Catalyst Center의 템플릿 이해

소개

이 문서에서는 Cisco Catalyst Center에 대해 설명하고 3계층 또는 축소된 코어 캠퍼스 아키텍처의 구성 템플릿에 대한 경험을 설명합니다.

배경 정보

이 문서는 Cisco Catalyst Center에 대한 기본적인 이해와 구성 템플릿에 대한 경험을 갖춘 엔터프라이즈 전문가를 위한 것입니다. 특히 Three-Tier 또는 Collapsed Core 캠퍼스 아키텍처에서 근무했거나 근무할 계획인 사용자에게 적합합니다.

주요 목표는 독자가 Cisco Catalyst Center 내의 템플릿을 사용하여 구성 및 관리 솔루션을 구현하고 자동화할 수 있도록 돕는 것입니다. 이 문서에서는 고급 통찰력, 실용적인 기법 및 실제 사례를 제시하여 LAN 인프라 기술을 개선하고 자동화와 템플릿 기반 관리를 통해 워크플로를 최적화하려는 고객을 위한 실용적인 리소스로 활용할 수 있습니다.

개요

엔터프라이즈 네트워크가 계속 진화함에 따라 확장 가능하고 일관되며 자동화된 관리가 더욱 절실해지고 있습니다. Cisco Catalyst Center는 캠퍼스 네트워크 전반의 구성, 프로비저닝 및 보증을 간소화하는 중앙 집중식의 인텐트 기반 플랫폼을 제공합니다. 이 백서에서는 네트워크 전문가들이 Cisco Catalyst Center의 CLI 템플릿 편집기 및 자동화 기능을 활용하여 네트워크 운영을 간소화하고, 구성 오류를 줄이며, 3계층 및 축소된 핵심 아키텍처 전반의 구축을 가속화하는 방법에 대해 살펴봅니다. 모듈형 Jinja2 기반 템플릿을 설계하고, 자동화를 day-0 및 day-N 워크플로에 통합하며, 코어, 디스트리뷰션 및 액세스 레이어 전반에서 운영 일관성을 달성하기 위한 모범 사례를 자세히 소개합니다. 이 문서에 설명된 전략을 채택하면 기존의 수동 네트워크 관리를 Cisco의 의도 기반 네트워킹 비전에 부합하는 민첩하고 표준화되고 자동화에 기초한 모델로 전환할 수 있습니다.

캠퍼스 네트워크의 과제

캠퍼스 네트워크가 현대 조직의 요구 사항을 충족하기 위해 발전함에 따라 다음과 같은 몇 가지 주요 과제에 직면하게 됩니다.

2a. 네트워크 관리의 복잡성

많은 네트워크 기능이 여전히 수동으로 관리되므로 작업자 오류의 위험이 증가합니다. 이로 인해 유지 보수 작업이 늘어날 뿐만 아니라 IT 리소스의 부담이 가중됩니다. 특히 예산이 정적이거나 한정되어 있습니다.

20억 구축 및 자동화 과제

유/무선 네트워크 모두에 대해 새로운 디바이스를 온보딩하는 것은 많은 시간이 소요되고 복잡하기 때문에 구축이 지연되고 관리 오버헤드가 증가합니다.

2c. 소프트웨어 이미지 관리

네트워크 전반에 걸쳐 일관된 "골든 이미지"를 유지하는 것은 쉽지 않습니다. 많은 네트워크에서 유무선 장치를 위한 여러 운영 체제가 사용되면서 비효율성과 관리 문제가 발생하고 있습니다.

2d. 일관성 없는 네트워크 컨피그레이션

네트워크 구성이 달라지면 컴플라이언스 문제가 발생하고 운영상의 비효율성이 초래되어 안정적이고 안전한 네트워크를 유지하는 것이 더 어려워질 수 있습니다.
2e. 사용자 기대치 상승

사용자는 위치나 장치에 상관없이 중단 없는 연결과 원활한 애플리케이션 환경을 원합니다. 이러한 기대치를 충족하려면 네트워크는 복원력이 뛰어나고 지능적이며 실시간 변화에 적응할 수 있어야 합니다.

이러한 과제 외에도, 최신 LAN 인프라는 다양한 복잡성에 직면해 있습니다.

Cisco Catalyst Center로 캠퍼스 네트워크 간소화

Cisco Catalyst Center는 본사, 지사, 유무선 연결 및 IT/OT 환경을 지원하는 캠퍼스 네트워크를 위한 중앙 집중식 네트워크 관리 솔루션입니다. 물리적 어플라이언스, VMware ESXi 서버 또는 AWS 클라우드를 비롯한 유연한 구축 옵션을 제공합니다. 포괄적인 기능을 갖춘 Catalyst Center는 운영을 간소화하고 성능을 향상하며 보안을 강화합니다.

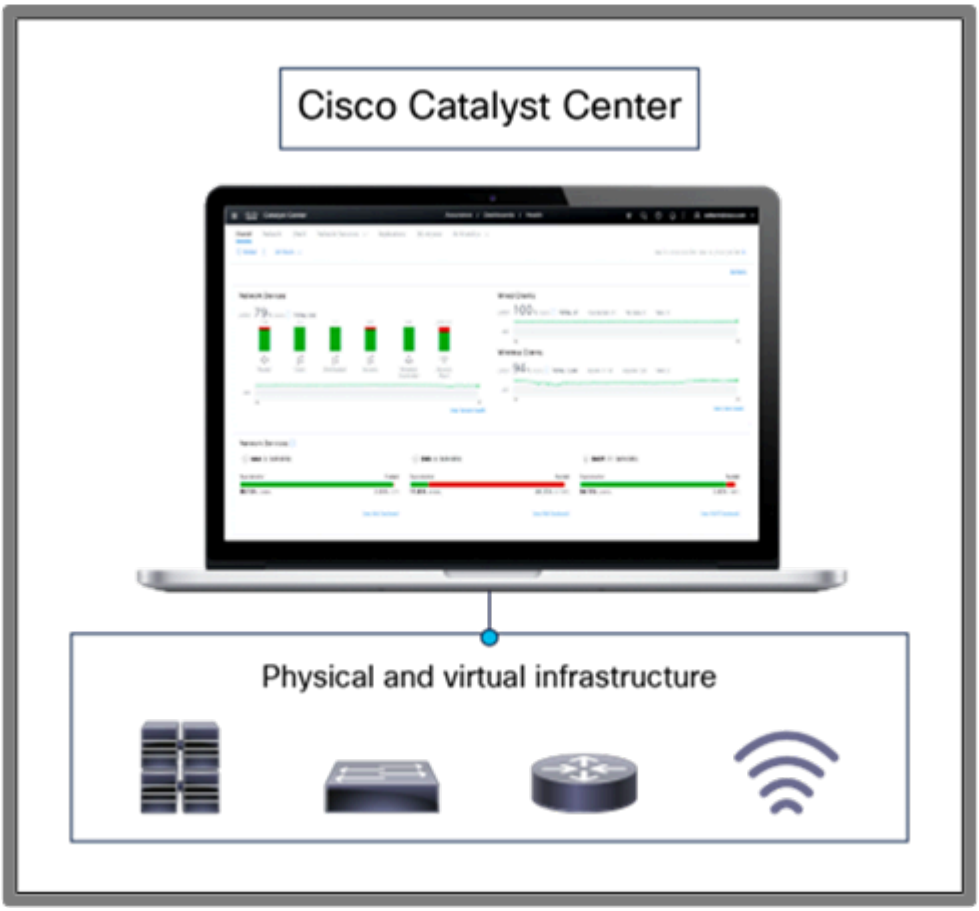


그림 1: Cisco Catalyst Center로 인프라 관리

주요 기능 및 혜택

Cisco Catalyst Center(CC)는 네트워크 관리 및 자동화를 간소화하는 고급 기능을 제공합니다.

ZTP(Zero-Touch Provisioning): 디바이스 온보딩을 자동화하여 수작업 및 구축 시간을 단축합니다.

소프트웨어 이미지 관리(SWIM): 업그레이드 전/후 확인을 통해 장치 간에 일관된 소프트웨어 버전을 보장하여 문제를 방지합니다.

의도 기반 자동화: 네트워크 용도를 유선 및 무선 네트워크에 대한 디바이스 구성으로 변환하여 구축을 간소화합니다.

LAN 자동화: 레이어 3 IP 주소 지정 및 라우팅을 자동화하여 엔드 투 엔드 토폴로지를 생성합니다.

무선 네트워크 자동화: PnP(Plug and Play)와 같은 기능으로 무선 액세스 포인트를 신속하게 프로비저닝할 수 있습니다.

계층적 네트워크 관리: 여러 위치에 걸쳐 일관된 구축을 위해 사이트별 프로파일(예: SSID, RF 매개변수, VLAN)을 허용합니다.

CLI 템플릿: Catalyst Center 템플릿 편집기를 사용하면 관리자가 CLI 기반 구성 템플릿을 쉽게 생성 및 관리할 수 있어 여러 디바이스에 일관되고 효율적인 구축이 가능합니다.

보증: 보증은 CC를 통해 관리되는 디바이스에 대한 중앙 집중식 모니터링을 허용합니다.

Cisco Catalyst Center는 이러한 기능 외에도 본 문서의 범위를 벗어나는 여러 기능을 제공합니다. 이 백서는 주로 Catalyst Center를 사용하여 CLI 템플릿을 설계하는 데 중점을 둡니다.

Catalyst Center를 사용한 LAN 캠퍼스 아키텍처의 개괄적 개요

기존 LAN 캠퍼스 네트워크는 엔터프라이즈 연결의 백본을 이루며, 유선 및 무선 장치에 대해 안정적이고 확장 가능한 통신을 보장합니다. 이러한 네트워크는 일반적으로 조직의 규모와 복잡성에 따라 3-Tier 아키텍처 또는 Collapsed Core 아키텍처를 사용하여 설계됩니다.

3계층 아키텍처

3-Tier 아키텍처는 코어 레이어, 디스트리뷰션 레이어 및 액세스 레이어로 구성된 기본 네트워크 설계 모델입니다. 이 아키텍처는 확장성, 고성능, 효율적인 트래픽 관리를 제공합니다. 각 레이어의 개요를 참조하십시오.

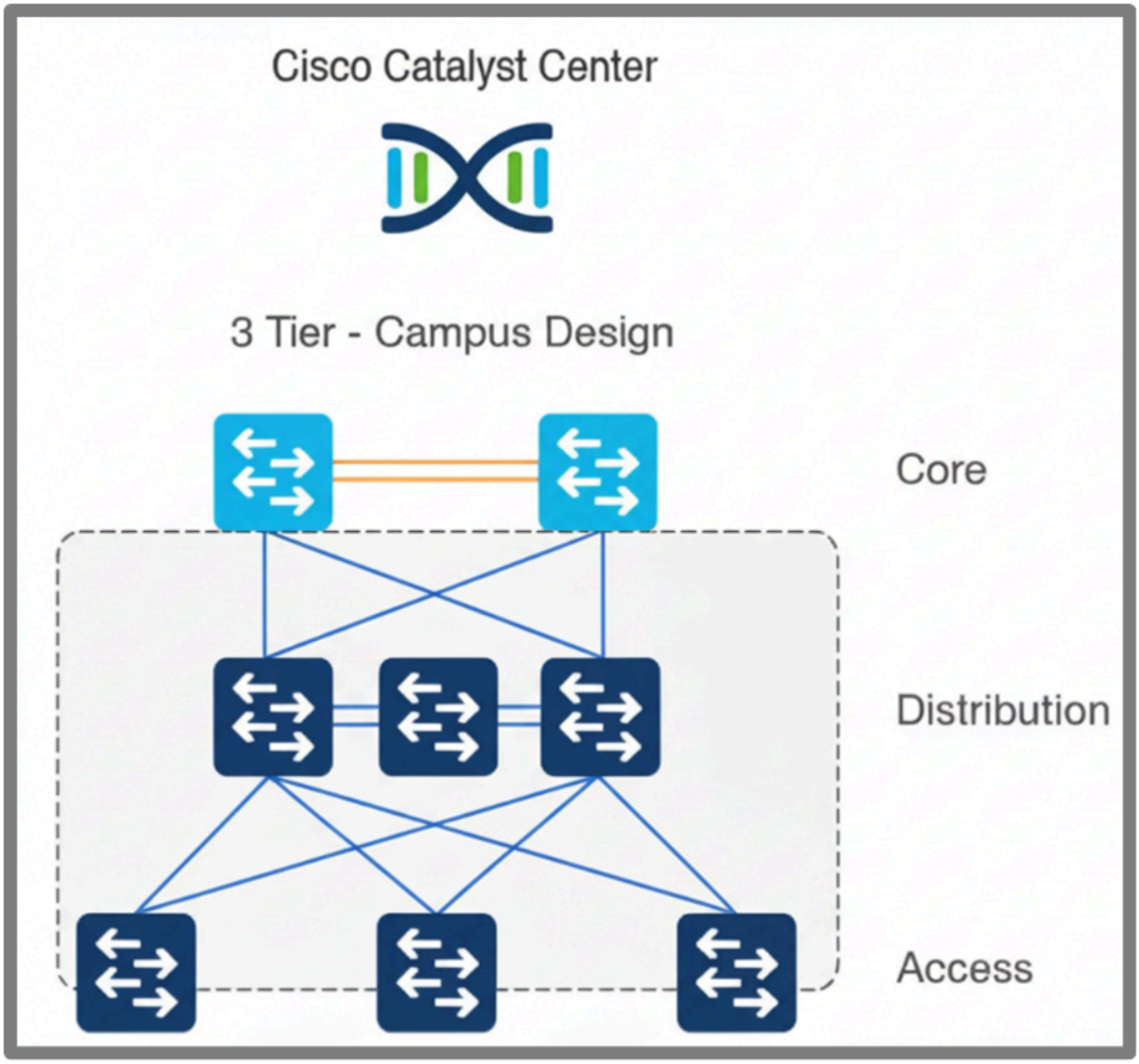


그림 2: 3계층 캠퍼스 아키텍처

코어 레이어

코어 레이어는 네트워크의 백본 역할을 하며 고속 연결 및 확장성을 제공합니다. 주요 구성에는 노스바운드 및 사우스바운드 라우팅 프로토콜(OSPF 및 BGP 등), 경로 정책, 다운링크 및 업링크 인터페이스 구성, 보안 강화 등이 포함됩니다

디스트리뷰션 레이어

디스트리뷰션 레이어는 코어 레이어와 액세스 레이어를 브리징하여 트래픽 어그리게이션, 정책 시행, 이중화를 처리합니다. 주요 구성으로는 이중화를 위한 HSRP/VRRP, 루프 방지를 위한 STP, 레이어 2 및 레이어 3 VLAN, 업링크 및 다운링크 인터페이스 구성, 보안을 위한 ACL, 보안 강화 등이 있습니다.

액세스 레이어

액세스 레이어는 엔드포인트를 네트워크에 연결하여 안전하고 신뢰할 수 있는 액세스를 지원합니다. 주요 컨피그레이션에는 액세스 인터페이스 컨피그레이션, 업링크 인터페이스 컨피그레이션, 레이어 2 VLAN, 디바이스에 대한 액세스를 제한하는 ACL, 보안 강화 등이 포함됩니다.

축소된 코어 아키텍처

Collapsed Core Architecture는 코어 레이어와 디스트리뷰션 레이어를 단일 레이어로 결합하여 성능과 확장성을 유지하면서 복잡성과 비용을 줄입니다. 이 접근 방식은 별도의 코어 레이어가 필요하지 않은 중소 규모 네트워크에 적합합니다. 이 아키텍처의 레이어 개요를 참조하십시오.

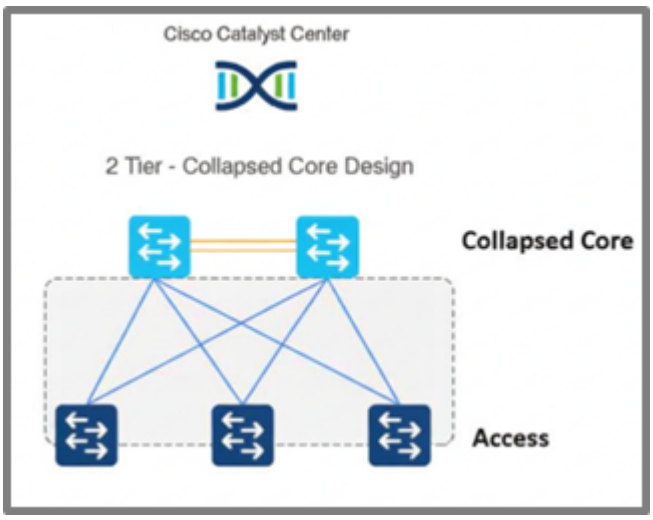


그림 3: 축소된 코어 캠퍼스 아키텍처

축소된 코어 레이어

축소된 코어 레이어는 코어 레이어와 디스트리뷰션 레이어의 기능을 결합하여 백본 연결, 트래픽 어그리게이션, 정책 시행을 제공합니다. 주요 컨피그레이션에는 노스바운드 및 사우스바운드 라우팅 프로토콜(OSPF 및 BGP 등), 경로 정책, 다운링크 및 업링크 인터페이스 컨피그레이션, 결함 탐지를 위한 BFD, SVI를 사용한 VLAN 간 라우팅, 게이트웨이 이중화를 위한 HSRP/VRRP, 루프 방지를 위한 STP, 보안 강화 등이 포함됩니다. Cisco Catalyst Center의 템플릿을 활용하면 이러한 구성을 자동화하여 일관되고 효율적인 구축을 보장할 수 있습니다.

액세스 레이어

앞서 설명한 것처럼 액세스 레이어는 엔드포인트를 네트워크에 연결하여 안전하고 신뢰할 수 있는 액세스를 지원합니다. 주요 컨피그레이션에는 액세스 인터페이스 컨피그레이션, 업링크 인터페이스 컨피그레이션, 레이어 2 VLAN, 디바이스에 대한 액세스를 제한하는 ACL, 보안 강화 등이 포함됩니다.

템플릿 설계 고려 사항

이 섹션에서는 Cisco Catalyst Center에서 디바이스 컨피그레이션 생성을 위한 템플릿을 설계하는 방법에 대해 설명합니다. 템플릿 편집기는 재사용 가능한 CLI 템플릿 생성을 활성화하고 네트워크에 맞춤화된 구성의 동적 구축을 지원하여 프로비저닝을 간소화합니다. Catalyst Center는 두 가지 템플릿 언어를 지원합니다. Jinja2와 Velocity입니다 이러한 언어는 디바이스에 대한 컨피그레이션 관리에 도움이 됩니다.

Jinja는 HTML, XML 또는 기타 텍스트 기반 형식과 같은 동적 콘텐츠를 생성하기 위해 주로 Python과 함께 사용되는 디자이너 친화적인 템플릿 언어입니다. 이를 통해 템플릿 내에 변수 및 제어 구조(예: 루프 및 조건)를 포함시켜 동적 출력을 생성할 수 있습니다.

Apache Velocity는 Java 기반 템플릿 엔진으로, VTL(Velocity Template Language)을 사용하여 웹 페이지, XML 또는 소스 코드를 비롯한 다양한 문서의 동적 콘텐츠를 활성화합니다. Java 객체의 데이터를 템플릿과 병합하여 최종 출력을 생성합니다.

이 문서에서는 Jinja2 템플릿만 다룹니다.

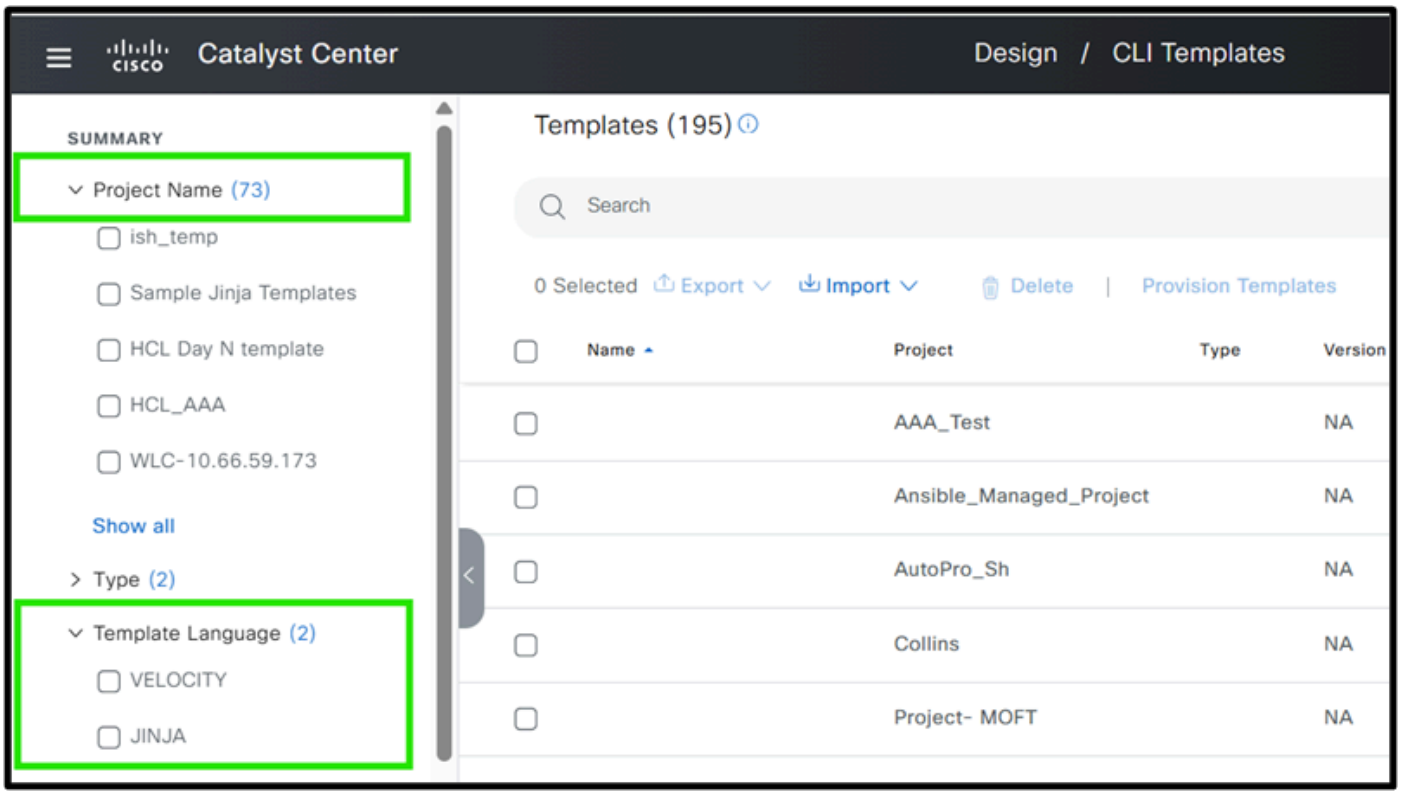


그림 4: Cisco Catalyst Center 템플릿 편집기

이 문서에서는 유연성이 뛰어나 Jinja2를 사용합니다. Jinja2에 대한 심층적인 탐색보다는 템플릿 설계를 위한 실제적인 적용에 초점을 맞추고 있다. Catalyst Center의 Jinja2 템플릿에 대한 자세한 내용은 다음 링크를 참조하십시오.

https://ciscolearning.github.io/cisco-learning-codelabs/posts/cat-center-j2-part-1/#0

Cisco 캠퍼스 네트워크를 위한 템플릿 설계 전략을 살펴보기 전에 템플릿으로 작업할 때 효율성과 관리 용이성을 보장하기 위해 주요 모범 사례를 활용하는 것이 중요합니다.

템플릿 구조 및 모범 사례/모범 전략 지침

Cisco Catalyst Center를 사용하여 네트워크 디바이스 구성을 자동화할 때는 구조화된 전략과 모범 사례를 채택하는 것이 중요합니다. 이러한 단계를 통해 네트워크 인프라 전반에서 일관성, 확장성 및 관리 용이성을 보장할 수 있습니다.

장치 역할별로 구성 나누기

먼저 네트워크 토폴로지에서의 역할에 따라 디바이스를 분류합니다. 일반적인 역할은 다음과 같습니다.

- 코어
- 배포
- 액세스

예: 코어 스위치로 작동하는 디바이스는 액세스 스위치와 비교하여 다른 구성 요구 사항을 가져야 합니다.

모듈형 블록으로 구성 구획화

각 디바이스 역할 내에서 유사한 기능 또는 구성을 함께 그룹화하여 구성을 모듈형 블록으로 나눕니다. 이러한 모듈식 접근 방식은 자동화, 문제 해결 및 향후 업데이트를 간소화합니다.

코어 디바이스의 예:

- OSPF 컨피그레이션 블록
- BGP 컨피그레이션 블록

QoS 정책 차단

역할에 구애받지 않는 컨피그레이션 블록 식별

일부 컨피그레이션 블록은 모든 디바이스 역할에 전체적으로 적용됩니다. 이러한 블록을 식별하고 표준화하면 네트워크 전체에서 모범 사례와 일관성이 보장됩니다.

역할에 구애받지 않는 일반 컨피그레이션 블록:

기본 컨피그레이션: 호스트 이름, 로그인 배너

관리 프로토콜: DHCP, DNS, NTP, SNMP

액세스 정책: 표준 보안 구성

이러한 블록은 Core, Distribution, Access 디바이스에 재사용될 수 있어 자동화 프로세스가 간소화됩니다.

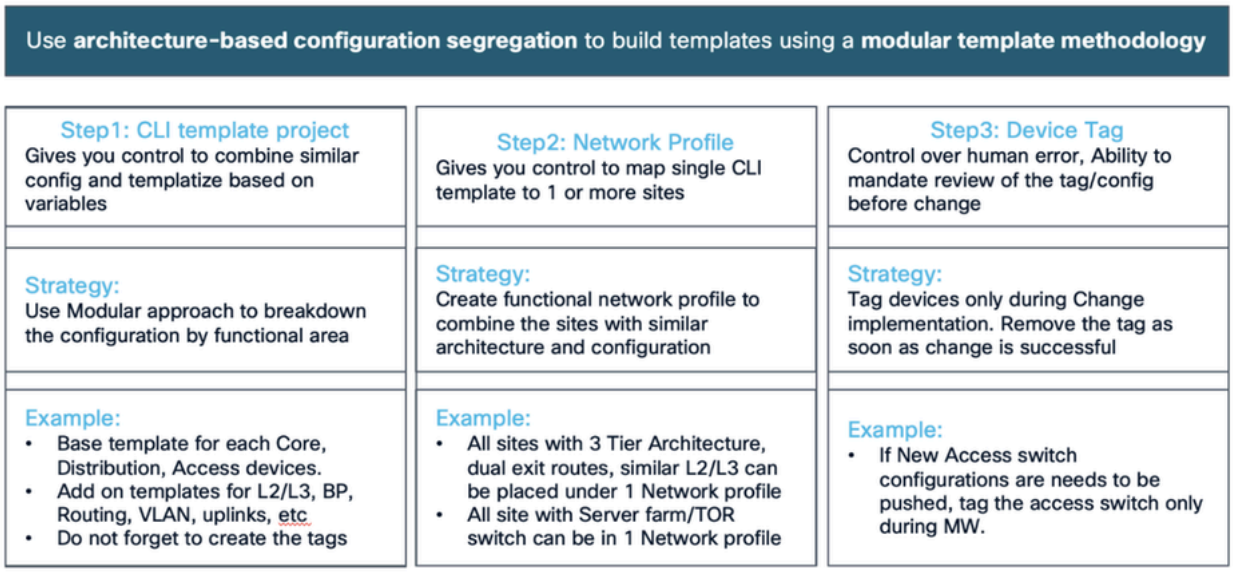


그림 1: 모범 사례

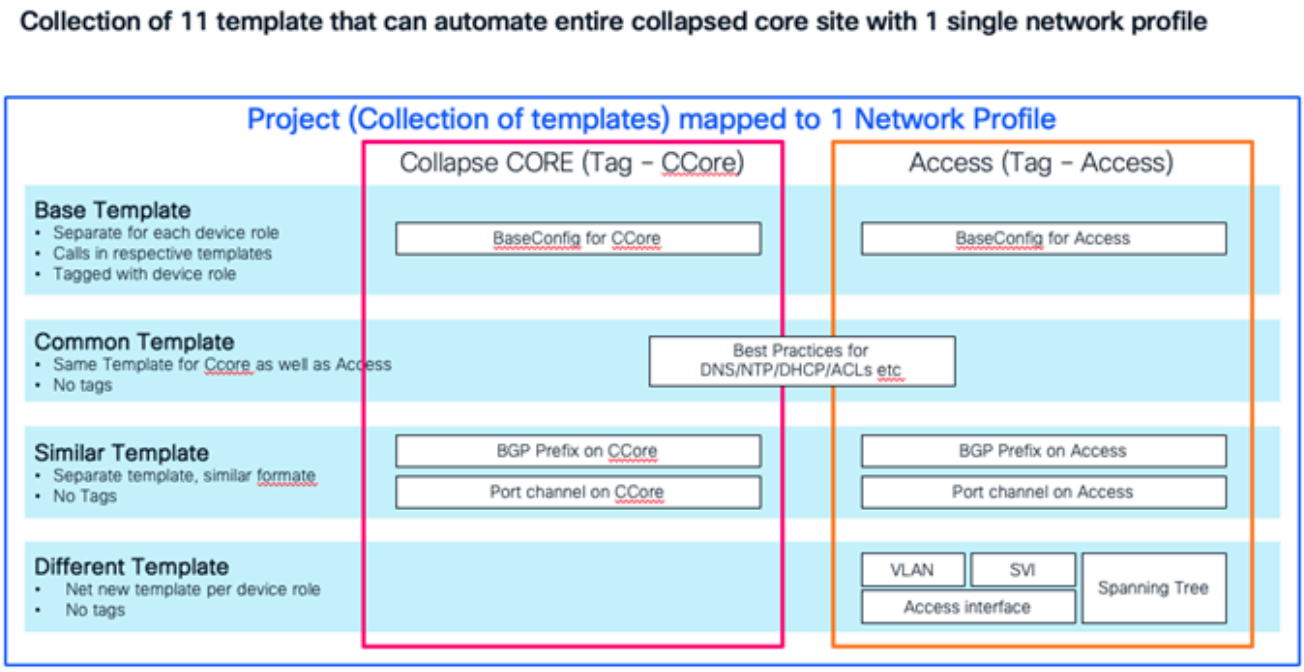


그림 2: 축소된 코어 템플릿 예

템플릿 작업에 대한 모범 사례

자동화된 구성을 위한 모듈식 템플릿 설계

Cisco Catalyst Center에서 장치 구성을 자동화할 때 모든 구성을 단일 템플릿에 포함시키지 마십시오. 그 대신 모듈형 접근 방식을 채택하십시오.

더 작은 용도별 템플릿(모듈)을 참조하는 기본 템플릿을 생성합니다.

컨피그레이션을 논리적 모듈(예: 인터페이스 설정, 라우팅 프로토콜, 보안 기능)로 세분화합니다.

이러한 구조를 통해 업데이트 효율성을 높일 수 있습니다. 특정 모듈에 대한 변경 사항은 해당 모듈이 사용되는 모든 위치에 자동으로 반영되므로 오류와 복잡성이 크게 줄어듭니다.

예: 브랜치 디바이스에 대한 모듈형 컨피그레이션

브랜치 디바이스에 대한 컨피그레이션을 자동화하고 있다고 가정합니다.

기본 템플릿:

주요 컨피그레이션 영역에 대한 모듈 템플릿에 대한 참조를 포함합니다.

필요에 따라 변수를 사용자 지정을 위해 각 모듈에 전달합니다.

모듈 템플릿:

interface_settings: 인터페이스 구성을 관리합니다.

routing_protocols: OSPF, EIGRP 또는 BGP 설정을 포함합니다.

보안_기능: ACL, 방화벽 규칙 또는 기타 보안 정책을 정의합니다.

```
{% include "Branch/Interface Configuration" %}
{% include "Branch/Routing Protocol Configuration" %}
{% include "Branch/Security Configuration" %}

{{ Branch_Interface_Configuration(branch_id) }}
{{ Branch_Routing_Protocol_Configuration(branch_id, ospf_area) }}
{{ Branch_Security_Configuration(branch_id) }}
```

기본 템플릿 구조 예:

이 구조를 사용하면 라우팅 또는 보안 컨피그레이션을 변경하는 작업은 해당 모듈에서만 수행하면 되며, 이러한 변경 사항은 기본 템플릿을 사용하는 모든 위치에 즉시 반영됩니다. 따라서 모든 브랜치 라우터에서 컨피그레이션을 더욱 쉽게 관리할 수 있고 일관되게 구성할 수 있습니다.

여기서 프로젝트 이름은 Branch이고 다른 3개의 모듈이 프로젝트 아래에 정의됩니다. 이러한 모든 항목은 기본 템플릿에서 결합됩니다.

템플릿에서 변수 최소화

복잡성과 오류를 줄이기 위해 템플릿의 변수 수를 최소한으로 유지합니다. 특히 대규모 네트워크 전체에서 더 적은 수의 변수가 구축을 단순화하므로 프로세스의 효율성과 일관성이 향상됩니다.

템플릿에 디바이스 태그 사용

Cisco Catalyst Center에서 위치, 역할, 사이트 등의 디바이스 태그를 활용하여 동적이고 확장

가능한 Jinja2 템플릿을 생성합니다. 이러한 태그는 조건부 논리를 활성화하여 적절한 디바이스에 올바른 컨피그레이션이 적용되도록 합니다. 이 접근 방식은 오류를 최소화하고 다양한 네트워크 환경에서 템플릿 관리를 간소화합니다.

가능한 경우 하드 코드 정적 값

고정 값을 하드코딩하면 템플릿을 간소화하고 구축 효율성을 높일 수 있습니다. 일반적인 예로는 DNS, NTP 또는 Syslog 서버에 대한 IP 주소가 있으며, 이는 일반적으로 여러 디바이스에서 일관성을 유지합니다. 마찬가지로, 액세스 스위치에서 표준 VLAN ID를 사용하면 이러한 값을 하드코딩하여 가변성을 줄이고 구축을 가속화할 수 있습니다.

2단계 접근 방식 채택: Day 0 및 Day N 템플릿

Cisco Plug and Play와 같은 서비스를 사용하여 디바이스를 온보딩할 경우, 2단계 템플릿 전략을 사용합니다.

Day 0 템플릿: 기본 컨피그레이션을 푸시하여 디바이스가 Cisco Catalyst Center와 통신할 수 있도록 합니다.

N일 템플릿: 디바이스에 연결할 수 있게 되면 고급 기능 및 컨피그레이션을 구축합니다.

모범 사례에서는 효율적이고 확장 가능한 템플릿을 지원하여 Cisco 캠퍼스 네트워크 구축을 간소화합니다.

Jinja 템플릿 매크로의 공백 컨트롤

진자어를 사용하여 템플릿을 만들 때는 특히 매크로 내에서 동적 콘텐츠를 렌더링할 때, 공백 및 줄 바꿈을 신중하게 처리하는 것이 중요합니다. 공백 누적 또는 의도하지 않은 새 행은 생성된 출력의 서식 문제로 이어질 수 있으며, 이는 다운스트림 처리에서 잘못된 해석 또는 오류를 유발해야 합니다. 이를 해결하기 위해 Jinja는 공백을 제어하기 위한 구문을 제공합니다. 구분 기호({{- ... -}} 또는 {%- ... -%}) 바로 내부에 빼기 기호(-)를 배치하면 식 주위의 선행 또는 후행 공백이 모두 제거됩니다. 예를 들어 {{item[1]}}을(를) {{- item[1] -}}(으)로 바꾸면 매크로를 렌더링할 때 추가 공백 또는 새 줄이 제거됩니다. 이 방법은 템플릿 스니펫에 나와 있는 것처럼 목록을 반복하거나 컨피그레이션 파일을 생성할 때 특히 유용합니다. 이러한 시나리오에서는 항상 whitespace 컨트롤을 적용하여 깨끗하고 예측 가능한 출력을 유지하는 것이 좋습니다.

예(권장 사용량):

```
{% 와일드카드 목록 %}의 항목에 대한
   {% if item[0] == prefix -%}
      {{- 항목[1] -}}
```

{%- endif %}
{%- %} 종료

3계층 아키텍처

이 백서는 코어 스위치에 대한 액세스 스위치의 템플릿을 개발하는 것으로 시작하여 각 레이어의 요구 사항을 간략하게 설명합니다.

액세스 레이어 스위치

액세스 스위치는 플러그 앤 플레이를 사용하여 온보딩되며 Day 0 템플릿이 필요합니다. Catalyst Center의 플러그 앤 플레이 프로세스에 대한 자세한 내용은 다음 링크를 참조하십시오.

https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/catalyst-center/2-3-7/user_guide/b_cisco_catalyst_center_user_guide_237/m_onboard-and-provision-devices-with-plug-and-play.html

앞서 설명한 대로 Catalyst Center는 Velocity 및 Jinja2 템플릿 언어를 모두 지원합니다. 이 문서에서는 Jinja2를 사용하여 템플릿 구조를 설명하는데, 유연성이 뛰어납니다. Day-0 및 Day-N 템플릿을 사용하여 액세스 레이어 스위치 컨피그레이션을 구축할 수 있습니다.

기본 Day 0 템플릿은 구성할 수 있습니다. 1단계:

1단계: 템플릿 정의

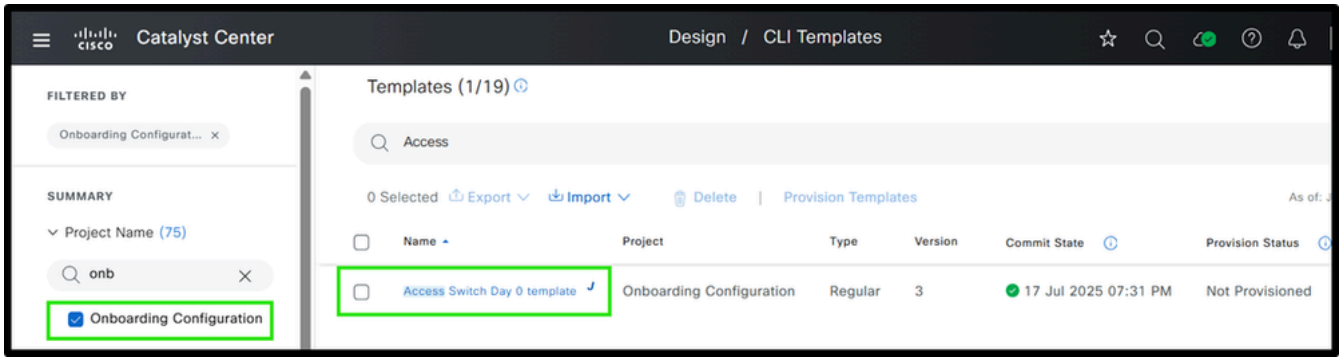
```
username admin privilege 15 password SamplePass123
!
enable secret EnableSecret123
!
ip routing
!
vlan {{ branch_number * 100 + 13 }}
 name SW_MGMT
!
interface vlan {{ branch_number * 100 + 13 }}
 ip address {{ ip_address }} 255.255.255.128
 no ip redirects
 no ip unreachables
 no ip proxy-arp
!
ip route 0.0.0.0 0.0.0.0 {{ nexthop }} name Default-Gateway
!
interface range Te1/1/1 - 2
 switchport
 switchport mode trunk
 no shutdown
!
```

1단계: 템플릿 정의

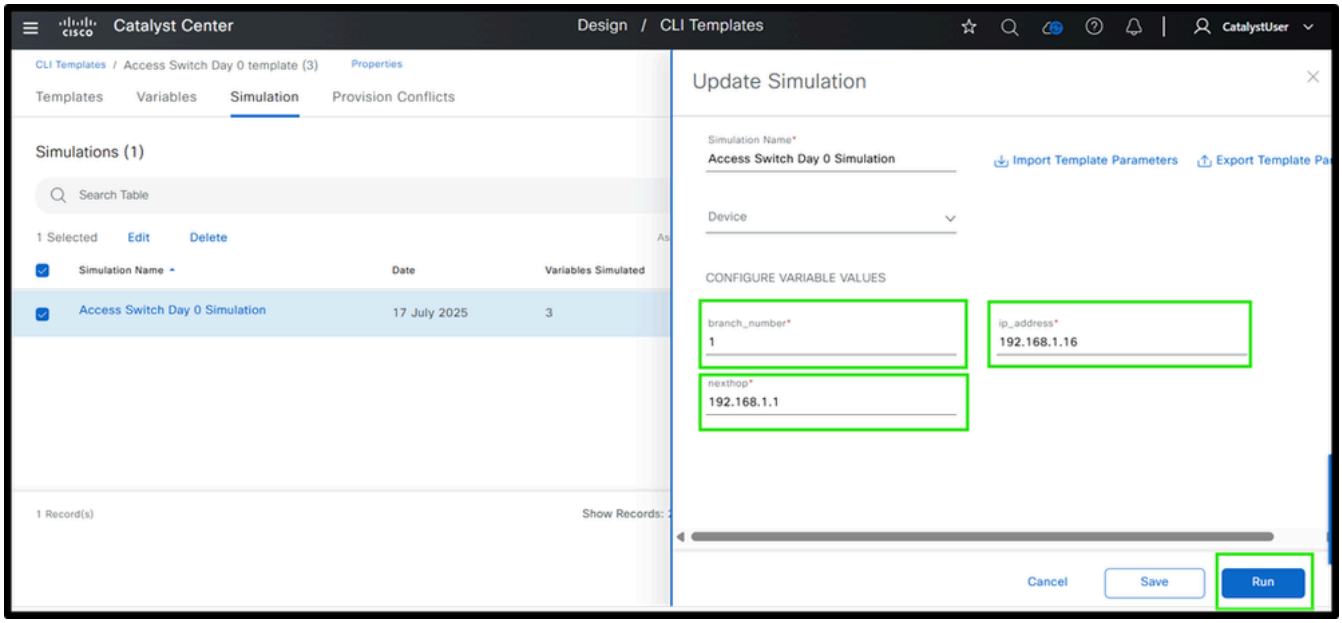
이 템플릿은 사용자 이름, 비밀번호, enable 비밀, 서브넷 마스크 등의 상수를 하드 코딩하여 구성을 간소화합니다. 브랜치의 모든 스위치는 동일한 관리 VLAN 서브넷 마스크를 공유하기 때문입니다. 그러나 관리 IP 주소는 스위치마다 고유하며 변수로 정의됩니다. 이 Day 0 템플릿을 사용하는 Day N 템플릿에서 포괄적인 템플릿 구조를 제공해야 합니다. Day N 템플릿에서 액세스 스위치의 각 기능은 전용 모듈에 의해 관리됩니다. 예를 들어, 한 모듈에서 레이어 2 VLAN을 처리하고, 별도의 모듈에서 업링크 및 다운링크 액세스 인터페이스를 관리하며, 다른 모듈에서 보안 강화에 중점을 둡니다.

일관된 VLAN ID가 선호되지만, 브랜치 번호를 기반으로 하는 공식을 사용하여 가변 ID를 동적으로 생성할 수 있습니다(예: 브랜치 1 = VLAN 113, 브랜치 2 = VLAN 213). 이렇게 하면 여러 브랜치에서 템플릿을 재사용할 수 있습니다. 마찬가지로 next-hop IP는 연결된 분산 클러스터에 따라 브랜치마다 달라야 하므로 변수입니다.

2단계: 시뮬레이션 수행 및 변수 제공

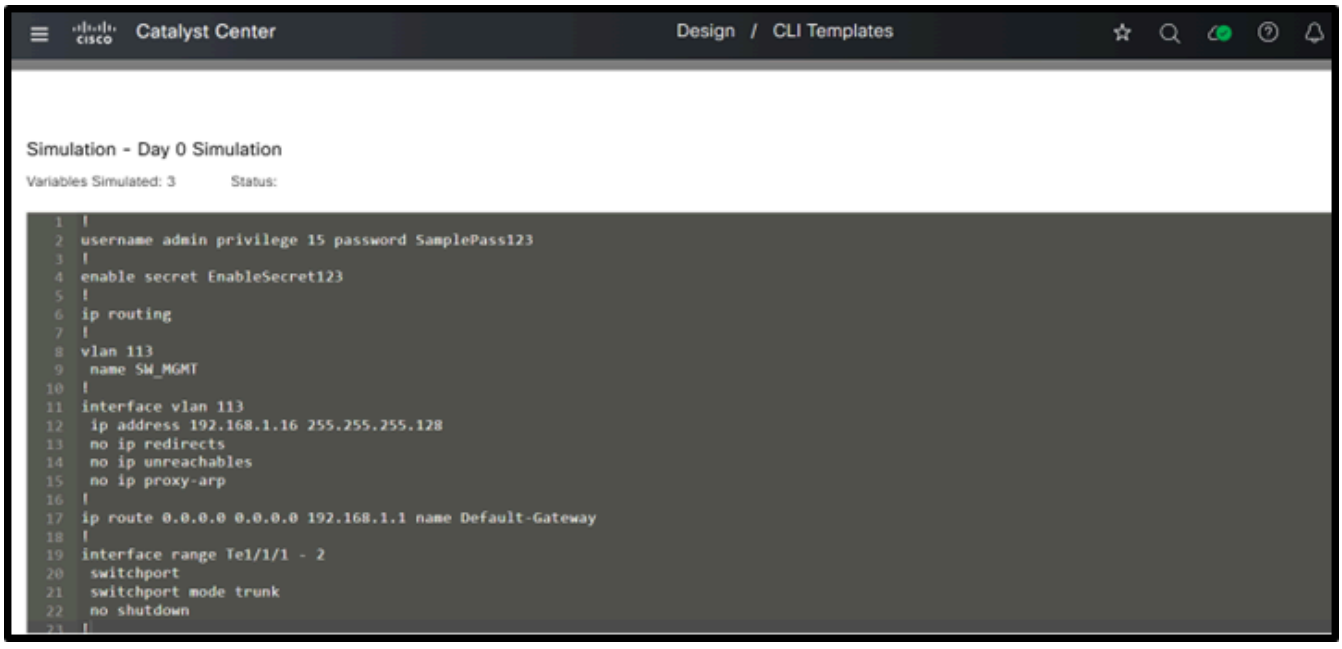


시뮬레이션 입력 및 출력이 있는 Switch Day 0 템플릿 구조



예: 시뮬레이션 입력

항상 구축 전에 템플릿을 시뮬레이션하는 것이 좋습니다. 스크린샷은 변수를 입력한 후 최종 컨피그레이션을 보여줍니다.



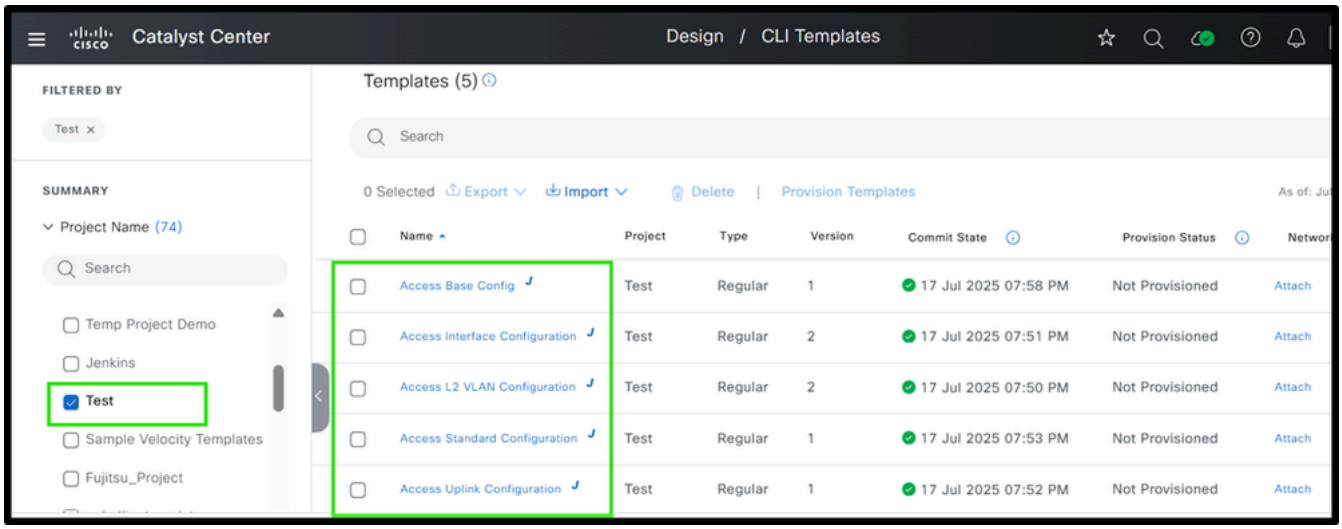
값을 입력한 후 FInal 컨피그레이션

이제 모듈형 Day N 템플릿을 만드는 방법을 살펴보겠습니다.

액세스 스위치 구성은 다양한 모듈들로 분할될 수 있고, 이들 모두는 기본 모듈 내에 결합될 수 있다. 액세스 스위치의 기본 템플릿은 다음과 같이 구성됩니다.

기본 템플릿과 해당 모듈은 모두 Cisco Catalyst Center의 "테스트"라는 프로젝트 내에서 생성됩니다.

1단계: 기본 템플릿을 비롯한 다양한 템플릿 정의



액세스 스위치 Day N 템플릿 구조

2단계: 다양한 모듈 정의

액세스 기반 구성:

스크린샷은 기본 컨피그레이션의 예를 보여줍니다.

```
{% include "Test/Access L2 VLAN Configuration" %}
{% include "Test/Access Interface Configuration" %}
{% include "Test/Access Uplink Configuration" %}
{% include "Test/Access Standard Configuration" %}

{{ Access_L2_VLAN_Configuration(branch_number, is_poe) }}
{{ Access_Uplink_Configuration(branch_number, is_poe)}}
{{ Access_Interface_Configuration(branch_number, is_poe) }}
{{ Access_Standard_Configuration(branch_number) }}
```

액세스 기반 구성

이 모듈형 컨피그레이션 템플릿은 다음 네 부분으로 구성됩니다. VLAN 컨피그레이션, 업링크 인터페이스 컨피그레이션, 액세스 인터페이스 컨피그레이션, 표준 컨피그레이션. 두 가지 변수만 사용합니다. branch_number 및 is_poe를 통해 간단하고 관리하기 쉬움

branch_number는 Day 0 템플릿과 같이 브랜치 특정 VLAN ID를 계산하고 is_poe는 액세스 스위치가 PoE인지 또는 PoE가 아닌 스위치인지를 결정합니다. 이러한 변수는 프로비저닝 중에

제공되며 모듈에 전달하여 올바른 컨피그레이션을 생성하므로 수고를 줄이고 효율성을 향상합니다.

이제 각 모듈이 전체 구성의 특정 부분을 생성하는 데 어떻게 기여하는지 살펴보겠습니다.

액세스 L2 VLAN 컨피그레이션

```
{% macro Access_L2_VLAN_Configuration (branch_number, is_poe) %}
!
vlan {{ 100 * branch_number + 11 }}
 name DATA_VLAN
!
vlan {{ 100 * branch_number + 12 }}
 name VOICE_VLAN
!
{% if is_poe == 'Yes' %}
vlan {{ 100 * branch_number + 14 }}
 name AP_Mgmt
{% endif %}
!
{% endmacro %}
```

액세스 L2 VLAN 컨피그레이션

이 모듈은 앞서 설명한 대로 브랜치 번호를 기반으로 VLAN을 생성합니다. 데이터 및 음성 VLAN은 PoE 지원 여부에 관계없이 모든 스위치에서 생성됩니다. AP 관리 VLAN(예: 브랜치 1의 114)은 is_poe가 "Yes"로 설정된 경우에만 생성됩니다. 즉 스위치가 PoE를 지원한다는 의미입니다. is_poe가 "No"인 경우 비 PoE 스위치에서 액세스 포인트를 지원할 수 없으므로 AP 관리 VLAN을 건너뜁니다. 이는 if 조건을 사용하여 관리됩니다.

```
{% macro common_access_settings() %}
 switchport port-security maximum 2
 switchport port-security
 switchport port-security violation shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
 storm-control broadcast level 2.00
 storm-control multicast level 2.00
 storm-control unknown-unicast 2.00
{% endmacro %}

{% macro Access_Interface_Configuration(branch_number, is_poe) %}
!
interface range Gi1/0/1 - 6
{% if is_poe == 'Yes' %}
 description *** AP ***
 switchport mode access
 switchport access vlan {{ 100 * branch_number + 14 }}
{% else %}
 description *** User Ports ***
 switchport mode access
 switchport access vlan {{ 100 * branch_number + 11 }}
 switchport voice vlan {{ 100 * branch_number + 12 }}
{% endif %}
{{ common_access_settings() }}
!
interface range Gi1/0/7 - 24
 description *** User Ports ***
 switchport mode access
 switchport access vlan {{ 100 * branch_number + 11 }}
 switchport voice vlan {{ 100 * branch_number + 12 }}
{{ common_access_settings() }}
!
{% endmacro %}
```

액세스 인터페이스 컨피그레이션

이 모듈은 액세스 인터페이스 컨피그레이션을 처리하고 앞에서 설명한 PoE 스위치와 동일한 접근 방식을 사용합니다. is_poe 변수가 "Yes"이면 스위치가 PoE 스위치임을 의미합니다. 처음 6개 포트(1-6)는 AP 관리 VLAN으로 구성해야 합니다. 그렇지 않으면 처음 6개 포트를 사용자 액세스 포트로 설정해야 합니다.

스위치가 24포트 모델이라고 가정하면 나머지 포트(7~24)는 스위치가 PoE인지 여부에 관계없이 항상 사용자 액세스 포트로 구성됩니다.

인터페이스 범위가 표준화되어 더 이상 입력 변수로 사용되지 않으므로 템플릿의 변수 수를 최소화하는 것이 좋습니다. 또한 모듈에는 common_access_settings라는 매크로가 포함되어 있습니다. 이 매크로는 반복적인 구성을 통합하여 템플릿 크기를 최소화합니다. 이 매크로는 인터페이스 설정 내에서 호출되므로 여러 번 지정할 필요가 없습니다.

참고: 이 템플릿은 모든 액세스 인터페이스에 동일한 설명을 적용합니다. 각 인터페이스에 고유한 설명이 필요한 경우 별도의 Python 스크립트 또는 유사한 자동화 툴을 사용하여 푸시하는 것이 좋습니다.

업링크 인터페이스에 대한 컨피그레이션을 생성하는 모듈을 검토합니다.

```
{% macro Access_Uplink_Configuration(branch_number, is_poe) %}
{% if is_poe == 'Yes' %}
!
interface range Te 1/1/1 - 2
 switchport
 switchport mode trunk
 switchport trunk allowed vlan {{ branch_number * 100 + 11 }},{{ branch_number * 100 + 12 }},{{ branch_number * 100 + 13 }},{{
branch_number * 100 + 14 }}
 no shutdown
!
{% else %}
!
interface range Te 1/1/1 - 2
 switchport
 switchport mode trunk
 switchport trunk allowed vlan {{ branch_number * 100 + 11 }},{{ branch_number * 100 + 12 }},{{ branch_number * 100 + 13 }}
 no shutdown
!
{% endif %}
{% endmacro %}
```

액세스 업링크 구성

이 모듈은 업링크 인터페이스에 대한 컨피그레이션을 생성하고 VLAN 프루닝을 처리합니다. 스위치가 PoE를 지원하는 경우 AP 관리 VLAN이 허용되는 VLAN 목록에 포함됩니다. 그렇지 않은 경우에는 제외됩니다. 이 논리는 앞에서 설명한 것처럼 코드에서 if 조건을 사용하여 관리됩니다.

모범 사례 및 보안 강화 등 표준 구성을 보여 주는 최종 모듈을 검토합니다.

⚠ 주의: 이는 예시 목적으로만 사용되며, 컨피그레이션이 특정 요건에 따라 달라질 수 있으므로 실제 네트워크 설정에 대한 참조로 사용하지 않아야 합니다

```
{% macro Access_Standard_Configuration (branch_number) %}
!
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vtp mode off
no errdisable recovery cause all
crypto key generate rsa modulus 2048
!
ip ssh version 2
ip ssh time-out 120
ip ssh source-interface vlan {{ branch_number * 100 + 13 }}
no ip http server
no ip http secure-server
ip http client source-interface vlan {{ branch_number * 100 + 13 }}
!
logging buffered informational
logging host 192.168.1.10
logging host 192.168.2.20
logging source-interface vlan {{ branch_number * 100 + 13 }}
!
ntp authentication
ntp authentication-key 10 md5 Netw0rkAuthKey
ntp source vlan {{ branch_number * 100 + 13 }}
ntp server 192.168.3.1 key 10
ntp server 192.168.3.2 key 10
!
snmp-server enable traps
snmp-server trap-source vlan {{ branch_number * 100 + 13 }}
snmp-server group NMSNWDEVICE v3 priv access SNMPHOST
snmp-server user netadmin NMSNWDEVICE v3 auth sha AuthKey123 priv aes 128 PrivKey123
!
ip access-list standard SNMPHOST
permit 192.168.4.0 0.0.0.255
!
ip access-list standard VTYACL
permit 192.168.5.10
```

```
permit 192.168.5.11
!
aaa new-model
ip tacacs source-interface vlan {{ branch_number * 100 + 13 }}
tacacs server TACACS_1
 address ipv4 192.168.6.1
 key TACACSKey123
 timeout 4
tacacs server TACACS_2
 address ipv4 192.168.6.2
 key TACACSKey123
 timeout 4
aaa group server tacacs+ TACACS-SERVER
 server name TACACS_1
 server name TACACS_2
!
aaa authentication login default group TACACS-SERVER local
aaa authorization exec default group TACACS-SERVER local
aaa accounting exec default start-stop group TACACS-SERVER
!
line console 0
login authentication default
exec-timeout 5 0
!
line vty 0 15
login authentication default
access-class VTYACL in
exec-timeout 5 0
!
banner login ^
********************************** WARNING ************************************
All systems/network should be used/accessed by authorized persons only
  If you are not authorized to do so, you should log off immediately
    Access to and usage of this system /network may be monitored
    All users must comply with information security policies
        Any Violation may lead to disciplinary action.
*******************************************************************************^
{% endmacro %}
```

2부: 액세스 표준 구성

이 모듈에서는 모범 사례, 보안 강화, 보안 장치 관리를 위한 주요 기능이 통합된 표준 컨피그레이션을 생성합니다. 대부분의 값은 브랜치 간의 일관성을 위해 하드코딩됩니다. 단, 각 브랜치의 스위치에 대한 관리 VLAN을 계산하는 데 사용되며 여러 컨피그레이션의 소스 인터페이스 역할을 하는 branch_number는 제외합니다.

3단계: 스위치를 구성하기 전에 시뮬레이션을 수행합니다. 기본 구성만 시뮬레이션해야 합니다

.

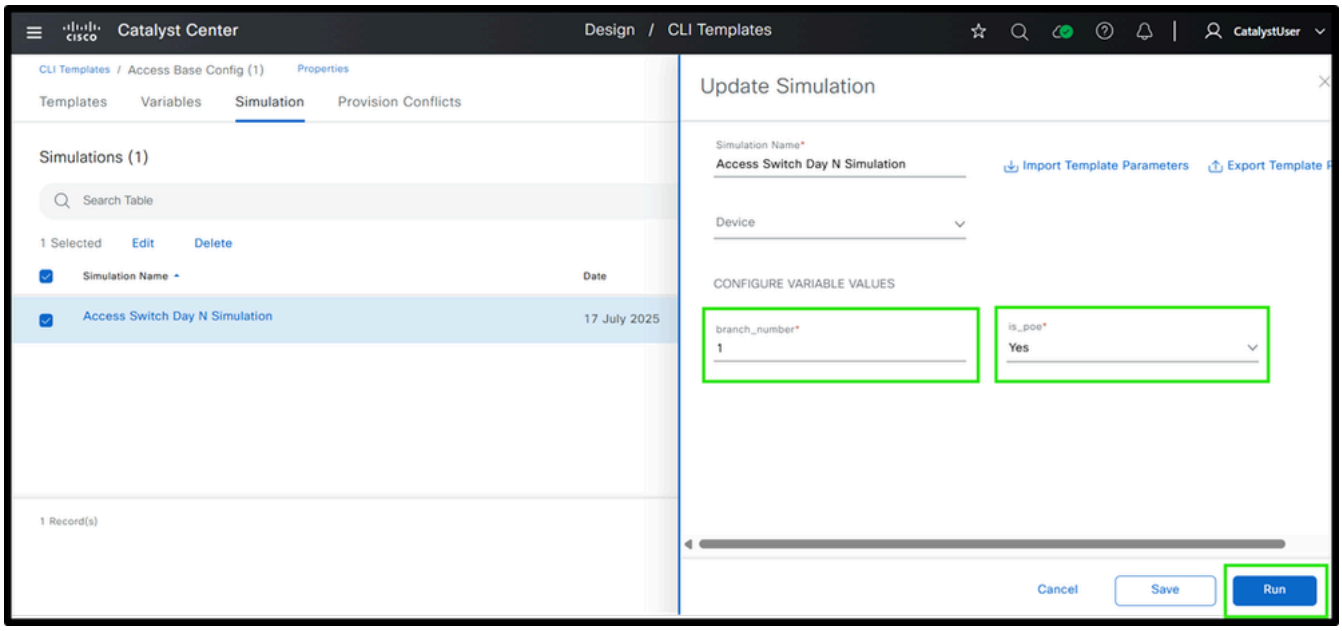
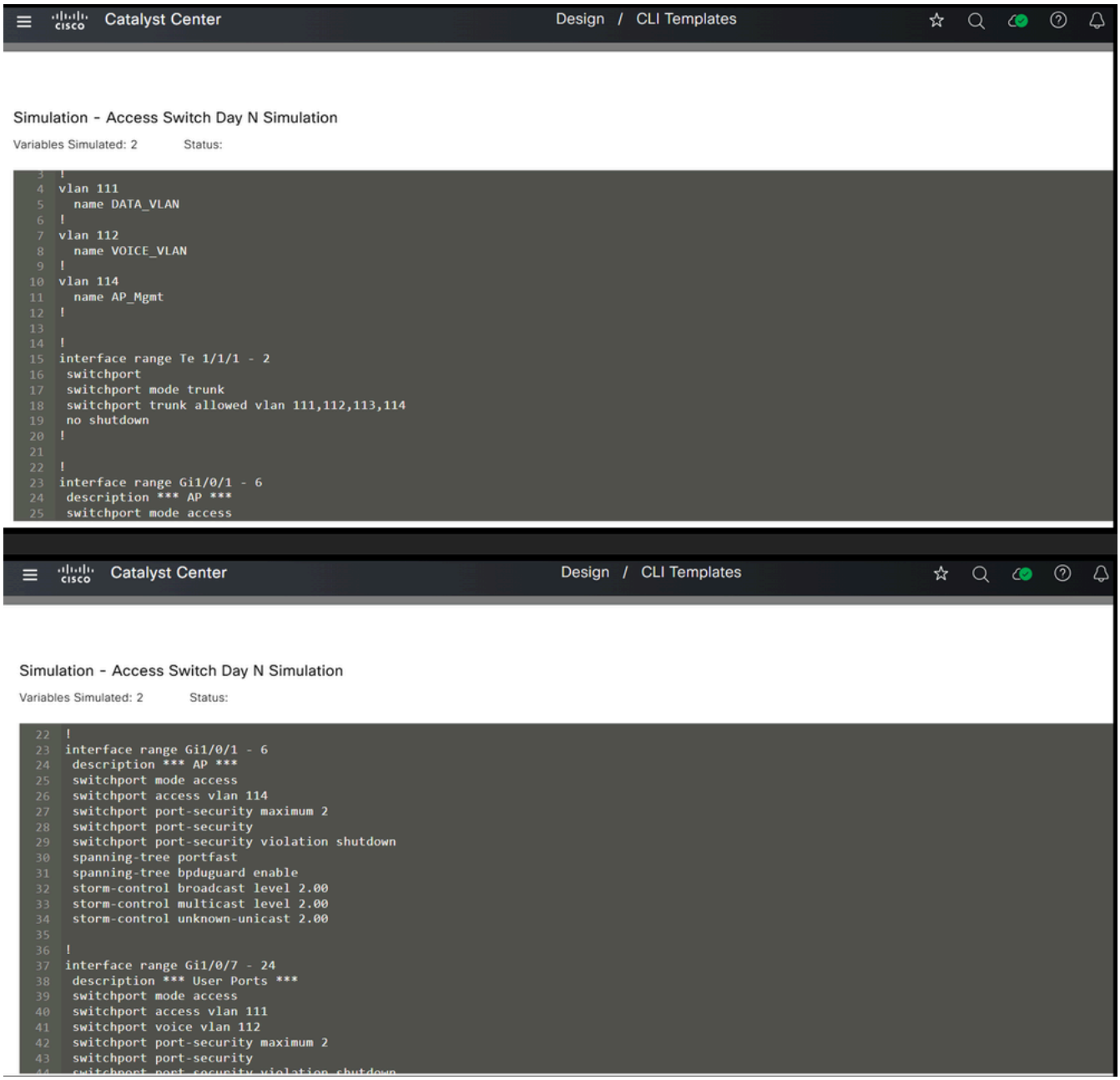


그림 7: 액세스 스위치 Day N 템플릿 시뮬레이션 입력 및 출력



시뮬레이션

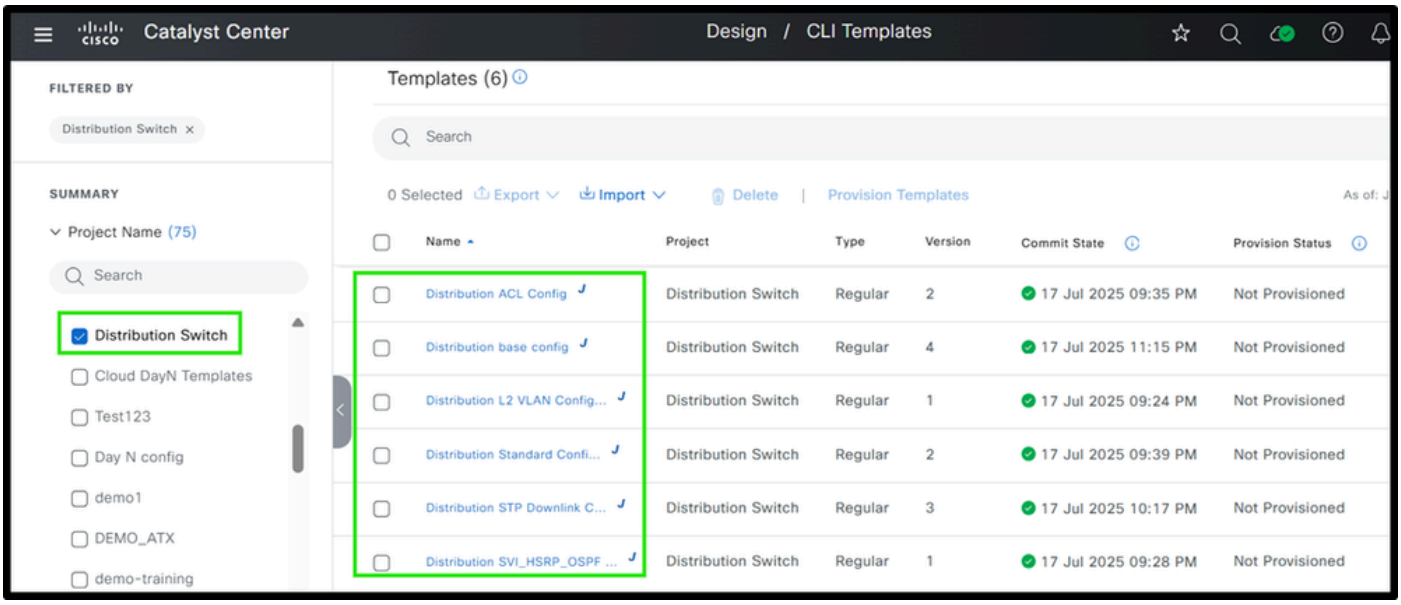
이는 액세스 레이어에서 템플릿을 사용하여 컨피그레이션을 생성하는 방법입니다.

이제 디스트리뷰션 레이어 장치에 템플릿을 적용할 수 있는 방법을 살펴보겠습니다.

디스트리뷰션 레이어 스위치

이제 분산 스위치용 모듈형 템플릿을 설계합니다. 기본 템플릿과 해당 모듈은 Cisco Catalyst Center의 'Distribution Switch' 프로젝트의 일부입니다.

1단계: 디스트리뷰션 스위치 템플릿 구조



예: 배포 템플릿

2단계: 각 모듈 정의

제공된 기본 컨피그레이션에서는 각 모듈을 정의하고 모든 모듈을 참조합니다.

```
{% include "Distribution Switch/Distribution L2 VLAN Configuration" %}
{% include "Distribution Switch/Distribution STP Downlink Config" %}
{% include "Distribution Switch/Distribution SVI_HSRP_OSPF Config" %}
{% include "Distribution Switch/Distribution ACL Config" %}
{% include "Distribution Switch/Distribution Standard Configuration" %}

{{ Distribution_L2_VLAN_Configuration(branch_number, is_poe) }}}}
{{ Distribution_STP_Downlink_Config(branch_number, is_poe, distribution_number) }}
{{ Distribution_SVI_HSRP_OSPF_Config(branch_number, is_poe, distribution_number) }}
{{ Distribution_ACL_Config(branch_number) }}
{{ Distribution_Standard_Config() }}
```

예: Distribution Base 템플릿 모듈

액세스 스위치와 마찬가지로, 모든 템플릿은 'Distribution Switch' 프로젝트 내에서 만들어지고 기본 템플릿에서 참조됩니다. 일부 템플릿은 액세스 스위치에 사용되는 템플릿과 동일하지만, 이 섹션에서는 분산 스위치의 차이점에 대해 설명합니다. 모듈 "Distribution L2 VLAN Configuration"은 앞서 액세스 스위치에 대해 설명한 것과 동일합니다. 이 정보를 제공하는 Access L2 VLAN Configuration 모듈을 확인하십시오. 변수에 제공된 입력 값을 기반으로 필요한 VLAN을 생성합니다.

이제 "Distribution STP Downlink Config" 모듈을 검토하십시오. 이 모듈은 디스트리뷰션 스위치에 대한 스패닝 트리 및 업링크 컨피그레이션의 생성을 처리합니다.

```
{% macro Distribution_STP_Downlink_Config (branch_number, is_poe, distribution_number) %}
!
spanning-tree mode rapid-pvst

{% set base_vlan = branch_number * 100 %}
{% set vlans = [base_vlan + 11, base_vlan + 12, base_vlan + 13] %}

{% if is_poe == 'Yes' %}
  {% set vlans = vlans + [base_vlan + 14] %}
{% endif %}

{% if distribution_number == 1 %}
   {% set stp_priority = 4096 %}
{% else %}
   {% set stp_priority = 8192 %}
{% endif %}

spanning-tree vlan {{ vlans | join(',') }} priority {{ stp_priority }}
!
interface range TWE 1/0/1 - 2
 switchport
 switchport mode trunk
 switchport trunk allowed vlan {{ vlans | join(',') }}
 no shutdown
!
{% endmacro %}
```

분산 STP 다운링크 구성

여기서 Jinja2 매크로 기능이 사용되고 있으며, 이는 기반 모듈에서 참조됩니다. 따라서 이 구조는 모듈형 접근 방식을 구축하는 데 도움이 됩니다.

이 모듈은 "branch_number" 및 스위치가 PoE를 활성화하는지 여부에 따라 STP(Spanning Tree Protocol) 및 다운링크 인터페이스를 구성합니다. "branch_number" 변수는 각 브랜치에 대해 고유한 기본 VLAN을 생성하는 데 사용되므로, 액세스 스위치에 대해 이미 강조 표시된 방식과 유사하게 고유한 VLAN을 보장합니다. 스위치가 PoE가 활성화된 경우("is_poe" == 'Yes'), AP 관리 VLAN과 같은 추가 VLAN이 목록에 추가됩니다. "distribution_number" 변수는 STP 우선순위를 결정하며, Distribution 1(기본 루트 브리지로 설정)에는 4096을 설정하고 보조 배포 스위치에는 8192를 설정합니다. 마지막으로, 적절한 VLAN이 트렁크 인터페이스에 적용되므로 스위치의 PoE 활성화 여부에 따라 관련 VLAN만 허용됩니다.

이제 효율적인 네트워크 라우팅 및 이중화를 위해 SVI, HSRP 및 OSPF의 설정에 중점을 두는 "Distribution SVI_HSRP_OSPF Config" 모듈을 검토합니다.

```
{% macro Distribution_SVI_HSRP_OSPF_Config (branch_number, is_poe, distribution_number) %}
!
interface loopback0
ip address {{ loopback_ip }} 255.255.255.255
!
router ospf 1
router-id {{ loopback_ip }}
!
key chain HSRP_KEY
 key 0
  key-string cisco@7875
!
interface vlan {{ 100 * branch_number + 11 }}
description Data_Endpoints
ip address 172.17.{{ (branch_number - 1) * 16 }}.{{ distribution_number + 1 }} 255.255.240.0
standby bfd
standby version 2
standby {{ 100 * branch_number + 11 }} ip 172.17.{{ (branch_number - 1) * 16 }}.1
{% if distribution_number == 1 %}
standby {{ 100 * branch_number + 11 }} priority 255
{% else %}
standby {{ 100 * branch_number + 11 }} priority 250
{% endif %}
standby {{ 100 * branch_number + 11 }} authentication md5 key-chain HSRP_KEY
standby {{ 100 * branch_number + 11 }} preempt delay minimum 120
no ip redirects
no ip unreachables
no ip proxy-arp
ip ospf 1 area 0
bfd interval 100 min_rx 100 multiplier 3
!
! uplink interfaces
interface TWE1/1/1
 no switchport
 ip address {{ twe1_1_1_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
interface TWE1/1/2
 no switchport
 ip address {{ twe1_1_2_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
{% endmacro %}
```

배포 SVI_HSRP_OSPF 구성

이 모듈인 Distribution_SVI_HSRP_OSPF_Config는 디스트리뷰션 스위치에 대한 SVI, HSRP, OSPF 및 업링크 인터페이스를 구성하는 데 도움이 됩니다. 이 예에서는 데이터 서브넷에 대한 SVI에 초점을 맞추고 있지만, 음성 또는 관리와 같은 다른 SVI에도 동일한 방법을 사용할 수 있습니다.

데이터 서브넷에 대한 IP 주소 계획이 이미 완료된 경우, branch_number 및 distribution_number 변수에 따라 각 SVI에 대해 IP 주소가 자동으로 계산될 수 있습니다. 예를 들어, 브랜치 1에 서브넷 172.17.0.0/20이 있고 브랜치 2에 172.17.16.0/20이 있으며 브랜치 3에 172.17.32.0/20이 있는 경우 게이트웨이 IP는 172.17.x.1(여기서 x는 브랜치 번호)입니다. 첫 번째 분산 스위치의 두 번째 IP는 172.17.x.2이고, 두 번째 분산 스위치의 세 번째 IP는 172.17.x.3입니다. 이렇게 하면 IP 주소가 자동으로 계산되어 오류가 줄어들고 프로세스가 단순화됩니다.

루프백 인터페이스에는 OSPF 라우터 ID 역할을 하는 variable loopback_ip의 IP가 할당되어 네트워크 전체에서 안정적이고 일관된 라우팅을 보장합니다. OSPF 컨피그레이션에서 이 루프백 IP는 라우터 ID로 사용되며 관련 인터페이스가 OSPF 영역 0에 추가됩니다. HSRP의 경우 우선순위 값이 설정됩니다. 첫 번째 디스트리뷰션 스위치에 255, 두 번째 디스트리뷰션 스위치에 250을 제공하여 적절한 페일오버를 보장합니다. 또한 보안을 강화하기 위해 키 체인(HSRP_KEY)을 사용하여 HSRP 인증을 구성합니다.

컨피그레이션을 깔끔하고 관리 가능하게 유지하기 위해 일부 값은 하드코딩됩니다. 예를 들어, 서브넷 마스크(255.255.240.0)와 특정 HSRP 설정(예: 버전 및 BFD)은 모든 브랜치에서 동일하기 때문에 변수 수가 줄어듭니다. 이렇게 하면 구성이 더 간단해지고, 적용이 쉬워지며, 실수가 발생할 가능성이 줄어듭니다. 마지막으로, 업링크 인터페이스는 IP로 구성되고 스위치 간의 적절한 라우팅을 위해 OSPF 영역 0에 추가됩니다. 이러한 접근 방식을 통해 구성 프로세스를 더 쉽게 관리하고 오류 발생 가능성을 줄이는 동시에 여러 지사에 대해 유연하게 대처할 수 있습니다.

이제 디스트리뷰션 레이어에서 세그멘테이션을 제공하는 "디스트리뷰션 ACL 컨피그레이션" 모듈을 검토합니다.

```
{% macro Distribution_ACL_Config (branch_number) %}
!
ip access-list extended BLOCK_BRANCH
 deny ip 172.17.{{ 16 * (branch_number - 1) }}.0 0.0.15.255 172.16.{{ 16 * (branch_number - 1) }}.0 0.0.15.255
 deny ip any host 239.255.255.250
 permit ip any any
!
interface vlan {{ 100 * branch_number + 11 }}
 ip access-group BLOCK_BRANCH in
!
{% endmacro %}
```

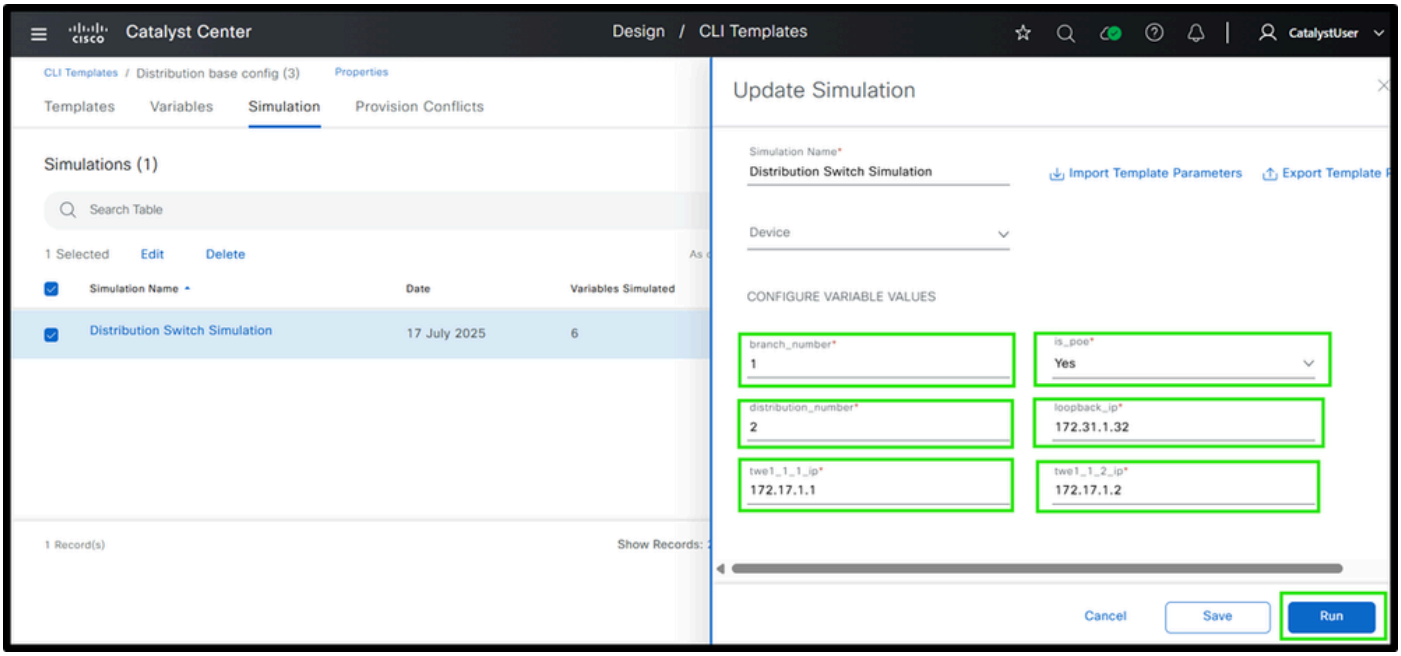
배포 ACL 구성

이 모듈에서는 Jinja2 템플릿을 사용하여 디스트리뷰션 레이어에서 세그멘테이션을 보여 줍니다. 또한 branch_number 변수를 사용하여 동적으로 서브넷 주소를 계산하므로 자동화된 확장 가능한 ACL 컨피그레이션이 가능합니다. 각 브랜치에 대해 ACL은 이러한 범위 간의 IP 트래픽을 거부하여 서브넷 1(172.17.X.0)과 서브넷 2(172.16.X.0) 간의 통신을 차단합니다. 또한 멀티캐스트 주소 239.255.255.250에 대한 트래픽은 거부하고, 다른 모든 트래픽은 허용합니다. VLAN 인터페이스는 브랜치 번호에 따라 동적으로 할당되며 ACL은 해당 인터페이스에서 인바운드로 적용됩니다. 이 자동화된 접근 방식은 효과적인 지점별 세그멘테이션을 보장하고 수동 컨피그레이션 오류를 줄이며

네트워크 정책 시행을 간소화합니다.

마지막으로 마지막 모듈인 "Distribution Standard Configuration(배포 표준 컨피그레이션)"은 Access Standard Configuration(액세스 표준 컨피그레이션) 모듈에 설명된 것과 거의 동일합니다(자세한 내용은 해당 섹션 참조). 여기에는 모범 사례, 보안 강화, 보안 장치 관리를 위한 주요 기능이 포함됩니다. 유일한 차이점은 소스 인터페이스에 있습니다. 액세스 스위치 템플릿에서는 VLAN {{ branch_number * 100 + 13 }}(으)로 정의되지만, 분산 스위치 컨피그레이션에서는 루프백0(으)로 하드코딩할 수 있습니다.

3단계: 구성을 배포하기 전에 시뮬레이션을 수행합니다.



(1) 분산 스위치 템플릿 시뮬레이션 입력 및 출력

(2) 분산 스위치 템플릿 시뮬레이션 입력 및 출력

(3) 분산 스위치 템플릿 시뮬레이션 입력 및 출력

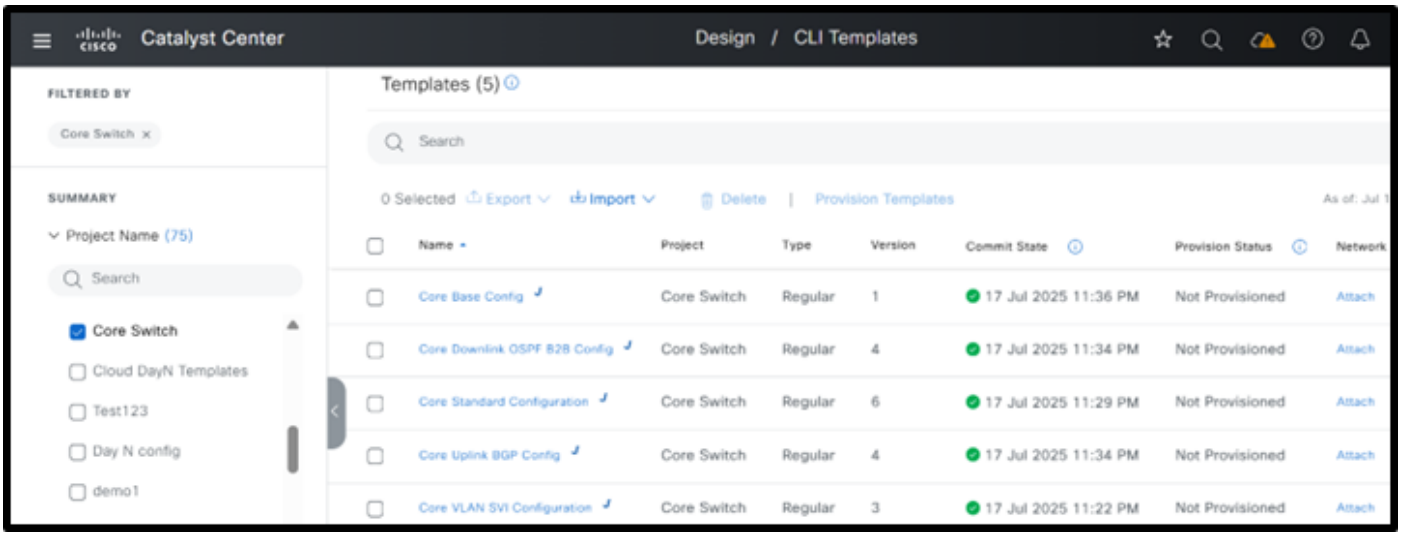
(4) 분산 스위치 템플릿 시뮬레이션 입력 및 출력

이는 템플릿을 디스트리뷰션 레이어에서 사용하여 컨피그레이션을 생성하는 방법입니다. 이제 코어 레이어 장치를 살펴보고 템플레이팅을 적용하는 방법을 알아보겠습니다.

코어 레이어 스위치

이제 코어 스위치용 모듈형 템플릿을 설계합니다. 기본 템플릿과 해당 모듈은 Cisco Catalyst Center의 'Core Switch' 프로젝트의 일부입니다. 1단계의 기본 템플릿을 참조하십시오.

1단계: 다양한 코어 스위치 구조 정의



코어 스위치 템플릿 구조

2단계: 다양한 모듈 정의

```
{% include "Core Switch/Core VLAN SVI Configuration" %}
{% include "Core Switch/Core Downlink OSPF B2B Config" %}
{% include "Core Switch/Core Uplink BGP Config" %}
{% include "Core Switch/Core Standard Configuration" %}

{{ Core_VLAN_SVI_Configuration () }}
{{ Core_Downlink_OSPF_B2B_Config () }}
{{ Core_Uplink_BGP_Config () }}
{{ Core_Standard_Config () }}
```

코어 기본 구성

대부분의 코어 스위치 컨피그레이션은 모든 브랜치에서 유사하므로 공통 값을 하드코딩할 수 있습니다. 일반적으로 IP 주소만 변경되며, 변수를 사용하여 이를 설정할 수 있습니다. 각 브랜치에는 일반적으로 두 개의 코어 스위치만 있기 때문에 이러한 변수를 쉽게 관리할 수 있습니다. 일부 브랜치에 더 많은 코어 스위치가 있다고 해도 액세스 또는 디스트리뷰션 스위치 수보다 적은 수가 있습니다. 따라서 액세스 및 배포 스위치의 경우 변수가 너무 많이 사용되고 변수가 너무 많으면 구성에 많은 시간이 소요될 수 있으므로, 모범 사례로서 변수를 최소화하는 것이 더 중요합니다.

이제 첫 번째 모듈부터 시작합니다. "코어 VLAN SVI 컨피그레이션" 이 예에서 코어 스위치는 방화벽 뒤에 위치하며 방화벽과 eBGP 피어링을 설정해야 합니다. 이 모듈은 eBGP 피어링 및 OSPF 인접 디바이스에 필요한 VLAN 및 해당 SVI를 생성합니다. 방화벽은 액티브/스탠바이 컨피그레이션에서 작동하는 것으로 간주됩니다.

```
{% macro Core_VLAN_SVI_Configuration () %}
!
vlan 2001
 name eBGP_peering_to_FW
!
vlan 2002
 name OSPF_neighborship
!
interface vlan 2001
 description eBGP Peering to Firewall
 ip address {{ VLAN2001_IP }} 255.255.255.248
 bfd interval 100 min_rx 100 multiplier 3
 no ip redirects
 no ip unreachables
 no ip proxy-arp
!
interface vlan 2002
 description OSPF neighborship to Core SW 2
 ip address {{ VLAN2002_IP }} 255.255.255.248
 bfd interval 100 min_rx 100 multiplier 3
 ip ospf 1 a 0
 no ip redirects
 no ip unreachables
 no ip proxy-arp
!
{% endmacro %}
```

코어 VLAN SVI 컨피그레이션

이 모듈은 앞서 설명한 대로 OSPF 및 BGP 네이버 관계를 설정하는 데 필요한 VLAN 및 관련 SVI를 생성합니다. SVI IP 주소를 제외한 모든 매개변수는 하드코딩됩니다. IP 주소 지정 계획과 일치하는 경우 서브넷 마스크도 포함됩니다. 이 방법은 변수를 제한하고 구성 오류의 가능성을 줄이는 데 도움이 됩니다.

이제 "Core Downlink OSPF B2B Config" 모듈을 살펴보겠습니다. 이 모듈은 다운링크 인터페이스, OSPF 및 Core Switch 1과 Core Switch 2 간의 백투백 링크에 대한 컨피그레이션을 생성합니다.

```
{% macro Core_Downlink_OSPF_B2B_Config () %}
!
interface loopback0
ip address {{ loopback_ip }} 255.255.255.255
!
router ospf 1
router-id {{ loopback_ip }}
default-information originate
!
! downlink interfaces
interface TWE1/0/1
 ip address {{ twe1_0_1_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
interface TWE1/0/2
 ip address {{ twe1_0_2_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
interface TWE1/0/24
 description Towards_Core_SW
 switchport mode trunk
 switchport trunk allowed vlan 2001,2002
 channel-group 10 mode active
 spanning-tree portfast trunk
 no shutdown
!
interface TWE1/0/48
 description Towards_Core_SW
 switchport mode trunk
 switchport trunk allowed vlan 2001,2002
 channel-group 10 mode active
 spanning-tree portfast trunk
 no shutdown
!
interface Port-channel10
 description Towards_Core_SW
 switchport mode trunk
 switchport trunk allowed vlan 2001,2002
 spanning-tree portfast trunk
 no shutdown
!
{% endmacro %}
```

코어 다운링크 OSPF B2B 컨피그레이션

이전 모듈과 유사하게, 이 모듈의 대부분의 값들은 변수들의 수를 최소화하도록 하드코딩된다. 루프백 및 다운링크 인터페이스의 IP 주소만 가변적입니다. 또한 백투백 포트 채널 및 VLAN은 여러 브랜치에 걸쳐 표준화됩니다.

이제 BGP 컨피그레이션을 생성하고 방화벽에 연결된 업링크를 관리하는 "Core Uplink BGP

Config"모듈을 살펴보겠습니다.

```
{% macro Core_Uplink_BGP_Config () %}
!
router bgp {{ AS_Number }}
bgp log-neighbor-changes
bgp router-id interface Loopback0
bgp graceful-restart
!
! eBGP Peering with Firewall
neighbor {{ BGP_NEIGHBOR }} remote-as {{ REMOTE_AS }}
neighbor {{ BGP_NEIGHBOR }} description eBGP Peering with Firewall
neighbor {{ BGP_NEIGHBOR }} update-source vlan 2001
neighbor {{ BGP_NEIGHBOR }} fall-over bfd
aggregate-address {{ AGGREGATE_IP }} {{ AGGREGATE_MASK }} summary-only
!
address-family ipv4
network {{ loopback_ip }} mask 255.255.255.255
neighbor {{ BGP_NEIGHBOR }} activate
exit-address-family
!
! Redistribute OSPF into BGP
redistribute ospf
!
! Uplink intefaces
interface TWE1/0/23
 description Towards_Firewall
 switchport mode access
 switchport access vlan 2001
 channel-group 10 mode active
 spanning-tree portfast
 no shutdown
!
interface TWE1/0/47
 description Towards_Firewall
 switchport mode access
 switchport access vlan 2001
 channel-group 10 mode active
 spanning-tree portfast
 no shutdown
!
interface Port-channel10
 description Towards_Firewall
 switchport mode access
 switchport access vlan 2001
 spanning-tree portfast
 no shutdown
!
{% endmacro %}
```

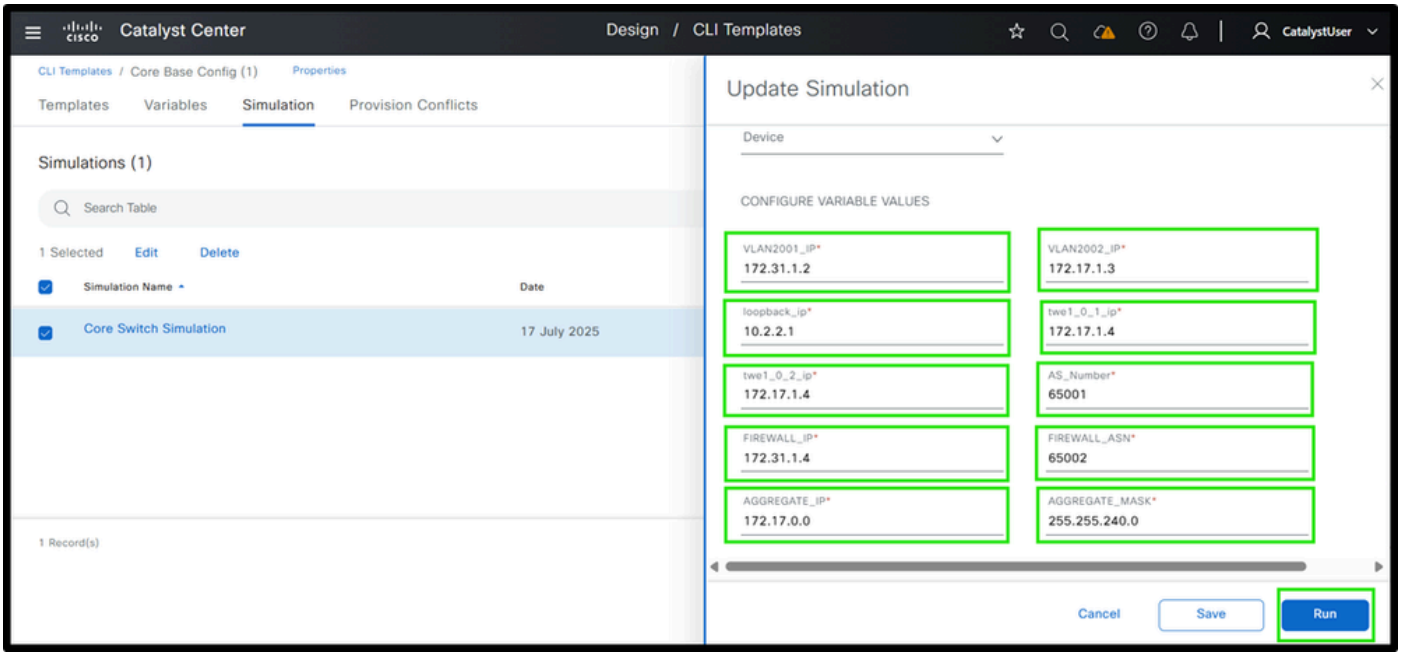
코어 업링크 BGP 컨피그레이션

이 모듈은 방화벽과의 eBGP 네이버 관계를 설정하는 데 필요한 BGP 컨피그레이션을 생성합니다. 위에 표시된 것처럼 대부분의 값은 서로 다른 브랜치에서 일관성을 유지하기 때문에 하드코딩됩니다. IP 주소와 AS 번호만 입력 변수로 가져와 필요한 컨피그레이션을 생성하는 데 사용됩니다. 대부

분의 다른 설정은 변수의 수를 최소화하기 위해 표준화되었습니다. 방화벽에 연결된 업링크 인터페이스는 이전 모듈에서 생성한 eBGP 네이버십에 사용되는 VLAN과 함께 지정됩니다.

마지막으로 마지막 모듈인 "Core Standard Configuration"은 Access Standard Configuration에 설명된 것과 거의 동일합니다(자세한 내용은 해당 섹션 참조). 여기에는 모범 사례, 보안 강화, 보안 장치 관리를 위한 주요 기능이 포함됩니다. 디스트리뷰션 스위치 컨피그레이션과 일관되게 이 모듈에서도 소스 인터페이스를 루프백0으로 설정할 수 있으며, 이 값을 하드코딩할 수 있습니다.

3단계: 시뮬레이션 수행

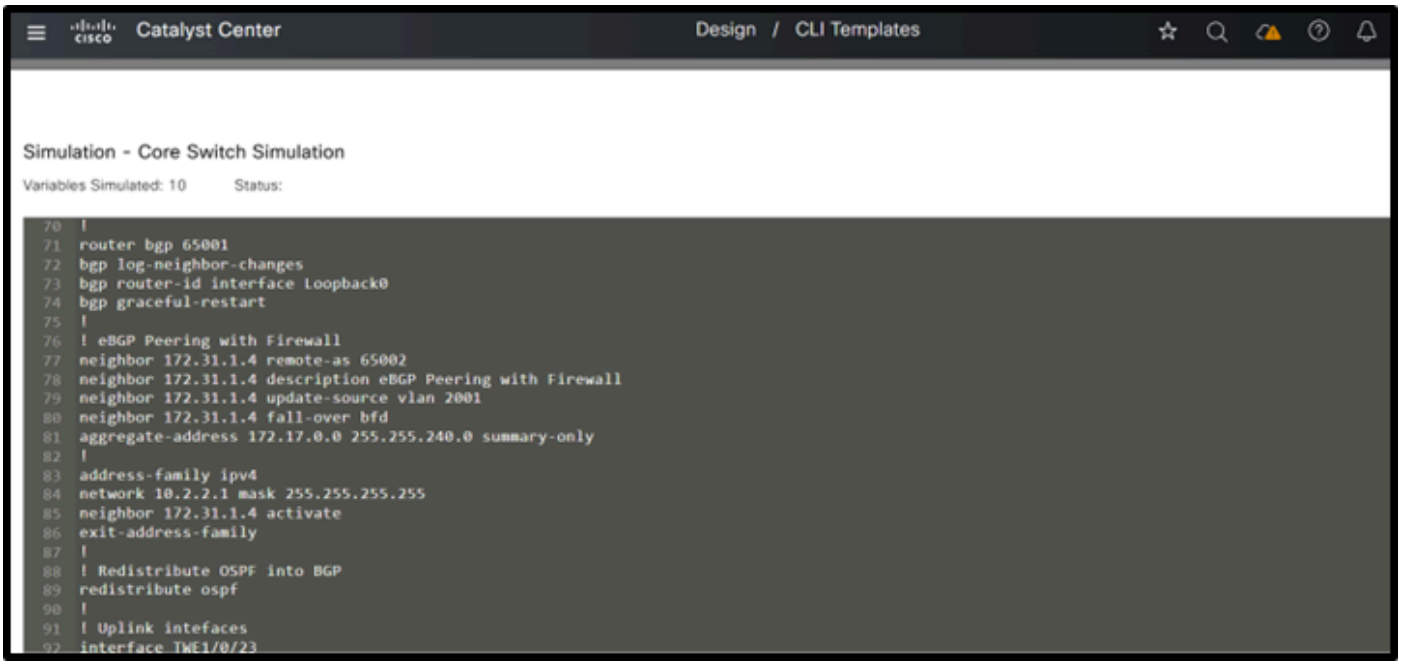


(1) 코어 스위치 템플릿 시뮬레이션 입력 및 출력

(2) 코어 스위치 템플릿 시뮬레이션 입력 및 출력

(3) 코어 스위치 템플릿 시뮬레이션 입력 및 출력

(4) 코어 스위치 템플릿 시뮬레이션 입력 및 출력



(5) 코어 스위치 템플릿 시뮬레이션 입력 및 출력

각 모듈의 구조와 컨피그레이션을 간략하게 살펴보면서 3-Tier 아키텍처에 대한 템플릿 설계에 대한 자세한 설명을 마치겠습니다.

이 모든 모듈은 앞서 설명한 모범 사례를 활용합니다.

참고: 축소된 핵심 아키텍처에 대한 템플릿을 설계할 때는 3-Tier 아키텍처에 대해 제공된 설명을 참조하십시오. 템플릿 구조는 동일하게 유지됩니다. 그러나 이전에는 코어 레이어와 디스트리뷰션 레이어에서 개별적으로 구현되었던 기능이 축소된 코어 레이어에서 결합되었습니다. 기본 템플릿을 생성하고 그 안의 관련 모듈을 참조하여 동일한 모듈식 템플릿 접근 방식을 사용할 수도 있습니다.

요약

기존 3-tier 캠퍼스 아키텍처는 코어, 디스트리뷰션 및 액세스 레이어 전반에서 광범위한 수동 구성에 의존하는 경우가 많습니다. 이러한 접근 방식은 시간이 많이 소요될 뿐만 아니라 사람의 실수가 일어나기 쉽다. 자동화와 중앙 집중식 관리가 없으면 운영 오버헤드가 크게 증가하므로 현대적이고 동적인 캠퍼스 네트워크를 효과적으로 확장하고 관리하기가 어려워집니다. Catalyst Center CLI 템플릿 기능 구성을 통해 기존 LAN 네트워크에 대해 자동화할 수 있습니다. 장치를 프로비저닝할 때 모듈식 접근 방식을 활용하는 것이 중요합니다. 모듈들은 상이한 층들에서 사용되는 다양한 특징들에 기초할 수 있다. 마지막으로 이 모든 모듈을 기본 모듈에 바인딩합니다.

실천 방안

이 백서에 제시된 모듈형 템플릿 방법론을 채택하여 스위치 구성을 표준화하고 3계층 및 축소된 핵심 아키텍처 전반의 네트워크 운영을 최적화하는 모범 사례를 채택하도록 조직을 초대합니다.

- 네트워크 팀은 모듈형 템플릿을 구현하여 다음과 같은 효과를 거둘 수 있습니다.
- 일관되고 반복 가능한 구성 방식을 통해 운영 효율성을 개선합니다.
- 인적 오류를 최소화하고 문제 해결 시간을 단축합니다.
- 확장성이 향상되어 성장과 비즈니스 요구 사항의 진화를 지원합니다.
- 다양한 환경 전반에 걸쳐 구성 일관성을 보장합니다.

이 접근 방식은 일상적인 관리를 간소화할 뿐만 아니라 더 신속한 구축을 지원하고, 업데이트 주기를 간소화하며, 보안 및 규정 준수 요구 사항에 더 잘 부합합니다. 모듈식 템플릿을 채택하면 끊임없이 변화하는 IT 환경에서 민첩성, 탄력성 및 장기적인 성공을 거둘 수 있는 네트워크가 구축됩니다.

실제 데모를 보려면 템플릿에 대해 자세히 알아보려면 YouTube 시리즈를 참조하십시오.

1 Catalyst Center에서 템플릿을 생성하고 관리하는 방법

https://youtu.be/SyUqEEcwye0

2 Catalyst Center의 CLI 템플릿에서 시스템 바인딩 변수를 사용하는 방법

https://youtu.be/gV1QBuHYJdo

작성자

Naveen Kumar, Customer Delivery Architect, Cisco Customer Experience

Risabh Mishra, 컨설팅 엔지니어, Cisco Customer Experience

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.