

# Articulation des services

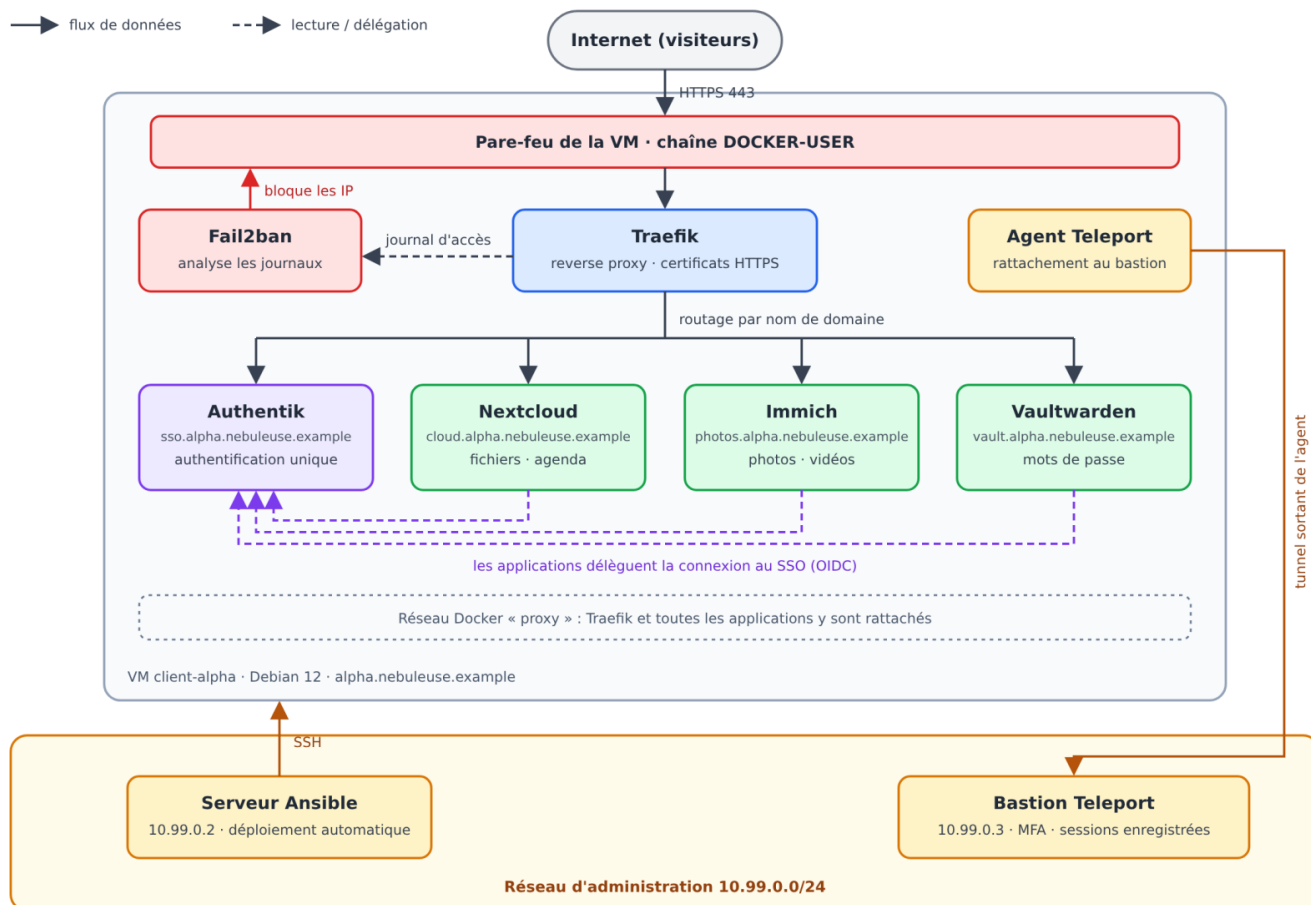
## Pourquoi cette page ?

Les services d'une VM Nébuleuse ne sont pas indépendants. Une application attend que le SSO existe, le SSO attend que le reverse proxy le publie, et le pare-feu applicatif lit les journaux des deux. Cette page donne la vue d'ensemble : qui dépend de qui, ce que chaque service doit savoir des autres, et dans quel ordre Ansible doit les installer. Les pages suivantes détaillent chaque service ; celle-ci explique pourquoi ils s'emboîtent.

| Service        | Rôle dans l'ensemble                                                                     |
|----------------|------------------------------------------------------------------------------------------|
| Traefik        | Reçoit tout le trafic web et le dirige selon le nom de domaine                           |
| Authentik      | Identifie les utilisateurs pour toutes les applications                                  |
| Applications   | S'appuient sur Traefik pour être publiées, sur Authentik pour connecter les utilisateurs |
| Fail2ban       | Lit les journaux et bloque les adresses fautives                                         |
| Agent Teleport | Rattache la VM au bastion, indépendamment du reste                                       |

**Utilisateur → Traefik → Application → redirection vers Authentik → retour à l'application connecté**

## Vue d'ensemble



# Les relations, service par service

Chaque ligne ci-dessous est une information qu'un service doit connaître d'un autre. Si l'une d'elles est fausse, la fonctionnalité correspondante ne marche pas, et le symptôme est rarement évident. C'est pourquoi ces valeurs sont des variables partagées, définies une seule fois.

| Service            | A besoin de                         | Information échangée                                              |
|--------------------|-------------------------------------|-------------------------------------------------------------------|
| Traefik            | Le réseau Docker <code>proxy</code> | Nom du réseau, pour trouver les conteneurs                        |
| Traefik            | Un e-mail ACME                      | Pour obtenir les certificats HTTPS                                |
| Chaque application | Le réseau <code>proxy</code>        | Pour être joignable par Traefik                                   |
| Chaque application | Des labels Traefik                  | Nom de domaine ( <code>Host(...)</code> ) et port interne         |
| Chaque application | Authentik                           | Adresse de l'émetteur (issuer), identifiant client, secret client |
| Authentik          | Chaque application                  | Adresse de retour (redirect URI) exacte                           |
| Fail2ban           | Le journal d'accès de Traefik       | Chemin du fichier, format des lignes                              |

| Service        | A besoin de                        | Information échangée                   |
|----------------|------------------------------------|----------------------------------------|
| Fail2ban       | La chaîne <code>DOCKER-USER</code> | Le point d'insertion des blocages      |
| Agent Teleport | Le bastion                         | Adresse du proxy et jeton d'enrôlement |

Deux relations demandent une attention particulière :

1. **L'adresse de retour (redirect URI)** doit être identique, caractère pour caractère, dans Authentik et dans l'application. Un `/` final en trop suffit à provoquer une erreur de connexion.
2. **Les adresses réelles des visiteurs** : les applications ne voient que l'adresse de Traefik, sauf si elles sont configurées pour lire l'en-tête `X-Forwarded-For` envoyé par un proxy de confiance. Sans cela, les journaux ne contiennent que l'adresse interne, et Fail2ban bloquerait la mauvaise adresse, ou aucune.

# Ordre de déploiement

L'ordre des rôles dans `site.yml` n'est pas arbitraire. Chaque étape suppose que la précédente est terminée.

1. **Socle et Docker** : le système est à jour, Docker est installé, le réseau `proxy` existe et la chaîne `DOCKER-USER` est disponible.
2. **Agent Teleport** : l'accès administrateur est établi avant les services applicatifs. Si un déploiement ultérieur échoue, la machine reste accessible par le bastion.
3. **Traefik** : il écoute sur les ports 80 et 443 et obtient les certificats. Il doit exister avant les applications, qui déclarent leurs routes par labels.
4. **Authentik** : une fois démarré derrière Traefik, son API est joignable. Les applications en ont besoin pour créer leurs connexions SSO.
5. **Applications commandées** : chacune crée sa connexion dans Authentik (API), récupère l'identifiant et le secret, puis génère son fichier Compose.
6. **Fail2ban** : il vient en dernier, car ses règles pointent vers des journaux qui n'existent qu'une fois Traefik et les applications démarrés.

```
# site.yml (extrait : ordre d'application des rôles)
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] }
```

Note — La condition `when` d'un rôle s'applique à chacune de ses tâches : si l'application n'est pas commandée, aucune tâche du rôle ne s'exécute.

## Comment une valeur voyage d'un service à l'autre

Un identifiant client créé par Authentik doit se retrouver dans le fichier Compose de l'application. Il ne vient pas d'une variable écrite à l'avance : il est créé pendant le déploiement. Ansible le récupère donc dans la réponse de l'API, le mémorise, puis le fournit au modèle de fichier.

**Appel API Authentik → réponse (identifiant, secret) → variable de la tâche suivante → modèle `docker-compose.yml.j2` → fichier sur la VM**

C'est ce qui oblige à placer Authentik avant les applications, et à rendre ces appels rejouables : si la connexion existe déjà, la tâche récupère ses valeurs au lieu de la recréer. Le détail est donné dans la page Authentik.

## Attendre qu'un service soit prêt

Démarrer un conteneur ne signifie pas que le service répond. Authentik met plusieurs dizaines de secondes à devenir joignable, et une tâche qui l'appelle trop tôt échoue sans raison valable. Le rôle attend donc que l'adresse de santé réponde, en réessayant à intervalles réguliers.

```
# roles/authentik/tasks/main.yml (extrait)
- name: Attendre que l'API d'Authentik réponde
  # curl -f : retourne une erreur si le serveur répond autre chose qu'un succès.
  # -s -S : silencieux, mais affiche les erreurs.
  ansible.builtin.command: >-
    curl -fsS https://sso.{{ client_domain }}/-/health/live/
  register: authentik_health
  # Réessaie jusqu'à ce que la commande réussisse : 30 essais, 10 secondes d'écart.
  until: authentik_health.rc == 0
  retries: 30
  delay: 10
  # Ne modifie rien : l'indicateur "changed" n'a pas de sens ici.
  changed_when: false
```

## Vérification

```
# Depuis le serveur Ansible : liste l'ordre exact des tâches pour un client,
# sans rien exécuter, pour contrôler que les rôles sont enchaînés comme prévu.
ansible-playbook site.yml --limit client-alpha --list-tasks

# Sur la VM : tous les conteneurs doivent être "Up" et sur le réseau "proxy".
docker ps --format '{{.Names}}\t{{.Status}}'
docker network inspect proxy --format '{{range .Containers}}{{.Name}} {{end}}'
```

## Pièges

1. **Dépendance circulaire apparente** : Authentik est publié par Traefik, et les applications ont besoin d'Authentik. Il n'y a pas de cycle tant que l'ordre Traefik, puis Authentik, puis applications est respecté.
2. **Rejouer une partie seulement** : relancer uniquement le tag d'une application ne recrée pas les éléments de l'étape précédente. Si une valeur de Traefik ou d'Authentik change, il faut relancer aussi les rôles qui en dépendent.
3. **Fail2ban lancé trop tôt** : une jail dont le fichier journal n'existe pas ne démarre pas. D'où sa position en dernier.
4. **Résolution du nom du SSO depuis un conteneur** : une application doit contacter `sso.<domaine>` depuis l'intérieur de la VM. Selon la manière dont l'hébergeur attache l'adresse publique, ce chemin peut ne pas aboutir. Le cas est traité dans la page Traefik.

Revision #2

Created 2026-10-15 07:18:13 UTC by clement-derouet

Updated 2026-10-15 07:50:23 UTC by clement-derouet