Capitolo 10 – Logging, Monitoraggio e Troubleshooting

La gestione dei server non si limita a deploy e configurazione: è fondamentale **monitorare lo stato** e poter **diagnosticare problemi** rapidamente.

Ansible offre strumenti integrati per log e debug, ma è utile combinarli con best practice sysadmin.

10.1 Verbosità e Output

Per ottenere informazioni dettagliate durante l'esecuzione:

```
ansible-playbook playbooks/site.yml -v       # livello base
ansible-playbook playbooks/site.yml -vv      # più dettagli
ansible-playbook playbooks/site.yml -vvv     # debug SSH e task
ansible-playbook playbooks/site.yml -vvvv    # dettagli estremi (output completo)
```

- Utilizzare i livelli più alti solo in fase di debug
- Per grandi infrastrutture, può generare molto output

10.2 Modulo Debug

Il modulo `debug` è fondamentale per verificare variabili o output intermedi.

Esempio:

```
- name: Mostra valore di una variabile
  debug:
    var: http_port

- name: Mostra messaggio personalizzato
  debug:
    msg: "Deploy completato su {{ inventory_hostname }}"
```

10.3 Check Mode

Permette di simulare l’esecuzione senza modificare i server.

```
ansible-playbook playbooks/base.yml --check
```

Utile per:

- Verificare impatto dei cambiamenti
- Evitare errori in produzione
- Validare logica dei playbook

10.4 Test di Syntax

Prima di eseguire un playbook, controllarne la sintassi:

```
ansible-playbook playbooks/base.yml --syntax-check
```

Permette di:

- Evitare errori di YAML
- Verificare correttezza struttura tasks e ruoli

10.5 Registrare Output

È possibile salvare l’output di un task in una variabile per uso successivo:

```
- name: Controlla spazio disco
  command: df -h /var
  register: disco

- name: Mostra spazio disco
  debug:
    var: disco.stdout
```

- `register` salva l’output del task

- Permette logica condizionale e troubleshooting

10.6 Gestione Errori

- Task possono essere ignorati con `ignore_errors: yes`
- Condizioni di fallback con `failed_when`
- Retry automatici con `retries` e `delay` (modulo `until`)

Esempio:

```
- name: Esegue comando con retry
  command: /usr/bin/check_service
  register: result
  retries: 3
  delay: 10
  until: result.rc == 0
```

10.7 Log Centralizzato

Per infrastrutture grandi, utile salvare output dei playbook:

```
ansible-playbook playbooks/site.yml | tee /var/log/ansible/site.log
```

- Consente audit e storico modifiche
- Utile per verificare esecuzioni su più server

10.8 Best Practice Troubleshooting

- Testare playbook in staging prima di prod
- Usare `--check` e `--diff` per evitare modifiche indesiderate
- Usare `debug` per variabili e output
- Registrare log per audit e rollback
- Isolare task problematici per esecuzione separata

10.9 Conclusione

Un corretto approccio a logging e troubleshooting permette di:

- Rilevare problemi rapidamente
- Migliorare sicurezza e affidabilità
- Automatizzare interventi di controllo
- Mantenere infrastruttura coerente

Revision #2
Created 2026-02-26 14:14:14 UTC by Pe
Updated 2026-02-26 14:15:57 UTC by Pe