credativ

From Playbook to Role — Participant Handout

Ansible for Proxmox VE · Module 04

credativ GmbH

Table of contents

1 When to structure code into roles

A single playbook file is the right answer for a while. It stops being the right answer when one of these becomes true:

- you want the same tasks in a second project
- values and logic are tangled in one file and you edit the wrong one
- more than one person maintains it
- the file no longer fits on one screen

A role is not a new technology. It is **a directory layout with conventions**: Ansible loads `main.yml` from a fixed set of subdirectories, so you do not wire anything together yourself.

2 The layout

```
roles/pve_node/
├── defaults/main.yml    # values, lowest precedence - meant to be overridden
├── vars/main.yml         # values that are NOT meant to be overridden
├── tasks/main.yml        # the tasks (no "hosts:", no play keys)
├── handlers/main.yml     # the handlers
├── templates/            # Jinja2 templates (.j2)
├── files/                # files copied verbatim
└── meta/main.yml         # dependencies and metadata
```

Only the directories you need have to exist. A role with just `tasks/main.yml` is a valid role.

3 The conversion

The playbook becomes three lines:

```
---
- name: Prepare the Proxmox VE nodes
  hosts: pve
  gather_facts: true
  roles:
    - pve_node
```

`roles/pve_node/tasks/main.yml` receives the task list, and *only* the task list. No `hosts:`, no `tasks:` key, no leading play. A role task file is a YAML list of tasks at the top level:

```
---
- name: Disable the enterprise repositories
  ansible.builtin.file:
    path: "{{ item }}"
    state: absent
  loop: "{{ pve_enterprise_repo_files }}"
  notify: Update apt cache
  tags: [repos]

- name: Enable the no-subscription repository
  ansible.builtin.deb822_repository:
    name: pve-no-subscription
    types: [deb]
    uris: "{{ pve_repo_uri }}"
    suites: "{{ ansible_distribution_release }}"
    components: ["{{ pve_repo_component }}"]
    signed_by: "{{ pve_keyring }}"
  notify: Update apt cache
  tags: [repos]
```

Handlers move to `handlers/main.yml` in exactly the same way.

4 `defaults/` is the role's public interface

Every value that a user of the role might reasonably want to change belongs here:

```
---
pve_packages:
  - vim
  - curl
  - jq
  - tmux

```

```
pve_repo_uri: http://download.proxmox.com/debian/pve
pve_repo_component: pve-no-subscription
pve_keyring: /usr/share/keyrings/proxmox-archive-keyring.gpg

pve_enterprise_repo_files:
  - /etc/apt/sources.list.d/pve-enterprise.sources
  - /etc/apt/sources.list.d/ceph.sources
```

`vars/main.yml` is the mirror image: values the role needs internally and callers should not override. It sits high in the precedence order, so use it sparingly: a value in `vars/` beats `group_vars/`, which surprises people.

5 Variable precedence

The full list has 22 levels. These are the ones you will actually meet, from weakest to strongest:

#	Source
1	`roles/<role>/defaults/main.yml`
2	`inventory/group_vars/all.yml`
3	`inventory/group_vars/<group>.yml`
4	`inventory/host_vars/<host>.yml`
5	`roles/<role>/vars/main.yml`
6	`vars:` in the play
7	`--extra-vars` on the command line

The working rule: **define in `defaults/`, override in `group_vars/`.** Reach for `--extra-vars` only for one-off runs, never as part of a documented procedure, since it is invisible to anyone reading the repository.

6 Templates

A template renders a file from inventory data and facts, with the logic in its own file instead of inside the YAML.

```
- name: Render the message of the day from the inventory
  ansible.builtin.template:
    src: motd.j2
    dest: /etc/motd
    owner: root
    group: root
    mode: "0644"
  tags: [motd]
```

```
  This node is managed by Ansible (role pve_node).

    node    : {{ inventory_hostname }} ({{ ansible_host }})
    cluster : {{ pve_cluster_name | default('not configured') }}
    peers   : {{ groups['pve'] | difference([inventory_hostname]) | sort
| join(', ') }}
```

Every value comes from somewhere you already know: `inventory_hostname` and `ansible_host` from the inventory, `pve_cluster_name` from `group_vars`, `groups['pve']` from the inventory at run time.

> **i** Note
>
> The `| sort` is not cosmetic. `difference()` returns a set, and a set has no guaranteed order, so without sorting the rendered file can differ between runs and the task reports `changed` forever. Any template that joins an unordered collection needs a `sort` to stay idempotent.

6.1 `template` owns the whole file

	`blockinfile`	`template`
Owns	a marked region	the whole file
Changes by others	survive	are overwritten
Readability	Jinja inside YAML	logic in its own file
Use when	the file has other owners	the file is yours alone

> **⚠** Pick the file before you pick the module
>
> `/etc/motd` is yours: nothing else writes it, so owning it completely is safe.
>
> `/etc/hosts` is not. The provisioning writes it, together with the node's real fully qualified domain name, and Proxmox VE resolves its **own** node name through it. A template that rewrites that file with an invented domain changes the identity of every node in one run. Module 02 therefore *checks* name resolution instead of configuring it.

7 Tags

```
  tags: [repos]
```

```
ansible-playbook site.yml --tags repos
ansible-playbook site.yml --skip-tags packages
ansible-playbook site.yml --list-tags
```

Tags make iteration fast while you develop a role. They are not a structuring tool: a tagged partial run can produce a host state that no complete run would ever produce, and nobody will remember which tag combination was used six months later.

8 `ansible-lint`

```
ansible-lint site.yml roles/
```

On the workshop desktop it is already installed, inside the virtualenv.

Two findings show up in this project, and the difference between them is the point:

- **`var-naming[no-role-prefix]`** wants every variable in `roles/pve_node` to be called `pve_node_*`, so that two roles in one play cannot collide. Here the same names live in `group_vars` and are read by the plays as well, so prefixing only the role's copies would give one value two spellings. The project records that decision in `.ansible-lint` rather than renaming.
- **`args[module]`** claims `missing required arguments: api_host, api_user` for every Proxmox task in `30-operations.yml`. That is wrong: the play sets them once through `module_defaults`, which ansible-lint does not resolve. It is reported as a warning.

Skipping a rule is legitimate when the reason is written down next to the skip. Skipping it to make the output quiet is how a linter stops being useful.

It flags missing task names, `shell` where `command` is enough, deprecated module names, unspecified file modes, and a long list of smaller things. Running it once before committing is a cheap habit with a high return, and it teaches idiomatic Ansible faster than any tutorial.

9 Further reading

Structure

- Roles (the directory layout, `roles:` versus `include_role`, role dependencies)
- Variable precedence: where should I put a variable? (the complete list of 22 levels, when you need it)
- Sample Ansible setup (a worked directory layout for a real project, worth copying)
- Tips and tricks (the closest thing to an official style guide)
- Tags

Templating

- Templating with Jinja2
- `ansible.builtin.template`
- Jinja2 template designer documentation (the upstream reference for loops, conditionals and filters)

Tooling

- `ansible-galaxy` command (`ansible-galaxy role init` creates the skeleton for you)
- ansible-lint documentation, including the rule index, which explains *why* each rule exists rather than only what it flags
- Ansible Galaxy: before writing a role, check whether someone has already written it

Offline

```
ansible-doc ansible.builtin.template
ansible-galaxy role init --init-path roles my_role
```