프로그래밍 가능한 설치 프로그램 사용

목차

소개

이 문서에서는 NSO 및 CNC와 같은 Cisco 소프트웨어 스택을 구축하기 위한 사양 기반 자동화 플랫폼인 프로그램 가능 설치 프로그램에 대해 설명합니다.

프로그램 가능 설치 프로그램

필드	가치
제품	프로그램 가능 설치 프로그램
문서 유형	기술 백서 — 아키텍처, 구현 및 성과
주요 대상	솔루션 설계자, 플랫폼 엔지니어, DevOps/SRE, 딜리버리 리드
2차 대상	엔지니어링 관리, 보안 검토자, 프로그램 관리자

요약

프로그래밍 가능한 설치 프로그램은 RHEL 제품군 및 해당하는 관련 인프라(VMware vCenter, OpenShift, KVM, air-gapped Kubernetes)에 NSO(Network Services Orchestrator), CNC(Crosswork Network Controller), CDG(Crosswork Data Gateway), BPA(Business Process Automation)를 비롯한 Cisco 소프트웨어 스택을 배포 및 운영하기 위한 사양 기반 자동화 플랫폼입니다. 시스템은 아티팩트를 패키지화하고, 장기 실행 설치 전에 번들을 검증하고, 암호화된 비밀을 준비하고, 검증 게이트를 오케스트레이션하는 Python 컨트롤 플레인을 사용하여 선언적 의도(YAML 사양 및 선택적인 안내식 의도 생성)를 실행(Ansible 역할 및 플레이북)과 분리합니다.

이 백서에서는 아키텍처 레이어, 기본 데이터 흐름, 구현 패턴(하이브리드 데이터 기반 아티팩트 검증 모델 포함), 구축 모드(네이티브, 컨테이너화된, 온라인, 에어 갭(air-gapped)), 검증 및 로깅 프레임워크에 대해 설명합니다. 딜리버리 및 플랫폼 조직의 경우, 이 플랫폼은 수작업, 표면 오구성 및 누락된 바이너리를 조기에 줄이고, 제품 및 토폴로지 전반의 자동화를 표준화하는 동시에 환경별 파라미터화를 유지하는 것을 목표로 합니다.

키워드: 인프라 자동화, 선언형 배포, Ansible, YAML 사양, 공극 패키징, 아티팩트 확인, Crosswork, NSO, CNC, CDG, BPA, 검증 정책, DevOps.

이 문서를 읽는 방법

역할	주요 제안
의사 결정자/리드	요약; §1 가치 제안 §8 편익 및 위험 상태 §10 결론
솔루션 설계자	§3-§6(아키텍처, 사양 모델, 구현, 구축 패턴)
DevOps/SRE/딜리버리 엔지니어	§5의§7 부록 B; 관련 내부 백서 부록
보안 검토자	§7 보안 및 규정 준수 상태 §3.2의 신뢰 범위

---

1. 가치 제안

1.1 전달 문제

멀티티어 네트워크 자동화 및 오케스트레이션 제품의 엔터프라이즈 설치는 전통적으로 하이터치 방식이었습니다. 긴 Runbook, 많은 수동 단계, 사이트 간 버전 기울이기, 프로세스에 몇 시간이 걸리는 장애(NED 누락, 잘못된 OVA 경로, 불완전한 Air-Gap 이미지 세트). 이러한 패턴은 비용을 증가시키고, 변경 기간을 늘리며, 감사를 더 어렵게 만듭니다.

1.2 프로그래밍 가능한 설치 프로그램 방식

프로그램 가능 설치 프로그램은 설치를 세부 항목에 의해 매개 변수가 지정된 프로그램으로 취급합니다. 토폴로지, 버전, 플랫폼 선택(vCenter 대 OpenShift 대 바닐라 VM), 파일 경로, 자격 자동화는 가능한 경우 동질적이며, 고객 간에 반복 가능하고, 클러스터 또는 제품 설치 전에 "준비 안 됨"이 빠르고 명시적으로 나타나도록 검사와 함께 미리 로드됩니다.

1.3 프로그래밍 가능성의 의미

- 선언형: 서비스 사업자는 배포해야 할 사항을 설명합니다. 플레이북은 어떻게 구현하는지 보여줍니다.
- 데이터 기반 확인: 예상 아티팩트는 패턴이 안정적인 릴리스당 임시 스크립트가 아닌 테이블과 규칙에서 파생됩니다.
- 정책 제어 품질: 구조적 보고서와 선택적인 티켓팅 통합으로 계층적 정책에 따라 구축 전/후 검증이 실행됩니다.
- 다중 모드 운영: 최초 사용자를 위한 대화형 메뉴 CI/CD 스타일 반복을 위한 CLI 및 고정 이진 파일 표준화된 런타임 이미지를 위한 Docker 기반 흐름.

---

2. 시스템 컨텍스트

2.1 작업자 및 환경

액터/시스템	역할
배달 엔지니어	작성자 또는 사양 생성, 패키지 실행, 자격 증명 모음 준비, 검증, 오케스트레이터 및 Ansible
설치 프로그램 호스트	Python, Ansible 컨피그레이션, 아티팩트용 디스크가 있는 Linux 제어 노드(네이티브 또는 컨테이너)
대상 인프라	vCenter, OpenShift/KubeVirt 또는 사양에 따른 바닐라 VM
아티팩트 소스	내부 미러, 엔타이틀먼트 레이아웃, 소프트웨어 배포—환경별
다운스트림 시스템	모니터링, 변경 관리, 선택적 JIRA 워크플로

2.2 신뢰 경계

1. 사양 명시적 의도; 경로 및 비보안 매개변수를 참조할 수 있습니다. 암호는 피할 수 있는 일반 텍스트 사양 필드가 아닌 Ansible Vault 워크플로를 통해 이동해야 합니다.
2. 아티팩트 스토리지는 무결성을 보호해야 합니다. 확인은 세부 항목에 대한 프레즌스 및 이름 지정에 중점을 둡니다. 정책이 필요한 조직 제어(체크섬, 서명된 번들)로 확장합니다.
3. 설치 관리자-대상 SSH는 높은 권한 경로입니다. 설치 프로그램 호스트의 보안 침해가 큰 영향을 미칩니다. 강화 및 액세스 제어는 운영상의 전제 조건입니다.

---

3. 건축원칙

1. 선언적 우선: YAML의 사용자 의도; 자동화는 이를 일관되게 해석합니다.
2. 계획 및 실행의 분리: 파이썬이 게이트를 계획, 확인, 조직합니다. Ansible은 인프라 및 제품 단계를 실행합니다.
3. 구성 가능한 자동화: 사이트 수준 플레이북은 포커스가 있는 플레이북을 가져옵니다(사전 설치, Kubernetes 트랙, 제품 설치).
4. 점진적 공개: 온보딩을 위한 대화형 설정 고급 자동화를 위한 플래그 및 스크립트
5. 코드베이스가 동일하며 런타임이 여러 개 있는 경우: 네이티브 AlmaLinux/RHEL 경로와 Docker 기반 경로가 하나의 저장소 레이아웃을 공유합니다.
6. 명시적 Air-Gap 지원: 연결된 시스템의 패키지; 번들 전송; 필수 구성 요소를 설치하고 공용 네트워크에 대한 런타임 종속성을 사용하지 않고 배포합니다.

---

4. 논리적 아키텍처

4.1 레이어

레이어	책임
사양	토폴로지, 버전, 플랫폼, 경로, 자격
컨트롤 플레인	패키지 구성, 번들 확인, 볼팅 도우미, 검증 드라이버, orchestrator CLI
자동화 평면	호스트 준비, Kubernetes 라이프사이클, 제품 설치 및 즉각적인 구성
아티팩트	바이너리, 이미지, 차트, OVA, 타볼

4.2 엔드 투 엔드 데이터 흐름

1. 패키지 흐름: 파일을 다운로드하거나 특정 위치에 스테이징하는 데 필요한 사전 구성 도구이며, 선택적으로 오프라인 설치를 위해 전송 가능한 tarball을 생성합니다.
2. 흐름 확인: Orchestrator는 사양을 구문 분석하고, 예상 아티팩트(플랫폼 필터 및 자격 목록 포함)를 확인하며, 설치 전에 준비/누락된 보고서를 확인합니다.
3. 플로우 구축: Spec plus Vault(및 선택 사항 사전 검증)를 통해 Ansible 플레이북을 계약에서 파생된 인벤토리에 대비할 수 있습니다.

4.3 지원 제품(설치 프로그램 범위)

제품/번들
NSO
CNC
CDG
BPA
CNC + NSO

---

5. 사양 및 의도 모델

5.1 사용자 사양

사양은 플랫폼(예: vCenter, OCP, VM, KVM), 호스트, 버전이 있는 애플리케이션, 토폴로지(예: NSO CFS/RFS 레이아웃), 자격(NED 및 애드온 패키지), OVA, qcow2 이미지 및 애플리케이션 계층 tarball에 대한 파일 경로를 설명하는 YAML 문서입니다.

5.2 공통 조각

특정 애플리케이션 user_spec은 CNC/CDG에 대한 기본값 및 경로 폴백을 제공합니다.
Orchestrator의 파서는 사용자 사양을 진실의 소스로 취급하며 사용자 키가 없는 경우 일반적인 사양 엔트리를 사용합니다.

5.3 의도 생성

의도 생성기는 일련의 설문지, 규칙 엔진을 통해 요구 사항의 안내된 수집을 지원합니다 (intent.yaml에 대한 스키마 지원 매핑.

---

6. 심층 분석 구현

6.1 Deployment Orchestrator(cx_deploy_orchestrator.py)

Orchestrator는 스크립팅 또는 대화형 GENERATE-intent, verify-bundle 및 설치 조정을 위한 단일 항목입니다. 설계는 명시적으로 혼합되어 있습니다.

- ARTIFACT_DEFS: 애플리케이션별 아티팩트 유형 및 명명 패턴(NSO 설치 관리자, NED 서명 패키지, 선택적 패키지; CNC OVA/qcow2/tier tarball, CDG 이미지; BPA 차트 및 Air-Gap Image Tarballs).
- APP_CONFIG: CLI —app 값(nso, crossworksuite, bpa)을 사양 폴더 이름 및 기본 의도 파일 이름에 매핑합니다.
- 파서/핸들러: 사용자 사양 및 공통 사양을 사용하여 CNC/CDG 경로를 확인합니다. 사용자 지정 핸들러는 차트 및 tarball 경로에 대해 비균일 이름 지정(예: TSDN/DLM) 및 BPA 버전 형식을 다룹니다.

준비도:AReportaggregates discovered artificates is_readyis는 사양 구문 분석 후 필요한 파일이 누락되지 않을 때 true입니다. 이 모듈은 sys.frozen 경로 확인을 통해 PyInstaller가 고정된 바이너리를 지원합니다.

6.2 필수 구성 요소 및 포장 도구(setup_cxinstaller_prereqs)

이 구성 요소는 아티팩트 패키지 및 호스트 사전 요구 사항 설치를 위한 대화형 메뉴 및 CLI 모드를 제공합니다. 온라인, Air-Gap 또는 자동 탐지 combinedCNC_NSO를 포함하는 다중 애플리케이션 패키징. Ansible 및 verify-bundle 논리에 의해 예상되는 아티팩트 트리를 채웁니다.

6.3 Ansible Automation Plane

- 구성: 스택의 멀티 트랙 특성을 보여 줍니다. 서로 다른 Kubernetes 부트스트랩 경로를 가져옵니다. 즉, 서로 다른 제품이 서로 다른 Kubernetes 부트스트랩 경로를 대상으로 한다는 점을 반영합니다.
- 역할(대표 제품군): 사전 설치(SELinux, 방화벽, SSH, 레지스트리 도우미), k8s_install / rke2, deploy_nso, deploy_cnc, deploy_cdg, deploy_bpa, 사후 설치, 제거 및 인증서 갱신 도우미. 소

유권과 테스트는 역할 경계에서 가장 잘 관리됩니다. 중첩된 작업 폴더는 하위 워크플로(예: NSO L3 HA)를 구현합니다.

6.4 검증 확인 프레임워크(validation_checks/)

이 프레임워크는 계층적 정책 제어(글로벌 → 앱 → 단계 → 개별 검사), 자동 검사 검색, 구조화된 로그에 대한 향상된 보고, 선택적인 JIRA 통합을 제공합니다. 운영자는 설치에 사용된 것과 동일한 사양에 대해 구축 전 또는 후 단계를 실행하며, 기업 변경 원칙에 적합한 품질 게이트에 자동화를 맞춥니다.

일반적인 브랜치에서 암시적 확장: BPA와 NSO에 대한 30개 확인 순서(목록 검증 확인-체크를 체크아웃할 때 확인해야 함)

6.5 저장소 암호 관리자(scripts/vault_secrets_manager.py)

사양에서 필요한 저장소 변수를 도출하고, 정책에서 암호를 묻거나 받아들이며, 암호화된 group_vars/all_secrets.yaml과 Ansible용 저장소 암호 파일을 함께 방출하여 플레이북에 임시 비밀 포함을 줄입니다.

---

7. 구축 패턴 및 런타임

패턴	요약
네이티브 (AlmaLinux/RHEL)	PYTHONPATH 및 ANSIBLE_CONFIG 설정; 제품 설명서별로 패키징, 보관소, 검증, 오케스트레이터 및 플레이북 실행
Docker 기반 설치 관리자	대규모 아티팩트에 대한 호스트 마운트가 있는 scripts/setup_installer.sh 및 scripts/start_installer.sh
에어갭	연결된 시스템의 패키지; 전송 번들; 표적에 추출; install with—airgap
macOS 번들 생성	번들을 준비하기 위해 Mac에서 Usepython3 ./setup_cxinstaller_prereqs.py; 대상 구축은 프로젝트 문서당 Linux 지향적

---

8. 보안, 검증 및 관찰 가능성

8.1 보안 상태(설계 의도)

- 암호:Ansible Vault 암호화 그룹 변수 선호; 해당되는 경우 저장소 관리자 엄격 모드를 사용합니다.
- 설치 프로그램 호스트:높은 신뢰도의 컨트롤 플레인으로 취급; 액세스 제한 및 모니터링.

- 아티팩트:획득 채널 보호; 조직 프로세스는 암호화 확인을 통해 verify-bundle을 확장할 수 있습니다.
- 로깅:애플리케이션 및 Ansible 로그 언더구축/로그/

8.2 운영 게이트로서의 검증

배포 전 호출 예:

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

선택적 플래그:

- -p <policy_file>— 사용자 지정 검증 정책을 사용합니다(기본값은 validation_checks/validation_policies/default.yaml).
- -a <app> - 특정 앱(소문자, 예: cnc, nso, bpa, cdg)으로 확인 제한
- —report-file <path> — 독립형 JSON 사전 검사 보고서 작성

8.3 릴리스 규정

이는 내부 소싱에 대한 모델로서, 더 많은 CISCO 애플리케이션 설치를 위해 동일한 원칙을 따르고 저장소를 포킹할 수 있습니다.

---

9. 편익 및 성과

테마	결과
시간 및 노동	수동 단계 감소; Ansible 또는 제품 설치 프로그램에서 늦지 않고 verify-bundle 및 검증 단계에서 장애가 탐지됨
일관성	여러 계약의 스키마, 역할 및 객체 레이아웃을 공유하여 "눈송이" 차이를 줄입니다.
연결 해제된 작업	문서화된 번들 전송으로 런타임 다운로드 없이 규제 대상 네트워크 지원
관리	구조화된 검증 보고서 및 선택적인 JIRA 훅은 변경 기록 및 후속 조치를 지원합니다.
확장성	확장 지점 지우기: ARTIFACT_DEFS, 핸들러, 새 역할/플레이북, 의도 스키마

정량적 메트릭(설치 기간, 불량률)은 조직별로 다릅니다. 팀은 비교 가능한 토폴로지에 대한 레거시 runbook과 비교하여 기준을 설정해야 합니다.

---

10. 확장성 및 유지보수

10.1 아티팩트 유형 추가

1. ExtendARTIFACT_DEFS(및 필요한 경우 레이블) incx_deploy_orchestrator.py.
2. 패턴만으로 이름을 캡처할 수 없는 경우 사용자 지정 핸들을 추가합니다.
3. 다운로드가 자동화되면 setup_cxinstaller_prereqs에서 패키징 로직을 업데이트합니다.

10.2 실행 가능한 동작 추가

응집력 있는 역할의 새로운 업무 선호 경계가 명확할 경우 새로운 역할을 소개합니다. 철사 플레이북은 import_playbookor 문서화된 항목 플레이북을 참조하십시오. group_vars/vars에서 안전한 기본값을 유지합니다.

10.3 Intent Ggenerator 진화

YAML 스키마 언더인텐트-생성기/스키마/및 챗봇 입력을 업데이트합니다. 생성된 파일이 APP_CONFIG에 필요한 파일 이름과 일치하는지 확인합니다.

10.4 로드맵 고려 사항(그림)

- 심층적인 SBOM 또는 이미지 서명 확인 통합.
- CNC/CDG 시나리오에 대한 검증 범위 확대
- 사양 연결 및 Ansible 구문 확인을 위한 CI 참조

---

11. 결론

CX Programmable Installer는 선언적 사양, 패키지 및 확인을 위한 Python 컨트롤 플레인, 다양한 인프라 및 연결 모델 전반에 걸쳐 Crosswork 관련 제품을 확장 가능하고 반복 가능한 방식으로 구축할 수 있는 Ansible 자동화 플레인을 결합합니다. 이 아키텍처는 의도적으로 의도와 실행을 구분하고, 실용적인 데이터 기반 아티팩트 기대치를 적용하며, 엔터프라이즈 제공에 적합한 검증 및 볼팅(vaulting) 워크플로우를 포함합니다. 플레이북 테이블, 문제 해결 매트릭스, 연결 매트릭스 및 확장 명령 참조와 같은 전체 운영 부속서는 함께 제공되는 내부 백서를 참조하십시오.

참조 및 문서 맵

문서	경로
운영자 가이드	README.md
출시 플레이북	RELEASE_GUIDE.md
내부 아키텍처 부속서	docs/CX_INSTALLER_TECHNICAL_WHITE_PAPER_INTERNAL.md을 참조하십시오.
Docker(온라인/공극/사용)	SETUP_ONLINE_DOCKER.md, SETUP_AIRGAPPED_DOCKER.md, USAGE_DOCKER.md
검증 프레임워크	docs/validation_checks/README.md을 참조하십시오.
저장소 관리자	docs/scripts/VAULT_SECRETS_MANAGER.md을 참조하십시오.
제품 설명서	docs/nso.md, docs/bpa.md, docs/CNC_VCENTER_DEPLOYMENT_GUIDE.md, docs/CNC_OCP_DEPLOYMENT_GUIDE.md, docs/CNC_NSO_DEPLOYMENT_GUIDE.md
의도 생성기	intent-generator/README.md을 참조하십시오.
챗봇/규칙 흐름 개요	docs/HowItWorks.md을 참조하십시오.

부록 A — 리포지토리 레이아웃(요약)

```
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
```

설정 후 아티팩트 초점(일반):ansible_playbooks/files/artificates/, files/bin/, files/charts/, files/images/

부록 B — Orchestrator CLI(요약)

스크립트:cx_deploy_orchestrator.py

인수	설명
—앱 / -a	nso | crossworksuite | bpa
—spec / -s	YAML 사양에 대한 경로
- 단계	generate-intent | verify-bundle | install
- 검증 전용	번들 확인; 준비되지 않은 경우 0이 아닌 종료
—dry-run	지원되는 경우 리허설
—목록 사양	알려진 사양 나열

환경(일반 세션):

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

부록 C - 용어집

용어	정의
사양	YAML 사용자 사양: 플랫폼, 앱, 토폴로지, 경로, 자격
의도	의도 생성기 또는 수동 작성에서 표준화된 YAML
번들	에어 갭(air-gap) 전송을 위한 패키지 설치 프로그램 트리(종종 tarball)
오케스트레이터	cx_deploy_orchestrator.py — 조정 확인/인텐트/설치
아티팩트 확인	파일 시스템은 사양에 따라 필요한 바이너리/이미지가 존재하는지를 확인합니다.
자격 증명 모음	암호용 Enable Vault 암호화 변수 파일
네드	NSO(Network Element Driver) 패키지
CFS / RFS	NSO 클러스터 전달자/중복 전달자 토폴로지 개념
에어갭	패키지 다운로드 엔드포인트에 대한 설치 프로그램 시간 액세스 권한이 없는 환경

문서 개정 기록

버전	날짜	참고
1.0	2026-03-27	초기 출판 준비 기술 백서(프로그래밍 가능 설치 프로그램 프레이밍)

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.