

Inventaire et variables du client

Inventaire et variables, c'est quoi ?

L'inventaire dit **sur quelles machines** Ansible agit. Les variables disent **comment** configurer chacune. Dans Nébuleuse, la commande d'un client se résume à un dossier de variables : le code (les rôles) ne change jamais d'un client à l'autre.

Emplacement	Contenu	Chiffré
<code>inventory/hosts.yml</code>	Liste des machines et leur adresse d'administration	Non
<code>inventory/group_vars/all/vars.yml</code>	Valeurs communes à tous les clients	Non
<code>inventory/host_vars/<client>/vars.yml</code>	Valeurs propres à un client, dont la liste des applications	Non
<code>inventory/host_vars/<client>/vault.yml</code>	Secrets du client	Oui (Ansible Vault)

Commande web → `vars.yml` **du client** → `nebuleuse_apps` → **rôles activés par** `site.yml`

Concrètement, le formulaire de commande génère deux fichiers dans `host_vars/<client>/` : `vars.yml` (le domaine et les applications choisies) et `vault.yml` (des secrets générés aléatoirement, puis chiffrés). Un nouveau client, c'est donc un dossier de plus, sans modifier de rôle.

Prérequis

1. Le serveur Ansible est en place (page précédente), avec `.vault_pass` créé.
2. L'inventaire contient `client-alpha` dans le groupe `clients`.

Précédence des variables

Une même variable peut être définie à plusieurs niveaux. Ansible retient la valeur du niveau le plus fort. Dans ce projet, seuls ces niveaux sont utilisés, du plus faible au plus fort :

1. Les valeurs par défaut du rôle (`roles/<rôle>/defaults/main.yml`)
2. Les variables communes (`group_vars/all`)
3. Les variables du client (`host_vars/<client>`)
4. Les variables passées en ligne de commande (`-e`)

Les valeurs par défaut portent donc les choix raisonnables, `group_vars` les choix de l'hébergeur, et `host_vars` ce qui est propre au client.

Les variables communes

```
# inventory/group_vars/all/vars.yml
# Valeurs partagées par tous les clients de Nébuleuse. Aucun secret ici.

# Domaine de l'hébergeur : sert au bastion et aux adresses techniques.
nebuleuse_domain: nebuleuse.example

# Adresse e-mail déclarée auprès de l'autorité de certification (Let's Encrypt).
# Elle reçoit les alertes d'expiration de certificat.
nebuleuse_acme_email: certificats@nebuleuse.example

# Réseau Docker partagé entre Traefik et les applications.
docker_proxy_network: proxy

# Réseau d'administration : seul à pouvoir joindre le port SSH.
admin_network_cidr: 10.99.0.0/24

# Adresse du bastion Teleport (proxy) auquel chaque agent se rattache.
teleport_proxy_addr: bastion.nebuleuse.example:443

# Catalogue : applications que Nébuleuse sait déployer.
# Une application commandée doit figurer ici, sinon le déploiement s'arrête.
nebuleuse_catalogue:
  - nextcloud
  - immich
  - vaultwarden
  - jellyfin
```

- papermerge
- homeassistant
- matrix

Les variables d'un client

```
# inventory/host_vars/client-alpha/vars.yml
# Généré à partir de la commande du client. Aucun secret ici.

# Identifiant court du client : sert de préfixe aux conteneurs (alpha_nextcloud...).
client_name: alpha

# Domaine du client : chaque application aura un sous-domaine.
client_domain: alpha.nebuleuse.example

# Contact administrateur du client (compte initial du SSO, notifications).
client_admin_email: admin@alpha.nebuleuse.example

# Applications commandées. C'est la seule variable qui pilote ce qui est déployé.
nebuleuse_apps:
  - nextcloud
  - immich
  - vaultwarden

# Sous-domaines de chaque service. Le SSO est toujours présent (socle).
client_subdomains:
  sso: sso
  nextcloud: cloud
  immich: photos
  vaultwarden: vault
```

Les secrets d'un client

Les secrets suivent une convention : chaque secret est défini dans `vault.yml` sous un nom préfixé par `vault_`, puis référencé dans `vars.yml` ou dans un rôle sous son nom normal. Ainsi, on repère d'un coup d'œil les variables sensibles, et on voit où une valeur est utilisée sans déchiffrer quoi que ce soit.

```
# inventory/host_vars/client-alpha/vault.yml (AVANT chiffrement)
# Toutes les valeurs ci-dessous sont fictives.

vault_authentik_secret_key: "FICTIF-ab12cd34ef56ab12cd34ef56ab12cd34ef56"
vault_authentik_db_password: "FICTIF-db-motdepasse-1"
vault_nextcloud_db_password: "FICTIF-db-motdepasse-2"
vault_nextcloud_admin_password: "FICTIF-admin-motdepasse-3"
vault_immich_db_password: "FICTIF-db-motdepasse-4"
vault_vaultwarden_admin_token: "FICTIF-token-admin-5"
```

```
# Chiffrement du fichier avec le mot de passe de .vault_pass (voir ansible.cfg).
ansible-vault encrypt inventory/host_vars/client-alpha/vault.yml

# Pour modifier plus tard : ouvre le fichier déchiffré dans l'éditeur, puis le rechiffre à la
sauvegarde.
ansible-vault edit inventory/host_vars/client-alpha/vault.yml
```

Après chiffrement, le fichier commence par `$ANSIBLE_VAULT;1.1;AES256` suivi d'un bloc illisible. Il peut être versionné sans risque tant que le mot de passe du Vault reste secret.

Dans un rôle, le secret est consommé sous son nom propre :

```
# roles/authentik/defaults/main.yml
# Le rôle ne connaît que "authentik_secret_key" : il ignore d'où vient la valeur.
authentik_secret_key: "{{ vault_authentik_secret_key }}"
authentik_db_password: "{{ vault_authentik_db_password }}"
```

Vérifier la commande avant de déployer

Un garde-fou en début de playbook arrête le déploiement si le client commande une application inconnue ou oublie un secret. Mieux vaut échouer tout de suite que découvrir l'erreur au milieu d'un déploiement.

```
# site.yml (extrait : tâches de contrôle)
pre_tasks:
  - name: Vérifier que toutes les applications commandées existent au catalogue
```

```

ansible.builtin.assert:
  that:
    # "difference" retourne les éléments de la commande absents du catalogue.
    - nebuleuse_apps | difference(nebuleuse_catalogue) | length == 0
  fail_msg: >-
    Application(s) inconnue(s) : {{ nebuleuse_apps | difference(nebuleuse_catalogue) }}
  quiet: true

- 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

```

Vérification

```

# Affiche toutes les variables fusionnées d'un client (le Vault est déchiffré).
# Attention : la sortie contient les secrets en clair, ne pas la partager.
ansible-inventory --host client-alpha --yaml

# Contrôle la syntaxe du playbook sans rien exécuter.
ansible-playbook site.yml --syntax-check

```

Pour s'assurer qu'aucun fichier de secrets n'est resté en clair avant un commit :

```

# Liste les vault.yml qui NE commencent PAS par l'en-tête de chiffrement.
# Aucune ligne ne doit s'afficher.
grep -L '^$ANSIBLE_VAULT' inventory/host_vars/*/vault.yml

```

Pièges

1. **Variable** `vault_...` **non définie** : un secret référencé dans un rôle mais absent de `vault.yml` provoque « is undefined » au milieu du déploiement. Les tâches de contrôle ci-dessus évitent ce cas pour les variables essentielles.

2. **Un fichier** `host_vars/client-alpha.yml` **et un dossier** `host_vars/client-alpha/` : les deux sont lus, ce qui rend l'origine d'une valeur difficile à retrouver. Il faut choisir un seul format, ici le dossier.
3. **Secret affiché dans un journal** : une tâche qui manipule un secret doit porter `no_log: true`, sinon la valeur apparaît dans la sortie d'Ansible.
4. **Fichier versionné avant chiffrement** : si `vault.yml` est commité en clair, le secret reste dans l'historique Git même après chiffrement. Il faut alors le renouveler, pas seulement chiffrer le fichier.
5. **Mot de passe du Vault perdu** : les fichiers sont illisibles et les secrets doivent être régénérés. D'où la sauvegarde hors serveur recommandée à la page précédente.

Page suivante : **Articulation des services.**

Revision #1

Created 2026-10-15 07:17:34 UTC by clement-derouet

Updated 2026-10-15 07:17:55 UTC by clement-derouet