

# Capitolo 4 – Inventory: Struttura e Gestione Professionale

L'inventory è il punto di partenza di qualsiasi operazione con Ansible.

Definisce:

- Quali host devono essere gestiti
- Come connettersi agli host
- Come raggrupparli logicamente
- Quali variabili applicare

Una gestione corretta dell'inventory è fondamentale in ambienti sysadmin.

---

## 4.1 Inventory Statico (Formato INI)

È il formato più semplice e immediato.

Esempio file `inventory/production`:

```
[web]
web01 ansible_host=192.168.10.10
web02 ansible_host=192.168.10.11

[db]
db01 ansible_host=192.168.10.20

[all:vars]
ansible_user=ansible
ansible_python_interpreter=/usr/bin/python3
```

## Spiegazione

- `[web]` → gruppo logico di server
- `web01` → alias host
- `ansible_host` → IP reale o FQDN

- `[all:vars]` → variabili applicate a tutti gli host

Test di connettività:

```
ansible all -m ping
```

## 4.2 Parametri Host Specifici

È possibile definire parametri direttamente sull'host:

```
web01 ansible_host=192.168.10.10 ansible_port=2222 ansible_user=deploy
```

Parametri comuni:

- `ansible_host`
- `ansible_user`
- `ansible_port`
- `ansible_ssh_private_key_file`
- `ansible_python_interpreter`

Questo permette di gestire ambienti eterogenei.

## 4.3 Inventory in YAML

Alternativa più leggibile in ambienti complessi.

Esempio:

```
all:
  vars:
    ansible_user: ansible
    ansible_python_interpreter: /usr/bin/python3
  children:
    web:
      hosts:
        web01:
          ansible_host: 192.168.10.10
        web02:
          ansible_host: 192.168.10.11
```

```
db:
  hosts:
    db01:
      ansible_host: 192.168.10.20
```

Vantaggi:

- Struttura gerarchica chiara
- Migliore gestione di ambienti complessi
- Più coerente con il resto dell'ecosistema YAML di Ansible

---

## 4.4 Gruppi e Sottogruppi

È possibile creare gruppi annidati.

Esempio:

```
[web]
web01
web02

[frontend]
web01

[backend]
web02

[production:children]
web
db
```

Questo permette di:

- Applicare playbook a gruppi specifici
- Creare logiche di infrastruttura multilivello

Esempio esecuzione su gruppo specifico:

```
ansible web -m command -a "uptime"
```

---

## 4.5 File `group_vars` e `host_vars`

Separare le variabili dall'inventary è una best practice.

Struttura:

```
group_vars/  
├─ web.yml  
└─ db.yml  
  
host_vars/  
└─ web01.yml
```

Esempio `group_vars/web.yml`:

```
http_port: 80  
max_clients: 200
```

Esempio `host_vars/web01.yml`:

```
http_port: 8080
```

Regola importante:

Le variabili host specifiche hanno priorità su quelle di gruppo.

---

## 4.6 Separazione Ambienti

Non utilizzare mai un unico inventory per tutto.

Struttura consigliata:

```
inventory/  
├─ production  
├─ staging  
└─ dev
```

Esecuzione su ambiente specifico:

```
ansible-playbook -i inventory/staging playbooks/site.yml
```

Questo evita errori critici su produzione.

---

## 4.7 Best Practice Sysadmin

- Non inserire password in chiaro nell'inventary
  - Utilizzare ansible-vault per credenziali
  - Separare ambienti in file distinti
  - Usare alias host leggibili
  - Definire sempre ansible\_python\_interpreter
  - Versionare l'inventary con Git (con attenzione ai segreti)
- 

## 4.8 Verifica Inventory

Comandi utili:

Visualizzare host:

```
ansible-inventory -i inventory/production --list
```

Visualizzare struttura grafica:

```
ansible-inventory -i inventory/production --graph
```

Questi strumenti aiutano nel troubleshooting.

---

## Conclusione

L'inventary non è solo un elenco di server.  
È la rappresentazione logica della tua infrastruttura.

Una progettazione corretta permette:

- Scalabilità
- Chiarezza operativa
- Riduzione errori
- Maggiore controllo sugli ambienti

Nel prossimo capitolo entreremo nei Comandi Ad-Hoc e nell'operatività quotidiana da sysadmin.

---

Revision #1

Created 2026-02-26 13:53:39 UTC by Pe

Updated 2026-02-26 13:54:27 UTC by Pe