Exercise 04 — From Playbook to Role

Ansible for Proxmox VE · Module 04

credativ GmbH

~10 min · in `~/ansible-proxmox` · needs exercise 02; exercise 03 is not required

> **Objective:** turn `playbooks/10-node-prep.yml` into the role `pve_node` and prove that the behaviour did not change.

> **i** Where to look
>
> Everything here was on the slides; the handout has the finished role file by file. Roles · Variable precedence · `template` · ansible-lint

The directories, `site.yml` and the template are already in the starter. What is missing is the content.

1 — Give the role its defaults

`roles/pve_node/defaults/main.yml` has the values as comments. Uncomment them, then **delete `pve_packages` from `inventory/group_vars/pve.yml`**.

Leave `pve_cluster_name` and `pve_root_password` where they are: those describe your environment, not the role.

Done when: `pve_packages` appears exactly once in the project.

```
grep -rn pve_packages inventory/ roles/
```

2 — Move the tasks and the handler

From `playbooks/10-node-prep.yml` into `roles/pve_node/tasks/main.yml` and `handlers/main.yml`: the tasks only — no `hosts:`, no `tasks:` key, indentation shifted left. Replace the hard-coded repository values with the variables you just uncommented.

Leave the old playbook alone. It stays as the before-picture.

Done when: no `ansible.builtin.debug` placeholder is left in the role.

3 — Render the message of the day

`roles/pve_node/templates/motd.j2` is written for you — read it first, it is the whole point of the module. Add the task that renders it:

```
ansible-doc ansible.builtin.template    # src, dest, owner, group, mode
```

> **Note**
>
> `/etc/motd` is a deliberate choice. A template owns the **whole** file, so pick one nothing else writes. `/etc/hosts` would be wrong: the provisioning owns it and Proxmox VE resolves its own node name through it.

Done when: the role has a `template` task with `tags: [motd]`.

4 — Run it three times

```
ansible-playbook site.yml --check --diff
ansible-playbook site.yml
ansible-playbook site.yml
```

Done when:

```
# first real run
pve01 : ok=8  changed=1  unreachable=0  failed=0
# second run
pve01 : ok=8  changed=0  unreachable=0  failed=0
```

The single `changed` is `/etc/motd`, because the flat playbook never wrote it. Everything else was already in the state the role describes — that is the proof the refactoring changed the structure and not the outcome.

If you finish early

- Put `pve_repo_component: from-group-vars` into `inventory/group_vars/pve.yml`, then `ansible pve01 -m ansible.builtin.debug -a 'var=pve_repo_component'`. Which value wins over the role default, and why?
- Add `--extra-vars 'pve_repo_component=from-cli'`. Which wins now? Use `debug` only — do not run `site.yml` with a bogus repository component. Remove the override afterwards.
- `ansible-playbook site.yml --list-tags`, then run only `--tags motd`.
- `ansible-lint site.yml roles/ playbooks/`. Then read `.ansible-lint` in the project: which finding was fixed, which was recorded as a decision, and why?

Solutions

Solution — Task 1: the defaults

The point participants miss: the values must be **removed** from `group_vars/pve.yml`. Leave them there and `group_vars` keeps outranking the role defaults, so the role's interface is decorative.

```
grep -rn pve_packages inventory/ roles/
# roles/pve_node/defaults/main.yml:5:pve_packages:
```

Exactly one hit, in the role.

> ⚠ Warning
>
> Do **not** have them check this with `ansible pve01 -m ansible.builtin.debug -a 'var=pve_packages'`. Role defaults are in scope only while the role runs, so an ad-hoc command prints `<< error 1 - 'pve_packages' is undefined >>` — which is correct behaviour and looks exactly like a broken refactoring. If someone tries it anyway, that is a good thirty seconds on variable scope.

The directories, `site.yml` and `templates/motd.j2` are already in the starter. `ansible-galaxy role init` would additionally create `vars/`, `files/`, `meta/` and `tests/`; empty directories do no harm, and only the ones actually used need to exist.

Solution — Task 2: tasks and handler

```yaml
# roles/pve_node/tasks/main.yml
---
- 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]

- name: Apply pending handlers now
  ansible.builtin.meta: flush_handlers

- name: Install the admin tools
  ansible.builtin.apt:
    name: "{{ pve_packages }}"
    state: present
```

```
  tags: [packages]

- name: Verify that every node resolves the other cluster nodes
  ansible.builtin.command: "getent hosts {{ item }}"
  register: pve_resolve
  changed_when: false
  failed_when: pve_resolve.rc != 0
  loop: "{{ groups['pve'] | difference([inventory_hostname]) }}"
  loop_control: {label: "{{ item }}"}
  tags: [hosts]

- 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]

- name: Check whether the clock is synchronised
  ansible.builtin.command: timedatectl show -p NTPSynchronized --value
  register: ntp_state
  changed_when: false
  check_mode: false
  tags: [time]

- name: Fail early if time is not in sync
  ansible.builtin.assert:
    that: ntp_state.stdout == 'yes'
    fail_msg: "Time is not synchronised - corosync will be unhappy"
  tags: [time]
```

```
# roles/pve_node/handlers/main.yml
---
- name: Update apt cache
  ansible.builtin.apt:
    update_cache: true
```

Reference copy: `courses/ansible-proxmox/code/roles/pve_node/`.

Errors to expect:

- `hosts:` or `tasks:` left in the role file → *"Invalid task"* or a play that runs nothing. A role task file is a bare list of tasks.
- Indentation shifted by two instead of four spaces → the task list parses as something else entirely.
- Handler name mismatch after the move → the handler never fires, `apt` fails on a stale cache.

Solution — Task 3: the template

Template as printed in the task.

Verify:

```
ansible pve01 -m ansible.builtin.command -a 'cat /etc/motd'
ssh root@192.168.0.1 true    # the motd appears on the next interactive login
```

Watch for the `| sort` in the peers line. Without it, `difference()` returns an unordered set, the file flaps between runs, and the task reports `changed` forever. That is the most common cause of a template that will not settle, and it is worth showing on purpose if someone's second run is not clean.

The point to make: a template owns the file completely. That is fine for `/etc/motd` and wrong for `/etc/hosts`, which the provisioning writes and which carries the node's real FQDN. If someone asks why we do not generate `/etc/hosts` from the inventory — that is the answer, and it is the more valuable lesson of the two.

Solution — Task 4: the proof

```
# site.yml
---
- name: Prepare the Proxmox VE nodes
  hosts: pve
  gather_facts: true
  roles:
    - pve_node
```

Expected: first run `changed=1` (the motd template), second run `changed=0`.

Make the point explicitly: the refactoring was verified by the tool itself. No test suite was needed to show the behaviour is unchanged, because `changed=0` already says so.

Solution — Optional tasks

Command	Winning value	Why
default only	`pve-no-subscription`	`defaults/` is the only source
`group_vars/pve.yml` set	`from-group-vars`	`group_vars` outranks `defaults`
plus `--extra-vars`	`from-cli`	`--extra-vars` outranks everything

```
ansible-playbook site.yml --list-tags
#   TASK TAGS: [hosts, packages, repos, time]
ansible-playbook site.yml --tags hosts

ansible-lint site.yml roles/
```

Measured against the reference project with ansible-lint 26.8:

```
var-naming[no-role-prefix]  7x   roles/pve_node/...   -> exit 2
args[module]                9x   playbooks/...        -> warning only
```

`var-naming[no-role-prefix]` is real. The rule wants every variable in `roles/pve_node` to be called `pve_node_*`, so that two roles combined in one play cannot collide. In this project the same names are set in `group_vars`, read by the plays and read by the role; prefixing only the role's copies would give one value two spellings. The reference project therefore records the deviation in `.ansible-lint` instead of renaming. Show that file: silencing a linter is fine when the reason is written down, and only then.

`args[module]` reports `missing required arguments: api_host, api_user` for every `community.proxmox` task in `30-operations.yml`. That one is a **false positive**: the play sets those once through `module_defaults`, which ansible-lint does not resolve. It is a warning and does not fail the run. Worth a minute, because knowing when to overrule a linter is the actual skill.

> **i** Note
>
> `ansible-lint --version` crashes with a traceback in this combination of ansible-lint 26.8 and Python 3.13. Linting itself works. If a participant tries the version flag first and concludes the tool is broken, that is why.

Automated verification

```
./tests/run.sh 04
```

Checks: the role directory layout exists, `site.yml` references the role, values are no longer duplicated in `group_vars`, `/etc/hosts` contains the node's own entry plus both peers, and two consecutive `site.yml` runs report `changed=0`.