neueda

futureproof your workforce

MODULE 4: HANDLERS, ROLES AND CODE REUSE

NEUEDA.COM

Poll Question

How much code-reuse is evident in your work environment?

A. I can’t see any
B. There’s a bit
C. It’s a real company policy
D. I don’t know

neueda
futureproof your workforce

Module Overview

- Creating Re-usable Code
- Tags and Handlers
- Include and Import
- Ansible Roles

CREATING RE-USABLE CODE

Creating Re-Usable Code

- In this module we will look at three levels of code re-usability that build on each other
  - **Tags** and **Handlers** make your playbooks more flexible and widely usable
  - **Includes** and **Imports** allow you to directly bring code from other playbooks into your current playbook
  - **Roles** wrap up everything into flexible structure that can be reused cleanly

TAGS AND HANDLERS

Ansible Tags

- Tags are used at resource level to attach labels to a specific resource
- The `ansible-playbook` command has two tag-related options
  - The `--tags` option will select resources based on a list of tag values
  - The `--skip-tags` option will specifically exclude resources with the specified tags
- There are two reserved tag names
  - An `always` tag will always run that play or task unless you explicitly specify `--skip-tags always`
  - A `never` tag will never run that play or task unless you explicitly specify `--tags never`
- Tags can be added to tasks, blocks of tasks, plays, includes, imports and roles.

Ansible Tags Example:

- By using a combination of the --tags and --skip-tags arguments we can select subsets of the tasks

```yaml
---
- name: Ansible Tags
  hosts: localhost
  gather_facts: false
  tasks:
    - debug:
        msg: "This is the gold medal"
      tags:
        - first
        - always
    - debug:
        msg: "This is the silver medal"
      tags: second
    - debug:
        msg: "This is the bronze medal"
      tags:
        - third
        - always
    - debug:
        msg: "Thanks for coming"
      tags:
      - fourth
      - never
```

Handlers

- Some tasks should only be run when a certain change is made
- E.g., if webserver configuration is updated, restart httpd, but not otherwise.
- This is achieved using handlers
  - Handlers run only when notified
  - Handlers must have a globally unique name

Example: Apache Server

- This example will install the latest version of the Apache web server
- But the handler will only be notified if a change is made

```
---
- name: Handlers Example
  hosts: web
  gather_facts: false
  tasks:
    - name: Install httpd latest version
      yum:
        name: httpd
        state: latest
      become: true
      notify: restart_httpd
  handlers:
    - name: restart_httpd
      become: true
      service:
        name: httpd
        state: started
```

Example: flush_handlers

- By default, handlers are run at the end of a play.
- Despite being called 3 times, the `run_handler` here will only be run once
- The `flush_handlers` task will force the handlers to run each time

```yaml
---
- name: Handlers Example
  hosts: db
  gather_facts: false
  become: true
  tasks:
    - name: Sleep for 2 seconds
      command: sleep 2
      notify: run_handler

    # - name: Flush handlers
    #   meta: flush_handlers

    - name: Sleep for 5 seconds
      command: sleep 5
      notify: run_handler

    # - name: Flush handlers
    #   meta: flush_handlers

    - name: Sleep for 7 seconds
      command: sleep 7
      notify: run_handler

  handlers:
    - name: run_handler
      debug:
         msg: "Today's date and time: {{ '%d-%m-%Y %H:%M:%S:%s' | strftime }}"
```

INCLUDE AND IMPORT

12

Bringing tasks in from external files

- Two options for splicing the tasks of another file into the current one:
  - `include_tasks`: In this case, tasks are added dynamically
    - Included tasks can be affected by the results of earlier tasks
    - Similar to handlers - they may or may not run depending on the results of other tasks
  - `import_tasks`: tasks are added statically
    - Imported files are preprocessed before any tasks are run
    - Imported content is never affected by other tasks in the top level playbook
- The included/imported files can be only a list of tasks

Demo: include_tasks

Demo: import_tasks

Importing a Playbook

- Ansible has an `import_playbook` module, which statically incorporates a playbook
- It occurs at the top level of a playbook, as a list with the other plays
- This can be used to collect related playbooks and run them in order

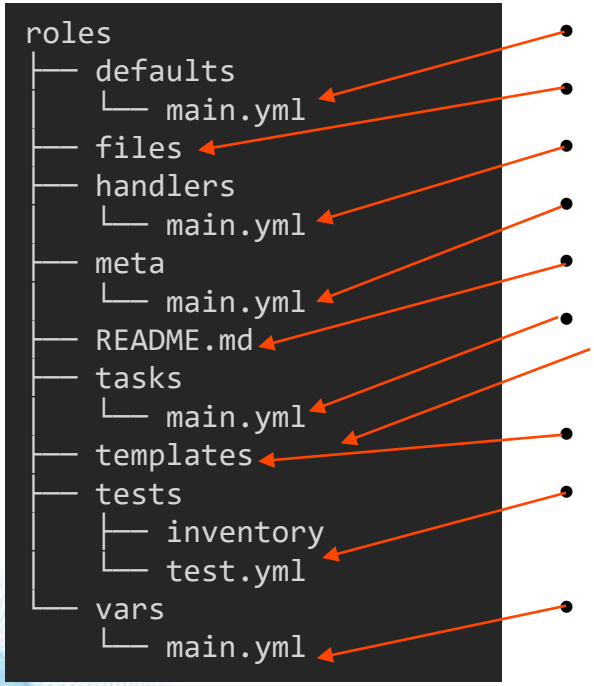
```
---
- import_playbook: check.yml
- hosts: localhost
  tasks:
    - ansible.builtin.debug:
        msg: task1
```

ANSIBLE ROLES

Ansible Roles

- Ansible **Roles** are the next level of re-use
- A Role is composed of a standardized file structure with multiple folders
- This allows Ansible playbooks to automatically load from separate YAML files:
  - Variables
  - Tasks
  - Handlers
  - Templates
  - Default values

Role Folder Structure



- A configuration file to define variable defaults
- All the extra files required by the role tasks
- Handlers for service management
- Role metadata like author info, licenses, dependencies
- Good citizenship to have a clear README.md!
- These are the tasks that will be executed when your role runs - **THIS IS THE MAIN BIT**.
- Jinja2 template engine scripts to create configuration files
- Test environment with its own inventory file and playbook script
- Place to store variables that do no require changes between environments

Ansible Roles Locations

- The first place Ansible looks for a role is in the local `./roles` directory
- Next is in the `.ansible` directory of the users home directory: `~/.ansible/roles`
- Finally `/etc/ansible/roles`
- You can also call a role with a fully qualified path:

```
---
- hosts: webservers
  roles:
    - role:
'/path/to/my/roles/common'
```

Using Roles

- You can use roles in three ways:
  - At the play level with the `roles` option (this is the classic way to use roles).
  - At the tasks level with `include_role`.
    - You can reuse roles dynamically anywhere in the tasks section of a play using `include_role`.
  - At the tasks level with `import_role`.
    - You can reuse roles statically anywhere in the tasks section of a play using `import_role`.

Using Roles at the Play Level

- For a the common role (say), in the `roles/common/` directory:

If exists…	Ansible adds…
`tasks/main.yml`	Any tasks to the play
`handlers/main.yml`	Any handlers to the play
`vars/main.yml`	Any variables to the play
`defaults/main.yml`	Any variables to the play
`meta/main.yml`	Any role dependencies to the list of roles

```
---
- hosts: webservers
  roles:
    - common
    - webservers
```

Creating your first role

- The ansible-galaxy command can initialize a role directory structure with all the right files in place:

```
mkdir roles
cd roles
ansible-galaxy role init my-test-role
```

- To actually create a role you only need to have some meaningful tasks in the tasks/main.yml file.
- Filling in the other files will make your role more flexible and useful.

LAB-3

- Go through the instructions in Lab 3.

REVIEW

25

neueda
future-proof your workforce

Module Review

In this module you learned about:

- ✓ Creating Re-usable Code
- ✓ Tags and Handlers
- ✓ Include and Import
- ✓ Ansible Roles

Next we will do a short quiz

- Knowledge Check

More Info

- Re-using Ansible artifacts — Ansible Documentation

KNOWLEDGE CHECK

neueda
future-proof your workforce

Knowledge check question 1

What is the meta module argument to ensure handlers run at a specific point?

Choice	Response
A	run_handlers
B	force_handlers
C	flush_handlers

Knowledge check question 1

What is the meta module argument to ensure handlers run at a specific point?

Choice	Response
A	run_handlers
B	force_handlers
C	flush_handlers

Knowledge check question 2

Which module would you use if you wanted to add a tag to a list of tasks in a file?

Choice	Response
A	import_tasks
B	include_tasks

Knowledge check question 2

Which module would you use if you wanted to add a tag to a list of tasks in a file?

Choice	Response
A	import_tasks
B	include_tasks

Knowledge check question 3

Where are the role dependencies listed in an Ansible Role?

Choice	Response
A	meta/main.yml
B	defaults/main.yml
C	variables/main.yml
D	tasks/main.yml

Knowledge check question 3

Where are the role dependencies listed in an Ansible Role?

Choice	Response
A	meta/main.yml
B	defaults/main.yml
C	variables/main.yml
D	tasks/main.yml