Vers une démarche qualité : le Runbook

🖥️ Workflow de création de VM

🏅 Une approche qualité pour l'infrastructure

📋 Le Runbook permet la traçabilité des interventions

Contexte et objectifs

- 🎯 **Objectif** : Mettre à disposition des VM à des utilisateurs
  - Chaque utilisateur peut avoir une ou plusieurs VM
  - Chaque VM est attribuée à un utilisateur (ou un groupe)

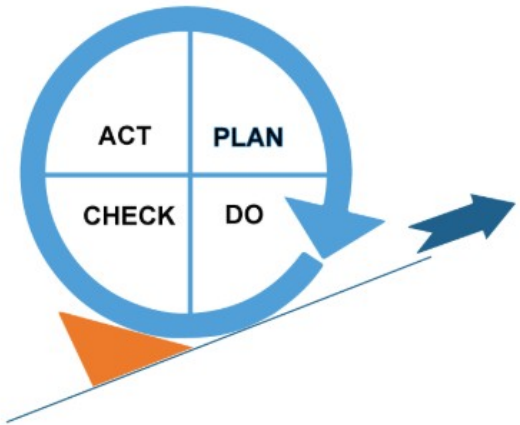
- 🚧 **Contraintes** (exigences qualité) :
  - Monitoring (NUT, Munin)
  - Sauvegarde (UrBackup)
  - NTP, locale, exim client
  - Client GLPI (remontée DSIN)
  - Outils communs standard

- 🎁 **Facilités** (pour les utilisateurs) :
  - Compte automatique avec clé SSH
  - Groupes Ansible pré-paramétrés
  - bashrc fonctionnel, ...
  - Descriptions synchronisées sur VM et sur la GUI de PVE

Mise en place d'un workflow ?

📐 La roue de Deming (PDCA)

- **Plan** : définir le processus (ici, notre workflow)
- **Do** : exécuter (créer des VM, les configurer)
- **Check** : vérifier (évaluer l'état des VM)
- **Act** : agir pour améliorer le processus (les interventions)



🔒 La « cale » qui empêche de reculer

- La **traçabilité** (écrire ce qu'on fait) est la cale
- Elle verrouille les progrès et empêche de revenir en arrière
- Permet de mesurer, d'analyser et d'optimiser en continu

Objectif : s'inscrire dans une démarche d'**amélioration continue**

Le Runbook : le cœur du processus

- 🌐 Application web Symfony + PostgreSQL

- 🛢️ Une base de données centralisée pour :
  - Déclarer et décrire les machines (*machine*)
  - Gérer les utilisateurs (*utilisateur*)
  - Gérer les groupes (*groupe*)
  - Suivre les **interventions** (*intervention*)

- 🔑**Fonction clé** : génération des fichiers YAML pour Ansible
  - Inventaire dynamique
  - Groupes Ansible
  - Playbooks de post-installation

➡️ C'est le point central qui permet de TOUT tracer

Les VM sont créées à partir d'un template cloud-init

- 🚀 **Cloud-init** = standard d'initialisation dans les environnements cloud
  - Injection d'IP, hostname, clé SSH au **démarrage**
  - Réduction des interventions manuelles

- 🖥️ **Intégration Proxmox** : support natif dans la GUI

- 🏅 **Avantages qualité** :
  - Standardisation des VM
  - Reproductibilité
  - Diminution des erreurs humaines (saisie IP dans la GUI, exit la console)

📌 **Procédure** :

1. Clone du template
2. Saisie de l'adresse IP dans cloud-init (GUI Proxmox)
3. Démarrage → configuration automatique

Vue d'ensemble du workflow

```
A["👤 Administrateur"]

  A -->|"① Déclare"| B["📋 Runbook<br/>IP + Utilisateur + Groupes Ansible<br/>+ Description"]
  A -->|"② Clone"| C["🖥️ Proxmox GUI<br/>Template Debian 13 cloud-init"]

  B -->|"📌 IP sélectionnée"| C
  B --> D["📁 Fichiers YAML<br/>- inventaire<br/>- groupes Ansible"]
  C -->|"③ Saisie IP + Démarre"| E["⚙️ VM démarrée<br/>avec cloud-init"]

  D --> DNS["🌐 Ansible<br/>bind.yml"]
  DNS -->|"DNS mis à jour"| F["🔧 Ansible<br/>post-install.yml"]
  E --> F

  F --> T1["👤 Compte sudoer"]
  F --> T2["📁 bashrc"]
  F --> T3["🐳 Groupes"]
  F --> T4["📊 NUT/Munin"]
  F --> T5["💾 UrBackup"]
  F --> T6["🕐 NTP/Locale"]
  F --> T7["🧰 Outils"]
  F --> T8["📝 Mise à jour /etc/motd<br/>+ Description Proxmox"]
  F --> T9["💻 Client GLPI"]

  T1 & T2 & T3 & T4 & T5 & T6 & T7 & T8 & T9 --> G["✅ VM configurée"]
```

Vous connaissez Mark Down ?
Bienvenue dans Mermaid !

```bash
apt install chrony
```

```
apt install chrony
```

```mermaid
graph LR
  A[Serveur] --> B[Ceph]
  B --> C[Stockage distribué]
```

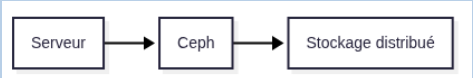
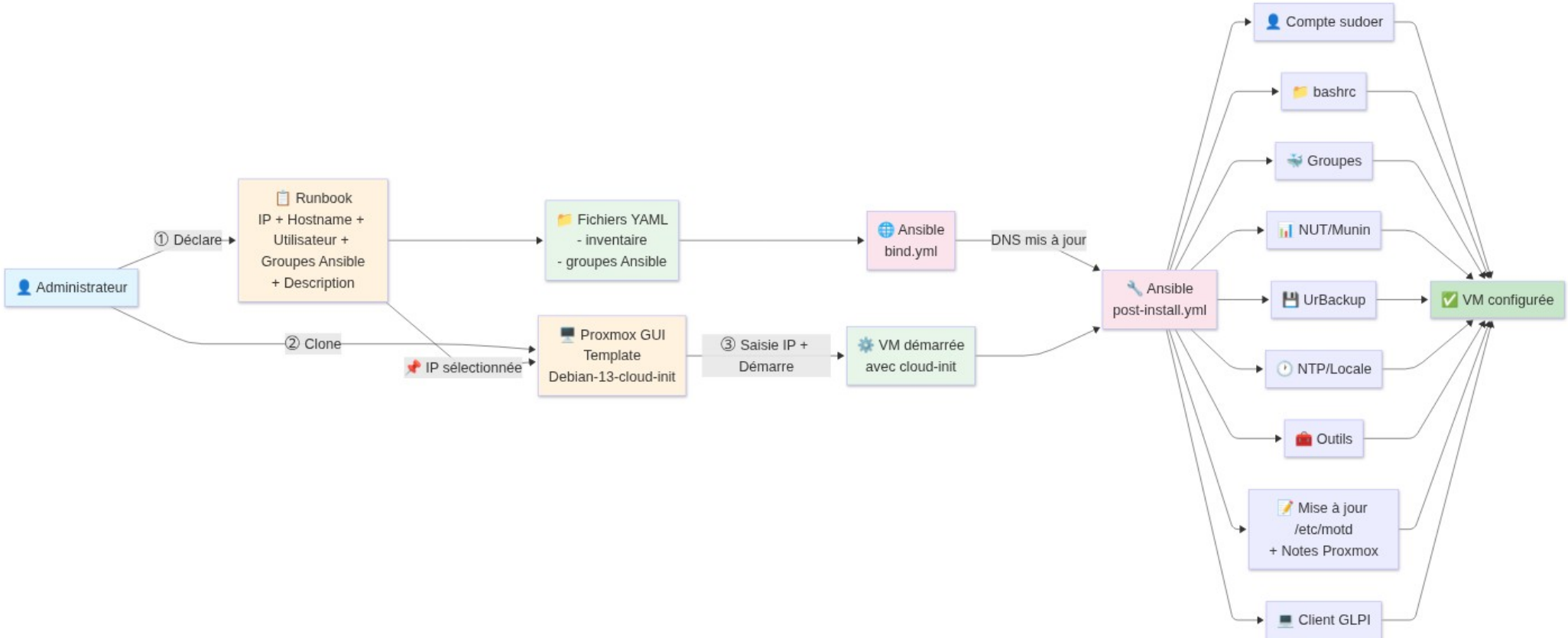


Diagramme du workflow



Étapes 1 à 3 : déclaration, Yaml/DNS, PVE

① ② ③ – Les trois premières étapes

- Déclaration dans le Runbook
- IP disponible, utilisateur, groupes Ansible, description
- Enregistrement dans les tables machine, utilisateur, groupe
- Préparation YAML + mise à jour DNS
- Génération de l'inventaire et des groupes Ansible
- Exécution de bind.yml pour ajouter l'enregistrement DNS
- Condition : la VM doit être joignable par son FQDN
- Création par clonage du template Debian 13 cloud-init
- Saisie de l'IP dans la section cloud-init
- Démarrage de la VM

À ce stade : la VM est démarrée, son IP est configurée, mais elle n'est pas encore configurée (pas de comptes, pas de monitoring, etc.)

Etape 4 : post-installation Ansible

④ Post-installation Ansible – Le « Do » qualité

Le playbook post-install.yml exécute toutes les tâches de configuration :

Tâche Ansible	Rôle qualité
👤 `Compte sudoer`	`Utilisateur prêt, clé SSH injectée`
📁 bashrc	Environnement de travail familier
🐳 Groupes Ansible	VM intégrée aux groupes déclarés
📊 NUT/Munin	Monitoring garanti
💾 UrBackup	Sauvegarde configurée
🕐 NTP/Locale	Conformité fuseau/locale
🧰 Outils communs	Base d'outils standard
📝 MOTD + Description Proxmox	Synchronisation de la description Runbook
💻 Client GLPI	Remontée d'informations vers la DSIN

Résultat : VM configurée conformément aux standards, prête à l'emploi.

Traçabilité : la « cale » qui évite les régressions

📝 Traçabilité des interventions : la « cale »

Principe : **écrire ce qu'on fait**

Outils : la table *intervention*

Ce qu'on peut tracer :

- Qui (utilisateur ou admin)
- Sur quelle machine
- Quel groupe concerné
- Titre, contenu détaillé, date
- Statut de l'intervention

Pourquoi c'est une « cale » :

- On ne peut pas « oublier » ce qu'on a fait
- L'historique est figé, daté, et interrogeable par mots
- Permet le Check (vérification) et l'Act (amélioration)
- Empêche de revenir en arrière sans trace

Démarche qualité : on planifie, on fait, on évalue, on ajuste => on améliore

Conclusion: bénéfices qualité

✅ Bénéfices de cette approche qualité

Standardisation : toutes les VM sont identiques dans leur base

Automatisation : réduction des erreurs humaines

Traçabilité : historique complet des actions et interventions

Amélioration continue : le PDCA est en place (Plan → Do → Check → Act)

La cale en place :

Impossible de revenir en arrière sans trace

On verrouille les bonnes pratiques

Résultat :

Infrastructure stable, maîtrisée, documentée

Équipe admin sereine, utilisateurs satisfaits

Conformité avec les exigences qualité (ISO, etc.)