



# 24.Nexus Python实战

## 教主技术进化论

主讲人：现任明教教主  
北京乾颐堂网络实验室出品

Cisco live!





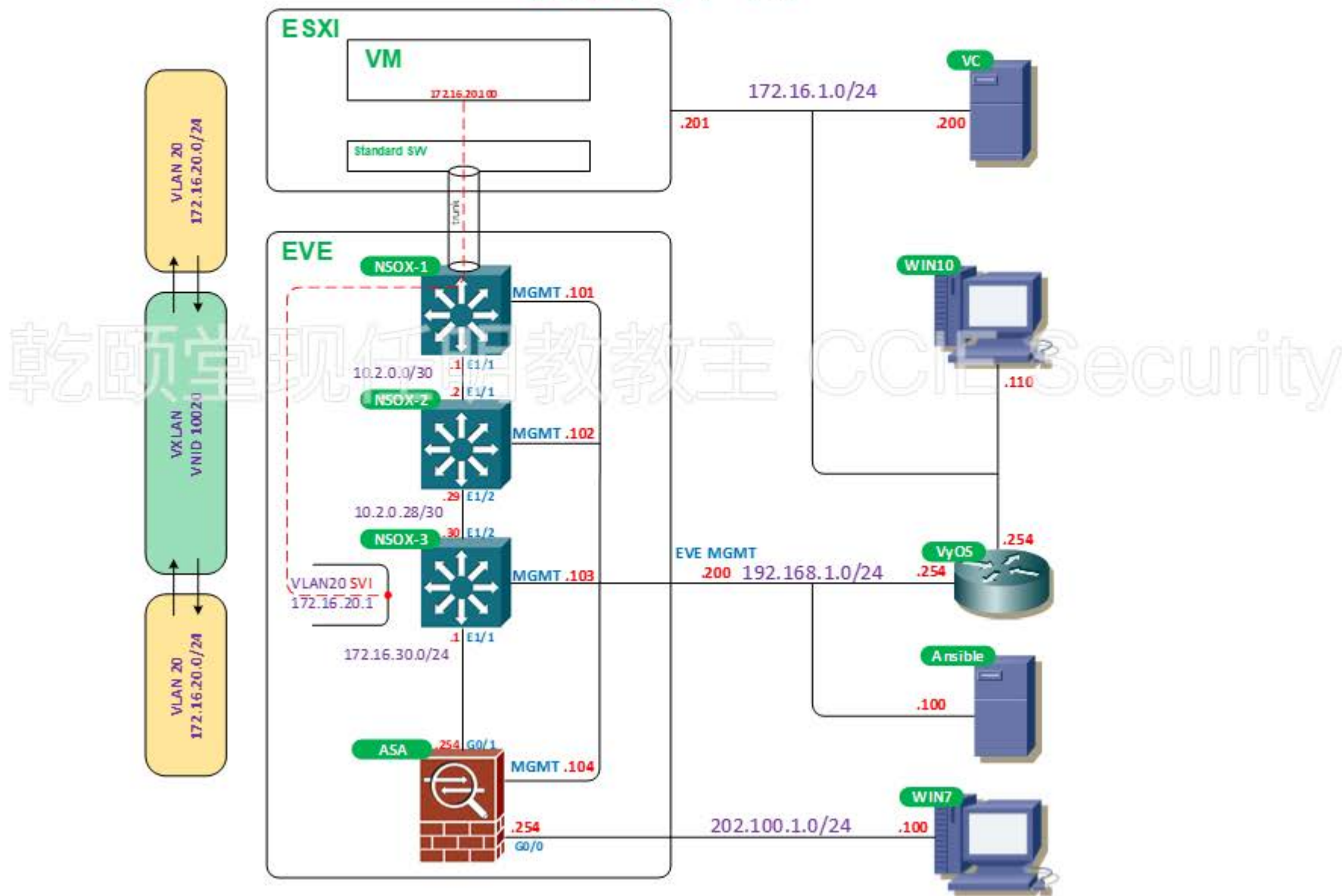
# 内容简介

1. Nexus内建Python
2. Ansible配置Nexus
3. Nexus API
4. 简易云实战

乾颐堂现任明教教主 CCIE Security



# 拓扑介绍



乾颐堂现任首席教主 CCIE Security



# 第一部分

乾颐堂现任明教教主 CCIE Security

# Nexus内置Python





## 关于内置Python

The Cisco Nexus 9000 Series devices support **Python v2.7.5** in both interactive and non-interactive (script) modes and is available in the Guest Shell.

The Python scripting capability gives programmatic access to the device's command-line interface (CLI) to perform various tasks and PowerOn Auto Provisioning (POAP) or Embedded Event Manager (EEM) actions. Python can also be accessed from the Bash shell.

The Python interpreter is available in the Cisco NX-OS software.

For information about using Python with Cisco Nexus devices, see the Cisco Nexus 9000 Series Python SDK User Guide and API Reference at this URL:  
<https://developer.cisco.com/site/nx-os/docs/apis/python/>.

# Cisco Python Package

Cisco NX-OS provides a Cisco Python package that enables access to many core network device modules, such as interfaces, VLANs, VRFs, ACLs and routes.

The screenshot shows the Cisco documentation page for the Python package on Nexus 9000 Series switches. The page is titled "Documentation > Open NX-OS" and is under the "Python" section. The left sidebar contains a navigation menu with the following items: "Cisco Nexus 9000 Series Python SDK User Guide and API Reference", "About the Python Programming Language" (selected), "Features of Nexus 9000 Python Scripting", "Getting Started with the Cisco Nexus 9000 Series Python SDK Using Python", and "Executing Scripts". Below these are several module categories: ACL Module, BGP Module, CLI Module, FEATURE Module, INTERFACE Module, KEY Module, MAC\_ADDRESS\_TABLE Module, OSPFSession Module, ROUTEMAP Module, ROUTES Module, SSH Module, SYSTEM Module, TACACS Module, VLAN Module, and VRF Module. The main content area is titled "About the Python Programming Language" and includes a note: "Note: This guide is intended for users who are familiar with the Python programming language and who have experience configuring Cisco Nexus switches. For detailed configuration information, refer to other Cisco documentation, such as Cisco Nexus 9000 Series NX-OS configuration guides and CLI command references." Below the note, the text states: "The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python website at [www.python.org](http://www.python.org). The Python website also distributes and links to many free third-party Python modules, programs and tools, and additional documentation." It also mentions: "The Cisco Nexus 9000 Series switches support all of the features available in Python v2.7.5." The next section is titled "Features of Nexus 9000 Python Scripting" and lists the following tasks: "Run a script to verify configuration on switch bootup.", "Back up a configuration.", "Perform proactive congestion management by monitoring and responding to buffer utilization characteristics.", "Integrate with the Power-On Auto Provisioning for EEM modules.", "Perform a job at a specific time interval (such as Port Auto Description).", "Programmatically access the switch command-line interface (CLI) to perform various tasks.", and "Python can also be accessed from the Bash shell."



## 进入Python交互界面的方式

```
NXOS1# python
Python 2.7.5 (default, Jun  7 2016, 10:49:13)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

进入底层

```
NXOS1(config)# feature bash-shell
bash-4.2$
```

如果全局路由表可以访问互联网,可以使用pip install xxx

```
bash-4.2$ pip install xxx
```

# cli模块

PYTHON

```
>>> from cli import *
```

PYTHON

Example:

```
>>> cli ('configure terminal', interface loopback 5, no shut)
```

## Arguments:

- cmd: Single CLI command or a batch of CLI commands. Delimiter for multiple CLI commands is **a space followed by a semicolon followed by a space**(空格 分号 空格). Configuration commands must be in a fully qualified form.

## Returns:

- string: CLI output string for show commands or an empty string for configuration commands.

## Raises:

- cli\_syntax\_error: CLI command is not a valid NX-OS command.
- cmd\_exec\_error: Execution of CLI command is not successful.



# clid模块

## PYTHON

Example:

```
>>> import json
>>> from cli import *
>>> jversion = json.loads(clid("show version"))
>>> jversion['bios_ver_str']
'08.06'
```

乾颐堂现任明教教主 CCIE Security

### Arguments:

- cmd: Single CLI command or a batch of CLI commands. Delimiter for multiple CLI commands is a space followed by a semicolon. Configuration commands must be in a fully qualified form.

### Returns:

- string: JSON-formatted output of show commands.

### Raises:

- cli\_syntax\_error: CLI command is not a valid NX-OS command.
- cmd\_exec\_error: Execution of CLI command is not successful.

# 激活sftp-server

NXOS1(config)# feature sftp-server

CAUsers\Administrator\Desktop

名字	大小	类型	已修改
..		上级目录	2018/6/2 21:24:51
ansible_bak		文件夹	2018/6/2 21:55:47
Python脚本		文件夹	2018/4/23 9:19:30
安装包		文件夹	2018/4/21 18:50:51
抓包		文件夹	2017/8/15 19:47:04
config_if_desc.py	2 KB	PY 文件	2018/6/2 17:48:46
Microsoft EdgeJnk	2 KB	快捷方式	2018/5/16 20:15:45
PutTYJnk	1 KB	快捷方式	2018/4/20 16:51:21
SecureCRTJnk	2 KB	快捷方式	2018/4/20 15:28:50
VMware-VMRC-10....	30,172	压缩(zipped)文件	2018/4/18 15:27:24
复制网站.txt	1 KB	文本文档	2018/4/20 23:26:58

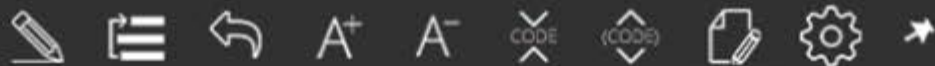
名字	大小	已修改	权限	所有者
..		2018/6/4 14:19:52	rxwxrwx	0
home		2018/6/4 14:05:58	rxwxrwx	2002
scripts		2018/6/4 13:18:51	rxwxrwx	0
virt_strg_pool_bf_vd		2018/6/4 13:18:55	rxw-----	0
virtual-instance		2018/6/4 13:17:24	rxwxrwx	0
20180604_051903_...	224 KB	2018/6/4 13:32:11	rw-rw-rw-	0
nxos.7.0.3J5.1.bin	739,698	2016/10/29 22:04:30	rw-rw-rw-	0
virtual-instance.conf	1 KB	2018/6/4 13:17:15	rw-rw-rw-	0

使用WINSCP可以直接读取bootflash

0 B / 30,177 KB, 0 / 10      1已隐藏 0 B / 723 MB, 0 / 7      2已隐藏 SFTP-3 0:00:54

# 创建Python文件

```
10 |         intf=intflist['TABLE_interface']['ROW_interface'][i]
11 |         i=i+1
12 |         if intf['state'] == 'up':
13 |             if re.match('.*Ether.*', intf['interface']):
14 |                 upintflist.append(intf['interface'])
15 |     return upintflist
16 |
17 | def getneiname(ifname):
18 |     try:
19 |         neidetail = json.loads(cli('show cdp neighbors interface %s' % ifname))
20 |         #print(neidetail)
21 |         return neidetail['TABLE_cdp_neighbor_brief_info']['ROW_cdp_neighbor_brief_info']['device_id']
22 |     except:
23 |         return None
24 |
25 | def config_description(ifname):
26 |     enter_inf = 'interface %s' % ifname
27 |     description = 'description link to %s' % getneiname(ifname)
28 |     cli_cmd = 'configure terminal ; ' + enter_inf + ' ; ' + description
29 |     #print(cli_cmd)
30 |     cli(cli_cmd)
31 |
32 | if __name__ == '__main__':
33 |     #print(getupintflist())
34 |     #print(getneiname('ethernet1/3'))
35 |     #config_description('ethernet1/1')
36 |
37 |     up_interfaces = getupintflist()
38 |     for x in up_interfaces:
39 |         if getneiname(x):
```





## 运行config\_if\_desc.py

```
NXOS1# python bootflash:///config_if_desc.py
```

```
NXOS1# show run inter e1/1
```

```
!Command: show running-config interface Ethernet1/1
```

```
!Time: Mon Jun  4 07:06:29 2018
```

```
version 7.0(3)I5(1)
```

```
interface Ethernet1/1
```

```
  description link to NXOS2(9DU4P5RCQ2O)
```

```
  no switchport
```

```
  ip address 10.2.0.1/30
```

```
  ip router ospf 1 area 0.0.0.0
```

```
  ip pim sparse-mode
```

```
  no shutdown
```

乾颐堂现任明教教主 CCIE Security



# 任务计划调度Python脚本

```
feature scheduler
scheduler job name run_python
    python bootflash:///config_if_desc.py
```

```
scheduler schedule name schedule_run_python
    job name run_python
    time start now repeat 0:0:1
end
```

顾乾堂 明教教主 CCIE Security



# 第二部分

# Nexus API

乾頤堂現任明教教主 ©CIE Security





## 当前的挑战

Most networks in use today were built on hardware with tightly coupled software intended to be managed and administered through the command-line interface (CLI).

These systems worked well in a world of static network configurations, static workloads, and predictable slower change rates for application scaling. As data center networks have been virtualized and begun moving to cloud and agile IT models, this model is no longer the optimal method.

Therefore, vendors are working to layer programmability onto existing offerings and device operating systems - for example, improving the accessibility of the CLIs by making them available outside of the switch by using HTTP/HTTPS and using JSON/XML payloads.

Although this approach increases capabilities, it is not an ideal method for incorporating programmability. The CLI based programming, whether locally on switch or remotely using HTTP/HTTPS, is still synchronous, proprietary, order dependent and sequential. Hence automating using CLI makes the task harder



## 协议与数据编码语言

协议:

- CLI
- NETCONF(The NETCONF protocol can be layered on any transport protocol)
- RESTCONF(HTTP传输的NETCONF)
- standard RESTful access

乾颐堂现任明教教主 CCIE Security

编码语言:

- JSON
- XML



# 协议比较

	SNMP	NETCONF	SOAP	REST
Standard	IETF	IETF	W3C	-
Resources	OIDs	Paths		URLs
Data models	Defined in MIBs	YANG Core Models		
Data Modeling Language	SMI	YANG	(WSDL, not data)	Undefined, (WSDL), WADL, text...
Management Operations	SNMP	NETCONF	In the XML Schema, not standardized	HTTP operations
Encoding	BER	XML	XML	XML, JSON,...
Transport Stack	UDP	SSH TCP	SSL HTTP TCP	SSL HTTP TCP

“RESTConf”

© All rights reserved

MAY 27, 2013 22



# YANG Data Modeling

YANG is a data modeling language that supports data models for the NETCONF protocol. (The NETCONF protocol supports network configuration management.)

YANG provides a way to define the objects and data in NETCONF requests and replies. YANG models the network configuration, operational, and RPC data; and it provides semantics to better define NETCONF data in terms of constraints, reusable structures, and built-in/derived types.

For more details about YANG data modeling, visit YANG Central <http://www.yang-central.org>.

# MO DN RN DME

## MO:

Object instances are referred to as managed objects (MOs)

## DN:

Every managed object in the system can be identified by a unique distinguished name (DN). This approach allows the object to be referred to globally.

## DME:

internal data management engine (DME). The DME validates and rejects incorrect attributes.

## RN:

In addition to its distinguished name, each object can be referred to by its relative name (RN). The relative name identifies an object relative to its parent object. Any given object's distinguished name is derived from its own relative name appended to its parent object's distinguished name.

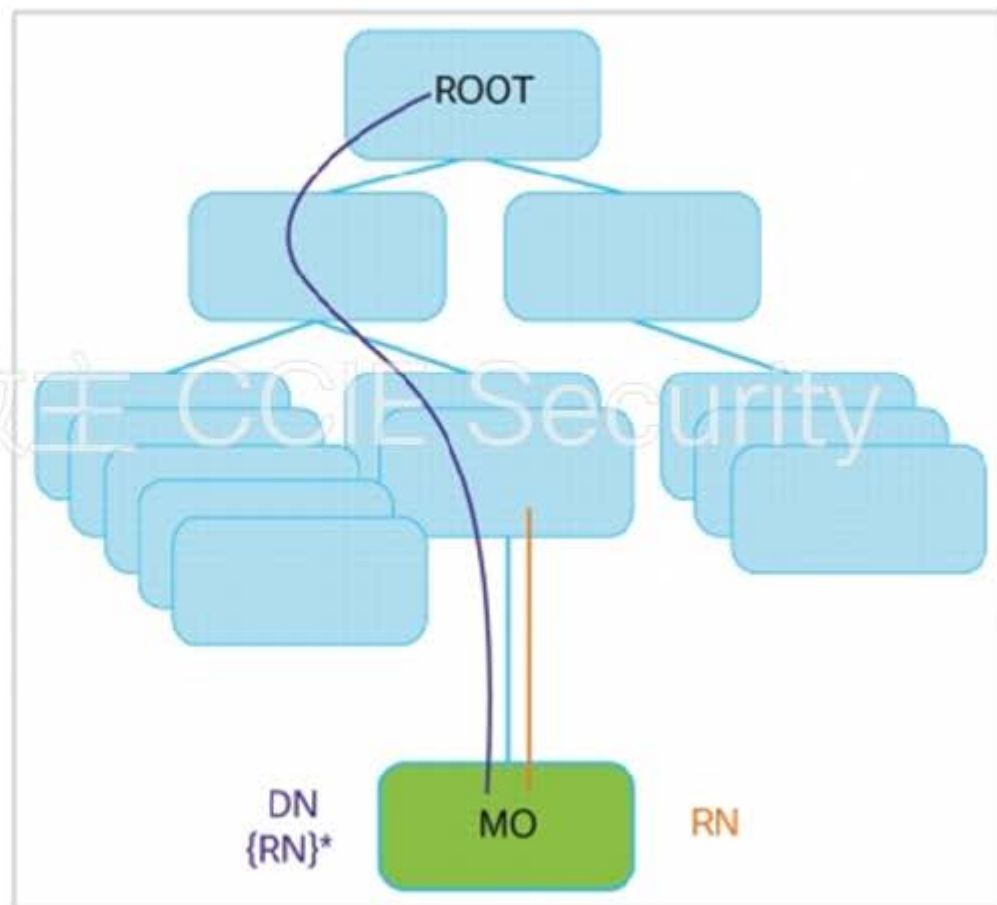


Figure 2: Managed Objects, Relative Names and Distinguished Names



# URL formate

The URL format used can be represented as follows:

`<system>/api/[mo|class]/[dn|class][:method].[xml|json]?{options}`

The various building blocks of the preceding URL are as follows:

- **System**: System identifier; an IP address or DNS-resolvable host name
- **mo | class**: Indication of whether this is a managed object or tree (MIT) or class-level query
- **class**: Managed-object class (as specified in the information model) of the objects queried; the class name is represented as
- **dn**: Distinguished name (unique hierarchical name of the object in the MIT tree) of the object queried
- **method**: Optional indication of the method being invoked on the object; applies only to POST requests
- **xml | json**: Encoding format
- **options**: Query options, filters, and arguments

# 激活API

NXOS1 (config)# feature nxapi

Cisco NX-API Sandbox

不安全 https://192.168.1.101

EVE | Topology vSphere Web Client

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

Enter CLI commands here, one command per line.

Message format: json-rpc xml json

nx-api rest nxapi

Command type: json xml

Convert Reset

REQUEST Copy

ERROR Copy

Copyright © 2014-2016 Cisco Systems, Inc. All rights reserved. NX-API version 1.1

# sandbox测试 json\_rpc 1

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

`vlan 50`

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

Command type:  cli  cli\_ascii

**POST** **Reset**

**REQUEST**

```
[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "vlan 50",
      "version": 1
    },
    "id": 1
  }
]
```

**RESPONSE**

乾颐堂现任明教教主 CCIE Security

## sandbox测试 json\_rpc 2

文件:  
N9K\_Core\_Info.py

```
1 import json
2 from urllib3 import *
3 from base64 import b64encode
4
5 disable_warnings()
6 http = PoolManager()
7
8 username = "admin"
9 password = "Cisc0123"
10 nxos1_ip = "192.168.1.101"
11 nxos3_ip = "192.168.1.103"
12 nxos1_url = "https://" + nxos1_ip + "/ins"
13 nxos3_url = "https://" + nxos3_ip + "/ins"
14
15
16 my_headers = {'content-type': 'application/json-rpc'}
17
18 user_pass_str = username + ':' + password
19 user_pass_str_encode = user_pass_str.encode()
20 userAndPass = b64encode(user_pass_str_encode).decode("ascii")
21
22 my_headers["Authorization"] = 'Basic %s' % userAndPass
23
```

## sandbox测试 json\_rpc 3

文件:  
test\_json\_rpc.py

```
1  from N9K_Core_Info import *
2
3
4  def test_json_rpc():
5      payload = [
6          {
7              "jsonrpc": "2.0",
8              "method": "cli",
9              "params": {
10                 "cmd": "vlan 60",
11                 "version": 1
12             },
13             "id": 1
14         }
15     ]
16     r = http.request('POST', nxos1_url, headers=my_headers, body=json.dumps(payload))
17
18
19 if __name__ == "__main__":
20     test_json_rpc()
21
22
```



# sandbox测试 json\_dme 1

Creating a VLAN

DME YANG

这里有url

```
POST http://<IP_Address>/api/mo/sys/bd.json
{
  "bdEntity": {
    "children": [
      {
        "12BD": {
          "attributes": {
            "fabEncap": "vlan-50",
            "pcTag": "1"
          }
        }
      }
    ]
  }
}
```

这个逗逼的数据格式是不闭合的

Response

```
{
  imdata : []
}
```

vlan 50

Message format:   
 json-rpc xml json   
 nx-api rest nx-yang   
 Command type:   
 cli

POST Reset Convert

REQUEST

```
{
  "topSystem": {
    "children": [
      {
        "bdEntity": {
          "children": [
            {
              "12BD": {
                "attributes": {
                  "fabEncap": "vlan-50",
                  "pcTag": "1"
                }
              }
            }
          ]
        }
      }
    ]
  }
}
```

ERROR

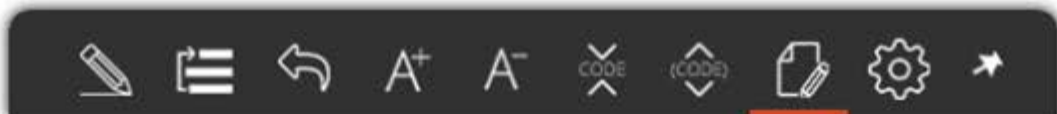
Copy Python Copy

<https://developer.cisco.com/docs/nx-os-n3k-n9k-api-ref/#!/configuring-vlans/creating-a-vlan>

## sandbox测试 json\_dme 2

文件:  
test\_token.py

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import requests
5  import json
6
7  def get_token():
8      base_url = 'http://192.168.1.101/api/'
9
10     # create credentials structure
11     name_pwd = {'aaaUser': {'attributes': {'name': 'admin', 'pwd': 'Cisc0123'}}}
12     json_credentials = json.dumps(name_pwd)
13
14     # log in to API
15     login_url = base_url + 'aaaLogin.json'
16     post_response = requests.post(login_url, data=json_credentials)
17
18     # get token from login response structure
19     auth = json.loads(post_response.text)
20     login_attributes = auth['imdata'][0]['aaaLogin']['attributes']
21     auth_token = login_attributes['token']
22
23     return auth_token
24
25  if __name__ == "__main__":
26      print(get_token())
27
28
```



## sandbox测试 json\_dme 3

```
1 from N9K_Core_Info import *
2 from test_token import get_token
3
4 nxos1_url = "http://192.168.1.101/api/mo/sys/bd.json"
5
6 yang_headers = {'content-type': 'application/json', 'Cookie': "APIC-Cookie=" + get_token()}
7
8
9 def test_yang():
10     payload = {"bdEntity": {"children": [{"l2BD": {"attributes": {"fabEncap": "vlan-98", "pcTag": "1"}}}]}
11
12     r = http.request('POST', nxos1_url, headers=yang_headers, body=json.dumps(payload))
13     print(r.data)
14
15
16 if __name__ == "__main__":
17     test_yang()
18
19
```

文件:  
test\_dme.py

# 实战中采用的是JSON\_RPC

The screenshot shows a PyCharm IDE with the following code in the editor:

```
from N9K_Core_Info import *

def nxos3_edit_svi(VLANid):

    SVIipadd = '172.16.' + str(VLANid) + '.1'

    payload = [
        {"jsonrpc": "2.0", "method": "cli", "params": {"cmd": "configure terminal", "version": 1}, "id": 1},
        {"jsonrpc": "2.0", "method": "cli", "params": {"cmd": "interface vlan " + str(VLANid), "version": 1}, "id": 2},
        {"jsonrpc": "2.0", "method": "cli", "params": {"cmd": "ip address "+str(SVIipadd)+"/24", "version": 1}, "id": 3},
        {"jsonrpc": "2.0", "method": "cli", "params": {"cmd": "no shut", "version": 1}, "id": 4}
    ]

    http_request('POST', nxos_url, headers=my_headers, body=json.dumps(payload))

    print(SVIipadd)

if __name__ == "__main__":
    nxos3_edit_svi()
```

The Run window shows the following output:

```
C:\Users\Administrator\PycharmProjects\Nexus_API\venv\Scripts\python.exe C:/Users/Administrator/PycharmProjects/Nexus_API/qytang_6.3/qytang/qytang/mo
b'{"imdata":[]}'

Process finished with exit code 0
```

Parameter "VLANid" unfilled



# 第三部分

乾颐堂现任明教教主 ©CIE Security

# Ansible配置Nexus



教主技术进化论

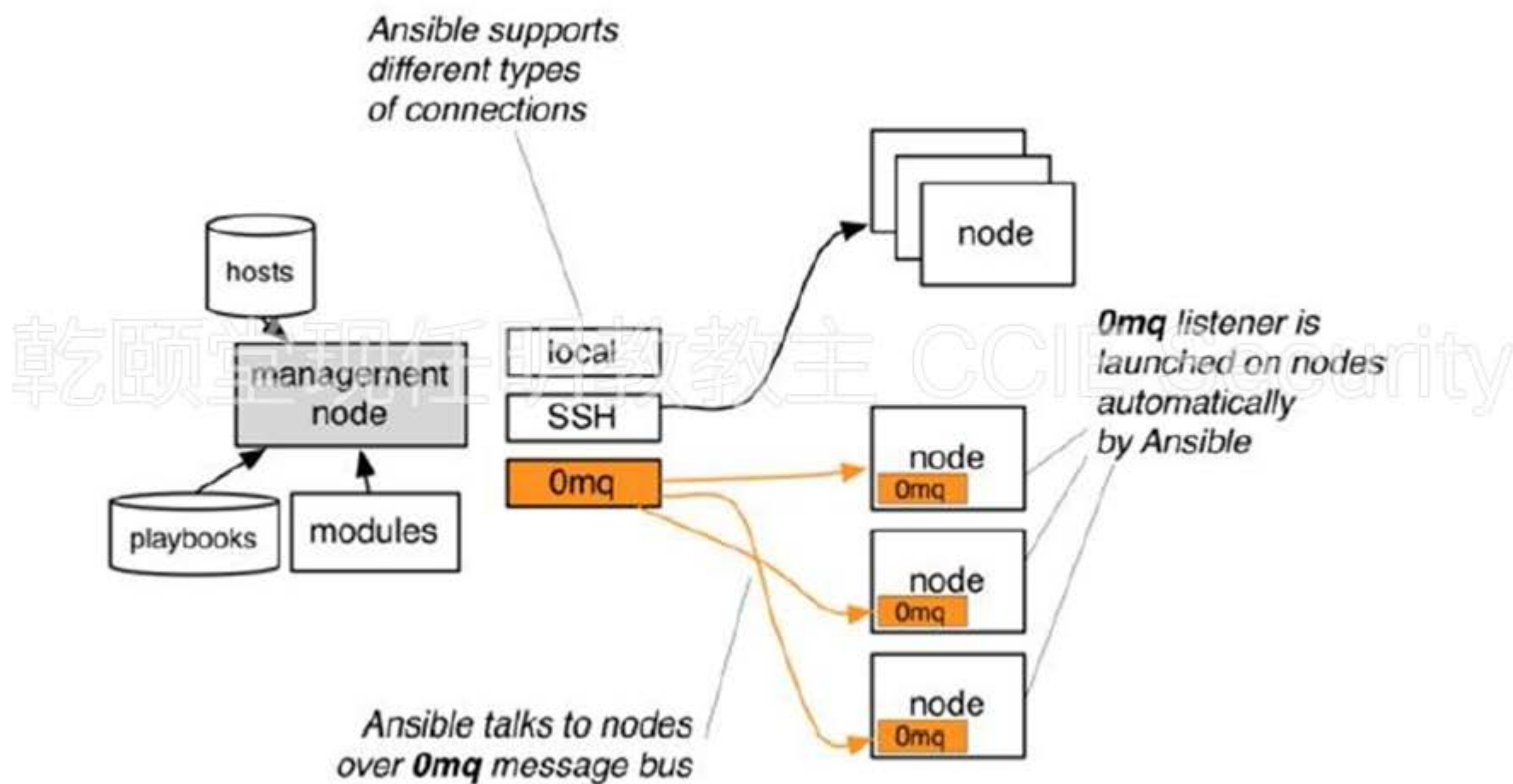
## Ansible简介

Ansible是新出现的自动化运维工具，基于Python开发，集合了众多运维工具（Puppet、CFengine、Chef、SaltStack）的优点，实现了批量系统配置、批量程序部署、批量运行命令等功能。

乾颐堂现任明教教主 CCIE Security

Ansible是基于模块工作的，本身没有批量部署的能力。真正具有批量部署的是ansible所运行的**模块**，ansible只是提供一种框架。

# Ansible工作原理



# Ansible丰富的网络相关模块

[http://docs.ansible.com/ansible/latest/list\\_of\\_network\\_modules.html](http://docs.ansible.com/ansible/latest/list_of_network_modules.html)

## 本次实验主要讲解NXOS模块

### Nxos

- `nxos_aaa_server` - Manages AAA server global configuration.
- `nxos_aaa_server_host` - Manages AAA server host-specific configuration.
- `nxos_acl` - Manages access list entries for ACLs.
- `nxos_acl_interface` - Manages applying ACLs to interfaces.
- `nxos_banner` - Manage multiline banners on Cisco NXOS devices
- `nxos_bgp` - Manages BGP configuration.
- `nxos_bgp_af` - Manages BGP Address-family configuration.
- `nxos_bgp_neighbor` - Manages BGP neighbors configurations.
- `nxos_bgp_neighbor_af` - Manages BGP address-family's neighbors configuration.
- `nxos_command` - Run arbitrary command on Cisco NXOS devices
- `nxos_config` - Manage Cisco NXOS configuration sections
- `nxos_evpn_global` - Handles the EVPN control plane for VXLAN.
- `nxos_evpn_vni` - Manages Cisco EVPN VXLAN Network Identifier (VNI).
- `nxos_facts` - Gets facts about NX-OS switches
- `nxos_feature` - Manage features in NX-OS switches.
- `nxos_file_copy` - Copy a file to a remote NXOS device over SCP.
- `nxos_gir` - Trigger a graceful removal or insertion (GIR) of the switch.
- `nxos_gir_profile_management` - Create a maintenance-mode or normal-mode profile for GIR.
- `nxos_hsrp` - Manages HSRP configuration on NX-OS switches.
- `nxos_igmp` - Manages IGMP global configuration.

乾颐堂 网络工程师 CCIE Security





## 安装Ansible

```
[root@localhost ~]# yum install -y ansible
```

```
[root@localhost ~]# ansible --version
```

```
ansible 2.4.2.0
```

```
config file = /etc/ansible/ansible.cfg
```

```
configured module search path = [u'/root/.ansible/plugins/modules',  
u'/usr/share/ansible/plugins/modules']
```

```
ansible python module location = /usr/lib/python2.7/site-packages/ansible
```

```
executable location = /usr/bin/ansible
```

```
python version = 2.7.5 (default, Nov 6 2016, 00:28:07) [GCC 4.8.5 20150623 (Red Hat 4.8.5-11)]
```



## Ansible重要文件

```
[root@localhost ansible]# ls  
ansible.cfg hosts roles
```

默认安装后，在/etc/ansible目录，会出现两个重要的文件

- 1.ansible.cfg (系统配置)
- 2.hosts (管理主机信息)

乾颐堂 教主 CCIE Security

## 配置Hosts

文件:/etc/ansible/hosts

```
[N9Ks]
192.168.1.101
192.168.1.102
192.168.1.103

[N9Ks:vars]
ansible_ssh_pass='Cisc0123'
ansible_ssh_user='admin'
ansible_connection=local
```

乾颐堂 网络工程师 教主 CCIE Security



## 修改ansible.cfg

文件:/etc/ansible/ansible.cfg

不检查host\_key

```
~~~忽略其它~~~  
# uncomment this to disable SSH key host checking  
host_key_checking = False  
~~~忽略其它~~~
```

乾颐堂现任明教教主 CCIE Security



## Ping模块 测试

使用ping模块测试连通性（其实是使用ping和ssh）

```
[root@localhost ansible]# ansible N9Ks -m ping -o
192.168.1.101 | SUCCESS => {"changed": false, "ping": "pong"}
192.168.1.102 | SUCCESS => {"changed": false, "ping": "pong"}
192.168.1.103 | SUCCESS => {"changed": false, "ping": "pong"}
```

乾颐堂现任明教教主 CCIE Security



## setup模块 测试

使用setup模块提取设备基本信息

```
[root@localhost ansible]# ansible N9Ks -m setup
```

乾颐堂现任明教教主 CCIE Security

## Ad-Hoc (nxos\_config)

```
- name: configure top level configuration and save it
nxos_config:
  lines: hostname {{ inventory_hostname }}
```

改编为Ad-Hoc格式:

```
ansible 192.168.1.101 -m nxos_config -a 'lines="hostname NXOS1"'
```

```
- name: for idempotency, use full-form commands
nxos_config:
  lines:
    # - shut
    - shutdown
  # parents: int eth1/1
  parents: interface Ethernet1/1
```

改编为Ad-Hoc格式:

```
ansible 192.168.1.101 -m nxos_config -a 'lines="ip address 2.2.2.2
255.255.255.0" parents="interface loop222"'
```



## Ad-Hoc (ios\_static\_route)

```
- nxos_static_route:  
  prefix: "192.168.20.64/24"  
  next_hop: "3.3.3.3"  
  route_name: testing  
  pref: 100
```

改编为Ad-Hoc格式(添加):

```
ansible 192.168.1.101 -m nxos_static_route -a 'prefix=6.6.6.0/24  
next_hop=10.1.1.254'
```

乾颐堂现任助教教主 CCIE Security

改编为Ad-Hoc格式(删除):

```
ansible 192.168.1.101 -m nxos_static_route -a 'prefix=6.6.6.0/24  
next_hop=10.1.1.254 state=absent'
```



## Playbook简介

Playbook是由一个或多个“play”组成的列表。play的主要功能在于将事先归为一组的主机装扮成事先通过ansible中的task定义好的角色。从根本上来讲，所谓的task无非是调用ansible的一个module。将多个play组织在一个playbook中，即可以让他们联通起来按事先编排的机制同唱一台大戏。

乾颐堂现任明教教主 CCIE Security



# YAML简介 (1)

<http://docs.ansible.com/ansible/latest/YAMLSyntax.html>

## 特点简介:

We use YAML because it **is easier for humans to read and write** than other common data formats like **XML or JSON**. Further, there are libraries available in most programming languages for working with YAML.

## 开始与结束:

There's another small quirk to YAML. All YAML files (regardless of their association with Ansible or not) can optionally **begin with --- and end with ...**. This is part of the YAML format and indicates the start and end of a document.

## YAML简介 (2)

### 列表:

All members of a list are lines beginning at the same indentation level starting with a "-" (a dash and a space):

```
---
# A list of tasty fruits
fruits:
  - Apple
  - Orange
  - Strawberry
  - Mango
...
```

**严重注意**"-"后面的空格

### 字典:

A dictionary is represented in a simple key: value form (the colon must be followed by a space):

```
# An employee record
martin:
  name: Martin D'vloper
  job: Developer
  skill: Elite
```

**严重注意**":"后面的空格

### 混合嵌套:

```
# Employee records
- martin:
  name: Martin D'vloper
  job: Developer
  skills:
    - python
    - perl
    - pascal
- tabitha:
  name: Tabitha Bitumen
  job: Developer
  skills:
    - lisp
    - fortran
    - erlang
```

## YAML简介 (3)

依然可以采用类JSON的格式

```
---  
martin: {name: Martin D'vloper, job: Developer, skill: Elite}  
fruits: ['Apple', 'Orange', 'Strawberry', 'Mango']
```

乾颐堂现任明教教主 CCIE Security



## 剧本实例

```
---
- hosts: csr
  gather_facts: false
  connection: local
  vars_files:
    - creds.yaml
    - customer_config.yaml

  vars:
    creds:
      username: "{{ username }}"
      password: "{{ password }}"
  tasks:
    - name: config interfaces
      ios_config:
        provider: "{{ creds }}"
        lines:
          - ip address {{ item.ip }} 255.255.255.255
        parents:
          - interface {{ item.port }}
      with_items: "{{ interfaces }}"
      when: (item.router == inventory_hostname)

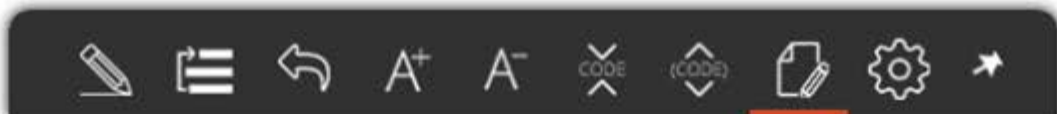
    - name: Configure ACL on Cisco CSR
      ios_config:
        provider: "{{ creds }}"
        lines:
          - access-list 99 permit 1.1.1.1
```

乾颐堂现任明教教主 CCIE Security

# 实战1:creds.yaml

```
1 username: admin
2 password: Cisc0123
3
```

乾颐堂现任明教教主 CCIE Security



## 实战2:customer\_config.yaml

```
1  loopinterfaces:
2    - {router: 192.168.1.101, ip: 10.2.7.1/32, port: loopback0}
3    - {router: 192.168.1.102, ip: 10.2.7.2/32, port: loopback0}
4    - {router: 192.168.1.103, ip: 10.2.7.3/32, port: loopback0}
5
6  etherinterfaces:
7    - {router: 192.168.1.101, ip: 10.2.0.1/30, port: Ethernet1/1}
8    - {router: 192.168.1.102, ip: 10.2.0.2/30, port: Ethernet1/1}
9    - {router: 192.168.1.102, ip: 10.2.0.29/30, port: Ethernet1/2}
10   - {router: 192.168.1.103, ip: 172.16.200.1/24, port: Ethernet1/1}
11   - {router: 192.168.1.103, ip: 10.2.0.30/30, port: Ethernet1/2}
12
13  nvesourceinterfaces:
14    - {router: 192.168.1.101, port: loopback0}
15    - {router: 192.168.1.103, port: loopback0}
16
17  piminterfaces:
18    - {router: 192.168.1.101, port: Ethernet1/1}
19    - {router: 192.168.1.101, port: loopback0}
20    - {router: 192.168.1.102, port: Ethernet1/1}
21    - {router: 192.168.1.102, port: Ethernet1/2}
22    - {router: 192.168.1.102, port: loopback0}
23    - {router: 192.168.1.103, port: Ethernet1/2}
24    - {router: 192.168.1.103, port: loopback0}
25
26  ospfinterfaces:
27    - {router: 192.168.1.101, port: Ethernet1/1}
28    - {router: 192.168.1.102, port: Ethernet1/1}
29    - {router: 192.168.1.102, port: Ethernet1/2}
30    - {router: 192.168.1.103, port: Ethernet1/2}
```



## 实战3:final\_config.yaml

```
88 - name: enable ospf process
89   nxos_ospf:
90     ospf: 1
91     state: present
92
93 - name: enable ospf on interface
94   nxos_interface_ospf:
95     interface: "{{ item.port }}"
96     ospf: 1
97     area: 0
98     cost: default
99   with_items: "{{ ospfinterfaces }}"
100   when: (item.router == inventory_hostname)
101
102 - name: nve interface
103   nxos_vxlan_vtep:
104     interface: nve1
105     source_interface: "{{ item.port }}"
106     shutdown: false
107   with_items: "{{ nvesourceinterfaces }}"
108   when: (item.router == inventory_hostname)
109
110 - name: SAVE CONFIG
111   nxos_config:
112     provider: "{{ creds }}"
113     save_when: modified
114 ...
115
```







## 实战3:运行Ansible剧本

```
[root@localhost ansible]# ansible-playbook final_config.yaml
```

乾颐堂现任明教教主 CCIE Security



# 第四部分

## 简易云实战

乾颐堂现任明教教主 CCIE Security



教主技术进化论

# 实战页面



简易云实战

请选择内存

- 1G内存
- 2G内存

请选择CPU

- 1核CPU
- 2核CPU

提交

```
Script started!!!
```

# 使用Python自动化配置虚拟机,网络和ASA

乾颐堂网络实验室 参数测试 图表 - WINSOCKET 简易云实战 图片 -  
为您想的更多!

## 简易云实战

请选择内存

- 1G内存
- 2G内存

请选择CPU

- 1核CPU
- 2核CPU

提交

```
Script started!!!  
开始配置N9K网络!  
N9K网络配置完毕!  
开始配置ASA!  
ASA配置完毕!  
开始配置vSphere虚拟机  
vSphere虚拟机配置结束!  
全部配置配置完毕! 可能需要再等待3-4分钟直到虚拟机配置结束
```

乾颐堂现任明教教主 CCIE Security