# Verbeter de doorvoer op Catalyst 8000V met Multi-TxQs in AWS

## Inhoud


Inleiding

Achtergrondinformatie

Katalysator 8000V-gedrag bij niet-gebruik van Multi-TxQs

Wat zijn Multi-TXQ's in AWS-infrastructuur

Hoe verkeer wordt gehasht in Multi-TxQs

Catalyst 8000V-softwareversies die Multi-TxQs ondersteunen

Hoe het IP-adresseringsschema te ontwerpen om de hashing te berekenen

Voorwaarden

Virtuele omgeving maken

Bereken het IP-adresschema met behulp van Python Hash Index Script voor 17.7 en 17.8 releases (Afbreken)

Bereken IP-adresschema met behulp van Python Hash Index Script voor 17.9 en latere releases

Voorbeeldtopologie en CLI-configuratie met 8 TXQ's met loopback-interfaces

Voorbeeldtopologie en CLI-configuratie met behulp van 12 TXQ's met loopback-interfaces

Voorbeeldtopologie en CLI-configuratie met 12 TXQ's met secundaire IP-adressen

autonome modus

SD-WAN-modus

Handige CLI-opdrachten voor probleemoplossing

Voorbeeld CLI-uitvoer


## Inleiding

In dit document wordt beschreven hoe u Multi-TXQ's op Catalyst 8000V kunt inschakelen en gebruiken die in AWS-omgevingen worden geïmplementeerd om de doorvoerprestaties te verbeteren.

## Achtergrondinformatie

De aanwezigheid van meerdere wachtrijen vereenvoudigt en versnelt het proces van het toewijzen van inkomende en uitgaande pakketten aan een bepaalde vCPU. Het gebruik van Multi-TXQ's op Catalyst 8000V maakt efficiënt kerngebruik mogelijk over de beschikbare toegewezen dataplanekernen, wat resulteert in hogere doorvoerprestaties. Dit artikel biedt een licht overzicht van hoe Multi-TXQ's werken, hoe het is geconfigureerd, toont voorbeeld CLI-configuraties voor zowel Autonomous als SD-WAN Catalyst 8000V-implementaties en herziet opdrachten voor probleemoplossing om prestatieknelpunten te helpen vinden.

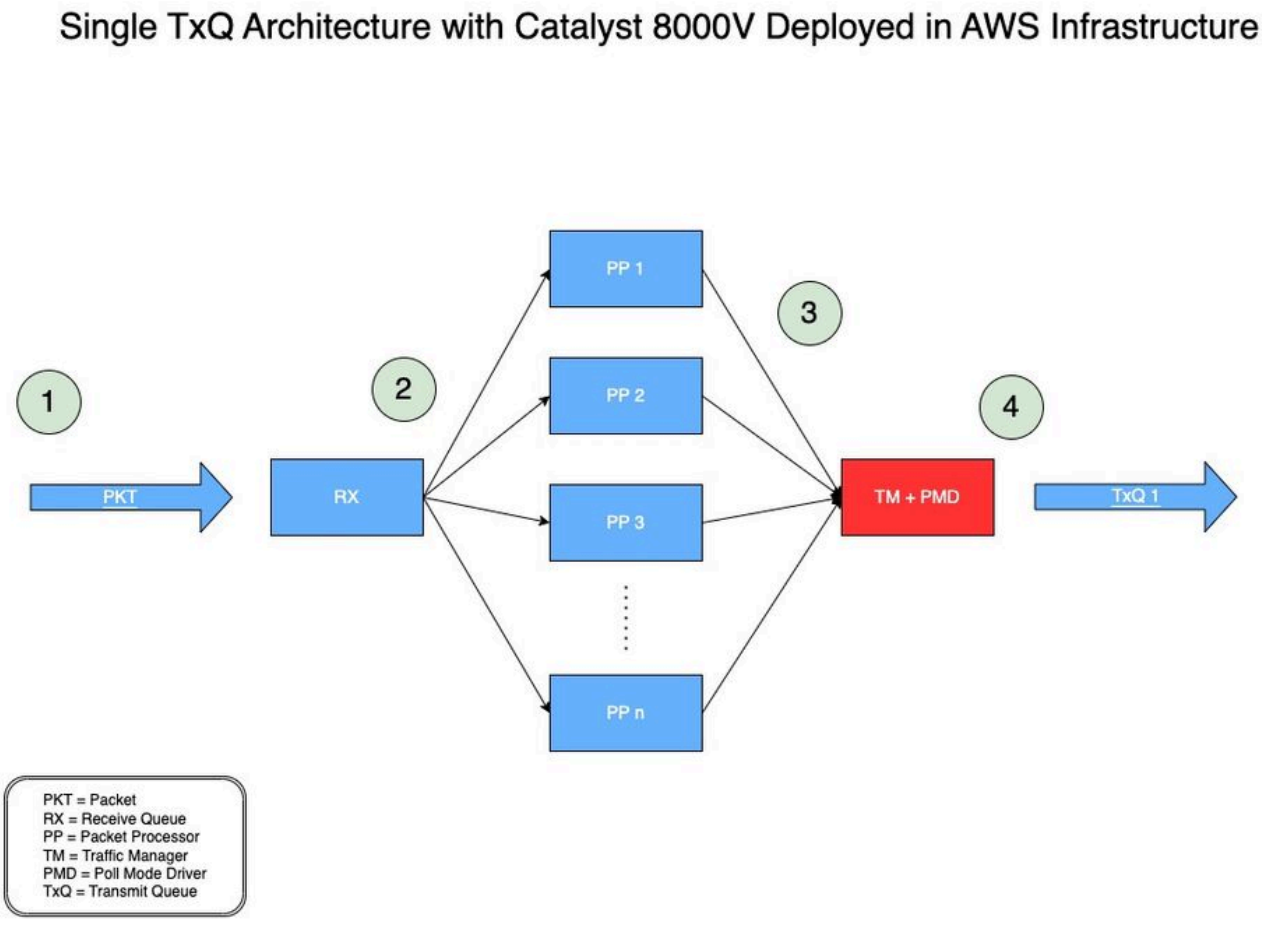
# Katalysator 8000V-gedrag bij niet-gebruik van Multi-TxQs

Tot 17.18 software release worden pakketten die Catalyst 8000V binnenkomen gedistribueerd naar alle vCPU (Packet Processing cores), ongeacht de stromen. Zodra PP de pakketverwerking heeft voltooid, wordt de stroomvolgorde hersteld om op een interface te worden verzonden.

Voordat het pakket in een zendwachtrij (TxQ) wordt geplaatst, maakt de Catalyst 8000V één TxQ per interface. Daarom, als er slechts één uitgang-interface beschikbaar is, gaan meerdere stromen naar één TxQ.

De Catalyst 8000V kan niet profiteren van dit Multi-TxQ-proces als er slechts één interface beschikbaar is. Dit leidt tot knelpunten in de doorvoerprestaties en een ongelijke verdeling van de belasting over de beschikbare cores van het gegevensvliegtuig. Als er slechts één uitgang-interface wordt gebruikt om gegevens uit de C8000V-instantie te verzenden, is er slechts één TxQ beschikbaar om netwerkverkeer te verzenden en mogelijk pakketten te laten vallen omdat de enkele wachtrij sneller wordt gevuld.

Ter referentie kunt u het enkele TxQ-architectuurmodel voor Catalyst 8000V dat in AWS is geïmplementeerd, hier vinden in afbeelding 1.



Afbeelding 1: Single TxQ-architectuurmodel voor Catalyst 8000V geïmplementeerd in AWS.

1. Een netwerkpakket (PKT) doorkruist een VPC en wordt ontvangen op de ingangsinterface van een C8000V.
2. De PKT wordt geplaatst op een ontvangstwachtrij (RX) en wordt vervolgens doorgestuurd naar een Packet Processor (PP)-engine die wordt bepaald door een algoritme.
3. Nadat de Packet Processor (PP) het pakket heeft verwerkt, verzendt deze het pakket naar Traffic Manager (TM).
4. Aan het einde van de TM-verwerking is één kern verantwoordelijk voor het plaatsen van het pakket in de enige beschikbare TxQ die vervolgens wordt doorgestuurd naar de uitgang-interface van de Catalyst 8000V.

## Wat zijn Multi-TXQ's in AWS-infrastructuur

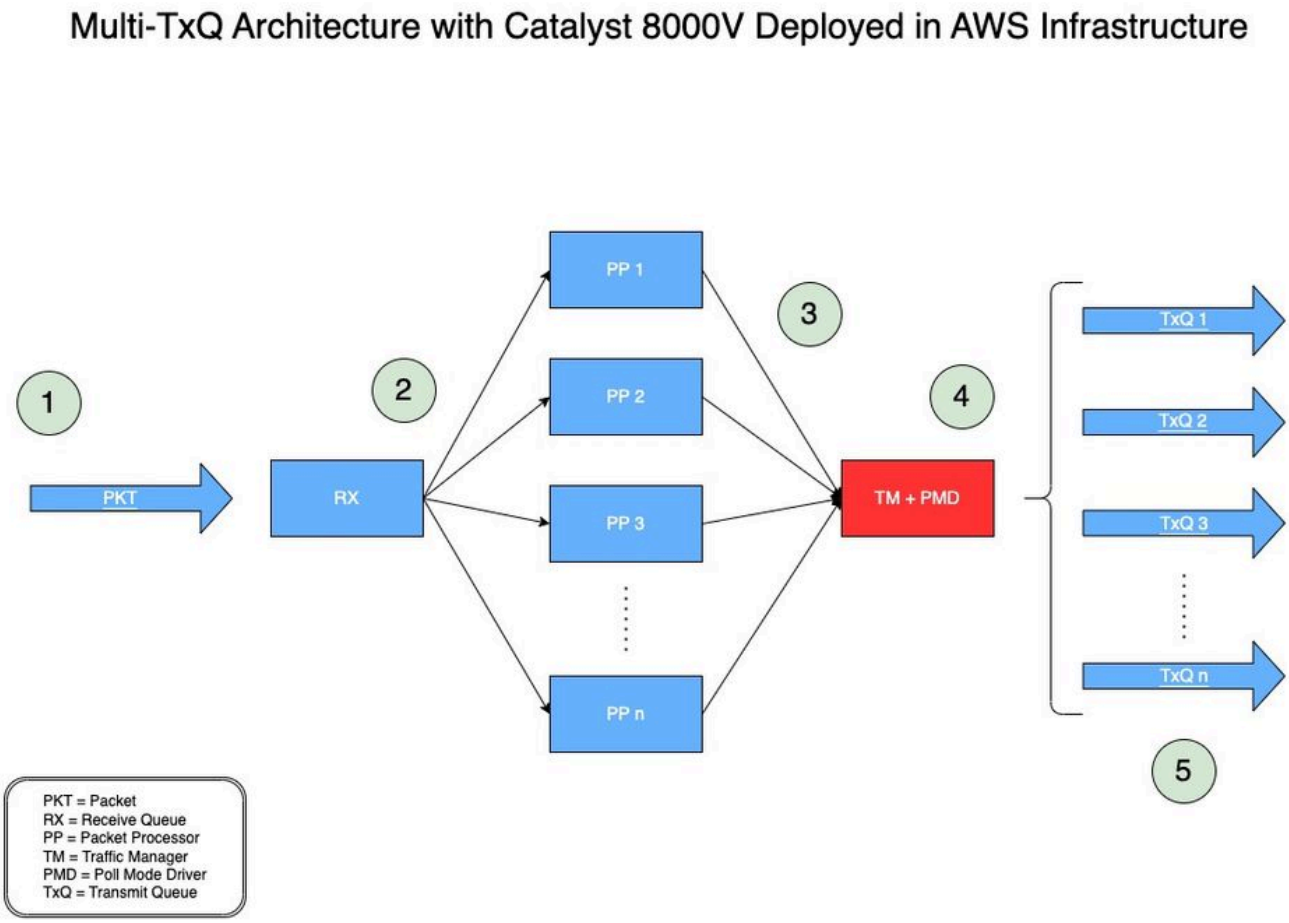
AWS ENA biedt meerdere zendwachtrijen (Multi-TxQ's) om de interne overhead te verminderen en de schaalbaarheid te vergroten. De aanwezigheid van meerdere wachtrijen vereenvoudigt en versnelt het proces van het toewijzen van inkomende en uitgaande pakketten aan een bepaalde vCPU. Het AWS- en DPDK-netwerkreferentiemodel is op stroom gebaseerd, waarbij elke vCPU een stroom verwerkt en pakketten van die stroom naar een toegewezen transmissiewachtrij (TxQ) verzendt. Het RX/TX-wachtrijpaar voor elke vCPU is geldig op basis van het stroomgebaseerde model.

Omdat de Catalyst 8000V NIET op flow is gebaseerd, is de instructie "RX/TX-wachtrijpaar voor elke vCPU" niet van toepassing op de Catalyst 8000V.

In dit geval zijn RX/TX-wachtrijen niet per vCPU maar per interface. RX/TX-wachtrijen fungeren als interfaces tussen de toepassing (Catalyst 8000V) en AWS-infrastructuur/hardware voor het verzenden van gegevens/netwerkverkeer. AWS bepaalt per instantie hoe snel en hoeveel RX/TX-wachtrijen beschikbaar zijn per interface.

De Catalyst 8000V moet meerdere interfaces hebben om meerdere TxQ's te maken. Om de flow op orde te houden met meerdere stromen die uit een interface gaan (zodra de Catalyst 8000V meerdere TxQs inschakelt na dit proces), stroomt de Catalyst 8000V hashes op basis van de 5-tuples om de juiste TxQ te kiezen. Een gebruiker kan meerdere interfaces op de Catalyst 8000V maken met dezelfde fysieke NIC die aan de instantie is gekoppeld door Loopback-interfaces of secundaire IP-adressen te gebruiken.

In Afbeelding 2 kunt u zien hoe een pakket wordt verwerkt met behulp van de Multi-TxQ-architectuur met Catalyst 8000V in AWS.




Afbeelding 2: Multi-TxQ-architectuurmodel voor Catalyst 8000V geïmplementeerd in AWS.

1. Een netwerkpakket (PKT) doorkruist een VPC en wordt ontvangen op de ingangsinterface van een C8000V.
2. De PKT wordt geplaatst op een ontvangstwachtrij (RX) en wordt vervolgens doorgestuurd naar een Packet Processor (PP)-engine die wordt bepaald door een algoritme.
3. Nadat de Packet Processor (PP) het pakket heeft verwerkt, verzendt deze het pakket naar Traffic Manager (TM).
4. Aan het einde van de TM-verwerking, voordat het pakket in een verzendwachtrij (TxQ) wordt geplaatst, kijkt de TM naar de pakketheader en hasht het pakket (uitgelegd in de volgende sectie). Een andere component, de Poll Mode Driver (PMD), wordt gebruikt om het aantal TXQ's te configureren dat door de instantie wordt ondersteund. Eén kern is gewijd aan de TM + PMD-functie die het hashen en verzenden van het pakket naar de toegewezen TxQ doet.
5. De TxQ wordt gekozen op basis van de vijf tuples hashed en modulo met het aantal TxQ ondersteund door de instantie. Pakketten worden naar de geselecteerde TxQ geplaatst en doorgestuurd naar de uitgang-interface van de Catalyst 8000V.

## Hoe verkeer wordt gehasht in Multi-TxQs

Aan het einde van de TM-verwerking, zoals weergegeven in stap 4 van figuur 2, kijkt de TM naar de pakketheader en haalt de 5 tuples (bestemmingsadres, bronadres, protocol, bestemmingspoort en bronpoort) uit het pakket en hasht het naar een TxQ.

De TxQ wordt gekozen op basis van de vijf tuples hashed en modulo met het aantal TxQ ondersteund door de instantie.

## Catalyst 8000V-softwareversies die Multi-TxQs ondersteunen

AWS EC2-instanties van hetzelfde type instantiefamilie ondersteunen allemaal een verschillend aantal TXQ's, afhankelijk van de instantiegrootte. De C8000V begon ondersteuning te bieden voor meerdere TxQ's vanaf IOS® XE 17.7.

Vanaf IOS® XE 17.7 ondersteunt de C8000V meerdere TxQ's op de C5n.9xlarge die tot 8 TXq's kunnen hebben.

Vanaf IOS® XE 17.9 ondersteunt de C8000V de C5n.18xlarge instantiegrootte, die tot 12 TXQ's kan hebben (50% meer dan C5n.9xlarge).

Hoewel Multi-TxQ wordt ondersteund vanuit IOS® XE 17.7, wordt het ten zeerste aanbevolen om IOS® XE 17.9 te gebruiken voor zowel de levenscyclus van de software als de hogere doorvoerprestaties met 12 TxQ-ondersteuning.

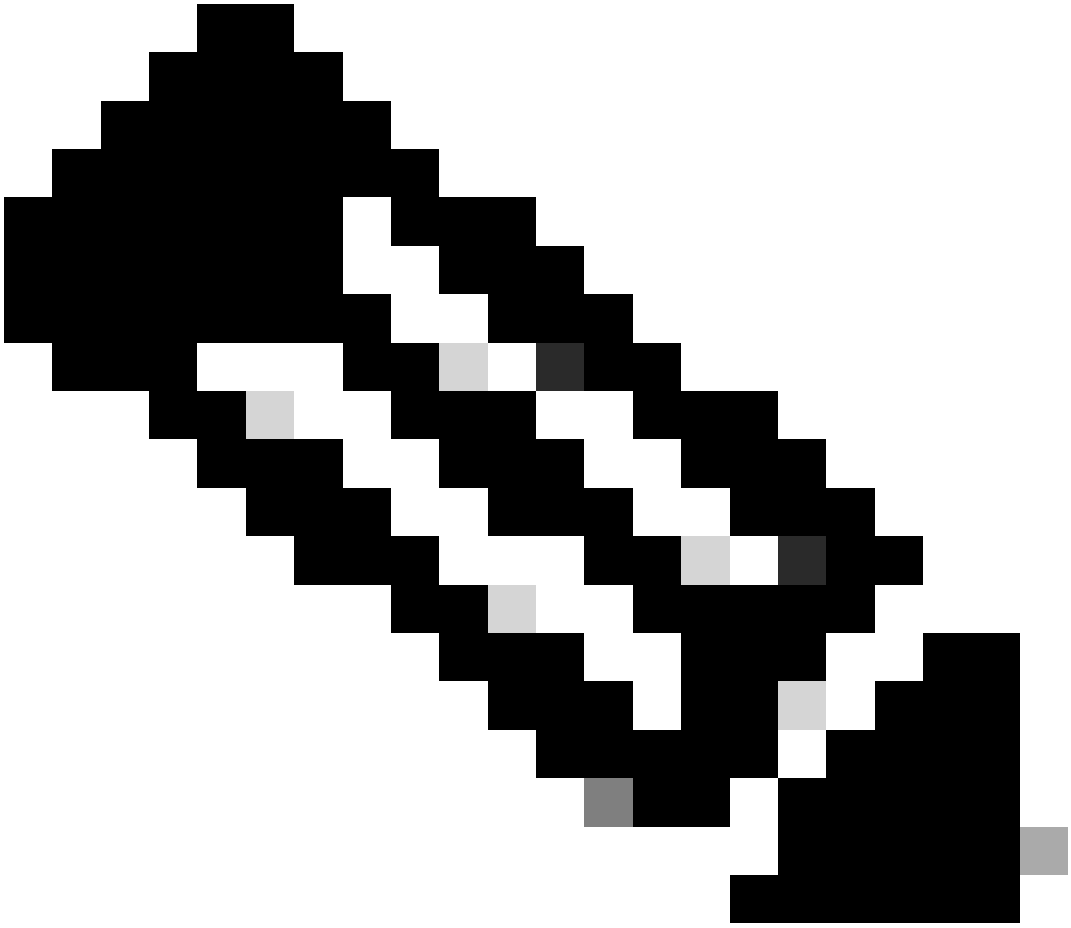
## Hoe het IP-adresseringsschema te ontwerpen om de hashing te berekenen

Om het hashverkeer tussen alle beschikbare TxQ’s gelijk te verdelen, moeten speciale IP-adressen worden gebruikt wanneer de Catalyst 8000V IPsec/GRE-tunnels beëindigt.

Er zijn openbare scripts beschikbaar om deze speciale IP-adressen te genereren die kunnen worden gebruikt om de Catalyst 8000V-interfaces te configureren die verantwoordelijk zijn voor het beëindigen van deze tunnels. Deze sectie bevat instructies over het downloaden en gebruiken van de scripts om de vereiste IP-adressen te engineeren voor zelfs Multi-TxQ-hashing.

Als de Catalyst 8000V helder tekstverkeer zoals TCP/UDP afhandelt, is er geen speciaal IP-adresseringsschema vereist.

Originele instructies zijn hier te vinden: https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/



Opmerking: voor Catalyst 8000V met 17.18 of hoger worden pakketten verschillend verdeeld. Er moet dus een ander hashing-algoritme worden gebruikt.

## Voorwaarden

- Moet Linux/MacOS of een Windows-machine hebben die Python-scripts kan uitvoeren.
- Controleer of de Python-versie 3.8.9 of hoger is; controleer de Python-versie met 'python3 --versie'
- Installeer PIP als deze nog niet is geïnstalleerd. Zo niet, voer uit:
  - curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  - Python3 Get-Pip.Py

U kunt controleren welke Python-versie uw machine gebruikt met het commando 'python3 --versie'.

```
user@computer ~ % python3 --version
```

```
Python 3.9.6
```

Zodra de Python-versie is geverifieerd en wordt uitgevoerd, installeert u de nieuwste versie van PIP met een versie die gelijk is aan of hoger is dan die 3.8.9.

```
user@computer ~ % curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current

                                 Dload  Upload   Total   Spent    Left  Speed

100 2570k  100 2570k    0     0  6082k      0 --:--:-- --:--:-- --:--:-- 6135k
```

<#root>

```
user@computer ~ % python3 get-pip.py

Defaulting to user installation because normal site-packages is not writeable
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl.metadata (3.5 kB)
Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)

   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 7.4 MB/s eta 0:00:00

Installing collected packages: pip

  WARNING: The scripts pip, pip3 and pip3.9 are installed in '/Users/name/Library/Python/3.9/bin' which

  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-scri

Successfully installed pip-23.3.1


[

notice

]

 A new release of pip is available: 21.2.4 -> 23.3.1


[

notice

]

 To update, run: /Applications/Xcode.app/Contents/Developer/usr/bin/python3 -m pip install --upgrade pi
```

# Virtuele omgeving maken

Nadat de vereisten zijn geïnstalleerd, maakt u de virtuele omgeving en downloadt u het IP-adres-hashing-script dat wordt gebruikt om het unieke IP-adresschema voor Multi-TxQ te genereren.

Overzicht van opdrachten:

1. Python3 -M VENV C8KV-hash
2. CD C8KV-hash
3. Bronbak/activeren
4. Git Clone https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/
5. CD C8KV-AWS-PMD-Hash
6. Python3 -M Pip Install --Upgrade Pip
7. Pip install -r requirements.txt

Virtuele omgevingen in Python worden gebruikt om geïsoleerde werkruimten te maken die geen invloed hebben op andere projecten of afhankelijkheden. Maak de virtuele omgeving 'c8kv-hash' met behulp van deze opdracht:

```
user@computer Desktop % python3 -m venv c8kv-hash
```

Navigeer binnen de virtuele omgeving naar de map 'c8kv-hash' (eerder gemaakt).

```
user@computer Desktop % cd c8kv-hash
```

Activeer de virtuele omgeving.

```
user@computer c8kv-hash % source bin/activate
```

Kloon de repository die het Multi-TxQ hashing python script heeft.

```
(c8kv-hash) user@computer c8kv-hash % git clone https://github.com/CiscoDevNet/python-c8000v-aws-multit

Cloning into 'c8kv-aws-pmd-hash'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (59/59), done.
remote: Total 82 (delta 34), reused 57 (delta 19), pack-reused 0
Receiving objects: 100% (82/82), 13.01 KiB | 2.60 MiB/s, done.
Resolving deltas: 100% (34/34), done.
```

Zodra de repository is gekloond, navigeert u naar de map 'c8kv-aws-pmd-hash'. Aangezien deze zich in de virtuele omgeving bevindt die is gemaakt, installeert u de nieuwste versie van PIP.

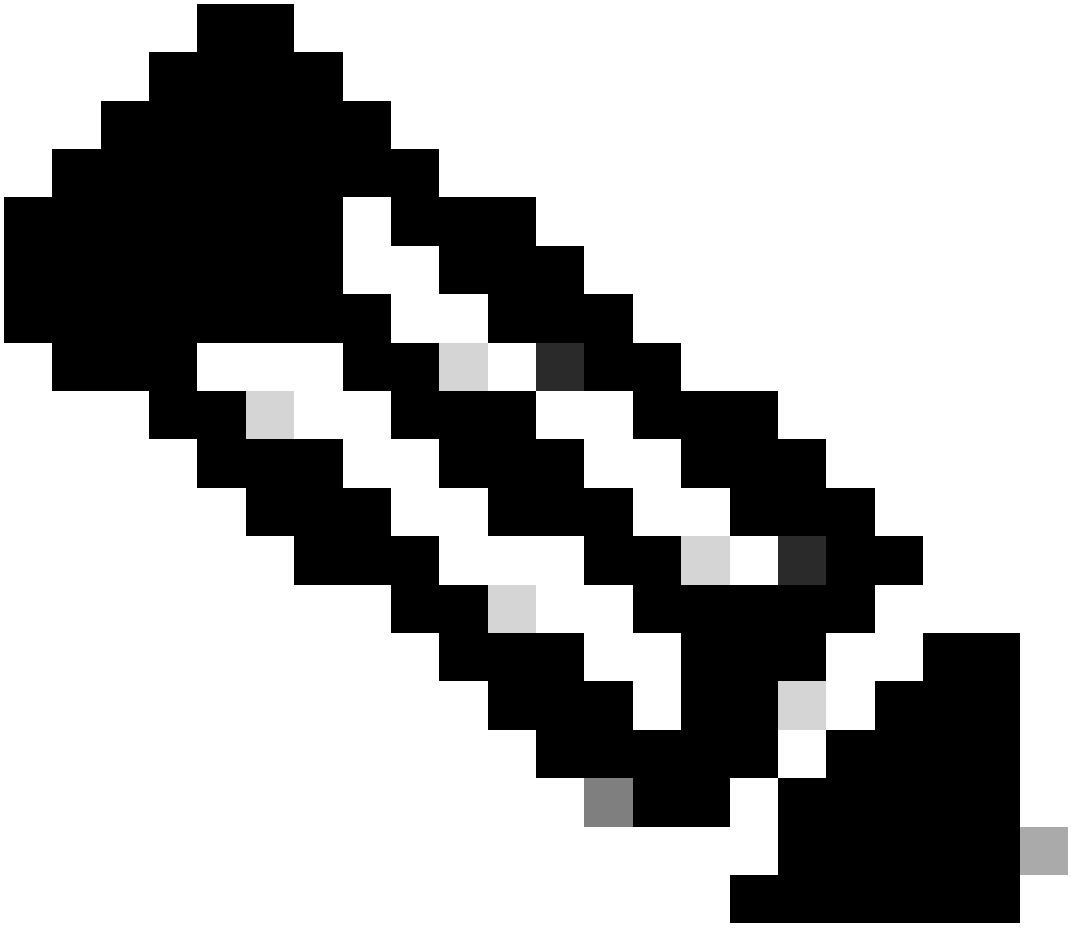
```
(c8kv-hash) user@computer c8kv-hash % cd c8kv-aws-pmd-hash
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 -m pip install --upgrade pip

Requirement already satisfied: pip in /Users/name/Desktop/c8kv-hash/lib/python3.9/site-packages (21.2.4)
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)
     |████████████████████████████████| 2.1 MB 2.7 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 21.2.4
    Uninstalling pip-21.2.4:
      Successfully uninstalled pip-21.2.4
Successfully installed pip-23.3.1
```

Nadat PIP is bijgewerkt, installeert u de afhankelijkheden in het bestand requirements.txt in de map.

```
(c8kv-hash) user@computer c8kv-aws-pmd-hash % pip install -r requirements.txt
Collecting crc32c==2.3 (from -r requirements.txt (line 1))
  Downloading crc32c-2.3-cp39-cp39-macosx_11_0_arm64.whl (27 kB)
Installing collected packages: crc32c
Successfully installed crc32c-2.3
```

De virtuele omgeving is nu up-to-date en kan worden gebruikt om het IP-adresschema voor Multi-TxQ te genereren.

## Bereken het IP-adresschema met behulp van Python Hash Index Script voor 17,7 en 17,8 releases (Afbreken)



OPMERKING: 7.7 EN 17.8 HASHSCRIPTS WORDEN BINNENKORT VERWIJDERD. HET IS AAN TE RADEN OM 17.9 HASH SCRIPT TE GEBRUIKEN

Overzicht van opdrachten:

- Python3 c8kv_multitxq_hash.py --old_crc 1 --dest_network 192.168.1.0/24 --src_network 192.168.2.0/24 --unique_hash 1

'-old_crc 1' genereert hash-index op basis van 17.7 en 17.8 release met modulo 8 om overeen te komen met de ondersteunde PMD TXQ (niet wijzigen)

'--dest_network' definieert het subnet van het bestemmingsnetwerkadres (wijzigen op basis van het IP-adresschema van uw netwerk)

'--src_network' definieert het subnet van het bronnetwerkadres (wijzigen op basis van het IP-adresschema van uw netwerk)

'-unique_hash 1' genereert één set (8 paren voor 8 TXQ's) unieke gehashte IP-adressen. Dit kan

worden gewijzigd.

<#root>

```
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 c8kv_multitxq_hash.py --old_crc 1 --dest_network 1

Dest:            Src:             Prot  dstport  srcport          Hash: Rev-hash:

192.168.1.0      192.168.2.0      2     5
192.168.1.0      192.168.2.1      2     7
192.168.1.0      192.168.2.2      2     1
192.168.1.0      192.168.2.3      2     3
192.168.1.0      192.168.2.4      2     5
192.168.1.0      192.168.2.5      2     7
192.168.1.0      192.168.2.6      2     1
192.168.1.0      192.168.2.7      2     3
192.168.1.0      192.168.2.8      2     5
192.168.1.0      192.168.2.9      2     7
192.168.1.0      192.168.2.10     2     1
.
. ### trimmed output ###
.
192.168.1.255    192.168.2.247    5     2
192.168.1.255    192.168.2.248    5     4
192.168.1.255    192.168.2.249    5     6
192.168.1.255    192.168.2.250    5     0
192.168.1.255    192.168.2.251    5     2
192.168.1.255    192.168.2.252    5     4
192.168.1.255    192.168.2.253    5     6
192.168.1.255    192.168.2.254    5     0
192.168.1.255    192.168.2.255    5     2

Unique hash:

------ Tunnels set 0 ---------

192.168.1.37<===>192.168.2.37<===>0


192.168.1.129<===>192.168.2.129<===>1


192.168.1.36<===>192.168.2.36<===>2


192.168.1.128<===>192.168.2.128<===>3


192.168.1.39<===>192.168.2.39<===>4


192.168.1.131<===>192.168.2.131<===>5


192.168.1.38<===>192.168.2.38<===>6
```

**192.168.1.130<===>192.168.2.130<===>7**

# Bereken IP-adresschema met behulp van Python Hash Index Script voor 17.9 en latere releases

Overzicht van opdrachten:

- Python3 c8kv_multitxq_hash.py --dest_network 192.168.1.0/24 --src_network 192.168.2.0/24 --prot UDP --src_port 12346 --dst_port 12346 --unique_hash 1

Merk op dat in IOS® XE versie 17.9 en hoger, het script modulo 12 gebruikt zonder de optie --old_crc, die overeenkomt met de ondersteunde PMD TXQ.

'--dest_network' definieert het subnet van het bestemmingsnetwerkadres (wijzigen op basis van het IP-adresschema van uw netwerk)

'--src_network' definieert het subnet van het bronnetwerkadres (wijzigen op basis van het IP-adresschema van uw netwerk)

'--Prot UDP' definieert het gebruikte protocol. De gebruiker kan de protocolparameter opgeven als "gre" of "tcp" of "udp" of een decimale waarde (OPTIONEEL)

'--src_port' definieert de gebruikte bronpoort (OPTIONEEL)

'--dst_port' definieert de gebruikte bestemmingspoort (OPTIONEEL)

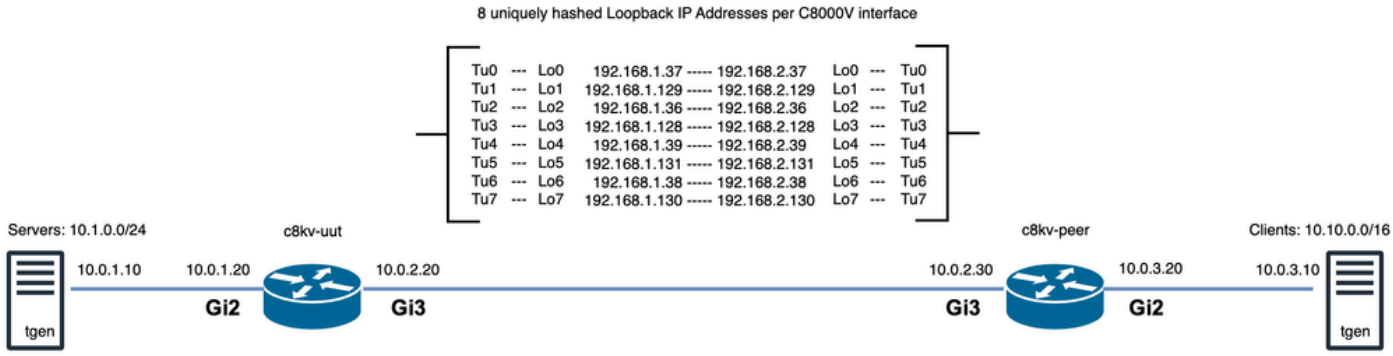
'-unique_hash 1' genereert één set (12 paren voor 12 TXQ's) unieke gehashte IP-adressen. Dit kan worden gewijzigd.

<#root>

```
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 c8kv_multitxq_hash.py --dest_network 192.168.1.0/

Dest:            Src:             Prot  dstport  srcport          Hash: Rev-hash:

192.168.1.0      192.168.2.0      17    12346    12346       ==>    4        4       <-- Unique Hash Va
192.168.1.0      192.168.2.1      17    12346    12346       ==>    4        4
192.168.1.0      192.168.2.2      17    12346    12346       ==>    8        8       <-- Unique Hash Va
192.168.1.0      192.168.2.3      17    12346    12346       ==>    0        0       <-- Unique Hash Va
192.168.1.0      192.168.2.4      17    12346    12346       ==>    0        0
192.168.1.0      192.168.2.5      17    12346    12346       ==>    0        0
192.168.1.0      192.168.2.6      17    12346    12346       ==>    4        4
192.168.1.0      192.168.2.7      17    12346    12346       ==>    0        0
192.168.1.0      192.168.2.8      17    12346    12346       ==>    9        9       <-- Unique Hash Va
192.168.1.0      192.168.2.9      17    12346    12346       ==>    9        9
192.168.1.0      192.168.2.10     17    12346    12346       ==>    9        9
192.168.1.0      192.168.2.11     17    12346    12346       ==>    1        1       <-- Unique Hash Va
192.168.1.0      192.168.2.12     17    12346    12346       ==>    1        1
.
. ### trimmed output ###
.
```

```
192.168.1.255    192.168.2.250    17    12346    12346       ==>    1        1
192.168.1.255    192.168.2.251    17    12346    12346       ==>    1        1
192.168.1.255    192.168.2.252    17    12346    12346       ==>    9        9
192.168.1.255    192.168.2.253    17    12346    12346       ==>    1        1
192.168.1.255    192.168.2.254    17    12346    12346       ==>    5        5       <-- Unique Hash Va
192.168.1.255    192.168.2.255    17    12346    12346       ==>    9        9

Unique hash:

------ Tunnels set 0 ---------


192.168.1.38 <===> 192.168.2.38<===>0


192.168.1.37 <===> 192.168.2.37<===>1


192.168.1.53 <===> 192.168.2.53<===>2


192.168.1.39 <===> 192.168.2.39<===>3


192.168.1.48 <===> 192.168.2.48<===>4


192.168.1.58 <===> 192.168.2.58<===>5


192.168.1.42 <===> 192.168.2.42<===>6


192.168.1.46 <===> 192.168.2.46<===>7


192.168.1.40 <===> 192.168.2.40<===>8


192.168.1.43 <===> 192.168.2.43<===>9


192.168.1.36 <===> 192.168.2.36<===>10


192.168.1.56 <===> 192.168.2.56<===>11
```

## Voorbeeldtopologie en CLI-configuratie met 8 TXQ's met loopback-interfaces




Afbeelding 3: Voorbeeldtopologie die acht TxQs gebruikt met Loopback-interfaces.

Dit is een voorbeeld CLI-configuratie voor 'c8kv-uut' (Afbeelding 3) die acht IPsec-tunnels met Loopback-interfaces maakt met behulp van de berekende hashed IP-adressen (192.168.1.X) uit de vorige sectie.

Een soortgelijke configuratie zou van toepassing zijn op het andere routereindpunt (c8kv-peer) met de resterende acht berekende hashed IP-adressen (192.168.2.X).

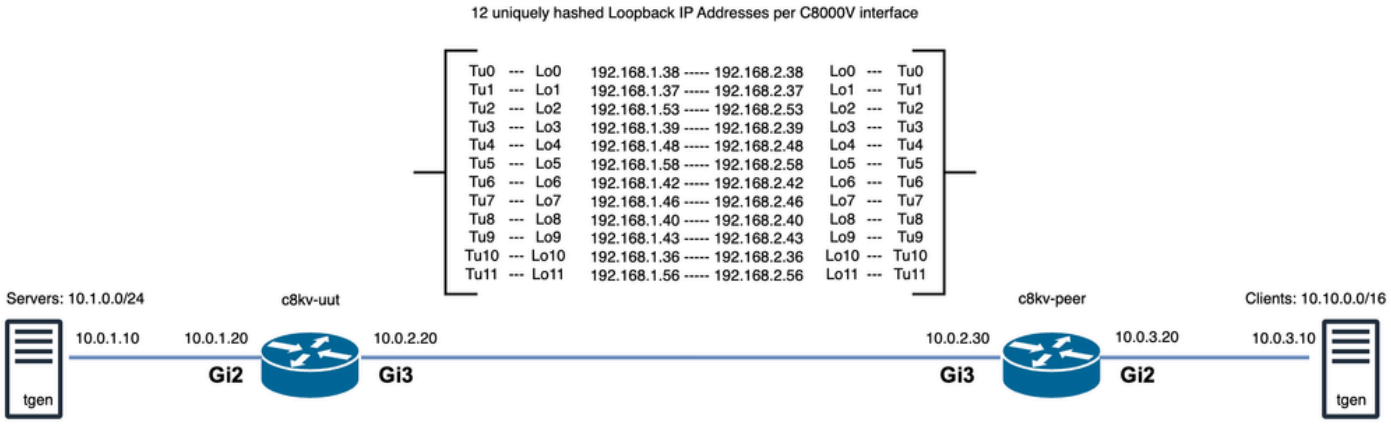
```
ip cef load-sharing algorithm include-ports source destination 00ABC123

crypto keyring tunnel0
  local-address Loopback0
  pre-shared-key address 192.168.2.37 key cisco
crypto keyring tunnel1
  local-address Loopback1
  pre-shared-key address 192.168.2.129 key cisco
crypto keyring tunnel2
  local-address Loopback2
  pre-shared-key address 192.168.2.36 key cisco
crypto keyring tunnel3
  local-address Loopback3
  pre-shared-key address 192.168.2.128 key cisco
crypto keyring tunnel4
  local-address Loopback4
  pre-shared-key address 192.168.2.39 key cisco
crypto keyring tunnel5
  local-address Loopback5
  pre-shared-key address 192.168.2.131 key cisco
crypto keyring tunnel6
  local-address Loopback6
  pre-shared-key address 192.168.2.38 key cisco
crypto keyring tunnel7
  local-address Loopback7
  pre-shared-key address 192.168.2.130 key cisco

crypto isakmp policy 200
 encryption aes
 hash sha
 authentication pre-share
 group 16
 lifetime 28800
```

```
crypto isakmp profile isakmp-tunnel0
   keyring tunnel0
   match identity address 0.0.0.0
   local-address Loopback0
crypto isakmp profile isakmp-tunnel1
   keyring tunnel1
   match identity address 0.0.0.0
   local-address Loopback1
crypto isakmp profile isakmp-tunnel2
   keyring tunnel2
   match identity address 0.0.0.0
   local-address Loopback2
crypto isakmp profile isakmp-tunnel3
   keyring tunnel3
   match identity address 0.0.0.0
   local-address Loopback3
crypto isakmp profile isakmp-tunnel4
   keyring tunnel4
   match identity address 0.0.0.0
   local-address Loopback4
crypto isakmp profile isakmp-tunnel5
   keyring tunnel5
   match identity address 0.0.0.0
   local-address Loopback5
crypto isakmp profile isakmp-tunnel6
   keyring tunnel6
   match identity address 0.0.0.0
   local-address Loopback6
crypto isakmp profile isakmp-tunnel7
   keyring tunnel7
   match identity address 0.0.0.0
   local-address Loopback7

crypto ipsec transform-set ipsec-prop-vpn-tunnel esp-gcm 256
 mode tunnel
crypto ipsec df-bit clear

crypto ipsec profile ipsec-vpn-tunnel
 set transform-set ipsec-prop-vpn-tunnel
 set pfs group16

interface Loopback0
 ip address 192.168.1.37 255.255.255.255
!
interface Loopback1
 ip address 192.168.1.129 255.255.255.255
!
interface Loopback2
 ip address 192.168.1.36 255.255.255.255
!
interface Loopback3
 ip address 192.168.1.128 255.255.255.255
!
interface Loopback4
 ip address 192.168.1.39 255.255.255.255
!
interface Loopback5
 ip address 192.168.1.131 255.255.255.255
!
interface Loopback6
 ip address 192.168.1.38 255.255.255.255
```

```
!
interface Loopback7
 ip address 192.168.1.130 255.255.255.255
!

interface Tunnel0
 ip address 10.101.100.101 255.255.255.0
 load-interval 30
 tunnel source Loopback0
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.37
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel1
 ip address 10.101.101.101 255.255.255.0
 load-interval 30
 tunnel source Loopback1
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.129
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel2
 ip address 10.101.102.101 255.255.255.0
 load-interval 30
 tunnel source Loopback2
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.36
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel3
 ip address 10.101.103.101 255.255.255.0
 load-interval 30
 tunnel source Loopback3
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.128
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel4
 ip address 10.101.104.101 255.255.255.0
 load-interval 30
 tunnel source Loopback4
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.39
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel5
 ip address 10.101.105.101 255.255.255.0
 load-interval 30
 tunnel source Loopback5
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.131
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel6
 ip address 10.101.106.101 255.255.255.0
 load-interval 30
 tunnel source Loopback6
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.38
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel7
```

```
 ip address 10.101.107.101 255.255.255.0
 load-interval 30
 tunnel source Loopback7
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.130
 tunnel protection ipsec profile ipsec-vpn-tunnel
!


interface GigabitEthernet2
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet3
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!

 !  ### IP route from servers to c8kv-uut

ip route 10.1.0.0 255.255.0.0 GigabitEthernet2 10.0.1.10

!  ### IP routes from c8kv-uut to clients on c8kv-peer side, routes are evenly distributed to all 8 TXQ

ip route 10.10.0.0 255.255.0.0 Tunnel0
ip route 10.10.0.0 255.255.0.0 Tunnel1
ip route 10.10.0.0 255.255.0.0 Tunnel2
ip route 10.10.0.0 255.255.0.0 Tunnel3
ip route 10.10.0.0 255.255.0.0 Tunnel4
ip route 10.10.0.0 255.255.0.0 Tunnel5
ip route 10.10.0.0 255.255.0.0 Tunnel6
ip route 10.10.0.0 255.255.0.0 Tunnel7

 !  ### IP route from c8kv-uut Loopback int tunnel endpoint to c8kv-peer Loopback int tunnel endpoints

ip route 192.168.2.0 255.255.255.0 GigabitEthernet3 10.0.2.30
```

# Voorbeeldtopologie en CLI-configuratie met behulp van 12 TXQ's met loopback-interfaces




Afbeelding 4. Voorbeeldtopologie die twaalf TxQs gebruikt met behulp van Loopback-interfaces.

Dit is een voorbeeld CLI-configuratie voor 'c8kv-uut' (Afbeelding 4) die twaalf IPsec-tunnels met Loopback-interfaces maakt met behulp van de berekende hashed IP-adressen (192.168.1.X) uit de vorige sectie.

Een soortgelijke configuratie zou van toepassing zijn op het andere routereindpunt (c8kv-peer) met de resterende acht berekende hashed IP-adressen (192.168.2.X).

```
ip cef load-sharing algorithm include-ports source destination 00ABC123

crypto keyring tunnel0
  local-address Loopback0
  pre-shared-key address 192.168.2.38 key cisco
crypto keyring tunnel1
  local-address Loopback1
  pre-shared-key address 192.168.2.37 key cisco
crypto keyring tunnel2
  local-address Loopback2
  pre-shared-key address 192.168.2.53 key cisco
crypto keyring tunnel3
  local-address Loopback3
  pre-shared-key address 192.168.2.39 key cisco
crypto keyring tunnel4
  local-address Loopback4
  pre-shared-key address 192.168.2.48 key cisco
crypto keyring tunnel5
  local-address Loopback5
  pre-shared-key address 192.168.2.58 key cisco
crypto keyring tunnel6
  local-address Loopback6
  pre-shared-key address 192.168.2.42 key cisco
crypto keyring tunnel7
  local-address Loopback7
  pre-shared-key address 192.168.2.46 key cisco
crypto keyring tunnel8
  local-address Loopback8
  pre-shared-key address 192.168.2.40 key cisco
crypto keyring tunnel9
  local-address Loopback9
  pre-shared-key address 192.168.2.43 key cisco
```

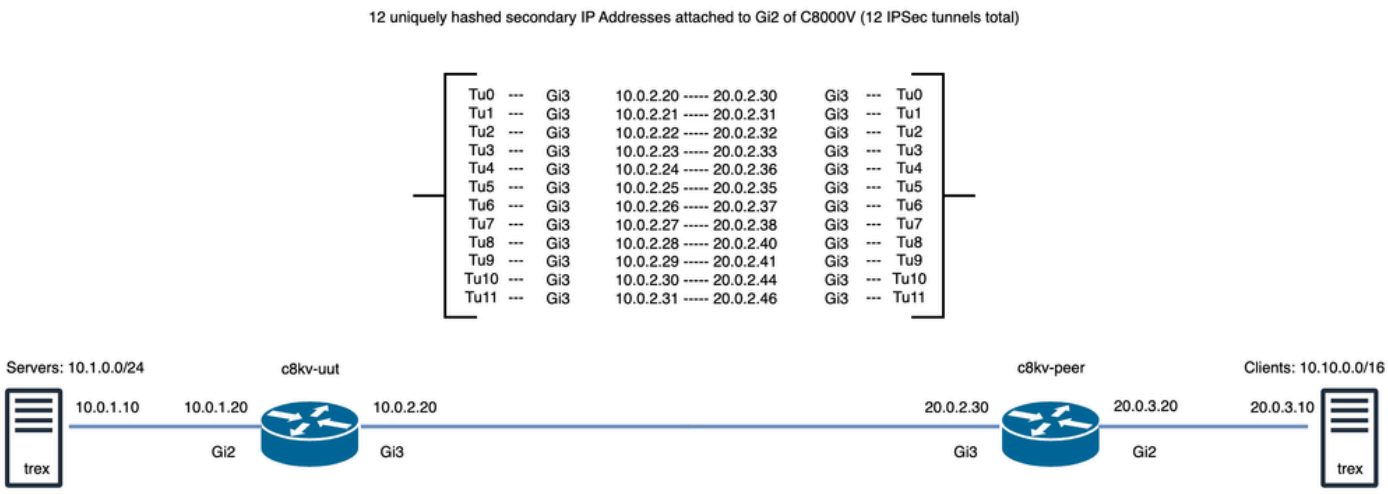
```
crypto keyring tunnel10
  local-address Loopback10
  pre-shared-key address 192.168.2.36 key cisco
crypto keyring tunnel11
  local-address Loopback11
  pre-shared-key address 192.168.2.56 key cisco


crypto isakmp policy 200
 encryption aes
 hash sha
 authentication pre-share
 group 16
 lifetime 28800
crypto isakmp profile isakmp-tunnel0
   keyring tunnel0
   match identity address 0.0.0.0
   local-address Loopback0
crypto isakmp profile isakmp-tunnel1
   keyring tunnel1
   match identity address 0.0.0.0
   local-address Loopback1
crypto isakmp profile isakmp-tunnel2
   keyring tunnel2
   match identity address 0.0.0.0
   local-address Loopback2
crypto isakmp profile isakmp-tunnel3
   keyring tunnel3
   match identity address 0.0.0.0
   local-address Loopback3
crypto isakmp profile isakmp-tunnel4
   keyring tunnel4
   match identity address 0.0.0.0
   local-address Loopback4
crypto isakmp profile isakmp-tunnel5
   keyring tunnel5
   match identity address 0.0.0.0
   local-address Loopback5
crypto isakmp profile isakmp-tunnel6
   keyring tunnel6
   match identity address 0.0.0.0
   local-address Loopback6
crypto isakmp profile isakmp-tunnel7
   keyring tunnel7
   match identity address 0.0.0.0
   local-address Loopback7
crypto isakmp profile isakmp-tunnel8
   keyring tunnel8
   match identity address 0.0.0.0
   local-address Loopback8
crypto isakmp profile isakmp-tunnel9
   keyring tunnel9
   match identity address 0.0.0.0
   local-address Loopback9
crypto isakmp profile isakmp-tunnel10
   keyring tunnel10
   match identity address 0.0.0.0
   local-address Loopback10
crypto isakmp profile isakmp-tunnel11
   keyring tunnel11
   match identity address 0.0.0.0
   local-address Loopback11
```

```
crypto ipsec transform-set ipsec-prop-vpn-tunnel esp-gcm 256
 mode tunnel
crypto ipsec df-bit clear

crypto ipsec profile ipsec-vpn-tunnel
 set transform-set ipsec-prop-vpn-tunnel
 set pfs group16

interface Loopback0
 ip address 192.168.1.38 255.255.255.255
!
interface Loopback1
 ip address 192.168.1.37 255.255.255.255
!
interface Loopback2
 ip address 192.168.1.53 255.255.255.255
!
interface Loopback3
 ip address 192.168.1.39 255.255.255.255
!
interface Loopback4
 ip address 192.168.1.48 255.255.255.255
!
interface Loopback5
 ip address 192.168.1.58 255.255.255.255
!
interface Loopback6
 ip address 192.168.1.42 255.255.255.255
!
interface Loopback7
 ip address 192.168.1.46 255.255.255.255
!
interface Loopback8
 ip address 192.168.1.40 255.255.255.255
!
interface Loopback9
 ip address 192.168.1.43 255.255.255.255
!
interface Loopback10
 ip address 192.168.1.36 255.255.255.255
!
interface Loopback11
 ip address 192.168.1.56 255.255.255.255

interface Tunnel0
 ip address 10.101.100.101 255.255.255.0
 load-interval 30
 tunnel source Loopback0
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.38
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel1
 ip address 10.101.101.101 255.255.255.0
 load-interval 30
 tunnel source Loopback1
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.37
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel2
```

```
 ip address 10.101.102.101 255.255.255.0
 load-interval 30
 tunnel source Loopback2
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.53
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel3
 ip address 10.101.103.101 255.255.255.0
 load-interval 30
 tunnel source Loopback3
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.39
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel4
 ip address 10.101.104.101 255.255.255.0
 load-interval 30
 tunnel source Loopback4
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.48
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel5
 ip address 10.101.105.101 255.255.255.0
 load-interval 30
 tunnel source Loopback5
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.58
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel6
 ip address 10.101.106.101 255.255.255.0
 load-interval 30
 tunnel source Loopback6
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.42
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel7
 ip address 10.101.107.101 255.255.255.0
 load-interval 30
 tunnel source Loopback7
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.46
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel8
 ip address 10.101.108.101 255.255.255.0
 load-interval 30
 tunnel source Loopback8
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.40
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel9
 ip address 10.101.109.101 255.255.255.0
 load-interval 30
 tunnel source Loopback9
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.43
 tunnel protection ipsec profile ipsec-vpn-tunnel
```

```
!
interface Tunnel10
 ip address 10.101.110.101 255.255.255.0
 load-interval 30
 tunnel source Loopback10
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.36
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel11
 ip address 10.101.111.101 255.255.255.0
 load-interval 30
 tunnel source Loopback11
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.56
 tunnel protection ipsec profile ipsec-vpn-tunnel


interface GigabitEthernet2
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet3
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!

  !  ### IP route from c8kv-uut to local servers

ip route 10.1.0.0 255.255.0.0 GigabitEthernet2 10.0.1.10

!  ### IP routes from c8kv-uut to clients on c8kv-peer side, routes are evenly distributed to all 12 TX

ip route 10.10.0.0 255.255.0.0 Tunnel0
ip route 10.10.0.0 255.255.0.0 Tunnel1
ip route 10.10.0.0 255.255.0.0 Tunnel2
ip route 10.10.0.0 255.255.0.0 Tunnel3
ip route 10.10.0.0 255.255.0.0 Tunnel4
ip route 10.10.0.0 255.255.0.0 Tunnel5
ip route 10.10.0.0 255.255.0.0 Tunnel6
ip route 10.10.0.0 255.255.0.0 Tunnel7
ip route 10.10.0.0 255.255.0.0 Tunnel8
ip route 10.10.0.0 255.255.0.0 Tunnel9
ip route 10.10.0.0 255.255.0.0 Tunnel10
ip route 10.10.0.0 255.255.0.0 Tunnel11

 !  ### IP route from c8kv-uut Loopback int tunnel endpoint to c8kv-peer Loopback int tunnel endpoints

ip route 192.168.2.0 255.255.255.0 GigabitEthernet3 10.0.2.30
```

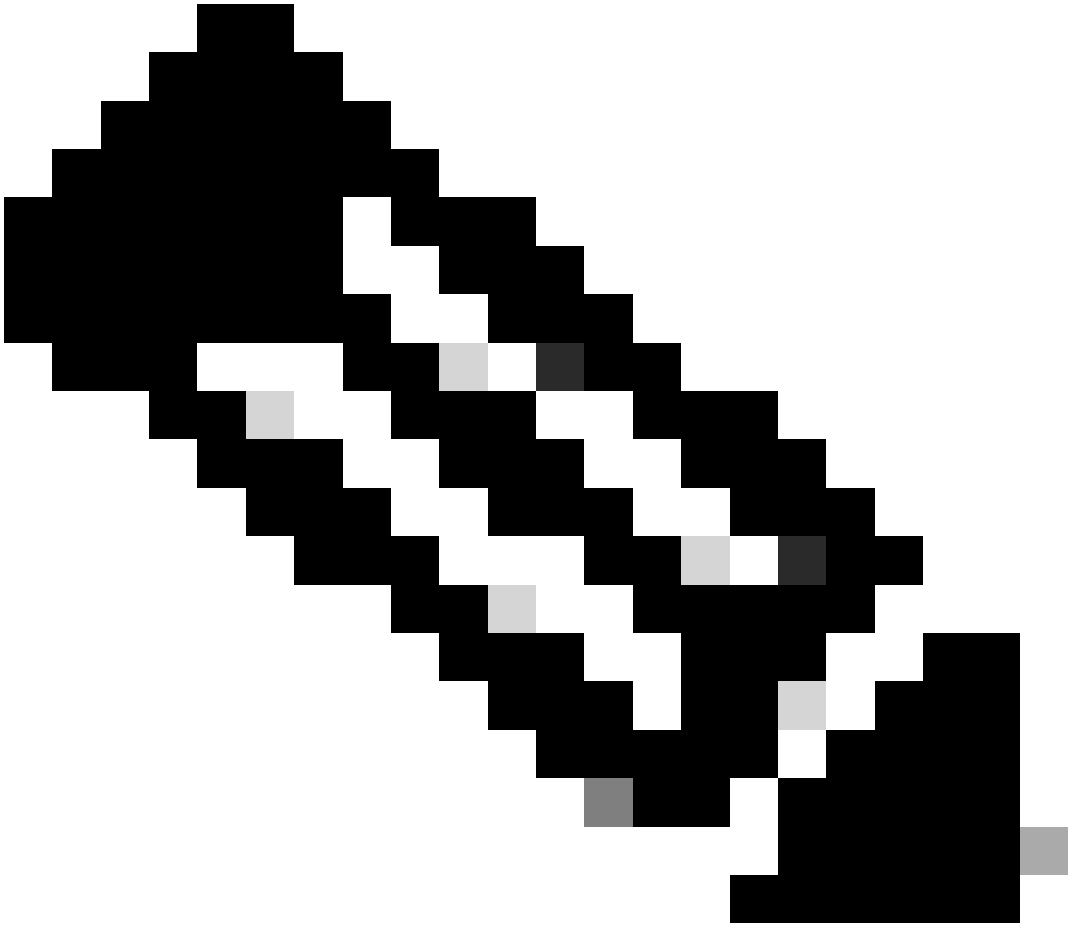
# Voorbeeldtopologie en CLI-configuratie met 12 TXQ's met secundaire IP-adressen




Afbeelding 5. Voorbeeldtopologie die twaalf TxQs gebruikt met behulp van secundaire IP-adressen.

Als Loopback-adressen niet kunnen worden gebruikt in de AWS-omgeving, kunnen in plaats daarvan secundaire IP-adressen worden gebruikt die aan de ENI zijn gekoppeld.

Dit is een voorbeeld van een CLI-configuratie voor 'c8kv-uut' (Afbeelding 5) die twaalf IPsec-tunnels maakt met als bron 1 primair IP-adres + 11 secundaire IP-adressen die zijn aangesloten op de GigabitEthernet3-interface met behulp van berekende hashed IP-adressen (10.0.2.X). Een soortgelijke configuratie zou van toepassing zijn op het andere routereindpunt (c8kv-peer) met de resterende twaalf berekende hashte IP-adressen (20.0.2.X).



Opmerking: In dit voorbeeld gebruiken we een tweede C8000V als tunneleindpunt, maar andere eindpunten voor cloudnetwerken zoals TGW of DX kunnen ook worden gebruikt.

```
ip cef load-sharing algorithm include-ports source destination 00ABC123

crypto keyring tunnel0
  local-address 10.0.2.20
  pre-shared-key address 20.0.2.30 key cisco
crypto keyring tunnel1
  local-address 10.0.2.21
  pre-shared-key address 20.0.2.31 key cisco
crypto keyring tunnel2
  local-address 10.0.2.22
  pre-shared-key address 20.0.2.32 key cisco
crypto keyring tunnel3
  local-address 10.0.2.23
  pre-shared-key address 20.0.2.33 key cisco
crypto keyring tunnel4
  local-address 10.0.2.24
  pre-shared-key address 20.0.2.36 key cisco
crypto keyring tunnel5
```

```
  local-address 10.0.2.25
  pre-shared-key address 20.0.2.35 key cisco
crypto keyring tunnel6
  local-address 10.0.2.26
  pre-shared-key address 20.0.2.37 key cisco
crypto keyring tunnel7
  local-address 10.0.2.27
  pre-shared-key address 20.0.2.38 key cisco
crypto keyring tunnel8
  local-address 10.0.2.28
  pre-shared-key address 20.0.2.40 key cisco
crypto keyring tunnel9
  local-address 10.0.2.29
  pre-shared-key address 20.0.2.41 key cisco
crypto keyring tunnel10
  local-address 10.0.2.30
  pre-shared-key address 20.0.2.44 key cisco
crypto keyring tunnel11
  local-address 10.0.2.31
  pre-shared-key address 20.0.2.46 key cisco


crypto isakmp policy 200
 encryption aes
 hash sha
 authentication pre-share
 group 16
 lifetime 28800
crypto isakmp profile isakmp-tunnel0
   keyring tunnel0
   match identity address 20.0.2.30 255.255.255.255
   local-address 10.0.2.20
crypto isakmp profile isakmp-tunnel1
   keyring tunnel1
   match identity address 20.0.2.31 255.255.255.255
   local-address 10.0.2.21
crypto isakmp profile isakmp-tunnel2
   keyring tunnel2
   match identity address 20.0.2.32 255.255.255.255
   local-address 10.0.2.22
crypto isakmp profile isakmp-tunnel3
   keyring tunnel3
   match identity address 20.0.2.33 255.255.255.255
   local-address 10.0.2.23
crypto isakmp profile isakmp-tunnel4
   keyring tunnel4
   match identity address 20.0.2.36 255.255.255.255
   local-address 10.0.2.24
crypto isakmp profile isakmp-tunnel5
   keyring tunnel5
   match identity address 20.0.2.35 255.255.255.255
   local-address 10.0.2.25
crypto isakmp profile isakmp-tunnel6
   keyring tunnel6
   match identity address 20.0.2.37 255.255.255.255
   local-address 10.0.2.26
crypto isakmp profile isakmp-tunnel7
   keyring tunnel7
   match identity address 20.0.2.38 255.255.255.255
   local-address 10.0.2.27
crypto isakmp profile isakmp-tunnel8
   keyring tunnel8
```

```
   match identity address 20.0.2.40 255.255.255.255
   local-address 10.0.2.28
crypto isakmp profile isakmp-tunnel9
   keyring tunnel9
   match identity address 20.0.2.41 255.255.255.255
   local-address 10.0.2.29
crypto isakmp profile isakmp-tunnel10
   keyring tunnel10
   match identity address 20.0.2.44 255.255.255.255
   local-address 10.0.2.30
crypto isakmp profile isakmp-tunnel11
   keyring tunnel11
   match identity address 20.0.2.46 255.255.255.255
   local-address 10.0.2.31


crypto ipsec transform-set ipsec-prop-vpn-tunnel esp-gcm 256
 mode tunnel
crypto ipsec df-bit clear

crypto ipsec profile ipsec-vpn-tunnel
 set transform-set ipsec-prop-vpn-tunnel
 set pfs group16

interface Tunnel0
 ip address 10.101.100.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.20
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.30
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel1
 ip address 10.101.101.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.21
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.31
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel2
 ip address 10.101.102.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.22
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.32
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel3
 ip address 10.101.103.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.23
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.33
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel4
 ip address 10.101.104.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.24
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.36
```

```
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel5
 ip address 10.101.105.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.25
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.35
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel6
 ip address 10.101.106.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.26
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.37
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel7
 ip address 10.101.107.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.27
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.38
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel8
 ip address 10.101.108.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.28
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.40
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel9
 ip address 10.101.109.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.29
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.41
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel10
 ip address 10.101.110.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.30
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.44
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel11
 ip address 10.101.111.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.31
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.46
 tunnel protection ipsec profile ipsec-vpn-tunnel
!


interface GigabitEthernet2
 mtu 9216
```

```
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet3
 mtu 9216
 ip address 10.0.2.20 255.255.255.0
 ip address 10.0.2.21 255.255.255.0 secondary
 ip address 10.0.2.22 255.255.255.0 secondary
 ip address 10.0.2.23 255.255.255.0 secondary
 ip address 10.0.2.24 255.255.255.0 secondary
 ip address 10.0.2.25 255.255.255.0 secondary
 ip address 10.0.2.26 255.255.255.0 secondary
 ip address 10.0.2.27 255.255.255.0 secondary
 ip address 10.0.2.28 255.255.255.0 secondary
 ip address 10.0.2.29 255.255.255.0 secondary
 ip address 10.0.2.30 255.255.255.0 secondary
 ip address 10.0.2.31 255.255.255.0 secondary
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!

  !  ### IP route from c8kv-uut to local servers

ip route 10.1.0.0 255.255.255.0 GigabitEthernet2 10.0.1.10

!  ### IP routes from c8kv-uut to clients on c8kv-peer side, routes are evenly distributed to all 12 TX

ip route 10.10.0.0 255.255.0.0 Tunnel0
ip route 10.10.0.0 255.255.0.0 Tunnel1
ip route 10.10.0.0 255.255.0.0 Tunnel2
ip route 10.10.0.0 255.255.0.0 Tunnel3
ip route 10.10.0.0 255.255.0.0 Tunnel4
ip route 10.10.0.0 255.255.0.0 Tunnel5
ip route 10.10.0.0 255.255.0.0 Tunnel6
ip route 10.10.0.0 255.255.0.0 Tunnel7
ip route 10.10.0.0 255.255.0.0 Tunnel8
ip route 10.10.0.0 255.255.0.0 Tunnel9
ip route 10.10.0.0 255.255.0.0 Tunnel10
ip route 10.10.0.0 255.255.0.0 Tunnel11


!  ### IP route from c8kv-uut Gi3 int tunnel endpoint to c8kv-peer Gi3

int tunnel endpoints (secondary IP addresses on c8kv-peer side)

ip route 20.0.2.30 255.255.255.255 10.0.2.1
ip route 20.0.2.31 255.255.255.255 10.0.2.1
ip route 20.0.2.32 255.255.255.255 10.0.2.1
ip route 20.0.2.33 255.255.255.255 10.0.2.1
ip route 20.0.2.36 255.255.255.255 10.0.2.1
ip route 20.0.2.35 255.255.255.255 10.0.2.1
ip route 20.0.2.37 255.255.255.255 10.0.2.1
ip route 20.0.2.38 255.255.255.255 10.0.2.1
ip route 20.0.2.40 255.255.255.255 10.0.2.1
ip route 20.0.2.41 255.255.255.255 10.0.2.1
```

```
ip route 20.0.2.44 255.255.255.255 10.0.2.1
ip route 20.0.2.46 255.255.255.255 10.0.2.1
```

# Typische Catalyst 8000V-implementatie in AWS

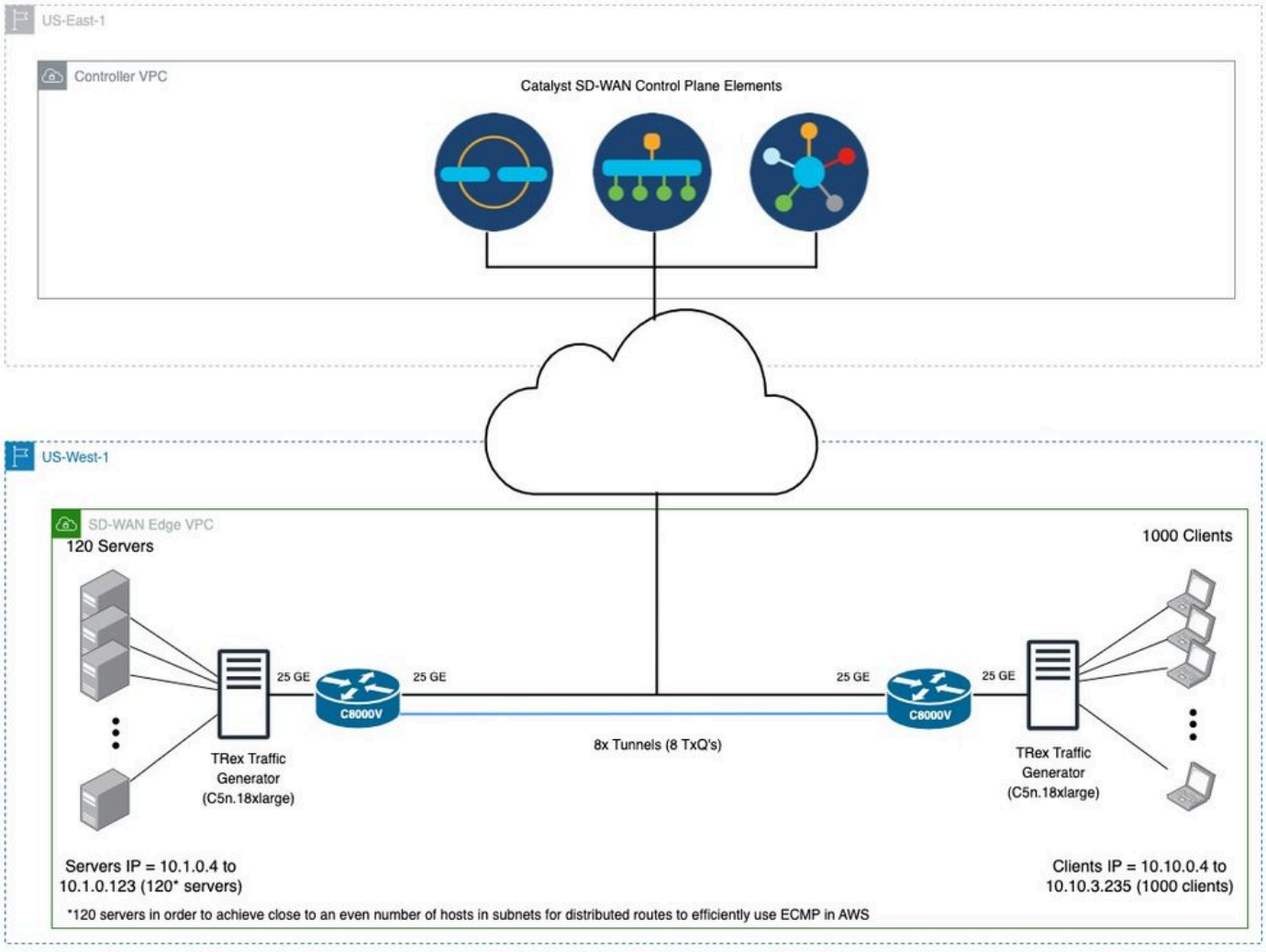
## autonome modus

Zie de vorige voorbeelden van CLI-configuraties en topologieën. CLI-configuratie kan worden gekopieerd en gewijzigd op basis van netwerkadresseringsschema en gegenereerde hashed IP-adressen.

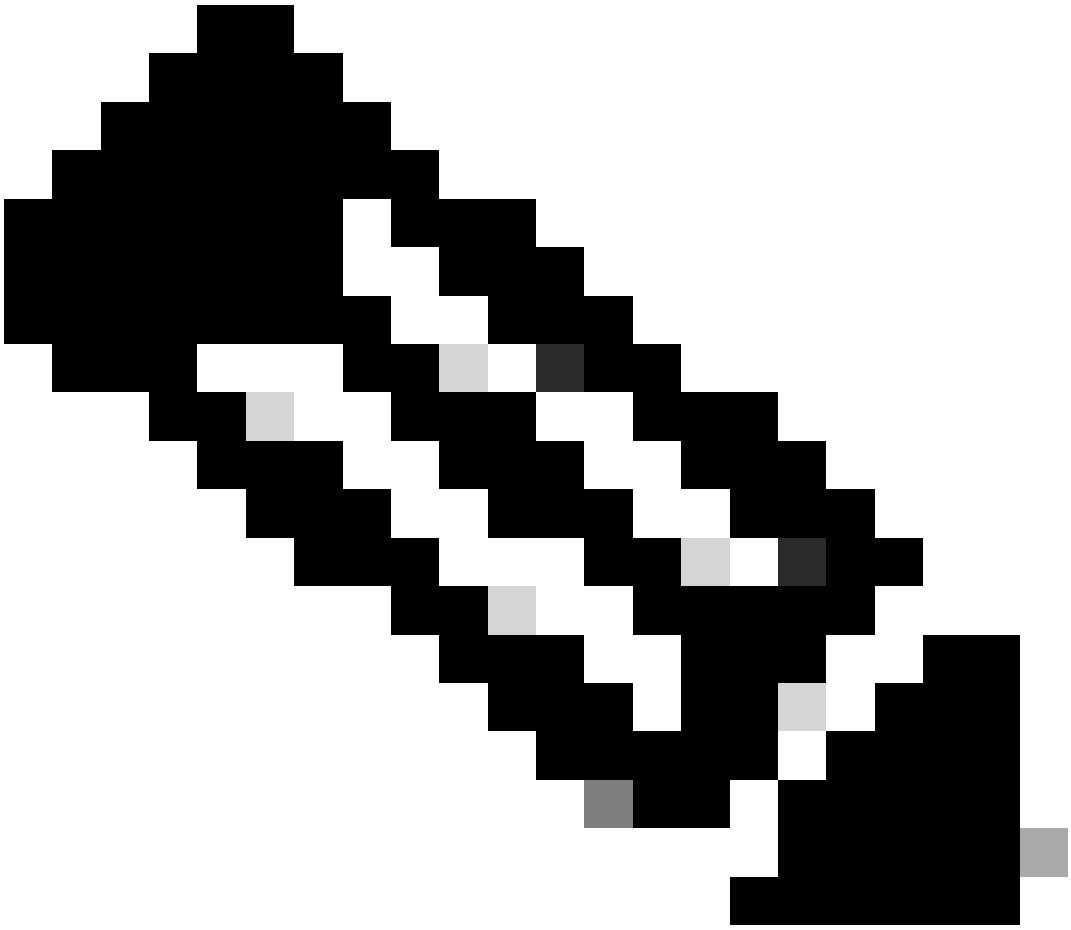
Voor een succesvolle tunnelcreatie moet u IP-routes maken op zowel de C8000V als routeringstabellen op AWS VPC.

## SD-WAN-modus

Dit is een voorbeeldtopologie en SD-WAN-configuratie die TLOC's maakt met behulp van Loopback-interfaces op C8000V's die zich in een AWS VPC bevinden.



Afbeelding 6. Voorbeeld van SD-WAN-topologie die TLOC's gebruikt met Loopback-interfaces op C8000V's die zich in een AWS VPC bevinden.



> Opmerking: in afbeelding 6 vertegenwoordigt de zwartgekleurde verbinding Control (VPN0)-verbinding tussen elementen van het SD-WAN-besturingsvlak en SD-WAN-randapparaten. Blauwkleurige verbindingen vertegenwoordigen tunnels tussen de twee SD-WAN edge-apparaten met behulp van TLOC's.

U kunt een voorbeeld van een SD-WAN CLI-configuratie vinden voor Afbeelding 6 (hier).

```
csr_uut#show sdwan run
system
system-ip             29.173.249.161
site-id               5172
admin-tech-on-failure
sp-organization-name  SP_ORG_NAME
organization-name     ORG_NAME
upgrade-confirm       15
```

```
vbond X.X.X.X
!
memory free low-watermark processor 68484
service timestamps debug datetime msec
service timestamps log datetime msec
no service tcp-small-servers
no service udp-small-servers
platform console virtual
platform qfp utilization monitor load 80
platform punt-keepalive disable-kernel-core
hostname csr_uut
username ec2-user privilege 15 secret 5 $1$4P16$..ag88eFsOMLIemjNcWSt0
vrf definition 11
address-family ipv4
exit-address-family
!
address-family ipv6
exit-address-family
!
!
vrf definition Mgmt-intf
address-family ipv4
exit-address-family
!
address-family ipv6
exit-address-family
!
!
no ip finger
no ip rcmd rcp-enable
no ip rcmd rsh-enable
no ip dhcp use class
ip route 0.0.0.0 0.0.0.0 X.X.X.X
ip route 0.0.0.0 0.0.0.0 X.X.X.X
ip route 0.0.0.0 0.0.0.0 X.X.X.X
ip route vrf 11 10.1.0.0 255.255.0.0 X.X.X.X
ip route vrf Mgmt-intf 0.0.0.0 0.0.0.0 X.X.X.X
no ip source-route
ip ssh pubkey-chain
username ec2-user
key-hash ssh-rsa 353158c28c7649710b3c933da02e384b ec2-user
!
!
!
no ip http server
ip http secure-server
ip nat settings central-policy
ip nat settings gatekeeper-size 1024
ipv6 unicast-routing
class-map match-any class0
match dscp 1
!
class-map match-any class1
match dscp 2
!
class-map match-any class2
match dscp 3
!
class-map match-any class3
match dscp 4
!
class-map match-any class4
```

```
match dscp 5
!
class-map match-any class5
match dscp 6
!
class-map match-any class6
match dscp 7
!
class-map match-any class7
match dscp 8
!
policy-map qos_map1
class class0
priority percent 20
!
class class1
bandwidth percent 18
random-detect
!
class class2
bandwidth percent 15
random-detect
!
class class3
bandwidth percent 12
random-detect
!
class class4
bandwidth percent 10
random-detect
!
class class5
bandwidth percent 10
random-detect
!
class class6
bandwidth percent 10
random-detect
!
class class7
bandwidth percent 5
random-detect
!
!
interface GigabitEthernet1
no shutdown
ip address dhcp
no mop enabled
no mop sysid
negotiation auto
exit
interface GigabitEthernet2
no shutdown
ip address dhcp
load-interval 30
speed 10000
no negotiation auto
service-policy output qos_map1
exit
interface GigabitEthernet3
shutdown
ip address dhcp
```

```
load-interval 30
speed 10000
no negotiation auto
exit
interface GigabitEthernet4
no shutdown
vrf forwarding 11
ip address X.X.X.X 255.255.255.0
load-interval 30
speed 10000
no negotiation auto
exit
interface Loopback1
no shutdown
ip address 192.168.1.21 255.255.255.255
exit
interface Loopback2
no shutdown
ip address 192.168.1.129 255.255.255.255
exit
interface Loopback3
no shutdown
ip address 192.168.1.20 255.255.255.255
exit
interface Loopback4
no shutdown
ip address 192.168.1.128 255.255.255.255
exit
interface Loopback5
no shutdown
ip address 192.168.1.23 255.255.255.255
exit
interface Loopback6
no shutdown
ip address 192.168.1.131 255.255.255.255
exit
interface Loopback7
no shutdown
ip address 192.168.1.22 255.255.255.255
exit
interface Loopback8
no shutdown
ip address 192.168.1.130 255.255.255.255
exit
interface Tunnel1
no shutdown
ip unnumbered GigabitEthernet1
tunnel source GigabitEthernet1
tunnel mode sdwan
exit
interface Tunnel14095001
no shutdown
ip unnumbered Loopback1
no ip redirects
ipv6 unnumbered Loopback1
no ipv6 redirects
tunnel source Loopback1
tunnel mode sdwan
exit
interface Tunnel14095002
no shutdown
ip unnumbered Loopback2
```

```
no ip redirects
ipv6 unnumbered Loopback2
no ipv6 redirects
tunnel source Loopback2
tunnel mode sdwan
exit
interface Tunnel14095003
no shutdown
ip unnumbered Loopback3
no ip redirects
ipv6 unnumbered Loopback3
no ipv6 redirects
tunnel source Loopback3
tunnel mode sdwan
exit
interface Tunnel14095004
no shutdown
ip unnumbered Loopback4
no ip redirects
ipv6 unnumbered Loopback4
no ipv6 redirects
tunnel source Loopback4
tunnel mode sdwan
exit
interface Tunnel14095005
no shutdown
ip unnumbered Loopback5
no ip redirects
ipv6 unnumbered Loopback5
no ipv6 redirects
tunnel source Loopback5
tunnel mode sdwan
exit
interface Tunnel14095006
no shutdown
ip unnumbered Loopback6
no ip redirects
ipv6 unnumbered Loopback6
no ipv6 redirects
tunnel source Loopback6
tunnel mode sdwan
exit
interface Tunnel14095007
no shutdown
ip unnumbered Loopback7
no ip redirects
ipv6 unnumbered Loopback7
no ipv6 redirects
tunnel source Loopback7
tunnel mode sdwan
exit
interface Tunnel14095008
no shutdown
ip unnumbered Loopback8
no ip redirects
ipv6 unnumbered Loopback8
no ipv6 redirects
tunnel source Loopback8
tunnel mode sdwan
exit
no logging console
aaa authentication enable default enable
```

```
aaa authentication login default local
aaa authorization console
aaa authorization exec default local none
login on-success log
license smart transport smart
license smart url https://smartreceiver.cisco.com/licservice/license
line aux 0
!
line con 0
stopbits 1
!
line vty 0 4
transport input ssh
!
line vty 5 80
transport input ssh
!
sdwan
interface GigabitEthernet1
tunnel-interface
encapsulation ipsec
color private1 restrict
allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface GigabitEthernet2
exit
interface GigabitEthernet3
exit
interface Loopback1
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private2 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
```

```
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback2
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private3 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback3
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private4 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
```

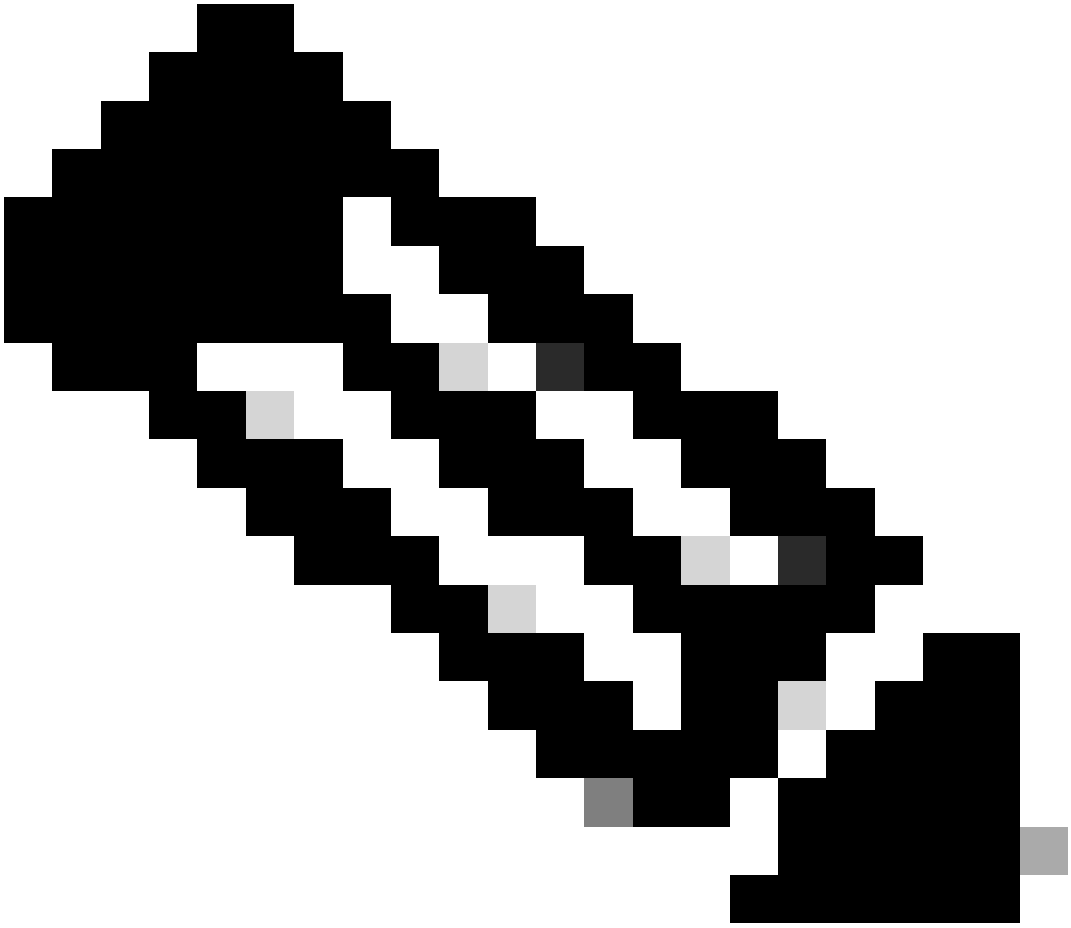
```
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback4
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private5 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback5
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private6 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
```

```
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback6
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color red restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback7
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color blue restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
```

```
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback8
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color green restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
appqoe
no tcpopt enable
no dreopt enable
no httpopt enable
!
omp
no shutdown
send-path-limit  16
ecmp-limit       16
graceful-restart
no as-dot-notation
timers
graceful-restart-timer 43200
exit
address-family ipv4
advertise connected
advertise static
!
address-family ipv6
advertise connected
advertise static
```

```
!
!
!
security
ipsec
replay-window  8192
integrity-type ip-udp-esp esp
!
!
sslproxy
no enable
rsa-key-modulus      2048
certificate-lifetime 730
eckey-type           P256
ca-tp-label          PROXY-SIGNING-CA
settings expired-certificate  drop
settings untrusted-certificate drop
settings unknown-status       drop
settings certificate-revocation-check none
settings unsupported-protocol-versions drop
settings unsupported-cipher-suites drop
settings failure-mode         close
settings minimum-tls-ver      TLSv1
dual-side optimization enable
!
policy
app-visibility
flow-visibility
!
```

# Problemen met doorvoerprestaties oplossen in AWS



Opmerking: het uitvoeren van prestatietests in openbare cloudomgevingen introduceert nieuwe variabelen die de doorvoerprestaties kunnen beïnvloeden. Hier zijn een paar om te overwegen tijdens het uitvoeren van dit soort tests:

- Onderliggend gebruik van bronnen door de peers op het moment van het uitvoeren van de tests
- Geen gebruik van dedicated hosts (gebruik van dedicated hosts verhoogt de kosten van de cloud met 16x)
- Cloud draait in verschillende regio's, de prestaties kunnen variëren
- In sommige gevallen zijn de cijfers vergelijkbaar, ongeacht het functieprofiel; dit is mogelijk te wijten aan AWS-beperking op interface per instantiegrootte
- AWS gooit pakketten per seconde af op EC2-instanties, wat ook kan leiden tot pakketverlies
- AWS maakt de throttling rate niet bekend, maar daalt als gevolg van de pps throttling kan worden waargenomen via de 'pps_allowance_exceeded' teller

## Handige CLI-opdrachten voor probleemoplossing

Bij het uitvoeren van doorvoerprestatietests kunnen deze opdrachten voor probleemoplossing worden gebruikt om knelpunten of redenen voor prestatievermindering aan te wijzen.

"Toon Platform Hardware QFP Active Statistics Drop" - hiermee kunnen we zien of er een daling is op de C8KV. We moeten ervoor zorgen dat er geen significante staartdruppels of relevante tellers toenemen.

"toon platform hardware qfp actieve statistieken drop clear" - Deze opdracht wist de tellers.

"show platform hardware qfp active datapath infrastructure sw-cio" - Deze opdracht geeft ons gedetailleerde informatie over het percentage Packet Processor (PP), Traffic Manager (TM) dat tijdens de uitvoering wordt gebruikt. Dit stelt ons in staat om te bepalen of er voldoende verwerkingscapaciteit is of niet van de c8kv.

"show platform hardware qfp active datapath util summary" - Deze opdracht geeft ons de volledige informatie van de invoer/uitvoer die de c8kv verzendt/ontvangt van alle poorten.

Controleer de invoer-/uitvoersnelheid en kijk of er een daling is. Controleer ook het percentage verwerkingsbelasting. Als het 100% bereikt, betekent dit dat de c8kv zijn capaciteit heeft bereikt.

"Toon platte hardware qfp active infrastructure bqs interface GigabitEthernetX" - Met deze opdracht kunnen we de statistieken op interfaceniveau controleren in termen van het wachtrijnummer, bandbreedte, achteruitval.

"show controller" - Deze opdracht geeft ons veel gedetailleerde informatie over de rx / tx goede pakketten, gemiste pakketten.

Deze opdracht kan worden gebruikt in een scenario waarin we geen staartdalingen zien, maar de verkeersgenerator ons nog steeds laat vallen.

Dit kan gebeuren in een scenario waarbij het datagebruik al 100% bereikt en ook de PP op 100%.

Als de rx_missed_errors-tellers blijven toenemen, betekent dit dat de CSR de cloudinfrastructuur onder druk zet omdat deze geen verkeer meer kan verwerken.

"show platform hardware qfp active datapath infrastructure sw-hqf" - kan worden gebruikt om te controleren of er congestie optreedt als gevolg van de tegendruk van AWS.

"show plat hardware qfp active datapath infrastructure sw-nic" - Bepaalt hoe de belasting van het verkeer over meerdere wachtrijen wordt verdeeld. Na 17.7 hebben we 8 Multi-TXQ's.

Het kan ook bepalen of er een bepaalde wachtrij is die al het verkeer afneemt of de belasting goed wordt verdeeld.

"show controllers | in errors|exceeded|Giga" - Toont de pakketdruppels als gevolg van pps-throttling gedaan vanaf AWS-zijde, die kan worden waargenomen via pps_allowance_exceeded-teller.

# Voorbeeld CLI-uitvoer

Voorbeelduitvoer waarbij de teller van de staartdruppels blijft toenemen - Geef de opdracht meerdere keren uit om te zien of de tellers toenemen, zodat we kunnen bevestigen dat het inderdaad staartdruppels zijn.

<#root>

```
csr_uut#show platform hardware qfp active statistics drop
Last clearing of QFP drops statistics : never
-----------------------------------------------------------------------
Global Drop Stats Packets Octets
-----------------------------------------------------------------------
Disabled 30 3693
IpFragErr 192 290976
Ipv4NoRoute 43 3626
Ipv6NoRoute 4 224
SdwanImplicitAclDrop 31 3899
```

**TailDrop 19099700 22213834441**

```
UnconfiguredIpv6Fia 3816 419760
```

Voorbeelduitvoer hier weergegeven - Geef de opdracht elke 30 seconden uit om de realtime gegevens op te halen

<#root>

```
csr_uut#show platform hardware qfp active datapath infrastructure sw-cio
Credits Usage:

ID Port Wght Global WRKR0 WRKR1 WRKR2 WRKR3 WRKR4 WRKR5 WRKR6 WRKR7 WRKR8 WRKR9 WRKR10 WRKR11 WRKR12 WR
1 rcl0 16: 455 0 4 1 2 3 2 2 4 4 4 4 0 4 23 512
1 rcl0 32: 496 0 0 0 0 0 0 0 0 0 0 0 0 0 16 512
2 ipc 1: 468 4 2 4 3 0 1 1 4 0 2 0 4 0 18 511
3 vxe_punti 4: 481 0 0 0 0 0 0 0 0 0 0 0 0 0 31 512
4 Gi1 4: 446 0 0 1 1 0 2 3 0 3 2 0 1 1 52 512
5 Gi2 4: 440 4 4 4 3 2 1 1 3 2 4 4 3 2 59 504
6 Gi3 4: 428 1 1 1 0 4 4 1 0 4 4 0 0 2 43 494
7 Gi4 4: 427 1 1 0 1 4 2 0 4 3 4 1 1 7 56 512

Core Utilization over preceding 12819.5863 seconds
-------------------------------------------------
ID: 0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

**% PP**

```
: 6.11 6.23 6.09 6.09 6.04 6.05 6.06 6.07 6.05 6.03 6.04 6.06 0.00 0.00
% RX: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 2.23
```

**% TM:**

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 4.79 0.00
% IDLE: 93.89 93.77 93.91 93.91 93.96 93.95 93.94 93.93 93.95 93.97 93.96 93.94 95.21 97.77
```

Hier weergegeven voorbeeld van uitvoer - Controleer de invoer-/uitvoersnelheid en kijk of er een daling is. Controleer ook het percentage verwerkingsbelasting. Als het 100% bereikt, betekent dit dat de node zijn capaciteit heeft bereikt.

<#root>

```
csr_uut#show platform hardware qfp active datapath util summary
CPP 0: 5 secs 1 min 5 min 60 min

Input: Total (pps)

900215 980887 903176 75623
(bps) 10276623992 11197595912 10310265440 863067008

Output: Total (pps)

900216 937459 865930 72522
(bps) 10276642720 10712432752 9894215928 828417104

Processing: Load (pct)

56 58 54 4
```

Hier weergegeven voorbeelduitvoer voor statistieken op interfaceniveau:

<#root>

```
csr_uut#sh plat hardware qfp active infrastructure bqs interface GigabitEthernet2
Interface: GigabitEthernet2, QFP interface: 7
Queue: QID: 111 (0x6f)
bandwidth (cfg) : 0 , bandwidth (hw) : 1050000000
shape (cfg) : 0 , shape (hw) : 0
prio level (cfg) : 0 , prio level (hw) : n/a
limit (pkts ) : 1043
Statistics:
depth (pkts ) : 0

tail drops (bytes): 0 , (packets) : 0


total enqs (bytes): 459322360227 , (packets) : 374613901
licensed throughput oversubscription drops:
(bytes): 0 , (packets) : 0
Schedule: (SID:0x8a)
Schedule FCID : n/a
bandwidth (cfg) : 10500000000 , bandwidth (hw) : 10500000000
shape (cfg) : 10500000000 , shape (hw) : 10500000000
Schedule: (SID:0x87)
Schedule FCID : n/a
bandwidth (cfg) : 200000000000 , bandwidth (hw) : 200000000000
shape (cfg) : 200000000000 , shape (hw) : 200000000000
Schedule: (SID:0x86)
Schedule FCID : n/a
bandwidth (cfg) : 500000000000 , bandwidth (hw) : 500000000000
shape (cfg) : 500000000000 , shape (hw) : 500000000000
```

```
csr_uut#sh plat hardware qfp active infrastructure bqs interface GigabitEthernet3 | inc tail
tail drops (bytes): 55815791988 , (packets) : 43177643
```

Steekproefuitvoer voor de goede RX/TX-pakketten, gemiste pakketstatistieken

<#root>

```
c8kv-aws-1#show controller
GigabitEthernet1 - Gi1 is mapped to UIO on VXE
rx_good_packets 346
tx_good_packets 243
rx_good_bytes 26440
tx_good_bytes 31813
rx_missed_errors 0
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
tx_q0packets 0
tx_q0bytes 0
GigabitEthernet2 - Gi2 is mapped to UIO on VXE
rx_good_packets 96019317
tx_good_packets 85808651
rx_good_bytes 12483293931
tx_good_bytes 11174853219
```

**rx_missed_errors 522036**

```
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
tx_q0packets 0
tx_q0bytes 0
GigabitEthernet3 - Gi3 is mapped to UIO on VXE
rx_good_packets 171596935
tx_good_packets 191911304
rx_good_bytes 11668588022
tx_good_bytes 13049984257
```

**rx_missed_errors 21356065**

```
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
tx_q0packets 0
tx_q0bytes 0
GigabitEthernet4 - Gi4 is mapped to UIO on VXE
```

```
rx_good_packets 95922932
tx_good_packets 85831238
rx_good_bytes 12470124252
tx_good_bytes 11158486786
```

**rx_missed_errors 520328**

```
rx_errors 46
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
tx_q0packets 0
tx_q0bytes 0
```

Steekproefuitvoer om te controleren of er congestie optreedt als gevolg van de tegendruk van AWS:

<#root>

```
csr_uut#show platform hardware qfp active datapath infrastructure sw-hqf
Name : Pri1 Pri2 None / Inflight pkts
GigabitEthernet4 : XON XON XOFF / 43732

HQF[0] IPC: send 514809 fc 0 congested_cnt 0
HQF[0] recycle: send hi 0 send lo 228030112
fc hi 0 fc lo 0
cong hi 0 cong lo 0
HQF[0] pkt: send hi 433634 send lo 2996661158
fc/full hi 0 fc/full lo 34567275
```

**cong hi 0 cong lo 4572971630**********Congestion counters keep incrementing**

```
HQF[0] aggr send stats 3225639713 aggr send lo state 3225206079
aggr send hi stats 433634
max_tx_burst_sz_hi 0 max_tx_burst_sz_lo 0
HQF[0] gather: failed_to_alloc_b4q 0
HQF[0] ticks 662109543, max ticks accumulated 348
HQF[0] mpsc stats: count: 0
enq 3225683472 enq_spin 0 enq_post 0 enq_flush 0
sig_cnt:0 enq_cancel 0
deq 3225683472 deq_wait 0 deq_fail 0 deq_cancel 0
deq_wait_timeout
```

Voorbeeld van uitvoer voor de verdeling van de verkeerslading over meerdere wachtrijen:

```
um-csr-uut#sh plat hardware qfp active datapath infrastructure sw-nic
pmd b1c5a400 device Gi1
RX: pkts 50258 bytes 4477620 return 0 badlen 0
pkts/burst 1 cycl/pkt 579 ext_cycl/pkt 996
Total ring read 786244055, empty 786197491
```

```
TX: pkts 57860 bytes 6546349
pri-0: pkts 7139 bytes 709042
pkts/send 1
pri-1: pkts 3868 bytes 451352
pkts/send 1
pri-2: pkts 1875 bytes 219403
pkts/send 1
pri-3: pkts 2417 bytes 242527
pkts/send 1
pri-4: pkts 8301 bytes 984022
pkts/send 1
pri-5: pkts 10268 bytes 1114859
pkts/send 1
pri-6: pkts 1740 bytes 175353
pkts/send 1
pri-7: pkts 22252 bytes 2649791
pkts/send 1
Total: pkts/send 1 cycl/pkt 1091
send 56756 sendnow 0
forced 56756 poll 0 thd_poll 0
blocked 0 retries 0 mbuf alloc err 0
TX Queue 0: full 0 current index 0 hiwater 0
TX Queue 1: full 0 current index 0 hiwater 0
TX Queue 2: full 0 current index 0 hiwater 0
TX Queue 3: full 0 current index 0 hiwater 0
TX Queue 4: full 0 current index 0 hiwater 0
TX Queue 5: full 0 current index 0 hiwater 0
TX Queue 6: full 0 current index 0 hiwater 0
TX Queue 7: full 0 current index 0 hiwater 0
pmd b1990b00 device Gi2
RX: pkts 1254741010 bytes 511773562848 return 0 badlen 0
pkts/burst 16 cycl/pkt 792 ext_cycl/pkt 1342
Total ring read 1012256968, empty 937570790
TX: pkts 1385120320 bytes 564465308380
pri-0: pkts 168172786 bytes 68650796972
pkts/send 1
pri-1: pkts 177653235 bytes 72542203822
pkts/send 1
pri-2: pkts 225414300 bytes 91947701824
pkts/send 1
pri-3: pkts 136817435 bytes 55908224442
pkts/send 1
pri-4: pkts 256461818 bytes 104687120554
pkts/send 1
pri-5: pkts 176043289 bytes 71879529606
pkts/send 1
pri-6: pkts 83920827 bytes 34264110122
pkts/send 1
pri-7: pkts 160636635 bytes 64585622696
pkts/send 1
Total: pkts/send 1 cycl/pkt 442
send 1033104466 sendnow 41250092
forced 1776500651 poll 244223290 thd_poll 0
blocked 1060879040 retries 3499069 mbuf alloc err 0
TX Queue 0: full 0 current index 0 hiwater 31
TX Queue 1: full 718680 current index 0 hiwater 255
TX Queue 2: full 0 current index 0 hiwater 31
TX Queue 3: full 0 current index 0 hiwater 31
TX Queue 4: full 15232240 current index 0 hiwater 255
TX Queue 5: full 0 current index 0 hiwater 31
TX Queue 6: full 0 current index 0 hiwater 31
TX Queue 7: full 230668 current index 0 hiwater 224
```

```
pmd b1712d00 device Gi3
RX: pkts 1410702537 bytes 498597093510 return 0 badlen 0
pkts/burst 18 cycl/pkt 269 ext_cycl/pkt 321
Total ring read 1011915032, empty 934750846
TX: pkts 754803798 bytes 266331910366
pri-0: pkts 46992577 bytes 16616415156
pkts/send 1
pri-1: pkts 49194201 bytes 17379760716
pkts/send 1
pri-2: pkts 46991555 bytes 16616509252
pkts/send 1
pri-3: pkts 49195026 bytes 17381741474
pkts/send 1
pri-4: pkts 48875656 bytes 17283423414
pkts/send 1
pri-5: pkts 417370776 bytes 147056906106
pkts/send 6
pri-6: pkts 46992860 bytes 16617923068
pkts/send 1
pri-7: pkts 49191147 bytes 17379231180
pkts/send 1
Total: pkts/send 2 cycl/pkt 0
send 339705775 sendnow 366141927
forced 3138709511 poll 2888466204 thd_poll 0
blocked 1758644571 retries 27927046 mbuf alloc err 0
TX Queue 0: full 0 current index 0 hiwater 0
TX Queue 1: full 0 current index 0 hiwater 0
TX Queue 2: full 0 current index 0 hiwater 0
TX Queue 3: full 0 current index 0 hiwater 0
TX Queue 4: full 0 current index 1 hiwater 0
TX Queue 5: full 27077270 current index 0 hiwater 224
TX Queue 6: full 0 current index 0 hiwater 0
TX Queue 7: full 0 current index 0 hiwater 0
```

Voorbeelduitvoer die de pakketdalingen laat zien als gevolg van pps-throttling uitgevoerd vanaf AWS-zijde, die kan worden waargenomen via pps_allowance_exceeded-teller:

```
C8k-AWS-2#show controllers | in errors|exceeded|Giga

GigabitEthernet1 - Gi1 is mapped to UIO on VXE
  rx_missed_errors 1750262
  rx_errors 0
  tx_errors 0
  rx_mbuf_allocation_errors 0
  rx_q0_errors 0
  rx_q1_errors 0
  rx_q2_errors 0
  rx_q3_errors 0
  bw_in_allowance_exceeded 0
  bw_out_allowance_exceeded 0
  pps_allowance_exceeded 11750
  conntrack_allowance_exceeded 0
  linklocal_allowance_exceeded 0
```

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.