Netero Documentation

Release 1.1.0

Renato Almeida de Oliveira

Jun 15, 2020

Netero is a simple utiliy to help network manangement, that aims to encpsulate vendors' specifics sintax in YAML models based on YANG data model, in this realease it is possible to perfom the following:

- Manage your configuration Backups
- Integrate your backups with Gogs API, with git push and commit
- Consume PeeringDB API for prospection of when some Autonomous System (AS) lies on the same IXP as your AS
- Consume PeeringDB API for gather AS informations as max IPv4/IPv6 prefixes, interfaces address, IRR-ASSET
- Encapsulate BGPq3 or BGPq4 for generation of prefix-list of a given IRR-ASSET

CHAPTER 1

Quick Start

Clone our sample repository , where you're going to find some playbooks samples, and a Jinja2 template for RouterOS policy configurations.

CHAPTER 2

Next Steps

- The configuration module, that will read model files and configure the network devices

2.1 Installing Netero

2.1.1 Requiriments

This collection requires the following packages:

- [git]
- [bgpq3] or [bgpq4]
- [requests]
- [netmiko]
- [ncclient]

2.1.2 Installing

Ansible Galaxy is the default source of Ansible collections for the ansible-galaxy tool. We can install Netero Ansible collection by running:

```
$ ansible-galaxy collection install renatoalmeidaoliveira.netero
```

2.2 Use Cases

2.2.1 Configuration Backup

For the configuration backup you can use the backup mode and netero roles or use the modules directly.

Using the roles

The netero roles encapsulate the configuration gathering of the devices, and for utilization you must setup the netero mode to backup, and separate your devices in groups of vendors, i.e., IOS, IOS-XR, ROUTEROS, etc.

So for configuration management your playbook must perform the following tasks:

1. Create the repository in your favorite repository manager, in the example the gogs_createrepo are going to be used.
2. Clone the previously created repositories.

```
- name: Setup repositories
  collections:
    - renatoalmeidaoliveira.netero

  hosts: all

  tasks:

   - name: Create Repository
     gogs_createrepo:
        gogsURL: "http://gogs.local:3000/"
        organization: "netero"
        name: "{{ inventory_hostname }}"
        accessToken: "0bba381ce3df8208591e067a4abae72a556974ce"
     delegate_to: localhost

   - name: Clone Repository
     git:
        repo: "git@gogs.local:netero/{{ inventory_hostname }}.git"
        dest: "{{ inventory_hostname }}"
     delegate_to: localhost
```

3. Create a play for each of your device vendors and set the respective group.

```
- name: Collect IOS-XR configuration
  collections:
    - renatoalmeidaoliveira.netero
  vars:
    - netero_mode: "backup"
  hosts: iosxr
  roles:
    - iosxr
- name: Collect MK configuration
  collections:
    - renatoalmeidaoliveira.netero
  vars:
    - netero_mode: "backup"
  hosts: routeros
  roles:
    - routeros
```

Warning: Remember to configure the netero_mode variable to "backup"

Suported Vendors:

- IOS

- IOS-XR
- MikroTik
- Fortgate

4. Commit and push the repositories .

```
- name: Commit and push reporitories

  collections:
    - renatoalmeidaoliveira.netero

  hosts: all

  tasks:

  - name: Commit
    git_commit:
      path: "{{ inventory_hostname }}"
    delegate_to: localhost
  - name: Push
    git_push:
      path: "{{ inventory_hostname }}"
    delegate_to: localhost
```

Using the modules

For make your backup with the modules you could use the following steps

1. Create the repository on Gogs, if the repository already exists the module runs without changes

```
- name: Create Repository
  gogs_createrepo:
     gogsURL: "<Gogs URL>"
     organization: "acme"
     name: "{{ inventory_hostname }}"
     accessToken: <accessToken>
   delegate_to: localhost
```

2. Clone the configuration repository

```
- name: Clone Repository
  git:
    repo: "<Gogs URL>:<org|user>/{{ inventory_hostname }}.git"
    dest: "{{ inventory_hostname }}"
  delegate_to: localhost
```

Tip: Setup the SSH Keys with Gogs and the Server runnig Ansible, instead of using username and password

3. Extract your device configuration using any module you want.

```
- name: Gather device configuration
  routeros_facts:
```

(continues on next page)

(continued from previous page)

```
    gather_subset:
      - config
```

4. Create the configuration File

```
- name: Create configuration File
  copy:
    content: "{{ansible_net_config}}"
    dest: "{{ inventory_hostname }}/{{ inventory_hostname }}.cfg"
  delegate_to: localhost
```

5. Sanitize your configuration file

In this step remove any line containing passwords, and the timestamp of the collect, ie usually the first line

```
- name: Sanitize Configuration File
  lineinfile:
    path: "{{ inventory_hostname }}/{{ inventory_hostname }}.cfg"
    state: absent
    regexp: '# \w+/\d+/\d+ \d+:\d+:\d+.*'
  delegate_to: localhost
```

6. Commit the local repository

Important: Make sure that the user runnig the ansible-playbook have the git user.name and user.email configured

```
- name: Commit
  git_commit:
     path: "{{ inventory_hostname }}"
  delegate_to: localhost
```

7. Push the repository

```
- name: Push
  git_push:
    path: "{{ inventory_hostname }}"
  delegate_to: localhost
```

2.2.2 Creating routing policies

Using the modules

For creating routing policies we gonna use the peergindb_getasn and irr_prefix modules, for extract all the ASN informations and then using Jinja2 templates it is possible to create the desired configuration

1. Consultando a API do PeeringDB para extrair as informações do ASN:

```
- name: Get ASN Data
  peeringdb_getasn:
    asn: 204092
    ix-id: 1670
  register: ASNData
```

SAMPLE OUTPUT

```
  "ASNData.message": {
    "ASN": 204092,
    "info_ipv6": true,
    "info_prefixes4": 20,
    "info_prefixes6": 20,
    "info_unicast": true,
    "interfaces": [
        {
            "ipaddr4": "185.1.89.10",
            "ipaddr6": "2001:7f8:b1::a",
            "speed": 1000
        }
    ],
    "irr_as_set": [
        "AS-GRIFON"
    ],
    "poc_set": []
}
```

2. Using the ASN Data as input for irr_prefix:

```
- name: Get IRR Prefix
  irr_prefix:
    asn32Safe: True
    IPv: 4
    asSet: "{{ item }} "
    aggregate: true
  with_items:
    - "{{ ASNData.message.irr_as_set }}"
  register: IRRData
```

SAMPLE OUTPUT

```
"IRRData.results": [
        {
            "ansible_loop_var": "item",
            "changed": true,
            "failed": false,
            "invocation": {
                "module_args": {
                    "IPv": "4",
                    "aggregate": true,
                    "asSet": "AS-GRIFON ",
                    "asn32Safe": true
                }
            },
            "item": "AS-GRIFON",
            "message": {
                "irr_prefix": [
                    {
                        "exact": true,
                        "prefix": "23.128.24.0/24"
                    },
                    {
                        "exact": true,
                        "prefix": "23.128.25.0/25"
                    },
```

(continues on next page)

(continued from previous page)

```
                    {
                        "exact": true,
                        "prefix": "23.128.25.240/28"
                    }
                ]
            }
        }
    ]
}
```

3. Create a Jinja2 template for create your device configuration
4. Apply the configuration to your device

2.2.3 Prospect ASN

Using the modules

This module was created to simplify ASN information gathering, imagine the following scenario:

- Your NetFlow monitoring system shows you that 30% of your traffic goes to some ASN, and to optimize your traffic you want to make an peering agreement with that ASN but you don't know any contact number and if that ASN is on the same IXP with your.
- After getting that information you want to send the Policy contact an email asking for the peering agreement

That can be configured as follow:

1. Configure the module with your ASN in src-asn and the desired ASNs in dst-asn, and with your peeringDB username and password:

```
- name: Prospect ASN Data
  peeringdb_prospect:
    src-asn: 1916
    dst-asn: 1251
    username : Joe
    password: secret
```

SAMPLE OUTPUT

```
"prospectData": {
    "changed": false,
    "failed": false,
    "message": [
        {
            "1251": {
                "IXs": [
                    {
                        "id": 171,
                        "name": "IX.br (PTT.br) São Paulo: ATM/MPLA"
                    },
                    {
                        "id": 119,
                        "name": "Equinix São Paulo: Equinix IX - SP Metro"
                    }
                ],
```

(continues on next page)

(continued from previous page)

```
                "name": "ANSP",
                "poc_set": [
                    {
                        "created": "*************",
                        "email": "*****@*****",
                        "id": *******,
                        "name": "********",
                        "phone": "**********",
                        "role": "Technical",
                        "status": "ok",
                        "updated": "************",
                        "url": "*******",
                        "visible": "Users"
                    }
                ]
            }
        }
    ]
}
```

Warning: Contact data sanitized.

2. Create a template with Jinja using ASN data
3. Send an email asking for your peering session

2.3 Modules

2.3.1 git_commit – Makes git commit on repository

Synopsis

This module runs git status and if there are any changes on the repository makes git add * ana git commit

Requirements

The below requirements are needed on the host that executes this module.

- git>=1.7.1 (the command line tool)

Parameters

path (True, any, None) The repository path

commitMessage (False, any, None) Sets the commit message, if none uses timestamp

Examples

```
- name: Commit repo
  git_commit:
    path: /home/repository

- name: Commit with message
  git_commit:
    path: /home/repository
    commitMessage: "Commit executed by Ansible"
```

Return Values

message (success, dict,) object

Status

- This is not guaranteed to have a backwards compatible interface. *[preview]*
- This is maintained by community.

Authors

- Renato Almeida de Oliveira (renato.a.oliveira@pm.me)

2.3.2 git_push – Makes git push on repository

Synopsis

This module runs git status -sb and if there are any changes on the repository make git push

This module assumes that Ansible server and the Git Server can connect

Requirements

The below requirements are needed on the host that executes this module.

- git>=1.7.1 (the command line tool)

Parameters

path (True, any, None) The repository path

Examples

```
- name: Push repo
  git_push:
    path: /home/repository
```

Return Values

message (success, dict,) object

Status

- This is not guaranteed to have a backwards compatible interface. *[preview]*
- This is maintained by community.

Authors

- Renato Almeida de Oliveira (renato.a.oliveira@pm.me)

2.3.3 gogs_createrepo – Create a repository on Gogs

Synopsis

This module encapusles Gogs API to create a repository

Parameters

gogsURL (True, any, None) The Gogs Server URL

user (False, any, None) The user that owns the repository, This argument is mutually exclusive with organization.

organization (False, any, None) The organization that owns the repository, This argument is mutually exclusive with user.

name (True, any, None) The repository name

description (False, any, None) A short description of the repository

private (False, any, False) Either true to create a private repository, or false to create a public one

autoInit (False, any, False) Pass true to create an initial commit with README, .gitignore and LICENSE.

gitignores (False, any, None) Desired language .gitignore templates to apply. Use the name of the templates. For example, 'Go' or 'Go,SublimeText'.

license (False, any, default) Desired LICENSE template to apply. Use the name of the template. For example, 'Apache v2 License' or 'MIT License'.

readme (False, any, None) Desired README template to apply. Use the name of the template.

accessToken (True, any, None) The user Access Token

Examples

```
- name: Create Repository
  gogs_createRepo:
    gogsURL: "http://gogs.local:3000/"
    organization: "acme"
    name: "Test Inventory"
    accessToken: "Token"
```

Return Values

message (success, dict,) object

Status

- This is not guaranteed to have a backwards compatible interface. *[preview]*
- This is maintained by community.

Authors

- Renato Almeida de Oliveira (renato.a.oliveira@pm.me)

2.3.4 irr_prefix – Generater IRR prefix-list

Synopsis

This modules runs bgpq3 to generate model based prefix-list

Requirements

The below requirements are needed on the host that executes this module.

- bgpq3

Parameters

asn32Safe (False, any, False) assume that your device is asn32-safe

IPv (True, any, None) IP protocol version

aggregate (False, any, False) If true aggregate the prefix

asSet (True, any, None)

host(False, any, None) Host running IRRD database

Examples

```
- name: Get prefix-list
  irr_prefix:
    asn32_safe: true
    IPv: 4
    as-set: AS1234
```

Return Values

message (success, dict,) object containing the IRR prefixes

Status

- This is not guaranteed to have a backwards compatible interface. *[preview]*
- This is maintained by community.

Authors

- Renato Almeida de Oliveira (renato.a.oliveira@pm.me)

2.3.5 peeringdb_getasn – Searches for an ASN policy and interfaces

Synopsis

This modules encapsules peeringDB API to search for an specific ASN his interfaces and policy indormations

Parameters

asn (True, any, None) The searched ASN

username (False, any, None) Your peeringDB User

password (False, any, None) Your peeringDB password

ix-id (False, any, None) The peeringDB IXP ID

ix-name (False, any, None) The peerigDB IXP Name

Examples

```
- name: Search ASN 15169
  peeringdb_getasn:
    asn: 15169
    ix-id: 171
```

Return Values

object (success, dict,) object representing ASN data

Status

- This is not guaranteed to have a backwards compatible interface. *[preview]*
- This is maintained by community.

Authors

- Renato Almeida de Oliveira (renato.a.oliveira@pm.me)

2.3.6 peeringdb_prospect – Searches for common IXP

Synopsis

This modules uses peeringDB API to lookup for IXP that dst-ASN has in commoon with src-ASN

Providing username and password allows peeringDB to provide restricted information on the query

Parameters

src-asn (True, any, None) The source ASN you whant to lookup for matches on IXP

dst-asn (True, any, None) The destination ASN you whant to lookup for matches on IXP

username (False, any, None) The peeringdb Username

password (False, any, None) The peeringDB password

Examples

```
- name: Get ASN Data
  peeringdb_prospect:
    dst-asn: 15169
    src-asn: 2906
```

Return Values

object (success, dict,) object representing ASN data

Status

- This is not guaranteed to have a backwards compatible interface. *[preview]*
- This is maintained by community.

Authors

- Renato Almeida de Oliveira (renato.a.oliveira@pm.me)