白皮书

思科网络服务编排器 NSO-通向自动化的桥梁

摘要

构建一个服务编排工具链还是构建一个 DevOps 环境，自动化策略通常都需要开发人员了解基础架构如何工作，同时需要基础架构所有者熟悉应用程序开发和服务创建流程。而现实世界中的经验告诉我们这种方法是有缺陷的。思科网络服务编排器（NSO）提供了一种更为可行的方法，通过充当应用程序或服务所有者与基础架构所有者之间的桥梁，让每个团队在其本地环境中进行操作，但同时仍然可以有效地进行协作。这些团队多年的运营经验使得他们已经形成了一个具有各种功能和运营能力的平台，而这个平台无论在企业领域或服务提供商领域都是自动化目标中的很有价值的一部分。

目录

自动化的重要性

强大的自动化能力通常被认为是一种竞争优势，因为它们可以帮助改善客户体验，增加收入并降低成本。市场先驱将自动化与敏捷开发，持续集成和持续部署（CI / CD）等概念相结合；并重塑其运作方式，取得了积极的成果。而成功达成这些计划的关键之一是支持自动化战略的强大先进的工具。

行业领导者都有一些共同特征：他们行动迅速，他们努力提供良好的客户体验，并且他们能够处理好价格和成本。通常情况下，结果是收入和盈利能力的增长，这使得这些组织能够让股东感到高兴并持续投资于业务。成功地开启和维持这种良性循环需要持续致力于调整人员和技能、过程与文化、以及技术，这即使对于最坚定的公司也可能是很大的挑战。虽然所有这三个要素都很重要，但本文主要关注的是最后一个要素：技术（尽管不可能避免讨论到其他两个因素）。Google，Netflix，Intuit 和 WalMart 等公司的开创性工作表明，流程自动化可以加速服务创建，改善客户体验并降低成本。自动化被认为是一种竞争优势，在过去的几年里，问题已经从“我应该自动化吗？”改为“我如何实现自动化？”

如何从自动化中获益以及为何自动化经常令人失望

无论您是移动运营商，零售商，媒体公司还是银行，您的企业都会根据众多的定期运行的核心流程进行运营。您能多好的重复这些流程在很大程度上决定了您的客户满意程度，员工的满意程度以及您能保业务持续多久。您能多快相应客户的要求？ 您的订单流程中有多少摩擦？ 您能以多快的速度推出新产品和服务？

前文中提到的市场领导者们通过三个基本概念应用自动化，这有助于提高软件，产品，服务等从概念到客户处落地的速度和质量：

敏捷开发

敏捷理念专注于通过消除流程障碍，促进客户与工程师之间的协作以及持续快速增量改进来加速交付。虽然“敏捷”最初是作为一种软件开发方法开始的，但其概念已经成功应用于从制造到营销再到服务创建的各个方面。自动化能够压缩开发和测试的时间。

持续集成/持续交付（CI / CD）

CI / CD 流水线关注快速，可靠的实施。持续集成是一种软件实践。虽然这种理念通常会应用与将错误修复程序和新功能集成到生产代码中，但它同样可以应用在路由器配置更改或防火墙规则之中。持续交付主要涉及软件和相关基础设施的部署。“基础设施作为代码”的概念是 CI 和 CD 在基础设施软件和配置管理中的应用。CI / CD 本质上依赖于自动化，因为手动过程不仅会引入错误，差异和延迟，同时还会降低可扩展性并增加成本。

DevOps

DevOps 的核心是推动文化转变，打破传统的组织结构，培养一种合作和共享所有权的文化。但具有讽刺意味的是，DevOps 成功的关键是允许团队之间更加独立的工具化和自动化。

理论与现实

自动化是神奇的，因为它完成了三件事:

1. 通过自动化来处理占据大部分时间的单调过程（例如用户申请一个 Web 服务器的 IP 地址），可以让您的团队从事像更好地了解客户和利益相关者的需求等更高价值的任务(例如了解业务应用程序和服务的重要性)。
2. 自动化流程以一致的方式执行，因此下游流程无需处理错误和可变性，客户可获得可预测性和更好的体验。
3. 自动化流程不分日夜的以机器速度按需运行。 这缩短了每个人的开发，测试和部署周期。 对于组织而言，这意味着更快的产品上市时间，更高的生产力和更低的成本。

理论是简单明了的，但现实情况可能更为复杂。
自动化依赖于软件能够对物理世界进行程序化控制 – 即从概念上的目标过渡到实际上的行动，即使虚拟资源（VM，容器等）仍然需要在物理基础设施上提供服务。例如，当服务所有者点击其屏幕上的按钮以部署其应用程序的另一个实例时，该操作最终必须转换为启动虚拟机，启动网络端口，分配 IP 地址，更新防火墙和负载平衡器，以及其它操作。

现实世界的客户需求可能既复杂又苛刻，基础设施也同样复杂和苛刻，因此成功取决于连接它们的桥梁的稳健性（见图 1）。
这是思科 NSO 发挥作用的一个角色：确保基础设施的意图与基础设施的现实相匹配。

思科 NSO 已在世界上一些最大，最复杂的多厂商生产环境中部署了近十年。 本白皮书将探讨这些经验如何转化为一组开发人员和运营团队可以用来高效，全面地实施自动化策略的功能和工具。

图 1 连接世界

建立更好的桥梁

多年帮助客户实现一级环境自动化的经验告诉我们一些支持 DevOps 目标的关键设计原则:

可扩展的复杂性

自动化工具就像数学一样。如果需要计算 2 份馅饼的食谱用量，那么小学的算术知识就足够了。但如果需要在火星上降落一枚火箭，那就需要微积分之类的知识了。同样，您需要的工具非常简单，易于上手，但功能强大，足以支持您更复杂的计划。随着您的目标变得更加复杂，您需要确保自己有足够的方法来表达它们。规模不仅仅是增加复杂性，它还涉及能够处理越来越多的服务，应用程序和设备。

灵活且适配性强的边缘层

业务变更是不可避免的，所以在北向，你的桥接层必须能够轻松容纳新的应用和工具。同样在南向，你的桥接层也必须能够适应基础设施的变化，比如新的厂商设备，虚拟化/容器化和对云的适配。

标准化

桥接层要提供抽象的视图，借以隐藏底层资源的复杂性和异质性。借助抽象层，应用层应该直接请求基础设施资源，而不必去关心底层的实现细节。同样，无论请求者是应用程序，工具还是系统，基础设施层看到也都是一致的请求接口。因为标准化和抽象，桥接层充当了的权威的中间层如实的反映了真实的请求过程。

开发者为中心

桥接层提供的双向的可编程接口的质量和可用性，直接决定了平台的功能和实现的速度。

总之，这四项原则有助于实现 DevOps 这一目标，在整个组织允许每个团队独立工作的同时，也保障了整体的凝聚性。在不影响到应用程序和服务正常运行的情况下,基础设施层可以改变和优化资源。

同样，在更快地推出新的应用和服务的同时，也不用考虑底层基础设施的变动。通过提高敏捷性，效率和生产力，降低成本。速度更快，质量可预测,提供给客户满意的产品和服务。

NSO

NSO 代表了思科近十年在智能自动化大型复杂环境的积累成果。在 NSO 根植于运营商市场同时，近年来，也注意到越来越多的大企业青睐于自动化解决方案。图 2 提供了 NSO 的组件和构成。

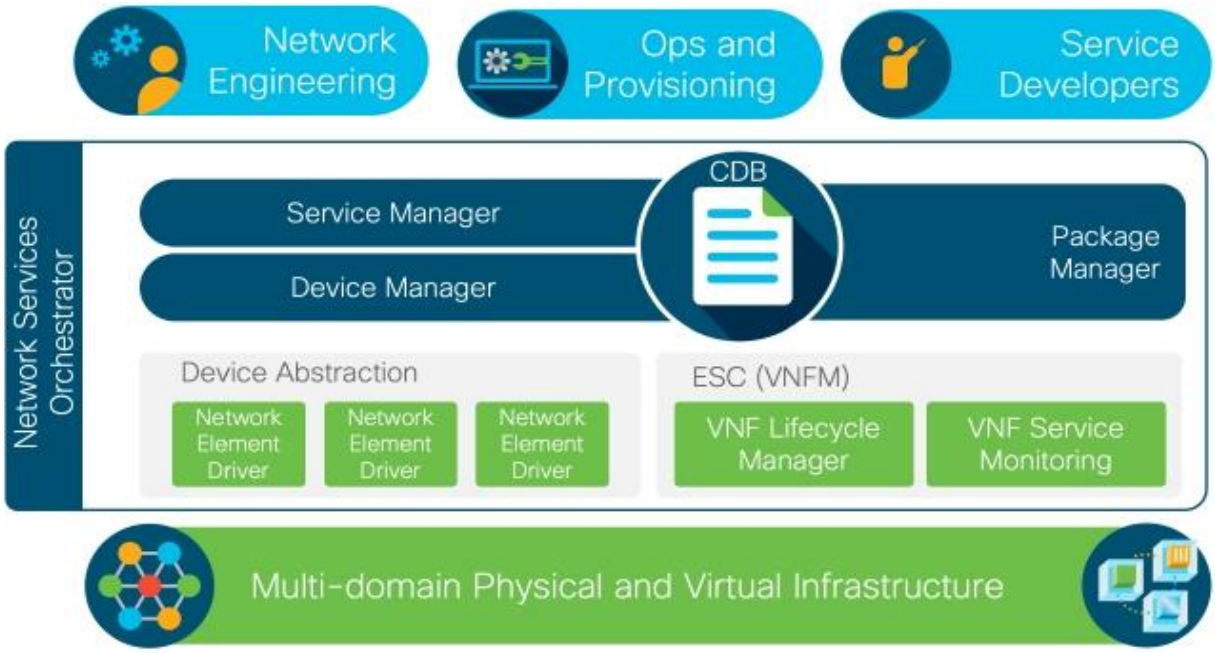
在一个高级别层面上，NSO 有三个组成部分:

1.基于模型的编程接口，从控制简单的设备开启到管理复杂的配置，NSO 提供了完整生命周期服务管理

2.快速，高可扩展，高可用的配置数据作为单一数据来源提供服务

3.使用网络设备抽象层驱动程序（NED）驱动支持思科和超过 150 个非思科的物理和虚拟设备。

图 2 - NSO 架构



总之， NSO 为所有网络设备和服务 - 物理和虚拟的，提供统一网络接口。通过通用建模语言和数据存储。 开发团队可以定义服务并将其交付运营团队，可以快速自动化激活和变更服务，上到基于高阶的基于意图的网络，下到设备层级的颗粒度配置，都可以包含一个交易内完成。
让我们来看一下 NSO 的一些关键属性，包括:

真正的模型驱动系统

针对整个网络环境，NSO 可以自动生成单一，定义良好的 API。使用标准化的 yang 建模语言，你可以建模并自动化驱动任何类型的 1 到 7 层设备，无论是物理还是虚拟的，传统物理网络还是软件定义网络（SDN）的叠加网络。你可以为任何类型的服务或策略建模。

实时配置数据库（CDB）

NSO 捕获网络中设备和服务的实时状态。在这个领域里，网络置备和运营团队获取的数据，高达 70%不准确，NSO 可以提供单一的，可扩展的，持续的准确数据。

有状态的收敛

为实现端到端的自动化，一个编排器应该能够接收服务的“意图”并将其转化为网络中的真正配置。现实中，许多网络公司目前依赖于工作流程达成这一目标，为了应对每个独特的案例，工作流程复杂性不断增长，大家不堪重负。
NSO 有不同解决方案，基于状态收敛的概念，使用通用数据模型来对服务和设备建模，通过灵活的服务数据模型，NSO 完全自动化创建，删除和运行时修改网络服务，实现了设计期服务和运行期的网络配置的映射，NSO 的状态性收敛算法推导出所需的最小网络更改，并应用于网络设备。

多域编排

自动化工具通常被绑定到某个技术领域：数据中心网络的工具，广域网的工具，光网络的工具，也许还有管理防火墙和其他 L4-7 的设备的工具。
NSO 可以跨越多个技术领域
允许您更容易可靠地对跨域服务链实现自动化。对于虚拟领域，NSO 引入 ESC 控制器作为核心平台。
ESC 提供了基础的全生命周期管理，管理对象包括虚拟资源，虚拟机（VM），或者基于容器的网络元素作为端到端服务的组成。
作为单一服务模型的一部分，NSO 可以触发，发布，配置，持续监控单个和复杂的服务链中的 VNF，还可以管理 VNF 的证书许可。

由此及彼

NSO 提供了支持 DevOps 目标的工具，在机构组织中建立专业流程，转变组织文化，从 DevOps 获得最大利益应该是一个增量，协作，迭代过程。从小处开始，实现简单的东西，从中学习，然后重复一些具挑战性的事情。
我们建议进行以下实践:

阶段 1 - 使用 NSO 作为网络编程接口

使用 NSO 为网络提供单个 API。运维获得网络置备和配置工具，利用命令行界面（CLI）从一个简单 interface 配置开始，执行一次交易提交。而不是单独接触多个设备盒子，使用不同的特定设备命令。

阶段 2 - 使用 NSO 进行服务抽象

NSO 借助设备和服务模型，全面地自动化实现激活服务和变更。你可以把端到端的服务视图视为一个整体，而不仅仅是视作个别设备的配置。

阶段 3 - 使用 NSO 进行 DevOps 基础架构自动化

当你变更人员和流程来支持敏捷开发，CI / CD，NSO 支持产品开发人员，网络工程师，配置和运营团队全程参与 - 共同努力设计，持续快速的执行新服务和处理变更。

让我们更详细地看一下这些阶段。

第 1 阶段：构建可编程的网络接口

尽管事实上，网络工程师们可能像杂耍一样完成成千上万设备的配置，但这通常依赖于手工过程 - 或者主要是手工过程，比如 CLI 脚本 - 脆弱且劳动密集繁重。

NSO 提供可更好的数据存储（CDB）和抽象/标准化网元驱动程序（NEDs），通过两种机制，提供了更强大，更富有弹性的配置管理:

交易

配置变更类似数据库交易：一次性提交所有更改，如果有部分变更失败，则整个交易回滚。

同步机制和 diff 引擎

NSO 可以比较 CDB 中和设备中的配置并突出高亮差异。NSO 也可以在任一方向进行同步。它可以带来符合 CDB 期望的设备更新，或符合设备预期的的 CDB 配置更新（即，捕获带外更新）。

这两种机制相结合以确保配置更改以可靠方式实施:

1. NSO 接受意图（网络应该是什么样子）
2. NSO 把预期的状态与当前状态进行比较，呈现“差异”。
3. NSO 更新设备配置以匹配预期的状态
4. NSO 回读设备配置以确保它匹配预期的状态

这个过程是 NSO 核心，它本身也可以让运维变得更加简单:

使用网络 CLI 和 REST 接口自动配置设备

通过单一接口和一致性语义，NSO 可以管理整个网络的设备配置，网络工程师和运维团队可以使用类似工具 - CLI 脚本或 REST 接口 - 把数百或数千个设备作为一个集合，管理配置生命周期。他们可以将网络元素分组，并应用基于模板的配置变更，一次性批量完成网络配置。

黄金配置

网络工程师可以使用模板，确保设备基于特殊配置特征进行分组，他们可以更新黄金配置模板并将其应用于组中的所有设备。

利用模板来描述所有设备的正确配置给网络工程师和运维团队带来巨大便利，在此之前他们依靠分布式的，手动流程来尝试处理庞大的异构网络规模蔓延。

配置合规性报告

一旦应用了黄金配置，网络工程师和运营团队可以使用 NSO 测验出任何偏离模板的网络元素。他们可以直接访问所有设备，立即捕获到配置中带外元素。然后工程师可以更新黄金配置，做例外处理，或重新运行模板以使设备进入合规性检查。

阶段 2：服务抽象

这个阶段使用 NSO 来帮助服务所有者设计，部署和修改服务，同时服务运营团队也可以更好地了解正在运行的内容，置备和运营团队现在可以进行高级别的自动更改，不必在服务中明确编程处理每个步骤并声明每个设备。网络运营团队现在可以更深入地了解服务。而不是检查来自网络设备的低级数据，试图推断每个服务正在做什么，以面向客户的视角，他们可以查看并跟踪网络上运行的服务。

开发人员：消除设计期和运行期之间的差距

在传统的环境中，负责设计编译（应用程序，服务，产品等）的人很少与负责运维的交谈。有文化和组织障碍，但这两个群体也往往缺乏有效的工具和语言进行合作。这种障碍拖累整个组织。没有基础设施团队的前期参与，服务模型设计中重要的需求和挑战会被忽视。看起来像开发人员的简单的“问”，运维层面可能实际上是非常复杂的操作。问题在开发阶段没有被发现，甚至在发布后也成为客户的潜在问题。

试图解决这个问题是昂贵的，“计划外工作”对所有相关团队产生了连带的负面影响。 随着组织开始围绕 DevOps 原则并追求服务抽象，他们开始通过 NSO 等工具弥补设计意图和网络实施之间的空白。 现在，服务设计人员和开发人员以人们可读的 YANG 数据模型定义了新服务。网络团队可以更快地测试和部署它们，因为服务是用他们（和他们的网络工具）已经熟悉的语言编写的。

服务交付：推向市场的时间

服务交付的准则很简单：越快越好！服务交付的延迟直接影响到收入损失和不良客户体验，庆幸的是 NSO 具备多种能力以加速服务提供:

全业务生命周期自动化

NSO 自动化提供整个端到端的服务流程。NSO 涵盖所有网络设备和资源，VNFs，应用和网络服务,无论是粗颗粒度的服务内容还是细颗粒的配置。此外 NSO 允许交付团队修改正在运行的服务以及创建和删除，以便他们可以更快速和精准的变更服务。

交易模型

如前所述，NSO 使用数据库样式的交易模型来配置新服务或更改现有服务，以确保网络和任何客户的服务永远不会处于未知状态。

激活测试

您可以构建 canary 测试。通过在新（或新更改的）服务上发送活动流量，测量面向客户的关键性能指标（KPI）并验证服务是否按预期执行。

运行时服务修改

NSO 可以对活动服务进行更改，而不是删除和重新创建服务以实现更改。 NSO 的状态收敛功能自动生成满足服务修改意图所需的最小配置。例如，客户可以登录他们的自服务门户以改变他们的服务级别（例如铜到金）或修改安全规则。

一旦请求更改，NSO 就会对细粒度的网络配置进行更改以满足客户的请求。

网络变更试运行能力

NSO 网络更改工具显示计划的更改在执行之前将如何影响网络和服务。 在进行更改之前，团队可以执行试运行并查看在执行更改时网络中将发生的最小更改集。

解耦 OSS 与网络

由于 NSO 充当桥梁，OSS 层可以屏蔽网络的复杂性，反之亦然。 运维人员可以独立管理各自领域的生命周期，使用稳定的接口允许每个层与另一个层进行通信。更新 OSS 无需担心与特定网络设备的依赖关系。 同样，网络基础架构可以更灵活，因为不再需要将新的供应商或设备明确集成到 OSS 中。

运营：拥有客户体验

运营团队通常是管理客户体验的第一道防线，他们往往缺乏理解实际情况的能力。 NSO 为运营团队提供了更好的工具来应对这一挑战。

可追溯性

NSO 不仅向运维团队展示了网络上发生的事情，而且还让他们在相关的面向客户的服务的环境中检查网络服务来实现这一点。 运维团队可以在多个设备上跟踪单个服务，并准确了解每个配置（或更改）如何影响每个客户的服务。 此功能可以更轻松地为客户解决问题，了解软件版本更新的影响，以及更有效地执行其他客户操作和支持功能。

深入了解服务配置

NSO 使运营团队能够了解如何配置服务和分配哪些资源以快速了解服务实例与网络中实际运行的内容之间的关系。 这便于运营人员了解设备的行为，配置方式和服务期望之间的不匹配。运维人员可以识别哪台设备的哪些配置服务于哪个业务（例如，谁需要 VLAN 99？）。

最后，资源故障可以与受影响的服务相关联（例如，哪些服务使用的链路是持续振荡的）。

服务规划

对于新服务正在使用响应式 FastMap（在下一节中讨论）进行部署，运维人员可以使用 NSO 的计划工具立即查看配置进展的程度，以及来自网络的实时状态和配置。此功能对于自动化和由快速实施和更耗时的操作组成的跟踪服务至关重要。

服务健康状况

NSO 允许网络管理者将编排保证纳入服务模型，以便他们可以跟踪反映真实客户体验的服务 KPI，并在适用的情况下验证服务是否符合服务级别协议（SLA）。

阶段 3：DevOps 基础架构自动化

准备开始实现利用敏捷开发方法和 CI/CD 流程将网络服务联合起来的竞争优势？ 让我们仔细看看 NSO DevOps 功能如何提供帮助。

基于模型的架构

如前所述，NSO 的一个明确特征是它完全基于模型。NSO 在其模型中捕获服务的各个方面。 YANG 服务模型成为服务的精确黑盒规范。 通过自动将服务意图映射到设备配置，NSO 显著减少了所需的手动编程量，因为数据模型中的任何更改都会自动触发整个系统的实时重新呈现：UI，API，数据存储和南向抽象。开发人员可以通过前所未有的敏捷性对服务功能进行设计时更改，而不依赖或中断基础架构团队。

状态化收敛

因为 NSO 通过如下两种机制不断地将网络收敛到所需的状态，使得动态服务的创建和修改是可能的：

FastMap

对于一个服务开发人员只需要描述“创建”操作，FastMap 自动确定任何类型的运行时服务修改所需的更新，删除和修复操作，从而为开发人员节省了为每个可想到的服务生命周期场景定义工作流的时间和精力。

响应式 FastMap

响应式 FastMap 是多域和分布式环境的理想选择，它采用非线性方法实现达到所需状态所需的必要更改。 某些更改（例如：应用新的防火墙规则）可能需要几秒钟才能实现，而其他更改（例如，启动新的 VM）可能需要几分钟。响应式 FastMap 不是等待更改完成，而是在可能的地方进行更改，并不断重新评估仍需要完成的工作。

包管理

NSO 通过安装，更新和卸载软件包的整个生命周期，为开发人员提供了一种全面，系统的软件包管理方法，以及在平台上管理应用程序的工具。该平台应用严格的版本控制规则，并允许开发人员捕获包之间的依赖关系。

北向集成 API

为了支持 DevOps 流程和服务以及有效的桥接，NSO 提供了一个稳定，灵活的软件接口。

丰富的北向 API

NSO 支持 API，从基于程序或基于 RPC 的协议（如 NETCONF / RESTCONF）到语言绑定，如 Erlang，Java，Python 和 C。NSO 还提供人机交互界面，如 Web UI 和一组的 CLI。 所有这些接口都是从开发人员创建的模型中自动呈现的。

API 媒介

服务提供商开发人员面临常见的障碍是现有的 OSS / BSS 系统，其具有对基础设施的硬编码南向接口。 使用传统的编排系统，服务提供商必须进行广泛的集成项目，以改变 OSS 系统如何将参数解析到编排器。NSO 只是适应 OSS 使用的现有 API。 开发人员可以使用该 API 创建数据模型，将其加载到 NSO 中，并将其映射到现有服务包。 该示例是用于特定 SP，但 NSO 可以为企业系统提供类似的 API 媒介。

交易安全操作

如前所述，NSO 默认使用自带交易式的模型，这意味着开发人员不必费力地自己开发和维护一个这样的模型。

幂等运算

DevOps 的核心原则，NSO 的差异和收敛操作隐含地提供了幂等性。当与运营商 OSS/BSS 系统集成时，交易性和幂等性显示出特殊的价值，意味着当这些系统调用 NSO 时，他们永远不必应对变化被部分执行的情况。 任何新服务或变更交易要么完全应用，要么根本不应用。此外，幂等性意味着基于事件 OSS / BSS 层中的系统不必编写逻辑，以避免发送太多配置。OSS 系统不再需要收集和维护状态-因而在 OSS 层完全消除了巨大的复杂性。 上层系统变得适应性更强，整合起来更简单，成本更低。

通过 NED 进行多厂商抽象

NSO 围绕 NED 的概念构建设备抽象层，允许 NSO 以编程方式操做每一个网络中的设备。 NED 计算出设备相关的有序的一系列命令，能够使网元从当前配置状态到所需的配置状态。这使开发人员不必为多厂商环境编写和维护特定的设备的代码和逻辑。 NED 可覆盖几乎任何物理或虚拟思科网络设备，以及超过 150 种非思科设备。 为了更进一步的灵活性，NSO 5 引入了 NED 创建工具允许客户为 NETCONF 设备创建自己的 NED。

开发工具和 SDK 内容

NSO 包含了综合工具，帮助开发者在每个开发周期的每一个阶段都能更快，更有效地工作。

创建

NSO 使开发人员能够在现实的开发环境中运行系统的完整生产级安装，以便他们可以立即开始编程。它具有丰富的 yang 工具，包括 yang 验证器和编译器。这简化了开发人员在 "创建" 过程中的工作。

图 3 - DevOps 周期

验证

NSO 通过其 NetSim 工具提供开发和生产测试功能。 这个网络模拟器允许开发人员快速而廉价地在模拟真实生产的环境中测试他们的代码。 NSO 还提供用于验证版本迁移的离线工具。 这些工具验证了某一服务的客户端或消费者（即一个编排器或 OSS / BSS 系统）需要更新的程度，从而使开发人员可以避免引入无意识的破坏性变化。

包

当开发人员发布新代码时，NSO 提供一种独立的和版本化的包格式。 这个意味着开发人员可以构建和打包他们的工作这样的包是他们唯一需要导入到正在运行的系统中的东西。NSO 也提供无中断的包安装和版本迁移，所以开发人员可以运行时引入新的包或更新现有的包，不影响系统的运行。

配置

NSO 可以集成到 CI / CD 流水线中，所以基础设施和基础设施配置可以与相关软件包协同无缝部署。 除了简化初始部署之外，这个功能还有助于自动伸缩，所以添加或删除应用实例可以自动包括相关的基础设施。

监控

NSO 提供了解应用程序或服务如何与基础设施进行交互的方法 - 有没有性能瓶颈或耗尽资源的服务。 CDB（配置数据库）也提供单一来源性能管理，健康监测，系统保证和类似工具，轻松收集关于基础设施状况的运营数据。

结论

NSO 使得将开发工具和方法论从软件世界带入网络世界这件事变得很容易。每个人都说相同的语言和使用相同的工具，信息可以全方向地自由流动，从提出一个好主意和到进入投产实施所需时间将从根本上缩短。

接下来做什么？

过桥需要走很多步。保持渐进且持续的方式:

- 映射技术，人员和流程之间的链接
- 投资提升员工的技能
- 采取小步骤，进行许多试点，并获得经验
- 认识到会有变革的阻力
- 拥抱不确定性
- 让每个团队按自己的进度移动
- 面向客户（内部或外部），以客户为中心
- 从小而有用的东西开始：盈得胜利，获得自信，重复一些更具挑战性的事情

通过打破分隔服务设计师，应用程序开发人员和基础设施运营团队的墙壁，NSO 帮助组织加速对 DevOps 原则和文化的采用，为组织及其客户都带来处。

更多信息
cisco.com/go/nso

NSO on DevNet
developer.cisco.com/nso

NSO on GitHub
github.com/NSO-developer