ANSIBLE – 01/01/2026

```
## ad-hoc commands
ansible all -i rh94-20, -m setup

ansible all -i rh94-20, -m service -a "name=chronyd
state=started enabled=yes"

ansible all -i rh94-20, -m file -a "dest=/tmp/test

# fetch the file from remote m/c to ansible controller

ansible all -i rh94-20, -m fetch -a "src=/etc/hosts
dest=/tmp"

# > multiple lines , | as is (example below)
command:  >                  |   ansible.builtin.copy:
  cp httpd.conf /tmp;        |   content: |
  validate=flkdnfkldn        |     { line1,
                             |       line2 }
```

```
## Misc
become: yes
## we run as l_build, the above means if needed to run a
task then use sudo, add l_build to sudoers list

become-user: 'elastica'
## this means when running the task become the elastica
user so that the files are placed as if elastica user is
doing. E.g. the container file


## Magic vars
var = hostvars[inventory_hostname]
var = play_hosts
var = inventory_hostname
```

```
## include_tasks/ include_role /import_playbook

tasks:
  - name: include tasks example
    ansible.builtin.include_tasks: tasks/query_netbox.yml

  - name: include role example
    ansible.builtin.include_role:
      name: el_paching

- import_playbook: web.yml

## with_first_found

- hosts: example

  pre_tasks:
     - include_vars: "{{ item }}"
       with_first_found:
         - "apache_{{ ansible_os_family }}.yml"
         - "apache_default.yml"
 tasks:
   ..
```

```
## block / when /rescue / always
tasks:
  # Install blah blah
  - block:
      - name: blah blah
        ansible.builtin.command: ls /tmp/missingfile
        when: ansible_os_family == 'RedHat'
        become: yes

  - block:
      - name: blah blah
        script: ls /tmp/missingfile
        when: ansible_os_family == 'RedHat'

  - rescue:
      - name: Run only in case of error in block
        debug: msg="There was an error in the block"

  - always:
       - name: This will always run no matter what"
         debug: msg="This always exectes"
```

```
## variable options
#playbooks/inventory/prod/ny4
[dma-infra]
infra-host1
infra-host2

[dma-prod]
dma-host1

[dma-children]
dma-infra
dma-prod

[dma-children:vars]
k1=v1

[all:vars]
K2=v2
```

```
## Important pb control tricks
ansible.builtin.meta: end_host

ansible.builtin.add_host:
  name: {{ item }}
  group: dynamic_targets
loop: host_lists

ansible_local.ec_tags.app != app ..

## serial || max_fail_percentage
## failed_when || fail_msg

serial: 10 # run pb 10 hosts
max_fail_percentage: 10 # if >10% hosts fail
then whole pb stop running. Ref: Pg – 267
...

ansible_date_time.date <
ansible_local.ec_no_patch.do_not_patch_until
```

```
## vars prompt
vars_prompt:
    - name: loginid
      prompt: "Enter your username"
      private: no
    - name: password
      prompt: "Enter your password"
      private: yes
## will create /tmp/foo in rh94-6 !!
File name: localhost1
[local]
localhost ansible_connection=local

ansible -i localhost1 -m file -a
"path=/tmp/foo state=touch" all
OR File name: localhost2
[local]
rh94-1 ansible_connection=local

ansible -i localhost2 -m file -a
"path=/tmp/foo state=touch" all
# uses local connection instead of ssh
ansible-playbook -i localhost, -c
local redis-playbook.yml
```

```
## run_playbook.yml
- name: Run a playbook "{{ playbook_name }}"
  hosts: all
  gather_facts: true
  become: true
  vars_files:
    - ../ansible-vault.yml

- name: Import "{{playbook_name }}"
  import_playbook: "{{ playbook_name }}.yml"
```

```
## run_one_role.yml
- name: Run on a host "{{ role }}"
  hosts: all
  gather_facts: true
  become: true
  vars_files:
    - ../ansible-vault.yml
  roles:
    - role: "{{ role }}"
```

```
## changed_when / failed_when success_msg || fail_msg

#Ex-1
- name: Check if a file exists
  hosts: localhost
  tasks:
    - name: Run ls on a file that might not exist
      ansible.builtin.command: ls /tmp/missingfile
      register: result
      ignore_errors: yes

    - name: Fail only if "No such file" is *not* in stderr
      ansible.builtin.debug:
        msg: "File does not exist, which is fine."
      failed_when: "'No such file' not in result.stderr"

#Ex-2
- name: Check service status
  hosts: localhost
  tasks:
    - name: Run systemctl status
      ansible.builtin.command: systemctl status sshd
      register: service_status
      changed_when: "'inactive' in service_status.stdout"
```

```
## handlers
- hosts: "{{ hosts }}"

  vars_files:
    - vars/webserver_vars.yml
    #webserver_port: 8080

  tasks:
    - name: install httpd
      ansible.builtin.package:
        name: httpd
        state: present

    - name: configure httpd
      ansible.builtin.template:
        src: templates/custom_httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
      notify:
        - restart httpd

  handlers:
    - name: restart httpd
      ansible.builtin.service:
      name: httpd
      state: restarted  state: started
```

```
## lineinfile
#Ex-1
lineinfile:
   dest: "/etc/php/php.conf"
   regexp: "^opcache.memory"
   line: "opcache.memory = 96"
   state: present
notify: restart apache

#Ex-2
- name: configure proxy
   lineinfile:
     dest: /etc/environment
     regexp: "{{ item.regexp }}"
     line: "{{ item.line }}"
     state: "{{ proxy_state }}"
   with_items:
     - regexp: "^http_proxy="
       line: "http_proxy=http://example-proxy:80/"
     - regexp: "^https_proxy="
       line: "https_proxy=https://example-proxy:443/"
     - regexp: "^ftp_proxy="
       line: "ftp_proxy=http://example-proxy:80/"
```

```
## wait_for

- name: Wait for sshd to come online, blocks subsequent
tasks
  local_action:
  module: wait_for
  host: "{{ inventory_hostname }}"
  port: 22
  delay: 10
  timeout: 300
  state: started

- name: Wait for web server port
  local_action:
  module: wait_for
  host: "{{ inventory_hostname }}"
  port: "{{ webserver_port }}"
  delay: 10
  timeout: 300
  state: started
```

```
## shell >> copy >> lineinfile >> synchronize
- name: Create local temp dir
  local_action: shell mktemp -d | sed -e 's/\/tmp\///'
  register: tmpname

- name: Create remote temp dir
  shell: mktemp -d | sed -e 's/\/tmp\///'
  register: remotetemp

- name: Copy a script to remote m/c
  copy:
    src: timeval.py
    dest: "/tmp/{{ remotetemp.stdout }}/timeval.py"
    mode: 0700
    owner: root
    group: root

- name: Add external values
  lineinfile:
    dest: "/tmp/{{ remotetemp.stdout }}/svc.conf"
    regexp: "contacts"
    insertafter: "contact_groups"
    line: "contacts"

- name: Fill in external recipients
  replace:
  dest: "/tmp/{{ remotetemp.stdout }}/svc.conf"
  regexp: "^(.*contacts.*)$"
  replace: "\1 {{ item }}"
  with_items: alert_recipietns
```

```
- name: Pre-generate templates
  shell: timeval.py  chdir="/tmp/{{ remotetemp.stdout }}/'

- name: Fetch template
  synchronize:  # rsync binary must be present
    mode: pull
    src: /tmp/{{ remotetemp.stdout }}/
    dest: /tmp/{{ tmpname.stdout }}/

- name: Get all file names
  local_action: >
    shell cd /tmp/{{ tmpname.stdout }}/ ;
    /bin/ls svc-*.cfg
  register: all-svc-files

- name: Copy all svc files to remote m/c
  copy:
    src: timeval.py
    dest: "/etc/blah/{{ ansible_hostname }}.{{item}}"
    mode: 0700
    owner: root
    group: root
  with_items: all-svc-files.stdout_lines
  deligate_to: "{{ nagios_host }}"
```

```
## stat
- name: check if bios is uefi or legacy
  stat:
    path: /sys/firmware/efi
    register: bios

- name: copy file
  copy:
    src: …
    dest: …
  when: bios.stat.exists


## register variable.stdout || variable.stderr
- name: Fetch cpu list
  shell: “numactl -H | grep ‘node 0 cpus’ | sed …”
  register: sock0

- lineinfile:
    dest: “/etc/php/php.conf”
    regexp: “^opcache.memory”
    line: “opcache.memory = 96. {{ sock0.stdout }}”
    state: present

## run_once, runs the task per batch, serial: 5
tasks:
  - name: Upgrade database schema
    debug:
      msg: Upgrading database schema..
    run_once: true
```

```
## async/poll
- name: Demonstrate asynchronous tasks
  hosts: frontends
  become: true

  tasks:
    - name: A simulated long running task
      shell: "sleep 20"
      async: 30
      poll: 5

# Ex-2 # To check the status of the task submitted above
 tasks:
   - name: A simulated long running task
     shell: "sleep 20"
     async: 30
     poll: 0
     register: long_task

.. do some other tasks here, and then ..

   - name: Check on the asynchronous task
     async_status:
       jid: "{{ long_task.ansible_job_id }}"
     register: async_result
     until: async_result.finished
     retries: 30

## Ex-3
- name: reboot server
  shell: 'sleep 1 && shutdown -r now "Reboot triggered
by Ansible" && sleep 1'
  async: 1
  poll: 0

- name: wait for server to come back
  local_action:
  module: wait_for
  host: "{{ inventory_hostname }}"
  port: 22
  delay: 10
  timeout: 300
```

```
## assert that success_msg || fail_msg
- name: Example play using assert
  hosts: localhost
  gather_facts: no

  vars:
    app_port: 8080

  tasks:
    - name: Verify that app_port has valid range
      ansible.builtin.assert:
        that:
          - app_port >= 1024
          - app_port <= 65535
        success_msg: "Port {{ app_port }} is valid!"
        fail_msg: "Port {{ app_port }} is invalid!"
```

```
## assert that
- name: Assert service is running on correct host
  hosts: webserver
  tasks:
    - name: Ensure environment is production
      ansible.builtin.assert:
        that:
          - env == "production"
        fail_msg: "Environment is production, not {{ env }}"
        success_msg: "Environment verified: {{ env }}"


ansible-playbook -i myenventory assert1.yml \
-e env=production
```

```
## ansible-galaxy commands
ansible-config dump | grep COLLECTIONS_PATHS

COLLECTIONS_PATHS(default) =
['/home/james/.ansible/collections',
'/usr/share/ansible/collections']

# to list installed collections
ansible-galaxy collection list

# to install collection in a specific path
ansible-galaxy collection install [--force] -p
/usr/share/ansible/collections
davidban77.gns3

# to remove collections
rm -rf /usr/share/ansible/collections/ansible_collections/
davidban77/gns3/
```

```
## Install collection using requirements.yml
---
collections:
- name: davidban77.gns3
  version: '>1.2.0,<1.5.0'
- marmorag.ansodium

# install collections
ansible-galaxy collection install -r requirements.yml
```

Additional Ref Code Block:

```
## Add ip address to a host
---
- name: Configure a static IP on RH9 host
  hosts: all
  become: true

  tasks:
    - name: Add a static Ethernet connection
      community.general.nmcli:
        conn_name: enp0s5_custom
        ifname: enp0s5  # The actual interface
        type: ethernet
        ip4: 10.0.0.64/24
        gw4: 10.0.0.1
        dns4:
          - 75.75.75.75
          - 75.75.76.76
        state: present
        autoconnect: yes

# ansible-playbook -i rh94-tmp, addip.yml

## OS patching recap

1. Place tags: /etc/ansible/facts.d/my_tags.fact

2. Stitch the playbook:
    1. Query netbox (ansible.builtin.uri)
    2. Create host group: dynamic_targets
    3. Create host group: reachable_host
    4. Run role:  el_patching || el_patching_roll

3. Role el_patching:
    1. Preflight: my_tags.fact exists? & validate
       with survey input
    2. Create vmsnapshot
    3. Capture grubby -get-default, yum-deug-dump
    4. Run: dnf update | --bugfix | --security |
       --advisory=RHSA-XXXX
    5. Capture post-patch metadata

4. Role el_patching_rollback:
    1. Preflight: my_tags.fact exists? & validate
       with survey input
    2. Capture pre-rollback metadata
    3. Restore vmsnapshot (for bruteforce)
       OR yum-debug-restore <file-name> (graceful)
    4. Capture post-rollback metadata
```

```
## Bulkload devices into netbox
1.  Create an yml device file new_devices.yaml
    - name: "device-01"
      site: "Site A"
      device_role: "network-edge-router"
      device_type: "C9300-48P"
      platform: "cisco_ios"
      primary_ip: "10.0.0.1/24"
    - name: "device-02"
      site: "Site A"
      device_role: "network-edge-router"
      device_type: "C9300-48P"
      platform: "cisco_ios"
      primary_ip: "10.0.0.2/24"
    # ... more devices

2.  Create playbook
---
- name: Bulk import devices and IP addresses to NetBox
  connection: local
  gather_facts: false
  collections:
    - netbox.netbox

  vars:
    netbox_url: "https://your.netbox.instance"
    netbox_token: "your_api_token"
    device_list: "{{ lookup('file', 'new_devices.yaml') |
from_yaml }}" # Load data from the YAML file

  tasks:
    - name: Create device in NetBox
      nb_device:
        netbox_url: "{{ netbox_url }}"
        netbox_token: "{{ netbox_token }}"
        data:
          name: "{{ item.name }}"
          site: "{{ item.site }}"
          device_role: "{{ item.device_role }}"
          device_type: "{{ item.device_type }}"
          platform: "{{ item.platform }}"
        state: present
      loop: "{{ device_list }}"
      register: new_devices

    - name: Create IP address in NetBox
      nb_ipam_ip_address:
        netbox_url: "{{ netbox_url }}"
        netbox_token: "{{ netbox_token }}"
        data:
          address: "{{ item.item.primary_ip }}"
          assigned_object_type: "dcim.device"
          assigned_object_id: "{{ item.device.id }}"
        state: present
      loop: "{{ new_devices.results }}"
      loop_control:
        loop_var: item
      register: new_ips

    - name: Set primary IP for the device
      nb_device:
        netbox_url: "{{ netbox_url }}"
        netbox_token: "{{ netbox_token }}"
        data:
          name: "{{ item.item.item.name }}"
          primary_ip4: "{{ item.ip_address.id }}" # Use the ID
of the created IP
        state: present
      loop: "{{ new_ips.results }}"
      loop_control:
        loop_var: item

3.  Run the playbook
    ansible-playbook import_devices.yaml
```

```
## Create your own modules
git clone https://github.com/ansible/ansible.git

cd ansible

python -m virtualenv moduledev

vi remote_filecopy.py

#!/usr/bin/env python
…
ANSIBLE_METADATA = {'metadata_version': '1.1',
                    'status': ['preview'],
                    'supported_by': 'community'}

DOCUMENTATION = '''
---
module: remote_filecopy
version_added: "2.15"
short_description: Copy a file on the remote host
description:
- The remote_copy module copies a file on the remote host
from a
given source to a provided destination.
options:
source:
description:
- Path to a file on the source file on the remote host
required: True
dest:
description:
- Path to the destination on the remote host for the
copy
required: True
author:
- Jesse Keating (@omgjlk)
'''

EXAMPLES = '''
# Example from Ansible Playbooks
- name: backup a config file
  remote_copy:
    source: /etc/herp/derp.conf
    dest: /root/herp-derp.conf.bak
'''

RETURN = '''
source:
  description: source file used for the copy
  returned: success
  type: str
  sample: "/path/to/file.name"
dest:
  description: destination of the copy
  returned: success
  type: str
  sample: "/path/to/destination.file"
gid:
  description: group ID of destination target
  returned: success
  type: int
  sample: 502
group:
  description: group name of destination target
  returned: success
  type: str
  sample: "users"
uid:
  description: owner ID of destination target
  returned: success
  type: int
  sample: 502
owner:
  description: owner name of destination target
  returned: success
  type: str
  sample: "fred"
mode:
  description: permissions of the destination target
  returned: success
  type: int
  sample: 0644
size:
  description: size of destination target
  returned: success
  type: int
  sample: 20
state:
  description: state of destination target
  returned: success
  type: str
  sample: "file"
'''
######

import shutil

def main():
    module = AnsibleModule(
        argument_spec = dict(
            source=dict(required=True, type='str'),
            dest=dict(required=True, type='str')
        ),
    )

try:
  shutil.copy(module.params['source'],
              module.params['dest'])
except:
  module.fail_json(msg="Failed to copy file")

module.exit_json(changed=True)

from ansible.module_utils.basic import *

if __name__ == '__main__':
   main()


cp ~/ansible/moduledev/remote_filecopy.py library/


## Module Test playbook.yml

---
- name: Playbook to test custom module
  hosts: all

tasks:
- name: Test the custom module
  remote_filecopy:
    source: /tmp/foo
    dest: /tmp/bar
  register: testresult

- name: Print the test result data
  ansible.builtin.debug:
    var: testresult
```

netbox protocols
ansible-linux/**requirements.yml:**

```
collections:
- name: community.general
  version: "9.1.0"
  source: https://galaxy.ansible.com

- name: ansible.posix
  version: "1.5.4"
  source: https://galaxy.ansible.com

- name: netbox.netbox
  version: "3.19.1"
  source: https://galaxy.ansible.com

- name: fedora.linux_system_roles
```

ansible-linux/**requirements.txt:**

```
pynetbox==7.3.4
pytz==2021.3
requests==2.31.0
```

ansible-linux/**execution-environment.yml:**

```
---
version: 3
dependencies:
  galaxy: requirements.yml
  python: requirements.txt
```

ansible-linux/**ansible.cfg:**

```
[defaults]
inventory =
./inventory/netbox_inventory.yml
roles_path = ./roles
host_key_checking = false
retry_files_enable = false
forks = 10
remote_tmp = /tmp
scp_if_ssh = True

# format errors in more readable yml
format
stdout_callback = yaml
bin_ansible_callbacks = True

callbacks _enabled =
ansible.posix.profile_tasks

[ssh_connection]
pipelining = True
control_path = /tmp/ansible-ssh-%%h-
%%p-%%r
ssh_args = -o ControlMaster=auto -o
ControlPersist=3600s

[inventory]
enable_plugins = yaml, script, netbox
```

ansible-linux/inventory/**netbox_inventory.yml:**

```
plugin: netbox.netbox.nb_inventory
api_endpoint:
https://netbox.example.com
token: "{{ lookup('env',
'NETBOX_TOKEN') }}"
validate_certs: false
config_context: false
compose:
   ansible_network_os: platform.slug
flatten_custom_fields: true

keyed_groups:
  - key: device_role
    prefix: "role_"
    separator: ""
    regex: "[-]"
    replace: "_"
  - key: platform
    prefix: "platform_"
    separator: ""
    regex: "[-]"
    replace: "_"
  - key: sites
    prefix: "sites_"
    separator: ""
    regex: "[-]"
    replace: "_"
  - key: os_flavor
    prefix: "os"
    separator: "_"
device_query_filters:
  - has_primary_ip: "true"
  - platform_id: 1
```

How to run a playbook

```
ANSIBLE_INVENTORY_ENABLED=host_list \
ansible-playbook -i <server_name>, \
playbooks/base_os_pb.yml \
--ask-pass -ask-become-pass \
--ask-vault-pass \
-e "sites=ny4 ansible_user=l_build
ctfy_zone=blah reboot=true"
```

##Tower

Tower >> Inventories

Name: NetBox_Inventory
Variables:

```
{
"centrify_zone": "ECM",
"ec_app_domain":"app.example.com",
"ec_domain": "ad.example.com",
"ec_ny_dc":
"vcenterny.exa,mple.com",
"snmp_v3_auth_protocol": "SHA",
"snmp_v3_priv_protocol": "AES:
}
```

NetBox_Inventory >> Sources

Name: Netbox Source
Execution environment: netbox_ee
Project: Linux Base OS Build
Inventory file:
inventory/netbox_inventory.yml

Enabled Options:
Overwrite local groups and hosts
from remote inventory source
Overwrite local variable from
remote inventory source
Update on launch

## ansible-pull - pros Instead of a central control server pushing configurations to managed nodes, each managed node runs a local script (often as a cron job) that pulls configuration playbooks from a central source repository (like Git) and executes them locally. Typically automated by setting up a cron job on each managed node to run ansible-pull periodically (e.g., every hour) to check for and apply updates.	## ansible-pull - cons Each target node acts as its own central server. This requires provisioning SSH keys or credentials to every machine to allow them to pull configurations from a git repository. It makes managing sensitive data via Ansible Vault harder to deploy securely. Pull mode is harder to monitor. It can be challenging to determine which nodes are successfully running the configuration, or to report failures centrally, requiring additional tooling (ansible-pull usually runs on a schedule (e.g., via cron), meaning there is a time gap between a change being committed to the repo and all nodes applying it, which is not ideal for emergency hotfixes While still generally "agentless," pull mode requires Ansible and Git to be installed on every target node, adding to the local footprint. If many nodes pull from the same git repository simultaneously, it can put significant load on that server. Debugging is often harder

Points to remember:

1. **assert | lineinfile | local_action | file | copy | fetch**
2. **regexp| insert_after | insert_before**
3. **changed_when | failed_when | success_msg | fail_msg**
4. **wait_for**
5. **async – poll**
6. **ansible_os_family**
7. **handlers**
8. **ansible.builtin.uri (used in solace SEMP v2)**
9. **ansible_local.ec_tags.app (reading /etc/ansible/facts.d/ec_tags.fact file with “app”:”value”)**
10. **ansible_date_time.date**
11. **ansible_builtin.add_host | ansible_builtin_meta: end_host**
12. **ansible_connection = local**

Patching playbook points:

1. **tagging each host with /etc/ansible/facts.d/ec_tags.fact**
2. **query_netbox > create dynamic_targets > crate reachable_hosts**
3. **run patching playbook for each reachable host with serial: 10 and max_fail_percentage: 10**
4. **patching preflight > prepatch metadata > create snapshot > patch**
5. **dnf update || dnf update --bugfix || dnf update --securityfix || dnf update --advisory=RHSA-3456,..**
6. **post patch metadata**