

Socle système et Docker Engine

Le socle, c'est quoi ?

Le socle est ce que reçoit toute VM Nébuleuse avant la moindre application : un système à jour, un accès SSH restreint, un pare-feu, puis Docker avec son réseau partagé. Il est porté par deux rôles, `base-hardening` et `docker-engine`, appliqués dans cet ordre.

Rôle	Rôle dans le socle
<code>base-hardening</code>	Mises à jour, SSH restreint au réseau d'administration, pare-feu <code>ufw</code>
<code>docker-engine</code>	Docker depuis le dépôt officiel, journaux limités, réseau <code>proxy</code>

VM Debian 12 vierge → `base-hardening` → `docker-engine` → **machine prête à recevoir Traefik**

Concrètement, la VM livrée par le provisioning possède déjà un utilisateur `deploy` autorisé par clé SSH et membre de `sudo` sans mot de passe. Le socle ne crée pas cet utilisateur : il en dépend, et il le verrouille comme seul compte autorisé à se connecter.

Prérequis

1. Les pages précédentes sont en place : inventaire, variables communes (`admin_network_cidr`) et `ansible ... -m ping` fonctionnel.
2. L'utilisateur `deploy` se connecte par clé et dispose de `sudo` sans mot de passe.
3. L'adresse `ansible_host` de chaque client est son adresse du réseau d'administration, car c'est celle sur laquelle SSH écoutera.

Arborescence des deux rôles

```
roles/
├─ base-hardening/
│   ├─ defaults/main.yml
│   ├─ handlers/main.yml
│   └─ tasks/main.yml
└─ templates/sshd-nebuleuse.conf.j2
├─ docker-engine/
│   ├─ defaults/main.yml
│   ├─ handlers/main.yml
│   └─ tasks/main.yml
└─ templates/daemon.json.j2
```

Le rôle `base-hardening`

Variables

```
# roles/base-hardening/defaults/main.yml

# Seul utilisateur autorisé à ouvrir une session SSH.
base_ssh_user: deploy

# Adresse sur laquelle SSH écoute : celle du réseau d'administration.
# "ansible_host" vaut cette adresse dans l'inventaire (voir page Inventaire).
base_ssh_listen_address: "{{ ansible_host }}"

# Réseau autorisé à joindre SSH (défini une fois pour tous dans group_vars/all).
base_admin_network_cidr: "{{ admin_network_cidr }}"

# Ports ouverts à tout Internet : uniquement le web.
base_public_tcp_ports:
  - 80
  - 443
```

Configuration SSH

```
# roles/base-hardening/templates/sshd-nebuleuse.conf.j2
# Fichier g  r   par Ansible : toute modification manuelle sera   cras  e.

# Pas de connexion directe en root.
PermitRootLogin no

# Authentification par cl   uniquement : aucun mot de passe n'est accept  .
PasswordAuthentication no
KbdInteractiveAuthentication no
PubkeyAuthentication yes

# Un seul compte peut se connecter.
AllowUsers {{ base_ssh_user }}

# SSH n'  coute que sur l'adresse d'administration, jamais sur l'adresse publique.
ListenAddress {{ base_ssh_listen_address }}
```

Note — Sous Debian 12, le fichier principal `/etc/ssh/sshd_config` commence par une directive `Include /etc/ssh/sshd_config.d/*.conf`. Pour chaque param  tre, SSH retient la premi  re valeur lue : le fichier d  pos   dans ce dossier l'emporte donc sur le fichier principal, sans avoir    le modifier.

T  ches

```
# roles/base-hardening/tasks/main.yml

- name: Mettre    jour l'index des paquets et les paquets install  s
  ansible.builtin.apt:
    update_cache: true
    # "dist"   quivaut    "apt dist-upgrade" : g  re aussi les changements de d  pendances.
    upgrade: dist
    # Ne relit l'index qu'au bout d'une heure :   vite un t  l  chargement inutile    chaque
    passage.
    cache_valid_time: 3600

- name: Installer les paquets du socle
  ansible.builtin.apt:
    name:
```

```

- ufw                                # pare-feu de l'hôte
- unattended-upgrades               # installation automatique des mises à jour de sécurité
- ca-certificates                   # certificats racine pour les téléchargements HTTPS
- curl                              # utilisé par d'autres rôles
state: present

- name: Activer les mises à jour de sécurité automatiques
  ansible.builtin.copy:
    dest: /etc/apt/apt.conf.d/20auto-upgrades
    content: |
      APT::Periodic::Update-Package-Lists "1";
      APT::Periodic::Unattended-Upgrade "1";
    owner: root
    group: root
    mode: "0644"

- name: Déposer la configuration SSH durcie
  ansible.builtin.template:
    src: sshd-nebuleuse.conf.j2
    dest: /etc/ssh/sshd_config.d/10-nebuleuse.conf
    owner: root
    group: root
    mode: "0644"

    # Vérifie la syntaxe avant d'écraser le fichier : une erreur ici couperait l'accès.
    validate: /usr/sbin/sshd -t -f %s
  notify: Redémarrer ssh

# Le pare-feu : l'ordre compte. La règle SSH est créée AVANT l'activation
# du pare-feu, sinon la connexion en cours serait coupée.

- name: Refuser par défaut tout trafic entrant
  community.general.ufw:
    default: deny
    direction: incoming

- name: Autoriser SSH depuis le réseau d'administration uniquement
  community.general.ufw:
    rule: allow
    port: "22"
    proto: tcp
    src: "{{ base_admin_network_cidr }}"

```

```
- name: Autoriser le web (HTTP et HTTPS) depuis Internet
  community.general.ufw:
    rule: allow
    port: "{{ item }}"
    proto: tcp
    loop: "{{ base_public_tcp_ports }}"

- name: Activer ufw
  community.general.ufw:
    state: enabled
```

```
# roles/base-hardening/handlers/main.yml
# Un handler ne s'exécute qu'à la fin des tâches, et seulement si une tâche l'a notifié.
- name: Redémarrer ssh
  ansible.builtin.service:
    name: ssh          # le service s'appelle "ssh" sous Debian, pas "sshd"
    state: restarted
```

Le rôle `docker-engine`

Variables

```
# roles/docker-engine/defaults/main.yml

# Nom du réseau partagé entre Traefik et les applications (voir group_vars/all).
docker_engine_proxy_network: "{{ docker_proxy_network }}"

# Taille maximale d'un fichier de journal de conteneur, et nombre de fichiers conservés.
# Sans limite, les journaux d'un conteneur bavard finissent par remplir le disque.
docker_engine_log_max_size: "10m"
docker_engine_log_max_files: "3"
```

Configuration du démon

```
# roles/docker-engine/templates/daemon.json.j2
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "{{ docker_engine_log_max_size }}",
    "max-file": "{{ docker_engine_log_max_files }}"
  },
  "live-restore": true
}
```

Le paramètre `live-restore` garde les conteneurs en marche pendant un redémarrage ou une mise à jour du démon Docker : une mise à jour du moteur ne coupe donc pas les services des clients.

Tâches

```
# roles/docker-engine/tasks/main.yml

- name: Créer le dossier des clés de dépôt
  ansible.builtin.file:
    path: /etc/apt/keyrings
    state: directory
    mode: "0755"

- name: Récupérer la clé de signature du dépôt Docker
  ansible.builtin.get_url:
    url: https://download.docker.com/linux/debian/gpg
    dest: /etc/apt/keyrings/docker.asc
    mode: "0644"

- name: Déclarer le dépôt officiel Docker
  ansible.builtin.apt_repository:
    # "signed-by" lie ce dépôt à sa clé : les paquets sont vérifiés avec elle.
    # ansible_distribution_release vaut "bookworm" sur Debian 12.
    repo: >-
      deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.asc]
        https://download.docker.com/linux/debian {{ ansible_distribution_release }} stable
    filename: docker
    state: present
```

```

- name: Installer Docker Engine et le plugin Compose
  ansible.builtin.apt:
    name:
      - docker-ce
      - docker-ce-cli
      - containerd.io
      - docker-buildx-plugin
      - docker-compose-plugin    # fournit la commande "docker compose"
      - python3-docker           # bibliothèque Python requise par les modules
community.docker
  state: present
  update_cache: true

- name: Déposer la configuration du démon Docker
  ansible.builtin.template:
    src: daemon.json.j2
    dest: /etc/docker/daemon.json
    owner: root
    group: root
    mode: "0644"
  notify: Redémarrer docker

- name: Activer et démarrer Docker
  ansible.builtin.service:
    name: docker
    state: started
    enabled: true

# Les handlers sont lancés ici plutôt qu'en fin de rôle : le réseau ci-dessous
# doit être créé sur un démon déjà configuré.
- name: Appliquer les changements de configuration du démon
  ansible.builtin.meta: flush_handlers

- name: Créer le réseau partagé entre Traefik et les applications
  community.docker.docker_network:
    name: "{{ docker_engine_proxy_network }}"
    driver: bridge

```

```

# roles/docker-engine/handlers/main.yml

- name: Redémarrer docker

```

```
ansible.builtin.service:
  name: docker
  state: restarted
```

Note — L'utilisateur `deploy` n'est volontairement pas ajouté au groupe `docker`. Appartenir à ce groupe revient à avoir les droits root sur la machine. Ansible utilise déjà `sudo`, ce qui laisse une trace de chaque commande.

Docker et le pare-feu : ce qu'il faut savoir

Docker gère lui-même ses règles de pare-feu. Quand un conteneur publie un port (comme Traefik sur 80 et 443), Docker l'ouvre directement, sans passer par `ufw`. Ainsi, une règle `ufw` ne bloque pas un port publié par un conteneur.

Ce comportement est connu, et il explique deux choix du projet : 1. Seul Traefik publie des ports vers l'extérieur. Les autres conteneurs restent sur le réseau `proxy`, sans port publié. 2. Les blocages dynamiques (Fail2ban) ne passent pas par `ufw` mais par la chaîne `DOCKER-USER`. Docker crée cette chaîne et ne la modifie jamais ; les règles qu'on y place sont donc appliquées avant celles de Docker.

Le playbook du socle

Pour tester dès maintenant, un `site.yml` minimal suffit. Il sera complété à la page de déploiement.

```
# site.yml
- name: Socle Nébuleuse
  hosts: clients
  become: true          # les tâches s'exécutent en root via sudo
  roles:
    - { role: base-hardening, tags: [base] }
    - { role: docker-engine, tags: [docker] }
```

```
# Simulation : montre ce qui changerait, sans rien modifier.
ansible-playbook site.yml --limit client-alpha --check --diff
```

```
# Déploiement réel.  
ansible-playbook site.yml --limit client-alpha
```

Vérification

```
# Depuis le serveur Ansible : un second passage ne doit rien modifier.  
# La ligne finale doit afficher "changed=0".  
ansible-playbook site.yml --limit client-alpha  
  
# Sur la VM : valeurs SSH effectives.  
sudo sshd -T | grep -E 'listenaddress|passwordauthentication|permitrootlogin|allowusers'  
  
# Sur la VM : règles du pare-feu.  
sudo ufw status verbose  
  
# Sur la VM : version de Docker, pilote de journaux et live-restore.  
docker version --format '{{.Server.Version}}'  
docker info --format 'journaux={{.LoggingDriver}} live-restore={{.LiveRestoreEnabled}}'  
  
# Sur la VM : le réseau partagé existe et la chaîne DOCKER-USER est disponible.  
docker network ls --filter name=proxy  
sudo iptables -L DOCKER-USER -n
```

Résultat attendu pour `sshd -T` : `listenaddress 10.99.0.21:22`, `passwordauthentication no`, `permitrootlogin no`, `allowusers deploy`.

Pièges

1. **Se couper l'accès en modifiant SSH** : une erreur de `ListenAddress` ou une règle de pare-feu oubliée coupe la connexion. La validation `sshd -t` et l'ordre « règle avant activation » limitent le risque. Avant de fermer une session de test, ouvrir une seconde connexion pour vérifier que l'accès fonctionne encore.
2. `ansible_host` **qui n'est pas l'adresse d'administration** : SSH n'écouterait pas là où Ansible se connecte. L'inventaire de la première page respecte la règle ; une machine ajoutée avec son adresse publique fausserait le rôle.
3. `apt` **bloqué par un autre processus** : juste après le démarrage d'une VM neuve, `unattended-upgrades` peut tenir le verrou `apt`. Ansible patiente un moment, puis échoue avec « Could not get lock ». Relancer le playbook suffit.

4. **Pare-feu `ufw` qui « ne bloque rien » pour un conteneur** : voir la section précédente.
Ce n'est pas un défaut de configuration.
5. **`docker_network` en erreur sur la bibliothèque Python** : le paquet `python3-docker` doit être installé avant toute tâche `community.docker`. Il fait partie de la liste des paquets du rôle.
6. **Journaux de conteneurs qui remplissent le disque** : sans `max-size`, aucune rotation n'est faite. La configuration `daemon.json` ne s'applique qu'aux conteneurs créés après elle ; les conteneurs existants doivent être recréés.

Page suivante : **Bastion Teleport : agent et enrôlement.**

Revision #1

Created 2026-10-15 11:13:58 UTC by clement-derouet

Updated 2026-10-15 11:15:39 UTC by clement-derouet