Informazioni sui modelli in Catalyst Center

Introduzione

Questo documento descrive il Cisco Catalyst Center e l'esperienza con i modelli di configurazione per architetture campus a tre livelli o con core compressi.

Premesse

Questo documento è destinato ai professionisti aziendali con una conoscenza di base del Cisco Catalyst Center e un'esperienza con i modelli di configurazione. È particolarmente importante per coloro che hanno lavorato o hanno intenzione di lavorare con architetture campus a tre livelli o con core compressi.

L'obiettivo principale è aiutare i lettori a implementare e automatizzare le soluzioni di configurazione e gestione utilizzando i modelli all'interno di Cisco Catalyst Center. Presentando informazioni approfondite, tecniche pratiche ed esempi concreti, questo documento rappresenta una risorsa pratica per coloro che desiderano migliorare le proprie competenze nell'infrastruttura LAN e ottimizzare i flussi di lavoro attraverso l'automazione e la gestione basata su modelli.

Sintesi

Con la continua evoluzione delle reti aziendali, la necessità di una gestione scalabile, coerente e automatizzata non è mai stata così elevata. Cisco Catalyst Center offre una piattaforma centralizzata basata sulle finalità che semplifica la configurazione, il provisioning e la garanzia sulle reti dei campus. In questo white paper viene illustrato come i professionisti di rete possono sfruttare l'editor di modelli CLI e le funzionalità di automazione di Cisco Catalyst Center per semplificare le operazioni di rete, ridurre gli errori di configurazione e accelerare le installazioni su architetture core a tre livelli e compresse. Illustra le procedure ottimali per la progettazione di modelli modulari basati su Jinja2, l'integrazione dell'automazione nei flussi di lavoro giornalieri e giornalieri e l'ottenimento della coerenza operativa tra i livelli Core, Distribution e Access. Adottando le strategie descritte in questo documento, è possibile trasformare la tradizionale gestione manuale della rete in un modello agile, standardizzato e basato sull'automazione in linea con la visione di rete basata sulle finalità di Cisco.

Sfide delle reti dei campus

Con l'evoluzione delle reti universitarie per soddisfare le esigenze delle organizzazioni moderne, queste si trovano ad affrontare diverse sfide fondamentali:

2 bis. Complessità nella gestione della rete

Molte funzioni di rete sono ancora gestite manualmente, aumentando il rischio di errore umano. Ciò non solo aumenta gli sforzi di manutenzione, ma mette a dura prova anche le risorse IT, soprattutto con budget limitati o statici.

2 ter. Problematiche di installazione e automazione

L'onboarding di nuovi dispositivi per reti cablate e wireless è spesso dispendioso in termini di tempo e complesso, con conseguenti ritardi nell'installazione e un aumento del sovraccarico amministrativo.

2 quater. Gestione delle immagini software

Mantenere un'"immagine d'oro" coerente sulla rete è una sfida. Molte reti finiscono con sistemi operativi multipli per dispositivi cablati e wireless, con conseguente inefficienza e difficoltà di gestione.

2g. Configurazioni di rete incoerenti

Le variazioni nelle configurazioni di rete possono causare problemi di conformità e inefficienze operative, rendendo più difficile mantenere una rete affidabile e sicura.
2 sexies. Aspettative crescenti degli utenti

Gli utenti richiedono connettività ininterrotta e un'esperienza applicativa ottimale, indipendentemente dalla loro posizione o dal dispositivo utilizzato. Per soddisfare queste aspettative, le reti devono essere resilienti, intelligenti e in grado di adattarsi ai cambiamenti in tempo reale.

Oltre a queste problematiche, le infrastrutture LAN moderne devono affrontare una serie di altre complessità.

Semplificazione delle reti dei campus con Cisco Catalyst Center

Cisco Catalyst Center è una soluzione di gestione di rete centralizzata per reti di campus, che supporta sedi centrali, filiali, connessioni cablate e wireless e ambienti IT/OT. Offre opzioni di installazione flessibili, tra cui appliance fisiche, server VMware ESXi o cloud AWS. Grazie a

funzionalità complete, Catalyst Center semplifica le operazioni, migliora le prestazioni e aumenta la sicurezza.

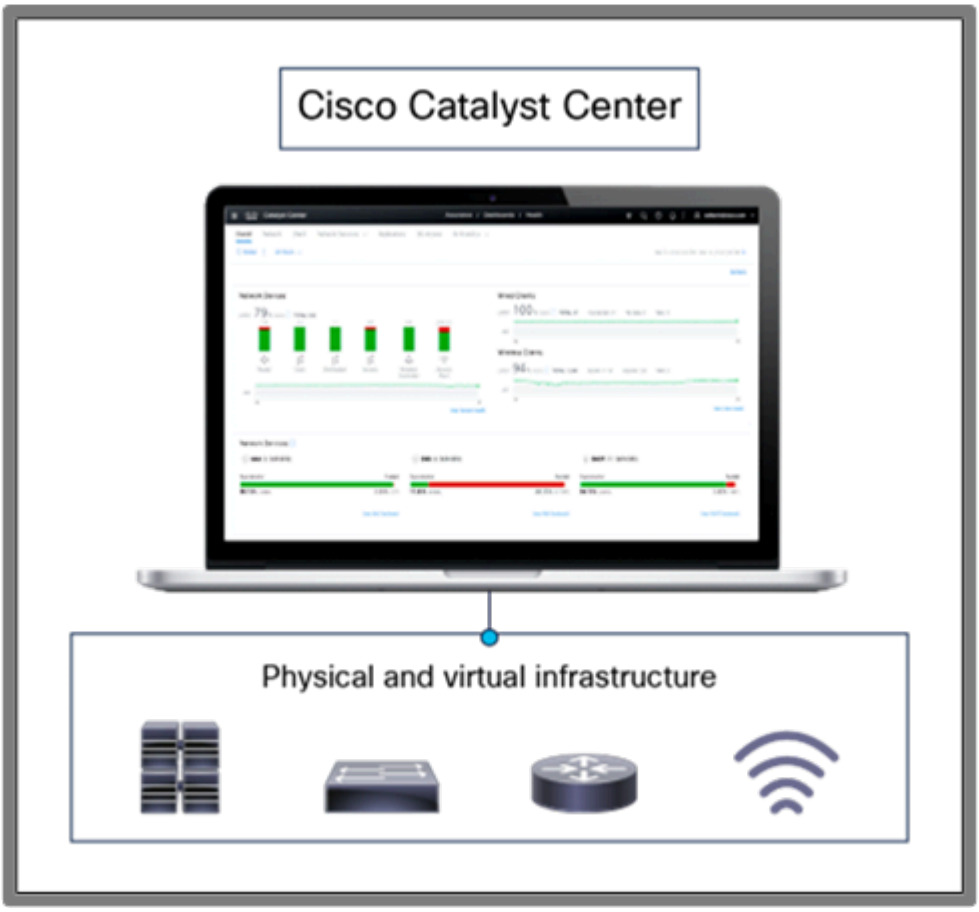


Figura 1: Gestione dell'infrastruttura con Cisco Catalyst Center

Caratteristiche e vantaggi principali

Cisco Catalyst Center (CC) offre funzionalità avanzate che semplificano la gestione e l'automazione della rete:

ZTP (Zero-Touch Provisioning): Automatizza l'onboarding dei dispositivi, riducendo le operazioni manuali e i tempi di installazione.

Gestione immagini software (SWIM): Assicura la coerenza delle versioni software tra i dispositivi con controlli precedenti e successivi all'aggiornamento per evitare problemi.

Automazione basata sugli intenti: Semplifica le installazioni convertendo l'intento della rete in configurazioni di dispositivi per reti cablate e wireless.

Automazione LAN: Automatizza l'indirizzamento e il routing IP di layer 3 per creare topologie complete.

Automazione rete wireless: Funzioni quali Plug and Play (PnP) consentono il provisioning rapido dei punti di accesso wireless.

Gestione gerarchica della rete: Consente profili specifici del sito (ad esempio SSID, parametri RF, VLAN) per implementazioni coerenti tra più sedi.

Modelli CLI : Catalyst Center Template Editor consente agli amministratori di creare e gestire facilmente modelli di configurazione basati su CLI, consentendo una distribuzione coerente ed efficiente tra i dispositivi.

Garanzia : Assurance consente il monitoraggio centralizzato dei dispositivi gestiti tramite CC.

Oltre a queste caratteristiche, Cisco Catalyst Center offre molte altre caratteristiche che esulano dall'ambito di questo documento. Questo documento si concentra principalmente sulla progettazione di modelli CLI utilizzando Catalyst Center.

Panoramica di alto livello dell'architettura del campus LAN con Catalyst Center

Le reti LAN tradizionali costituiscono la spina dorsale della connettività aziendale, garantendo una comunicazione affidabile e scalabile per i dispositivi cablati e wireless. Queste reti sono in genere progettate utilizzando l'architettura a 3 livelli o l'architettura core compressa, a seconda delle dimensioni e della complessità dell'organizzazione.

Architettura a tre livelli

L'architettura a tre livelli è un modello di progettazione di rete di base costituito dal livello principale, dal livello di distribuzione e dal livello di accesso. Questa architettura fornisce scalabilità, prestazioni elevate e gestione efficiente del traffico. Consultate la panoramica di ciascun livello.

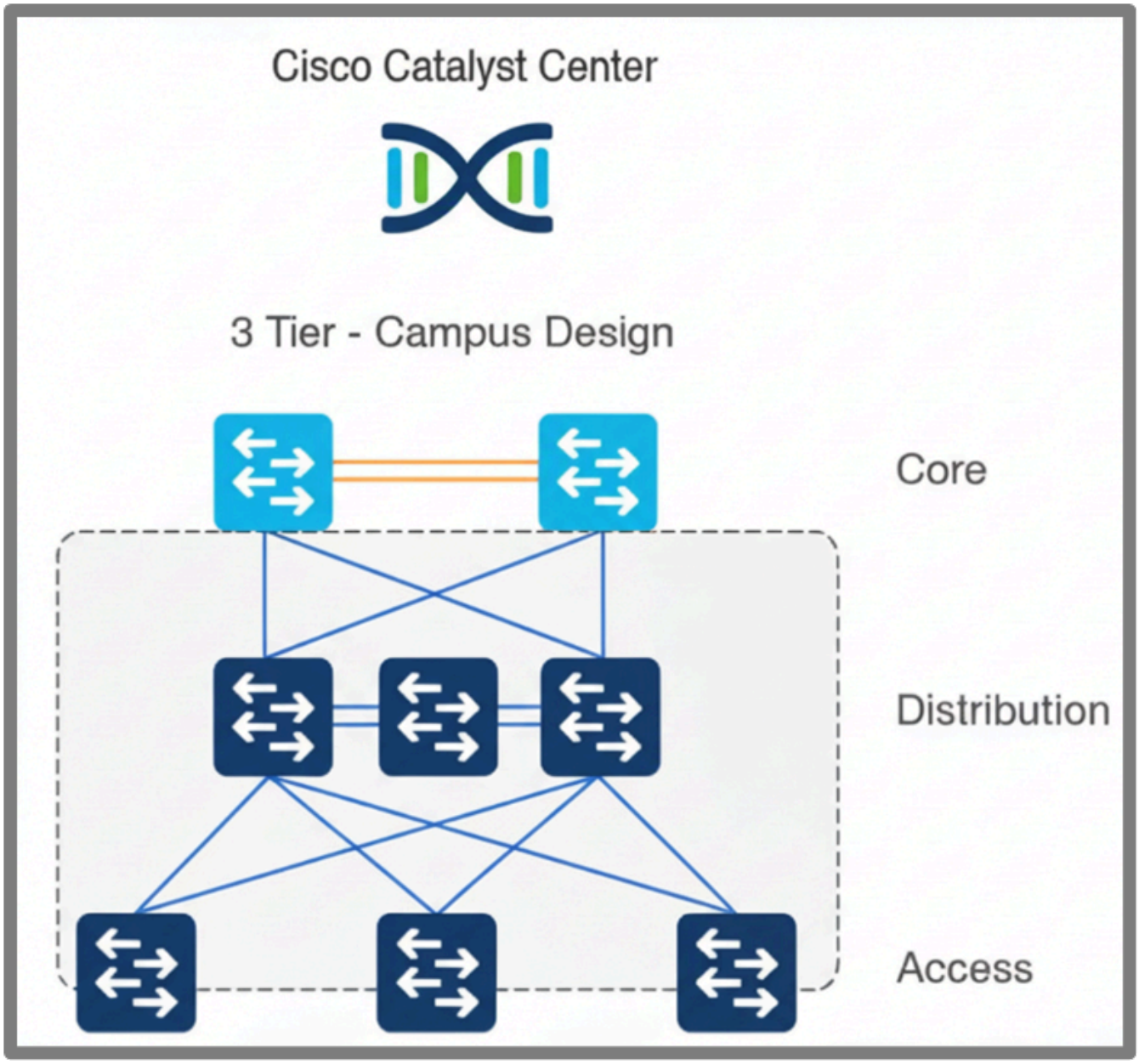


Figura 2: Architettura a tre livelli per campus

Livello core

Il layer core funge da backbone della rete, offrendo connettività e scalabilità ad alta velocità. Le configurazioni principali includono protocolli di routing in direzione nord e sud (ad esempio OSPF e BGP), policy di routing, configurazioni di interfaccia di downlink e uplink, protezione avanzata, ecc.

Livello di distribuzione

Il livello di distribuzione crea un ponte tra i livelli Core e Access, gestendo l'aggregazione del traffico, l'applicazione delle policy e la ridondanza. Le configurazioni principali includono HSRP/VRRP per la ridondanza, STP per la prevenzione dei loop, VLAN di layer 2 e layer 3,

configurazioni delle interfacce uplink e downlink, ACL per la sicurezza e protezione avanzata.

Livello di accesso

Il livello di accesso connette gli endpoint alla rete, consentendo un accesso sicuro e affidabile. Le configurazioni principali includono la configurazione dell'interfaccia di accesso, la configurazione dell'interfaccia uplink, le VLAN di layer 2, gli ACL per limitare l'accesso al dispositivo e la protezione avanzata.

Architettura di base compressa

L'architettura Core compressa combina i livelli Core e Distribution in un singolo livello, riducendo la complessità e i costi e mantenendo al contempo le prestazioni e la scalabilità. Questo approccio è indicato per le reti di piccole e medie dimensioni in cui non è richiesto un Core Layer separato. Visualizza la panoramica dei livelli in questa architettura.

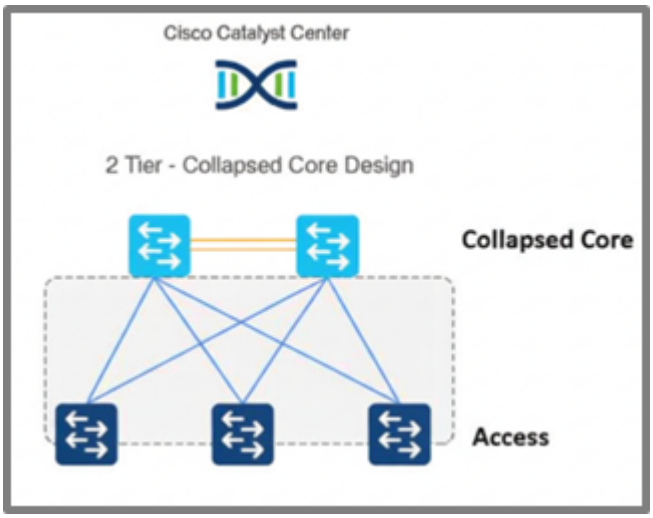


Figura 3: Architettura Core Campus compressa

Livello core compresso

Il livello di base compresso combina le funzioni dei livelli di base e di distribuzione, fornendo la connettività della backbone, l'aggregazione del traffico e l'applicazione delle policy. Le configurazioni chiave includono protocolli di routing in direzione nord e sud (ad esempio OSPF e BGP), policy di routing, configurazioni dell'interfaccia di downlink e uplink, BFD per il rilevamento degli errori, routing tra VLAN con SVI, HSRP/VRRP per la ridondanza del gateway, STP per la prevenzione dei loop e protezione avanzata. Sfruttando i modelli di Cisco Catalyst Center, queste configurazioni possono essere automatizzate, garantendo installazioni coerenti ed efficienti.

Livello di accesso

Come descritto in precedenza, il livello di accesso connette gli endpoint alla rete, consentendo un accesso sicuro e affidabile. Le configurazioni principali includono la configurazione dell'interfaccia di accesso, la configurazione dell'interfaccia uplink, le VLAN di layer 2, gli ACL per limitare l'accesso al dispositivo e la protezione avanzata.

Considerazioni sulla progettazione dei modelli

In questa sezione viene descritto come progettare modelli in Cisco Catalyst Center per generare configurazioni di dispositivi. L'Editor modelli semplifica il provisioning consentendo la creazione di modelli CLI riutilizzabili e supportando la distribuzione dinamica di configurazioni personalizzate per la rete. Catalyst Center supporta due linguaggi di modello: Jinja2 e Velocity. Questi linguaggi facilitano la gestione della configurazione dei dispositivi.

Jinja è un linguaggio di modellazione molto diffuso e di facile utilizzo, utilizzato principalmente con Python per la generazione di contenuti dinamici come HTML, XML o altri formati basati su testo. Consente di incorporare variabili e strutture di controllo (come loop e condizionali) all'interno dei modelli per creare output dinamico.

Apache Velocity è un motore di modellazione basato su Java che utilizza il linguaggio VTL (Velocity Template Language) per abilitare il contenuto dinamico in vari documenti, tra cui pagine Web, XML o codice sorgente. Unisce i dati degli oggetti Java con i modelli per produrre l'output finale.

Questo documento copre solo i modelli Jinja2.

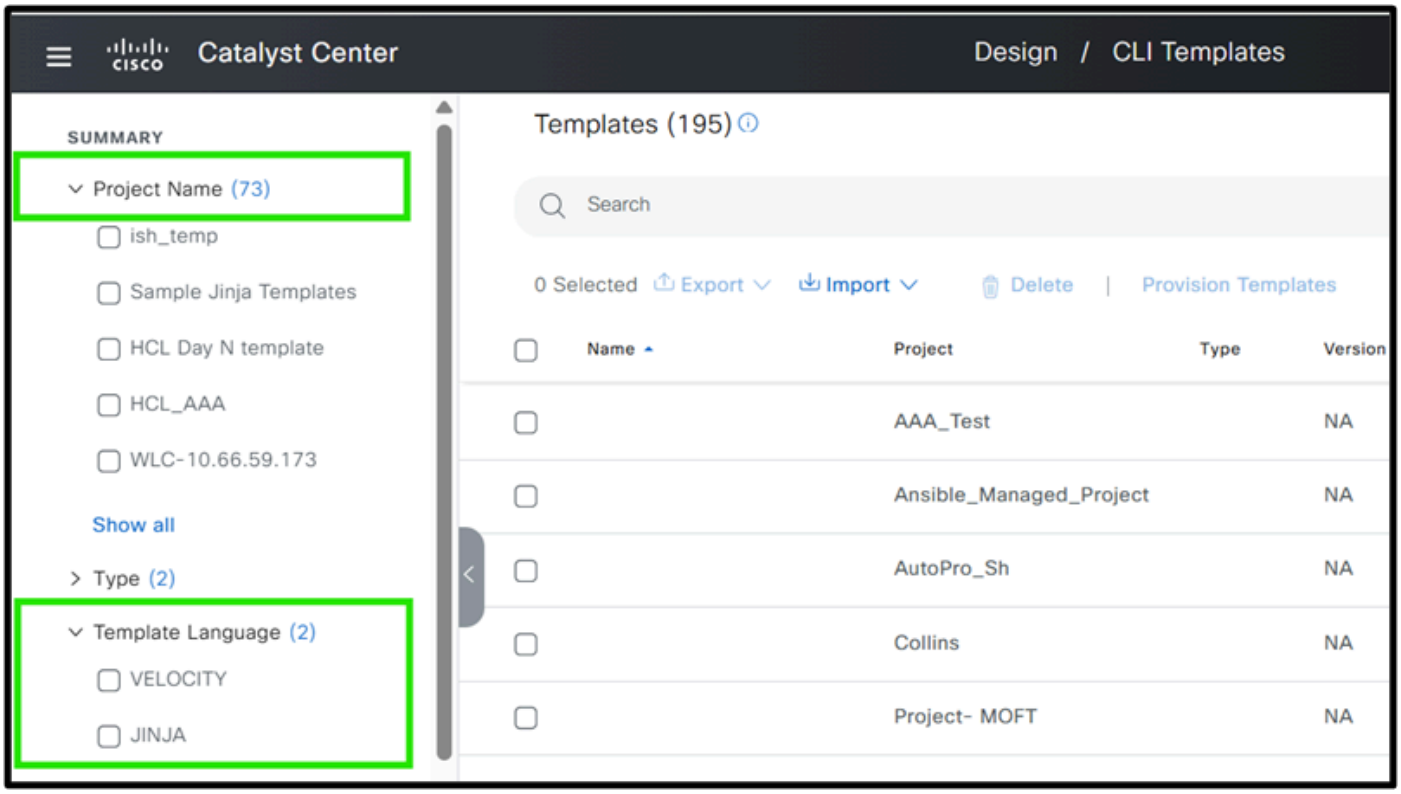


Figura 4: Cisco Catalyst Center Template Editor

In questo documento, viene utilizzato Jinja2 per la sua flessibilità. Anziché un'esplorazione approfondita di Jinja2, l'attenzione si concentra sull'applicazione pratica per la progettazione di modelli. Per ulteriori informazioni sul modello Jinja2 nel Catalyst Center, fare riferimento al collegamento:

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

Prima di addentrarsi nelle strategie di progettazione dei modelli per una rete campus Cisco, è importante utilizzare le best practice chiave per garantire efficienza e gestibilità quando si utilizzano i modelli.

Struttura dei modelli e best practice / Linee guida per la migliore strategia

Quando si automatizza la configurazione dei dispositivi di rete utilizzando Cisco Catalyst Center, è essenziale adottare strategie strutturate e best practice. Questi passaggi assicurano coerenza, scalabilità e facilità di gestione nell'infrastruttura di rete.

- Dividi configurazione per ruolo dispositivo

Iniziare classificando i dispositivi in base al loro ruolo nella topologia di rete. I ruoli comuni includono:

- Nucleo
- Distribuzione
- Accesso

Esempio: I dispositivi che funzionano come switch Core devono avere requisiti di configurazione diversi rispetto agli switch Access.

Suddivisione della configurazione in blocchi modulari

All'interno di ogni ruolo del dispositivo, suddividere la configurazione in blocchi modulari raggruppando insieme funzionalità o configurazioni simili. Questo approccio modulare semplifica l'automazione, la risoluzione dei problemi e gli aggiornamenti futuri.

Esempi per un dispositivo di base:

- Blocco di configurazione OSPF
- Blocco configurazione BGP
- Blocco criteri QoS

Individuazione blocchi di configurazione indipendenti dai ruoli

Alcuni blocchi di configurazione si applicano universalmente a tutti i ruoli dei dispositivi. L'identificazione e la standardizzazione di questi blocchi garantisce procedure ottimali e coerenza in tutta la rete.

Blocchi di configurazione comuni indipendenti dai ruoli:

- Configurazione di base: Nome host, banner di accesso
- Protocolli di gestione: DHCP, DNS, NTP, SNMP

Criteri di accesso: configurazioni di protezione standard

Questi blocchi possono essere riutilizzati per dispositivi Core, di distribuzione e di accesso, semplificando il processo di automazione.

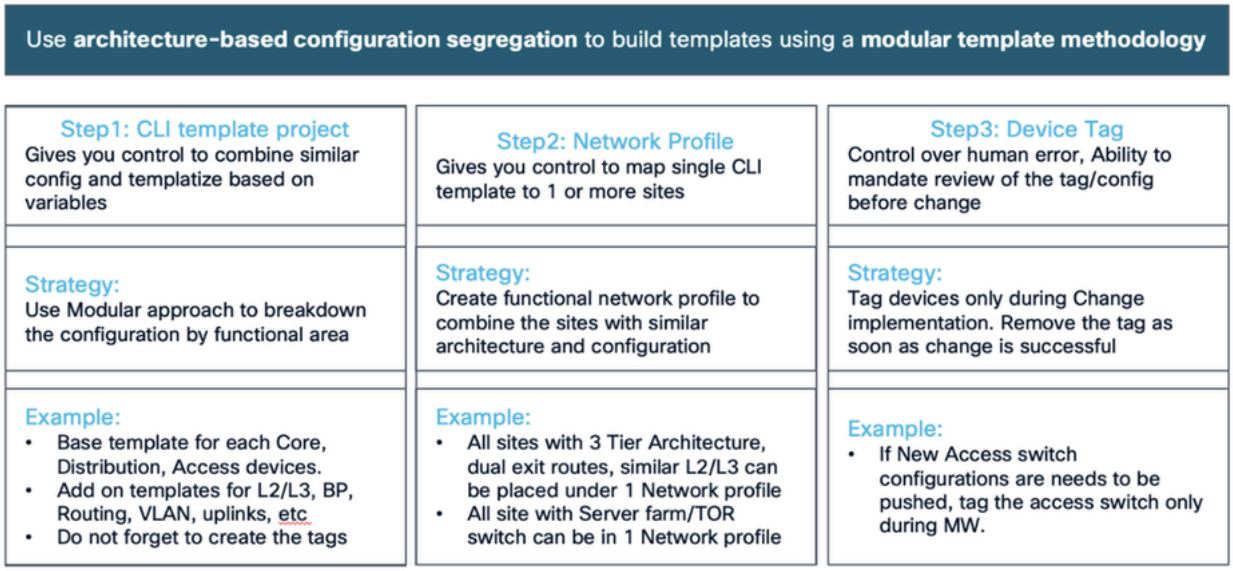


Figura 1: Procedura consigliata con esempio

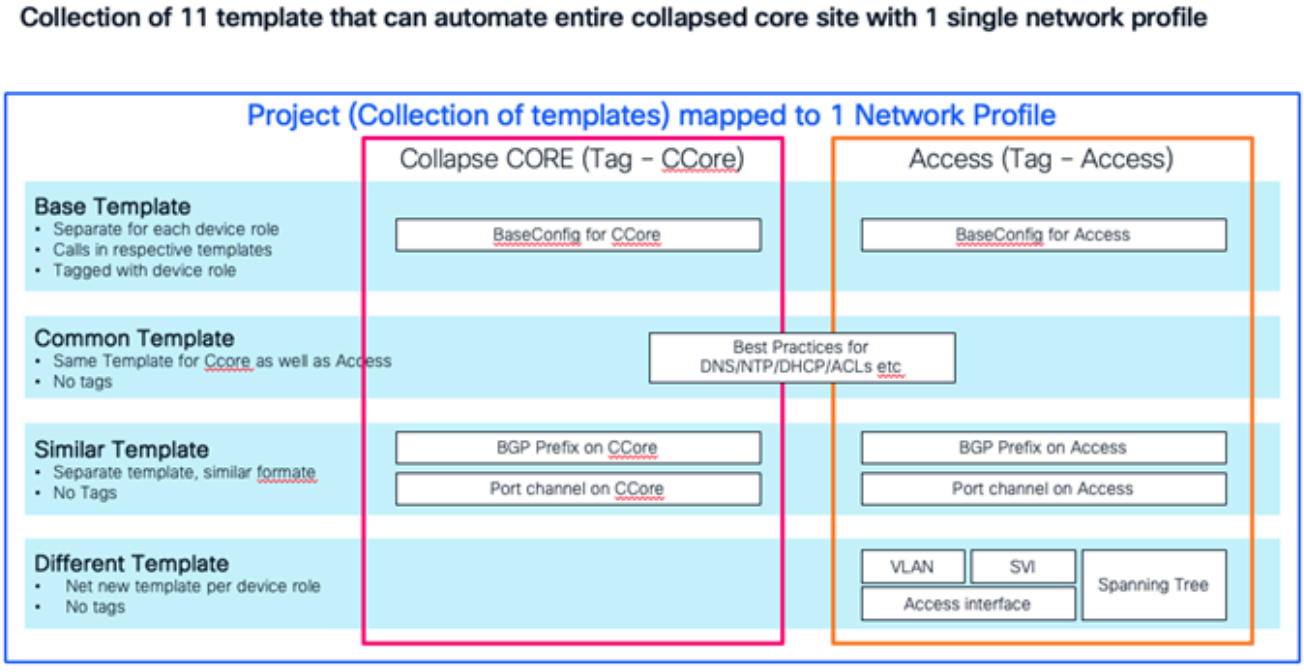


Figura 2: Esempio di modello di base compresso

Procedure ottimali per l'utilizzo dei modelli

Progettazione di modelli modulari per la configurazione automatizzata

Quando si automatizzano le configurazioni dei dispositivi in Cisco Catalyst Center, evitare di incorporare tutte le configurazioni in un unico modello monolitico. Adottare invece un approccio modulare:

Creare un modello di base che faccia riferimento a modelli (moduli) più piccoli e specifici per uno scopo:

Suddividere la configurazione in moduli logici (ad esempio impostazioni dell'interfaccia, protocolli di routing, funzionalità di sicurezza).

Questa struttura rende gli aggiornamenti più efficienti: le modifiche a un modulo specifico vengono automaticamente applicate ovunque venga utilizzato il modulo, riducendo in modo significativo gli errori e la complessità.

Esempio: Configurazione modulare per un dispositivo di succursale

Si supponga di automatizzare la configurazione di un dispositivo di diramazione.

Modello di base:

Include riferimenti ai modelli di modulo per le aree di configurazione principali.

Trasmette le variabili necessarie a ciascun modulo per la personalizzazione.

Modelli di modulo:

impostazioni_interfaccia: Gestisce le configurazioni dell'interfaccia.

protocolli_routing: Contiene le impostazioni OSPF, EIGRP o BGP.

caratteristiche_sicurezza: Definisce ACL, regole firewall o altri criteri di sicurezza.

```
{% 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) }}
```

Esempio di struttura del modello di base:

Con questa struttura, qualsiasi modifica alle configurazioni di routing o di sicurezza deve essere apportata solo nei rispettivi moduli e tali modifiche vengono immediatamente applicate a ogni utilizzo del modello di base. Ciò rende le configurazioni più gestibili e coerenti su tutti i router delle filiali.

Il nome del progetto è Branch e altri 3 moduli diversi sono definiti in project. Tutti questi elementi vengono combinati nel modello di base.

Riduci a icona variabili nel modello

Ridurre al minimo il numero di variabili nel modello per ridurre la complessità e gli errori. Un numero inferiore di variabili semplifica l'installazione, in particolare nelle reti di grandi dimensioni, rendendo il processo più efficiente e coerente.

Utilizzo dei tag dei dispositivi per i modelli

Utilizzare i tag dei dispositivi in Cisco Catalyst Center, ad esempio la posizione, il ruolo o il sito, per creare modelli Jinja2 dinamici e scalabili. Questi tag abilitano la logica condizionale, assicurando che le configurazioni corrette vengano applicate ai dispositivi appropriati. Questo approccio riduce al minimo gli errori e semplifica la gestione dei modelli in diversi ambienti di rete.

Valori Statici Hardcode Dove Possibile

L'hardcode di valori statici può semplificare i modelli e migliorare l'efficienza dell'installazione. Esempi comuni sono gli indirizzi IP per i server DNS, NTP o Syslog, che in genere rimangono coerenti tra i dispositivi. Analogamente, l'uso di ID VLAN standard sugli switch di accesso consente di codificare questi valori, riducendo la variabilità e accelerando l'implementazione.

Adottare un approccio in due fasi: Modelli Giorno 0 e Giorno N

Quando si caricano dispositivi utilizzando servizi come Cisco Plug and Play, utilizzare una strategia di modello in due fasi:

Modelli giorno 0: Eseguire il push delle configurazioni di base per verificare che il dispositivo possa comunicare con Cisco Catalyst Center.

Modelli Giorno N: Distribuire funzionalità e configurazioni avanzate quando il dispositivo è raggiungibile.

Le best practice consentono di creare modelli efficienti e scalabili che semplificano le installazioni di reti all'interno di campus Cisco.

Controllo degli spazi vuoti nelle macro dei modelli Jinja

Quando si creano modelli utilizzando il linguaggio Jinja, è essenziale gestire con attenzione gli spazi vuoti e le nuove righe, in particolare quando si esegue il rendering del contenuto dinamico all'interno delle macro. Gli spazi vuoti accumulati o le righe nuove non intenzionali possono causare problemi di formattazione nell'output generato, che devono causare errori di interpretazione o errori nell'elaborazione a valle. Per risolvere questo problema, Jinja fornisce la sintassi per il controllo dello spazio vuoto: inserendo un segno meno (-) direttamente all'interno dei delimitatori ({{- ... -}} o {%- ... -%}) viene rimosso qualsiasi spazio vuoto iniziale o finale intorno all'espressione. Sostituendo ad esempio {{item[1]}} con {{- item[1] -}} si garantisce che eventuali spazi o nuove righe aggiuntive vengano rimossi durante il rendering della macro. Questa procedura è particolarmente utile quando si scorrono gli elenchi o si generano i file di configurazione, come illustrato nello snippet di modello. È consigliabile applicare sempre il controllo degli spazi vuoti in tali scenari per mantenere risultati puliti e prevedibili.

Esempio (uso consigliato):

```
{% per l'elemento in wildcard_list %}
   {% se elemento[0] == prefisso -%}
      {{- elemento[1] -}}
   {%- endif %}
{%- fine %}
```

Architettura a tre livelli

Questo white paper inizia con lo sviluppo di modelli per gli switch di accesso fino agli switch principali e descrive i requisiti per ogni livello.

Access Layer Switch

Gli switch di accesso sono integrati tramite Plug and Play e devono richiedere un modello per il giorno 0. Per ulteriori informazioni sul processo Plug and Play in Catalyst Center, fare riferimento al collegamento:

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

Come descritto in precedenza, Catalyst Center supporta sia i linguaggi di modellazione Velocity che Jinja2. Questo documento utilizza Jinja2 per illustrare la struttura del modello, grazie alla sua flessibilità. La configurazione dello switch del livello di accesso può essere distribuita utilizzando il modello Giorno-0 e Giorno-N.

È possibile strutturare un modello base Giorno 0, vedere Passo 1:

Passaggio 1: Definisci modello

```
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
!
```

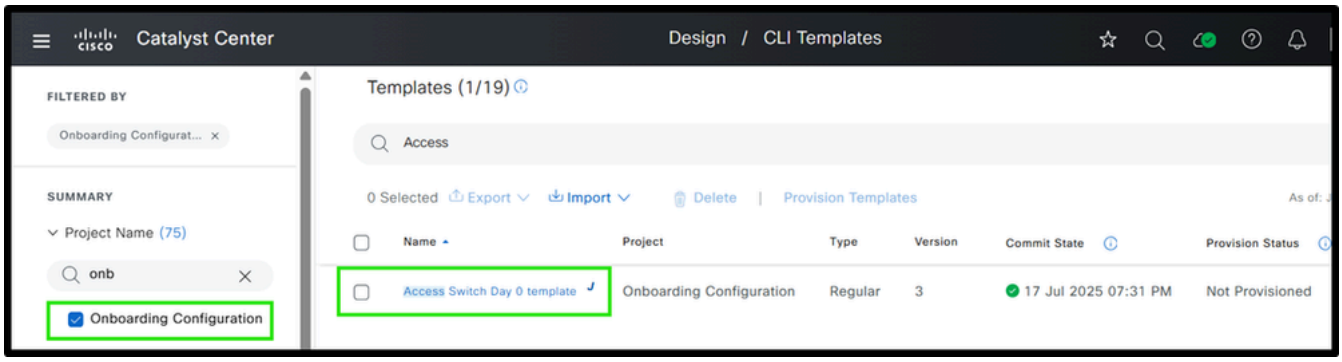
Passaggio 1: Definisci modello

Il modello semplifica la configurazione mediante costanti hardcode quali nome utente, password, enable secret e subnet mask, in quanto tutti gli switch di una diramazione condividono la stessa subnet mask della VLAN di gestione. L'indirizzo IP di gestione, tuttavia, è univoco per ciascuno switch e viene definito come variabile. Nel modello Giorno N, che

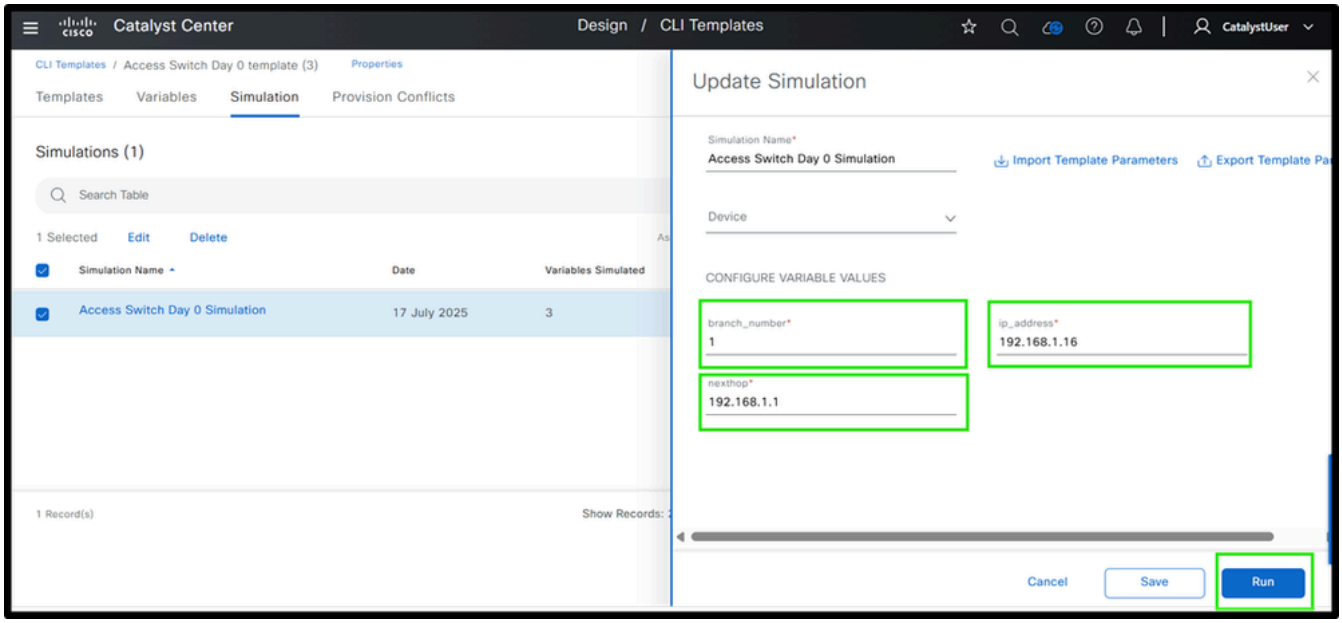
utilizza questo modello Giorno 0, deve essere fornita una struttura di modello completa. Nel modello Day N, ciascuna funzionalità dello switch di accesso è gestita da un modulo dedicato. Ad esempio, un modulo gestisce VLAN di layer 2, moduli separati gestiscono interfacce di accesso uplink e downlink, un altro modulo è incentrato sulla protezione avanzata e così via.

Anche se si preferiscono ID VLAN coerenti, è possibile generare dinamicamente ID variabili utilizzando una formula basata sul numero di filiale (ad esempio, Filiale 1 = VLAN 13, Filiale 2 = VLAN 213). In questo modo il modello è riutilizzabile tra i rami. Analogamente, l'indirizzo IP dell'hop successivo è una variabile, in quanto deve essere diverso per ogni branch a seconda del cluster di distribuzione connesso.

Passaggio 2: Esegui simulazione e fornisci variabile

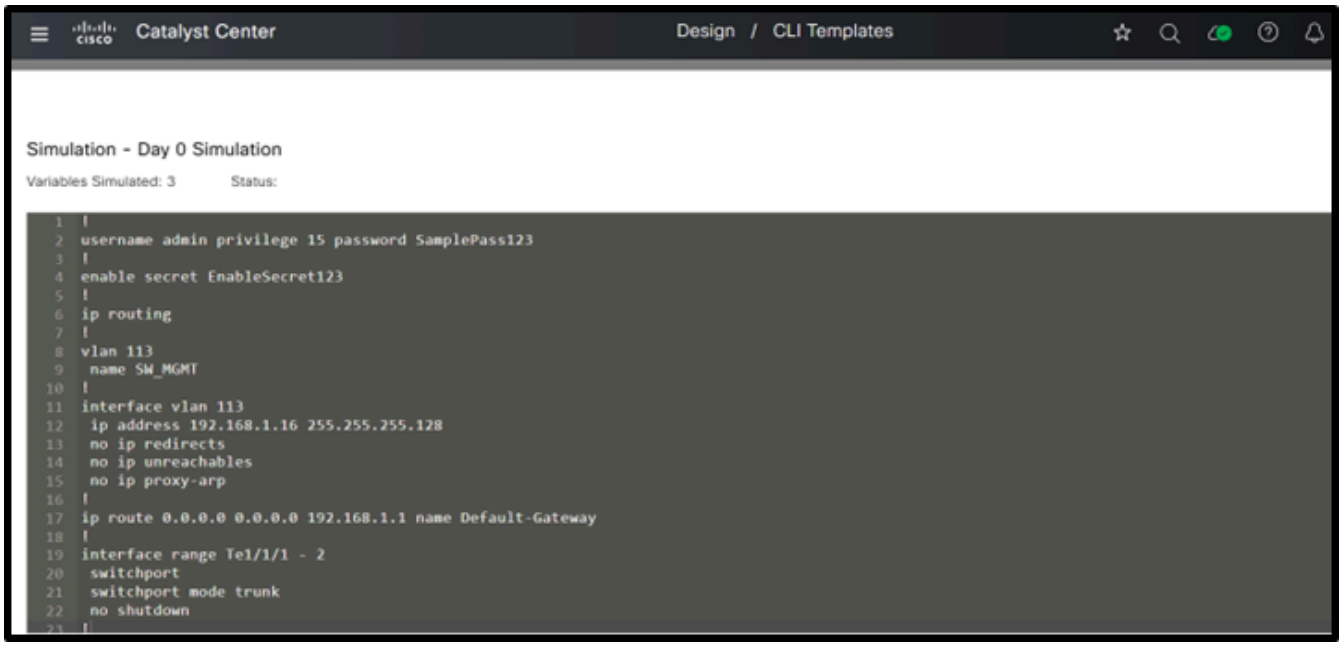


Struttura del modello Access Switch Day 0 con input e output di simulazione



Es. Input simulazione

Si consiglia sempre di simulare il modello prima della distribuzione. La schermata mostra la configurazione finale dopo l'immissione delle variabili.



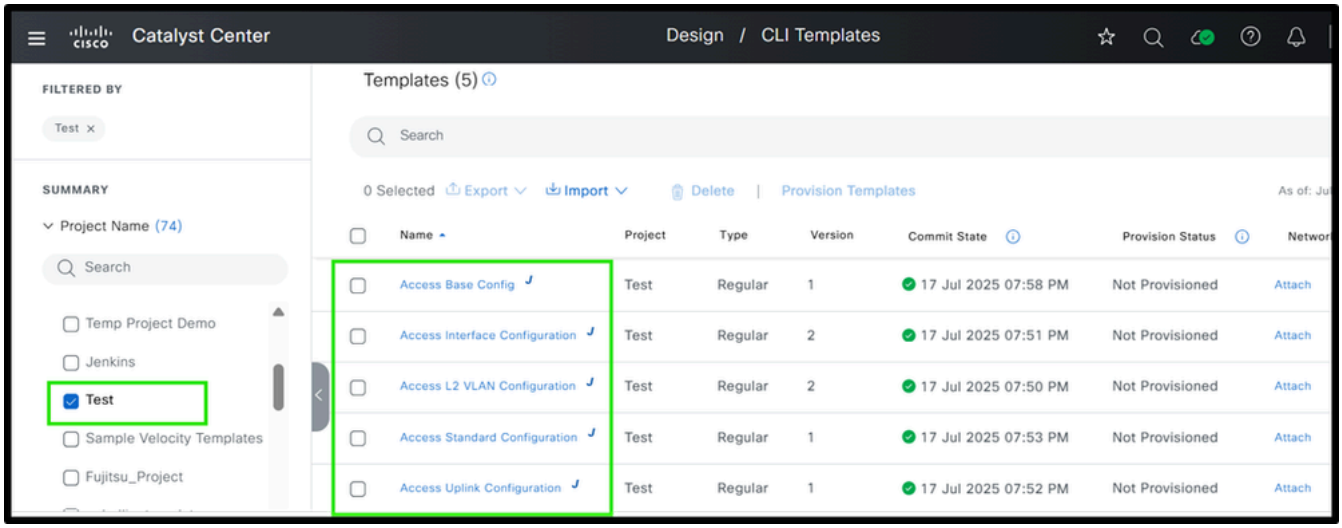
Config finale dopo l'immissione dei valori

A questo punto verrà illustrato come creare un modello modulare per il giorno N.

La configurazione dello switch di accesso può essere suddivisa in vari moduli, che possono essere tutti combinati all'interno di un modulo di base. Il modello base per gli switch di accesso è strutturato come mostrato.

Sia il modello di base che i relativi moduli vengono creati all'interno di un progetto denominato "Test" in Cisco Catalyst Center.

Passaggio 1: Definizione di vari modelli, incluso il modello di base



Struttura del modello Access Switch Day N

Passaggio 2: Definire vari moduli

Configurazione base di accesso:

Nello screenshot è illustrato un esempio della configurazione di base.

```
{% 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) }}
```

Configurazione di base di Access

Questo modello di configurazione modulare comprende quattro parti: Configurazione VLAN, configurazione dell'interfaccia uplink, configurazione dell'interfaccia di accesso e configurazione standard. Vengono utilizzate solo due variabili: branch_number e is_poe, che ne semplificano la gestione.

Il valore branch_number calcola gli ID VLAN specifici di una filiale, come mostrato nel modello Giorno 0, e is_poe determina se lo switch di accesso è uno switch PoE o uno switch non PoE. Queste variabili vengono fornite durante il provisioning e passate ai moduli per creare le configurazioni corrette, riducendo gli sforzi e migliorando l'efficienza.

Esaminiamo ora ciascun modulo per verificare in che modo contribuisce alla generazione di parti specifiche della configurazione complessiva.

Accesso alla configurazione della VLAN 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 %}
```

Accesso alla configurazione della VLAN L2

Questo modulo crea le VLAN in base al numero di filiale, come spiegato in precedenza. Le VLAN dati e voce vengono create su tutti gli switch, che supportino o meno la PoE. La VLAN di gestione dell'access point (ad esempio, 114 per la diramazione 1) viene creata solo se is_poe è impostato su "Sì", ossia lo switch supporta PoE. Se is_poe è "No", la VLAN di gestione dell'access point viene ignorata perché gli switch non PoE non possono supportare i punti di accesso. Questa condizione viene gestita tramite una condizione 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 %}
```

Configurazione interfaccia di accesso

Questo modulo gestisce la configurazione dell'interfaccia di accesso e utilizza lo stesso approccio dello switch PoE descritto in precedenza. Se la variabile is_poe è "Yes", ossia lo switch è uno switch PoE, le prime sei porte (1-6) devono essere configurate con la VLAN di gestione dell'access point. In caso contrario, le prime sei porte devono essere impostate come porte di accesso utente.

Supponendo che lo switch sia un modello a 24 porte, le porte rimanenti (7-24) vengono sempre configurate come porte di accesso utente, indipendentemente dal fatto che lo switch sia PoE o meno.

L'intervallo di interfacce è stato standardizzato e non è più considerato una variabile di input, il che è considerato una buona pratica per ridurre al minimo il numero di variabili nel modello. Inoltre, il modulo include una macro denominata common_access_settings, che riduce al minimo le dimensioni del modello consolidando le configurazioni ripetute. Questa macro viene chiamata semplicemente all'interno delle impostazioni dell'interfaccia, evitando di specificarle

più volte.

Nota: Questo modello applica la stessa descrizione a tutte le interfacce di accesso. Se sono necessarie descrizioni univoche per ciascuna interfaccia, si consiglia di spingerle utilizzando script Python separati o strumenti di automazione simili.

Esaminare il modulo che genera le configurazioni per le interfacce 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 %}
```

Configurazione di Access Uplink

Questo modulo genera la configurazione per le interfacce uplink e gestisce l'eliminazione delle VLAN. Se lo switch supporta PoE, la VLAN di gestione AP è inclusa nell'elenco delle VLAN consentite; in caso contrario, è esclusa. Questa logica viene gestita utilizzando la condizione if nel codice, come descritto in precedenza.

Esaminare il modulo finale, che illustra le configurazioni standard, incluse le procedure ottimali e la protezione avanzata.

Attenzione: Questa operazione ha solo scopo illustrativo e non deve essere utilizzata come riferimento per le impostazioni di rete effettive, in quanto le configurazioni possono variare in base a requisiti specifici

---

```
{% 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: Accesso alla configurazione standard

```
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: Accesso alla configurazione standard

Questo modulo genera una configurazione standard che incorpora best practice, protezione avanzata e funzionalità chiave per una gestione sicura dei dispositivi. La maggior parte dei valori è hardcoded per la coerenza tra le diramazioni, ad eccezione di branch_number, che viene utilizzato per calcolare la VLAN di gestione per gli switch in ciascuna diramazione e funge da interfaccia di origine per diverse configurazioni.

Passaggio 3: Eseguire una simulazione prima di configurare gli switch. È necessario simulare solo la configurazione di base.

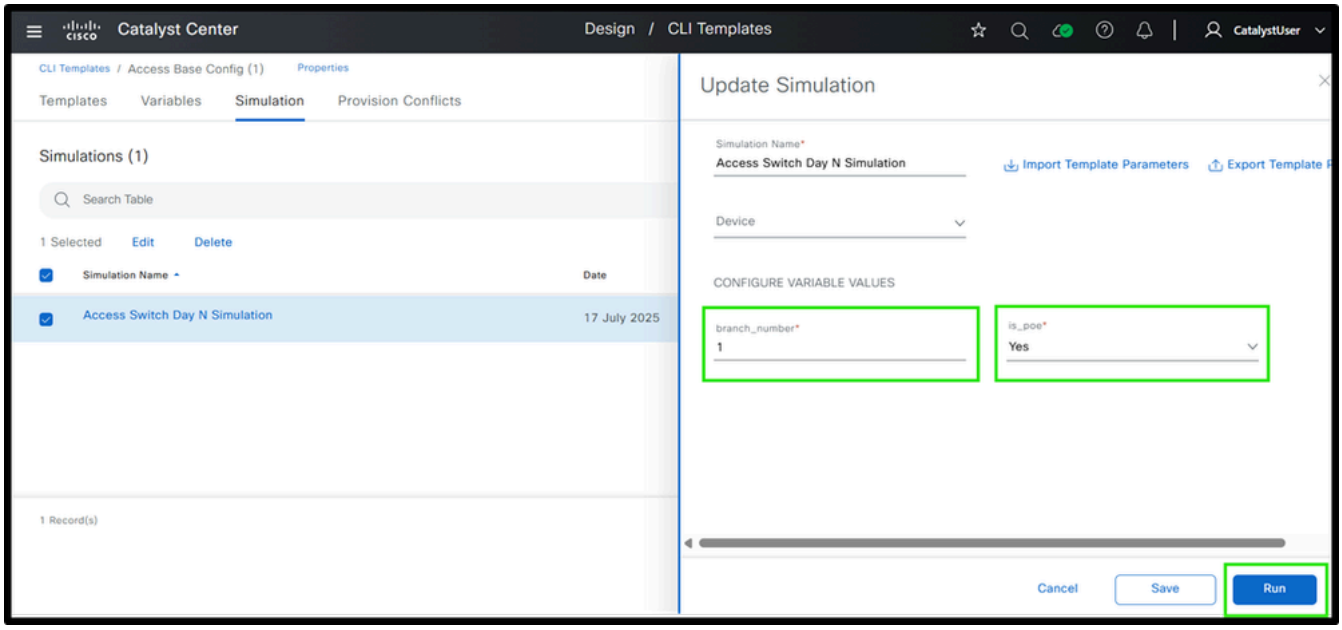
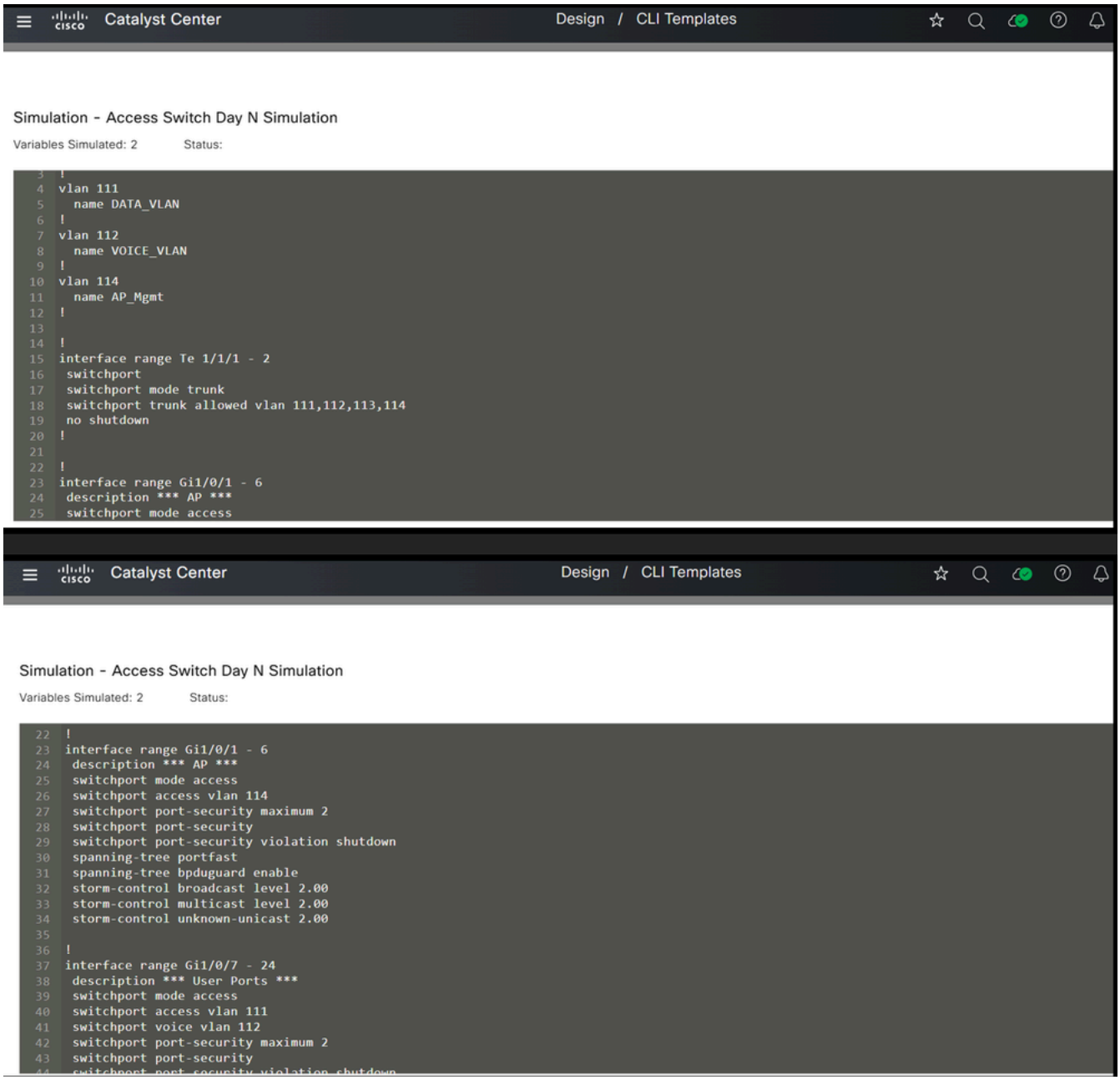


Figura 7: Access Switch Day N - Input e output simulazione modello



Simulazione

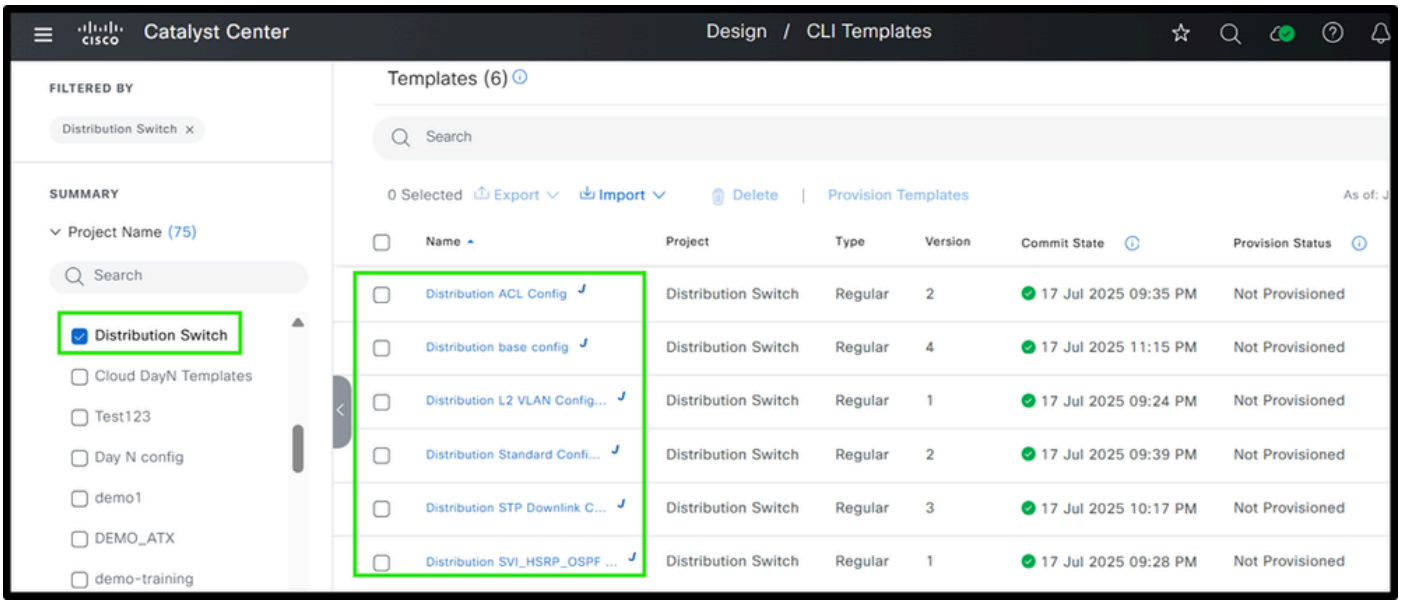
In questo modo è possibile utilizzare i modelli a livello di accesso per generare le configurazioni.

Ora, diamo un'occhiata ai dispositivi del livello di distribuzione per vedere come la modellazione può essere applicata a loro.

Switch per livelli di distribuzione

A questo punto è necessario progettare un modello modulare per gli switch di distribuzione. Il modello di base e i relativi moduli fanno parte del progetto 'Distribution Switch' in Cisco Catalyst Center.

Passaggio 1: Struttura modello commutatore di distribuzione



Es. Modelli di distribuzione

Passaggio 2: Definire ogni modulo

La configurazione di base fornita definisce ogni modulo e fa riferimento a tutti i moduli.

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

Es. Moduli modello base di distribuzione

Analogamente agli switch di accesso, tutti i modelli vengono creati all'interno del progetto 'Distribution Switch' e sono menzionati nel modello di base. Mentre alcuni modelli sono identici a quelli usati per gli switch di accesso, questa sezione spiega le differenze specifiche degli switch di distribuzione. Il modulo "Distribution L2 VLAN Configuration" è identico a quello descritto sopra per gli switch di accesso. Controllare il modulo di configurazione VLAN L2 di Access che fornisce queste informazioni. e genera le VLAN richieste in base ai valori di input forniti per le variabili.

Esaminare ora il modulo "Distribution STP Downlink Config", che gestisce la generazione delle

configurazioni spanning tree e uplink per gli switch di distribuzione.

```
{% 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 %}
```

Configurazione downlink STP distribuzione

In questo caso viene utilizzata la funzionalità macro Jinja2, a cui viene fatto riferimento nel modulo basato. Quindi questa struttura aiuta a costruire un approccio modulare.

Questo modulo configura lo Spanning Tree Protocol (STP) e le interfacce di downlink in base al "branch_number" e se lo switch è abilitato o meno per PoE. La variabile "branch_number" viene utilizzata per generare VLAN di base univoche per ciascuna filiale, garantendo VLAN distinte, in modo simile all'approccio già evidenziato per gli switch di accesso. Se lo switch è abilitato per PoE ("is_poe" == 'Yes'), viene aggiunta una VLAN aggiuntiva, ad esempio la VLAN di gestione dell'access point. La variabile "distribution_number" determina la priorità STP, impostando 4096 per la distribuzione 1 (rendendola il bridge principale preferito) e 8192 per gli switch di distribuzione secondari. Infine, all'interfaccia trunk vengono applicate le VLAN appropriate, in modo da consentire solo le VLAN appropriate a seconda che lo switch sia abilitato o meno per PoE.

Esaminare ora il modulo "Distribution SVI_HSRP_OSPF Config", che si concentra sulla configurazione di SVI, HSRP e OSPF per un routing e una ridondanza di rete efficienti.

```
{% 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 %}
```

Configurazione SVI_HSRP_OSPF di distribuzione

Questo modulo, Distribution_SVI_HSRP_OSPF_Config, aiuta a configurare le interfacce SVI, HSRP, OSPF e uplink per gli switch di distribuzione. Nell'esempio, ci stiamo concentrando sulla SVI per le subnet di dati, ma lo stesso metodo può essere utilizzato per altre SVI come la voce o la gestione.

Se la pianificazione degli indirizzi IP per le subnet di dati è già stata eseguita, gli indirizzi IP possono essere calcolati automaticamente per ciascuna SVI in base alle variabili branch_number e distribution_number. Ad esempio, se la diramazione 1 ha la subnet 172.17.0.0/20, la diramazione 2 ha 172.17.16.0/20, e la diramazione 3 ha 172.17.32.0/20, l'IP del gateway è 172.17.x.1 (dove x è il numero della diramazione). Il secondo IP per il primo switch di distribuzione è 172.17.x.2, il terzo IP per il secondo switch di distribuzione è 172.17.x.3. In questo modo, gli indirizzi IP vengono calcolati automaticamente, riducendo gli errori e semplificando il processo.

All'interfaccia di loopback viene assegnato un IP dalla variabile loopback_ip, che funge da ID router OSPF per garantire un routing stabile e coerente attraverso la rete. Nella configurazione OSPF, questo IP di loopback viene utilizzato come ID del router e le interfacce rilevanti vengono aggiunte all'area OSPF 0. Per HSRP, vengono impostati i valori di priorità riportati di seguito. 255 per il primo switch di distribuzione e 250 per il secondo, garantendo un failover corretto. Inoltre, l'autenticazione HSRP viene configurata utilizzando una catena di chiavi (HSRP_KEY) per migliorare la sicurezza.

Per mantenere la configurazione pulita e gestibile, alcuni valori sono hardcoded. Ad esempio, la subnet mask (255.255.240.0) e alcune impostazioni HSRP (come version e BFD) sono uguali in tutte le diramazioni, riducendo il numero di variabili. In questo modo la configurazione è più semplice, facile da applicare e meno soggetta a errori. Infine, le interfacce uplink vengono configurate con IP e aggiunte all'area OSPF 0 per consentire il corretto routing tra gli switch. Questo approccio rende il processo di configurazione più semplice da gestire e meno soggetto a errori, oltre ad essere flessibile per le diverse filiali.

Esaminare ora il modulo "Distribution ACL Config", che fornisce la segmentazione a livello di distribuzione.

```
{% 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 %}
```

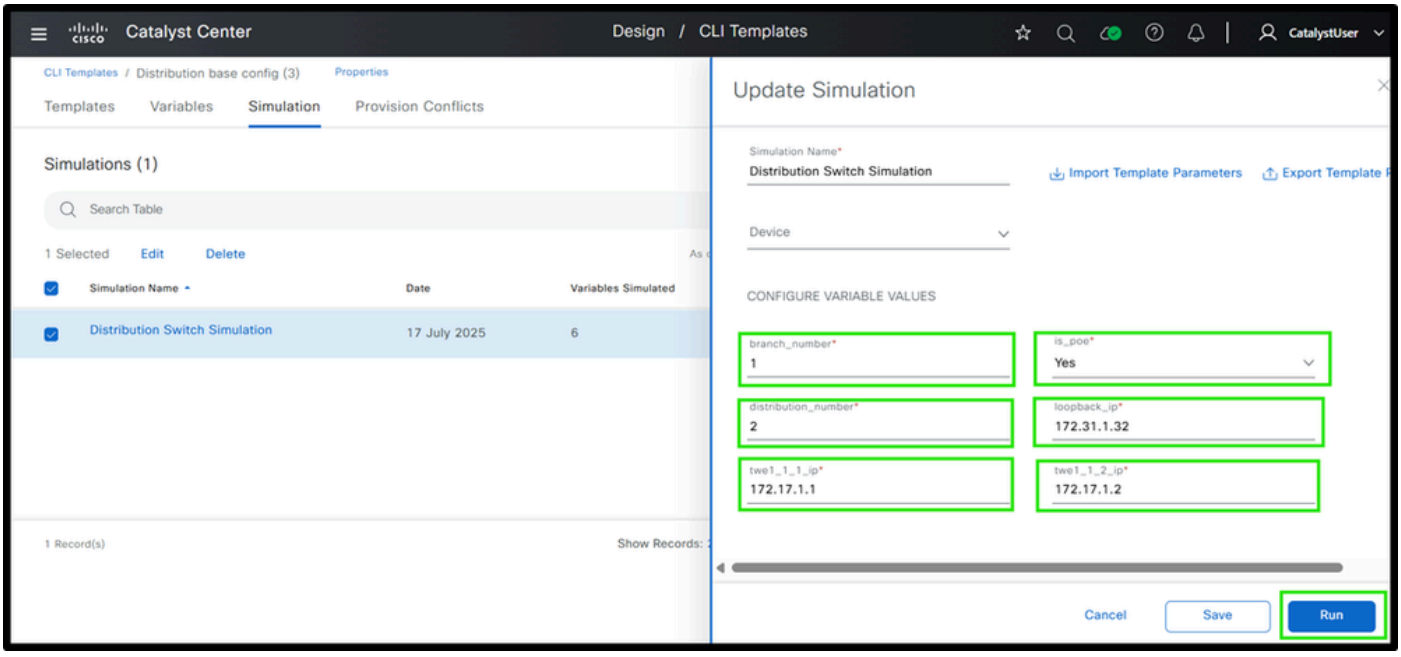
Configurazione ACL di distribuzione

In questo modulo viene illustrata la segmentazione a livello di distribuzione utilizzando un modello Jinja2. e utilizza la variabile branch_number per calcolare dinamicamente gli indirizzi delle subnet, abilitando configurazioni ACL automatizzate e scalabili. Per ciascuna diramazione, l'ACL blocca la comunicazione tra la subnet 1 (172.17.X.0) e la subnet 2 (172.16.X.0) impedendo il traffico IP tra

questi intervalli. Inoltre, nega il traffico diretto all'indirizzo multicast 239.255.255.250, autorizzando tutto il resto del traffico. L'interfaccia VLAN viene assegnata dinamicamente in base al numero di ramo e l'ACL viene applicato sul traffico in entrata sull'interfaccia. Questo approccio automatizzato garantisce un'efficace segmentazione per filiale, riduce gli errori di configurazione manuali e semplifica l'applicazione delle policy di rete.

Infine, l'ultimo modulo, "Distribution Standard Configuration" è quasi identico a quello descritto nel modulo Access Standard Configuration (per ulteriori informazioni, fare riferimento a tale sezione). Include best practice, protezione avanzata e funzionalità chiave per una gestione sicura dei dispositivi. L'unica differenza risiede nell'interfaccia di origine: nel modello Access Switch, è definita come VLAN {{ branch_number * 100 + 13 }}, mentre nella configurazione dello switch di distribuzione, può essere hardcoded come Loopback0.

Passaggio 3: Eseguire la simulazione prima di distribuire la configurazione.



(1) Input e output della simulazione del modello dello switch di distribuzione

(2) Input e output della simulazione del modello dello switch di distribuzione

(3) Input e output della simulazione del modello dello switch di distribuzione

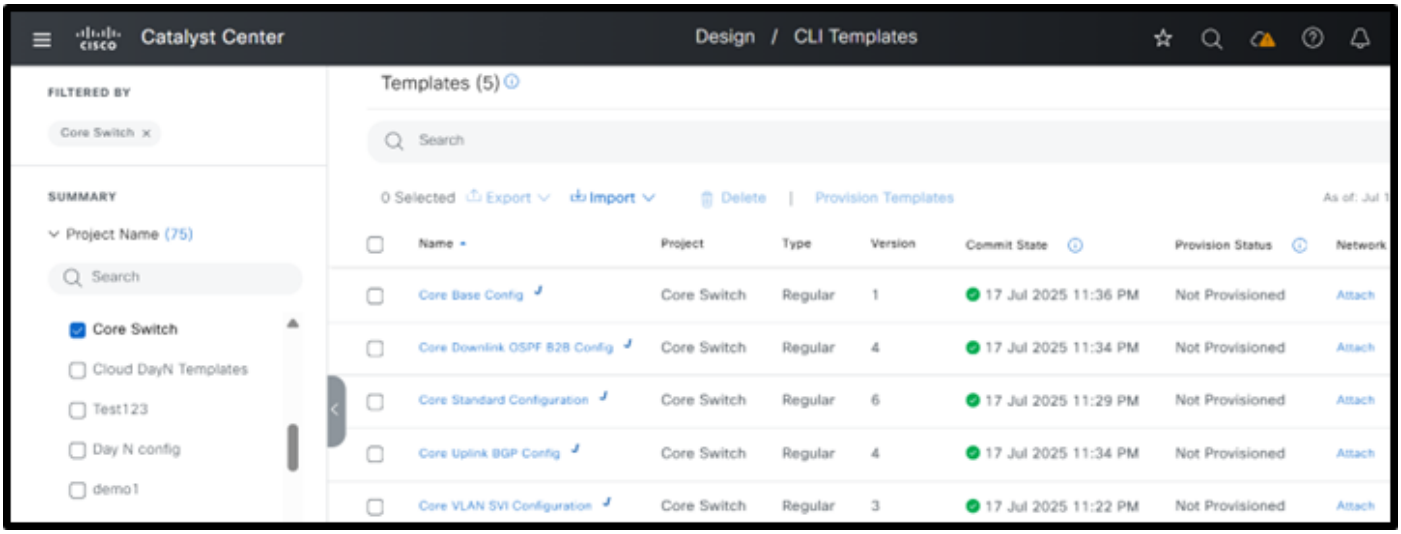
(4) Ingressi e uscite della simulazione del modello dello switch di distribuzione

In questo modo è possibile utilizzare i modelli a livello di distribuzione per generare le configurazioni. Ora, diamo un'occhiata ai dispositivi del livello principale per vedere come la modellazione può essere applicata lì.

Core Layer Switch

A questo punto è possibile progettare un modello modulare per gli switch principali. Il modello di base e i relativi moduli fanno parte del progetto 'Core Switch' in Cisco Catalyst Center. Vedere la maschera di base al passaggio 1.

Passaggio 1: Definizione della struttura dei vari switch di base



Struttura del modello di switch principale

Passaggio 2: Definire vari moduli

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

Configurazione base principale

La maggior parte delle configurazioni di switch di base sono simili in tutte le diramazioni, quindi i valori comuni possono essere hardcoded. In genere, vengono modificati solo gli indirizzi IP, che possono essere impostati tramite variabili. Poiché in genere ogni filiale dispone solo di due switch core, la gestione di queste variabili è semplice. Anche se alcune filiali hanno più switch core, il loro numero è comunque inferiore al numero di switch di accesso o distribuzione. Per questo motivo, come buona norma, è più importante ridurre al minimo le variabili per gli switch di accesso e distribuzione, in quanto vengono utilizzati in un numero maggiore di switch e la presenza di troppe variabili può rendere la configurazione più dispendiosa in termini di tempo.

Iniziare ora con il primo modulo: "Configurazione Core VLAN SVI". In questo esempio, i core switch sono posizionati dietro un firewall e devono stabilire il peering eBGP con esso. Questo modulo è responsabile della generazione delle VLAN e delle SVI corrispondenti richieste per il peer eBGP e il protocollo OSPF. Si presume che il firewall funzioni in una configurazione attiva/standby.

```
{% 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 %}
```

Configurazione Core VLAN SVI

Come spiegato in precedenza, questo modulo crea le VLAN richieste e le SVI associate per stabilire le relazioni adiacenti OSPF e BGP. Tutti i parametri, ad eccezione degli indirizzi IP delle SVI, sono hardcoded, inclusa la subnet mask se allineata al piano di indirizzamento IP. Questo metodo consente di limitare le variabili e di ridurre la possibilità di errori di configurazione.

Esaminiamo ora il modulo "Core Downlink OSPF B2B Config", che genera configurazioni per interfacce downlink, OSPF e collegamenti back-to-back tra Core Switch 1 e Core Switch 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 %}
```

Configurazione Core Downlink OSPF B2B

Analogamente al modulo precedente, la maggior parte dei valori di questo modulo sono hardcoded per ridurre al minimo il numero di variabili. Solo gli indirizzi IP delle interfacce di loopback e downlink sono variabili. Inoltre, i canali delle porte back-to-back e le VLAN sono standardizzati su diverse filiali.

Esaminiamo ora il modulo "Core Uplink BGP Config", che genera configurazioni BGP e gestisce uplink connessi ai firewall.

```
{% 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 %}
```

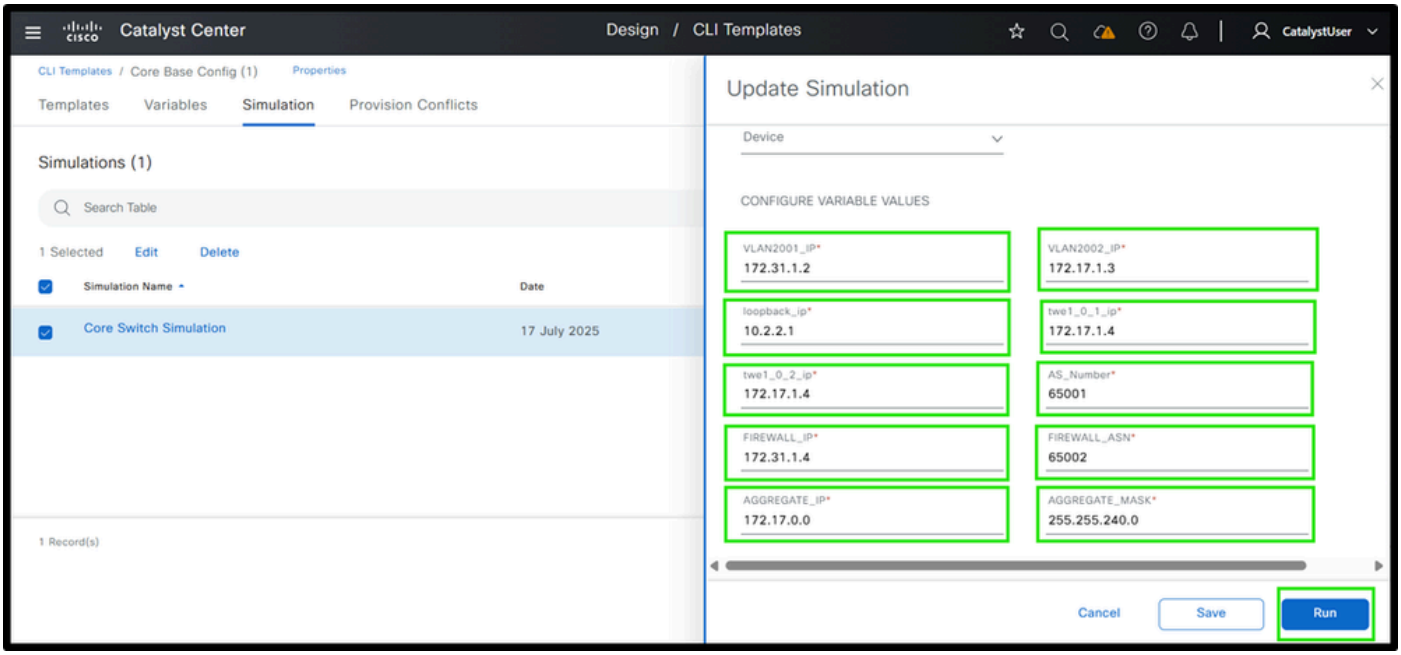
Configurazione Core Uplink BGP

Questo modulo genera la configurazione BGP necessaria per stabilire una relazione di tipo adiacente eBGP con il firewall. Come illustrato in precedenza, la maggior parte dei valori è

hardcoded, in quanto rimangono coerenti tra diramazioni diverse. Solo gli indirizzi IP e i numeri AS, che possono variare per ciascuna diramazione, vengono presi come variabili di input e utilizzati per generare la configurazione necessaria. La maggior parte delle altre impostazioni è stata standardizzata per ridurre al minimo il numero di variabili. Le interfacce uplink connesse al firewall vengono specificate insieme alla VLAN utilizzata per il protocollo eBGP adiacente, generato dal modulo precedente.

Infine, l'ultimo modulo, "Core Standard Configuration", è quasi identico a quello descritto in Access Standard Configuration (per ulteriori dettagli, fare riferimento a quella sezione). Include best practice, protezione avanzata e funzionalità chiave per una gestione sicura dei dispositivi. Coerentemente con la configurazione dello switch di distribuzione, anche in questo modulo l'interfaccia di origine può essere impostata su loopback0 e questo valore può essere hardcoded.

Passaggio 3: Esegui simulazione

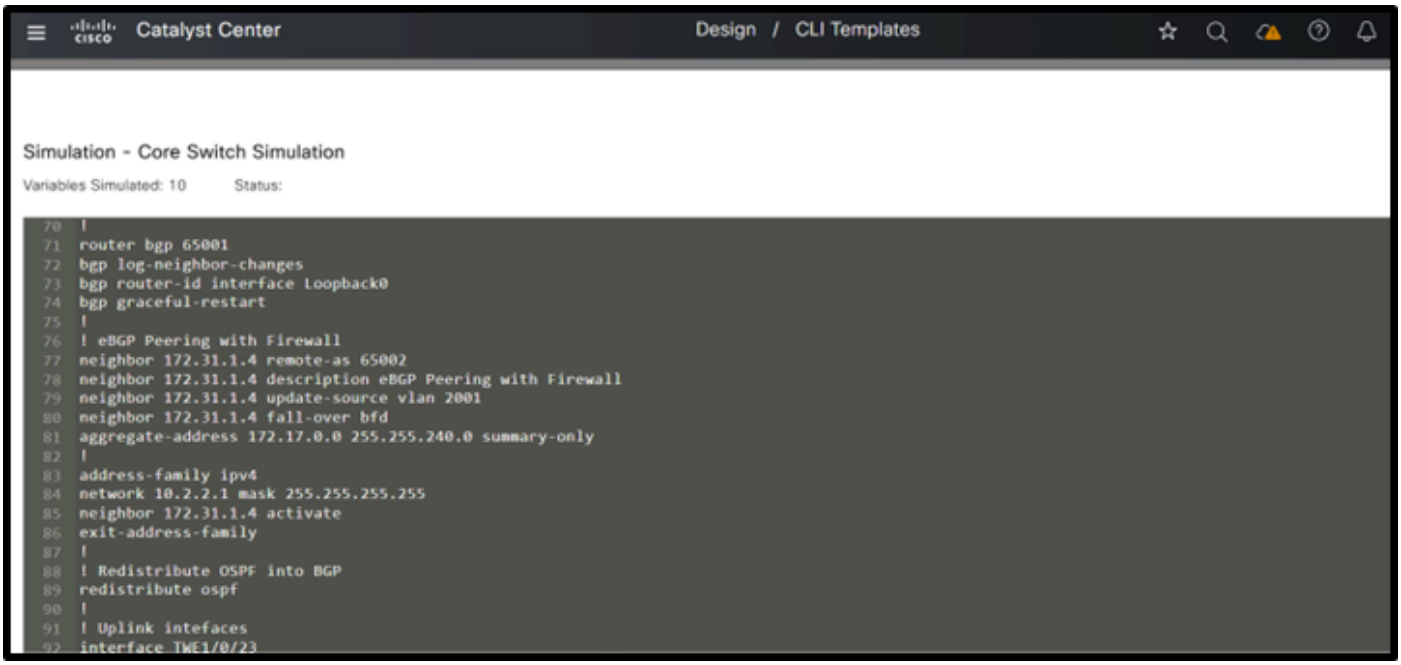


(1) Ingressi e uscite della simulazione del modello di switch core

(2) Ingressi e uscite della simulazione del modello di switch core

(3) Ingressi e uscite della simulazione del modello di switch core

(4) Ingressi e uscite di simulazione del modello di switch core



(5) Ingressi e uscite della simulazione del modello di switch core

In questo modo viene completata la spiegazione dettagliata della progettazione di modelli per l'architettura a tre livelli, delineando sia la struttura che la configurazione di ogni modulo.

Tutti questi moduli utilizzano le procedure ottimali descritte in precedenza.

Nota: Quando si progettano modelli per un'architettura di base compressa, fare riferimento alle spiegazioni fornite per l'architettura a tre livelli. La struttura del modello

---

rimane invariata; tuttavia, le feature che in precedenza erano implementate separatamente nei livelli core e distribuzione vengono ora combinate nel livello core compresso. È possibile utilizzare lo stesso approccio di modello modulare anche in questo caso, creando una maschera di base e facendo riferimento ai moduli rilevanti al suo interno.

Riepilogo

L'architettura tradizionale del campus a 3 livelli si basa spesso su un'ampia configurazione manuale a tutti i livelli di core, distribuzione e accesso. Questo approccio non è solo dispendioso in termini di tempo, ma è anche soggetto all'errore umano. L'assenza di automazione e di gestione centralizzata aumenta in modo significativo il sovraccarico operativo, rendendo difficile scalare e gestire in modo efficace le moderne reti universitarie dinamiche. Tramite Catalyst Center CLI le configurazioni delle funzionalità dei modelli possono essere automatizzate per le reti LAN tradizionali. È importante utilizzare l'approccio modulare durante il provisioning dei dispositivi. I moduli possono essere basati sulle varie funzioni utilizzate a diversi livelli. Infine, associare tutti i moduli al modulo di base.

Azioni da intraprendere

Invitiamo le organizzazioni ad adottare la metodologia dei modelli modulari presentata in questo white paper come una best practice per standardizzare le configurazioni degli switch e ottimizzare le operazioni di rete su architetture core sia a tre livelli che compresse.

- Implementando modelli modulari, i team di rete possono:
- Migliorare l'efficienza operativa attraverso procedure di configurazione coerenti e ripetibili.
- Riduzione al minimo dell'errore umano e dei tempi di risoluzione dei problemi.
- Maggiore scalabilità per supportare la crescita e l'evoluzione delle esigenze aziendali.
- Garanzia di coerenza della configurazione in ambienti diversi.

Questo approccio non solo semplifica la gestione quotidiana, ma consente anche installazioni più rapide, semplifica i cicli di aggiornamento e migliora l'allineamento con i requisiti di sicurezza e conformità. L'adozione di modelli modulari consente di posizionare la rete in modo da ottenere agilità, resilienza e successo a lungo termine in un panorama IT in continua evoluzione.

Per dimostrazioni pratiche, saperne di più sui modelli , vedi la serie su YouTube

1 Come creare e gestire i modelli in Catalyst Center

https://youtu.be/SyUqEEcwye0

2 Come utilizzare le variabili di binding di sistema nei modelli CLI in Catalyst Center

https://youtu.be/gV1QBuHYJdo

Autori

Naveen Kumar, Customer Delivery Architect, Cisco Customer Experience

Risabh Mishra, Consulente Tecnico, Cisco Customer Experience

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).