



Communauté Cisco

# Datacenter NX-OS EVPN & ACI

Automatisation grâce aux APIs NX-OS et Terraform

**Francesco Molino**

Network and Security Consultant - CCIE Enterprise Infrastructure | SP | Security #35050 | Firejumper Élite #154

**Xavier Crèvecoeur**

Network and Security Consultant - CCIE Security #11010 | Firejumper Élite #135

5 mai 2020

# Nouveautés et prochains événements



# Événement : Demandez-moi N'importe Quoi



## Data Center – Introduction à VxLAN (BGP-EVPN)

Foire aux Questions jusqu'au  
vendredi 8 mai 2020

avec Michael Di Bartolomeo.  
*Événement privé*

Suivez le lien

<http://bit.ly/AMA-apr20>

**Questions**

Demandez-moi N'importe Quoi à propos de ...  
Data Center Introduction à VxLAN (BGP-EVPN)

Avec Michael Di Bartolomeo  
**Foire aux Questions | Jusqu'au 8 mai**

**Posez vos Questions !**

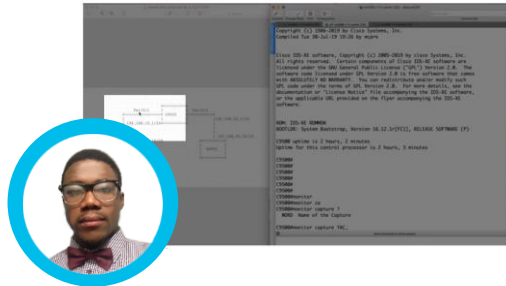
# Vidéo Tips avec nos experts



Proposez-nous des sujets à traiter sur les technologies qui vous intéressent !



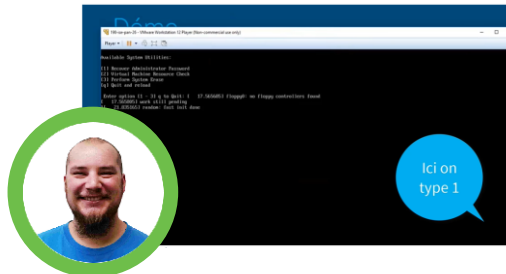
Écrivez-nous : [french-cisco-community@external.cisco.com](mailto:french-cisco-community@external.cisco.com)



Routage et Commutation avec Gary Bolivar  
« *Problème de connectivité avec un switch Cisco famille Catalyst C9000* »

Suivez le lien

<https://bit.ly/VTips-RS1>



Sécurité avec Polo Arroyo Folange  
« *Récupération de mot de passe pour Cisco ISE 2.6* »

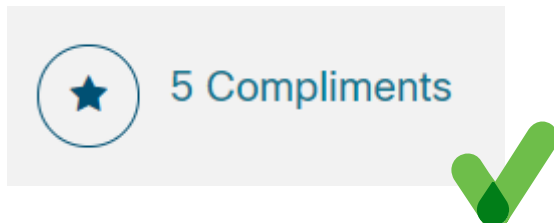
Suivez le lien

<https://bit.ly/VTips-SE2>

# Évaluez le contenu de la Communauté Cisco

## Discussions, Documents, Blogs et Vidéos

Aidez-nous à identifier les contenus de qualité et à reconnaître l'effort des membres de la Communauté Cisco en français.



Identifiez les experts



Repérez les solutions

Apprenez à mieux utiliser la plateforme et exploiter toutes ses ressources.

Suivez le lien

<http://bit.ly/PilotVideoFR>

# Reconnaissance aux Top Contributeurs

## Devenez un Top Contributeur pour le mois de Mai !

La reconnaissance aux **Top Contributeurs** est conçue pour reconnaître et remercier ceux qui ont collaboré avec nous en fournissant des contenus techniques de qualité ainsi que les participants plus actifs qui ont permis à notre communauté de devenir un des Top sites pour les passionnés de la technologie de Cisco.

### Suivez le lien

<http://bit.ly/FRCC-SpotlightAwards>



The Community Spotlight Awards recognizes members whose significant contributions designate leadership and commitment to their peers within their respective communities, including Cisco Community, Cisco Learning Network (CLN), and Cisco Developers Network (CDN). Spotlight awards are designed to recognize and thank individuals who help make our communities the premier online destination for Cisco enthusiasts. [FAQs](#)

**2020** 2019 2018 2017 2016 2015 2014 2013 2012 2011

janvier février **mars** avril mai juin juillet août septembre octobre novembre décembre

French Community Rookie, March 2020



Mohamed Mady SIDIBE  
2020 March

# Les experts de la Communauté Cisco

## Francesco Molino

Sr Network & Security Consultant

CCIE Enterprise Infrastructure,  
SP et Security #35050  
Firejumper Elite #154



Présentateur



# Les experts de la Communauté Cisco

## Xavier Crèvecoeur

Sr Network & Security Consultant

CCIE Security #11010 &  
Firejumper Elite #135



Q&R  
Question Manager



# Participez avec nous et posez des questions

La présentation comprendra aussi quelques questions du public.  
Nous vous invitons cordialement à participer activement aux questions que vous pourrez poser pendant cette séance sur le panneau à droite « Q&R ».

Résolvez vos doutes et partagez votre opinion



# Introduction

# Ordre du jour



Nexus et les APIs



ACI et les APIs



Démonstration (x2)

# Nexus et les APIs

# Python / APIs

- Qu'est ce que Python?
- Qu'est ce qu'API?



# Notre quotidien...

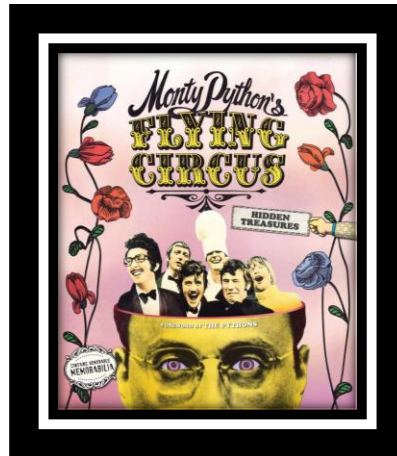
- Différentes tâches à réaliser pour configurer les équipements.
- Coordination pour préparer et envoyer les équipements sur site.
- Petits changements (Moves, Adds & Changes) à réaliser qui prennent du temps.
- Des erreurs humaines arrivent parfois.
- Pas ou peu d'homogénéité des configurations surtout lors des « MAC\* ».
- ...





# Qu'est ce que Python\*?

- Langage Open Source créé en 1991 / Version 3 en Décembre 2008
- Créé par Guido Van Rossum
- Le nom Python vient de Monty Python's Flying Circus (Fan de la série TV)
- Langage de programmation interprété (pas besoin d'être compilé)
- Fonctionne avec n'importe quel système d'exploitation (Linux, macOS, Windows)





# APIs

- API = Application Programming Interface
- C'est une interface qui permet d'exposer des données à d'autres systèmes de manière structurée.
- Cette interface permet d'obtenir des données sans connaître la complexité du système en arrière plan
- 2 « modèles » de API = SOAP et REST
- REST API (Plus rapide et plus léger)

| Difference       | SOAP   | REST   |
|------------------|--|--|
| Style            | Protocol   | Architectural style  |
| Function         | Function-driven: transfer structured information   | Data-driven: access a resource for data                              |
| Data format      | Only uses XML  | Permits many data formats, including plain text, HTML, XML, and JSON |
| Security         | Supports WS-Security and SSL   | Supports SSL and HTTPS   |
| Bandwidth        | Requires more resources and bandwidth  | Requires fewer resources and is lightweight                          |
| Data cache       | Can not be cached  | Can be cached  |
| Payload handling | Has a strict communication contract and needs knowledge of everything before any interaction | Needs no knowledge of the API  |
| ACID compliance  | Has built-in ACID compliance to reduce anomalies   | Lacks ACID compliance  |



Quand a été créée  
la dernière version  
de Python ?

### Polling Question - Sondage 1

- A. 1990
- B. 1991
- C. 2008



# Automatisation

- Nexus et APIs



# Nexus et automatisation

Différentes méthodes:

- SSH
- json-rpc
- NX-API CLI (celui que nous allons voir aujourd'hui)
- NX-API REST

NX-API REST = Programmation en mode Objet orienté (basé sur des réelles fonctions repérées par un dn\*).

\*Dn = *Distinguished Name*

→ [https://fqdn\\_nexus/visore.html](https://fqdn_nexus/visore.html)

→ explications: <https://developer.cisco.com/docs/cisco-nexus-3000-and-9000-series-nx-api-rest-sdk-user-guide-and-api-reference-release-9-3x/#!configuring-an-ethernet-interface/configuring-an-ethernet-interface>

NX-API CLI = Encapsulation de commandes CLI au travers du protocole HTTPS.

Les 2 services sont exposés par un serveur NGINX sur le NX-OS.

SSH vs NX-API ? Données structurées, maintien des « outputs » des commandes.



# Comment activer NX-API ?

- Activer la fonctionnalité avec la commande: `feature nxapi`
- Activer le transport sur HTTPS: `nxapi https port 8443` → Vous pouvez choisir le port sur lequel le serveur NGINX écoute (NGINX = serveur web)
- Attachement d'un certificat pour le protocole https:
  - `nxapi certificate https crt certfile bootflash:/cert.pem`
  - `nxapi certificate https key keyfile bootflash:/key.pem`
  - `nxapi certificate enable`

Accès au NX-API Sandbox: <https://nexus fqdn:8443/>

Authentification basée sur les comptes habituelles (exemple: comptes locaux)

Résultats obtenus sous différent formats :

- JSON (par défaut)
- XML

Documentation: [https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/programmability/guide/b\\_Cisco\\_Nexus\\_9000\\_Series\\_NX-OS\\_Programmability\\_Guide\\_7x/NX\\_API.html](https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_7x/NX_API.html)



# Exemple : NX-API Sandbox

**CISCO NX-API Developer Sandbox** Quick Start Logout

```
sh ip int bri vrf management
```

Message format:  json-rpc  xml  json  nx-api rest  
 rx yang

Command type:  cli\_show  cli\_show\_array  
 cli\_show\_ascii  cli\_conf  bash

**REQUEST:**

```
{
  "ins_api": {
    "version": "1.0",
    "type": "cli_show",
    "chunk": "0",
    "sid": "1",
    "input": "sh ip int bri vrf management",
    "output_format": "json"
  }
}
```

**RESPONSE:**

```
{
  "ins_api": {
    "type": "cli_show",
    "version": "1.0",
    "sid": "eoc",
    "outputs": {
      "output": {
        "input": "sh ip int bri vrf management",
        "msg": "Success",
        "code": "200",
        "body": {
          "TABLE_intf": {
            "ROW_intf": {
              "intf-name": "mgmt0",
              "prefix": "172.16.2.100",
              "ip-disabled": "FALSE",
              "id": 2,
              "proto-state": "up",
              "link-state": "up",
              "admin-state": "up"
            }
          },
          "TABLE_vrf": {
            "ROW_vrf": {
              "vrf-name-out": "management"
            }
          }
        }
      }
    }
  }
}
```



# Exemple : NX-API Sandbox (suite)

```
show ip int bri vrf management
```

POST Reset

**REQUEST:**

```
import requests
import json

"""
Modify these please
"""
url='http://YOURIP/ins'
switchuser='USERID'
switchpassword='PASSWORD'

myheaders={'content-type':'application/json'}
payload={
  "ins_api": {
    "version": "1.0",
    "type": "cli_show",
    "chunk": "0",
    "sid": "1",
    "input": "show ip int bri vrf management",
    "output_format": "json"
  }
}
response = requests.post(url,data=json.dumps(payload), headers=myheaders,auth=(switchuser
```

Copy Python



# Exemple : NX-API CLI avec Python

## Exemples de connexions en utilisant Requests

```
myheaders = {'content-type': 'application/json', 'Accept': "application/json"}
myheaders['Authorization'] = " ".join(("Basic", authencode))
payload = {
    "ins_api": {
        "version": "1.0",
        "type": "cli_show", --> (ou cli_conf pour configurer le Nexus)
        "chunk": "0",
        "sid": "1",
        "input": "show cdp nei", --> (commandes à exécuter. Séparer les multiples commandes par;)
        "output_format": "json" → Format de sortie
    }
}
response = requests.request("POST", url, headers=myheaders, data=json.dumps(payload), verify=False)
return response
```

→ `response.status_code` : Succès ou Échec (200 ou 400)

→ `response.reason` : Raison de l'échec. Détail de l'échec avec `response.text`



# Exemple : NX-API CLI Logs

## Comment voir les logs d'accès

Sur le Nexus, exécuter commande: **show nxapi-server logs**

```
SPINE-1# show nxapi-server logs
2020 April 25 19:35:34.780 ngx_http_auth_pam_authenticate:381 PAM: client ip len:12
2020 April 25 19:35:34.780 ngx_http_auth_pam_authenticate:389 PAM: remote ip:10.100.1.107
2020 April 25 19:35:34.819 ngx_http_cookie_set:642 creating cookie cleanup_cnt:5
2020 April 25 19:35:34.821 ngx_http_ins_api_post_body_handler:550 Input Message
{
  "ins_api": {
    "version": "1.0",
    "type": "cli_show",
    "chunk": "0",
    "sid": "1",
    "input": "show ip int brief vrf management ;show vlan",
    "output_format": "json"
  }
}
2020 April 25 19:35:34.822 _parse_ins_req_msg_json_ez:616 Rollback type set to 0
2020 April 25 19:35:34.822 parse_user_from_request:41 cookie had user 'admin'
2020 April 25 19:35:34.822 parse_user_from_request:55 auth header had user 'admin'
```





# Exemple : NX-OS CLI avec outputs en JSON

Recevoir la sortie de chacune des commandes en JSON

```
SPINE-1# sh vlan
```

| VLAN Name | Status | Ports  |
|-----------|--------|--|
| 1 default | active | Eth1/1, Eth1/2, Eth1/3, Eth1/4<br>Eth1/5, Eth1/6, Eth1/7, Eth1/8<br>Eth1/9, Eth1/10, Eth1/11<br>Eth1/12, Eth1/13, Eth1/14<br>Eth1/15, Eth1/16, Eth1/17<br>Eth1/18, Eth1/19, Eth1/20<br>Eth1/21, Eth1/22, Eth1/23<br>Eth1/24, Eth1/25, Eth1/26<br>Eth1/27, Eth1/28, Eth1/29<br>Eth1/30, Eth1/31, Eth1/32<br>Eth1/33, Eth1/34, Eth1/35<br>Eth1/36, Eth1/37, Eth1/38<br>Eth1/39, Eth1/40, Eth1/41<br>Eth1/42, Eth1/43, Eth1/44<br>Eth1/45, Eth1/46, Eth1/47<br>Eth1/48, Eth1/49, Eth1/50<br>Eth1/51, Eth1/52, Eth1/53<br>Eth1/54, Eth1/55, Eth1/56<br>Eth1/57, Eth1/58, Eth1/59<br>Eth1/60, Eth1/61, Eth1/62<br>Eth1/63, Eth1/64, Eth1/65<br>Eth1/66, Eth1/67, Eth1/68<br>Eth1/69, Eth1/70, Eth1/71<br>Eth1/72, Eth1/73, Eth1/74<br>Eth1/75, Eth1/76, Eth1/77 |

```
SPINE-1# sh vlan | json
```

```
{
  "TABLE_vlanbrief": {
    "ROW_vlanbrief": {
      "vanshowbr-vlanid": "1",
      "vanshowbr-vlanid-utf": "1",
      "vanshowbr-vlanname": "default",
      "vanshowbr-vlanstate": "active",
      "vanshowbr-shutstate": "noshutdown",
      "vanshowlist-idx": ["Ethernet1/1,Eth
        ernet1/2,Ethernet1/3,Ethernet1/4,Ethernet1/5,Ethernet1/6,Ethernet1/7,Ethernet1/
        8,Ethernet1/9,Ethernet1/10,Ethernet1/11,Ethernet1/12,Ethernet1/13,Ethernet1/14,E
        thernet1/15,Ethernet1/16,Ethernet1/17,Ethernet1/18,Ethernet1/19,Ethernet1/20,Eth
        ernet1/21,Ethernet1/22,Ethernet1/23,Ethernet1/24,Ethernet1/25,Ethernet1/26,Ether
        net1/27,Ethernet1/28,Ethernet1/29,Ethernet1/30,Ethernet1/31,Ethernet1/32,Etherne
        t1/33,Ethernet1/34,Ethernet1/35,Ethernet1/36,Ethernet1/37,Ethernet1/38,Ethernet1
        /39,Ethernet1/40,Ethernet1/41,Ethernet1/42,Ethernet1/43,Ethernet1/44,Ethernet1/4
        5,Ethernet1/46,Ethernet1/47,Ethernet1/48,Ethernet1/49,Ethernet1/50,Ethernet1/51,
        Ethernet1/52,Ethernet1/53,Ethernet1/54,Ethernet1/55,Ethernet1/56,Ethernet1/57,Et
        hernet1/58,Ethernet1/59,Ethernet1/60,Ethernet1/61,Ethernet1/62,Ethernet1/63,Ethe
        rnet1/64",
      "Ethernet1/65,Ethernet1/66,Ethernet1/67,Ethernet1/68,Ethernet1/69,Eth
        ernet1/70,Ethernet1/71,Ethernet1/72,Ethernet1/73,Ethernet1/74,Ethernet1/75,Ether
        net1/76,Ethernet1/77,Ethernet1/78,Ethernet1/79,Ethernet1/80,Ethernet1/81,Etherne
        t1/82,Ethernet1/83,Ethernet1/84,Ethernet1/85,Ethernet1/86,Ethernet1/87,Ethernet1
        /88,Ethernet1/89,Ethernet1/90,Ethernet1/91,Ethernet1/92,Ethernet1/93,Ethernet1/9
        4,Ethernet1/95,Ethernet1/96,Ethernet1/97,Ethernet1/98,Ethernet1/99,Ethernet1/100
        ,Ethernet1/101,Ethernet1/102,Ethernet1/103,Ethernet1/104,Ethernet1/105,Ethernet1
        /106,Ethernet1/107,Ethernet1/108,Ethernet1/109,Ethernet1/110,Ethernet1/111,Ether
        net1/112,Ethernet1/113,Ethernet1/114,Ethernet1/115,Ethernet1/116,Ethernet1/117,E
        thernet1/118,Ethernet1/119,Ethernet1/120,Ethernet1/121,Ethernet1/122,Ethernet1/1
        23,Ethernet1/124,Ethernet1/125,Ethernet1/126,Ethernet1/127,Ethernet1/128"}],
      "TABLE_mtuinfo": {
        "ROW_mtuinfo": {
          "vanshowinfo-vlanid": "1",
          "vanshowinfo-media-type": "enet",
          "vanshowinfo-vlanmode": "ce-vlan"}}
    }
  }
}
```

```
SPINE-1#
SPINE-1#
SPINE-1#
```



# NX-API CLI retourne également des infos en bash (Linux)

## Activation du mode bash (équivalence du mode interpréteur Linux)

feature bash-shell

Depuis le Nexus, en mode enable: **run bash**

ip link show

Message format: json-rpc xml json nx-api rest

Command type: cli\_show cli\_show\_array

POST Reset

REQUEST:

```
{
  "ins_api": {
    "version": "1.0",
    "type": "bash",
    "chunk": "0",
    "sid": "1",
    "input": "ip link show",
    "output_format": "json"
  }
}
```

Copy Python

RESPONSE:

```
{
  "ins_api": {
    "sid": "e0c",
    "type": "bash",
    "version": "1.0",
    "outputs": {
      "output": {
        "body": "1: lo: LOOPBACK,UP,LOWER_UP mtu 16436 qdisc noqueue sta",
        "code": "200",
        "msg": "Success"
      }
    }
  }
}
```

```
bash-4.2$ ip link show | more
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT
    link/ether c2:bf:e1:9e:20:27 brd ff:ff:ff:ff:ff:ff
4: eth2: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te UP mode DEFAULT qlen 1000
    link/ether 0c:14:66:01:c3:01 brd ff:ff:ff:ff:ff:ff
5: eth3: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te UP mode DEFAULT qlen 1000
    link/ether 0c:14:66:01:c3:02 brd ff:ff:ff:ff:ff:ff
6: eth4: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te UP mode DEFAULT qlen 1000
    link/ether 0c:14:66:01:c3:03 brd ff:ff:ff:ff:ff:ff
7: eth5: <NO-CARRIER,BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast s
tate DOWN mode DEFAULT qlen 1000
    link/ether 0c:14:66:01:c3:04 brd ff:ff:ff:ff:ff:ff
8: eth6: <NO-CARRIER,BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast s
tate DOWN mode DEFAULT qlen 1000
    link/ether 0c:14:66:01:c3:05 brd ff:ff:ff:ff:ff:ff
9: eth7: <NO-CARRIER,BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast s
tate DOWN mode DEFAULT qlen 1000
```

NX-API CLI est une  
méthode REST-API  
propriétaire au  
Nexus ?

Polling Question - Sondage 2

- A. OUI
- B. NON



# ACI

- APIC
- Terraform

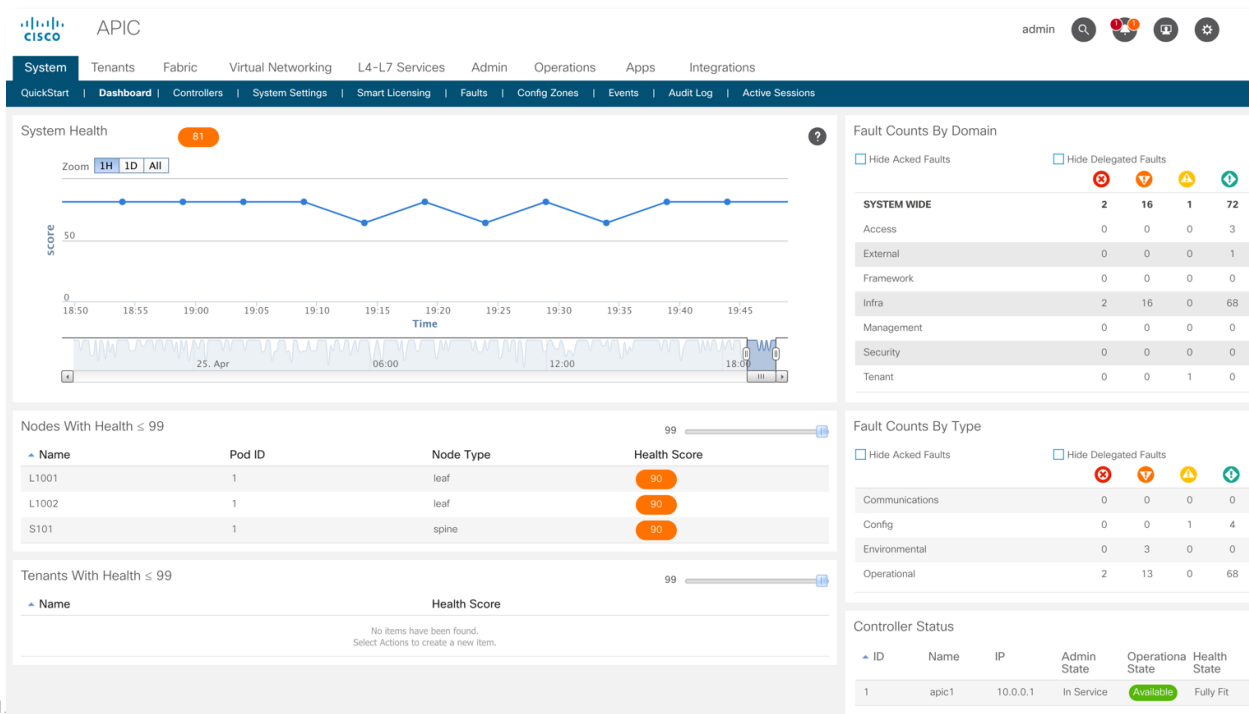


# Cisco ACI vs NX-OS

ACI est géré par l'orchestrateur APIC. Solution SDN de Cisco pour le Data Center

- Application Centric
- Meilleure intégration avec des solutions externes (VMWare, Load-Balancer,...)
- Gestion des images et de l'inventaire
- Simplicité de gestion (Tenants, Application Profile...)
- User-Friendly
- Abstraction des parties Underlay et Overlay
  - ✓ Underlay = Configuration « Fabric VXLAN, BGP EVPN, et routage entre équipements [« Full-Mesh »]
  - ✓ Overlay = Configuration réseau « data » client (VLANS et VRF client)

- APIC est le contrôleur de la « Fabric ACI »



The screenshot displays the Cisco APIC dashboard with the following sections:

- System Health:** A line graph showing a health score over time. The score is currently 91. The x-axis represents time from 18:50 to 19:45. The y-axis represents the score from 0 to 50.
- Fault Counts By Domain:** A table showing fault counts for various domains.
 

| Domain             | Count 1 | Count 2 | Count 3 | Count 4 |
|--------------------|---------|---------|---------|---------|
| <b>SYSTEM WIDE</b> | 2       | 16      | 1       | 72      |
| Access             | 0       | 0       | 0       | 3       |
| External           | 0       | 0       | 0       | 1       |
| Framework          | 0       | 0       | 0       | 0       |
| Infra              | 2       | 16      | 0       | 68      |
| Management         | 0       | 0       | 0       | 0       |
| Security           | 0       | 0       | 0       | 0       |
| Tenant             | 0       | 0       | 1       | 0       |
- Nodes With Health ≤ 99:** A table listing nodes with their health scores.
 

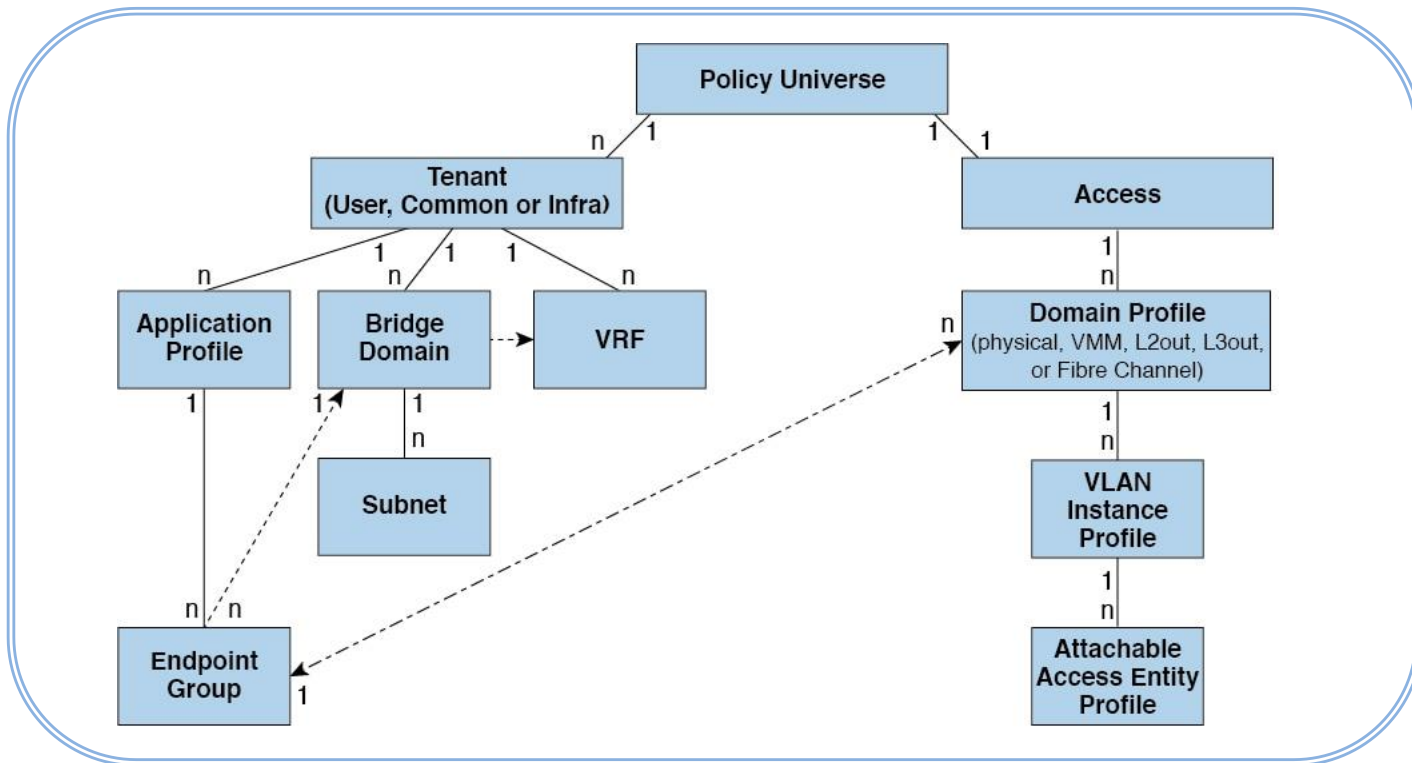
| Name  | Pod ID | Node Type | Health Score |
|-------|--------|-----------|--------------|
| L1001 | 1      | leaf      | 90           |
| L1002 | 1      | leaf      | 90           |
| S101  | 1      | spine     | 90           |
- Tenants With Health ≤ 99:** A section indicating that no items have been found for tenants with a health score of 99 or lower.
- Fault Counts By Type:** A table showing fault counts for different fault types.
 

| Type                  | Count 1 | Count 2 | Count 3 | Count 4 |
|-----------------------|---------|---------|---------|---------|
| <b>Communications</b> | 0       | 0       | 0       | 0       |
| Config                | 0       | 0       | 1       | 4       |
| Environmental         | 0       | 3       | 0       | 0       |
| Operational           | 2       | 13      | 0       | 68      |
- Controller Status:** A table showing the status of the APIC controller.
 

| ID | Name  | IP       | Admin State | Operation State | Health State |
|----|-------|----------|-------------|-----------------|--------------|
| 1  | apic1 | 10.0.0.1 | In Service  | Available       | Fully Fit    |



# Cisco ACI "Object Model"





# Cisco APIC API

Documentation API: <https://apic fqdn/doc/html/>

Outil disponible sur API: API Inspector

The screenshot shows the Cisco APIC web interface. At the top, there's a navigation bar with 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integratio'. Below this, there's a 'Tenants' section with a search bar and a list of tenants: 'ALL TENANTS', 'Add Tenant', 'Tenant Search: name or descr', 'common', 'DATACENTER', 'CORP', 'DMZ', 'OOB'. The 'DATACENTER' tenant is selected, and the 'API Inspector' tool is open, showing a list of API requests and responses. The tool displays details such as the URL, method, and response body for each request. A dropdown menu is visible in the top right corner, with 'Show API Inspector' highlighted.





# Cisco APIC API (Outils/Librairies)

Acitoolkit:

<https://acitoolkit.readthedocs.io/en/latest/introduction.html>

Cobra:

<https://cobra.readthedocs.io/en/latest/>

Et plein d'autres:

<https://developer.cisco.com/docs/aci/#!getting-started/sample-scripts>

## Guides

APIC REST API Guide [↗](#)

Cisco Network Assurance  
Engine REST API User Guide [↗](#)

ACI Policy Model Guide [↗](#)

APIC Management Information  
Model Reference [↗](#)

Cobra (Python) [↗](#)

Cobra SDK Downloads

pyaci (Python) [↗](#)

ACI Toolkit [↗](#)

ACIrb (Ruby)

## Automation

Ansible

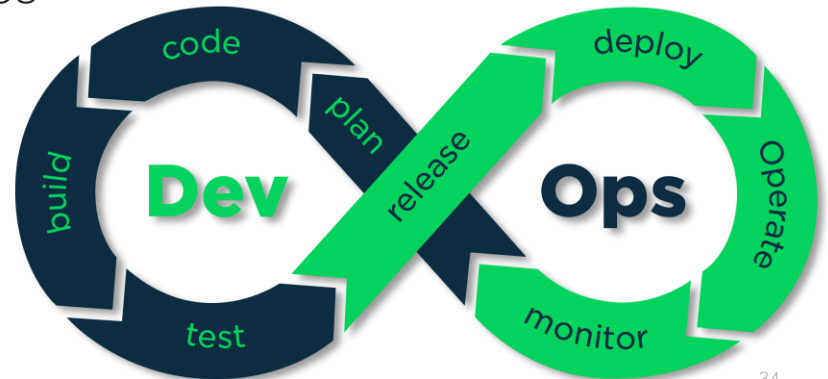
Puppet



# Cisco APIC API (Terraform)

Les besoins d'aujourd'hui:

- Interagir avec notre infrastructure (Sur site et dans le cloud) en mode programmable
  - Cela s'appelle « Infrastructure-as-Code » aussi dit IAC en abréviation
- Pouvoir provisionner, changer et gérer les versions de changements de manière sécuritaire et facile.
- Être indépendant: « vendor agnostic »
- Support Entreprise





# Cisco APIC API (Terraform) suite

- Terraform est donc un outil de « provisioning » appartenant à la société HashiCorp (<https://www.terraform.io>)
- Une centaine de providers disponibles: <https://www.terraform.io/docs/providers/index.html>
- Cisco ACI est un provider compatible Terraform: <https://github.com/terraform-providers/terraform-provider-aci>
- Développé en GO et lancé en Juillet 2014
- Très grande communauté pour aider/supporter



# Cisco APIC API (Terraform) suite

Extensions des fichiers terraform = tf

Pour un projet de base, 3 fichiers nécessaires (2 obligatoires):

- variables.tf ==> Optionnel. Contient des variables utilisées sur l'ensemble des modules (ex: nom utilisateur, mot de passe, url...)
- provider.tf ==> Configuration des providers utilisés dans notre projet
- main.tf ==> Fichier de provisionnement des ressources

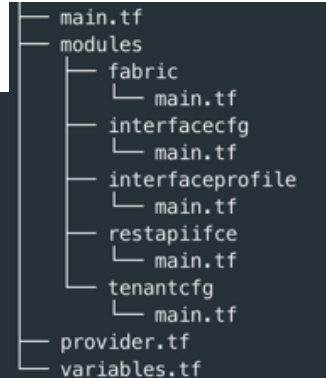
```
bash-3.2$ cat variables.tf
variable "username" {
    default = "admin"
}

variable "password" {
    default = "C1sc0123#"
}

variable "url" {
    default = "https://192.168.10.1"
}
```

```
bash-3.2$ cat provider.tf
provider "aci" {
    username = var.username
    password = var.password
    url      = var.url
    insecure = true
}
bash-3.2$
```

```
bash-3.2$ cat main.tf
module "fabricpool" {
    source = "./modules/fabric"
}
module "interfacecfg" {
    source = "./modules/interfacecfg"
    target_group_depends_on_1 = module.fabricpool.EOF_fabric_1
}
module "interfaceprofile" {
    source = "./modules/interfaceprofile"
    target_group_depends_on_2 = module.interfacecfg.EOF_fabric_2
}
module "restapiifce" {
    source = "./modules/restapiifce"
    target_group_depends_on_3 = module.interfaceprofile.EOF_fabric_3
}
module "tenant" {
    source = "./modules/tenantcfg"
    target_group_depends_on_4 = module.restapiifce.EOF_fabric_4
}
bash-3.2$
```





# Cisco APIC API (Terraform init)

**Terraform init** est la première commande à exécuter pour préparer l'exécution de notre projet.

Différentes options sont disponibles.

Pour notre exemple, cette commande va:

- Parcourir notre répertoire de travail pour trouver les providers utilisés.
- Les télécharger et les installer dans un répertoire **.terraform** dans notre répertoire de projet. (uniquement pour les providers officiels distribués par Terraform)
- Les providers personnalisés devront être installés manuellement dans `~/.terraform.d/plugins`
- Initialiser tous les modules présents dans ce répertoire.



# Cisco APIC API (Terraform init)

```
terraform terraform init
Initializing modules...
- fabricpool in modules/fabric
- interfacecfg in modules/interfacecfg
- interfaceprofile in modules/interfaceprofile
- restapiifce in modules/restapiifce
- tenant in modules/tenantcfg
Initializing the backend...
Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aci" (terraform-providers/aci) 0.2.1...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aci: version = "~> 0.2"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```



# Cisco APIC API (Terraform plan)

- **Terraform plan** va effectuer un « refresh » de ce qui a été déjà exécuté pour connaître ce qui sera créé, modifié ou supprimé.
- Ce plan peut être sauvegardé pour être utilisé lors de l'utilisation de la commande **terraform apply**.

```
# module.tenant.aci_vrf.perimeter_security_voice will be created
+ resource "aci_vrf" "perimeter_security_voice" {
  + annotation           = (known after apply)
  + bd_enforced_enable   = (known after apply)
  + description          = (known after apply)
  + id                   = (known after apply)
  + ip_data_plane_learning = (known after apply)
  + knw_mcast_act        = (known after apply)
  + name                 = "VOICE"
  + name_alias           = (known after apply)
  + pc_enf_dir           = (known after apply)
  + pc_enf_pref          = (known after apply)
  + tenant_dn            = (known after apply)
}
```

**Plan:** 234 to add, 0 to change, 0 to destroy.



# Cisco APIC API (Terraform apply)

**Terraform apply** va appliquer tous les changements.

Différentes options disponibles: (nous allons utiliser 2 options dans notre démo)

- `auto-approve` ==> Une confirmation est demandée lors de l'application de la configuration. Ici, nous acceptons sans demander une intervention utilisateur
- `parallelism=1` ==> Nombre d'opérations concurrentes. Par défaut, cette valeur est à 10. Avec le provider ACI, il faut le limiter à 1 lors d'utilisation du nom utilisateur et mot de passe. Pour l'exécuter avec plusieurs occurrences, une authentification par certificat est nécessaire (non compatible pour le moment avec la fonction `aci_rest`)





# Cisco APIC API (Certificate authentication)

- Générer un self signed certificat (seule méthode supportée):  
openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout francesco.key -out francesco.crt -subj '/CN=francesco'

```
variable "username" {
  default = "francesco"
}

variable "password" {
  default = "Cisc0123#"
}

variable "url" {
  default = "https://192.168.10.1"
}

variable "cert_name" {
  default = "francesco"
}

variable "private_key" {
  default = "/Users/francesco/Downloads/TMP/TEST/francesco.key"
}

provider "aci" {
  username = var.username
  private_key = var.private_key
  cert_name = var.cert_name
  url      = var.url
  insecure = true
}
```

Local User - francesco

Policy | Operational | Faults | History

Properties

| User Certificates: |                         |        | 🗑️ | + |
|--------------------|-------------------------|--------|----|---|
| Name               | Expiration Date         | State  |    |   |
| francesco          | Apr 1 23:04:58 2120 GMT | Active |    |   |

Create X509 Certificate

Name: francesco

Data: [/Wdwyk+pggz9UHPbxnrBn  
h4njsQ2Upp0QRlvoAq5dDeRCLK9Fb9fMQJzBcLY+GDt  
9T+IstUjZGnRT1UgtmRj  
4aD0XgY8Wp7YUpDU5cH+nviawrHBKik48Kfp1/bxox4  
Tnov5PVZeERinCvH0bETv  
EQ==  
-----END CERTIFICATE-----]



# Cisco APIC API (Troubleshooting)

- Activer debug Terraform:

`export TF_LOG=DEBUG` ==> Active le debug dans le terminal

`export TF_LOG_PATH=/Users/francesco/TerraformProject/terraform.log`

==> Sauvegarde le log de debug dans un fichier

- Log dans Cisco APIC au niveau du serveur NGINX:

`/var/log/dme/log/nginx.bin.log`

Quelle commande Terraform est optionnelle pour exécuter le projet ?

### Polling Question - Sondage 3

- A. Terraform init
- B. Terraform plan
- C. Terraform apply



# Ressources

- Ressources Démos

# Quelques ressources...

- Terraform ACI provider:  
<https://github.com/terraform-providers/terraform-provider-aci>
- Terraform:  
<https://www.terraform.io/docs/providers/aci/index.html>

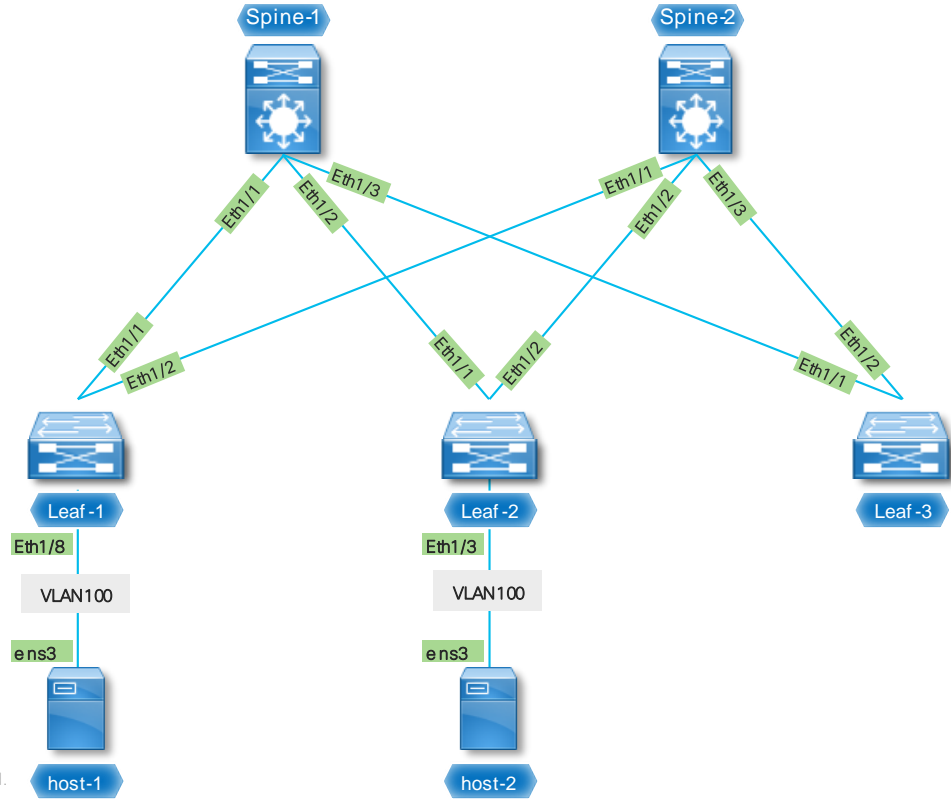


Démonstrations





# Démo 1 (exemple environnement Nexus)





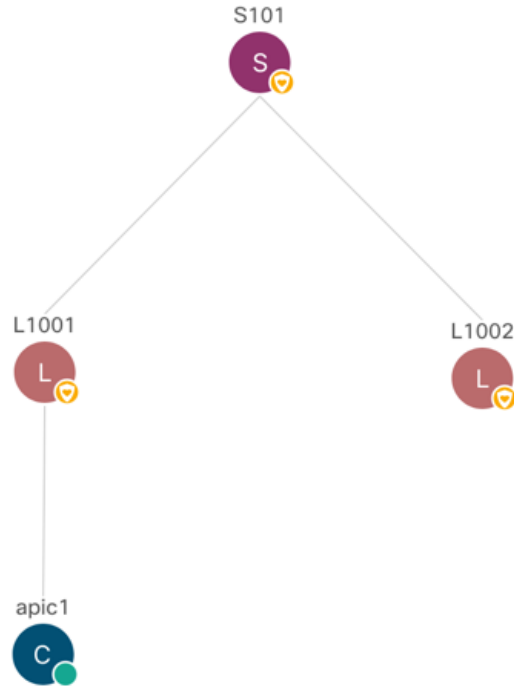
# Démo 1

- « Provisioning » d'une « Fabric EVPN » en automatique
- Utilisation de NX-API CLI
- Configuration de base effectuée (SSH, Certificats et service nxapi activé)





# Démo 2 (exemple environnement ACI)





## Démo 2

- Fabric ACI de LAB avec 1 Spine, 2 Leafs et 1 APIC
- Configuration vierge
- « Provisioning » des configurations de Fabric et Tenants

# Quelques ressources...

- Terraform ACI provider:  
<https://github.com/terraform-providers/terraform-provider-aci>
- Terraform:  
<https://www.terraform.io/docs/providers/aci/index.html>

Téléchargement des sources de la démo

➔ `docker pull supportlan/csc_demo_dc`

Exécution du docker

➔ `docker run -it --privileged supportlan/csc_demo_dc`



```
├── ACI
│   ├── MyTools.py
│   ├── Template.xls
│   ├── __pycache__
│   │   └── MyTools.cpython-37.pyc
│   ├── Logs
│   ├── start.py
│   └── template
│       ├── fabric.template
│       ├── interfacecfg.template
│       ├── interfaceprofile.template
│       ├── interfaceprofile_old.template
│       ├── restapiifce.template
│       ├── tenant.template
│       └── variables.template
├── terraform
│   ├── francesco.crt
│   ├── francesco.key
│   ├── main.tf
│   └── modules
│       ├── fabric
│       │   └── main.tf
│       ├── interfacecfg
│       │   └── main.tf
│       ├── interfaceprofile
│       │   └── main.tf
│       ├── restapiifce
│       │   └── main.tf
│       └── tenantcfg
│           └── main.tf
├── provider.tf
├── variables.tf
├── vault.kdbx
└── EVPN_PYTHON
    ├── INFOS.xls
    ├── MyTools.py
    ├── __pycache__
    │   └── MyTools.cpython-37.pyc
    ├── authentication.kdbx
    ├── evpn_leaf.template
    ├── evpn_spine.template
    ├── Logs
    └── start.py
```

Dissipez vos  
doutes



Utilisez le panneau « Q&R » pour  
poser vos questions

Avec le code source  
du laboratoire inclus!

Téléchargez la présentation sur

<http://bit.ly/WEBsld-may20>



# Cisco Community – Demandez-moi ...

Avez-vous encore des questions sur l'Automatisation Data Center ?

Foire aux Questions  
jusqu'au 15 mai

avec Francesco Molino  
*Événement public*

Suivez le lien

<http://bit.ly/AMA-may20>



**Questions**

Demandez-moi N'importe Quoi à propos de ...  
**Automatisation Data Center ACI et NXOS EVPN**

Avec Francesco Molino  
**Foire aux Questions | Jusqu'au 15 mai**

**Posez vos Questions !**

# La communauté est disponible dans d'autres langues

Si vous parlez anglais, espagnol, portugais, russe, chinois ou japonais, vous pouvez participer aussi dans les autres communautés Cisco.

[Cisco Community](#)

Anglais

[Сообщество Cisco](#)

Russe

[Comunidad de Cisco](#)

Espagnol

[Comunidade da Cisco](#)

Portugais

[思科服务支持社区](#)

Chinois

[シスココミュニティ](#)

Japonais

# Nous vous invitons à nous suivre dans les réseaux sociaux et à partager nos prochains événements

## Cisco Community

- Facebook/CiscoSupportCommunity
- Twitter @cisco\_support
- YouTube ciscosupportchannel
- LinkedIn Cisco Community  
<https://www.linkedin.com/showcase/3544800/>
- Instagram ciscosupportcommunity  
<https://www.instagram.com/ciscosupportcommunity/>





Votre avis nous  
intéresse !



Veillez remplir le sondage qui  
apparaîtra sur votre écran à la fin  
de cette présentation.



Merci pour votre participation !



