高效率数据中心网络人员的 7 大习惯

PACKET PUSHERS 白皮书

目录

简介

想要成为数据中心网络工程师，就要学会适应复杂性和不确定性。一个位置的变化可能会对其他位置产生意想不到的影响：性能可能会受到影响，工作负载可能会暴露，网络也有可能会崩溃。

想要在网络需求日新月异的动态环境中顺利运作，网络工程师需要养成良好的习惯。经验丰富的网络工程师往往谨慎仔细、思虑周全、一丝不苟。通常，在进行更改之前，他们会通过 SNMP 陷阱、系统日志、设备配置、数据包捕获和流式遥测来查找有关网络的信息。当出现问题时，他们会有一整套的流程来发现、诊断和解决问题。

虽然这些常规做法对工程师很有帮助，但本文接下来提出的 7 个具体习惯，可以帮助网络工程师更好地应对数据中心管理和运维中的不确定性。

这些习惯本身可以作为一套最佳实践，而且符合 Juniper Apstra 的要求，后者是一款建立在基于意图的网络 (IBN) 概念上的软件平台。IBN 可以为数据中心网络提供可靠的自动化和编排功能。Apstra 软件可以在多供应商环境中工作，并能自动执行、编排及验证变更，从而确保结果符合业务意图。

这 7 大习惯包括：

1. 专为业务成果而设计
2. 通过自动化实现可靠性和可重复性
3. 触类旁通，学以致用
4. 选择合适的设备
5. 时时验证
6. 主动出击，而非坐以待毙
7. 记录工作

本白皮书探讨了高效率数据中心网络工作人员的 7 大习惯，并阐明了 Apstra 如何与这些习惯相辅相成。

7 大习惯

1. 专为业务成果而设计

数据中心网络的存在是为了支持有助业务发展的应用和服务。然而，传统的数据中心设计往往从供应商选择入手，而非期望的业务成果。供应商选择通常会受业务目标以外的因素影响。时常与销售代表共进午餐的高管想要的产品，可能不同于在供应商认证上投入时间和资金的工程人员。

产品之间可能存在微小但颇具影响的差异。一家供应商的代码会受累于组织不想要或不需要的东西，但仍须维护和更新。另一家供应商的硬件规格很理想，但其网络操作系统可能有问题。软件如何实施密钥协议，可能需要工程师付出额外努力才能实现。

结果就是，企业必须调整实际业务应用的需求，以适应产品的特殊要求和限制。这意味着操作越来越复杂，出现问题的风险也越来越大。这会拖慢网络支持新应用和服务的速度，使网络成为一种瓶颈。

较好的习惯是从业务成果着手，然后围绕这些成果设计网络。Apstra 专注于意图，能够优先考虑成果，然后在设备层面上运行，进而以声明的方式实现这些成果。这款平台能够持续监控和验证网络状态，从而确保实现这些成果。

由于 Apstra 支持各种网络硬件和软件，组织可以选择最适合业务的产品，而不是让设计受制于某个供应商的特殊要求。

2. 通过自动化实现可靠性和可重复性

几十年来，网络行业一直在谈论自动化，但大多数企业并未放弃自行生成的脚本，工程师使用起来就像多功能袖珍刀一样：用于一些小任务的简便工具。

事实上，大多数企业数据中心都过于脆弱，无法支持跨多种设备和服务编排的广泛而可靠的自动化功能。其中一个原因是，自动化流程可能会引发一系列意外事件，从而导致整个网络崩溃。此外在设置之后，配置偏离基准的情况并不少见。编写脚本和指南的工程师必须不断调整代码来适应网络设备上运行的不同软件版本的细微差别。

此外，网络工程师还缺少支持广泛自动化的关键要素。其中包括对网络状态的深入了解、预期设备配置的可靠真实信息源、投入生产之前测试变更的能力，以及验证流程结果的机制。

Apstra 能够实现可靠且可重复的自动化，因为它具有防护措施，可以防止意外问题，并确保变更与操作人员的意图保持一致。例如，如果有工程师进行了与意图不一致的配置变更，无论是命令中的拼写错误，还是违反现有策略的配置变更，Apstra 都会通知该工程师并阻止变更。

Apstra 之所以能做到这一点，是因为它将整个网络状态保存在其图形数据库中（如下所述），从而能在投入生产之前识别潜在问题。通过这种方式，Apstra 简化了变更管理，并通过可预测且经过验证的方式进行常见的变更，防止出现意外后果。

此外，如果变更出现问题，Apstra 还能够将设备回滚到已知的良好状态。这样就实现了兼具可靠性与可重复性的自动化流程。

3. 触类旁通，学以致用

偶尔使用业务应用程序的人都知道，每次使用它来完成任务时，都要周而复始地重新熟悉其运行方式，非常浪费时间。

管理、监控和自动化工具亦是如此。如果它们不是常规工作流程的一部分，就必须在界面上进行摸索，继而耽误任务进度。熟悉工具是一个好习惯，这种习惯可以让您充分利用工具。

Apstra 简单直观，只需几天时间就能熟练上手。相比之下，其他解决方案在架构和操作上的变化巨大，而且需要几个月的时间来深入学习，这也凸显了 Apstra 的优势。而且，由于 Apstra 提供多种功能，包括监控、配置和分析，网络工程师可以快速熟悉软件，然后经常使用，进而熟练运用有用的工具。

同时，由于 Apstra 可以在多供应商环境中运行，它会将每个 NOS 和供应商协议实施的个别特殊问题抽象出来。这意味着网络工程师无需非常熟悉每个 NOS 的细微差别，即可轻松完成自己的工作。

4. 选择合适的设备

较为明智的做法是为手头的工作选择合适的设备，而不是让操作适应特定设备的运作方式。

Apstra 的多供应商战略使网络团队能够灵活选择合适的硬件和软件来支持公司的目标。这也意味着企业可以发挥采购杠杆作用，避免受到供应商的束缚。他们还可以从不同的供应商处采购设备，鉴于当前的供应链限制，这种能力非常重要。

5. 时时验证

科技社交媒体上充斥着各种草率的 CLI 命令、配置错误和拼写错误方面的奇闻轶事，这些命令和错误导致了各种刺激而富有挑战性的后果。事实上，人都会犯错，这就是为什么成功的工程师会在生产过程中检查自己的工作。

这些检查可以采用各种形式进行。某些网络操作系统会在输入命令字符串时检查其中的每个词，并在出现拼写错误时提醒工程师。工程师可能会让同事对配置进行健全性检查，或将脚本上传到存储库供他人审查。部分组织则会依赖基于 ITIL 的变更管理流程。

Apstra 通过不断验证设备变更来确保这些变更符合意图，从而强制执行这一良好习惯。例如，假设您创建了新的 VLAN，但意外将其添加到包含现有无标记 VLAN 的接口上，此时 Apstra 不仅会反馈错误，还会向您展示错误原因。

持续验证既可减少人为错误，又能确保新的变更达到预期目标；也就是说，新的变更符合业务定义的可访问性、性能、安全性和合规性参数。

6. 主动出击，而非坐以待毙

果断处理小问题以防止事态严重，是一种非常好的习惯。例如，如果交换机端口时好时坏，那么最好在端口彻底发生故障之前找到根本原因并加以解决。

如果坐以待毙，那么在试图寻找问题根源时，就会被各种警报和日志（可能还有源源不断的短信）所淹没。如果只是电缆松动，那您还算幸运。但如果是网卡或光学模块损坏，就只能寄希望于手头上有备件。

Apstra 可通过其分析功能支持主动运维。它会监控物理设备的状态，并且可以针对需要立即处理的异常情况发出警告，以免出现更严重的问题。它还会监控带宽问题、利用率和总容量，并提前向工程师发出信号，帮助他们平稳、高效地满足不断增长的需求，以免陷入火烧眉毛的窘境。

7. 记录工作

就像多吃蔬菜和定期锻炼一样，每个人都知道记录工作是一种好习惯，无论是更新网络图表还是对配置变更进行注释。记录文档中会阐明所执行的操作及原因，以及所发生的情况。这些记录可以在日后用于新的变更、故障排除、审计等。

记录文档很重要，因为工程师随时可能转岗或另谋他职。他们离开时会带走大量运营知识和经验。而文档可以记录这些知识，让组织得以保留经验并与操作人员分享。

文档记录并非一帆风顺，记录工作既繁琐又耗时。在处理棘手的新部署或应对危机时，记录往往是工程师最容易忽略的事情。在建议的所有习惯中，定期、可靠的记录可能是最难保持的。

Apstra 可以助您养成这一习惯，因为其本质就是自动记录。它会不断收集各个设备的遥测数据，维护整体网络状态，同时还会存储变更和版本的历史记录。Apstra 就像一台时间机器，让您能够回顾过去，查看所做的变更、更新及其结果。

正如上文所述，Apstra 可以在必要时将网络回滚到一个已知的良好状态。

Apstra 和 IBN 简介

Apstra 是围绕基于意图的网络 (IBN) 概念打造而成的一款平台。Apstra 可从业务层面的意图或预期结果着手，然后将这些意图转化为满足预期结果所需的设备层配置。

业务意图包含常规结果，例如确保一组应用的特定服务级别；同时也包含特定结果，例如部署交换矩阵、打开 VLAN 并连接正确的端口，或实施访问策略。

Apstra 可以持续监控网络的整体状态及个别网络设备，从而确保网络始终符合组织的意图。如果发生违背该意图的事件，如数据包丢失、拥塞或接口问题，Apstra 可以提醒工程师并提供问题的相关详情。在某些情况下，Apstra 可以提供后续步骤建议，或者根据需要自动修复问题。

如果工程师试图进行与先前意图相冲突的配置变更，Apstra 将通过提供变更前分析来提醒工程师，以免出现意外后果。随着新的网络服务添加到数据中心，Apstra 可确保这些服务所需的网络配置不会与现有意图相冲突。

核心原则

Apstra 基于参考设计、多供应商支持、持续验证和集成等关键原则打造而成，可以有效支持 IBN。

参考设计：数据中心网络通常在没有任何长期战略的情况下组合在一起，这是导致数据中心复杂性的一个重要原因。专用的自定义实施方案可解决当前业务需求。但随着时间的推移，这种方案将形成由复杂配置组成的网络。这些配置不受自动化的影响，需要手动维护，而且十分耗时。对于这样的网络，除了从一位工程师传给下一位工程师的经验知识，通常鲜有文档记录。积压的技术性难题可能会迫使工程师仅靠权宜之计来维持业务运行。

Apstra 通过使用有限数量的参考设计开辟了新的方向。参考设计介绍了物理基础架构的模板，包括交换硬件和布线。典型的参考设计是叶脊式或 Clos 网络。

虽然一些工程师可能有受到参考设计或模板限制的感受，但这种方法是以行业最佳实践为指导，大大减少了困扰传统数据中心网络的一次性方案、权宜之计以及碎片化实施方案所带来的问题。企业必须遵循这种严苛的原则，以确保单一真实信息源和对变更的控制。

Apstra 使用参考蓝图来指导整个“设计-部署-运营”生命周期的自动化，从而实现配置即代码，实施版本控制并提供依赖关系验证。这种方法为企业提供了来自云提供商的多种经验证的自动化技术，而无需大量内部专家。

多供应商支持：Apstra 支持越来越多的硬件供应商，包括 Cisco、Artista、Dell、瞻博网络，以及众多白盒硬件制造商。Apstra 还支持各种网络操作系统，从商业 NOS 到 SONiC 等开源选项一应俱全。

这种多供应商支持可确保各组织通过合适的设备来满足其业务和技术需求，并确保这些设备与其工程师的知识和技能相匹配。对于想要在机架或 Pod 中混合搭配供应商和 NOS，进而获得有竞争力的价格、考虑供应链的多样性或满足运营要求的客户，Apstra 还能为其提供更多选择和更高的灵活性。

闭环验证：Apstra 通过基于其单一真实信息源的闭环验证来确保符合意图。闭环验证可以确保变更或更新切实发生并与意图保持一致。相比之下，简单的自动化脚本可以自动执行一组命令，但该脚本只是运行，然后停止；对于是否进行了变更或变更是否产生了预期结果，自动化脚本无法提供上下文。

闭环验证对于可靠的自动化来说至关重要，因为它可以检查自动化流程是否真正产生了符合业务要求的结果，以及网络是否持续在所需状态下运行。这种验证提供了可靠性，使网络团队能够对自动化系统产生信任。

集成：Apstra 与 VMware NSX-T 和 VMware vSphere 紧密集成，可以简化服务器和网络团队之间的操作。Apstra 不仅可以实现跨物理底层和虚拟叠加层的端到端可见性，还可以确定叠加层中的变更何时需要在底层中进行调整，例如交换矩阵的 MTU 配置大小与叠加层的要求不匹配。Apstra 还可以检测和修复异常和配置错误。

如此一来，底层便能对不断变化的工作负载连接和容量做出响应并进行优化。此外，当出现问题时，Apstra 软件可以快速确定根本原因是在底层，还是在虚拟叠加层。Apstra 通过加快解决问题的速度，帮助组织大大减少平均修复时间和运营成本。

此外，开放式 API 还支持与常见工作流工具（如 ServiceNow、聊天机器人和 Slack）集成。

Apstra 组件

Apstra 产品由三个软件元素组成：设备代理、数据存储和图形数据库。这些元素协同合作，可以提供基于意图的网络。最后我们简要介绍一下每个元素。

代理：Apstra 在物理和虚拟的网络设备上使用设备代理来配置设备，并将遥测数据发送到 Apstra 的数据存储。对于无法运行第三方代理的网络设备，可以在 Apstra 服务器中作为 Linux 容器运行异机代理。该异机代理可以通过 SSH 或 API 从网络设备获得状态信息。

数据存储：数据存储在服务器上运行，可收集代理遥测数据、网络设计详细信息、网络异常和其他数据。用户意图也保留在数据存储中，图形数据库亦是如此。

图形数据库：Apstra 在图形数据库中展示数据中心网络中的每个元素或对象，以及所有网络配置。这就是 Apstra 对网络状态的“理解”。图形数据库作为网络的虚拟模型，是不同团队的单一真实信息源。该模型会不断收到网络和设备遥测数据，因此可以实时更新。系统会将用户意图与此模型进行比较，从而确保网络配置和设备配置提供合适的结果。