Entender modelos no Catalyst Center

Introdução

Este documento descreve o Cisco Catalyst Center e a experiência com modelos de configuração para arquiteturas de campus de núcleo triplo ou recolhido.

Informações de Apoio

Este documento destina-se a profissionais de empresas com conhecimento básico do Cisco Catalyst Center e experiência com modelos de configuração. Ele é particularmente relevante para aqueles que trabalharam ou planejam trabalhar com arquiteturas de campus de núcleo triplo ou recolhido.

O principal objetivo é ajudar os leitores a implementar e automatizar soluções de configuração e gerenciamento usando modelos no Cisco Catalyst Center. Apresentando insights avançados, técnicas práticas e exemplos reais, este documento serve como um recurso prático para aqueles que procuram aprimorar suas habilidades de infraestrutura de LAN e otimizar os fluxos de trabalho por meio da automação e do gerenciamento baseado em modelos.

Resumo executivo

À medida que as redes corporativas continuam a evoluir, a necessidade de gerenciamento escalável, consistente e automatizado nunca foi tão grande. O Cisco Catalyst Center oferece uma plataforma centralizada, baseada em intenção, que simplifica a configuração, o provisionamento e a garantia nas redes do campus. Este white paper explora como os profissionais de rede podem aproveitar os recursos de automação e o Editor de modelos de CLI do Cisco Catalyst Center para simplificar as operações de rede, reduzir os erros de configuração e acelerar as implantações em arquiteturas de núcleo agrupadas e de três camadas. Ele detalha as melhores práticas para projetar modelos modulares baseados em Jinja2, integrando automação em fluxos de trabalho de dia 0 e dia N, e alcançar consistência operacional nas camadas de Núcleo, Distribuição e Acesso. Ao adotar as estratégias descritas neste documento, você pode transformar o gerenciamento de rede manual tradicional em um modelo ágil, padronizado e orientado à automação, alinhado à visão de rede baseada em intenções da Cisco.

Desafios das redes de campus

À medida que as redes de campus evoluem para atender às demandas das empresas modernas, elas enfrentam vários desafios importantes:

2a. Complexidade no gerenciamento de rede

Muitas funções de rede ainda são gerenciadas manualmente, aumentando o risco de erro humano. Isso não apenas aumenta os esforços de manutenção, mas também sobrecarrega os recursos de TI, especialmente com orçamentos estáticos ou limitados.

2-B. Desafios de implantação e automação

A integração de novos dispositivos para redes com e sem fio geralmente é demorada e complexa, levando a atrasos na implantação e aumento da sobrecarga administrativa.

2-C. Gerenciamento de imagens de software

Manter uma "imagem de ouro" consistente em toda a rede é um desafio. Muitas redes terminam com vários sistemas operacionais para dispositivos com e sem fio, levando a ineficiências e dificuldades de gerenciamento.

2-D. Configurações de rede inconsistentes

Variações nas configurações de rede podem resultar em problemas de conformidade e ineficiências operacionais, dificultando a manutenção de uma rede confiável e segura.
2-E. Expectativas crescentes dos usuários

Os usuários exigem conectividade ininterrupta e experiências de aplicativo perfeitas, independentemente de sua localização ou dispositivo. Atender a essas expectativas exige que as redes sejam resilientes, inteligentes e capazes de se adaptar a mudanças em tempo real.

Além desses desafios, as infraestruturas de LAN modernas enfrentam uma variedade de outras complexidades.

Simplificando as redes de campus com o Cisco Catalyst Center

O Cisco Catalyst Center é uma solução de gerenciamento de rede centralizada para redes de

campus, com suporte a sedes, filiais, conexões com e sem fio e ambientes de TI/TO. Ele oferece opções de implantação flexíveis, incluindo dispositivos físicos, servidores VMware ESXi ou nuvem AWS. Com recursos abrangentes, o Catalyst Center simplifica as operações, melhora o desempenho e reforça a segurança.

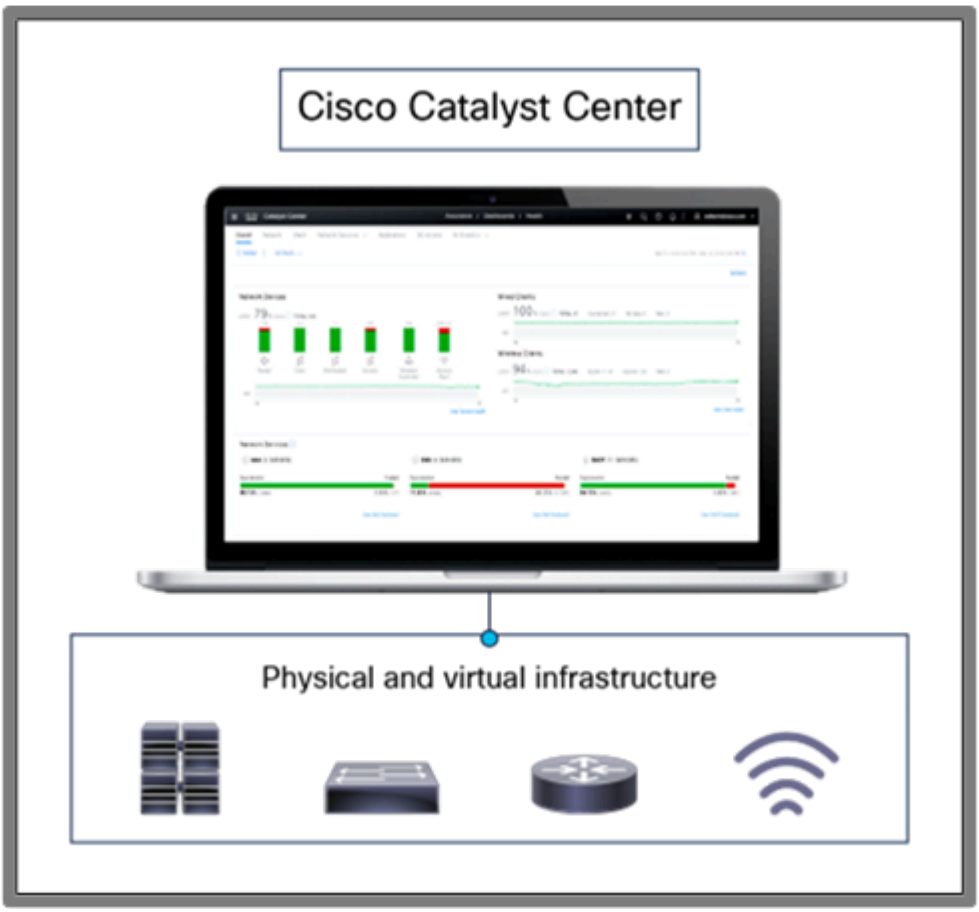


Figura 1: Gerenciando a infraestrutura com o Cisco Catalyst Center

Principais recursos e benefícios

O Cisco Catalyst Center (CC) fornece recursos avançados que otimizam o gerenciamento e a automação da rede:

Provisionamento automatizado (ZTP): Automatiza a integração de dispositivos, reduzindo o esforço manual e o tempo de implantação.

Gerenciamento de imagens de software (SWIM): Garante versões de software consistentes entre os dispositivos com verificações pré e pós-atualização para evitar problemas.

Automação baseada em intenção: Simplifica as implantações, convertendo a intenção da rede em configurações de dispositivos para redes com e sem fio.

Automação de LAN: Automatiza o roteamento e o endereçamento IP de Camada 3 para criar topologias fim-a-fim.

Automação de rede sem fio: Recursos como Plug and Play (PnP) permitem o rápido provisionamento de access points sem fio.

Gerenciamento de rede hierárquico: Permite perfis específicos do local (por exemplo, SSIDs, parâmetros de RF, VLANs) para implantações consistentes entre locais.

Modelos de CLI : O Catalyst Center Template Editor permite que os administradores criem e gerenciem facilmente modelos de configuração baseados em CLI, permitindo uma implantação consistente e eficiente entre dispositivos.

Garantia : A garantia permite o monitoramento centralizado de dispositivos gerenciados via CC.

Além desses recursos, o Cisco Catalyst Center oferece muitos outros recursos que estão além do escopo deste documento. Este documento concentra-se principalmente no projeto de modelos CLI usando o Catalyst Center.

Visão geral de alto nível da arquitetura do campus LAN com o Catalyst Center

As redes tradicionais de campus de LAN formam a espinha dorsal da conectividade empresarial, garantindo uma comunicação confiável e escalável para dispositivos com e sem fio. Essas redes são normalmente projetadas usando a arquitetura de 3 camadas ou a arquitetura de núcleo recolhido, dependendo do tamanho e da complexidade da organização.

Arquitetura de três camadas

A arquitetura de três camadas é um modelo de projeto de rede básico que consiste na camada central, camada de distribuição e camada de acesso. Essa arquitetura fornece escalabilidade, alto desempenho e gerenciamento de tráfego eficiente. Consulte a visão geral de cada camada.

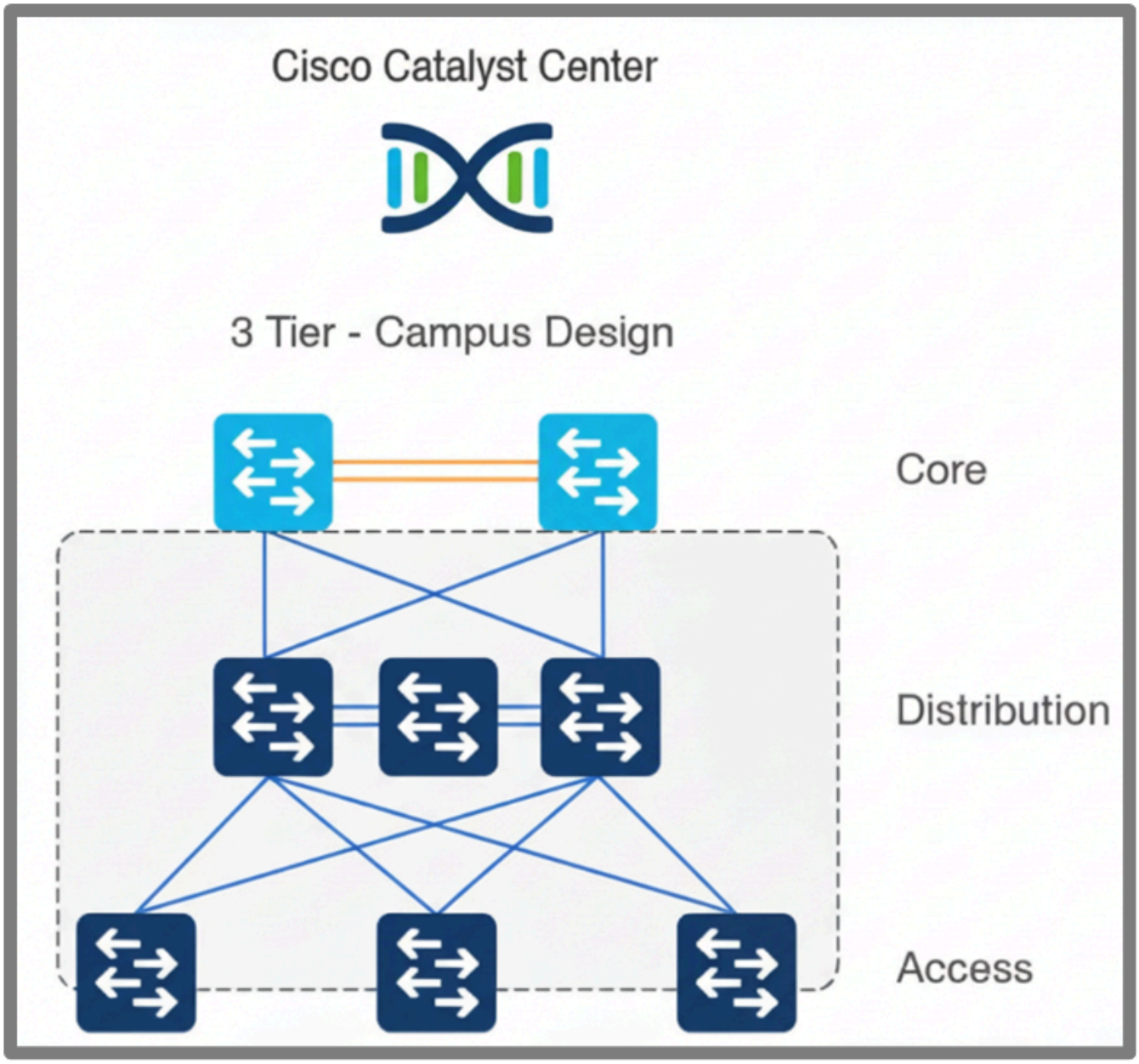


Figura 2: Arquitetura de campus de três camadas

Camada de núcleo

A camada central serve como o backbone da rede, fornecendo conectividade e escalabilidade de alta velocidade. As principais configurações incluem protocolos de roteamento ascendentes e descendentes (como OSPF e BGP), políticas de rota, configurações de interface de downlink e uplink, fortalecimento de segurança, etc.

Camada de distribuição

A camada de distribuição interliga as camadas de núcleo e de acesso, lidando com a agregação de tráfego, aplicação de políticas e redundância. As principais configurações incluem HSRP/VRRP para redundância, STP para prevenção de loop, VLANs de Camada 2 e

Camada 3, configurações de interface uplink e downlink, ACLs para segurança e fortalecimento de segurança.

Camada de acesso

A camada de acesso conecta terminais à rede, permitindo acesso seguro e confiável. As principais configurações incluem configuração de interface de acesso, configuração de interface de uplink, VLANs de Camada 2, ACLs para restringir o acesso ao dispositivo e proteção de segurança.

Arquitetura de núcleo recolhida

A arquitetura de núcleo recolhido combina as camadas de núcleo e de distribuição em uma única camada, reduzindo a complexidade e o custo enquanto mantém o desempenho e a escalabilidade. Essa abordagem é adequada para redes de pequeno a médio porte em que não é necessária uma camada central separada. Consulte a visão geral das camadas nesta arquitetura.

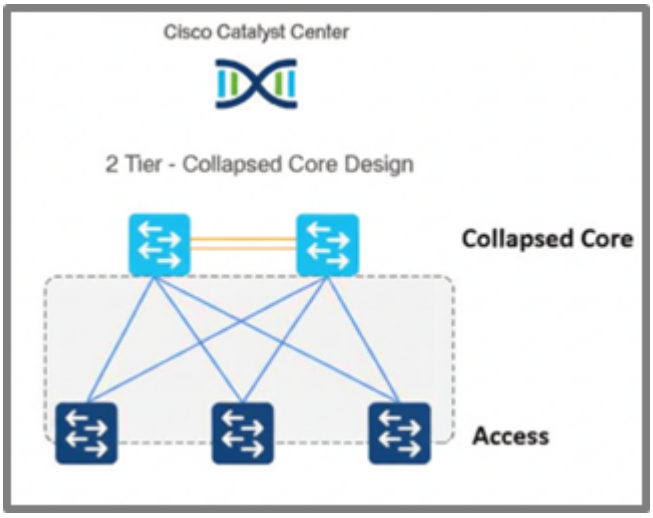


Figura 3: Arquitetura do núcleo recolhido do campus

Camada de núcleo recolhida

A camada central recolhida combina as funções das camadas central e de distribuição, fornecendo conectividade de backbone, agregação de tráfego e aplicação de políticas. As principais configurações incluem protocolos de roteamento ascendentes e descendentes (como OSPF e BGP), políticas de rota, configurações de interface de downlink e uplink, BFD para detecção de falhas, roteamento entre VLANs usando SVIs, HSRP/VRRP para redundância de gateway, STP para prevenção de loop e fortalecimento de segurança. Aproveitando os modelos no Cisco Catalyst Center, essas configurações podem ser automatizadas, garantindo implantações consistentes e eficientes.

Camada de acesso

Conforme descrito anteriormente, a camada de acesso conecta terminais à rede, permitindo acesso seguro e confiável. As principais configurações incluem configuração de interface de acesso, configuração de interface de uplink, VLANs de Camada 2, ACLs para restringir o acesso ao dispositivo e proteção de segurança.

Consideração do Design do Modelo

Esta seção descreve como projetar modelos no Cisco Catalyst Center para gerar configurações de dispositivos. O Editor de modelos agiliza o provisionamento, permitindo a criação de modelos CLI reutilizáveis e suportando a implantação dinâmica de configurações personalizadas para sua rede. O Catalyst Center oferece suporte a duas linguagens de modelo: Jinja2 e Velocity. Essas linguagens ajudam no gerenciamento de configuração para dispositivos.

Jinja é uma linguagem de modelagem popular e amigável para designers usada principalmente com Python para gerar conteúdo dinâmico como HTML, XML ou outros formatos baseados em texto. Ele permite incorporar variáveis e estruturas de controle (como loops e condicionais) dentro de modelos para criar saída dinâmica.

O Apache Velocity é um mecanismo de modelagem baseado em Java que usa a linguagem de modelo Velocity (VTL) para permitir conteúdo dinâmico em vários documentos, incluindo páginas da Web, XML ou até mesmo código-fonte. Ele mescla dados de objetos Java com modelos para produzir a saída final.

Este documento abrange apenas modelos Jinja2.

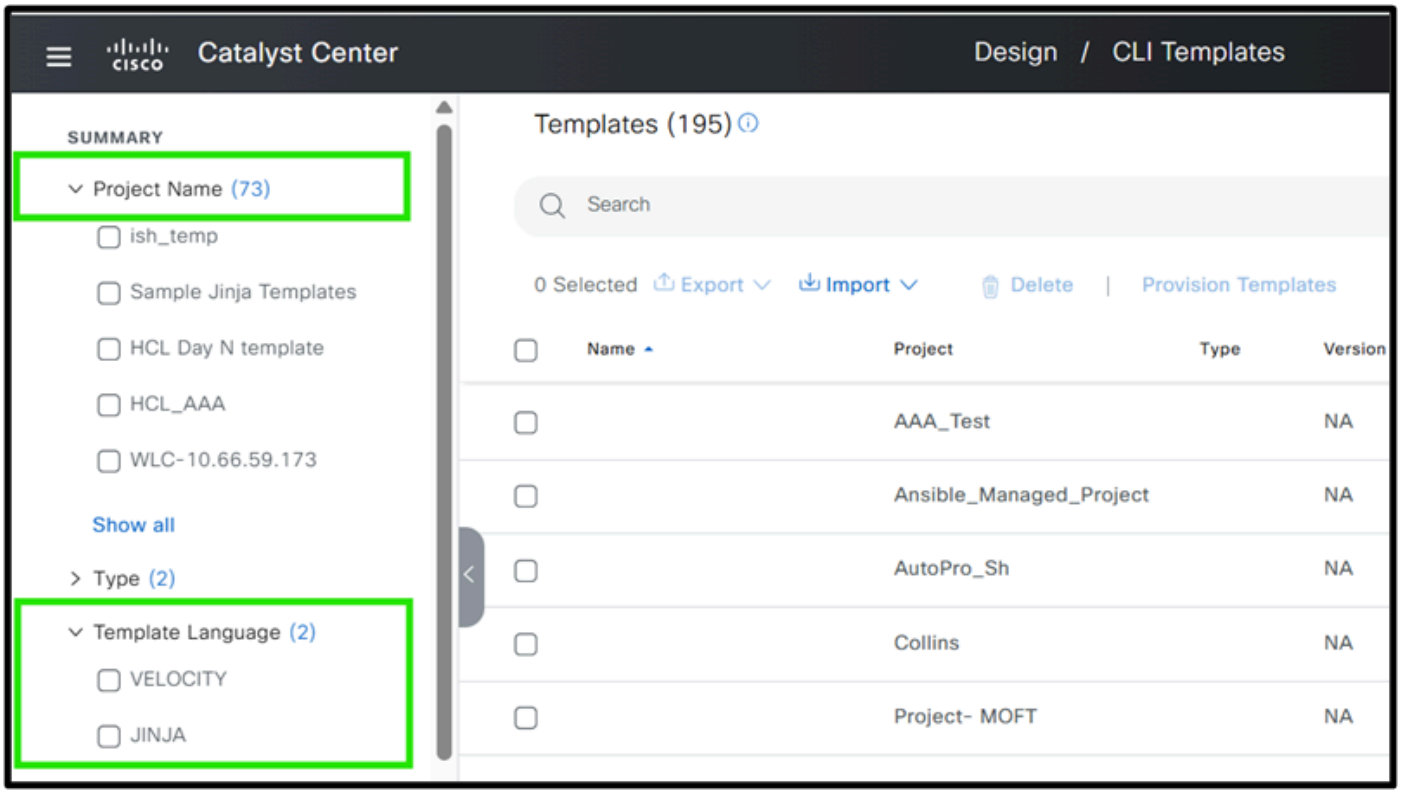


Figura 4: Editor de modelos do Cisco Catalyst Center

Neste documento, usamos Jinja2 devido à sua flexibilidade. Em vez de uma exploração aprofundada de Jinja2, o foco é na aplicação prática para o projeto de modelos. Para obter mais informações sobre a modelagem Jinja2 no Catalyst Center, consulte o link:

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

Antes de mergulhar nas estratégias de projeto de modelos para uma rede de campus da Cisco, é importante utilizar as principais práticas recomendadas para garantir a eficiência e a capacidade de gerenciamento ao trabalhar com modelos.

Estrutura do modelo e práticas recomendadas / Diretriz para a melhor estratégia

Ao automatizar a configuração de dispositivos de rede usando o Cisco Catalyst Center, é essencial adotar estratégias estruturadas e práticas recomendadas. Essas etapas ajudam a garantir a consistência, a escalabilidade e a facilidade de gerenciamento em toda a infraestrutura de rede.

- Dividir configuração por função de dispositivo

Comece categorizando os dispositivos de acordo com sua função na topologia de rede. As funções comuns incluem:

- Centro
- Distribuição
- Acesso

Exemplo: Um dispositivo que funciona como um switch central deve ter requisitos de configuração diferentes em comparação a um switch de acesso.

Compartimente a configuração em blocos modulares

Dentro da função de cada dispositivo, divida a configuração em blocos modulares agrupando recursos ou configurações semelhantes. Essa abordagem modular simplifica a automação, a solução de problemas e as atualizações futuras.

Exemplos de um dispositivo central:

- Bloco de Configuração do OSPF
- Bloco de configuração BGP
- Bloco de políticas de QoS

Identificar blocos de configuração independentes de função

Alguns blocos de configuração se aplicam universalmente em todas as funções de dispositivo. Identificar e padronizar esses blocos garante práticas recomendadas e consistência em toda a rede.

Blocos de configuração comuns independentes de função:

- Configuração básica: nome de host, banners de login
- Protocolos de gerenciamento: DHCP, DNS, NTP, SNMP

Políticas de acesso: configurações de segurança padrão

Esses blocos podem ser reutilizados para dispositivos Core, Distribution e Access, otimizando o processo de automação.

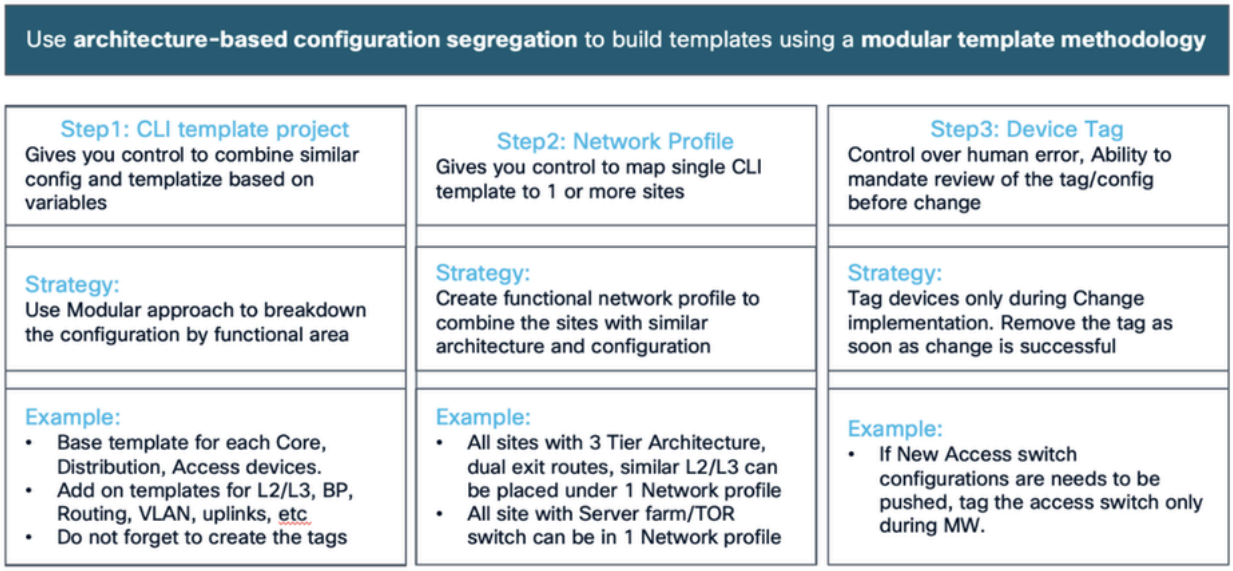


Figura 1: Prática recomendada com exemplo

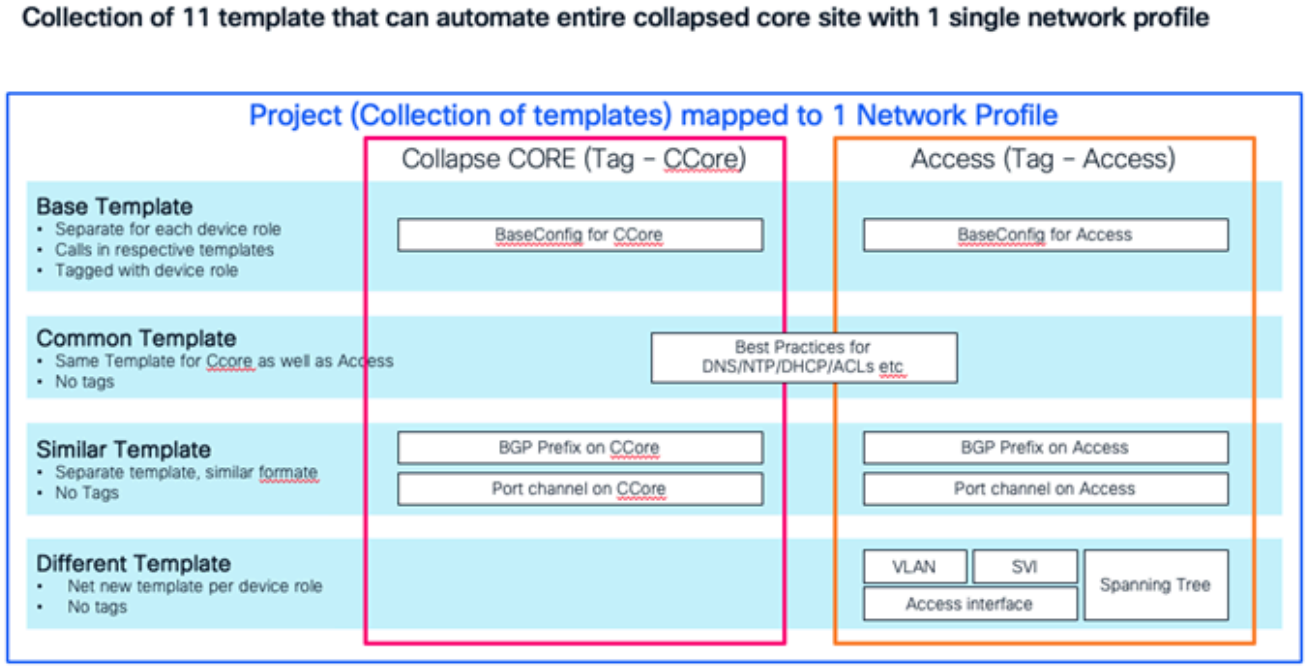


Figura 2: Exemplo de modelo de núcleo recolhido

Práticas recomendadas para trabalhar com modelos

Projeto de modelo modular para configuração automatizada

Ao automatizar as configurações do dispositivo no Cisco Catalyst Center, evite incorporar todas as configurações em um único modelo monolítico. Em vez disso, adote uma abordagem modular:

Crie um modelo base que faça referência a modelos (módulos) menores e específicos para a finalidade:

Divida a configuração em módulos lógicos (por exemplo, configurações de interface, protocolos de roteamento, recursos de segurança).

Essa estrutura torna as atualizações mais eficientes — as alterações em um módulo específico são refletidas automaticamente sempre que esse módulo é usado, reduzindo significativamente os erros e a complexidade.

Exemplo: Configuração modular para um dispositivo de filial

Suponha que você esteja automatizando a configuração de um dispositivo de filial.

Modelo base:

Inclui referências a modelos de módulo para as principais áreas de configuração.

Passa variáveis conforme necessário para cada módulo para personalização.

Modelos de módulo:

interface_settings: Gerencia configurações de interface.

protocolos de roteamento: Contém configurações de OSPF, EIGRP ou BGP.

recursos de segurança: Define ACLs, regras de firewall ou outras políticas de segurança.

```
{% 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) }}
```

Exemplo de estrutura de modelo base:

Com essa estrutura, qualquer alteração nas configurações de roteamento ou segurança só precisa ser feita em seus respectivos módulos, e essas alterações são refletidas instantaneamente sempre que o modelo base é usado. Isso torna suas configurações mais gerenciáveis e consistentes em todos os roteadores de filial.

Aqui o nome do projeto é Filial e outros 3 módulos diferentes são definidos no projeto. Todos eles são combinados no modelo base.

Minimizar Variáveis no Modelo

Mantenha o número de variáveis no modelo em um nível mínimo, para reduzir a complexidade e os erros. Menos variáveis simplificam a implantação, especialmente em redes grandes, tornando o processo mais eficiente e consistente.

Uso de marcas de dispositivo para modelos

Aproveite as tags de dispositivo no Cisco Catalyst Center, como local, função ou local, para criar modelos Jinja2 dinâmicos e escaláveis. Essas tags permitem a lógica condicional, garantindo que as configurações corretas sejam aplicadas aos dispositivos apropriados. Essa abordagem minimiza erros e simplifica o gerenciamento de modelos em diversos ambientes de rede.

Codificar Valores Estáticos Sempre Que Possível

Os valores estáticos de hardcode podem simplificar modelos e melhorar a eficiência da implantação. Exemplos comuns incluem endereços IP para servidores DNS, NTP ou Syslog, que normalmente permanecem consistentes entre os dispositivos. Da mesma forma, o uso de IDs de VLAN padrão em switches de acesso permite que esses valores sejam codificados, reduzindo a variabilidade e acelerando a implantação.

Adotar uma abordagem em duas fases: Modelos de Dia 0 e Dia N

Ao integrar dispositivos usando serviços como o Cisco Plug and Play, use uma estratégia de modelo em dois estágios:

Modelos do dia 0: Envie configurações básicas para garantir que o dispositivo possa se comunicar com o Cisco Catalyst Center.

Modelos do Dia N: Implante recursos e configurações avançados quando o dispositivo estiver acessível.

As práticas recomendadas permitem modelos eficientes e escaláveis que simplificam as implantações de rede do campus da Cisco.

Controle de espaço em branco em modelos de macros Jinja

Ao criar modelos usando a linguagem Jinja, é essencial lidar com espaços em branco e novas linhas cuidadosamente, especialmente ao renderizar conteúdo dinâmico dentro de macros. Espaços em branco acumulados ou novas linhas não intencionais podem levar a problemas de formatação na saída gerada, que devem causar erros de interpretação ou erros no processamento de downstream. Para resolver isso, Jinja fornece a sintaxe para controlar espaços em branco: colocando um sinal de menos (-) diretamente dentro dos delimitadores ({{- ... -}} ou {%- ... -%}), ele remove qualquer espaço em branco à esquerda ou à direita em torno da expressão. Por exemplo, substituir {{item[1]}} por {{- item[1] -}} garante que espaços extras ou novas linhas sejam removidos quando a macro for processada. Essa prática é especialmente útil ao percorrer listas ou gerar arquivos de configuração, como mostrado no snippet de modelo. Recomendamos sempre aplicar o controle de espaço em branco nesses cenários para manter saídas limpas e previsíveis.

Exemplo (uso recomendado):

```
{% para o item na lista_curinga %}
   {% se o item[0] == o prefixo -%}
      {{- item[1] -}}
   {%- endif %}
{%- endfor %}
```

Arquitetura de três camadas

Este whitepaper começa com o desenvolvimento de modelos para switches de acesso através de switches centrais e descreve os requisitos para cada camada.

Switches de camada de acesso

Os switches de acesso são integrados usando Plug and Play e devem exigir um modelo de Dia 0. Para obter mais informações sobre o processo Plug and Play no Catalyst Center, consulte o link :

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

Conforme discutido anteriormente, o Catalyst Center oferece suporte aos idiomas de modelo Velocity e Jinja2. Este documento utiliza Jinja2 para ilustrar a estrutura do modelo, devido à sua flexibilidade. A configuração do switch de camada de acesso pode ser implantada usando o modelo de Dia 0 e Dia N.

Um modelo básico de Dia 0 pode ser estruturado; consulte a Etapa 1:

Passo 1: Definir modelo

```
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
!
```

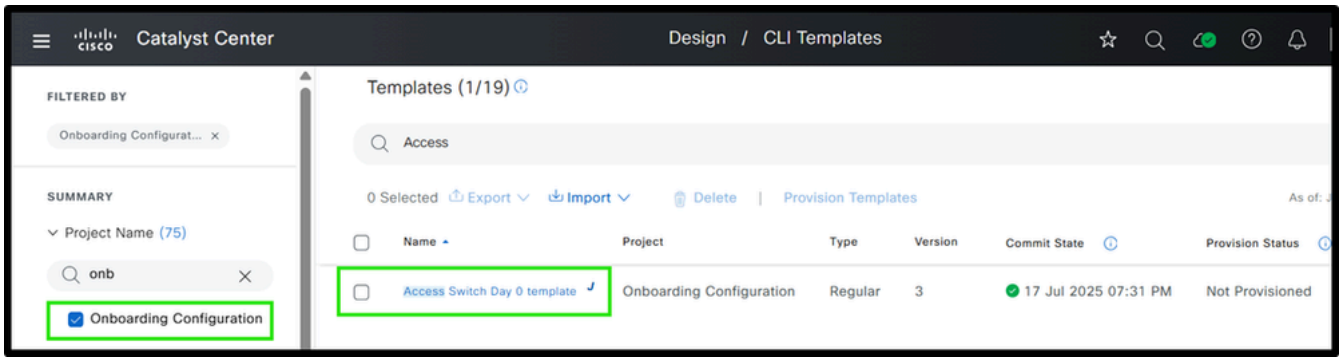
Passo 1: Definir modelo

O modelo simplifica a configuração codificando constantes como nome de usuário, senha, segredo de ativação e máscara de sub-rede, já que todos os switches em uma filial compartilham a mesma máscara de sub-rede VLAN de gerenciamento. O endereço IP de gerenciamento, no entanto, é exclusivo para cada switch e é definido como uma variável. Deve ser fornecida uma estrutura de modelo abrangente no modelo Dia N, que utiliza esse

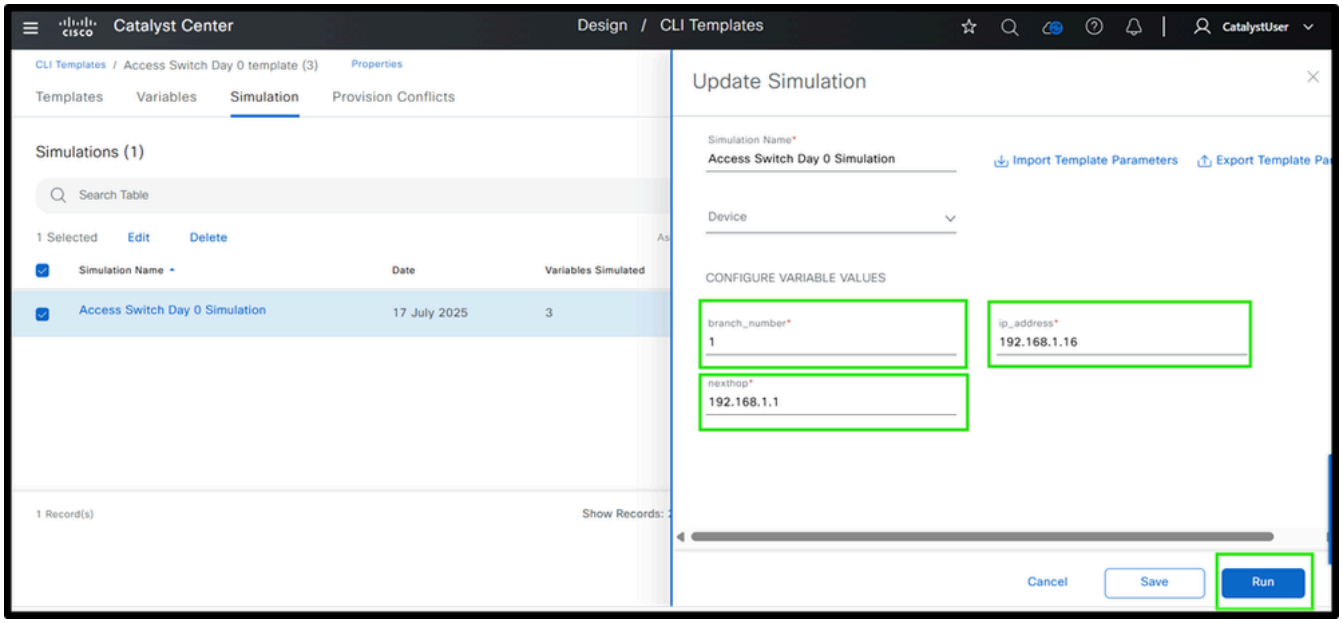
modelo Dia 0. No modelo do Dia N, cada recurso do switch de acesso é gerenciado por um módulo dedicado — por exemplo, um módulo lida com VLANs de Camada 2, módulos separados gerenciam interfaces de acesso de uplink e downlink, outro módulo se concentra no fortalecimento da segurança e assim por diante.

Embora as IDs de VLAN consistentes sejam preferenciais, as IDs variáveis podem ser geradas dinamicamente usando uma fórmula baseada no número da filial (por exemplo, Filial 1 = VLAN 113, Filial 2 = VLAN 213). Isso torna o modelo reutilizável entre filiais. Da mesma forma, o IP do próximo salto é uma variável, pois deve diferir por filial, dependendo do cluster de distribuição conectado.

Passo 2: Executar simulação e fornecer variável

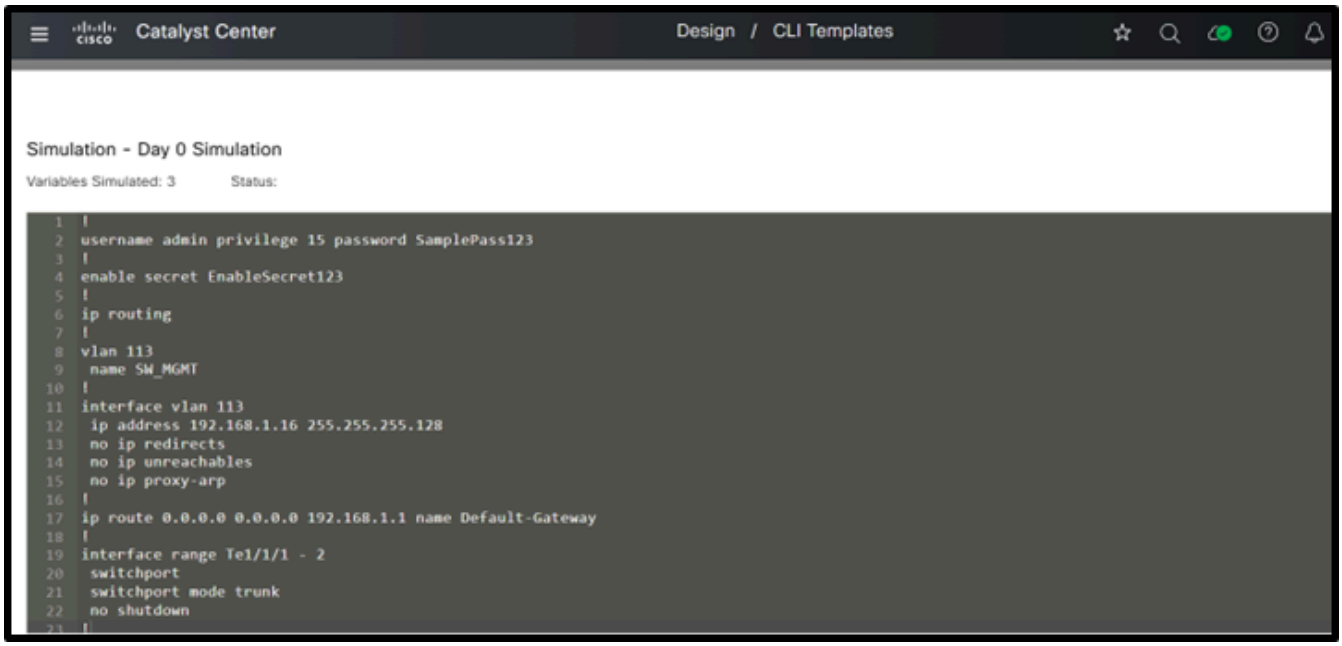


Estrutura de Modelo do Dia 0 do Switch de Acesso com Entradas e Saídas de Simulação



Ex.: Entradas de simulação

É sempre recomendável simular o modelo antes da disponibilização. A captura de tela mostra a configuração final após inserir as variáveis.



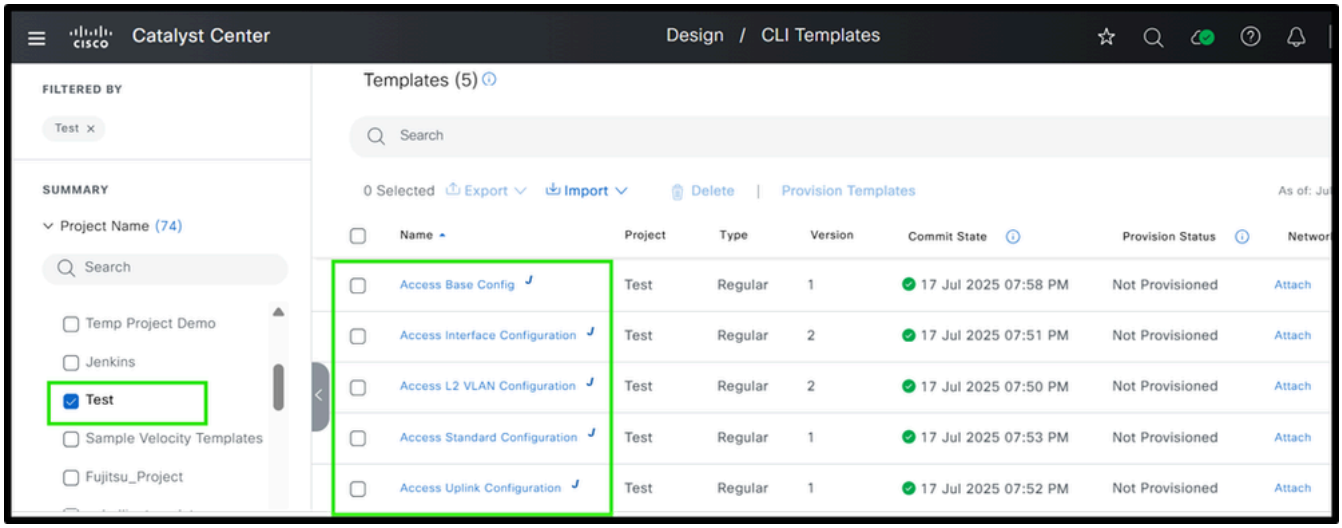
Configuração final após inserir valores

Agora, vamos ver como criar um modelo modular de Dia N.

A configuração do switch de acesso pode ser dividida em vários módulos, todos os quais podem ser combinados dentro de um módulo base. O modelo básico para switches de acesso é estruturado conforme mostrado.

Tanto o modelo básico quanto seus módulos são criados dentro de um projeto chamado "Teste" no Cisco Catalyst Center.

Passo 1: Definir vários modelos, incluindo o modelo Base



Estrutura do Modelo de Dia N do Switch de Acesso

Passo 2: Definir vários módulos

Configuração da base de acesso:

A captura de tela mostra um exemplo da configuração básica.

```
{% 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) }}
```

Configuração da base de acesso

Este modelo de configuração modular inclui quatro partes: Configuração de VLAN, configuração de interface de uplink, configuração de interface de acesso e configuração padrão. Ele usa apenas duas variáveis: branch_number e is_poe, mantendo-o simples e fácil de gerenciar.

Branch_number calcula IDs de VLAN específicos da filial, como mostrado no modelo Dia 0, e is_poe determina se o switch de acesso é um PoE ou não-PoE. Essas variáveis são fornecidas durante o provisionamento e passadas para os módulos para criar as configurações corretas, reduzindo o esforço e melhorando a eficiência.

Agora, vamos analisar cada módulo para ver como eles contribuem para gerar partes específicas da configuração geral.

Acessar a configuração de VLAN de L2

```
{% 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 %}
```

Acessar a configuração de VLAN de L2

Este módulo cria VLANs com base no número da filial, conforme explicado anteriormente. As VLANs de dados e voz são criadas em todos os switches, suportando ou não PoE. A VLAN de gerenciamento do AP (exemplo, 114 para a filial 1) será criada apenas se is_poe estiver definida como "Yes", o que significa que o switch suporta PoE. Se is_poe for "Não", a VLAN de gerenciamento do AP será ignorada, já que os switches não PoE não podem suportar pontos de acesso. Isso é gerenciado usando uma condição if.

```
{% 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 %}
```

Configuração da interface de acesso

Este módulo lida com a configuração da interface de acesso e utiliza a mesma abordagem que o switch PoE descrito anteriormente. Se a variável is_poe for "Yes", significando que o switch é um switch PoE, as primeiras seis portas (1-6) devem ser configuradas com a VLAN de gerenciamento do AP. Caso contrário, as primeiras seis portas devem ser definidas como portas de acesso do usuário.

Supondo que o switch seja um modelo de 24 portas, as portas restantes (7-24) sempre são configuradas como portas de acesso do usuário, independentemente de o switch ser PoE ou não.

O intervalo da interface foi padronizado e não é mais considerado como uma variável de entrada, o que é considerado uma prática recomendada para minimizar o número de variáveis no modelo. Além disso, o módulo inclui uma macro chamada common_access_settings, que minimiza o tamanho do modelo consolidando configurações repetidas. Essa macro é simplesmente chamada dentro das configurações da interface,

evitando a necessidade de especificá-las várias vezes.

Note: Este modelo aplica a mesma descrição a todas as interfaces de acesso. Se forem necessárias descrições exclusivas para cada interface, é recomendável enviá-las por push usando scripts Python separados ou ferramentas de automação semelhantes.

Reveja o módulo que gera configurações para interfaces de uplink.

```
{% 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 %}
```

Configuração de uplink de acesso

Este módulo gera a configuração para interfaces de uplink e manipula a remoção de VLAN. Se o switch suportar PoE, a VLAN de gerenciamento de AP é incluída na lista de VLANs permitidas; caso contrário, está excluído. Essa lógica é gerenciada usando a condição if no código, conforme descrito anteriormente.

Analise o módulo final, que demonstra configurações padrão, incluindo práticas recomendadas e fortalecimento da segurança.

Caution: Observe que isso é apenas para fins ilustrativos e não deve ser usado como referência para configurações de rede reais, pois as configurações podem variar com base em requisitos específicos

---

```
{% 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
```

Parte 1: Configuração padrão de acesso

```
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 %}
```

Parte 2: Configuração padrão de acesso

Este módulo gera uma configuração padrão que incorpora as melhores práticas, o fortalecimento da segurança e os principais recursos para o gerenciamento seguro de dispositivos. A maioria dos valores é codificada para consistência entre ramificações, exceto branch_number, que é usado para calcular a VLAN de gerenciamento para switches em cada ramificação e serve como interface de origem para várias configurações.

Passo 3: Faça uma simulação antes de configurar os switches. Somente a configuração base deve ser simulada.

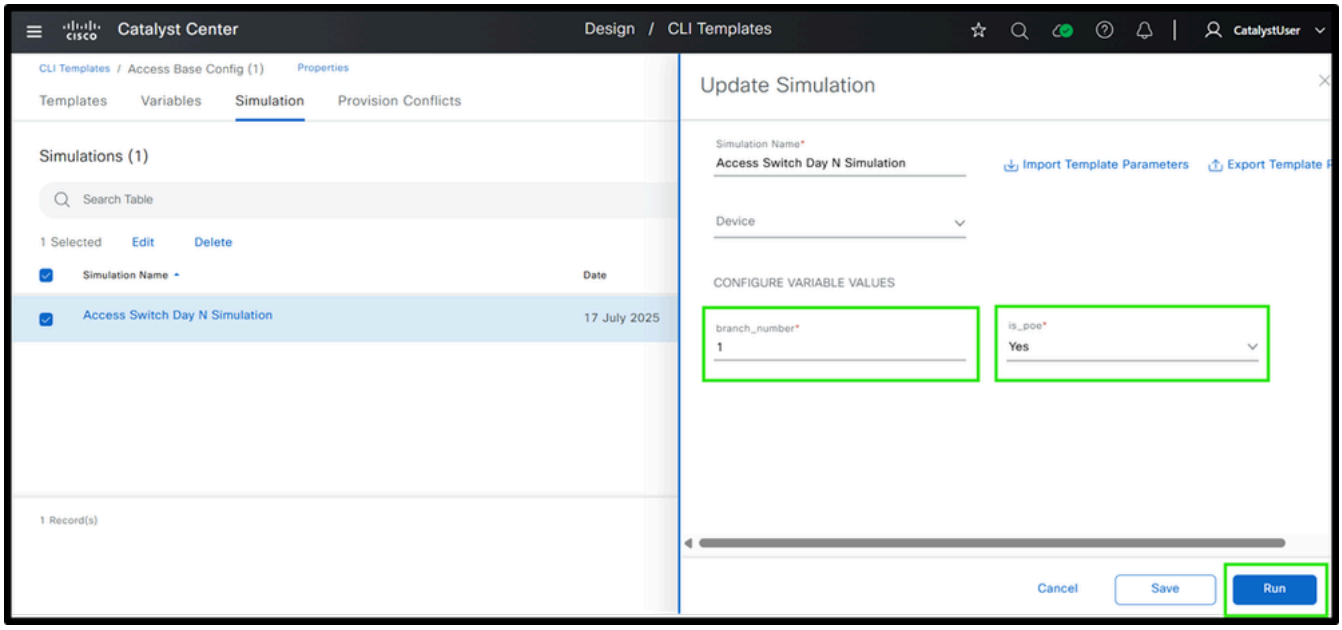
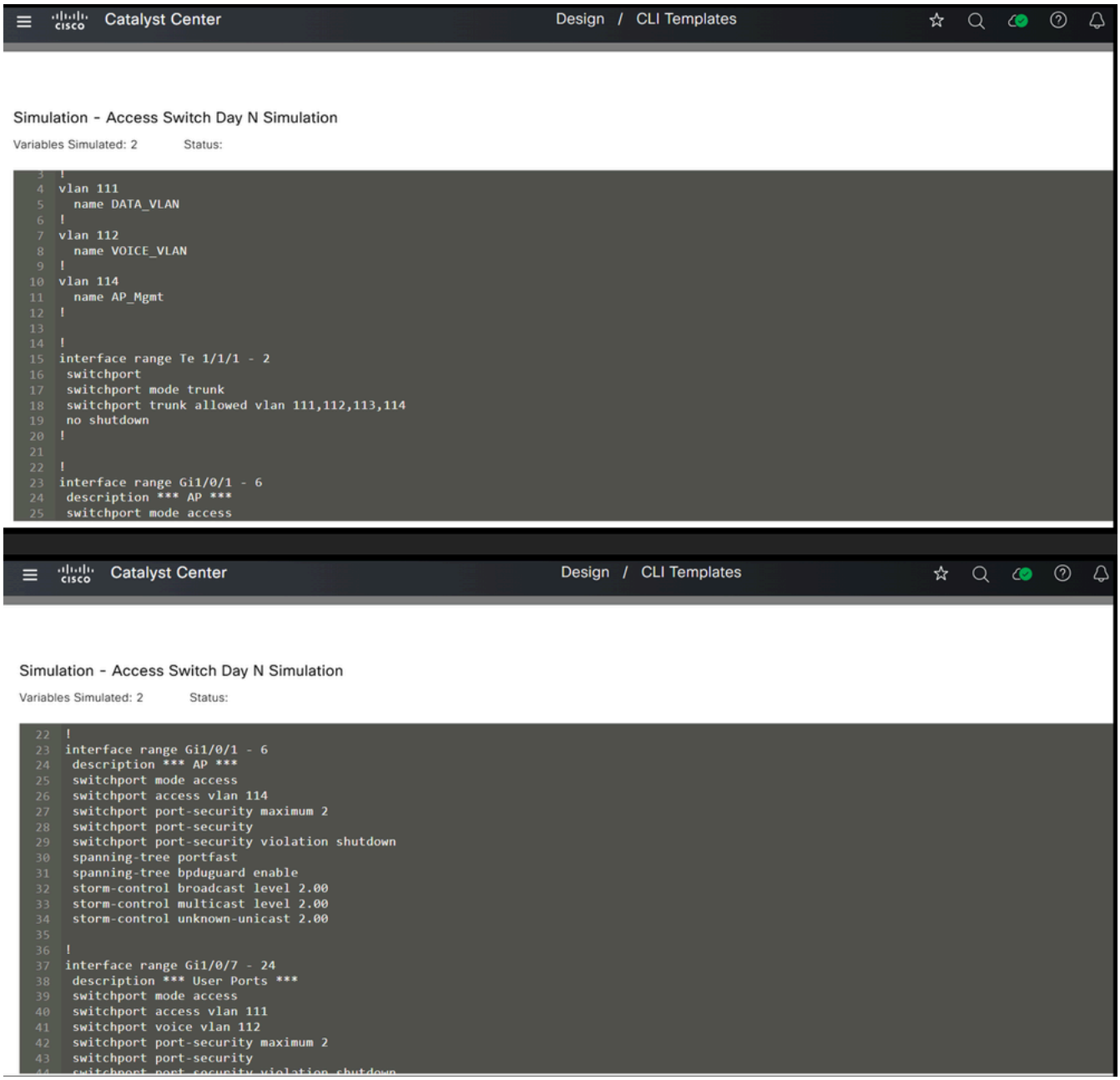


Figura 7: Entradas e Saídas de Simulação do Modelo N do Dia N do Switch de Acesso



Simulação

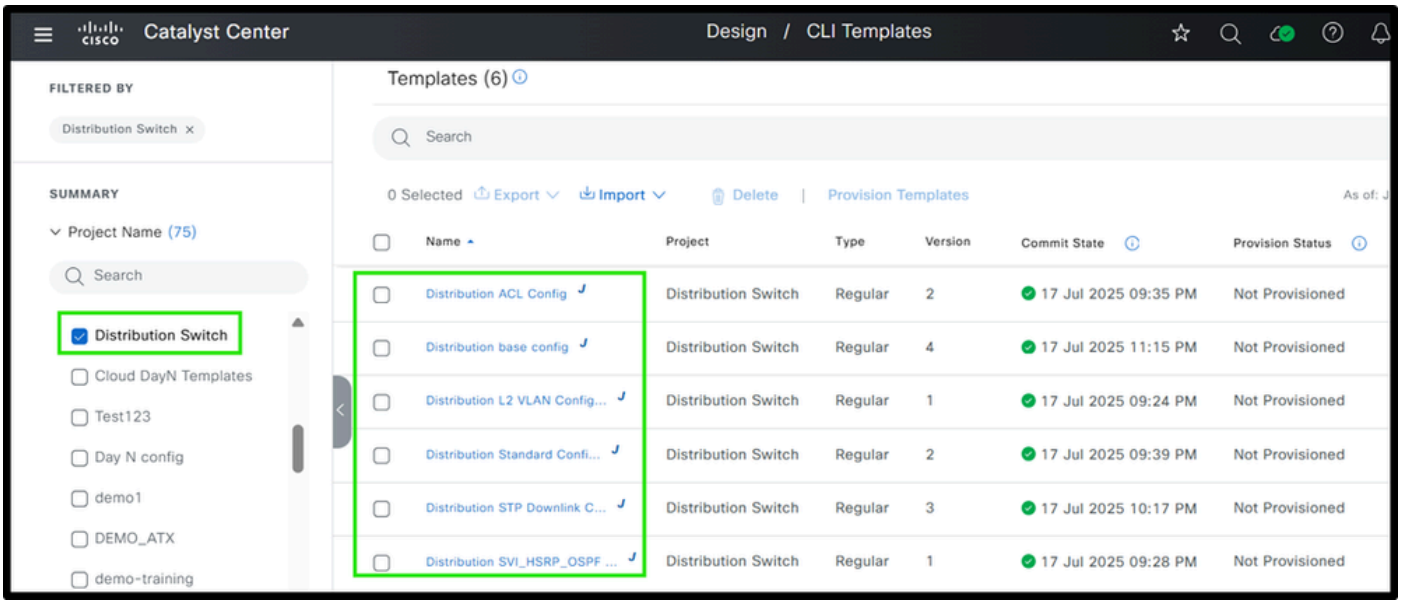
É assim que os modelos podem ser usados na camada de acesso para gerar configurações.

Agora, vamos dar uma olhada nos dispositivos da camada de distribuição para ver como a modelagem pode ser aplicada a eles.

Switches de camada de distribuição

Agora para projetar um modelo modular para switches de distribuição. O modelo base e seus módulos fazem parte do projeto 'Switch de distribuição' no Cisco Catalyst Center.

Passo 1: Estrutura do Modelo do Comutador de Distribuição



Ex.: Modelos de Distribuição

Etapa 2: Definir cada módulo

A configuração básica fornecida define cada módulo e todos são referenciados.

```
{% 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() }}
```

Ex.: Módulos de Modelo de Base de Distribuição

Semelhante aos switches de acesso, todos os modelos são criados dentro do projeto 'Switch de distribuição' e referenciados no modelo base. Embora alguns modelos sejam idênticos aos usados para switches de acesso, esta seção explica as diferenças específicas para switches de distribuição. O módulo "Configuração de VLAN de L2 de distribuição" é idêntico ao descrito anteriormente para switches de acesso. Verifique o módulo Access L2 VLAN Configuration que fornece essas informações. Gera as VLANs necessárias com base nos valores de entrada fornecidos para as variáveis.

Agora, reveja o módulo "Distribution STP Downlink Config", que lida com a geração de configurações de spanning tree e uplink para switches de distribuição.

```
{% 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 %}
```

Configuração de downlink de STP de distribuição

Aqui a funcionalidade macro Jinja2 está sendo usada , que é referenciada no módulo baseado. Essa estrutura ajuda na construção de uma abordagem modular.

Este módulo configura o Spanning Tree Protocol (STP) e interfaces de downlink com base no "branch_number" e se o switch está habilitado para PoE. A variável "branch_number" é usada para gerar VLANs de base exclusivas para cada filial, garantindo VLANs distintas, semelhantes à abordagem já destacada para switches de acesso. Se o switch for habilitado para PoE ("is_poe" == 'Yes'), uma VLAN adicional, como a VLAN de gerenciamento AP, será adicionada à lista. A variável "distribution_number" determina a prioridade de STP, definindo 4096 para Distribution 1 (tornando-a a bridge raiz preferencial) e 8192 para switches de distribuição secundários. Finalmente, as VLANs apropriadas são aplicadas à interface de tronco, garantindo que somente as VLANs relevantes sejam permitidas com base no fato de o switch estar ou não habilitado para PoE.

Agora, revise o módulo "Distribution SVI_HSRP_OSPF Config", que se concentra na configuração de SVIs, HSRP e OSPF para roteamento e redundância de rede eficientes.

```
{% 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 %}
```

Configuração de distribuição SVI_HSRP_OSPF

Este módulo, Distribution_SVI_HSRP_OSPF_Config, ajuda a configurar SVIs, HSRP, OSPF e interfaces de uplink para switches de distribuição. Neste exemplo, estamos nos concentrando no SVI para sub-redes de dados, mas o mesmo método pode ser usado para outros SVIs, como voz ou gerenciamento.

Se o planejamento de endereço IP para sub-redes de dados já estiver feito, os endereços IP poderão ser calculados automaticamente para cada SVI com base nas variáveis branch_number e distribution_number. Por exemplo, se a filial 1 tiver a sub-rede 172.17.0.0/20, a filial 2 tiver 172.17.16.0/20 e a filial 3 tiver 172.17.32.0/20, o IP do gateway será 172.17.x.1 (onde x é o número da filial). O segundo IP do primeiro switch de distribuição é 172.17.x.2 e o terceiro IP do segundo switch de distribuição é 172.17.x.3. Dessa forma, os endereços IP são calculados automaticamente, reduzindo erros e simplificando o processo.

A interface de loopback recebe um IP da variável loopback_ip, que serve como o ID do roteador OSPF para garantir roteamento estável e consistente através da rede. Na configuração do OSPF, esse IP de loopback é usado como o ID do roteador e as interfaces relevantes são adicionadas à área 0 do OSPF. Para HSRP, os valores de prioridade são definidos: 255 para o primeiro switch de distribuição e 250 para o segundo, garantindo o failover adequado. Além disso, a autenticação do HSRP é configurada usando uma cadeia de chaves (HSRP_KEY) para melhorar a segurança.

Para manter a configuração limpa e gerenciável, alguns valores são codificados. Por exemplo, a máscara de sub-rede (255.255.240.0) e determinadas configurações de HSRP (como versão e BFD) são as mesmas em todas as ramificações, reduzindo o número de variáveis. Isso torna a configuração mais simples, mais fácil de aplicar e menos propensa a erros. Por fim, as interfaces de uplink são configuradas com IPs e adicionadas à área 0 do OSPF para o roteamento adequado entre switches. Essa abordagem torna o processo de configuração mais fácil de gerenciar e menos propenso a erros, ao mesmo tempo em que é flexível para diferentes filiais.

Agora, reveja o módulo "Distribution ACL Config", que fornece segmentação na camada de distribuição.

```
{% 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 %}
```

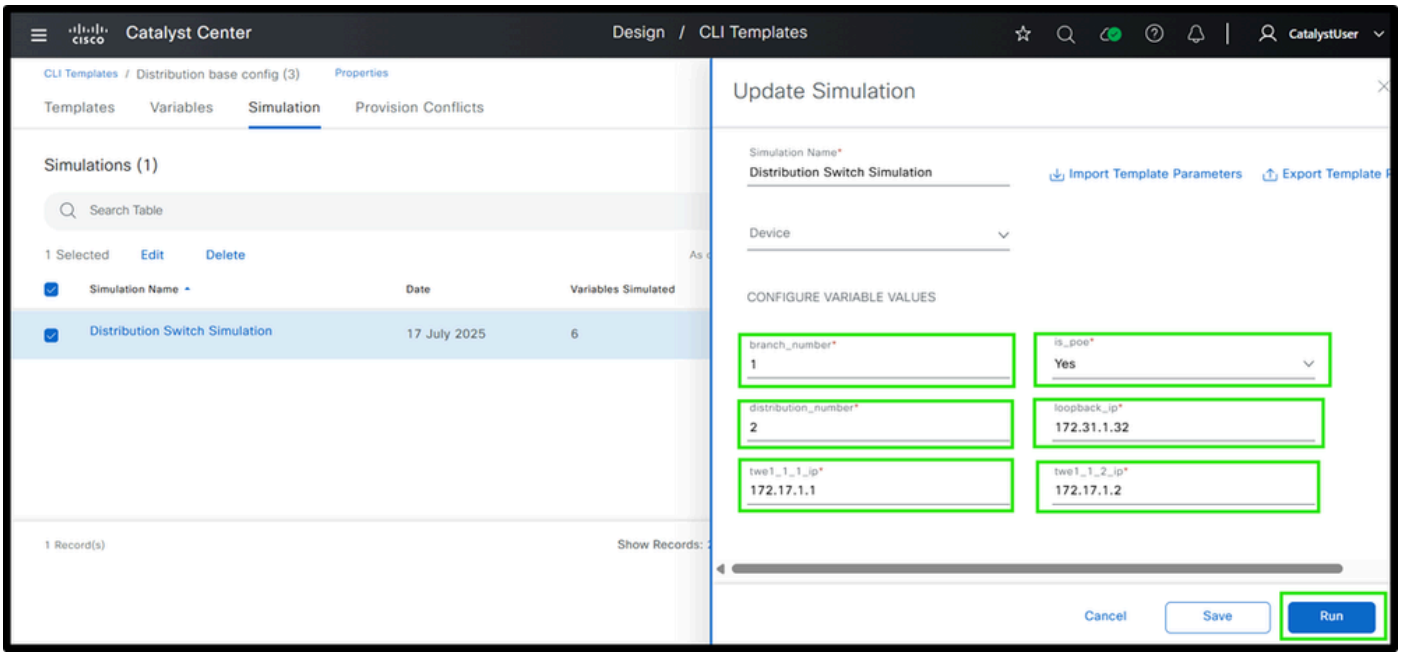
Configuração da ACL de distribuição

Este módulo demonstra segmentação na camada de distribuição usando um modelo Jinja2. Ele utiliza a variável branch_number para calcular dinamicamente os endereços de sub-rede, permitindo configurações de ACL automatizadas e escaláveis. Para cada filial, a ACL bloqueia a comunicação entre a sub-rede 1 (172.17.X.0) e a sub-rede 2 (172.16.X.0) negando o tráfego IP entre esses intervalos. Ele também nega o tráfego para o endereço multicast 239.255.255.250, ao

mesmo tempo em que permite todo o tráfego restante. A interface VLAN é atribuída dinamicamente com base no número da filial, e a ACL é aplicada na entrada dessa interface. Essa abordagem automatizada garante uma segmentação eficaz por filial, reduz os erros de configuração manual e simplifica a aplicação da política de rede.

Por fim, o último módulo, "Configuração padrão de distribuição", é quase idêntico ao descrito no módulo Configuração padrão de acesso (consulte essa seção para obter detalhes). Ele inclui práticas recomendadas, proteção de segurança e recursos importantes para o gerenciamento seguro de dispositivos. A única diferença está na interface de origem: no modelo Switch de acesso, ele é definido como VLAN {{ branch_number * 100 + 13 }}, enquanto na configuração do Switch de distribuição, ele pode ser codificado como Loopback0.

Passo 3: Execute a simulação antes de implantar a configuração.



(1) Entradas e Saídas de Simulação de Modelo de Switch de Distribuição

(2) Entradas e Saídas de Simulação de Modelo de Switch de Distribuição

(3) Entradas e Saídas de Simulação de Modelo de Switch de Distribuição

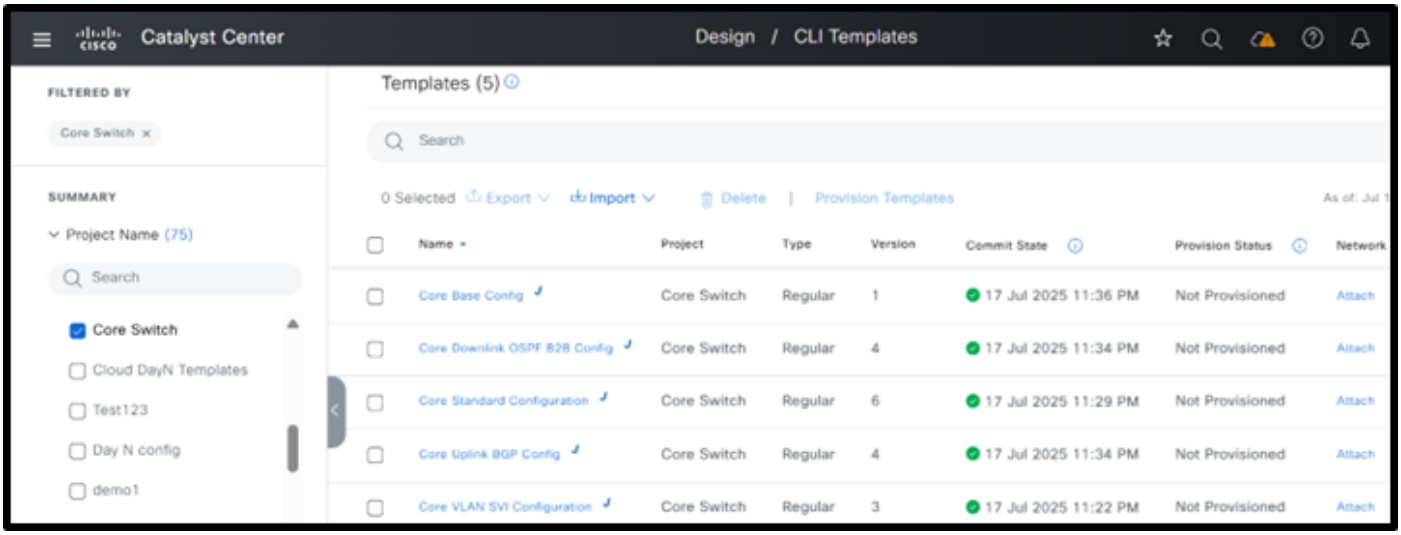
(4) Entradas e Saídas de Simulação de Modelo de Switch de Distribuição

É assim que os modelos podem ser usados na camada de distribuição para gerar configurações. Agora, vamos observar os dispositivos da camada central para ver como a modelagem pode ser aplicada ali.

Switches da camada central

Agora, projete um modelo modular para switches centrais. O modelo base e seus módulos fazem parte do projeto 'Core Switch' no Cisco Catalyst Center. Consulte o modelo básico na etapa 1.

Passo 1: Definir a estrutura de vários switches centrais



Estrutura do modelo do switch central

Passo 2: Definir vários módulos

```
{% 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 () }}
```

Configuração básica principal

A maioria das configurações de switch principais é semelhante em todas as ramificações, portanto, os valores comuns podem ser codificados. Geralmente, somente os endereços IP são alterados e eles podem ser definidos usando variáveis. Como cada filial normalmente tem apenas dois switches centrais, gerenciar essas variáveis é simples. Mesmo que algumas filiais tenham mais switches centrais, seu número ainda é menor que o número de switches de acesso ou distribuição. É por isso que, como prática recomendada, é mais importante minimizar as variáveis dos switches de acesso e distribuição, pois elas são usadas em maior número e ter muitas variáveis pode tornar a configuração mais demorada.

Agora comece com o primeiro módulo: "Configuração de SVI da VLAN principal." Neste exemplo, os switches centrais estão posicionados atrás de um firewall e devem estabelecer o peering eBGP com ele. Este módulo é responsável pela geração de VLANs e SVIs correspondentes necessários para o peering e vizinhança de OSPF do eBGP. Supõe-se que o firewall opere em uma configuração ativa/em espera.

```
{% 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 %}
```

Configuração do SVI da VLAN principal

Este módulo, conforme explicado anteriormente, cria as VLANs necessárias e as SVIs associadas para estabelecer relações de vizinhança OSPF e BGP. Todos os parâmetros, exceto os endereços IP do SVI, são codificados, incluindo a máscara de sub-rede, se ela estiver alinhada com o plano de endereçamento IP. Esse método ajuda a limitar variáveis e reduz a possibilidade de erros de configuração.

Agora, vamos rever o módulo "Configuração B2B do OSPF de downlink principal", que gera configurações para interfaces de downlink, OSPF e links back-to-back entre o switch central 1 e o switch central 2.

```
{% 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 %}
```

Configuração de OSPF B2B de downlink central

Semelhante ao módulo anterior, a maioria dos valores neste módulo são codificados para minimizar o número de variáveis. Somente os endereços IP das interfaces de loopback e downlink são variáveis. Além disso, os canais de porta back-to-back e as VLANs são padronizadas em diferentes filiais.

Agora, vamos revisar o módulo "Core Uplink BGP Config", que gera configurações de BGP e gerencia uplinks conectados aos firewalls.

```
{% 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 %}
```

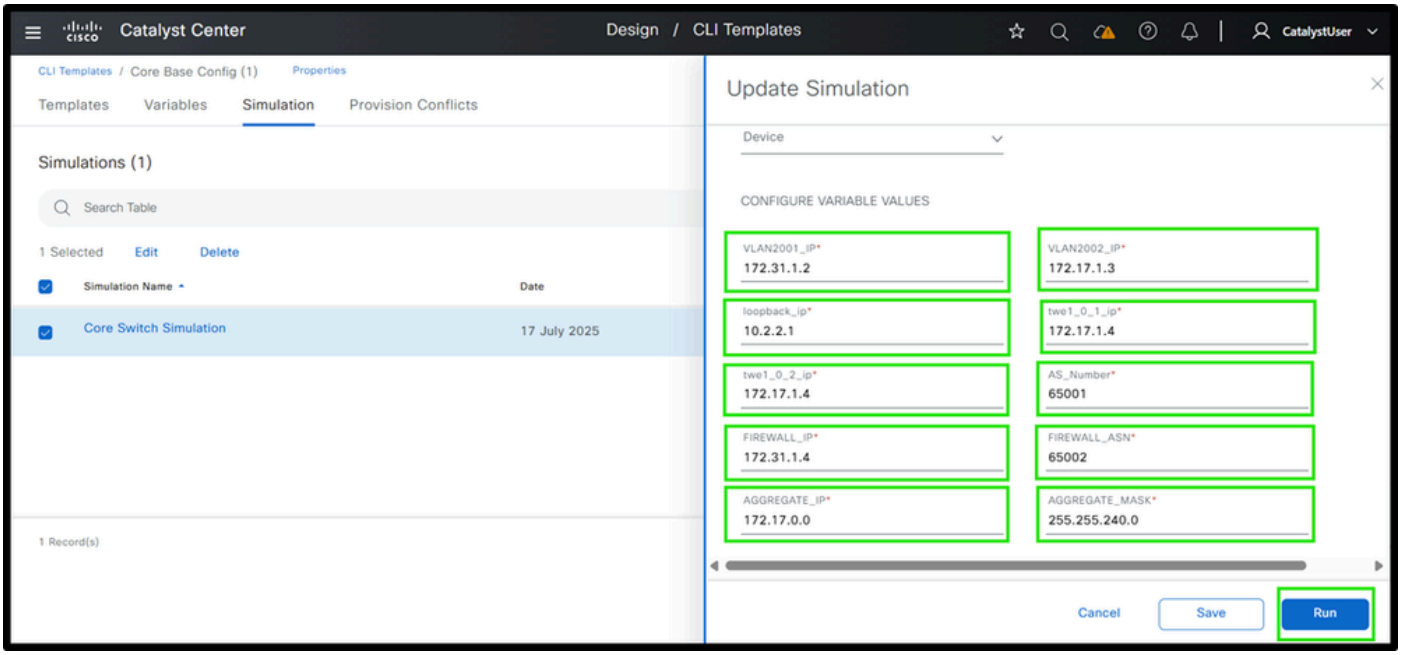
Configuração de BGP de uplink central

Este módulo gera a configuração de BGP necessária para estabelecer uma relação de vizinhança eBGP com o firewall. Como mostrado acima, a maioria dos valores é codificada, pois eles

permanecem consistentes em diferentes ramificações. Somente os endereços IP e os números AS, que podem variar para cada ramificação, são considerados como variáveis de entrada e usados para gerar a configuração necessária. A maioria das outras configurações foram padronizadas para minimizar o número de variáveis. As interfaces de uplink conectadas ao firewall são especificadas junto com a VLAN usada para a vizinhança eBGP, que foi gerada pelo módulo anterior.

Por fim, o último módulo, "Core Standard Configuration", é quase idêntico ao descrito na Access Standard Configuration (consulte essa seção para obter mais detalhes). Ele inclui práticas recomendadas, proteção de segurança e recursos importantes para o gerenciamento seguro de dispositivos. Consistente com a configuração do switch de distribuição, a interface de origem também pode ser definida como loopback0 neste módulo, e este valor pode ser codificado.

Passo 3: Executar simulação

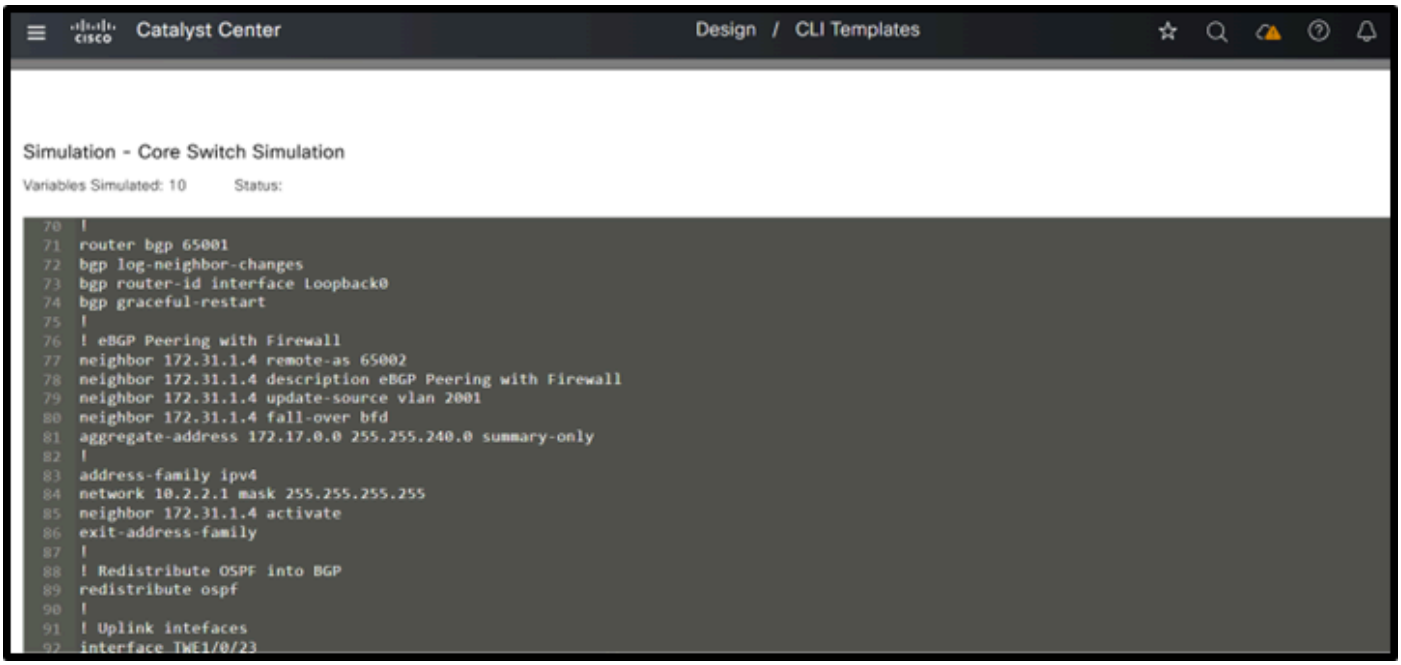


(1) Entradas e saídas de simulação do modelo de switch central

(2) Entradas e saídas de simulação do modelo de switch central

(3) Entradas e saídas de simulação do modelo de switch central

(4) Entradas e saídas de simulação do modelo de switch central



(5) Entradas e saídas de simulação do modelo de switch principal

Isso completa a explicação detalhada do projeto de modelos para a arquitetura de três camadas, descrevendo a estrutura e a configuração de cada módulo.

Todos esses módulos utilizam as práticas recomendadas explicadas anteriormente.

Note: Ao projetar modelos para uma arquitetura de núcleo recolhido, consulte as explicações fornecidas para a arquitetura de três camadas. A estrutura do modelo

---

permanece a mesma; no entanto, os recursos que antes eram implementados separadamente nas camadas de núcleo e de distribuição agora são combinados na camada de núcleo recolhido. A mesma abordagem de modelo modular também pode ser usada aqui, criando um modelo base e fazendo referência aos módulos relevantes dentro dele.

Summary

A arquitetura tradicional de campus de três níveis geralmente depende de uma extensa configuração manual nas camadas de núcleo, distribuição e acesso. Essa abordagem não só é demorada, como também é propensa a erros humanos. A ausência de automação e gerenciamento centralizado aumenta significativamente a sobrecarga operacional, dificultando o dimensionamento e o gerenciamento eficaz de redes de campus modernas e dinâmicas. Através do Catalyst Center, as configurações de recursos de modelo CLI podem ser automatizadas para as redes LAN tradicionais. É importante utilizar a abordagem modular ao provisionar os dispositivos. Os módulos podem ser baseados nos vários recursos usados em diferentes camadas. E, finalmente, vincular todos esses módulos ao módulo base.

Plano de ação

Convidamos as empresas a adotarem a metodologia de modelo modular apresentada neste white paper como uma prática recomendada para padronizar configurações de switch e otimizar operações de rede em arquiteturas de núcleo de três camadas e recolhidas.

- Com a implementação de modelos modulares, as equipes de rede podem:
- Melhorar a eficiência operacional por meio de práticas de configuração consistentes e reproduzíveis.
- Minimize o erro humano e reduza o tempo de solução de problemas.
- Obtenha maior escalabilidade para apoiar o crescimento e as necessidades empresariais em evolução.
- Garanta a consistência da configuração em diversos ambientes.

Essa abordagem não apenas simplifica o gerenciamento diário, como também permite implantações mais rápidas, simplifica os ciclos de atualização e melhora o alinhamento com os requisitos de segurança e conformidade. A adoção de modelos modulares posiciona sua rede para obter agilidade, resiliência e sucesso a longo prazo em um cenário de TI em constante mudança.

Para demonstrações práticas, saiba mais sobre modelos , consulte a série no YouTube

1 Como criar e gerenciar modelos no Catalyst Center

https://youtu.be/SyUqEEcwye0

2 Como usar variáveis de ligação do sistema em modelos CLI no Catalyst Center

https://youtu.be/gV1QBuHYJdo

Autores

Naveen Kumar, arquiteto de entrega ao cliente, experiência do cliente da Cisco

Risabh Mishra, engenheiro de consultoria, experiência do cliente da Cisco

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.