Ansible

Session 9 - 28th Jan 2023 Summary

➢ To understand what is happening when we run the playbook, we can run it with the verbose (-v) option.

```
[root@ip-172-31-5-229 code]# ansible-playbook  -v a.yml
Using /etc/ansible/ansible.cfg as config file

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

TASK [Gathering Facts] *********************************************************
ok: [3.7.253.131]

TASK [command] *****************************************************************
changed: [3.7.253.131] => {"changed": true, "cmd": ["date"], "delta": "0:00:00.0
03289", "end": "2023-01-28 08:57:54.156228", "msg": "", "rc": 0, "start": "2023-
01-28 08:57:54.152939", "stderr": "", "stderr_lines": [], "stdout": "Sat Jan 28
08:57:54 AM UTC 2023", "stdout_lines": ["Sat Jan 28 08:57:54 AM UTC 2023"]}

TASK [package] *****************************************************************
ok: [3.7.253.131] => {"changed": false, "msg": "Nothing to do", "rc": 0, "result
s": []}

PLAY RECAP *********************************************************************
3.7.253.131                : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

➢ When we want to store output of some module then a register keyword will be used. It stores output in a variable then we can use that variable anywhere in the playbook.

root@ip-172-31-5-229:/code

```
- hosts: all
  tasks:
  - command: date

  - package: "name=vim state=present"
    register: x


  - debug:
      var: x
~
```

➢ This is the output of the above “x” variable which store output of package module and we can print using the debug module.

```
TASK [debug] *******************************************************************
ok: [3.7.253.131] => {
    "x": {
        "changed": false,
        "failed": false,
        "msg": "Nothing to do",
        "rc": 0,
        "results": []
    }
}
```

➢ Above output returns a dictionary with key value pairs. So we can use that key value pairs for some conditions.

➢ If the value of the failed key appears to be false then we can say the task runs successfully. Then we can use this for conditions.

```
root@ip-172-31-5-229:/code
- hosts: all
  tasks:
  - command: date

  - package: "name=vim state=present"
    register: x


  - debug:
      var: x

  - debug:
       msg: "my vim software installed ssss.."
    when:  not x.failed
~
```

➢ There are multiple ways for handling exceptions. One way is ignore_errors. If an error occurs in a task then ignore it and run next tasks.

```
root@ip-172-31-5-229:/code
- hosts: all
  tasks:
  - command: date

  - package: "name=vimal state=present"
    register: x
    ignore_errors: true


  - debug:
      var: x
```

```
root@ip-172-31-5-229:/code

TASK [command] *****************************************************************
changed: [3.7.253.131]

TASK [package] *****************************************************************
fatal: [3.7.253.131]: FAILED! => {"changed": false, "failures": ["No package vim
al available."], "msg": "Failed to install some of the specified packages", "rc"
: 1, "results": []}
...ignoring

TASK [debug] *******************************************************************
ok: [3.7.253.131] => {
    "x": {
        "changed": false,
        "failed": true,
        "failures": [
            "No package vimal available."
        ],
        "msg": "Failed to install some of the specified packages",
        "rc": 1,
        "results": []
    }
}
```

➢ Another way for handling exceptions is block and rescue. Write some tasks in block and if something fails in block then and then only rescue code will run. There is one more keyword ie. always. Doesn't matter if it is a block code run or rescue code run, always code always runs.

```
root@ip-172-31-5-229:/code
- hosts: all
  tasks:
  - name: i m block
    block:
    - package:
        name: "httpd123"
        state: present

    - copy:
        content: "hi Vimal.."
        dest: "/var/www/html/index.html"

    - service:
        name: "httpd"
        state: started

    rescue:
     - debug:
        msg: "actual code not working..."

    always:
    - debug:
        msg: "i m always run...."
```

➢ Idempotent nature does not work for all modules in ansible for example command module.

➢ When we use the command module then they always show true for the changed key. Use the changed_when keyword to control how a task reports that it has changed something on the managed host. Means if we want to make false for changed key then use “changed_when: false”.

```
root@ip-172-31-5-229:/code
- hosts: all
  tasks:
  - package: "name=httpd state=present"

  - command: date
    changed_when: false
~
```

➢ We can use the failed_when keyword on a task to specify which conditions indicate that the task has failed.

```
root@ip-172-31-5-229:/code
- hosts: all
  tasks:
  - command: date

  - package: "name=vim state=present"
    register: x
    failed_when: true
    ignore_errors: true


  - debug:
      var: x

  - debug:
      msg: "my vim software installed ssss.."
    when:  not x.failed
~
```

➢ Ansible might need access to sensitive data, such as passwords or keys, to configure managed hosts. Normally, this information is stored as plain text in inventory variables or other Ansible files.

➢ In that case, however, any user with access to the Ansible files, would have access to this sensitive data. Ansible Vault can be used to encrypt and decrypt any data file used by Ansible.

➢ To create a new encrypted file : Command prompts for the new Vault password and then opens a file using the editor then we can type our data.

```
[root@ip-172-31-5-229 code]# ansible-vault  create myvar.yml
New Vault password:
Confirm New Vault password:
```

➢ After creating an encrypted file if we try to see data then it will show encrypted data which is encrypted using AES256 algorithm.

```
[root@ip-172-31-5-229 code]# cat myvar.yml
$ANSIBLE_VAULT;1.1;AES256
353737643665316139316638383036316465666665616161383366623734303837643231643938666
626666316630343230373733363035633133643536333931 0a303064316636633565643137393237
333763333766376431303139323732346239333363636563306633653435646135643733666466396
6131313365383631310a353664326266303161653731653034363733633562616534616362643438
3037
[root@ip-172-31-5-229 code]#
```

➢ If we try to use that encrypted data in the playbook then it will ask for a vault password.

```
[root@ip-172-31-5-229 code]# ansible-playbook  m.yml --ask-vault-password
Vault password:

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

TASK [Gathering Facts] *********************************************************
ok: [3.7.253.131]

TASK [package] *****************************************************************
ok: [3.7.253.131]

PLAY RECAP *********************************************************************
3.7.253.131                : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

➢ We can also create a file for storing a vault password and provide that file to ansible-playbook command.

```
[root@ip-172-31-5-229 code]# vim /etc/mysecret
[root@ip-172-31-5-229 code]# cat /etc/mysecret
redhat@123
[root@ip-172-31-5-229 code]# ansible-playbook  m.yml --vault-password-file=/etc/
mysecret

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

TASK [Gathering Facts] *********************************************************
```

➢ Ansible provides multiple pre-create roles. For this install the rhel-system-roles package.

```
[root@ip-172-31-5-229 ~]#
[root@ip-172-31-5-229 ~]# yum  install rhel*roles*
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscripti
on-manager to register.

Last metadata expiration check: 0:09:16 ago on Sat 28 Jan 2023 09:57:09 AM UTC.
Dependencies resolved.
================================================================================
 Package            Arch    Version           Repository                   Size
================================================================================
Installing:
 rhel-system-roles  noarch  1.20.1-1.el9_1    rhel-9-appstream-rhui-rpms  2.0 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 2.0 M
Installed size: 9.7 M
Is this ok [y/N]:
```

➢ To see the all roles :

```
[root@ip-172-31-5-229 ~]# ansible-galaxy  list
```

➢ For configuring a ntp client they provide a pre-created role we have just provide their value for example ntp server.

```
- hosts: all
  vars:
    timesync_ntp_servers:
      - hostname: 0.in.pool.ntp.org
        iburst: yes
  roles:
    - rhel-system-roles.timesync
~
```

```
[root@ip-172-31-5-229 code]# ansible-playbook  n.yml

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

TASK [Gathering Facts] *********************************************************
ok: [3.7.253.131]

TASK [rhel-system-roles.timesync : Set version specific variables] *************
included: /usr/share/ansible/roles/rhel-system-roles.timesync/tasks/set_vars.yml
 for 3.7.253.131

TASK [rhel-system-roles.timesync : Ensure ansible_facts used by role] **********
ok: [3.7.253.131]

TASK [rhel-system-roles.timesync : Set platform/version specific variables] ****
ok: [3.7.253.131]

TASK [rhel-system-roles.timesync : Populate service facts] *********************
```

➢ lineinfile module ensures a particular line is in a file, or replace an existing line using a back-referenced regular expression.