Labo 4 : Network Automation avec Ansible

VirtuRéseaux 2024
Kevin DOBRETZ

Ce laboratoire est divisé en deux parties :

- Dans la première partie il s'agira de configurer un petit réseau composé de deux routeurs et deux hôtes avec l'outil ifupdown 1.
- La deuxième partie installera un tunnel wireguard entre les deux hôtes et installera un serveur web basé sur nginx disponible uniquement sur l'interface VPN de l'hôte.

Installation

1. Je reprends le script python3:
   python3 ssh_config.py ansible-simple
   J'ai 4 machines, R1,R2,H1,H2
2. Je fais un script de config reprenant les commandes du labo2 pour nommer les hôtes: labo4.sh:

```
#!/bin/sh

#à faire exécuter depuis le code python peut être

#removal of previous labo1 config.
rm ~/.ssh/config.d/openflow-basic
rm ~/.ssh/config.d/switched_config

#pyhton3 script
python3 ssh_config.py ansible-simple

#hostnames changing
echo "*hostnames*"
for m in R1 R2 H1 H2
do
   echo "ssh "$m "hostname $m";
   ssh -q $m "hostname $m";
done
```

Première commande:

ansible -m ping -i "H1,H2,R1,R2" all

résultat:

```
H2 | SUCCESS => {
    "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
R2 | SUCCESS => {
    "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
H1 | SUCCESS => {
    "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
R1 | SUCCESS => {
    "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Expliquez ce que fait cette commande ? utilisez la commande `ansible-doc` pour répondre à la question.

> A trivial test module, this module always returns `pong' on
> successful contact. It does not make sense in playbooks, but
> it is useful from `/usr/bin/ansible' to verify the ability to
> login and that a usable Python is configured. This is NOT ICMP
> ping, this is just a trivial test module that requires Python
> on the remote-node. For Windows targets, use the
> [ansible.windows.win_ping] module instead. For Network
> targets, use the [ansible.netcommon.net_ping] module instead.
> …
> RETURN VALUES:
> - ping
>
> Value provided with the `data' parameter.
> returned: success
> sample: pong
> type: str

Pas d'explication sur le -m et le -i mais quand je les retire, l'aide d'ansible pointe sur les manques:

-m MODULE_NAME, --module-name MODULE_NAME
Name of the action to execute (default=command)

-i INVENTORY, --inventory INVENTORY, --inventory-file INVENTORY
specify inventory host path or comma separated host
list. --inventory-file is deprecated. This argument
may be specified multiple times.

Ainsi nous voyons que dans ansible -m ping -i "H1,H2,R1,R2" all
le -m permet d'enchaîner avec un nom de module, le -i permet de proposer un inventaire de paramètre à la suite de l'appel de module, ici il s'agit des noms des machines. Le "all" à la fin n'est pas une option de ping mais d'ansible, pour dire de faire faire cette commande à tous les hôtes (alt. webservers).

- Donner la commande ad-hoc qui permet d'afficher l'uptime de toutes les machines, sans écrire de fichier inventaire. Démontrez que cette commande est correcte en l'exécutant sur la topologie.

  ansible -i "H1,H2,R1,R2" all -m command –args "uptime"

  trouvé dans la doc:

  -a MODULE_ARGS, --args MODULE_ARGS

  The action's options in space separated k=v format: -a 'opt1=val1 opt2=val2'

  or a json string: -a '{"opt1": "val1", "opt2": "val2"}'

  Output:

```
H1 | CHANGED | rc=0 >>
 02:41:01 up 18 min,  3 users,  load average: 0.02, 0.01, 0.00
R2 | CHANGED | rc=0 >>
 02:41:02 up 18 min,  2 users,  load average: 0.02, 0.01, 0.00
H2 | CHANGED | rc=0 >>
 02:41:02 up 18 min,  2 users,  load average: 0.00, 0.00, 0.00
R1 | CHANGED | rc=0 >>
 02:41:02 up 18 min,  2 users,  load average: 0.02, 0.01, 0.00
```

- Donner la commande ad-hoc qui va créer le fichier `/tmp/hello.txt` sur toutes les machines. Montrez sa sortie par une capture d'écran.

  ansible -i "H1,H2,R1,R2" all -m copy --args "content='Hello, world' dest=/tmp/hello.txt"

sortie:


```
R2 | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
        },
        "changed": false,
        "checksum": "e02aa1b106d5c7c6a98def2b13005d5b84fd8dc8",
        "dest": "/tmp/hello.txt",
        "gid": 0,
        "group": "root",
        "mode": "0644",
        "owner": "root",
        "path": "/tmp/hello.txt",
        "size": 12,
        "state": "file",
        "uid": 0
}
H1 | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
        },
        "changed": false,
        "checksum": "e02aa1b106d5c7c6a98def2b13005d5b84fd8dc8",
        "dest": "/tmp/hello.txt",
        "gid": 0,
        "group": "root",
        "mode": "0644",
        "owner": "root",
        "path": "/tmp/hello.txt",
        "size": 12,
        "state": "file",
        "uid": 0
}
R1 | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
        },
        "changed": false,
        "checksum": "e02aa1b106d5c7c6a98def2b13005d5b84fd8dc8",
        "dest": "/tmp/hello.txt",
        "gid": 0,
        "group": "root",
        "mode": "0644",
        "owner": "root",
        "path": "/tmp/hello.txt",
        "size": 12,
        "state": "file",
        "uid": 0
}
H2 | SUCCESS => {
        "ansible_facts": {
```

```
        "discovered_interpreter_python": "/usr/bin/python3"
        },
        "changed": false,
        "checksum": "e02aa1b106d5c7c6a98def2b13005d5b84fd8dc8",
        "dest": "/tmp/hello.txt",
        "gid": 0,
        "group": "root",
        "mode": "0644",
        "owner": "root",
        "path": "/tmp/hello.txt",
        "size": 12,
        "state": "file",
        "uid": 0
}
```


- Quelle est la différence entre les modules `command`, `shell` et `raw` ? Expliquez et donnez des exemples.

Je regarde la doc:

ansible-doc shell:

The [ansible.builtin.shell] module takes the command name followed by a list of space-delimited arguments. Either a free form command or `cmd' parameter is required, see the examples. It is almost exactly like the [ansible.builtin.command] module but runs the command through a shell (`/bin/sh') on the remote node. For Windows targets, use the [ansible.windows.win_shell] module instead.

ansible-doc command:

The [ansible.builtin.command] module takes the command name followed by a list of space-delimited arguments. The given command will be executed on all selected nodes. The command(s) will not be processed through the shell, so variables like `$HOSTNAME' and operations like `"*"', `"<"', `">"', `"|"', `";"' and `"&"' will not work. Use the [ansible.builtin.shell] module if you need these features. To create `command' tasks that are easier to read than the ones using space-delimited arguments, pass parameters using the `args' task keyword <https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#task> or use `cmd' parameter. Either a free form command or `cmd' parameter is required, see the examples. For Windows targets, use the [ansible.windows.win_command] module instead.

ansible-doc raw

Executes a low-down and dirty SSH command, not going through the module subsystem. This is useful and should only be done in a few cases. A common case is installing `python' on a system without python installed by default. Another is speaking to any devices such as routers that do not have any Python installed. In any other case, using the

[ansible.builtin.shell] or [ansible.builtin.command] module is much more appropriate. Arguments given to `raw' are run directly through the configured remote shell. Standard output, error output and return code are returned when available. There is no change handler support for this module. This module does not require python on the remote system, much like the [ansible.builtin.script] module. This module is also supported for Windows targets. If the command returns non UTF-8 data, it must be encoded to avoid issues. One option is to pipe the output through `base64'.

2 - Routage et adressage

A - Playbook Ansible

Ecrire un playbook qui met en place une configuration persistante des interfaces de H1,H2, R1 et R2 avec les contraintes suivantes :

1. L'adressage de toutes les interfaces sera statique : `iface <ifname> inet static`.
2. Le routage sera statique et seuls H1 et H2 utilisent une route par défaut vers leur routeur respectif. Les routes seront installées avec la directive `post-up` de `/etc/network/interfaces` 1
3. Les hôtes et les routeurs devront être séparés en deux groupes dans votre inventaire Ansible : `routers` et `hosts`.

-> je regarde la documentation:
https://docs.ansible.com/ansible/latest/getting_started/get_started_playbook.html

Je configure un ansible en commandes shell selon cette logique:

```
- name: Configuration de H1
  hosts: H1
  become: true
  tasks:
        - name: Définir les variables pour H1
        set_fact:
        interface_name: eth0
        interface_address: 1.0.0.3
        interface_netmask: 255.255.255.0
        interface_gateway: 1.0.0.1

        - name: Remplir le fichier /etc/network/interfaces.d/eth0
        shell: |
        echo "auto {{ interface_name }}" > /etc/network/interfaces.d/{{ interface_name }}
        echo "iface {{ interface_name }} inet static" >> /etc/network/interfaces.d/{{ interface_name }}
        echo "address {{ interface_address }}" >> /etc/network/interfaces.d/{{ interface_name }}
        echo "netmask {{ interface_netmask }}" >> /etc/network/interfaces.d/{{ interface_name }}
        echo "gateway {{ interface_gateway }}" >> /etc/network/interfaces.d/{{ interface_name }}
        echo "post-up ip link set {{ interface_name }} up" >> /etc/network/interfaces.d/{{ interface_name }}

        register: filleth0

        - name: Afficher la sortie de la commande de remplissage
        debug:
        var: filleth0.stdout_lines
```

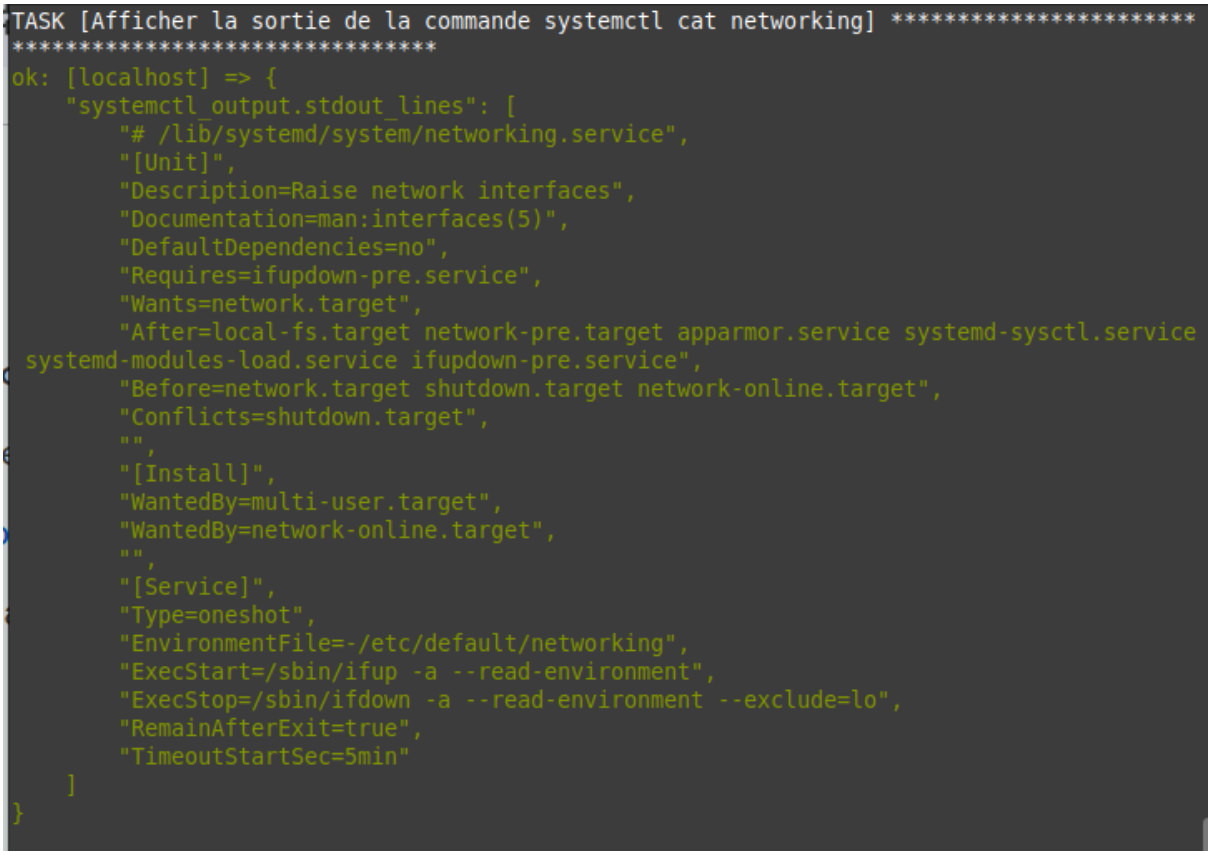
```
- name: Remove old eth0 and restart networking
shell: |
ifup {{ interface_name }}
systemctl restart networking

register: netrestart
ignore_errors: yes

- name: Afficher la sortie de la commande de restart
debug:
var: netrestart.stdout_lines
```

4. Tout changement dans la configuration IP devra redémarrer le service systemd `networking`, qui est un wrapper au dessus d'ifup/ifdown, vous pouvez le vérifier avec la commande :

systemctl cat networking



5. Votre configuration devra être idempotente, autrement dit, si il n'y a pas de changements dans les fichiers de configuration, Ansible ne devra pas les re-écrire, ni redémarrer le service `networking`.

Ah zut, je ferai ça plus tard avec des modules ansible pour pouvoir check les status

6. Votre playbook devra utiliser des variables pour le nom des interfaces, l'adresse IP, le masque et la passerelle par défaut, indiqué dans le champ `vars` de vos `tasks` ansible puis générer le fichier `/etc/network/interfaces.d` sur la base de ces variables à l'aide du module template d'Ansible 2

7. À quoi servent les options `--syntax-check` et `--check` de la commande ansible-playbook ?

je teste:
ansible-playbook --syntax-check labo4.yaml

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost
does not match 'all'
[WARNING]: Could not match supplied host pattern, ignoring: H1
[WARNING]: Could not match supplied host pattern, ignoring: H2
[WARNING]: Could not match supplied host pattern, ignoring: R1
[WARNING]: Could not match supplied host pattern, ignoring: R2
```

et

ansible-playbook --check labo4.yaml

```
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost
does not match 'all'
[WARNING]: Could not match supplied host pattern, ignoring: H1

PLAY [Configuration de H1] *****************************************************************************
skipping: no hosts matched
[WARNING]: Could not match supplied host pattern, ignoring: H2

PLAY [Configuration de H2] *****************************************************************************
skipping: no hosts matched
[WARNING]: Could not match supplied host pattern, ignoring: R1

PLAY [Configuration de R1] *****************************************************************************
skipping: no hosts matched
[WARNING]: Could not match supplied host pattern, ignoring: R2

PLAY [Configuration de R2] *****************************************************************************
skipping: no hosts matched

PLAY [Faire systemctl cat networking sur mon laptop] ***************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [localhost]

TASK [exec systemctl cat networking] *******************************************************************
skipping: [localhost]

TASK [Afficher la sortie de la commande systemctl cat networking] **************************************
ok: [localhost] => {
        "systemctl_output.stdout_lines": []
}

PLAY RECAP *********************************************************************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0
ignored=0
```

en ajoutant l'inventaire, ansible-playbook --syntax-check labo4.yaml -i "H1,H2,R1,R2" je retire les warnings

J'en conclu qu'il s'agit d'un test comme celui d'une compilation.

B - Vérification du bon fonctionnement

A la fin de cette implémentation, H1 doit pouvoir pinger H2, ce que votre playbook devra vérifier dans une tâche finale.

```
TASK [Afficher le résultat du ping] ********************************************************************
**********************
ok: [H1] => {
    "msg": "Ping de H1 à H2 : {'changed': True, 'stdout': 'PING 3.0.0.3 (3.0.0.3) 56(84) bytes of d
ata.\\n\\n--- 3.0.0.3 ping statistics ---\\n1 packets transmitted, 1 received, 0% packet loss, time
 0ms\\nrtt min/avg/max/mdev = 4.571/4.571/4.571/0.000 ms', 'stderr': '', 'rc': 0, 'cmd': 'ping 3.0.
0.3 -q -c 1', 'start': '2024-05-07 13:22:26.072825', 'end': '2024-05-07 13:22:26.082039', 'delta':
'0:00:00.009214', 'msg': '', 'stdout_lines': ['PING 3.0.0.3 (3.0.0.3) 56(84) bytes of data.', '', '
--- 3.0.0.3 ping statistics ---', '1 packets transmitted, 1 received, 0% packet loss, time 0ms', 'r
tt min/avg/max/mdev = 4.571/4.571/4.571/0.000 ms'], 'stderr_lines': [], 'failed': False}"
}
```

A - Playbook Ansible mais pas en shell, avec des modules

Avec l'aide de mes camarades, google et ansible.com
J'ai pu refaire la partie A avec un code à modules, c'était en fait pas très différent une fois que le code shell marchait bien.
fichier *modular_labo4.yaml*

```
TASK [Afficher la sortie de la commande systemctl cat networking] ********************************************************************************************
ok: [localhost] => {
    "systemctl_output.stdout_lines": [
        "# /lib/systemd/system/networking.service",
        "[Unit]",
        "Description=Raise network interfaces",
        "Documentation=man:interfaces(5)",
        "DefaultDependencies=no",
        "Requires=ifupdown-pre.service",
        "Wants=network.target",
        "After=local-fs.target network-pre.target apparmor.service systemd-sysctl.service systemd-modules-load.service ifupdown-pre.service",
        "Before=network.target shutdown.target network-online.target",
        "Conflicts=shutdown.target",
        "",
        "[Install]",
        "WantedBy=multi-user.target",
        "WantedBy=network-online.target",
        "",
        "[Service]",
        "Type=oneshot",
        "EnvironmentFile=-/etc/default/networking",
        "ExecStart=/sbin/ifup -a --read-environment",
        "ExecStop=/sbin/ifdown -a --read-environment --exclude=lo",
        "RemainAfterExit=true",
        "TimeoutStartSec=5min"
    ]
}

PLAY [Vérifier la connectivité de H1 à H2] ***********************************************************************************************************************

TASK [Gathering Facts] *******************************************************************************************************************************************
ok: [H1]

TASK [Ping de H1 à H2] *******************************************************************************************************************************************
ok: [H1]

TASK [Afficher le résultat du ping] ******************************************************************************************************************************
ok: [H1] => {
    "msg": {
        "changed": false,
        "failed": false,
        "ping": "Ping test"
    }
}
```