

Déploiement d'un client

En bref

Niveau : intermédiaire

Durée : 45 minutes

Prérequis : tous les rôles des pages précédentes écrits ; une VM Debian 12 vierge

Niveau de test : reconstitué

Le déploiement, c'est quoi ?

Les pages précédentes ont décrit les rôles un par un. Cette page les assemble : le fichier `site.yml` complet, la procédure pour intégrer un nouveau client, et le test qui prouve que le déploiement est reproductible. C'est la page qui répond à l'objectif du projet : un client, une commande.

Élément	Rôle
<code>site.yml</code>	Playbook unique : contrôles, puis rôles dans l'ordre
<code>inventory/host_vars/<client>/</code>	Variables et secrets du client
Étiquettes (<code>tags</code>)	Permettent de rejouer une seule partie
<code>--limit</code>	Restreint l'exécution à un client

Nouvelle commande → dossier `host_vars` → ligne dans l'inventaire → `ansible-playbook site.yml --limit <client>`

Concrètement, déployer un client revient à créer ses fichiers de variables, l'ajouter à l'inventaire, et lancer le même playbook que pour tous les autres. Aucun rôle n'est modifié.

Prérequis

1. Le serveur Ansible est en place, avec `.vault_pass` (page 1).
2. Les rôles `base-hardening`, `docker-engine`, `teleport-agent`, `traefik`, `authentik`, `nextcloud`, `immich`, `vaultwarden` et `fail2ban` existent dans `roles/`.
3. La VM du client est livrée : Debian 12, utilisateur `deploy` autorisé par clé SSH, adresse du réseau d'administration.
4. Les enregistrements DNS du client pointent vers l'adresse publique de la VM.

Les secrets attendus par application

Chaque application commandée exige des secrets dans le Vault. Une table dans les variables communes les liste, pour que le playbook puisse vérifier qu'aucun ne manque avant de commencer.

```
# inventory/group_vars/all/secrets_catalogue.yml
# Secrets obligatoires : ceux du socle, puis ceux de chaque application.

nebuleuse_core_secrets:
  - vault_authentik_secret_key
  - vault_authentik_db_password
  - vault_authentik_bootstrap_password
  - vault_authentik_bootstrap_token

# Une entrée par application du catalogue, même sans secret (liste vide).
nebuleuse_app_secrets:
  nextcloud:
    - vault_nextcloud_db_password
    - vault_nextcloud_admin_password
  immich:
    - vault_immich_db_password
    - vault_immich_admin_password
  vaultwarden:
    - vault_vaultwarden_admin_token
  jellyfin: []
  papermerge: []
  homeassistant: []
  matrix: []
```

Le playbook complet

```
# site.yml

- name: Déployer un client Nébuleuse
```

```

hosts: clients

become: true          # les tâches s'exécutent en root via sudo

# Un client à la fois : une erreur sur l'un n'interrompt pas la préparation des autres.
serial: 1

pre_tasks:
  # --- Contrôles : échouer tout de suite plutôt qu'au milieu du déploiement ---
  - name: Vérifier que toutes les applications commandées existent au catalogue
    ansible.builtin.assert:
      that:
        - nebuleuse_apps | difference(nebuleuse_catalogue) | length == 0
      fail_msg: >-
        Application(s) inconnue(s) : {{ nebuleuse_apps | difference(nebuleuse_catalogue) }}
      quiet: true
    tags: always

  - name: Vérifier que le client a bien un domaine
    ansible.builtin.assert:
      that:
        - client_domain is defined
        - client_domain | length > 0
      fail_msg: "La variable client_domain doit être définie pour {{ inventory_hostname }}."
      quiet: true
    tags: always

  - name: Vérifier que les secrets nécessaires existent dans le Vault
    ansible.builtin.assert:
      that:
        # "vars" contient toutes les variables visibles pour ce client, Vault déchiffré
        compris.
        - nebuleuse_secrets_manquants | length == 0
      fail_msg: "Secrets manquants dans le Vault : {{ nebuleuse_secrets_manquants }}"
      quiet: true
    vars:
      nebuleuse_secrets_attendus: >-
        {{ nebuleuse_core_secrets +
          (nebuleuse_apps | map('extract', nebuleuse_app_secrets) | flatten | list) }}
      nebuleuse_secrets_manquants: >-
        {{ nebuleuse_secrets_attendus | reject('in', vars) | list }}
    tags: always

```

roles:

```
# 1. Socle : système, pare-feu, Docker et réseau "proxy".
- { role: base-hardening, tags: [base] }
- { role: docker-engine, tags: [docker] }

# 2. Accès administrateur par le bastion.
- { role: teleport-agent, tags: [teleport] }

# 3. Publication web et certificats.
- { role: traefik, tags: [traefik] }

# 4. SSO : toujours présent, il fait partie du socle.
- { role: authentik, tags: [sso] }

# 5. Applications : déployées uniquement si elles figurent dans la commande.
- { role: nextcloud, tags: [nextcloud], when: "'nextcloud' in nebuleuse_apps" }
- { role: immich, tags: [immich], when: "'immich' in nebuleuse_apps" }
- { role: vaultwarden, tags: [vaultwarden], when: "'vaultwarden' in nebuleuse_apps" }

# 6. Protection : en dernier, elle lit les journaux des rôles précédents.
- { role: fail2ban, tags: [fail2ban] }
```

Les contrôles portent l'étiquette `always` : ils s'exécutent même quand un seul rôle est rejoué avec `--tags`. Le premier passage d'un nouveau client doit en revanche se faire sans étiquette.

Intégrer un nouveau client

La procédure est la même pour chaque client. L'exemple prend un troisième client, de numéro 3 : adresses `10.99.0.23` et `192.0.2.23`.

```
# 1. Créer le dossier de variables à partir du client existant le plus proche.
cp -r inventory/host_vars/client-alpha inventory/host_vars/client-<nom>
```

```
# 2. Ajouter la machine à l'inventaire.
# inventory/hosts.yml (extrait)
all:
```

```
children:
  clients:
    hosts:
      client-<nom>:
        ansible_host: 10.99.0.23
```

```
# 3. Éditer vars.yml (domaine, e-mail, applications) puis vault.yml (secrets neufs).
#   Chaque client a des secrets distincts : ne jamais recopier ceux d'un autre.
ansible-vault edit inventory/host_vars/client-<nom>/vault.yml

# 4. Vérifier la connexion et la syntaxe.
ansible client-<nom> -m ping
ansible-playbook site.yml --syntax-check

# 5. Simuler : montre ce qui changerait, sans rien modifier.
ansible-playbook site.yml --limit client-<nom> --check --diff

# 6. Déployer.
ansible-playbook site.yml --limit client-<nom>
```

La simulation (`--check`) s'arrête vite sur un client vierge : plusieurs tâches dépendent de ce que les précédentes ont réellement installé. Elle est surtout utile sur un client déjà déployé, pour prévoir l'effet d'une modification.

Le test d'idempotence

Un déploiement est idempotent quand le rejouer sur un client à jour ne modifie rien. C'est le critère qui distingue un playbook fiable d'un script.

```
# Premier passage : de nombreuses tâches "changed".
ansible-playbook site.yml --limit client-<nom>

# Second passage immédiat : la ligne finale doit afficher "changed=0".
ansible-playbook site.yml --limit client-<nom>
```

Si le second passage signale des changements, la tâche concernée est à corriger : un `command` sans `changed_when`, un modèle qui change à chaque exécution (une date, une valeur aléatoire), ou un module qui ne compare pas l'état réel.

Rejouer une partie seulement

```
# Un seul service : par exemple après un changement de version de Nextcloud.
ansible-playbook site.yml --limit client-alpha --tags nextcloud

# Plusieurs services, dans l'ordre du playbook.
ansible-playbook site.yml --limit client-alpha --tags traefik,sso

# Ajouter une application à un client existant : modifier nebuleuse_apps dans vars.yml,
# puis rejouer son étiquette et celle du SSO (qui crée la connexion).
ansible-playbook site.yml --limit client-alpha --tags sso,immich,fail2ban
```

Vérification globale

```
# Sur la VM : tous les conteneurs sont "Up".
docker ps --format '{{.Names}}\t{{.Status}}'

# Les certificats sont délivrés : chaque adresse répond en HTTPS.
for h in sso cloud photos vault; do
    curl -sS -o /dev/null -w "${h}.penhoat.example : %{http_code}\n" "https://${h}.penhoat.example/"
done

# Le bastion voit la VM (depuis le bastion).
tctl nodes ls

# Les jails Fail2ban sont actives.
sudo fail2ban-client status
```

Pièges

1. **Secret manquant découvert en fin de parcours** : sans le contrôle des secrets, l'erreur « is undefined » survient après plusieurs minutes de déploiement. Le tableau `nebuleuse_app_secrets` doit être tenu à jour à chaque nouvelle application.
2. **Rejouer une application sans le SSO** : une application a besoin de la connexion créée par Authentik. Pour la première installation d'une application, inclure l'étiquette `sso`.

3. `--limit` **oublié** : sans lui, le playbook cible tous les clients, l'un après l'autre. `serial: 1` limite les dégâts, mais le temps et les effets d'une erreur s'accumulent.
4. **Secrets copiés d'un client à l'autre** : un secret commun annule l'isolation entre clients. Chaque `vault.yml` doit avoir ses propres valeurs.
5. **Certificats de test restés actifs** : si `traefik_acme_staging` est resté à `true` après les essais, les navigateurs refusent le certificat. La variable est à repasser à `false`, puis le dossier `acme` à vider avant de rejouer l'étiquette `traefik`.
6. **Idempotence qui se dégrade** : une tâche ajoutée sans `changed_when` fait échouer le test d'idempotence. Le test se rejoue après chaque modification de rôle.

Page suivante : **Pièges et dépannage.**

Revision #1

Created 2026-10-17 13:54:44 UTC by clement-derouet

Updated 2026-10-17 13:54:44 UTC by clement-derouet