲区缓存读取，这比从主内存或磁盘读取要快。

暖数据

不常访问的数据。查询此类数据时，通常可以接受中速查询。

窗口函数

一种对与当前记录有某种关联的一组行执行计算的 SQL 函数。窗口函数对于处理任务很有用，例如计算移动平均值或根据当前行的相对位置访问行的值。

工作负载

一系列资源和代码，它们可以提供商业价值，如面向客户的应用程序或后端过程。

工作流

迁移项目中负责一组特定任务的职能小组。每个工作流都是独立的，但支持项目中的其他工作流。例如，组合工作流负责确定应用程序的优先级、波次规划和收集迁移元数据。组合工作流将这些资产交付给迁移工作流，然后迁移服务器和应用程序。

WORM

请参阅一次写入多次读取。

WQF

请参阅 AWS 工作负载资格鉴定框架。

一次写入多次读取（WORM）

一种存储模型，可一次写入数据并防止数据被删除或修改。授权用户可以根据需要多次读取数据，但无法对其进行更改。此数据存储基础设施被认为不可变。

Z

零日漏洞利用

一种利用零日漏洞的攻击，通常为恶意软件。

零日漏洞

生产系统中不可避免的缺陷或漏洞。威胁主体可能利用这种类型的漏洞攻击系统。开发人员经常因攻击而意识到该漏洞。

零样本提示

为 LLM 提供执行任务的说明，但没有可以帮助指导的示例（样本）。LLM 必须使用预先训练的知识来处理任务。零样本提示的有效性取决于任务的复杂性和提示的质量。另请参阅少样本提示。

僵尸应用程序

平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常会停用这些应用程序。

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。