→ Chapter:08 Simplifying playbooks with roles

Describing roles structure

- An ansible role is defined by a standardized structure of subdirectories and files.
- → Roles group content
- → Roles make larger projects more manageable
- → Roles can be developed in parallel by different admins.

Structure of role ⇒ $ tree user_role.example.

```
user_role.example
   └→ defaults
          └→ main.yml
   └→ files
   └→ Handlers
          └→ main.yml
   └→ meta
          └→ main.yml
   └→ README.md
   └→ tasks
          └→ main.yml
   └→ templates
   └→ tests
          └→ inventory
          └→ test.yml
   └→ vars
          └→ main.yml
```

1> defaults

main.yml => default values of role variable
overwritten by vars in playbook.

2> files

Static files that are referenced by role tasks.

3> handlers

main.yml => handler definitions

4> meta

main.yml => info about role ; includes author,
license, platforms & optional role depend.

5> tasks

main.yml => task definitions.

6> templates

Contains Jinja2 templates

7> tests

Contains inventory & test.yml playbook that
can be used to test the role.

8> vars

main.yml => role's variable values.
not overwritten by vars in playbook

ø using ansible roles in a playbook.

```
---
- hosts: remote.example.com
  roles:
    - role1
    - role2
```

In such way all the info & content inside of role any dependencies will be imported into the playbook.

When using roles in a play; the roles will run first, before any tasks that is defined in the play.

```
---
- hosts: remote.example.com
  roles:
    - role: role1
    - role: role2        OR   - {role: role1, var1: var1,
        var1 : var1                  var2: var2}
        var2 : var2
```

any vaiable in 'default' or 'vars' are now replaced by 'var1: var1' & 'var2 : var2'.

ø Order of execution.

tasks of role1 => then => tasks of role 2 => then Normal tasks.

Roles handler => then => Normal play's handler

```
- name: illustration of order of execution
  hosts: remote.example.com
  pre_tasks:
        - task 1
            :
        - task 2
            :
        - task 3
            :
          notify: myhandler.
```

these will execute before execution of roles. (independent)

```
  roles:
      - role 1
```

execution of role

```
  tasks:
      - task 1
        notify: my handler.
```

execution of tasks after execution of role.

```
  post_tasks:
        - task 1
            :
        - task 2
          notify: my hander.
  handles:
        - name: myhandler.
              :
```

execution of tasks after execution of role (independendent)

After handler is notified by 'pre_tasks' & 'tasks' & 'post_tasks' therefore handler is run 3 times after execution of each of these tasks.

Roles can be added in playbook also as a task
- include_role: 'role 1' & - import_role: 'role 1'

Reusing content with system roles

rhel-system-roles package:

RedHat subscription is registered on C.N
$ yum install rhenl-system-roles
→ roles are located in /usr/share/ansible/roles dir
→ This is the default path ; which could by overridden by mentioning 'roles-path' in ansible.cfg file or by changing environmental variable ANSIBLE_ROLES_PATH

/usr/share/doc/rhel-system-roles-<version>/ directory
⇒ Documentation for the RHEL system Roles.

Each documentation directory have README.md file which contains a description of the role.

for timesync example;
two major attributes are 'hostname' & 'iburst'

```
- name : timesync
  hosts : servers

  vars :
     timesync_ntp_servers :
       - hostname : xyz
         iburst   : ~~pqrs~~ yes/NO

  roles :
       - rhel-system-roles.timesync .
```

ɸ Selinux role example

this can do;

1) set enforcing or permissive mode.
2) Run restorecon on ports
3) set SELinux Boolean values.
4) set SELinux file context persistent
5) SELinux user mapping

```
vars:
  selinux_state: enforcing.
```

Set SElinux stat.

~~vars:~~

```
  selinux_booleans:
    - name: 'httpd_enable_homedirs'
      state: on
      persistent: yes.
```

Set SElinux boolean.

```
  selinux_fcontexts:
    - target: '/srv/www(/.*)?'
      setype: 'httpd_sys_content_t'
      state : present.
```

Set SElinux Context persistent.

```
  selinux_restore_dirs:
    - /srv/www
```

Works as restorecon

```
  selinux_ports :
    - ports : '82'
      setype : 'httpd_port_t'
      proto : 'tcp'
      state : 'present'
```

Set ports on selinux type.

timedatectl ⇒ Check current clock settings

tzselect ⇒ look up other valid values to set diffⁿ time zone.

Creating Roles.

- Create the role directory structure
- Define the role content
- Use the role in a playbook

By default, ansible looks for roles in a subdirectory called 'roles' in the directory containing your ansible playbook.

If not found above, ansible looks for directory mentioned in ansible.cfg under 'roles_path'

By default : /usr/share/ansible/roles
If-a . /etc/ansible/roles.

In role directory structure; if a subdirectory exists but is empty, then it is ignored.

```
$ cd roles
$ ansible-galaxy init myfirstrole
```
— (Creates Role Skeleton)

In roles directory 'tasks' can have 'template' module & template module can have variables mentioned in 'defaults'

in meta/main.yml file dependencies are mentioned.

```
---
dependencies:
    - role: role2
      port: 8080
```

While playbook execution prefix of role name can be seen while execution.

Variable precedence:

default directory:
Could be override by almost all

vars directory:
Could not be override by inventory variables & vars from playbook.

variables mentioned below roles:
Highest precedence.

Deploying roles with ansible galaxy.

- Ansible galaxy is a public library of ansible content written by a variety of ansible administrators and users.

Searching roles from the comand line.

```
$ ansible-galaxy search 'keyword'
```

--author, --platforms, --galaxy-tags to narrow down the search results.

```
$ ansible-galaxy info 'role_name'
```

→ info about the role.

```
$ ansible-galaxy install role-name
```

→ downlads the role from ansible-galaxy & install it on local machine.

It gets downladed in the directory mentioned in ansible.cfg roles-path section.
Otherwise in users ~/.ansible/roles directory
OR in the dir ANSIBLE_ROLES_PATH env is mentioned.

```
$ ansible-galaxy install role_name

$ ansible-galaxy install role_name -p directory.
```

$ Installing roles using a requirement file.
if a playbook requires specific roles to be present. you can download them by:

```
$ vim roles/requirements.yml.
    - src: role-name
      version: xyz

$ ansible-galaxy install -r roles/requirements.yml
    -p roles.
```

'-src' can have
- ansible-galaxy role name
- Git-based repo url by https
- Git-based repo url by ssh

If role is hosted on source control repository.
- Attribute 'scm' for git
- Attribute 'hg' for mercurial-based software repo.

'name' keyword is used to override the local name of the role.
'version' keyword => specifies version to be downloaded.

```
$ ansible-galaxy list
```

→ lists the roles that are found locally.

```
$ ansible-galaxy remove role-name
```

→ Removes role locally.