

Dynamic Inventory — Participant Handout

Ansible for Proxmox VE · Module 06

credativ GmbH

Table of contents

1	Why static inventories go stale	1
2	Configuring the plugin	1
3	Using it	2
3.1	What the plugin actually calls	3
4	Facts	3
5	Groups from tags	3
6	Combining static and dynamic	4
7	Caching	4
8	Further reading	5

1 Why static inventories go stale

A hand-maintained list of guests is correct on the day you write it. After that, every VM created in the GUI, every one deleted, every rename and every migration is a small divergence between your file and reality. You notice at the worst possible time, when a playbook fails against a host that no longer exists.

The cluster already knows all of it. An **inventory plugin** asks the API at run time instead of reading a file.

Read the general concept first, it applies to every cloud and platform plugin: [Working with dynamic inventory](#).

2 Configuring the plugin

The plugin is `community.proxmox.proxmox`, from the collection you installed in module 03. Its complete option list is in the [plugin documentation](#).

```

1  plugin: community.proxmox.proxmox
2  url: https://192.168.0.1:8006
3  user: automation@pve
4  token_id: ansible
5  token_secret: "{{ lookup('ansible.builtin.env', 'PVE_TOKEN_SECRET') }}"
6  validate_certs: false
7  want_facts: true

```

! The file name matters

An inventory source for this plugin **must** end in `.proxmox.yml` or `.proxmox.yaml`. With any other name the plugin declines the file silently and you get an empty inventory with no error. `ansible-inventory -vvv` shows the reason.

i Least privilege in production: PVEAuditor

In this lab, we reuse the `automation@pve` token created in module 05 (PVEVMAdmin). In production, never grant administrative roles to an inventory plugin. A dedicated, read-only service account with the `PVEAuditor` role on / provides all the visibility the dynamic inventory needs without exposing cluster management privileges.

3 Using it

```
1  ansible-inventory -i inventory/lab.proxmox.yml --graph
2  ansible-inventory -i inventory/lab.proxmox.yml --list
3  ansible-inventory -i inventory/lab.proxmox.yml --host web01
```

```
@all:
  |--@proxmox_all_qemu:
  | |--web01
  |--@proxmox_all_running:
  | |--web01
  |--@proxmox_all_stopped:
  |--@proxmox_nodes:
  | |--pve01
  | |--pve02
  | |--pve03
  |--@proxmox_pve02_qemu:
  | |--web01
```

The generated groups follow a fixed scheme (the prefix is configurable with `group_prefix`):

Group	Contains
<code>proxmox_nodes</code>	the cluster nodes themselves
<code>proxmox_all_qemu</code> / <code>proxmox_all_lxc</code>	all VMs / all containers
<code>proxmox_all_running</code> / <code>proxmox_all_stopped</code>	by current status
<code>proxmox_all_templates</code>	guests marked as templates
<code>proxmox_<node>_qemu</code> / <code>proxmox_<node>_lxc</code>	guests per node
<code>proxmox_pool_<pool></code>	guests in a resource pool

Nothing in that list is maintained by you. Create a guest in the GUI, run `ansible-inventory --graph` again, and it is there.

3.1 What the plugin actually calls

```

1  TOKEN='PVEAPIToken=automation@pve!ansible=xxxxxxx-xxxx-xxxx-xxxx-
   xxxxxxxxxxxx'
2  PVE=https://192.168.0.1:8006/api2/json
3
4  # everything the cluster knows about, in one call - this is the inventory
5  curl -sk -H "Authorization: $TOKEN" "$PVE/cluster/resources" \
6      | jq '.data[] | select(.type=="qemu") | {vmid, name, node, status,
   tags}'
7
8  # the nodes
9  curl -sk -H "Authorization: $TOKEN" "$PVE/nodes" | jq '.data[] | {node,
   status}'
10
11 # the configuration of one guest - this is where want_facts gets its data
12 curl -sk -H "Authorization: $TOKEN" "$PVE/nodes/pve02/qemu/9001/config"
   | jq

```

API reference: `/cluster/resources`.

Running that first command is the fastest way to understand what the plugin can and cannot know: if a field is not in the response, no inventory option will conjure it up. It is also a useful reality check when a group comes out empty: compare what the API returns with what you filtered on.

4 Facts

With `want_facts: true` each guest carries its Proxmox VE configuration as host variables:

```

{
  "proxmox_vmid": 9001,
  "proxmox_node": "pve02",
  "proxmox_status": "stopped",
  "proxmox_maxmem": 536870912,
  "proxmox_tags_parsed": ["web", "staging"]
}

```

These are available in playbooks like any other variable, and also to the grouping expressions below.

5 Groups from tags

Proxmox VE lets you tag guests. Tags are the natural place for “what is this machine for”, because the person creating the VM sets them, in the GUI, at creation time.

```

1  want_facts: true
2
3  keyed_groups:
4    - key: proxmox_tags_parsed
5      separator: ""
6      prefix: tag
7
8  groups:
9    webserver: "'web' in (proxmox_tags_parsed | list)"
10   production: "'prod' in (proxmox_tags_parsed | list)"
11
12  compose:
13    ansible_host: proxmox_ipconfig0.ip | default(proxmox_net0.ip) |
ansible.utils.ipaddr('address')

```

keyed_groups, groups and compose come from Ansible's *constructed* inventory features and work the same way in every inventory plugin. See [Using inventory plugins](#) and the [constructed plugin](#).

i Note

compose: ansible_host: ... needs care. Without the QEMU guest agent installed, a VM cannot report its IP address at all, and the plugin falls back to the guest name, which only works if your DNS knows it. The plugin documentation contains several worked variants; pick the one that matches how your environment resolves guests.

6 Combining static and dynamic

The cluster nodes have to exist before anything can be dynamic, so they stay in the static inventory. The guests come from the API.

```
ansible-playbook site.yml -i inventory/hosts.yml -i inventory/lab.proxmox.yml
```

[defaults]

```
inventory = inventory/hosts.yml,inventory/lab.proxmox.yml
```

Both sources are merged into one inventory. Groups from both are available in the same run, and host variables from the later source win.

7 Caching

Every run queries the API. On a large cluster, or in a pipeline that runs many playbooks, enable the inventory cache:

```

cache: true
cache_plugin: ansible.builtin.jsonfile

```

```
cache_connection: /tmp/ansible_inventory_cache
cache_timeout: 300
```

The settings come from Ansible's [cache plugins](#). Be aware of the trade-off: a cached inventory is a static inventory for the length of the timeout, with all the staleness that implies.

8 Further reading

- [Inventory plugin](#) `community.proxmox.proxmox` (every option, with examples)
- [Working with dynamic inventory](#)
- [How to build your inventory](#)
- `ansible-inventory` [command](#)
- [Proxmox VE Administration Guide](#): guest tags are described in the *Qemu/KVM Virtual Machines* chapter
- [Proxmox VE API viewer](#): look at `/cluster/resources`, which is what the plugin calls