Templates op Catalyst Center begrijpen

Inleiding

Dit document beschrijft het Cisco Catalyst Center en de ervaring met configuratiesjablonen voor drielaags- of samengevouwen kerncampusarchitecturen.

Achtergrondinformatie

Dit document is bedoeld voor zakelijke professionals met een fundamenteel begrip van Cisco Catalyst Center en ervaring met configuratiesjablonen. Het is met name relevant voor degenen die hebben gewerkt of van plan zijn te werken met drielaags of ingestorte kerncampusarchitecturen.

Het belangrijkste doel is om lezers te helpen bij het implementeren en automatiseren van configuratie- en beheeroplossingen met behulp van sjablonen binnen Cisco Catalyst Center. Door geavanceerde inzichten, praktische technieken en praktijkvoorbeelden te presenteren, dient dit document als een praktische bron voor diegenen die hun LAN-infrastructuurvaardigheden willen verbeteren en workflows willen optimaliseren door automatisering en sjabloongebaseerd beheer.

samenvatting

Naarmate bedrijfsnetwerken blijven evolueren, is de behoefte aan schaalbaar, consistent en geautomatiseerd beheer nog nooit zo groot geweest. Cisco Catalyst Center biedt een gecentraliseerd, op opzet gebaseerd platform dat configuratie, provisioning en beveiliging in campusnetwerken vereenvoudigt. In deze whitepaper wordt onderzocht hoe netwerkprofessionals gebruik kunnen maken van de CLI Template Editor- en automatiseringsmogelijkheden van Cisco Catalyst Center om netwerkactiviteiten te stroomlijnen, configuratiefouten te verminderen en implementaties in drielaagse en ingestorte kernarchitecturen te versnellen. Het beschrijft best practices voor het ontwerpen van modulaire Jinja2-gebaseerde sjablonen, het integreren van automatisering in dag-0- en dag-N-workflows en het bereiken van operationele consistentie in de kern-, distributie- en toegangslagen. Door de in dit document beschreven strategieën te gebruiken, kunt u het traditionele handmatige netwerkbeheer transformeren in een flexibel, gestandaardiseerd en automatiseringsgestuurd model dat is afgestemd op de op opzet gebaseerde netwerkvisie van Cisco.

Uitdagingen van campusnetwerken

Naarmate campusnetwerken evolueren om aan de eisen van moderne organisaties te voldoen, worden ze geconfronteerd met verschillende belangrijke uitdagingen:

2 bis. Complexiteit in netwerkbeheer

Veel netwerkfuncties worden nog steeds handmatig beheerd, waardoor het risico op menselijke fouten toeneemt. Dit verhoogt niet alleen de onderhoudsinspanningen, maar ook de IT-middelen, vooral met statische of beperkte budgetten.

2 ter. Uitdagingen op het gebied van implementatie en automatisering

Het onboarden van nieuwe apparaten voor zowel bekabelde als draadloze netwerken is vaak tijdrovend en complex, wat leidt tot vertragingen in de implementatie en verhoogde administratieve overhead.

2 quater. Beheer van software-images

Het onderhouden van een consistent "gouden beeld" in het hele netwerk is een uitdaging. Veel netwerken hebben meerdere besturingssystemen voor bekabelde en draadloze apparaten, wat leidt tot inefficiëntie en beheerproblemen.

2d. Inconsistente netwerkconfiguraties

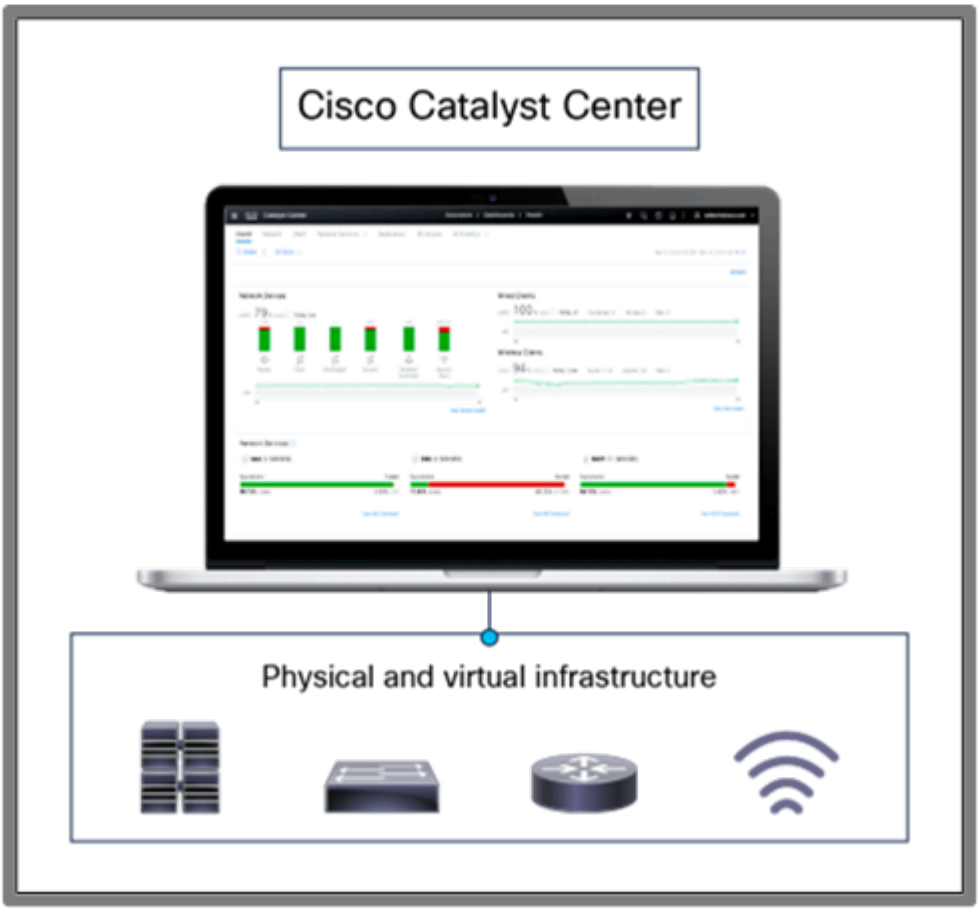
Variaties in netwerkconfiguraties kunnen leiden tot nalevingsproblemen en operationele inefficiënties, waardoor het moeilijker wordt om een betrouwbaar en veilig netwerk te onderhouden.
2 sexies. Stijgende gebruikersverwachtingen

Gebruikers eisen ononderbroken connectiviteit en naadloze toepassingservaringen, ongeacht hun locatie of apparaat. Om aan deze verwachtingen te voldoen, moeten netwerken veerkrachtig, intelligent en in staat zijn om zich aan te passen aan realtime veranderingen.

Naast deze uitdagingen worden moderne LAN-infrastructuren geconfronteerd met een verscheidenheid aan andere complexiteit.

Campusnetwerken vereenvoudigen met Cisco Catalyst Center

Cisco Catalyst Center is een gecentraliseerde netwerkbeheeroplossing voor campusnetwerken, die hoofdkantoren, vestigingen, bekabelde en draadloze verbindingen en IT/OT-omgevingen ondersteunt. Het biedt flexibele implementatieopties, waaronder fysieke apparaten, VMware ESXi-servers of AWS-cloud. Met uitgebreide functies vereenvoudigt Catalyst Center de bedrijfsvoering, verbetert het de prestaties en versterkt het de beveiliging.



Afbeelding 1: Infrastructuur beheren met Cisco Catalyst Center

Belangrijkste kenmerken en voordelen

Cisco Catalyst Center (CC) biedt geavanceerde functies voor het stroomlijnen van netwerkbeheer en automatisering:

- Zero-Touch Provisioning (ZTP): automatiseert de onboarding van apparaten, waardoor handmatige inspanningen en implementatietijd worden verminderd.
- Software Image Management (SWIM): zorgt voor consistente softwareversies op apparaten met controles vóór en na de upgrade om problemen te voorkomen.
- Intent-Based Automation: vereenvoudigt implementaties door netwerkintentie te vertalen naar apparaatconfiguraties voor bekabelde en draadloze netwerken.

LAN-automatisering: automatiseert Layer 3 IP-adressering en -routering om end-to-end topologieën te maken.

Draadloze netwerkautomatisering: functies zoals Plug and Play (PnP) maken snelle provisioning van draadloze toegangspunten mogelijk.

Hiërarchisch netwerkbeheer: maakt locatiespecifieke profielen (bijvoorbeeld SSID's, RF-parameters, VLAN's) mogelijk voor consistente implementaties op verschillende locaties.

CLI-sjablonen: met de Catalyst Center Template Editor kunnen beheerders eenvoudig CLI-gebaseerde configuratiesjablonen maken en beheren, waardoor een consistente en efficiënte implementatie op verschillende apparaten mogelijk is.

Assurance zorgt voor gecentraliseerde monitoring voor beheerde apparaten via CC.

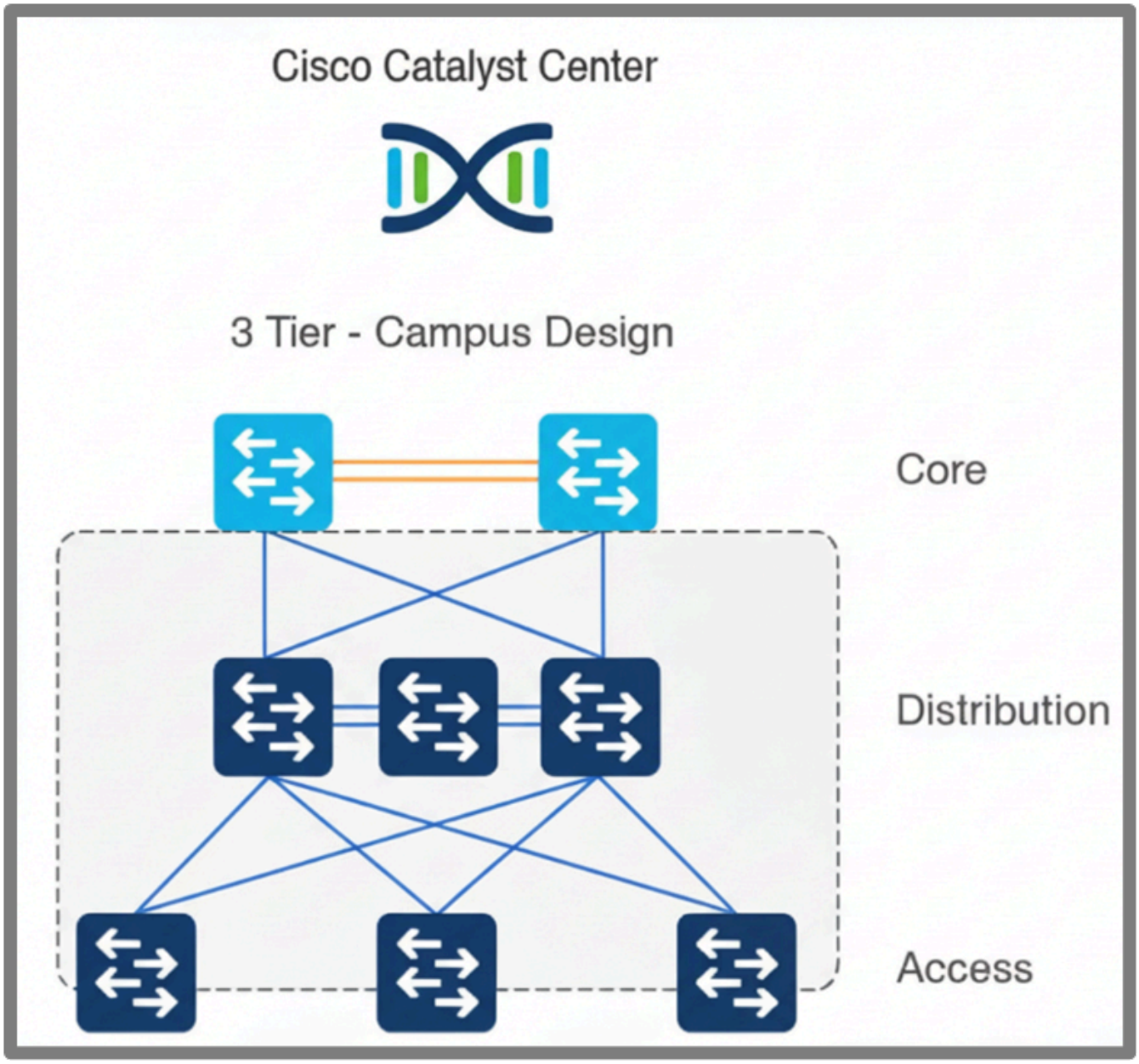
Naast deze functies biedt Cisco Catalyst Center nog veel meer functies die buiten het bereik van dit document vallen. Dit artikel richt zich voornamelijk op het ontwerpen van CLI-sjablonen met behulp van Catalyst Center.

Overzicht op hoog niveau van LAN Campus-architectuur met Catalyst Center

Traditionele LAN-campusnetwerken vormen de ruggengraat van zakelijke connectiviteit en zorgen voor betrouwbare en schaalbare communicatie voor bekabelde en draadloze apparaten. Deze netwerken zijn meestal ontworpen met behulp van de 3-tier architectuur of de Collapsed Core architectuur, afhankelijk van de grootte en complexiteit van de organisatie.

Drie lagen architectuur

De drielaagse architectuur is een fundamenteel netwerkontwerpmodel dat bestaat uit de kernlaag, de distributielaag en de toegangslaag. Deze architectuur biedt schaalbaarheid, hoge prestaties en efficiënt verkeersbeheer. Zie het overzicht van elke laag.



Afbeelding 2: Campusarchitectuur met drie lagen

kernlaag

De Core Layer vormt de ruggengraat van het netwerk en biedt snelle connectiviteit en schaalbaarheid. Belangrijke configuraties zijn onder meer noordelijke en zuidelijke routeringsprotocollen (zoals OSPF en BGP), routebeleid, configuraties voor downlink- en uplinkinterfaces, beveiligingsharding enz

distributielaag

De distributielaag vormt een brug tussen de kern- en toegangslagen en behandelt verkeersaggregatie, beleidshandhaving en redundantie. Belangrijke configuraties zijn onder meer HSRP/VRRP voor redundantie, STP voor luspreventie, Layer 2- en Layer 3-VLAN's,

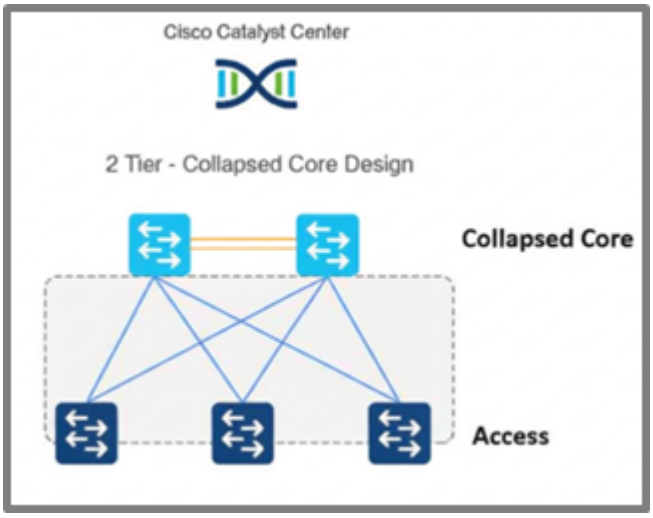
configuraties voor uplink- en downlink-interfaces, ACL's voor beveiliging en beveiligingshardening.

toegangslaag

De toegangslaag verbindt eindpunten met het netwerk en maakt veilige en betrouwbare toegang mogelijk. Belangrijke configuraties zijn onder meer configuratie van de toegangsinterface, configuratie van de uplinkinterface, Layer 2-VLAN's, ACL's voor het beperken van de toegang tot het apparaat en beveiligingsherstel.

Samengevouwen kernarchitectuur

De Collapsed Core-architectuur combineert de kern- en distributielagen in één laag, waardoor complexiteit en kosten worden verminderd en de prestaties en schaalbaarheid behouden blijven. Deze aanpak is zeer geschikt voor kleine tot middelgrote netwerken waarbij een aparte Core Layer niet nodig is. Bekijk het overzicht van de lagen in deze architectuur.



Afbeelding 3: Ingestorte kerncampusarchitectuur

Ingeklapte kernlaag

De Collapsed Core Layer combineert de functies van de kern- en distributielagen en biedt backboneconnectiviteit, verkeersaggregatie en beleidshandhaving. Belangrijke configuraties zijn onder meer noordgebonden en zuidgebonden routeringsprotocollen (zoals OSPF en BGP), routebeleid, downlink- en uplinkinterfaceconfiguraties, BFD voor foutdetectie, inter-VLAN-routering met behulp van SVI's, HSRP/VRRP voor gatewayredundantie, STP voor luspreventie en beveiligingsherstel. Door gebruik te maken van sjablonen in Cisco Catalyst Center kunnen deze configuraties worden geautomatiseerd, waardoor consistente en efficiënte implementaties worden gegarandeerd.

toegangslaag

Zoals eerder beschreven, verbindt de toegangslaag eindpunten met het netwerk, waardoor veilige en betrouwbare toegang mogelijk wordt. Belangrijke configuraties zijn onder meer configuratie van de toegangsinterface, configuratie van de uplinkinterface, Layer 2-VLAN's, ACL's voor het beperken van de toegang tot het apparaat en beveiligingsherstel.

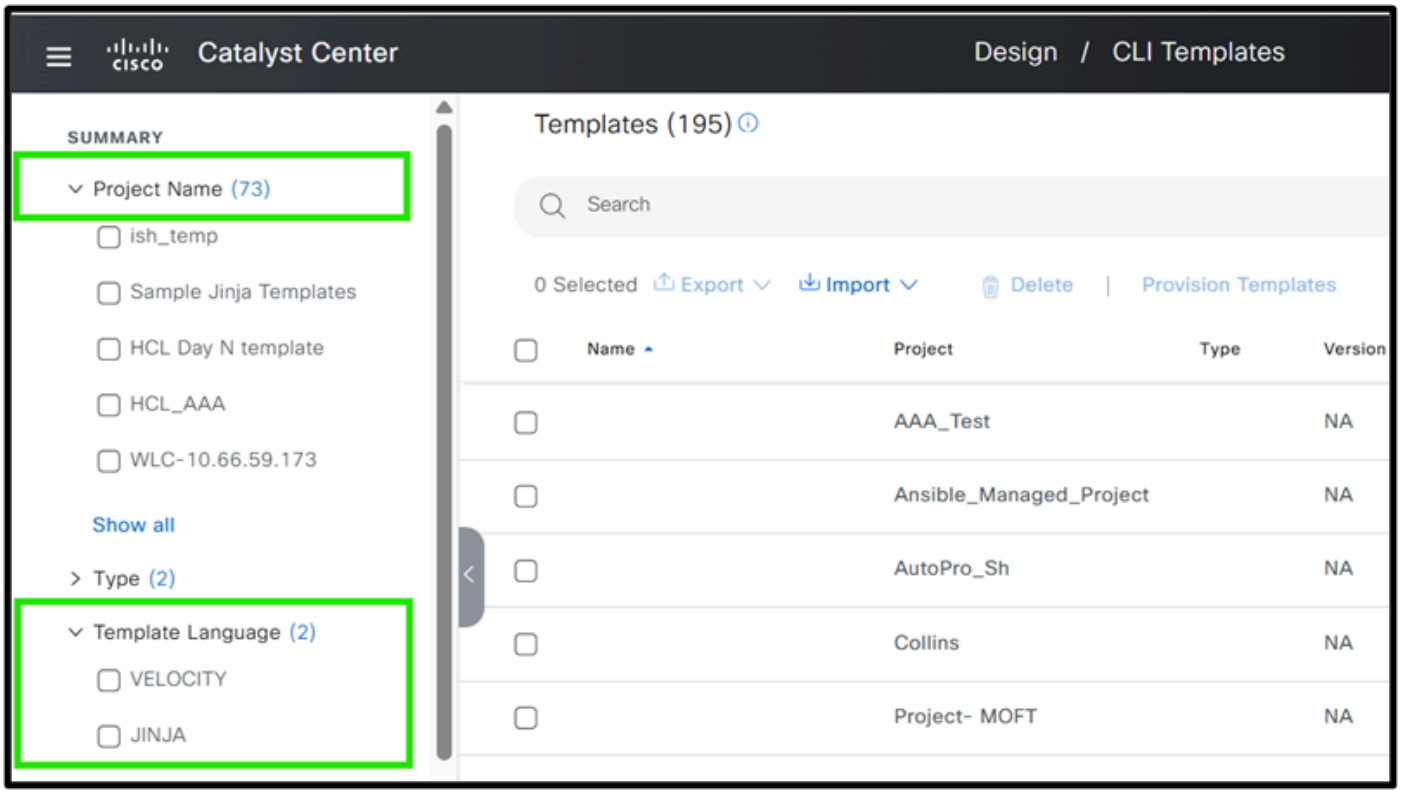
Overweging sjabloonontwerp

In dit gedeelte wordt beschreven hoe u sjablonen ontwerpt in Cisco Catalyst Center voor het genereren van apparaatconfiguraties. De sjablooneditor stroomlijnt de provisioning door het maken van herbruikbare CLI-sjablonen mogelijk te maken en de dynamische implementatie van configuraties op maat van uw netwerk te ondersteunen. Catalyst Center ondersteunt twee sjabloontalen: Jinja2 en Velocity. Deze talen helpen bij configuratiebeheer voor apparaten.

Jinja is een populaire, ontwerpvriendelijke sjabloontaal die voornamelijk wordt gebruikt met Python voor het genereren van dynamische inhoud zoals HTML, XML of andere op tekst gebaseerde indelingen. Hiermee kunnen variabelen en besturingsstructuren (zoals lussen en conditionals) in sjablonen worden ingebed om dynamische uitvoer te maken.

Apache Velocity is een op Java gebaseerde sjabloonengine die de Velocity Template Language (VTL) gebruikt om dynamische inhoud in verschillende documenten mogelijk te maken, waaronder webpagina's, XML of zelfs broncode. Het combineert gegevens van Java-objecten met sjablonen om de uiteindelijke uitvoer te produceren.

Dit document heeft alleen betrekking op Jinja2-sjablonen.



Afbeelding 4: Sjablooneditor Cisco Catalyst Center

In dit document gebruiken we Jinja2 vanwege zijn flexibiliteit. In plaats van een diepgaande verkenning van Jinja2, ligt de focus op praktische toepassing voor sjabloonontwerp. Voor meer informatie over Jinja2-sjablonen in Catalyst Center, raadpleegt u de link:

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

Voordat u zich verdiept in sjabloonontwerpstrategieën voor een Cisco-campusnetwerk, is het belangrijk om de belangrijkste best practices te gebruiken om efficiëntie en beheerbaarheid te garanderen bij het werken met sjablonen.

Sjabloonstructuur en best practices / Richtsnoer voor de beste strategie

Bij het automatiseren van de configuratie van netwerkapparaten met behulp van Cisco Catalyst Center is het essentieel om gestructureerde strategieën en best practices aan te nemen. Deze stappen zorgen voor consistentie, schaalbaarheid en beheergemak in uw netwerkinfrastructuur.

Configuratie verdelen op apparaatrol

Begin met het categoriseren van apparaten op basis van hun rol in de netwerktopologie. Gemeenschappelijke rollen zijn onder meer:

- kern
- verdeling
- toegang

Voorbeeld: een apparaat dat als Core-switch fungeert, moet andere configuratievereisten hebben dan een Access-switch.

Configuratie opdelen in modulaire blokken

Binnen elke apparaatrol moet u de configuratie opsplitsen in modulaire blokken door vergelijkbare functies of configuraties samen te groeperen. Deze modulaire aanpak vereenvoudigt automatisering, probleemoplossing en toekomstige updates.

Voorbeelden van een Core Device:

- OSPF-configuratieblok
- BGP-configuratieblok
- QoS-beleid blokkeren

Rol-agnostische configuratieblokken identificeren

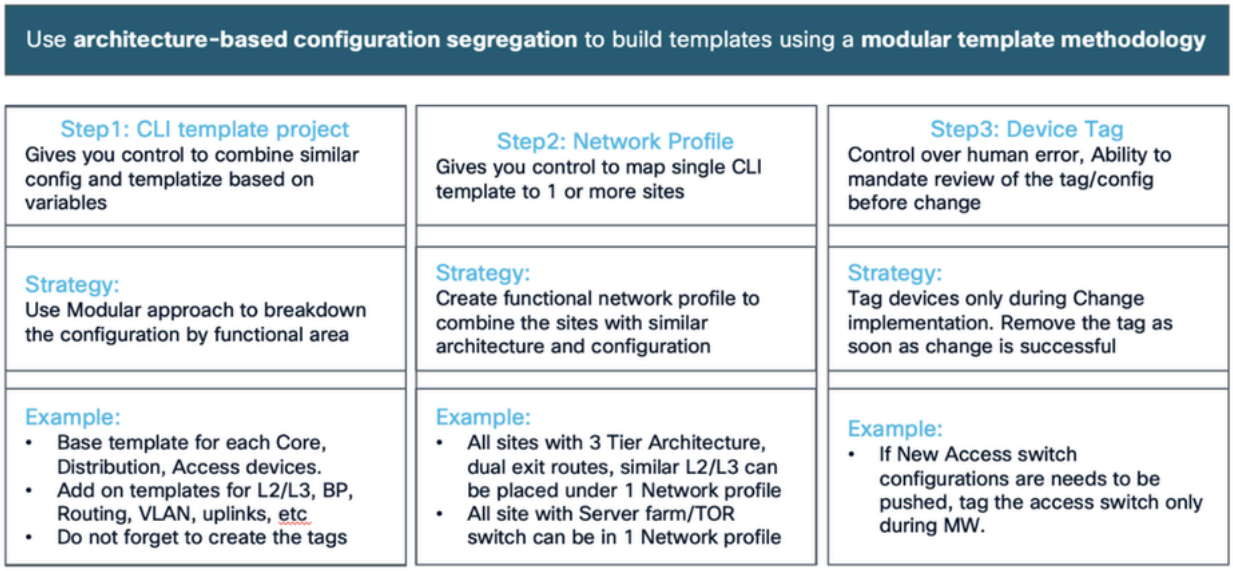
Sommige configuratieblokken zijn universeel van toepassing op alle apparaatrollen. Het identificeren en standaardiseren van deze blokken zorgt voor best practices en consistentie in het hele netwerk.

Common Role-Agnostic Configuration Blocks:

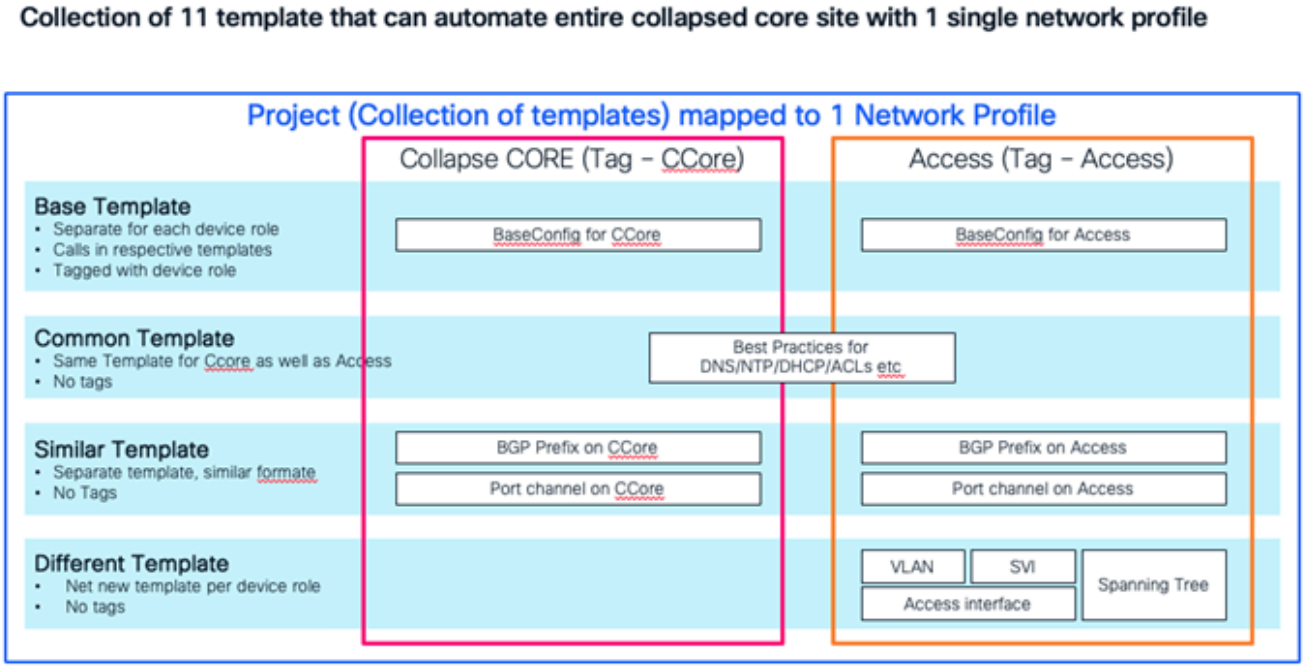
- Basisconfiguratie: hostnaam, aanmeldingsbanners
- Beheerprotocollen: DHCP, DNS, NTP, SNMP

Toegangsbeleid: standaardbeveiligingsconfiguraties

Deze blokken kunnen worden hergebruikt voor Core-, Distribution- en Access-apparaten, waardoor het automatiseringsproces wordt gestroomlijnd.



Figuur 1: Best practice met voorbeeld



Afbeelding 2: Voorbeeld van de samengevouwen kernsjabloon

Best practices voor het werken met sjablonen

Modulair sjabloonontwerp voor geautomatiseerde configuratie

Vermijd bij het automatiseren van apparaatconfiguraties in Cisco Catalyst Center het insluiten van alle configuraties in één monolithische sjabloon. Kies voor een modulaire aanpak:

Maak een basissjabloon die verwijst naar kleinere, doelspecifieke sjablonen (modules):

Verdeel de configuratie in logische modules (bijvoorbeeld interface-instellingen, routeringsprotocollen, beveiligingsfuncties).

Deze structuur maakt updates efficiënter: wijzigingen in een specifieke module worden automatisch weerspiegeld waar die module wordt gebruikt, waardoor fouten en complexiteit aanzienlijk worden verminderd.

Voorbeeld: Modulaire configuratie voor een vertakkingsapparaat

Stel dat u de configuratie voor een vertakkingsapparaat automatiseert.

Basissjabloon:

Bevat verwijzingen naar modulesjablonen voor belangrijke configuratiegebieden.

Doorgeeft variabelen zoals nodig aan elke module voor aanpassing.

Modulesjablonen:

interface_settings: Interfaceconfiguraties beheren.

routing_protocols: Bevat OSPF-, EIGRP- of BGP-instellingen.

security_features: definieert ACL's, firewallregels of ander beveiligingsbeleid.

```
{% 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) }}
```

Voorbeeld structuur basissjabloon:

Met deze structuur hoeven eventuele wijzigingen in routing- of beveiligingsconfiguraties alleen in hun respectieve modules te worden aangebracht en die wijzigingen worden onmiddellijk weerspiegeld waar de basissjabloon wordt gebruikt. Dit maakt uw configuraties beter beheersbaar en consistenter voor alle filiaalrouters.

Hier project naam is Branch en 3 andere verschillende modules zijn gedefinieerd onder project. Al deze worden gecombineerd in een basissjabloon.

Variabelen in de sjabloon minimaliseren

Houd het aantal variabelen in uw sjabloon tot een minimum beperkt om complexiteit en fouten te verminderen. Minder variabelen vereenvoudigen de implementatie, vooral in grote netwerken, waardoor het proces efficiënter en consistenter wordt.

Apparaattags gebruiken voor sjablonen

Maak gebruik van apparaattags in Cisco Catalyst Center, zoals locatie, rol of site, om dynamische en schaalbare Jinja2-sjablonen te maken. Deze tags maken voorwaardelijke logica mogelijk en zorgen ervoor dat de juiste configuraties op de juiste apparaten worden toegepast. Deze aanpak minimaliseert fouten en vereenvoudigt sjabloonbeheer in verschillende netwerkomgevingen.

Statische hardcodewaarden waar mogelijk

Statische waarden voor hardcodering kunnen sjablonen vereenvoudigen en de implementatie efficiënter maken. Veelvoorkomende voorbeelden zijn IP-adressen voor DNS-, NTP- of Syslog-servers, die doorgaans consistent blijven op verschillende apparaten. Ook het gebruik van standaard VLAN ID's op access switches maakt het mogelijk deze waarden worden hardcoded, het verminderen van variabiliteit en het versnellen van de implementatie.

Kies een tweefasenaanpak: sjablonen voor dag 0 en dag N

Gebruik bij het onboarden van apparaten met services zoals Cisco Plug and Play een sjabloonstrategie in twee stappen:

Sjablonen voor dag 0: Druk op basisconfiguraties om ervoor te zorgen dat het apparaat kan communiceren met het Cisco Catalyst Center.

Day N-sjablonen: implementeer geavanceerde functies en configuraties zodra het apparaat bereikbaar is.

Best practices maken efficiënte en schaalbare sjablonen mogelijk die de implementatie van Cisco-campusnetwerken vereenvoudigen.

Whitespace-besturing in Jinja-sjabloonmacro's

Bij het maken van sjablonen met behulp van de Jinja-taal is het essentieel om witruimte en nieuwe regels zorgvuldig te behandelen, vooral bij het renderen van dynamische inhoud in macro's. Gecumuleerde witruimte of onbedoelde nieuwe lijnen kunnen leiden tot formatteringsproblemen in de gegenereerde uitvoer, die een verkeerde interpretatie of fouten in de stroomafwaartse verwerking moeten veroorzaken. Om dit aan te pakken, biedt Jinja syntaxis voor het regelen van witruimte: het plaatsen van een minteken (-) direct in de scheidingstekens ({- ... -} of {%- ... -%}) het strippen van elke leidende of achtervolgende witruimte rond de uitdrukking. Bijvoorbeeld, het vervangen van {{item[1]} door {{- item[1] -}} zorgt ervoor dat eventuele extra spaties of nieuwe lijnen worden verwijderd wanneer de macro wordt gerenderd. Deze oefening is vooral handig bij het herhalen van lijsten of het genereren van configuratiebestanden, zoals weergegeven in het sjabloonfragment. We raden aan om witruimteregeling altijd toe te passen in dergelijke scenario's om schone en voorspelbare uitgangen te behouden.

Voorbeeld (aanbevolen gebruik):

```
{% voor object in wildcard_list %}
   {% if item[0] == prefix -%}
      {{- item[1] -}}
   {%- endif %}
{%- end voor %}
```

Drie lagen architectuur

Deze whitepaper begint met het ontwikkelen van templates voor toegang tot switches tot de core switches en schetst de eisen voor elke laag.

Switches voor toegangslaag

Access-switches zijn aan boord met Plug and Play en moeten een Day 0-sjabloon hebben. Voor meer informatie over het Plug and Play-proces in Catalyst Center, raadpleegt u de 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

Zoals eerder besproken, ondersteunt Catalyst Center zowel Velocity- als Jinja2-sjabloontalen. Dit document maakt gebruik van Jinja2 om de sjabloonstructuur te illustreren, vanwege de flexibiliteit. De configuratie van de Access Layer switch kan worden geïmplementeerd met behulp van de Day-0- en Day-N-sjabloon.

Een standaard sjabloon voor dag 0 kan worden gestructureerd, zie stap 1:

Stap 1: sjabloon definiëren

```
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
!
```

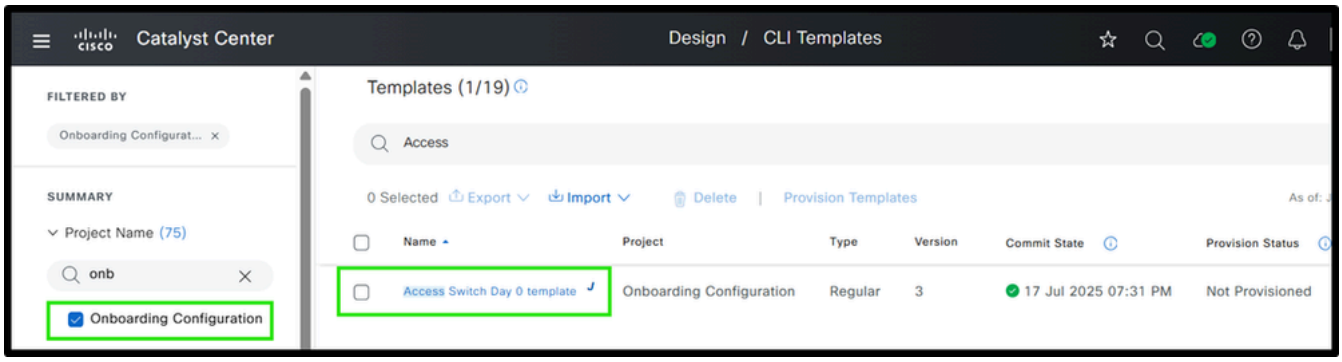
Stap 1: sjabloon definiëren

De sjabloon vereenvoudigt de configuratie door hardcoderingsconstanten zoals gebruikersnaam, wachtwoord, geheim inschakelen en subnetmasker te gebruiken, omdat alle switches in een filiaal hetzelfde beheer-VLAN-subnetmasker delen. Het IP-adres voor beheer is echter voor elke switch uniek en wordt gedefinieerd als een variabele. Een uitgebreide sjabloonstructuur moet worden verstrekt in de Day N-sjabloon, die deze Day 0-sjabloon

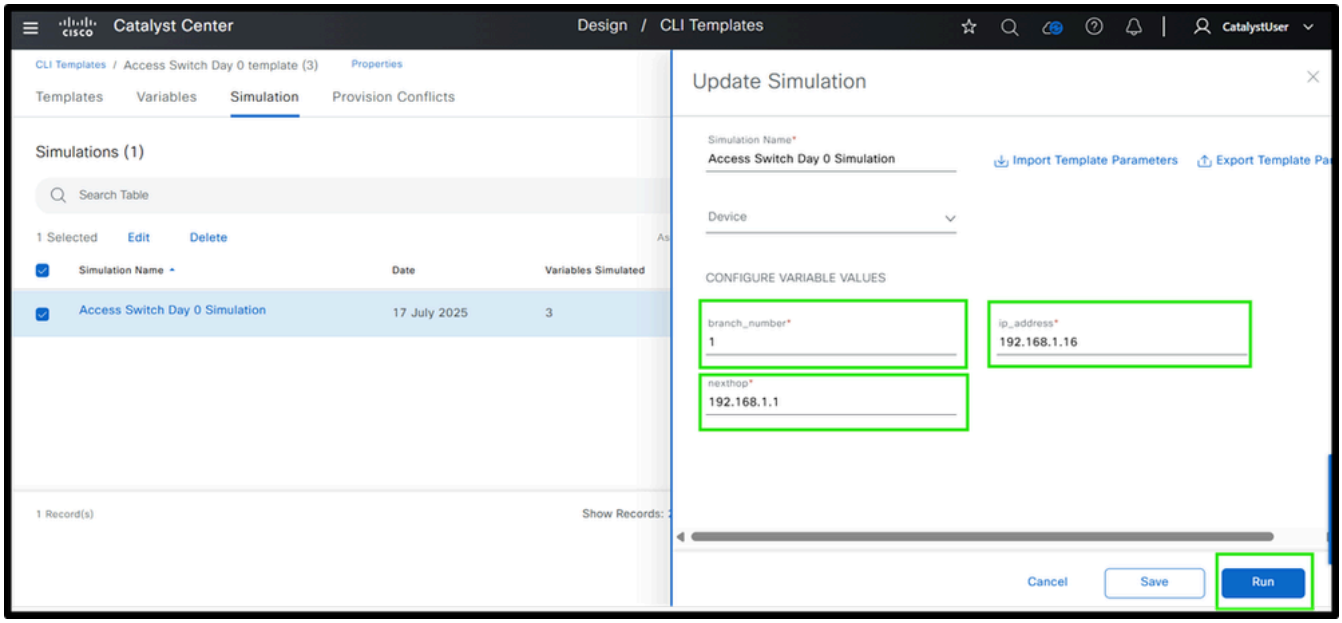
gebruikt. In de Day N-template wordt elke functie van de toegangsmodule beheerd door een speciale switch, bijvoorbeeld één module behandelt Layer 2 VLAN's, aparte modules beheren uplink- en downlink-toegangsinterfaces, een andere module richt zich op beveiligingsherstel, enzovoort.

Terwijl consistente VLAN-ID's de voorkeur hebben, kunnen dynamische variabele ID's worden gegenereerd met behulp van een formule op basis van het filiaalnummer (bijvoorbeeld Branch 1 = VLAN 113, Branch 2 = VLAN 213). Dit maakt de sjabloon herbruikbaar voor alle branches. Evenzo is de next-hop IP een variabele, omdat deze per tak moet verschillen, afhankelijk van het aangesloten distributiecluster.

Stap 2: Simulatie uitvoeren en variabele opgeven



Toegang tot de Switch Dag 0-sjabloonstructuur met simulatie-ingangen en -uitgangen



Ex. Simulatie-ingangen

Het wordt altijd aanbevolen om de sjabloon te simuleren voordat deze wordt geïmplementeerd. De screenshot toont de uiteindelijke configuratie na het invoeren van de

variabelen.

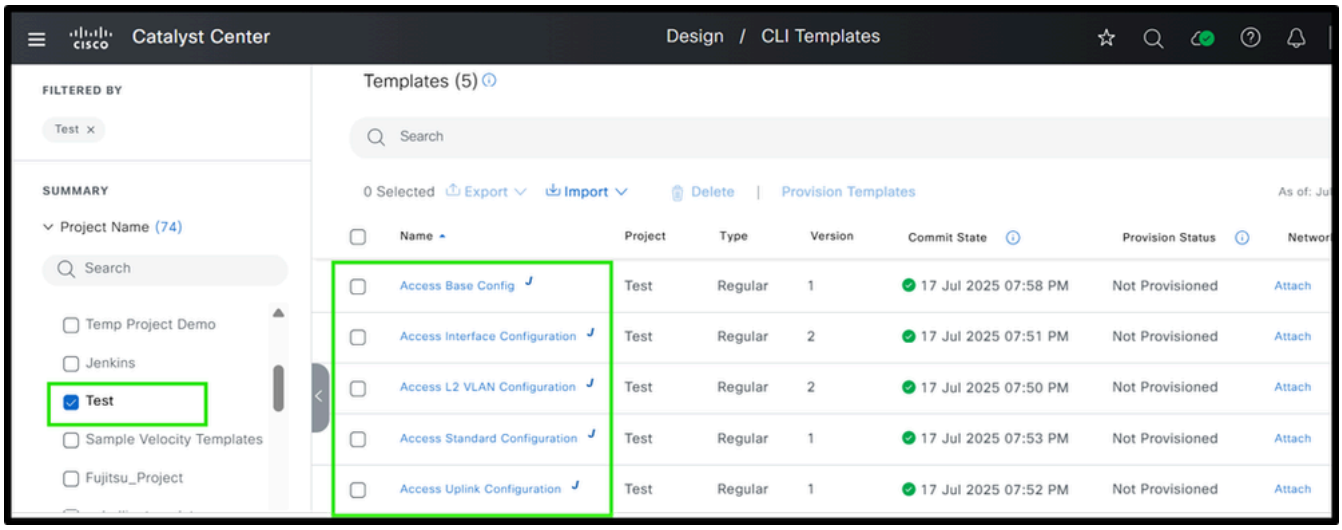
Definitieve configuratie na het invoeren van waarden

Laten we nu eens kijken naar hoe u een modulaire Day N-sjabloon kunt maken.

De toegangsmodules kunnen worden onderverdeeld in verschillende switches, die allemaal kunnen worden gecombineerd in een basismodule. De basissjabloon voor toegang switches is gestructureerd zoals weergegeven.

Zowel de basissjabloon als de bijbehorende modules worden gemaakt binnen een project met de naam "Test" in Cisco Catalyst Center.

Stap 1: Definieer verschillende sjablonen, waaronder Basissjabloon



Stap 2: Definieer verschillende modules

Access Base Config:

De screenshot toont een voorbeeld van de basisconfiguratie.

```
{% 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) }}
```

Access Base Config

Deze modulaire configuratiesjabloon bestaat uit vier onderdelen: VLAN-configuratie, uplinkinterfaceconfiguratie, toegangsinterfaceconfiguratie en standaardconfiguratie. Het gebruikt slechts twee variabelen: branch_number en is_poe, waardoor het eenvoudig en gemakkelijk te beheren is.

De branch_number berekent branch-specifieke VLAN ID's, zoals weergegeven in de Day 0 template, en is_poe bepaalt of de access switch een PoE of niet-PoE switch is. Deze variabelen worden verstrekt tijdens de provisioning en doorgegeven aan de modules om de juiste configuraties te maken, waardoor de inspanning wordt verminderd en de efficiëntie wordt verbeterd.

Laten we nu elke module bekijken om te zien hoe ze bijdragen aan het genereren van specifieke delen van de algemene configuratie.

Toegang tot L2 VLAN-configuratie

```
{% 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 %}
```

Toegang tot L2 VLAN-configuratie

Deze module maakt VLAN's op basis van het filiaalnummer, zoals eerder uitgelegd. VLAN's voor gegevens en spraak worden gemaakt op alle switches, of ze nu PoE ondersteunen of niet. Het AP Management VLAN (bijvoorbeeld 114 voor tak 1) wordt alleen gemaakt als is_poe is ingesteld op "Ja", wat betekent dat de switch PoE ondersteunt. Als is_poe "Nee" is, wordt het AP Management VLAN genegeerd omdat niet-PoE-switches geen toegangspunten kunnen ondersteunen. Dit wordt beheerd met behulp van een if-voorwaarde.

```
{% 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 %}
```

Configuratie van toegangsinterface

Deze module behandelt de toegangsinterfaceconfiguratie en maakt gebruik van dezelfde aanpak als de eerder beschreven PoE-switch. Als de variabele is_poe "Ja" is, wat betekent dat de switch een PoE-switch is, moeten de eerste zes poorten (1-6) worden geconfigureerd met het AP-beheer-VLAN. Anders moeten de eerste zes poorten worden ingesteld als toegangspoorten voor gebruikers.

Ervan uitgaande dat de switch een model met 24 poorten is, worden de resterende poorten (7-24) altijd geconfigureerd als toegangspoorten voor gebruikers, ongeacht of de switch PoE is of niet.

Het interfacebereik is gestandaardiseerd en wordt niet langer beschouwd als een invoervariabele, wat wordt beschouwd als een beste praktijk om het aantal variabelen in de template te minimaliseren. Bovendien bevat de module een macro met de naam common_access_settings, die de sjabloongrootte minimaliseert door herhaalde configuraties te consolideren. Deze macro wordt gewoon binnen de interface-instellingen genoemd,

waardoor ze niet meerdere keren hoeven te worden opgegeven.

Opmerking: deze sjabloon past dezelfde beschrijving toe op alle toegangsinterfaces. Als er voor elke interface unieke beschrijvingen nodig zijn, is het raadzaam om ze te pushen met behulp van afzonderlijke Python-scripts of vergelijkbare automatiseringstools.

Controleer de module die configuraties voor uplinkinterfaces genereert.

```
{% 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 %}
```

Uplinkconfiguratie openen

Deze module genereert de configuratie voor uplinkinterfaces en verwerkt het snoeien van VLAN's. Als de switch PoE ondersteunt, wordt het AP Management VLAN opgenomen in de lijst met toegestane VLAN's. Anders wordt het uitgesloten. Deze logica wordt beheerd met de voorwaarde if in de code, zoals eerder beschreven.

Bekijk de laatste module, die standaardconfiguraties demonstreert, inclusief best practices en beveiligingsherstel.

Let op: dit is alleen ter illustratie en mag niet worden gebruikt als referentie voor daadwerkelijke netwerkinstellingen, omdat configuraties kunnen variëren op basis van specifieke vereisten

---

```
{% 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
```

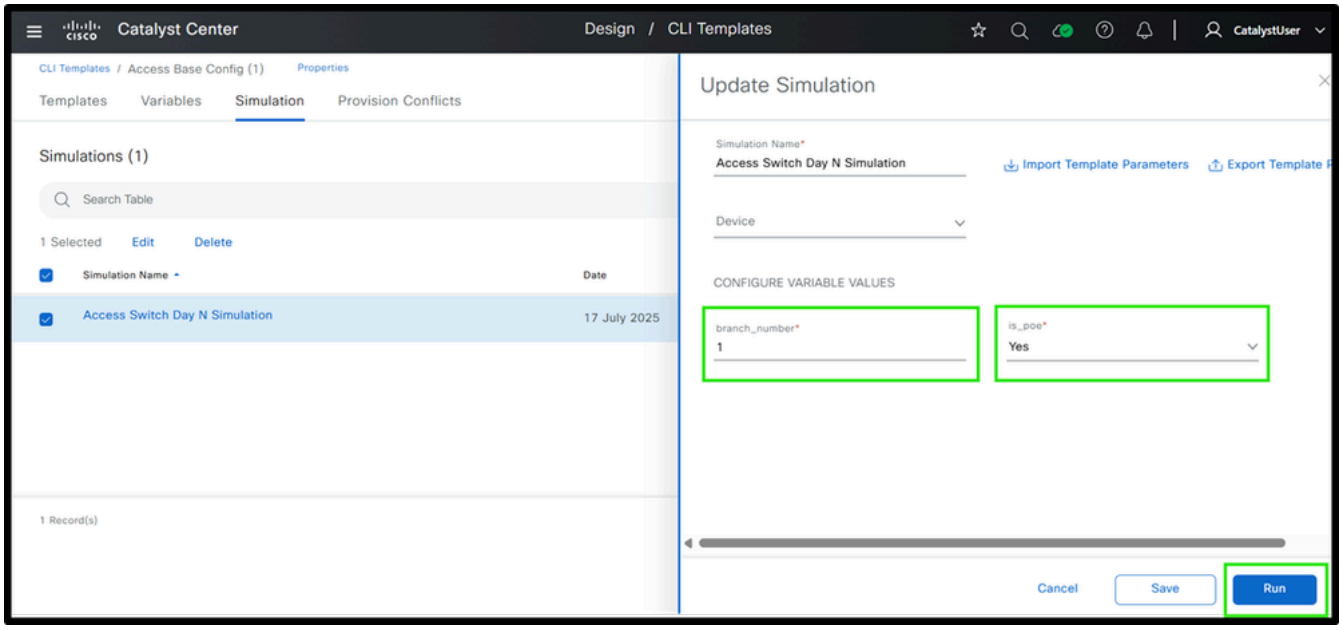
Deel 1: Toegang tot standaardconfiguratie

```
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 %}
```

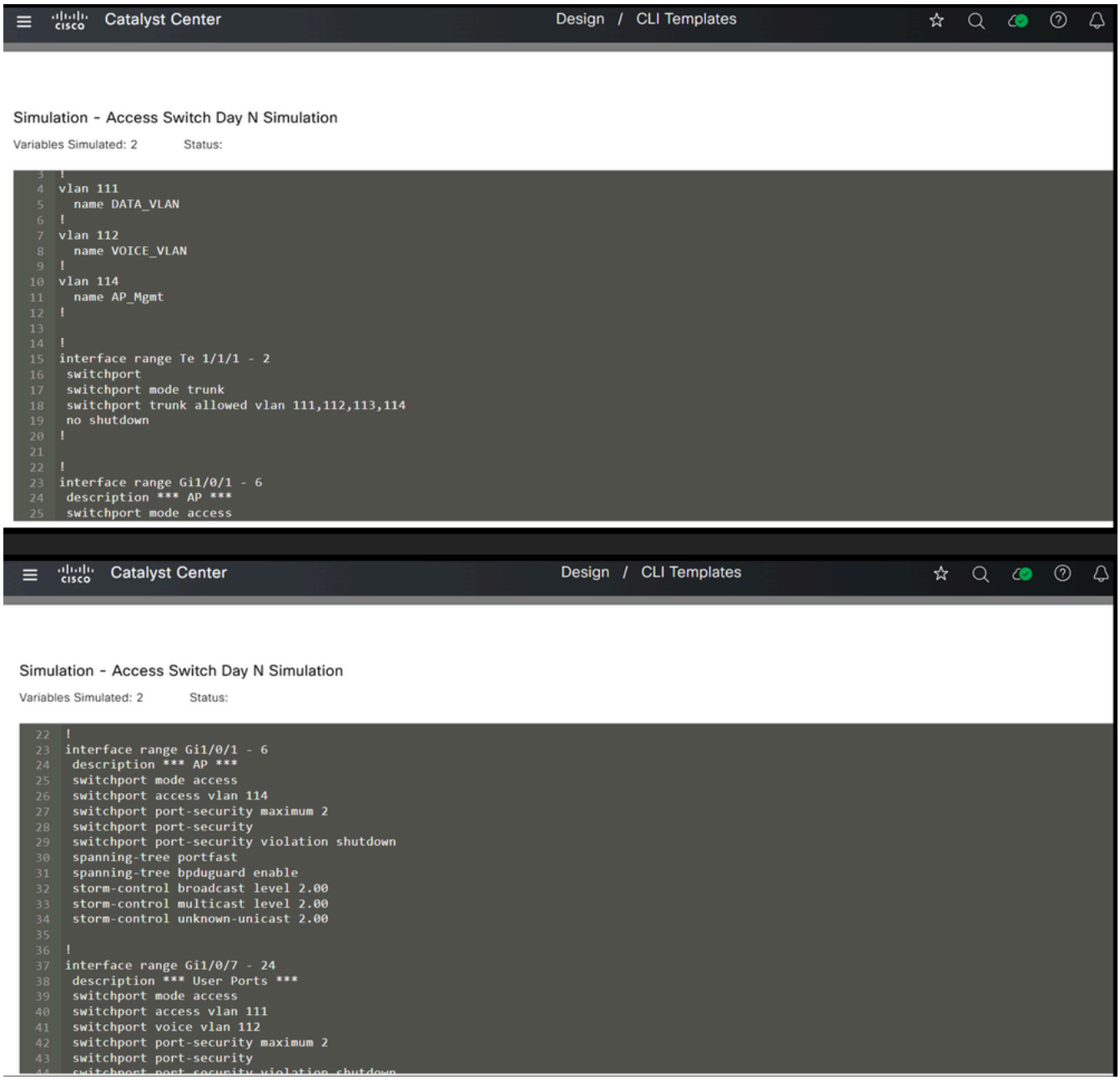
Deel 2: Toegang tot standaardconfiguratie

Deze module genereert een standaardconfiguratie met best practices, beveiligingshardening en belangrijke functies voor veilig apparaatbeheer. De meeste waarden zijn gehardcodeerd voor consistentie tussen branches, behalve branch_number, dat wordt gebruikt om het beheer-VLAN voor switches in elke branch te berekenen en dient als de broninterface voor verschillende configuraties.

Stap 3: Voer een simulatie uit voordat u de switches configureert. Alleen de basisconfiguratie moet worden gesimuleerd.



Afbeelding 7: In- en uitvoer van Access Switch Day N-sjabloonsimulatie



simulatie

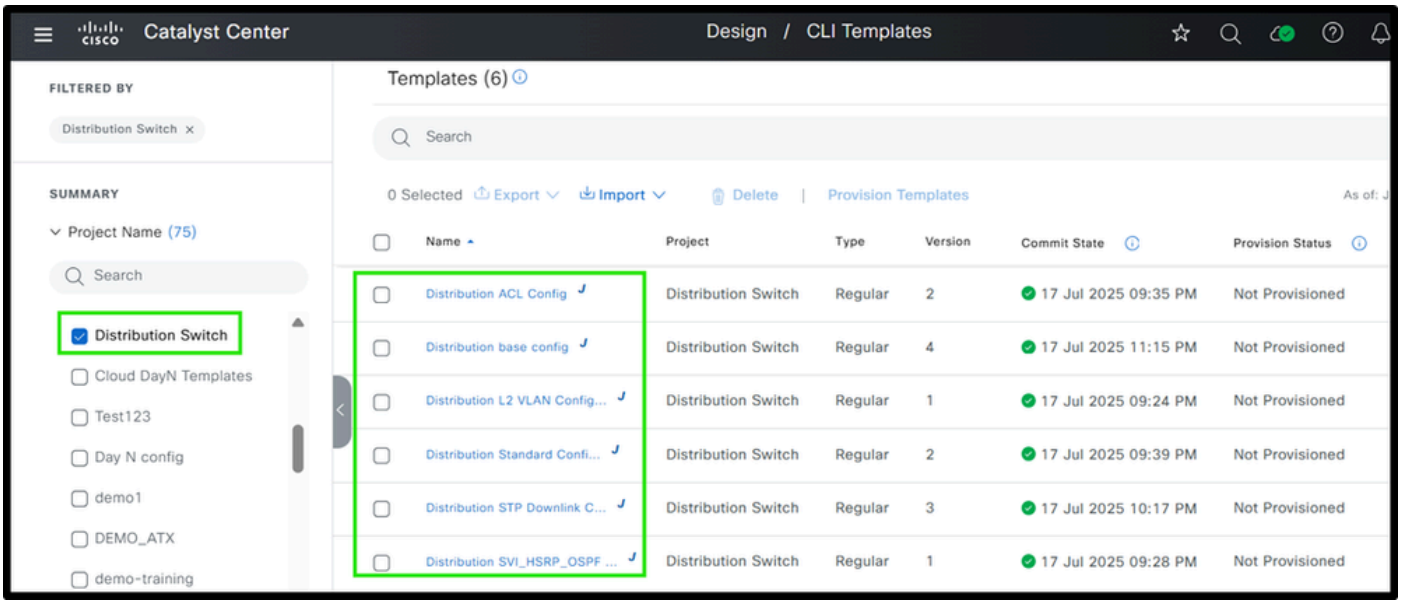
Zo kunnen sjablonen worden gebruikt op de toegangslaag om configuraties te genereren.

Laten we nu eens kijken naar de distributielaagapparaten om te zien hoe sjablonen erop kunnen worden toegepast.

Distributielaag Switches

Nu om een modulaire sjabloon voor de verdeling switches te ontwerpen. De basissjabloon en de bijbehorende modules maken deel uit van het project van de 'Distribution Switch' in Cisco Catalyst Center.

Stap 1: Distributie Switch Template Structure



Ex. distributiesjablonen

Stap 2: Definieer elke module

De aangeboden basisconfiguratie definieert alle modules waarnaar wordt verwezen.

```
{% 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. Distributiebasissjabloonmodules

Net als bij Access-switches worden alle templates gemaakt binnen het project 'Distributie Switch' en waarnaar wordt verwezen in de basistemplate. Hoewel sommige templates identiek zijn aan de templates die worden gebruikt voor access switches, worden in dit gedeelte de specifieke verschillen voor distribution switches uitgelegd. De module "Distribution L2 VLAN Configuration" is gelijk aan de eerder beschreven module voor access switches. Controleer de Access L2 VLAN Configuration-module die deze informatie biedt. Het genereert de vereiste VLAN's op basis van de invoerwaarden die voor de variabelen zijn opgegeven.

Bekijk nu de "Distribution STP Downlink Config"-module, die de productie van Spanning Tree- en uplinkconfiguraties voor distributie-switches regelt.

```
{% 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 %}
```

Distributie STP Downlink Config

Hier wordt Jinja2 macrofunctionaliteit gebruikt, waarnaar wordt verwezen in de op basis van module. Deze structuur helpt bij het bouwen van een modulaire aanpak.

Deze module configureert Spanning Tree Protocol (STP)- en downlink-interfaces op basis van het "branch_number" en of de switch PoE-enabled is. De variabele "branch_number" wordt gebruikt om voor elke branch unieke basis-VLAN's te genereren, waardoor verschillende VLAN's worden gegarandeerd, vergelijkbaar met de benadering die al is gemarkeerd voor access-switches. Als de switch PoE-enabled is ("is_poe" == 'Yes'), wordt een extra VLAN, zoals het AP Management VLAN, aan de lijst toegevoegd. De variabele "distribution_number" bepaalt de STV-prioriteit en stelt 4096 in voor Distributie 1 (waardoor het de voorkeurswaarde is voor root-brug) en 8192 voor secundaire distributie-switches. Ten slotte worden de juiste VLAN's toegepast op de hoofdinterface, zodat alleen de betreffende VLAN's zijn toegestaan op basis van de vraag of de switch PoE-enabled is.

Bekijk nu de module "Distributie SVI_HSRP_OSPF Config", die zich richt op de installatie van SVI's, HSRP en OSPF voor efficiënte netwerkroutering en redundantie.

```
{% 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 %}
```

Distributie SVI_HSRP_OSPF-configuratie

Deze module, Distribution_SVI_HSRP_OSPF_Config, helpt bij het configureren van SVI's, HSRP, OSPF en uplink-interfaces voor distributie-switches. In dit voorbeeld richten we ons op de SVI voor gegevenssubnetten, maar dezelfde methode kan worden gebruikt voor andere SVI's zoals spraak of beheer.

Als de IP-adresplanning voor gegevenssubnetten al is voltooid, kunnen de IP-adressen automatisch voor elke SVI worden berekend op basis van de variabelen branch_number en distribution_number. Bijvoorbeeld, als tak 1 het subnet 172.17.0.0/20 heeft, tak 2 heeft 172.17.16.0/20, en tak 3 heeft 172.17.32.0/20, is de gateway IP 172.17.x.1 (waarbij x het filiaalnummer is). Het tweede IP voor de eerste switch is 172.17.x.2, en het derde IP voor de tweede switch is 172.17.x.3. Op deze manier worden de IP-adressen automatisch berekend, waardoor fouten worden verminderd en het proces wordt vereenvoudigd.

De loopback-interface krijgt een IP toegewezen van de variabele loopback_ip, die dient als de OSPF-router-ID om een stabiele en consistente routering over het netwerk te garanderen. In de OSPF-configuratie wordt deze loopback-IP gebruikt als de router-ID en worden de relevante interfaces toegevoegd aan OSPF-gebied 0. Voor HSRP worden prioriteitswaarden vastgesteld: 255 voor de eerste distributie-switch en 250 voor de tweede, waardoor een goede failover wordt gegarandeerd. Bovendien wordt HSRP-verificatie geconfigureerd met behulp van een sleutelketen (HSRP_KEY) om de beveiliging te verbeteren.

Om de configuratie schoon en beheersbaar te houden, zijn sommige waarden hardcoded. Het subnetmasker (255.255.240.0) en bepaalde HSRP-instellingen (zoals versie en BFD) zijn bijvoorbeeld hetzelfde in alle branches, waardoor het aantal variabelen wordt verminderd. Dit maakt de configuratie eenvoudiger, gemakkelijker toe te passen en minder kans op fouten. Ten slotte worden de uplinkinterfaces geconfigureerd met IP's en toegevoegd aan OSPF-gebied 0 voor een goede routering tussen switches. Deze aanpak maakt het configuratieproces gemakkelijker te beheren en minder vatbaar voor fouten, terwijl het ook flexibel is voor verschillende branches.

Bekijk nu de module "Distributie ACL Config", die segmentatie op de distributielaag biedt.

```
{% 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 %}
```

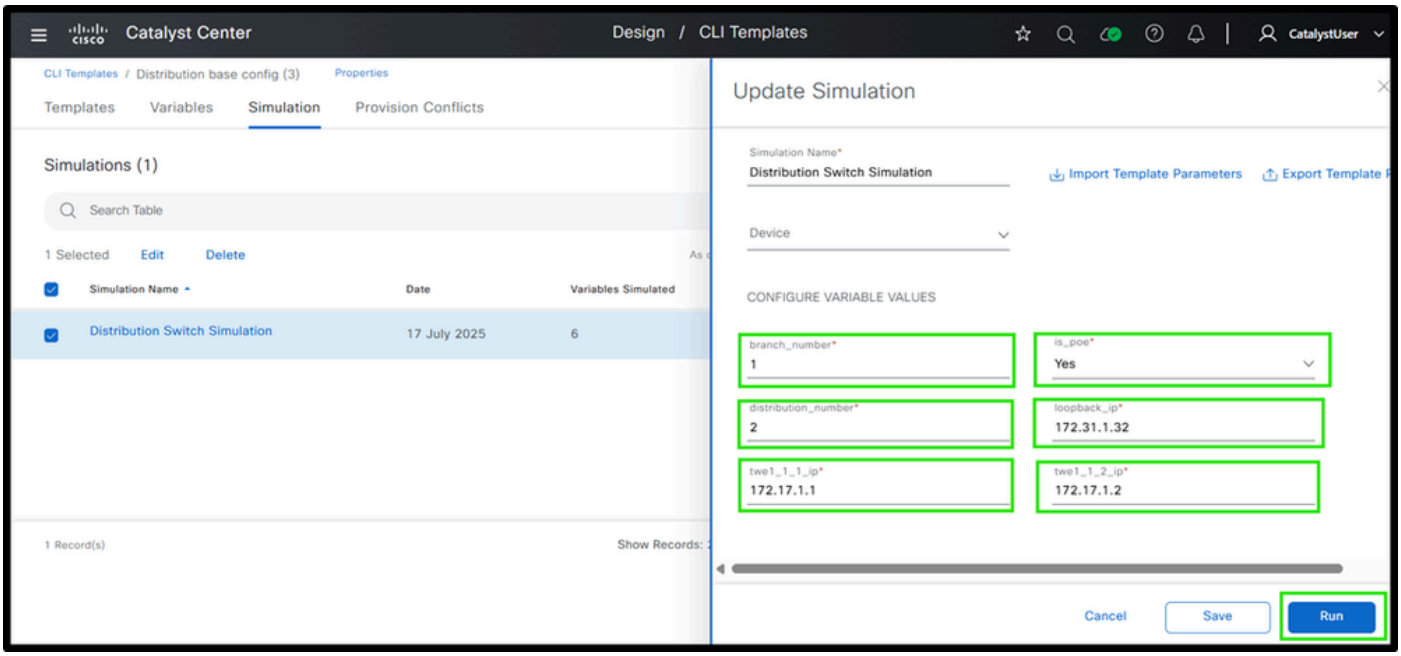
Distributie ACL Config

Deze module demonstreert segmentatie op de distributielaag met behulp van een Jinja2-sjabloon. Het maakt gebruik van de branch_number variabele om subnetadressen dynamisch te berekenen, waardoor geautomatiseerde en schaalbare ACL-configuraties mogelijk zijn. Voor elke tak blokkeert de ACL de communicatie tussen subnet 1 (172.17.X.0) en subnet 2 (172.16.X.0) door IP-verkeer tussen deze bereiken te weigeren. Het weigert ook verkeer naar het multicast-adres

239.255.255.250, terwijl al het andere verkeer wordt toegestaan. De VLAN-interface wordt dynamisch toegewezen op basis van het filiaalnummer en de ACL wordt inbound op die interface toegepast. Deze geautomatiseerde aanpak zorgt voor effectieve segmentatie per branche, vermindert handmatige configuratiefouten en vereenvoudigt de handhaving van het netwerkbeleid.

Tot slot is de laatste module, "Distributiestandaardconfiguratie", bijna identiek aan de module die wordt beschreven in de module Standaardconfiguratie voor toegang (raadpleeg die sectie voor meer informatie). Het bevat best practices, beveiligingshardening en belangrijke functies voor veilig apparaatbeheer. Het enige verschil zit hem in de broninterface: in de Access Switch template is het gedefinieerd als VLAN {{ branch_number * 100 + 13 }}, terwijl het in de Distribution Switch configuration kan worden gehardcodeerd als Loopback0.

Stap 3: Simulatie uitvoeren voordat u de configuratie implementeert.



(1) Distributie Switch Template Simulation Inputs en Outputs

(2) Distributie Switch Template Simulation Inputs en Outputs

(3) Distributie Switch Template Simulation Inputs en Outputs

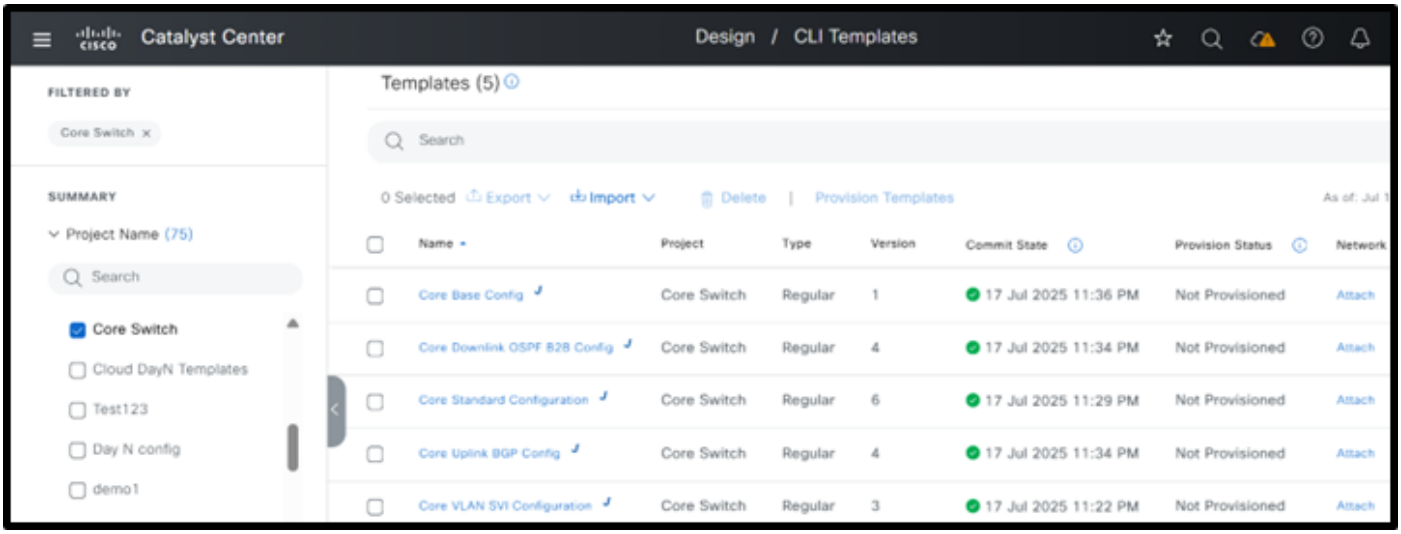
(4) Distributie Switch Template Simulation Inputs en Outputs

Op deze manier kunnen sjablonen worden gebruikt in de distributielaag om configuraties te genereren. Laten we nu eens kijken naar de kernlaagapparaten om te zien hoe sjablonen daar kunnen worden toegepast.

Kernlaag Switches

Ontwerp nu een modulaire template voor core switches. De basissjabloon en de bijbehorende modules maken deel uit van het project 'Core Switch' in Cisco Catalyst Center. Zie het basissjabloon in stap 1.

Stap 1: Definieer de verschillende switches



Core Switch Template Structure

Stap 2: Definieer verschillende modules

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

Core Base Config

De meeste core switch configuraties zijn vergelijkbaar in alle branches, zodat gemeenschappelijke waarden kunnen worden hardcoded. Meestal veranderen alleen de IP-adressen en deze kunnen worden ingesteld met behulp van variabelen. Omdat elke branche meestal slechts twee kernvariabelen heeft, is het eenvoudig om deze switches te beheren. Zelfs als sommige vestigingen meer core switches hebben, is hun aantal nog steeds minder dan het aantal access of distribution switches. Daarom is het als best practice belangrijker om de switches voor toegang en distributie tot een minimum te beperken, omdat ze in grotere aantallen worden gebruikt en omdat het hebben van te veel variabelen de configuratie tijdrovender kan maken.

Begin nu met de eerste module: "Core VLAN SVI Configuration." In dit voorbeeld worden de switches achter een firewall geplaatst en moeten ze eBGP-peering ermee instellen. Deze module is verantwoordelijk voor het genereren van de VLAN's en bijbehorende SVI's die nodig zijn voor de eBGP-peering en OSPF-omgeving. De firewall wordt geacht in een actieve/standby-configuratie te werken.

```
{% 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 %}
```

Core VLAN SVI-configuratie

Deze module maakt, zoals eerder uitgelegd, de vereiste VLAN's en bijbehorende SVI's voor het vaststellen van OSPF- en BGP-buurrelaties. Alle parameters, met uitzondering van de SVI IP-adressen, zijn hardcoded, inclusief het subnetmasker als dit overeenkomt met het IP-adresplan. Deze methode helpt variabelen te beperken en vermindert de kans op configuratiefouten.

Laten we nu de "Core Downlink OSPF B2B Config" module bekijken, die configuraties genereert voor downlink interfaces, OSPF en back-to-back koppelingen tussen Core Switch 1 en 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 %}
```

Core Downlink OSPF B2B Config

Net als bij de vorige module zijn de meeste waarden in deze module hardcoded om het aantal variabelen te minimaliseren. Alleen de IP-adressen voor loopback- en downlink-interfaces zijn variabel. Bovendien zijn de back-to-back poortkanalen en VLAN's gestandaardiseerd in verschillende branches.

Laten we nu de "Core Uplink BGP Config" module bekijken, die BGP-configuraties genereert en uplinks beheert die zijn aangesloten op de 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 %}
```

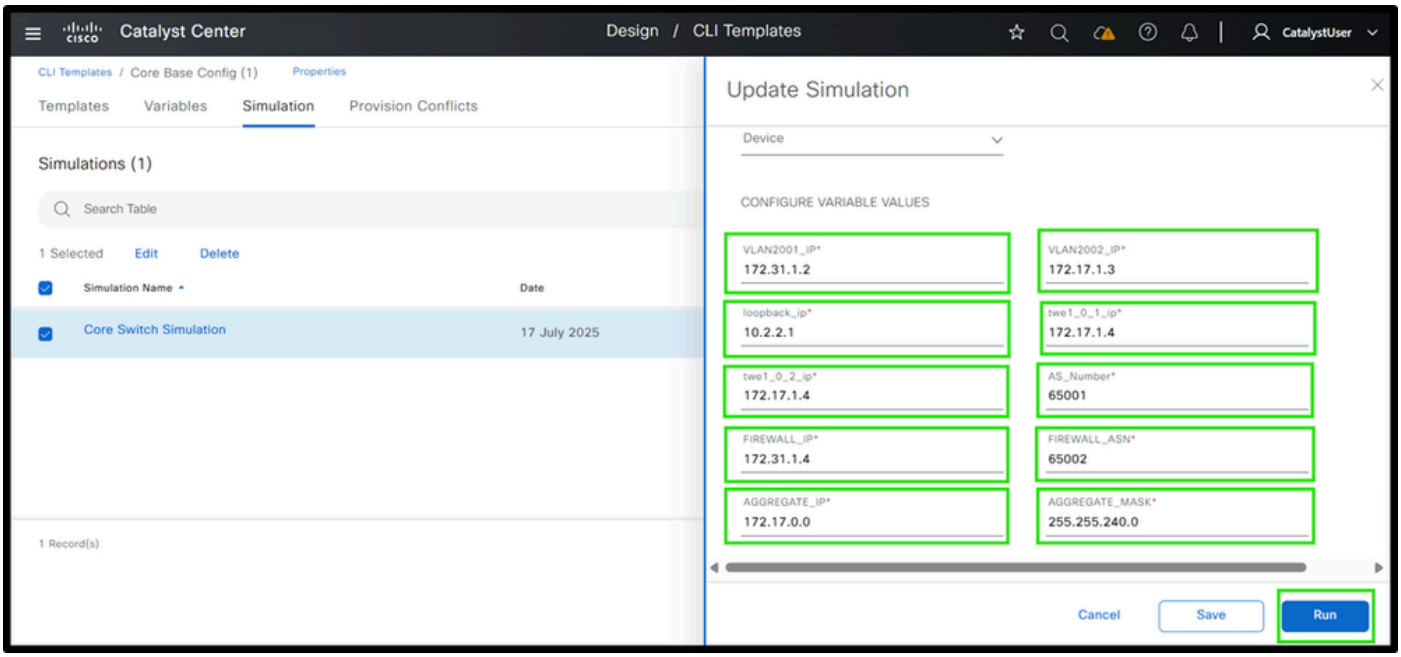
Core Uplink BGP-configuratie

Deze module genereert de BGP-configuratie die nodig is om een eBGP-buurrelatie met de firewall tot stand te brengen. Zoals hierboven weergegeven, zijn de meeste waarden hardcoded omdat ze

consistent blijven in verschillende branches. Alleen de IP-adressen en AS-nummers, die voor elke tak kunnen variëren, worden als invoervariabelen gebruikt en gebruikt om de nodige configuratie te genereren. De meeste andere instellingen zijn gestandaardiseerd om het aantal variabelen te minimaliseren. De uplinkinterfaces die zijn aangesloten op de firewall worden gespecificeerd samen met het VLAN dat wordt gebruikt voor de eBGP-omgeving en dat is gegenereerd door de vorige module.

Tot slot is de laatste module, "Core Standard Configuration", bijna identiek aan de module die wordt beschreven in de Access Standard Configuration (raadpleeg die sectie voor meer informatie). Het bevat best practices, beveiligingshardening en belangrijke functies voor veilig apparaatbeheer. In overeenstemming met de configuratie van de distributiemodule kan de broninterface ook in deze switch worden ingesteld op loopback0 en kan deze waarde worden hardcoded.

Stap 3: Simulatie uitvoeren

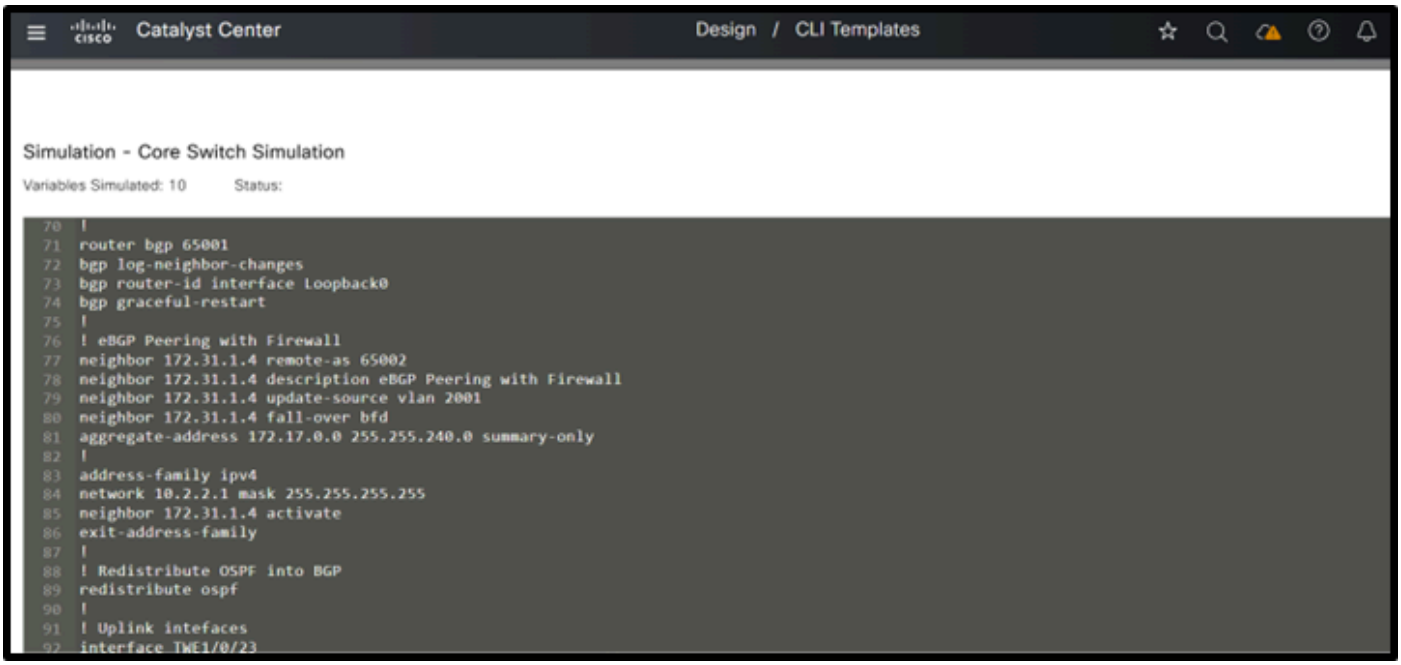


(1) Invoer en uitvoer van Core Switch Template Simulation

(2) Invoer en uitvoer van Core Switch Template Simulation

(3) Invoer en uitvoer van Core Switch Template Simulation

(4) Invoer en uitvoer van Core Switch Template Simulation



(5) Core Switch Template Simulation Inputs and Outputs

Dit completeert de gedetailleerde uitleg van het ontwerpen van sjablonen voor de drielaagse architectuur, waarin zowel de structuur als de configuratie van elke module wordt beschreven.

Al deze modules maken gebruik van de best practices die eerder zijn uitgelegd.

Opmerking: Raadpleeg bij het ontwerpen van sjablonen voor een samengevouwen kernarchitectuur de uitleg voor de drielaagse architectuur. De structuur van de sjabloon

---

blijft hetzelfde; functies die voorheen afzonderlijk werden geïmplementeerd in de kern- en distributielagen, worden nu gecombineerd in de samengevouwen kernlaag. Dezelfde modulaire sjabloonbenadering kan ook hier worden gebruikt, door een basissjabloon te maken en te verwijzen naar de relevante modules erin.

Samenvatting

De traditionele 3-tier campusarchitectuur is vaak gebaseerd op uitgebreide handmatige configuratie in de kern-, distributie- en toegangslagen. Deze aanpak is niet alleen tijdrovend, maar ook vatbaar voor menselijke fouten. De afwezigheid van automatisering en gecentraliseerd beheer verhoogt de operationele overhead aanzienlijk, waardoor het moeilijk is om moderne, dynamische campusnetwerken effectief te schalen en te beheren. Via Catalyst Center kunnen CLI-sjabloonfunctieconfiguraties worden geautomatiseerd voor de traditionele LAN-netwerken. Het is belangrijk om de modulaire aanpak te gebruiken bij het leveren van de apparaten. Modules kunnen worden gebaseerd op de verschillende functies die op verschillende lagen worden gebruikt. En bind dan eindelijk al deze modules aan de basismodule.

Oproep tot actie

We nodigen organisaties uit om de modulaire sjabloonmethodologie die in dit witboek wordt gepresenteerd, over te nemen als een best practice voor het standaardiseren van switch-configuraties en het optimaliseren van netwerkactiviteiten in zowel drielaags- als samengevouwen kernarchitecturen.

- Door modulaire sjablonen te implementeren, kunnen netwerkteams:
- Verbeter de operationele efficiëntie door consistente, herhaalbare configuratiepraktijken.
- Minimaliseer menselijke fouten en verminder de tijd die nodig is voor het oplossen van problemen.
- Grotere schaalbaarheid realiseren om groei en veranderende bedrijfsbehoeften te ondersteunen.
- Zorgen voor consistentie van de configuratie in verschillende omgevingen.

Deze aanpak stroomlijnt niet alleen het dagelijks beheer, maar maakt ook snellere implementaties mogelijk, vereenvoudigt updatecycli en verbetert de afstemming op beveiligings- en nalevingsvereisten. Door modulaire sjablonen te gebruiken, positioneert u uw netwerk voor flexibiliteit, veerkracht en succes op de lange termijn in een steeds veranderend IT-landschap.

Voor praktische demonstraties, meer informatie over sjablonen, zie YouTube-serie

1 Sjablonen maken en beheren in Catalyst Center

https://youtu.be/SyUqEEcwye0

2 Hoe systeembindvariabelen in CLI-sjablonen in Catalyst Center te gebruiken

https://youtu.be/gV1QBuHYJdo

Auteurs

Naveen Kumar, Customer Delivery Architect, Cisco Customer Experience

Risabh Mishra, Consulting Engineer, Cisco Customer Experience

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.