Documentation

- Architecture DevOps : Refonte CI/CD et Gestion de Configuration

? Architecture DevOps : Refonte CI/CD et Gestion de Configuration

Auteur : Administrateur Homelab
Date de révision : Avril 2026

---

1. Le Changement de Paradigme : Pourquoi cette refonte ?

Historiquement, notre cluster Kubernetes (via Tekton) gérait l'intégralité de la chaîne logicielle : de la fabrication des images Docker jusqu'à l'exécution de scripts Ansible via des Pods éphémères pour mettre à jour des serveurs. Cela créait plusieurs problèmes :

- **Surcharge de K8s :** Création de Pods, PVC, et gestion de secrets lourds juste pour lancer un `apt-get update` ou un `rsync`.
- **Manque de visibilité :** Les logs d'exécution Ansible étaient perdus dès la destruction du Pod Tekton.
- **Mauvaise gestion du temps :** Les tâches planifiées (Cron) saturaient l'API Kubernetes.

La Solution : La séparation des responsabilités (Séparation CI / CD).

- **Tekton (CI - Continuous Integration) :** Son rôle strict est désormais de "fabriquer". Il écoute Git, clone le code, compile, et pousse des artefacts (ex: images Docker via Kaniko).
- ⚙ **AWX (CD - Continuous Deployment / Configuration) :** Son rôle est de "déployer et configurer". Il détient les clés SSH, se connecte aux machines (Proxmox, Synology, HA) et applique les Playbooks Ansible.

---

2. L'Organisation Ansible : Le "Clean Code"

Pour supporter cette architecture, l'inventaire monolithique a été découpé selon les meilleures pratiques (DRY - Don't Repeat Yourself) en exploitant la **pyramide de préséance d'Ansible**.

La pyramide de préséance (Du plus faible au plus fort) :

1. **Defaults des Rôles (Niveau 1) :** `roles/mon_role/defaults/main.yml`. Ex: `allow_reboot: false`. C'est la sécurité par défaut.
2. **Variables de Groupes (Niveau 2) :** `group_vars/*.yml`. Ex: `ansible_user: root` pour tout le monde, écrasé par `ansible_user: nancre` dans `group_vars/vm.yml`.
3. **Variables d'Hôtes (Niveau 3) :** `host_vars/nom_host.yml`. Ex: Port SSH `22022` spécifique au NAS Synology.
4. **Variables AWX (Niveau 4 - God Mode) :** Paramètres passés via l'interface AWX (Extra Vars) lors d'un lancement manuel pour écraser toute autre règle.

Structure de l'Inventaire (Aperçu)

```
all:
  children:
    proxmox_hypervisors:
      hosts:
        proxmox-server:
          ansible_host: 10.151.151.249
        proxmox-ms:
          ansible_host: 10.151.151.239
    # ... Les variables spécifiques sont désormais dans group_vars/ et host_vars/
```

---

3. Les Workflows Clés : Comment on fait ?

A. La Synchronisation Événementielle (Ex: Homer)

Logique : Quand un fichier de configuration change sur Git, Tekton s'en aperçoit, mais délègue le déploiement à AWX.

- **Git :** Push sur `infra/kubernetes/homer/config/`.
- **Tekton :** Le Trigger s'active. La Task appelle la commande générique `trigger-awx-job` avec l'ID du Job Template.
- **AWX :** Réceptionne l'API call, lance le Playbook `sync_homer_ha.yml`, et se connecte en SSH au Home Assistant (port 22) avec son *Machine Credential* chiffré pour copier les fichiers dans `/addon_configs/`.

B. Les Tâches Planifiées (Ex: Maintenance OS & Docker Prune)

Logique : On supprime les `CronJobs` de Kubernetes. Le *Scheduler* natif d'AWX prend le relais.

- **AWX (Job Template) :** Pointage sur `maintenance_update_os.yml` ou `maintenance_docker_prune.yml`.
- **Schedules :** Planifiés nativement via l'UI AWX (ex: Tous les dimanches à 23h00). Attention au bug d'interface d'AWX lié au fuseau horaire (UTC vs Europe/Paris) : il faut définir l'heure de départ (DTSTART) et laisser l'heure de la règle (RRULE) vide, ou passer par un appel API `curl` en cas de blocage.
- **Robuste (Multi-Nœuds) :** Le rôle de mise à jour détecte dynamiquement sur quel hyperviseur tourne une VM (grâce à la variable `pve_node`) pour exécuter les commandes d'API de secours (ex: `qm reset`) en ciblant le bon hôte Proxmox ou Synology.

C. La Fabrication pure (Ex: code-server)

Logique : Reste 100% dans Kubernetes, car c'est du "Build" (CI).

- **Git :** Création et Push d'un Tag (ex: `v2.8`).
- **Tekton :**
  1. Utilise `custom-git-clone` pour récupérer le `Dockerfile` avec les clés SSH.
  2. Utilise la Task `kaniko` pour compiler l'image dans un Pod (utilise les ressources K8s, sans daemon Docker).
  3. Pousse l'image vers la registry privée Harbor : `docker-registry.numericare.fr/private/code-server:v2.8`.

---

4. Bilan et Bénéfices

- **Baisse de la charge K8s :** Moins de pods éphémères, moins de montages de PVC inutiles.

- □□ **Sécurité renforcée :** Les clés SSH et les accès NAS/Proxmox sont chiffrés et gérés par le gestionnaire d'identifiants d'AWX, plus dans des ConfigMaps/Secrets K8s éparpillés.
- □□ **Observabilité :** Chaque exécution Ansible a désormais une interface visuelle, un historique, et un rapport d'erreurs détaillé dans AWX.