Exercise 03 — Forming the Cluster

Ansible for Proxmox VE · Module 03

credativ GmbH

~20 min · in `~/ansible-proxmox` · after exercise 02

> **Objective:** create the cluster on `pve01`, join the other two one at a time, and end with a playbook you can run again without it doing anything.

> **i** Where to look
>
> Everything here was on the slides; the handout has the same playbook with every line explained, and the `curl` equivalents. `proxmox_cluster` · `proxmox_cluster_join_info` · `delegate_to` · `serial` · Proxmox VE Cluster Manager

1 — Give the playbook the password

`inventory/group_vars/pve.yml` ships with a placeholder. The API modules authenticate with it, so replace it:

```
pve_root_password: "password123"
```

Done when:

```
ansible pve01 -m ansible.builtin.debug -a 'var=pve_root_password'
```

prints the password rather than `VARIABLE IS NOT DEFINED`.

The collection is already installed — `deploy.sh` put it in place offline. Check with `ansible-doc -l community.proxmox | wc -l`; it lists 62 modules.

2 — Fill in the playbook

`playbooks/20-cluster.yml` has the structure already: two plays, `serial: 1` on the second, and a `TODO` in each of the five task bodies. The comments say which module and which options.

```
ansible-doc community.proxmox.proxmox_cluster
ansible-doc community.proxmox.proxmox_cluster_join_info
```

Three things the skeleton does not do for you:

- every API task needs `delegate_to: localhost` — the module talks HTTPS, it does not run on the node

- the two waiting tasks are `until` / `retries` / `delay` loops, not `wait_for`
- the fingerprint for the join sits in the registered join info, under the primary's entry in `cluster_join.nodelist`

Done when: the file has no `ansible.builtin.debug` left in it.

3 — Run it

```
ansible-playbook playbooks/20-cluster.yml
```

Open the GUI of `pve01` while it runs. One to three minutes.

> **i** Note
>
> **No `--check` here.** A dry run creates no cluster, so the join play has nothing to read the join information from and stops with *"Node is not part of a cluster and does not have any join information"*. That is correct behaviour: check mode cannot rehearse a sequence whose second step depends on the first having really happened.

Done when:

```
pve01 : ok=2  changed=1  unreachable=0  failed=0
pve02 : ok=3  changed=1  unreachable=0  failed=0
pve03 : ok=3  changed=1  unreachable=0  failed=0
```

and all three nodes appear under *Datacenter* in the GUI.

4 — Prove it, twice

```
ansible pve01 -m ansible.builtin.command -a 'pvecm status'
ansible pve -m ansible.builtin.command -a 'ls /etc/pve/nodes'
ansible-playbook playbooks/20-cluster.yml
```

Done when: `Quorate: Yes` with three nodes, the same three directories on every node — that is pmxcfs replicating — and the second playbook run reports `changed=0` everywhere.

If you finish early

- Look at `/etc/pve/corosync.conf` on any node. Find your `link0` addresses and the `config_version` counter.
- Write a file into `/etc/pve/` on `pve02` and find it on `pve03`: `ansible pve02 -m ansible.builtin.copy -a 'content=hello dest=/etc/pve/hello.txt'`. Remove it again afterwards.
- Stop corosync on `pve03` (`systemctl stop corosync`), then run `pvecm status` on `pve01` and on `pve03` and compare the two answers. Start it again.
- `ssh root@192.168.0.1` and look at `/root/.ssh/authorized_keys`. Where does it actually point, and what does that mean for a node that joins later?

Solutions

Solution — Task 1: the password and the collection

```
ansible-galaxy collection install -r requirements.yml
ansible-doc -l community.proxmox | wc -l          # well above 50
python3 -c 'import proxmoxer, requests; print("ok")'
```

If it fails:

- `Failed to import the required Python library (proxmoxer)` → installed for a different interpreter than the one Ansible uses. `ansible --version` prints the Python path; install into that one, or use the distribution package `python3-proxmoxer`.
- No network on the desktop → install the collection from a local tarball. Prepare one before the session: `ansible-galaxy collection install community-proxmox-2.0.0.tar.gz`

Solution — Task 2: the playbook

```
# inventory/group_vars/pve.yml  (added below the module 02 variables)
pve_cluster_name: training
pve_root_password: "password123"
```

Verify:

```
ansible pve01 -m ansible.builtin.debug -a 'var=pve_root_password'
```

prints the configured password. Say once that this belongs in a secret store in production, then move on.

Solution — Task 2, continued

The complete playbook is printed in the task; the reference copy is `courses/ansible-proxmox/code/playbooks/20-cluster.yml`.

The four things participants get wrong, in order of frequency:

1. **`delegate_to: localhost` missing** on an API task → Ansible tries to run the module on the Proxmox node, which has no `proxmoxer`.
2. **`serial: 1` missing** on the join play → both nodes join simultaneously. It may even work, which is worse, because it teaches the wrong lesson. If someone hits the race, use it: show `pvecm status` disagreeing between nodes.
3. **The fingerprint expression** typed with `'eq'` instead of `'equalto'`, or filtering on the wrong key. Both fail with a Jinja error naming the filter.
4. **Indentation of the multi-line `fingerprint:` expression.** It is one YAML scalar split across three lines; the continuation lines must be indented further than the key.

Solution — Task 3: the run

```
ansible-playbook playbooks/20-cluster.yml --check
ansible-playbook playbooks/20-cluster.yml
```

If someone runs `--check` anyway, it fails in the join play, and that is worth two minutes: the first play would create the cluster, but in check mode it does not, so `proxmox_cluster_join_info` has nothing to read. Check mode rehearses individual tasks, not a sequence where step two depends on step one having really happened.

The real run takes roughly one to three minutes. Expected recap:

```
pve01 : ok=2  changed=1  failed=0
pve02 : ok=3  changed=1  failed=0
pve03 : ok=3  changed=1  failed=0
```

Solution — Task 4: proving it

```
ansible pve01 -m ansible.builtin.command -a 'pvecm status'
#   Cluster name:  training
#   Quorate:       Yes
#   Expected votes: 3     Total votes: 3

ansible pve01 -m ansible.builtin.command -a 'pvecm nodes'  # three membership
lines
ansible pve  -m ansible.builtin.command -a 'ls /etc/pve/nodes'
#   pve01 pve02 pve03  - identical on all three nodes
```

The last command is the clearest demonstration of pmxcfs replication in the whole workshop. Spend a minute on it.

The second playbook run reports `changed=0`: `proxmox_cluster` reads the cluster status first and returns *"Cluster 'training' already present"* respectively *"Node already in the cluster"*.

Solution — Optional tasks

```
ansible pve01 -m ansible.builtin.command -a 'cat /etc/pve/corosync.conf'
# one ring0_addr per node, config_version incremented once per join

ansible pve02 -m ansible.builtin.copy -a 'content=hello dest=/etc/pve/
hello.txt'
ansible pve03 -m ansible.builtin.command -a 'cat /etc/pve/hello.txt'  # hello
ansible pve03 -m ansible.builtin.file -a 'path=/etc/pve/hello.txt
state=absent'

ansible pve03 -m ansible.builtin.systemd_service -a 'name=corosync
state=stopped'
ansible pve01 -m ansible.builtin.command -a 'pvecm status'  # Quorate: Yes
(2 of 3)
ansible pve03 -m ansible.builtin.command -a 'pvecm status'  # Quorate: No
ansible pve03 -m ansible.builtin.systemd_service -a 'name=corosync
state=started'
```

On the isolated node, `/etc/pve` is read-only. Have someone try to write a file there and read the error aloud. This is split-brain protection working as designed, and it sticks with people better than the slide does.

Recovery: a cluster broken beyond repair

The Proxmox VE API can create and join clusters, but it cannot make a node leave one. A node that joined the wrong cluster, or a repeated lab run against a dirty environment, cannot be fixed from Ansible. Options during a live session, in order:

1. Pair the participant with a neighbour and continue on the working cluster.
2. Reset the affected VM from the snapshot the environment provides.
3. Continue with the remaining two nodes. Modules 04 to 06 work fine on a two-node cluster as long as it is quorate.

Do not attempt a manual `pvecm delnode` recovery during the workshop. It costs more time than it is worth and takes attention away from the room.

Automated verification

```
./tests/run.sh 03
```

Checks: exactly one cluster exists with the configured name, all three nodes are members and online, the cluster is quorate, `link0` matches the management addresses from the inventory, `/etc/pve/nodes` is identical on all nodes, and a repeated playbook run reports `changed=0`.