



乾颐学院

乾颐学院 现任明教教主

21. Cisco Netbox实战



乾颐学院 现任明教教主

教主技术进化论 2022

教主VIP, 聊点高级的!

乾颐学院 现任明教教主

课程目录

1. Nexus从Netbox同步配置

2. NSO从Netbox同步配置

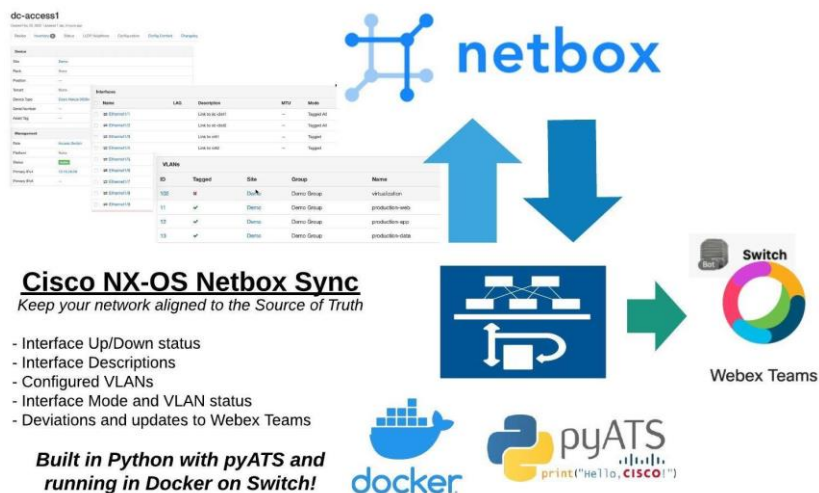
1. Nexus从Netbox同步配置

内部项目地址:

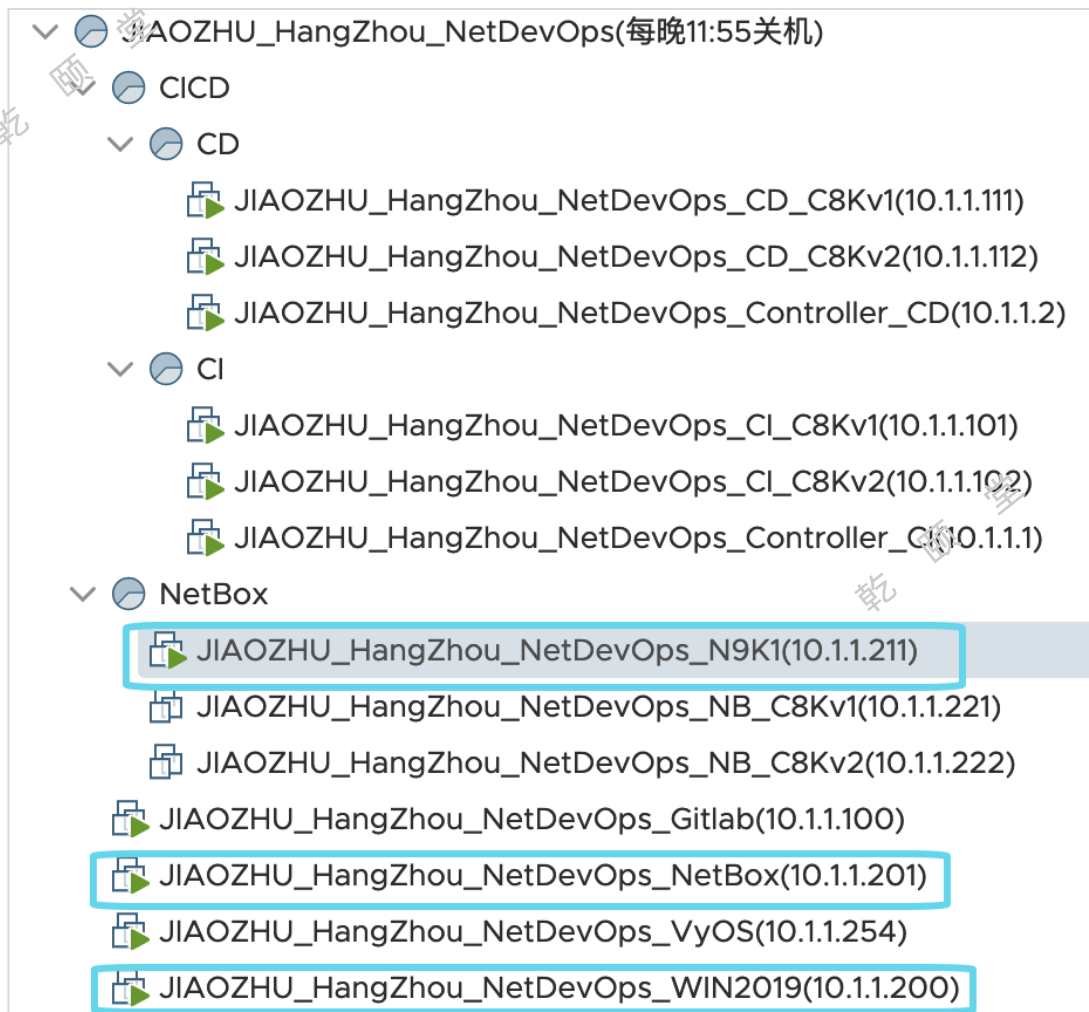
http://pygitlab.qytang.com/root/2022_21_nexus_from_netbox.git

原始项目:

<https://developer.cisco.com/codeexchange/github/repo/hprestor>



设备



```
[root@NetBox netbox-docker]# pwd
/root/netbox-docker
[root@NetBox netbox-docker]# docker-compose up -d
```

- 本地网络设备使用Nexus9K
- 主要检查Nexus接口和VLAN配置
- 设备配置数据位于Netbox

Netbox N9K1

The screenshot shows the Netbox web interface. On the left is a navigation sidebar with categories: Organization, DEVICES, and DEVICE COMPONENTS. The 'DEVICES' section is expanded, and 'Devices' is selected. The main content area is titled 'Devices' and shows a table of 5 devices. The 'N9K1' device is highlighted with a red box and a red circle with the number '3'. Below the table are buttons for '+ Add Components', 'Edit Selected', and 'Delete Selected'. At the bottom right, there is a 'Per Page' dropdown and 'Showing 1-5 of 5'.

<input type="checkbox"/>	Name	Status	Tenant	Site	Location	Rack	Role	Manufacturer	Type	IP Address	<input type="checkbox"/>
<input type="checkbox"/>	C8Kv1	Active	—	QYTANG_NSO	Beijing-NSO-Location	Rack02	Qytang-NSO-Devices	Cisco	C8Kv	192.168.1.1/24	<input type="checkbox"/>
<input type="checkbox"/>	C8Kv2	Active	—	QYTANG_NSO	Beijing-NSO-Location	Rack02	Qytang-NSO-Devices	Cisco	C8Kv	192.168.1.2/24	<input type="checkbox"/>
<input checked="" type="checkbox"/>	N9K1	Active	—	QYTANG BEIJING	亿城国际516	Rack01	QYT-LAB-Device	Cisco	N9K	10.1.1.211/24	<input type="checkbox"/>
<input type="checkbox"/>	R1	Active	—	QYTANG BEIJING	亿城国际516	Rack01	QYT-LAB-Device	Cisco	CSR1Kv	10.1.1.221/24	<input type="checkbox"/>
<input type="checkbox"/>	R2	Active	—	QYTANG BEIJING	亿城国际516	Rack01	QYT-LAB-Device	Cisco	CSR1Kv	10.1.1.222/24	<input type="checkbox"/>

Netbox N9K1接口

Devices > QYTANG BEIJING

N9K1

Created 2022-07-22 00:50 · Updated 2 days ago

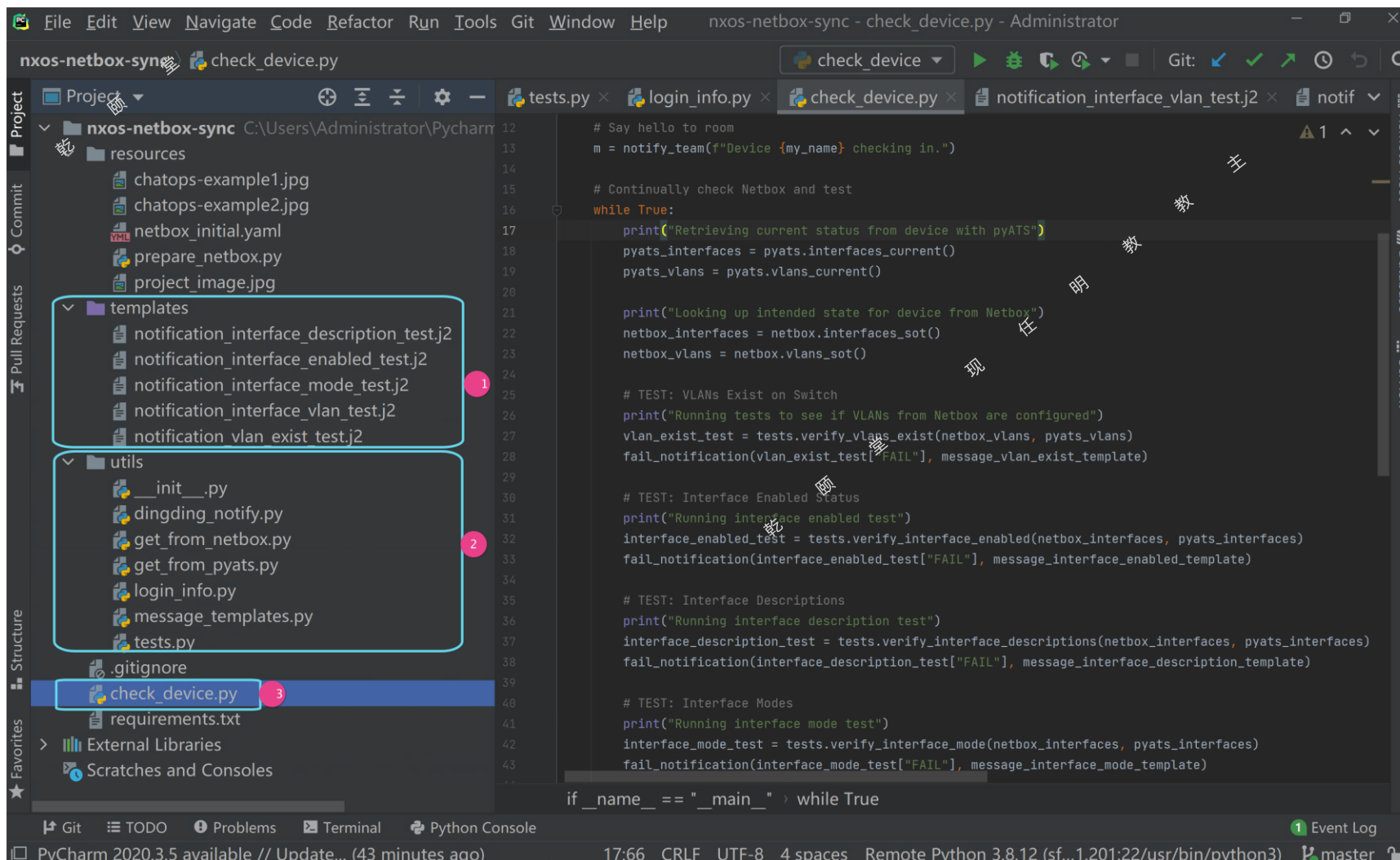
Device **Interfaces 4** Config Context Journal Change Log

Quick search

Name	Label	Enabled	Type	Parent interface	LAG	MTU	Mode	Description	IP Addresses	Cable	Connection	Untagged VLAN
<input type="checkbox"/> Ethernet1/1		<input checked="" type="checkbox"/>	1000BASE-F (1GE)				Access	qytang netbox new e1/1				qytang102 (102)
<input type="checkbox"/> Ethernet1/2		<input checked="" type="checkbox"/>	1000BASE-T (1GE)				Access	qytang netbox e1/2				qytang102 (102)
<input type="checkbox"/> Ethernet1/3		<input checked="" type="checkbox"/>	1000BASE-T (1GE)				Tagged	qytang netbox e1/3				qytang101 (101)
<input type="checkbox"/> mgmt0		<input checked="" type="checkbox"/>	1000BASE-T (1GE)						10.1.1.211/24			

Per Page
Showing 1-4 of 4

代码视图



```

# Say hello to room
m = notify_team(f"Device {my_name} checking in.")

# Continually check Netbox and test
while True:
    print("Retrieving current status from device with pyATS")
    pyats_interfaces = pyats.interfaces_current()
    pyats_vlans = pyats.vlans_current()

    print("Looking up intended state for device from Netbox")
    netbox_interfaces = netbox.interfaces_sot()
    netbox_vlans = netbox.vlans_sot()

    # TEST: VLANs Exist on Switch
    print("Running tests to see if VLANs from Netbox are configured")
    vlan_exist_test = tests.verify_vlans_exist(netbox_vlans, pyats_vlans)
    fail_notification(vlan_exist_test["FAIL"], message_vlan_exist_template)

    # TEST: Interface Enabled Status
    print("Running interface enabled test")
    interface_enabled_test = tests.verify_interface_enabled(netbox_interfaces, pyats_interfaces)
    fail_notification(interface_enabled_test["FAIL"], message_interface_enabled_template)

    # TEST: Interface Descriptions
    print("Running interface description test")
    interface_description_test = tests.verify_interface_descriptions(netbox_interfaces, pyats_interfaces)
    fail_notification(interface_description_test["FAIL"], message_interface_description_template)

    # TEST: Interface Modes
    print("Running interface mode test")
    interface_mode_test = tests.verify_interface_mode(netbox_interfaces, pyats_interfaces)
    fail_notification(interface_mode_test["FAIL"], message_interface_mode_template)

if __name__ == "__main__":
    while True:
  
```

第一部分: 通知markdown模板

第二部分: 工具代码

- dingding_notify.py [钉钉通知]
- get_from_netbox.py [从netbox获取设备配置数据]
- get_from_pyats.py [pyats获取设备当前配置状态,并且配置设备]
- login_info.py [登录认证信息]
- message_templates.py [读取markdown模板]
- tests.py [测试netbox和pyats数据的差别]

第三部分: check_device.py [最终汇总]

check_device.py步骤分析

1. 从netbox获取配置
2. pyats从设备获取配置
3. 比较netbox和pyats配置的区别
4. 使用pyats补齐或者纠正配置

值得参考代码.1

get_from_netbox.py

从netbox获取接口和vlan的配置

```
import pynetbox
from utils.login_info import netbox_url, netbox_token, switch_name

netbox = pynetbox.api(netbox_url, token=netbox_token)
device = netbox.dcim.devices.get(name=switch_name)
switch_mgmt_ip = str(device.primary_ip).split('/')[0]

def interfaces_sot():
    interfaces = netbox.dcim.interfaces.filter(device_id=device.id, mgmt_only=False)
    return interfaces

def vlans_sot():
    vlans = netbox.ipam.vlans.filter(site_id=device.site.id)
    return vlans
```


值得参考代码.2

```
get_from_pyats.py
```

直接从Python字典读取,
不必使用testbed文件

```
from genie.testbed import load
from genie.libs.conf.vlan import Vlan
from genie.libs.conf.interface import Interface
from utils.login_info import switch_name, swtich_mgmt_port
from utils.login_info import device_username, device_password
from utils.get_from_netbox import switch_mgmt_ip

device_details = {"devices": {
    switch_name: {
        "protocol": "ssh",
        "ip": switch_mgmt_ip,
        "port": swtich_mgmt_port,
        "username": device_username,
        "password": device_password,
        "os": "nxos",
        "ssh_options": "-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null",
    }}

testbed = load(device_details)
device = testbed.devices[switch_name]
# device.connect(learn_hostname=True)
device.connect(learn_hostname=True,
               log_stdout=False,
               ssh_options='-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null')
```

值得参考代码.3

```
def interface_access_configure(netbox_interface):  
    if netbox_interface.mode.label == "Access":  
        # Configure native and tagged vlans on a trunk  
        if netbox_interface.name in device.interfaces.keys():  
            new_interface = device.interfaces[netbox_interface.name]  
        else:  
            new_interface = Interface(name=netbox_interface.name, device=device)  
            new_interface.switchport_enable = True  
  
            new_interface.switchport_mode = "access"  
            new_interface.access_vlan = str(netbox_interface.untagged_vlan.vid)  
            return new_interface  
    else:  
        print(f"Interface {netbox_interface.name} is NOT an access interface.")  
        return False
```

get_from_pyats.py

配置交换机接口的模式与vlan

值得参考代码.4

test.p
y

比较VLAN配置,找到那些缺少的和名称不一致的VLAN

```
def verify_vlans_exist(netbox_vlans, pyats_vlans):
    results = {
        "status": False,
        "PASS": [],
        "FAIL": [],
    }
    for vlan in netbox_vlans:
        if str(vlan.vid) in pyats_vlans.keys():
            # print(f" ✓ {vlan.vid} exists on switch")
            if vlan.name == pyats_vlans[str(vlan.vid)]["name"]:
                print(f" ✓ {vlan.display} exists with correct name on switch")
                results["PASS"].append(vlan)
            else:
                print(f" ✗ {vlan.display} exists but with WRONG name on switch")
                results["FAIL"].append(vlan)
        else:
            print(f" ✗ {vlan.vid} MISSING from switch")
            results["FAIL"].append(vlan)

    return results
```

执行效果

Console显示效果

check_device x

```
ssh://root@10.1.1.201:22/usr/bin/python3 -u /nxos-netbox-sync/check_device.py
Retrieving current status from device with pyATS
Looking up intended state for device from Netbox
Running tests to see if VLANs from Netbox are configured
X 101 MISSING from switch
X 102 MISSING from switch
Running interface enabled test
✓ Ethernet1/1 was correctly found to be UP/UP on switch
✓ Ethernet1/2 was correctly found to be UP/UP on switch
✓ Ethernet1/3 was correctly found to be UP/UP on switch
Running interface description test
X Ethernet1/1 incorrectly has NO description configured on switch. It should be 'qytang netbox new e1/1'
X Ethernet1/2 incorrectly has NO description configured on switch. It should be 'qytang netbox e1/2'
X Ethernet1/3 incorrectly has NO description configured on switch. It should be 'qytang netbox e1/3'
Running interface mode test
✓ Ethernet1/1 is correctly configured as an access port
```

钉钉通告效果



乾颐堂服务机器人 机器人

Device switch checking in.

收到

回复



乾颐堂服务机器人 机器人

The following interfaces from Netbox have incorrect descriptions configured.

- Ethernet1/2
- Ethernet1/3

收到

回复



乾颐堂服务机器人 机器人

The following interfaces from Netbox have incorrect access and/or trunk vlan configurations.

- Ethernet1/1

收到

回复



乾颐堂服务机器人 机器人

I am updating my Interface switchport configurations.

收到

回复

2. NSO从Netbox同步配置

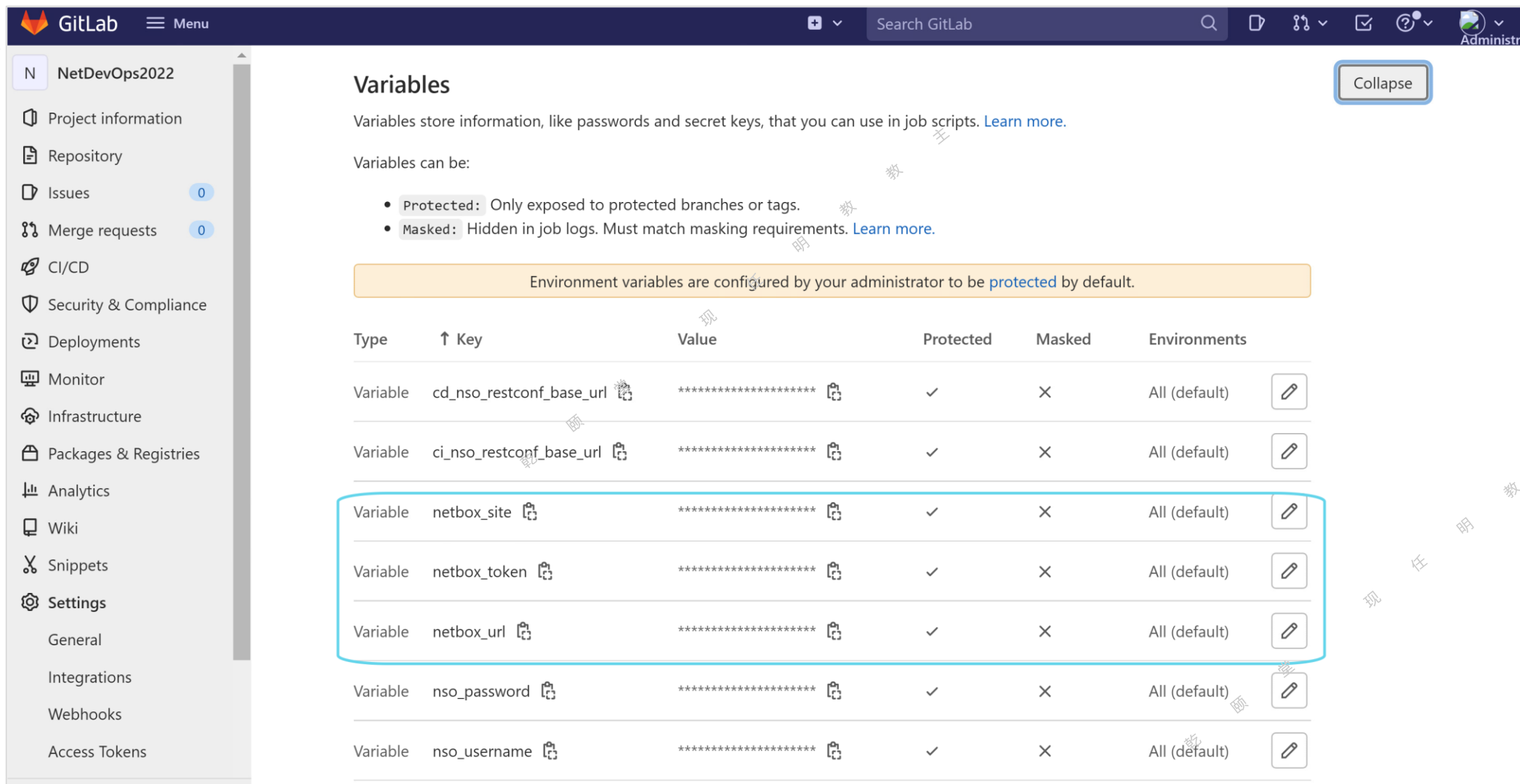
内部项目地址:

<http://pygitlab.qytang.com/root/netdevops2022.git>

分支:

[netbox_ci](#)

Variables



The screenshot shows the GitLab interface for the 'NetDevOps2022' project. The 'Variables' section is active, displaying a list of environment variables. A blue box highlights the 'netbox_site', 'netbox_token', and 'netbox_url' variables. A yellow box highlights a message stating that environment variables are configured by the administrator to be protected by default.








Variables

Variables store information, like passwords and secret keys, that you can use in job scripts. [Learn more.](#)

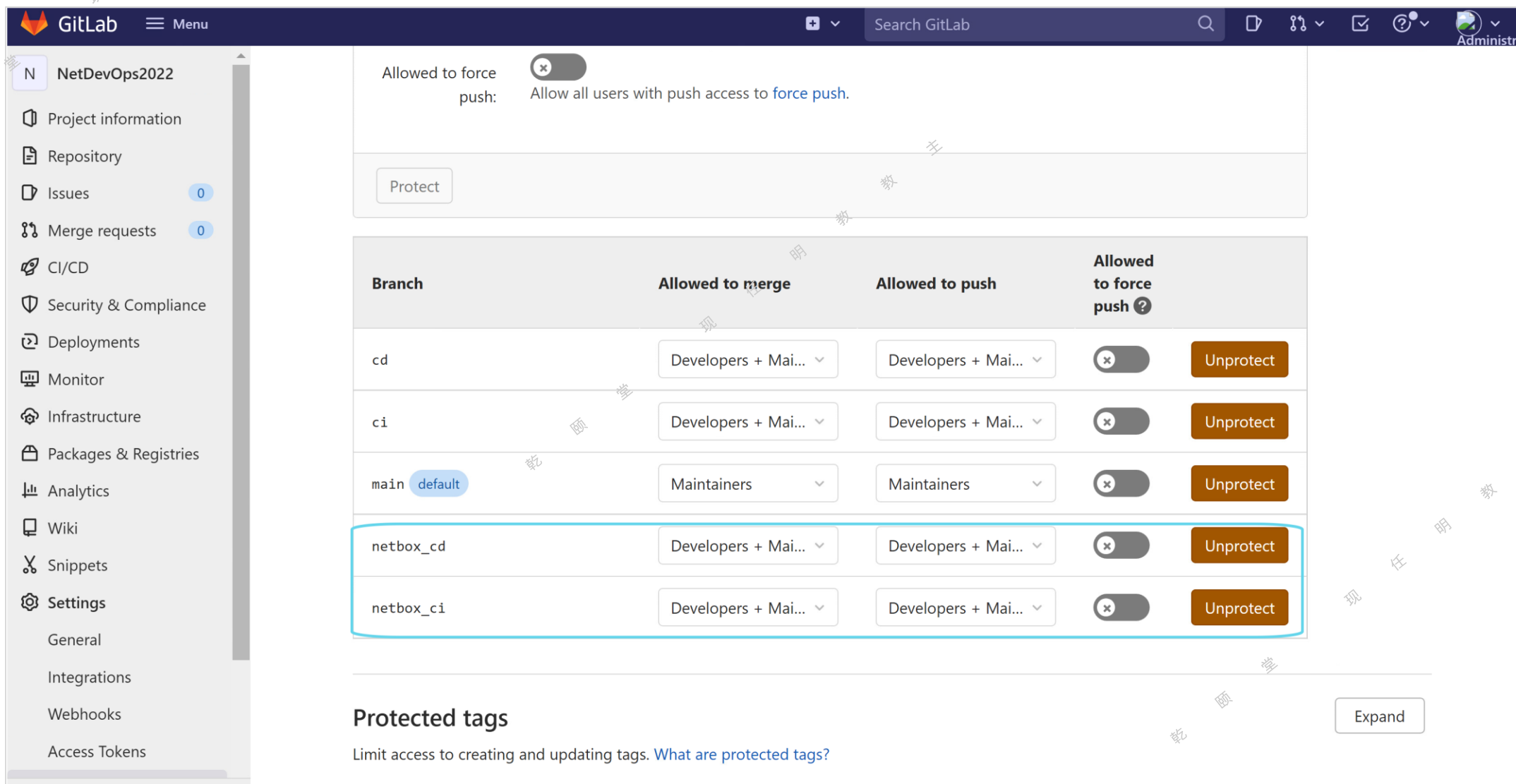
Variables can be:

- **Protected:** Only exposed to protected branches or tags.
- **Masked:** Hidden in job logs. Must match masking requirements. [Learn more.](#)

Environment variables are configured by your administrator to be **protected** by default.

Type	Key	Value	Protected	Masked	Environments	
Variable	cd_nso_restconf_base_url	*****	✓	✗	All (default)	
Variable	ci_nso_restconf_base_url	*****	✓	✗	All (default)	
Variable	netbox_site	*****	✓	✗	All (default)	
Variable	netbox_token	*****	✓	✗	All (default)	
Variable	netbox_url	*****	✓	✗	All (default)	
Variable	nso_password	*****	✓	✗	All (default)	
Variable	nso_username	*****	✓	✗	All (default)	

保护分支



Allowed to force push: Allow all users with push access to [force push](#).

Protect

Branch	Allowed to merge	Allowed to push	Allowed to force push ?	
cd	Developers + Mai...	Developers + Mai...	<input checked="" type="checkbox"/>	Unprotect
ci	Developers + Mai...	Developers + Mai...	<input checked="" type="checkbox"/>	Unprotect
main <small>default</small>	Maintainers	Maintainers	<input checked="" type="checkbox"/>	Unprotect
netbox_cd	Developers + Mai...	Developers + Mai...	<input checked="" type="checkbox"/>	Unprotect
netbox_ci	Developers + Mai...	Developers + Mai...	<input checked="" type="checkbox"/>	Unprotect

Protected tags Expand

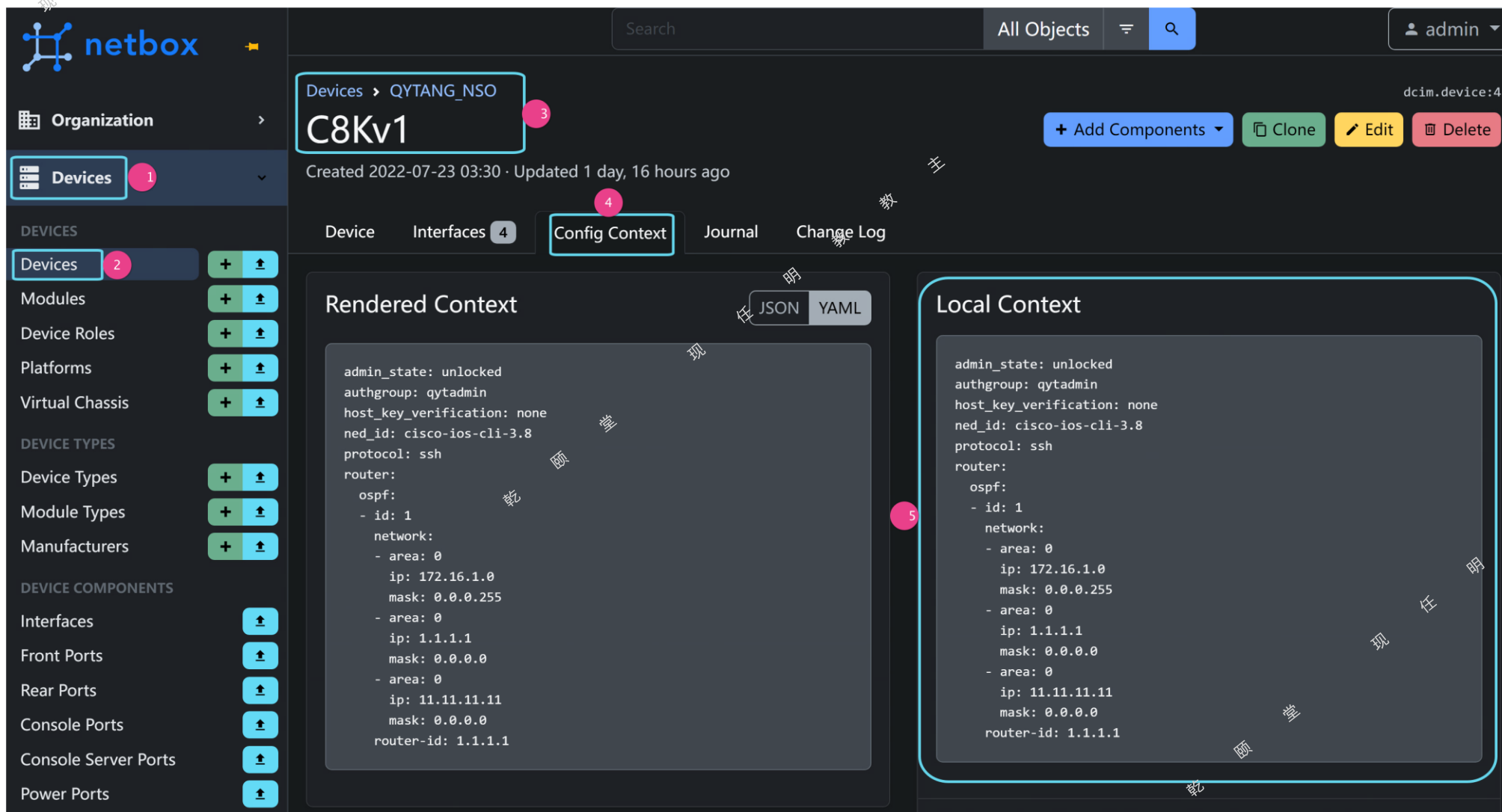
Limit access to creating and updating tags. [What are protected tags?](#)

Netbox 设备配置

The screenshot shows the Netbox web interface. On the left is a navigation sidebar with categories like Organization, DEVICES, DEVICE TYPES, and DEVICE COMPONENTS. The 'Devices' page is active, displaying a table of 5 devices. The first two rows are highlighted with a blue selection box. At the bottom of the table, there are buttons for '+ Add Components', 'Edit Selected', and 'Delete Selected'. The footer shows the date and time: 2022-07-25 02:55 UTC.

<input type="checkbox"/>	Name	Status	Tenant	Site	Location	Rack	Role	Manufacturer	Type	IP Address	<input type="checkbox"/>
<input type="checkbox"/>	C8Kv1	Active	—	QYTANG_NSO	Beijing-NSO-Location	Rack02	Qytang-NSO-Devices	Cisco	C8Kv	192.168.1.1/24	<input type="checkbox"/>
<input type="checkbox"/>	C8Kv2	Active	—	QYTANG_NSO	Beijing-NSO-Location	Rack02	Qytang-NSO-Devices	Cisco	C8Kv	192.168.1.2/24	<input type="checkbox"/>
<input type="checkbox"/>	N9K1	Active	—	QYTANG BEIJING	亿城国际516	Rack01	QYT-LAB-Device	Cisco	N9K	10.1.1.211/24	<input type="checkbox"/>
<input type="checkbox"/>	R1	Active	—	QYTANG BEIJING	亿城国际516	Rack01	QYT-LAB-Device	Cisco	CSR1Kv	10.1.1.221/24	<input type="checkbox"/>
<input type="checkbox"/>	R2	Active	—	QYTANG BEIJING	亿城国际516	Rack01	QYT-LAB-Device	Cisco	CSR1Kv	10.1.1.222/24	<input type="checkbox"/>

Netbox 设备Config Context



The screenshot displays the Netbox web interface for configuring a device. The left sidebar shows the navigation menu with 'Devices' highlighted. The main content area shows the configuration for device 'C8Kv1' under the 'QYTANG_NS0' organization. The 'Config Context' tab is selected, showing two context views: 'Rendered Context' and 'Local Context'. The 'Rendered Context' view shows the configuration in JSON format, and the 'Local Context' view shows the configuration in YAML format. The configuration includes details such as admin state, authgroup, host key verification, ned_id, protocol, and router configuration with OSPF settings.

Organization: QYTANG_NS0

Device: C8Kv1

Created 2022-07-23 03:30 · Updated 1 day, 16 hours ago

Device | Interfaces 4 | **Config Context** | Journal | Change Log

Rendered Context (JSON)

```
admin_state: unlocked
authgroup: qytadmin
host_key_verification: none
ned_id: cisco-ios-cli-3.8
protocol: ssh
router:
  ospf:
    - id: 1
      network:
        - area: 0
          ip: 172.16.1.0
          mask: 0.0.0.255
        - area: 0
          ip: 1.1.1.1
          mask: 0.0.0.0
        - area: 0
          ip: 11.11.11.11
          mask: 0.0.0.0
      router-id: 1.1.1.1
```

Local Context (YAML)

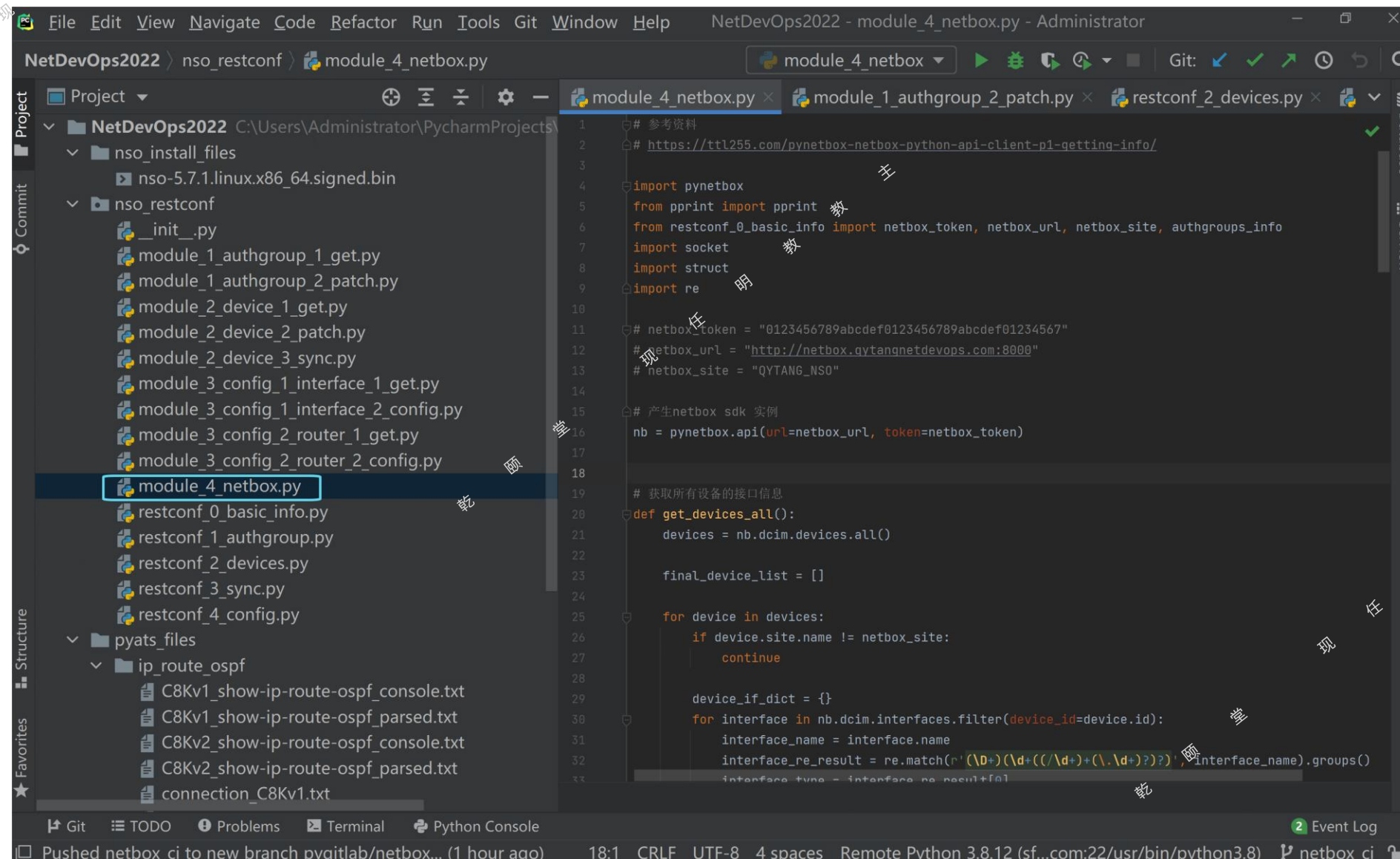
```
admin_state: unlocked
authgroup: qytadmin
host_key_verification: none
ned_id: cisco-ios-cli-3.8
protocol: ssh
router:
  ospf:
    - id: 1
      network:
        - area: 0
          ip: 172.16.1.0
          mask: 0.0.0.255
        - area: 0
          ip: 1.1.1.1
          mask: 0.0.0.0
        - area: 0
          ip: 11.11.11.11
          mask: 0.0.0.0
      router-id: 1.1.1.1
```

Netbox 接口配置

Netbox interface configuration for device C8Kv1. The interface table is as follows:

Name	Label	Enabled	Type	Parent interface	LAG	MTU	Mode	Description	IP Addresses	Cable	Connection	Untagged VLAN
GigabitEthernet1		✓	1000BASE-T (1GE)						172.16.1.1/24	#4	C8Kv2 > GigabitEthernet1	
GigabitEthernet2		✓	1000BASE-T (1GE)						192.168.1.1/24	#3	C8Kv2 > GigabitEthernet2	
Loopback0		✓	Virtual						1.1.1.1/24			
Loopback1		✗	Virtual						11.11.11.11/24			

从Netbox获取数据的代码



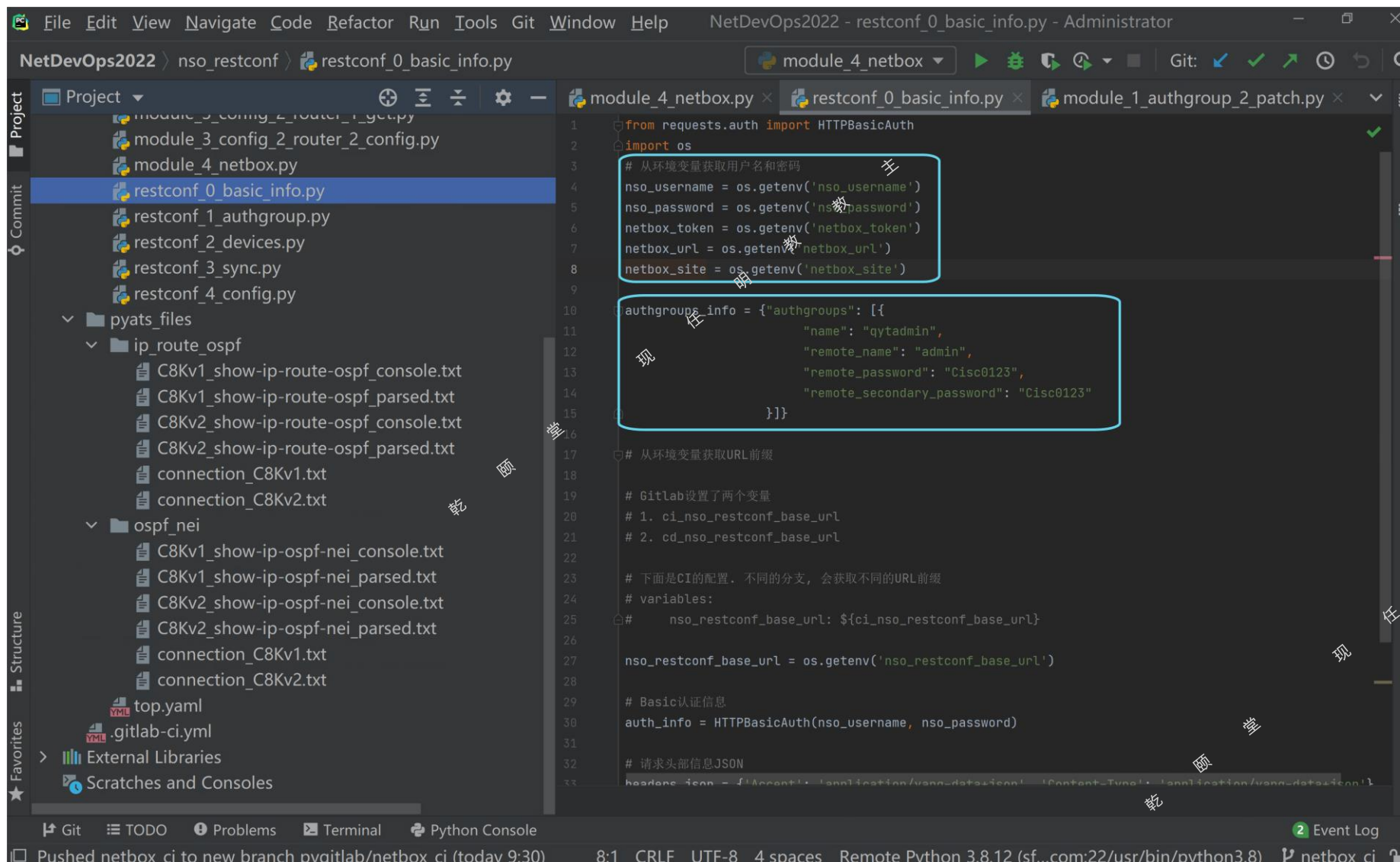
```
File Edit View Navigate Code Refactor Run Tools Git Window Help NetDevOps2022 - module_4_netbox.py - Administrator
NetDevOps2022 > nso_restconf > module_4_netbox.py
Project
  NetDevOps2022 C:\Users\Administrator\PycharmProjects
  nso_install_files
    nso-5.7.1.linux.x86_64.signed.bin
  nso_restconf
    _init_.py
    module_1_authgroup_1_get.py
    module_1_authgroup_2_patch.py
    module_2_device_1_get.py
    module_2_device_2_patch.py
    module_2_device_3_sync.py
    module_3_config_1_interface_1_get.py
    module_3_config_1_interface_2_config.py
    module_3_config_2_router_1_get.py
    module_3_config_2_router_2_config.py
    module_4_netbox.py
    restconf_0_basic_info.py
    restconf_1_authgroup.py
    restconf_2_devices.py
    restconf_3_sync.py
    restconf_4_config.py
  pyats_files
    ip_route_ospf
      C8Kv1_show-ip-route-ospf_console.txt
      C8Kv1_show-ip-route-ospf_parsed.txt
      C8Kv2_show-ip-route-ospf_console.txt
      C8Kv2_show-ip-route-ospf_parsed.txt
      connection_C8Kv1.txt
Commit
Structure
Favorites
Database
SciView

1  # 参考资料
2  # https://ttl255.com/pynetbox-netbox-python-api-client-p1-getting-info/
3
4  import pynetbox
5  from pprint import pprint
6  from restconf_0_basic_info import netbox_token, netbox_url, netbox_site, authgroups_info
7  import socket
8  import struct
9  import re
10
11 # netbox_token = "0123456789abcdef0123456789abcdef01234567"
12 # netbox_url = "http://netbox.qytangnetdevops.com:8000"
13 # netbox_site = "QYTANG_NS0"
14
15 # 产生netbox sdk 实例
16 nb = pynetbox.api(url=netbox_url, token=netbox_token)
17
18
19 # 获取所有设备的接口信息
20 def get_devices_all():
21     devices = nb.dcim.devices.all()
22
23     final_device_list = []
24
25     for device in devices:
26         if device.site.name != netbox_site:
27             continue
28
29         device_if_dict = {}
30         for interface in nb.dcim.interfaces.filter(device_id=device.id):
31             interface_name = interface.name
32             interface_re_result = re.match(r'(\d+)(\d+((\d+)(\.\d+)?))?', interface_name).groups()
33             interface_type = interface.re_result[0]
```

Git | TODO | Problems | Terminal | Python Console | Event Log

Pushed netbox ci to new branch pygitlab/netbox... (1 hour ago) 18:1 CRLF UTF-8 4 spaces Remote Python 3.8.12 (sf...com:22/usr/bin/python3.8) netbox ci

从环境变量读取数据，手动配置的authgroups信息



```
File Edit View Navigate Code Refactor Run Tools Git Window Help NetDevOps2022 - restconf_0_basic_info.py - Administrator
NetDevOps2022 > nso_restconf > restconf_0_basic_info.py
module_4_netbox
restconf_0_basic_info.py
module_1_authgroup_2_patch.py
module_5_config_2_router_1_get.py
module_3_config_2_router_2_config.py
restconf_1_authgroup.py
restconf_2_devices.py
restconf_3_sync.py
restconf_4_config.py
pyats_files
ip_route_ospf
C8Kv1_show-ip-route-ospf_console.txt
C8Kv1_show-ip-route-ospf_parsed.txt
C8Kv2_show-ip-route-ospf_console.txt
C8Kv2_show-ip-route-ospf_parsed.txt
connection_C8Kv1.txt
connection_C8Kv2.txt
ospf_nei
C8Kv1_show-ip-ospf-nei_console.txt
C8Kv1_show-ip-ospf-nei_parsed.txt
C8Kv2_show-ip-ospf-nei_console.txt
C8Kv2_show-ip-ospf-nei_parsed.txt
connection_C8Kv1.txt
connection_C8Kv2.txt
top.yaml
.gitlab-ci.yml
External Libraries
Scratches and Consoles
Git TODO Problems Terminal Python Console Event Log
Pushed netbox ci to new branch pygitlab/netbox ci (today 9:30) 8:1 CRLF UTF-8 4 spaces Remote Python 3.8.12 (sf...com:22/usr/bin/python3.8) netbox ci

1 from requests.auth import HTTPBasicAuth
2 import os
3 # 从环境变量获取用户名和密码
4 nso_username = os.getenv('nso_username')
5 nso_password = os.getenv('nso_password')
6 netbox_token = os.getenv('netbox_token')
7 netbox_url = os.getenv('netbox_url')
8 netbox_site = os.getenv('netbox_site')
9
10 authgroups_info = {"authgroups": [{
11     "name": "qytadmin",
12     "remote_name": "admin",
13     "remote_password": "Cisco0123",
14     "remote_secondary_password": "Cisco0123"
15 }]}
16
17 # 从环境变量获取URL前缀
18
19 # Gitlab设置了两个变量
20 # 1. ci_nso_restconf_base_url
21 # 2. cd_nso_restconf_base_url
22
23 # 下面是CI的配置，不同的分支，会获取不同的URL前缀
24 # variables:
25 # ci_nso_restconf_base_url: ${ci_nso_restconf_base_url}
26
27 nso_restconf_base_url = os.getenv('nso_restconf_base_url')
28
29 # Basic认证信息
30 auth_info = HTTPBasicAuth(nso_username, nso_password)
31
32 # 请求头部信息JSON
33 headers_json = {'Accept': 'application/vnd.yang.data+json', 'Content-Type': 'application/vnd.yang.data+json'}
```