使用可编程安装程序

目录

简介

本文档介绍可编程安装程序，它是用于部署NSO和CNC等思科软件堆栈的规范驱动自动化平台。

可编程安装程序

字段	价值
产品	可编程安装程序
文档类型	技术白皮书 — 架构、实施和成果
主要受众	解决方案架构师、平台工程师、DevOps/SRE、交付主管
二级受众	工程管理、安全审核员、项目经理

摘要

可编程安装程序是一个规范驱动的自动化平台，用于在企业Linux（RHEL系列）和相关基础设施（VMware vCenter、OpenShift、KVM、气隙式Kubernetes）上部署和操作思科软件堆栈(包括网络服务协调器(NSO)、Crosswork Network Controller(CNC)、Crosswork Data Gateway(CDG)和Business Process Automation(BPA))。 该系统将声明性意图（YAML规范和可选的指导性意图生成）与执行（Ansible角色和手册）分开，并使用Python控制平面封装对象、在长时间运行安装之前验证捆绑包、准备加密密钥以及协调验证网关。

本白皮书介绍架构层、主数据流、实施模式（包括混合数据驱动的工件验证模型）、部署模式（本地、容器化、在线、气隙）以及验证和日志记录框架。对于交付和平台组织，该平台旨在尽早减少手动工作、表面错误配置和缺少二进制文件，并在保持特定于环境的参数化的同时，跨产品和拓扑实现自动化标准化。

关键词：基础设施自动化、声明性部署、Ansible、YAML规范、气隙封装、工件验证、Crosswork、NSO、CNC、CDG、BPA、验证策略、DevOps。

如何阅读此文档

角色	建议的焦点
决策者/领导者	摘要；§1价值主张；§8收益和风险状况；§10结论
解决方案架构师	§3-§6（架构、规格模型、实施、部署模式）
DevOps/SRE/交付工程师	§5-§7;附录 B;配套内部白皮书附件
安全审核员	§7安全与合规状况；§3.2中的信任边界

1.价值主张

1.1交付问题

企业安装多层网络自动化和协调产品历来是人性化的工作：长的Runbook、许多手动步骤、站点之间的版本偏差，以及数小时进入流程的故障（缺少NED、错误的OVA路径、不完整的气隙映像集）。 这种模式会增加成本，延长更改窗口，并加大审核的难度。

1.2可编程安装方法

可编程安装程序将安装视为按规范参数化的程序：拓扑、版本、平台选择（vCenter与OpenShift与vanilla VM）、文件路径和授权。自动化在可能的情况下具有等效，可在所有客户间重复执行，并会通过检查提前加载，因此在群集或产品安装之前，“未就绪”是一个快速、明确的结果。

1.3在这种情况下，可编程意味着什么

- 声明：操作员描述必须部署的内容；实战手册实现方式。
- 数据驱动验证：当模式稳定时，每个版本期望的对象从表和规则中导出，而不是临时脚本。
- 策略控制质量：部署前和部署后验证在分层策略下运行，具有结构化报告和可选的故障单集成。
- 多模式操作：首次用户交互式菜单；CLI和冻结的二进制文件，用于CI/CD样式重复；用于标准化运行时映像的基于Docker的流。

2.系统情景

2.1行为者和环境

参与者/系统	角色
交付工程师	编写或生成规格、运行打包、存储库准备、验证、协调器和Ansible
安装程序主机	具有Python的Linux控制节点（本地或容器）、Ansible配置、用于工件的磁盘
目标基础设施	vCenter、OpenShift/KubeVirt或按规范的Vanilla VM

参与者/系统	角色
对象源	内部镜像、授权布局、软件分发 — 特定于环境
下游系统	监控、变更管理、可选JIRA工作流程

2.2信任边界

1. 说明明示意图；它们可以参照路径和非加密参数。机密必须流经Ansible Vault工作流程，而不是可避免的纯文本规范字段。
2. 工件存储必须受到完整性保护；验证重点在于在线状态和命名与规范的一致性 — 在策略需要时，通过组织控制（校验和、带签名的捆绑包）进行扩展。
3. 安装程序到目标SSH是高权限路径；安装程序主机受到危害影响极大。强化和访问控制是运营必备条件。

3.架构原则

1. 声明性优先：YAML中的用户意图；自动化会始终如一地解释它。
2. 规划和执行分离：Python规划、验证和协调门户；Ansible执行基础设施和产品步骤。
3. 可组合自动化：站点级攻略导入重点攻略（预安装、Kubernetes路径、产品安装）。
4. 渐进式披露：入职互动式设置；用于高级自动化的标志和脚本。
5. 相同的代码库、多个运行时间：本地AlmaLinux/RHEL路径和基于Docker的路径共享一个存储库布局。
6. 显式气隙支持：在连接的机器上包装；传输捆绑包；安装前提条件并进行部署，无需依赖公共网络。

4.逻辑架构

4.1层

层	责任
规格	拓扑、版本、平台、路径、授权
控制层面	打包、捆绑包验证、保险存储助手、验证驱动程序、协调器CLI
自动化平面	主机准备、Kubernetes生命周期、产品安装和第一天配置
工件	二进制文件、图像、图表、OVA、tarball

4.2端到端数据流

1. 包装流程：必备工具将文件下载或暂存到特定位置，可选择为离线安装生成可转移的跟踪球。
2. 验证流程：在安装之前，协调器会解析规范、解析预期对象（包括平台过滤器和授权列表）以及报告就绪/丢失。

3. 部署流程：Spec plus vault（和可选的预先验证）根据从项目衍生的资产驱动Ansible手册。

4.3支持的产品（安装程序范围）

产品/捆绑包
NSO
CNC
CDG
BPA
CNC + NSO

5.规格和意图模型

5.1用户规格

规范是描述平台（例如vCenter、OCP、VM、KVM）、主机、具有版本的应用程序、拓扑（例如NSO CFS/RFS布局）、授权（NED和附加软件包）以及OVA、qcow2映像和应用层目标的文件路径的YAML文档。

5.2通用分段

特定应用程序user_spec为CNC/CDG提供缺省值和路径回退。协调器的解析器将用户规范视为真实源，并在用户键缺失时使用通用规范条目。

5.3意图生成

意图生成器通过一组调查表、规则引擎(日期驱动逻辑)和到intent.yaml的模式支持映射。

6.实施深入探讨

6.1部署协调器(cx_deploy_orchestrator.py)

协调器是脚本化或交互式生成意图、验证捆绑包和安装协调的单个条目。其设计明显是混合的：

- ARTIFACT_DEFS：声明每个应用程序的对象类型和命名模式(NSO安装程序、NED签名软件包、可选软件包；CNC OVA/qcow2/tier tarball;CDG图像；BPA图表和气隙图像标签)。
- APP_CONFIG：将CLI — 应用值(nso、crossworksuite、bpa)映射到规范文件夹名称和默认目的文件名。
- 解析器/处理程序：使用用户规范和通用规范解析CNC/CDG路径；自定义处理程序涵盖非统一命名（例如TSDN/DLM）以及图表路径和tarball路径的BPA版本格式。

就绪:AReportaggregates发现的对象；is_readyis在规范分析后没有缺少必需文件时为true。模块通过sys.frozen路径解析支持PyInstaller冻结的二进制文件。

6.2必备条件和打包工具(setup_cxinstaller_prereqs)

此组件提供交互式菜单和CLI模式，用于项目打包和主机前提安装：在线、气隙或自动检测；多应用程序包，包括combinedCNC_NSO。它填充Ansible和verify-bundle逻辑预期的人工对象树。

6.3 Ansible自动化平面

- 组合：这说明了堆栈的多路径特性：它导入不同的Kubernetes bootstrap路径 — 反映不同产品针对不同的Kubernetes bootstrap路径。
- 角色（代表性系列）：预安装(SELinux、防火墙、SSH、注册表帮助程序)、k8s_install / rke2、deploy_nso、deploy_cnc、deploy_cdg、deploy_bpa、postinstall、卸载和证书续订帮助程序。所有权和测试最好在角色界限进行管理；嵌套任务文件夹实施子工作流（例如NSO L3 HA）。

6.4验证检查框架(validation_checks/)

→此框架提供分层策略控制(全局策略应用→阶段→单个检查)、自动发现检查、增强结构化日志报告以及可选的JIRA集成。操作员根据用于安装的相同规格运行部署前或部署后阶段，使自动化与适合企业变更规则的质量门保持一致。

典型分支机构的指示规模：按照对BPA和NSO进行30次检查的顺序(在结帐时使用make list-validation-checkson确认的计数)。

6.5 Vault Secrets Manager(scripts/vault_secrets_manager.py)

从规范中导出所需的保管库变量，根据策略提示或接受密码，并为Ansible发出加密的group_vars/all_secrets.yaml和保管库密码文件，从而减少在攻略手册中临时嵌入的密码。

7.部署模式和运行时间

模式	摘要
本地(AlmaLinux / RHEL)	设置PYTHONPATH和ANSIBLE_CONFIG;根据产品指南运行包装、保险存储、验证、协调器和手册
基于Docker的安装程序	scripts/setup_installer.sh和scripts/start_installer.sh,带有大型项目的主机装载；
气隙	包装在连接的机器上；传输包；目标提取；随 — airgap安装
macOS捆绑创建	在Mac上使用Epython3 ./setup_cxinstaller_prereqs.py准备捆绑包；根据项目文档，目标部署保持面向Linux的状态

8.安全性、验证和可观察性

8.1安全状态（设计意图）

- 秘密：首选Ansible Vault加密组变量；在适当情况下使用vault manager strict模式。
- 安装程序主机：视为高信任的控制平面；限制访问和监控。
- 工件：保护采集通道；组织流程可以通过加密验证来增强verify-bundle。
- 日志记录：应用和Ansible日志未部署/日志/

8.2验证为操作门

部署前调用示例：

```
cd /opt/cx-installer
python3 validation_checks/run_validation_checks.py -t pre -s specification/your_spec.yaml
```

使用可选标志：

- -p <policy_file> — 使用自定义验证策略(默认值为 validation_checks/validation_policies/default.yaml)
- -a <app> — 将检查限制到特定应用（小写，如cnc、nso、bpa、cdg）
- —report-file <path> — 编写独立JSON预检查报告

8.3发布规则

这是一个内部采购模型，对于更多思科应用安装，可以遵循相同的原则，对存储库进行分叉。

9.优势和成果

主题	结果
时间和辛劳	减少手动步骤；在验证捆绑包和验证阶段检测到故障，而不是在Ansible或产品安装程序中检测到故障
一致性	跨活动的共享架构、角色和项目布局减少了“雪花”差异
断开的操作	有文档记录的捆绑传输支持受管制的网络，无需运行时下载
监管	结构化验证报告和可选的JIRA挂钩支持变更记录和跟进
可扩展性	清除扩展点：ARTIFACT_DEFS、处理程序、新角色/手册、意图架构

量化指标（安装持续时间、缺陷率）特定于组织;团队必须参照可比较拓扑上的旧版操作手册。

10.可扩展性和维护

10.1添加对象类型

1. ExtendARTIFACT_DEFS(如果需要，可添加标签)incx_deploy_orchestrator.py。
2. 如果命名不能仅由模式捕获，则添加acustom句柄。
3. 自动下载时，更新setup_cxinstaller_prereqs中的打包逻辑。

10.2添加Ansible行为

更倾向于在紧密结合的角色中执行新任务；在边界明确时引入新角色。Wire playbooks viaimport_playbookor已记录的入门手册。在group_vars / vars中保留安全默认值。

10.3意图生成器演进

更新YAML架构下intent-generator/schema/和chatbot输入；确保生成的文件与APP_CONFIG预期的文件名匹配。

10.4规划图注意事项（示例）

- 更深入的SBOM或图像签名验证集成。

- 扩展了CNC/CDG方案的验证范围。
- 用于规范链接和Ansible语法检查的CI参考。

11.结论

CX可编程安装程序结合了声明性规范、用于打包和验证的Python控制平面和Ansible自动化平面，可在各种基础设施和连接模式下进行可扩展、可重复的Crosswork相关产品部署。其架构有意将意图与执行分开，在实际中应用数据驱动的对象期望，并嵌入适合企业交付的验证和保险存储工作流程。有关完整的操作附件（攻略表、故障排除矩阵、连接矩阵和扩展命令参考），请参阅配套的内部白皮书。

参考和文档地图

文档	路径
操作员指南	README.md
发布手册	RELEASE_GUIDE.md
内部架构附件	docs/CX_INSTALLER_TECHNICAL_WHITE_PAPER_INTERNAL.md
Docker（在线/空气间隙/使用情况）	SETUP_ONLINE_DOCKER.md、SETUP_AIRGAPPED_DOCKER.md、USAGE_DOCKER.md
验证框架	docs/validation_checks/README.md
保管库管理器	docs/scripts/VAULT_SECRETS_MANAGER.md
产品指南	docs/nso.md、docs/bpa.md、docs/CNC_VCENTER_DEPLOYMENT_GUIDE.md、docs/CNC_OCP_DEPLOYMENT_GUIDE.md、docs/CNC_NSO_DEPLOYMENT_GUIDE.md
意图生成器	intent-generator/README.md
聊天机器人/规则流概述	docs/HowItWorks.md

附录A — 存储库布局（摘要）

```
cx-installer/
├── ansible_playbooks/     # ansible.cfg, files/, group_vars/, playbooks/, roles/, vars/
├── apps/                  # App-specific supporting content
├── deploy/                # Python deploy helpers, logging utilities
├── docs/                  # Technical documentation
├── intent-generator/      # Chatbot, rule engine, schemas, output/
├── scripts/               # Docker setup/start, vault_secrets_manager.py, ...
```

```
├── specification/         # User specs, samples, common fragments
├── validation_checks/     # Policies, runners, reports
├── cx_deploy_orchestrator.py
├── setup_cxinstaller_prereqs*
├── requirements.txt
└── README.md
```

安装后工件焦点（典型）：ansible_playbooks/files/artifacts/、files/bin/、files/charts/、files/images/。

附录B — Orchestrator CLI（摘要）

脚本:cx_deploy_orchestrator.py

参数	描述
— 应用/ -a	nso | crossworksuite | bpa
—spec / -s	YAML规范的路径
— 步骤	generate-intent | verify-bundle | install
— 仅验证	验证捆绑包；如果尚未就绪，则退出非零
— 干跑	在支持的地方进行试运行
—list-specs	列出已知规格

环境（典型会话）：

```
export PYTHONPATH=$(pwd)
export ANSIBLE_CONFIG=$(pwd)/ansible_playbooks/ansible.cfg
```

附录C — 术语表

期限	定义
规格	YAML用户规范：平台、应用、拓扑、路径、授权
意图	从目的生成器或手写的等效项进行规范化YAML
捆绑包	用于气隙传输的打包安装程序树（通常为tarball）
协调器	cx_deploy_orchestrator.py — 验证/意图/安装协调
项目验证	文件系统检查每个规范中是否存在所需的二进制文件/映像
保险存储	Ansible Vault-encrypted variable file for secrets

期限	定义
NED	网络元件驱动程序包(NSO)
CFS/RFS	NSO集群转发器/冗余转发器拓扑概念
气隙	没有安装程序到时间访问软件包下载端点的环境

文档修订历史记录

version	日期	备注
1.0	2026-03-27	初始发布就绪技术白皮书（可编程安装程序成帧）

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。