

Principes des Systèmes d'exploitation

Roles

Guillaume Duvillé

1. Roles

2. Ansible galaxy

Roles

What are roles? (1)

- Idiomatic way to split playbooks into different files,
- Allows to reuse tasks,
- Offers a lot of flexibility:
 - Can include specific variables,
 - Can define handlers,
 - Can define specific modules,
 - Offers a per task file management,
 - ...
- Can be easily shared.

What are roles? (2)

- Simply a convention based formatted file tree:

- defaults: default variables for the role (with the lowest priority),
- files: files that the role deploys,
- handlers: handlers of the role,
- library: custom modules,
- meta: roles metadata,
- tasks: main list of tasks executed by the role,
- templates: templates of the role,
- vars: other variables.

```
ansible_root
|-- roles
|   |-- role_name
|       |-- defaults
|           |-- main.yml
|       |-- files
|           |-- main.yml
|       |-- handlers
|           |-- main.yml
|       |-- library
|           |-- custom_module.py
|       |-- meta
|           |-- main.yml
|       |-- tasks
|           |-- main.yml
|       |-- templates
|           |-- main.yml
|       |-- vars
|           |-- main.yml
|-- ansible.cfg
-- hosts
```

Storing roles

- **ansible** looks for roles in the following locations:
 - ① `roles/` directory (in directory containing the playbook file),
 - ② `/etc/ansible/roles/`
- Fully qualified paths may be used in playbook:

```
1 ---
2 - hosts: all
3   roles:
4     role: '/home/users/git_repos/ansible_dir/role'
```

Three different ways to use roles:

- ① play level with **roles:** option,
- ② task level in static mode with **import_role:** option,
- ③ task level in dynamic mode with **include_role:** option.

Play level (1)

When file is parsed:

- ① Add `foo/tasks/main.yml` to the play,
- ② Add `foo/handlers/main.yml` to the play,
- ③ Add `foo/vars/main.yml` variables to the play,
- ④ Add `foo/defaults/main.yml` variables to the play,
- ⑤ Add `foo/meta/main.yml` role dependencies to the play.
- ⑥ Do the same with `bar/tasks/main.yml`, and so on...

```
1 ---
2 - hosts: all
3   roles:
4     - foo
5     - bar
```

Templates and files of the role can be referenced without giving path.

- At this level, roles are statically processed (processed when playbook is read).

Play level (2)

- Options can be passed to the role,
- Including:
 - Variables definitions (or override),
 - Tags definition,
 - ...

```
1  ---
2  - hosts: all
3    roles:
4      - foo
5      - role: bar
6        vars:
7          variable1: value1
8          variable2: value2
9        tags:
10       - list
11       - of
12       - tags
```

Task level: static mode

```
1  ---
2  - hosts: all
3    tasks:
4      - name: Import role foo
5        import_role:
6          name: foo
7      - name: Import role bar
8        import_role:
9          name: bar
10     vars:
11       variable1: value1
12       variable2: value2
13     tags: list, of, tags
```

Works the same way as play level.

Task level: dynamic mode

```
1  ---
2  - hosts: all
3    tasks:
4      - name: Import role foo
5        import_role:
6          name: foo
7      - name: Import role bar
8        import_role:
9          name: bar
10     vars:
11       variable1: value1
12       variable2: value2
13     tags: list, of, tags
```

Roles are included as they are encountered (at execution time),

What is the difference?

Static:

- Roles are included at processing time,
- If pre tasks are defined in roles, they are executed before any other tasks of the playbooks,
- Tasks of the role appear in the `--list-tasks` report,
- Can lead to errors when using loops.

Dynamic:

- Roles are included at execution time (when they are hit),
- Execution is sequential: if role contains pre tasks, those will be executed **after** previously defined tasks,
- The include is considered as a task:
 - will appear in the `--list-tasks` report,
 - but tasks of the role won't!
- Tags will apply on the include task,
- So does conditions: use of **when:** in playbook won't apply on the tasks of the role but on the inclusion.

Multiple runs of roles

- `foo` is executed only once,
- unless called with different parameters
- Or **`allow_duplicates: true`** is defined in `roles/foo/meta/main.yml`.

```
1  ---
2  - hosts: all
3    roles:
4      - foo
5      - bar
6      - foo
```

Dependencies

- roles/foo/meta/main.yml can define role dependencies,
- Use of **dependencies:** option.
- If roles/foo/meta/main.yml contains:

```
1  ---
2  dependencies:
3    - role: bar
4      vars:
5        variable1: value1
6  allow_duplicates: true
```

- bar role will be run with given options.

Ansible galaxy

What is it?

- Galaxy is a hub for **ansible** content (especially roles but not only),
- It contains list of user created content,
- Content is not stored on Galaxy but on github or gitlab.
- Search of content can be done on <https://galaxy.ansible.com/search>

Usage

- Roles can be install with **ansible-galaxy** command.
- To install **glances** installation role located at <https://galaxy.ansible.com/vincentclee/glances>:

```
1 $ ansible-galaxy install vincentclee.glances
```

- Default install dir is `/etc/ansible/roles` (requires admin privileges),
- Can be overridden with **--role-path** (short **-p**) option.