Capitolo 8 – Deploy Applicativo e Template Jinja2

Dopo aver appreso ruoli, variabili e gestione dei segreti, il passo successivo è imparare a **deployare applicazioni** in modo automatizzato e a utilizzare **template Jinja2** per configurazioni dinamiche.

8.1 Cos’è un Template Jinja2

I template Jinja2 permettono di creare file di configurazione dinamici.

Caratteristiche principali:

- Sintassi semplice simile a Python
- Variabili sostituite al momento dell’esecuzione
- Supporto per cicli, condizioni e filtri
- Ideale per configurazioni come nginx, apache, systemd, ecc.

Esempio minimo di template `nginx.conf.j2`:

```
server {
    listen {{ http_port }};
    server_name {{ server_name }};

    root {{ web_root }};
    index index.html;
}
```

8.2 Creazione di un Template per il Web Server

Esempio ruolo `roles/nginx/templates/nginx.conf.j2`:

```
user www-data;
worker_processes auto;

pid /run/nginx.pid;

events {
    worker_connections 768;
}

http {
    server {
        listen {{ http_port }};
        server_name {{ server_name }};
        root {{ web_root }};
        index index.html index.htm;

        location / {
            try_files $uri $uri/ =404;
        }
    }
}
```

Variabili tipiche da definire in `group_vars/web.yml`:

```
http_port: 80
server_name: example.com
web_root: /var/www/html
```

8.3 Copiare Template con il Modulo `template`

Playbook esempio: `playbooks/deploy_web.yml`

```
- name: Deploy Web Server
  hosts: web
  become: true
```

```
  roles:
    - nginx

  tasks:
    - name: Deploy configurazione nginx
      template:
        src: nginx.conf.j2
        dest: /etc/nginx/sites-available/default
        owner: root
        group: root
        mode: '0644'

    - name: Riavvia nginx se configurazione cambiata
      service:
        name: nginx
        state: restarted
      when: ansible_os_family == "Debian"
```

8.4 Gestione Deploy Applicativo Completo

Oltre al web server, un deploy tipico include:

- Creazione cartella applicazione
- Copia codice (modulo `git` o `copy`)
- Gestione dipendenze (es. `pip`, `npm`, `apt`)
- Configurazione variabili ambiente
- Avvio servizi (es. systemd o docker)

Esempio snippet per copia codice:

```
- name: Clona repository applicativo
  git:
    repo: 'https://github.com/tuo-progetto/app.git'
    dest: /var/www/html
    version: main
```

8.5 Gestione Segreti nel Deploy

Variabili sensibili (API key, password DB) mai hardcodate.

Esempio `group_vars/web/secrets.yml` cifrato con Vault:

```
db_password: "SuperSegreta123"
api_key: "XYZ987654321"
```

Utilizzo nel template:

```
DB_PASSWORD={{ db_password }}
API_KEY={{ api_key }}
```

Esecuzione playbook con Vault:

```
ansible-playbook playbooks/deploy_web.yml --ask-vault-pass
```

8.6 Best Practice per Deploy

- Creare ruoli per ogni componente (nginx, app, db)
- Usare template per tutte le configurazioni modificabili
- Separare variabili di ambiente da variabili di default
- Testare sempre con `--check` prima di eseguire in produzione
- Versionare codice e playbook su Git
- Automatizzare restart servizi solo se necessario (`notify` e `handlers`)

8.7 Conclusione

L’utilizzo di template Jinja2 e playbook modulare permette di:

- Standardizzare deploy
- Rendere configurazioni dinamiche e riutilizzabili
- Gestire segreti in sicurezza
- Automatizzare tutte le operazioni di setup applicativo

Nei prossimi capitoli si può approfondire:

- Hardening e sicurezza server
- Logging e troubleshooting avanzato

- Deploy multi-tier e orchestrazione

Revision #1
Created 2026-02-26 14:08:02 UTC by Pe
Updated 2026-02-26 14:08:21 UTC by Pe