AWS 白皮书

在 AWS 上进行 5G 网络的持续集成和持续交付

在 AWS 上进行 5G 网络的持续集成和持续交付: AWS 白皮书

Table of Contents

在 AWS 上进行 5G 网络的持续集成和持续交付

发布日期：2021 年 3 月 8 日（文档修订）

摘要

本白皮书介绍了 5G 网络的持续集成和持续交付（CI/CD），以及如何使用 Amazon Web Services（AWS）工具和服务来完全自动化 5G 网络功能的部署和升级。本白皮书详细介绍了 5G 网络功能的 CI/CD 的不同阶段，包括网络设置、基础设施部署、云原生网络功能部署以及网络功能的持续更新。它还详细介绍了与用于测试、可观测性和编排的开源和第三方工具的集成。

本白皮书面向通信服务提供商（CSP）和独立软件供应商（ISV）。

引言

一直以来，蜂窝网络中新网络节点或新功能的开发、实验室和现场集成测试以及生产部署需要数周甚至数月的时间，才能确保关键任务和关键业务电信（电信）服务的稳定性。部署周期漫长的原因是传统网络节点的整体架构、多供应商环境以及 2G、3G 和 4G 移动网络中网络实体之间的许多点对点接口。

正如借 AWS 之力促进 5G 网络发展白皮书中所介绍的那样，由 3GPP 标准化的 5G 移动网络现在支持通过虚拟化和容器化实现的云原生架构。更具体地说，5G 网络引入并支持微服务、无状态和基于服务的架构的新范式。

这种 5G 架构意味着不同的网络功能可以作为松散耦合的独立服务工作，通过明确定义的接口和 API 相互通信。最重要的是，每项网络功能都可以独立更新。5G 的这种架构转变使 CSP 能够更轻松且更频繁地推出网络功能更新，同时通过自动化来维持测试、安全要求和标准，从而实现更高的敏捷性和运营效率。

对于 CSP 来说，新功能的集成和部署通常在网络功能供应商发布新的网络功能软件包（例如基于容器的网络功能中的 Docker 镜像）或新的配置文件（如 Kubernetes 应用程序案例中的 Helm Chart）时开始。（Helm Chart 是描述一组相关的 Kubernetes 资源的文件集合。）

将 CI/CD 范式用于 5G 网络功能部署的想法越来越受追捧，但这一想法的实际实现一直是电信行业面临的挑战。

AWS 率先开发了用于软件交付的新 CI/CD 工具，以帮助各行各业快速开发和推出软件变更，同时保持系统的稳定性和安全性。这些工具包括一组软件开发和运营（DevOps）服务，例如 AWS CodeStar、CodeCommit、CodePipeline、CodeBuild 和 CodeDeploy。

AWS 还使用 AWS Cloud Development Kit（AWS CDK）、AWS CloudFormation 以及基于 API 的第三方工具（例如 Terraform）宣传基础设施即代码（IaC）的理念。使用这些工具，AWS 可以将网络功能的部署流程作为源代码存储在 AWS 中，并在 CI/CD 管道中维护此 IaC 源代码，实现持续交付。

本白皮书详细介绍了利用 AWS IaC 和 CI/CD 工具部署和更新 5G 网络功能的流程。此外，本白皮书还介绍了与用于测试、可观测性和编排的第三方工具的集成。

AWS CI/CD 工具不限于 5G 网络功能。它们还用于自动部署 4G 网络，从而使 CSP 能够快速高效地部署和更新 4G 网络功能。大多数 4G 网络功能都是基于虚拟网络功能（VNF）。类似 AWS CloudFormation 的 AWS CI/CD 工具集可用于自动部署 4G VNF，从而为 4G 网络部署带来规模和时间效率。

持续集成和持续交付

持续集成

持续集成（CI）是开发人员定期将其代码推送到中央存储库（如 AWS CodeCommit 或 GitHub）的软件流程。每次代码推送都会触发自动构建，然后运行测试。CI 的主要目标是在早期阶段发现代码问题，提高代码质量，并缩短验证和发布新软件更新所需的时间。

持续交付和部署

持续交付（CD）是将工件部署到测试环境、暂存环境和生产环境的软件流程。持续交付可以是完全自动化的，也可以在关键时刻有审批阶段。这样可以确保部署前所有必需的批准（例如版本管理批准）都已到位。当持续交付得以正确实施时，开发人员将始终能够获得一个已通过标准化测试流程的部署就绪型构建工件。

采用持续部署 时，系统会在未经开发人员明确批准的情况下自动将修订部署到生产环境中，从而实现整个软件发布流程的自动化。这允许在产品生命周期的早期阶段建立持续的客户反馈循环。

采用持续部署时，提交并通过自动化测试的每项更改都会自动发布到生产环境中。持续交付并非旨在发布所有已提交的每项更改并立即将自动化测试传递到生产环境中，而是为了确保每项更改都已准备好投入生产。

基础设施即代码

正如借 AWS 之力促进 5G 网络发展白皮书中所详述的那样，IaC 是实现应用程序及其环境的调配流程和生命周期管理自动化的关键驱动因素。网络/IT 管理员和开发人员都能利用配置文件将基础设施实例化，而不必依赖手动执行的步骤。IaC 将这些配置文件视为软件代码。这些文件可用于生成一组对象，即构成运行环境的计算、存储、网络和应用程序服务。IaC 借助自动化消除了配置漂移，从而提高了基础设施部署的速度和敏捷性。

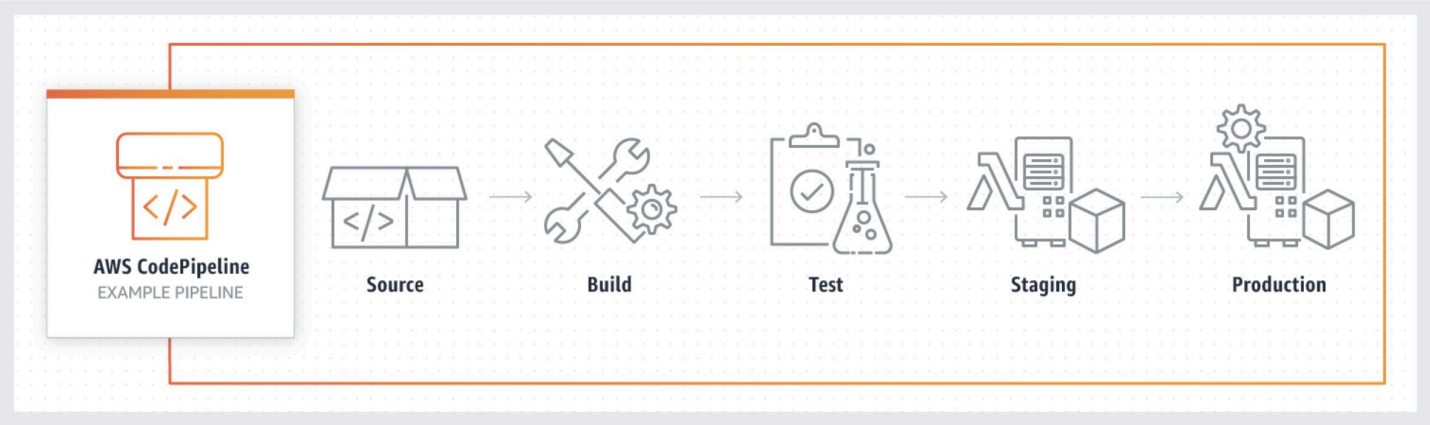
对于在 AWS 上实施网络功能虚拟化（NFV），此 IaC 框架从编排的角度带来了价值。从创建 Virtual Private Cloud（VPC）到网络功能部署，每个步骤都可以作为源代码进行编程、管理，并通过 AWS CodeCommit 中的版本控制进行维护。

这种用于网络功能的 IaC 框架可实现可重复且可靠的基础设施和网络功能的创建和部署，从而扩展到网络切片管理和服务生命周期管理的端到端（E2E）自动化。AWS 提供了一个全面的工具集，用于以

编程、描述性和声明方式创建、维护和部署基础设施，使用 AWS CloudFormation、AWS CDK、AWS CDK for Kubernetes 和 API 暴露的所有 AWS 服务。

AWS 上的 CI/CD

可以将 CI/CD 想象成一个管道，其中新代码在一端提交，在一系列阶段（源代码、构建、测试、暂存和生产）中进行测试，然后作为生产就绪代码发布。



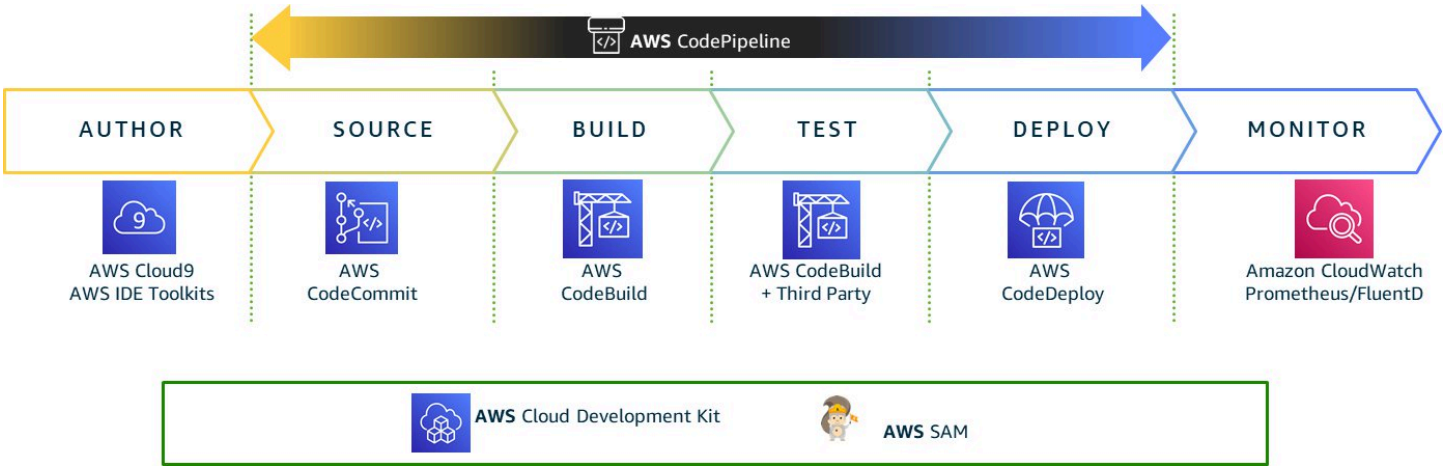
CICD 管道概述

CI/CD 管道的每个阶段都被构造为交付过程中的一个逻辑单元。每个阶段都充当审查代码某个方面的一扇门。随着代码在管道中的进展，假设代码的质量在后面的阶段会更高（因为代码的更多方面持续得到验证）。在早期阶段发现的问题会阻止代码在管道中进展。测试的结果会立即发送给团队，如果软件没有通过该阶段，所有后续的构建和发布工作都会停止。

AWS 引入了一套完整的 CI/CD 开发工具来加快软件开发和发布周期。每当代码发生更改时，AWS CodePipeline 都会根据定义的发布模型，自动执行发布流程的构建、测试和部署阶段。这样可以快速、可靠地交付功能和更新。

代码管道可以与其他服务集成。它们可以是 AWS 服务，例如 Amazon Simple Storage Service（Amazon S3），也可以是 GitHub 等第三方产品。AWS CodePipeline 可以解决各种开发和运营使用案例，包括：

- 使用 AWS CodeBuild 编译、构建和测试代码
- 将基于容器的应用程序持续交付到云
- 部署前验证网络服务或特定云原生网络功能所需的构件（如描述符和容器镜像）
- 容器化网络功能/虚拟网络功能（CNF/VNF）的功能、集成和性能测试，包括基线测试和回归测试
- 可靠性和灾难恢复（DR）测试。



AWS CICD 管道组件

AWS 可以使用以下 AWS 开发工具设置 CI/CD 管道：

- AWS CodeCommit
- AWS CodeBuild
- AWS CodePipeline
- AWS CodeDeploy
- Amazon Elastic Container Registry
- AWS CodeStar

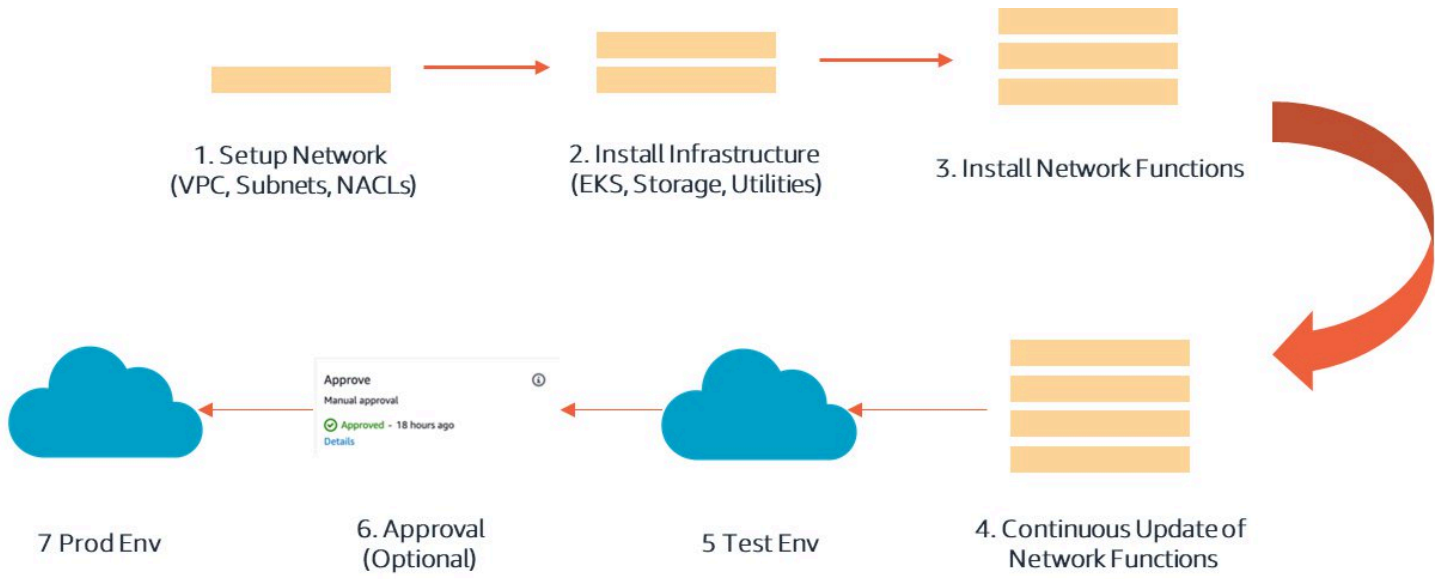
使用 AWS CDK 和 AWS CloudFormation 可以自动创建 CI/CD 管道。在 NFV 域中，这种 AWS 原生自动化可以集成到管理和编排（MANO）框架和 CSP 的服务编排框架中。

CI/CD 过程包括以下步骤：

- 网络设置 – AWS CDK 和 AWS CloudFormation 启动网络先决条件的创建：
    - 网络堆栈（VPC、子网、网络地址转换（NAT）网关、路由表和互联网网关）
- 基础设施部署 – AWS CDK 和AWS CloudFormation 启动以下资源堆栈的创建：
    - 计算堆栈（Amazon Elastic Kubernetes Service（Amazon EKS）集群创建、EKS Worker 节点、AWS Lambda）

  - 存储堆栈（Amazon S3 存储桶、Amazon Elastic Block Store（Amazon EBS）卷和 Amazon Elastic File System（Amazon EFS））
  - 监控堆栈（CloudWatch、Amazon OpenSearch Service（OpenSearch Service））
  - 安全堆栈（AWS Identity and Access Management（AWS IAM）、Amazon Elastic Compute Cloud（Amazon EC2）安全组、VPC 网络访问控制列表（NACL））

- 云网络功能（CNF）部署 – 在这个阶段，CNF 是使用 Kubectl 和 Helm Chart 工具部署到 EKS 集群上的。此阶段还部署了 CNF 高效工作所需的所有特定应用程序或工具（例如 Prometheus 或 Fluentd）。CNF 可以通过 Lambda 函数部署，也可以使用 AWS CodeBuild 部署。
- 持续更新和部署 – 这些是迭代执行的一系列步骤，用于部署作为导致升级的容器/配置更改一部分的更改。与 CNF 部署案例类似，持续更新和部署可以使用 AWS 服务、来自 AWS CodeCommit、Amazon Elastic Container Registry（Amazon ECR）的触发器或第三方源系统（如 GitLab Webhook）实现自动化。



AWS CICD 管道流程图

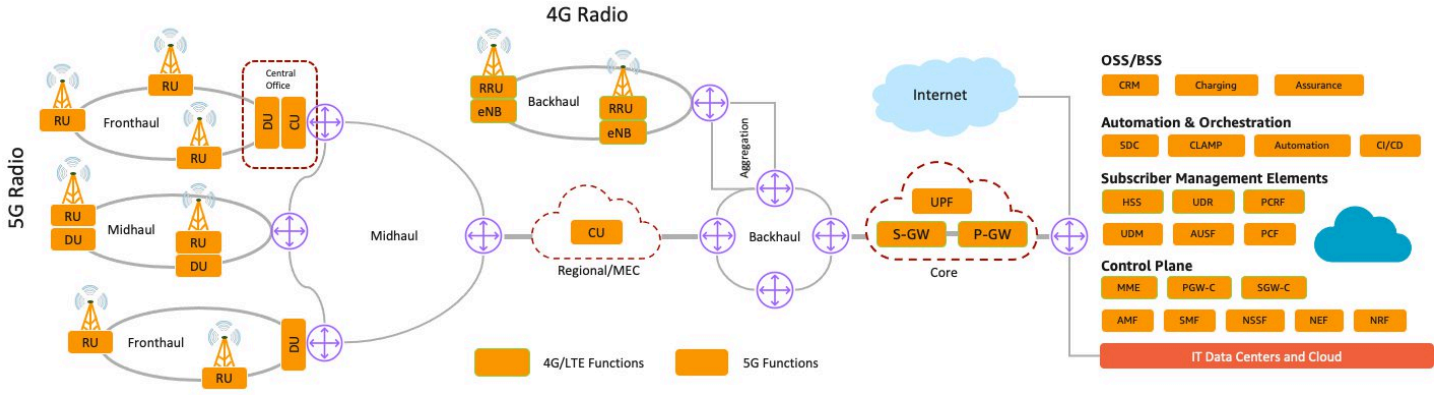
CI/CD 管道是使用 AWS CodePipeline 构建的，它利用持续交付服务对软件发布所需的步骤进行建模、可视化和自动化。通过在管道中定义阶段，您可以从源代码存储库中检索代码，将该源代码构建为可发布的构件，测试构件，然后将其部署到生产环境中。只有成功通过所有这些阶段的代码才会被部署。您可以选择向管道添加其他要求（例如手动批准），以帮助确保仅将批准的更改部署到生产环境中。

AWS 上的 5G 网络

5G 网络基础设施的典型模型由 4G/5G 无线电站点、前传/中传/后传网络、核心网络站点和电信/IT 数据中心组成。CSP 可以使用 AWS 服务创建可扩展、灵活的 5G 网络基础设施，同时降低前期投资成本。AWS 可用于在托管运营支持系统/业务支持系统（OSS/BSS）和大多数控制面板核心网络功能的区域中实施虚拟网络运营中心（NOC）。

也可以利用 AWS 来实施本地中心机构（CO）或分布式数据中心，AWS Outposts 实例队列主要托管用户平面功能，例如 UPF（用户面功能）、RAN 中央单元（CU）和多接入边缘计算（MEC）。借 AWS 之力促进 5G 网络发展白皮书中详细讲述了参考架构以及在 AWS 上实施 5G 网络的好处。

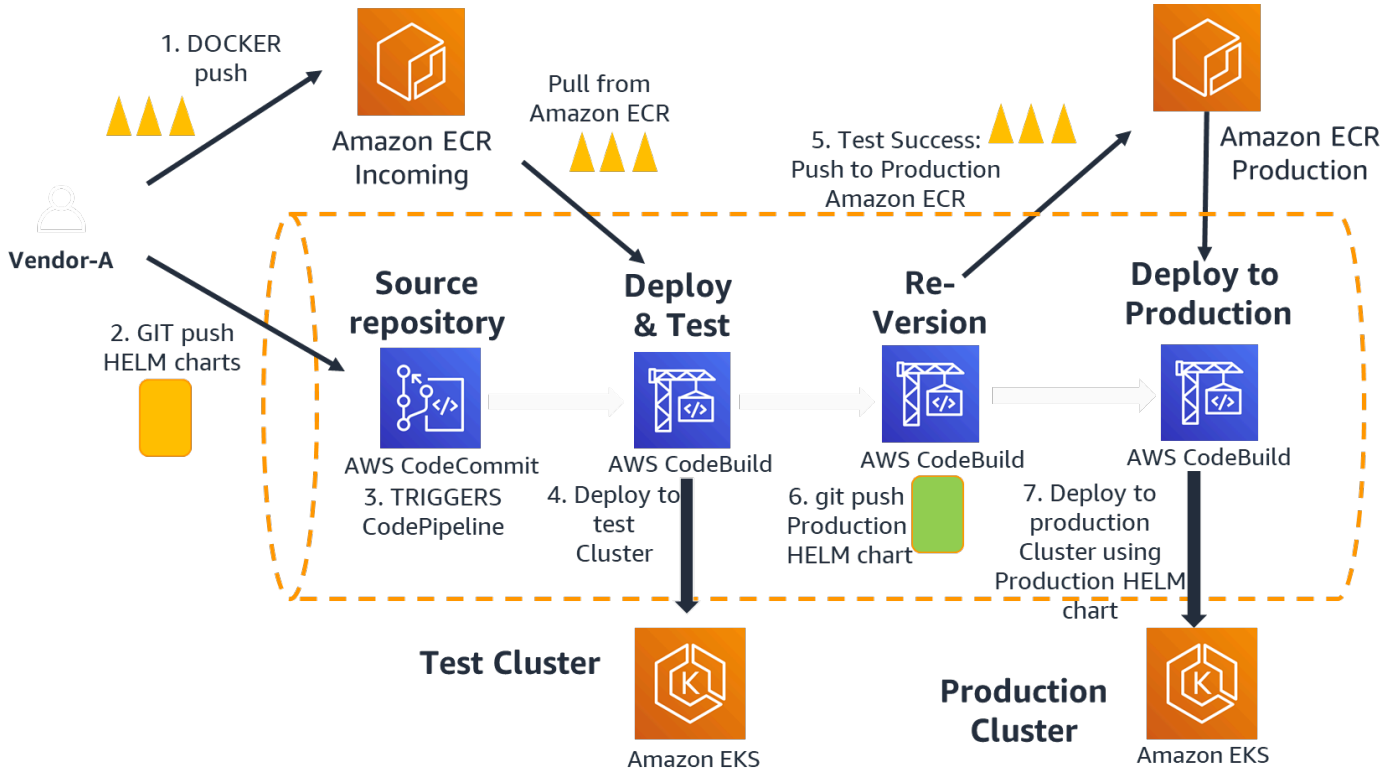
当需要在 AWS 上实施 5G 网络时，本白皮书的以下章节中介绍的 AWS CI/CD 工具可以推进 5G 网络功能的部署、升级和生命周期管理的完全自动化。



5G 网络端到端架构

5G 网络中的 CI/CD

基础设施的设计构造以使用声明式语言的代码形式存储。这使得 CSP 能够根据需要以相同的预期行为再现基础设施。代码在代码存储库中维护，并设置管道以协调对已部署堆栈的更新（例如 AWS CDK 和 AWS CloudFormation）。AWS 可以帮助构建基础设施即代码（IaC），以便敏捷地加入独立软件供应商（ISV）功能。



代码管道流

通过 Helm chart 更改云原生网络功能配置被视为网络功能自动执行 CI/CD 管道的触发器。

AWS CodeCommit 可用于维护配置文件，Amazon ECR 可用于保存容器镜像。

如代码管道流图所示，当 ISV 将新的代码更改推送到代码存储库（Helm Chart、配置文件或属性文件）中时，将触发代码管道。代码管道从 ECR 中提取镜像，然后使用 Helm Chart 来部署应用程序。新的应用程序测试可以与第三方测试自动化框架集成。根据结果，CSP 可以批准生产部署。

CodePipeline 的源代码阶段会查找配置文件中的更改。源代码阶段的有效提供商是 CodeCommit、Amazon S3、GitHub 或 AWS CloudFormation。通过使用 Lambda 函数来实现 Webhook，可以集成替代源系统，从而实现 Gitlab 和 AWS CodePipeline 之间的事件驱动型集成。有关详细的实施指南，请参阅以下链接。

- Webhook 与 GitLab
- 容器注册表集成

CI/CD 管道设计应考虑关键的部署步骤，例如初始部署、测试和在测试结果与预期保持一致并根据基线进行验证后升级到生产环境。管道流程的每个阶段都提供数据构件，从而实现比较和数据驱动型决策。

应用程序 CI/CD 管道步骤

每个阶段都可以视为一项单独的任务，允许合并足以支持网络服务和云原生网络功能的验证和部署工作流。运行任务可以结合其他第三方工具，例如流量生成器和模拟器，从而实现端到端网络服务验证。

AWS 提供复杂的 AWS Step Functions（云原生状态机）服务，该服务可与其他 AWS 服务进行原生集成，还可以与 Jira 或测试自动化框架等外部系统集成。

CI/CD 详细步骤

可以将 CI/CD 想象成一个管道，其中新代码在一端提交，在一系列阶段（源代码、构建、测试、暂存和生产）中进行测试，然后作为生产就绪代码发布。

以下是部署和测试步骤。部署和配置主要分为四个主要部分：

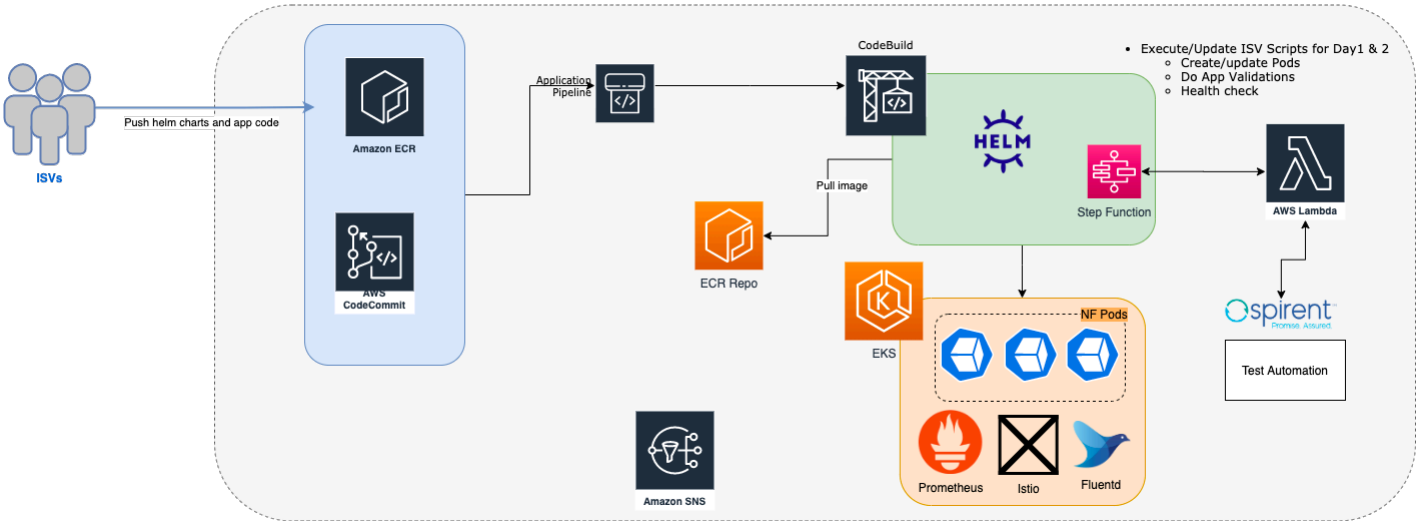
- 网络设置
- 基础设施部署
- 云原生网络功能部署
- CNF 持续交付

网络设置

重点关注基础设施先决条件的设置。这包括创建 VPC、网络、子网、NACL 等。设计 IP 网络计划时要考虑 ISV 和客户的实施计划（例如多租户和静态与动态分配）。此计划可以在 AWS CDK 或 AWS CloudFormation 中编码。运行此代码将部署云基础设施网络先决条件。

基础设施部署

基础设施部署会预置所有基础设施组件。它包括生成 EKS 集群和支持基础设施，如 EFS、EKS Worker 节点、ELB，以及根据云原生网络功能要求配置集群。根据 CNF 的要求，AWS 还会为节点部署额外的网络接口，包括 Multus 接口。大多数部署和配置步骤都是应用程序的一次性任务，并且仅在需要时作为应用程序的更新进行更新。

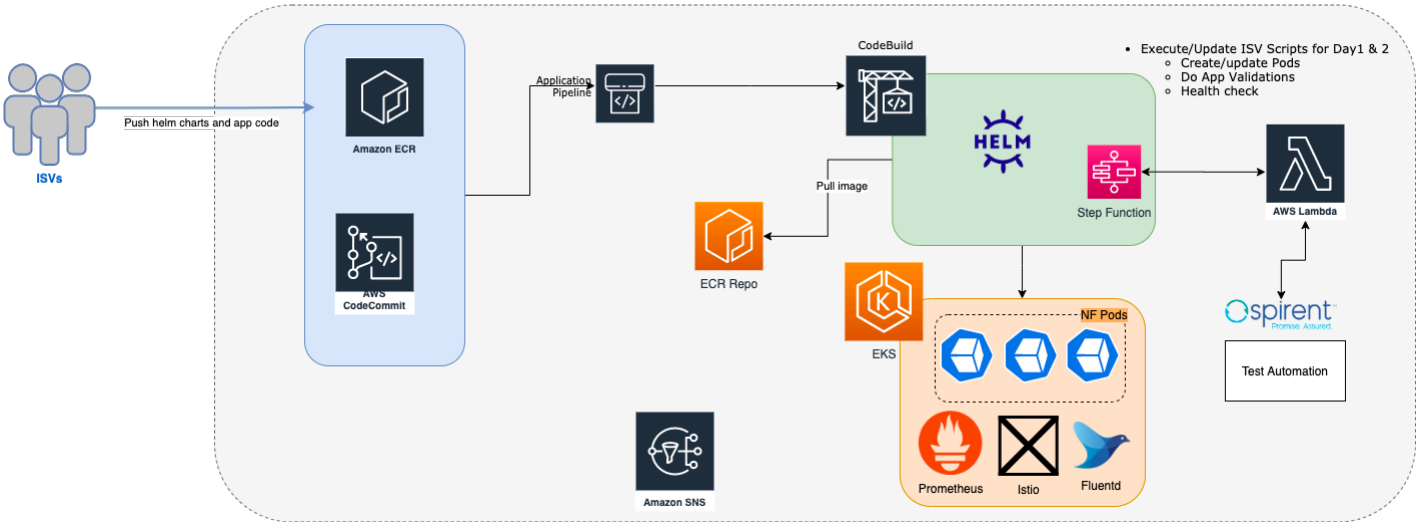


使用 CDK 进行基础设施部署

云原生网络功能部署

CNF 部署与应用程序部署有关。在 CNF 部署过程中，应用程序的 Helm Chart 是通过 CI/CD 代码管道实现的。合并了运行主要涉及检查前后的各个应用程序特定脚本的回调。Helm Chart 是根据应用程序的需求按顺序实现的，在进入部署的下一步之前检查 Kubernetes POD 的状态。通常，ISV 会提供一个包装脚本来运行 Helm Chart 和健全性检查。这些 ISV 脚本是从 AWS CodePipeline 内部调用的。在此阶段中，除了记录和监控应用程序的云基础设施的 Amazon CloudWatch 之外，还部署了诸如 Prometheus 和 Fluentd 之类的日志记录和监控代理。

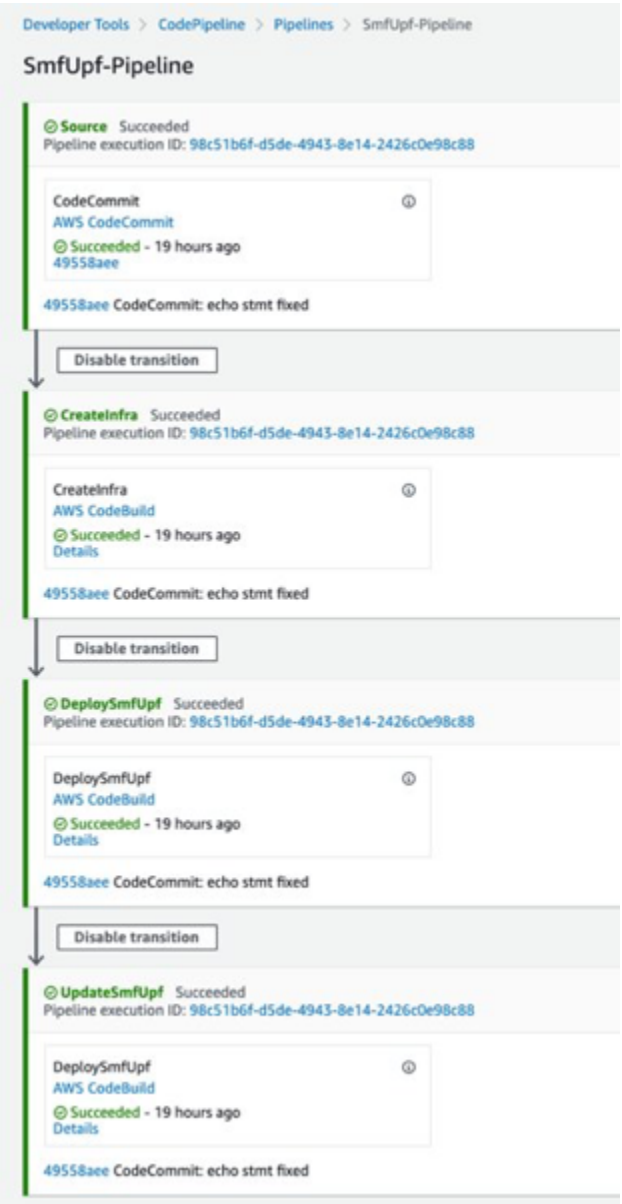
代码管道与第三方测试自动化框架集成。代码管道可以直接调用测试自动化框架 API，以在部署的应用上运行测试，查询测试结果和分析结果。这简化了应用程序的部署和测试。



应用程序部署和更新

以下是通过 AWS CodePipeline 部署用户面功能/会话管理功能（UPF/SMF）CNF 的示例。

- 使用 CodeCommit、CodeBuild 和 CodePipeline 自动执行完整的 CI/CD 流程。
- 基础设施创建和应用程序安装任务作为管道的一部分进行了集成。
- FluentD 和 Prometheus 代理是在 Amazon CloudWatch 控制面板中安装和创建的。



UPF/SMF CNF 的部署示例

CNF 持续交付

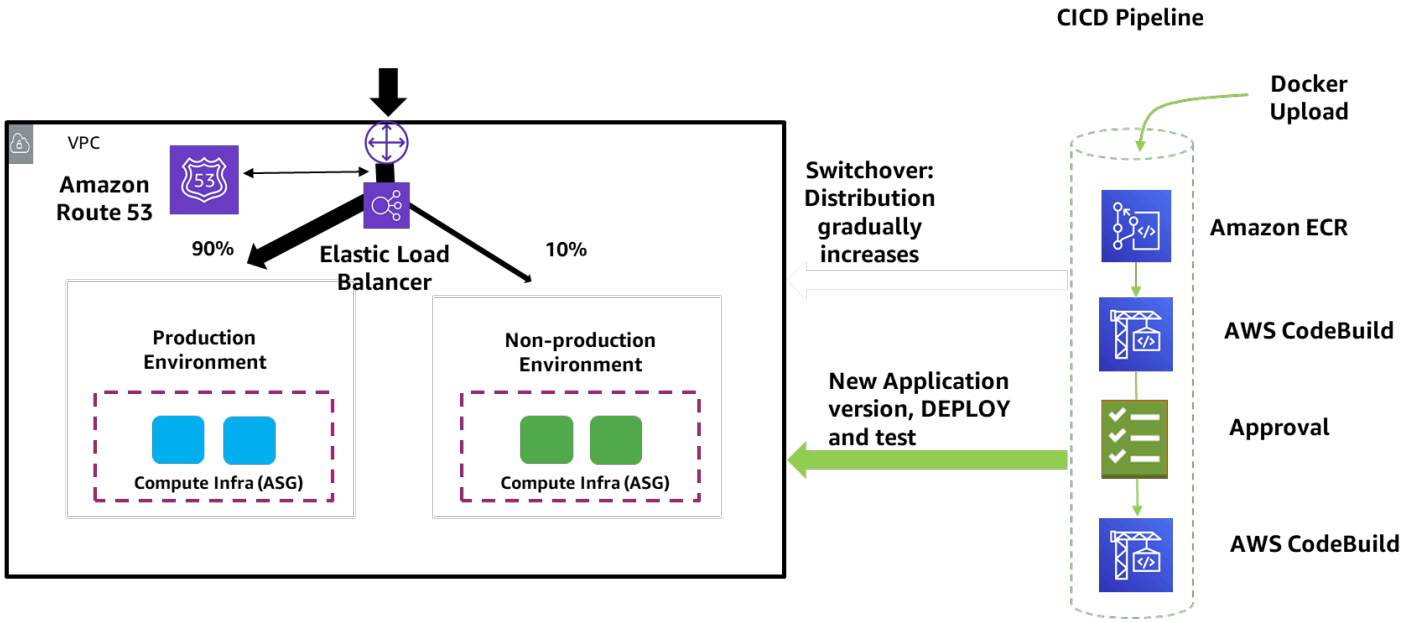
此步骤包含一系列重复执行的步骤，用于部署作为导致升级的容器/配置更改一部分的更改。CNF 持续交付是通过管道实现自动化的，并且特定于个别应用程序。AWS 使用标准的 Helm Chart 来更新特定的 CNF。代码管道具有检查前后的应用程序更新状态。更新后的 CI/CD 管道还与测试自动化框架集成在一起，以运行自动化测试。这种抽象允许干净地部署网络功能。

CNF 持续交付和部署可以大致分为以下几类：

- 应用程序升级 – 大多数应用程序升级都是 Kuberbetes 应用程序 POD 中的更改。这些更新可以通过代码管道自动应用。大多数 CNF 通过提供应用程序 POD 的多个实例来支持就地升级。多个实例允许滚动升级方法。并非所有的应用程序 POD 更改都支持 Helm 升级。管道会考虑这些变化，并根据需要使用 Helm 安装/删除。
- 主要升级– 主要升级主要是数据库架构更改。如果不导致停机，则无法应用此更改。这些更改的标准方法是删除应用程序并重新创建相关的 pod。在此过程中，应用程序可能不可用。以下工具用于升级：

    - AWS CloudFormation 使客户能够在 JSON 或 YAML 模板中描述和预置所有基础设施资源。AWS CloudFormation 通过 Lambda 支持的自定义资源提供强大的扩展机制。客户可以将 AWS CloudFormation 扩展到 AWS 资源以外，并在其他环境中预置所需的资源；例如，混合环境中的本地部署资源。AWS CDK 使开发人员能够使用 Python、TypeScript、JavaScript、Java 和 C# 等更高级别的熟悉编程语言构建代码，然后将代码编译为较低级别的 AWS CloudFormation JSON 格式，随后可以进行部署。
    - BlueGreen 部署 – AWS 支持并建议在测试环境和生产环境中进行蓝/绿部署和基于 canary 的部署。蓝/绿部署使客户能够在受控环境中测试新的应用程序版本。它们提供了一种简单而优雅的方法来切换生产流量。基于 Canary 的部署扩展了这一概念，它允许使用一小部分生产流量测试非生产绿色环境，从而发现由生产流量引起的任何问题。针对内部模拟测试流量和少量生产流量测试新的应用程序版本，这使用户在切换生产流量之前充满信心。在切换完成之前，生产流量会逐渐增加。实施涉及加权 DNS 和加权 ELB 目标组。
    - 自动化可以通过为 AWS CodePipeline 配置蓝/绿部署阶段和基于 canary 的部署阶段来实现。批准阶段最初在调配期间可能是手动进行的，但稍后应该完全自动化。在测试环境中，在部署到生产环境之前，最好始终使用回滚操作进行测试，以验证向前和向后兼容性。具有服务网格的集群上的蓝/绿部署取决于终端应用程序和路由网关为服务网格提供的支持才能实现平稳过渡。

- AWS Systems Manager 提供了一个统一的用户界面，供您查看 CI/CD 部署的网络功能所使用的多个 AWS 服务的运行数据。Systems Manager 使您能够跨 AWS 资源自动执行运营任务。



Canary 版本部署

安全性

安全性是一个关键要素。以下是 AWS CI/CD 流程在部署应用程序时要考虑的安全步骤的列表。

- 源 – 分配给供应商的 ECR 存储库在配置时启用了“推送时扫描”标志，因此上传的所有 Docker 镜像都将立即接受安全扫描。任何已知的常见漏洞和分析（CVE）都将通过通知进行标记。除了 ECR 之外，当供应商将图表放入 AWS CodeCommit 存储库时，还会要求他们加密任何用于 Secrets Manager 的密码，而不是纯文本密码。
- 构件完整性 – 无论是静态（使用 AWS 托管密钥）还是传输（使用 SSL/TLS），整个管道中使用的项目都会经过加密。
- IAM 用户和角色 – 向用户或资源提供的权限基于最低权限原则。如果您要跨不同服务中的资源运营，则可能需要配置跨 IAM 角色信任关系。例如，AWS CodeBuild 需要在 Amazon EKS 集群上运行命令的权限。
- 审计 – AWS CloudTrail 提供的审核功能可跟踪跨服务和用户操作的每个 API 调用，并允许对过去的事件进行评估。

- 镜像漏洞扫描 – 系统会自动扫描上传到 Amazon ECR 的 CNF 镜像是否存在安全漏洞。AWS Management Console 中提供了扫描结果的报告，也可以通过 API 检索。然后，可以将调查结果发送给 CSP 运营商以采取纠正措施，包括更换 CNF 镜像。

安全检查在管道的各个阶段进行，以确保新上传的镜像是安全的，并符合所需的合规性检查，因此可以向 CSP 发送通知以供批准：

- 容器注册表会扫描是否存在任何未解决的 CVE 漏洞。
- 在测试阶段检查配置是否存在信息泄漏、已知个人身份信息（PII）模式，从而触发合规性检查规则，以解决意外开放的 TCP/UDP 端口和 DOS 漏洞等问题。
- 向后和向前兼容性已经过升级/回滚安全验证。

除了应用程序之外，通过确保在各阶段传输（无论是静态还是在传输中）的构件已加密来预置管道安全性至关重要。

可观测性

AWS 为默认情况下部署在 AWS 上的 5G CNF 启用了可观测性。这是通过 Amazon CloudWatch 实现的。CloudWatch 让您全面了解您的云资源和应用程序。

在此过程中，Amazon CloudWatch 有四个主要步骤：

1. 收集 – 从在 AWS 和本地服务器上运行的所有 AWS 资源、应用程序和服务中收集指标和日志。
2. 监控 – 使用 CloudWatch 控制面板直观呈现应用程序和基础设施，并排关联日志和指标以进行故障排除，并使用 CloudWatch 告警设置告警。
3. 行动 – 使用 CloudWatch Events 和 AWS Auto Scaling 自动响应操作更改。
4. 分析 – 使用 CloudWatch Metric Math，获得最长一秒钟的指标、更长的数据留存期（15 个月）和实时分析。

Amazon CloudWatch 代理安装在客户的 Kubernetes 集群中。该代理支持 Prometheus 配置、发现和指标提取功能，丰富所有高保真 Prometheus 指标和元数据并以 嵌入式指标格式（EMF）将其发布到 CloudWatch Logs。

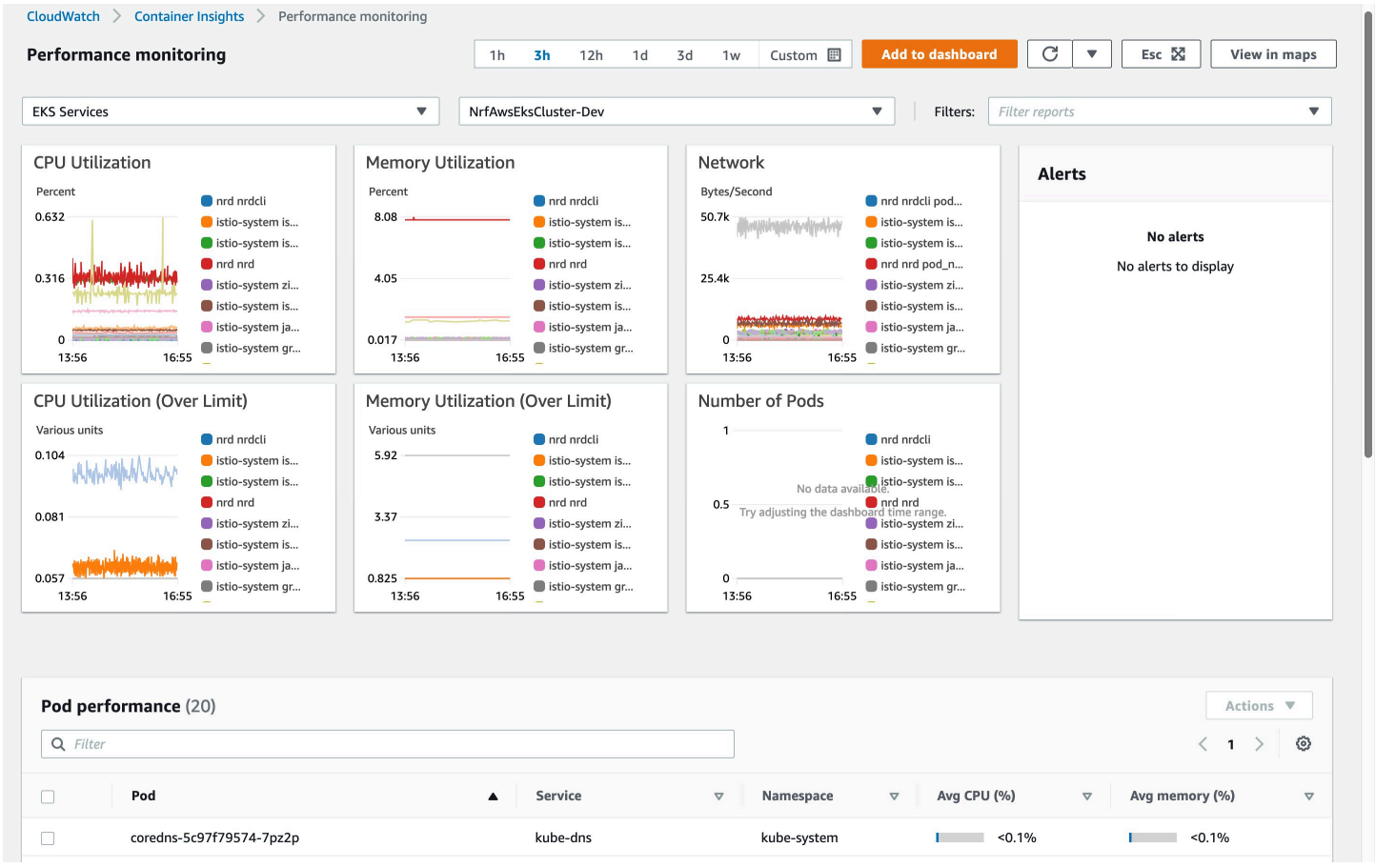
Amazon CloudWatch Container Insights 可自动发现和收集来自容器化应用程序的 Prometheus 指标。它会自动收集、筛选和创建在控制面板中直观呈现的聚合自定义 CloudWatch 指标。

每个事件都会为一组完全可配置的精选指标维度创建指标数据点，作为 CloudWatch 自定义指标。将聚合的 Prometheus 指标发布为 CloudWatch 自定义指标统计信息可以减少监控、告警和排除性能问题和故障所需的指标数量。您还可以使用 CloudWatch Logs Insights 查询语言分析高保真度 Prometheus 指标，以隔离影响容器化环境运行状况和性能的特定 pod 和标签。

AWS CloudTrail 提供这种可见性，并记录各种服务间的每次 API 调用。AWS Config 提供合规性验证功能。AWS 使用 AWS X-Ray 和 AWS CloudTrail 等各种服务，为客户提供针对应用程序、基础设施和管道的其他指标、日志、事件监控选项。

- AWS 可以在本地集成开源指标工具，例如 Prometheus、Fluentd 等。
- Prometheus 指标可以进一步导入 Amazon CloudWatch 或 OpenSearch Service 中进行进一步分析。
- AWS 使用 FluentD 作为标准机制来从各种系统收集日志。为这个项目使用和配置了同样的机制。

有关如何配置此机制的详细信息，请参阅将 FluentD 设置为守护进程集以将日志发送到 CloudWatch Logs。



Amazon CloudWatch 监控指标的示例

使用第三方和开源工具进行 CI/CD 编排

编排层使用 IaC 来部署和配置运行 5G 网络功能所需的底层基础设施。该层应设计为模块化、便携式且可重复使用。

该基础设施遵循云原生最佳实践，具有高可用性、冗余性和可扩展性。

如前面几节所示，可以使用 AWS Cloud Development Kit 部署底层基础设施。这可以使用 Hashicorp 提供的Terraform 来完成。

Terraform

Terraform 是一款开源 IaC 软件工具，可提供一致的命令行界面（CLI）工作流来管理数百种云服务。Terraform 将云 API 编码为声明式配置文件。

要使用 Terraform 进行部署，请使用与 CDK 中使用的相同原则。该代码采用模块结构，允许根据供应商的要求自定义和重复使用网络组件。

配置全都是参数化的，这样就可以根据提供商和 ISV 的建议完全定制部署。

网络功能部署分为两个阶段：

- 所需的 AWS 基础设施是通过中央存储库创建和管理的。
- 配置和代码集中存储在 GitHub 存储库中。

创建先决条件后，可以使用在前一阶段设置的应用程序管道部署网络功能。

基础设施部署

基础设施部署包括成功部署和配置网络功能的所有先决条件。

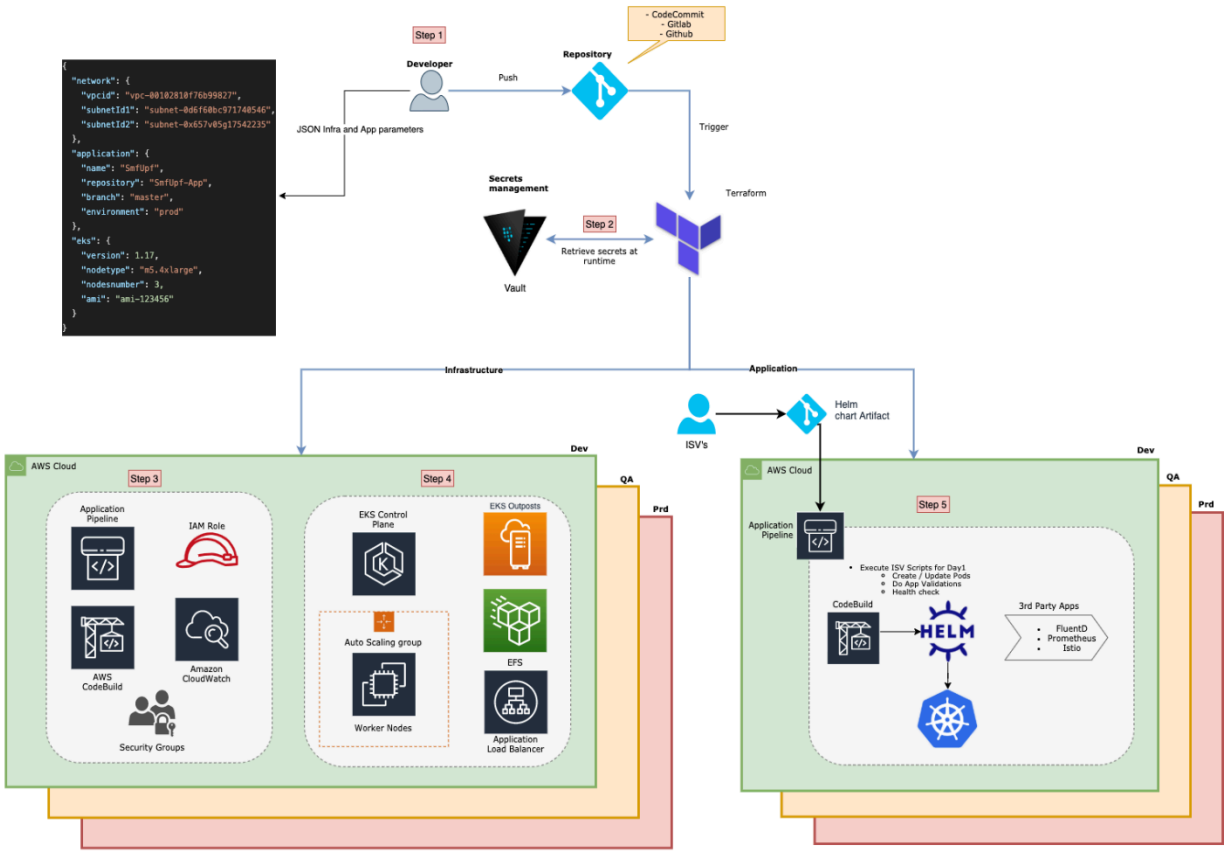
在此阶段中创建的一些组件包括：

- 联网 – VPC、公有和私有子网、路由、负载均衡器
- 计算 – Kubernetes（VMware Tanzu、Amazon EKS 或 AWS Outposts）、Amazon EC2 实例主节点和 Worker 节点、Auto Scaling 组
- 存储 – Amazon EFS、Amazon EBS、Amazon S3 存储桶
- 安全性 – IAM 角色、安全组

- 管道 – CodePipeline、CodeBuild
- 可观测性 – CloudWatch、Prometheus、FluentD

以下是由 Terraform 编排的基础设施序列，在下图中进行了说明：

1. 开发人员使用 IaC 代码填充存储在中央存储库中的 JSON 文件。该文件包含有关所需基础设施配置的信息，例如实例大小、Kubernetes 版本、网络信息和应用程序存储库详细信息。
2. 在运行时从 HashiCorp 文件库或 AWS Secrets Manager 中检索密钥。
3. 部署和配置基础设施组件（网络、计算、存储和安全）。
4. 部署了包含托管网络功能 pod 的 Worker 节点的 Amazon EKS 集群。也可以在 AWS Outposts 上部署 Amazon EKS 以支持需要靠近数据中心的工作负载。
5. 创建应用程序管道并将其配置为侦听网络功能存储库中的更改。每次将代码推送到已配置的存储库分支时，该管道都会自动触发网络功能的构建、测试和部署。
6. 收集和集中化日志和指标的可观测性工具作为服务部署在所有节点中，并提供几乎实时的数据，这些数据可以在 Grafana 或 OpenSearch 控制面板中可视化



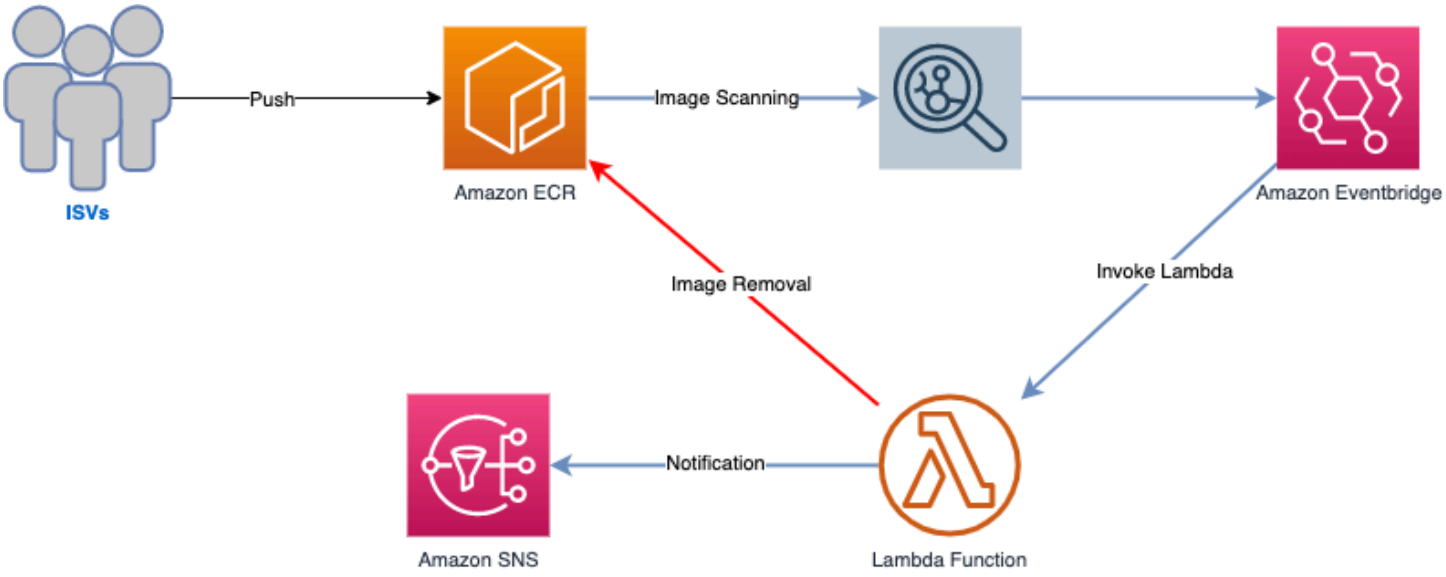
网络功能部署和配置

网络功能部署和配置

在前一阶段创建的管道使 ISV 和提供商都能够分散和优化网络功能的部署。该管道已连接并侦听应用程序存储库中的更改，该存储库在上图步骤 1 中的 JSON 文件中进行了配置。

为了审核第三方发布的镜像，我们部署并配置了一个漏洞扫描解决方案，该解决方案可帮助识别容器镜像中的软件漏洞。扫描解决方案会自动检查推送到 Amazon ECR 的所有新镜像。有关 ECR 镜像扫描的详细信息，请参阅镜像扫描。

下图演示了镜像漏洞扫描解决方案的架构。



镜像漏洞扫描解决方案的架构

可以将应用程序管道配置为由扫描结果后镜像中的更改触发，或者由存储库中的直接更改触发。例如，当创建新的 Helm 镜像时。

以下列表是创建/升级网络功能的顺序，如下图所示：

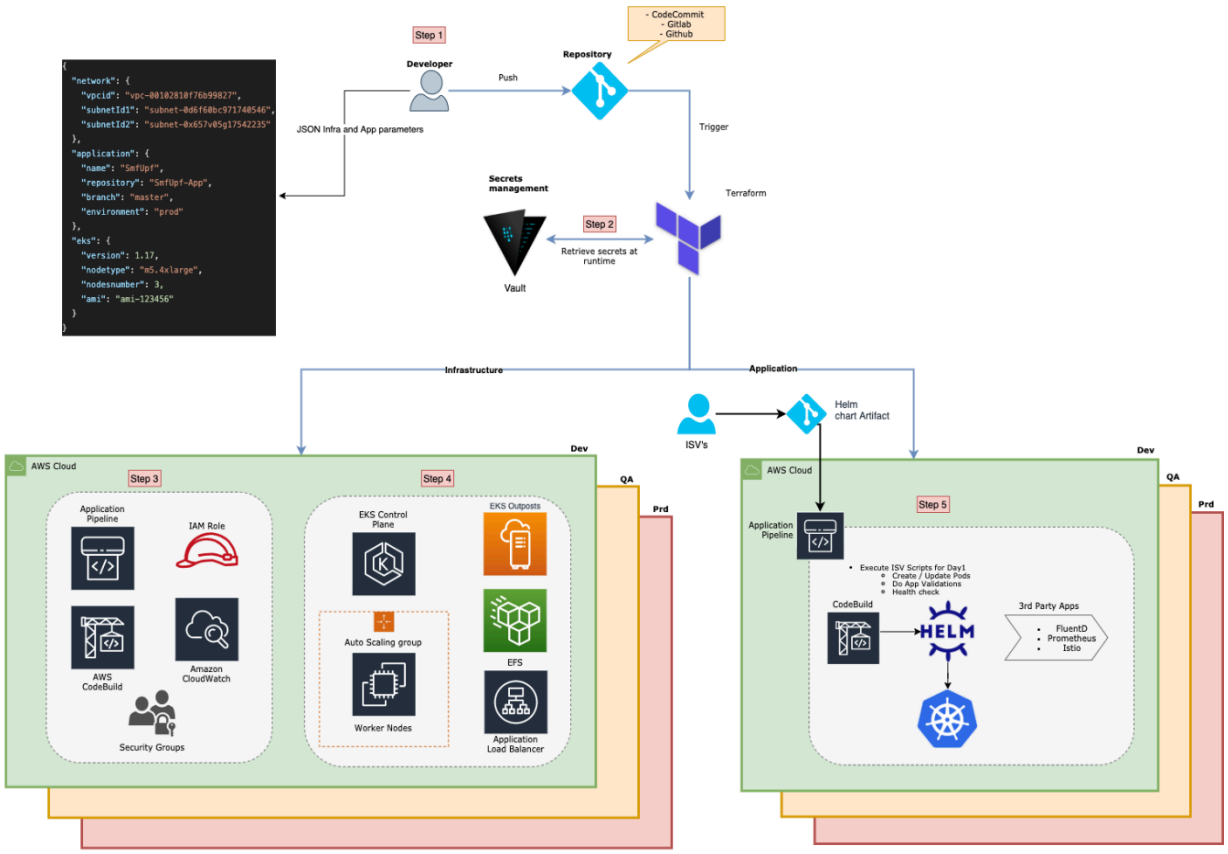
ISV 向 Amazon ECR 发布新镜像。如果映像获得批准，则会触发应用程序管道。

CodePipeline 从 Amazon ECR 中提取新镜像，然后使用 CodeBuild 将镜像部署到 Kubernetes。Helm 命令可以用来升级网络功能。

部署镜像后，将触发测试即服务（TaS）。TaS 对新部署进行验证，并集中处理有关网络功能在压力下的性能的数据和指标。

在 OpenSearch 和 Grafana 中收集和集中化日志和指标。诸如 Datadog、Istio 和 Prometheus 之类的第三方也可以配置为提供额外的可观测性。

还可以部署能够协调网络资源的 MANO，并与该解决方案集成。它使用收集的数据来执行自动化操作，例如网络切片和服务质量（QoS）自动调整。



应用程序管道

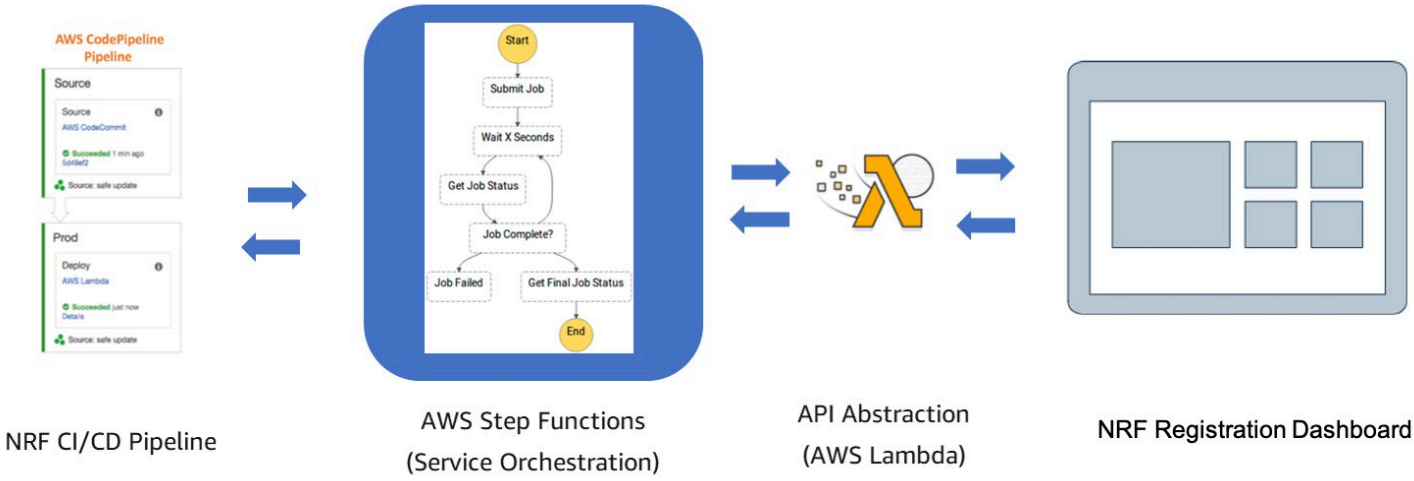
测试

专用于电信的测试自动化框架可以集成到代码管道中。代码管道与 Step Functions 集成，以协调与测试自动化框架的集成。AWS Step Functions 使用多个 Lambda 函数通过 API 调用来调用测试自动化框架（第三方工具）。Step Functions 首先获取测试 ID，运行测试，然后从测试自动化框架中获取结果。然后，Step Functions 会分析结果并将其传递给代码管道，以便批准应用程序进行生产部署。可以根据需要在代码管道中自动化或手动维护批准。对于 CSP 来说，这是将部署从测试环境推向生产环境的重要一步。集成所需的高级别 API 分为以下几类：

- 获取上下文

- 执行特定的测试用例
- 停止测试用例
- 获取结果

调用外部 REST API 的复杂性是使用 AWS Step Functions 建模的，该服务允许标准结构调用并行流、等待结果、根据条件分支以及将 REST API 与 AWS CodePipeline 集成。



测试流程

CI/CD 和编排

持续集成和持续交付是云原生架构及其如何应用于 5G 的整体自动化理念的一部分。编排是这种理念的另一个方面，它应该是动态的，并且可以对网络上发生的任何变化做出反应。预计编排和 CI/CD 将紧密结合，以确保正常的服务并最大限度地减少服务中断。预计 CI/CD 与编排之间的集成将在两个层面进行：

- 在对将补丁和升级应用到系统中进行管理和编排时，需要最大限度地减少对任何实时服务的中断。例如，编排可以动态地确定应推出更新的最佳时间。
- CI/CD 感知编排允许在部署升级期间根据采用的部署模型策略（canary、线性或一次性全部）转移流量。

通常，编排解决方案在 CI/CD 管道之上运行，以允许编排将治理阶段引入到这些管道中，并有机会参与正在进行的升级周期。

总结

CI/CD 为开发人员和应用程序团队提供了在几分钟内部署新应用程序代码的清晰、高效的途径。AWS 拥有丰富的工具，可以帮助开发人员集成、测试和部署新代码，包括 AWS CodePipeline、AWS CodeCommit、AWS CodeBuild、AWS CodeDeploy 和许多其他工具。本文档探讨了如何使用 AWS 服务创建 CI/CD 流程，以完全自动化的方式部署 5G 网络功能，包括完成新代码部署所需的不同步骤。它还介绍了如何将第三方测试自动化框架集成到 CI/CD 流程中，以及如何使用 Terraform 等第三方工具。

贡献者

本文档的贡献者包括：

- Amazon Web Services AWS 电信高级顾问 Hisham Elshaer
- Amazon Web Services AWS 电信首席顾问 Vara Prasad Talari
- Amazon Web Services AWS 电信首席顾问 Rabi Abdel
- Amazon Web Services 共享交付高级顾问 Franco Bontorin
- Amazon Web Services 共享交付顾问 Pragtideep Singh
- Amazon Web Services 全球客户云基础设施架构师 Subbarao Duggisetty
- Amazon Web Services AWS 电信高级合作伙伴解决方案架构师 Young Jung

文档修订

要获得有关此白皮书更新的通知，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
初次发布	白皮书首次发布	2021 年 3 月 8 日

延伸阅读

如需更多信息，请参阅：

- 在 AWS 上练习持续集成和持续交付（白皮书）
- AWS 上的运营商级移动分组核心网络（白皮书）
- 借 AWS 之力促进 5G 网络发展（白皮书）

首字母缩略词

- AMF – 访问和移动管理功能
- API – 应用程序编程接口
- AUSF – 身份验证服务器功能
- BSS – 业务支持系统
- CDK – Cloud Development Kit
- CI/CD – 持续集成和持续交付
- CLI – 命令行界面
- CNF – 云原生或容器化网络功能
- CSP – 通信服务提供商
- CU – RAN 中央单元
- CVE – 常见漏洞和风险
- DoS – 拒绝服务
- DR – 灾难恢复
- DU – RAN 分布式单元
- E2E – 端到端
- ECR– Elastic Container Registry
- EFS – Elastic File System
- EKS – Elastic Kubernetes Service
- EPC – Evolved Packet Core
- IaC – 基础设施即代码
- ISV – 独立软件供应商
- MANO – 管理和编排
- MEC – 多接入边缘计算
- NACL – 网络访问控制列表
- NAT – 网络地址转换
- NF – 网络功能
- NFV – 网络功能虚拟化
- NFVO – 网络功能虚拟化编排器

- NOC – 网络运营中心
- NRF – 网络存储库功能
- OSS – 运营支持系统
- PII– 个人身份信息
- QoS – 服务质量
- RAN – 无线电接入网络
- SBI– 基于服务的接口
- SMF – 会话管理功能
- SSL – 安全套接字层
- TaS– 测试即服务
- TCP – 传输控制协议
- TLS – 传输层安全性
- UDM – 统一数据管理
- UDP – 用户数据报协议
- UPF – 用户面功能
- VIM – 虚拟化基础设施管理器
- VNF – 虚拟网络功能
- VPC– Virtual Private Cloud

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 对其客户的责任和义务由 AWS 协议决定，本文档与 AWS 和客户之间签订的任何协议无关，亦不影响任何此类协议。