Support pédagogique Ansible

AWX

Introduction à AWX et Ansible Tower

- **AWX** : Projet open source offrant une interface web pour Ansible.
- **Ansible Tower** : Version commerciale basée sur AWX, proposée par Red Hat.
- **Relation** : AWX sert de base à Ansible Tower, offrant des fonctionnalités similaires.

Dernière Release d'AWX

- **Date de la dernière release** : 2 juillet 2024.
- **Pause des releases** : Suspension des nouvelles versions pour permettre un refactoring majeur.
- **Objectif** : Transformer AWX en une architecture plus modulaire et orientée services.

Pourquoi ce Refactoring ?

- **Problèmes actuels** : Difficulté à implémenter des changements en raison de la structure monolithique.
- **Solution proposée** : Décomposer AWX en services modulaires pour améliorer la flexibilité et la maintenabilité.

Objectifs du Refactoring

- **Modularité** : Faciliter l'ajout de nouvelles fonctionnalités.
- **Réutilisation du code** : Améliorer l'utilisation du code dans d'autres projets Ansible.
- **Efficacité** : Accélérer le processus de développement et de déploiement.

Implications pour les Utilisateurs d'AWX

- **Pause des nouvelles fonctionnalités** : Suspension temporaire des mises à jour pendant le refactoring.
- **Stabilité** : Les versions existantes restent fonctionnelles, mais sans nouvelles mises à jour.
- **Suivi des évolutions** : Recommandation de suivre les annonces officielles pour rester informé.

Red Hat Ansible Automation Platform

- **Présentation** : Plateforme commerciale offrant un cadre complet pour l'automatisation avec Ansible.
- **Relation avec AWX** : Ansible Automation Platform intègre des composants basés sur AWX, offrant des fonctionnalités stables et supportées.

Nouveautés de la Version 2.4

- **Support ARM** : Exécution sur plateformes ARM, incluant le plan de contrôle et les environnements d'exécution.
- **Ansible Builder v3** : Simplification de la création d'environnements d'exécution.
- **Améliorations de sécurité** : Mises à jour de Django et sqlparse pour corriger des vulnérabilités.

Conclusion

- **AWX en transition** : Refactoring en cours pour une architecture plus modulaire.
- **Ansible Automation Platform** : Continue d'évoluer avec de nouvelles fonctionnalités et améliorations.

Architecture d’AWX

- **Composants** :
  - Interface Web
  - API REST
  - Moteur Ansible
- **Utilisation de conteneurs** : Docker ou Kubernetes
- **Schéma** : Web → Control Node → Managed Nodes

Concepts Clés

- Projets
- Templates de tâches
- Inventaires
- Identifiants (Credentials)
- Jobs
- Workflows
- Plannings (Schedules)

Projet (Project)

- **Source des playbooks** : Git, SVN, etc.
- **Synchronisation** : Manuelle ou planifiée
- **Exemple** : Lien vers un dépôt GitHub contenant des playbooks

Inventaire (Inventory)

- **Groupe de machines cibles**
- **Types** :
  - Statique (manuel)
  - Dynamique (AWS, Azure, GCE, etc.)
- **Variables** : D’hôtes et de groupe

Inventaire Dynamique

- **Intégration avec des fournisseurs cloud**
- **Plugins d'inventaire dynamique**
- **Exemple** : Inventaire AWS EC2 avec filtres

Identifiants (Credentials)

- **Types** :
  - SSH
  - Vault
  - Cloud (AWS, Azure, GCP)
  - Machine
- **Sécurité** : Chiffrement, RBAC

Template de Tâche (Job Template)

- **Définition** : Job réutilisable
- **Sélection** : Projet, inventaire, identifiants, playbook
- **Variables** : Passage via survey ou extra vars

Exécution d’un Job

- **Interface** : Lancer, suivre et consulter l’historique
- **Logs** : Détails de l’exécution
- **Exemple visuel** : Job en cours

Workflow Template

- **Chaînage** : Plusieurs jobs
- **Logique conditionnelle** : if/else
- **Exemple** : Déploiement multi-environnement

Surveys

- **Formulaire interactif** : Renseigner les variables au lancement d’un job
- **Types de champs** : Texte, choix, booléens
- **Validation** : Des entrées utilisateur

Plannings (Schedules)

- **Exécution automatique** : Jobs à intervalles définis
- **Format** : Cron
- **Exemple** : Exécution tous les lundis à 2h

RBAC (Role-Based Access Control)

- **Contrôle d'accès** : Par rôle (admin, user, auditor)
- **Attribution** : Par organisation, projet, inventaire
- **Objectif** : Sécurité et gouvernance

Organisation & Équipes

- **Organisation** : Regroupement logique
- **Équipe** : Groupe d’utilisateurs avec permissions partagées
- **Utilisation** : Grandes entreprises

API REST d'AWX

- **Accès** : Toutes les fonctionnalités via API
- **Utilisation** : Scripts ou applications externes
- **Documentation** : Swagger UI intégré

CLI : `awx` et `tower-cli`

- **Ligne de commande** : Interaction avec AWX
- **Exemples** : Commandes courantes
- **Utilité** : Pipelines CI/CD

Journalisation & Audit

- **Historique des actions** : Jobs, changements, connexions
- **Logs** : Exécution, erreurs, succès
- **Intégration** : Possible avec ELK/Prometheus

Cas d’Usage Réel

- **Exemple** : CI/CD
- **Déploiement automatisé** : De VM
- **Gestion des correctifs** : Sur 1000 serveurs

Bonnes Pratiques

- **Versionner** : Les playbooks
- **Utiliser** : Surveys et workflows
- **Restreindre les accès** : Via RBAC
- **Superviser** : Avec alertes