





# Microsoft Windows Automation with Red Hat Ansible



**Red Hat Ansible Engine 2.8 DO417**  
**Microsoft Windows Automation with Red Hat Ansible**  
**Édition 120200501**  
**Date de publication 20200106**

Auteurs: Razique Mahroua, Chris Caillouet, Joel Birchler, Artur Glogowski,  
Morgan Weetman, Rajat Agrawal  
Éditeur: Steven Bonneville, David O'Brien, Nicole Muller

Copyright © 2019 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are  
Copyright © 2019 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed, please send email to [training@redhat.com](mailto:training@redhat.com) or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, Hibernate, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

The OpenStack® word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation in the United States and other countries, and are used with the OpenStack Foundation's permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

<b>Conventions de la documentation</b>	<b>ix</b>
<b>Introduction</b>	<b>xi</b>
Microsoft Windows Automation with Red Hat Ansible .....	xi
Organisation de l'environnement de formation .....	xii
<b>1. Présentation de Red Hat Ansible Automation</b>	<b>1</b>
Présentation de Red Hat Ansible Automation .....	2
Quiz: Présentation de Red Hat Ansible Automation .....	9
Architecture de l'automatisation Windows avec Red Hat Ansible Tower .....	11
Exercice guidé: Architecture de l'automatisation Windows avec Red Hat Ansible Tower ....	19
Gestion de fichiers dans Git avec Visual Studio Code .....	26
Exercice guidé: Gestion de fichiers dans Git avec Visual Studio Code .....	35
Open Lab: Présentation de Red Hat Ansible Automation .....	39
Résumé .....	48
<b>2. Exécution de commandes simples d'automatisation</b>	<b>49</b>
Préparation des hôtes Microsoft Windows pour l'automatisation .....	50
Quiz: Préparation des hôtes Microsoft Windows pour l'automatisation .....	55
Préparation de Red Hat Ansible Tower pour gérer des hôtes .....	59
Exercice guidé: Préparation de Red Hat Ansible Tower pour gérer des hôtes .....	66
Exécution de commandes ad hoc .....	71
Exercice guidé: Exécution de commandes ad hoc .....	79
Open Lab: Exécution de commandes simples d'automatisation .....	85
Résumé .....	100
<b>3. Mise en œuvre de playbooks Ansible</b>	<b>101</b>
Écriture de playbooks .....	102
Exercice guidé: Écriture de playbooks .....	107
Exécution de playbooks dans Red Hat Ansible Tower .....	111
Exercice guidé: Exécution de playbooks dans Red Hat Ansible Tower .....	122
Mise en œuvre de plusieurs plays .....	128
Exercice guidé: Mise en œuvre de plusieurs plays .....	134
Open Lab: Mise en œuvre de playbooks Ansible .....	142
Résumé .....	153
<b>4. Gestion des variables et des faits</b>	<b>155</b>
Gestion des variables .....	156
Exercice guidé: Gestion des variables .....	165
Gestion des secrets .....	176
Quiz: Gestion des secrets .....	183
Gestion des faits .....	185
Exercice guidé: Gestion des faits .....	192
Open Lab: Gestion des variables et des faits .....	198
Résumé .....	205
<b>5. Installation et mise à jour de logiciels</b>	<b>207</b>
Installation et mise à jour de logiciels .....	208
Exercice guidé: Installation et mise à jour de logiciels .....	213
Modification du Registre .....	228
Exercice guidé: Modification du Registre .....	232
Open Lab: Installation et mise à jour de logiciels .....	238
Résumé .....	253
<b>6. Mise en œuvre d'un contrôle de tâche</b>	<b>255</b>
Écriture de boucles et de tâches conditionnelles .....	256
Exercice guidé: Écriture de boucles et de tâches conditionnelles .....	267
Mise en œuvre des gestionnaires .....	273

Exercice guidé: Mise en œuvre des gestionnaires .....	276
Gestion des échecs de tâche .....	280
Exercice guidé: Gestion des échecs de tâche .....	285
Open Lab: Mise en œuvre d'un contrôle de tâche .....	290
Résumé .....	298
<b>7. Déploiement de fichiers sur des hôtes gérés</b>	<b>299</b>
Modification et transfert de fichiers sur des hôtes .....	300
Exercice guidé: Modification et transfert de fichiers sur des hôtes .....	307
Création de modèles de fichiers à l'aide de Jinja2 .....	324
Exercice guidé: Création de modèles de fichiers à l'aide de Jinja2 .....	329
Open Lab: Déploiement de fichiers sur des hôtes gérés .....	337
Résumé .....	347
<b>8. Interaction avec les utilisateurs et les domaines</b>	<b>349</b>
Gestion des comptes d'utilisateur locaux .....	350
Exercice guidé: Gestion des comptes d'utilisateur locaux .....	355
Gestion des domaines Active Directory .....	358
Exercice guidé: Gestion des domaines Active Directory .....	365
Génération d'inventaires dynamiques à partir d'Active Directory .....	374
Exercice guidé: Génération d'inventaires dynamiques à partir d'Active Directory .....	381
Open Lab: Interaction avec les utilisateurs et les domaines .....	386
Résumé .....	395
<b>9. Automatisation des tâches d'administration Windows</b>	<b>397</b>
Exécution des commandes et planification des tâches sur les hôtes .....	398
Exercice guidé: Exécution des commandes et planification des tâches sur les hôtes .....	414
Configuration et gestion du stockage .....	431
Exercice guidé: Configuration et gestion du stockage .....	435
Open Lab: Automatisation des tâches d'administration Windows .....	440
Résumé .....	448
<b>10. Gestion de projets volumineux</b>	<b>449</b>
Inclusion et importation de fichiers .....	450
Exercice guidé: Inclusion et importation de fichiers .....	455
Création de rôles .....	460
Exercice guidé: Création de rôles .....	470
Déploiement de rôles avec Ansible Galaxy .....	477
Exercice guidé: Déploiement de rôles avec Ansible Galaxy .....	481
Intégration d'Ansible avec des ressources de configuration de l'état souhaité .....	489
Exercice guidé: Intégration d'Ansible avec des ressources de configuration de l'état souhaité .....	493
Open Lab: Gestion de projets volumineux .....	496
Résumé .....	505
<b>11. Construction de workflows Ansible Tower</b>	<b>507</b>
Création et gestion d'utilisateurs Ansible Tower .....	508
Exercice guidé: Gestion de l'accès des utilisateurs et des équipes .....	519
Création et utilisation de questionnaires .....	527
Exercice guidé: Création et utilisation de questionnaires .....	533
Planification de lancement de tâches .....	538
Exercice guidé: Planification de lancement de tâches .....	540
Exécution de workflows complexes .....	543
Exercice guidé: Exécution de workflows complexes .....	549
Open Lab: Construction de workflows Ansible Tower .....	554
Résumé .....	565
<b>12. Révision complète</b>	<b>567</b>

Révision complète .....	568
Open Lab: Examen général de l'atelier 1 .....	571
Open Lab: Examen général de l'atelier 2 .....	577
Open Lab: Examen général de l'atelier 3 .....	583
Open Lab: Examen général de l'atelier 4 .....	589
<b>A. Sujets supplémentaires</b>	<b>597</b>
Gestion des rôles utilisateur Red Hat Ansible Tower .....	598
<b>B. Mentions légales</b>	<b>607</b>
Licence du rôle arillso.chocolatey .....	608



# Conventions de la documentation



## Références

Les « références » indiquent où trouver de la documentation externe se rapportant à un sujet.



## Note

Une « remarque » est un conseil, un raccourci ou une approche alternative pour la tâche considérée. Le fait d'ignorer une remarque ne devrait pas entraîner de conséquences négatives, mais vous pourriez passer à côté d'une astuce qui vous simplifierait la vie.



## Important

Les cadres « Important » détaillent des éléments qui pourraient aisément être négligés : des changements de configuration qui ne s'appliquent qu'à la session en cours ou des services qui doivent être redémarrés pour qu'une mise à jour soit appliquée. Ignorer un cadre « Important » ne vous fera perdre aucune donnée, mais cela pourrait être source de frustration et d'irritation.



## Mise en garde

Un « avertissement » ne doit pas être ignoré. Le fait d'ignorer un avertissement risque fortement d'entraîner une perte de données.



# Introduction

## Microsoft Windows Automation with Red Hat Ansible

Microsoft Windows Automation with Red Hat Ansible (DO417) est conçu pour les professionnels Windows Server qui souhaitent utiliser Ansible pour écrire des playbooks d'automatisation afin d'effectuer des tâches d'administration système courantes, de façon reproductible et à l'échelle, pour les systèmes Microsoft Windows. Vous allez utiliser Red Hat Ansible Tower pour gérer et exécuter vos playbooks Ansible en toute sécurité à partir d'une interface utilisateur web centrale.

### Objectifs du cours

- Les stagiaires vont écrire et exécuter des tâches d'automatisation Microsoft Windows à l'aide de Red Hat Ansible Automation dans un environnement Microsoft Windows. Ils effectuent des tâches d'administration courantes, écrivent des projets Ansible à partir de leur poste de travail Microsoft Windows, stockent l'historique de toutes les modifications dans un système de contrôle de version basé sur Git, puis exécutent et résolvent ces tâches à partir de l'interface web de Red Hat Ansible Tower.

### Public

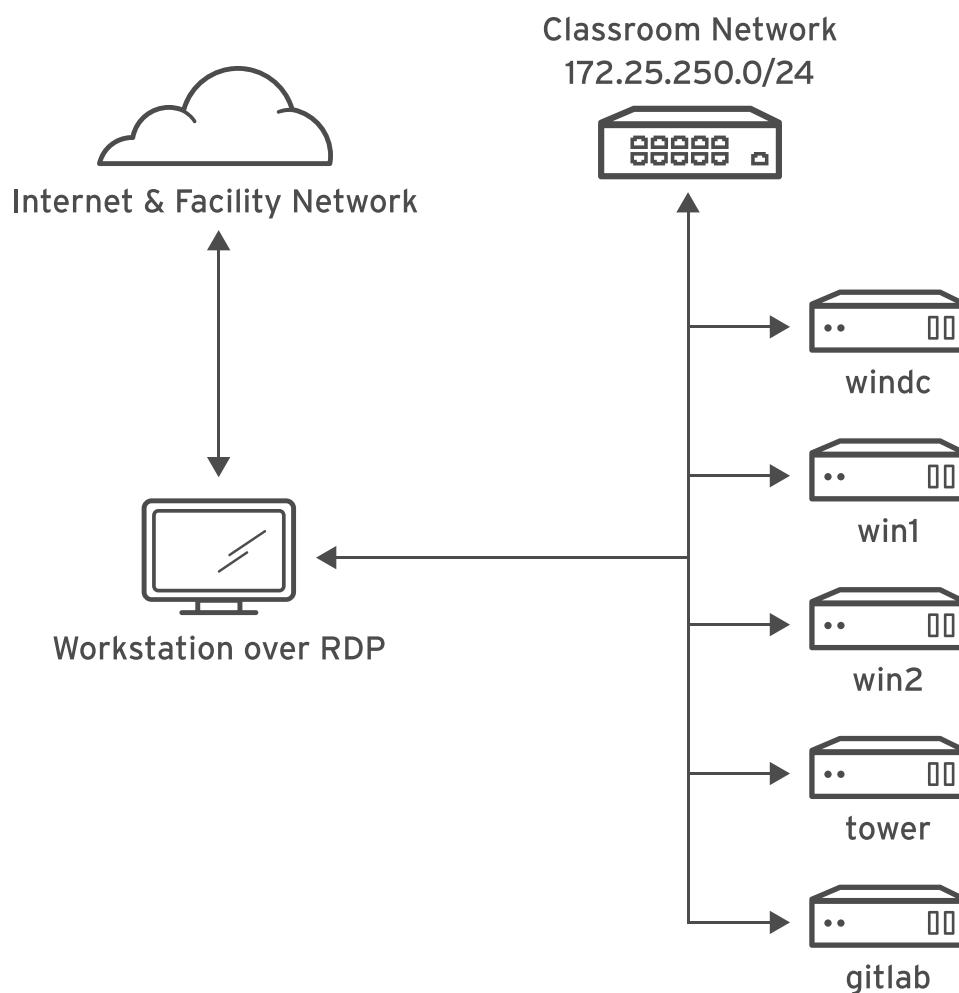
- Administrateurs Microsoft Windows Server désireux d'automatiser les tâches de gestion et d'utiliser des outils d'automatisation pour mettre en œuvre un workflow DevOps.

### Conditions préalables

- Les stagiaires doivent normalement avoir une expérience en tant qu'administrateur de Microsoft Windows Server. Aucune expérience préalable avec Red Hat Ansible Automation ou Linux n'est requise.

# Organisation de l'environnement de formation

---



**Figure 0.1: Environnement de formation**

Dans ce cours, le système informatique principal utilisé pour les travaux pratiques est **workstation**. Les informations d'identification permettant d'accéder à ce système s'affichent dans votre interface Red Hat Online Learning (ROL), sous l'onglet **Online Lab**. Ce cours utilise un ensemble de six machines virtuelles s'exécutant dans un environnement de cloud distant :

Tous les systèmes sont connectés à un sous-réseau privé, **172.25.250.0/24**. Les adresses IP sont générées au cours du provisionnement et restent statiques pendant tout le cycle de vie de l'environnement.

- **workstation** est le principal système que vous utiliserez de manière interactive dans le cadre de ce cours. Vous allez vous connecter à cet hôte à l'aide du protocole RDP (Remote Desktop)

## Introduction

Protocol) à partir de votre ordinateur ou de votre ordinateur portable. Les instructions relatives à la réalisation de ces connexions sont abordées en détail plus loin dans cette introduction.

Le système **workstation** est le seul système directement connecté à Internet. Il est également connecté à un réseau privé utilisé par tous les autres systèmes de la salle de classe.

- **windc** est le contrôleur de domaine Microsoft Active Directory pour le domaine **example.com**.
- **win1** et **win2** sont les deux hôtes gérés par Ansible.
- **tower** fournit le service Red Hat Ansible Tower que vous utiliserez pour exécuter votre code d'automatisation.
- **gitlab** fournit un service GitLab CE qui fournit les référentiels de contrôle de version Git, dans lesquels sont stockés votre code et d'autres supports d'exercice.

#### Résumé des machines de la salle de classe

Nom de la machine	Rôle
<b>workstation.example.com</b>	Hôte Windows Server utilisé en tant que poste de travail
<b>windc.example.com</b>	Contrôleur de domaine Active Directory pour le domaine <b>example.com</b>
<b>win1.example.com</b>	Premier hôte Windows Server géré par Ansible
<b>win2.example.com</b>	Deuxième hôte Windows Server géré par Ansible
<b>tower.example.com</b>	Serveur Red Hat Ansible Tower
<b>gitlab.example.com</b>	Serveur GitLab

La liste suivante décrit les comptes d'utilisateurs qui sont disponibles :

- Sur **workstation**, vous allez utiliser l'utilisateur **training** comme compte d'utilisateur standard. Le mot de passe de ce compte est automatiquement généré et disponible dans la plateforme d'apprentissage en ligne (ROL), sous l'onglet **Online Lab**.
- Le compte **Administrator** local sur ce système utilise le même mot de passe.
- Sur **windc**, le compte d'utilisateur local **Administrator** utilise **RedHat123@!** comme mot de passe. Il s'agit également du mot de passe du compte d'administrateur de domaine **example.com**.
- Sur **win1** et **win2**, l'utilisateur **devops** est utilisé par Red Hat Ansible Tower pour des raisons de gestion et d'automatisation.

Ce compte utilise **RedHat123@!** comme mot de passe.

- L'URL <http://tower.example.com> fournit un accès à l'interface utilisateur web de Red Hat Ansible Tower. Vous pouvez vous connecter à l'interface web en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.
- L'URL <https://gitlab.example.com> fournit un accès à l'interface utilisateur web de GitLab CE dans votre environnement de formation. Vous pouvez vous connecter en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

## Contrôle de vos systèmes

Les ordinateurs que vous allez utiliser pour les activités se trouvent dans une salle de classe Red Hat Online Learning à distance. Vous pouvez les contrôler à l'aide d'une application web hébergée sur <http://rol.redhat.com>. Vous devez vous connecter à ce site à l'aide de vos informations d'identification du Portail Client Red Hat.

Pour accéder à la salle de classe à distance, vous avez besoin d'un ordinateur disposant d'un accès à Internet et d'un client RDP. Ce périphérique doit avoir une résolution d'écran d'au moins 1280 x 800 pixels. Nous supposons que vous fournissez votre propre ordinateur pour ce cours de formation.



### Important

Si vous participez à une session de formation dispensée par un instructeur et que vous ne disposez pas de votre propre ordinateur avec vous, ou si vous rencontrez des difficultés avec votre client RDP, veuillez contacter l'instructeur pour obtenir de l'aide.

## Installation d'un client RDP

Si aucun client RDP n'est installé sur votre ordinateur, vous devez être en mesure d'installer le logiciel sur celui-ci. Le tableau suivant répertorie certains clients RDP suggérés pour les systèmes d'exploitation les plus courants.

### Clients RDP recommandés

Système d'exploitation	Nom du client	Installation
Windows 2000 ou version ultérieure	Bureau à distance Microsoft	Installé sur tous les systèmes Windows, Windows 2000 ou version ultérieure
macOS 10.12 ou version ultérieure, processeur 64 bits	Bureau à distance Microsoft	Installez à partir de <a href="https://apps.apple.com/us/app/microsoft-remote-desktop-10/id1295203466">https://apps.apple.com/us/app/microsoft-remote-desktop-10/id1295203466</a> , ou directement à partir du Mac App Store.
Distributions Linux, y compris Fedora, Red Hat Enterprise Linux, Ubuntu et Debian	Remmina	<p>Fedora fournit Remmina dans la distribution. Installez avec <b>sudo dnf install remmina</b>.</p> <p>Des instructions pour d'autres distributions Linux sont disponibles à l'adresse <a href="https://remmina.org/how-to-install-remmina/">https://remmina.org/how-to-install-remmina/</a>. Sur ces distributions, assurez-vous que le programme et son plug-in RDP pour Remmina sont installés.</p> <p>Remmina présente certains avantages en matière de fonctionnalités par rapport à <b>xfreerdp</b>.</p>

Système d'exploitation	Nom du client	Installation
Red Hat Enterprise Linux 7 ou version ultérieure	xfreerdp	<p>Fourni par la distribution. Installez avec <b>sudo yum install freerdp</b>.</p> <p>Vous devez également vous assurer que les plug-ins requis pour la prise en charge de Windows RDP sont installés (<i>freerdp-plug-ins</i> pour RHEL 7 et <i>freerdp-libs</i> pour RHEL 8).</p> <p>Ce client dispose de moins de fonctionnalités que Remmina mais il est facile à installer sur Red Hat Enterprise Linux.</p>

## Contrôle des machines virtuelles

Les machines virtuelles de l'environnement de formation sont contrôlées sur une Page Web. L'état de chaque machine virtuelle de la salle de classe est affiché sur la page de l'onglet **Online Lab**.

### États de la machine

État de la machine virtuelle	Description
STARTING	La machine virtuelle est en cours de démarrage.
STARTED	La machine virtuelle est en cours d'exécution et disponible (ou, pendant le démarrage, le sera bientôt.)
STOPPING	La machine virtuelle est en cours d'arrêt.
STOPPED	La machine virtuelle est complètement arrêtée. Au démarrage, la machine virtuelle affiche le même état que lors de son arrêt (le disque est préservé).
PUBLISHING	La création initiale de la machine virtuelle est en cours.
WAITING_TO_START	La machine virtuelle est en attente du démarrage d'autres machines virtuelles.

Selon l'état de la machine, une sélection des actions suivantes est disponible.

### Actions de machine/salle de classe

Bouton ou action	Description
<b>PROVISION LAB</b>	Permet de créer la salle de classe ROL. Crée toutes les machines virtuelles nécessaires pour la salle de classe et les démarre. Peut prendre jusqu'à une heure.

Bouton ou action	Description
<b>DELETE LAB</b>	Permet de supprimer la salle de classe ROL. Détruit toutes les machines virtuelles de la salle de classe. <b>Attention : tous les travaux générés sur les disques seront perdus.</b>
<b>START LAB</b>	Permet de démarrer toutes les machines virtuelles de la salle de classe.
<b>SHUTDOWN LAB</b>	Permet d'arrêter toutes les machines virtuelles de la salle de classe.
<b>ACTION → Start</b>	Permet de démarrer (allumer) la machine virtuelle.
<b>ACTION → Shutdown</b>	Permet d'éteindre correctement la machine virtuelle, en conservant le contenu sur son disque.
<b>ACTION → Power Off</b>	Force l'arrêt de la machine virtuelle, en conservant le contenu du disque. Cela équivaut à couper l'alimentation d'une machine physique.

Au début d'un exercice, si vous êtes invité à réinitialiser un seul nœud de machine virtuelle, cliquez sur **ACTION → Reset** pour la machine virtuelle spécifique que vous voulez réinitialiser.

Au début d'un exercice, si vous êtes invité à réinitialiser l'ensemble des machines virtuelles, cliquez sur **ACTION → Reset**

Si vous souhaitez que l'environnement de formation retourne à son état d'origine du début du cours, cliquez sur **DELETE LAB** pour supprimer l'ensemble de l'environnement de formation. Une fois l'atelier supprimé, vous pouvez cliquer sur **PROVISION LAB** pour déployer un nouvel ensemble de systèmes de salle de classe.



#### Mise en garde

L'opération **DELETE LAB** ne peut pas être annulée. Tous les travaux que vous avez terminés jusqu'ici dans l'environnement de formation sont perdus.

## Minuterie d'arrêt automatique

L'inscription à la formation en ligne Red Hat (ROL) confère aux stagiaires le droit d'utiliser l'ordinateur pendant un temps donné. Afin de les aider à conserver le temps d'utilisation de l'ordinateur qui leur est alloué, une minuterie d'arrêt automatique est associée à la salle de classe ROL. Celle-ci ferme l'environnement de la salle de classe à l'expiration du temps prévu.

Pour régler la minuterie, cliquez sur **MODIFY** afin que la fenêtre **New Autostop Time** s'affiche. Configurez le nombre d'heures jusqu'à l'arrêt automatique de la salle de classe. Notez qu'il existe une durée maximale de dix heures. Cliquez sur **ADJUST TIME** pour appliquer la modification aux paramètres de la minuterie.

## Cycle de vie de la salle de classe

L'inscription à Red Hat Online Learning limite le cycle de vie de cette classe à nonante (90) jours. Après 90 jours, l'environnement est désactivé et toutes les données sont perdues. Cependant, vous pouvez approvisionner un nouvel environnement si nécessaire. Si nécessaire, suivez les instructions mentionnées ci-dessus pour approvisionner l'environnement.

Lors de la mise en service, l'onglet **Online Lab** affiche les informations de connexion pour l'environnement. Cela inclut la liste des systèmes qui sont approvisionnés et les différentes informations d'identification que vous allez utiliser tout au long du cours.



## chapitre 1

# Présentation de Red Hat Ansible Automation

### Objectif

Décrire l'objectif et les avantages de l'automatisation des tâches d'administration de Windows Server, ainsi que l'architecture de base d'une solution basée sur Red Hat Ansible Automation.

### Résultats

- Décrire les concepts fondamentaux de l'automatisation avec Red Hat Ansible Automation, sa conception de base et les cas d'utilisation courants.
- Décrire l'architecture d'une mise en œuvre d'automatisation Windows à l'aide de la solution Red Hat Ansible Tower et d'un référentiel Git comme point central de contrôle.
- Récupérer, gérer, modifier et stocker des fichiers dans un système de contrôle de version Git distant, tel que GitHub ou GitLab, à l'aide de Visual Studio Code.

### Sections

- Présentation de Red Hat Ansible Automation (et quiz)
- Description de l'architecture de l'automatisation Windows Automation avec Red Hat Ansible Tower (et exercice guidé)
- Gestion de fichiers dans Git avec Visual Studio Code (et exercice guidé)

### Atelier

Présentation de Red Hat Ansible Automation

# Présentation de Red Hat Ansible Automation

---

## Résultats

Au terme de cette section, vous devez pouvoir décrire les concepts fondamentaux de l'automatisation avec Red Hat Ansible Automation, sa conception de base et les cas d'utilisation courants.

### Présentation de l'automatisation et de l'administration système Windows

Pendant de nombreuses années, la gestion de l'infrastructure et l'administration du système étaient basées, en grande partie, sur les tâches manuelles exécutées par le biais d'interfaces utilisateur graphiques ou par ligne de commande. Les administrateurs système ont généralement recours à des listes de contrôle, des sources de documentation ou une routine mémorisée pour effectuer des tâches standard.

Cependant, cette approche est sujette aux erreurs. Il est facile pour un administrateur système d'ignorer une étape ou d'effectuer une étape de manière incorrecte. Une vérification limitée est souvent mise en place pour s'assurer que les étapes ont été correctement exécutées ou qu'elles donnent le résultat escompté.

De plus, compte tenu de la gestion manuelle et indépendante de chaque serveur, il n'est pas exclu que de nombreux serveurs dont la configuration doit être identique présentent, en fait, de légères différences, voire des différences importantes. Ces incohérences peuvent rendre la maintenance plus complexe et entraîner des erreurs ou une instabilité au sein de l'environnement informatique.

L'*automatisation* peut vous permettre d'éviter les problèmes causés par l'administration du système et la gestion de l'infrastructure manuelles. En tant qu'administrateur système, utilisez l'automatisation pour vous assurer que tous vos systèmes sont déployés et configurés rapidement et correctement. L'automatisation vous évite d'exécuter des tâches répétitives dans votre planning quotidien, et ainsi de gagner du temps et de vous concentrer sur des tâches plus sensibles. Pour votre entreprise, cela signifie que vous pouvez déployer plus rapidement la prochaine version d'une application ou les mises à jour d'un service.

### Infrastructure-as-Code

Un bon système d'automatisation vous permet de mettre en œuvre des pratiques *Infrastructure-as-Code* (*IaC*). *IaC* signifie que vous pouvez utiliser un langage d'automatisation lisible par ordinateur pour définir et décrire l'état dans lequel vous voulez que se trouve votre infrastructure informatique. Idéalement, ce langage d'automatisation est facile à lire pour un humain, de sorte qu'il soit aisément compréhensible et d'y apporter des modifications. Ce code est ensuite appliqué à votre infrastructure pour vous assurer qu'elle se trouve effectivement dans cet état.

Si le langage d'automatisation est représenté sous la forme de fichiers texte simples, alors vous pouvez le gérer dans un système de contrôle de version, tel que Git, de la même façon que vous gériez du code logiciel. Cela a pour avantage que chaque modification peut alors être archivée dans le système de contrôle de version, de sorte que vous disposez d'un historique des modifications que vous effectuez au fil du temps. Si vous souhaitez revenir à une précédente configuration valide connue, vous pouvez extraire cette version du code et l'appliquer à votre infrastructure.

En gérant votre infrastructure en tant que code (IaC), vous pouvez vous assurer que vos environnements de pré-production et de production sont cohérents. Cela vous permet également de mettre en œuvre des pratiques d'*intégration continue* (CI) et de *déploiement continu* (CD). Vous pouvez intégrer des outils tels que lint et des tests unitaires pour vérifier le code qui définit la configuration de votre infrastructure avant de valider vos modifications. L'utilisation de ces outils peut vous aider à améliorer la qualité globale de votre code et à appliquer les normes de conformité, en créant une base pour prendre en charge les meilleures pratiques DevOps.

Les développeurs peuvent définir la configuration de leur choix dans le langage d'automatisation. Les opérateurs peuvent examiner ces modifications plus facilement afin de formuler des commentaires, et ils peuvent utiliser cette automatisation pour s'assurer, de manière reproductible, que les systèmes se trouvent bien dans l'état attendu par les développeurs. Enfin, cela vous aide à établir les meilleures pratiques et à vous assurer qu'elles sont alignées sur les différentes implémentations d'équipes.

## Atténuation des erreurs humaines

En réduisant le nombre de tâches effectuées manuellement sur les serveurs à l'aide de l'automatisation des tâches et des pratiques IaC, vos serveurs seront plus souvent configurés de manière cohérente. Cela signifie que vous devez vous habituer à apporter des modifications en mettant à jour le code d'automatisation, plutôt que de les appliquer manuellement à vos serveurs. Sinon, vous courez le risque de perdre les modifications appliquées manuellement la prochaine fois que vous apporterez des changements via l'automatisation.

L'automatisation permet la révision du code par plusieurs experts et prend en charge la documentation automatisée du processus de révision afin de réduire les risques opérationnels.

Au final, vous pouvez faire en sorte que les modifications apportées à votre infrastructure informatique soient effectuées par le biais de l'automatisation pour limiter les erreurs humaines.

## Qu'est-ce qu'Ansible ?

*Ansible* est une plateforme d'automatisation Open Source et un *langage simple d'automatisation* qui peut décrire l'infrastructure d'application informatique dans les *playbooks Ansible*. C'est aussi un *moteur d'automatisation* qui exécute des playbooks Ansible.

Ansible peut gérer des tâches puissantes d'automatisation et s'adapter à de nombreux workflows et environnements différents. De plus, les nouveaux utilisateurs d'Ansible peuvent l'utiliser pour devenir rapidement productifs.

## Ansible est simple

Les playbooks Ansible fournissent une automatisation lisible par l'utilisateur. Cela signifie que les playbooks sont des outils d'automatisation faciles à lire, à comprendre et à modifier. Les utilisateurs n'ont besoin d'aucune compétence particulière en codage pour les écrire. Les playbooks exécutent des tâches dans l'ordre. La simplicité de la conception des playbooks les rend utilisables par toutes les équipes, ce qui permet aux personnes novices dans Ansible d'être rapidement productives.

## Ansible est puissant

Vous pouvez utiliser Ansible pour le déploiement d'applications, la gestion de la configuration ainsi que l'automatisation du workflow et du réseau. Ansible vous permet dès lors d'orchestrer le cycle de vie complet d'une application.

## Ansible n'utilise pas d'agent

Ansible s'articule autour d'une *architecture sans agent*. En général, Ansible se connecte aux hôtes qu'il gère en utilisant OpenSSH, Windows Remote Management (*WinRM*) ou PowerShell Remoting Protocol (*PSRP*), puis exécute des tâches, la plupart du temps en transmettant par push de petits programmes, appelés *Ansible modules*, à ces hôtes. Ces programmes servent à placer le système dans un état spécifique. Tout module transmis par push est supprimé lorsqu'Ansible a terminé ses tâches. Vous pouvez commencer à utiliser Ansible quasiment immédiatement, car aucun agent n'exige d'approbation pour être utilisé et déployé sur les hôtes gérés. En l'absence d'agent ou d'autre infrastructure de sécurité personnalisée, Ansible s'avère plus efficace et sécurisé que d'autres solutions.

Ansible présente de nombreux atouts :

- *Prise en charge multiplateforme* : Ansible offre une prise en charge sans agent de Linux, Windows, UNIX et des périphériques réseau dans des environnements physiques, virtuels, de cloud et de conteneurs.
- *Automatisation visible par l'utilisateur* : les playbooks Ansible, écrits sous la forme de fichiers texte YAML, offrent une facilité de lecture et la garantie que tout le monde comprenne les actions effectuées.
- *Description parfaite des applications* : chaque modification peut être effectuée par des playbooks Ansible et chaque aspect de l'environnement d'application peut être décrit et documenté.
- *Gestion facile grâce au contrôle de versions* : les playbooks et les projets Ansible sont en texte brut. Ils peuvent être traités comme du code source et placés dans le système existant de contrôle de versions.
- *Prise en charge d'inventaires dynamiques* : la liste de machines gérées par Ansible peut être mise à jour de manière dynamique à partir de sources externes. Ainsi, la liste correcte et actuelle de tous les serveurs gérés est constamment capturée, indépendamment de l'infrastructure ou de l'emplacement.
- *Intégration facile de l'orchestration avec d'autres systèmes* : HPSA (HP Server Automation), Puppet, Jenkins, Red Hat Satellite et d'autres systèmes existants dans l'environnement peuvent être exploités et intégrés dans votre workflow Ansible.

## Ansible : le langage des DevOps



Figure 1.1: Ansible tout au long du cycle de vie des applications

La communication est essentielle pour les DevOps. Ansible est le premier langage d'automatisation qui peut être lu et écrit sur des systèmes sans connaissances préalables en programmation.

## La manière Ansible

Ansible décrit votre infrastructure de manière simple et ne requiert pas de compétences avancées en langage de programmation. Il a été conçu en tenant compte de la simplicité et de l'efficacité.

### La complexité tue la productivité

La simplicité paie. De par sa conception, Ansible est doté d'outils simples à utiliser. Par conséquent, l'automatisation est facile à écrire et à lire. Tirez parti de cette simplicité pour simplifier la façon dont vous créez l'automatisation.

### Meilleure lisibilité

Le langage d'automatisation Ansible repose sur des fichiers texte simples, déclaratifs et faciles à lire pour les utilisateurs. Correctement écrits, les playbooks Ansible peuvent clairement documenter l'automatisation du workflow.

### Langage déclaratif

Ansible est un *moteur qui permet d'appliquer l'état souhaité*. Il cerne le problème de l'automatisation des déploiements informatiques en exprimant ces derniers en termes d'état système souhaité. Ansible configure vos systèmes dans l'état souhaité uniquement en effectuant les modifications nécessaires. Ansible n'est pas un langage de script et ne doit pas être traité en tant que tel.

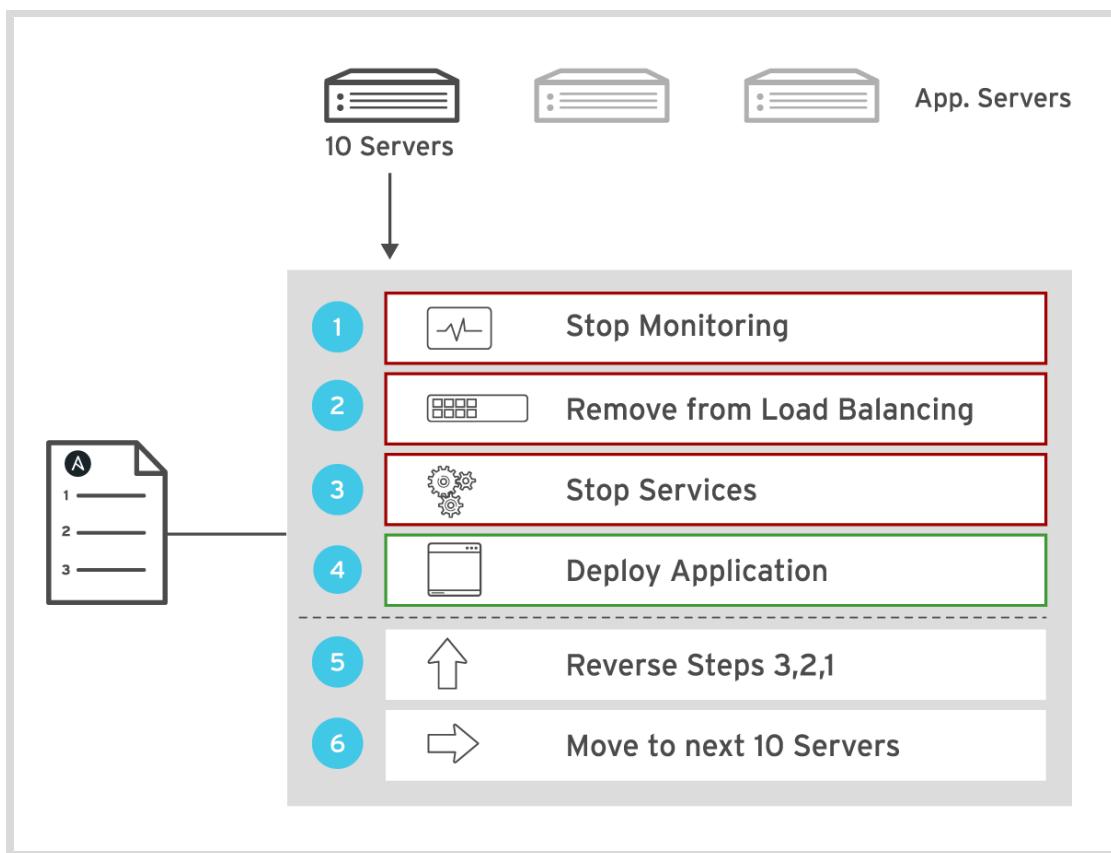


Figure 1.2: Ansible offre une automatisation complète

## Concepts et architecture Ansible

L'architecture Ansible propose deux types de machine : les *nœuds de contrôle* et les *hôtes gérés*. Ansible est installé et exécuté à partir d'un nœud de contrôle, qui est une machine unique contenant également des copies des fichiers de projet Ansible. Un nœud de contrôle qui s'exécute sur Linux peut être l'ordinateur portable d'un administrateur, un système partagé par plusieurs administrateurs, une machine virtuelle Linux ou un serveur exécutant Red Hat Ansible Tower.

Les hôtes gérés sont listés dans un *inventaire*, qui organise également ces systèmes en groupes afin de faciliter la gestion collective. Vous pouvez définir l'inventaire dans un fichier texte statique ou le déterminer de manière dynamique au moyen de scripts qui obtiennent des informations provenant de sources externes.

Au lieu d'écrire des scripts complexes, les utilisateurs Ansible créent des *plays* de haut niveau pour s'assurer qu'un hôte ou un groupe d'hôtes soit dans un état particulier. Un play effectue une série de *tâches* sur les hôtes, dans l'ordre spécifié par le play. Ces plays sont exprimés au format YAML dans un fichier texte. Un fichier contenant un ou plusieurs plays est appelé un *playbook*.

Chaque tâche exécute un *module*, qui est une petite section de code (écrite en Python, PowerShell ou un autre langage) avec des arguments spécifiques. Ansible est fourni avec de nombreux modules utiles qui peuvent accomplir une grande variété de tâches d'automatisation. Chaque module constitue un outil de votre boîte à outils. Les modules peuvent agir sur des fichiers système, installer des services ou des utilisateurs, ou interagir avec des configurations DSC PowerShell.

Lorsqu'un module est utilisé dans une tâche, il permet généralement de s'assurer qu'un aspect particulier concernant la machine est dans un état particulier. Par exemple, une tâche utilisant un module peut garantir qu'un fichier existe et dispose d'autorisations et de contenus particuliers, tandis qu'une tâche utilisant un module différent peut garantir qu'un système de fichiers particuliers est monté. Si le système ne se trouve pas dans cet état, c'est la tâche qui l'y mettra. À l'inverse, si le système est déjà au bon état, il ne fait rien. Si une tâche échoue, alors par défaut, Ansible va interrompre le reste du playbook pour les hôtes qui ont échoué.

Les tâches, les plays et les playbooks sont conçus pour être *idempotents*. Cela signifie que vous pouvez exécuter en toute sécurité un playbook sur le même hôte plusieurs fois. Lorsque vos systèmes sont dans le bon état, le playbook n'effectue aucune modification lors de son exécution. Par conséquent, l'exécution d'un playbook à plusieurs reprises sur le même hôte est sûre. Il existe quelques modules que vous pouvez utiliser pour exécuter des commandes arbitraires. Cependant, vous devez les utiliser avec précaution pour vous assurer qu'ils s'exécutent de manière idempotente.

Comme expliqué précédemment, l'architecture d'Ansible ne repose pas sur l'utilisation d'agents. Par conséquent, lorsque vous exécutez un playbook Ansible ou une commande ad hoc, le nœud de contrôle se connecte à l'hôte géré, à l'aide de SSH (configuration par défaut) ou de WinRM (pour la plupart des hôtes Windows). Les clients n'ont besoin que d'un écouteur WinRM installé sur les hôtes, plutôt que d'un agent spécifique à Ansible, et, par conséquent, il n'est pas nécessaire d'autoriser le trafic réseau spécial sur un port non standard.

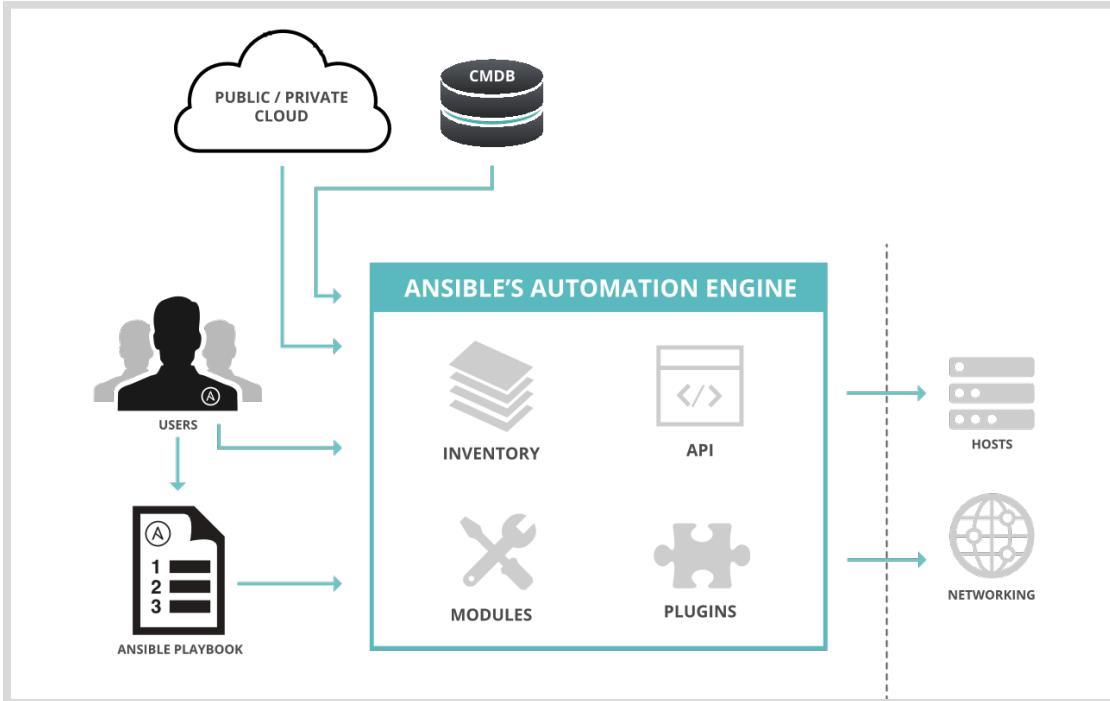


Figure 1.3: Architecture Ansible

*Red Hat Ansible Tower* est une structure d'entreprise qui vous permet de contrôler, de sécuriser et de gérer l'automatisation Ansible à l'échelle. Il agit en tant que nœud de contrôle pour exécuter vos playbooks Ansible. Il fournit une interface utilisateur Web et une API RESTful. Les administrateurs Linux peuvent utiliser les outils de ligne de commande fournis avec Ansible pour exécuter des playbooks, mais Red Hat Ansible Tower fournit une interface graphique centralisée qui vous aide à utiliser Ansible de manière plus efficace avec une équipe ou à grande échelle. Cela permet aux utilisateurs sans expérience Linux d'exécuter facilement l'automatisation Ansible, tout en fournissant également un point central d'autorisation, de gestion et de journalisation des activités d'automatisation.

## Exemples d'utilisation

Contrairement à d'autres outils, Ansible associe et réunit l'orchestration et la gestion de configuration, le provisionnement et le déploiement d'applications dans une seule plateforme facile à utiliser.

Parmi les cas d'utilisation d'Ansible, nous pouvons citer :

### Gestion de la configuration

De nombreux utilisateurs expérimentés découvrent pour la première fois la plateforme d'automatisation Ansible au travers du cas d'utilisation courant d'Ansible qui porte sur la centralisation de la gestion et du déploiement des fichiers de configuration.

### Déploiement d'applications

Lorsque vous définissez votre application avec Ansible et gérez le déploiement avec Red Hat Ansible Tower, les équipes peuvent traiter efficacement l'ensemble du cycle de vie de l'application, du développement à la production.

### Déploiement

Vous devez déployer ou installer les applications sur les systèmes. Ansible et Red Hat Ansible Tower permettent de rationaliser le processus des systèmes de provisionnement, que vous utilisez un environnement PXE (*Preboot Execution Environment*) avec une installation

Kickstart sur des serveurs sans système d'exploitation ou des machines virtuelles, ou que vous créez des machines virtuelles ou des instances cloud à partir de modèles.

#### Livraison continue

La création d'un pipeline CI/CD impose la coordination et l'adhésion de nombreuses équipes. Dans ce contexte, la plateforme d'automatisation simple est l'élément essentiel qui peut être utilisé par tous les membres de votre organisation. Avec les playbooks Ansible, les applications sont correctement déployées (et gérées) tout au long de leur cycle de vie.

#### Sécurité et conformité

Lorsque votre politique de sécurité est définie dans les playbooks Ansible, l'analyse et l'application des règles de sécurité à l'échelle du site peuvent être intégrées dans d'autres processus automatisés. Au lieu d'être considérés en aval, ces aspects sont intégrés dans l'ensemble des déploiements.

#### Orchestration

Les configurations seules ne définissent pas votre environnement. Vous devez également définir la façon dont plusieurs configurations interagissent et garantir que les éléments hétérogènes puissent être gérés dans leur ensemble.



#### Références

##### Ansible

<https://www.ansible.com>

##### Fonctionnement d'Ansible

<https://www.ansible.com/how-ansible-works>

## ► Quiz

# Présentation de Red Hat Ansible Automation

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

► 1. Quel terme décrit le mieux l'architecture Ansible ?

- a. Sans agent
- b. Client/Serveur
- c. Axé sur les événements
- d. Sans état

► 2. Parmi les deux protocoles suivants, lequel Ansible utilise-t-il pour communiquer avec des nœuds Windows gérés ? (Choisissez-en deux.)

- a. HTTP
- b. HTTPS
- c. WinRM
- d. WinSCP
- e. Protocole PowerShell à distance

► 3. Quel fichier définit les actions effectuées par Ansible sur les nœuds gérés ?

- a. Inventaire d'hôtes
- b. Manifeste
- c. Playbook
- d. Script

► 4. Quelle syntaxe est utilisée pour définir les playbooks Ansible ?

- a. Bash
- b. Perl
- c. Python
- d. YAML

## ► Solution

# Présentation de Red Hat Ansible Automation

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

► 1. Quel terme décrit le mieux l'architecture Ansible ?

- a. Sans agent
- b. Client/Serveur
- c. Axé sur les événements
- d. Sans état

► 2. Parmi les deux protocoles suivants, lequel Ansible utilise-t-il pour communiquer avec des nœuds Windows gérés ? (Choisissez-en deux.)

- a. HTTP
- b. HTTPS
- c. WinRM
- d. WinSCP
- e. Protocole PowerShell à distance

► 3. Quel fichier définit les actions effectuées par Ansible sur les nœuds gérés ?

- a. Inventaire d'hôtes
- b. Manifeste
- c. Playbook
- d. Script

► 4. Quelle syntaxe est utilisée pour définir les playbooks Ansible ?

- a. Bash
- b. Perl
- c. Python
- d. YAML

# Architecture de l'automatisation Windows avec Red Hat Ansible Tower

## Résultats

Au terme de cette section, vous devez pouvoir décrire l'architecture d'une mise en œuvre d'automatisation Windows à l'aide de la solution Red Hat Ansible Tower et d'un référentiel Git comme point central de contrôle.

## Intégration d'Ansible dans un environnement Microsoft Windows

Il existe un certain nombre de méthodes que vous pouvez utiliser pour mettre en œuvre l'automatisation Ansible dans un environnement entièrement ou principalement composé d'hôtes gérés par Microsoft Windows. Red Hat a collaboré avec un large éventail de clients, et sur la base de notre expérience, nous vous recommandons le modèle d'architecture suivant :

- Utilisez *Red Hat Ansible Tower* en tant que nœud de contrôle pour exécuter des playbooks Ansible.
- Stockez des playbooks Ansible et d'autres supports dans un système de contrôle de version distribué, de préférence un serveur basé sur *Git*.
- Utilisez des éditeurs de code source natifs Windows qui peuvent utiliser directement des fichiers dans votre système de contrôle de version afin de modifier des playbooks Ansible et d'autres supports.

L'un des avantages de cette solution est que la plupart de vos administrateurs Windows n'ont pas besoin d'utiliser la ligne de commande Linux pour utiliser Ansible. Au lieu de cela, ils peuvent utiliser l'interface utilisateur graphique de Red Hat Ansible Tower, l'interface utilisateur web ou son API REST. Cela réduit la courbe d'apprentissage Ansible pour les administrateurs et fournit également des capacités intéressantes d'Ansible Tower, telles que la planification d'événements, la journalisation et la gestion des secrets.

L'utilisation d'un système de contrôle de version pour gérer votre code est une pratique recommandée, même si vous n'utilisez pas Ansible Tower. Vous pouvez ainsi effectuer le suivi des personnes responsables de l'élaboration de modifications spécifiques au code, et pouvez revenir à des révisions plus anciennes, si nécessaire. Lorsque vous utilisez Ansible pour définir un élément aussi important que votre infrastructure IaC, il est important que vous disposiez de pratiques solides pour gérer ce code.

L'une des raisons pour lesquelles nous recommandons Git est que ses compétences sont courantes, et il existe des services puissants permettant de gérer le référentiel Git pour l'utiliser à la fois dans le cloud et en tant qu'installation sur site. Ceux-ci incluent GitHub, GitLab et Bitbucket, entre autres. Dans ce cours, nous utiliserons une installation locale de GitLab, plutôt que l'un des services cloud, mais ils fonctionnent de façon similaire.

De plus, de nombreux éditeurs de programmation basés sur Windows fournissent une intégration directe entre l'environnement d'édition de texte et les référentiels Git qui stockent les fichiers de projet. Un environnement populaire est Visual Studio Code, qui est utilisé pour ce cours.

La section suivante fournit un aperçu rapide de l'utilisation d'un référentiel Git à l'aide de Visual Studio Code. Dans la suite de cette section, vous en apprendrez davantage sur Red Hat Ansible Tower.

## Présentation de Red Hat Ansible Tower

Le fait de partager une infrastructure Ansible existante pour effectuer une mise à l'échelle de l'automatisation des services informatiques peut présenter certains défis. Bien que vous puissiez tirer parti de playbooks Ansible correctement dans différentes équipes, Ansible ne fournit aucune fonctionnalité permettant de gérer l'accès partagé. En outre, bien que les playbooks puissent autoriser la délégation de tâches complexes, l'exécution de celles-ci peut exiger certaines informations d'identification d'administrateur avec des niveaux de privilèges et de protection élevés. Red Hat Ansible Tower limite ces défis en fournissant un cadre efficace d'exécution et de gestion d'Ansible à l'échelle d'une entreprise.

Ansible Tower facilite l'administration résultant du partage d'une infrastructure Ansible tout en maintenant la sécurité de l'organisation en introduisant des fonctionnalités telles qu'une interface utilisateur Web centralisée par la gestion des playbooks, un *contrôle d'accès basé sur les rôles (RBAC)* et une journalisation et un audit centralisés.

L'API REST vous permet d'intégrer Ansible Tower aux workflows et aux ensembles d'outils organisationnels existants. L'API et les fonctionnalités de notification d'Ansible Tower permettent d'associer très facilement des playbooks Ansible à d'autres outils comme Jenkins, Red Hat CloudForms ou Red Hat Satellite pour permettre une intégration et un déploiement continu. Elle fournit des mécanismes permettant l'utilisation et le contrôle centralisés des informations d'identification des machines et autres secrets sans les exposer aux utilisateurs d'Ansible Tower.

L'abonnement Red Hat Ansible Automation offre une prise en charge à la fois pour Red Hat Ansible Tower et Red Hat Ansible Engine. Pour plus d'informations, consultez <https://www.redhat.com/en/technologies/management/ansible>.

## Architecture Red Hat Ansible Tower

Red Hat Ansible Tower est une application web Django conçue pour fonctionner sur un serveur Linux en tant que solution autohébergée sur site qui se dispose en couche au-dessus d'une infrastructure Ansible d'entreprise existante.

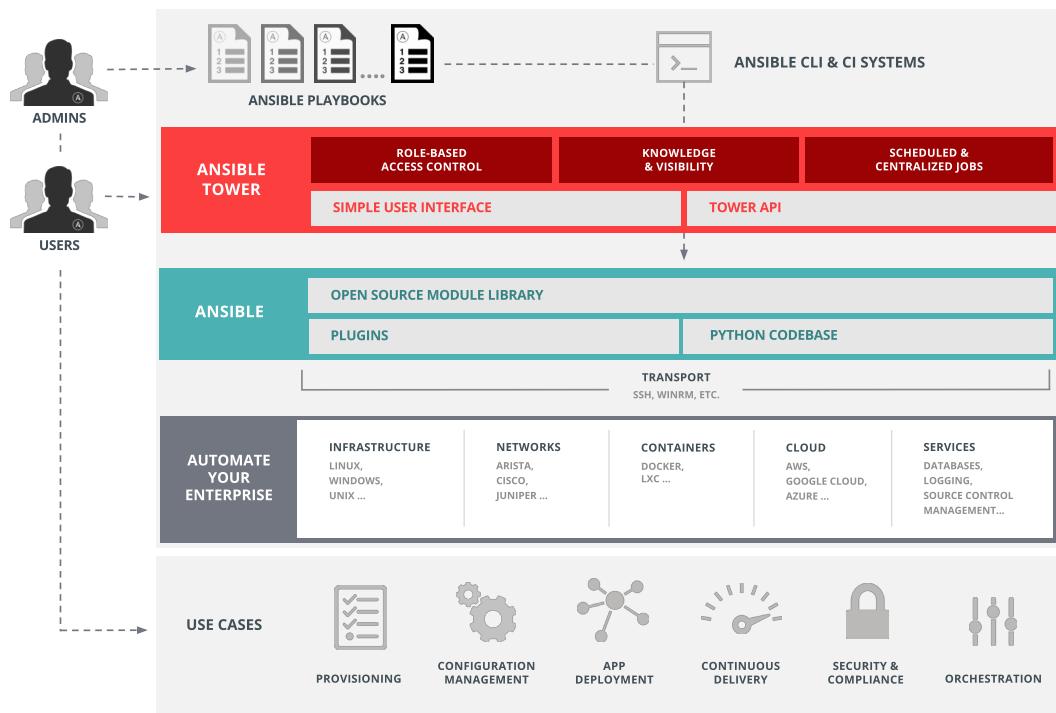


Figure 1.4: Architecture d'Ansible Tower

Vous pouvez interagir avec l'infrastructure Ansible sous-jacente pour l'entreprise via l'interface utilisateur Web d'Ansible Tower ou via l'API RESTful. L'interface utilisateur Web d'Ansible Tower est une interface graphique qui effectue des actions en exécutant des appels vers l'API Ansible Tower. Toute action disponible via l'interface utilisateur web d'Ansible Tower peut également être effectuée via l'API RESTful d'Ansible Tower. L'API RESTful est essentielle pour les utilisateurs cherchant à intégrer Ansible avec des outils et processus logiciels existants.

Ansible Tower stocke ses données dans une base de données backend PostgreSQL et utilise le système de messagerie RabbitMQ. Les versions d'Ansible Tower antérieures à la version 3.0 se basaient également sur une base de données MongoDB. Cette dépendance a depuis lors été supprimée et les données sont désormais stockées uniquement dans une base de données PostgreSQL.

Selon les besoins de votre entreprise, Ansible Tower peut être mis en œuvre à l'aide d'une des architectures suivantes :

- Une seule machine avec une base de données intégrée
- Une seule machine avec une base de données distante
- Un cluster de plusieurs machines haute disponibilité
- Pod OpenShift avec base de données distante

#### Note

Ce cours se concentre sur l'architecture la plus directe à déployer : à savoir, un seul serveur Red Hat Ansible Tower avec une base de données intégrée.

## Fonctions de Red Hat Ansible Tower

Les fonctions suivantes sont quelques-unes des fonctions proposées par Red Ansible Tower pour contrôler, sécuriser et gérer Ansible dans un environnement d'entreprise :

### Coffres-forts d'informations d'identification externes

Ansible Tower 3.5 prend en charge les coffres-forts d'informations d'identification externes. Cela vous permet de gérer les informations d'identification d'un grand nombre de fournisseurs, notamment HashiCorp Vault, CyberArk AIM et CyberArk Conjur, ainsi que le coffre-fort de clés Microsoft Azure.

### Tableau de bord visuel

L'interface utilisateur web d'Ansible Tower affiche un tableau de bord qui fournit une vue de synthèse de l'intégralité de l'environnement Ansible pour l'entreprise. Le tableau de bord Ansible Tower vous permet de voir facilement l'état actuel des hôtes et des inventaires, ainsi que les résultats des exécutions de tâches récentes.

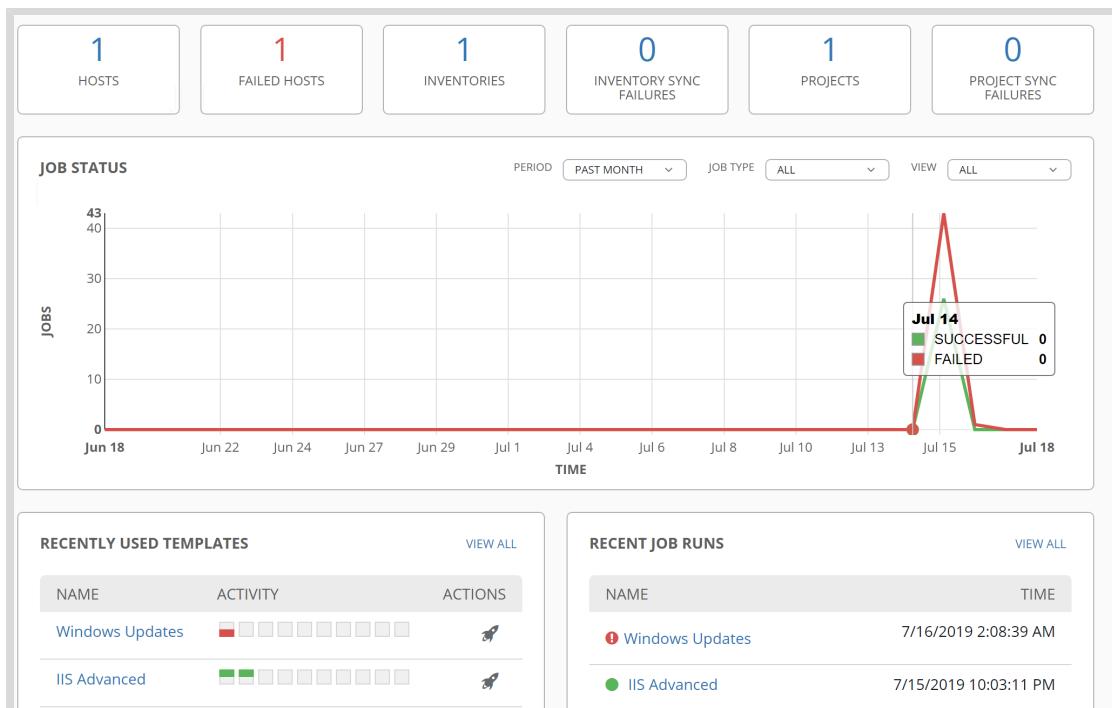


Figure 1.5: Le tableau de bord d'Ansible Tower

### Contrôle d'accès basé sur les rôles

Ansible Tower utilise un système de contrôle d'accès basé sur les rôles (RBAC) qui maintient la sécurité tout en rationalisant la gestion de l'accès des utilisateurs. Ce système simplifie la délégation de l'accès des utilisateurs aux objets Ansible Tower comme les organisations, les projets et les inventaires.

### Gestion d'inventaire graphique

Utilisez l'interface utilisateur Web d'Ansible Tower pour créer des groupes d'inventaire et ajouter des hôtes d'inventaire. Mettez à jour les inventaires depuis une source d'inventaire externe, comme des fournisseurs de cloud public, des environnements de virtualisation locaux et la base de données de gestion de la configuration (CMDB, configuration management database) personnalisée d'une organisation.

## Planification des tâches

Utilisez Ansible Tower pour planifier l'exécution de playbooks et les mises à jour depuis des sources de données externes soit de façon ponctuelle, soit à intervalles réguliers. Cela vous permet d'effectuer des tâches de routine sans surveillance et s'avère particulièrement utile pour des tâches telles que les routines de sauvegarde, qui devraient dans l'idéal être exécutées hors des heures d'exploitation.

## Création de rapports d'état des tâches en temps réel et historiques

Lorsque vous lancez une exécution de playbook dans Ansible Tower, l'interface utilisateur web affiche la sortie du playbook et les résultats d'exécution en temps réel. Les résultats des tâches exécutées précédemment et des exécutions de tâches planifiées sont également disponibles dans Ansible Tower.

The screenshot shows two main panels. On the left, a 'DETAILS' panel displays job information: STATUS (Successful), STARTED (7/15/2019 10:16:50 PM), FINISHED (7/15/2019 10:16:54 PM), JOB TYPE (Check), PROJECT (Ansible Workshop Project), CREDENTIAL (Git Credential), EXECUTION NODE (localhost), and INSTANCE GROUP (tower). On the right, a larger panel titled 'Ansible Workshop Project' shows a log of task execution. The log includes:

```

43 TASK [Repository Version] ****
44 ****
45 ok: [localhost,] => {
46     "msg": "Repository Version 6c356fcfc347f5c964b1bdd9f6b0d10c7
47 4344a0"
48 }
49 TASK [Write Repository Version] ****
50 ****
51 changed: [localhost, -> localhost] => {"changed": true, "checksum": "7
52 d9813fb1cef945afe686313c01178588a72d9eb", "dest": "/var/lib/awx/projec
ts/tmp2t_8vl5q", "gid": 993, "group": "awx", "md5sum": "767f3e1bad91d8
92be70d04b3af84c4", "mode": "0600", "owner": "awx", "secontext": "sys
tem_u:object_r:var_lib_t:s0", "size": 40, "src": "/var/lib/awx/.ansib
le/tmp/ansible-tmp-1563229013.76-11223192835061/source", "state": "fil
e", "uid": 996}
53 PLAY [all] ****
54 ****

```

**Figure 1.6: Gestion des tâches dans Ansible Tower**

## Automatisation déclenchée par l'utilisateur

Ansible simplifie l'automatisation des services informatiques et Ansible Tower va encore plus loin en permettant aux utilisateurs le libre service. L'interface utilisateur Web simplifiée d'Ansible Tower, associée à la flexibilité de son système RBAC, vous permet de réduire des tâches complexes à des routines simples faciles à utiliser.

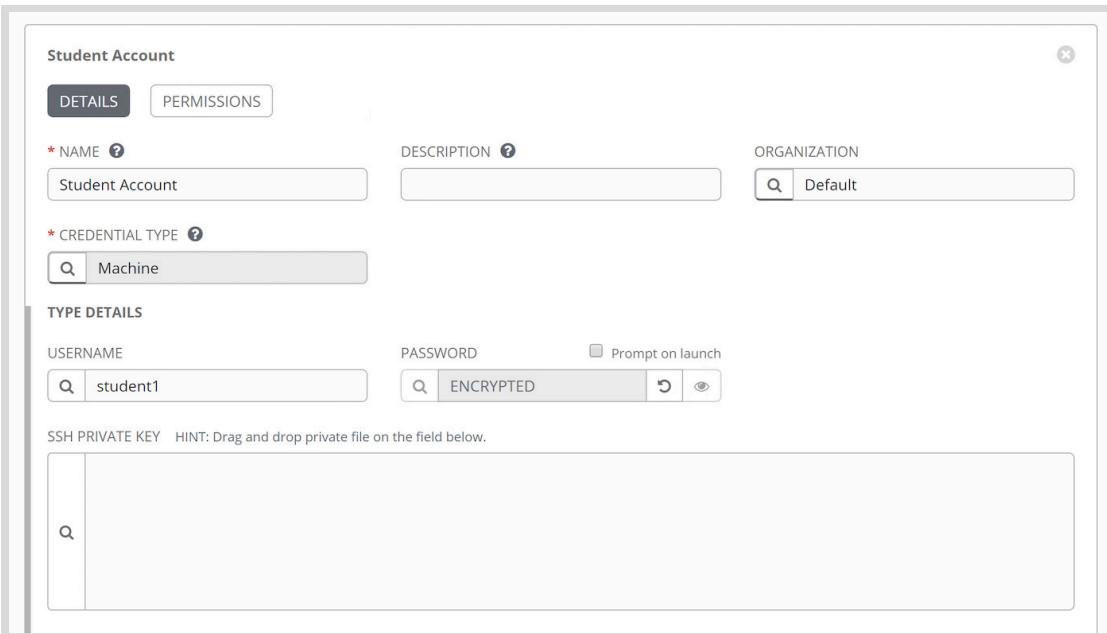
## Exécution de commandes à distance

Ansible Tower rend la flexibilité à la demande des commandes ad hoc d'Ansible disponible par le biais de sa fonctionnalité d'exécution de commandes à distance. Les autorisations d'utilisateurs pour l'exécution de commandes à distance sont mises en application à l'aide du système RBAC d'Ansible Tower.

## Gestion des informations d'identification

Ansible Tower gère centralement les informations d'identification d'authentification. Cela signifie que vous pouvez exécuter les plays Ansible sur des hôtes gérés, synchroniser des informations à partir de sources d'inventaire dynamiques et importer du contenu de projet Ansible à partir de systèmes de contrôle de version. Ansible Tower chiffre les mots de passe ou les clés fournis de sorte qu'ils ne puissent pas être récupérés par les utilisateurs Ansible Tower. Vous pouvez accorder aux utilisateurs la possibilité d'utiliser ou de remplacer ces informations d'identification sans réellement les exposer à l'utilisateur.

Ansible prend en charge plus de 15 types d'informations d'identification différents, ce qui inclut des fournisseurs majeurs tels que Google Compute Engine, Microsoft Azure et VMware.



**Figure 1.7: Gestion des informations d'identification Ansible Tower**

#### Journalisation et audit centralisés

Ansible Tower consigne tous les playbooks et l'exécution des commandes à distance. Cela vous permet de vérifier quand et par qui chaque tâche a été effectuée. En outre, Ansible Tower offre la possibilité d'intégrer les données de son journal dans des solutions d'agrégation de journalisation de tiers, telles que Splunk et Sumologic.

#### Notifications intégrées

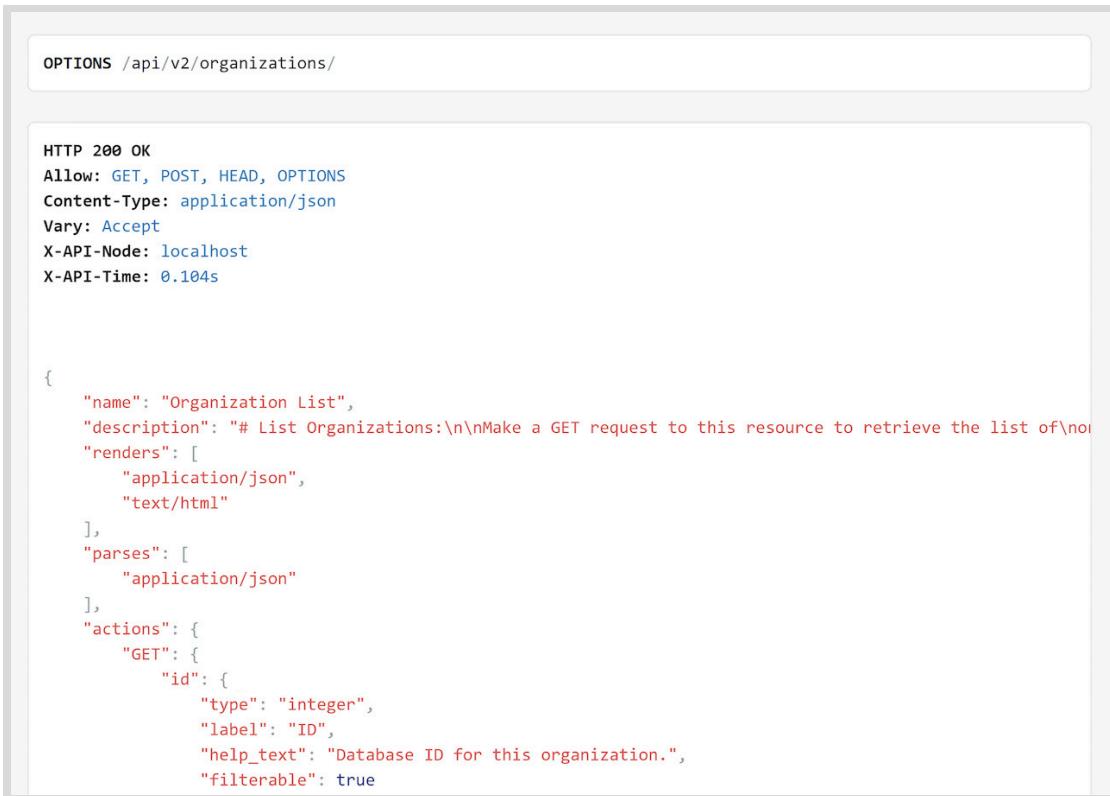
Ansible Tower vous avertit en cas de réussite ou d'échec d'exécution des tâches. Ansible Tower peut fournir des notifications au moyen de nombreuses applications, dont l'e-mail, Slack et HipChat.

#### Workflows à plusieurs playbooks

Les opérations complexes impliquent souvent l'exécution en série de plusieurs playbooks. Les workflows à plusieurs playbooks d'Ansible Tower permettent aux utilisateurs d'enchaîner plusieurs playbooks pour faciliter l'exécution de routines complexes impliquant la mise en service, la configuration, le déploiement et l'orchestration. Un éditeur de workflows intuitif simplifie également la modélisation des workflows à plusieurs playbooks.

#### API RESTful

L'API RESTful d'Ansible Tower expose toutes les fonctions Ansible Tower disponibles par le biais de l'interface utilisateur Web. Le format navigable de l'API la rend auto-documentée et simplifie la recherche d'informations relatives à l'utilisation de l'API.



```

OPTIONS /api/v2/organizations/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
X-API-Node: localhost
X-API-Time: 0.104s

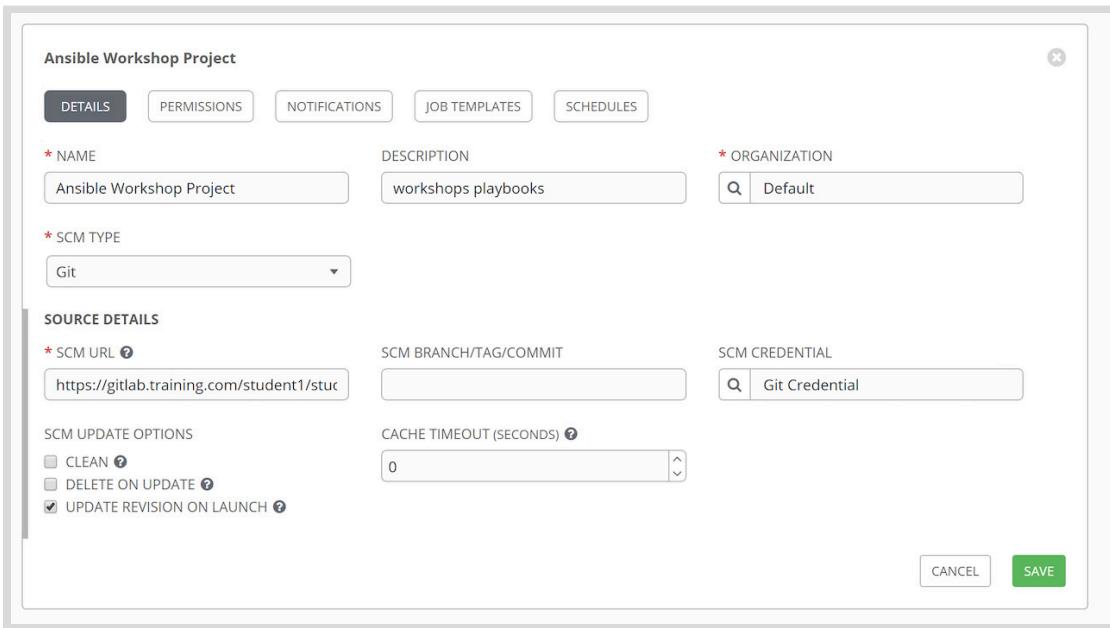
{
    "name": "Organization List",
    "description": "# List Organizations:\n\nMake a GET request to this resource to retrieve the list of\norganizations.\n",
    "renders": [
        "application/json",
        "text/html"
    ],
    "parses": [
        "application/json"
    ],
    "actions": {
        "GET": {
            "id": {
                "type": "integer",
                "label": "ID",
                "help_text": "Database ID for this organization.",
                "filterable": true
            }
        }
    }
}

```

Figure 1.8: Interaction avec l'API Ansible Tower

## Interaction avec les référentiels Git

Les projets Ansible Tower peuvent récupérer vos playbooks à partir d'un système de contrôle de version tel que Git, SVN ou Mercurial et de les exécuter. Pour un référentiel Git, vous pouvez extraire du code d'une branche, d'une balise ou d'une validation spécifique, et exécuter cette version du code. Vous pouvez également configurer Ansible Tower pour fournir les informations d'authentification dont elle a besoin pour accéder au référentiel.



The screenshot shows the 'Ansible Workshop Project' configuration page. The 'DETAILS' tab is selected. The project name is 'Ansible Workshop Project', the description is 'workshops playbooks', and it is associated with the 'Default' organization. The 'SCM TYPE' is set to 'Git'. Under 'SOURCE DETAILS', the 'SCM URL' is 'https://gitlab.training.com/student1/stuc', and the 'SCM BRANCH/TAG/COMMIT' field is empty. The 'SCM CREDENTIAL' is 'Git Credential'. Under 'SCM UPDATE OPTIONS', the 'UPDATE REVISION ON LAUNCH' checkbox is checked. At the bottom right are 'CANCEL' and 'SAVE' buttons.

Figure 1.9: Gestion de VCS avec Ansible Tower

Grâce à cette approche flexible, vous pouvez tester vos modifications à partir d'une branche de développement par rapport à un ensemble de serveurs de test. En fonction des résultats, vous pouvez décider de fusionner vos modifications dans la branche principale pour les utiliser dans l'environnement de production.



## Références

Pour plus d'informations, reportez-vous au *Guide d'administration d'Ansible Tower* à l'adresse  
<https://docs.ansible.com/ansible-tower/latest/html/administration/>

### Gérer Windows comme Linux avec Ansible

<https://www.redhat.com/en/about/videos/summit-2018-manage-windows-linux-ansible>

### Intégration : Ansible et Windows

<https://www.ansible.com/integrations/infrastructure/windows>

### Vue d'ensemble de la configuration de l'état souhaité de Windows PowerShell

<https://docs.microsoft.com/en-us/powershell/dsc/overview/overview>

## ► Exercice guidé

# Architecture de l'automatisation Windows avec Red Hat Ansible Tower

Au cours de cet exercice, vous allez vous connecter à un serveur Red Hat Ansible Tower, explorer son interface utilisateur Web, et vous familiariser avec celle-ci, puis l'utiliser pour lancer un modèle d'automatisation existant.

## Résultats

Vous devez être en mesure de vous connecter à l'interface utilisateur web Ansible Tower, d'explorer le panneau de configuration et de lancer un modèle à partir de la console.

## Avant De Commencer

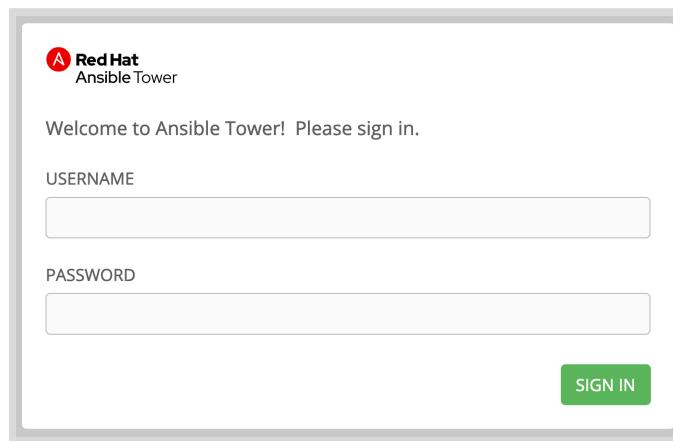
Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.

### ► 1. Accédez à l'interface utilisateur web Ansible Tower.

- 1.1. Double-cliquez sur le raccourci du Bureau nommé Ansible Tower et authentifiez-vous avec l'utilisateur **admin** et le mot de passe **RedHat123@!**.



### ► 2. Passez en revue la configuration d'Ansible Tower.

- 2.1. Dans le tableau de bord, cliquez sur **Users** dans la barre latérale pour afficher les comptes disponibles dans Ansible Tower.

Ce menu affiche les comptes disponibles, y compris le compte **admin** en cours d'utilisation.

- 2.2. Cliquez sur **Settings** au bas de la barre latérale de la console pour accéder à la configuration d'Ansible Tower.

- 2.3. Cliquez sur **Authentication** pour afficher la configuration actuelle.

À partir de ce panneau, vous pouvez configurer l'authentification à partir de divers fournisseurs et applications, tels que GitHub ou LDAP. Cliquez sur le bouton **LDAP** pour afficher la configuration actuelle de votre instance.

Dans une configuration de production, configurez l'implémentation de l'authentification qui convient le mieux à votre environnement.

- 2.4. Revenez à la page **Settings**, puis cliquez sur **License** pour afficher les informations de licence pour Ansible Tower.

À partir de ce panneau, vous pouvez consulter les informations relatives aux licences pour cette instance d'Ansible Tower, ainsi que télécharger un nouveau fichier de licence.

► 3. Accédez aux modèles de tâche et exécutez-les.

- 3.1. Dans la barre latérale de la console, cliquez sur **Templates** pour accéder aux modèles disponibles ou en configurer d'autres. Les modèles sont un ensemble d'instructions qui exécutent des playbooks Ansible et effectuent des tâches à l'aide des modules Ansible.
- 3.2. Dans la liste **Templates**, sélectionnez le modèle **Run architecture project**.

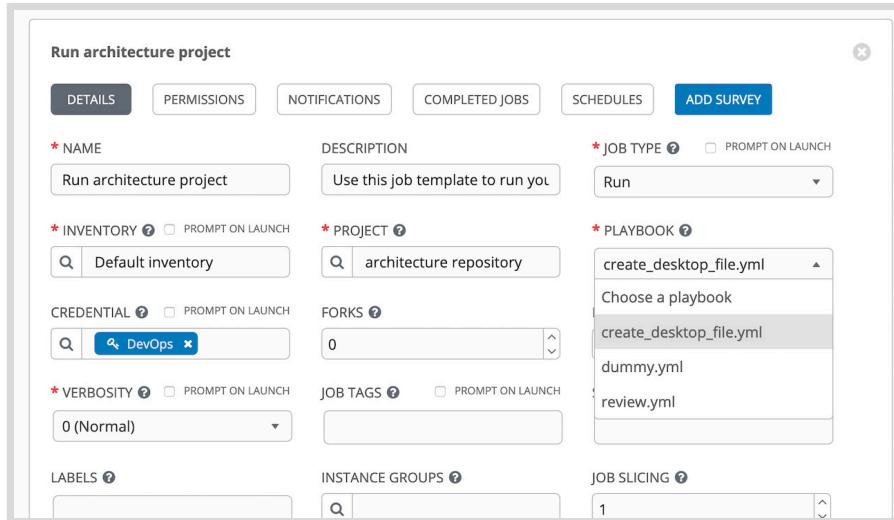
 **Note**

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 3.3. Sélectionnez le playbook **create\_desktop\_file.yml** dans la liste **PLAYBOOK**, puis cliquez sur **Save** pour mettre à jour le modèle de tâche.

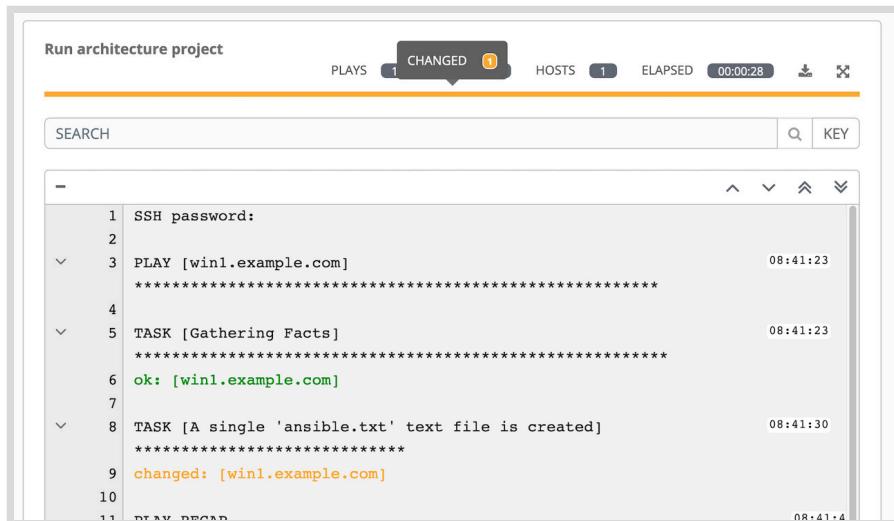
Le playbook indique à Ansible de créer un fichier texte **ansible.txt** sur le Bureau **devops** du serveur **win1.example.com**.

Sélectionnez **LAUNCH** pour commencer l'exécution.



- 3.4. Pendant l'exécution de la tâche, Ansible Tower affiche la sortie des tâches à mesure qu'elles sont exécutées.

Ansible récupère des informations relatives au système, puis exécute les tâches. Une tâche dont l'état est **changed** indique que le fichier n'était pas présent et que la tâche le crée.



La sortie affiche également les statistiques relatives aux tâches, telles que l'heure de début et les informations du modèle en cours d'utilisation.

- 3.5. Une fois l'opération terminée, l'état passe de **Running** à **Successful**. Si la tâche rencontre des problèmes, cet état passe à **Failed** et la sortie contient des informations sur les erreurs.

DETAILS	
STATUS	● Successful
STARTED	8/22/2019 8:41:19 AM
FINISHED	8/22/2019 8:41:47 AM
JOB TEMPLATE	<a href="#">Run architecture project</a>
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	<a href="#">Default inventory</a>
PROJECT	● <a href="#">architecture repository</a>
REVISION	8b4b33b
PLAYBOOK	<a href="#">create_desktop_file.yml</a>
CREDENTIAL	DevOps
ENVIRONMENT	/var/lib/awx/venv/ansible

Sélectionnez une entrée dans la sortie pour accéder à plus d'informations sur le résultat de l'exécution.

The screenshot shows a detailed view of an Ansible task execution. The task was created on 8/22/2019 at 8:41:47 AM, has ID 74, and is part of a play named 'win1.example.com'. The task itself is a 'win\_copy' module that created a single 'ansible.txt' file. The JSON output of this task is displayed below, showing the file path and other metadata.

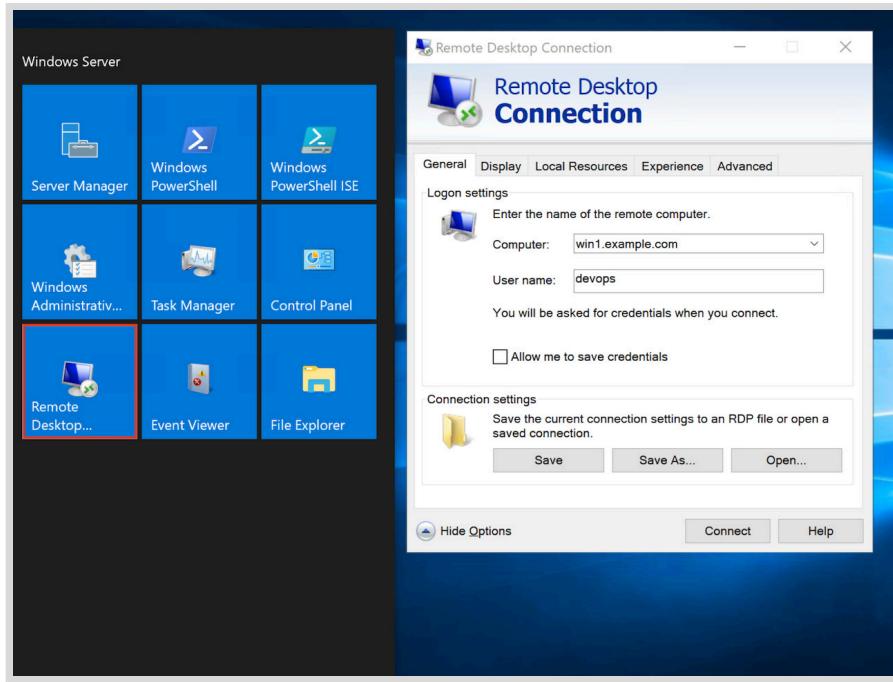
```

1 {
2   "src": null,
3   "_ansible_no_log": false,
4   "dest": "C:\\\\Users\\\\student\\\\desktop\\\\ansible.txt",
5   "checksum": "0b3343554d393d8e0ca219194d018b912139c9ab",
6   "changed": true,
7   "original_basename": "tmp2AAIph",
8   "operation": "file_copy",
9   "size": 47
10 }
  
```

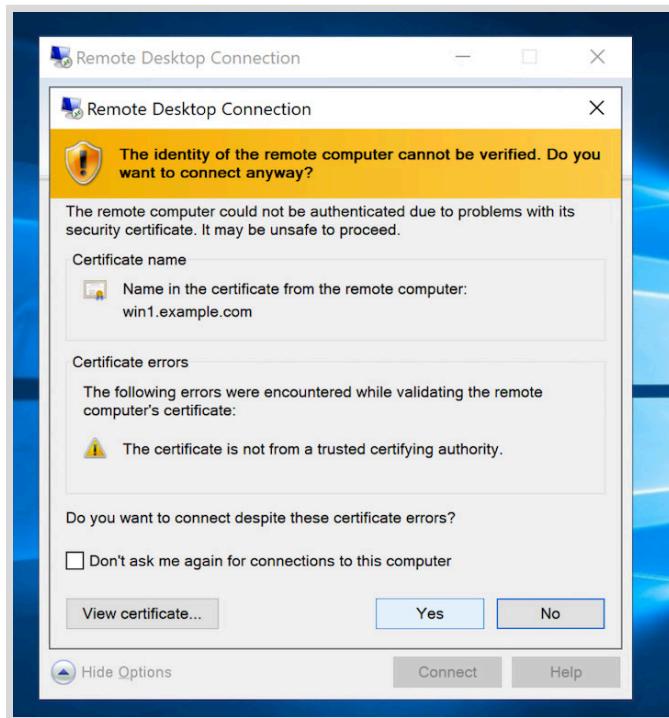
#### ► 4. Accédez au fichier texte **ansible.txt**.

- 4.1. Pour vous assurer que le fichier est présent sur **win1.example.com**, accédez au client **Connexion Bureau à distance** disponible dans le menu **Démarrer**. Le client est accessible à partir de la liste des applications **Windows Server**.

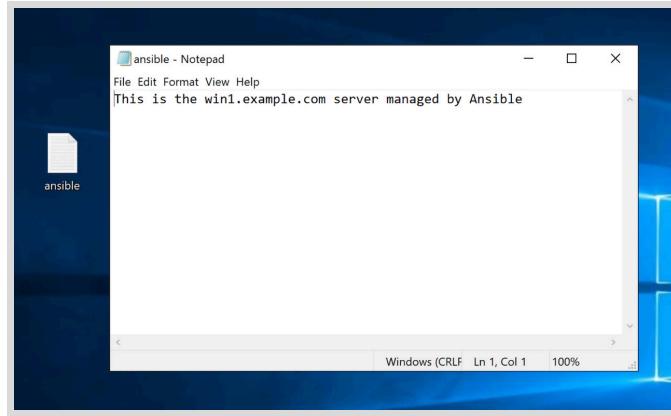
Accédez à **win1** en utilisant **devops** comme nom d'utilisateur. Cliquez sur **Connect** pour lancer la connexion. Lorsque vous y êtes invité, utilisez le mot de passe **RedHat123 @!**, puis cliquez sur **OK**.



4.2. Acceptez le certificat auto-signé en sélectionnant **Yes**.



4.3. Assurez-vous que le fichier texte **ansible.txt** est présent sur le Bureau.



L'exercice guidé est maintenant terminé.

# Gestion de fichiers dans Git avec Visual Studio Code

## Résultats

Au terme de cette section, vous serez en mesure de récupérer, gérer, modifier et stocker des fichiers dans un système de contrôle de version Git distant, tel que GitHub ou GitLab, à l'aide de Visual Studio Code.

## Infrastructure-as-Code

L'un des concepts clés de DevOps est la notion d'infrastructure sous forme de code (IaC, Infrastructure-as-Code), abordée précédemment dans ce chapitre. Au lieu de gérer votre infrastructure manuellement, vous définissez et créez vos systèmes en exécutant votre code d'automatisation. Red Hat Ansible Automation est un outil clé qui peut vous aider à mettre en œuvre cette approche.

Si les projets Ansible sont le code utilisé pour définir l'infrastructure, un *système de contrôle de version* tel que Git doit être utilisé pour suivre et contrôler les modifications apportées au code.

Le contrôle de version vous permet de mettre en œuvre un cycle de vie pour les différents stades de votre code d'infrastructure, comme le développement, l'assurance qualité et la production. Vous pouvez valider les modifications apportées à une branche, puis tester ces modifications dans les environnements de développement et d'assurance qualité non critiques. Une fois que les modifications vous conviennent, vous pouvez les fusionner avec le code de production principal et appliquer les modifications à votre infrastructure de production.

## Présentation de Git

Git est un *système de contrôle de version distribué* (DVCS, *Distributed Version Control System*) qui vous permet de gérer les modifications apportées aux fichiers d'un projet de manière collaborative. Chaque révision d'un fichier est validée dans le système. Les anciennes versions des fichiers peuvent être restaurées, et un journal de l'auteur des modifications est conservé.

Les systèmes de contrôle de version présentent de nombreux avantages, notamment :

- la possibilité de réviser et de restaurer les anciennes versions des fichiers ;
- la possibilité de comparer deux versions du même fichier pour identifier les modifications ;
- un enregistrement ou un journal de l'auteur des modifications, ainsi que le moment où ces modifications ont été apportées ;
- des mécanismes permettant à plusieurs utilisateurs de modifier des fichiers de manière collaborative, de résoudre les modifications conflictuelles et de fusionner les modifications ensemble.

Chaque développeur peut commencer par *cloner* un projet partagé existant à partir d'un *référentiel distant*. Le clonage d'un projet crée une copie complète du dépôt distant d'origine en tant que *dépôt local*. Il s'agit d'une copie locale de l'intégralité de l'historique des fichiers dans le système de contrôle de version, et pas seulement de l'instantané le plus récent des fichiers de projet.

Le développeur apporte des modifications dans une *arborescence de travail*, qui est une zone de fonctionnement pour les fichiers nouveaux et modifiés. Un ensemble de modifications connexes est ensuite indexé et validé dans le référentiel local. À ce stade, aucune modification n'a été apportée au dépôt distant partagé.

Lorsque le développeur est prêt à partager son activité, il *transmet par push* les modifications au dépôt distant. Sinon, si le référentiel local est accessible à partir du réseau, le propriétaire du référentiel distant peut *extraire* les modifications du référentiel local du développeur vers le référentiel distant.

De même, lorsqu'un développeur est prêt à mettre à jour son référentiel local avec les dernières modifications apportées au référentiel distant, il peut extraire les modifications du référentiel distant et les fusionner dans le référentiel local.

Pour utiliser Git de manière efficace, un utilisateur doit connaître les trois états possibles d'un fichier dans l'arborescence de travail : *modifié*, *indexé* ou *validé*.

- *Modifié* : la copie du fichier dans l'arborescence de travail a été éditée et diffère de la version la plus récente figurant dans le dépôt.
- *Indexé* : le fichier modifié a été ajouté à une liste de fichiers modifiés pour être validé en tant qu'ensemble, mais il n'a pas encore été validé.
- *Validé* : le fichier modifié a été validé dans le dépôt local.

Une fois le fichier validé dans le dépôt local, la validation peut être transmise par push au dépôt distant ou extraite par celui-ci.

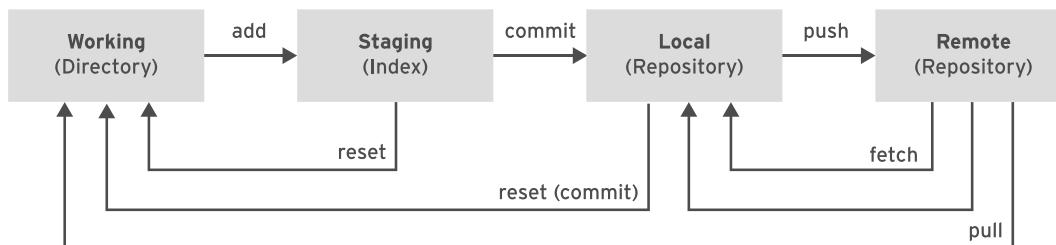


Figure 1.23: Les quatre zones dans lesquelles Git gère les fichiers

Un lien vers des informations détaillées sur Git se trouve dans les références à la fin de cette section.

## Présentation de Visual Studio Code

Visual Studio Code est un *environnement de développement intégré (IDE)* multiplateformes et léger qui permet de modifier et de déboguer du code. Le projet lui-même est principalement développé sous une licence MIT Open Source, sur <https://github.com/microsoft/vscode>, mais les versions officielles incluent des personnalisations publiées sous une licence Microsoft propriétaire. Les scripts de compilation de fichiers binaires entièrement Open Source basés sur ce projet, dont tous les codes propriétaires sont supprimés, sont gérés sur <https://github.com/VSCodium/vscodium>.

Dans ce cours, vous allez utiliser Visual Studio Code comme éditeur principal pour les fichiers Ansible et outil pour travailler avec les référentiels Git.

D'autres éditeurs de texte sont disponibles sous Windows avec la prise en charge de Git, y compris Atom et Sublime Text.

## Installation de Git sur Microsoft Windows

Git pour Windows est disponible à l'adresse <https://git-scm.com/download/win>. L'installation est simple. Si vous utilisez Git avec Visual Studio Code, vous devez sélectionner Visual Studio Code comme éditeur par défaut lorsque vous y êtes invité.

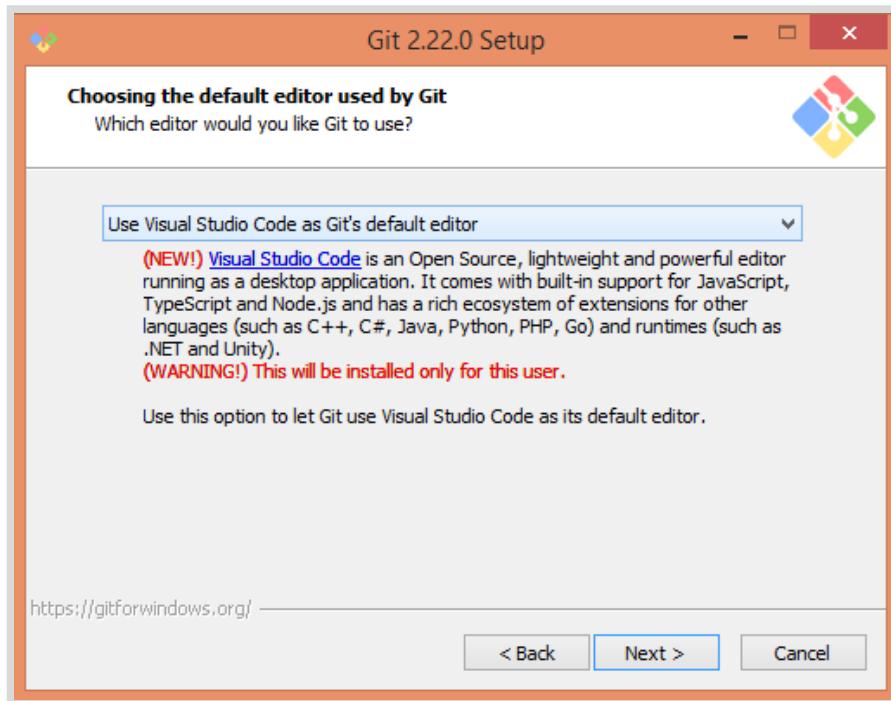


Figure 1.24: L'éditeur par défaut pour Git

Après avoir installé Git, exécutez le programme Git GUI accessible à partir du menu de démarrage rapide. Après le clonage d'un référentiel, vous pouvez configurer les paramètres de nom d'utilisateur et d'e-mail pour le référentiel, ou globalement.

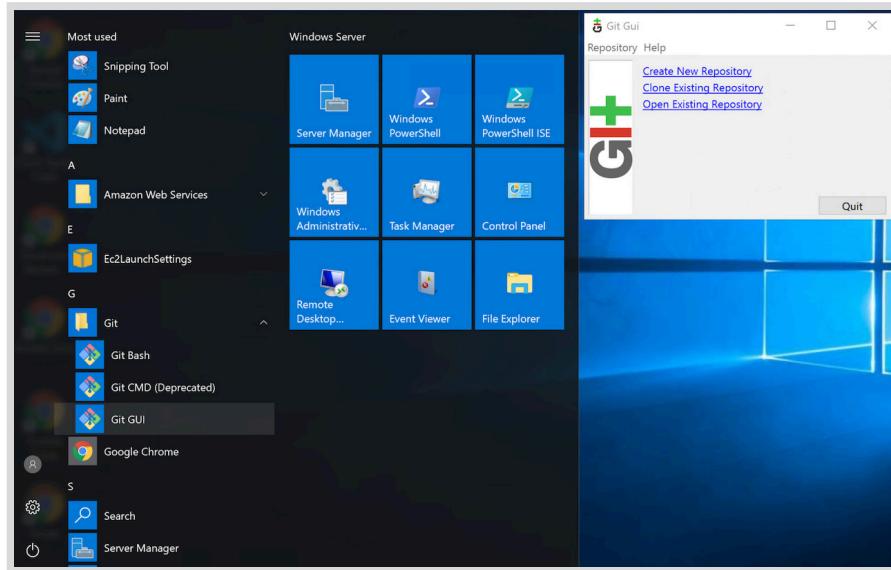


Figure 1.25: Accès à Git GUI

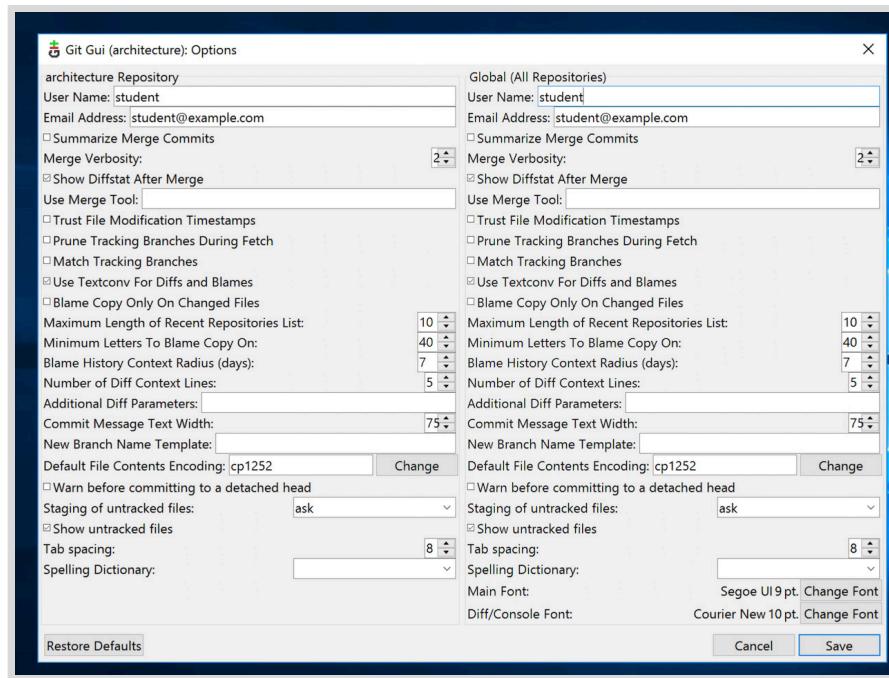


Figure 1.26: Gestion des options Git par défaut à l'aide de Git GUI

## Configuration de Visual Studio Code

La majeure partie de l'intégration entre Visual Studio Code et Git s'effectue au sein de Visual Studio Code. Accédez à **File** → **Preferences** → **Settings**, développez **Extensions**, puis sélectionnez **Git**. Faites défiler jusqu'au paramètre **Path**, puis sélectionnez le lien **Edit in settings.json** dans la fenêtre principale. Cela ouvre le fichier JSON qui stocke les paramètres.

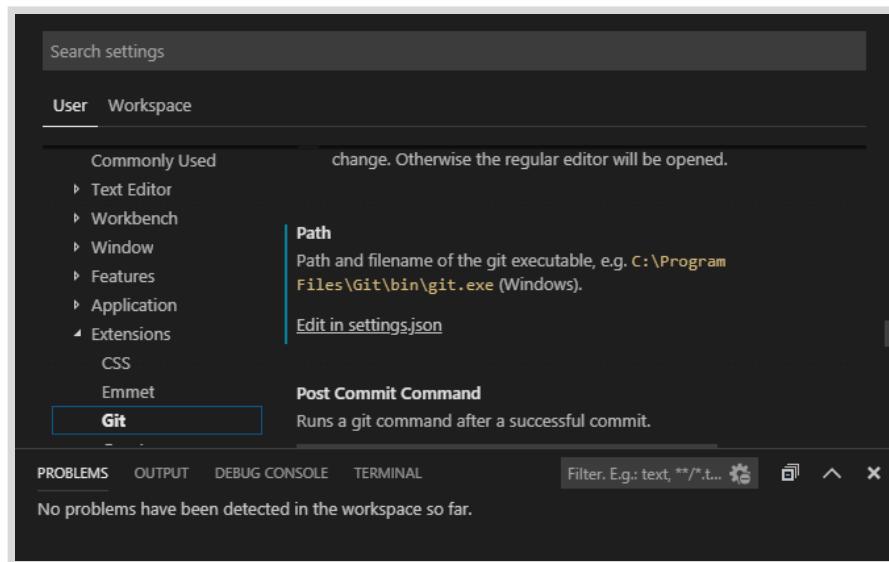


Figure 1.27: Configurer les paramètres Git avec Visual Studio Code

Ajoutez l'option **Path** avec la valeur correspondant au chemin d'accès absolu de l'exécutable Git.

**Note**

Veillez à échapper toutes les barres obliques inverses à l'aide d'une barre oblique inverse supplémentaire, comme illustré dans la figure suivante.

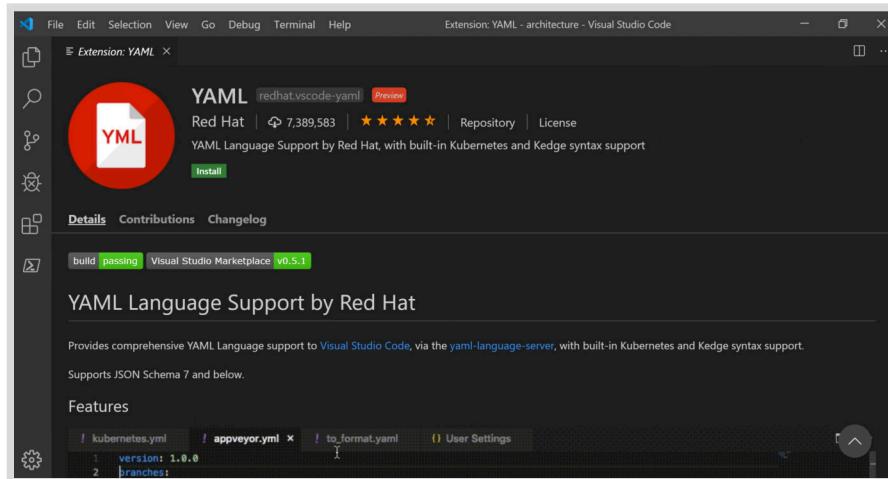
```
C: > Users > user > AppData > Roaming > Code > User > settings.json > ...
1  {
2    "scm.alwaysShowProviders": true,
3    "scm.alwaysShowActions": true,
4    "git.alwaysShowStagedChangesResourceGroup": true,
5    "git.fetchOnPull": true,
6    "git.path": "C:\\Program Files\\Git\\bin\\git.exe"
7 }
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    Filter. E.g.: text, \*\*/\*.t...   

No problems have been detected in the workspace so far.

**Figure 1.28: Configurer le chemin d'accès de l'exécutable Git.**

Étant donné que vous travaillez avec des playbooks Ansible, qui sont au format YAML, installez l'extension YAML en accédant à **View → Extensions**, puis recherchez ou faites défiler vers le bas jusqu'à l'extension YAML. Cliquez sur **Install**.



**Figure 1.29: Installez l'extension YAML**

Vous pouvez cloner un référentiel Git dans Visual Studio Code en accédant à la *palette de commandes*, qui est accessible en sélectionnant **View → Command Palette** ou en appuyant sur **Ctrl+Maj+P** sur le clavier. Saisissez **Git: Clone** pour accéder à la fonction de clone Git.

La palette de commandes vous invite à saisir l'URL du référentiel distant et le répertoire parent prévu pour le référentiel local.

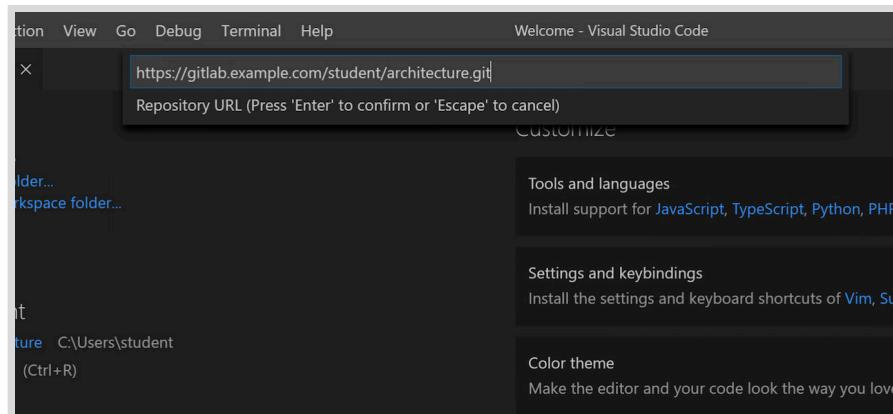


Figure 1.30: Clonage des référentiels Git à l'aide de la palette de commandes

## Gestion de playbooks dans Visual Studio Code

Pour les besoins de cet exemple, supposons que vous avez cloné un référentiel à partir d'un serveur Git central. Accédez à **File** → **Open Folder...**, sélectionnez l'emplacement du référentiel cloné, puis cliquez sur **Select Folder**. La vue **Explorer** doit être active et afficher le contenu du référentiel cloné.

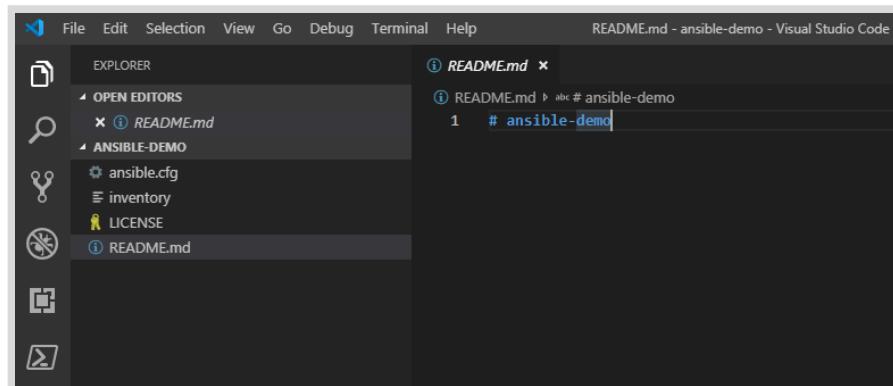


Figure 1.31: Affichage Explorer

Notez les icônes de statut de contrôle de version dans la partie inférieure gauche de l'écran. Elles indiquent actuellement qu'il n'y a eu aucune modification.

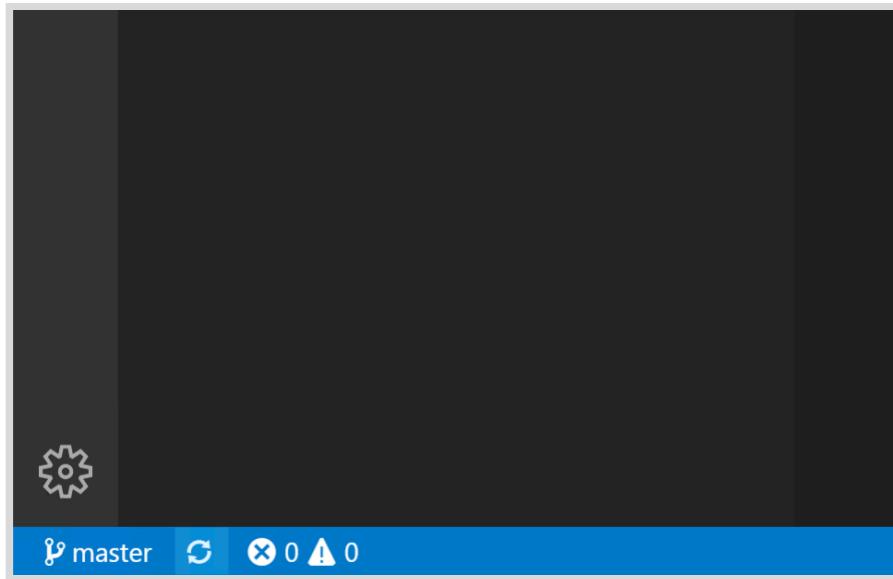


Figure 1.32: Icônes de statut de contrôle de version

L'ajout d'un nouveau fichier **site.yml** entraîne l'ajout d'une notification à l'icône **Source Control**, ainsi que l'affichage d'un **U** en regard de **site.yml**. Le **U** indique que **site.yml** est **untracked (non suivi)**, ce qui signifie que Git ne vérifie pas s'il y a eu des modifications. Si vous accédez à l'affichage **Source Control**, vous verrez **site.yml** apparaître sous **Changes**.

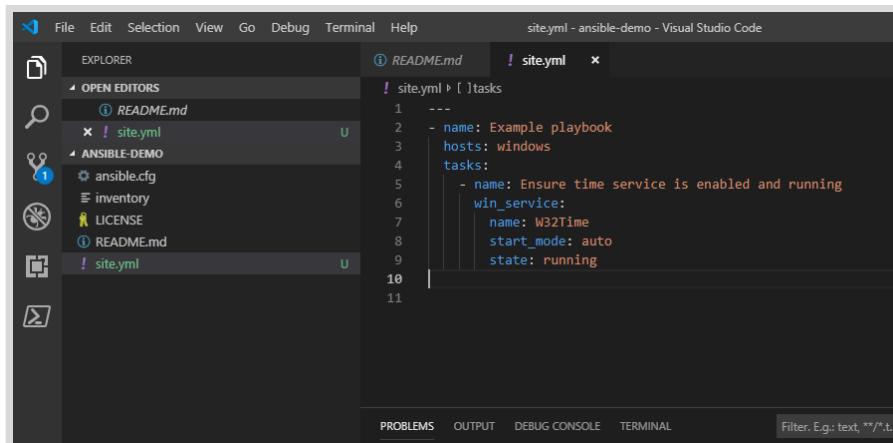


Figure 1.33: Un nouveau fichier est créé

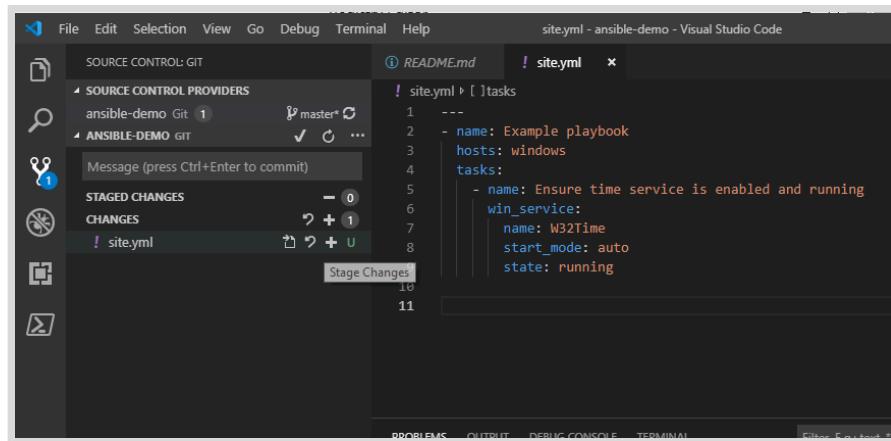


Figure 1.34: Le nouveau fichier listé sous Changes dans l'affichage Source Control

Voici quelques-uns des codes communs pour les fichiers modifiés :

Indicateur	Description
U (Untracked)	Ce fichier n'est pas encore sous contrôle de version.
M (Modified)	Ce fichier est suivi par git et a été modifié.
A (Added)	Ce fichier a été ajouté à l'index et sera intégré lors de la validation suivante.
D (Deleted)	Ce fichier a été supprimé. Le fichier n'apparaît plus dans l'affichage Explorer, uniquement dans l'affichage Source Control. La suppression du fichier sera incluse lors de la validation suivante.

Cliquez sur **+** en regard de **site.yml** pour ajouter le fichier à Git, puis saisissez un message de validation décrivant vos modifications. Idéalement, votre message de validation doit être suffisamment succinct pour tenir sur une seule ligne. Appuyez sur **Ctrl+Entrée** pour valider vos modifications.

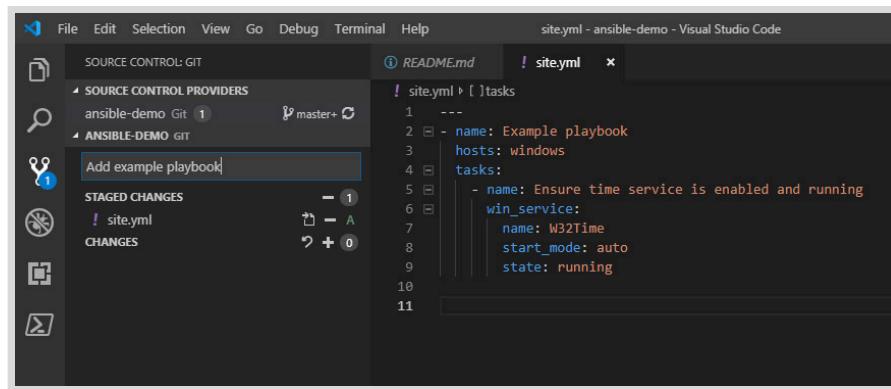


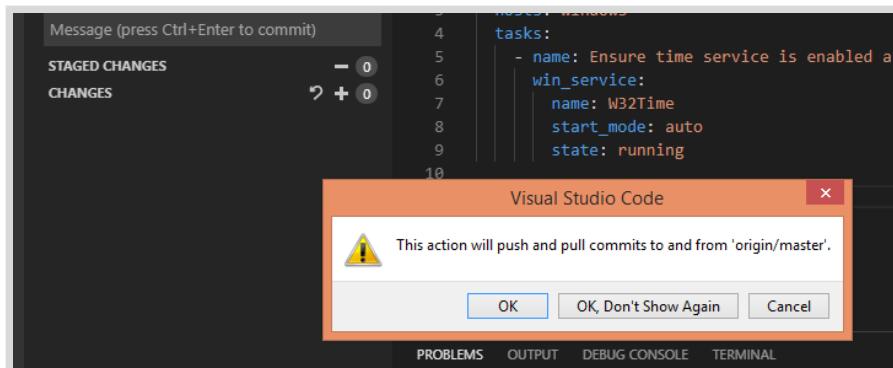
Figure 1.35: Validation des modifications

Maintenant qu'une modification locale a été validée, le référentiel local est en avance sur le référentiel distant en termes de modifications. Notez l'icône d'état située dans la partie inférieure gauche, qui indique qu'une modification peut être envoyée au référentiel distant.



**Figure 1.36: Le référentiel local est en avance par rapport au référentiel distant**

Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur Synchronize Changes pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.



**Figure 1.37: Un push vers le référentiel distant est requis**

Les référentiels sont désormais synchronisés à nouveau, et la partie suivante du travail peut commencer.



## Références

### **Pro Git de Scott Chacon et Ben Straub (livre gratuit)**

<https://git-scm.com/book/en/v2>

### **Visual Studio Code**

<https://code.visualstudio.com/>

### **Comment écrire un message de validation Git**

<https://chris.beams.io/posts/git-commit/>

## ► Exercice guidé

# Gestion de fichiers dans Git avec Visual Studio Code

Au cours de cet exercice, vous allez modifier des fichiers dans une copie locale d'un référentiel Git distant, valider vos modifications, puis pousser ces modifications vers le référentiel distant.

## Résultats

Vous devez être en mesure de configurer les paramètres Git et de modifier les fichiers stockés dans un référentiel Git existant.

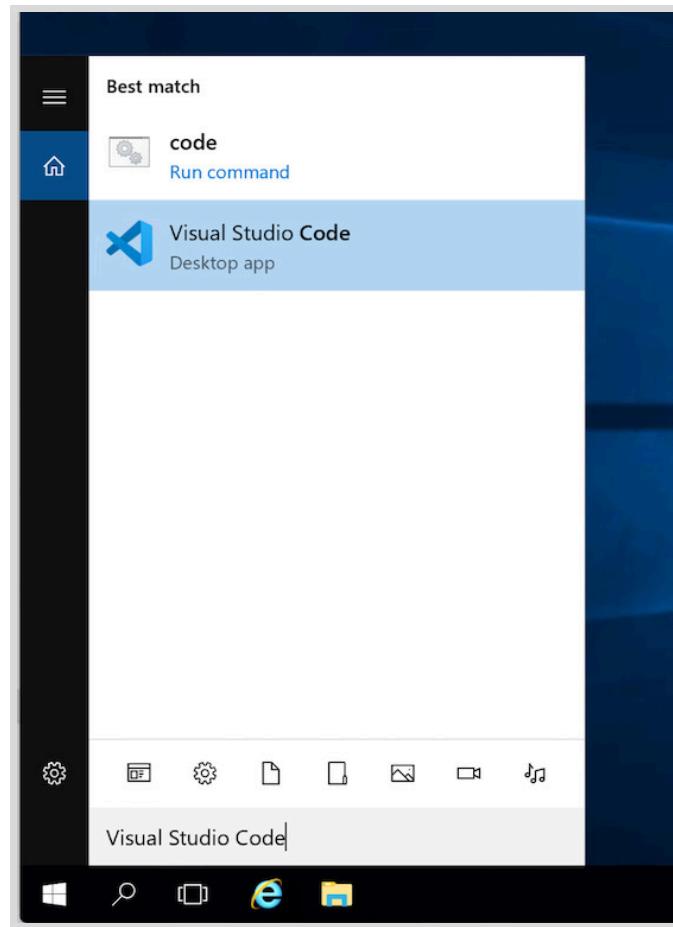
## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.

- ▶ 1. Lancez l'éditeur Visual Studio Code et clonez le référentiel **architecture** sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, tapez **code** comme terme à rechercher, puis ouvrez Visual Studio Code.

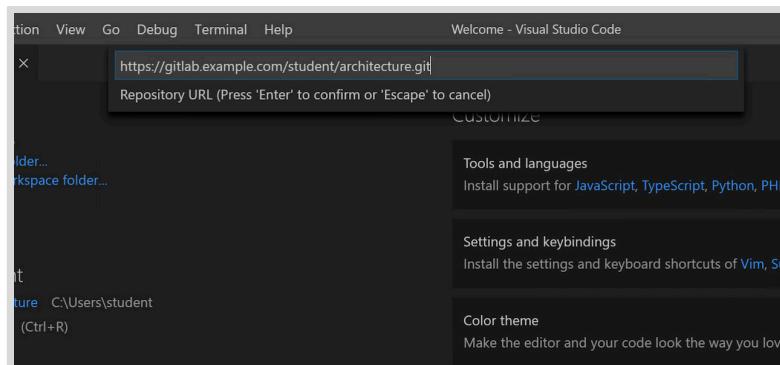


- Ouvrez la palette de commandes en accédant à **View → Command palette** ou en tapant **Ctrl+Maj+P**.

Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.

Utilisez l'URL de référentiel `https://gitlab.example.com/student/architecture.git`. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **architecture**.

Vous pouvez éventuellement afficher les fichiers en sélectionnant **Open** dans la fenêtre qui s'affiche après le clonage.



- ▶ 2. Ouvrez le répertoire **c:\Users\student\Documents\architecture** en accédant à **File → Open Folder**, puis sélectionnez le dossier **c:\Users\student\Documents\architecture**.
- ▶ 3. Modifiez le fichier **site.yml**, puis validez vos modifications.
- 3.1. Sélectionnez le fichier **site.yml**, puis modifiez le nom de la tâche et le nom du service Windows géré de **W32Time** en **Spooler**.

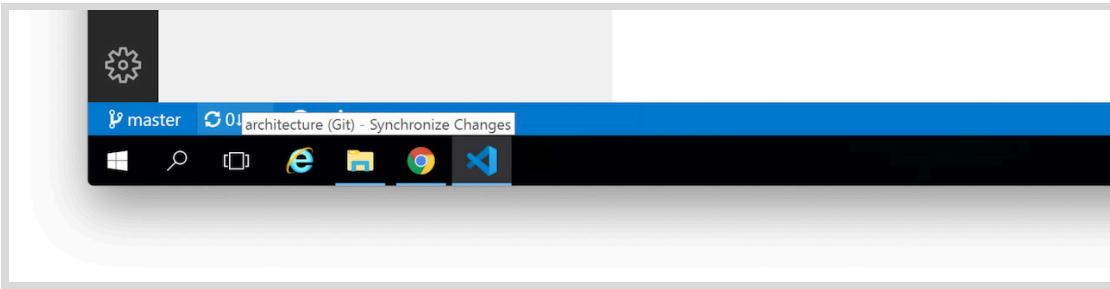
```
...output omitted...
- name: Spooler service is enabled and running
  win_service:
    name: Spooler
    start_mode: auto
    state: running
```

- 3.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été apportées.
- 3.3. Passez à l'affichage **Source Control**, puis sélectionnez **+**, en regard de **site.yml**, pour effectuer les modifications.
- 3.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- ▶ 4. Modifiez le fichier **README.md**, validez vos modifications, puis synchronisez votre référentiel local avec le référentiel distant.
- 4.1. Sélectionnez l'affichage **Explorer**, puis le fichier **README.md**.
- 4.2. Remplacez le nom du service **w32time** dans la description par **Spooler**.

```
This is a sample README for the Managing Files in Git exercise in D0417

### Description: Manages the Spooler service
...output omitted...
```

- 4.3. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche à nouveau en regard de **master**, ce qui indique que des modifications ont été apportées.
- 4.4. Passez à l'affichage **Source Control**, puis sélectionnez **+** en regard de **README.md** pour effectuer les modifications.
- 4.5. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 4.6. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes**, représenté par une flèche circulaire en bas de l'éditeur, pour envoyer vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.



À l'invite, cliquez sur **Yes** ou **Ask Me Later**.

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Présentation de Red Hat Ansible Automation

### Liste de contrôle des performances

Au cours de cet atelier, vous allez utiliser Visual Studio Code pour cloner un référentiel Git à partir d'un serveur distant, apporter des modifications à votre référentiel local et les valider, puis envoyer les modifications au référentiel distant.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Accéder à la console Web Webhook gitlab et inspecter les référentiels
- Cloner un référentiel distant à l'aide de Visual Studio Code
- Mettre à jour un playbook Ansible
- Valider et pousser vos modifications vers un référentiel Git distant

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.

1. À partir de **workstation**, accédez à votre instance GitLab disponible sur <https://gitlab.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.  
Accédez au référentiel **architecture** dans l'espace de noms **student** et assurez-vous que le playbook **review.yml** existe. Inspectez le contenu du playbook.
2. À partir de **workstation**, accédez à Visual Studio Code. Si vous n'avez pas encore cloné le référentiel **architecture** disponible sur <https://gitlab.example.com/student/architecture.git>, clonez-le sur **C:\Users\student\Documents\architecture**. Ajoutez le dossier **architecture** à l'espace de travail Visual Studio Code.
3. Utilisez Visual Studio Code pour modifier le fichier **review.yml** dans le projet **architecture**. Ces modifications vont modifier un playbook Ansible. Pour le moment, ne vous inquiétez pas de l'impact de ces modifications, car cet exercice est destiné à vous permettre de vous exercer à utiliser Visual Studio Code avec Git.  
Lisez le fichier et procédez aux modifications suivantes :
  - Sur la ligne **hosts:**, insérez un espace et le texte **win1.example.com** après les deux-points.

- Pour la première tâche, à savoir la ligne - **name: Install service**, remplacez « Install service » par **IIS Web Server is installed**.
- Pour la deuxième tâche (- **name: Start service**), remplacez la ligne **name: W3Svc** par **name: IIS Web Server is started**. Ne modifiez pas la mise en retrait de cette ligne.

Enregistrez vos modifications.

- 4.** Ajoutez vos modifications à l'index Git, puis validez vos modifications à l'aide du message **Use more descriptive task names and target the correct host**.

Validez et transmettez vos modifications à la branche **master**.

- 5.** Accédez à la console Web GitLab et passez en revue le playbook **review.yml** dans le référentiel **architecture** pour vous assurer que les modifications sont disponibles sur le serveur distant.

L'atelier est maintenant terminé.

## ► Solution

# Présentation de Red Hat Ansible Automation

### Liste de contrôle des performances

Au cours de cet atelier, vous allez utiliser Visual Studio Code pour cloner un référentiel Git à partir d'un serveur distant, apporter des modifications à votre référentiel local et les valider, puis envoyer les modifications au référentiel distant.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Accéder à la console Web Webhook gitlab et inspecter les référentiels
- Cloner un référentiel distant à l'aide de Visual Studio Code
- Mettre à jour un playbook Ansible
- Valider et pousser vos modifications vers un référentiel Git distant

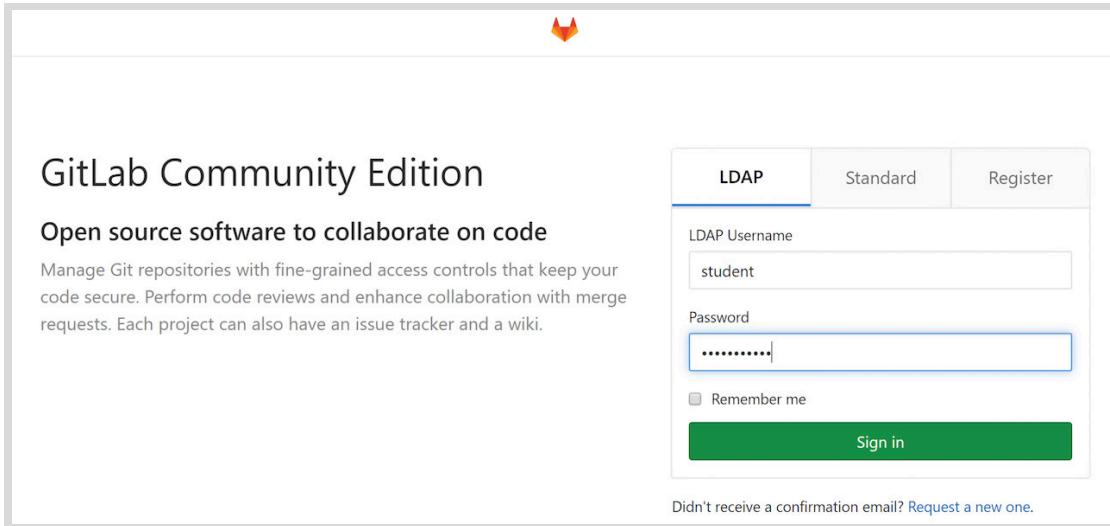
### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.

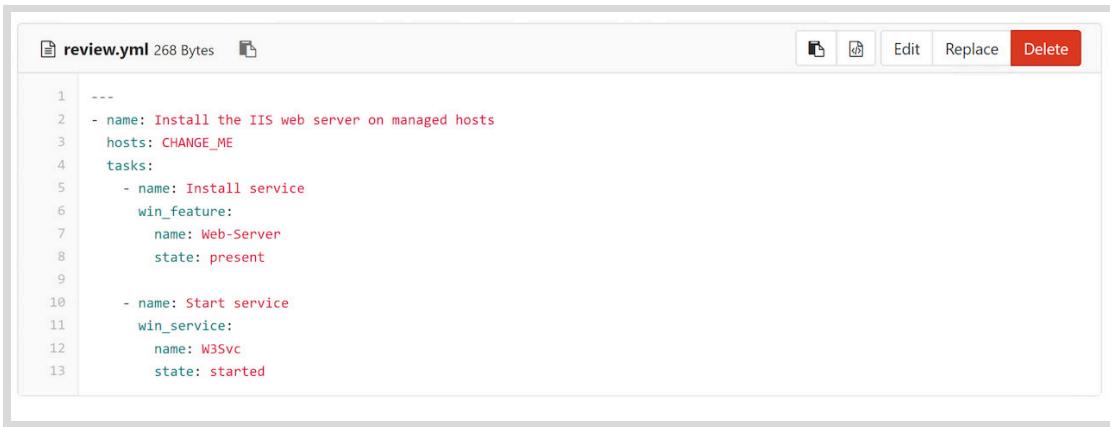
1. À partir de **workstation**, accédez à votre instance GitLab disponible sur <https://gitlab.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.  
Accédez au référentiel **architecture** dans l'espace de noms **student** et assurez-vous que le playbook **review.yml** existe. Inspectez le contenu du playbook.
  - 1.1. À partir de **workstation**, double-cliquez sur l'icône **GitLab** sur votre Bureau pour accéder à votre instance GitLab située sur <https://gitlab.example.com>.
  - 1.2. Connectez-vous à GitLab en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Assurez-vous que **LDAP** est sélectionné et cliquez sur **Sign In**.



- 1.3. La page vous redirige vers la liste des projets de l'utilisateur **student**. Dans l'onglet **Your projects**, sélectionnez le projet **architecture** pour afficher le contenu du référentiel.
- 1.4. La page répertorie tous les fichiers du référentiel. Cliquez sur **review.yml** pour inspecter le playbook.

Name	Last commit	Last Update
solutions	Establish initial architecture repository	a day ago
create_desktop_file.yml	Establish initial architecture repository	about 4 hours ago
dummy.yml	Establish initial architecture repository	a day ago
review.yml	Establish initial architecture repository	about a minute ago
sample_file.txt	Establish initial architecture repository	about 4 hours ago
test.yml	Establish initial architecture repository	about a minute ago

- 1.5. Le playbook appelle le module **win\_feature** pour installer le serveur Web IIS sur vos hôtes gérés. Une fois l'installation terminée, le playbook démarre le service à l'aide du module **win\_service**.



```

1  ---
2  - name: Install the IIS web server on managed hosts
3  hosts: CHANGE_ME
4  tasks:
5    - name: Install service
6      win_feature:
7        name: Web-Server
8        state: present
9
10   - name: Start service
11     win_service:
12       name: W3SVC
13       state: started

```

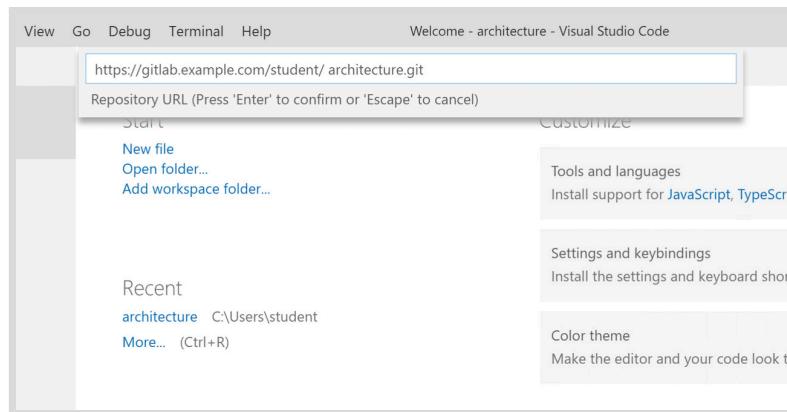
2. À partir de **workstation**, accédez à Visual Studio Code. Si vous n'avez pas encore cloné le référentiel **architecture** disponible sur <https://gitlab.example.com/student/architecture.git>, clonez-le sur **C:\Users\student\Documents\architecture**. Ajoutez le dossier **architecture** à l'espace de travail Visual Studio Code.

- 2.1. Cliquez sur **Recherche Windows**, puis tapez le **code** à rechercher, puis ouvrez Visual Studio Code.
- 2.2. Pour cloner le référentiel **architecture**, ouvrez la palette de commandes en accédant à **View → Command Palette** ou en tapant **Ctrl+Maj+P**.

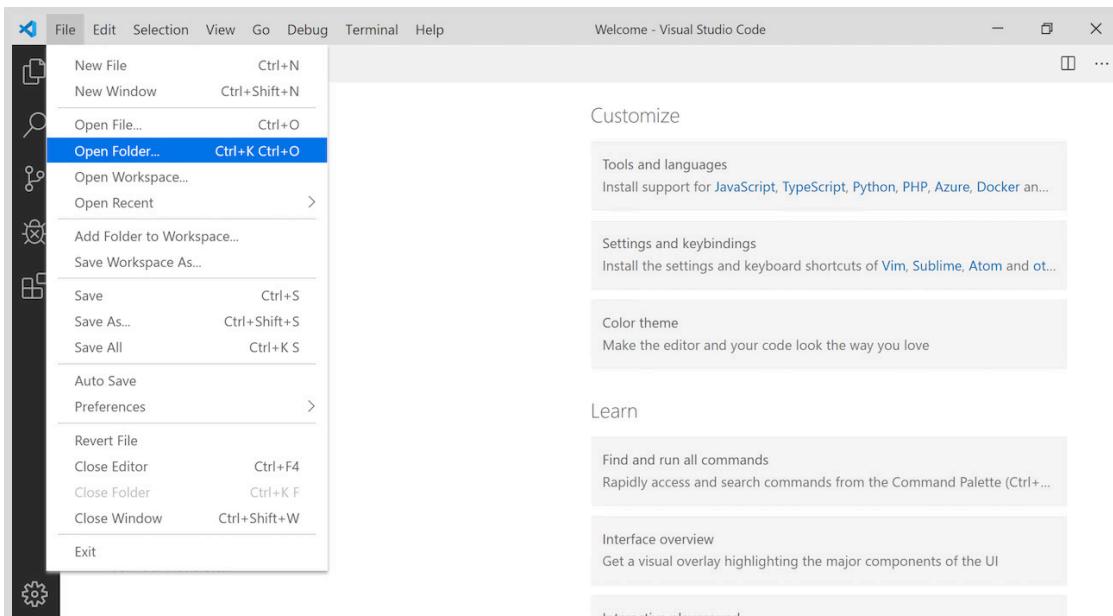
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.

Utilisez l'URL de référentiel <https://gitlab.example.com/student/architecture.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **architecture**.

Vous pouvez éventuellement afficher les fichiers en sélectionnant **Open** dans la fenêtre qui s'affiche après le clonage.



- 2.3. Pour ajouter le dossier **architecture** à votre espace de travail, accédez à **File → Open Folder**.



Dans l’Explorateur Windows, accédez à **Local Disk (C:)** → **Users** → **student** → **architecture**, puis cliquez sur **Select Folder**.

3. Utilisez Visual Studio Code pour modifier le fichier **review.yml** dans le projet **architecture**. Ces modifications vont modifier un playbook Ansible. Pour le moment, ne vous inquiétez pas de l’impact de ces modifications, car cet exercice est destiné à vous permettre de vous exercer à utiliser Visual Studio Code avec Git.

Lisez le fichier et procédez aux modifications suivantes :

- Sur la ligne **hosts:**, insérez un espace et le texte **win1.example.com** après les deux-points.
- Pour la première tâche, à savoir la ligne **- name: Install service**, remplacez « **Install service** » par **IIS Web Server is installed**.
- Pour la deuxième tâche (**- name: Start service**), remplacez la ligne **name: W3Svc** par **name: IIS Web Server is started**. Ne modifiez pas la mise en retrait de cette ligne.

Enregistrez vos modifications.

- 3.1. À partir de Visual Studio Code, ouvrez le playbook **review.yml** dans le dossier **architecture**.

Sur la ligne **hosts:**, insérez un espace et le texte **win1.example.com** après les deux-points.

```
---  
- name: Install the IIS web server on managed hosts  
  hosts: win1.example.com  
...output omitted...
```

- 3.2. Pour la première tâche, à savoir la ligne **- name: Install service**, remplacez « **Install service** » par **IIS Web Server is installed**.

```
...output omitted...
tasks:
  - name: IIS Web Server is installed
    win_feature:
      name: Web-Server
      state: present
...output omitted...
```

- 3.3. Pour la deuxième tâche (- **name: Start service**), remplacez la ligne **name: W3Svc** par **name: IIS Web Server is started**.

```
...output omitted...
  - name: IIS Web Server is started
    win_service:
      name: W3Svc
      state: started
```

- 3.4. Enregistrez votre fichier. Après vos mises à jour, le fichier doit se présenter comme suit :

```
---
- name: Install the IIS web server on managed hosts
  hosts: win1.example.com
  tasks:
    - name: IIS Web Server is installed
      win_feature:
        name: Web-Server
        state: present

    - name: IIS Web Server is started
      win_service:
        name: W3Svc
        state: started
```



#### Note

Le fichier de solution terminé est disponible dans **solutions/review.sol** dans le projet **architecture**.

4. Ajoutez vos modifications à l'index Git, puis validez vos modifications à l'aide du message

**Use more descriptive task names and target the correct host.**

Validez et transmettez vos modifications à la branche **master**.

- 4.1. À partir de la barre latérale de Visual Studio Code, sélectionnez **Source Control** pour accéder à la liste des modifications.

```

! review.yml x
! review.yml > [ ]tasks > abcname
1 --- 
2 - name: Install the IIS web server on managed hosts
3   hosts: win1.example.com
4   tasks:
5     - name: IIS Web Server is installed
6       win_feature:
7         name: Web-Server
8         state: present
9
10    - name: IIS Web Server is started
11      win_service:
12        name: W3Svc
13        state: started
14

```

- 4.2. Survolez **review.yml** pour accéder aux options disponibles pour le fichier. Cliquez sur **Stage Changes** pour ajouter vos modifications à l'index.

```

SOURCE CONTROL: GIT ✓ ⌂ ...
Use more descriptive tasks name and target the correct host
CHANGES
! review.yml M
Stage Changes
! review.yml M

```

```

! review.yml x
! review.yml > [ ]tasks > abcname
1 --- 
2 - name: Install the IIS web server on managed hosts
3   hosts: win1.example.com
4   tasks:
5     - name: IIS Web Server is installed
6       win_feature:
7         name: Web-Server
8         state: present
9
10    - name: IIS Web Server is started
11      win_service:
12        name: W3Svc
13        state: started
14

```

- 4.3. Dans le fichier texte **Message**, saisissez le message **Use more descriptive task names and target the correct host**

- 4.4. Cliquez sur **Commit** pour valider les modifications dans votre référentiel local.

```

! review.yml x
! review.yml > [ ]tasks > abcname
1   ---
2   - name: Install the IIS web server on managed hosts
3     hosts: win1.example.com
4     tasks:
5       - name: IIS Web Server is installed
6         win_feature:
7           name: Web-Server
8           state: present
9
10      - name: IIS Web Server is started
11        win_service:
12          name: W3Svc
13          state: started
14

```

- 4.5. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur Synchronize Changes pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
5. Accédez à la console Web GitLab et passez en revue le playbook **review.yml** dans le référentiel **architecture** pour vous assurer que les modifications sont disponibles sur le serveur distant.
- 5.1. À partir de **workstation**, actualisez votre navigateur ou accédez à **Student → Personal projects → student / architecture** pour accéder au référentiel. Sélectionnez le playbook **review.yml**. Le résultat doit afficher vos modifications.

```

! review.yml 299 Bytes
1   ---
2   - name: Install the IIS web server on managed hosts
3     hosts: win1.example.com
4     tasks:
5       - name: IIS Web Server is installed
6         win_feature:
7           name: Web-Server
8           state: present
9
10      - name: IIS Web Server is started
11        win_service:
12          name: W3Svc
13          state: started

```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Infrastructure as Code (IaC) vous permet d'utiliser un langage d'automatisation lisible par ordinateur pour définir et décrire l'état dans lequel vous voulez que se trouve votre infrastructure informatique.
- Ansible est une plateforme d'automatisation Open Source et un langage d'automatisation simple qui peut décrire l'infrastructure d'application informatique.
- Ansible est installé et exécuté à partir d'un nœud de contrôle, qui exécute des playbooks sur des hôtes gérés...
- Ansible Tower fournit des outils et des fonctions natifs pour la gestion des environnements Microsoft Windows.
- Git est un système de contrôle de version distribué (*DVCS, Distributed Version Control System*) qui vous permet de gérer les modifications apportées aux fichiers d'un projet de manière collaborative.

## chapitre 2

# Exécution de commandes simples d'automatisation

### Objectif

Préparer des hôtes Microsoft Windows en vue de l'automatisation et Red Hat Ansible Tower en tant que système de contrôle d'automatisation central, et exécuter des tâches d'automatisation uniques sur ces hôtes à partir d'Ansible Tower.

### Résultats

- Configurer les systèmes Microsoft Windows pour fournir les conditions préalables et l'accès nécessaires à la gestion par Red Hat Ansible Automation.
- Configurer Red Hat Ansible Tower avec un inventaire des hôtes Windows à gérer, ainsi que les informations d'identification nécessaires pour vous authentifier et vous connecter à ces hôtes.
- Décrire ce qu'est un module Ansible et exécuter une seule opération sur les hôtes gérés sous la forme d'une commande ad hoc à l'aide de Red Hat Ansible Tower.

### Sections

- Préparation des hôtes Microsoft Windows pour l'automatisation (et quiz)
- Préparation de Red Hat Ansible Tower pour gérer des hôtes (et exercice guidé)
- Exécution de commandes ad hoc (et exercice guidé)

### Atelier

Exécution de commandes simples d'automatisation

# Préparation des hôtes Microsoft Windows pour l'automatisation

## Résultats

Au terme de cette section, vous devez pouvoir configurer les systèmes Microsoft Windows pour fournir les conditions préalables et l'accès nécessaires à la gestion par Red Hat Ansible Automation.

## Hôtes gérés

L'un des avantages des hôtes gérés, c'est qu'ils n'ont pas besoin qu'un agent Ansible spécial soit installé. Le nœud de contrôle Ansible se connecte aux hôtes gérés à l'aide d'un protocole réseau standard afin de garantir que les systèmes sont dans l'état spécifié. Sur les systèmes Windows, Ansible se connecte au service *Windows Remote Management (WinRM)* et à son écouteur.

## Configuration minimale requise

Avant de commencer, vérifiez que la configuration minimale requise du système d'exploitation hôte est satisfaita.

Vous pouvez utiliser Ansible pour gérer toutes les versions de Windows disposant d'un support Microsoft standard et étendu. Les versions antérieures de Windows, telles que Windows Server 2008 ou Windows 7, nécessitent des mises à jour PowerShell et .NET Framework. Appliquez toujours les derniers correctifs de sécurité.

Les hôtes gérés ont besoin des exigences de base *minimales* suivantes :

- Windows Server (2008, 2008 R2, 2012, 2012 R2, 2016 ou 2019) ou Windows 7, 8.1 ou 10
- PowerShell 3.0 ou version ultérieure
- .NET Framework 4.0 ou version ultérieure

Les versions de Windows depuis Windows Server 2012 et Windows 8 sont livrées avec le minimum de dépendances .NET et PowerShell requises. Les versions de Windows antérieures à cette opération nécessitent le téléchargement et l'installation du dernier Service Pack, de .NET Framework et de PowerShell.

Ce sont les exigences de base pour qu'Ansible puisse fonctionner en général. Cependant, certains modules Ansible peuvent avoir des exigences supplémentaires qui sont répertoriées dans la documentation de ces modules sur [https://docs.ansible.com/ansible/latest/user\\_guide/modules.html](https://docs.ansible.com/ansible/latest/user_guide/modules.html).



### Important

En raison d'un problème de mémoire WinRM, PowerShell 3.0 requiert un hotfix. Reportez-vous à l'article de support Microsoft disponible à l'adresse <https://support.microsoft.com/en-us/help/2842230/out-of-memory-error-on-a-computer-that-has-a-customized-maxmemorypersh> pour plus d'informations.

Installez les dernières mises à jour stables pour garantir la sécurité et la fiabilité de vos systèmes.

## Configuration de Windows pour l'accès à Ansible

Ansible communique avec les hôtes Windows à l'aide de Windows Remote Management. WinRM est une mise en œuvre Microsoft du protocole standard de gestion WS basé sur SOAP et fournit un identifiant de connexion non interactif pour contrôler un système distant.

WinRM est activé par défaut depuis Windows Server 2012. Cependant, vous devez toujours configurer un écouteur WinRM, les règles de pare-feu correspondantes et le mécanisme d'authentification.

Ansible utilise le protocole PSRP (*PowerShell Remoting Protocol*) par dessus WinRM pour exécuter des commandes PowerShell sur une cible. PSRP fournit une connexion directe plus rapide à PowerShell.

Étant donné que la connexion Ansible dépend généralement de WinRM et de PowerShell, Ansible ne peut pas interagir directement avec les écouteurs WinRM ni mettre à niveau PowerShell.



### Note

Ansible 2.8 fournit une fonctionnalité expérimentale qui vous permet d'utiliser SSH au lieu de WinRM pour vous connecter à des hôtes Windows. Il n'est pas encore pris en charge et peut changer dans les versions ultérieures d'Ansible. Ce cours utilise le protocole WinRM standard pour se connecter à des hôtes gérés par Windows.

## Configuration pour des environnements de développement

Le projet Ansible fournit un exemple de script PowerShell permettant de configurer les hôtes gérés Windows pour l'accès à distance. Ce script est conçu pour être utilisé dans des environnements d'atelier et de développement dans lesquels les niveaux de sécurité élevés ne sont pas un problème. Ansible ne recommande pas d'utiliser l'exemple de script dans des environnements de production en raison de l'affaiblissement potentiel du mécanisme d'authentification.

Trouvez le script dans le référentiel officiel <https://github.com/ansible/ansible/blob/devel/examples/scripts/ConfigureRemotingForAnsible.ps1>.

Le script **ConfigureRemotingForAnsible.ps1** effectue les actions suivantes :

1. Confirme l'installation de PowerShell version 3 ou version ultérieure.
2. Exécute le service WinRM, s'il n'est pas déjà en cours d'exécution, et le configure pour qu'il s'exécute automatiquement au démarrage.
3. Active la communication à distance PowerShell et un écouteur SSL.
4. Définit la clé de Registre **LocalAccountTokenFilterPolicy** sur 1.
5. En option, configure l'authentification de base ou CredSSP. Cela est décrit en détail ci-dessous.
6. Configure le pare-feu Windows pour les connexions WinRM à la fois sur HTTP et HTTPS.

Exécutez **ConfigureRemotingForAnsible.ps1** en tant qu'administrateur avec les options appropriées pour votre environnement. Ce cours utilise les options **-DisableBasicAuth** et **-EnableCredSSP** comme préférence de sécurité.

**ConfigureRemotingForAnsible.ps1** crée également un certificat auto-signé pour les communications HTTPS. Il existe plusieurs options pour personnaliser le certificat SSL ou forcer la création d'un nouveau certificat.

La liste suivante décrit la liste des arguments que vous pouvez utiliser avec le script PowerShell :

#### Liste d'arguments **ConfigureRemotingForAnsible.ps1**

-EnableCredSSP

Active le protocole d'authentification CredSSP.

-DisableBasicAuth

Désactive l'authentification de base. Cette option est recommandée, car l'authentification de base est la méthode d'authentification WinRM la moins sécurisée.

-CertValidityDays

Spécifie l'expiration du certificat SSL auto-signé créé pour HTTPS. La validité par défaut est de 1 095 jours (3 ans).

-ForceNewSSLCert

Force la création d'un nouveau certificat SSL.

-SubjectName

Spécifie le nom CN (Common Name) du certificat SSL. Par défaut, il s'agit du nom d'hôte.

-SkipNetworkProfileCheck

Permet d'activer PS Remoting sans vérifier le profil réseau.

-Verbose

Active la sortie détaillée. Cette option est utile pour le débogage.

## Configuration pour les environnements de production

L'exécution de **ConfigureRemotingForAnsible.ps1** démarre rapidement les systèmes pour les environnements d'atelier et de développement, mais cette commande n'est pas destinée à des systèmes de production robustes. Utilisez le script comme point de départ, puis adaptez-le pour qu'il respecte vos exigences en matière de sécurité.

Pour éviter les incohérences entre les systèmes et réduire les étapes manuelles, intégrez le script dans le cadre d'un processus de provisionnement d'hôte automatisé. Les méthodes spécifiques permettant de réaliser cette opération varient d'un environnement à l'autre et ne sont pas abordées dans le cadre de ce cours.

## Authentification

Ansible peut utiliser divers protocoles pour authentifier sa connexion avec les hôtes Windows. À savoir : Basic, Certificate, CredSSP, NTLM et Kerberos. Vous devez configurer l'hôte géré pour qu'il accepte le protocole le mieux adapté à votre environnement.

Option	Comptes locaux	Comptes Active Directory	Délégation des informations d'identification	Chiffrement HTTP
<b>Basique</b>	Oui	Non	Non	Non
<b>Certificat</b>	Oui	Non	Non	Non
<b>Kerberos</b>	Non	Oui	Oui	Oui

Option	Comptes locaux	Comptes Active Directory	Délégation des informations d'identification	Chiffrement HTTP
<b>AUTHENTIFICATION</b>	Oui	Oui	Non	Oui
<b>CredSSP</b>	Oui	Oui	Oui	Oui

**Mise en garde**

Les stratégies d'authentification de base et par certificat ne sont pas chiffrées. Il est vivement recommandé de n'autoriser ces protocoles que par le biais de connexions HTTPS sécurisées.

Les protocoles d'authentification de base et par certificat sont simples mais moins sécurisés. De plus, ces options peuvent uniquement utiliser des comptes locaux. La plupart des configurations de production nécessitent un protocole conçu pour les comptes de domaine.

CredSSP, NTLM et Kerberos prennent en charge les comptes de domaine. CredSSP et Kerberos sont préférés en raison de leur sécurité, de leur vitesse et de la prise en charge de la délégation des informations d'identification.

## CredSSP

CredSSP (Credential Security Support Provider) prend en charge l'authentification à la fois pour les comptes locaux et les comptes de domaine.

Contrairement à l'authentification de base, CredSSP chiffre les informations d'identification avant de les transmettre à l'hôte cible. Les messages chiffrés peuvent être envoyés via HTTP.

**Note**

Ce cours utilise CredSSP pour l'environnement de formation.

La délégation d'informations d'identification est une fonctionnalité principale de CredSSP. Cela permet à votre serveur cible de s'authentifier auprès d'un deuxième serveur à l'aide de vos informations d'identification.

**Important**

Utilisez uniquement la délégation d'informations d'identification CredSSP avec des hôtes approuvés. Les informations d'identification de l'utilisateur sont transmises directement aux serveurs.

Exécutez la commande **ConfigureRemotingForAnsible.ps1** avec l'option **-EnableCredSSP** pour activer l'authentification CredSSP. Cela exécute la commande PowerShell **Enable-WSManCredSSP -role server -Force**.

## NTLM (New Technology Local-Area Network Manager)

NT (nouvelle technologie) LAN (réseau local) Manager (NTLM) est un protocole d'authentification par hachage à sens unique.

Le protocole NTLM ne requiert aucune installation supplémentaire et le service WinRM l'active par défaut.

NTLM est largement utilisé pour la compatibilité héritée, mais il n'est pas préféré pour les applications de production en raison des faiblesses de sécurité et des vulnérabilités connues. Kerberos a remplacé NTLM comme le protocole d'authentification recommandé.

## Kerberos

Kerberos est le protocole d'authentification réseau Microsoft par défaut. Il fournit une option d'authentification plus sophistiquée, qui est recommandée pour une utilisation avec les comptes de domaine.

Kerberos est conçu pour authentifier de manière sécurisée les entités d'un réseau non approuvé, tel qu'Internet. Il utilise la cryptographie à clé symétrique et une tierce partie de confiance pour générer et valider un système de tickets échangés.

Les clients s'authentifient auprès d'un *Centre de distribution de clés Kerberos (KDC, Key Distribution Center)* pour récupérer un ticket de session. Le contrôleur de domaine Active Directory peut indiquer le rôle du KDC. Les secrets partagés sont utilisés pour chiffrer et déchiffrer les messages entre les entités.

Contrairement à CredSSP et NTLM, Kerberos ne prend pas en charge les comptes locaux.

La configuration d'Active Directory et de l'authentification Kerberos est décrite dans Chapitre 8, *Interaction avec les utilisateurs et les domaines*.



### Références

#### **Ansible Windows Guides &mdash; Ansible Documentation**

[https://docs.ansible.com/ansible/latest/user\\_guide/windows.html](https://docs.ansible.com/ansible/latest/user_guide/windows.html)

#### **Windows Remote Management**

<https://docs.microsoft.com/en-us/windows/win32/winrm/portal>

## ► Quiz

# Préparation des hôtes Microsoft Windows pour l'automatisation

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

► 1. Quelles sont les deux technologies requises sur les hôtes gérés Windows ? (Choisissez-en deux.)

- a. OpenSSH 8.0
- b. .NET Framework 4.0
- c. Visual Studio 2019
- d. PowerShell 3.0

► 2. Quelles sont les deux limitations de la connexion Ansible WinRM ? (Choisissez-en deux.)

- a. Ansible ne peut pas exécuter de commandes qui nécessitent un accès en tant qu'administrateur.
- b. Ansible ne peut pas interagir avec l'écouteur WinRM.
- c. Ansible ne peut pas mettre à niveau PowerShell.
- d. Ansible ne peut pas redémarrer l'hôte.

► 3. Quelle est l'utilité de l'exemple de script PowerShell

`ConfigureRemotingForAnsible.ps1` fourni par Ansible ?

- a. Configurer rapidement des hôtes gérés pour l'accès à distance.
- b. Présenter les meilleures pratiques pour les déploiements de production.
- c. Configurer Ansible Tower pour une opération à distance.
- d. Installer à distance les mises à jour logicielles les plus récentes sur des hôtes gérés.

► 4. Quelles sont les fonctions qui font que le protocole CredSSP est plus sécurisé que l'authentification de base ?

- a. CredSSP ne transmet pas les informations d'identification pendant un deuxième tronçon de serveur.
- b. CredSSP ne prend pas en charge les comptes locaux.
- c. Un tiers de confiance fournit un échange chiffré des tickets.
- d. CredSSP chiffre les informations d'identification avant de les envoyer à l'hôte cible.

► **5. Quelles sont les trois fonctionnalités qui font de Kerberos l'option préférée pour les environnements de production ? (Choisissez-en trois.)**

- a. Kerberos est le protocole d'authentification réseau Microsoft par défaut.
- b. Kerberos prend en charge les comptes de domaine Active Directory.
- c. Kerberos utilise la cryptographie à clé symétrique et un système de tickets au lieu de transmettre des informations d'identification de compte directement à l'hôte cible.
- d. Kerberos ne peut pas fonctionner de manière sécurisée sur des réseaux non approuvés.

## ► Solution

# Préparation des hôtes Microsoft Windows pour l'automatisation

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

► 1. **Quelles sont les deux technologies requises sur les hôtes gérés Windows ? (Choisissez-en deux.)**

- a. OpenSSH 8.0
- b. .NET Framework 4.0
- c. Visual Studio 2019
- d. PowerShell 3.0

► 2. **Quelles sont les deux limitations de la connexion Ansible WinRM ? (Choisissez-en deux.)**

- a. Ansible ne peut pas exécuter de commandes qui nécessitent un accès en tant qu'administrateur.
- b. Ansible ne peut pas interagir avec l'écouteur WinRM.
- c. Ansible ne peut pas mettre à niveau PowerShell.
- d. Ansible ne peut pas redémarrer l'hôte.

► 3. **Quelle est l'utilité de l'exemple de script PowerShell**

**ConfigureRemotingForAnsible.ps1 fourni par Ansible ?**

- a. Configurer rapidement des hôtes gérés pour l'accès à distance.
- b. Présenter les meilleures pratiques pour les déploiements de production.
- c. Configurer Ansible Tower pour une opération à distance.
- d. Installer à distance les mises à jour logicielles les plus récentes sur des hôtes gérés.

► 4. **Quelles sont les fonctions qui font que le protocole CredSSP est plus sécurisé que l'authentification de base ?**

- a. CredSSP ne transmet pas les informations d'identification pendant un deuxième tronçon de serveur.
- b. CredSSP ne prend pas en charge les comptes locaux.
- c. Un tiers de confiance fournit un échange chiffré des tickets.
- d. CredSSP chiffre les informations d'identification avant de les envoyer à l'hôte cible.

► **5. Quelles sont les trois fonctionnalités qui font de Kerberos l'option préférée pour les environnements de production ? (Choisissez-en trois.)**

- a. Kerberos est le protocole d'authentification réseau Microsoft par défaut.
- b. Kerberos prend en charge les comptes de domaine Active Directory.
- c. Kerberos utilise la cryptographie à clé symétrique et un système de tickets au lieu de transmettre des informations d'identification de compte directement à l'hôte cible.
- d. Kerberos ne peut pas fonctionner de manière sécurisée sur des réseaux non approuvés.

# Préparation de Red Hat Ansible Tower pour gérer des hôtes

## Résultats

Au terme de cette section, vous devez pouvoir configurer Red Hat Ansible Tower avec un inventaire des hôtes Windows à gérer, ainsi que les informations d'identification nécessaires pour vous authentifier et vous connecter à ces hôtes.

## Gestion de systèmes Windows avec Ansible Tower

Red Hat Ansible Tower peut être un panneau de contrôle central pour la gestion de vos systèmes Microsoft Windows. Il permet de protéger la sécurité des informations d'identification d'authentification, d'ajouter et d'organiser des machines hôtes en groupes, et de contrôler l'accès des utilisateurs à ces projets et à ces informations d'identification.

## Spécification d'hôtes gérés dans Ansible Tower

Ansible Tower obtient la liste des hôtes à gérer à partir des *inventaires*. Un inventaire décrit une série d'hôtes individuels ou de groupes d'hôtes sur lesquels des tâches d'automatisation peuvent être exécutées.

Vous pouvez définir d'autres informations par le biais de *variables d'inventaire* sur des hôtes individuels, des groupes d'hôtes ou tous les membres d'un inventaire particulier. Les variables d'inventaire peuvent modifier la manière dont Ansible Tower interagit avec les hôtes ou se connecte à ces derniers, et les tâches d'automatisation peuvent les utiliser pour déterminer ce qu'il convient de faire.

Vous pouvez configurer plusieurs inventaires dans Ansible Tower et utiliser différents inventaires à des fins différentes. Spécifiez manuellement les hôtes et les groupes d'hôtes dans un inventaire, ou générez des paramètres d'inventaire de manière dynamique à partir d'une source d'informations distante, telle que Microsoft Azure Resource Manager ou Amazon Web Services EC2.

## Création d'un inventaire dans l'interface utilisateur web d'Ansible Tower

Les inventaires décrivent un ensemble d'hôtes sur lesquels des tâches sont exécutées. Les inventaires sont divisés en *groupes* et ces groupes contiennent les hôtes. Vous pouvez utiliser une adresse IP, un nom d'hôte ou des expressions régulières lorsque les noms d'hôtes utilisent un modèle de dénomination. Les groupes sont similaires aux catégories qui décrivent un ensemble d'hôtes similaires, par exemple, un groupe **développement** contiendra tous vos serveurs de développement.

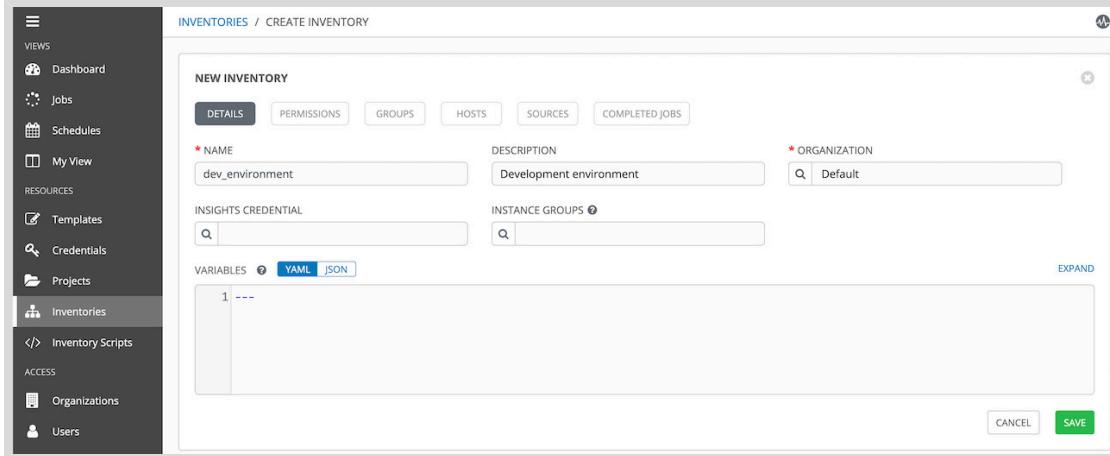
Vous pouvez créer des inventaires manuellement ou les provisionner dynamiquement. Les inventaires requièrent un nom et doivent être attribués à une organisation.

L'exemple suivant décrit la procédure de création manuelle d'un inventaire dans l'interface utilisateur web :

1. Connectez-vous à l'interface utilisateur web d'Ansible Tower.
2. Cliquez sur **Inventories** dans le volet de navigation.

**chapitre 2 |** Exécution de commandes simples d'automatisation

3. Dans la fenêtre **INVENTORIES**, cliquez sur le bouton **+**. Choisissez **Inventory** dans la liste. Dans la fenêtre **NEW INVENTORY**, entrez le **NAME** de l'inventaire et son **ORGANIZATION**. Cliquez sur l'icône de loupe de la zone **ORGANIZATION** pour afficher la liste des organisations disponibles. Cliquez sur **SAVE**.



**Figure 2.1: Création de nouveaux inventaires dans Ansible Tower**

Une fois que l'inventaire existe, vous pouvez y ajouter des hôtes, affecter ces hôtes à des groupes et définir des valeurs pour les variables d'inventaire.

Il existe deux façons principales d'ajouter des hôtes à des inventaires. Les *inventaires statiques* sont gérés manuellement. Les *inventaires dynamiques* sont mis à jour de manière dynamique, avec les hôtes et les groupes dans l'inventaire, en extrayant une liste à partir d'une source centrale d'informations, telle que Microsoft Azure Resource Manager, Amazon EC2 ou VMware vCenter. Il est souvent plus facile et plus efficace de configurer un inventaire dynamique qui obtient des informations à jour à partir d'une source centrale précise.

## Gestion des hôtes dans un inventaire

Les hôtes Ansible définissent les systèmes à gérer. Pour ajouter un hôte à un inventaire statique :

1. Dans l'interface utilisateur web d'Ansible Tower, cliquez sur **Inventories** dans le volet de navigation, puis cliquez sur le nom de l'inventaire auquel vous souhaitez ajouter un hôte.
2. Dans la fenêtre de détail d'inventaire, cliquez sur le bouton **HOSTS**. Ensuite, cliquez sur le bouton **+**, qui affiche l'info-bulle « *Create a New Host* ».
3. Dans le champ **HOST NAME**, saisissez le nom d'hôte ou l'adresse IP de l'hôte utilisé par les outils d'automatisation pour gérer cet hôte.
4. Vous pouvez définir des valeurs pour les variables d'inventaire Ansible qui s'appliquent uniquement à cet hôte dans la zone **VARIABLES**, au format YAML ou JSON. Les variables et leur utilisation seront traitées plus en détail ultérieurement dans le cours. Pour les hôtes Windows, vous devrez peut-être définir des variables qui contrôlent la manière dont Ansible Tower se connectera à cet hôte pour des raisons de gestion. Ces variables sont abordées plus tard dans cette section.
5. Cliquez sur **SAVE**.

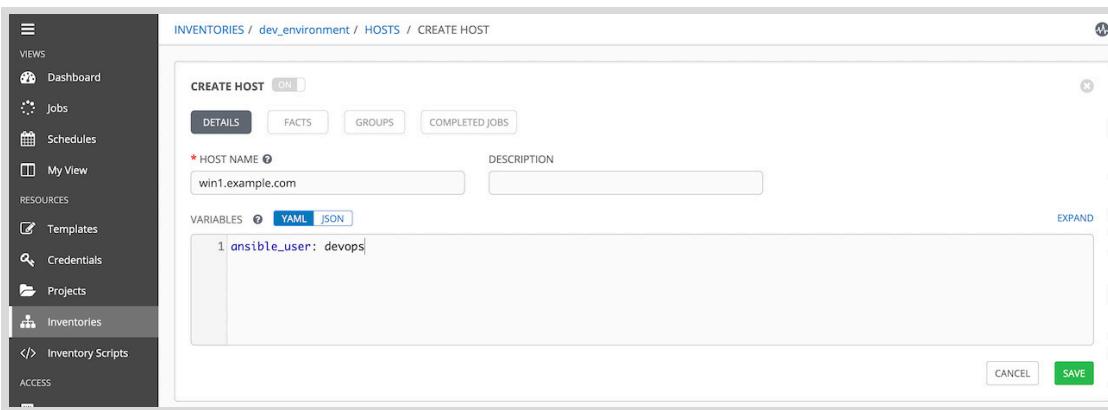


Figure 2.2: Gestion des hôtes dans des inventaires Ansible

**Note**

Pour configurer un inventaire dynamique, sélectionnez le bouton **SOURCES** dans l'interface d'inventaire. Cliquez sur le bouton **+** pour ajouter une source, puis attribuez un nom à la source, sélectionnez le type de source que vous utilisez et renseignez d'autres informations nécessaires pour ce type de source. Des informations supplémentaires sont disponibles dans la documentation d'Ansible Tower sur <https://docs.ansible.com>.

## Organisation d'hôtes en groupes

Utilisez les **groupes** pour organiser les hôtes à des fins de gestion collective. Par exemple, vous pouvez créer un groupe **développement** pour les serveurs de développement et exécuter une tâche d'automatisation sur tous les hôtes de ce groupe. Vous pouvez également créer un groupe **production** pour les serveurs de production.

Les hôtes peuvent être placés dans plusieurs groupes simultanément. Par exemple, vous pouvez créer un ensemble de groupes pour les systèmes de développement, de test et de production. Vous pouvez également créer des groupes en fonction de l'emplacement physique ou du datacenter dans lequel se trouve un hôte. Vous pouvez configurer des groupes en fonction de l'objectif ou de l'application sur ces systèmes. Ensuite, vous pouvez créer un ensemble de groupes final basé sur le fait qu'un système exécute Windows Server 2016, Windows Server 2019 ou Red Hat Enterprise Linux 8. Vous pouvez même imbriquer des groupes, de sorte qu'un groupe **us** puisse avoir pour membres les groupes **us-west-1** et **us-east-1**. En utilisant des groupes comme celui-ci, vous pouvez plus facilement appliquer des tâches d'automatisation à de nombreux hôtes du même type sans utiliser des tests conditionnels complexes dans les playbooks Ansible.

Pour créer un groupe et lui affecter des hôtes :

1. Dans l'interface utilisateur web d'Ansible Tower, cliquez sur **Inventories** dans le volet de navigation, puis cliquez sur le nom de l'inventaire auquel vous souhaitez ajouter un hôte.
2. Dans l'écran de détails de l'inventaire, cliquez sur le bouton **GROUPS**. Ensuite, cliquez sur le bouton **+**, qui affiche l'info-bulle « Create a new group ».
3. Dans le champ **NAME**, saisissez le nom à utiliser pour le groupe.
4. Vous pouvez définir des valeurs pour les variables d'inventaire Ansible qui s'appliquent à tous les membres de ce groupe dans la zone **VARIABLES**, au format YAML ou JSON. Si

## chapitre 2 | Exécution de commandes simples d'automatisation

une variable de groupe et une variable d'hôte définissent une valeur pour le même nom de variable, la valeur de la variable d'hôte est prioritaire.

5. Cliquez sur **SAVE**.
6. Les groupes imbriqués sont créés, et les hôtes ajoutés au groupe, de la même manière que pour les groupes d'inventaire de niveau supérieur. Dans l'écran de détail du groupe, cliquez sur le bouton **GROUPS** pour créer un groupe imbriqué pour cet inventaire, en ajoutant les hôtes compris dans chaque groupe imbriqué au nouveau groupe. Vous pouvez également cliquer sur le bouton **HOSTS** pour ajouter des hôtes au groupe.

Vous pouvez définir n'importe quelle variable dans le champ **VARIABLES**. Les playbooks Ansible lisent ces variables et appliquent leur valeur aux hôtes de ce groupe. Cela s'avère utile si certaines valeurs dépendent de l'environnement, par exemple, un chemin d'accès différent entre un environnement de développement et un environnement de production.

La capture d'écran ci-dessous montre comment utiliser des variables de groupe pour définir le répertoire root d'un serveur Web dans un environnement de pré-production.

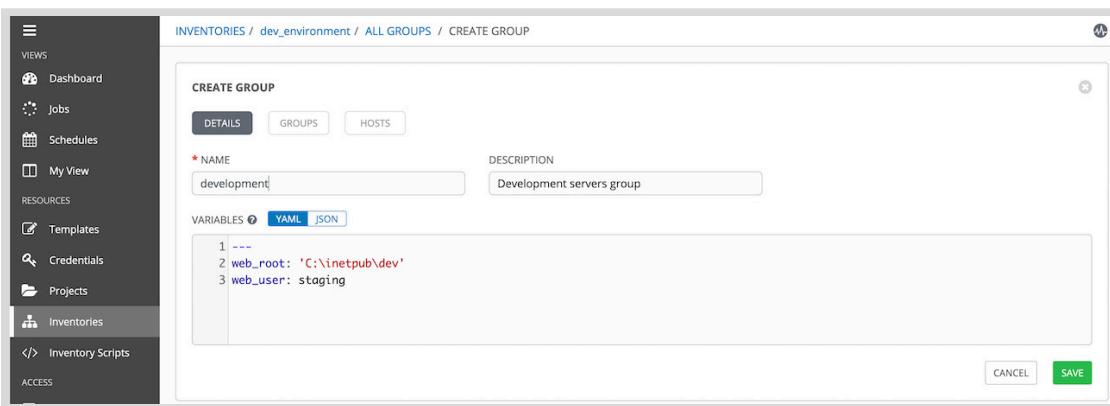


Figure 2.3: Gestion des variables de groupe

## Variables d'inventaire et hôtes Microsoft Windows

Lors de la gestion d'hôtes Microsoft Windows, quelques variables d'inventaire doivent être définies pour les hôtes Windows, de sorte qu'Ansible Tower sache comment s'y connecter. Il existe quatre paramètres clés :

### Variables de connexion pour les systèmes Microsoft Windows

Variable	Objet
<b>ansible_connection</b>	Protocole à utiliser pour se connecter à l'hôte
<b>ansible_port</b>	Port réseau à utiliser avec ce protocole
<b>ansible_winrm_transport</b>	Méthode d'authentification à utiliser avec les connexions WinRM
<b>ansible_winrm_server_cert_validation</b>	Indique s'il faut valider le certificat TLS pour les connexions WinRM

Bien que vous puissiez définir ces variables pour chaque hôte, il est utile et plus pratique de les définir pour l'inventaire, ou pour un groupe, en particulier si tous les hôtes de l'inventaire sont des hôtes qui doivent avoir les mêmes paramètres. Si vous avez un mélange d'hôtes Windows et d'autres systèmes d'exploitation et périphériques dans l'inventaire, vous pouvez créer un groupe pour les hôtes Windows et appliquer les paramètres à tous les hôtes de ce groupe.

Ces paramètres doivent être définis dans le cadre des **variables** appropriées. L'exemple suivant montre comment les définir pour l'ensemble de l'inventaire.

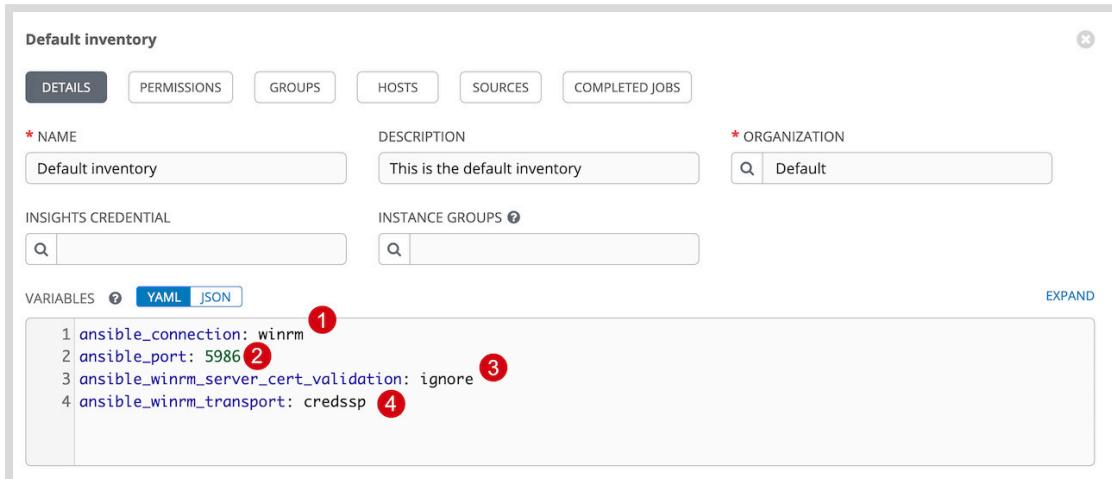


Figure 2.4: Gestion des variables de connexion dans un inventaire

1. Ordonne à Ansible Tower d'utiliser une autre méthode de connexion pour un hôte Windows, en l'occurrence, WinRM. Par défaut, Ansible utilise le protocole SSH pour se connecter à n'importe quel hôte.
2. Ordonne à Ansible d'utiliser le port 5986 pour le trafic WinRM avec chiffrement TLS.
3. Ordonne à Ansible d'ignorer la signature de l'autorité de certification (CA) sur le certificat WinRM présenté par le serveur et de simplement l'accepter. Ce paramètre est nécessaire si le certificat WinRM est auto-signé. Définissez l'option sur **validate** si vous utilisez des certificats WinRM signés par une autorité de certification valide.



### Note

Si votre certificat WinRM est signé par une autorité de certification publique, il doit valider. Si vous utilisez un certificat d'autorité de certification personnalisé pour signer le certificat WinRM, ajoutez le certificat d'autorité de certification personnalisé à votre installation Ansible Tower. Les détails sur la manière de configurer cette validation de certificat ne sont pas abordés dans ce cours, mais vous pouvez trouver plus d'informations dans le guide de l'utilisateur Ansible dans la section « Validation des certificats HTTPS » [[https://docs.ansible.com/ansible/latest/user\\_guide/windows\\_winrm.html#https-certificate-validation](https://docs.ansible.com/ansible/latest/user_guide/windows_winrm.html#https-certificate-validation)].

Pour plus d'informations sur les certificats WinRM, suivez le lien *How to: configure WinRM for HTTPS* répertorié dans la section References.

4. Indique à Windows Ansible d'utiliser la méthode de transport CredSSP, qui est décrite dans Préparation des hôtes Microsoft Windows pour l'automatisation. Windows dispose de plusieurs méthodes d'authentification que vous pouvez utiliser pour vous connecter à vos serveurs.

Le schéma suivant montre comment Ansible Tower utilise les informations d'identification stockées dans sa base de données pour se connecter aux hôtes Windows par le biais de la méthode de transport que vous définissez. Il utilise la connexion pour exécuter des playbooks ou des modules ad hoc.

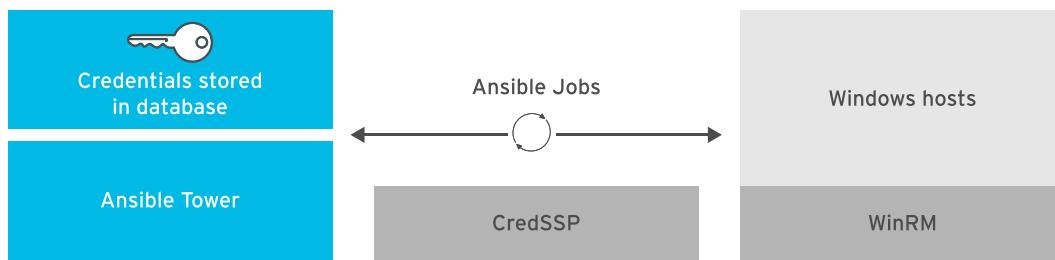


Figure 2.5: Connexion aux hôtes Windows

## Authentification de l'accès aux hôtes gérés

Pour qu'Ansible Tower puisse accéder à des hôtes gérés dans votre inventaire, il doit fournir des informations d'identification d'authentification à ces hôtes. Les *informations d'identification* d'Ansible Tower stockent en toute sécurité les données sensibles, telles que les informations d'authentification pour les hôtes gérés, les sources d'informations d'inventaire dynamiques et les référentiels Git. Ces informations d'identification sont stockées dans un format chiffré et peuvent être configurées de sorte que les utilisateurs d'Ansible Tower puissent les utiliser mais sans les voir.

Il existe de nombreux types d'informations d'identification pris en charge par Ansible Tower. Utilisez les *informations d'identification de la machine* pour vous authentifier auprès de vos hôtes d'inventaire. Ce type d'informations d'identification doit contenir des informations d'identification que votre méthode de connexion peut utiliser pour s'authentifier. Par exemple, elles peuvent contenir un nom d'utilisateur et un mot de passe pour un utilisateur sur vos systèmes Windows disposant de privilèges d'administrateur, de sorte qu'Ansible Tower puisse gérer les composants. Après avoir créé les informations d'identification, le mot de passe est chiffré de façon à ce que personne ne puisse le lire, mais qu'Ansible Tower puisse les déchiffrer et les utiliser.

Pour créer de nouvelles informations d'identification de machine :

1. Dans l'interface utilisateur web d'Ansible Tower, cliquez sur **Credentials** dans le volet de navigation.
2. Créez de nouvelles informations d'identification. Dans la fenêtre **CREDENTIALS**, cliquez sur le bouton **+**, qui affiche l'info-bulle « Create a new credential ».
3. Dans la fenêtre **NEW CREDENTIAL**, saisissez un nom pour les informations d'identification. Dans le champ **CREDENTIAL TYPE**, cliquez sur l'icône de la loupe et sélectionnez le type d'informations d'identification **Machine**. Elles peuvent se trouver sur la deuxième page des résultats de la recherche. Cliquez sur le bouton **SELECT**.  
Vous pouvez également saisir le mot **Machine** dans le champ.
4. Pour l'authentification basée sur un mot de passe, ajoutez un nom d'utilisateur et un mot de passe dans les champs **TYPE DETAILS** qui s'affichent. Si vous ne renseignez pas le champ **PASSWORD**, vous pouvez également activer la case à cocher **Prompt on launch**. Si cette case est cochée, Ansible Tower invite l'utilisateur à saisir le mot de passe lors de l'exécution des tâches d'automatisation qui utilisent ces informations d'identification.

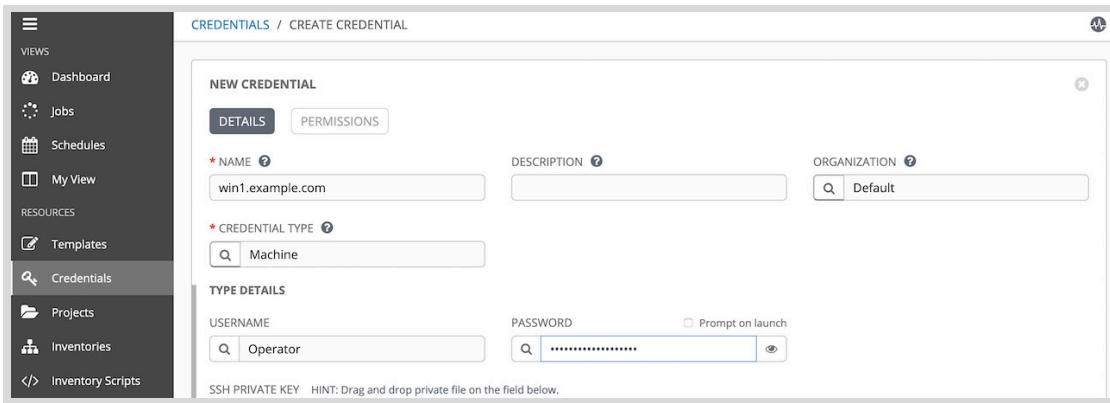


Figure 2.6: Création d'informations d'identification de machine

5. Cliquez sur **SAVE**.

La capture d'écran ci-dessous illustre l'affichage du champ de mot de passe après la création des informations d'identification de la machine.

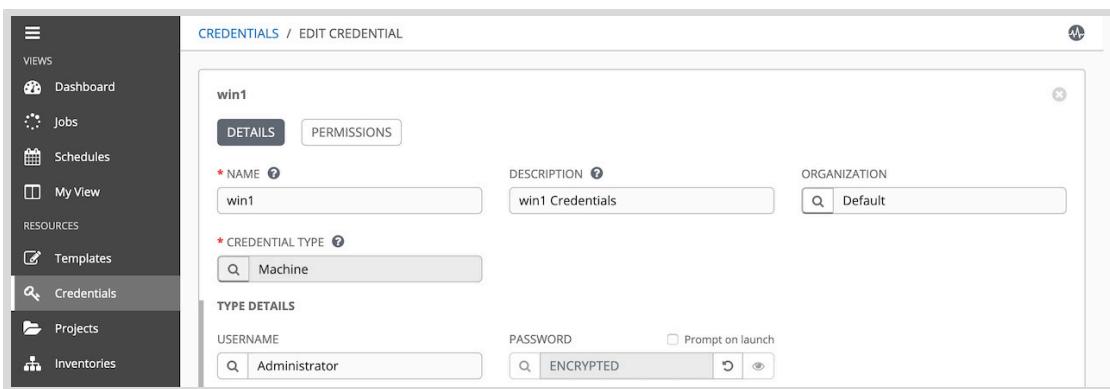


Figure 2.7: Affichage des informations d'identification de machine dans Ansible Tower



## Références

### Prise en charge de Windows — Documentation Ansible

[https://docs.ansible.com/ansible/2.3/intro\\_windows.html](https://docs.ansible.com/ansible/2.3/intro_windows.html)

### Types d'informations d'identification — Documentation Ansible

<https://docs.ansible.com/ansible-tower/latest/html/userguide/credentials.html#credential-types>

### Procédure : configurer WINRM pour HTTPS

<https://support.microsoft.com/en-ca/help/2019527/how-to-configure-winrm-for-https>

## ► Exercice guidé

# Préparation de Red Hat Ansible Tower pour gérer des hôtes

Au cours de cet exercice, vous allez configurer Red Hat Ansible Tower avec un inventaire des hôtes Windows à gérer, ainsi que les informations d'identification nécessaires pour vous authentifier et vous connecter à ces hôtes.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Gérer les inventaires et les variables d'inventaire dans Ansible Tower.
- Créer des groupes et des hôtes dans Ansible Tower.
- Authentifiez-vous auprès des hôtes à l'aide des informations d'identification de la machine.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. À partir de **workstation**, cliquez sur l'icône Ansible Tower sur votre Bureau pour accéder à Ansible Tower.  
Connectez-vous à la console Web en tant qu'**admin** avec le mot de passe **RedHat123@!**.
- ▶ 2. Avant de pouvoir interagir avec les hôtes, vous devez créer un inventaire et définir les informations de connexion. Dans les étapes suivantes, vous créez le groupe et ajoutez un hôte.
  - 2.1. Accédez à **Inventories** pour gérer les inventaires. Cliquez sur **Create a new inventory**, puis sélectionnez **Inventory** pour créer un nouvel inventaire statique.

2.2. Créez un nouvel inventaire à l'aide des informations suivantes :

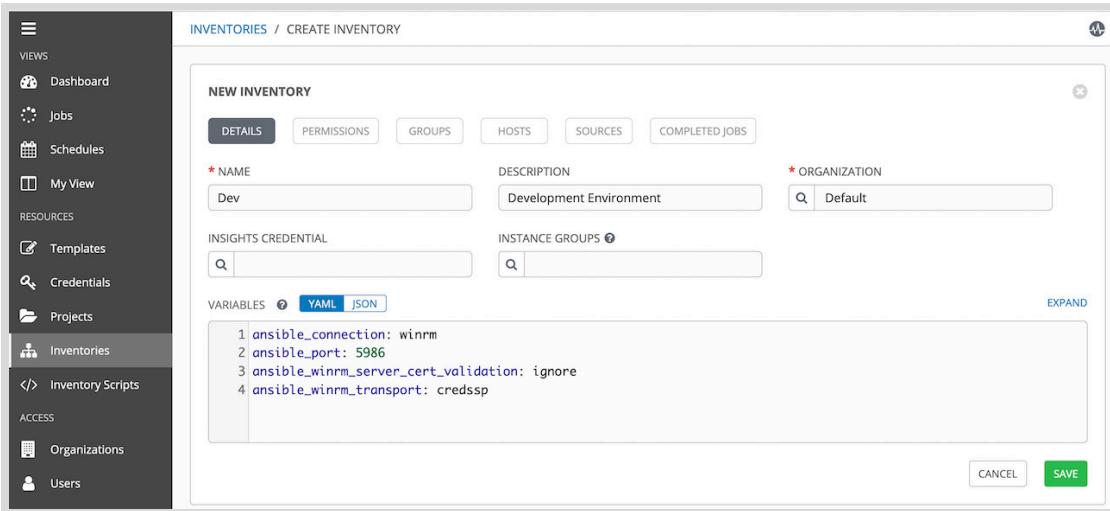
Champ	Valeur
Nom	Dev
Description	Environnement de développement
Organisation	Valeur par défaut

Laissez la valeur par défaut pour tous les autres champs.

2.3. Définissez des variables d'inventaire afin que tous les hôtes de l'inventaire héritent des paramètres. Saisissez les variables suivantes dans la zone **Variables**, puis cliquez sur **SAVE** pour créer l'inventaire.

```
ansible_connection: winrm ①
ansible_port: 5986 ②
ansible_winrm_server_cert_validation: ignore ③
ansible_winrm_transport: credssp ④
```

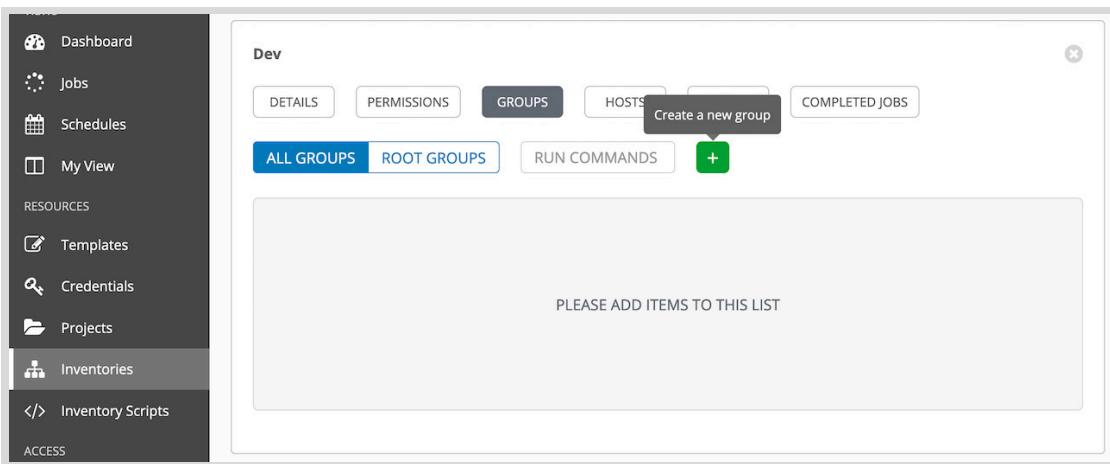
- ① Par défaut, Ansible utilise le protocole SSH pour se connecter à n'importe quel hôte. Vos serveurs Windows sont configurés avec un écouteur WinRM à la place, vous devez donc utiliser cette variable pour ordonner à Ansible d'utiliser la méthode de connexion WinRM.
- ② Cette variable ordonne à Ansible d'utiliser le port WinRM TLS chiffré 5986.
- ③ Cette variable ordonne à Ansible d'ignorer la validation du certificat WinRM, car l'environnement utilise des certificats auto-signés.
- ④ Cette variable ordonne à Ansible de s'authentifier à l'aide de CredSSP pour l'authentification.



### Note

YAML utilise trois tirets (---) pour séparer les directives dans un fichier YAML. Cela sert également à signaler le début d'un document si aucune directive n'est présente. Le fait de supprimer ou de laisser les tirets dans ce cas ne fait aucune différence.

- 2.4. Créez un groupe pour cet inventaire de développement en sélectionnant **GROUPS**, puis en cliquant sur le bouton **+**, qui affiche l'info-bulle **Create a new group**.



Nommez le groupe **dev\_servers**, puis cliquez sur **SAVE** pour créer le groupe.

- 2.5. Créez un nouvel hôte pour le groupe **dev\_servers** en sélectionnant **HOSTS**. Cliquez sur le bouton **+**, qui affiche l'info-bulle **Add a host**, puis sélectionnez **New Host** dans le menu affiché.

Utilisez le champ **HOST NAME** pour nommer le nouvel hôte **win2.example.com**. Étant donné que l'inventaire définit les informations de connexion, vous n'avez pas besoin de le spécifier dans la définition de l'hôte.

Cliquez sur **SAVE** pour créer l'hôte.

### Note

La piste en fil d'Ariane située en haut de la page affiche la structure de l'inventaire.

- 2.6. Examinez tous les hôtes de l'inventaire à partir de la page affichée lors de l'enregistrement du nouvel hôte. Cette page fournit l'accès aux opérations suivantes :
  - Inclusion ou exclusion d'hôtes dans les tâches ; cela est indiqué par le commutateur **ON/OFF**. Tous les hôtes créés sont inclus par défaut.
  - Modification des hôtes, y compris leur renommage ou leur suppression.
  - Exécution de commandes en invoquant des modules ad hoc.
  - Gestion des hôtes, y compris leur renommage ou leur suppression.
  
- ▶ 3. Passez en revue les informations d'identification de la machine fournies dans l'environnement.

- 3.1. Pour passer en revue les informations d'identification utilisées par Ansible Tower pour se connecter à vos hôtes, sélectionnez **Credentials** dans le menu principal.

Notez que le contrôleur de domaine Windows a un utilisateur **devops** qui appartient au groupe **Domain Admins**.

Cliquez sur l'icône de crayon **Edit credential** pour afficher les informations d'identification de **DevOps**.

NAME	KIND	OWNERS	ACTIONS
DevOps	Machine	Default	
GitLab	Source Control	Default	

Ces informations d'identification utilisent un type de **Machine**, qui est utilisé pour se connecter directement aux serveurs. Notez le nom de l'utilisateur qui correspond au nom de l'utilisateur dans Active Directory. Le mot de passe est chiffré et illisible. Des champs supplémentaires sont disponibles pour configurer l'authentification, mais la méthode de connexion que nous utilisons pour les systèmes Windows dans ces activités ne les utilise pas.

Cliquez sur **CANCEL** pour revenir à la page **CREDENTIALS**, puis déconnectez-vous.

L'exercice guidé est maintenant terminé.

# Exécution de commandes ad hoc

## Résultats

Au terme de cette section, vous devez pouvoir décrire ce qu'est un module Ansible et exécuter une seule opération sur les hôtes gérés sous la forme d'une commande ad hoc à l'aide de Red Hat Ansible Tower.

## Exécution de tâches d'automatisation uniques

Une *commande ad hoc* est un moyen d'exécuter rapidement une tâche Ansible unique que vous n'avez pas besoin de sauvegarder pour l'exécuter plus tard. Ce sont des opérations simples, en ligne, que vous pouvez exécuter sans écrire de playbook.

Les commandes ad hoc sont utiles pour les tests et les modifications rapides. Par exemple, vous pouvez utiliser une commande ad hoc pour vous assurer qu'une certaine ligne existe dans le fichier **%SystemRoot%\System32\drivers\etc\hosts** sur un groupe de serveurs. Vous pouvez utiliser une autre commande ad hoc pour redémarrer efficacement un service sur de nombreuses machines différentes, ou pour vous assurer qu'un paquetage logiciel particulier est à jour.

Cependant, les commandes ad hoc ne peuvent pas exécuter des tâches complexes ou répétitives ; utilisez les playbooks Ansible pour tirer pleinement parti de la puissance d'Ansible pour l'automatisation reproductible.



### Note

Les commandes ad hoc interagissent avec les *modules* Ansible, qui sont de petits programmes ou scripts qui sont exécutés sur les hôtes gérés pour implémenter une tâche d'automatisation. Chaque module a une utilisation particulière, telle que la création d'un utilisateur, l'installation d'un paquetage, l'exécution d'un script PowerShell ou la création d'une tâche planifiée.

Ansible est fourni avec de nombreux modules pour la gestion de l'environnement Windows. Chapitre 3, *Mise en œuvre de playbooks Ansible*, traite en détail des modules.

## Exécution d'une commande ad hoc dans Ansible Tower

Pour exécuter une commande ad hoc sur un ensemble de nœuds gérés :

1. Accédez à l'affichage **Inventories**, puis sélectionnez un inventaire.
2. Activez les cases à cocher en regard des hôtes sur lesquels vous souhaitez exécuter la commande.
3. Cliquez sur **RUN COMMANDS**.

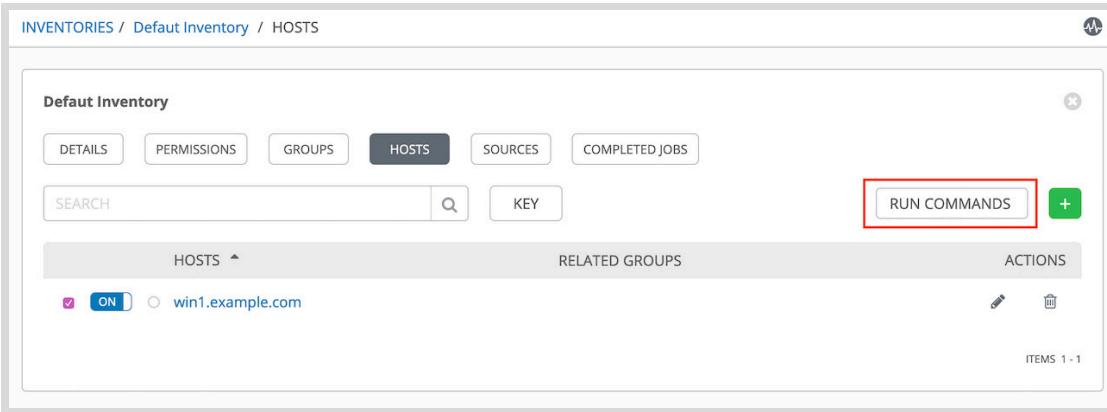


Figure 2.14: Utilisation de la fonction Run Commands

- In the **EXECUTE COMMAND** window, select a *module* from the **MODULE** list. Dans le champ **ARGUMENTS**, spécifiez les arguments requis par le module sélectionné. Bien que tous les modules ne requièrent pas d'arguments supplémentaires, nombreux d'entre eux en requièrent.

Dans le champ **MACHINE CREDENTIAL**, sélectionnez les informations d'identification de la machine devant s'authentifier auprès des hôtes gérés sur lesquels vous allez exécuter cette commande ad hoc. Utilisez l'icône de la loupe pour afficher la liste des informations d'identification de la machine disponible.

Le module **win\_ping** dans l'exemple suivant n'a besoin d'aucun argument, et vérifie si les hôtes sur lesquels il s'exécute peuvent exécuter l'automatisation Ansible.

Figure 2.15: Exécution de commandes sur les hôtes

Le point d'interrogation du champ **ARGUMENTS** est dynamique. Après avoir sélectionné un module, cliquez sur l'icône pour afficher un texte qui contient un lien ; ce lien pointe vers la documentation du module. Il s'agit d'une fonctionnalité utile pour déterminer rapidement les fonctionnalités du module et comment l'utiliser.

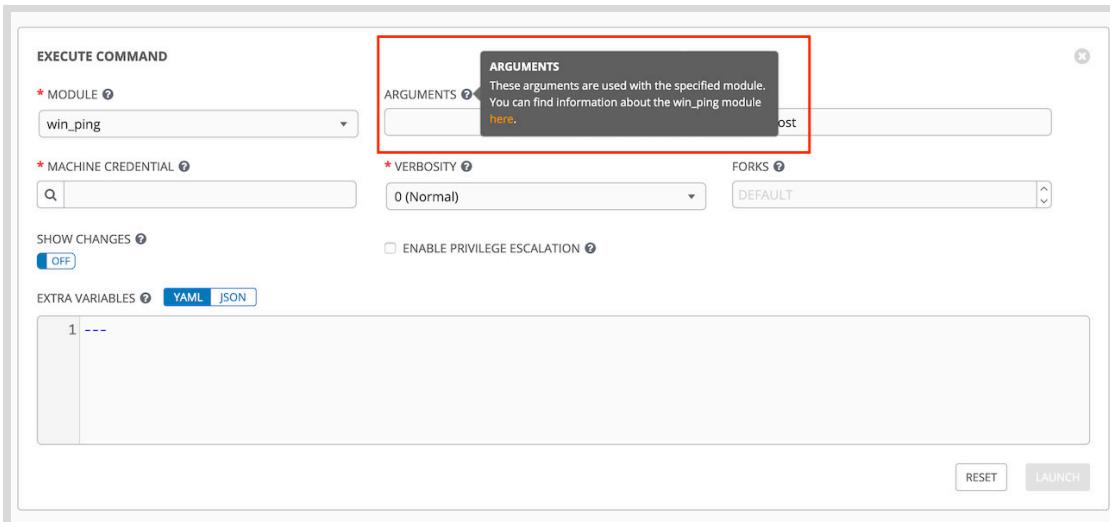


Figure 2.16: Accès à la documentation des modules

5. Cliquez sur **LAUNCH** pour exécuter la commande ad hoc.

```

win_ping
ELAPSED 00:00:18
SEARCH Q KEY
1 SSH password:
2 win1.example.com | SUCCESS => {
3   "changed": false,
4   "ping": "pong"
5 }

```

Figure 2.17: Résultat des commandes ad hoc Ansible Tower

6. Ansible Tower affiche un nouvel écran dans votre navigateur web pour la tâche que vous venez de démarrer. Lors de l'exécution de la tâche, le volet **DETAILS** est continuellement mis à jour avec les informations relatives à l'état de la tâche. Le volet de droite affiche le nom du module en cours d'exécution, ainsi que le résultat de la commande ad hoc Ansible. Ce résultat fournit davantage d'informations sur les réussites et les échecs des tâches, les avertissements et d'autres détails utiles résultant de la commande Ansible.

## Exécution de tâches avec les modules Ansible

Les modules sont les outils utilisés par Ansible pour effectuer des tâches. Ansible fournit des centaines de modules accomplissant différentes choses. Vous pouvez généralement trouver un module spécial testé qui fait ce dont vous avez besoin dans le cadre de l'installation standard.

**Note**

Par défaut, un sous-ensemble limité de modules peut être utilisé dans des commandes ad hoc exécutées par Ansible Tower. Cependant, tout module disponible peut être intégré à un playbook Ansible qui est exécuté par Ansible Tower.

Un module Ansible est PowerShell (pour les modules utilisés sous Windows) ou le code Python qui effectue une tâche spécifique, comme l'installation d'un logiciel ou la configuration d'un service. Le produit Ansible de base inclut de nombreux modules, et le tableau suivant répertorie plusieurs des composants spécifiques à Windows.

**Modules Ansible sélectionnés pour Windows**

Nom	Description
<b>win_command</b>	Permet l'exécution d'une commande sur des nœuds gérés, en tant qu'utilisateur spécifique, ou avec l'augmentation des priviléges sur un compte d'administration. En raison de sa simplicité et de sa flexibilité, le module n'est pas idempotent, c'est pourquoi il faut veiller à l'utiliser avec précaution.
<b>win_shell</b>	Active l'exécution de scripts PowerShell sur les nœuds gérés.
<b>win_package</b>	Active l'installation des fichiers MSI (Microsoft Software Installation) ou des paquetages logiciels exécutables.
<b>win_feature</b>	Gère les rôles ou les fonctions système pour les systèmes Windows.
<b>win_update</b>	Gère les mises à jour Windows.
<b>win_reboot</b>	Gère les redémarrages pour les systèmes Windows, par exemple, une fois que <b>win_update</b> a appliqué les mises à jour du système.
<b>win_partition</b>	Gère le partitionnement des disques.
<b>win_format</b>	Garantit que les volumes sont formatés.
<b>win_hostname</b>	Contrôle le nom de la machine système.

**Important**

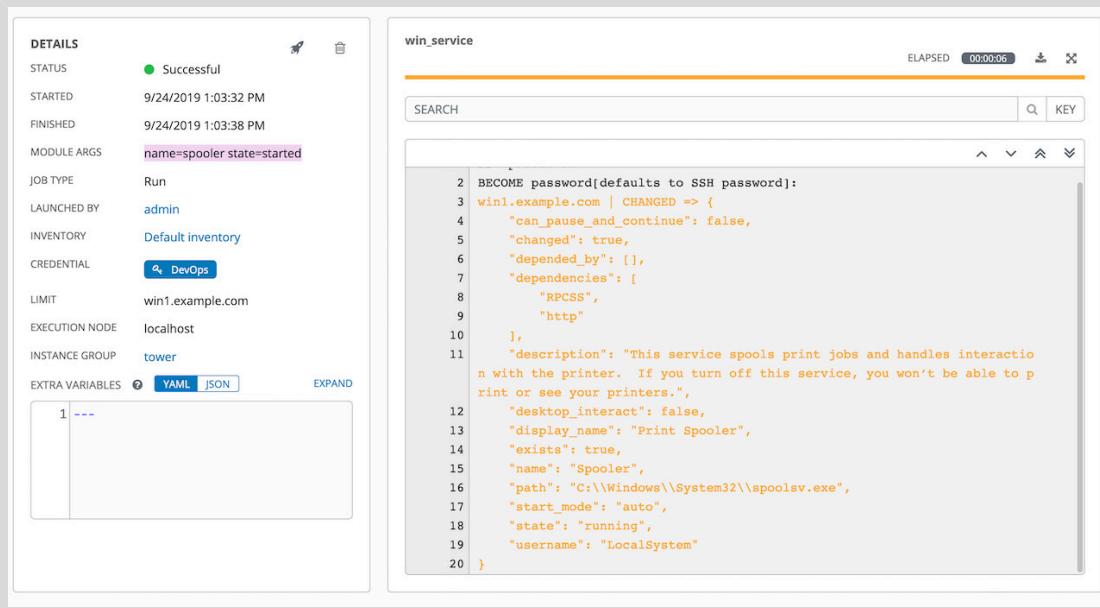
Pour obtenir une liste complète des modules disponibles et de la documentation relative à leur utilisation, voir Modules Windows — Documentation Ansible [[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_windows\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html)].

La plupart des modules nécessitent des arguments pour contrôler leur fonctionnement. Une liste des arguments disponibles pour un module est disponible dans la documentation de ce dernier. Lorsque vous exécutez un module dans une commande ad hoc via Ansible Tower, spécifiez n'importe quel argument dans le champ **ARGUMENTS** de la page **EXECUTE COMMAND**. Ce champ accepte les arguments entrés sous la forme d'une liste d'éléments *argument=value* séparés par des espaces.

## chapitre 2 | Exécution de commandes simples d'automatisation

Par exemple, pour démarrer le service **spooler** sur des hôtes gérés sélectionnés à l'aide d'une commande ad hoc, sélectionnez le module **win\_service**, puis saisissez le texte **name=spooler state=started** dans le champ **ARGUMENTS**.

Cela indique à Ansible de gérer le service **spooler** pour s'assurer qu'il est en cours d'exécution. Cette opération est effectuée par l'utilisation de deux arguments : **name** et **state**.



The screenshot shows the Ansible web interface. On the left, the 'DETAILS' panel displays job information: STATUS Successful, STARTED 9/24/2019 1:03:28 PM, FINISHED 9/24/2019 1:03:38 PM, MODULE ARG name=spooler state=started, JOB TYPE Run, LAUNCHED BY admin, INVENTORY Default inventory, CREDENTIAL DevOps, LIMIT win1.example.com, EXECUTION NODE localhost, INSTANCE GROUP tower, EXTRA VARIABLES. On the right, the 'win\_service' panel shows the module code:

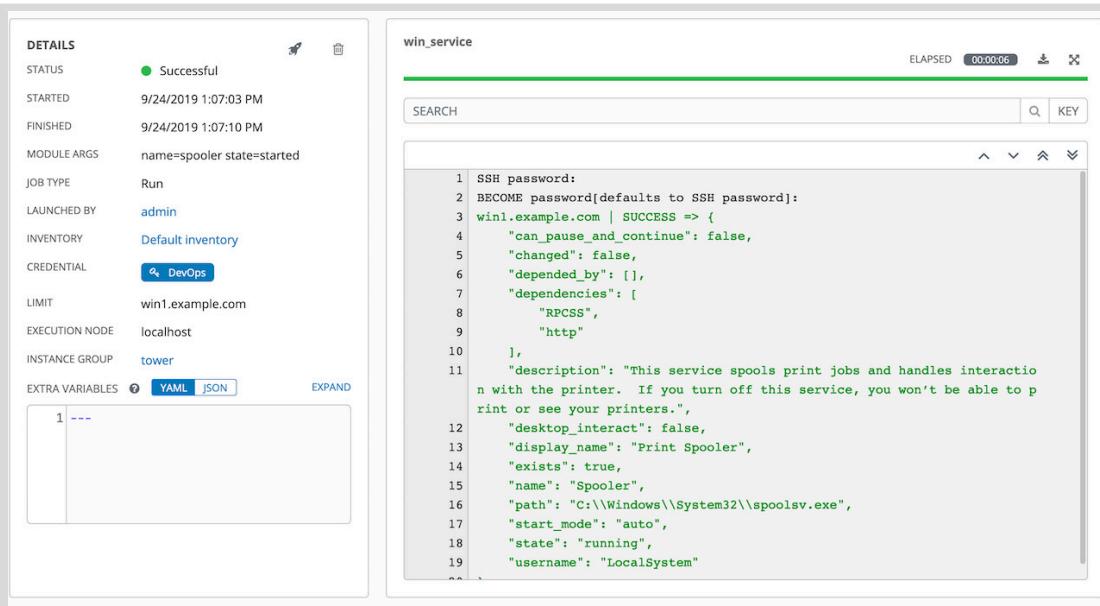
```

2 BECOME password[defaults to SSH password];
3 win1.example.com | CHANGED => {
4     "can_pause_and_continue": false,
5     "changed": true,
6     "depended_by": [],
7     "dependencies": [
8         "RPCSS",
9         "http"
10    ],
11    "description": "This service spools print jobs and handles interaction with the printer. If you turn off this service, you won't be able to print or see your printers.",
12    "desktop_interact": false,
13    "display_name": "Print Spooler",
14    "exists": true,
15    "name": "Spooler",
16    "path": "C:\\Windows\\System32\\spoolsv.exe",
17    "start_mode": "auto",
18    "state": "running",
19    "username": "LocalSystem"
20 }

```

Figure 2.18: Exécution de commandes ad hoc dans Ansible

La plupart des modules sont *idempotents*, ce qui signifie que vous pouvez les exécuter en toute sécurité plusieurs fois, et que si le système est déjà à l'état correct, ils ne font rien. Par exemple, si vous réexécutez la commande ad hoc précédente, elle ne devrait signaler aucun changement étant donné que le service **spooler** est déjà en cours d'exécution. Cette opération est affichée en tant que modification de la couleur du texte.



The screenshot shows the Ansible web interface. The 'DETAILS' panel is identical to Figure 2.18. The 'win\_service' panel shows the module code with color-coded changes, indicating that the module has already been run and no changes were made:

```

1 SSH password:
2 BECOME password[defaults to SSH password];
3 win1.example.com | SUCCESS => {
4     "can_pause_and_continue": false,
5     "changed": false,
6     "depended_by": [],
7     "dependencies": [
8         "RPCSS",
9         "http"
10    ],
11    "description": "This service spools print jobs and handles interaction with the printer. If you turn off this service, you won't be able to print or see your printers.",
12    "desktop_interact": false,
13    "display_name": "Print Spooler",
14    "exists": true,
15    "name": "Spooler",
16    "path": "C:\\Windows\\System32\\spoolsv.exe",
17    "start_mode": "auto",
18    "state": "running",
19    "username": "LocalSystem"
20 }

```

Figure 2.19: Ré-exécution de commandes ad hoc dans Ansible

## Gestion des modules Ansible pour les commandes ad hoc

Dans Ansible Tower, vous pouvez gérer la liste des modules à utiliser pour les commandes ad hoc. L'option est disponible dans **Settings → Jobs**. La zone **ANSIBLE MODULES ALLOWED FOR AD HOC JOBS** liste tous les modules que vous pouvez utiliser pour les commandes ad hoc. À partir de cette liste, vous pouvez ajouter ou supprimer un module : pour supprimer un module, cliquez sur l'icône de suppression en regard du nom du module, et pour ajouter un module, tapez son nom, puis appuyez sur **Entrée**. Pour supprimer un module, vous pouvez également cliquer sur le champ pour afficher une liste indiquant tous les modules actifs et sélectionner un module dans cette liste.

La capture d'écran suivante montre comment ajouter le module **win\_scheduled\_task** à la liste.

The screenshot shows the 'SETTINGS / JOBS' page in Ansible Tower. In the 'JOBS' section, under 'ANSIBLE MODULES ALLOWED FOR AD HOC JOBS', there is a list of modules. A red box highlights the input field where 'win\_scheduled\_task' has been typed. Other listed modules include ping, selinux, setup, win\_ping, win\_service, win\_updates, win\_group, win\_user, win\_shell, win\_feature, and win\_reboot.

Figure 2.20: Gestion des modules Ansible pour les travaux ad hoc

Après l'avoir ajouté, le module est disponible pour exécuter des commandes ad hoc :

The screenshot shows the 'EXECUTE COMMAND' dialog. In the 'MODULE' dropdown, the 'win\_scheduled\_task' module is selected and highlighted with a red box. Other options in the dropdown include win\_shell, win\_feature, win\_reboot, yum, and command.

Figure 2.21: Création d'une liste des modules Ansible pour les travaux ad hoc

## Exécution de commandes arbitraires sur des hôtes gérés

Le module **win\_command** vous permet d'exécuter des commandes arbitraires sur des hôtes gérés. La commande à exécuter est transmise au module en tant qu'argument. Par exemple, pour exécuter la commande **hostname** sur les hôtes gérés en tant que commande ad hoc, vous devez simplement placer **hostname** dans la zone **ARGUMENTS**.



The screenshot shows the Ansible Win Shell interface. At the top, it says "win\_shell". In the center, there is a text area containing the following command:

```

1 SSH password:
2 BECOME password[defaults to SSH password]:
3 win1.example.com | CHANGED | rc=0 >>
4 win1
5
6

```

At the bottom right of the interface, there are navigation icons: up, down, left, and right arrows.

Figure 2.22: Récupération des noms d'hôtes à l'aide de commandes ad hoc



### Note

Si un module ne figure pas dans la liste des modules que vous pouvez utiliser pour des commandes ad hoc, accédez à **Settings** → **Jobs** et ajoutez le module à la zone **MODULE**.

Le module **win\_command** vous permet d'exécuter rapidement des commandes à distance sur des hôtes gérés. Ces commandes ne sont pas traitées par le shell sur les hôtes gérés. Par conséquent, elles ne peuvent pas accéder aux variables d'environnement ni effectuer des opérations shell telles que `<`, `>`, `|` et `;`.

Pour les situations dans lesquelles les commandes ont besoin d'un traitement shell, les administrateurs peuvent utiliser le module **win\_shell**. Comme pour le module **win\_command**, vous passez les commandes à exécuter sous forme d'arguments à un module dans la commande ad hoc. Ansible exécute ensuite la commande à distance sur les hôtes gérés. Contrairement au module **win\_command**, les commandes sont traitées à travers un shell (par défaut, PowerShell) sur les hôtes gérés. Par conséquent, les variables d'environnement sont accessibles et les opérations shell seront également disponibles.

**Important**

Dans la plupart des cas, il est recommandé d'éviter les modules « run command » **win\_command** et **win\_shell** si vous pouvez utiliser d'autres modules pour accomplir la même chose.

La plupart des autres modules sont idempotents et peuvent effectuer le suivi des modifications automatiquement. Ils peuvent tester l'état des systèmes et ne rien faire si ces derniers présentent déjà l'état correct. En revanche, il est beaucoup plus compliqué d'utiliser les modules « run command » d'une manière idempotente. Si vous dépendez d'eux, il est plus difficile de s'assurer que la réexécution d'une commande ad hoc ou d'un playbook ne provoquera pas un échec inattendu.

Lorsque le module **win\_shell** ou **win\_command** s'exécute, il signale généralement un statut **CHANGED**, car ces modules ne peuvent pas détecter si l'état de la machine est déjà correct. Un module idempotent peut détecter si l'état de la machine est correct, ne rien faire et signaler **OK**.

Dans certains cas, les modules « run command » sont des outils précieux et constituent une solution appropriée à un problème. Si vous n'avez pas besoin de les utiliser, il est préférable d'essayer d'abord le module **win\_command**, en ayant recours au module **win\_shell** uniquement si vous avez besoin de ses fonctions spéciales.

## Comparaison des commandes ad hoc et des playbooks Ansible

Les commandes ad hoc sont utiles lorsque vous devez exécuter une tâche une seule fois, qu'elle est très simple et que vous n'envisagez pas de la ré-exécuter. Cependant, les commandes ad hoc présentent certains inconvénients.

Une limitation des commandes ad hoc est que vous ne pouvez utiliser qu'un seul module comme tâche unique. Un playbook Ansible vous permet d'écrire une automatisation plus complexe qui exécute une séquence de tâches.

Une autre limitation est qu'une commande ad hoc est créée et exécutée en fonction des besoins. Autrement dit, vous l'inventez « à la volée » et n'enregistrez pas la commande après son exécution. Les systèmes de contrôle de version gèrent normalement les playbooks Ansible pour que vous puissiez les réutiliser facilement.

Enfin, Ansible Tower ne fournit normalement qu'un ensemble limité de modules à utiliser en tant que commandes ad hoc. L'administrateur de serveur Ansible Tower peut modifier ces paramètres dans **Settings → Jobs**, mais c'est toujours moins souple que d'écrire et utiliser un playbook.

Le chapitre suivant de ce cours vous apprend à écrire et utiliser les playbooks Ansible.

**Références****Contrôle avec Ansible Tower, 2e partie**

<https://www.ansible.com/blog/control-with-ansible-tower-part-2>

**Modules Windows — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_windows\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html)

## ► Exercice guidé

# Exécution de commandes ad hoc

Au cours de cet exercice, vous allez exécuter une seule opération sur un hôte géré en tant que commande ad hoc dans Red Hat Ansible Tower.

## Résultats

Vous devez être en mesure d'exécuter des commandes ad hoc sur un hôte géré à partir d'Ansible Tower.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

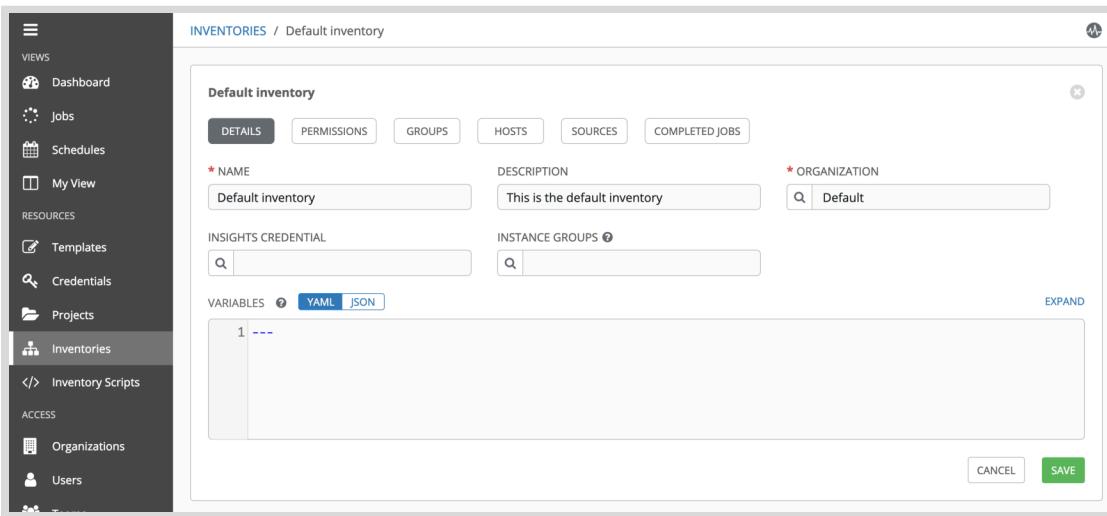
Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. À partir de **workstation**, accédez à l'instance Ansible Tower disponible sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**. Accédez à la page **Default inventory** dans la zone **Inventories** du panneau de configuration Ansible Tower, puis ouvrez l'écran d'exécution de la commande ad hoc pour l'hôte **win1.example.com**.
  - 1.1. À partir de **workstation**, double-cliquez sur l'icône de raccourci **Ansible Tower** sur votre Bureau pour accéder à votre instance Ansible Tower située sur `http://tower.example.com`.
  - 1.2. Connectez-vous à Ansible Tower en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

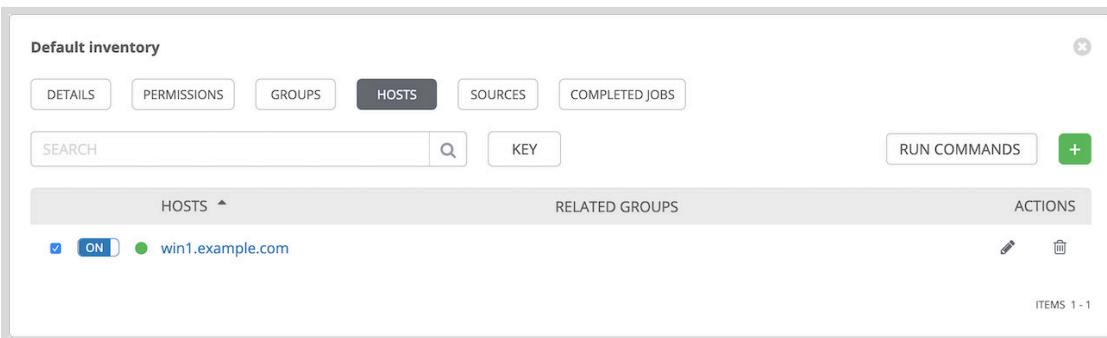
The screenshot shows the Ansible Tower sign-in interface. At the top left is the Red Hat logo with the text "Red Hat Ansible Tower". Below this, a message reads "Welcome to Ansible Tower! Please sign in.". There are two input fields: one labeled "USERNAME" containing "student" and another labeled "PASSWORD" containing ".....". At the bottom right is a green "SIGN IN" button.

## chapitre 2 | Exécution de commandes simples d'automatisation

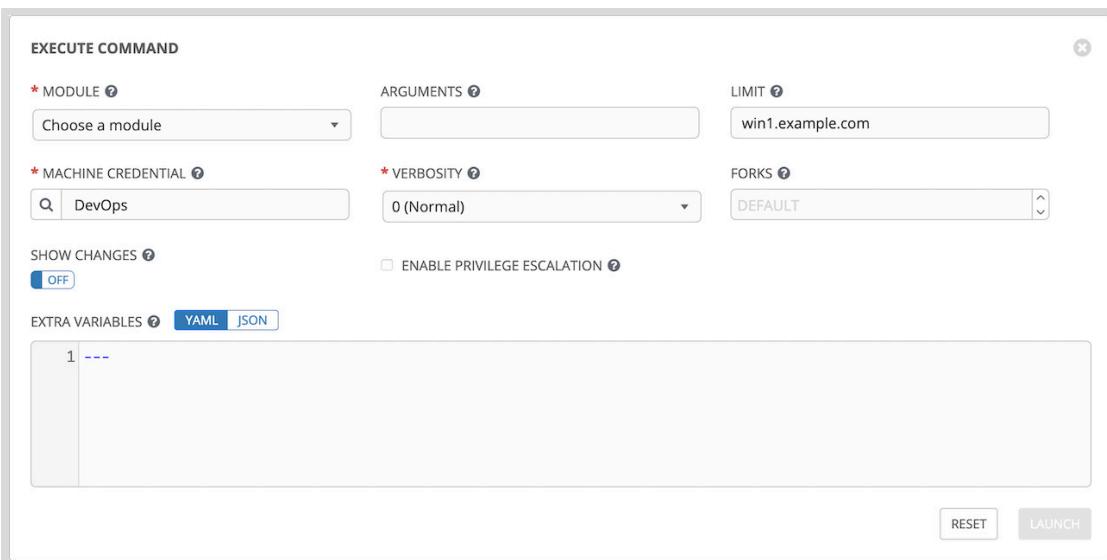
13. Dans la page principale du tableau de bord, sélectionnez **Inventories** dans la barre latérale et choisissez **Default inventory** dans les inventaires listés.



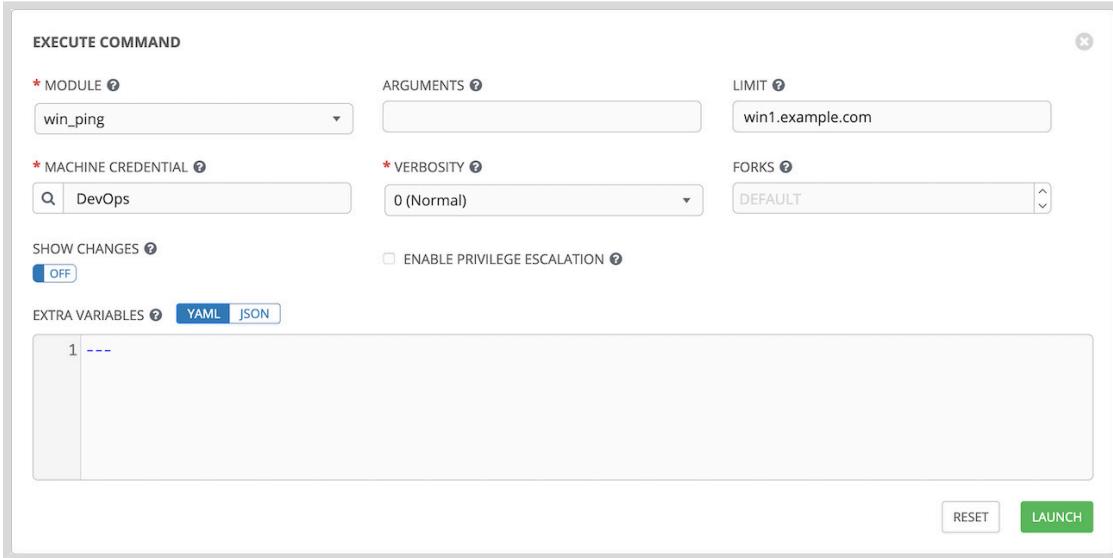
14. Dans la page **Default inventory**, cliquez sur **HOSTS**, puis sélectionnez l'entrée d'hôte **win1.example.com**.



15. Après avoir vérifié **win1.example.com**, cliquez sur **RUN COMMANDS** pour accéder à l'écran d'exécution de la commande ad hoc.



- 2. À l'aide du module **win\_ping**, vérifiez que le serveur **win1.example.com** est en cours d'exécution et accessible par Ansible.
- 2.1. Sur la page **RUN COMMAND**, sélectionnez le module **win\_ping** dans la liste **MODULE**.
  - 2.2. Cliquez sur l'icône du point d'interrogation dans le champ **ARGUMENTS**. Affiche une fenêtre qui contient un lien qui pointe vers la documentation de **win\_ping**.
  - 2.3. Cliquez sur **LAUNCH** pour exécuter la commande ad hoc.



- 2.4. Examinez la sortie pour vérifier que le module a bien été exécuté, ce qui confirme qu'Ansible a correctement communiqué avec le serveur.

```
win_ping
ELAPSED 00:00:05
SEARCH KEY
^ v ⌂ ⌃
1 SSH password:
2 win1.example.com | SUCCESS => {
3   "changed": false,
4   "ping": "pong"
5 }
```

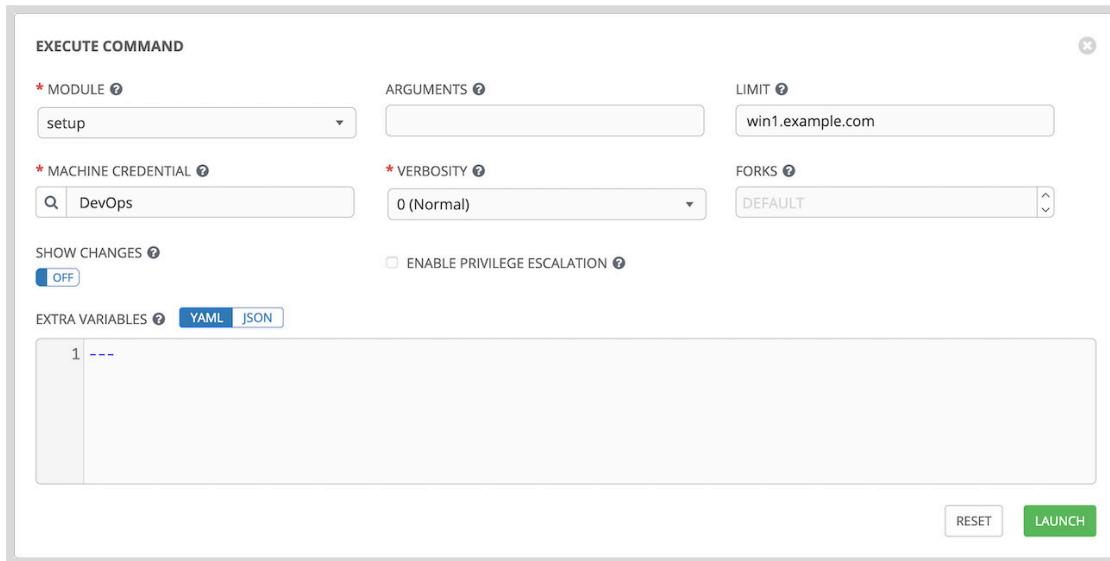
- 3. Revenez à la page **RUN COMMANDS** pour le serveur **win1.example.com**.

À l'aide du module **setup**, rassemblez les faits Ansible pour le serveur **win1.example.com**. Vous allez explorer l'utilisation de ces informations de sortie et utiliser ces faits disponibles pour Ansible, dans les chapitres et exercices suivants.

**chapitre 2 |** Exécution de commandes simples d'automatisation

3.1. Revenez à la page **RUN COMMANDS** pour le serveur **win1.example.com**.

Sur la page **RUN COMMAND**, sélectionnez le module **setup** dans la liste **MODULE**, puis cliquez sur **LAUNCH**.



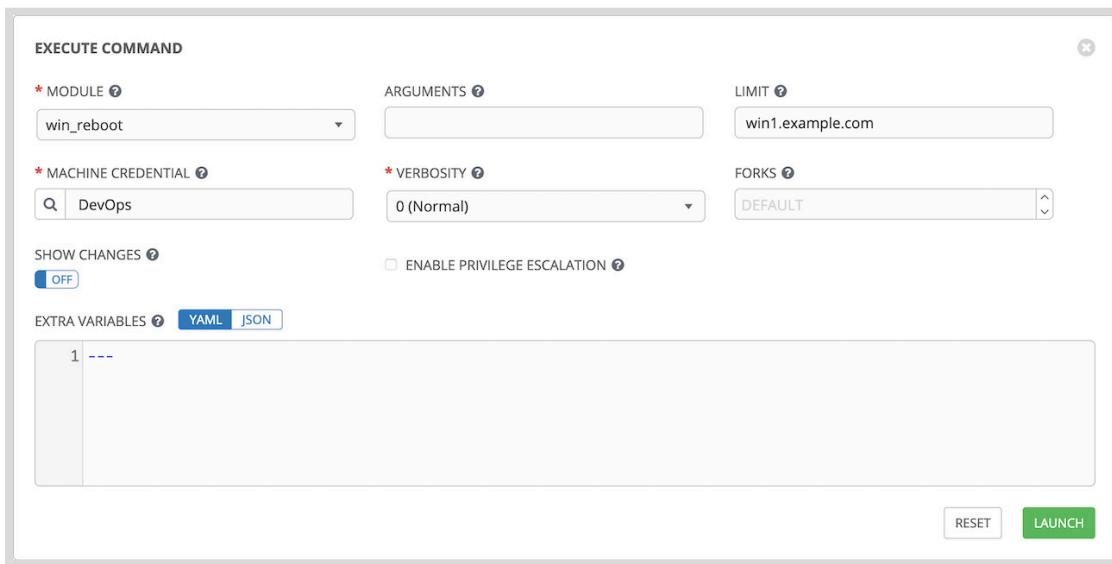
3.2. Examinez la sortie pour vérifier que le module a bien été exécuté, qu'Ansible a contacté le système et profilé divers aspects de l'environnement. Notez les différents faits collectés par Ansible, car vous les utiliserez dans des exercices de chapitre suivants.

```
setup
ELAPSED 00:00:09
SEARCH 🔎 KEY ↑ ↓ ↺ ↻

1 SSH password:
2 win1.example.com | SUCCESS => {
3     "ansible_facts": {
4         "ansible_architecture": "64-bit",
5         "ansible_bios_date": "10/16/2017",
6         "ansible_bios_version": "1.0",
7         "ansible_date_time": {
8             "date": "2019-08-26",
9             "day": "26",
10            "epoch": "1566855668.84827",
11            "hour": "21",
12            "iso8601": "2019-08-26T21:41:08Z",
13            "iso8601_basic": "20190826T214108848270",
14            "iso8601_basic_short": "20190826T214108" } }
```

► 4. À l'aide du module **win\_reboot**, redémarrez le serveur **win1.example.com**.

- 4.1. Sur la page **RUN COMMAND**, sélectionnez le module **win\_reboot** dans la liste **MODULE**, puis cliquez sur **LAUNCH**.

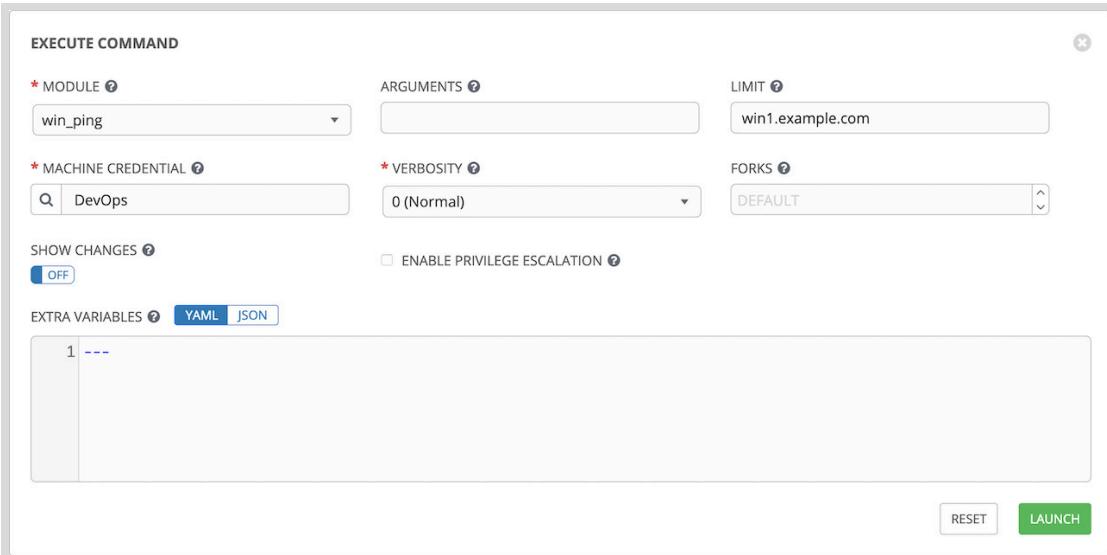


- 4.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement et qu'Ansible a redémarré le serveur.

```
1 SSH password:
2 win1.example.com | CHANGED => {
3     "changed": true,
4     "elapsed": 33,
5     "rebooted": true
6 }
```

- 5. À l'aide du module **win\_ping**, vérifiez que le serveur **win1.example.com** est en cours d'exécution et accessible par Ansible.

- 5.1. Sur la page **RUN COMMAND**, sélectionnez le module **win\_ping** dans la liste **MODULE**, puis cliquez sur **LAUNCH**.



- 5.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement, que le système s'exécute après le redémarrage et qu'Ansible peut correctement communiquer avec celui-ci.

The screenshot shows the execution results for the 'win\_ping' module. The output is as follows:

```
1 SSH password:  
2 win1.example.com | SUCCESS => {  
3     "changed": false,  
4     "ping": "pong"  
5 }
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Exécution de commandes simples d'automatisation

### Liste de contrôle des performances

Au cours de cet atelier, vous allez utiliser Ansible Tower pour exécuter plusieurs commandes ad hoc sur un hôte Windows dans votre inventaire.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Accédez à la console Web Ansible Tower et examinez l'inventaire par défaut.
- Sélectionnez un hôte pour l'exécution de commandes ad hoc.
- Exécutez plusieurs commandes ad hoc sur l'hôte cible.
- Connectez-vous à l'hôte cible pour vérifier la réussite des commandes ad hoc.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.  
Accédez à la fenêtre **Default inventory** dans la zone **Inventories** du panneau de configuration Ansible Tower, puis ouvrez l'écran d'exécution de la commande ad hoc pour l'hôte **win1.example.com**.
2. À l'aide du module **win\_ping**, vérifiez que le serveur **win1.example.com** est en cours d'exécution et accessible par Ansible.
3. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.  
À l'aide du module **setup**, rassemblez les faits Ansible pour le serveur **win1.example.com**. Vous allez explorer l'utilisation de ces informations de sortie et utiliser ces faits disponibles pour Ansible dans les chapitres et exercices suivants.
4. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.  
À l'aide du module **win\_feature**, installez la fonction **Web-Server** sur le serveur **win1.example.com**.
5. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

À l'aide du module **win\_service**, démarrez le service de serveur Web **W3Svc** sur le serveur **win1.example.com**.

6. Connectez-vous au serveur **win1.example.com** pour vérifier l'installation du service Web, et qu'il est en cours d'exécution.

Après avoir vérifié qu'Ansible a correctement installé la fonction de serveur Web et démarré le service, déconnectez-vous de **win1.example.com**.

7. Utilisez des commandes ad hoc supplémentaires Ansible pour rétablir les modifications apportées à l'environnement. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

À l'aide du module **win\_service**, arrêtez le serveur Web **W3Svc** sur le serveur **win1.example.com**.

8. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

À l'aide du module **win\_feature**, désinstallez la fonction de serveur Web sur le serveur **win1.example.com**.

9. À l'aide du module **win\_reboot**, redémarrez le serveur **win1.example.com**.

10. À l'aide du module **win\_ping**, vérifiez que le serveur **win1.example.com** est en cours d'exécution et accessible par Ansible.

L'atelier est maintenant terminé.

## ► Solution

# Exécution de commandes simples d'automatisation

### Liste de contrôle des performances

Au cours de cet atelier, vous allez utiliser Ansible Tower pour exécuter plusieurs commandes ad hoc sur un hôte Windows dans votre inventaire.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Accédez à la console Web Ansible Tower et examinez l'inventaire par défaut.
- Sélectionnez un hôte pour l'exécution de commandes ad hoc.
- Exécutez plusieurs commandes ad hoc sur l'hôte cible.
- Connectez-vous à l'hôte cible pour vérifier la réussite des commandes ad hoc.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

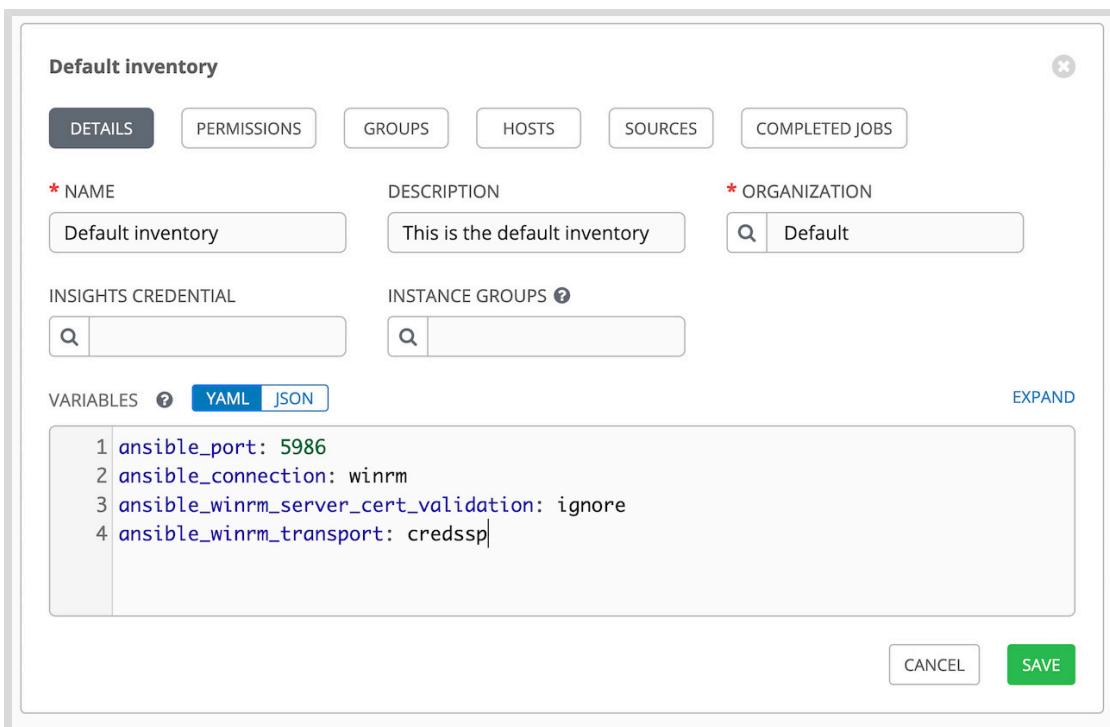
Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.  
Accédez à la fenêtre **Default inventory** dans la zone **Inventories** du panneau de configuration Ansible Tower, puis ouvrez l'écran d'exécution de la commande ad hoc pour l'hôte **win1.example.com**.
  - 1.1. À partir de **workstation**, double-cliquez sur l'icône de raccourci **Ansible Tower** sur votre Bureau pour accéder à votre instance Ansible Tower située sur `http://tower.example.com`.
  - 1.2. Connectez-vous à Ansible Tower en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.



- 1.3. Dans la fenêtre principale du tableau de bord, sélectionnez **Inventories** dans la barre latérale et choisissez **Default inventory** dans les inventaires listés.



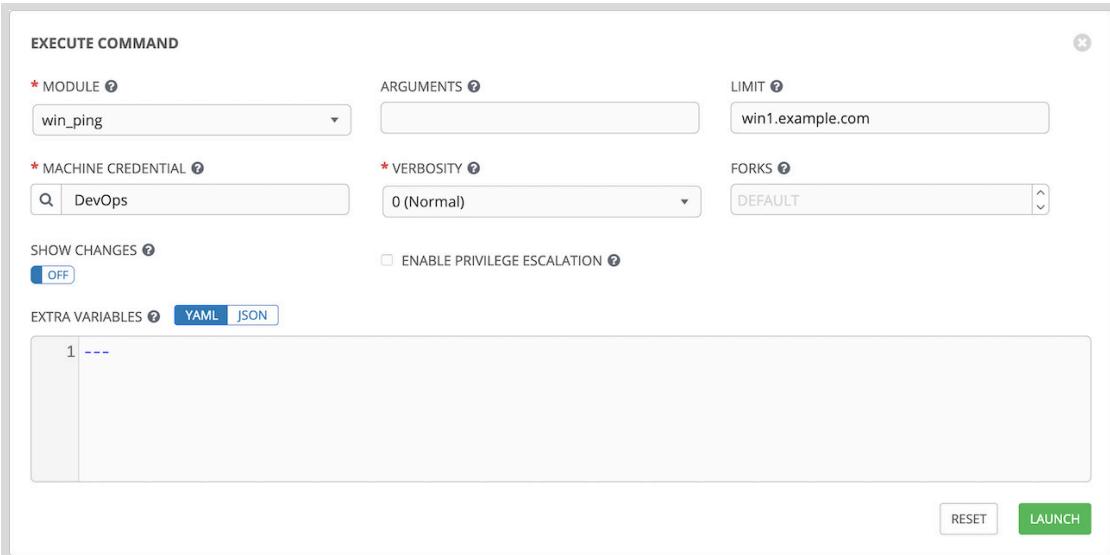
- 1.4. Dans la fenêtre **Default inventory**, sélectionnez le bouton **HOSTS** et cochez la case en regard de l'entrée d'hôte **win1.example.com**.

The screenshot shows the 'Default inventory' interface. At the top, there are tabs: DETAILS, PERMISSIONS, GROUPS, HOSTS (which is selected), SOURCES, and COMPLETED JOBS. Below the tabs is a search bar with a magnifying glass icon and a 'KEY' button. To the right of the search bar are 'RUN COMMANDS' and a '+' button. The main area has three columns: 'HOSTS' (with a dropdown arrow), 'RELATED GROUPS', and 'ACTIONS'. A single host entry, 'win1.example.com', is listed with a status of 'ON'. Below the table, it says 'ITEMS 1 - 1'.

- 1.5. Après avoir sélectionné **win1.example.com**, cliquez sur **RUN COMMANDS** pour accéder à la fenêtre **EXECUTE COMMAND** qui permet d'exécuter des commandes ad hoc.

The screenshot shows the 'EXECUTE COMMAND' interface. It includes fields for 'MODULE' (set to 'Choose a module'), 'ARGUMENTS' (empty), 'LIMIT' (set to 'win1.example.com'), 'MACHINE CREDENTIAL' (set to 'DevOps'), 'VERBOSITY' (set to '0 (Normal)'), 'FORKS' (set to 'DEFAULT'), 'SHOW CHANGES' (set to 'OFF'), and 'ENABLE PRIVILEGE ESCALATION' (unchecked). Below these are 'EXTRA VARIABLES' sections for YAML and JSON, both containing a single entry: '1 ---'. At the bottom are 'RESET' and 'LAUNCH' buttons.

2. À l'aide du module **win\_ping**, vérifiez que le serveur **win1.example.com** est en cours d'exécution et accessible par Ansible.
- 2.1. Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **win\_ping** dans la liste **MODULE**, puis cliquez sur **LAUNCH**.



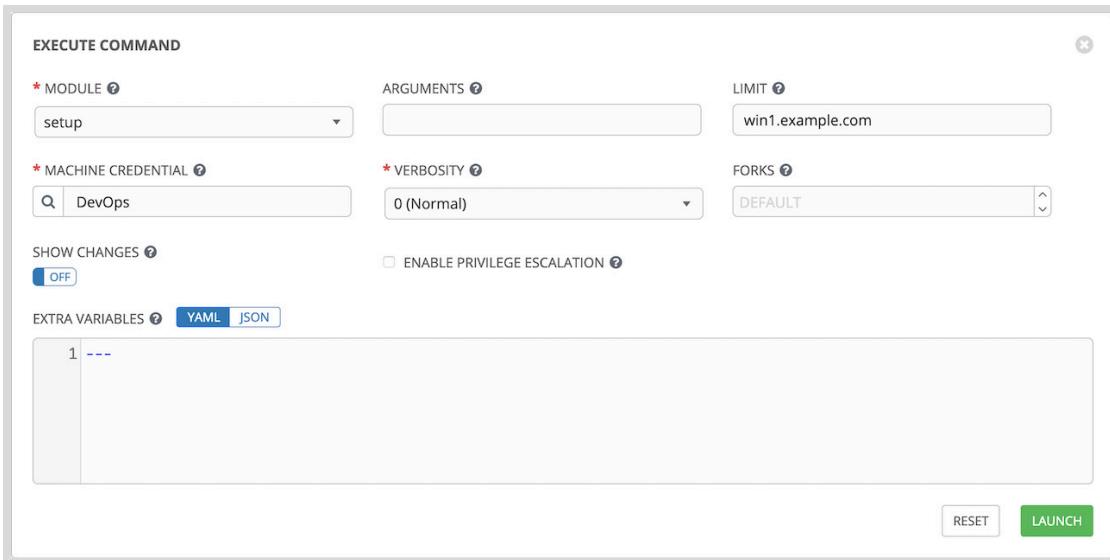
- 2.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement, ce qui confirme qu'Ansible a pu communiquer correctement avec le serveur.

The screenshot shows the execution output for the 'win\_ping' module. The output is as follows:

```
1 SSH password:  
2 win1.example.com | SUCCESS => {  
3     "changed": false,  
4     "ping": "pong"  
5 }
```

3. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**. À l'aide du module **setup**, rassemblez les faits Ansible pour le serveur **win1.example.com**. Vous allez explorer l'utilisation de ces informations de sortie et utiliser ces faits disponibles pour Ansible dans les chapitres et exercices suivants.

- 3.1. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**. Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **setup** dans la liste **MODULE**, puis cliquez sur **LAUNCH**.



- 3.2. Examinez la sortie pour vérifier que le module a bien été exécuté qu'Ansible a contacté le système et profilé divers aspects de l'environnement. Notez les différents faits collectés par Ansible, car vous les utiliserez dans des exercices de chapitre suivants.

setup

ELAPSED 00:00:09  

---

SEARCH  KEY

```
SSH password:  
win1.example.com | SUCCESS => {  
    "ansible_facts": {  
        "ansible_architecture": "64-bit",  
        "ansible_bios_date": "10/16/2017",  
        "ansible_bios_version": "1.0",  
        "ansible_date_time": {  
            "date": "2019-08-26",  
            "day": "26",  
            "epoch": "1566855668.84827",  
            "hour": "21",  
            "iso8601": "2019-08-26T21:41:08Z",  
            "iso8601_basic": "20190826T21410884  
8270",  
            "iso8601_basic_short": "20190826T21  
4108",  
        }  
    }  
}
```

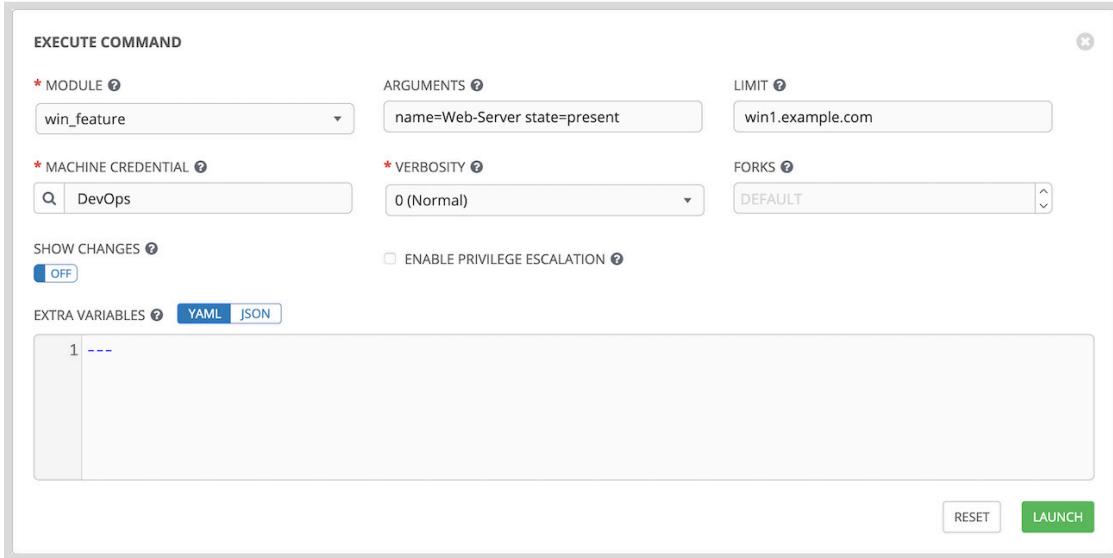
4. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

**chapitre 2 |** Exécution de commandes simples d'automatisation

À l'aide du module **win\_feature**, installez la fonction **Web-Server** sur le serveur **win1.example.com**.

- Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **win\_feature** dans la liste **MODULE**. Dans le champ **ARGUMENTS**, fournissez les arguments **name=Web-Server state=present**, puis cliquez sur **LAUNCH**.



- Examinez la sortie pour vérifier que le module s'est exécuté correctement et qu'Ansible a ajouté la fonction.

```

{
  "changed": true,
  "exitcode": "Success",
  "feature_result": [
    {
      "display_name": "Common HTTP Features",
      "id": 141,
      "message": [],
      "reboot_required": false,
      "restart_needed": false,
      "skip_reason": "NotSkipped",
      "success": true
    }
  ]
}
  
```

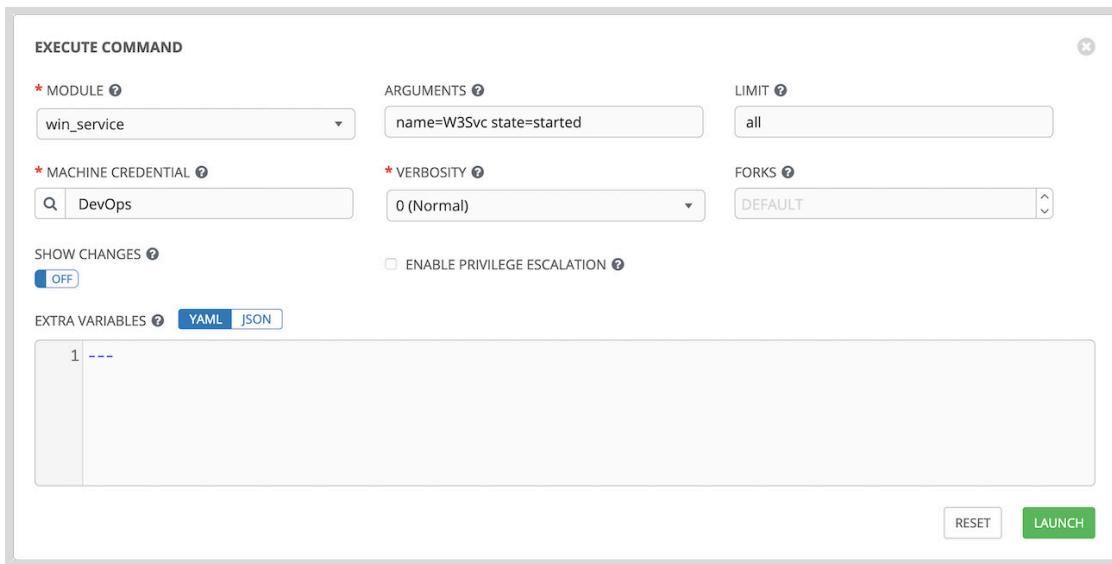
- Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

À l'aide du module **win\_service**, démarrez le service de serveur Web **W3Svc** sur le serveur **win1.example.com**.

- Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

**chapitre 2 |** Exécution de commandes simples d'automatisation

Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **win\_service** dans la liste **MODULE**. Dans le champ **ARGUMENTS**, fournissez les arguments **name=W3Svc state=started**, puis cliquez sur **LAUNCH**.



- 5.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement et qu'Ansible a démarré le service.

The screenshot shows the execution details and the module output. On the left, the 'DETAILS' panel shows the status as 'Successful', started at 8/26/2019 9:28:23 AM and finished at 8/26/2019 9:28:30 AM. The 'win\_service' module was run by 'student' using 'Default inventory' and credential 'DevOps'. The 'win\_service' module output is displayed on the right, showing the configuration and dependencies of the service.

```

SSH password:
win1.example.com | SUCCESS => {
    "can_pause_and_continue": false,
    "changed": false,
    "depended_by": [],
    "dependencies": [
        "WAS",
        "HTTP"
    ],
    "description": "Provides Web connectivity and administration through the Internet Information Services Manager",
    "desktop_interact": false,
    "display_name": "World Wide Web Publishing Service",
    "exists": true,
}

```

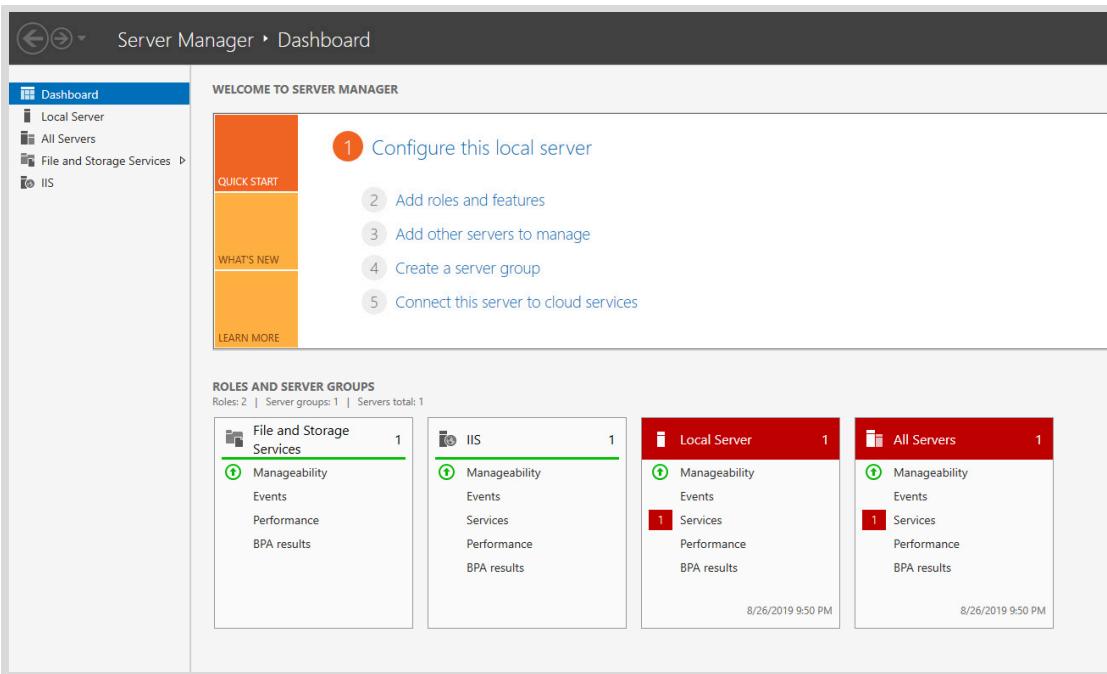
6. Connectez-vous au serveur **win1.example.com** pour vérifier l'installation du service Web, et qu'il est en cours d'exécution.

Après avoir vérifié qu'Ansible a correctement installé la fonction de serveur Web et démarré le service, déconnectez-vous de **win1.example.com**.

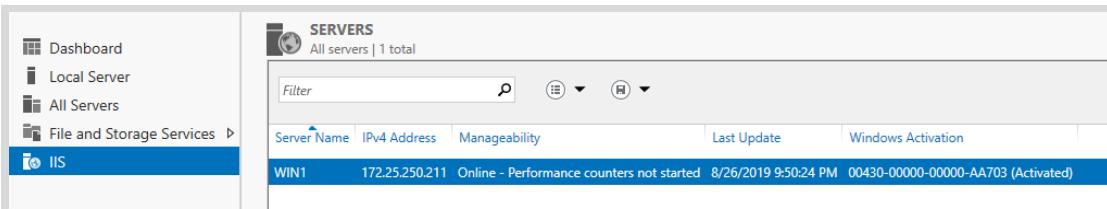
- 6.1. Lancez un client RDP sur le système **workstation.example.com** pour vous connecter au serveur **win1.example.com**. Utilisez le domaine **example**, le nom d'utilisateur **student** et le mot de passe **RedHat123@!**.

## chapitre 2 | Exécution de commandes simples d'automatisation

Une fois que vous vous êtes connecté au serveur **win1.example.com**, ouvrez le **Gestionnaire de serveur** en le sélectionnant dans le menu **Démarrer**.



- 6.2. Sélectionnez **IIS** dans le menu pour vérifier que le service Web est installé et en cours d'exécution.

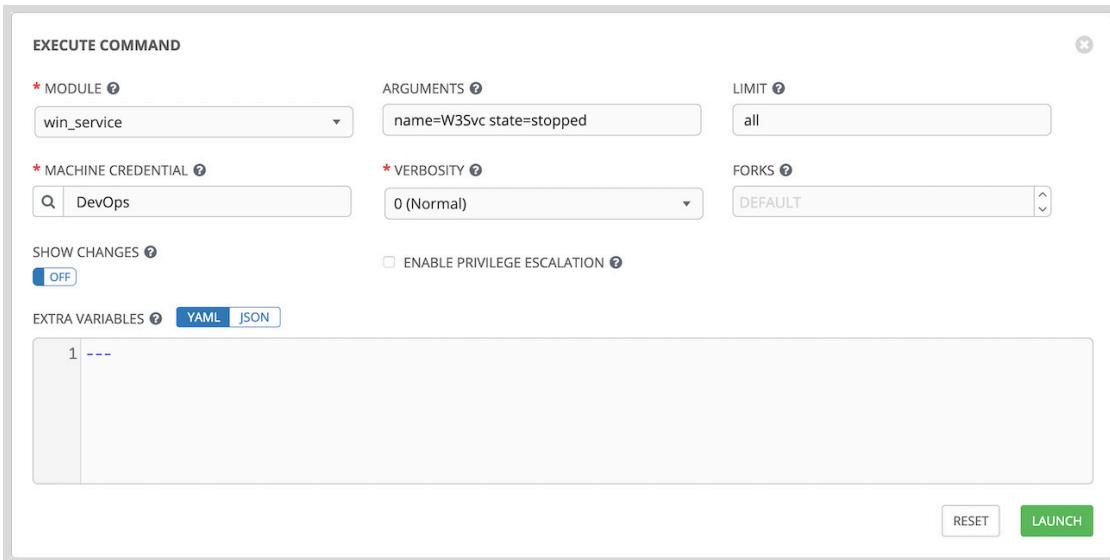


7. Utilisez des commandes ad hoc supplémentaires Ansible pour rétablir les modifications apportées à l'environnement. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

À l'aide du module **win\_service**, arrêtez le serveur Web **W3Svc** sur le serveur **win1.example.com**.

- 7.1. Revenez à la fenêtre **EXECUTE COMMANDS** pour le serveur **win1.example.com**.

Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **win\_service** dans la liste **MODULE**. Dans le champ **ARGUMENTS**, fournissez les arguments **name=W3Svc state=stopped**, puis cliquez sur **LAUNCH**.



7.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement et qu'Ansible a arrêté le service.

## win\_service

ELAPSED 00:00:06  

---

SEARCH KEY 

```
1 SSH password:  
2 win1.example.com | CHANGED => {  
3     "can_pause_and_continue": false,  
4     "changed": true,  
5     "depended_by": [],  
6     "dependencies": [  
7         "WAS",  
8         "HTTP"  
9     ],  
10    "description": "Provides Web connectivity and  
11    administration through the Internet Information Se  
12    rvices Manager",  
13    "desktop_interact": false,  
14    "display_name": "World Wide Web Publishing  
15    Service",  
16    "exists": true,
```

8. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

À l'aide du module **win\_feature**, désinstallez la fonction de serveur Web sur le serveur **win1.example.com**.

- 8.1. Revenez à la fenêtre **EXECUTE COMMAND** pour le serveur **win1.example.com**.

Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **win\_feature** dans la liste **MODULE**. Dans le champ **ARGUMENTS**, fournissez les arguments **name=Web-Server state=absent**, puis cliquez sur **LAUNCH**.

The screenshot shows the 'EXECUTE COMMAND' window. In the 'MODULE' field, 'win\_feature' is selected. The 'ARGUMENTS' field contains 'name=Web-Server state=absent'. The 'LIMIT' field is set to 'win1.example.com'. Under 'MACHINE CREDENTIAL', 'DevOps' is chosen. 'VERBOSITY' is set to '0 (Normal)'. 'FORKS' is set to 'DEFAULT'. The 'SHOW CHANGES' switch is set to 'OFF'. The 'ENABLE PRIVILEGE ESCALATION' checkbox is unchecked. Below the main fields is an 'EXTRA VARIABLES' section with a table containing one row (ID 1). At the bottom right are 'RESET' and 'LAUNCH' buttons.

- 8.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement et qu'Ansible a supprimé la fonction.

## win\_feature

ELAPSED 00:00:34  

---

SEARCH KEY

```
1 SSH password:  
2 win1.example.com | CHANGED => {  
3     "changed": true,  
4     "exitcode": "SuccessRestartRequired",  
5     "feature_result": [  
6         {  
7             "display_name": "Common HTTP Features",  
8             "id": 141,  
9             "message": [],  
10            "reboot_required": true,  
11            "restart_needed": true,  
12            "skip_reason": "NotSkipped",  
13            "success": true  
14        },  
15    ]
```

9. À l'aide du module `win_reboot`, redémarrez le serveur `win1.example.com`.

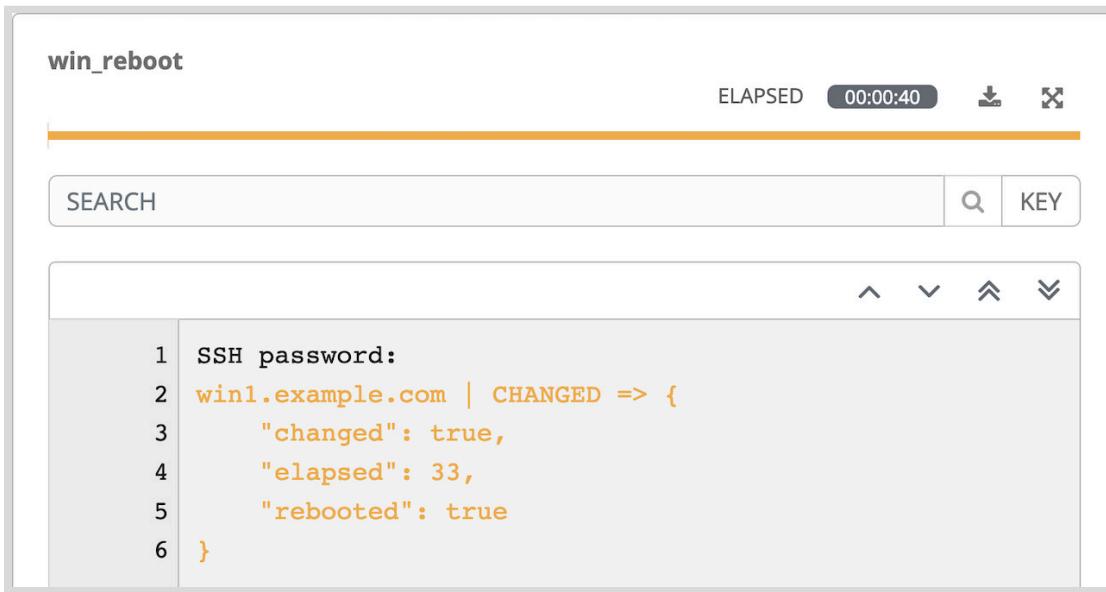
- 9.1. Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **win\_reboot** dans la liste **MODULE**, puis cliquez sur **LAUNCH**.

**EXECUTE COMMAND**

* MODULE <a href="#">?</a>	ARGUMENTS <a href="#">?</a>	LIMIT <a href="#">?</a>
win_reboot		win1.example.com
* MACHINE CREDENTIAL <a href="#">?</a>	* VERBOSITY <a href="#">?</a>	FORKS <a href="#">?</a>
DevOps	0 (Normal)	DEFAULT
SHOW CHANGES <a href="#">?</a>		
<input checked="" type="checkbox"/> OFF		
<input type="checkbox"/> ENABLE PRIVILEGE ESCALATION <a href="#">?</a>		
EXTRA VARIABLES <a href="#">?</a>		
<span style="border: 1px solid #ccc; padding: 2px;">YAML</span> <span style="border: 1px solid #ccc; padding: 2px;">JSON</span>		
<pre>1 ---</pre>		
<input style="border: 1px solid #ccc; padding: 2px; margin-right: 10px;" type="button" value="RESET"/> <input style="background-color: #00aaff; color: white; border: 1px solid #00aaff; padding: 2px;" type="button" value="LAUNCH"/>		

**chapitre 2 |** Exécution de commandes simples d'automatisation

- 9.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement et qu'Ansible a redémarré le serveur.

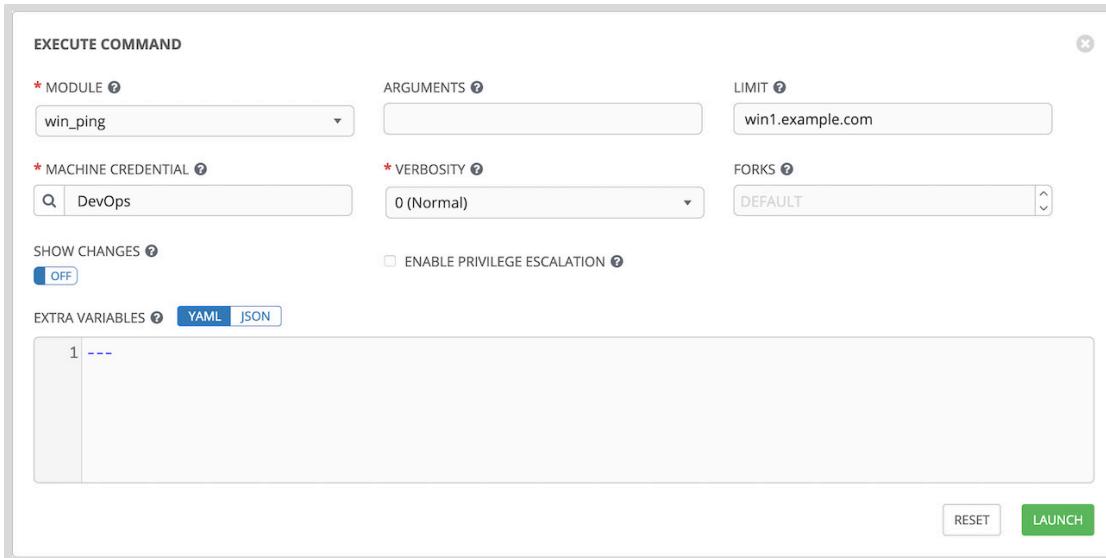


The screenshot shows the Ansible Tower interface with the title "win\_reboot". At the top right, it says "ELAPSED 00:00:40" with download and delete icons. Below is a search bar with a magnifying glass icon and a "KEY" button. The main area displays the command output:

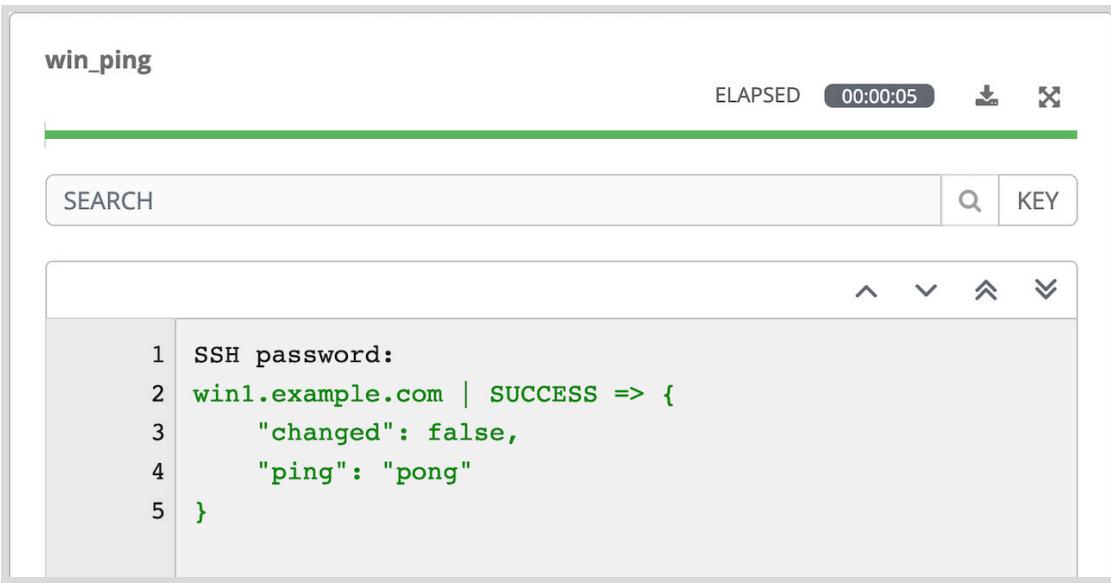
```
1 SSH password:  
2 win1.example.com | CHANGED => {  
3     "changed": true,  
4     "elapsed": 33,  
5     "rebooted": true  
6 }
```

10. À l'aide du module **win\_ping**, vérifiez que le serveur **win1.example.com** est en cours d'exécution et accessible par Ansible.

- 10.1. Dans la fenêtre **EXECUTE COMMAND**, sélectionnez le module **win\_ping** dans la liste **MODULE**, puis cliquez sur **LAUNCH**.



- 10.2. Examinez la sortie pour vérifier que le module s'est exécuté correctement, que le système s'exécute après le redémarrage et qu'Ansible peut correctement communiquer avec celui-ci.



The screenshot shows a terminal window titled "win\_ping". At the top right, there are buttons for "ELAPSED" (00:00:05), download, and close. Below the title is a search bar with a magnifying glass icon and a "KEY" button. The main area contains the following text:

```
1 SSH password:  
2 win1.example.com | SUCCESS => {  
3     "changed": false,  
4     "ping": "pong"  
5 }
```

Navigation arrows are located at the top right of the code editor.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Ansible communique avec les hôtes Windows à l'aide de Windows Remote Management (WinRM), qui est activé par défaut depuis Windows Server 2012.
- Vous pouvez définir les paramètres de connexion de vos hôtes Windows dans les variables d'inventaire au niveau de l'inventaire, pour des groupes spécifiques ou hôte par hôte.
- Les informations d'identification d'Ansible Tower vous permettent de stocker les données d'authentification sensibles, telles que les mots de passe ou les informations d'authentification utilisés pour accéder aux référentiels de code source.
- Les commandes ad hoc vous permettent d'exécuter un module Ansible en tant que tâche ponctuelle sur un hôte géré, ou sur une sélection d'hôtes dans un inventaire.

## chapitre 3

# Mise en œuvre de playbooks Ansible

### Objectif

Rédiger un simple playbook pour automatiser les tâches sur plusieurs hôtes basés sur Microsoft Windows, puis utiliser Red Hat Ansible pour l'exécuter.

### Résultats

- Rédiger un playbook de base et le stocker dans un référentiel Git.
- Configurer un modèle de tâche pour un playbook dans Red Hat Ansible Tower, puis l'utiliser pour exécuter le playbook sur les hôtes gérés.
- Rédiger un playbook qui inclut plusieurs plays, puis utiliser l'augmentation des priviléges par play pour effectuer des tâches en tant qu'utilisateurs particuliers.

### Sections

- Écriture de playbooks (et exercice guidé)
- Exécution de playbooks dans Red Hat Ansible Tower (et exercice guidé)
- Mise en œuvre de plusieurs plays (et exercice guidé)

### Atelier

Mise en œuvre de playbooks Ansible

# Écriture de playbooks

---

## Résultats

Au terme de cette section, vous devez pouvoir écrire un playbook Ansible de base et le stocker dans un référentiel Git.

## Playbooks Ansible et commandes ad hoc

Les commandes ad hoc peuvent exécuter une tâche simple unique sur un ensemble d'hôtes ciblés en tant que commande ponctuelle. Toutefois, la puissance réelle d'Ansible réside dans l'utilisation des playbooks pour exécuter plusieurs tâches complexes sur un ensemble d'hôtes ciblés d'une manière facilement reproductible.

Un *play* est un ensemble ordonné de tâches exécutées sur des hôtes choisis dans votre inventaire. Un *playbook* est un fichier texte contenant une liste d'un ou plusieurs plays à exécuter dans l'ordre.

Les plays vous permettent de transformer un ensemble complexe et fastidieux de tâches administratives manuelles en opérations de routine facilement reproductibles, dont les résultats sont positifs et prévisibles. Dans un playbook, vous pouvez enregistrer la séquence de tâches d'un play sous une forme lisible par l'homme et immédiatement exécutable. Les playbooks peuvent être créées aux formats YAML ou JSON. En raison de la façon dont elles sont écrites, les tâches proprement dites décrivent les étapes nécessaires pour déployer votre application ou infrastructure.

## Formatage d'un playbook Ansible

Pour mieux comprendre comment écrire un playbook Ansible simple, réfléchissez à la manière dont vous exécutez des commandes ad hoc dans Red Hat Ansible Tower. Figure 3.1 illustre un utilisateur d'Ansible Tower lançant une commande ad hoc qui exécute le module **win\_service** avec les arguments **name=spooler state=started** sur l'hôte **win1.example.com**.

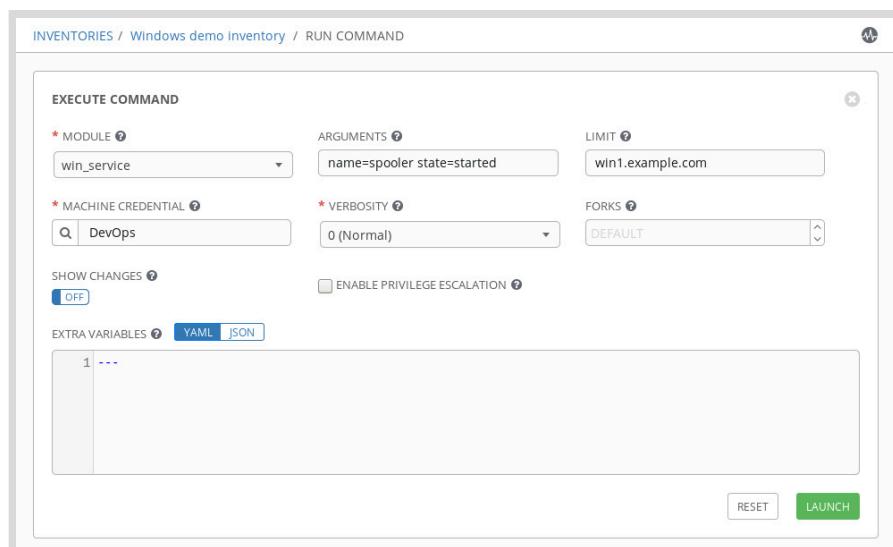


Figure 3.1: Exécution d'un exemple de commande ad hoc dans Ansible Tower

### chapitre 3 | Mise en œuvre de playbooks Ansible

Toute commande ad hoc peut être réécrite sous la forme d'un seul play de tâche, puis enregistrée dans un playbook pour une utilisation et une réutilisation ultérieure.

Un playbook est un fichier texte écrit au format YAML, et est généralement enregistré au moyen de l'extension de fichier standard **.yml**. Le playbook utilise la mise en retrait avec des espaces pour indiquer la structure de ses données. YAML n'énonce pas d'exigences strictes sur le nombre d'espaces utilisés pour la mise en retrait, mais pose deux règles de base :

- Les éléments de données situés au même niveau hiérarchique (tels que les éléments d'une même liste) doivent adopter la même mise en retrait.
- La mise en retrait des enfants d'un élément doit être plus importante que celle de leurs parents.

La commande ad hoc de Figure 3.1 peut être convertie en playbook Ansible, comme illustré ci-dessous :

```
---
```

```
- name: Converted ad hoc command example ①
  hosts: win1.example.com ②
  tasks: ③
    - name: Make sure the spooler service is started ④
      win_service:
        name: spooler ⑤
        state: started ⑥
```

- ① Un nom pour le play afin d'aider à documenter son rôle prévu.
- ② Les hôtes de l'inventaire sur lequel les tâches seront exécutées.
- ③ Le début d'une liste des tâches du play. Il y a une tâche dans ce play.
- ④ Effacez les noms descriptifs de chaque tâche décrivant la fonction de cette tâche. Comme indiqué dans le nom, cette tâche s'assure que le service de spouleur est correctement configuré sur chaque hôte.
- ⑤ Chaque tâche utilise un module Ansible pour effectuer le travail, en l'occurrence **win\_service**.
- ⑥ Cette ligne est un argument pour le module **win\_service**.
- ⑦ Cette ligne est mise en retrait du même montant que la ligne précédente, c'est donc également un argument pour le module.

Vous pouvez également ajouter des lignes vierges afin d'améliorer la lisibilité.



#### Important

Seuls les espaces peuvent être utilisés pour la mise en retrait ; les tabulations ne sont pas autorisées. Plus précisément, une mise en retrait de deux espaces est standard dans les fichiers YAML écrits pour Ansible.

Dans ce cours, vous utilisez Visual Studio Code comme éditeur de texte. Cependant, il existe de nombreuses autres options à prendre en compte lorsque vous travaillez avec Ansible dans un environnement de production. Visual Studio Code présente des extensions qui autorisent les personnalisations pour diverses langues et préférences, notamment l'assistance à la syntaxe pour YAML. Chaque éditeur dispose de paramètres que vous pouvez ajuster pour faciliter la modification de vos playbooks YAML.

### chapitre 3 | Mise en œuvre de playbooks Ansible

Un playbook commence par une ligne composée de trois tirets (---) en tant que marqueur de début de document. Il peut se terminer par trois points (...) comme un marqueur de fin de document, bien que, dans la pratique, les marqueurs sont souvent omis.

Entre ces marqueurs, le playbook est défini comme une liste de plays. Un élément d'une liste YAML commence avec un seul tiret suivi d'un espace. Par exemple, une liste YAML peut ressembler à ce qui suit :

```
- apple  
- orange  
- grape
```

Dans l'exemple de playbook, la ligne après --- commence par un tiret et marque le début du premier (et unique) play de la liste de plays.

Le play en soi est une collection de paires clé-valeur. Les clés du même play doivent présenter la même mise en retrait. L'exemple suivant montre un extrait de code YAML avec trois clés. Les deux premières clés présentent des valeurs simples. La troisième clé est composée d'une liste de trois éléments en guise de valeur.

```
- name: just an example  
  hosts: webservers  
  tasks:  
    - first  
    - second  
    - third
```

L'exemple de play original a trois clés, **name**, **hosts** et **tasks** ; ces clés présentent toutes la même mise en retrait.

La première ligne de l'exemple commence par un tiret et un espace, indiquant que le play est le premier élément d'une liste, suivi de la première clé, qui est l'attribut **name**. La clé **name** associe une chaîne arbitraire au play comme libellé. Cette étiquette identifie l'objectif du play. La clé **name** est facultative, toutefois son utilisation est recommandée, car elle permet de donner des indications sur le playbook. Cet attribut se révèle particulièrement utile lorsqu'un playbook contient plusieurs plays.

```
- name: Converted ad hoc command example
```

La deuxième clé du play est l'attribut **hosts** qui spécifie les hôtes pour lesquels les tâches dans le play sont exécutées. Comme pour l'argument de la commande **ansible**, l'attribut **hosts** prend un modèle hôte comme valeur, tel que les noms des hôtes ou des groupes gérés dans l'inventaire.

```
  hosts: win1.example.com
```

Enfin, la dernière clé du play est l'attribut **tasks**, dont la valeur spécifie une liste de tâches à exécuter pour ce play. Cet exemple a une seule tâche qui exécute le module **win\_service** avec des arguments spécifiques pour s'assurer que le service **Spooler** a bien été démarré.

```
tasks:
  - name: Make sure the spooler service is started
    win_service:
      name: spooler
      state: started
```

L'attribut **tasks** est la partie du play qui liste dans l'ordre les tâches à exécuter sur les hôtes gérés. Chaque tâche de la liste est, en elle-même, une collection de paires clé/valeur.

Dans l'exemple précédent, la seule tâche du play contient deux clés :

- **name** est un libellé facultatif qui décrit l'objet de la tâche. Il est judicieux de nommer toutes vos tâches pour donner des indications sur l'objet de chaque étape du processus d'automatisation. Ce libellé est destiné à être lisible par l'homme et est affiché dans la sortie au cours de l'exécution du play, car Ansible effectue chaque tâche.
- **win\_service** est le module à exécuter pour cette tâche. Les arguments sont transmis en tant que collection de paires clé-valeur qui sont les enfants du module (**name** et **state**).

Voici un autre exemple d'attribut **tasks** avec plusieurs tâches, utilisant le module **win\_service** pour s'assurer que plusieurs services réseau sont configurés pour s'exécuter au démarrage : le module **win\_regedit** pour supprimer une clé de registre et le module **win\_region** pour définir le format de la région sur l'anglais des États-Unis :

```
tasks:
  - name: Ensure that WinRM is started when the system has settled
    win_service:
      name: WinRM
      start_mode: delayed

  - name: Registry path MyClassroom and its entries are absent
    win_regedit:
      path: HKCU:\Software\MyClassroom
      state: absent
      delete_key: yes

  - name: Region format is set to English (United States)
    win_region:
      format: en-US
```



### Important

L'ordre dans lequel les plays et les tâches sont listés dans un playbook est important, car Ansible les exécute dans le même ordre.

Les playbooks que nous vous avons présentés jusqu'alors sont des exemples de base ; des exemples plus complexes de plays et de tâches sont présentés ultérieurement dans ce cours. Des informations sur les différents modules fournis avec Ansible sont disponibles sur le Web à l'adresse [https://docs.ansible.com/ansible/latest/modules/modules\\_by\\_category.html](https://docs.ansible.com/ansible/latest/modules/modules_by_category.html), mais un certain nombre de modules souvent utiles seront présentés dans les prochaines sections.



## Références

### Présentation des playbooks — Documentation Ansible

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_intro.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html)

### Utilisation des playbooks — Documentation Ansible

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks.html)

### Index des modules — Documentation Ansible

[https://docs.ansible.com/ansible/latest/modules/modules\\_by\\_category.html](https://docs.ansible.com/ansible/latest/modules/modules_by_category.html)

## ► Exercice guidé

# Écriture de playbooks

Au cours de cet exercice, vous allez écrire un playbook Ansible de base et le stocker dans un référentiel Git distant.

## Résultats

Vous serez en mesure d'utiliser Visual Studio Code pour écrire un simple playbook Ansible et de le stocker dans un référentiel Git.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

### ► 1. Lancez Visual Studio Code et installez l'extension Red Hat **YAML**.

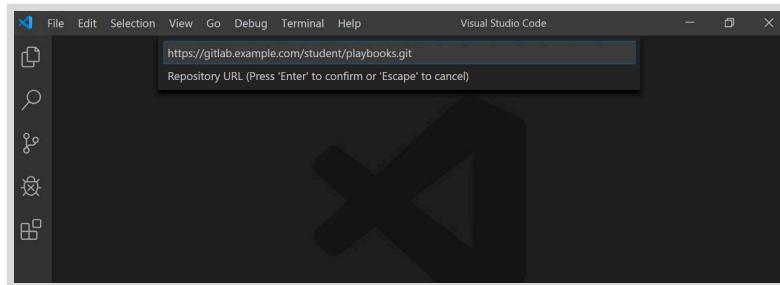
- 1.1. Double-cliquez sur **Visual Studio Code** sur le Bureau.
- 1.2. Accédez à l'affichage **Extensions** et recherchez **yaml** sur le marché des extensions.
- 1.3. Sélectionnez l'extension **YAML** en haut de la liste et cliquez sur **Install**.



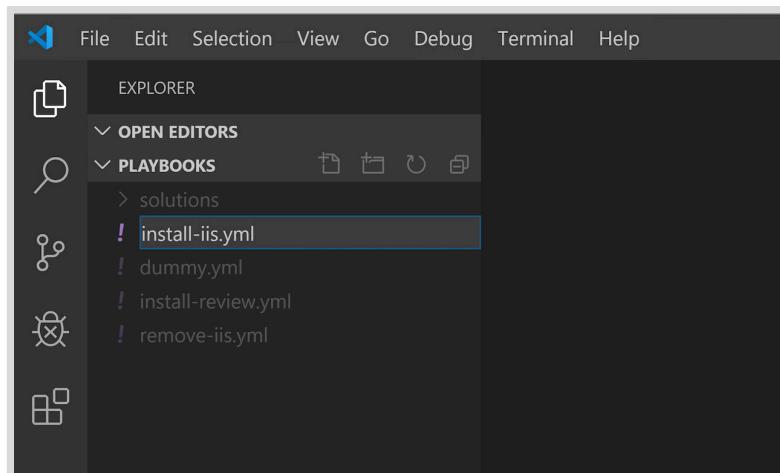
### ► 2. Clonez le référentiel **playbooks** sur votre instance **workstation**.

- 2.1. Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.

- 2.2. Utilisez l'URL de référentiel `https://gitlab.example.com/student/playbooks.git`. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **playbooks**.
- 2.3. Dans la boîte de dialogue qui s'affiche après le clonage, sélectionnez **Open** pour afficher les fichiers.



- 3. Cliquez sur l'icône située en regard de **PLAYBOOKS** pour créer un fichier, puis nommez ce fichier **install-iis.yml**.



- 4. Dans le nouveau fichier, commencez votre playbook à l'aide de trois tirets.

```
---
```

- 5. Ajoutez le nom du playbook et les hôtes qui doivent être ciblés par son exécution.

```
---
```

```
- name: Install the IIS web service
  hosts: win1.example.com
```

- 6. Commencez la section des tâches par une mise en retrait à deux espaces.

```
---
- name: Install the IIS web service
  hosts: win1.example.com

  tasks:
```

- ▶ 7. Ajoutez une tâche pour installer la fonction **Web-Server** sur le serveur cible. Utilisez le module Ansible **win\_feature** pour effectuer cette installation.

```
---
- name: Install the IIS web service
  hosts: win1.example.com

  tasks:

    - name: IIS service installed
      win_feature:
        name: Web-Server
        state: present
```

- ▶ 8. Ajoutez une tâche pour démarrer le service **W3SVC** sur le serveur cible. Utilisez le module Ansible **win\_service** pour effectuer cette tâche de gestion.

```
---
- name: Install the IIS web service
  hosts: win1.example.com

  tasks:

    - name: IIS service installed
      win_feature:
        name: Web-Server
        state: present

    - name: IIS service started
      win_service:
        name: W3Svc
        state: started
```

- ▶ 9. Ajoutez une tâche finale pour placer un fichier **index.html** simple sur le serveur cible contenant la phrase « Hello World! ». Utilisez le module Ansible **win\_copy** pour effectuer cet ajout de contenu.

```
---
- name: Install the IIS web service
  hosts: win1.example.com

  tasks:

    - name: IIS service installed
      win_feature:
```

```
name: Web-Server
state: present

- name: IIS service started
  win_service:
    name: W3Svc
    state: started

- name: Website index.html created
  win_copy:
    content: "Hello World!"
    dest: C:\Inetpub\wwwroot\index.html
```



### Note

Le fichier terminé est disponible dans le fichier **write.sol** dans le répertoire **solutions**.

- ▶ **10.** Cliquez sur **Save**. Notez les icônes de statut situées en bas. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été apportées.
- ▶ **11.** Accédez à la vue **Source Control**, puis cliquez sur **+** en regard du fichier **install-iis.yml** pour indexer les modifications.
- ▶ **12.** Saisissez un message de validation qui décrit le travail que vous avez effectué, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- ▶ **13.** Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.  
À l'invite, cliquez sur **Yes** ou **Ask Me Later**.

L'exercice guidé est maintenant terminé.

# Exécution de playbooks dans Red Hat Ansible Tower

## Résultats

Au terme de cette section, vous devez pouvoir configurer un modèle de tâche pour un playbook dans Red Hat Ansible Tower, et l'utiliser pour exécuter le playbook sur les hôtes gérés.

## Préparation d'Ansible Tower et exécution de playbooks

Red Hat Ansible Tower offre un moyen simple de gérer et d'exécuter vos playbooks Ansible à partir d'une interface Web centrale. En tant qu'administrateur du serveur Ansible Tower, ou en tant qu'utilisateur Ansible Tower configuré avec les permissions appropriées (appelées *rôles*), vous pouvez exécuter des playbooks, voir les résultats des exécutions précédentes et mettre à jour automatiquement un playbook à partir de votre système de contrôle de version.

Avant de pouvoir exécuter un playbook dans Ansible Tower, vous devez d'abord définir un *modèle de tâche* spécifiant l'emplacement où Ansible Tower peut trouver le playbook, l'inventaire et les informations d'identification à utiliser, ainsi que tout paramètre spécial qui contrôle la manière dont vous souhaitez que le playbook soit exécuté. Pour créer un modèle de tâche, vous devez effectuer les étapes suivantes, qui seront abordées plus en détail plus loin dans cette section :

- Créez des *informations d'identification SCM* qu'Ansible Tower peut utiliser pour s'authentifier auprès de votre système de contrôle de version.
- Créez un *projet* qui spécifie l'emplacement du référentiel de playbook dans votre système de contrôle de version, les informations d'identification permettant d'accéder au système de contrôle de version, ainsi que la façon de mettre à jour la copie du référentiel utilisée par Ansible Tower.
- Créez un *modèle de tâche* qui spécifie la manière dont vous souhaitez que le playbook soit exécuté, et que le playbook puisse être réutilisé à multiples reprises de la même façon.

En général, vous n'effectuez ces étapes qu'une seule fois. Ensuite, vous « lancez » le modèle de tâche pour exécuter votre playbook, et les résultats sont indiqués dans l'interface Ansible Tower. Chaque exécution du playbook est enregistrée en tant que *tâche*.

Le reste de cette section examine ces étapes en détail et explique comment passer en revue la sortie de votre exécution de playbook.

**Note**

Certaines des instructions mentionnent le rôle (les autorisations) dont un utilisateur a besoin pour effectuer des étapes spécifiques. L'utilisateur **admin** dispose de toutes les permissions et peut effectuer toutes les étapes. Les autres utilisateurs créés par l'administrateur peuvent disposer de moins de permissions.

La plupart des installations Ansible Tower ont une seule *organisation* configurée pour organiser les utilisateurs, les inventaires, les playbooks, etc. Les installations Ansible Tower peuvent avoir plusieurs organisations définies. Cependant, même si vous n'avez qu'une seule organisation configurée, vous devrez peut-être la spécifier dans l'interface utilisateur au cours de l'installation.

## Création d'informations d'identification SCM

Dans une section précédente de ce cours, vous avez appris à utiliser des informations d'identification de machine, qui stockent les informations d'authentification que les playbooks utilisent pour se connecter aux hôtes gérés et effectuer des tâches d'authentification sur ces derniers. Les *informations d'identification du contrôle de code source*, également appelées informations d'identification SCM, stockent les informations d'authentification utilisées par Ansible Tower pour accéder aux supports de projet stockés dans un système de contrôle de version tel que Git. Les informations d'identification SCM stockent le nom d'utilisateur et le mot de passe ou la clé privée (ainsi que la phrase de passe de la clé privée, le cas échéant) nécessaires pour authentifier l'accès au référentiel de contrôle de code source.

Voici un aperçu de la procédure utilisée pour créer des informations d'identification SCM, afin qu'Ansible Tower puisse récupérer des playbooks, des rôles ou d'autres supports auprès du référentiel Git.

1. Connectez-vous en tant qu'utilisateur ayant l'attribution de rôle appropriée :
    - a. Si vous créez des informations d'identification SCM privées, aucune rôle spécifique n'est exigé.
    - b. Si vous créez des informations d'identification SCM appartenant à une organisation, connectez-vous en tant qu'utilisateur avec le rôle **Admin** de l'organisation.
  2. Cliquez sur **Credentials** pour accéder à l'interface de gestion des informations d'identification.
- Dans le volet **CREDENTIAL**, cliquez sur **+** pour créer de nouvelles informations d'identification.
3. Dans le volet **CREATE CREDENTIAL**, saisissez les informations requises pour les nouvelles informations d'identification.
    - a. Saisissez un nom unique pour les informations d'identification.

Si vous créez les informations d'identification d'une organisation, cliquez sur la loupe située en regard du champ **ORGANIZATION** et sélectionnez l'organisation. Ignorez cette étape si vous créez des informations d'identification privées.

    - b. Dans la liste **CREDENTIAL TYPE**, sélectionnez **Source Control**.
  4. Après avoir sélectionné le type d'informations d'identification **Source Control**, les champs appropriés s'affichent dans la section **TYPE DETAILS**.

Saisissez les données d'authentification dans leurs champs respectifs. Par exemple, vous pouvez avoir besoin de spécifier un nom d'utilisateur. Si un mot de passe est nécessaire, vous devez le saisir dans le champ de mot de passe. Si vous utilisez une clé privée SSH pour vous authentifier, copiez et collez ou glissez et déposez votre clé privée dans le champ **SCM PRIVATE KEY**. Cette clé peut être une clé privée SSH chiffrée par phrase de passe, auquel cas vous pouvez fournir la phrase de passe à Ansible Tower dans le champ **PRIVATE KEY PASSPHRASE**.

The screenshot shows the 'Create New Credential' form in Ansible Tower. The 'NAME' field is filled with 'testing-repo'. The 'DESCRIPTION' field contains 'Credential for the testing respository'. The 'ORGANIZATION' field is set to 'Default'. The 'CREDENTIAL TYPE' is selected as 'Source Control'. In the 'TYPE DETAILS' section, the 'USERNAME' is 'testing' and the 'PASSWORD' is masked. There is a note: 'SCM PRIVATE KEY HINT: Drag and drop private file on the field below.' Below this is a large empty box for dragging files. The 'PRIVATE KEY PASSPHRASE' field is also masked. At the bottom right are 'CANCEL' and 'SAVE' buttons.

Figure 3.5: Création de nouvelles informations d'identification SCM dans Ansible Tower

## Création d'un projet

Dans Ansible Tower, la ressource *project* représente un référentiel disponible dans un système de contrôle de version (également désigné par Ansible Tower en tant que système de *gestion de contrôle de source* ou *SCM*), et a au moins un playbook et ses ressources associées, tels que des fichiers et des modèles. Dans ce cours, vous utilisez Git pour gérer ces fichiers. La conception d'Ansible Tower suppose que la plupart des projets Ansible figurent dans un système de contrôle de version, et qu'il peut automatiquement récupérer des supports mis à jour pour un projet à partir de plusieurs systèmes de contrôle de version couramment utilisés.

Ansible Tower permet de télécharger et récupérer automatiquement les mises à jour des supports de projet à partir de systèmes SCM, à l'aide de Git, Subversion ou Mercurial.

La procédure suivante montre comment créer un projet dans Ansible Tower qui se connecte à un référentiel Git existant, qui contient des playbooks Ansible.

1. Connectez-vous à l'interface web d'Ansible Tower en tant qu'utilisateur avec le rôle **Admin**.
2. Sélectionnez **Projects** pour accéder à la liste des projets. Cliquez sur **+** pour créer le projet.
3. Saisissez un nom unique pour votre projet.

Si vous le souhaitez, vous pouvez saisir une description pour votre projet.

### chapitre 3 | Mise en œuvre de playbooks Ansible

4. Pour affecter le projet à une organisation spécifique, cliquez sur l'icône de loupe correspondant au champ **ORGANIZATION**, puis sélectionnez une organisation.
  5. Dans la liste **SCM TYPE**, sélectionnez l'élément **Git** pour ordonner à Ansible Tower de récupérer un référentiel Git.
  6. Saisissez l'emplacement du référentiel Git dans le champ **SCM URL**.
- Si vous le souhaitez, dans le champ **SCM BRANCH/TAG/COMMIT**, spécifiez la branche, la balise ou la validation du référentiel auprès duquel obtenir le contenu.
- Si le serveur référentiel nécessite une authentification, sélectionnez de nouvelles informations d'identification dans le champ **SCM CREDENTIAL**.
7. Enfin, configurez la manière dont le projet obtient les mises à jour du système de contrôle de version (SCM). Les paramètres disponibles sont **Clean**, **Delete on Update** et **Update Revision on Launch**.

**Figure 3.6: Création de nouveaux projets dans Ansible Tower**

#### Clean

Cette option de mise à jour SCM supprime les modifications locales apportées aux supports du projet sur Ansible Tower avant d'obtenir la dernière révision auprès du référentiel de contrôle de source.

#### Delete on Update

Cette option SCM supprime complètement le référentiel de projet local sur Ansible Tower avant d'extraire la dernière révision auprès du référentiel de contrôle de source. Pour les référentiels volumineux, cette opération prend plus de temps que l'option **Clean**.

#### Update on Launch

Cette option SCM met à jour automatiquement le projet à partir du référentiel de contrôle de source chaque fois que vous utilisez le projet pour lancer une tâche. Ansible Tower effectue le suivi de cette mise à jour sous la forme d'une tâche distincte. Si cette option n'est pas sélectionnée, vous devez mettre à jour le projet manuellement.

Si vous ne souhaitez pas utiliser ces paramètres automatiques, vous pouvez mettre à jour manuellement un projet vers la dernière version du référentiel de contrôle de source. La procédure suivante décrit les étapes nécessaires pour la mise à jour manuelle d'un projet à partir d'une source SCM.

1. Connectez-vous en tant qu'utilisateur avec le rôle **Update** pour le projet.

2. Cliquez sur **Projects** pour accéder à l'interface de gestion de projet.
3. Une icône de synchronisation s'affiche sous la colonne **ACTIONS** si l'utilisateur a le rôle **Update** pour un projet donné. Cliquez sur l'icône pour déclencher une mise à jour immédiate d'un projet.

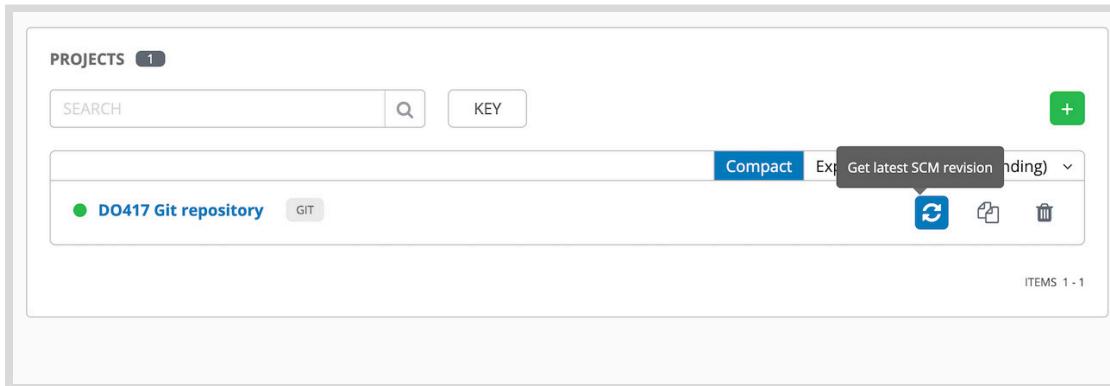


Figure 3.7: Mise à jour d'un projet SCM

## Création d'un modèle de tâche

Un *modèle de tâche* est utilisé pour exécuter votre playbook. Il associe le projet contenant votre playbook à un inventaire, aux informations d'identification de la machine permettant de s'authentifier auprès des hôtes de votre inventaire, ainsi qu'aux paramètres qui contrôlent le fonctionnement de votre playbook.

Une fois que vous avez créé un modèle de tâche dans Ansible Tower, vous pouvez le réutiliser pour exécuter à nouveau le playbook, aussi souvent que nécessaire. Cela vous permet de relancer le playbook pour corriger toute mauvaise configuration ou modification inattendue qui a été apportée à vos serveurs, ou pour préparer des serveurs supplémentaires qui ont été ajoutés à l'inventaire. Par conséquent, vous ne configurez normalement le modèle de tâche qu'une seule fois, mais vous l'exécutez chaque fois que vous mettez à jour votre playbook, ou périodiquement pour vous assurer que la configuration d'hôte est prévue. Le bouton, ou icône, **Launch** d'un modèle de tâche est utilisé pour exécuter son playbook.

La procédure suivante détaille la façon de créer un modèle de tâche pour l'exécution d'un playbook :

1. Connectez-vous à l'interface Web d'Ansible Tower en tant qu'utilisateur auquel est attribué le rôle **Use** pour un inventaire, un projet et les informations d'identification de machine.
2. Cliquez sur **Templates** pour accéder à l'interface de gestion des modèles.  
Cliquez sur **+**, puis sélectionnez **Job Template** pour créer un nouveau modèle de tâche.
3. Saisissez un nom pour le modèle de tâche, puis sélectionnez **Run** comme type de tâche.
4. Cliquez sur l'icône de loupe correspondant au champ **INVENTORY**, puis sélectionnez l'inventaire souhaité pour spécifier les hôtes gérés sur lesquels la tâche doit être exécutée.
5. Cliquez sur l'icône de loupe correspondant au champ **PROJECT**, puis sélectionnez le projet contenant le playbook que la tâche va exécuter.
6. Spécifiez le playbook que la tâche va exécuter en sélectionnant un playbook dans la liste **PLAYBOOK**. La liste répertorie tous les playbooks existants dans le projet.
7. Spécifiez les informations d'identification à utiliser pour l'authentification auprès des hôtes gérés.

### chapitre 3 | Mise en œuvre de playbooks Ansible

- a. Cliquez sur l'icône de loupe correspondant au champ **CREDENTIAL**.
- b. Sélectionnez les informations d'identification utilisées par le modèle de tâche.
8. Sélectionnez un paramètre de votre choix dans la liste **VERBOSITY**. Cette sélection détermine le niveau de détail généré dans la sortie de l'exécution de la tâche.

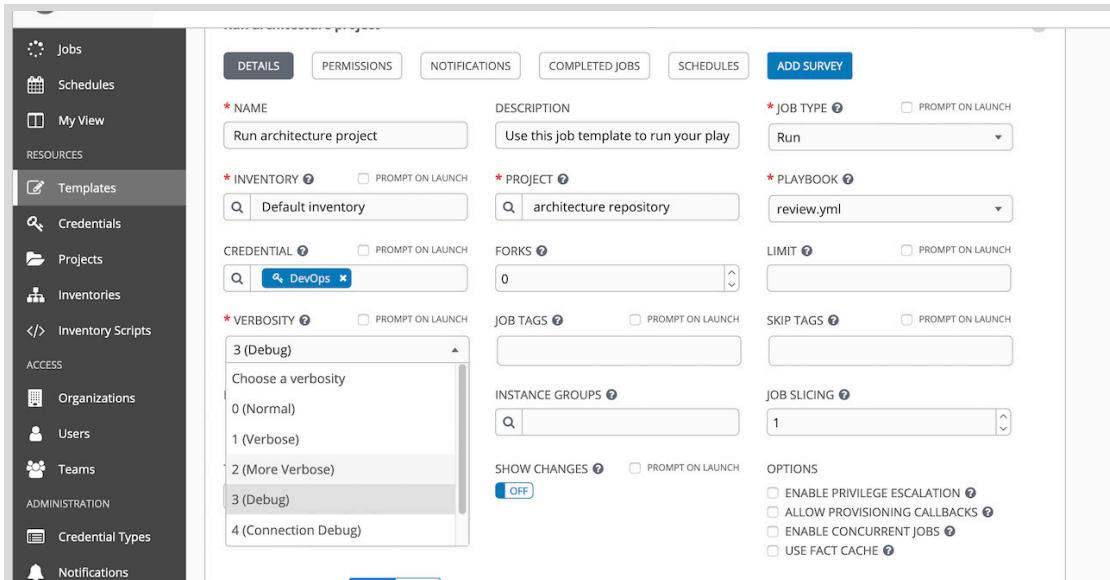


Figure 3.8: Création de modèles de tâche dans Ansible Tower



#### Note

Pour plus d'informations sur les différents champs, consultez le lien *Modèles de tâche – Documentation Ansible* fourni dans la section des références.

## Demande de paramètres de tâche au lancement

Ansible Tower fournit une certaine flexibilité en permettant d'inviter l'utilisateur à saisir certains paramètres des modèles de tâches au moment de l'exécution de la tâche. Cette option *Prompt on launch* est disponible pour une série de paramètres de tâches.

Cette possibilité de modification des paramètres des tâches au moment de l'exécution de celles-ci favorise la réutilisation des playbooks. Par exemple, plutôt que créer plusieurs modèles de tâches pour exécuter le même playbook sur différents ensembles d'hôtes gérés, créez un modèle de tâche unique et sélectionnez **Prompt on launch** pour le champ d'inventaire. Lors de l'exécution de la tâche, vous pouvez spécifier l'inventaire pour le play. Les utilisateurs ne peuvent sélectionner que les inventaires pour lesquels ils disposent du rôle **Use**.

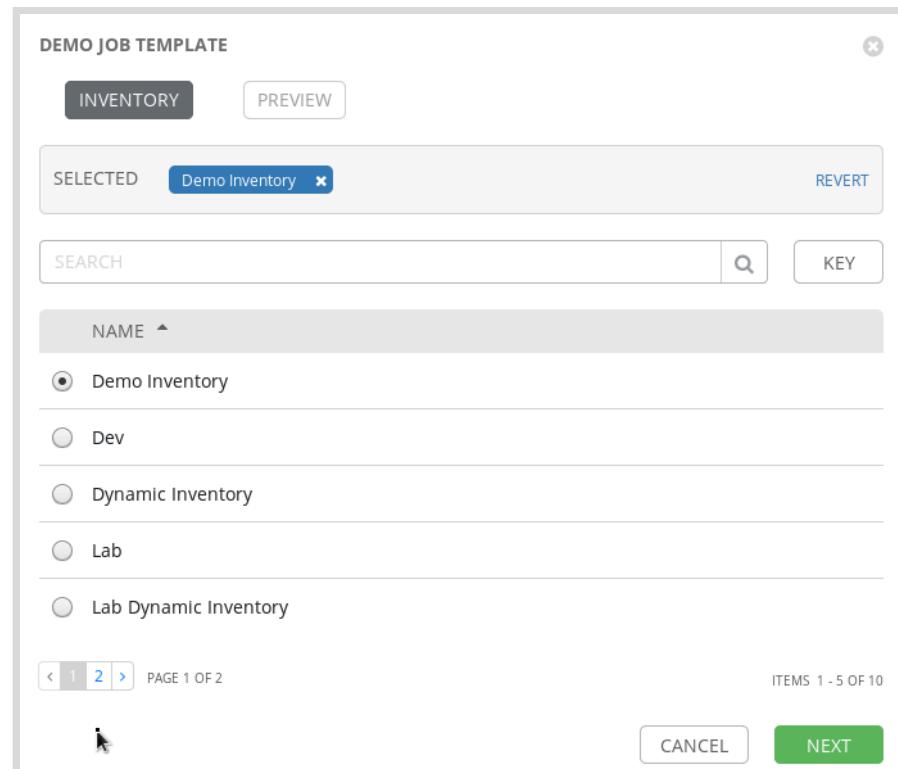


Figure 3.9: Modèle de tâche invitant à fournir un inventaire lors du lancement d'une tâche

## Lancement de tâches

La plupart du temps, votre modèle de tâche et tous ses composants sont déjà configurés. Tout ce que vous effectuerez généralement consiste à exécuter votre playbook en lançant une tâche à partir d'un modèle de tâche existant.

Voici une description de la façon d'utiliser un modèle de tâche existant pour exécuter un playbook :

1. Connectez-vous en tant qu'utilisateur avec le rôle **Execute** pour le modèle de tâche.
2. Cliquez sur **Templates** pour accéder à la liste de modèles.
3. Recherchez le modèle de tâche à exécuter dans la liste des modèles, puis cliquez sur l'icône de fusée sous la colonne **ACTIONS** pour lancer la tâche.
4. Si vous avez activé l'option **Prompt on launch** pour l'un des champs, Ansible Tower vous invite à saisir des informations avant d'exécuter la tâche. Saisissez les entrées du paramètre, puis cliquez sur **LAUNCH** pour lancer la tâche, comme illustré dans Figure 3.10.

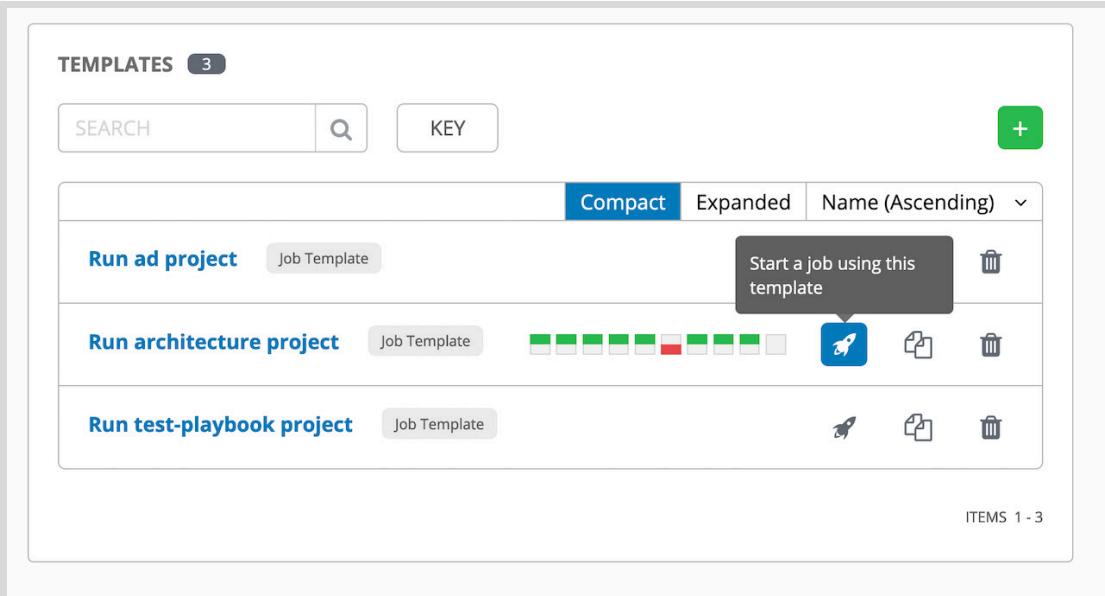


Figure 3.10: Lancement d'une tâche

## Évaluation des résultats de la tâche

Après avoir exécuté une tâche, Ansible Tower vous redirige vers la page de détails de la tâche. Vous pouvez également accéder à la page de détails en sélectionnant **Jobs**, puis en sélectionnant la tâche dont vous souhaitez afficher les détails.

La page des détails de la tâche est divisée en deux volets. Le volet **DETAILS** affiche les détails des paramètres de la tâche, tandis que le volet de sortie de la tâche affiche la sortie du playbook exécuté par la tâche.

La partie supérieure du volet de sortie de la tâche contient une synthèse détaillant le nombre de plays et de tâches exécutés, le nombre d'hôtes qu'Ansible Tower a utilisés pour exécuter la tâche, ainsi que la durée d'exécution de la tâche. De plus, vous pouvez utiliser des commandes pour afficher ce volet en plein écran ou pour télécharger la sortie de l'exécution de la tâche.

Sur le côté gauche de la section de sortie, vous pouvez utiliser les commandes **+** et **-** pour développer ou réduire la sortie de chaque tâche du playbook. Les commandes dans la section de sortie permettent de faire défiler la sortie, ainsi que d'accéder directement au début ou à la fin de la sortie, comme illustré dans Figure 3.11.

```

PLAYS 9 TASKS 276 HOSTS 7 ELAPSED 00:40:37
SEARCH 🔎 KEY 🔍

-
  tasks/mytitle.yml for workstation
  988
  989 TASK [windows-workstation : Install IIS and .Net 4.5 on Server] **** 09:39:22
    changed: [workstation]
  990
  991
  992 TASK [windows-workstation : Install IIS and .Net 4.5 on Non-Server] **** 09:41:05
    skipping: [workstation] => (item=IIS-WebServerRole)
  993 skipping: [workstation] => (item=IIS-WebServer)
  994 skipping: [workstation] => (item=NetFx4Extended-ASPNET45)
  995 skipping: [workstation] => (item=IIS-NetFxExtensibility45)
  996 skipping: [workstation] => (item=IIS-ISAPIExtensions)
  997 skipping: [workstation] => (item=IIS-ISAPIFilter)
  998 skipping: [workstation] => (item=IIS-ASPNET45)
  999 skipping: [workstation] => (item=IIS-ASPNET)
  1000
  1001 TASK [windows-workstation : Create temp directory] **** 09:41:05
    changed: [workstation]
  1002

```

Figure 3.11: Accès aux résultats de l'exécution de la tâche

Pour mieux comprendre la façon d'interpréter la sortie du playbook exécuté dans l'enregistrement de tâche, examinez le playbook suivant :

```

---
- name: Converted ad hoc command example
  hosts: win1.example.com
  tasks:
    - name: Make sure the spooler service is started
      win_service:
        name: spooler
        state: started

```

Une exécution réussie du playbook peut générer la sortie de tâche suivante :

```

PLAY [Converted ad hoc command example] ****

TASK [Gathering Facts] ****
ok: [win1.example.com]

TASK [Make sure the spooler service is started] ****
changed: [win1.example.com]

PLAY RECAP ****
win1.example.com : ok=2    changed=1    unreachable=0    failed=0
                  skipped=0   rescued=0   ignored=0

```

Une ligne est imprimée pour afficher le nom du play et le nom de chaque tâche dans le play. Notez que la tâche « Gathering Facts » est une tâche spéciale qui s'exécute normalement par défaut et utilise le module **setup** automatiquement pour définir certaines variables spécifiques à la machine au début d'un play. Les faits et la collecte de faits seront abordés plus tard dans le cours.

Chaque tâche liste tous les hôtes sur lesquels la tâche a été exécutée, ainsi que le résultat de la tâche :

**chapitre 3 |** Mise en œuvre de playbooks Ansible

- **ok** signifie que la tâche n'a apporté aucune modification à l'hôte.
- **changed** signifie que la tâche a modifié le système, et que le système est maintenant en bon état.
- **fatal** signifie qu'une erreur irrécupérable s'est produite lors de l'exécution de cette tâche sur cet hôte. Ansible va normalement cesser d'essayer d'exécuter le playbook sur cet hôte pour cette tâche. Des informations supplémentaires sur l'erreur fatale sont incluses dans la sortie.
- **skipping** signifie que la tâche du playbook a été ignorée pour cet hôte, probablement en raison d'une exécution conditionnelle, ce qui sera abordé plus tard dans ce cours.

À la fin du play se trouve un récapitulatif du play qui contient une ligne pour chaque hôte du play, ainsi que le nombre de tâches correspondant à certains résultats pour cet hôte. Si une tâche est signalée comme **changed**, elle sera également considérée comme **ok** dans le récapitulatif du play. Outre **ok**, **changed** et **skipped**, il existe un certain nombre d'autres résultats signalés dans le récapitulatif du play :

- **unreachable** est comptabilisé si l'hôte n'a pas pu être contacté sur le réseau ou si une erreur irrécupérable s'est produite.
- **failed** compte les tâches qui ont échoué, arrêtant l'exécution du playbook pour cet hôte.
- **rescued** compte les tâches qui ont échoué, mais qui ont été récupérées via la gestion du bloc rescue, qui est abordée plus tard dans ce cours.
- **ignored** compte les tâches qui ont échoué, mais pour lesquelles l'exécution du playbook a été autorisée à se poursuivre en cas d'échec. Ceci doit être configuré par l'auteur du playbook.

Pour plus d'informations, consultez Détails de la tâche &mdash; Exécution du playbook [<https://docs.ansible.com/ansible-tower/latest/html/userguide/jobs.html#job-details-playbook-run>] dans le *Guide de l'utilisateur Ansible Tower*.

En général, les playbooks Ansible sont idempotents. Vous pouvez donc exécuter le playbook plusieurs fois sans risque. Si l'état des hôtes gérés ciblés est déjà correct, vous ne devez apporter aucune modification. Par exemple, supposons que le playbook de l'exemple précédent est à nouveau exécuté :

```
PLAY [Converted ad hoc command example] ****
TASK [Gathering Facts] ****
ok: [win1.example.com]

TASK [Make sure the spooler service is started] ****
ok: [win1.example.com]

PLAY RECAP ****
win1.example.com : ok=2    changed=0    unreachable=0    failed=0
                  skipped=0   rescued=0   ignored=0
```

Cette fois, toutes les tâches ont réussi avec le statut **ok** et aucune modification n'a été signalée.



## Références

### Guide de l'utilisateur d'Ansible Tower

<http://docs.ansible.com/ansible-tower/latest/html/userguide/index.html>

### Informations d'identification &mdash; Guide d'utilisation d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/userguide/credentials.html>

### Projets &mdash; Guide d'utilisation d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/userguide/projects.html>

### Modèles de tâches &mdash; Guide d'utilisation d'Ansible Tower

[https://docs.ansible.com/ansible-tower/latest/html/userguide/job\\_templates.html](https://docs.ansible.com/ansible-tower/latest/html/userguide/job_templates.html)

## ► Exercice guidé

# Exécution de playbooks dans Red Hat Ansible Tower

Dans cet exercice, vous allez configurer Red Hat Ansible Tower avec un modèle de projet et de tâche pour un playbook existant, et utiliser le modèle de tâche pour exécuter le playbook.

## Résultats

Vous serez en mesure de créer un modèle de projet et de tâche, et de lancer une tâche à partir de l'interface Web d'Ansible Tower.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

Terminez la section appelée « Exercice guidé: Écriture de playbooks » présentée plus haut dans ce chapitre.

- ▶ 1. Cliquez sur le raccourci **Ansible Tower** sur votre Bureau, ou accédez à <https://tower.example.com> pour ouvrir l'interface Web d'Ansible Tower. Connectez-vous en utilisant le compte **admin** et **RedHat123@!** comme mot de passe.
- ▶ 2. Créez de nouvelles informations d'identification de contrôle de source. Le projet que vous créez à une étape ultérieure utilisera ces informations d'identification.
  - 2.1. Cliquez sur **Credentials** pour gérer les informations d'identification.
  - 2.2. Cliquez sur **+** pour ajouter de nouvelles informations d'identification.

NAME	KIND	OWNERS	ACTIONS
DevOps	Machine	Default	
GitLab	Source Control	Default	

- 2.3. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
NAME	Run Practice Git
DESCRIPTION	Student Git credential
ORGANIZATION	Valeur par défaut
TYPE	Source Control
USERNAME	<b>student</b>
PASSWORD	<b>RedHat123@!</b>

Ne renseignez pas les champs **SCM PRIVATE KEY** et **PRIVATE KEY PASSPHRASE**.

- 2.4. Cliquez sur **SAVE** pour créer les informations d'identification.
- 3. Créez un nouveau projet appelé **Run Practice**.
- 3.1. Cliquez sur **Projects**.
  - 3.2. Cliquez sur **+** pour ajouter un nouveau projet.
  - 3.3. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
NAME	Run Practice
DESCRIPTION	Single playbook practice project
ORGANIZATION	Valeur par défaut
SCM TYPE	Git
SCM URL	<a href="https://gitlab.example.com/student/playbooks.git">https://gitlab.example.com/student/playbooks.git</a>
SCM CREDENTIAL	Run Practice Git
SCM UPDATE OPTIONS	Sélectionner <b>CLEAN</b> et <b>UPDATE REVISION ON LAUNCH</b>

The screenshot shows the 'New Project' dialog box in Ansible Tower. The 'DETAILS' tab is active. The project name is 'Run Practice', the description is 'Single playbook practice project', and the organization is 'Default'. The SCM type is set to 'Git'. The SCM URL is 'https://gitlab.example.com/student/playbooks.git'. There is no SCM branch/tag/commit specified. The SCM credential is 'Run Practice Git'. Under 'SCM UPDATE OPTIONS', 'CLEAN' is checked, while 'DELETE ON UPDATE' and 'UPDATE REVISION ON LAUNCH' are unchecked. The cache timeout is set to 0 seconds.

- 3.4. Cliquez sur **SAVE** pour créer le projet. Cela déclenche automatiquement la mise à jour SCM du projet.

Les options **CLEAN** et **UPDATE REVISION ON LAUNCH** s'assurent que la tâche Ansible actualise le playbook à partir de Git avant d'exécuter la tâche. Ansible Tower utilise les valeurs fournies dans les champs **SCM URL** et **SCM CREDENTIAL** pour extraire une copie locale du référentiel.

► 4. Observez la mise à jour SCM automatique du projet **Run Practice**.

- 4.1. Faites défiler la page vers le bas et attendez quelques secondes. Dans la liste des projets, une icône d'état est présente à gauche du projet **Run Practice**. Cette icône est blanche au début, rouge avec un point d'exclamation en cas d'échec et verte en cas de réussite.
- 4.2. Cliquez sur l'icône d'état pour afficher la page d'état détaillé de la tâche de mise à jour SCM. Comme vous pouvez le voir dans la fenêtre **DETAILS**, la tâche de mise à jour SCM s'exécute comme tout autre playbook Ansible.
- 4.3. Vérifiez que le statut de la tâche dans la section **DETAILS** affiche **Successful**.

► 5. Créez un nouveau modèle de tâche appelé **Install IIS**.

- 5.1. Cliquez sur **Templates**.
- 5.2. Cliquez sur **+** pour créer le nouveau modèle de tâche.
- 5.3. Sélectionnez **Job Template** dans la liste.
- 5.4. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
NAME	Installer les services Internet (IIS)
DESCRIPTION	Installe et démarre les services Internet (IIS)
JOB TYPE	Run
INVENTORY	Inventaire par défaut
PROJECT	Run Practice
PLAYBOOK	install-iis.yml
CREDENTIAL	DevOps

The screenshot shows the 'New Job Template' interface. The 'DETAILS' tab is selected. Key configuration parameters include:

- NAME:** Install IIS
- DESCRIPTION:** Installs and starts IIS
- JOB TYPE:** Run
- INVENTORY:** Default inventory
- PROJECT:** Run Practice
- PLAYBOOK:** install-iis.yml
- CREDENTIAL:** DevOps
- FORKS:** 0
- LIMIT:** (empty)
- Skip Tags:** (empty)
- Job Slicing:** 1
- OPTIONS:** Enable Privilege Escalation, Allow Provisioning Callbacks

**Important**  
Un **solutions/install-iis.sol** fichier de solution est disponible dans les **règles** référentiel git.

- 5.5. Ne modifiez pas les autres champs et cliquez sur **SAVE** pour créer le modèle de tâche.
  
- ▶ 6. Lancez une tâche à l'aide du modèle de tâche **Install IIS**.
  - 6.1. Cliquez sur **Templates**.
  - 6.2. Sur la même ligne que le modèle de tâche **Install IIS**, cliquez sur l'icône de fusée située à droite pour lancer la tâche. Cela vous redirige vers une page d'état détaillée de la tâche en cours d'exécution.

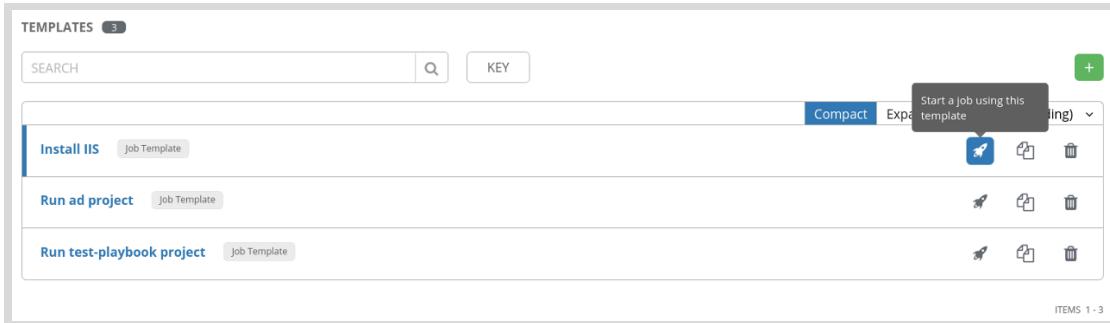


Figure 3.16: Lancement d'une tâche

6.3. Observez la sortie en direct de la tâche en cours d'exécution pendant une minute.

The screenshot shows the Ansible Tower 'JOBS' page for a job named '47 - Install IIS'. On the left, the 'DETAILS' panel shows the job status as 'Successful', started at 8/27/2019 9:53:43 AM and finished at 8/27/2019 9:54:52 AM. The 'LOG' panel on the right shows the execution log for the 'Install IIS' task. The log output includes:

```

PLAY [The IIS web service is started] ****
TASK [Gathering Facts] ****
ok: [win1.example.com]

TASK [IIS is installed] ****
changed: [win1.example.com]

TASK [IIS service is started] ****
changed: [win1.example.com]

TASK [Website index.html is created] ****
changed: [win1.example.com]

PLAY RECAP
*****
win1.example.com : ok=4    changed=3    unreachable=0
failed=0   skipped=0    rescued=0    ignored=0

```

6.4. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.

6.5. Parcourez le journal de sortie pour vous assurer que toutes les tâches ont été correctement exécutées. Le récapitulatif du play situé au bas du journal doit indiquer que trois tâches ont entraîné des modifications à l'hôte **win1.example.com**.

```

SSH password:
PLAY [The IIS web service is started] ****
TASK [Gathering Facts] ****
ok: [win1.example.com]

TASK [IIS is installed] ****
changed: [win1.example.com]

TASK [IIS service is started] ****
changed: [win1.example.com]

TASK [Website index.html is created] ****
changed: [win1.example.com]

```

```
PLAY RECAP ****
win1.example.com : ok=4    changed=3    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

 **Note**

Si le format d'un playbook est incorrect, Ansible affiche l'état FAILED et un point d'exclamation rouge. Le message d'erreur dans la sortie de journal peut vous aider à déboguer le problème en vous indiquant directement la ligne problématique.

Vous pouvez également comparer votre playbook au playbook de la solution **solutions/install-iis.sol** dans le référentiel Git **playbooks** pour terminer cet exercice guidé.

▶ **7.** Vérifiez que le serveur IIS est en cours d'exécution sur **win1.example.com**.

- 7.1. Dans Chrome, ouvrez un nouvel onglet et rendez-vous à l'adresse `http://win1.example.com/`. Le résultat devrait être semblable à ce qui suit :

```
Hello, World!
```

▶ **8.** Affichez les tâches exécutées précédemment.

- 8.1. Revenez à l'interface d'Ansible Tower, puis cliquez sur **Jobs**. Vous verrez une liste des tâches qui ont déjà été exécutées.
- 8.2. Cliquez sur la tâche du haut dans la liste. Il s'agit de la tâche que vous venez d'exécuter.

L'exercice guidé est maintenant terminé.

# Mise en œuvre de plusieurs plays

---

## Résultats

Au terme de cette section, vous serez en mesure de rédiger un playbook qui utilise plusieurs plays, et l'augmentation des priviléges par play, pour effectuer des tâches en tant qu'utilisateurs particuliers.

## Écriture de plusieurs plays

Un playbook est un fichier YAML contenant une liste d'un ou de plusieurs plays. Gardez à l'esprit qu'un seul play est une liste ordonnée de tâches à exécuter sur des hôtes sélectionnés de l'inventaire. Par conséquent, si un playbook contient plusieurs plays, chacun d'eux peut appliquer ses tâches à un ensemble distinct d'hôtes.

Cela peut s'avérer très utile lors de l'orchestration d'un déploiement complexe impliquant des tâches sur différents hôtes. Vous pouvez écrire un playbook pour exécuter un play sur la base d'un ensemble d'hôtes, et une fois cette opération terminée, un autre play traite un autre ensemble d'hôtes.

L'écriture d'un playbook contenant plusieurs plays est très simple. Chaque play du playbook est écrit comme un élément de liste de premier niveau, et chaque élément de liste suit la même syntaxe de play.

L'exemple suivant illustre un playbook simple avec deux plays. Le premier play s'exécute sur les hôtes **web**, et le second play s'exécute sur les hôtes **app**.

```
---
# This is a playbook with two plays

- name: first play ①
  hosts: web
  tasks:
    - name: Web server is installed
      win_feature:
        name: Web-Server
        status: present

- name: second play ②
  hosts: app
  tasks:
    - name: App server is installed
      win_feature:
        name: Application-Server
        status: present
```

- ① Commencez chaque play par **-name :** et sans mise en retrait.
- ② Le deuxième play est de structure identique à la première, et contient les paramètres **name**, **hosts** et **tasks**.

## Utilisateurs distants et augmentation des privilèges dans les plays

Les paramètres définis dans le play peuvent remplacer l'utilisateur Ansible par défaut, en spécifiant un utilisateur personnalisé pour la connexion à distance ou l'exécution de tâches. Ces paramètres doivent être placés au même niveau que les mots-clés **hosts** et **tasks**.

### Attributs d'augmentation des privilèges

Il peut arriver que vous souhaitiez exécuter un play ou des tâches spécifiques dans le play en tant qu'utilisateur différent de celui que vous utilisez pour vous connecter à vos hôtes gérés. Par exemple, vous pouvez exécuter une commande comme si vous utilisiez un outil comme **runas** pour obtenir des privilèges d'administration.

Vous pouvez activer l'augmentation des privilèges pour l'ensemble d'un playbook en activant la case à cocher **ENABLE PRIVILEGE ESCALATION** pour votre modèle de tâche. La méthode d'augmentation des privilèges à utiliser, l'utilisateur alternatif à exécuter et le mot de passe de cet utilisateur sont généralement fournis comme partie intégrante de vos informations d'identification de la machine.

Vous pouvez utiliser des mots-clés spéciaux dans votre playbook pour spécifier les paramètres d'augmentation des privilèges pour un play ou des tâches spécifiques dans le play, qui sont différents de ceux spécifiés dans les informations d'identification de votre machine. Ces mots-clés remplacent la sélection de **ENABLE PRIVILEGE ESCALATION** pour ce play ou cette tâche.

Utilisez le mot-clé **become** pour activer ou désactiver l'augmentation des privilèges, indépendamment de la manière dont cette fonctionnalité est définie dans le fichier de configuration Ansible. Utilisez les valeurs **yes** ou **true** pour activer l'augmentation des privilèges, et **no** ou **false** pour la désactiver.

```
become: true
```

Si vous activez l'augmentation des privilèges, vous pouvez utiliser le mot-clé **become\_method** pour définir la méthode d'augmentation des privilèges à appliquer dans le cadre d'un play spécifique. L'exemple ci-dessous illustre que vous devez utiliser **runas** pour l'augmentation des privilèges.

```
become_method: runas
```

En outre, vous pouvez utiliser le mot-clé **become\_user** pour définir le compte d'utilisateur afin d'exécuter des tâches pour un play spécifique.

```
become_user: privileged_user
```

L'exemple suivant illustre l'utilisation de ces mots-clé dans un play :

```
- name: Escalate privilege
hosts: datacenter-west
become: yes
become_method: runas
become_user: System
tasks:
  - name: Display information about the System user
    win_whoami:
```

L'exemple suivant illustre ce playbook en spécifiant que l'augmentation des priviléges doit être utilisée pour une tâche, et les paramètres par défaut utilisés pour une autre, avec les détails de l'augmentation des priviléges définis par le play :

```
- name: Selected privilege escalation example
hosts: datacenter-west
become_method: runas
become_user: System
tasks:
  - name: Display information about the privilege escalation user
    win_whoami:
    become: yes

  - name: Display user information (using current default "become" settings)
    win_whoami:
```

## Utilisation de la documentation du module pour trouver des modules

Ansible est fourni avec un grand nombre de modules. Les administrateurs disposent ainsi de nombreux outils pour les tâches administratives courantes. Nous avons abordé, précédemment dans ce cours, le site Web de la documentation Ansible à l'adresse <http://docs.ansible.com>. L'*index des modules* sur le site Web est un moyen facile de parcourir la liste des modules fournis avec Ansible. Par exemple, les modules destinés à la gestion des utilisateurs et des services sont situés dans *Systems Modules*, tandis que ceux relatifs à l'administration de base de données sont accessibles dans *Database Modules*.

Pour chaque module, le site Web de documentation Ansible propose un résumé des fonctions et des instructions, en détaillant la façon dont chaque fonction spécifique peut être appelée avec des options dans le module. La documentation fournit également des exemples utiles vous montrant comment utiliser chaque module et définir leurs mots-clés dans une tâche.

Dans les sections suivantes, ce cours va vous présenter de nombreux modules particulièrement utiles pour les tâches d'automatisation de Microsoft Windows.

## Discussion relative aux variations de la syntaxe des playbooks

La dernière partie de ce chapitre est consacrée à certaines variations de la syntaxe YAML ou des playbooks Ansible que vous pouvez rencontrer.

## Commentaires YAML

Utilisez des commentaires pour faciliter la lisibilité des playbooks. Dans YAML, tout ce qui se trouve à droite du symbole dièse (#) est un commentaire. Si du contenu se trouve à gauche du commentaire, faites précéder le symbole dièse d'un espace.

```
# This is a YAML comment

some data # This is also a YAML comment
```

## Chaînes YAML

Les guillemets autour des chaînes ne sont pas obligatoires, même si la chaîne contient des espaces. Vous avez la possibilité de placer les chaînes entre guillemets droits simples ou doubles.

```
this is a string
'this is another string'
"this is yet another a string"
```

Les chaînes entre guillemets doubles analysent les séquences d'échappement identifiées par un caractère de barre oblique inverse préfixée. Utilisez les séquences d'échappement pour insérer des caractères spéciaux tels que `\t` pour une tabulation ou `\b` pour un retour arrière. Si une chaîne contient une barre oblique inverse \ qui n'est pas utilisée pour démarrer une séquence d'échappement, mettez la chaîne entre apostrophes pour éviter l'analyse. Les variables incorporées dans des chaînes comprises entre apostrophes continueront de fonctionner correctement.

Pour écrire des chaînes de plusieurs lignes, vous disposez de deux méthodes. Vous pouvez utiliser la barre verticale (|) pour indiquer que les sauts de ligne à l'intérieur des chaînes doivent être conservés.

```
include_newlines: |
    Example Company
    123 Main Street
    Atlanta, GA 30303
```

Vous pouvez également utiliser le symbole « plus grand que » (>) pour indiquer que les sauts de ligne doivent être convertis en espaces et que les espaces de début de lignes doivent être supprimés. Cette méthode est généralement utilisée pour scinder les chaînes longues au niveau des caractères d'espacement, de sorte qu'elles occupent plusieurs lignes afin de garantir une meilleure lisibilité.

```
fold_newlines: >
    This is an example
    of a long string,
    that will become
    a single sentence once folded.
```

Supprimez la nouvelle ligne de fin de la chaîne multiligne en ajoutant le symbole moins (-) aux indicateurs > ou |.

```
no_newlines: >-
    This example will
    contain no newlines
    at all.
```

## Dictionnaires YAML

Nous connaissez les collections de paires clé-valeur écrites en tant que blocs mis en retrait, comme dans l'exemple suivant :

```
name: svcrole
svcservice: W3Svc
svcport: 80
```

Vous connaissez peut-être les dictionnaires écrits dans un format condensé et inclus dans des accolades, comme suit :

```
{name: svcrole, svcservice: W3Svc, svcport: 80}
```

Dans la plupart des cas, vous devez éviter le format condensé, car il est plus difficile de lire et d'identifier les modifications dans les outils diff.

## Listes YAML

Vous connaissez les listes qui sont écrites à l'aide de la syntaxe normale à un tiret :

```
hosts:
  - servera
  - serverb
  - serverc
```

Les listes respectent également un format condensé délimité par des crochets, comme suit :

```
hosts: [servera, serverb, serverc]
```

Comme avec les dictionnaires, évitez la syntaxe condensée en ligne pour des raisons de lisibilité.

## Écriture abrégée obsolète d'un playbook avec des paires clé=valeur

Certains playbooks peuvent utiliser une ancienne méthode d'écriture abrégée pour définir des tâches, qui consiste à placer les paires clé-valeur du module sur la même ligne que le nom du module. Par exemple, vous pouvez rencontrer ce type de syntaxe :

```
tasks:
  - name: shorthand form
    service: name=W3Svc enabled=true state=started
```

Écrivez cette tâche comme suit :

```
tasks:  
  - name: preferred expanded form  
    service:  
      name: W3Svc  
      enabled: true  
      state: started
```

Évitez la forme abrégée et utilisez la forme étendue avec une paire clé-valeur par ligne.

La forme étendue contient davantage de lignes, mais elle facilite le travail. Les mots-clés pour la tâche sont empilés verticalement et sont plus faciles à différencier. Vos yeux peuvent lire le play directement de haut en bas, ce qui limite les mouvements oculaires de gauche à droite. En outre, contrairement à l'écriture abrégée, la syntaxe étendue utilise le langage YAML natif. Les outils de mise en surbrillance de la syntaxe dans les éditeurs de texte modernes sont plus efficaces avec le format étendu que le format abrégé.



## Références

### **Présentation des playbooks — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_intro.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html)

### **Utilisation des playbooks — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks.html)

### **Syntaxe YAML — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/reference\\_appendices/YAMLSyntax.html](https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html)

### **Utilisation d'Ansible et Windows : Style YAML — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/windows\\_usage.html#yaml-style](https://docs.ansible.com/ansible/latest/user_guide/windows_usage.html#yaml-style)

## ► Exercice guidé

# Mise en œuvre de plusieurs plays

Au cours de cet exercice, vous allez créer un playbook contenant plusieurs plays, puis l'utiliser pour vous assurer que les hôtes gérés sont correctement configurés.

## Résultats

Vous serez en mesure de créer et d'exécuter un playbook afin de gérer la configuration et d'effectuer des tâches administratives sur un hôte géré.

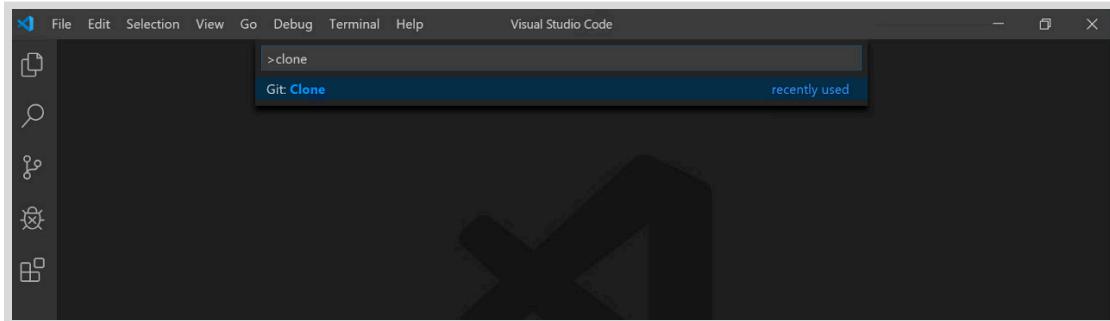
## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

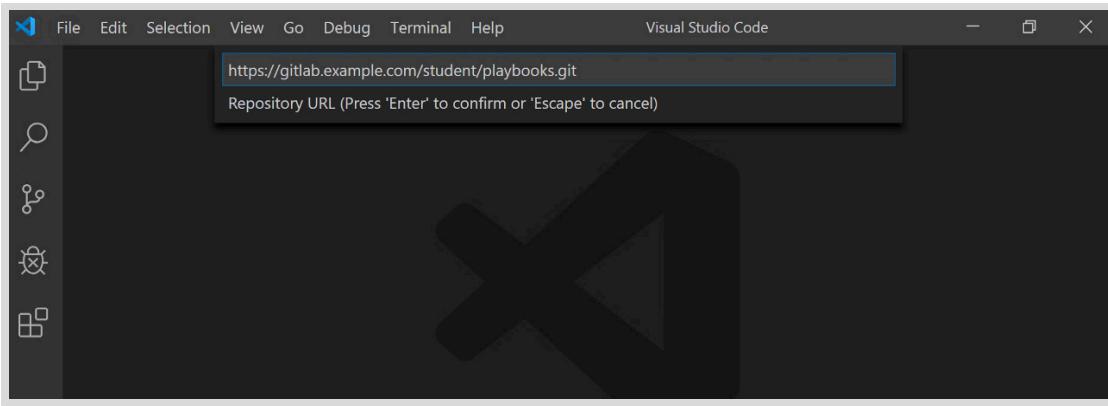
Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

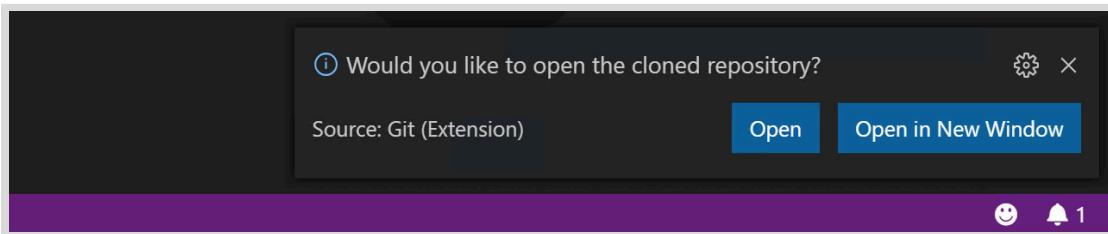
- ▶ 1. Lancez l'éditeur Visual Studio Code et, si vous n'avez pas déjà cloné le référentiel **playbooks**, clonez-le sur votre **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Ouvrez la palette de commandes en accédant à **View → Command Palette** ou en appuyant sur **Ctrl+Maj+P**.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.



- 1.3. À l'invite, fournissez l'URL de référentiel <https://gitlab.example.com/student/playbooks.git>, puis sélectionnez le dossier **Documents** dans le répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **playbooks** sur l'instance **workstation**.



1.4. Une invite d'ouverture du projet s'affiche. Cliquez sur **Open**.



- ▶ 2. Créez un playbook, nommez-le **multi.yml**, puis ajoutez-y les lignes nécessaires pour démarrer le premier play. Le play crée un partage de fichiers Windows sur **win1.example.com**.

2.1. Accédez à **File → New File** ou appuyez sur **Ctrl+N** pour créer un nouveau fichier.

2.2. Ajoutez une ligne au début du fichier composée de trois tirets. Cela indique le début du document YAML.

```
---
```

2.3. Ajoutez la ligne suivante pour indiquer le nom du play qui décrit l'état souhaité.

```
- name: Backup directory is shared on win1
```

2.4. Ajoutez la ligne suivante pour spécifier que le play s'applique à l'hôte géré **win1.example.com**. Mettez en retrait la ligne à l'aide de deux espaces (en l'alignant sur le mot-clé **name** au-dessus) pour indiquer qu'elle fait partie du premier play.

```
hosts: win1.example.com
```

2.5. Ajoutez la ligne suivante pour définir le début de la liste **tasks**. Mettez en retrait la ligne à l'aide de deux espaces (en l'alignant sur les mots-clés au-dessus) pour indiquer qu'elle fait partie du premier play.

```
tasks:
```

- 2.6. Enregistrez le fichier en accédant à **File** → **Save** ou en appuyant sur **Ctrl+S**. Nommez le playbook **multi.yml**, puis enregistrez-le dans le référentiel **playbooks** que vous avez créé dans le dossier **Documents**.

- 3. La première tâche du play vérifie l'existence d'un répertoire **C:\shares\backup** sur **win1.example.com**. Si le répertoire n'existe pas, Ansible le crée.

Mettez en retrait la première ligne de la tâche à l'aide de quatre espaces. Sous le mot-clé **tasks** du premier play, ajoutez les lignes suivantes.

```
- name: Backup directory exists ①
  win_file: ②
    path: C:\shares\backup ③
    state: directory ④
```

- ① Fournissez un nom déclaratif descriptif pour la tâche.
- ② La deuxième ligne est mise en retrait avec deux espaces supplémentaires et appelle le module **uri**. Le module **win\_file** est utilisé pour créer et modifier des fichiers et des répertoires sur des hôtes Windows.
- ③ Mettez en retrait deux espaces supplémentaires et fournissez le **path** du répertoire requis.
- ④ Directement sous l'option **path**, spécifiez qu'un **répertoire** doit exister au niveau du chemin d'accès fourni à l'aide du paramètre **state**.

- 4. Ajoutez une tâche dans le play qui utilise le module **win\_share** pour vous assurer que le répertoire **C:\shares\backup** est partagé sur le réseau. Ajoutez les lignes suivantes au playbook, sous la tâche précédente. Veillez à mettre en retrait la première ligne avec quatre espaces afin que les deux tâches soient alignées verticalement.

```
- name: Share is present ①
  win_share: ②
    name: backup ③
    description: Critical backup files
    path: C:\shares\backup ④
    full: devops ⑤
    state: present
```

- ① Fournissez un nom déclaratif descriptif pour la tâche.
- ② Mettez en retrait la deuxième ligne avec deux espaces supplémentaires et appelez le module **win\_share**. Les lignes restantes sont mises en retrait de plus de deux espaces et contiennent des paramètres pour le module **win\_share**.
- ③ Ansible s'assurera qu'un partage de fichiers nommé **backup** existe sur l'hôte géré.
- ④ Le paramètre **path** spécifie le répertoire à partager.
- ⑤ L'utilisateur **devops** disposera d'un accès **full**.

- 5. Démarrez un nouveau play dans le même playbook pour configurer l'hôte **win2.example.com** afin de copier un fichier vers le partage **win1.example.com** à des fins de sauvegarde.

- 5.1. Ajoutez la ligne suivante sans retrait pour créer un second play.

```
- name: File is backed up from win2
```

- 5.2. Spécifiez **win2.example.com** comme hôte cible pour le play. Utilisez deux espaces devant **hosts**, afin que **hosts** soit aligné horizontalement avec **name** juste au-dessus.

```
hosts: win2.example.com
```

- 5.3. Ajoutez la ligne suivante pour définir le début de la liste **tasks** pour le deuxième play.

```
tasks:
```

- ▶ 6. Créez une tâche qui copie des exemples de contenu dans un fichier sur **win2.example.com** pour la sauvegarde.

Mettez en retrait la première ligne de la tâche à l'aide de quatre espaces. Sous le mot-clé **tasks** du deuxième play, ajoutez les lignes suivantes.

```
- name: Example file created for backup ①
  win_copy: ②
    content: Hello. This is an example. ③
    dest: C:\Users\devops\Documents\logs.txt ④
```

- ① Fournissez un nom déclaratif descriptif pour la tâche.
- ② Utilisez le module **win\_copy** pour copier ou créer des fichiers sur les systèmes Windows.
- ③ Définissez le contenu du fichier sur une chaîne littérale à l'aide du paramètre **content**.
- ④ Spécifiez le chemin d'accès au fichier de destination pour le contenu à l'aide du paramètre **dest**.

- ▶ 7. Ajoutez une tâche finale au playbook qui copie le fichier dans le partage de fichiers **win1**. Mettez en retrait la déclaration de tâche en l'alignant sur la tâche précédente.

```
- name: Logs copied to backup share ①
  win_copy: ②
    src: C:\Users\devops\Documents\logs.txt ③
    dest: \\WIN1\backup\logs.txt ④
    remote_src: yes ⑤
```

- ① Fournissez un nom déclaratif descriptif pour la tâche.
- ② Appelez le module **win\_copy** utilisé précédemment, mais dans cette instance, copiez le journal à partir d'un fichier source au lieu d'un contenu littéral.
- ③ Spécifiez le fichier source avec le paramètre **src**.
- ④ Spécifiez le fichier de destination avec le paramètre **dest**.
- ⑤ Définissez **remote\_src** sur **yes** pour qu'Ansible localise le fichier source sur l'hôte géré **win2.example.com**.

- ▶ 8. Vérifiez que le playbook **multi.yml** final reflète le contenu structuré suivant.

```
---
- name: Backup directory is shared on win1
  hosts: win1.example.com
  tasks:
    - name: Backup directory exists
```

```

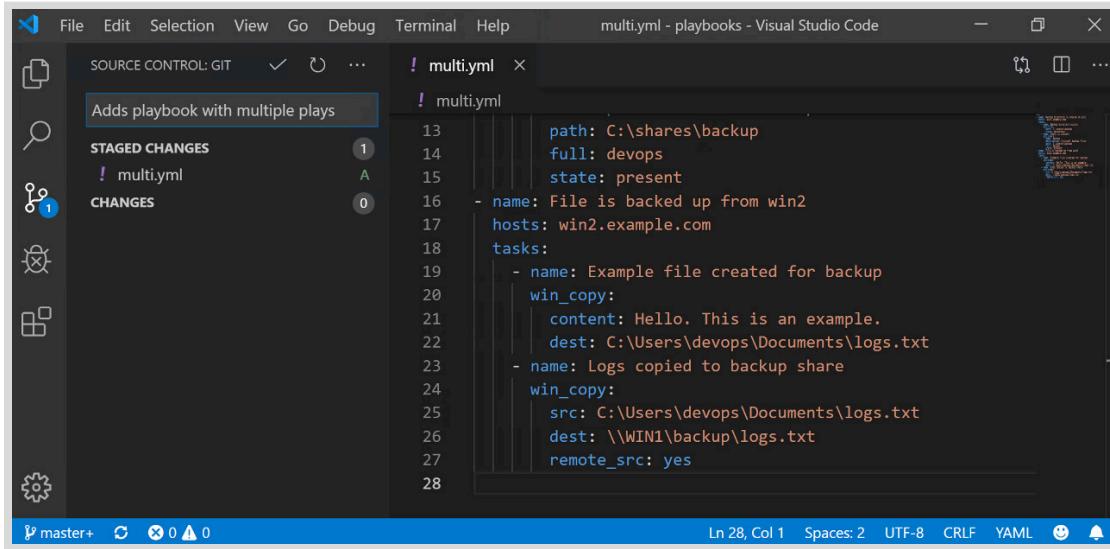
win_file:
  path: C:\shares\backup
  state: directory
- name: Share is present
  win_share:
    name: backup
    description: Critical backup files
    path: C:\shares\backup
    full: devops
    state: present

- name: File is backed up from win2
  hosts: win2.example.com
  tasks:
    - name: Example file created for backup
      win_copy:
        content: Hello. This is an example.
        dest: C:\Users\devops\Documents\logs.txt
    - name: Logs copied to backup share
      win_copy:
        src: C:\Users\devops\Documents\logs.txt
        dest: \\WIN1\backup\logs.txt
        remote_src: yes

```

- ▶ 9. Validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.

- 9.1. Cliquez sur **File** → **Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.
- 9.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard du fichier **multi.yml** pour indexer les modifications.
- 9.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.



- 9.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez

sur Synchronize Changes pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.

- ▶ 10. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- ▶ 11. Ajoutez `win2.example.com` à **Default inventory**.
  - 11.1. Cliquez sur **Inventories** dans le volet de navigation.
  - 11.2. Cliquez sur **Default inventory** dans la page **INVENTORIES**.
  - 11.3. Cliquez sur l'onglet **GROUPS**, puis sélectionnez le groupe **Windows**.
  - 11.4. Cliquez sur l'onglet **HOSTS** pour accéder à la liste d'hôtes dans le groupe.
  - 11.5. Cliquez sur **+**, puis sélectionnez **New host** pour ajouter un nouvel hôte au groupe.
  - 11.6. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
HOST NAME	win2.example.com

- 11.7. Cliquez sur **SAVE** pour ajouter le nouvel hôte.
- 11.8. Vérifiez la présence de `win1.example.com` et de `win2.example.com`.

- ▶ 12. Dans le volet de navigation, cliquez sur **Projects**. Cliquez sur l'icône d'actualisation située en regard du nom du projet **playbooks repository** pour vous assurer que le projet extrait le nouveau fichier.
- ▶ 13. Créez un modèle de tâche appelé **Backup win2 Log**.
  - 13.1. Cliquez sur **Templates**.
  - 13.2. Cliquez sur **+** pour créer un nouveau modèle de tâche, puis sélectionnez **Job Template**.

13.3. Renseignez le formulaire du nouveau modèle de tâche comme suit :

Champ	Valeur
NAME	Backup win2 Log
DESCRIPTION	Copie le journal de win2 vers le partage de fichiers win1
JOB TYPE	Run
INVENTORY	Inventaire par défaut
PROJECT	playbooks repository
PLAYBOOK	multi.yml
CREDENTIAL	DevOps

NEW JOB TEMPLATE

DETAILS    PERMISSIONS    COMPLETED JOBS    SCHEDULES    ADD SURVEY

* NAME Backup win2 Log	DESCRIPTION Copies log from win2 to win1 file share	* JOB TYPE Run	PROMPT ON LAUNCH
* INVENTORY Default inventory	* PROJECT playbooks repository	* PLAYBOOK multi.yml	
CREDENTIAL DevOps	FORKS 0	LIMIT PROMPT ON LAUNCH	
* VERTIBILITY 0 (Normal)	JOB TAGS PROMPT ON LAUNCH	SKIP TAGS PROMPT ON LAUNCH	

13.4. Cliquez sur **SAVE** pour créer le modèle de tâche.

- ▶ 14. Cliquez sur **LAUNCH** pour lancer la tâche **Backup win2 log**.
- ▶ 15. Examinez la sortie et vérifiez que l'option **STATUS** de la tâche dans le volet **DETAILS** affiche **Successful**. Lisez le journal de résultat pour vérifier que les plays **Backup directory is shared on win1** et **File is backed up from win2** sont tous deux exécutés.

```

JOBS / 25 - Backup win2 Log

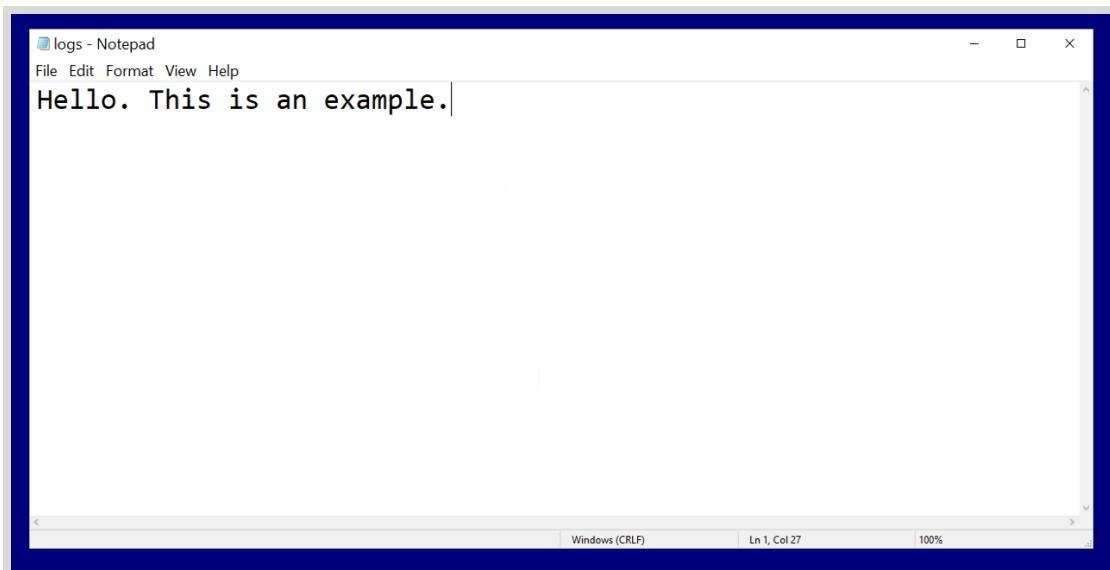
DETAILS
STATUS Successful
STARTED 8/30/2019 4:53:08 PM
FINISHED 8/30/2019 4:53:57 PM
JOB TEMPLATE Backup win2 Log
JOB TYPE Run
LAUNCHED BY student
INVENTORY Default inventory
PROJECT playbooks repository
REVISION 741d709
PLAYBOOK multi.yml
CREDENTIAL DevOps
ENVIRONMENT /var/lib/awx/venv/ansible
EXECUTION NODE localhost

Backup win2 Log
PLAYS 2 TASKS 6 HOSTS 2 ELAPSED 00:00:49
SEARCH Q KEY

22 TASK [Logs copied to backup share]
*****
23 changed: [win2.example.com]
24
25 PLAY RECAP *****
26 win1.example.com : ok=3    change
d=1 unreachable=0 failed=0 skipped
=0 rescued=0 ignored=0
27 win2.example.com : ok=3    change
d=2 unreachable=0 failed=0 skipped
=0 rescued=0 ignored=0
28

```

- ▶ 16. Connectez-vous à l'hôte géré **win1.example.com** pour vérifier que le fichier a été copié à partir de **win2.example.com**.
- 16.1. Cliquez sur **Recherche Windows**, recherchez **distance**, puis ouvrez Connexion Bureau à distance.
  - 16.2. Saisissez **win1.example.com** pour **Computer** et **EXAMPLE\student** pour **User Name** et **RedHat123@!** comme mot de passe.
  - 16.3. Cliquez sur **Recherche Windows** dans la fenêtre Connexion Bureau à distance **win1.example.com**, recherchez **C:\shares\backup\logs.txt**, puis ouvrez le fichier journal dans le Bloc-notes.



L'exercice guidé est maintenant terminé.

## ► Open Lab

# Mise en œuvre de playbooks Ansible

### Liste de contrôle des performances

Au cours de cet atelier, vous allez écrire et exécuter un playbook pour vous assurer que vos hôtes gérés sont correctement configurés.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Inspecter les playbooks Ansible à l'aide de Gitlab.
- Exécuter des playbooks à plusieurs plays à l'aide d'Ansible Tower.
- Mettre à jour les playbooks à l'aide de Visual Studio Code.
- Ajouter des hôtes à des inventaires.
- Accéder aux pages Web pour vous assurer que les playbooks s'exécutent correctement.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. Lors d'une activité précédente, vous avez installé et démarré le serveur Web. Exécutez le playbook **remove-iis.yml** pour supprimer le service et la page Web. Pour exécuter ce playbook, accédez à Ansible Tower à partir de **workstation**, accessible à l'adresse <https://tower.lab.example.com>, ou en double-cliquant sur l'icône de Bureau **Ansible Tower**.  
Connectez-vous à Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Mettez à jour le modèle de tâche **Run playbooks project** en sélectionnant le playbook **remove-iis.yml**.  
Exécutez le modèle de tâche.
2. À partir de **workstation**, accédez à votre instance GitLab sur <https://gitlab.example.com>. Vous pouvez soit utiliser l'icône du Bureau, soit utiliser Chrome pour accéder à l'URL. Connectez-vous en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Veillez à sélectionner la méthode d'authentification LDAP.  
Accédez au référentiel **playbooks** et passez en revue le playbook **install-review.yml**. Le playbook installe le service **Web-Server**, démarre le service associé et crée un fichier **index.html** personnalisé.

3. À partir de **workstation**, accédez à Visual Studio Code et clonez le référentiel Git, accessible à l'adresse <https://gitlab.example.com/student/playbooks.git>, sur **C:\Users\student\Documents\playbooks**.

Ajoutez le dossier **playbooks** à votre espace de travail, puis procédez aux modifications suivantes dans le playbook **install-review.yml**. Les commentaires figurant dans le playbook indiquent les modifications à apporter.

- Donnez au premier play (qui indique - **name: Install web service**) le nom **The IIS web server is deployed and running**.
  - Renommez la première tâche (qui indique - **name: Install service**) en **IIS service is installed**.
  - Renommez la deuxième tâche (qui indique - **name: Start service**) en **IIS service is running**.
  - Mettez à jour le contenu du fichier HTML **index.html** pour les deux tâches qui ciblent les deux hôtes différents en mettant à jour le paramètre **content**. Donnez aux modules la valeur **The win1 web server has been provisioned by Ansible** et **The win2 web server has been provisioned by Ansible**.
4. Enregistrez vos modifications, validez le fichier à l'aide du message **Add better description to playbook**, et poussez vos modifications vers le référentiel distant.
5. À partir d'Ansible Tower, mettez à jour l'inventaire **Default inventory** en ajoutant **win2.example.com** en tant qu'hôte dans le groupe **Windows** de ce même inventaire.
6. À partir d'Ansible Tower, mettez à jour le modèle de tâche **Run playbooks project** en lui indiquant d'utiliser **install-review.yml** comme playbook.  
Exécutez le modèle de tâche, puis passez en revue la sortie de la tâche ; la sortie de la tâche indique qu'Ansible installe et configure le serveur Web sur les deux hôtes de l'inventaire par défaut, puis personnalise la page Web pour chaque hôte.
7. À partir de **workstation**, ouvrez Chrome et accédez à <http://win1.example.com>. Assurez-vous que la page d'accueil se présente comme suit :

The win1 web server has been provisioned by Ansible

À partir de **workstation**, ouvrez Chrome et accédez à <http://win2.example.com>. Assurez-vous que la page d'accueil indique ce qui suit :

The win2 web server has been provisioned by Ansible

L'atelier est maintenant terminé.

## ► Solution

# Mise en œuvre de playbooks Ansible

### Liste de contrôle des performances

Au cours de cet atelier, vous allez écrire et exécuter un playbook pour vous assurer que vos hôtes gérés sont correctement configurés.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Inspecter les playbooks Ansible à l'aide de Gitlab.
- Exécuter des playbooks à plusieurs plays à l'aide d'Ansible Tower.
- Mettre à jour les playbooks à l'aide de Visual Studio Code.
- Ajouter des hôtes à des inventaires.
- Accéder aux pages Web pour vous assurer que les playbooks s'exécutent correctement.

### Avant De Commencer

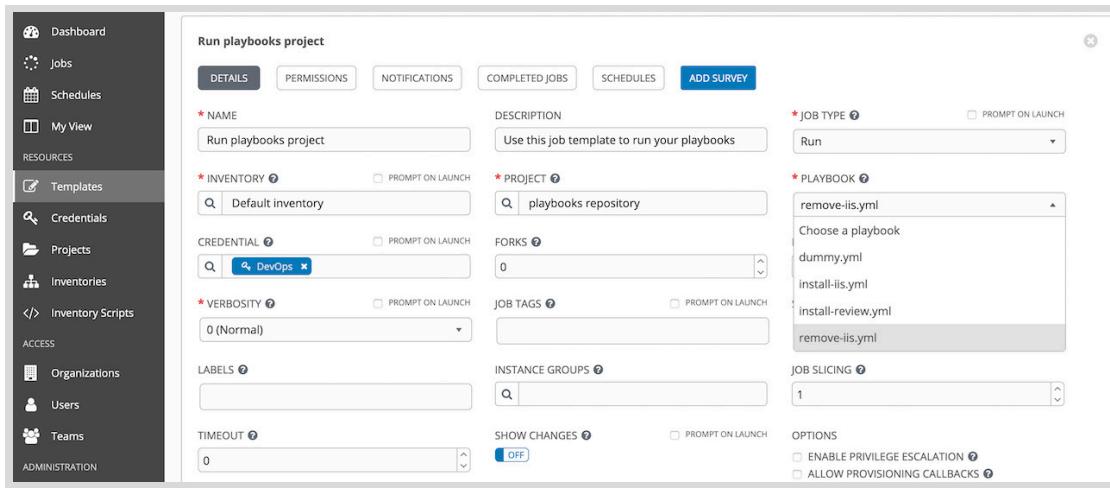
Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. Lors d'une activité précédente, vous avez installé et démarré le serveur Web. Exécutez le playbook **remove-iis.yml** pour supprimer le service et la page Web. Pour exécuter ce playbook, accédez à Ansible Tower à partir de **workstation**, accessible à l'adresse <https://tower.lab.example.com>, ou en double-cliquant sur l'icône de Bureau **Ansible Tower**.  
Connectez-vous à Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe Mettez à jour le modèle de tâche **Run playbooks project** en sélectionnant le playbook **remove-iis.yml**.  
Exécutez le modèle de tâche.
  - 1.1. À partir de **workstation**, double-cliquez sur l'icône du Bureau **Ansible Tower** pour ouvrir la console Web.  
Connectez-vous à Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe
  - 1.2. Accédez au modèle de tâche **Run playbooks project**, accessible à partir du lien **Templates**. Cliquez sur le modèle de tâche pour le modifier.
  - 1.3. Sélectionnez **remove-iis.yml** dans le champ **PLAYBOOKS**, puis cliquez sur **SAVE** pour enregistrer vos modifications.

### chapitre 3 | Mise en œuvre de playbooks Ansible



14. Pour exécuter le modèle de tâche, cliquez sur l'icône de fusée ou sur le bouton **LAUNCH** qui est disponible après avoir enregistré vos modifications.
15. Le résultat indique qu'Ansible supprime le fichier **index.html**, arrête le service de serveur Web et supprime le service IIS (**Web-Server**) de **win1.example.com**.

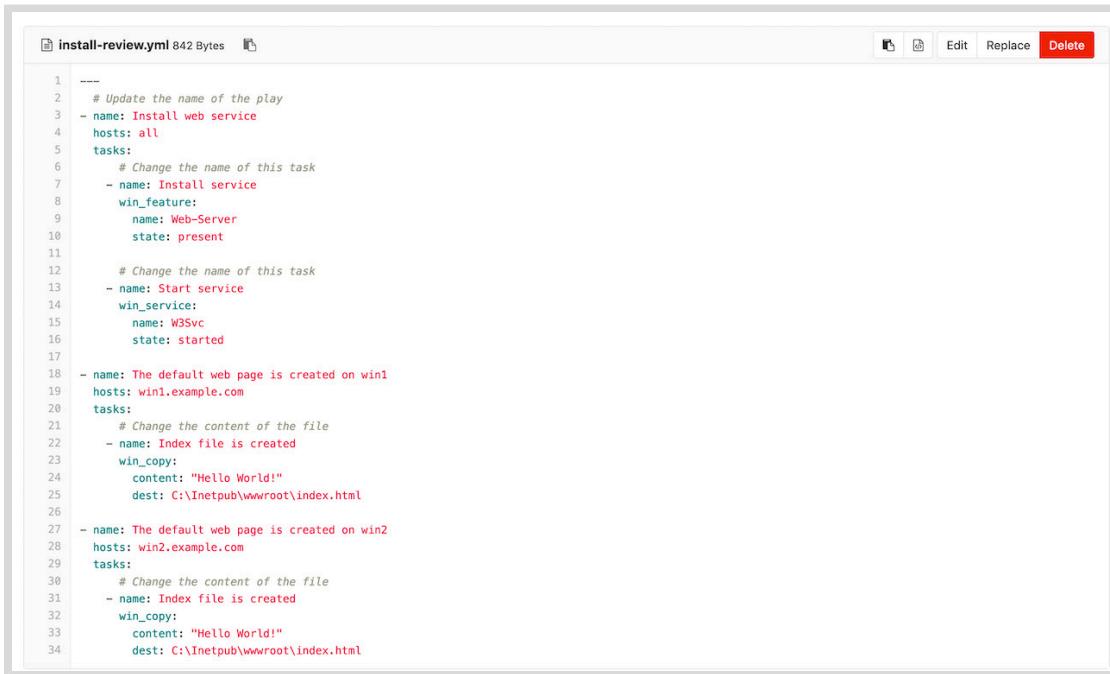
PLAY	TASK	HOST	STATE	MESSAGE
1	Gathering Facts	win1.example.com	ok	ok: [win1.example.com]
2	Website index.html is removed	win1.example.com	changed	changed: [win1.example.com]
3	IIS service is stopped	win1.example.com	ok	ok: [win1.example.com]
4	IIS service is removed	win1.example.com	ok	ok: [win1.example.com]
5	PLAY RECAP		ok=4 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0	win1.example.com : ok=4 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

2. À partir de **workstation**, accédez à votre instance GitLab sur **https://gitlab.example.com**. Vous pouvez soit utiliser l'icône du Bureau, soit utiliser Chrome pour accéder à l'URL. Connectez-vous en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Veillez à sélectionner la méthode d'authentification LDAP.  
Accédez au référentiel **playbooks** et passez en revue le playbook **install-review.yml**. Le playbook installe le service **Web-Server**, démarre le service associé et crée un fichier **index.html** personnalisé.

  - 2.1. Double-cliquez sur l'icône de Bureau **GitLab**, puis connectez-vous à GitLab en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

**chapitre 3 |** Mise en œuvre de playbooks Ansible

Examinez le playbook **install-review.yml** dans le référentiel **playbooks**. Pour ce faire, accédez à **Projects → Your Projects** et sélectionnez **student / playbooks**. Cliquez sur **install-review.yml** pour afficher le playbook. Les commentaires des playbooks indiquent les modifications que vous allez effectuer dans une étape ultérieure.



```

1  ---
2  # Update the name of the play
3  - name: Install web service
4  hosts: all
5  tasks:
6      # Change the name of this task
7      - name: Install service
8          win_feature:
9              name: Web-Server
10             state: present
11
12      # Change the name of this task
13      - name: Start service
14          win_service:
15              name: W3SVC
16              state: started
17
18      - name: The default web page is created on win1
19      hosts: win1.example.com
20      tasks:
21          # Change the content of the file
22          - name: Index file is created
23              win_copy:
24                  content: "Hello World!"
25                  dest: C:\Inetpub\wwwroot\index.html
26
27      - name: The default web page is created on win2
28      hosts: win2.example.com
29      tasks:
30          # Change the content of the file
31          - name: Index file is created
32              win_copy:
33                  content: "Hello World!"
34                  dest: C:\Inetpub\wwwroot\index.html

```

3. À partir de **workstation**, accédez à Visual Studio Code et clonez le référentiel Git, accessible à l'adresse <https://gitlab.example.com/student/playbooks.git>, sur **C:\Users\student\Documents\playbooks**.

Ajoutez le dossier **playbooks** à votre espace de travail, puis procédez aux modifications suivantes dans le playbook **install-review.yml**. Les commentaires figurant dans le playbook indiquent les modifications à apporter.

- Donnez au premier play (qui indique **- name: Install web service**) le nom **The IIS web server is deployed and running**.
- Renommez la première tâche (qui indique **- name: Install service**) en **IIS service is installed**.
- Renommez la deuxième tâche (qui indique **- name: Start service**) en **IIS service is running**.
- Mettez à jour le contenu du fichier HTML **index.html** pour les deux tâches qui ciblent les deux hôtes différents en mettant à jour le paramètre **content**. Donnez aux modules la valeur **The win1 web server has been provisioned by Ansible** et **The win2 web server has been provisioned by Ansible**.

- 3.1. À partir de **workstation**, accédez à Visual Studio Code en double-cliquant sur le raccourci du Bureau.

- 3.2. Clonez le référentiel **playbooks** en ouvrant la palette de commandes. Pour ce faire, accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P**.

Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.

- 3.3. Utilisez l'URL de référentiel `https://gitlab.example.com/student/playbooks.git`. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **playbooks**.

Vous pouvez éventuellement sélectionnez **Open** dans la fenêtre qui s'affiche après le clonage pour ouvrir le dossier.

- 3.4. Pour ajouter le dossier **playbooks** à votre espace de travail, accédez à **File → Open Folder**.

Dans l'Explorateur Windows, accédez à **Local Disk (C:) → Users → student → playbooks**, puis cliquez sur **Select Folder**.

- 3.5. Ouvrez le playbook `install-review.yml` dans le projet **architecture**. Procédez aux modifications suivantes :

- Nommez le premier play **The IIS web server is deployed and running**.

```
---  
# Update the name of the play  
- name: The IIS web server is deployed and running  
hosts: all  
...output omitted...
```

- Renommez la première tâche **IIS service is installed**.

```
...output omitted...  
# Change the name of this task  
- name: IIS service is installed  
win_feature:  
  name: Web-Server  
  state: present  
...output omitted...
```

- Renommez la deuxième tâche **IIS service is started**.

```
...output omitted...  
# Change the name of this task  
- name: IIS service is started  
win_service:  
  name: W3Svc  
  state: started  
...output omitted...
```

- Mettez à jour le contenu du fichier HTML `index.html` pour les deux tâches qui ciblent les deux hôtes différents en mettant à jour le paramètre **content**. Donnez aux modules la valeur **The win1 web server has been provisioned by Ansible** et **The win2 web server has been provisioned by Ansible**.

```
...output omitted...  
- name: The default web page is created on win1  
hosts: win1.example.com
```

```
tasks:  
# Change the content of the file  
- name: Index file is created  
  win_copy:  
    content: "The win1 web server has been provisioned by Ansible"  
    dest: C:\Inetpub\wwwroot\index.html  
  
- name: The default web page is created on win2  
hosts: win2.example.com  
tasks:  
# Change the content of the file  
- name: Index file is created  
  win_copy:  
    content: "The win2 web server has been provisioned by Ansible"  
    dest: C:\Inetpub\wwwroot\index.html
```

Une fois ces modifications terminées, le fichier doit ressembler à ce qui suit. La sortie omet intentionnellement les commentaires.

```
---  
- name: Web server is installed and running  
hosts: all  
tasks:  
- name: IIS service is installed  
  win_feature:  
    name: Web-Server  
    state: present  
  
- name: IIS service is started  
  win_service:  
    name: W3Svc  
    state: started  
  
- name: Default web page is created on win1  
hosts: win1.example.com  
tasks:  
- name: Index file is created  
  win_copy:  
    content: "The win1 web server has been provisioned by Ansible"  
    dest: C:\Inetpub\wwwroot\index.html  
  
- name: Default web page is created on win2  
hosts: win2.example.com  
tasks:  
- name: Index file is created  
  win_copy:  
    content: "The win2 web server has been provisioned by Ansible"  
    dest: C:\Inetpub\wwwroot\index.html
```



### Note

Le fichier de solution terminé est disponible dans **solutions/install-review.sol** du projet **playbooks**.

4. Enregistrez vos modifications, validez le fichier à l'aide du message **Add better description to playbook**, et poussez vos modifications vers le référentiel distant.

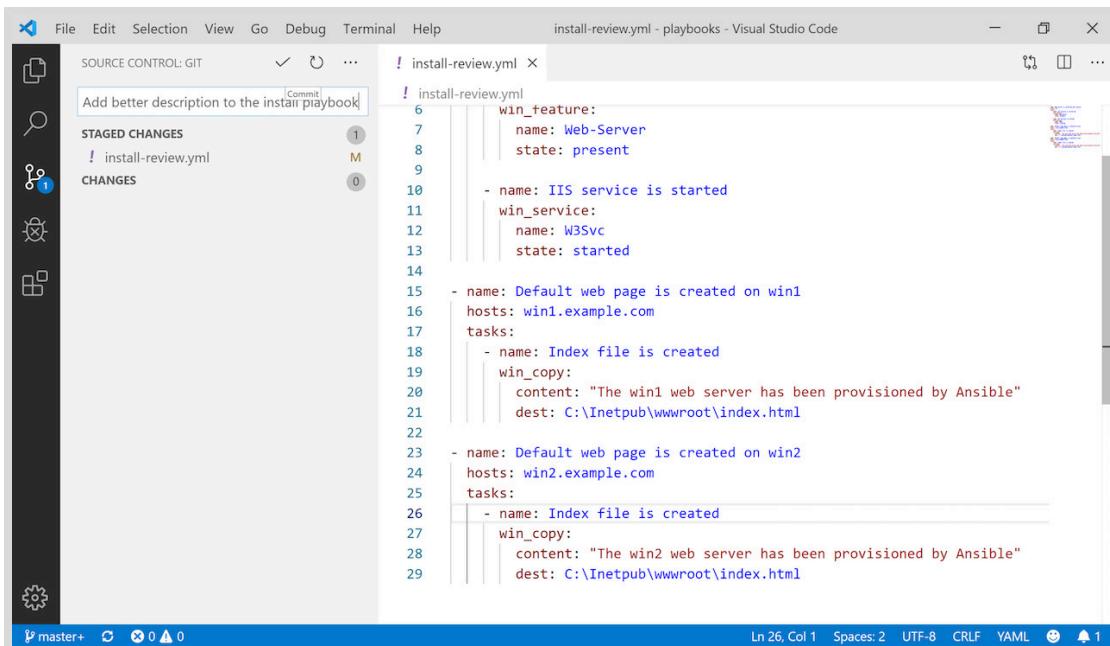
- 4.1. Enregistrez vos modifications.
- 4.2. À partir de la barre latérale de Visual Studio Code, sélectionnez **Source Control** pour accéder à la liste des modifications.
- 4.3. Survolez **install-review.yml** pour accéder aux options disponibles pour ce fichier. Cliquez sur **Stage Changes** pour ajouter vos modifications à l'index.

```

File Edit Selection View Go Debug Terminal Help
install-review.yml - playbooks - Visual Studio Code
SOURCE CONTROL: GIT
Message (press Ctrl+Enter to commit)
CHANGES
! install-review.yml 1 Stage Changes
! install-review.yml
  ! install-review.yml
  win_feature:
    name: Web-Server
    state: present
  - name: IIS service is started
    win_service:
      name: W3Svc
      state: started
  - name: Default web page is created on win1
    hosts: win1.example.com
    tasks:
      - name: Index file is created
        win_copy:
          content: "The win1 web server has been provisioned by Ansible"
          dest: C:\Inetpub\wwwroot\index.html
  - name: Default web page is created on win2
    hosts: win2.example.com
    tasks:
      - name: Index file is created
        win_copy:
          content: "The win2 web server has been provisioned by Ansible"
          dest: C:\Inetpub\wwwroot\index.html
Ln 26, Col 1  Spaces: 2  UTF-8  CRLF  YAML  1

```

- 4.4. Dans le fichier texte **Message**, saisissez le message **Add better description to the install playbook**.
- 4.5. Cliquez sur **Commit** pour valider les modifications dans votre référentiel local.



```

! install-review.yml
!
win_feature:
  name: Web-Server
  state: present
-
  - name: IIS service is started
    win_service:
      name: W3Svc
      state: started
-
  - name: Default web page is created on win1
    hosts: win1.example.com
    tasks:
      - name: Index file is created
        win_copy:
          content: "The win1 web server has been provisioned by Ansible"
          dest: C:\Inetpub\wwwroot\index.html
-
  - name: Default web page is created on win2
    hosts: win2.example.com
    tasks:
      - name: Index file is created
        win_copy:
          content: "The win2 web server has been provisioned by Ansible"
          dest: C:\Inetpub\wwwroot\index.html

```

Ln 26, Col 1 Spaces: 2 UTF-8 CRLF YAML ⚡ 1

- 4.6. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
5. À partir d'Ansible Tower, mettez à jour l'inventaire **Default inventory** en ajoutant **win2.example.com** en tant qu'hôte dans le groupe **Windows** de ce même inventaire.
  - 5.1. À partir d'Ansible Tower, cliquez sur **Inventories** dans le volet de navigation pour accéder à l'inventaire **Default inventory**.  
Cliquez sur **Default inventory** pour le modifier.
  - 5.2. Cliquez sur **GROUPS** pour lister tous les groupes de l'inventaire.
  - 5.3. Cliquez sur l'inventaire **Windows** pour accéder à son contenu, puis sélectionnez **HOSTS** pour accéder à la liste des hôtes.
  - 5.4. Cliquez sur le signe plus vert, puis sélectionnez **Existing Host** pour ajouter l'hôte **win2.example.com** créé précédemment.
  - 5.5. Activez la case à cocher **win2.example.com**, puis cliquez sur **SAVE**.
6. À partir d'Ansible Tower, mettez à jour le modèle de tâche **Run playbooks project** en lui indiquant d'utiliser **install-review.yml** comme playbook.  
Exécutez le modèle de tâche, puis passez en revue la sortie de la tâche ; la sortie de la tâche indique qu'Ansible installe et configure le serveur Web sur les deux hôtes de l'inventaire par défaut, puis personnalise la page Web pour chaque hôte.
  - 6.1. À partir d'Ansible Tower, cliquez sur **Templates** dans le volet de navigation pour accéder au modèle de tâche **Run playbooks project**.
  - 6.2. Cliquez sur le modèle de tâche pour le modifier. Sélectionnez **install-review.yml** pour le champ **PLAYBOOK**, puis cliquez sur **SAVE** pour mettre à jour le modèle de tâche.

### chapitre 3 | Mise en œuvre de playbooks Ansible

The screenshot shows the 'Run playbooks project' configuration dialog. Key settings include:

- NAME:** Run playbooks project
- DESCRIPTION:** Use this job template to run your playbook.
- JOB TYPE:** Run
- INVENTORY:** Default inventory
- PROJECT:** playbooks repository
- PLAYBOOK:** install-review.yml (selected)
- OPTIONS:** ENABLE PRIVILEGE ESCALATION, ALLOW PROVISIONING CALLBACKS, ENABLE CONCURRENT JOBS, USE FACT CACHE

6.3. Cliquez sur **LAUNCH** pour exécuter le modèle de tâche.

6.4. Cette action vous redirige vers la sortie de la tâche. Le résultat indique qu'Ansible est en mesure d'installer et de démarrer le serveur Web sur **win1** et **win2**, et de créer la page HTML **index.html** personnalisée sur chaque serveur.

The screenshot shows the Ansible Tower interface with the following details:

- DETAILS:**
  - Status: Successful
  - Started: 9/3/2019 3:33:27 PM
  - Finished: 9/3/2019 3:35:05 PM
  - Job Template: Run playbooks project
  - Job Type: Run
  - Launched By: admin
  - Inventory: Default inventory
  - Project: playbooks repository
  - Revision: 2bcafb8
  - Playbook: install-review.yml
  - Credential: DevOps
  - Environment: /var/lib/awx/venv/ansible
  - Execution Node: localhost
  - Instance Group: tower
  - Extra Variables: YAML (JSON selected)
- Run playbooks project:**
  - PLAYS: 3, TASKS: 7, HOSTS: 2, ELAPSED: 00:01:37
  - Log Output (partial):
 

```

11 changed: [win1.example.com]
12
13 TASK [IIS service is started]
*****
14 changed: [win1.example.com]
15 changed: [win2.example.com]
16
17 PLAY [Default web page is created on win1]
*****
18
19 TASK [Gathering Facts]
*****
20 ok: [win1.example.com]
21
22 TASK [Index file is created]
*****
23 changed: [win1.example.com]
24
25 PLAY [Default web page is created on win2]
      
```

7. À partir de **workstation**, ouvrez Chrome et accédez à <http://win1.example.com>. Assurez-vous que la page d'accueil se présente comme suit :

The win1 web server has been provisioned by Ansible

À partir de **workstation**, ouvrez Chrome et accédez à <http://win2.example.com>. Assurez-vous que la page d'accueil indique ce qui suit :

The win2 web server has been provisioned by Ansible

- 7.1. Pour vous assurer que le serveur Web est en cours d'exécution et que la page Web est disponible sur le serveur, ouvrez un nouvel onglet dans Chrome et accédez à <http://win2.example.com>.
- 7.2. La page Web par défaut doit se présenter comme suit :

The win1 web server has been provisioned by Ansible

- 7.3. Ouvrez un nouvel onglet et accédez à <http://win2.example.com>. La page Web par défaut doit se présenter comme suit :

The win2 web server has been provisioned by Ansible

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Un *play* est un ensemble ordonné de tâches exécutées sur des hôtes choisis dans votre inventaire.
- Un *playbook* est un fichier texte contenant une liste d'un ou plusieurs plays à exécuter dans un ordre spécifique.
- Les plays multiples peuvent s'avérer très utiles lors de l'orchestration d'un déploiement complexe impliquant des tâches différentes sur différents hôtes.
- Dans Red Hat Ansible Tower, un *projet* représente un référentiel qui est disponible dans un système de contrôle de version et qui contient au moins un playbook.
- Dans Red Hat Ansible Tower, un *modèle de tâche* représente une commande enregistrée qui peut être utilisée à plusieurs reprises pour exécuter un playbook à partir d'un projet avec un inventaire, des informations d'identification et des paramètres de configuration particuliers.



## chapitre 4

# Gestion des variables et des faits

### Objectif

Rédiger des playbooks qui utilisent des variables pour simplifier la gestion du playbook, et des faits afin de référencer des informations sur les hôtes gérés.

### Résultats

- Créer et référencer des variables qui affectent des hôtes ou groupes d'hôtes spécifiques, le play ou l'environnement global, et décrire le fonctionnement de la priorité des variables.
- Chiffrer les variables sensibles à l'aide d'Ansible Vault et exécuter des playbooks faisant référence à des fichiers de variables chiffrées par Vault.
- Référencer des données spécifiques à des hôtes gérés particuliers à l'aide de faits Ansible.

### Sections

- Gestion des variables (et exercice guidé)
- Gestion des secrets (et exercice guidé)
- Obtention d'informations sur le système avec des faits (et exercice guidé)

### Atelier

Gestion des variables et des faits

# Gestion des variables

---

## Résultats

Au terme de cette section, vous devez pouvoir définir et référencer des variables qui affectent des hôtes ou groupes d'hôtes spécifiques, le play ou l'environnement global, et décrire le fonctionnement de la priorité des variables.

## Présentation des variables Ansible

Ansible prend en charge des variables que vous pouvez utiliser pour stocker des valeurs en vue de leur réutilisation dans l'ensemble des fichiers au sein d'un projet Ansible. Ces variables simplifient la création et la maintenance d'un projet, et réduisent le nombre d'erreurs.

Elles facilitent également la gestion de valeurs dynamiques pour un environnement donné dans votre projet Ansible. Exemples de valeurs de variables :

- Utilisateurs à créer, modifier ou supprimer.
- Logiciels à installer ou à désinstaller.
- Services à arrêter, démarrer ou redémarrer.
- Fichiers à créer, modifier ou supprimer.
- Archives à récupérer ou extraire sur Internet.

## Noms des variables

Les noms de variables sont caractères composées de lettres, chiffres et traits de soulignement. Ils doivent commencer par une lettre.

Le tableau suivant illustre la différence entre les noms de variables valides et non valides.

### Exemples de noms de variables Ansible valides et non valides

Noms de variables non valides	Noms de variables valides
<code>web server</code>	<code>web_server</code>
<code>web-server</code>	
<code>remote.file</code>	<code>remote_file</code>
<code>1st file</code>	<code>file_1</code>
<code>1st_file</code>	<code>file1</code>
<code>remoteserver\$1</code>	<code>remote_server_1</code>
	<code>remote_server1</code>

## Définition de variables

Vous pouvez définir des variables à des emplacements divers au sein d'un projet Ansible. Néanmoins, cela peut être simplifié à trois niveaux d'étendue de base :

### Étendue des variables

#### Étendue globale

Variables définies à partir d'une ligne de commande ou d'une configuration Ansible.

#### Étendue de play

Variables définies dans le play et les structures correspondantes.

#### Étendue d'hôte

Variables définies sur des groupes d'hôtes et hôtes individuels par l'inventaire, une collecte de faits ou des tâches enregistrées.

Si vous définissez le même nom de variable sur plusieurs niveaux, le niveau avec la priorité la plus élevée l'emporte. Une étendue étroite a la priorité par rapport à une étendue plus vaste ; les variables que vous définissez dans un inventaire sont remplacées par celles que vous définissez dans le playbook. Ces dernières sont remplacées par les variables définies sur la ligne de commande.

La hiérarchie des variables est abordée en détail dans la documentation Ansible vers laquelle un lien est disponible dans les Références figurant au bas de cette section.

## Gestion de variables dans des playbooks

Les variables jouent un rôle important dans les playbooks Ansible car elles facilitent la gestion des données variables.

### Définition de variables dans les plays

Lorsque vous écrivez des playbooks, vous pouvez définir vos propres variables, puis invoquer ces valeurs dans une tâche.

Vous pouvez définir des variables de play de plusieurs manières. Une façon courante consiste à placer la variable dans un bloc **vars** au début d'un play :

```
- hosts: all
  vars:
    user_name: joe
    user_state: present
```

Il est également possible de définir des variables de play dans des fichiers externes. Dans ce cas, au lieu d'utiliser un bloc **vars** dans le play, vous pouvez utiliser la directive **vars\_files**, suivie d'une liste de noms de fichiers de variable externes correspondant à l'emplacement du playbook :

```
- hosts: all
  vars_files:
    - vars/users.yml
```

Vous définissez ensuite les variables de playbook au format YAML dans ces fichiers de variables externes :

```
user_name: joe
user_state: present
```

## Utilisation de variables dans les plays

Après avoir déclaré des variables, vous pouvez utiliser les variables dans des tâches. Vous référez les variables en plaçant le nom de la variable entre doubles accolades ({{ }}). Ansible remplace la variable par sa valeur lorsqu'il exécute la tâche.

```
- name: Example play
hosts: all
vars:
  user_name: joe

tasks:
  # This line will read: Creates the user joe
  - name: Creates the user {{ user_name }}
    win_user:
      # This line will create the user named Joe
      name: "{{ user_name }}"
```



### Important

Lorsque vous utilisez une variable comme premier élément commençant une valeur, l'utilisation de guillemets est impérative. Cela empêche Ansible d'interpréter la référence de variable comme le début d'un dictionnaire YAML. Si les guillemets ne sont pas utilisés, le message suivant apparaît :

```
win_service:
  name: {{ service }}
  ^ here
```

We could be wrong, but this one looks like it might be an issue with missing quotes. Always quote template expression brackets when they start a value. For instance:

```
with_items:
  - {{ foo }}
```

Should be written as:

```
with_items:
  - "{{ foo }}"
```

## Description des variables d'hôte et des variables de groupe

Les variables d'inventaire qui s'appliquent directement aux hôtes appartiennent à deux grandes catégories : les *variables d'hôtes*, qui s'appliquent à un hôte spécifique, et les *variables de groupe*, qui s'appliquent à tous les hôtes d'un groupe d'hôtes ou d'un groupe de groupes d'hôtes. Les variables d'hôte l'emportent sur les variables de groupe, mais celles que vous définissez dans un playbook l'emportent sur les deux.

## Définition des variables d'hôte et de groupe dans l'inventaire

Une façon de définir les variables d'hôte et les variables de groupe consiste à les définir dans l'inventaire. Les hôtes, les groupes et l'inventaire dans leur ensemble disposent d'une zone de texte **VARIABLES** que vous pouvez utiliser pour définir des variables.

Figure 4.1 illustre la définition de la variable **ansible\_user** sur **devops** pour l'hôte d'inventaire **win1.example.com** :

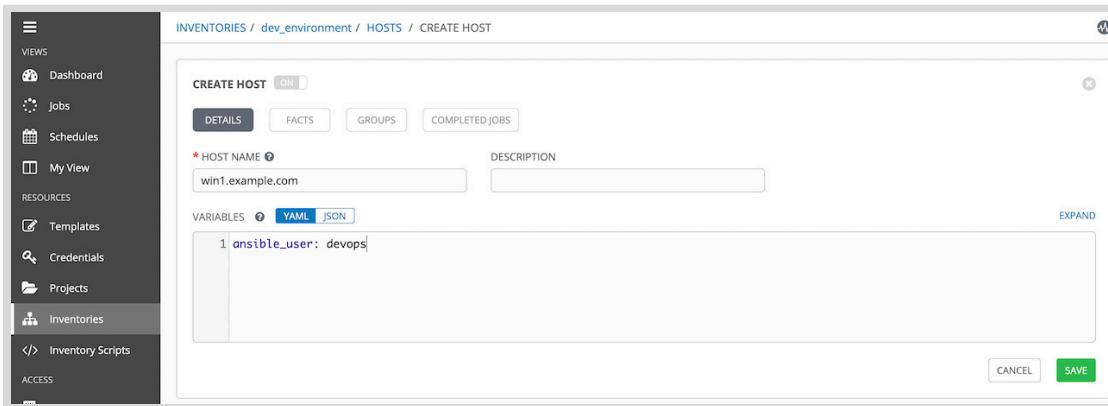


Figure 4.1: Définition des variables d'inventaire dans Ansible Tower

Cette approche comporte des avantages et des inconvénients. L'avantage est que les variables d'inventaire stockées dans Ansible Tower sont cohérentes pour tous les modèles de tâche qui utilisent cet inventaire. L'inconvénient est que vous trouverez peut-être plus facilement la gestion des variables spécifiques à l'hôte et au groupe pour un projet en particulier en utilisant des répertoires de variables de playbook ou en définissant les variables dans le playbook. Cette section couvre ensuite ces approches.

## Utilisation de répertoires pour remplir des variables d'hôte et de groupe

Une autre approche pour définir des variables pour les hôtes et les groupes d'hôtes consiste à créer deux répertoires, **group\_vars** et **host\_vars**, dans le répertoire de base Ansible. Ces répertoires contiennent des fichiers qui définissent respectivement des variables de groupe et des variables d'hôtes.

Pour définir des variables de groupe pour le groupe **servers**, vous devez créer un fichier YAML nommé **group\_vars/servers**, puis son contenu définit des variables sur des valeurs en utilisant la même syntaxe que celle figurant dans un playbook :

```
user_name: joe
```

De même, pour définir des variables d'hôte pour un hôte donné, créez un fichier contenant les variables d'hôte et dont le nom correspond à l'hôte dans le répertoire **host\_vars**.

Les exemples suivants illustrent cette approche plus en détail. Prenons un scénario dans lequel deux datacenters doivent être gérés et où les hôtes des datacenters sont définis dans l'inventaire :

- Le groupe **datacenter1** se compose des hôtes **demo1.example.com** et **demo2.example.com**.
- Le groupe **datacenter2** se compose des hôtes **demo3.example.com** et **demo4.example.com**.

**chapitre 4 |** Gestion des variables et des faits

- Le groupe **datacenters** se compose des groupes **datacenter1** et **datacenter2**; les quatre hôtes qui sont membres de ces groupes sont membres du groupe **datacenters**.

Avec cet inventaire, vous pouvez configurer les fichiers **host\_vars** et **group\_vars** pour définir des variables pour différents hôtes ou groupes. Vous n'avez pas besoin de disposer d'un fichier pour un hôte ou un groupe si celui-ci est vide. Si vous souhaitez définir des paramètres pour chaque hôte ou groupe individuellement, vous pouvez disposer d'une structure de répertoire comme ceci pour votre projet :

```
project
└── group_vars
    ├── all
    ├── datacenters
    │   ├── datacenters1
    │   └── datacenters2
    └── host_vars
        ├── demo1.example.com
        ├── demo2.example.com
        ├── demo3.example.com
        └── demo4.example.com
└── playbook.yml
```

Voici quelques opérations que vous pouvez effectuer avec cette structure :

Définition d'une valeur pour tous les serveurs

Si vous devez définir une valeur générale pour tous les serveurs dans les deux datacenters, définissez une variable de groupe pour le groupe d'hôtes **datacenters** dans le fichier **group\_vars/datacenters**:

```
time_server: 192.168.1.254
```

Vous pouvez également utiliser le groupe d'hôtes **all** pour affecter tous les hôtes de l'inventaire.

Définition de valeurs différentes pour chaque datacenter

Si la valeur à définir varie d'un datacenter à l'autre, définissez une variable de groupe pour chaque groupe d'hôtes de datacenter. Dans **group\_vars/datacenter1**:

```
time_server: 192.168.1.254
```

Dans **group\_vars/datacenter2**:

```
time_server: 192.168.2.254
```

Définition de valeurs différentes pour chaque hôte

Si la valeur à définir varie pour chaque hôte de chaque datacenter, définissez alors les variables dans des fichiers de variable d'hôte distincts. À titre d'exemple, définissez une variable pour **demo1.example.com** dans le fichier **host\_vars/demo1.example.com**:

```
smtp_relay: 192.168.1.20
```



### Note

La documentation Ansible Tower recommande de définir les variables d'inventaire dans l'inventaire proprement dit au lieu d'utiliser des répertoires (sur [https://docs.ansible.com/ansible-tower/latest/html/userguide/best\\_practices.html](https://docs.ansible.com/ansible-tower/latest/html/userguide/best_practices.html)). Aucune raison n'est spécifiée.

Les auteurs de ce cours pensent que cette approche présente des inconvénients. Par exemple, il n'y a aucune modification de contrôle des versions apportée aux champs d'inventaire pour les objets dans Ansible Tower, et la vérification des paramètres peut nécessiter un grand nombre de clics dans l'interface utilisateur.

Cependant, il est vrai que, en définissant les paramètres des variables dans l'inventaire lui-même, les variables seront cohérentes entre plusieurs projets partageant le même inventaire.

## Remplacement de variables du modèle de tâche

Vous pouvez utiliser des variables de playbook pour remplacer les variables d'inventaire. Les variables de modèle de tâche remplacent toutes les autres variables. Il s'agit des *variables supplémentaires*. Les valeurs de variables supplémentaires remplacent tous les autres paramètres pour cette variable.

Dans l'exemple suivant, la variable **prototype\_server** est définie sur **true** en tant que variable supplémentaire dans un modèle de tâche.

The screenshot shows the 'Variable testing' configuration page. In the 'EXTRA VARIABLES' section, under the 'YAML' tab, there is a code editor containing the following YAML:

```

1 ...
2 prototype_server: true
  
```

Figure 4.2: Définition de variables supplémentaires dans un modèle de tâche

Des variables supplémentaires peuvent être utiles lorsque vous devez remplacer la valeur définie par une variable pour une exécution ponctuelle d'un playbook.

## Utilisation de matrices en tant que variables

Au lieu d'attribuer des données de configuration correspondant à cet élément (liste de services, liste d'utilisateurs, etc.) à plusieurs variables, vous pouvez utiliser des *matrices*. L'utilisation de matrices présente l'avantage de permettre à Ansible de les parcourir.

Voyons le snippet suivant :

```
user1_first_name: Bob
user1_last_name: Jones
user1_full_name: Robert Richard Jones
user2_first_name: Anne
user2_last_name: Cook
user2_full_name: Anne Marie Cook
```

Ceci peut être réécrit sous la forme d'une matrice appelée **users**, où chaque utilisateur est un élément de la matrice. Chaque utilisateur est un hachage, ce qui signifie qu'il contient un ensemble de paires clé-valeur. Cela signifie que **users** est une matrice de hachages.

```
users:
bjones:
  first_name: Bob
  last_name: Jones
  full_name: Robert Richard Jones
acook:
  first_name: Anne
  last_name: Cook
  full_name: Anne Marie Cook
```

Vous pouvez ensuite utiliser les variables suivantes pour accéder aux données des utilisateurs :

```
# Returns 'Bob'
users.bjones.first_name

# Returns 'Anne Marie Cook'
users.acook.full_name
```

Parce que la variable est définie en tant que *dictionnaire* Python, une syntaxe alternative est disponible.

```
# Returns 'Bob'
users['bjones']['first_name']

# Returns 'Anne Marie Cook'
users['acook']['full_name']
```



### Important

L'utilisation du point peut poser problème si les noms des clés sont identiques à ceux des attributs ou des méthodes Python, tels que **discard**, **copy**, **add** et autres. L'utilisation de crochets permet d'éviter les conflits et les erreurs.

Les deux syntaxes sont correctes mais, pour détecter plus facilement les erreurs, Red Hat recommande d'utiliser la même syntaxe sur l'ensemble des fichiers d'un même projet Ansible.

## Capture de résultats de tâche à l'aide de variables enregistrées

Vous pouvez utiliser l'instruction **register** pour capturer les résultats d'une tâche. La sortie est alors enregistrée dans une variable temporaire qui pourra servir par la suite dans le playbook à des fins de débogage ou à d'autres fins, par exemple pour une configuration particulière basée sur les résultats d'une tâche.

Le playbook suivant montre comment capturer les résultats d'une tâche à des fins de débogage :

```
---
- name: Installs a package and prints the result
  hosts: demo.example.com

  tasks:
    - name: Copy install package
      win_copy:
        src: files/7z1900-x64.exe
        dest: C:\7z1900-x64.exe

    - name: Install the package
      win_package:
        path: c:\7z1900-x64.exe
        state: present
        product_id: 7-Zip
        arguments: /S
      register: install_result

    - debug:
        var: install_result
```

Lorsque vous exécutez le playbook, Ansible utilise le module **debug** pour afficher la valeur de la variable enregistrée **install\_result**.

```
PLAY [Installs a package and prints the result] ****
TASK [Gathering Facts] ****
ok: [demo.example.com]

TASK [Copy install package] ****
changed: [demo.example.com]

TASK [Install the package] ****
changed: [demo.example.com]

TASK [debug] ****
ok: [demo.example.com] => {
  "install_result": {
    "changed": true,
    "failed": false,
    "rc": 0,
    "reboot_required": false
  }
}
```

```
PLAY RECAP ****
demo.example.com: ok=4 changed=2 unreachable=0 failed=0 skipped=0 rescued=0
  ignored=0
```



## Références

### **Inventaire &— Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/intro\\_inventory.html](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html)

### **Variables &— Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_variables.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html)

### **Ordre des variables : Où placer une variable ?**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_variables.html#variable-precedence-where-should-i-put-a-variable](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable)

### **Syntaxe YAML &— Documentation Ansible**

[https://docs.ansible.com/ansible/latest/reference\\_appendices/YAMLSyntax.html](https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html)

## ► Exercice guidé

# Gestion des variables

Dans cet exercice, vous allez définir des variables et les utiliser dans un playbook.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Définir des variables de groupe dans Ansible.
- Refactoriser un playbook Ansible pour utiliser des variables.
- Exécuter des tâches dans Ansible Tower.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

Assurez-vous que l'hôte **win2.example.com** est listé dans **Default inventory**.



### Note

Cette activité nécessite **win2.example.com** dans l'inventaire **Default inventory**. Vous avez ajouté l'hôte dans un exercice précédent. Si l'hôte n'est pas présent, suivez les instructions de l'exercice guidé *Mise en œuvre de plusieurs plays* pour le créer.

Dans l'activité suivante, vous allez refactoriser un playbook Ansible pour utiliser des variables. Le playbook installe et configure le service IIS avec un site sur les deux hôtes. Un fichier **index.html** par défaut est transféré aux serveurs. La page Web par défaut est accessible à l'adresse <https://gitlab.example.com/student/variables/blob/master/files/index.html>.

- 1. À partir de votre station de travail **Windows**, accédez à votre instance GitLab sur <https://gitlab.example.com> et examinez le contenu du référentiel **variables**. Examinez le playbook **manage.yml** pour passer en revue les tâches que vous allez mettre à jour à une étape ultérieure.
- 1.1. Sur **workstation**, double-cliquez sur l'icône GitLab sur le Bureau pour accéder à votre instance GitLab.
  - 1.2. Connectez-vous à GitLab via LDAP en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Pour accéder au référentiel **variables**, cliquez sur student / variables dans la liste de projets.

Project	Last updated
P student / playbooks	updated a day ago
V student / variables	updated about 2 hours ago
T student / test-playbook	updated a day ago
A student / architecture	updated a day ago
A student / ad	updated a day ago

13. Cliquez sur le playbook **manage.yml** pour le passer en revue. Le playbook installe et configure les services Internet (IIS).

```

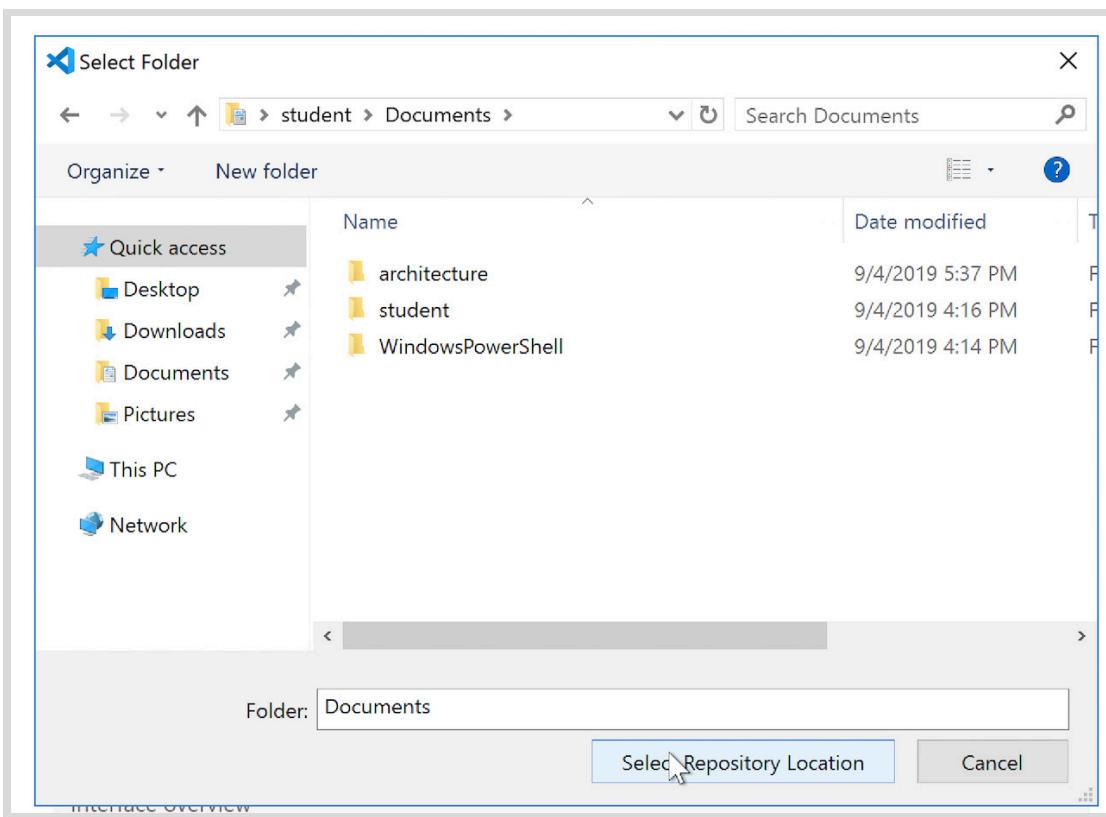
1 - name: Install and Configure an IIS site
2 hosts: all
3 tasks:
4
5 - name: IIS and .Net 4.5 are installed
6   win_feature:
7     # Convert the list to a variable
8     name:
9       - Web-Server
10      - NET-Framework-Core
11      - Web-Asp-Net45
12     include_management_tools: True
13     state: present
14
15 - name: Logs directory is created
16   win_file:
17     path: C:\sites\logs
18     state: directory
19
20 - name: Site directory is created
21   # Use a variable for the site path
22   win_file:
23     path: C:\sites\variables
24     state: directory
25
26 - name: Index page for site is installed
27   # Reuse the variable for the site path
28   win_copy:
29     src: 'files/index.html'
30     dest: C:\sites\variables\index.html
31

```

- 2. Sur **workstation**, ouvrez Visual Studio Code et clonez le référentiel <https://gitlab.example.com/student/variables.git> sur **C:\Users\student\Documents\variables**.

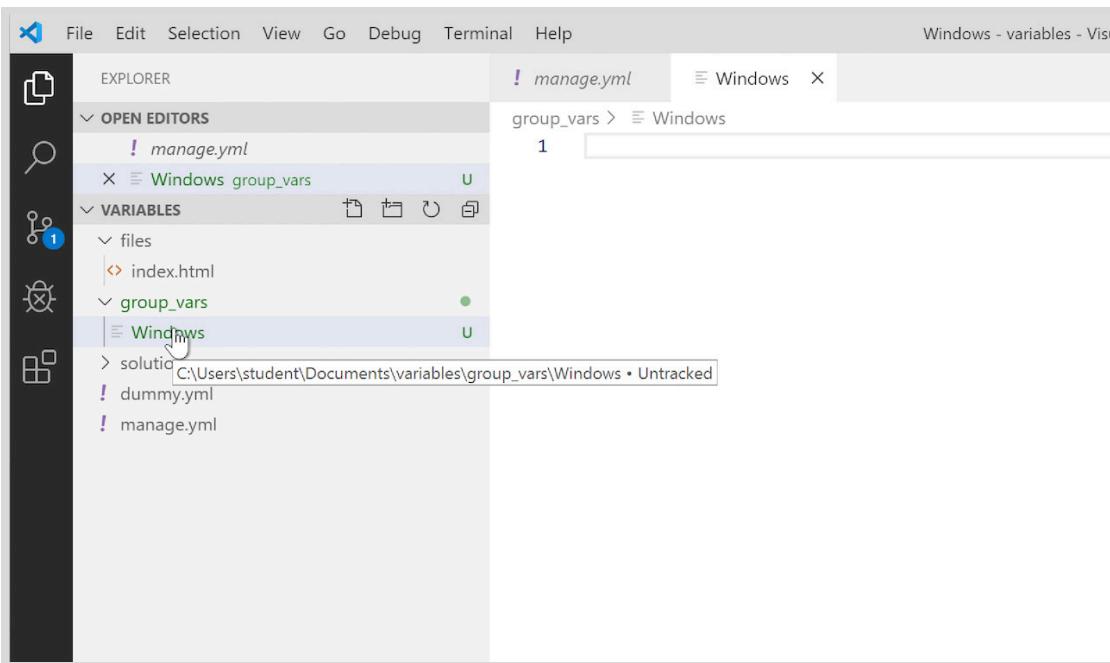
- 2.1. Double-cliquez sur l'icône Visual Studio Code sur le Bureau pour ouvrir l'éditeur Visual Studio Code.
- 2.2. Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **Git: Clone** pour cloner un référentiel.
- 2.3. Utilisez l'URL de référentiel <https://gitlab.example.com/student/variables.git> et appuyez sur **Entrée**.

Sélectionnez le dossier **Documents** et cliquez sur **Select Repository Location** pour cloner le référentiel **variables** dans le dossier **variables**.



2.4. Cliquez sur **Open** pour ajouter le dossier à l'espace de travail.

- ▶ 3. Au cours de cette étape, vous allez créer un nouveau dossier pour les variables de groupe et ajouter un fichier de variables pour le groupe **Windows** qui définit les variables suivantes :
  - La variable **iis\_packages\_name** est une liste qui définit les fonctions à installer. Il contient les trois entrées suivantes :
    - Web-Server
    - NET-Framework-Core
    - Web-Asp-Net45
  - La variable **iis\_site\_path** spécifie le répertoire root de votre site IIS. Attribuez à la variable la valeur **C:\sites\variables**.
  - La variable **iis\_site\_name** spécifie le nom du site IIS (**D0417-variables**).
  - La variable **iis\_site\_port** spécifie le port d'écoute du site sur (**8080**).
- 3.1. Créez un nouveau dossier **C:\Users\student\Documents\variables\group\_vars**. Pour ce faire, cliquez avec le bouton droit de la souris et sélectionnez **New Folder** dans l'arborescence de l'éditeur sur le côté gauche.
- 3.2. Cliquez avec le bouton droit de la souris sur le dossier **group\_vars** pour créer un fichier **Windows** dans celui-ci.



- 3.3. Dans le fichier **Windows**, définissez la variable **iis\_packages\_name** en tant que liste pour indiquer à Ansible la liste des fonctions à installer comme suit :

```
iis_packages_name:  
  - Web-Server  
  - NET-Framework-Core  
  - Web-Asp-Net45
```

- 3.4. Définissez la variable **iis\_site\_path**, qui spécifie le répertoire root de votre site IIS. Attribuez à la variable la valeur **C:\sites\variables**.

```
...output omitted...  
iis_site_path: C:\sites\variables
```

- 3.5. Définissez la variable **iis\_site\_name** avec la valeur **D0417-variables**:

```
...output omitted...  
iis_site_name: "D0417-variables"
```

- 3.6. Définissez la variable **iis\_site\_port** avec la valeur **8080**:

```
...output omitted...  
iis_site_port: 8080
```

- 3.7. Enregistrez vos modifications.

Le fichier terminé doit se présenter comme suit :

```
iis_packages_name:
  - Web-Server
  - NET-Framework-Core
  - Web-Asp-Net45
iis_site_path: C:\sites\variables
iis_site_name: "D0417-variables"
iis_site_port: 8080
```



### Note

Le fichier de variables terminé est accessible à l'adresse **solutions/group\_vars/Windows**.

- 4. Au cours des étapes suivantes, vous allez utiliser Visual Studio Code pour modifier le playbook **manage.yml** et apporter les modifications suivantes :
- Utilisez la variable **iis\_packages\_name** pour mettre à jour le module **win\_feature**.
  - Utilisez la variable **iis\_site\_path** pour mettre à jour le module **win\_file**.
  - Utilisez la variable **iis\_site\_path** pour mettre à jour le module **win\_copy**.
- Le contenu requiert à la fois la variable **iis\_site\_path** et **index.html** puisque la variable ne définit pas le fichier HTML à installer.
- Mettez à jour le module **win\_iis\_website** comme suit :

#### Variables du module **win\_iis\_website**

Paramètre	Variable
name	<b>iis_site_name</b>
port	<b>iis_site_port</b>
hostname	<b>inventory_hostname</b>
physical_path	<b>iis_site_path</b>

- Utilisez la variable **inventory\_hostname** pour mettre à jour le module **win\_firewall\_rule**.

4.1. Cliquez sur le playbook **manage.yml** pour l'ouvrir.

Pour le module **win\_feature**, déplacez la liste des fonctions (listées dans l'entrée **name**) vers une liste et demandez à Ansible d'utiliser la variable **iis\_packages\_name** à la place :

```
...output omitted...
- name: IIS and .Net 4.5 are installed
  win_feature:
    name: "{{ iis_packages_name }}"
    include_management_tools: True
    state: present
```

- 4.2. Pour le module **win\_file** utilisé dans la tâche **Site directory is created**, remplacez la valeur du paramètre **path** par la variable **iis\_site\_path**.

```
...output omitted...
- name: Site directory is created
  win_file:
    path: "{{ iis_site_path }}"
    state: directory
```

- 4.3. Pour le module **win\_copy**, remplacez le chemin d'accès du paramètre **dest** par la variable **iis\_site\_path**.

Le contenu doit utiliser à la fois la variable et **index.html**, car la variable ne définit pas le fichier HTML à installer.



### Important

Utilisez des guillemets simples pour vous assurer que la barre oblique inverse n'est pas interprétée comme une séquence d'échappement.

```
...output omitted...
- name: Index page for site is installed
  # Reuse the variable for the site path
  win_copy:
    src: files/index.html
    dest: '{{ iis_site_path }}\index.html'
```

- 4.4. Pour le module **win\_iis\_website**, remplacez les variables du module comme suit :

#### Variables du module **win\_iis\_website**

Paramètre	Variable
name	<b>iis_site_name</b>
port	<b>iis_site_port</b>
hostname	<b>inventory_hostname</b>
physical_path	<b>iis_site_path</b>

Le contenu doit correspondre à ce qui suit :

```
...output omitted...
- name: IIS site is created
  win_iis_website:
    name: "{{ iis_site_name }}"
    state: started
    port: "{{ iis_site_port }}"
    ip: "*"
    hostname: "{{ inventory_hostname }}"
```

```
application_pool: DefaultAppPool
physical_path: "{{ iis_site_path }}"
parameters: logfile.directory:C:\sites\logs
```

- 4.5. Pour le module **win\_firewall\_rule**, remplacez la valeur du paramètre **localport** par la variable **iis\_site\_port**. Le contenu doit correspondre à ce qui suit :

```
...output omitted...
- name: Firewall rule is enabled
  win_firewall_rule:
    name: HTTP
    localport: "{{ iis_site_port }}"
    action: allow
    direction: in
    protocol: tcp
    state: present
    enabled: yes
```

Le fichier final doit ressembler à ce qui suit :

```
- name: Install and Configure an IIS site
hosts: all
tasks:
  - name: IIS and .Net 4.5 are installed
    win_feature:
      name: "{{ iis_packages_name }}"
      include_management_tools: True
      state: present

  - name: Logs directory is created
    win_file:
      path: C:\sites\logs
      state: directory

  - name: Site directory is created
    win_file:
      path: "{{ iis_site_path }}"
      state: directory

  - name: Index page for site is installed
    win_copy:
      src: files/index.html
      dest: '{{ iis_site_path }}\index.html'

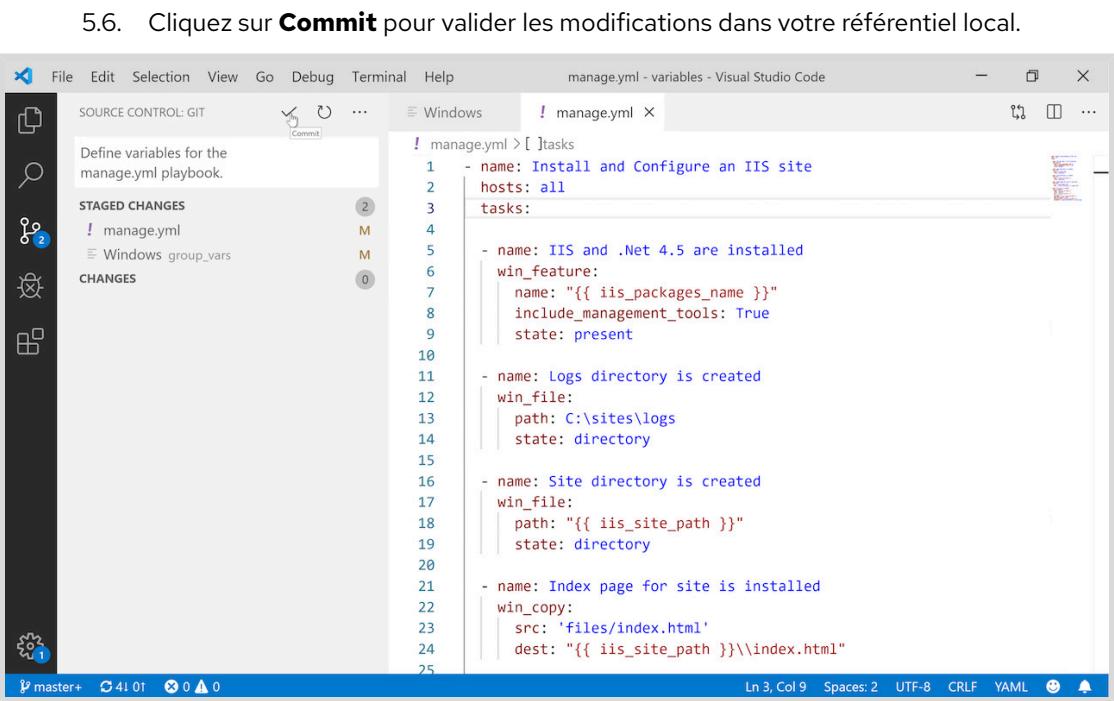
  - name: IIS site is created
    win_iis_website:
      name: "{{ iis_site_name }}"
      state: started
      port: "{{ iis_site_port }}"
      ip: "*"
      hostname: "{{ inventory_hostname }}"
      application_pool: DefaultAppPool
```

```

physical_path: "{{ iis_site_path }}"
parameters: logfile.directory:C:\sites\logs

- name: Firewall rule is enabled
  win_firewall_rule:
    name: HTTP
    localport: "{{ iis_site_port }}"
    action: allow
    direction: in
    protocol: tcp
    state: present
    enabled: yes
  
```

- ▶ 5. Enregistrez vos modifications, validez le fichier à l'aide du message **Define variables for the manage.yml playbook**, et poussez vos modifications vers le référentiel distant.
- 5.1. Enregistrez vos modifications.
  - 5.2. À partir de la barre latérale de Visual Studio Code, sélectionnez **Source Control** pour accéder à la liste des modifications.
  - 5.3. Survolez le playbook **manage.yml** pour accéder aux options disponibles pour le fichier. Cliquez sur **Stage Changes** pour ajouter vos modifications à l'index.
  - 5.4. Répétez l'étape pour le fichier **Windows**.
  - 5.5. Dans le fichier texte **Message**, saisissez le message **Define variables for the manage.yml playbook**.

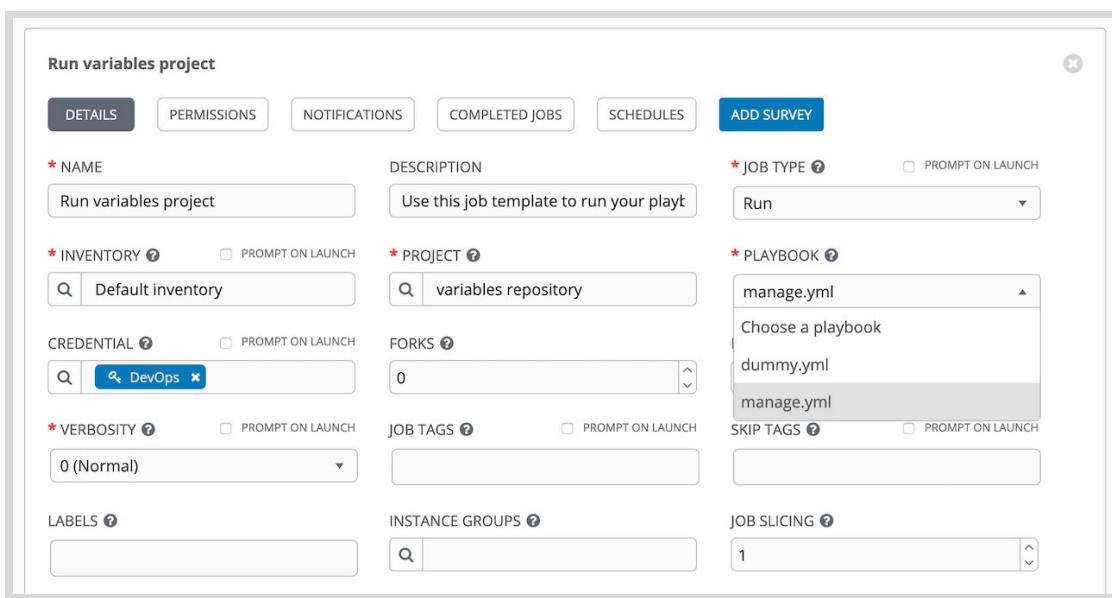


- 5.6. Cliquez sur **Commit** pour valider les modifications dans votre référentiel local.
- 5.7. Pour pousser vos modifications locales vers le référentiel distant, cliquez sur le bouton de synchronisation en bas de l'éditeur. Cela indique à Visual Studio Code de synchroniser votre référentiel avec la copie distante.

- 6. Dans Ansible Tower, modifiez la tâche **Run variables project** pour utiliser **manage.yml** comme playbook.

Après avoir mis à jour le modèle de tâche, exécutez-le et passez en revue la sortie de la tâche. La sortie de la tâche indique qu'Ansible a installé et configuré le serveur Web sur les deux hôtes de l'inventaire par défaut. Il a ensuite personnalisé la page Web pour chaque hôte.

- 6.1. Accédez à Ansible Tower en double-cliquant sur Ansible Tower à partir de votre Bureau. Connectez-vous à Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe
- 6.2. Dans Ansible Tower, cliquez sur **Templates**, puis sur le modèle de tâche **Run variables project** dans le menu latéral.
- 6.3. Cliquez sur le modèle de tâche pour le modifier. Sélectionnez **manage.yml** pour le champ **PLAYBOOK** et cliquez sur **SAVE** pour mettre à jour le modèle de tâche.



- 6.4. Cliquez sur **LAUNCH** pour exécuter une tâche à partir du modèle de tâche.
- 6.5. L'action vous redirige vers la sortie de la tâche. Le résultat indique qu'Ansible est en mesure d'installer et de démarrer le serveur Web sur **win1** et **win2**, et de créer le site personnalisé **DO417-variables** sur chaque serveur.  
Le play ouvre également le pare-feu pour autoriser les connexions sur le port 8080.

The screenshot shows the Ansible Tower interface. On the left, the 'DETAILS' panel displays job information: STATUS (Successful), STARTED (9/5/2019 3:43:50 PM), FINISHED (9/5/2019 3:44:43 PM), JOB TEMPLATE (Run variables project), JOB TYPE (Run), LAUNCHED BY (admin), INVENTORY (Default inventory), PROJECT (variables repository), REVISION (ba31f43), PLAYBOOK (manage.yml), CREDENTIAL (DevOps), ENVIRONMENT (/var/lib/awx/venv/ansible), EXECUTION NODE (localhost), INSTANCE GROUP (tower), and EXTRA VARIABLES (YAML, JSON). On the right, the 'Run variables project' panel shows the command output with tasks 22 through 36. Tasks 22 and 23 show 'changed' status for win2.example.com and win1.example.com respectively. Task 25 is a 'TASK [IIS site is created]' with a timestamp of 15:44:33. Task 29 is a 'TASK [Firewall rule is enabled]' with a timestamp of 15:44:38. Task 33 is a 'PLAY RECAP' with a timestamp of 15:44:42, showing results for win1.example.com and win2.example.com.

```

22 changed: [win2.example.com]
23 changed: [win1.example.com]
24
25 TASK [IIS site is created] 15:44:33
26 changed: [win2.example.com]
27 changed: [win1.example.com]
28
29 TASK [Firewall rule is enabled] 15:44:38
30 changed: [win1.example.com]
31 changed: [win2.example.com]
32
33 PLAY RECAP 15:44:42
34 win1.example.com : ok=7    changed=5    unreachable=0    fail=0
35 win2.example.com : ok=7    changed=5    unreachable=0    fail=0
36

```

- 7. Sur **workstation**, ouvrez Chrome et accédez à <http://win1.example.com:8080>. Assurez-vous que la page d'accueil indique ce qui suit :

## This is the default page of the variables IIS site

### This page has been provisioned by Ansible

To make changes to this page, edit the **files/index.html** file and rerun the job in Ansible Tower.

List of the modules used to create this site:

<b>win_feature</b>	Used to install IIS features
<b>win_file</b>	Used to create the directory structure
<b>win_copy</b>	Used to install that file
<b>win_iis_website</b>	Used to configure the IIS website
<b>win_firewall_rule</b>	Used to open the firewall

- 8. Sur **workstation**, ouvrez Chrome et accédez à `http://win2.example.com:8080` et assurez-vous que la page Web corresponde à celle renvoyée par `win1.example.com`.

 **Note**

Si vous n'êtes pas en mesure d'accéder aux pages Web et que vous devez nettoyer les ressources sur les hôtes, exécutez le playbook Ansible `remove-iis.yml` disponible dans le répertoire de projet **variables**. Le playbook supprime le site IIS et la règle de pare-feu.

Après avoir nettoyé les ressources, modifiez votre playbook et exécutez-le à nouveau.

L'exercice guidé est maintenant terminé.

# Gestion des secrets

---

## Résultats

À la fin de cette section, vous serez en mesure de décrire les approches de la protection des données sensibles utilisées par les playbooks.

## Protection des données sensibles

Il est possible qu'Ansible ait besoin d'accéder à des données sensibles telles que des mots de passe ou clés API pour configurer les hôtes gérés. Normalement, vous pouvez stocker ces informations sous forme de texte clair dans les variables d'inventaire ou d'autres fichiers Ansible. Dans ce cas, toutefois, n'importe quel utilisateur ayant accès aux fichiers Ansible ou à un système de contrôle de version qui stocke les fichiers Ansible a accès à ces données sensibles. Cela pose un problème de sécurité évident.

Vous pouvez gérer ce risque de plusieurs façons. Chacune présente ses avantages et ses inconvénients. Voici quelques façons :

- Protéger les fichiers structurés à l'aide de l'outil Ansible Vault fourni avec Ansible sur un système Linux ou un conteneur.
- Protéger les fichiers structurés à l'aide de l'outil Ansible Vault fourni avec Ansible sur un système Windows installé avec le sous-système Windows pour Linux (WSL).
- Protéger les fichiers structurés avec Ansible Vault en utilisant des *applets de commande* PowerShell non officielles de Jordan Borean sur un système Windows.
- Utiliser des informations d'identification personnalisées d'Ansible Tower.
- Utiliser un système de gestion de secret tiers.

## Protection des secrets avec Ansible Vault sur Linux

Ansible Vault, fourni avec les outils de ligne de commande Ansible, peut être utilisé pour chiffrer et déchiffrer n'importe quel fichier de données structurées utilisé par Ansible. Cela inclut les fichiers de variables d'hôte et de groupe, ainsi que les variables chargées par les directives **include\_vars** ou **vars\_files**. Certains modules Ansible, y compris **template**, peuvent accepter un fichier chiffré par Ansible Vault en tant que valeur pour sa directive **src** et déchiffrer le fichier avant de le placer à la destination sur l'hôte cible.

La principale limitation d'Ansible Vault est que les outils de ligne de commande Ansible ne sont pas pris en charge actuellement sur les systèmes d'exploitation Microsoft Windows. Cela signifie qu'il peut s'avérer difficile de préparer le fichier chiffré avec Ansible Vault si vous travaillez sur une station de travail Windows. Cependant, lorsque vous avez un fichier chiffré, Ansible Tower peut le déchiffrer lorsque vous exécutez la tâche à condition que vous fournissiez les informations d'identification Vault correctes en tant que l'une des informations d'identification du modèle de tâche.

**Important**

Ansible Vault n'implémente pas ses propres fonctions cryptographiques, mais utilise plutôt une boîte à outils Python externe. Les fichiers sont protégés par un chiffrement symétrique AES256 avec un mot de passe comme clé secrète. Notez que la façon dont cela est réalisé n'a pas fait l'objet d'un audit formel de la part d'un tiers.

## Chiffrement de fichiers sur la ligne de commande Linux

Pour chiffrer des fichiers avec Ansible Vault sur un système Linux, vous devez installer un serveur Linux ou une machine virtuelle avec les outils de ligne de commande Ansible. Pour plus d'informations sur la façon de procéder, reportez-vous à l'adresse [https://docs.ansible.com/ansible/latest/installation\\_guide/index.html](https://docs.ansible.com/ansible/latest/installation_guide/index.html). Vous devez également cloner le référentiel Git contenant votre projet Ansible sur le système Linux.

Pour chiffrer un fichier existant, vous pouvez utiliser la commande **ansible-vault encrypt** :

```
[student@demo project]$ ansible-vault encrypt secret.yml
New Vault password:
Confirm New Vault password:
Encryption successful
[student@demo project]$
```

Vous pouvez utiliser la commande **ansible-vault rekey** pour modifier le mot de passe du fichier et la commande **ansible-vault decrypt** pour déchiffrer définitivement le fichier.

L'inconvénient de cette approche dans un environnement axé sur Windows est que vous devez modifier vos fichiers secrets sur une station de travail Linux.

## Exécution de tâches avec des fichiers protégés par Ansible Vault

Sur Ansible Tower, pour déchiffrer un fichier Ansible protégé par Ansible Vault lors du lancement d'une tâche, le modèle de tâche doit inclure les mots de passe des fichiers protégés par Ansible Vault en tant qu'une ou plusieurs informations d'identification Vault.

Les informations d'identification Vault sont créées dans Ansible Tower de la même façon que les informations d'identification de machine, sauf que vous sélectionnez **Vault** comme type d'informations d'identification et que vous fournissez le mot de passe Vault aux informations d'identification.

The screenshot shows the 'API Vault Credential' configuration dialog. It has tabs for 'DETAILS' and 'PERMISSIONS'. Under 'DETAILS', there are fields for 'NAME' (API Vault Credential), 'DESCRIPTION' (empty), 'ORGANIZATION' (Default), 'CREDENTIAL TYPE' (Vault), 'VAULT PASSWORD' (ENCRYPTED), and 'VAULT IDENTIFIER' (empty). At the bottom are 'CANCEL' and 'SAVE' buttons.

Figure 4.11: Informations d'identification Vault

Incluez les informations d'identification Vault et les informations d'identification de la machine dans le modèle de tâche :

The screenshot shows the 'Test API Template' configuration dialog. It has tabs for 'DETAILS', 'PERMISSIONS', 'NOTIFICATIONS', 'COMPLETED JOBS', 'SCHEDULES', and 'ADD SURVEY'. Under 'DETAILS', there are fields for 'NAME' (Test API Template), 'DESCRIPTION' (empty), 'JOB TYPE' (Run), 'INVENTORY' (Demo Inventory), 'PROJECT' (Test API Project), 'PLAYBOOK' (api.yml), 'CREDENTIAL' (API Vault Credential, Demo Credential), 'VERBOSITY' (0 (Normal)), 'FORKS' (1), 'LIMIT' (empty), 'JOB TAGS' (empty), 'SKIP TAGS' (empty), and 'JOB SLICING' (empty). At the bottom are 'LARVIC', 'INSTANCE GROUPS', and 'JOB SLICING' buttons.

Figure 4.12: Modèle de tâche qui utilise des informations d'identification Vault

## Pratiques recommandées en matière de gestion des fichiers de variables

Pour simplifier la gestion, il est logique de configurer votre projet Ansible de sorte que les variables sensibles et toutes les autres soient conservées dans des fichiers distincts. Vous pouvez protéger les fichiers contenant des variables sensibles à l'aide de la commande **ansible-vault**.

Pour rappel, la meilleure façon de gérer des variables d'hôte et de groupe consiste à créer des répertoires au niveau du playbook. Le répertoire **group\_vars** contient généralement des fichiers de variables dont les noms correspondent aux groupes d'hôtes auxquels ils s'appliquent. Le répertoire **host\_vars** contient généralement des fichiers de variables dont les noms correspondent aux noms des hôtes gérés auxquels ils s'appliquent.

Cependant, au lieu d'utiliser les fichiers dans **group\_vars** ou **host\_vars**, vous pouvez également utiliser des répertoires pour chaque groupe d'hôtes ou hôte géré. Ces répertoires peuvent alors contenir plusieurs fichiers de variables, tous étant utilisés par le groupe d'hôtes ou l'hôte géré. Par exemple, dans le répertoire de projet suivant pour **playbook.yml**, les membres du groupe d'hôtes **webservers** utilisent les variables du fichier **group\_vars/webservers/vars**, et **demo.example.com** utilise les variables présentes à la fois dans **host\_vars/demo.example.com/vars** et **host\_vars/demo.example.com/vault** :

```
.
├── ansible.cfg
├── group_vars
│   └── webservers
│       └── vars
└── host_vars
    └── demo.example.com
        ├── vars
        └── vault
└── inventory
└── playbook.yml
```

Dans ce scénario, l'avantage est que vous pouvez placer la plupart des variables de **demo.example.com** dans le fichier **vars**, et vous pouvez protéger les variables sensibles en les plaçant séparément dans le fichier **vault**. Vous pouvez ensuite utiliser **ansible-vault** pour chiffrer **vault** et conserver le fichier **vars** en texte clair.

Il n'y a rien de particulier à signaler concernant les noms de fichiers utilisés dans cet exemple dans le répertoire **host\_vars/demo.example.com**. Ce répertoire peut contenir davantage de fichiers, certains chiffrés par Ansible Vault et d'autres non.

Vous pouvez également protéger les variables de playbook (contrairement aux variables d'inventaire) avec Ansible Vault. Vous pouvez placer les variables de playbook sensibles dans un fichier distinct, chiffré par Ansible Vault et inclus dans le playbook via une directive **vars\_files**. Cela peut s'avérer utile dans la mesure où les variables de playbook sont prioritaires sur les variables d'inventaire.

Si vous utilisez plusieurs mots de passe de Vault avec votre playbook, assurez-vous qu'un ID Vault est attribué à chaque fichier chiffré et que vous entrez le mot de passe correspondant à cet ID Vault lors de l'exécution du playbook. Cela garantit que le mot de passe correct est sélectionné en premier lors du déchiffrement du fichier chiffré par Vault, ce qui est plus rapide que d'obliger Ansible à essayer tous les mots de passe de Vault que vous avez fournis jusqu'à ce qu'il trouve le bon.

## Utilisation d'Ansible Vault sous Windows

L'outil **ansible-vault** n'est pris en charge que sous Linux. Cependant, certaines options ne sont pas prises en charge pour Windows.

Vous pouvez installer les outils Ansible dans l'environnement de sous-système Windows pour Linux (WSL) sur une station de travail Windows. Cette méthode peut fonctionner, mais elle n'est pas prise en charge ni recommandée par Red Hat.

L'un des développeurs Ansible, Jordan Borean, a écrit des applets de commande PowerShell non officiels que vous pouvez utiliser pour chiffrer et déchiffrer des fichiers Ansible Vault. Le référentiel GitHub pour son code se trouve à l'adresse <https://github.com/jborean93/PowerShell-AnsibleVault>. Ce code est écrit par un membre de la communauté Open Source, et n'est pas non plus pris en charge par Red Hat.

**Important**

Aucune de ces options n'est prise en charge par Red Hat, mais nous fournissons des informations à leur sujet pour votre référence.

Des liens vers des lectures supplémentaires sont disponibles dans la section Références ci-dessous.

## Stockage d'informations sensibles dans les informations d'identification personnalisées

Une nouvelle fonction dans Ansible Tower 3.2 est les *types d'informations d'identification personnalisées*. Vous pouvez utiliser cette fonction pour créer des types d'informations d'identification personnalisés, dont les instances sont stockées sous forme chiffrée dans la base de données Ansible Tower.

Les informations d'identification personnalisées sont composées d'*entrées* et d'*injecteurs*. Les entrées définissent les champs que les informations d'identification contiennent, tel qu'un nom d'utilisateur ou un jeton. Les injecteurs définissent la manière dont les informations sont disponibles pour une utilisation dans les playbooks.

Même s'il s'agit d'informations d'identification, vous n'êtes pas limité à leur utilisation à des fins d'authentification. Par exemple, vous pouvez stocker les informations relatives aux logiciels achetés, telles qu'une clé de licence, un numéro de série ou un code d'activation. Les images suivantes présentent des informations d'identification personnalisées simples utilisées pour stocker un nom d'utilisateur et un mot de passe pour l'authentification proxy.

The screenshot shows the 'CREATE CREDENTIAL TYPE' dialog in Ansible Tower. On the left is a sidebar with 'Credential Types' selected. The main area has a title 'NEW CREDENTIAL TYPE' with fields for 'NAME' (set to 'Demo') and 'DESCRIPTION' (set to 'Demonstration credential type'). Below this are two sections: 'INPUT CONFIGURATION' (YAML) and 'INJECTOR CONFIGURATION' (YAML). The 'INPUT CONFIGURATION' section contains the following YAML:

```

1 ---
2 fields:
3   - type: string
4     id: proxy_username
5     label: Proxy Username
6   - type: string

```

The 'INJECTOR CONFIGURATION' section contains the following YAML:

```

1 ---
2 extra_vars:
3   proxy_user: "{{ proxy_username }}"
4   proxy_pass: "{{ proxy_password }}"

```

At the bottom right are 'CANCEL' and 'SAVE' buttons.

**Figure 4.13: Création d'un nouveau type d'informations d'identification personnalisées**

The screenshot shows the 'CREATE CREDENTIAL' dialog in Ansible Tower. On the left is a sidebar with 'Views', 'Dashboard', 'Jobs', 'Schedules', 'My View', 'Resources', 'Templates', 'Credentials' (which is selected), 'Projects', 'Inventories', and 'Inventory Scripts'. The main area has tabs 'DETAILS' and 'PERMISSIONS' (disabled). Under 'DETAILS', there are fields for 'NAME' (Production Proxy), 'DESCRIPTION' (Production proxy credentials), 'ORGANIZATION' (Default), 'CREDENTIAL TYPE' (set to 'Demo'), and 'TYPE DETAILS' which includes 'PROXY USERNAME' (prod\_user) and 'PROXY PASSWORD' (redacted). At the bottom are 'CANCEL' and 'SAVE' buttons.

Figure 4.14: Création de nouvelles informations d'identification à l'aide d'un type personnalisé

## Intégration de systèmes de gestion de secret tiers

De nombreuses entreprises disposent déjà d'un système centralisé de gestion des secrets. Ansible Tower 3.5 et versions ultérieures peuvent s'intégrer à plusieurs systèmes de gestion secrets par le biais de plug-ins d'informations d'identification.

### Systèmes de gestion des secrets pris en charge

- CyberArk Application Identity Manager
- CyberArk Conjur
- HashiCorp Vault Key-Value Store
- HashiCorp Vault SSH Secrets Engine
- Système de gestion de clés Microsoft Azure

L'un des avantages d'un service secret centralisé est que les informations sensibles sont disponibles pour les membres de l'organisation sans avoir besoin d'accéder à Ansible Tower.

L'image suivante montre un exemple des champs supplémentaires requis lors de la création d'une mise en correspondance des informations d'identification avec un service secret externe.

This screenshot shows the 'CREATE CREDENTIAL' dialog with a different configuration. The 'CREDENTIAL TYPE' is set to 'HashiCorp Vault Secret Lookup'. In the 'TYPE DETAILS' section, there are fields for 'SERVER URL' (redacted), 'TOKEN' (redacted), and 'API VERSION' (set to 'v1'). The 'TEST' button is visible at the bottom right.

Figure 4.15: Création de nouvelles informations d'identification mises en correspondance avec un service secret externe

Ansible récupère les informations d'identification juste avant l'exécution du playbook.



## Références

### **Vault — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_vault.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_vault.html)

### **Variables et coffres-forts — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_best\\_practices.html#best-practices-for-variables-and-vaults](https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html#best-practices-for-variables-and-vaults)

### **PowerShell-AnsibleVault**

<https://github.com/jborean93/PowerShell-AnsibleVault>

### **Can Ansible run on windows? — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/windows\\_faq.html#can-ansible-run-on-windows](https://docs.ansible.com/ansible/latest/user_guide/windows_faq.html#can-ansible-run-on-windows)

### **Secret Management System — Documentation Ansible**

[https://docs.ansible.com/ansible-tower/latest/html/userguide/credential\\_plugins.html#ug-credential-plugins](https://docs.ansible.com/ansible-tower/latest/html/userguide/credential_plugins.html#ug-credential-plugins)

### **Ansible Tower Feature Spotlight: Custom Credentials**

<https://www.ansible.com/blog/ansible-tower-feature-spotlight-custom-credentials>

## ► Quiz

# Gestion des secrets

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. Parmi les méthodes suivantes, lesquelles sont des méthodes non prises en charge pour exécuter Ansible Vault sous Windows ? (Choisissez-en deux.)
  - a. Ansible PyInstaller
  - b. Module PowerShell Ansible
  - c. AnsiblePE
  - d. Bash pour Windows
  - e. Sous-système Windows pour Linux
- ▶ 2. Quel est le cipher de chiffrement utilisé par Ansible Vault ?
  - a. Blowfish
  - b. 3DES
  - c. RC4
  - d. AES256
  - e. SHA512
- ▶ 3. Quels sont les deux groupes d'informations requis pour un type d'informations d'identification personnalisé ? (Choisissez-en deux.)
  - a. Les entrées
  - b. Les pipes
  - c. Les extracteurs
  - d. Les exportations
  - e. Les injecteurs
- ▶ 4. Parmi les cinq systèmes de gestion des secrets suivants, lesquels sont pris en charge par Ansible Tower ? (Choisissez cinq réponses.)
  - a. Système de gestion de clés Microsoft Azure
  - b. HashiCorp Vault Key-Value Store
  - c. Cloudflare Red October
  - d. CyberArk Conjur
  - e. Lyft Confidant
  - f. CyberArk Application Identity Manager
  - g. HashiCorp Vault SSH Secrets Engine
  - h. Pinterest Knox

## ► Solution

# Gestion des secrets

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. Parmi les méthodes suivantes, lesquelles sont des méthodes non prises en charge pour exécuter Ansible Vault sous Windows ? (Choisissez-en deux.)
  - a. Ansible PyInstaller
  - b. Module PowerShell Ansible
  - c. AnsiblePE
  - d. Bash pour Windows
  - e. Sous-système Windows pour Linux
- ▶ 2. Quel est le cipher de chiffrement utilisé par Ansible Vault ?
  - a. Blowfish
  - b. 3DES
  - c. RC4
  - d. AES256
  - e. SHA512
- ▶ 3. Quels sont les deux groupes d'informations requis pour un type d'informations d'identification personnalisé ? (Choisissez-en deux.)
  - a. Les entrées
  - b. Les pipes
  - c. Les extracteurs
  - d. Les exportations
  - e. Les injecteurs
- ▶ 4. Parmi les cinq systèmes de gestion des secrets suivants, lesquels sont pris en charge par Ansible Tower ? (Choisissez cinq réponses.)
  - a. Système de gestion de clés Microsoft Azure
  - b. HashiCorp Vault Key-Value Store
  - c. Cloudflare Red October
  - d. CyberArk Conjur
  - e. Lyft Confidant
  - f. CyberArk Application Identity Manager
  - g. HashiCorp Vault SSH Secrets Engine
  - h. Pinterest Knox

# Gestion des faits

---

## Résultats

Au terme de cette section, vous serez en mesure de référencer les données sur les hôtes gérés à l'aide de faits Ansible et de configurer des faits personnalisés sur des hôtes gérés.

## Description des faits Ansible

Les faits *Ansible* sont des variables qu'Ansible détecte automatiquement sur un hôte géré. Les faits contiennent des informations propres à l'hôte que vous pouvez utiliser comme des variables régulières dans les plays, les conditions, les boucles ou toute autre instruction liée à une valeur collectée à partir d'un hôte géré.

Les faits collectés à partir des hôtes gérés peuvent inclure :

- le nom de l'hôte ;
- les noms d'interfaces réseau ;
- les adresses IP ;
- la version du système d'exploitation ;
- différentes variables d'environnement ;
- le nombre d'UC ;
- la mémoire totale ou libre ;
- l'espace disque disponible.

Les faits Ansible constituent un moyen pratique d'obtenir l'état d'un nœud géré et de décider de l'action à effectuer en fonction de cet état. Par exemple :

- Vous pouvez décider de redémarrer un serveur via une tâche conditionnelle, qui s'exécute ou non en fonction d'un fait contenant la version actuelle du noyau de l'hôte géré.
- Vous pouvez personnaliser une base de données en fonction de la mémoire disponible signalée par un fait.
- Vous pouvez définir l'adresse IPv4 en fonction de la valeur d'un fait.

Généralement, chaque play exécute automatiquement le module **setup** avant la première tâche afin de rassembler les faits. Il s'agit de la tâche **Gathering Facts** dans Ansible 2.3 et version ultérieure, ou simplement de **setup** dans les versions antérieures d'Ansible. Par défaut, vous n'avez pas besoin d'avoir une tâche pour exécuter **setup** dans votre play. Il est normalement exécuté automatiquement pour vous.

Pour déterminer les faits rassemblés par Ansible pour vos hôtes gérés, vous pouvez exécuter un court playbook qui rassemble des faits et utilise le module **debug** pour imprimer la valeur de la variable **ansible\_facts**.

```
- name: Fact dump
hosts: all
tasks:
  - name: Print all facts
    debug:
      var: ansible_facts
```

Lorsque vous exéutez le playbook, la sortie de la tâche affiche les faits :

```
PLAY [Fact dump] ****
TASK [Gathering Facts] ****
ok: [win1.example.com]

TASK [Print all facts] ****
ok: [win1.example.com] => {
  "ansible_facts": {
    "lastboot": "2019-08-21 03:14:00Z",
    "owner_contact": "",
    "domain": "example.com",
    "distribution_major_version": "10",
    "swaptotal_mb": 0,
    "kernel": "10.0.17763.0",
    "windows_domain_member": true,
    "system_vendor": "Amazon EC2",
    "module_setup": true,
    "processor_count": 1,
    "_facts_gathered": true,
    "gather_subset": [
      "interfaces": [
        {
          "macaddress": "06:77:83:A2:D5:29",
          "default_gateway": "172.25.250.1",
          "interface_name": "Amazon Elastic Network Adapter",
          "connection_name": "Ethernet 3",
          "dns_domain": "us-west-1.compute.internal",
          "interface_index": 5
        }
      ],
      "memtotal_mb": 4036,
      "windows_domain": "example.com",
      "os_family": "Windows",
      "processor_cores": 1,
      ...output omitted...
    ]
  }
}
```

Le playbook affiche le contenu de la variable **ansible\_facts** au format JSON sous la forme d'un hachage (dictionnaire) de variables. Vous pouvez parcourir la sortie pour voir les faits qui sont rassemblés, et pour rechercher ceux que vous souhaitez utiliser dans vos playbooks.

Le tableau suivant montre des faits collectés qu'Ansible est susceptible de collecter à partir d'un hôte géré et qui peuvent être utiles dans un playbook :

## Exemples de faits Ansible

Fait	Variable
Nom d'hôte abrégé	<code>ansible_facts['hostname']</code>
Nom de domaine complet	<code>ansible_facts['fqdn']</code>
Nombre de processeurs	<code>ansible_facts['processor_count']</code>
Noms de toutes les interfaces réseau	<code>ansible_facts['interfaces']</code>
Mémoire système totale	<code>ansible_facts['memtotal_mb']</code>
Version du noyau en cours d'exécution	<code>ansible_facts['kernel']</code>



### Note

N'oubliez pas que lorsque la valeur d'une variable est un hachage, deux syntaxes, vous pouvez utiliser l'une ou l'autre des syntaxes pour récupérer la valeur. Pour prendre deux exemples du tableau précédent :

- `ansible_facts['date_time']['month']` peut aussi être écrit `ansible_facts.date_time.month`
- `ansible_facts['env']['SystemDrive']` peut aussi être écrit `ansible_facts.env.SystemDrive`

Lors de l'utilisation de faits dans un playbook, Ansible remplace de manière dynamique le nom de la variable du fait par sa valeur correspondante :

```
---
- name: Fact usage
hosts: all
tasks:
  - name: Prints various Ansible facts
    debug:
      msg: >
        The current time on {{ ansible_facts['fqdn'] }}
        is {{ ansible_facts['date_time']['time'] }}
```

La sortie suivante montre comment Ansible peut interroger le nœud géré et utilise de manière dynamique les informations système pour mettre à jour la variable. Vous pouvez également utiliser les faits pour créer des groupes dynamiques d'hôtes qui correspondent à des critères particuliers.

```
PLAY ****
TASK [Gathering Facts] ****
ok: [win1.example.com]

TASK [Prints various Ansible facts] ****
ok: [win1.example.com] => {
```

```

        "msg": "The current time on win1.example.com is 13:16:45"
    }

PLAY RECAP ****
win1.example.com      : ok=2      changed=0      unreachable=0      failed=0

```

## Gestion des faits Ansible injectés sous forme de variables

Avant Ansible 2.5, les faits ont été injectés sous forme de variables individuelles précédées de la chaîne **ansible\_** au lieu de faire partie de la variable **ansible\_facts**. Par exemple, le fait **ansible\_facts['distribution']** aurait été appelé **ansible\_distribution**.

Beaucoup de playbooks plus anciens utilisent encore des faits injectés sous forme de variables, au lieu de la nouvelle syntaxe où ils apparaissent sous la variable **ansible\_facts**.

Le tableau suivant compare les anciens et les nouveaux noms de faits.

### Comparaison de certains noms de faits Ansible

Forme ansible_facts	Ancienne forme de variable de fait
<code>ansible_facts['hostname']</code>	<code>ansible_hostname</code>
<code>ansible_facts['fqdn']</code>	<code>ansible_fqdn</code>
<code>ansible_facts['interfaces']</code>	<code>ansible_interfaces</code>
<code>ansible_facts['kernel']</code>	<code>ansible_kernel</code>

**Important**

Actuellement, Ansible reconnaît à la fois le nouveau système de nommage des faits (à l'aide de **ansible\_facts**) et l'ancien système de nommage des « faits injectés en tant que variables séparées » des versions antérieures à la 2.5.

La valeur par défaut de **inject\_facts\_as\_vars** va probablement changer à **false** dans une future version d'Ansible. Si elle est définie sur **false**, vous ne pouvez référencer que des faits Ansible en utilisant le nouveau système de nommage **ansible\_facts.\***. Dans ce cas, toute tentative de référence à des faits au moyen de l'ancien espace de noms entraînera l'erreur suivante :

```
...output omitted...
TASK [Show me the facts] ****
fatal: [demo.example.com]: FAILED! => {"msg": "The task includes an option
with an \
undefined variable. The error was: 'ansible_distribution' is undefined\n\n
The error appears to have been in '/home/student/demo/playbook.yml': line 5, \
column 7, but may\nbe elsewhere in the file depending on the exact syntax
problem.\n
\nThe offending line appears to be:\n\n  tasks:\n    - name: Show me the facts\n      ^ here\n    }
...output omitted...
```

## Désactivation de la collecte de faits

Parfois, vous ne voulez pas collecter les faits pour votre play. Plusieurs raisons peuvent justifier ce cas de figure. Dans certains cas, il se peut que vous ne souhaitez pas collecter des faits pour votre play, par exemple si vous ne les utilisez pas et que vous souhaitez accélérer le play ou réduire la charge sur les hôtes gérés. Les hôtes gérés ne peuvent peut-être pas exécuter le module **setup** pour une raison quelconque, ou ils doivent installer certains logiciels prérequis avant de collecter des faits.

Afin de désactiver la collecte des faits pour un play, définissez le mot-clé **gather\_facts** sur **no** ou **false** :

```
---
- name: This play gathers no facts automatically
  hosts: large_farm
  gather_facts: no
```

Même si **gather\_facts: no** est défini pour un play, vous pouvez manuellement collecter les faits à tout moment en exécutant une tâche qui utilise le module **setup** :

```
tasks:
  - name: Manually gather facts
    setup:
    ...output omitted...
```

## Obtention d'informations à l'aide des variables magiques

Certaines variables ne sont pas des faits, ou elles sont configurées par le biais du module **setup**, mais elles sont définies automatiquement par Ansible. Ces *variables magiques* peuvent également être utilisées pour obtenir des informations spécifiques sur un hôte géré particulier.

Parmi les quatre variables les plus utiles, on compte les suivantes :

### **hostvars**

Contient les variables pour les hôtes gérés et peut être utilisée afin d'obtenir les valeurs des variables d'un autre hôte géré. Cela n'inclut pas les faits de l'hôte géré si Ansible ne les collecte pas pour cet hôte.

### **group\_names**

Dresse la liste de tous les groupes dans lesquels se trouve l'hôte géré actuel.

### **groups**

Dresse la liste de tous les groupes et hôtes de l'inventaire.

### **inventory\_hostname**

Contient le nom d'hôte de l'hôte géré actuel tel que configuré dans l'inventaire. Il peut, pour différentes raisons, être différent du nom d'hôte signalé par les faits.

Il existe également un certain nombre d'autres « variables magiques ». Pour plus d'informations, reportez-vous à *Variable Precedence – Ansible Documentation* dans la section Références ci-dessous. Un moyen d'obtenir leurs valeurs consiste à utiliser le module **debug** afin de consigner les contenus de la variable **hostvars** pour un hôte particulier :

```
...output omitted...
{
    "hostvars[inventory_hostname)": {
        "awx_job_template_name": "Run test-playbook project",
        "ansible_windows_domain_member": true,
        "tower_user_last_name": "student",
        "ansible_os_product_type": "server",
        "awx_user_email": "student@example.com",
        "module_setup": true,
        "tower_job_launch_type": "manual",
        "awx_job_template_id": 9,
        "gather_subset": [
            "all"
        ],
        "playbook_dir": "/var/lib/awx/projects/_7__test_playbook_repository",
        "ansible_distribution_version": "10.0.17763.0",
        "ansible_playbook_python": "/usr/bin/python2",
        "ansible_forks": 10,
        "ansible_owner_contact": "",
        "ansible_facts": {
            "lastboot": "2019-08-21 03:13:53Z",
            "owner_contact": "",
            "domain": "example.com",
            "distribution_major_version": "10",
            "swaptotal_mb": 0,
            "kernel": "10.0.17763.0",
        }
    }
}
...output omitted...
```

```

},
"tower_user_email": "student@example.com",
"ansible_windows_domain": "example.com",
"awx_job_id": 32,
"ansible_distribution_major_version": "10",
"groups": {
    "ungrouped": [
        "win1.example.com"
    ],
    "Windows": [],
    "all": [
        "win1.example.com"
    ]
},
"ansible_bios_date": "10/16/2017",
"inventory_hostname_short": "win1",
"tower_project_revision": "273396b95ae1d2724f38e4024fc4e1f7907f098f",
"omit": "__omit_place_holder__1594b921bbf58cf665e07686cdff4ae5d1a7125",
"ansible_user_gecos": "",
"ansible_diff_mode": false,
"ansible_processor_threads_per_core": 2,
"ansible_win_rm_certificate_expires": "2022-08-19 03:00:51",
"ansible_product_name": "t3.medium",
"ansible_user_sid": "S-1-5-21-2938333969-4197073335-1435044025-500",
"ansible_run_tags": [
    "all"
],
"awx_user_last_name": "student"
},
"changed": false,
"_ansible_verbose_always": true,
"_ansible_no_log": false
}

```



## Références

**setup - Collecte de faits sur les hôtes distants**; **Documentation Ansible**

[https://docs.ansible.com/ansible/latest/modules/setup\\_module.html](https://docs.ansible.com/ansible/latest/modules/setup_module.html)

## ► Exercice guidé

# Gestion des faits

Dans cet exercice, vous allez collecter des faits Ansible à partir de vos hôtes gérés et les utiliser dans des plays.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Collecter des faits à partir d'hôtes gérés à l'aide des modules **setup**, **win\_product\_facts** et **win\_disk\_facts**.
- Créer des tâches qui utilisent les faits collectés.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. Lancez l'éditeur de Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **variables**, clonez-le sur votre instance **workstation**.
  - 1.1. Cliquez **Search Windows** et tapez **code** pour rechercher et ouvrir Visual Studio Code.
  - 1.2. Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel. Utilisez l'URL de référentiel <https://gitlab.example.com/student/variables.git>. À l'invite, sélectionnez le dossier **Documents** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **C:\Users\student\Documents\variables**.
- ▶ 2. Cliquez sur **File → Open Folder**, puis sélectionnez le répertoire **c:\Users\student\Documents\variables**.
- ▶ 3. Ouvrez le playbook **facts.yml** pour ajouter des tâches qui recueillent des faits système et créent un fichier texte de synthèse.
- ▶ 4. Modifiez la tâche intitulée **Product facts are collected** pour invoquer le module **win\_product\_facts** afin de collecter des informations sur les produits et les licences Windows.

```
- name: Product facts are collected
  win_product_facts:
```

- ▶ 5. Modifiez la tâche intitulée **Disk facts are collected** pour invoquer le module **win\_disk\_facts** afin de collecter des informations sur les disques attachés.

```
- name: Disk facts are collected
  win_disk_facts:
```

- ▶ 6. Modifiez la tâche **Summary is written** pour écrire une sélection des faits dans un fichier texte de résumé sur le Bureau de l'hôte géré.

```
- name: Summary is written
  win_copy:
    content: |
      OS: {{ ansible_facts['os_name'] }}
      Machine ID: {{ ansible_facts['machine_id'] }}
      Last Boot: {{ ansible_facts['lastboot'] }}
      License Status: {{ ansible_facts['os_license_status'] }}
      Disk Size: {{ ansible_facts['disks'][0]['size'] }}
    dest: "{{ ansible_facts['user_dir'] }}\Desktop\summary-{{ ansible_facts['hostname'] }}.txt"
```

- ➊ Utilisez le paramètre **content** du module **win\_copy** pour écrire du contenu simple dans un fichier.
- ➋ Insérez des variables et des faits dans du texte à l'aide de doubles accolades **{{ variable }}**.
- ➌ Accédez aux valeurs de tableau à l'aide de clés numériques, comme dans **myArray[index]**.
- ➍ Identifiez l'emplacement du fichier cible à l'aide du paramètre **dest**. À l'instar des autres valeurs, les noms de fichiers peuvent également inclure la substitution de variables dynamiques. Ansible résout **ansible\_facts['user\_dir']** en **C:\Users\devops**.

- ▶ 7. Vérifiez que le **facts.yml** final correspond au playbook suivant.

```
---
- name: System summary is compiled
hosts: all
tasks:
  - name: Product facts are collected
    win_product_facts:

  - name: Disk facts are collected
    win_disk_facts:

  - name: Summary is written
    win_copy:
      content: |
        OS: {{ ansible_facts['os_name'] }}
        Machine ID: {{ ansible_facts['machine_id'] }}
```

```
Last Boot: {{ ansible_facts['lastboot'] }}
License Status: {{ ansible_facts['os_license_status'] }}
Disk Size: {{ ansible_facts['disks'][0]['size'] }}
dest: "{{ ansible_facts['user_dir'] }}\\Desktop\\summary-{{ ansible_facts['hostname'] }}.txt"
```

- 8. Enregistrez et validez les modifications sur votre référentiel Git local, puis transmettez-les par push au référentiel distant.
- 8.1. Cliquez sur **File** → **Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.
  - 8.2. Accédez au volet **Source Control** et cliquez sur **+** en regard de **facts.yml** pour indexer les modifications.
  - 8.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 8.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- 9. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- 10. Modifiez le modèle de tâche **Run variables project** pour tester votre playbook.

- 10.1. Dans le volet de navigation, cliquez sur **Templates**.
- 10.2. Cliquez sur le modèle de tâche **Run variables project** pour le modifier.
- 10.3. Sélectionnez le playbook **facts.yml** dans la liste **PLAYBOOK**.

The screenshot shows the 'Run variables project' job template configuration page. The 'DETAILS' tab is selected. The configuration includes:

- NAME:** Run variables project
- DESCRIPTION:** Use this job template to run your playbooks
- JOB TYPE:** Run
- INVENTORY:** Default inventory
- PROJECT:** variables repository
- PLAYBOOK:** facts.yml
- CREDENTIAL:** DevOps
- FORKS:** 0
- LIMIT:** (empty)
- VERBOSITY:** 0 (Normal)
- JOB TAGS:** (empty)
- SKIP TAGS:** (empty)
- LABELS:** (empty)
- INSTANCE GROUPS:** (empty)
- JOB SLICING:** 1

- 10.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

- 11. Vérifiez que le playbook a été exécuté correctement et examinez la sortie du journal.
- 11.1. Une fois toutes les tâches terminées, vérifiez que le statut de la section **DETAILS** affiche **Successful**. Si la tâche ne s'est pas correctement déroulée, comparez votre playbook avec le fichier de solution **solutions/facts.sol** dans le référentiel Git **variables**.

The screenshot shows the Ansible Tower interface. On the left, the 'Run variables project' run details are displayed, including the status as 'Successful'. On the right, the 'Run variables project' log output is shown, detailing the execution of tasks such as SSH password gathering, system summary compilation, and gathering facts from hosts win1.example.com and win2.example.com.

```

 1 SSH password:
 2
 3 PLAY [System summary is compiled] ****
 ****
 4
 5 TASK [Gathering Facts] ****
 ****
 6 ok: [win1.example.com]
 7 ok: [win2.example.com]
 8
 9 TASK [Product facts are collected] ****
 ****
10 ok: [win1.example.com]
11 ok: [win2.example.com]

```

- 11.2. Dans le panneau de sortie de la console, cliquez sur **ok: [win1.example.com]** sous **TASK [Gathering Facts]**. Observez le dictionnaire JSON "**ansible\_facts**" dans la zone de texte de la fenêtre modale.

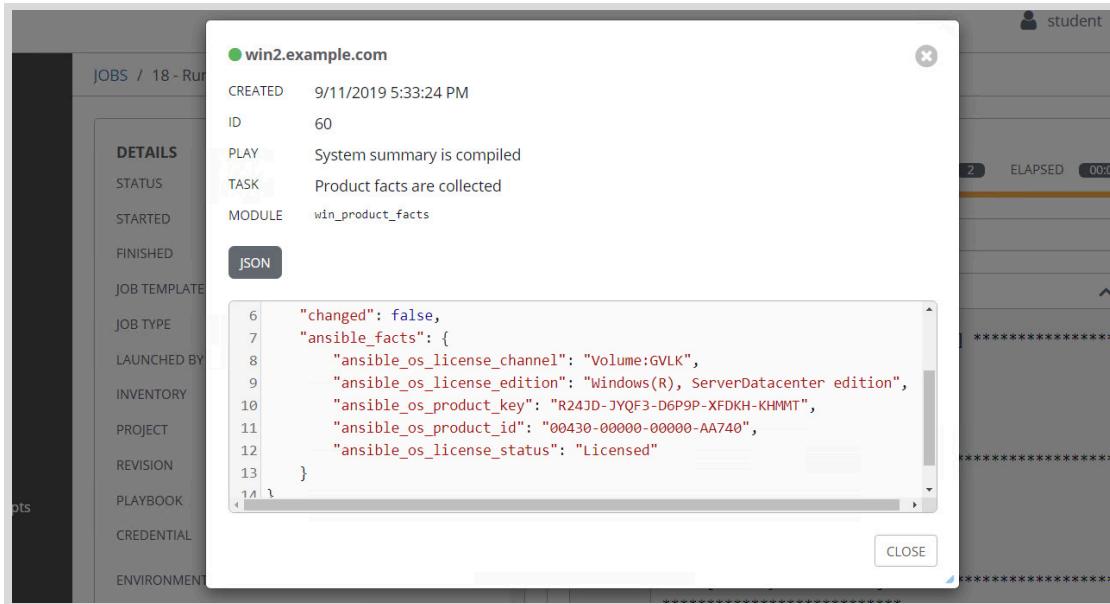
The screenshot shows a modal window for the 'Gathering Facts' task of the 'win1.example.com' job. The modal displays the JSON output of the 'gather\_facts' module, which includes details like the Windows domain membership, product serial, OS type, user ID, and user directory for the host.

```

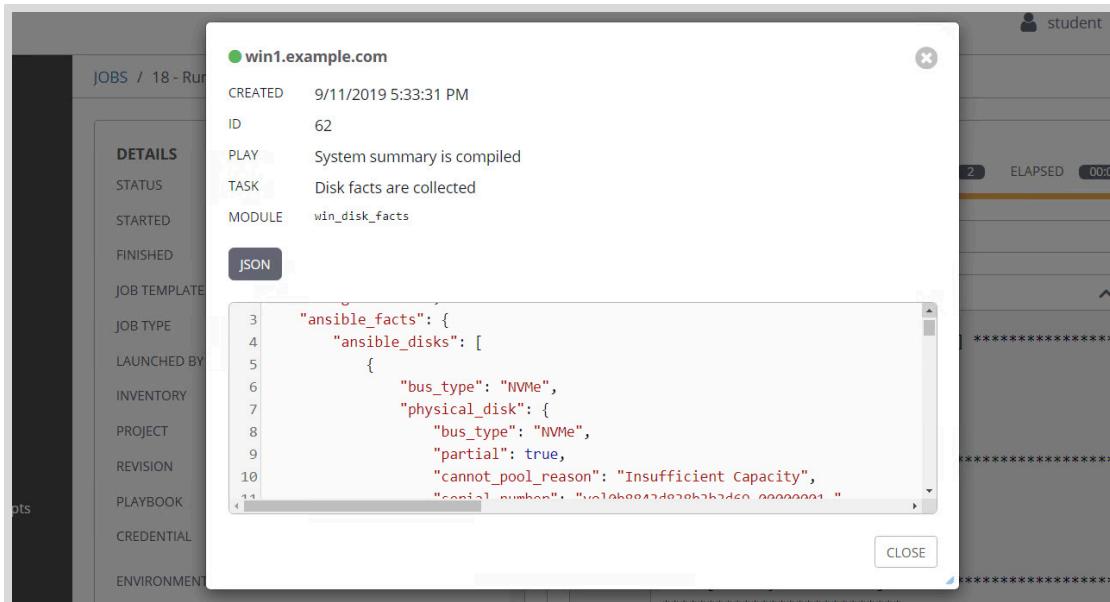
1 {
2   "changed": false,
3   "_ansible_verbose_override": true,
4   "ansible_facts": {
5     "ansible_windows_domain_member": true,
6     "ansible_product_serial": "ec2ad64a-ae66-f995-c660-2846d66b2be2",
7     "ansible_os_product_type": "server",
8     "ansible_user_id": "devops",
9     "ansible_user_dir": "c:\\Users\\devops"
}

```

- 11.3. Dans le panneau de sortie de la console, cliquez sur **ok: [win2.example.com]** sous **TASK [Product facts are collected]**. Observez le dictionnaire JSON "**ansible\_facts**" dans la zone de texte de la fenêtre modale.



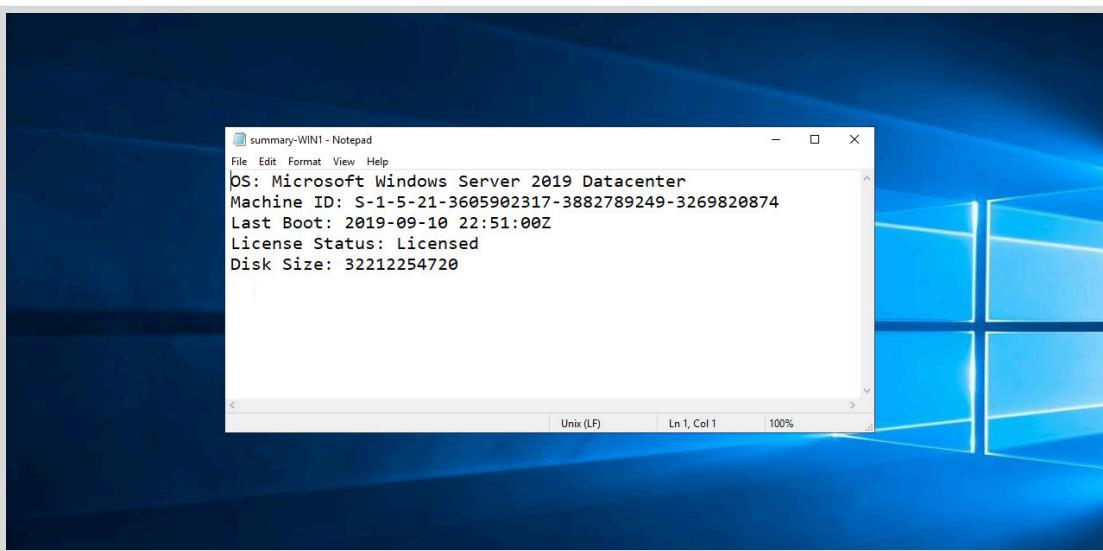
- 11.4. Dans le panneau de sortie de la console, cliquez sur **ok: [win1.example.com]** sous **TASK [Disk facts are collected]**. Observez le dictionnaire JSON "**ansible\_facts**" dans la zone de texte de la fenêtre modale.



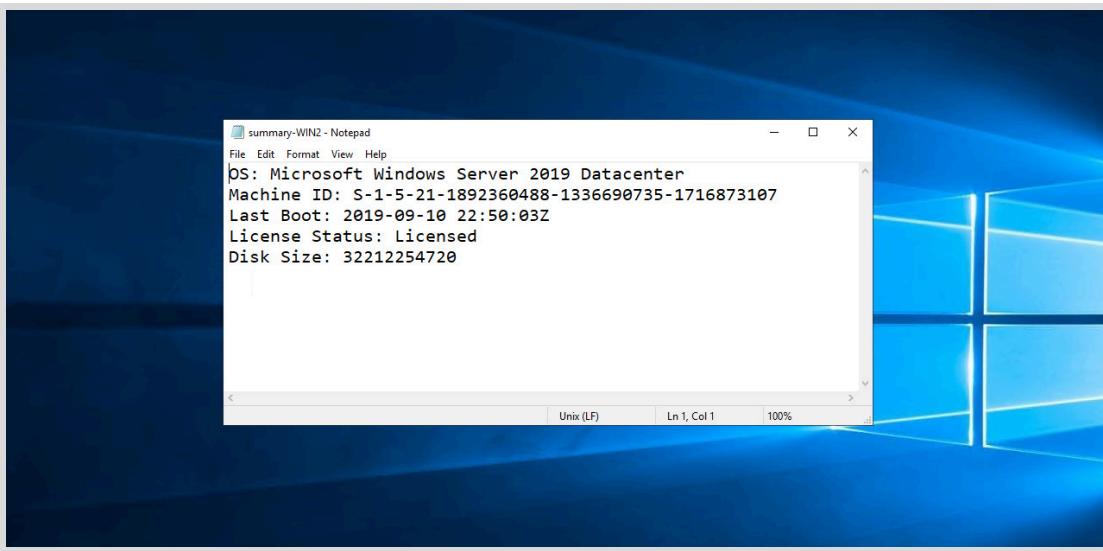
- 12. Connectez-vous à l'hôte géré **win1.example.com** et affichez le fichier de résumé du système qui a été écrit sur le bureau.

- 12.1. Cliquez sur **Search Windows** et tapez **remote** pour rechercher et ouvrir Connexion Bureau à distance.
- 12.2. Saisissez **win1.example.com** pour **Computer** et **devops** pour **User Name** et **RedHat123@!** comme mot de passe. Notez que **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.

- 12.3. Double-cliquez sur le document texte **summary-WIN1.txt** sur le bureau **win1.example.com** pour passer en revue son contenu.



- 13. Connectez-vous à l'hôte géré **win2.example.com** et affichez le fichier de résumé du système qui a été écrit sur le Bureau.
- 13.1. Cliquez sur **Search Windows** et tapez **remote** pour rechercher et ouvrir Connexion Bureau à distance.
  - 13.2. Saisissez **win2.example.com** pour **Computer** et **devops** pour **User Name** et **RedHat123@!** comme mot de passe.
  - 13.3. Double-cliquez sur le document texte **summary-WIN2.txt** sur le bureau **win2.example.com** pour passer en revue son contenu.



L'exercice guidé est maintenant terminé.

## ► Open Lab

# Gestion des variables et des faits

### Liste de contrôle des performances

Dans cet atelier, vous allez écrire et exécuter un playbook Ansible qui utilise des variables et des faits.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Définir des variables de groupe dans Ansible.
- Refactoriser un playbook Ansible pour utiliser des variables et des faits.
- Exécuter une tâche dans Ansible Tower.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



#### Note

Cette activité nécessite **win2.example.com** dans l'inventaire **Default inventory**. Vous avez ajouté l'hôte dans un exercice précédent. Si l'hôte n'est pas présent, suivez les instructions de l'exercice guidé *Mise en œuvre de plusieurs plays* pour le créer.

Dans cet atelier, vous allez refactoriser un playbook pour utiliser des variables de groupe et enregistrées, et pour afficher des informations sur le fichier d'échange et la mémoire.

1. À partir de **workstation**, ouvrez Visual Studio Code et clonez le référentiel **variables** accessible sur <https://gitlab.example.com/student/variables.git> vers **C:\Users\student\Documents\variables**, puis ouvrez le dossier.
2. Créez un dossier pour les variables de groupe, s'il n'existe pas, et un sous-dossier pour le groupe **Windows**. Si un fichier nommé **Windows** existe déjà, supprimez-le d'abord. Créez un fichier nommé **vars.yml** sous le dossier **Windows** qui définit la variable **pagefile**. Cette variable contient la liste suivante de variables qui contrôlent le module **win\_pagefile** :
  - **remove: yes**
  - **auto: no**

3. Ouvrez **review.yml** pour l'éditer. Localisez la tâche intitulée **Obtain page file information**. Cette tâche utilise le module **win\_pagefile** sans argument pour interroger la configuration actuelle. Complétez la tâche **Show original page file information** à l'aide du module **debug** pour afficher la valeur de la variable **orig\_pagefile\_info** enregistrée. Modifiez la tâche **Manage page files on virtual machines** pour inclure l'utilisation des variables **pagefile**. Complétez la tâche **Show new page file information** à l'aide du module **debug** pour afficher la variable **new\_pagefile\_info** enregistrée. Complétez la tâche **Show current memory information** à l'aide du module **debug** pour afficher le fait **ansible\_facts['memtotal\_mb']**.
4. Enregistrez le fichier, puis validez et transmettez par push vos modifications vers le référentiel **variables** distant.
5. Ouvrez l'interface utilisateur Web d'Ansible Tower et testez votre playbook à l'aide du modèle de tâche **Run variables project**. Passez en revue la sortie de la tâche et notez les différences entre la configuration originale du fichier d'échange et la nouvelle configuration.
6. Modifiez le modèle de tâche **Run variables project** pour qu'il utilise le playbook **review\_finish.yml**, puis lancez-le pour annuler vos modifications.

L'atelier est maintenant terminé.

## ► Solution

# Gestion des variables et des faits

### Liste de contrôle des performances

Dans cet atelier, vous allez écrire et exécuter un playbook Ansible qui utilise des variables et des faits.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Définir des variables de groupe dans Ansible.
- Refactoriser un playbook Ansible pour utiliser des variables et des faits.
- Exécuter une tâche dans Ansible Tower.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



#### Note

Cette activité nécessite **win2.example.com** dans l'inventaire **Default inventory**. Vous avez ajouté l'hôte dans un exercice précédent. Si l'hôte n'est pas présent, suivez les instructions de l'exercice guidé *Mise en œuvre de plusieurs plays* pour le créer.

Dans cet atelier, vous allez refactoriser un playbook pour utiliser des variables de groupe et enregistrées, et pour afficher des informations sur le fichier d'échange et la mémoire.

1. À partir de **workstation**, ouvrez Visual Studio Code et clonez le référentiel **variables** accessible sur <https://gitlab.example.com/student/variables.git> vers **C:\Users\student\Documents\variables**, puis ouvrez le dossier.
  - 1.1. Double-cliquez sur l'icône Visual Studio Code sur le Bureau.
  - 1.2. Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **Git: Clone** pour cloner un référentiel.
  - 1.3. Utilisez l'URL de référentiel <https://gitlab.example.com/student/variables.git> et appuyez sur **Entrée**.

Sélectionnez le dossier **Documents** et cliquez sur **Select Repository Location** pour cloner le référentiel **variables** dans le dossier **variables**.

- 1.4. Cliquez sur **Open** pour ajouter le dossier à l'espace de travail.
2. Créez un dossier pour les variables de groupe, s'il n'existe pas, et un sous-dossier pour le groupe **Windows**. Si un fichier nommé **Windows** existe déjà, supprimez-le d'abord. Créez un fichier nommé **vars.yml** sous le dossier **Windows** qui définit la variable **pagefile**. Cette variable contient la liste suivante de variables qui contrôlent le module **win\_pagefile** :
  - **remove: yes**
  - **auto: no**
  - 2.1. Créez un nouveau dossier **group\_vars** au niveau supérieur du projet s'il n'existe pas déjà. Pour ce faire, cliquez avec le bouton droit de la souris et sélectionnez **New Folder** dans l'arborescence de l'éditeur sur le côté gauche.
  - 2.2. Créez un nouveau dossier **Windows** dans le dossier **group\_vars**. Si un fichier nommé **Windows** existe déjà, supprimez-le d'abord.
  - 2.3. Créez un nouveau fichier **group\_vars/Windows/vars.yml**.
  - 2.4. Dans le fichier **vars.yml**, définissez la variable **pagefile** avec son contenu en tant que paramètres du module **win\_pagefile**.

```
---  
pagefile:  
  remove: yes  
  auto: no
```

- 2.5. Enregistrez vos modifications.
3. Ouvrez **review.yml** pour l'éditer. Localisez la tâche intitulée **Obtain page file information**. Cette tâche utilise le module **win\_pagefile** sans argument pour interroger la configuration actuelle. Complétez la tâche **Show original page file information** à l'aide du module **debug** pour afficher la valeur de la variable **orig\_pagefile\_info** enregistrée. Modifiez la tâche **Manage page files on virtual machines** pour inclure l'utilisation des variables **pagefile**. Complétez la tâche **Show new page file information** à l'aide du module **debug** pour afficher la variable **new\_pagefile\_info** enregistrée. Complétez la tâche **Show current memory information** à l'aide du module **debug** pour afficher le fait **ansible\_facts['memtotal\_mb']**.
- 3.1. Localisez la tâche intitulée **Obtain page file information**. Cette tâche utilise le module **win\_pagefile** sans argument pour interroger la configuration actuelle.

```
- name: Obtain page file information  
  win_pagefile:  
    register: orig_pagefile_info
```

- 3.2. Complétez la tâche **Show original page file information** à l'aide du module **debug** pour afficher la valeur de la variable **orig\_pagefile\_info** enregistrée.

```
- name: Show original page file information  
  debug:  
    var: orig_pagefile_info
```

- 3.3. Localisez la tâche intitulée **Manage page files on virtual machines**. Remplacez les instances de CHANGE\_ME par les variables **pagefile** appropriées. Cette tâche permet d'apporter des modifications à la configuration du fichier d'échange.

```
- name: Manage page files on virtual machines
  win_pagefile:
    remove_all: "{{ pagefile['remove'] }}"
    automatic: "{{ pagefile['auto'] }}"
  register: new_pagefile_info
```

- 3.4. Complétez la tâche **Show new page file information** à l'aide du module **debug** pour afficher la valeur de la variable **pagefile\_info** enregistrée.

```
- name: Show new page file information
  debug:
    var: new_pagefile_info
```

- 3.5. Complétez la tâche **Show current memory information** à l'aide du module **debug** pour afficher la valeur du fait **ansible\_facts['memtotal\_mb']**.

```
- name: Show current memory information
  debug:
    var: ansible_facts['memtotal_mb']
```

- 3.6. Le playbook complet doit ressembler à ceci :

```
---
- name: Manage page files
  hosts: Windows

  tasks:
    - name: Obtain page file information
      win_pagefile:
        register: orig_pagefile_info

    - name: Show original page file information
      debug:
        var: orig_pagefile_info

    - name: Manage page files on virtual machines
      win_pagefile:
        remove_all: "{{ pagefile['remove'] }}"
        automatic: "{{ pagefile['auto'] }}"
      register: new_pagefile_info

    - name: Show new page file information
      debug:
        var: new_pagefile_info
```

```
- name: Show current memory information
  debug:
    var: ansible_facts['memtotal_mb']
```

4. Enregistrez le fichier, puis validez et transmettez par push vos modifications vers le référentiel **variables** distant.
  - 4.1. Enregistrez vos modifications apportées au fichier **review.yml**.
  - 4.2. À partir de la barre latérale de Visual Studio Code, sélectionnez **Source Control** pour accéder à la liste des modifications.
  - 4.3. Survolez le playbook **review.yml** pour accéder aux options disponibles pour le fichier. Cliquez sur **+** pour ajouter vos modifications à l'index.
  - 4.4. Dans le champ **Message**, saisissez un message de validation concis, puis appuyez sur **Ctrl+Enter**.
  - 4.5. Pour pousser vos modifications locales vers le référentiel distant, cliquez sur le bouton **Synchronize Changes** en bas de l'éditeur. Cela indique à Visual Studio Code de synchroniser votre référentiel avec la copie distante.
5. Ouvrez l'interface utilisateur Web d'Ansible Tower et testez votre playbook à l'aide du modèle de tâche **Run variables project**. Passez en revue la sortie de la tâche et notez les différences entre la configuration originale du fichier d'échange et la nouvelle configuration.
  - 5.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 5.2. Cliquez sur le nom du modèle de tâche **Run variables project**.
  - 5.3. Sélectionnez le playbook **review.yml** dans la liste **PLAYBOOK**.
  - 5.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
  - 5.5. Examinez le résultat de la tâche **Show current page file information** pour confirmer que les modifications ont réussi.

```
...output omitted...
"orig_pagefile_info": {
  "failed": false,
  "automatic_managed_pagefiles": true,
  "changed": false,
  "pagefiles": []
}
...output omitted...
"new_pagefile_info": {
  "failed": false,
  "automatic_managed_pagefiles": false,
  "changed": true,
  "pagefiles": [
    {
      "caption": "C:\\\\ 'pagefile.sys'",
      "maximum_size": 0,
      "initial_size": 0,
      "status": "Normal"
    }
  ]
}
```

```
        "description": "'pagefile.sys' @ C:\\\\",
        "name": "C:\\\\pagefile.sys"
    }
```

6. Modifiez le modèle de tâche **Run variables project** pour qu'il utilise le playbook **review\_finish.yml**, puis lancez-le pour annuler vos modifications.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Les variables, qui ont une étendue globale, de play ou d'hôte, permettent aux plays d'être plus flexibles, en fournissant des valeurs par défaut qui peuvent être remplacées à différents niveaux.
- Lorsque vous utilisez des variables dans un play, vous devez les placer entre guillemets si elles débutent la ligne. Les variables sans guillemets au début de la ligne empêchent l'analyse du playbook.
- De nombreuses méthodes de gestion des informations sensibles dans les types d'informations d'identification personnalisés d'Ansible Tower vous permettent de stocker des informations d'identification et des clés non standard, chiffrées dans la base de données.
- Ansible Tower prend en charge l'intégration avec plusieurs services de stockage secrets centralisés à partir de CyberArk, HashiCorp et Microsoft.
- Ansible Vault gère le chiffrement des secrets et est fourni avec Ansible, mais il n'est pris en charge que sous Linux. Il existe des méthodes non prises en charge pour gérer les fichiers chiffrés Ansible Vault sous Windows.
- Les faits sont des informations relatives à un hôte géré, qui sont recueillies par le module **setup**. Vous pouvez utiliser des faits dans des instructions conditionnelles.



## chapitre 5

# Installation et mise à jour de logiciels

### Objectif

Installer, gérer et s'assurer que le logiciel est à jour à l'aide de playbooks Ansible.

### Résultats

- S'assurer que le logiciel est installé et à jour sur les systèmes Microsoft Windows.
- Inspecter le Registre Windows et s'assurer que les clés de Registre sont correctement configurées.

### Sections

- Installation et mise à jour de logiciels (et exercice guidé)
- Modification du Registre (et exercice guidé)

### Atelier

Installation et mise à jour de logiciels

# Installation et mise à jour de logiciels

---

## Résultats

Au terme de cette section, vous devez vous assurer que le logiciel est installé et à jour sur les systèmes Microsoft Windows.

## Installation de logiciels sur des systèmes Windows

Il est possible d'effectuer de nombreux types d'installation de logiciels sur un hôte Windows, tels que l'ajout de rôles et de fonctions Windows, l'installation de paquetages MSI ou d'exécutables, ainsi que l'application de hotfix ou de mises à jour Windows. Plusieurs modules Ansible sont disponibles pour prendre en charge ces différentes méthodes d'installation. Les plus courants sont décrits ci-dessous.

### Modules d'installation de logiciels

#### win\_package

Module d'origine pour l'installation de paquetages MSI ou d'exécutables, qui peuvent être disponibles localement ou à distance à partir d'une URL. Il existe plusieurs méthodes pour déterminer si le paquetage est déjà installé, comme **creates\_path** et **create\_service** par exemple, qui vérifient la présence de répertoires et de services respectivement. Vous pouvez également spécifier l'argument **product\_id** pour vérifier la présence du produit installé dans le Registre.

#### win\_chocolatey

Chocolatey est un gestionnaire de paquetages pour Windows, qui vous permet de rechercher, d'installer et de désinstaller des paquetages à partir du référentiel Chocolatey. Le module **win\_chocolatey**, ainsi que les sous-modules qui y sont associés, prennent en charge les fonctions de gestion de paquetages Chocolatey à partir d'Ansible.

#### win\_feature

Sur un serveur Windows, les rôles et les fonctions sont des caractéristiques que vous pouvez ajouter à un serveur et qui font partie intégrante du système d'exploitation. Par exemple, pour qu'un serveur propose le service DHCP, utilisez l'Assistant **Ajout de rôles et de fonctionnalités**. Le module **win\_feature** vous permet d'automatiser l'ajout et la suppression de rôles et de fonctions.

L'exemple suivant montre le module **win\_chocolatey** qui installe plusieurs paquetages en même temps, et spécifie un serveur proxy à utiliser avec l'option **proxy\_url**.

```
- name: Install developer packages
  win_chocolatey:
    name:
      - firefox
      - git
      - jdk8
    state: present
    proxy_url: http://192.168.0.254:3128
```

Contrairement à **win\_feature**, le module **win\_chocolatey** ne prend pas en charge l'évaluation de la nécessité d'un redémarrage. Cependant, vous pouvez passer en revue la sortie de la commande en la capturant avec une variable enregistrée.

L'exemple suivant montre comment installer un serveur DHCP à l'aide du module **win\_feature**. Il utilise une variable enregistrée pour déclencher le module **win\_reboot**, si un redémarrage est nécessaire.

```
- name: Install DHCP server
  win_feature:
    name: DHCP
    state: present
    include_management_tools: yes
  register: dhcp_feature

- name: Reboot if required by DHCP feature
  win_reboot:
    when: dhcp_feature.reboot_required
```

Lorsqu'une tâche **win\_feature** réussit, des informations supplémentaires sont disponibles dans la variable **feature\_result**, telles que la liste des articles de la base de connaissances qui s'appliquent à la fonction. Vous pouvez utiliser une variable enregistrée comme indiqué dans l'exemple précédent, puis afficher la variable **registered\_variable.feature\_result** à l'aide du module **debug**.

Vous pouvez obtenir la liste de tous les noms de fonctions disponibles sur un système en exécutant la commande PowerShell **Get-WindowsFeature**. Toutes les fonctions possédant des sous-fonctions sont listées sous la forme d'une arborescence, avec les sous-fonctions mises en retrait en dessous de la fonction principale.

Display Name	Name	Install State
[ ] Active Directory Certificate Services	AD-Certificate	Available
[ ] Certification Authority	ADCS-Cert-Authority	Available
[ ] Certificate Enrollment Policy Web Service	ADCS-Enroll-Web-Pol	Available
[ ] Certificate Enrollment Web Service	ADCS-Enroll-Web-Svc	Available
[ ] Certification Authority Web Enrollment	ADCS-Web-Enrollment	Available
[ ] Network Device Enrollment Service	ADCS-Device-Enrollment	Available
[ ] Online Responder	ADCS-Online-Cert	Available
[ ] Active Directory Domain Services	AD-Domain-Services	Available
[ ] Active Directory Federation Services	ADFS-Federation	Available
[ ] Active Directory Lightweight Directory Services	ADLDS	Available
[ ] Active Directory Rights Management Services	ADRMS	Available
[ ] Active Directory Rights Management Server	ADRMS-Server	Available
[ ] Identity Federation Support	ADRMS-Identity	Available
[ ] Device Health Attestation	DeviceHealthAttestat...	Available
[ ] DHCP Server	DHCP	Available
[ ] DNS Server	DNS	Available
[ ] Fax Server	Fax	Available
[X] File and Storage Services	FileAndStorage-Services	Installed
[ ] File and iSCSI Services	File-Services	Available
[ ] File Server	FS-FileServer	Available
[ ] BranchCache for Network Files	FS-BranchCache	Available
[ ] Data Deduplication	FS-Data-Deduplication	Available
[ ] DFS Namespaces	FS-DFS-Namespace	Available
[ ] DFS Replication	FS-DFS-Replication	Available
[ ] File Server Resource Manager	FS-Resource-Manager	Available
[ ] File Server VSS Agent Service	FS-VSS-Agent	Available
[ ] iSCSI Target Server	FS-iSCSITarget-Server	Available
[ ] iSCSI Target Storage Provider (VDS and V...	iSCSITarget-VSS-VDS	Available
[ ] Server for NFS	FS-NFS-Service	Available
[ ] Work Folders	FS-SyncShareService	Available
[ ] Storage Services	Storage-Services	Installed
[ ] Host Guardian Service	HostGuardianServiceRole	Available
[ ] Hyper-V	Hyper-V	Available
[ ] MultiPoint Services	MultiPointServerRole	Available
[ ] Network Controller	NetworkController	Available
[ ] Network Policy and Access Services	NPAS	Available
[ ] Print and Document Services	Print-Services	Available
[ ] Print Server	Print-Server	Available
[ ] Distributed Scan Server	Print-Scan-Server	Available
[ ] Internet Printing	Print-Internet	Available
[ ] LPD Service	Print-LPD-Service	Available
[ ] Remote Access	RemoteAccess	Available
[ ] DirectAccess and VPN (RAS)	DirectAccess-VPN	Available
[ ] Routing	Routing	Available
[ ] Web Application Proxy	Web-Application-Proxy	Available

Figure 5.1: Liste des noms de fonctions Windows

L'exemple suivant montre comment installer un paquetage MSI à l'aide du module **win\_package**. Il utilise le paramètre **creates\_path** pour vérifier si le paquetage est déjà installé.

```
- name: Install Registry Cleaner
  win_package:
    path: C:\temp\susregclean.msi
    state: present
    creates_path: C:\Program Files\Suspicious Registry Cleaner
```

Les tâches qui utilisent **win\_package** peuvent fournir n'importe lequel des paramètres **creates\_[path, service, version]**, afin de tester si un paquetage est installé.

## Mise à jour de logiciels sur des systèmes Windows

À l'instar de l'installation de logiciels, il existe plusieurs méthodes pour la mise à jour de logiciels. Les modules d'installation prennent en charge la mise à jour des paquetages vers une version plus récente, mais ils ne gèrent pas les mises à jour ou les hotfix Windows. Les modules de gestion des hotfix et des mises à jour Windows sont répertoriés ci-dessous.

### Modules de mise à jour logicielle

#### win\_updates

Ce module fonctionne avec le service **Windows Update** pour automatiser le téléchargement et l'installation des mises à jour Windows. Vous pouvez restreindre les catégories disponibles à partir desquelles installer les mises à jour en fournissant l'argument **category\_names**. Si nécessaire, ce module peut redémarrer le système et poursuivre l'installation des mises à jour, évitant ainsi les appels au module **win\_reboot**.

#### win\_hotfix

Ce module gère l'installation des fichiers **.msu** de hotfix. Le paquetage peut être vérifié par rapport à l'identifiant ou l'article de la base de connaissances, mais une incohérence par rapport à l'un d'entre eux empêchera l'installation.

#### win\_reboot

Utilisez ce module pour contrôler l'exécution d'un redémarrage. Par exemple, vous pouvez utiliser l'option **notify** dans une tâche **win\_hotfix** pour lancer une tâche gestionnaire de redémarrage.

L'exemple suivant montre le module **win\_updates**. Il est configuré pour installer uniquement les mises à jour critiques, pour redémarrer le système si nécessaire et pour utiliser une tâche planifiée au lieu de la méthode **become** permettant d'augmenter les priviléges.

```
- name: Install any critical updates, reboot if required
  win_updates:
    category_names:
      - CriticalUpdates
    use_scheduled_task: yes
    reboot: yes
    reboot_timeout: 600
```

Cet exemple montre le module **win\_hotfix**. Ce module prend également en charge la notification de la nécessité ou non d'un redémarrage.

```
- name: Install hotfix for SAP printing
  win_hotfix:
    hotfix_kb: KB7654321
    source: C:\temp\hotfix.msu
    state: present
    register: sap_hotfix

- name: Reboot if required for SAP printing hotfix
  win_reboot:
    when: sap_hotfix.reboot_required
```

Le module **win\_hotfix** permet de vérifier un paquetage de hotfix par rapport à l'identifiant du hotfix, ou au numéro d'article de la base de connaissances pour le hotfix. En cas d'échec de la vérification, l'installation du hotfix est annulée.

Il est peu probable que ces modules primaires soient utilisés eux-mêmes dans un play. Il existe de nombreux autres modules qui effectuent des tâches auxiliaires, telles que le redémarrage des services (**win\_service**), la décompression d'archives (**win\_unzip**) ou l'exécution de modifications de configuration dans le Registre (**win\_regedit**). Les modules dont vous avez besoin dépendent de la complexité du logiciel en cours d'installation ou de mise à jour.



## Références

### Modules Windows &mdash; Documentation Ansible

[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_windows\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html)

### Chocolatey - Le gestionnaire de paquetages pour Windows

<https://chocolatey.org/>

## ► Exercice guidé

# Installation et mise à jour de logiciels

Au cours de cet exercice, vous allez utiliser divers modules pour vous assurer que le logiciel est installé et à jour sur les hôtes gérés par Microsoft Windows.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Générer un jeton GitLab.
- Cloner un référentiel Git.
- Écrire deux plays dans un playbook Ansible.
- Mettre à jour les modèles de tâches Ansible Tower.
- Examiner les paquetages installés sur les hôtes gérés.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- 1. À partir de **workstation**, accédez à votre instance GitLab et examinez le référentiel **updates**.
- 1.1. À partir de **workstation**, double-cliquez sur l'icône GitLab sur votre Bureau pour accéder à votre instance GitLab.  
Connectez-vous à GitLab en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Veillez à sélectionner LDAP comme méthode d'authentification.
  - 1.2. La page principale liste les référentiels disponibles. Sélectionnez le référentiel **updates** pour l'espace de noms **student**.

The screenshot shows a list of projects in a GitLab interface. The projects are:

- U student / updates (highlighted with a red border)
- V student / variables
- A student / ad
- P student / playbooks
- A student / architecture
- S student / student

Each project has a star rating (0), a lock icon, and a timestamp indicating when it was last updated.

- 1.3. Ce référentiel contient un playbook Ansible **updates.yml** de niveau supérieur. Dans ce playbook, vous allez créer deux plays : un qui gère les mises à jour sur vos serveurs et un autre qui exécute Chocolatey pour installer des paquetages.

Cliquez sur le fichier pour l'inspecter.

```

# Update updates.yml
student committed less than a minute ago
ffc9c9cb

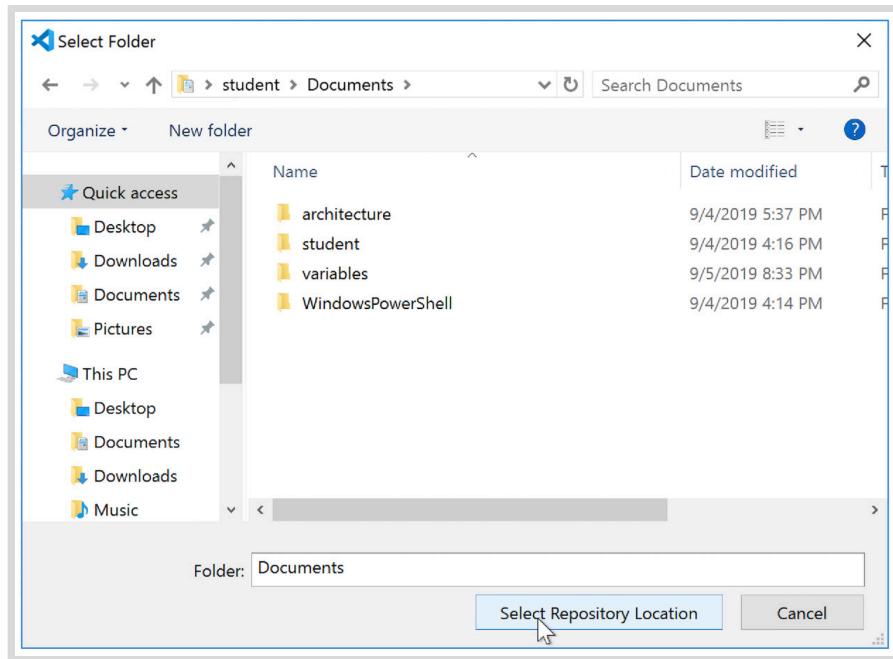
updates.yml 412 Bytes
Edit Replace Delete

1 # Give the first play a name,
2 # The plays ensures that updates are installed on the system. It also installs
3 # a .MSU package on the systems.
4 - name:
5   # Indicate the hosts to target for this play
6   hosts:
7   # Indicate the variables for this play:
8   vars:
9   tasks:
10
11  # Give the second play a name
12  # The plays invokes Chocolatey to install a set of packages to install
13  # packages on the server.
14  - name:

```

- 2. Ouvrez l'éditeur Visual Studio Code et clonez le référentiel **updates** sur votre **workstation**.

- 2.1. À partir de **workstation**, ouvrez l'éditeur Visual Studio Code en double-cliquant sur l'icône sur le Bureau.
- 2.2. Ouvrez la palette de commandes en accédant à **View → Command Palette** ou en appuyant sur **Ctrl+Maj+P**.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.
- 2.3. À l'invite, fournissez l'URL de référentiel <https://gitlab.example.com/student/updates.git>, puis sélectionnez le dossier **Documents** dans le répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel.  
Cette opération clone le référentiel distant dans le dossier **updates** sur l'instance **workstation**.



- 2.4. Lorsque vous êtes invité à ouvrir un projet, cliquez sur **Open**.
- 3. Dans Visual Studio Code, ouvrez le playbook **updates.yml**. Rédigez le premier play pour la gestion des mises à jour sur vos systèmes, comme décrit dans les étapes suivantes.
- 3.1. Mettez à jour le premier play en lui attribuant le nom **Manage updates on systems**.
- ```
- name: Manage updates on systems
```
- 3.2. Demandez à Ansible de cibler tous les hôtes en utilisant la valeur **all** pour la directive **hosts** sous le nom du play.
- ```
- name: Manage updates on systems
  hosts: all
```
- 3.3. Pour prendre en charge la réutilisabilité, créez trois nouvelles variables : **log\_file**, **KB** et **access\_token**.  
La variable **log\_file** pointe vers un fichier texte qui contient la liste des mises à jour disponibles. Utilisez la valeur **C:\Windows\Temp\ansible\_available\_updates.txt**.  
La variable **KB** contient le numéro de la base de connaissances pour la mise à jour que le playbook installe. Attribuez-lui la valeur **KB4465065**.  
Laissez la variable **access\_token** avec une valeur vide. Lors d'une étape ultérieure, vous allez générer et transmettre ce jeton d'accès à la variable dans Ansible Tower.

```
- name: Manage Updates on Systems
hosts: all
vars:
  log_file: C:\Windows\Temp\ansible_available_updates.txt
  KB: 'KB4465065'
  access_token: ""
```

- 3.4. Dans la section des tâches du play, créez la tâche initiale à l'aide du module **win\_updates**. Ce module initial interroge les serveurs de mise à jour Microsoft pour déterminer s'il existe des mises à jour de sécurité disponibles. Cependant, la tâche ne les installe pas.

Nommez cette tâche **Microsoft update servers are queried**. Utilisez le module **win\_updates** avec les paramètres suivants :

- Définissez **category\_names** pour **SecurityUpdates** afin de rechercher uniquement les mises à jour de sécurité.
- Utilisez **state** pour **searched** afin de rechercher de nouvelles mises à jour, mais ignorez l'installation.
- Enregistrez la sortie dans le fichier texte **C:\Windows\Temp\ansible\_available\_updates.txt** à l'aide de la variable **log\_file** dans le paramètre **log\_path**.

La tâche doit se présenter comme suit :

```
...output omitted...
tasks:
  - name: Microsoft update servers are queried
    win_updates:
      category_names: SecurityUpdates
      state: searched
      log_path: "{{ log_file }}"
```

- 3.5. Pour afficher le contenu du fichier, créez une nouvelle tâche en utilisant le module **win\_shell** pour exécuter la commande **type**. La commande **type** affiche le contenu d'un fichier.

Nommez cette tâche **Available security updates are captured**, et utilisez la variable **log\_file** de façon à ce que la commande accède au fichier **ansible\_available\_updates.txt**.

Utilisez l'instruction **register** pour capturer la sortie dans la variable **file\_output**. Le contenu de la variable est affiché à l'étape suivante.



#### Note

Notez la mise en retrait. L'instruction **Register** et le module **win\_shell** sont tous deux en retrait de deux espaces, en dessous du nom de la tâche.

```
...output omitted...
- name: Available security updates are captured
  win_shell: "type {{ log_file }}"
  register: file_output
```

- 3.6. Créez une tâche qui utilise le module **debug**. Ce module peut afficher un message ou le contenu d'une variable s'il existe des mises à jour disponibles. Le module utilise l'instruction **when** pour analyser le fichier et rechercher la chaîne **Adding update**. Nommez cette tâche **Available updates are displayed**.



### Note

L'instruction **when** est mise en retrait au même niveau que le module **debug**.

```
...output omitted...
- name: Available security updates are captured
  win_shell: "type {{ log_file }}"
  register: file_output

- name: Available updates found
  debug:
    msg: "Available updates found!"
  when: "'Adding update' in file_output.stdout"
```

- 3.7. Le référentiel Git **updates** fournit un fichier **.MSU**, qui est une mise à jour pour vos serveurs. Le fichier est accessible à l'adresse <https://gitlab.example.com/student/updates/raw/master/files/KB4465065.msu>.

Créez une tâche qui utilise le module **win\_uri** pour récupérer le fichier et l'enregistrer dans **C:\Windows\Temp\KB4465065.msu**. Attribuez-lui le nom **MSU package is retrieved on hosts**.

Étant donné que l'URL est protégée par un mot de passe, utilisez le paramètre **headers** pour transmettre le jeton d'accès à la requête sous la forme d'un en-tête. La variable **force\_basic\_auth: yes** indique à Ansible d'ajouter un en-tête d'authentification de base à la requête initiale.

Réutilisez la variable **KB**, qui contient le nom du fichier, comme le paramètre **src**.



### Note

L'instruction **validate\_certs: no** indique à Ansible d'accepter les certificats non sécurisés. Étant donné qu'il s'agit d'un environnement de développement, il est acceptable d'ignorer la vérification des certificats, mais il n'est pas recommandé de contourner cette vérification dans les environnements de production.

```
...output omitted...
- name: MSU package is retrieved on hosts
  win_uri:
    url: >
      https://gitlab.example.com/student/updates/raw/master/files/{{ KB }}.msu
    dest: C:\Windows\Temp\{{ KB }}.msu
    validate_certs: no
    force_basic_auth: yes
    method: GET
    headers:
      PRIVATE-TOKEN: "{{ access_token }}"
```

- 3.8. L'étape suivante installe le paquetage récupéré par le module précédent. Pour commencer, créez une nouvelle tâche qui utilise le module **win\_hotfix**. Transmettez le chemin d'accès du fichier MSU au module et l'identifiant de la base de connaissances (réutilisez la variable **KB**). Le paramètre **state: present** indique à Ansible d'installer le paquetage.

Nommez cette tâche **MSU package is installed**.

```
...output omitted...
- name: MSU package is installed
  win_hotfix:
    hotfix_kb: "{{ KB }}"
    source: C:\Windows\Temp\{{ KB }}.msu
    state: present
```

- 3.9. Un autre moyen d'installer les mises à jour consiste à utiliser le module **win\_updates**. Créez une nouvelle tâche qui appelle ce module ; nommez cette tâche **Security updates are applied**. Le module validera les paquetages à installer, sur la base d'une catégorie définie par les paramètres **category\_name** et **whitelist**.

Ajoutez les numéros de la base de connaissances **KB4494174** et **KB4346084** au paramètre **whitelist**. Le module renvoie une valeur booléenne dans la variable **reboot\_required** qui indique si les mises à jour nécessitent ou non un redémarrage du serveur.

Enregistrez la sortie dans la variable **updates\_status**.

```
...output omitted...
- name: Security updates are applied
  win_updates:
    category_name:
      - Updates
    whitelist:
      - KB4494174
      - KB4346084
    register: updates_status
```

- 3.10. Enfin, si les mises à jour nécessitent un redémarrage, ajoutez une tâche qui redémarre le serveur. Utilisez un conditionnel basé sur la sortie de la variable **updates\_status.reboot\_required**.

Nommez cette tâche **Server is rebooted**.

```
...output omitted...
- name: Server is rebooted
  win_reboot:
    when: updates_status.reboot_required
```

Une fois l'opération terminée, le premier play doit ressembler à ce qui suit :

```
- name: Manage Updates on Systems
hosts: all
vars:
  log_file: C:\Windows\Temp\ansible_available_updates.txt
  KB: 'KB4465065'
  access_token: ""

tasks:
  - name: Microsoft update servers are queried
    win_updates:
      category_names: SecurityUpdates
      state: searched
      log_path: "{{ log_file }}"

  - name: Available security updates are captured
    win_shell: "type {{ log_file }}"
    register: file_output

  - name: Available updates found
    debug:
      msg: "Available updates found!"
    when: "'Adding update' in file_output.stdout"

  - name: MSU package is retrieved on hosts
    win_uri:
      url: >
        https://gitlab.example.com/student/updates/raw/master/files/{{ KB }}.msu
      dest: C:\Windows\Temp\{{ KB }}.msu
      validate_certs: no
      force_basic_auth: true
      method: GET
      headers:
        PRIVATE-TOKEN: "{{ access_token }}"

  - name: MSU package is installed
    win_hotfix:
      hotfix_kb: "{{ KB }}"
      source: C:\Windows\Temp\{{ KB }}.msu
      state: present

  - name: Security updates are applied
    win_updates:
      category_name:
        - Updates
```

```
whitelist:  
  - KB4494174  
  - KB4346084  
register: updates_status  
  
- name: Server is rebooted  
  win_reboot:  
    when: updates_status.reboot_required
```

**Note**

Le fichier terminé est disponible dans le fichier **updates.sol**, dans le répertoire **solutions**.

- ▶ 4. Enregistrez le fichier avant de passer à l'étape suivante.
- ▶ 5. Dans ce second play, vous allez utiliser Chocolatey pour installer deux paquetages sur les hôtes gérés.
  - 5.1. Mettez à jour le deuxième play du fichier en lui donnant le nom **Install packages on systems** et en ciblant tous les hôtes pour le play. Pour ce faire, ajoutez la directive **hosts: all**.

**Note**

Veillez à mettre en retrait la directive **hosts** à l'aide de deux espaces.

```
...output omitted...  
- name: Install packages on systems  
  hosts: all
```

- 5.2. Ajoutez une directive **var** et créez une liste **packages** contenant deux paquetages à installer:
  - Putty
  - Wireshark

```
...output omitted...  
- name: Install packages on systems  
  hosts: all  
  vars:  
    packages:  
      - Putty  
      - Wireshark
```

- 5.3. Créez un bloc **tasks** sous la directive **hosts**.

```
...output omitted...  
- name: Install packages on systems  
  hosts: all  
  vars:
```

```
packages:  
  - Putty  
  - Wireshark
```

```
tasks:
```

- 5.4. Créez une tâche qui appelle le module **win\_chocolatey**. Ce module peut installer Chocolatey, s'il n'est pas déjà présent.

Nommez cette tâche **Chocolatey is installed**.

```
...output omitted...  
tasks:  
  - name: Chocolatey is installed  
    win_chocolatey:  
      name: chocolatey  
      state: present
```

- 5.5. Créez une nouvelle tâche qui installe les deux paquetages dans la liste **packages**. Utilisez le module **win\_chocolatey**, en transmettant la variable au module. Le paramètre **pinned** indique à Chocolatey de supprimer les mises à niveau pour les paquetages.

Nommez cette tâche **Packages are installed**.

```
...output omitted...  
  - name: Packages are installed  
    win_chocolatey:  
      name: "{{ packages }}"  
      state: present  
      pinned: true  
      register: install_result
```

- 5.6. Enfin, ajoutez une tâche pour redémarrer le serveur, si nécessaire. Cette tâche appelle le module **win\_reboot** et utilise l'instruction **when** pour déterminer s'il faut redémarrer ou non, sur la base des codes de retour de Chocolatey. Le code de retour 3010 indique que le serveur nécessite un redémarrage.

Nommez cette tâche **Server is rebooted**.

```
...output omitted...  
  - name: Server is rebooted  
    win_reboot:  
      when: install_result.changed and install_result.rc == 3010
```

- 5.7. Enregistrez le fichier.

Une fois l'opération terminée, le deuxième play doit ressembler à ce qui suit :

```
- name: Install packages on systems  
hosts: all  
vars:  
  packages:  
    - Putty  
    - Wireshark
```

```

tasks:
  - name: Chocolatey is installed
    win_chocolatey:
      name: chocolatey
      state: present

  - name: Packages are installed
    win_chocolatey:
      name: "{{ packages }}"
      state: present
      pinned: true
    register: install_result

  - name: Server is rebooted
    win_reboot:
      when: install_result.changed and install_result.rc == 3010
  
```

**Note**

Le fichier terminé est disponible dans le fichier **updates.sol**, dans le répertoire **solutions**.

## ▶ 6. Validez et envoyez vos modifications.

- 6.1. À partir de l'éditeur Visual Studio Code, indexez le playbook **updates.yml**. Pour ce faire, cliquez sur l'icône **Source Control**.

Cliquez sur le symbole + pour que le fichier **updates.yml** indexe vos modifications.

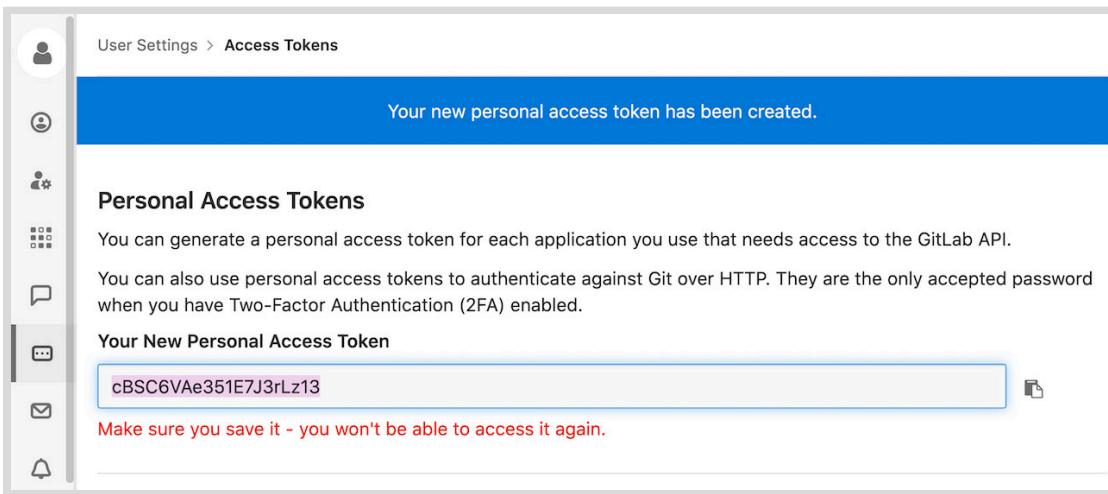
```

SOURCE CONTROL: GIT ✓ ⏪ ... ! updates.yml ✘ ...
Message (press Ctrl+Enter to commit)
CHANGES ! updates.yml 1 Stage Changes
! updates.yml
  32
  33 - name: MSU package is installed
  34   win_hotfix:
  35     hotfix_kb: "{{ KB }}"
  36     source: "C:\Windows\Temp\{{ KB }}.msu"
  37     state: present
  38
  39
  40 - name: Security updates are applied
  41   win_updates:
  42     category_name:
  43       - Updates
  44     whitelist:
  45       - KB4494174
  46       - KB4346084
  47     register: updates_status
  48
  49
  50 - name: Server is rebooted
  51   win_reboot:
  52     when: updates_status.reboot_required
  
```

- 6.2. Dans le champ du message de validation, entrez le message de validation **Define updates and Chocolatey tasks**, puis cliquez sur le bouton **commit**.
- 6.3. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur Synchronize Changes pour transmettre par push vos modifications locales

au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.

- ▶ 7. Créez un jeton d'accès GitLab qu'Ansible utilisera pour récupérer un fichier.
- 7.1. Dans GitLab, cliquez sur l'icône dans le coin supérieur droit et sélectionnez **Settings** pour accéder à vos paramètres de compte.
  - 7.2. Cliquez sur **Access Tokens**. Nommez le jeton **WindowsUpdates**, puis sélectionnez **api**. Accédez à votre **API** en tant qu'étendue. Cliquez sur **Create personal access token** pour créer le jeton.  
L'action génère un jeton et l'affiche dans le champ **Your New Personal Access Token**. Ne laissez pas cette page ouverte pour le moment, au cours d'une étape ultérieure, vous utiliserez le jeton.



- ▶ 8. Mettez à jour et exécutez le modèle de tâche **Run updates project** dans Ansible Tower.
- 8.1. À partir de **workstation**, double-cliquez sur l'icône Ansible Tower sur votre Bureau pour accéder à Ansible Tower. Connectez-vous à Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe
  - 8.2. Cliquez sur **Templates** pour accéder au modèle de tâche **Run updates project**. Cliquez sur le modèle pour le modifier.
  - 8.3. Mettez à jour le modèle en sélectionnant **updates .yml** dans le champ **PLAYBOOK**.  
Dans le cadre **EXTRA VARIABLES**, définissez la variable **access\_token**. Pour trouver cette valeur, affichez la page Chrome qui affiche le jeton d'accès que vous avez créé lors d'une étape précédente.

```
---  
access_token: cBSC6VAe351E7J3rLz13
```

Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.

## chapitre 5 | Installation et mise à jour de logiciels

**\* INVENTORY** Default inventory **PROMPT ON LAUNCH**

**\* PROJECT** updates repository **PROMPT ON LAUNCH**

**\* PLAYBOOK** updates.yml **PROMPT ON LAUNCH**

**CREDENTIAL** DevOps **PROMPT ON LAUNCH**

**FORKS** 0 **PROMPT ON LAUNCH**

**\* VERTBOSITY** 0 (Normal) **PROMPT ON LAUNCH**

**JOB TAGS** **PROMPT ON LAUNCH**

**LIMIT** **PROMPT ON LAUNCH**

**INSTANCE GROUPS** **PROMPT ON LAUNCH**

**SKIP TAGS** **PROMPT ON LAUNCH**

**JOB TAGS** **PROMPT ON LAUNCH**

**LABELS** **PROMPT ON LAUNCH**

**INSTANCE GROUPS** **PROMPT ON LAUNCH**

**JOB SLICING** 1 **PROMPT ON LAUNCH**

**TIMEOUT** 0 **PROMPT ON LAUNCH**

**SHOW CHANGES** OFF **PROMPT ON LAUNCH**

**OPTIONS**

- ENABLE PRIVILEGE ESCALATION
- ALLOW PROVISIONING CALLBACKS
- ENABLE CONCURRENT JOBS
- USE FACT CACHE

**EXTRA VARIABLES** **PROMPT ON LAUNCH**

```

1 ---
2 # Visit https://gitlab.example.com/profile/personal_access_tokens
3 # to generate an access token
4 access_token: cBSC6VAe351E7J3rLz13
  
```

- 8.4. Cliquez sur **LAUNCH** pour lancer une tâche à partir du modèle de tâche. Ansible Tower vous redirige vers la sortie de la tâche.

Inspectez la sortie pendant qu'Ansible Tower exécute le premier play qui récupère la liste des mises à jour de sécurité disponibles, installe le paquetage MSU et installe les deux KB. Le second play installe Chocolatey et les deux paquetages. Notez l'avertissement indiquant que si Chocolatey n'est pas présent sur le système, Ansible l'installe.

Ansible peut ignorer les tâches qui redémarrent les serveurs, ce qui est attendu si un paquetage ou une mise à jour est déjà présent.

DETAILS	
STATUS	Successful
STARTED	9/11/2019 3:41:48 PM
FINISHED	9/11/2019 3:43:44 PM
JOB TEMPLATE	Run updates project
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	Default inventory
PROJECT	updates repository
REVISION	db809c3
PLAYBOOK	updates.yml
CREDENTIAL	DevOps
ENVIRONMENT	/var/lib/awx/venv/ansible
EXECUTION NODE	localhost
INSTANCE GROUP	tower
EXTRA VARIABLES	<b>PROMPT ON LAUNCH</b>

```

1 access_token: cBSC6VAe351E7J3rLz13
  
```

```

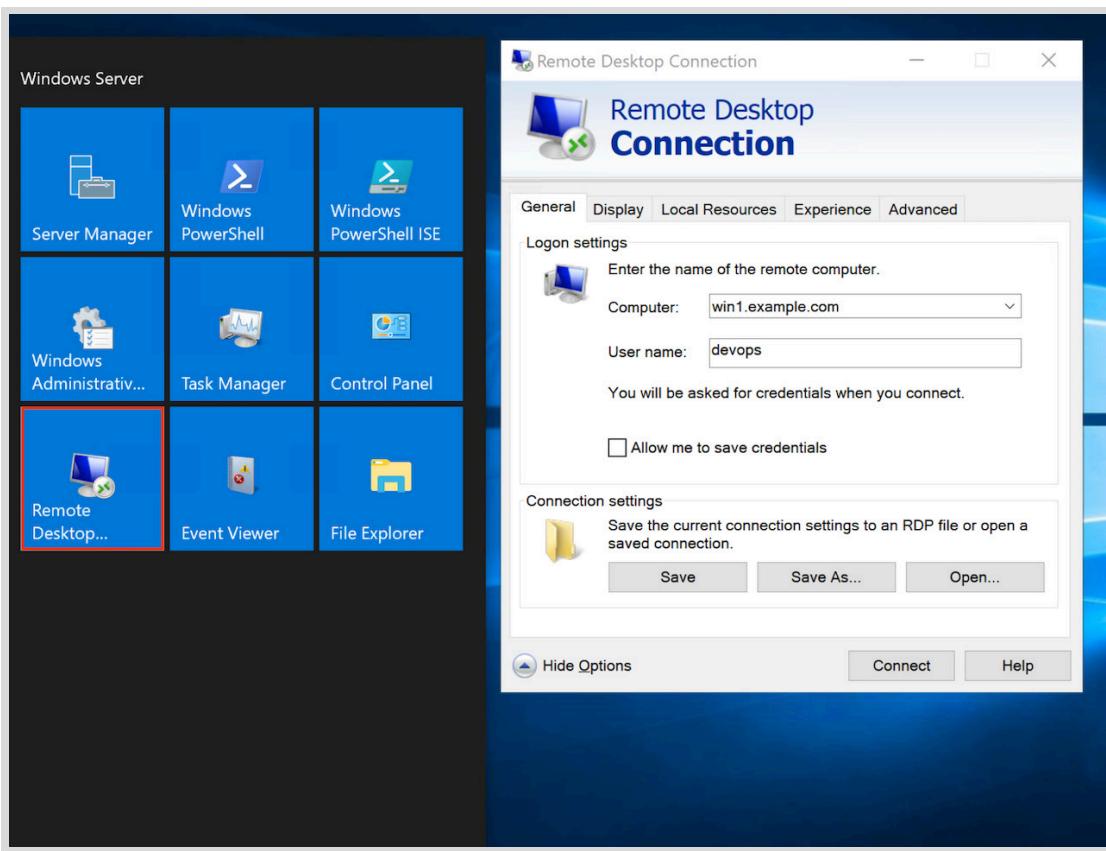
Run updates project
PLAYS 2 TASKS 11 HOSTS 2 ELAPSED 00:01:55
SEARCH Q KEY

44 *****
45 ok: [win1.example.com]
46 *****
47 TASK [Chocolatey is installed] 15:43:13
*****
48 changed: [win1.example.com]
49 changed: [win2.example.com]
50 *****
51 TASK [Packages are installed] 15:43:27
*****
52 changed: [win1.example.com]
53 changed: [win2.example.com]
54 *****
55 PLAY RECAP 15:43:44
*****
56 win1.example.com : ok=10 changed=4 unreachable=0 failed=0
d=0 skipped=1 rescued=0 ignored=0
57 win2.example.com : ok=10 changed=3 unreachable=0 failed=0
d=0 skipped=1 rescued=0 ignored=0
58
  
```

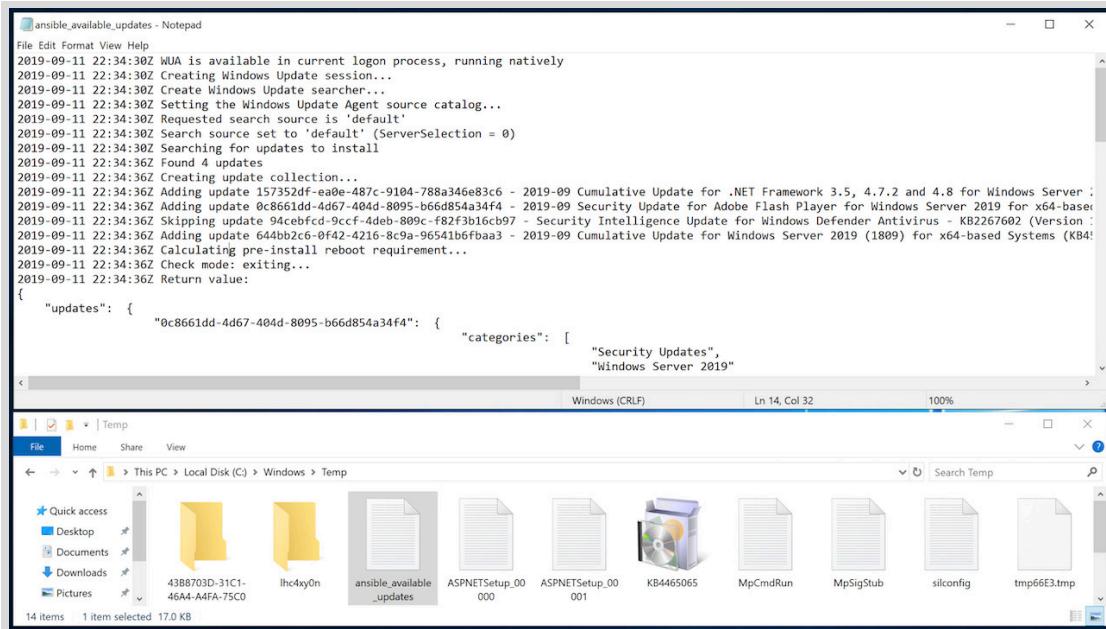
- 9. Examinez les modifications apportées aux hôtes gérés.

- 9.1. À partir de **workstation**, ouvrez le client du bureau à distance. Sélectionnez **Show options** pour afficher des options supplémentaires. Utilisez le nom d'ordinateur **win1.example.com** et le nom d'utilisateur **devops**.

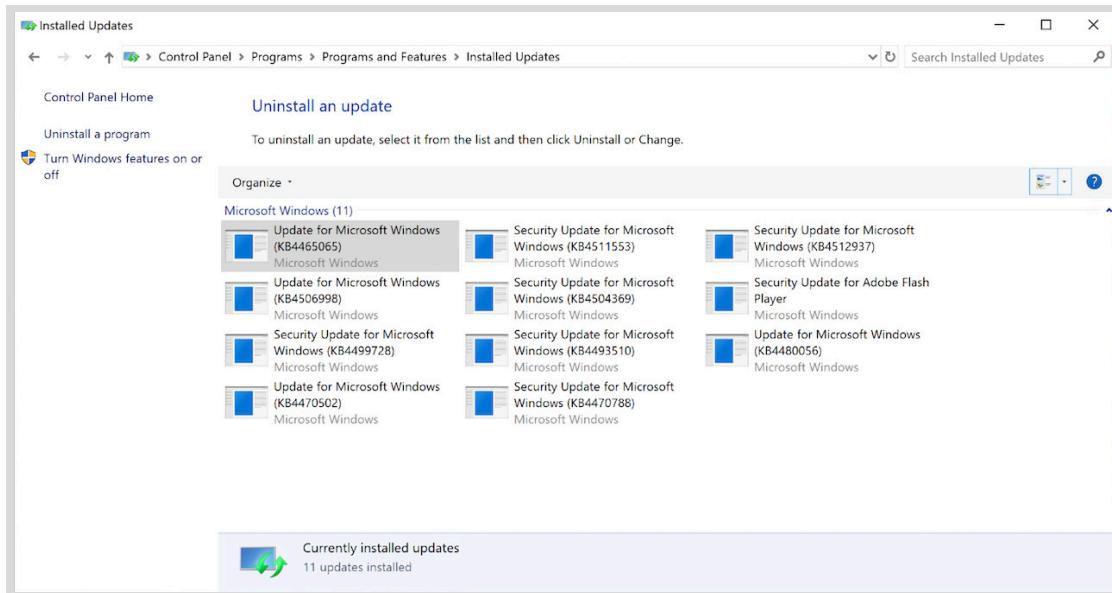
Si vous y êtes invité, acceptez le certificat auto-signé et utilisez **RedHat123@!** comme mot de passe.



- 9.2. Ouvrez l'explorateur de fichiers et accédez à **C:\Windows\Temp**. Dans l'avertissement qui s'affiche, cliquez sur **Continue** pour accéder au répertoire **Temp** protégé. Ouvrez le fichier texte **ansible\_available\_updates.txt** créé par Ansible. Ce fichier contient la liste des mises à jour de sécurité disponibles.



- 9.3. Accédez à **Start → Control Panel → Programs → Programs and Features** pour afficher la liste de tous les programmes installés.
- Sélectionnez **View installed updates** sur le côté de la fenêtre pour afficher la liste des mises à jour Windows. Recherchez la mise à jour nommée **KB4465065**, qui est celle qu'Ansible a installée.

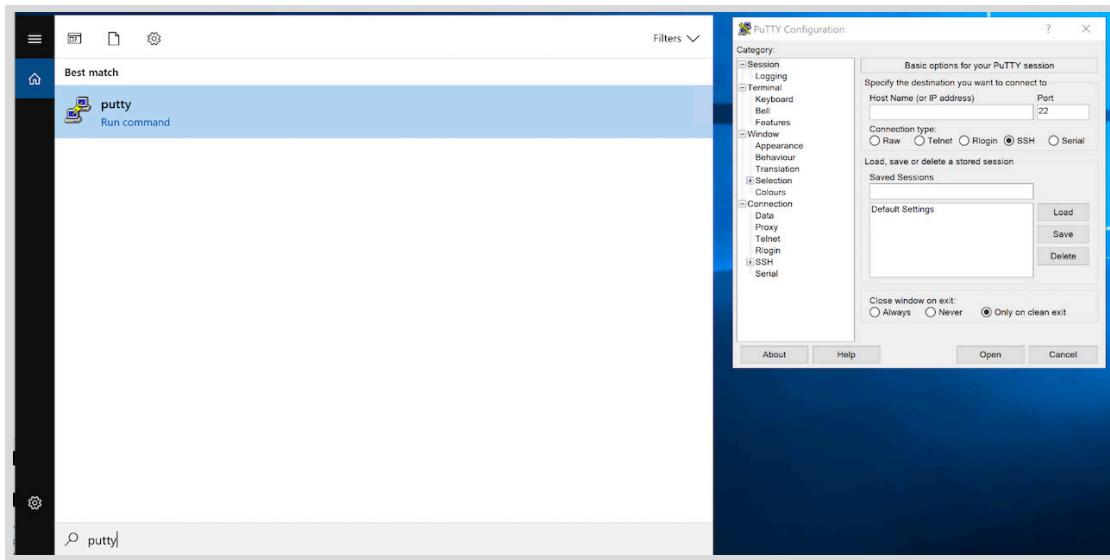


- 9.4. Fermez la session RDP sur **win1.example.com** et relancez le client RDP. Utilisez le nom d'ordinateur **win2.example.com** et le nom d'utilisateur **devops**. Si vous y êtes invité, acceptez le certificat auto-signé et utilisez **RedHat123@!** comme mot de passe.
- 9.5. Pour vous assurer que Putty est installé, cliquez sur l'icône de loupe et saisissez **Putty**. La recherche renvoie un résultat confirmant que Putty a été correctement installé.

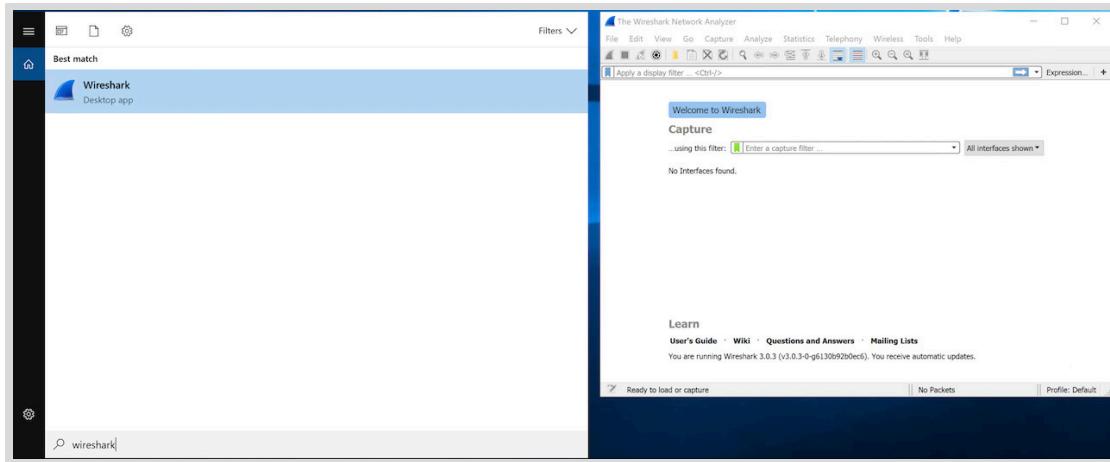


### Note

Si le paquetage n'est pas visible dans les résultats de la recherche, redémarrez le serveur.



9.6. Répétez les mêmes étapes pour le paquetage Wireshark. La recherche renvoie un résultat.



L'exercice guidé est maintenant terminé.

# Modification du Registre

---

## Résultats

Au terme de cette section, vous serez en mesure d'inspecter le Registre Windows et de vous assurer que les clés de Registre sont correctement configurées.

## Examen de la terminologie du Registre Windows

Le Registre Windows est une base de données hiérarchique, conçue pour mettre en œuvre les mises à jour atomiques. Les mises à jour atomiques garantissent que les mises à jour simultanées ne puissent pas interférer les unes avec les autres.

Le Registre sert à stocker les informations de configuration du système d'exploitation et de la plupart des paquetages logiciels. Bien qu'il ne soit pas obligatoire que le logiciel stocke sa configuration dans le Registre, cela est recommandé. Dans le Registre Windows, vous pouvez également trouver des informations telles que la configuration du service et le mode de démarrage, l'état de la licence logicielle, les informations spécifiques à l'utilisateur actuel et à tous les utilisateurs, ainsi que la configuration et les gestionnaires pour divers types de fichiers.

Les listes suivantes décrivent une partie de la terminologie du Registre Windows.

### Conditions générales du Registre

#### Ruche

Le Registre est stocké physiquement dans plusieurs fichiers appelés *ruches*. Une ruche est utilisée pour les données associées, par exemple le logiciel installé sur la machine locale.

#### Clé

Les clés fournissent la hiérarchie, car une clé peut contenir des valeurs ou d'autres clés. Vous pouvez faire correspondre les clés aux répertoires dans un système de fichiers.

#### Valeur

Une valeur a un nom et un type. Le type définit la manière dont les données qu'il contient seront interprétées. La liste suivante décrit les types de données possibles.

- Chaîne : série de caractères
- Binaire : vrai ou faux, oui ou non, 1 ou 0
- Dword ou Qword : valeurs 32 or 64 bits
- Multi-chaine : valeur qui peut contenir des données avec plusieurs lignes de chaînes
- Chaîne extensible : identique à une chaîne multiple, mais peut contenir des variables qui sont développées, comme des variables d'environnement telles que %HOMEPATH%

Les ruches du Registre sont stockées sous **C:\Windows\System32\Config**, en supposant que Windows soit installé sur le premier lecteur. Les fichiers sont nommés pour le type de configuration qu'ils contiennent, tels que **SOFTWARE**, **SYSTEM**, **SAM**, **SECURITY** et **DEFAULT**. Dans la liste précédente, **SAM** est l'abréviation de *Security Account Manager*, et cette ruche contient des informations sur le compte d'utilisateur.

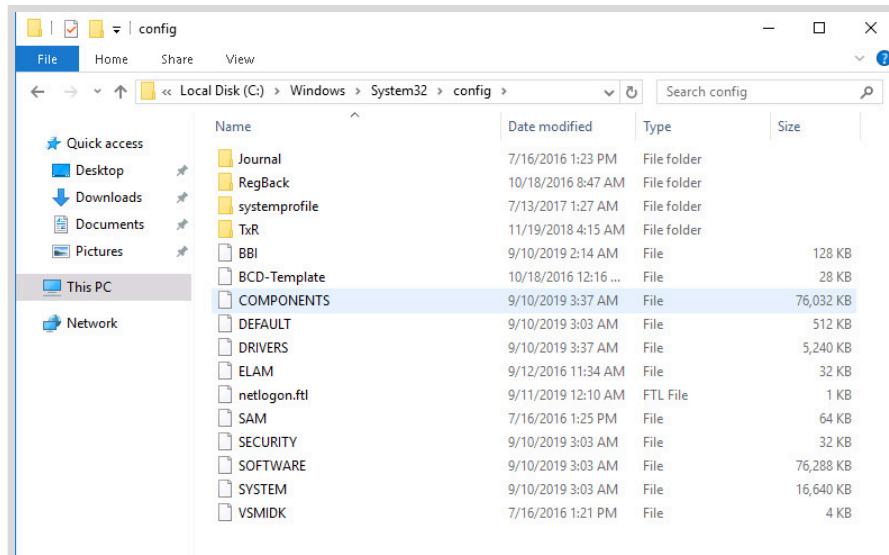


Figure 5.14: Fichiers de riche du Registre Windows

## Modification du Registre Windows

Étant donné que Windows stocke la plupart des configurations de programmes au sein du Registre, la possibilité de modifier la configuration du programme nécessite des modules qui peuvent interagir avec le Registre.

Windows fournit l'outil **regedit** permettant de modifier le Registre. Dans les versions précédentes de Windows, il existait un outil appelé **regedt32**, qui était la première version 32 bits. Il est désormais obsolète et lance simplement **regedit** à la place.

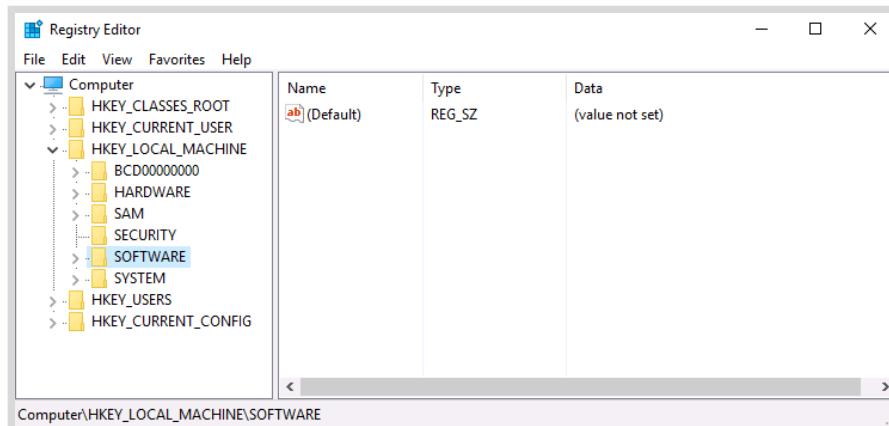


Figure 5.15: Éditeur du Registre Windows

Certaines des tâches courantes consistent à vérifier l'existence d'une clé, ou à renvoyer les sous-clés et les valeurs de la clé, ou encore à renvoyer une valeur spécifique. Le module **win\_reg\_stat** est conçu à cet effet et exécute une fonction similaire pour le module **stat**.

## Lecture des clés et des valeurs de Registre

Le module **win\_reg\_stat** peut être utilisé pour lire et vérifier les données contenues dans une valeur de Registre, ou pour confirmer qu'il existe une clé de Registre.

## chapitre 5 | Installation et mise à jour de logiciels

L'exemple suivant illustre l'utilisation de **win\_reg\_stat** pour obtenir des informations de licence. La variable inscrite dans le Registre permet d'effectuer les tâches suivantes de manière conditionnelle.

```
- name: Obtain myapp license information
  win_reg_stat:
    path: HKLM:\SOFTWARE\MySoft\MyApp\Licensing
    name: LicenseType
    register: myapp_license_type
```

## Modification des clés et des valeurs de Registre

Le module **win\_regedit** peut être utilisé pour créer une clé, pour ajouter une valeur ou pour modifier les données figurant dans une valeur.

Suite à l'exemple précédent, l'application est inscrite dans le Registre en fonction de la valeur stockée dans **myapp\_license\_type**.

```
- name: Set myapp registration type
  win_regedit:
    path: HKLM:\SOFTWARE\MySoft\MyApp\Licensing
    name: LicenseType
    data: "FULL"
    type: string
    when: myapp_license_type.value == "EVAL"

- name: Add myapp registration key
  win_regedit:
    path: HKLM:\SOFTWARE\MySoft\MyApp\Licensing
    name: LicenseKey
    data: "{{ myapp_key }}"
    type: dword
    when: myapp_license_type.value == "EVAL"
```

S'il y a trop de valeurs de Registre à gérer individuellement, il peut être plus simple d'exporter une clé de Registre à partir d'un système dans l'état souhaité, puis de la fusionner dans le Registre d'autres systèmes.

## Utilisation du module **win\_regmerge**

Le module **win\_regmerge** peut être utilisé pour fusionner un groupe de clés et de valeurs dans le Registre d'un hôte géré.



### Note

Sachez que les fichiers exportés à l'aide de **regedit** peuvent contenir une marque d'ordre d'octet comme premier caractère. Une marque d'ordre d'octet au début d'un fichier indique le codage Unicode et le mode endian pour le fichier. Ce caractère devra être supprimé pour utiliser le fichier exporté en tant que modèle.

Ce module ne force pas les paramètres du Registre à correspondre au fichier exporté. Vous devez d'abord supprimer la clé concernée, comme illustré dans l'exemple suivant.

```
- name: Remove myapp licensing key
  win_regedit:
    path: HKLM:\SOFTWARE\MySoft\MyApp\Licensing
    state: absent

- name: Register myapp
  win_regmerge:
    path: files/myapp-license.reg
```



## Références

**win\_reg\_stat : obtenir des informations sur les clés de Registre Windows**  
-> **Documentation Ansible**

[https://docs.ansible.com/ansible/latest/modules/win\\_reg\\_stat\\_module.html#win-reg-stat-module](https://docs.ansible.com/ansible/latest/modules/win_reg_stat_module.html#win-reg-stat-module)

**win\_regedit : ajouter, modifier ou supprimer des clés et des valeurs de Registre**  
-> **Documentation Ansible**

[https://docs.ansible.com/ansible/latest/modules/win\\_regedit\\_module.html#win-regedit-module](https://docs.ansible.com/ansible/latest/modules/win_regedit_module.html#win-regedit-module)

**win\_regmerge : fusionne le contenu d'un fichier de Registre dans le Registre Windows**  
-> **Documentation Ansible**

[https://docs.ansible.com/ansible/latest/modules/win\\_regmerge\\_module.html#win-regmerge-module](https://docs.ansible.com/ansible/latest/modules/win_regmerge_module.html#win-regmerge-module)

## ► Exercice guidé

# Modification du Registre

Dans cet exercice, vous allez déterminer si une clé de Registre Windows existe et vous assurer qu'une ou plusieurs clés spécifiques du Registre sont correctement définies.

## Résultats

Vous serez en mesure d'inspecter le Registre Windows et de vous assurer que les clés de Registre sont correctement configurées.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



### Note

Au cours de cet exercice, vous allez configurer le service de temps Windows pour vous synchroniser à une source de temps externe. La configuration du service de temps Windows pour une synchronisation avec une source de temps externe n'est généralement pas effectuée sur une machine qui est membre d'un domaine. Les membres du domaine synchronisent normalement l'heure sur un contrôleur de domaine automatiquement, ce qui évite les problèmes d'authentification.

- 1. Connectez-vous à **win1.example.com** pour vérifier la configuration actuelle du service W32Time.

- 1.1. Cliquez sur **Recherche Windows**, recherchez **mstsc**, puis ouvrez Connexion Bureau à distance.



### Note

Le programme Connexion Bureau à distance est également connu sous le nom de Microsoft Terminal Services Client (**mstsc.exe**).

- 1.2. Pour vous connecter au Bureau à distance, saisissez **win1.example.com** pour **Computer** et **devops** pour **User Name** et **RedHat123@!** comme mot de passe. Notez que **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE** dans le champ username.

Lorsque vous y êtes invité, accédez au certificat non sécurisé.

- 1.3. Cliquez sur **Recherche Windows**, puis tapez **regedit**. Ouvrez l'éditeur du Registre. Lorsque vous y êtes invité, cliquez sur **Yes** pour ouvrir l'éditeur du Registre.
- 1.4. Accédez à la clé **HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\services\W32Time**. Notez que la valeur de **Start** est actuellement **2**. Le tableau suivant contient les valeurs possibles pour les pilotes et les services.

#### Valeurs de départ de service/pilote

Valeur	Mode de démarrage	Description
0	Boot	Le pilote de périphérique est chargé au démarrage, et si aucune erreur ne se produit, il est démarré lors de l'initialisation du noyau.
1	System	Le pilote de périphérique est démarré lors de l'initialisation du noyau, après les pilotes de périphérique de démarrage.
2	Automatic	Le service est lancé automatiquement par le <i>Gestionnaire de contrôle des services (SCM)</i> .
3	Manual	Le service doit être démarré manuellement par un administrateur.
4	Disabled	Le service ne peut pas être démarré par le SCM ou un administrateur.

Un administrateur peut également définir un service sur **Delayed Start** à l'aide de l'applet services ou en modifiant le Registre.

Vérifiez que la valeur de **Parameters\NtpServer** fait référence à une adresse 169.254.x.x, que **Parameters\Type** est égal à **AllSync**, que **Config\AnnounceFlags** est égal à 0xA ou la décimale 10, et que **Config\MaxPosPhaseCorrection** est défini sur la valeur maximale possible pour 32 bits.

- 1.5. Fermez l'éditeur du Registre et déconnectez la session à distance.
- ▶ 2. Lancez l'éditeur Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **updates**, clonez-le sur votre instance **workstation**.
  - 2.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 2.2. Ouvrez la palette de commandes en accédant à **View → Command Palette** ou en appuyant sur **Ctrl+Maj+P**.
 

Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel. Utilisez l'URL de référentiel <https://gitlab.example.com/student/updates.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **updates**.

Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers, puis passez à l'étape suivante.

▶ 3. Ouvrez le répertoire **c:\Users\student\Documents\updates** en accédant à **File → Open Folder**, puis sélectionnez le dossier **C:\Users\student\Documents\updates**.

▶ 4. Ouvrez le playbook **time.yml**, puis effectuez les tâches dans le fichier. Notez que la variable **time\_key** a été définie pour vous.

4.1. Modifiez la tâche nommée **Configure time server type** pour définir le type de serveur sur **NTP**. Cela modifie la valeur **Parameters\Type** dans la clé W32Time.

```
- name: Configure time server type
  win_regedit:
    path: '{{ time_key }}\Parameters'
    name: Type
    data: NTP
    type: string
  notify: Restart windows time service
```

4.2. Modifiez la tâche nommée **Configure AnnounceFlags** pour définir la valeur sur 5. Cela modifie la valeur **Config\AnnounceFlags** dans la clé W32Time.

```
- name: Configure AnnounceFlags
  win_regedit:
    path: '{{ time_key }}\Config'
    name: AnnounceFlags
    data: 5
    type: dword
  notify: Restart windows time service
```

4.3. Modifiez la tâche nommée **Enable NTP Time Provider** pour activer le fournisseur **NTP**. Cela modifie la valeur **TimeProviders\NtpServer\Enabled** dans la clé W32Time.

```
- name: Enable NTP time provider
  win_regedit:
    path: '{{ time_key }}\TimeProviders\NtpServer'
    name: Enabled
    data: 1
    type: dword
  notify: Restart windows time service
```

4.4. Modifiez la tâche nommée **Configure upstream servers** pour définir **0.rhel.pool.ntp.org** en tant que serveur NTP en amont. Cela modifie la valeur **Parameters\NtpServer** dans la clé W32Time.

```
- name: Configure upstream servers
  win_regedit:
    path: '{{ time_key }}\Parameters'
    name: NtpServer
    data: 0.rhel.pool.ntp.org,0x1
    type: string
  notify: Restart windows time service
```

- 4.5. Modifiez les tâches nommées **Configure positive time correction** et **Configure negative time correction** pour définir la valeur de la correction de temps maximale sur 3 600 secondes. Cette opération modifie les valeurs **Config\MaxPosPhaseCorrection** et **Config\MaxNegPhaseCorrection** dans la clé W32Time.

```
- name: Configure positive time correction
  win_regedit:
    path: '{{ time_key }}\Config'
    name: MaxPosPhaseCorrection
    data: 3600
    type: dword
  notify: Restart windows time service
```

```
- name: Configure negative time correction
  win_regedit:
    path: '{{ time_key }}\Config'
    name: MaxNegPhaseCorrection
    data: 3600
    type: dword
  notify: Restart windows time service
```

- 5. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
- 5.1. Cliquez sur **File** → **Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.
  - 5.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard du fichier **time.yml** pour indexer les modifications.
  - 5.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 5.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- 6. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- 7. Testez votre playbook à l'aide du modèle de tâche **Run updates project**.
- 7.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 7.2. Cliquez sur le modèle de tâche **Run updates project** pour le modifier.
  - 7.3. Sélectionnez le playbook **time.yml** dans la liste **PLAYBOOK**.
  - 7.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

- 8. Une fois toutes les tâches terminées, vérifiez que l'option **STATUS** dans la section **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec le fichier de solution **solutions/time.sol** dans le référentiel Git **updates**.
- 9. Connectez-vous à l'hôte géré **win1.example.com** et vérifiez les modifications apportées au service W32Time.
- 9.1. Cliquez sur **Recherche Windows**, recherchez **mstsc**, puis ouvrez Connexion Bureau à distance.
  - 9.2. Saisissez **win1.example.com** pour **Computer** et **devops** pour **User Name** et **RedHat123@!** comme mot de passe. Notez que **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.
  - 9.3. Lancez l'éditeur du Registre, puis accédez à la clé W32Time. Notez que les valeurs affichées au début de l'exercice ont été modifiées, en raison des tâches du fichier de playbook **time.yml**. Fermez l'éditeur du Registre.
  - 9.4. Cliquez sur **Recherche Windows**, recherchez **cmd**, puis ouvrez le shell de commande.
  - 9.5. Exécutez la commande **w32tm /query /peers** pour afficher l'état de la synchronisation de l'heure.

```
C:\Users\devops>w32tm /query /peers
##Peers: 2

Peer: windc.example.com
State: Active
Time Remaining: 28.3535922s
Mode: 1 (Symmetric Active)
Stratum: 3 (secondary reference - syncd by (S)NTP)
PeerPoll Interval: 17 (out of valid range)
HostPoll Interval: 6 (64s)

Peer: 0.rhel.pool.ntp.org,0x1
State: Active
Time Remaining: 748.3535922s
Mode: 1 (Symmetric Active)
Stratum: 3 (secondary reference - syncd by (S)NTP)
PeerPoll Interval: 17 (out of valid range)
HostPoll Interval: 6 (64s)
```

- 9.6. Quittez le shell de commande, puis déconnectez-vous de **win1.example.com**.

► 10. Revenez à l'interface utilisateur Web d'Ansible Tower. Annulez vos modifications en mettant à jour le modèle de tâche **Run updates project**.




#### Note

Cette étape est nécessaire pour éviter d'éventuels problèmes d'authentification lors des exercices à venir.

- 10.1. Dans le volet de navigation, cliquez sur **Templates**.
- 10.2. Cliquez sur le modèle de tâche **Run updates project** pour le modifier.

10.3. Sélectionnez le playbook **revert\_time.yml** dans la liste **PLAYBOOK**.

10.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Installation et mise à jour de logiciels

### Liste de contrôle des performances

Au cours de cet atelier, vous allez vous assurer que des paquetages logiciels spécifiques sont installés et à jour, et que des clés de Registre spécifiques sont configurées.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Rédiger un playbook Ansible qui installe des paquetages Chocolatey et fusionne les données dans le Registre Windows.
- Générer un jeton d'accès d'API GitLab.
- Transmettre les variables d'Ansible Tower aux playbooks.
- Accéder au Registre Windows pour passer en revue les modifications.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. À partir du **poste de travail** Windows, ouvrez l'éditeur Visual Studio Code et clonez le référentiel **updates**, disponible à l'adresse <https://gitlab.example.com/student/updates.git>, vers **C:\Users\student\Documents\updates** si le dossier n'y figure pas déjà.
2. Ajoutez le dossier **updates** à votre espace de travail Visual Studio Code, puis procédez aux modifications suivantes dans le playbook **review.yml** :
  - Définissez la variable **package\_name** avec la valeur **Atom**. Il s'agit du nom du paquetage chocolaté pour l'installation de l'éditeur de texte Atom.
  - Définissez la variable **package\_url** avec la valeur <https://chocolatey.org/api/v2/package/Atom/1.39.1>. Cette URL spécifie l'emplacement du paquetage source Atom dans la version 1.39.1.
  - Définissez la variable **package\_local\_path** avec la valeur **C:\Windows\Temp\atom.nupkg**. Ce chemin indique l'emplacement d'enregistrement du fichier de paquetage.
  - Créez une tâche qui récupère le paquetage source Atom, tel qu'il est défini par la variable **package\_url**. Utilisez **package\_local\_path** pour le chemin d'accès de l'emplacement où le paquetage est enregistré.

Nommez la tâche **Atom package is retrieved on managed hosts**.

- Créez une tâche qui appelle le module **win\_chocolatey** pour installer le paquetage Atom. Configurez la tâche de sorte qu'elle utilise la variable **package\_name** comme nom du paquetage à installer et **package\_local\_path** comme emplacement du paquetage.

Nommez la tâche **Atom is installed from a local source**.

- Créez une tâche qui utilise le module **win\_uri** pour récupérer le fichier de Registre **RedHatTraining.reg**, disponible à l'adresse <https://gitlab.example.com/student/updates/blob/master/files/RedHatTraining.reg>. Nommez la tâche **Registry file is retrieved**.

Étant donné que l'URL est protégée par un mot de passe, utilisez le paramètre **headers** pour transmettre l'en-tête **PRIVATE-TOKEN** à la variable Ansible **access\_token**. Lors d'une étape ultérieure, vous allez générer ce jeton.

Enregistrez le fichier dans **C:\Windows\Temp\RedHatTraining.reg**.

- Créez une tâche qui appelle le module **win\_regmerge** pour mettre à jour le Registre en fusionnant les valeurs spécifiées dans le fichier de Registre.

Nommez la tâche **Registry is updated**.



#### Note

Le fichier terminé est disponible dans **solutions/review.sol** dans le dossier **updates**.

- Enregistrez vos modifications, puis validez le fichier avec le message **Configure Atom on managed hosts**.

Poussez vos modifications vers la branche master du référentiel **updates**.

- Accédez à votre instance GitLab sur <https://gitlab.example.com>, puis connectez-vous à l'aide du compte **student** et du mot de passe **Redhat123@!**.

Générez un jeton pour **student** disposant d'un accès à l'API. Nommez le jeton **UpdatesReview**.

- À partir de votre poste de travail, accédez à Ansible Tower sur <https://tower.example.com>. Connectez-vous en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Mettez à jour le modèle de tâche **Run updates project** comme suit :

- Sélectionnez **review.yml** comme playbook.
  - Limitez le play à **win1.example.com**.
  - Transmettez **access\_token** en tant que variable supplémentaire. Attribuez-lui la valeur du jeton GitLab que vous avez générée au cours de l'étape précédente.
- Exécutez le modèle de tâche **Run updates project** et assurez-vous qu'il s'exécute correctement.

7. À partir de **workstation**, ouvrez votre client RDP et connectez-vous à **win1.example.com** en utilisant **devops** comme nom d'utilisateur avec le mot de passe **RedHat123@!**. Lorsque vous y êtes invité, acceptez le certificat non sécurisé. Vérifiez qu'Atom est installé. Vous pouvez rechercher l'icône du Bureau. Accédez au registre pour vous assurer que la clé **RedHat** est disponible dans **HKCU:\Software**.

L'atelier est maintenant terminé.

## ► Solution

# Installation et mise à jour de logiciels

## Liste de contrôle des performances

Au cours de cet atelier, vous allez vous assurer que des paquetages logiciels spécifiques sont installés et à jour, et que des clés de Registre spécifiques sont configurées.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Rédiger un playbook Ansible qui installe des paquetages Chocolatey et fusionne les données dans le Registre Windows.
- Générer un jeton d'accès d'API GitLab.
- Transmettre les variables d'Ansible Tower aux playbooks.
- Accéder au Registre Windows pour passer en revue les modifications.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. À partir du **poste de travail** Windows, ouvrez l'éditeur Visual Studio Code et clonez le référentiel **updates**, disponible à l'adresse <https://gitlab.example.com/student/updates.git>, vers **C:\Users\student\Documents\updates** si le dossier n'y figure pas déjà.



### Note

Les étapes suivantes décrivent la façon de cloner le référentiel **updates**. Si vous avez déjà cloné le référentiel lors d'une étape précédente, vous pouvez ignorer ces étapes.

- 1.1. À partir du **poste de travail** Windows, ouvrez l'éditeur Visual Studio Code en double-cliquant sur le Bureau.
- 1.2. Pour cloner le référentiel **updates**, ouvrez la palette de commandes en accédant à **View → Command Palette** ou en appuyant sur **Ctrl+Maj+P**. Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel. Utilisez l'URL de référentiel <https://gitlab.example.com/student/updates.git>. À l'invite, sélectionnez le dossier **Documents** dans le répertoire

personnel de l'utilisateur **training** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **updates**.

Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage.

2. Ajoutez le dossier **updates** à votre espace de travail Visual Studio Code, puis procédez aux modifications suivantes dans le playbook **review.yml**:

- Définissez la variable **package\_name** avec la valeur **Atom**. Il s'agit du nom du paquetage chocolaté pour l'installation de l'éditeur de texte Atom.
- Définissez la variable **package\_url** avec la valeur <https://chocolatey.org/api/v2/package/Atom/1.39.1>. Cette URL spécifie l'emplacement du paquetage source Atom dans la version 1.39.1.
- Définissez la variable **package\_local\_path** avec la valeur C:\Windows\Temp\Atom.nupkg. Ce chemin indique l'emplacement d'enregistrement du fichier de paquetage.
- Créez une tâche qui récupère le paquetage source Atom, tel qu'il est défini par la variable **package\_url**. Utilisez **package\_local\_path** pour le chemin d'accès de l'emplacement où le paquetage est enregistré.

Nommez la tâche **Atom package is retrieved on managed hosts**.

- Créez une tâche qui appelle le module **win\_chocolatey** pour installer le paquetage Atom. Configurez la tâche de sorte qu'elle utilise la variable **package\_name** comme nom du paquetage à installer et **package\_local\_path** comme emplacement du paquetage.

Nommez la tâche **Atom is installed from a local source**.

- Créez une tâche qui utilise le module **win\_uri** pour récupérer le fichier de Registre **RedHatTraining.reg**, disponible à l'adresse <https://gitlab.example.com/student/updates/blob/master/files/RedHatTraining.reg>. Nommez la tâche **Registry file is retrieved**.

Étant donné que l'URL est protégée par un mot de passe, utilisez le paramètre **headers** pour transmettre l'en-tête **PRIVATE-TOKEN** à la variable Ansible **access\_token**. Lors d'une étape ultérieure, vous allez générer ce jeton.

Enregistrez le fichier dans **C:\Windows\Temp\RedHatTraining.reg**.

- Créez une tâche qui appelle le module **win\_regmerge** pour mettre à jour le Registre en fusionnant les valeurs spécifiées dans le fichier de Registre.

Nommez la tâche **Registry is updated**.

- 2.1. Dans Visual Studio Code, ouvrez le playbook **review.yml** et procédez aux modifications suivantes :

- 2.1.1. Définissez la variable **package\_name** avec la valeur **Atom**, comme suit :

```
---  
- name: Install Atom using Chocolatey and update the registry  
  hosts: all  
  vars:  
    package_name: Atom  
...output omitted...
```

- 2.1.2. Définissez la variable **package\_url** avec la valeur `https://chocolatey.org/api/v2/package/Atom/1.39.1`. Cette URL spécifie l'emplacement du paquetage source Atom `.nupkg`.

```
---
- name: Install Atom using Chocolatey and update the registry
hosts: all
vars:
  package_name: Atom
  package_url: https://chocolatey.org/api/v2/package/Atom/1.39.1
...output omitted...
```

- 2.1.3. Définissez la variable **package\_local\_path** avec la valeur `C:\Windows\Temp\atom.nupkg`. Ce chemin indique l'emplacement d'enregistrement du fichier.

```
---
- name: Install Atom using Chocolatey and update the registry
hosts: all
vars:
  package_name: Atom
  package_url: https://chocolatey.org/api/v2/package/Atom/1.39.1
  package_local_path: C:\Windows\Temp\atom.nupkg
...output omitted...
```

- 2.1.4. Créez une tâche qui récupère le paquetage source Atom, tel qu'il est défini par la variable **package\_url**, à l'aide du module **win\_uri**. Enregistrez le paquetage dans le chemin d'accès défini par la variable **package\_local\_path**.

Nommez la tâche **Atom package is retrieved on managed hosts**.

```
---
- name: Install Atom using Chocolatey and update the registry
hosts: all
vars:
  package_name: Atom
  package_url: https://chocolatey.org/api/v2/package/Atom/1.39.1
```

```
package_local_path: C:\Windows\Temp\atom.nupkg

tasks:
  - name: Atom package is retrieved on managed hosts
    win_uri:
      url: "{{ package_url }}"
      dest: "{{ package_local_path }}"
      method: GET
```

- 2.1.5. Créez une tâche qui appelle le module **win\_chocolatey** pour installer le paquetage Atom, à partir du fichier que vous avez récupéré. Nommez la tâche **Atom is installed from a local source**.

Configurez la tâche de sorte qu'elle utilise la variable **package\_name** comme nom du paquetage à installer et la variable **package\_local\_path** comme emplacement du paquetage.



#### Note

Si vous devez réinstaller le paquetage, transmettez le paramètre **force: yes** au module **win\_chocolatey**.

Le playbook doit se présenter comme suit :

```
---
- name: Install Atom using Chocolatey and update the registry
  hosts: all
  vars:
    package_name: Atom
    package_url: https://chocolatey.org/api/v2/package/Atom/1.39.1
    package_local_path: C:\Windows\Temp\atom.nupkg

  tasks:
    - name: Atom package is retrieved on managed hosts
      win_uri:
        url: "{{ package_url }}"
        dest: "{{ package_local_path }}"
        method: GET

    - name: Atom is installed from a local source
      win_chocolatey:
        name: "{{ package_name }}"
        source: "{{ package_local_path }}"
        state: present
```

- 2.1.6. Créez une tâche qui utilise le module **win\_uri** pour récupérer le fichier de Registre **RedHatTraining.reg**, disponible à l'adresse <https://>

gitlab.example.com/student/updates/blob/master/files/  
RedHatTraining.reg. Nommez la tâche **Registry file is retrieved**.

Étant donné que l'URL est protégée par un mot de passe, utilisez le paramètre **headers** pour transmettre l'en-tête **PRIVATE-TOKEN** à la variable Ansible **access\_token**. Lors d'une étape ultérieure, vous allez générer ce jeton.

Enregistrez le fichier dans **C:\Windows\Temp\RedHatTraining.reg**.



### Note

Notez la mise en retrait supplémentaire des données d'en-tête.

```
...output omitted...
- name: Registry file is retrieved
  win_uri:
    url: >
      https://gitlab.example.com/student/updates/blob/master/files/
      RedHatTraining.reg
    dest: C:\Windows\Temp\RedHatTraining.reg
    validate_certs: no
    force_basic_auth: yes
    method: GET
    headers:
      PRIVATE-TOKEN: "{{ access_token }}"
```

2.1.7. Créez une tâche qui appelle le module **win\_regmerge** pour mettre à jour le Registre en fusionnant les valeurs spécifiées dans le fichier de Registre.

Nommez la tâche **Registry is updated**.

La tâche doit se présenter comme suit :

```
...output omitted...
- name: Registry file is retrieved
  win_uri:
    url: >
      https://gitlab.example.com/student/updates/blob/master/files/
      RedHatTraining.reg
    dest: C:\Windows\Temp\RedHatTraining.reg
    validate_certs: no
    force_basic_auth: yes
    method: GET
```

```
headers:  
    PRIVATE-TOKEN: "{{ access_token }}"  
  
- name: Registry is updated  
  win_regmerge:  
    path: C:\Windows\Temp\RedHatTraining.reg
```

Le playbook final doit se présenter comme suit :

```
---  
- name: Install Atom using Chocolatey and update the registry  
  hosts: all  
  vars:  
    package_name: Atom  
    package_url: https://chocolatey.org/api/v2/package/Atom/1.39.1  
    package_local_path: C:\Windows\Temp\atom.nupkg  
  
  tasks:  
    - name: Atom package is retrieved on managed hosts  
      win_uri:  
        url: "{{ package_url }}"  
        dest: "{{ package_local_path }}"  
        method: GET  
  
    - name: Atom is installed  
      win_chocolatey:  
        name: "{{ package_name }}"  
        source: "{{ package_local_path }}"  
        state: present  
  
    - name: Registry file is retrieved  
      win_uri:  
        url: >  
          https://gitlab.example.com/student/updates/raw/master/files/  
RedHatTraining.reg  
        dest: C:\Windows\Temp\RedHatTraining.reg  
        validate_certs: no  
        force_basic_auth: yes  
        method: GET  
        headers:  
          PRIVATE-TOKEN: "{{ access_token }}"  
  
    - name: Registry is updated  
      win_regmerge:  
        path: C:\Windows\Temp\RedHatTraining.reg
```



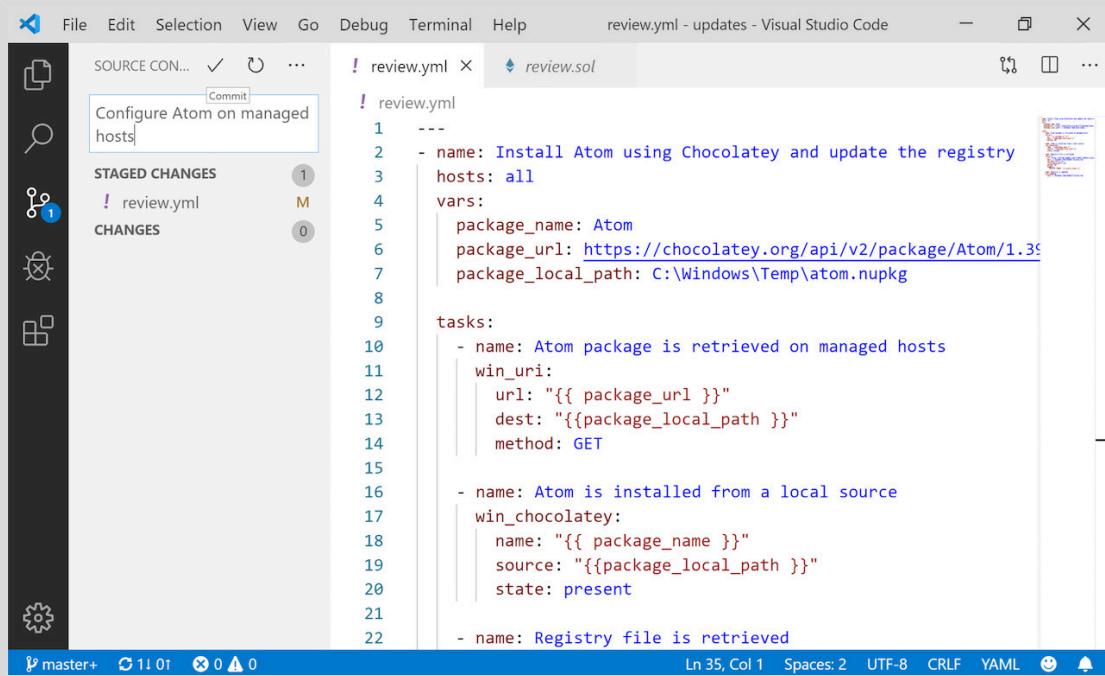
### Note

Le fichier terminé est disponible dans **solutions/review.sol** dans le dossier **updates**.

3. Enregistrez vos modifications, puis validez le fichier avec le message **Configure Atom on managed hosts**.

Poussez vos modifications vers la branche master du référentiel **updates**.

- 3.1. À partir de Visual Studio Code, enregistrez le fichier et indexez-le. Accédez au panneau **Source Control** et cliquez sur **+** en regard de **review.yml** pour indexer vos modifications.
- 3.2. Dans le champ du message de validation, tapez **Configure Atom on managed hosts** et cliquez sur le bouton **Commit**, ou appuyez sur **Ctrl+Entrée** pour valider vos modifications.



The screenshot shows the Visual Studio Code interface. On the left, the Source Control sidebar displays a commit message: "Configure Atom on managed hosts". It also shows two changes: "review.yml" (staged) and "review.sol" (untracked). The main editor area shows the content of the "review.yml" file:

```
! review.yml x ⚡ review.sol

! review.yml
1  ---
2  - name: Install Atom using Chocolatey and update the registry
3    hosts: all
4    vars:
5      package_name: Atom
6      package_url: https://chocolatey.org/api/v2/package/Atom/1.3
7      package_local_path: C:\Windows\Temp\atom.nupkg
8
9  tasks:
10   - name: Atom package is retrieved on managed hosts
11     win_uri:
12       url: "{{ package_url }}"
13       dest: "{{package_local_path }}"
14       method: GET
15
16   - name: Atom is installed from a local source
17     win_chocolatey:
18       name: "{{ package_name }}"
19       source: "{{package_local_path }}"
20       state: present
21
22   - name: Registry file is retrieved
```

- 3.3. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur Synchronize Changes pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
4. Accédez à votre instance GitLab sur <https://gitlab.example.com>, puis connectez-vous à l'aide du compte **student** et du mot de passe **Redhat123@!**. Générez un jeton pour **student** disposant d'un accès à l'API. Nommez le jeton **UpdatesReview**.
  - 4.1. À partir de **workstation**, double-cliquez sur l'icône du Bureau GitLab. Connectez-vous à GitLab en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Assurez-vous que l'onglet **LDAP** est sélectionné.
  - 4.2. Pour générer un jeton API, cliquez sur l'icône dans le coin supérieur droit et cliquez sur **Settings** pour accéder aux paramètres de l'utilisateur.

The screenshot shows the GitLab web interface. At the top, there is a navigation bar with the GitLab logo, 'Projects', 'Groups', 'More', and search/filter icons. On the right side, a user profile for 'student' (@student) is displayed with options for 'Profile', 'Settings' (which is highlighted with a red box), 'Help', and 'Sign out'. Below the profile, a sidebar lists 'Your projects': 'student / updates', 'student / control', 'student / variables', and 'student / test-playbook'. Each project entry includes a small icon, a name, an update status, and a star rating.

- 4.3. Cliquez sur l'entrée **Access Tokens** dans la barre latérale. Créez un jeton en fonction des informations suivantes.

- Nom : **UpdatesReview**
- Étendues : API

Cliquez sur **Create personal access token** pour générer le jeton.

The screenshot shows the 'User Settings' page in GitLab. The left sidebar has sections like 'Profile', 'Account', 'Applications', 'Chat', 'Access Tokens' (which is selected and highlighted with a grey background), 'Emails', 'Notifications', 'SSH Keys', and 'GPG Keys'. The main content area is titled 'User Settings > Access Tokens' and contains a section for 'Personal Access Tokens'. It explains that tokens can be generated for GitLab API or for authenticating over HTTP. It also mentions Two-Factor Authentication (2FA). A form is provided to add a token, with fields for 'Name' (set to 'UpdatesReview'), 'Expires at' (empty), and 'Scopes' (with 'api' checked). A green button at the bottom right says 'Create personal access token'.

Ne fermez pas la page. À l'étape suivante, vous allez utiliser le jeton.

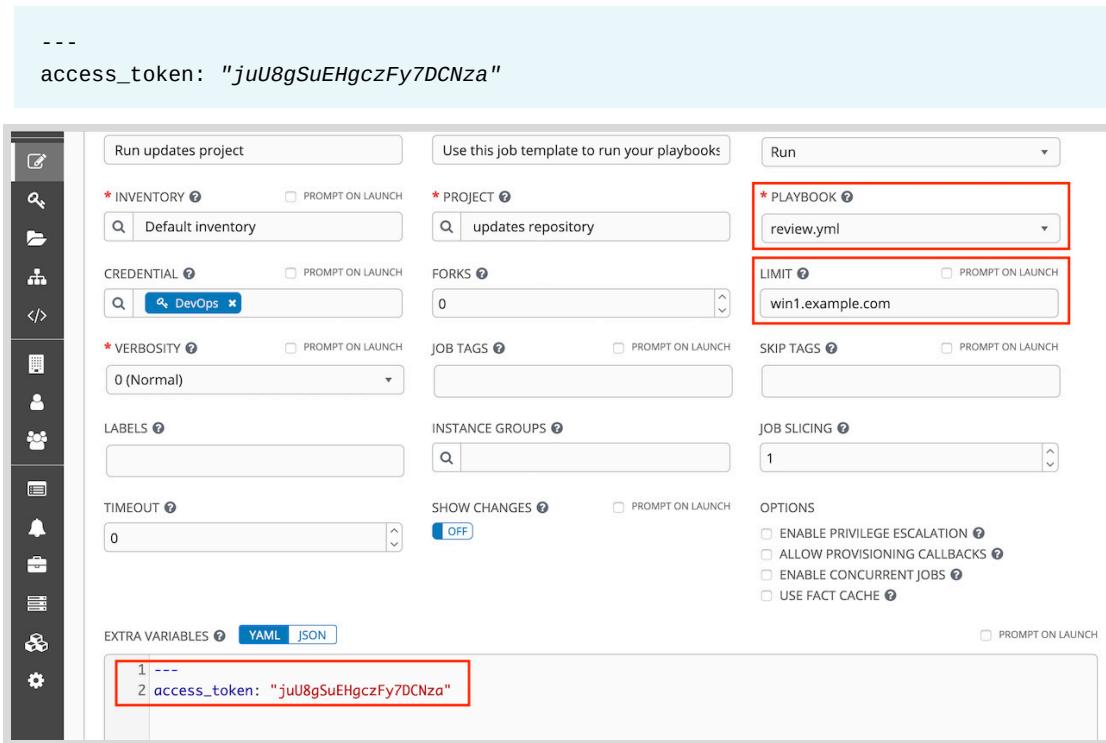
5. À partir de votre poste de travail, accédez à Ansible Tower sur <https://tower.example.com>. Connectez-vous en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Mettez à jour le modèle de tâche **Run updates project** comme suit :

- Sélectionnez **review.yml** comme playbook.
- Limitez le play à **win1.example.com**.
- Transmettez **access\_token** en tant que variable supplémentaire. Attribuez-lui la valeur du jeton GitLab que vous avez générée au cours de l'étape précédente.

- 5.1. À partir de **workstation**, double-cliquez sur l'icône du Bureau Ansible Tower. Connectez-vous à Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe

- 5.2. Cliquez sur **Templates** pour accéder aux modèles de tâches Ansible Tower.  
Sélectionnez le modèle **Run updates project** à modifier.
- 5.3. Mettez à jour le modèle en sélectionnant **review.yml** comme playbook Ansible.  
Dans le champ **LIMIT**, saisissez **win1.example.com**. Enfin, dans le champ **EXTRA VARIABLES**, ajoutez le **access\_token** dont la valeur correspond au jeton GitLab que vous avez créé à l'étape précédente.



- 5.4. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.
6. Exécutez le modèle de tâche **Run updates project** et assurez-vous qu'il s'exécute correctement.
- 6.1. Cliquez sur **LAUNCH** pour lancer une tâche à partir du modèle de tâche.
  - 6.2. Cette action vous redirige vers la sortie de la tâche. Notez la sortie des différentes tâches au fur et à mesure qu'elles s'exécutent. Ansible récupère le paquetage Atom et l'installe, puis récupère le fichier de Registre avant de le fusionner.

**Job Details:**

- STATUS: Successful
- STARTED: 9/19/2019 10:30:28 AM
- FINISHED: 9/19/2019 10:31:02 AM
- JOB TEMPLATE: Run updates project
- JOB TYPE: Run
- LAUNCHED BY: admin
- INVENTORY: Default inventory
- PROJECT: updates repository
- REVISION: 79fe7b3
- PLAYBOOK: review.yml
- CREDENTIAL: DevOps
- LIMIT: win1.example.com
- ENVIRONMENT: /var/lib/awx/venv/ansible
- EXECUTION NODE: localhost
- INSTANCE GROUP: tower
- EXTRA VARIABLES: access\_token: juU8gSuEHgczFy7DCNza

**Playbook Output:**

```

*****
9 changed: [win1.example.com]
10
11 TASK [Atom is installed]
*****
12 ok: [win1.example.com]
13
14 TASK [Registry file is retrieved]
*****
15 changed: [win1.example.com]
16
17 TASK [Registry is updated]
*****
18 changed: [win1.example.com]
19
20 PLAY RECAP
*****
21 win1.example.com : ok=5   changed=3   unreachable=0
                      failed=0  skipped=0   rescued=0  ignored=0

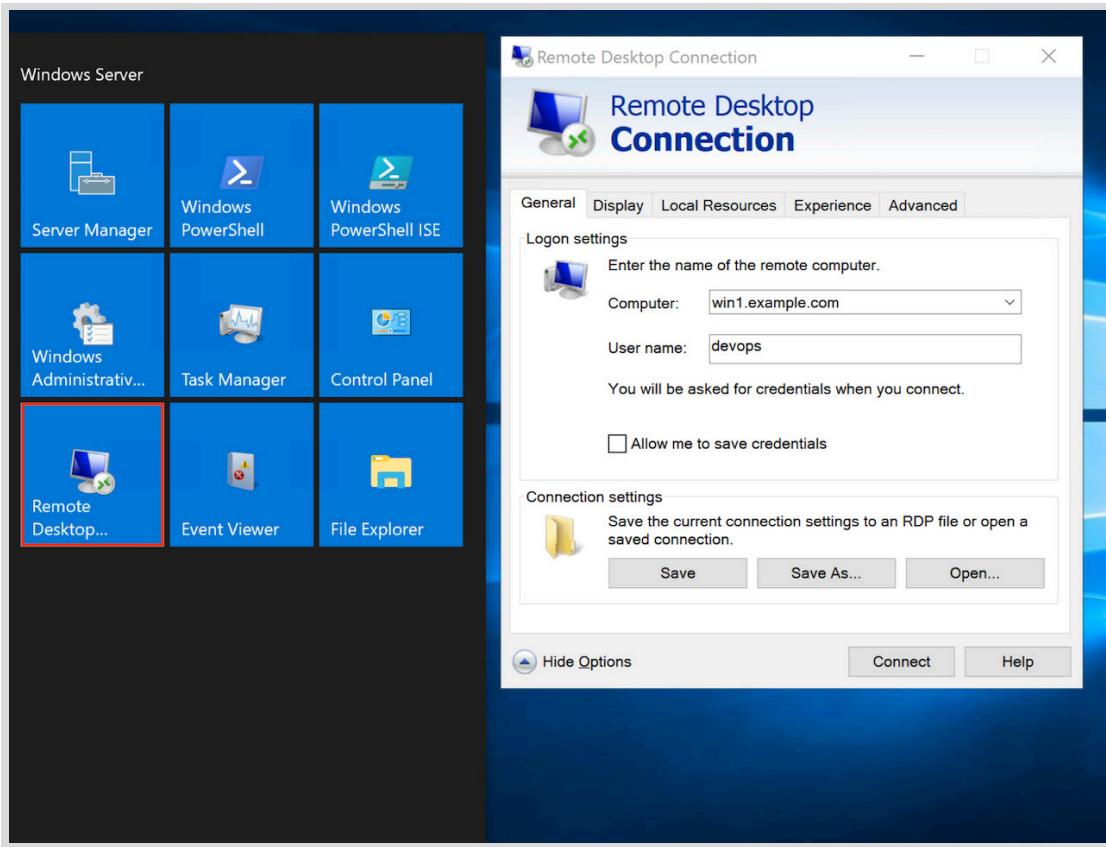
```

7. À partir de **workstation**, ouvrez votre client RDP et connectez-vous à **win1.example.com** en utilisant **devops** comme nom d'utilisateur avec le mot de passe **RedHat123@!**. Lorsque vous y êtes invité, acceptez le certificat non sécurisé.

Vérifiez qu'Atom est installé. Vous pouvez rechercher l'icône du Bureau.

Accédez au registre pour vous assurer que la clé **RedHat** est disponible dans **HKCU:\Software**.

- 7.1. À partir de **workstation**, ouvrez le client du bureau à distance. Sélectionnez **Show options** pour afficher des options supplémentaires. Utilisez le nom d'ordinateur **win1.example.com** et le nom d'utilisateur **devops**. Si vous y êtes invité, acceptez le certificat auto-signé et utilisez **RedHat123@!** comme mot de passe.



- 7.2. Pour vous assurer qu'Atom est installé, localisez l'icône du Bureau. Vous pouvez également exécuter la commande PowerShell **choco list --local-only**.

The screenshot shows a Windows PowerShell window with the title "Windows PowerShell". The command "choco list --local-only" was run, and the output is displayed:  
PS C:\Users\devops> choco list --local-only  
Chocolatey v0.10.15  
atom 1.39.1  
chocolatey 0.10.15  
chocolatey-core.extension 1.3.3  
chocolatey-windowsupdate.extension 1.0.4  
KB2919355 1.0.20160915  
KB2919442 1.0.20160915  
KB2999226 1.0.20181019  
KB3033929 1.0.5  
KB3035131 1.0.3  
putty 0.72  
putty.portable 0.72  
vcredist140 14.22.27821  
wireshark 3.0.4  
13 packages installed.  
PS C:\Users\devops>

**Note**

Si vous n'avez pas terminé les activités précédentes du chapitre, votre sortie peut être différente.

- 7.3. Cliquez sur l'icône de loupe et tapez **Registry Editor** pour accéder au Registre. Lorsque vous y êtes invité, cliquez sur **Yes** pour autoriser l'éditeur à effectuer des modifications.
- 7.4. Accédez à **Computer → HKEY\_CURRENT\_USER → Software**. Assurez-vous qu'une clé **RedHat** est associée à une sous-clé **Training**. Cliquez dessus pour passer en revue ses valeurs.

Computer\HKEY_CURRENT_USER\Software\RedHat\Training			
	Name	Type	Data
» Remote	(Default)	REG_SZ (value not set)	
» Software	Course	REG_SZ DO417	
» AppDataLow	Description	REG_SZ Microsoft Windows Auto...	
» Classes			
» Google			
» Microsoft			
» Policies			
» RedHat			
» Training			
» RegisteredApplications			
» Wow6432Node			
» System			

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Ansible fournit divers modules pour la prise en charge de l'installation de logiciels, y compris **win\_package**, **win\_chocolatey** et **win\_feature**.
- Vous pouvez exécuter la commande PowerShell **Get-WindowsFeature** pour obtenir la liste de tous les noms de fonctions disponibles sur un système.
- Vous pouvez utiliser les modules Ansible **win\_updates**, **win\_hotfix** et **win\_reboot** pour mettre à jour vos systèmes.
- Vous pouvez utiliser les modules Ansible **win\_regedit**, **win\_reg\_stat** et **win\_regmerge** pour gérer le Registre Windows.



## chapitre 6

# Mise en œuvre d'un contrôle de tâche

### Objectif

Gérer l'exécution des tâches à l'aide de boucles, de tests conditionnels et de gestionnaires, et les récupérer en cas d'échec des tâches.

### Résultats

- Rédiger des plays de manière efficace à l'aide de boucles et utiliser des conditions pour contrôler le moment où les tâches sont exécutées.
- Implémenter des tâches qui s'exécutent uniquement lorsqu'une autre tâche apporte une modification à l'hôte géré.
- Contrôler ce qui se passe lorsqu'une tâche échoue, récupérer après l'échec d'une tâche et contrôler les conditions amenant une tâche à signaler un échec.

### Sections

- Écriture de boucles et de tâches conditionnelles (et exercice guidé)
- Mise en œuvre de gestionnaires (et exercice guidé)
- Gestion des échecs de tâche (et exercice guidé)

### Atelier

Mise en œuvre d'un contrôle de tâche

# Écriture de boucles et de tâches conditionnelles

---

## Résultats

Au terme de cette section, vous serez en mesure de rédiger des plays de manière efficace à l'aide de boucles et utiliser des conditions pour contrôler le moment où les tâches sont exécutées.

## Itération des tâches avec des boucles

Répéter plusieurs fois les tâches à l'aide de boucles vous fait gagner du temps lorsque vous rédigez plusieurs tâches qui effectuent des actions utilisant le même module. Par exemple, au lieu d'écrire cinq tâches pour s'assurer que cinq utilisateurs ont été créés, vous pouvez écrire une tâche itérant une liste de cinq utilisateurs pour s'assurer que chacun a été créé à son tour.

Ansible prend en charge l'itération d'une tâche sur un ensemble d'éléments à l'aide du mot-clé **loop**. Vous pouvez configurer des boucles Ansible pour répéter une tâche à l'aide des éléments suivants :

- Les éléments d'une liste.
- Le contenu de chaque fichier dans une liste.
- Une séquence générée de nombres.
- Des structures plus complexes.

Cette section couvre les boucles simples effectuant une itération sur une liste d'éléments. Pour des scénarios de bouclage plus avancés, consultez le lien *Loop – Ansible Documentation* qui figure dans la section Références.

## Présentation des boucles simples

Une boucle simple itère une tâche sur une liste d'éléments. Utilisez le mot-clé **loop** dans une tâche et transmettez la liste des éléments sur lesquels la tâche doit itérer vers la boucle. La boucle **item** contient la valeur actuelle utilisée lors de chaque itération.

Examinez l'extrait de code suivant qui invoque le module **win\_chocolatey** trois fois pour installer un ensemble de paquetages logiciels à l'aide de Chocolatey :

```
- name: Install JRE 8
  win_chocolatey:
    name: jre8
    state: present

- name: Install Java Runtime
  win_chocolatey:
    name: javaruntime-preventasktoolbar
    state: present

- name: Install Dropbox
  win_chocolatey:
    name: dropbox
    state: present
```

Vous pouvez réécrire ces trois tâches à l'aide d'une boucle dans une seule tâche. La boucle itère sur les trois paquetages :

```
- name: Install packages
  win_chocolatey:
    name: "{{ item }}"
    state: present
  loop:
    - jre8
    - javaruntime-preventasktoolbar
    - dropbox
```

Vous pouvez également définir une variable pour la liste et la transmettre à la boucle. Dans l'exemple suivant, la variable **packages** contient la liste des paquetages à installer :

```
vars:
  packages:
    - jre8
    - javaruntime-preventasktoolbar
    - dropbox
tasks:
  - name: Installing Java and Dropbox
    win_chocolatey:
      name: "{{ item }}"
      state: present
    loop: "{{ packages }}"
```

## Itération avec les hachages et les dictionnaires

Il n'est pas nécessaire que les éléments de la liste de la boucle soient des valeurs simples telles que des chaînes ou des entiers. Chaque élément peut être un hachage ou un dictionnaire qui contient deux ou plusieurs clés.

Dans l'exemple suivant, chaque élément de la liste est un dictionnaire, et chaque dictionnaire définit trois clés : **name**, **description** et **groups**. Vous pouvez récupérer chaque clé dans la variable **item** actuelle en utilisant respectivement les variables **item.name**, **item.desc** et **item.grp**.

```
- name: Creating local users
  win_user:
    name: "{{ item.name }}"
    groups: "{{ item.grp }}"
    description: "{{ item.desc }}"
    state: present
  loop:
    - { name: 'itadmin', grp: 'Administrators', desc: 'New admin user' }
    - { name: 'ituser1', grp: 'Users', desc: 'local user' }
    - { name: 'ituser2', grp: 'Users', desc: 'local user' }
```

Vous pouvez réécrire le dictionnaire en utilisant le format YAML comme suit :

```

- name: Creating local users
  win_user:
    name: "{{ item.name }}"
    groups: "{{ item.grp }}"
    description: "{{ item.desc }}"
    state: present
  loop:
    - name: itadmin
      grp: Administrators
      desc: New admin user
    - name: ituser1
      grp: Users
      desc: local user
    - name: ituser2
      grp: Users
      desc: local user

```

La tâche s'assure que l'utilisateur **itadmin** est présent avec une description de **New admin user** et qu'il est membre du groupe **Administrators**. Elle s'assure également que **ituser1** et **ituser2** sont présents et sont membres du groupe **Users**.

## Itération à l'aide de la syntaxe Ansible précédente

Avant Ansible 2.5, la plupart des playbooks utilisaient une syntaxe différente pour les boucles. Plusieurs mots-clés ont été fournis, avec le préfixe **with\_**, suivi du nom d'un plug-in *lookup* Ansible (une fonctionnalité avancée qui n'est pas couverte en détail dans ce cours). Bien que cette syntaxe soit très courante dans les playbooks existants, elle sera peut-être obsolète à l'avenir.

### Exemples de syntaxe de bouclage précédente

Mot-clé	Description
<b>with_items</b>	Se comporte de la même façon que le mot-clé <b>loop</b> pour des listes simples, telles qu'une liste de chaînes ou une liste de dictionnaires. Contrairement aux boucles, si vous utilisez des listes imbriquées avec <b>with_items</b> , Ansible les aplati dans une liste à un seul niveau. La variable de boucle <b>item</b> contient l'élément de liste actuel utilisé lors de chaque itération.
<b>with_file</b>	Ce mot-clé nécessite une liste de fichiers. La variable de boucle <b>item</b> contient le contenu d'un fichier correspondant de la liste de fichiers à chaque itération.
<b>with_sequence</b>	Au lieu de requérir une liste, ce mot-clé nécessite des paramètres pour générer une liste de valeurs basée sur une séquence numérique. La variable de boucle <b>item</b> contient la valeur de l'un des éléments qui sont générés dans la séquence à chaque itération.

L'exemple suivant illustre l'utilisation du mot-clé **with\_items**.

```

vars:
  packages:
    - jre8

```

```
- javaruntime-preventasktoolbar
- dropbox
tasks:
  - name: Installing Java and Dropbox
    win_chocolatey:
      name: "{{ item }}"
      state: present
      with_items: "{{ packages }}"
```



### Important

Depuis Ansible 2.5, la méthode recommandée pour écrire des boucles consiste à utiliser le mot-clé **loop**.

Cependant, vous devez toujours comprendre l'ancienne syntaxe, en particulier **with\_items**, car elle est largement utilisée dans les playbooks existants. Vous êtes susceptible de rencontrer des playbooks et des rôles qui continuent à utiliser des mots-clés **with\_** pour le bouclage.

Vous pouvez convertir n'importe quelle tâche qui utilise la syntaxe précédente pour utiliser le mot-clé **loop** conjointement avec les filtres Ansible. Pour ce faire, vous n'avez pas besoin de savoir comment utiliser les filtres Ansible. Pour plus d'informations sur la conversion de boucles dans la nouvelle syntaxe et sur la façon de parcourir les éléments qui ne sont pas des listes simples, consultez la page *Migrating from with\_X to loop – Ansible Documentation* dans la section Références.



### Note

Les techniques avancées de bouclage dépassent le cadre de ce cours. Vous pouvez implémenter toutes les tâches d'itération de ce cours avec le mot-clé **with\_items** ou **loop**.

## Affichage de la sortie capturée avec des variables de registre et des boucles

Vous pouvez utiliser le mot-clé **register** pour capturer le résultat d'une tâche en boucle. L'exemple suivant illustre la structure de la variable **register** à partir d'une tâche qui contient une boucle.

```
---
- name: Loop register test
  hosts: windows
  tasks:
    - name: Executing commands and registering output in the variable "output"
      win_shell: "{{ item }}"
      loop:
        - whoami
        - hostname
    register: output ①
```

**chapitre 6 |** Mise en œuvre d'un contrôle de tâche

```
- name: showing the results from output variable
  debug:
    var: output ❷
```

- ❶ Enregistre la sortie de chaque itération dans la variable **output**.
- ❷ Affiche le contenu de la variable **output** dans la sortie.

Voici un exemple de sortie du play :

```
PLAY [testing] ****
TASK [Gathering Facts] ****
ok: [winhost1]

TASK [Executing commands and registering output in the variable "output"] **
changed: [winhost1] => (item=whoami)
changed: [winhost1] => (item=hostname)

TASK [showing the results from output variable] ****
ok: [winhost1] => {
  "output": { ❶
    "changed": true,
    "msg": "All items completed",
    "results": [ ❷
      { ❸
        "ansible_loop_var": "item",
        "changed": true,
        "cmd": "whoami",
        "delta": "0:00:00.343753",
        "end": "2019-08-16 04:48:23.980221",
        "failed": false,
        "item": "whoami",
        "rc": 0,
        "start": "2019-08-16 04:48:23.636468",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "ec2amaz-c5f1v2p\\administrator\r\n",
        "stdout_lines": [
          "ec2amaz-c5f1v2p\\administrator"
        ]
      },
      { ❹
        "ansible_loop_var": "item",
        "changed": true,
        "cmd": "hostname",
        "delta": "0:00:00.328126",
        "end": "2019-08-16 04:48:27.089488",
        "failed": false,
        "item": "hostname",
        "rc": 0,
        "start": "2019-08-16 04:48:26.761361",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "EC2AMAZ-C5F1V2P\r\n",
        "stdout_lines": [
          "EC2AMAZ-C5F1V2P"
        ]
      }
    ],
    "vars": {}
  }
}
```

```

        "stdout_lines": [
            "EC2AMAZ-C5F1V2P"
        ]
    ]
}
}

PLAY RECAP *****
winhost1 : ok=3    changed=1    unreachable=0    failed=0    skipped=0
              rescued=0   ignored=0

```

- ➊ La variable **output** est un dictionnaire de trois clés, dont chacune a la valeur : **changed**, **msg** et **results**. (Ansible n'utilise pas de liste ici, car l'ordre de ces clés n'est pas important.)
- ➋ La clé **results** contient les résultats de la tâche. Pour cette boucle, le résultat est une liste de dictionnaires. Les listes sont placées entre crochets « [] », et les dictionnaires sont placés entre accolades « {} ». Chaque élément de la liste est un dictionnaire de paires clé-valeur pour chaque élément de la boucle, dans l'ordre. Les éléments de liste et les paires clé-valeur sont séparés par des virgules.
- ➌ Marque le début des métadonnées de la tâche pour le premier élément. Les métadonnées se composent d'un ensemble de paires clé-valeur, et la clé **item** a pour valeur l'élément dans la liste. La sortie des commandes **whoami** et **hostname** est stockée dans la clé **stdout**.
- ➍ Le début des métadonnées de résultat de tâche pour le deuxième élément.
- ➎ Le caractère de crochet fermant () indique la fin de la liste de sortie.

Dans l'exemple qui précède, la clé **results** contient une liste. Dans l'exemple suivant, le playbook est modifié de sorte que la deuxième tâche itère une liste.

```

- name: Loop Register Test
hosts: winhost1
tasks:

  - name: Executing commands and registering output in the variable "output"
    win_shell: "{{ item }}"
    loop:
      - whoami
      - hostname
    register: output

  - name: showing the results from output variable
    debug:
      msg: "STDOUT from previous task: {{ item.stdout }}"
    loop: "{{ output.results }}"

```

La sortie de ce play peut apparaître comme suit :

```

PLAY [testing] *****
TASK [Gathering Facts] *****
ok: [winhost1]

TASK [Executing commands and registering output in the variable "output"] **
changed: [winhost1] => (item=whoami)
changed: [winhost1] => (item=hostname)

```

```

TASK [showing the results from output variable] *****
ok: [winhost1] => (item=...output omitted...) => {
    "msg": "STDOUT from previous task: ec2amaz-c5f1v2p\\administrator\r\n"
}
ok: [winhost1] => (item={...output omitted...}) => {
    "msg": "STDOUT from previous task: EC2AMAZ-C5F1V2P\r\n"
}

PLAY RECAP *****
winhost1 : ok=3    changed=1    unreachable=0    failed=0
            skipped=0   rescued=0   ignored=0

```

## Exécution de tâches de manière conditionnelle

Ansible peut utiliser des conditions pour exécuter des tâches lorsque certaines conditions sont remplies. Par exemple, vous pouvez définir une condition pour déterminer la quantité de mémoire disponible sur un hôte géré avant qu'Ansible n'installe ou ne configure un service.

Les conditions vous permettent de distinguer les hôtes gérés et de leur attribuer des rôles fonctionnels selon les conditions qu'ils remplissent. Les variables de playbook, les variables enregistrées et les faits Ansible ou les faits personnalisés peuvent tous être testés au moyen de conditions. Vous pouvez utiliser des opérateurs tels que supérieur à (`>`) ou inférieur à (`<`) pour comparer des chaînes, des données numériques ou des valeurs booléennes.

Les scénarios suivants illustrent l'utilisation de conditions dans Ansible :

- Définissez une limite fixe dans une variable (par exemple, `min_memory` pour la quantité minimale de mémoire qui est requise) et comparez-la à la mémoire disponible sur un hôte géré.
- Capturez la sortie d'une commande et évaluez-la pour déterminer s'il faut prendre d'autres mesures. Par exemple, vous pouvez ordonner à Ansible de copier un fichier `index.html` uniquement si le serveur Web IIS est installé et configuré correctement.
- Utilisez les faits Ansible pour déterminer la configuration réseau d'un hôte géré et décider quel fichier de modèle envoyer.
- Comparez une variable enregistrée avec une variable prédéfinie pour déterminer si un service a changé. Par exemple, testez la somme de contrôle MD5 d'un fichier de configuration de service pour voir si le service a changé.

## Écriture de tâches conditionnelles

Utilisez l'instruction `when` pour exécuter une tâche de manière conditionnelle. Il prend comme valeur la condition à tester. Si la condition que vous définissez est satisfaite, la tâche est exécutée, sinon, Ansible ignore la tâche.

Une condition simple que vous pouvez tester est si une variable booléenne a la valeur `true` ou `false`. L'instruction `when` de l'exemple suivant entraîne l'exécution de la première tâche uniquement si `run_my_task` a la valeur `true`. Dans la deuxième tâche, le système ne redémarre que si la `win_feature.reboot_required` variable est égale à `true`.

```

- name: Simple Boolean Task Demo
  hosts: winhost1
  vars:
    run_my_task: true

```

```

tasks:
  - name: Install IIS Web-Server with sub features and management tools
    win_feature:
      name: Web-Server
      state: present
      include_sub_features: yes
      include_management_tools: yes
    when: run_my_task
    register: feature_output

  - name: Reboot if installing Web-Server feature requires it
    win_reboot:
    when: feature_output.reboot_required

```

L'exemple suivant est un peu plus compliqué, car il teste si la variable **my\_service** contient une valeur. Si c'est le cas, Ansible utilise la valeur de la variable **my\_service** comme paquetage à installer. Si la variable **my\_service** n'est pas définie, Ansible ignore la tâche.

```

- name: Checking if the variable is defined
  hosts: winhost1
  vars:
    my_service: Web-Server
  tasks:
    - name: Install "{{ my_service }}" with sub features and management tools if
      my_service is defined
        win_feature:
          name: "{{ my_service }}"
          state: present
          include_sub_features: yes
          include_management_tools: yes
        when: my_service is defined

```

## Création de conditions à l'aide d'opérateurs Ansible

Le tableau suivant décrit quelques-unes des opérations que vous pouvez appliquer lorsque vous utilisez des conditions :

### Exemples d'utilisation d'opérateurs Ansible

Opération	Exemple
Est égal à (avec une chaîne)	<code>ansible_facts['architecture'] == "64-bit"</code>
Est égal à (avec une valeur numérique)	<code>max_memory == 1024</code>
Est supérieur à	<code>ansible_facts['powershell_version'] &gt; 2</code>
Est inférieur à	<code>ansible_facts['powershell_version'] &lt; 6</code>
Est supérieur ou égal à	<code>ansible_facts['processor_vcpus'] &gt;= 2</code>
Est inférieur ou égal à	<code>ansible_facts['processor_vcpus'] &lt;= 20</code>

Opération	Exemple
N'est pas égal à	<code>ansible_facts['memtotal_mb'] != 4000</code>
La variable existe	<code>my_service is defined</code>
La variable n'existe pas	<code>my_service is not defined</code>
Une variable booléenne en soi est implicitement un test pour savoir si elle est vraie. Les valeurs de 1, <b>True</b> ou <b>yes</b> sont évaluées comme étant vraies. Les valeurs de 0, <b>False</b> ou <b>no</b> sont évaluées comme fausses.	<code>reboot_required</code>
La variable booléenne est fausse	<code>not reboot_required</code>
La valeur de la première variable est présente en tant que valeur de la liste de la seconde variable.	<code>ansible_facts['distribution'] in supported_distros</code>

La dernière entrée du tableau précédent peut être troublante au premier abord. L'exemple suivant illustre son fonctionnement.

Dans l'exemple suivant, la variable `ansible_facts['distribution']` est un fait qui est déterminé au cours de la phase de collecte des faits, et qui identifie la distribution du système d'exploitation de l'hôte géré (pour Linux) ou la variante (pour Microsoft Windows). La variable `supported_distros` contient une liste des distributions de système d'exploitation prises en charge par le playbook. Si la valeur de `ansible_facts['distribution']` figure dans la liste `supported_distros`, la condition passe et la tâche est exécutée.

```
- name: Testing a condition
hosts: winhost1
vars:
  my_service: Web-Server
  supported_distros:
    - "Microsoft Windows Server 2016 Datacenter"
    - "Microsoft Windows Server 2016 Core"
    - "Microsoft Windows Server 2012 Datacenter"
    - "Microsoft Windows Server 2012 Core"
tasks:
  - name: Install "{{ my_service }}" with sub features and management tools
    win_feature:
      name: "{{ my_service }}"
      state: present
      include_sub_features: yes
      include_management_tools: yes
    when: ansible_facts['distribution'] in supported_distros
```



### Important

Notez la mise en retrait de l'instruction **when**. L'instruction n'étant pas une variable de module, vous devez la placer hors du module en la mettant en retrait au niveau supérieur de la tâche.

Une tâche est un hachage ou un dictionnaire YAML, et l'instruction **when** n'est qu'une clé supplémentaire dans la tâche similaire au nom de celle-ci et au module qu'elle utilise. Une convention d'usage consiste à placer le mot-clé **when** après le nom de la tâche, et le module et les arguments du module.

## Conditions multiples

Vous pouvez utiliser une instruction **when** pour évaluer plusieurs conditions. Pour ce faire, combinez des instructions conditionnelles à l'aide des opérateurs **and** ou **or**. Vous pouvez également regrouper des conditions avec des parenthèses lorsqu'il y a plus de deux opérateurs.

Les exemples suivants illustrent la façon d'exprimer plusieurs conditions.

- Si, pour que soit rempli un argument conditionnel, il suffit qu'une seule condition soit vraie, vous devez utiliser l'opérateur **or**. Par exemple, la condition suivante est remplie si la machine exécute soit Microsoft Windows Server 2016 Datacenter, soit Microsoft Windows Server 2012 Datacenter Edition.

```
when: ansible_facts['distribution'] == "Microsoft Windows Server 2016 Datacenter"
      or ansible_facts['distribution'] == "Microsoft Windows Server 2012 Datacenter"
```

- Avec l'opérateur **and**, les deux conditions doivent être vraies pour que l'ensemble de l'instruction conditionnelle soit vrai. Par exemple, la condition suivante est remplie si l'hôte distant est un hôte Microsoft Windows Server 2016 Datacenter avec une architecture 64 bits :

```
when: ansible_facts['distribution'] == "Microsoft Windows Server 2016 Datacenter"
      and ansible_facts['architecture'] == "64-bit"
```

Le mot-clé **when** prend également une liste permettant de décrire une liste de conditions. Lorsque vous fournissez une liste au mot-clé, Ansible combine toutes les conditions en utilisant l'opérateur **and**. L'exemple suivant illustre une autre façon de combiner plusieurs instructions conditionnelles à l'aide de l'opérateur **and**.

```
when:
  - ansible_facts['distribution'] == "Microsoft Windows Server 2016 Datacenter"
    - ansible_facts['architecture'] == "64-bit"
```

Ce format améliore la lisibilité, un objectif clé de playbooks Ansible bien écrits.

- Vous pouvez exprimer des instructions conditionnelles plus complexes en regroupant des conditions avec des parenthèses. Cela garantit qu'Ansible les interprète correctement.

Par exemple, l'instruction conditionnelle suivante est remplie si la machine exécute soit Microsoft Windows Server 2016 Datacenter, soit Microsoft Windows Server 2012 Datacenter Edition et dispose d'une architecture 64 bits.

```
when: >
  ( ansible_facts['distribution'] == "Microsoft Windows Server 2016 Datacenter"
  or
  ansible_facts['distribution'] == "Microsoft Windows Server 2012 Datacenter" )
and
  ansible_facts['architecture'] == "64-bit"
```

## Combinaison de boucles et de tâches conditionnelles

Vous pouvez combiner des boucles et des conditions pour gérer des déploiements plus granulaires. Dans l'exemple suivant, Ansible déploie le serveur Web IIS à l'aide du module **win\_feature** si le système d'exploitation est Microsoft Windows Server 2016 Core avec une architecture 64 bits. La tâche s'exécute uniquement si les deux conditions sont vraies.

```
- name: Testing Complex condition with loop
hosts: winhost1
vars:
  my_package:
    - jre8
    - javaruntime-preventasktoolbar
    - dropbox
tasks:
  - name: Install packages
    win_chocolatey:
      name: "{{ item }}"
      state: present
    loop: "{{ my_package }}"
    when:
      - ansible_facts['distribution'] == "Microsoft Windows Server 2016 Core"
      - ansible_facts['architecture'] == "64-bit"
```



### Important

Lorsque vous utilisez l'instruction **when** dans une boucle, Ansible évalue la condition pour chaque élément de la boucle.



### Références

#### Boucles — Documentation Ansible

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_loops.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html)

#### Migrating from with\_X to loop — Ansible Documentation

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_loops.html#migrating-from-with-x-to-loop](https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html#migrating-from-with-x-to-loop)

## ► Exercice guidé

# Écriture de boucles et de tâches conditionnelles

Dans cet exercice, vous allez écrire un playbook à l'aide de tâches comportant des boucles et des tests conditionnels.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Utiliser des boucles pour effectuer une itération sur les éléments et pour réduire le code répétitif.
- Utiliser des conditions logiques pour contrôler le moment où les tâches doivent s'exécuter.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.

- ▶ 1. Ouvrez l'éditeur Visual Studio Code et clonez le référentiel **control** sur **workstation**.
  - 1.1. Double-cliquez sur l'icône Visual Studio Code sur le Bureau.
  - 1.2. Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.
  - 1.3. À l'invite, fournissez l'URL de référentiel <https://gitlab.example.com/student/control.git> et choisissez le dossier **Documents** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **C:\Users\student\Documents\control** sur **workstation**.
  - 1.4. Dans le coin inférieur droit, cliquez sur **Open** pour ouvrir le projet.
- ▶ 2. Ouvrez le playbook **loops.yml** pour passer en revue son contenu. Notez la longue liste de tâches répétitives. Chaque tâche utilise le module **win\_feature** pour s'assurer qu'une fonction de serveur Web IIS est présente. Au cours des étapes suivantes, vous allez refactoriser le playbook pour améliorer la lisibilité et la facilité de maintenance.
- ▶ 3. Regroupez toutes les fonctions de débogage supplémentaires dans une seule tâche à l'aide d'une boucle. Remplacez toutes les tâches à l'exception de la tâche appelée **IIS is installed** avec la tâche mise à jour comme suit :

```
---
- name: IIS and debugging tools installed
  hosts: win1.example.com
  tasks:
    - name: IIS is installed
      win_feature:
        name: Web-Server
        state: present

    - name: Debug tools are installed
      win_feature:
        name: "{{ item }}"
        state: present
    loop:
      - Web-Custom-Logging
      - Web-Log-Libraries
      - Web-Request-Monitor
      - Web-Http-Tracing
      - Web-Mgmt-Tools

    - name: Server is rebooted
      win_reboot:
```

- ❶** Chaque itération de la boucle définit la valeur de la variable `item` sur l'élément suivant dans la liste `loop`.
- ❷** Consolidation des fonctions à installer dans une liste simple.

▶ **4.** Modifiez le playbook pour installer uniquement les outils de débogage supplémentaires lorsque l'espace disponible sur le disque est suffisant.

- 4.1. Insérez une tâche directement avant la tâche `Debug tools are installed` qui collecte des faits sur les disques présents sur l'hôte géré.

```
- name: Disk facts are gathered
  win_disk_facts:
```

- 4.2. Ajoutez les paramètres `when` et `vars` à la tâche `Debug tools are installed` comme suit. Mettez en retrait avec six espaces pour aligner les paramètres `when` et `vars` sur le paramètre `loop`.

```
when: size_remaining >= minimum_bytes ❶
vars: ❷
  minimum_bytes: "{{ '10 GB' | human_to_bytes }}"
  first_volume: "{{ ansible_facts.disks[0].partitions[0].volumes[0] }}" ❸
  size_remaining: "{{ first_volume.size_remaining }}
```

- ❶** Ansible évalue la clause `when` sur chaque itération de la boucle. Si la variable `size_remaining` est supérieure ou égale à la variable `minimum_bytes`, Ansible appelle le module `win_feature` pour s'assurer que la fonctionnalité est présente.
- ❷** Déclarez les variables de la tâche afin d'améliorer la lisibilité de la logique conditionnelle.

- ③ Par souci de simplicité, sélectionnez l'espace disponible à partir du premier volume de la première partition du premier disque sur notre hôte géré par la salle de classe.

► 5. Vérifiez que le **loops.yml** final correspond au playbook suivant.

```
---
- name: IIS and debugging tools installed
  hosts: win1.example.com
  tasks:
    - name: IIS is installed
      win_feature:
        name: Web-Server
        state: present

    - name: Disk facts are gathered
      win_disk_facts:

    - name: Debug tools are installed
      win_feature:
        name: "{{ item }}"
        state: present
      loop:
        - Web-Custom-Logging
        - Web-Log-Libraries
        - Web-Request-Monitor
        - Web-Http-Tracing
        - Web-Mgmt-Tools
    when: size_remaining >= minimum_bytes
  vars:
    minimum_bytes: "{{ '10 GB' | human_to_bytes }}"
    first_volume: "{{ ansible_facts.disks[0].partitions[0].volumes[0] }}"
    size_remaining: "{{ first_volume.size_remaining }}"

- name: Server is rebooted
  win_reboot:
```

- 6. Enregistrez et validez les modifications sur votre référentiel Git local, puis transmettez-les par push au référentiel distant.
- 6.1. Cliquez sur **File → Save** ou tapez **Ctrl+S** pour enregistrer le fichier.
  - 6.2. Accédez au volet **Source Control**, puis cliquez sur le **+** en regard du fichier **loops.yml** pour indexer les modifications.
  - 6.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 6.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.

**chapitre 6 |** Mise en œuvre d'un contrôle de tâche

- 7. À partir de **workstation**, accédez à votre instance Ansible Tower sur `http://tower.example.com`. Connectez-vous en tant que **student** avec **RedHat123@!** comme mot de passe.
- 8. Lancez une tâche à partir du modèle de tâche **Run control project**.
- 8.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 8.2. Cliquez sur le modèle de tâche **Run control project** pour le modifier.
  - 8.3. Sélectionnez le playbook **loops.yml** dans la liste **PLAYBOOK**.

The screenshot shows the 'Run control project' template configuration page. The 'DETAILS' tab is selected. The configuration includes:

- NAME:** Run control project
- DESCRIPTION:** Use this job template to run your playbooks
- JOB TYPE:** Run
- INVENTORY:** Default inventory
- PROJECT:** control repository
- PLAYBOOK:** loops.yml
- CREDENTIAL:** DevOps
- FORKS:** 0
- LIMIT:** (empty)
- VERBOSITY:** 0 (Normal)
- JOB TAGS:** (empty)
- SKIP TAGS:** (empty)
- LABELS:** (empty)
- INSTANCE GROUPS:** (empty)
- JOB SLICING:** 1

- 8.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
- 9. Vérifiez que le playbook a été exécuté correctement et examinez la sortie du journal.
- 9.1. Une fois toutes les tâches terminées, vérifiez que le statut de la section **DETAILS** affiche **Successful**. Si la tâche ne s'est pas correctement déroulée, comparez votre playbook avec le fichier de solution **solutions/loops.sol** dans le référentiel Git **control**.
  - 9.2. Examinez la sortie du journal et notez que la tâche **Debug tools are installed** s'exécute avec plusieurs modifications.

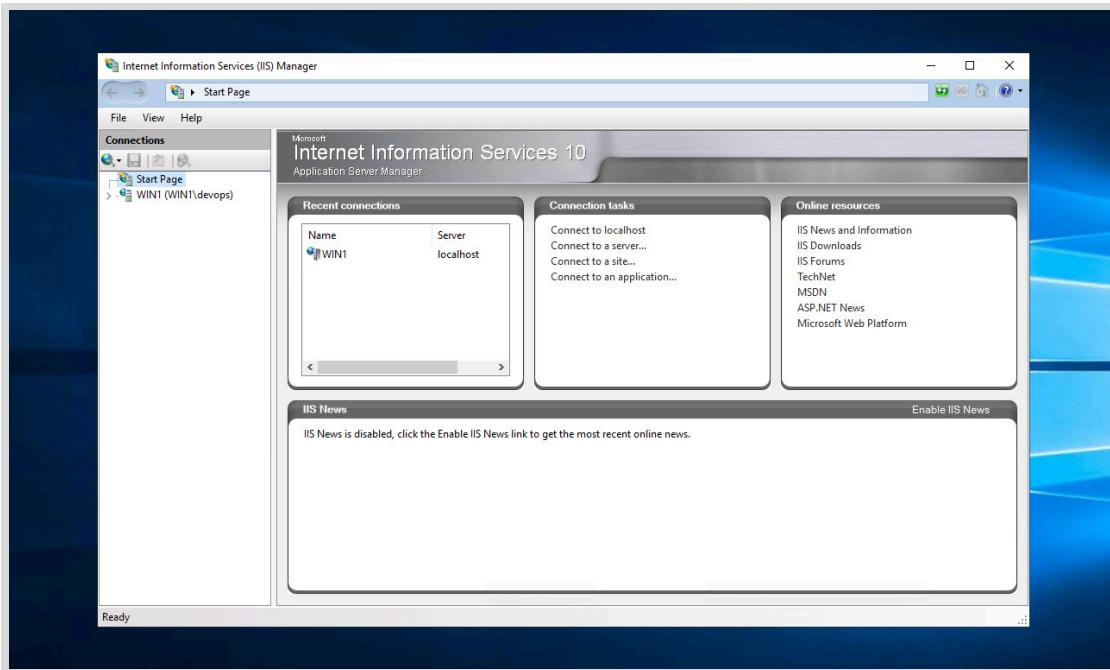
```

5 TASK [Gathering Facts] *****
6 ok: [win1.example.com]
7
8 TASK [IIS is installed] *****
9 ok: [win1.example.com]
10
11 TASK [Disk facts are gathered] *****
12 ok: [win1.example.com]
13
14 TASK [Debug tools are installed] *****
15 ok: [win1.example.com] => (item=Web-Custom-Logging)
16 ok: [win1.example.com] => (item=Web-Log-Libraries)
17 ok: [win1.example.com] => (item=Web-Request-Monitor)
18 ok: [win1.example.com] => (item=Web-Http-Tracing)
19 ok: [win1.example.com] => (item=Web-Mgmt-Tools)
20
21 PLAY RECAP *****
22 win1.example.com : ok=4    changed=0    unreachable=0    failed=0    skipped=0
                      rescued=0   ignored=0
23

```

- ▶ 10. Connectez-vous à l'hôte géré **win1.example.com** pour vérifier la présence des outils de gestion Web IIS.

- 10.1. Cliquez sur **Search Windows**, puis tapez **remote** pour rechercher et ouvrir Connexion Bureau à distance.
- 10.2. Saisissez **win1.example.com**, **devops** et **RedHat123@!** pour **Computer**, le nom d'utilisateur et le mot de passe, respectivement. Notez que **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.  
Cliquez sur **Yes** pour accepter le certificat non sécurisé utilisé dans votre environnement de formation.
- 10.3. Dans la session à distance **win1.example.com**, cliquez sur **Search Windows**, puis tapez **IIS** pour rechercher et ouvrir le gestionnaire IIS (Internet Information Services).



- 11. Exécutez le playbook **cleanup-loops.yml** à partir du modèle de tâche **Run control project**.
- 11.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 11.2. Cliquez sur le nom du modèle de tâche **Run control project**.
  - 11.3. Sélectionnez le playbook **cleanup-loops.yml** dans la liste **PLAYBOOK**.
  - 11.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'exercice guidé est maintenant terminé.

# Mise en œuvre des gestionnaires

---

## Résultats

Une fois cette section terminée, vous serez en mesure d'implémenter des gestionnaires qui ne s'exécutent que lorsqu'une autre tâche apporte une modification à l'hôte géré et l'en avertit.

## Présentation des gestionnaires

Les modules Ansible sont conçus pour être idempotents. Cela signifie que vous pouvez exécuter le playbook et les tâches correspondantes plusieurs fois sans modifier aucun composant sur l'hôte géré, à moins que vous n'indiquez à Ansible d'effectuer une modification pour amener l'hôte géré dans l'état que vous voulez.

Toutefois, à certains moments lorsqu'une tâche modifie le système, l'exécution, vous devrez peut-être exécuter des tâches supplémentaires. Par exemple, si vous installez une nouvelle fonctionnalité ou un paquetage sur votre serveur ou si vous associez un serveur à un domaine, vous devrez peut-être redémarrer le serveur pour que l'installation puisse être terminée ou que les modifications puissent être appliquées avec succès.

Les *gestionnaires* sont des tâches qui répondent à une notification déclenchée par d'autres tâches. Les tâches notifient leurs gestionnaires uniquement lorsque la tâche modifie quelque chose sur un hôte géré. Chaque gestionnaire possède un nom globalement unique et se déclenche à la fin d'un bloc de tâches dans un playbook. Si aucune tâche ne notifie le gestionnaire par son nom, le gestionnaire ne s'exécute pas. Si une ou plusieurs tâches notifient le gestionnaire, celui-ci s'exécute précisément une fois lorsque toutes les autres tâches sont terminées. Les gestionnaires étant des tâches, vous pouvez utiliser les mêmes modules dans les gestionnaires que dans les autres tâches. En général, vous utilisez des gestionnaires pour redémarrer des hôtes ou des services.

Vous pouvez considérer les gestionnaires comme des tâches inactives qui se déclenchent lorsqu'elles sont invoquées par une tâche modifiée qui a une instruction **notify**. L'exemple suivant montre comment le gestionnaire de **serveur de réinitialisation** s'exécute lorsqu'Ansible installe un nouveau paquetage dans une tâche.

```
- name: Testing Handlers
hosts: winhost1
tasks:
  - name: Install Dotnet4.6.1 ①
    win_chocolatey:
      name: dotnet4.6.1
      state: present
      notify: ②
        - reboot server ③

handlers:
  ④
    - name: reboot server ⑤
      win_reboot: ⑥
```

① Tâche de notification du gestionnaire.

- ❷ L'argument **notify** indique que la tâche a besoin de déclencher un gestionnaire.
- ❸ Nom du gestionnaire à exécuter.
- ❹ Le mot-clé **handlers** indique le début de la liste des tâches du gestionnaire.
- ❺ Nom du gestionnaire invoqué que la tâche invoque.
- ❻ Module à utiliser pour le gestionnaire.

Dans l'exemple précédent, le gestionnaire de **serveur de réinitialisation** se déclenche lorsque la tâche **win\_chocolatey** notify à Ansible qu'une modification s'est produite. Une tâche peut invoquer plusieurs gestionnaires dans la section **notify**. Ansible traite l'instruction **notify** comme une matrice et répète les noms des gestionnaires.

L'exemple suivant montre comment la tâche appelle deux gestionnaires ; le gestionnaire initial affiche un message, tandis que le deuxième gestionnaire redémarre le serveur.

```
- name: Testing complex condition with loop
hosts: winhost1
tasks:
  - name: Install Dotnet4.6.1
    win_chocolatey:
      name: dotnet4.6.1
      state: present
    notify:
      - printing message
      - reboot server

  handlers:
    - name: printing message
      debug:
        msg: Installation is done, rebooting the server now
    - name: reboot server
      win_reboot:
```

## Description des fonctions des gestionnaires

La liste suivante décrit certaines informations importantes à connaître sur les gestionnaires Ansible :

- Les gestionnaires fonctionnent toujours dans l'ordre spécifié par la section **handlers** du play. Ils ne s'exécutent pas dans l'ordre dans lequel ils sont listés par les instructions **notify** dans une tâche, ou dans l'ordre dans lequel les tâches les notifient.
- Dans un play simple avec une section unique, les gestionnaires sont exécutés lorsque toutes les autres tâches sont terminées. Un gestionnaire appelé par une tâche dans la partie **tasks** du playbook ne s'exécute que lorsqu'Ansible exécute toutes les tâches. Les plays plus complexes qui utilisent des sections, ou des rôles, **pre\_tasks** et **post\_tasks**, déclencheront des gestionnaires après chaque section.
- Les noms des gestionnaires existent dans un espace de noms par play. Si deux gestionnaires portent le même nom, Ansible n'exécute que le dernier.
- Même si plusieurs tâches notifient un gestionnaire, ce dernier n'est exécuté qu'une seule fois. Si aucune tâche ne le notifie, un gestionnaire ne s'exécute pas.
- Si une tâche qui inclut une instruction **notify** ne signale pas un résultat modifié (par exemple, si un paquetage est déjà installé et que la tâche indique qu'aucune autre action n'est requise),

le gestionnaire n'est pas notifié. Ansible ignore le gestionnaire à moins qu'il soit notifié par une autre tâche. Ansible notifie les gestionnaires uniquement si la tâche signale l'état **changed**.



### Important

Les gestionnaires sont censés effectuer une action supplémentaire lorsqu'une tâche modifie un hôte géré. Vous ne devez pas les utiliser en remplacement des tâches habituelles.

## Ordre d'exécution des gestionnaires

Ansible exécute les sections de play dans l'ordre suivant :

1. Les tâches que vous définissez dans la section **pre\_tasks**.
2. Les gestionnaires déclenchés par les tâches de la section **pre\_tasks**.
3. Les rôles que vous définissez dans la section **include\_role**.
4. Les tâches que vous définissez dans la section **tasks**.
5. Les gestionnaires que les tâches des sections **include\_roles** et **tasks** notifient.
6. Les tâches que vous définissez dans la section **post\_tasks**.
7. Les gestionnaires déclenchés par les tâches de la section **post\_tasks**.

L'ordre de ces sections dans un play ne modifie pas l'ordre d'exécution, comme indiqué ci-dessus. Par exemple, si vous écrivez la section **tasks** avant la section **roles**, Ansible exécute toujours les rôles avant les tâches de la section **tasks**. Toutefois, pour des raisons de lisibilité, il est recommandé d'organiser votre play dans l'ordre d'exécution suivant :

1. **pre\_tasks**
2. **roles**
3. **tâches**
4. **post\_tasks**
5. **handlers**

Ansible exécute et vide les gestionnaires que les tâches notifient à plusieurs stades au cours d'une exécution : après la section **pre\_tasks**, après les sections **include\_roles** et **tasks**, et après la section **post\_tasks**. Cela signifie qu'un gestionnaire peut s'exécuter plusieurs fois, à différents moments au cours de l'exécution du play, si les tâches le notifient dans plusieurs sections.



### Références

**Gestionnaires : Exécution d'opérations sur les modifications – Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_intro.html#handlers-running-operations-on-change](https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#handlers-running-operations-on-change)

## ► Exercice guidé

# Mise en œuvre des gestionnaires

Dans cet exercice, vous allez écrire un play qui utilise un gestionnaire pour exécuter une tâche si une autre tâche apporte une modification à l'hôte géré.

## Résultats

Vous serez en mesure d'implémenter des tâches qui s'exécutent uniquement lorsqu'une autre tâche apporte une modification à l'hôte géré.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. Ouvrez l'éditeur Visual Studio Code et clonez le référentiel **control** sur votre **workstation**.
  - 1.1. Sur **workstation**, ouvrez l'éditeur Visual Studio Code en double-cliquant sur son icône sur le Bureau.
  - 1.2. Ouvrez la palette de commandes en accédant à **View → Command palette** ou en tapant **Ctrl+Maj+P**.  
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.
  - 1.3. À l'invite, fournissez l'URL de référentiel <https://gitlab.example.com/student/control.git> et choisissez le dossier **Documents** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **C:\Users\student\Documents\control** sur **workstation**.
  - 1.4. Dans le coin inférieur droit, cliquez sur **Open** pour ouvrir le projet.
- ▶ 2. Créez un playbook nommé **handlers.yml** qui utilise le module Ansible **win\_updates** pour installer les mises à jour de sécurité Windows sur le serveur **win1.example.com**.

```
---
- name: System updated and restarted
  hosts: win1.example.com

  tasks:
    - name: Security updates applied
      win_updates:
        category_names: SecurityUpdates
        state: installed
```

- 3. Sous la section **tasks**, ajoutez un gestionnaire qui utilise le module **win\_reboot** auquel vous pouvez notifier le redémarrage du serveur. Bien que ce module fournit un paramètre **reboot**, cet exercice crée un gestionnaire permettant de gérer cette action, ce qui vous permet d'exécuter le redémarrage en fonction de votre cas d'utilisation.

```
---  
- name: System updated and restarted  
  hosts: win1.example.com  
  
  tasks:  
    - name: Security updates applied  
      win_updates:  
        category_names: SecurityUpdates  
        state: installed  
  
    handlers:  
      - name: Server is rebooted  
        win_reboot:
```

- 4. Ajoutez une instruction **notify** à votre tâche pour invoquer le gestionnaire lorsque la tâche modifie l'état de l'hôte cible.

```
---  
- name: System updated and restarted  
  hosts: win1.example.com  
  
  tasks:  
    - name: Security updates applied  
      win_updates:  
        category_names: SecurityUpdates  
        state: installed  
      notify:  
        - Server is rebooted  
  
    handlers:  
      - name: Server is rebooted  
        win_reboot:
```

- 5. Enregistrez et validez les modifications sur votre référentiel Git local, puis transmettez-les par push au référentiel distant.

- 5.1. Cliquez sur **File → Save** ou tapez **Ctrl+S** pour enregistrer le fichier.
- 5.2. Accédez au volet **Source Control**, puis cliquez sur le **+** en regard de **handlers.yml** pour indexer les modifications.
- 5.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 5.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.

**chapitre 6 |** Mise en œuvre d'un contrôle de tâche

- 6. À partir de **workstation**, accédez à votre instance Ansible Tower sur `http://tower.example.com`. Connectez-vous en tant que **student** avec **RedHat123@!** comme mot de passe.
- 7. Dans le volet de navigation, cliquez sur **Projects**. Cliquez sur le bouton de synchronisation pour récupérer les dernières modifications apportées au projet **control**.
- 8. Lancez une tâche à partir du modèle de tâche **Run control project**.
- 8.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 8.2. Cliquez sur le modèle de tâche **Run control project** pour le modifier.
  - 8.3. Sélectionnez le playbook **handlers.yml** dans la liste **PLAYBOOK**.

The screenshot shows the 'Run control project' configuration page. The 'DETAILS' tab is selected. The configuration includes:

- NAME:** Run control project
- DESCRIPTION:** Use this job template to run your playbooks
- JOB TYPE:** Run
- INVENTORY:** Default inventory
- PROJECT:** control repository
- PLAYBOOK:** handlers.yml
- CREDENTIAL:** DevOps
- FORKS:** 0
- LIMIT:** (empty)
- VERBOSITY:** 0 (Normal)
- JOB TAGS:** (empty)
- SKIP TAGS:** (empty)
- LABELS:** (empty)
- INSTANCE GROUPS:** (empty)
- JOB SLICING:** 1

- 8.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
- 9. Vérifiez que le playbook a été exécuté correctement et examinez la sortie du journal.
- 9.1. Une fois toutes les tâches terminées, vérifiez que le statut de la section **DETAILS** affiche **Successful**. Si la tâche ne s'est pas correctement déroulée, comparez votre playbook avec le fichier de solution **solutions/handlers.sol** dans le référentiel Git **control**.
  - 9.2. Examinez la sortie du journal et notez que la tâche **Security updates applied** s'exécute, signale l'état **CHANGED** et notifie le gestionnaire **Server is rebooted**.

```

-
1 SSH password:
2
3 PLAY [Apply system updates] **** 20:57:39
4
5 TASK [Gathering Facts] **** 20:57:39
6 ok: [win1.example.com]
7
8 TASK [Security updates applied] **** 20:57:47
9 changed: [win1.example.com]
10
11 RUNNING HANDLER [Server is rebooted] **** 21:04:21
12 changed: [win1.example.com]
13
14 PLAY RECAP **** 21:05:35
15 win1.example.com : ok=3    changed=2    unreachable=0    failed=0    skip
ped=0    rescued=0    ignored=0
16

```

- 10. Exécutez le même playbook une deuxième fois. Comparez la sortie du journal et notez que la tâche **Security updates applied** s'exécute, signale l'état **OK**, mais ne notify pas le gestionnaire **Server is rebooted** parce qu'aucune modification n'a été effectuée.

```

-
1 SSH password:
2
3 PLAY [Apply system updates] **** 21:08:11
4
5 TASK [Gathering Facts] **** 21:08:11
6 ok: [win1.example.com]
7
8 TASK [Security updates applied] **** 21:08:39
9 ok: [win1.example.com]
10
11 PLAY RECAP **** 21:08:58
12 win1.example.com : ok=2    changed=0    unreachable=0    failed=0    skip
ped=0    rescued=0    ignored=0
13

```

L'exercice guidé est maintenant terminé.

# Gestion des échecs de tâche

---

## Résultats

Après avoir terminé cette section, vous serez en mesure de contrôler ce qui se passe lorsqu'une tâche échoue, récupérer après l'échec d'une tâche et contrôler les conditions amenant une tâche à signaler un échec.

## Gestion des échecs de tâche dans les plays

Ansible évalue les codes de retour de chaque tâche pour déterminer si une tâche a réussi ou échoué. En règle générale, lorsqu'une tâche échoue, Ansible met immédiatement fin au reste du play sur cet hôte, en ignorant toutes les tâches suivantes.

Toutefois, dans certains cas, vous souhaiterez poursuivre l'exécution d'un play, même si une tâche échoue. Par exemple, il se peut qu'une tâche spécifique échoue, et vous pouvez souhaiter effectuer une récupération en exécutant une autre tâche de manière conditionnelle. Plusieurs fonctions Ansible peuvent être utilisées pour mieux gérer les erreurs liées aux tâches.

### Ignorance de l'échec d'une tâche

Par défaut, si une tâche échoue sur un hôte, le play est interrompu pour cet hôte. Toutefois, vous pouvez remplacer ce comportement en ignorant les tâches qui ont échoué. Vous pouvez utiliser le mot-clé **ignore\_errors** dans une tâche pour remplacer le comportement par défaut.

L'exemple suivant montre comment utiliser **ignore\_errors** dans une tâche pour poursuivre l'exécution d'un playbook sur l'hôte, même en cas d'échec d'une tâche. Par exemple, si **notafeature** n'est pas une fonction valide, l'installation de la fonction échoue, mais si **ignore\_errors** est défini sur **true**, Ansible doit continuer.

```
- name: Install notafeature with sub features and management tools
  win_feature:
    name: notafeature
    state: present
    include_sub_features: yes
    include_management_tools: yes
    ignore_errors: true
```

### Exécution forcée des gestionnaires après l'échec d'une tâche

Lorsqu'une tâche échoue et que le play est interrompu sur cet hôte, tout gestionnaire qui a été notifié par des tâches précédentes du play n'est pas exécuté. Si vous définissez le mot-clé **force\_handlers** sur **true** sur le play, Ansible exécute les gestionnaires que le play appelle même si ce dernier a été interrompu à cause de l'échec d'une tâche plus récente.

L'exemple suivant illustre comment utiliser le mot-clé **force\_handlers** dans un play pour forcer l'exécution du gestionnaire même si une tâche échoue :

```
- name: Testing Handlers
  hosts: demo
  force_handlers: true
  tasks:
    - name: Install Dotnet4.6.1
      win_chocolatey:
        name: dotnet4.6.1
        state: present
      notify:
        - reboot server

    - name: Install notafeature with sub features and management tools
      win_feature:
        name: notafeature
        state: present
        include_sub_features: yes
        include_management_tools: yes

    handlers:
      - name: reboot server
        win_reboot:
```



### Note

Gardez à l'esprit que les gestionnaires sont notifiés lorsqu'une tâche signale un résultat **changed**, mais ne le sont pas lorsque des tâches signalent **ok** ou **failed**.

## Spécification des conditions d'échec d'une tâche

Vous pouvez utiliser le mot-clé **failed\_when** sur une tâche pour spécifier les conditions indiquant que la tâche a échoué.

Par exemple, vous pouvez indiquer que la mise à jour d'un serveur Windows échoue uniquement si une variable **failed\_update\_count** est définie et qu'elle est supérieure à 0.

```
- name: Installing Windows Updates
  win_updates:
    category_names:
      - CriticalUpdates
      - SecurityUpdates
    state: installed
    reboot: true
    reboot_timeout: 600
    register: update_result
    failed_when:
      - update_result.failed_update_count is defined
      - update_result.failed_update_count > 0
```

Vous pouvez utiliser le module **fail** pour forcer un échec de tâche. Vous pouvez réécrire le scénario mentionné précédemment sous la forme de deux tâches :

```
- name: Installing Windows Updates
  win_updates:
    category_names:
      - CriticalUpdates
      - SecurityUpdates
    state: installed
    reboot: true
    reboot_timeout: 600
  register: update_result
  ignore_errors: true

- name: Reporting failure
  fail:
    msg: The update has failed
  when:
    - update_result.failed_update_count is defined
    - update_result.failed_update_count > 0
```

Le module **fail** vous permet de fournir un message d'échec clair pour la tâche. Cette approche permet également les échecs différés, vous permettant d'exécuter des tâches intermédiaires pour effectuer ou annuler d'autres modifications.

## Gestion des erreurs à l'aide de blocs Ansible

Dans les playbooks, les *blocs* sont des clauses qui regroupent des tâches de manière logique ; vous pouvez les utiliser pour contrôler la façon dont les tâches sont exécutées. Par exemple, un bloc de tâche peut avoir un mot-clé **when** pour appliquer une condition à plusieurs tâches.

L'exemple suivant montre comment définir un bloc dans lequel le module **win\_updates** s'exécute.

```
- name: Testing Block
  block:
    - win_updates:
        category_names:
          - CriticalUpdates
          - SecurityUpdates
        state: installed
        reboot: true
        reboot_timeout: 600
```

Vous pouvez également utiliser des blocs pour gérer les erreurs à l'aide des instructions **rescue** et **always**. Si une tâche d'un bloc échoue, Ansible exécute les tâches dans un bloc **rescue** pour récupérer l'exécution du play. Après l'exécution des tâches de la clause **block**, ou les tâches de la clause **rescue** en cas d'échec, Ansible exécute toutes les tâches de la clause **always**.

En résumé :

- **block** : définit les tâches principales à exécuter.
- **rescue** : définit les tâches devant s'exécuter si les tâches dans la clause **block** échouent.
- **always** : définit les tâches qui sont systématiquement exécutées, indépendamment de la réussite ou de l'échec des tâches définies dans les clauses **block** et **rescue**.

L'exemple suivant indique comment mettre en œuvre un bloc dans un playbook. Même si les tâches dans la clause **block** échouent, Ansible exécute celles des clauses **rescue** et **always**.

```
- name: Testing block, rescue and always
  hosts: demo
  tasks:
    - name: Block, Rescue and Always
      block:
        # Ansible starts by running these tasks...
        - name: Installing Windows Updates
          win_updates:
            category_names:
              - CriticalUpdates
              - SecurityUpdates
            state: installed
            reboot: true
            reboot_timeout: 600
          register: update_result
          failed_when:
            - update_result.failed_update_count is defined
            - update_result.failed_update_count > 0

      rescue:
        # ... these tasks only run if the tasks in the block fail.
        - name: Showing up a failure message
          debug:
            msg: The Critical and Security Updates has failed.

      always:
        # However, these tasks always run
        - name: Updating mandatory ServicePacks and FeaturePacks
          win_updates:
            category_names:
              - ServicePacks
              - FeaturePacks
            state: installed
            reboot: true
            reboot_timeout: 600
```



### Important

La condition **when** d'une clause **block** s'applique également à ses clauses **rescue** et **always** le cas échéant.

## Contrôle du moment où une tâche signale un changement d'état

Lorsqu'une tâche apporte une modification à un hôte géré, elle signale l'état **changed** et informe les gestionnaires. Lorsqu'une tâche n'a besoin d'apporter aucune modification, elle signale l'état **ok** et ne notifie pas les gestionnaires.

Utilisez le mot-clé **changed\_when** pour contrôler le moment auquel une tâche signale la modification d'un résultat. L'exemple suivant montre comment **win\_shell** est utilisé pour

exécuter une commande sur des serveurs. Généralement, il signale toujours un état **changed** lorsque le module est exécuté. Pour modifier ce comportement, définissez **changed\_when** sur **false** afin qu'Ansible signale uniquement un état **ok** ou **failed**.

```
- name: Testing changed_when by running a command
hosts: demo
tasks:
- name: Executing a command on Windows
  win_shell: hostname
  changed_when: false
```



## Références

### Blocs – Documentation Ansible

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_blocks.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_blocks.html)

### Error Handling In Playbooks — Ansible Documentation

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_error\\_handling.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_error_handling.html)

## ► Exercice guidé

# Gestion des échecs de tâche

Dans cet exercice, vous allez explorer différentes manières de gérer l'échec d'une tâche dans un playbook Ansible.

## Résultats

Vous serez en mesure de personnaliser la manière dont Ansible identifie et gère les erreurs et le résultat d'état.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. Ouvrez l'éditeur Visual Studio Code et clonez le référentiel **control** sur **workstation**.
  - 1.1. À partir de **workstation**, ouvrez l'éditeur Visual Studio Code en double-cliquant sur son icône sur le Bureau.
  - 1.2. Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour accéder à la palette de commandes.  
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.
  - 1.3. Saisissez l'URL de référentiel `https://gitlab.example.com/student/control.git` et choisissez le dossier **Documents** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier `C:\Users\student\Documents\control`.
  - 1.4. Dans le coin inférieur droit, cliquez sur **Open** pour ouvrir le projet.
- ▶ 2. Ouvrez le fichier dans le référentiel **control** nommé **install-iis.yml**.
- ▶ 3. Avec Ansible, vous pouvez utiliser des conditions pour les tâches qui vérifient la mise en œuvre des fonctions qui ne sont pas encore disponibles.  
Ajoutez une tâche qui installe une fonction personnalisée nommée **Team-Application** que votre équipe envisage d'ajouter ultérieurement à l'environnement. Assurez-vous que cette tâche n'interrompt pas l'exécution du playbook tant que la fonction n'est pas créée en ajoutant l'argument **ignore\_errors: yes**.

```
- name: Team-Application is installed
  win_feature:
    name: Team-Application
    state: present
  ignore_errors: yes
```

- 4. Ansible permet de s'assurer que le contenu de votre environnement correspond aux sorties attendues et atténue les éventuelles divergences. Utilisez une variable enregistrée pour capturer la sortie de la tâche, puis utilisez-la de manière conditionnelle pour exécuter les tâches suivantes.

- 4.1. Ajoutez une tâche qui lit le contenu du texte de la page d'accueil **win1.example.com** et le stocke dans la variable **index**.

```
- name: Homepage index is set
  win_uri:
    url: http://win1.example.com
    return_content: yes
  register: index
```

- 4.2. Ajoutez une autre ligne à la tâche qui utilise l'instruction conditionnelle **failed\_when** pour comparer la sortie de la page d'accueil stockée dans la variable **index** avec la chaîne "**Ansible is Great**". Cette comparaison échouera si le contenu de l'index ne correspond pas à cette phrase.

```
- name: Homepage index is set
  win_uri:
    url: http://win1.example.com
    return_content: yes
  register: index
  failed_when: index.content != "Ansible is great"
```

- 4.3. Une fois vos modifications apportées, le playbook doit se présenter comme suit :

```
---
- name: Install the IIS web service
  hosts: win1.example.com

  tasks:
    - name: IIS service installed
      win_feature:
        name: Web-Server
        state: present

    - name: IIS service started
      win_service:
        name: W3Svc
        state: started

    - name: Website index.html created
      win_copy:
```

```
content: "Hello world!"  
dest: C:\Inetpub\wwwroot\index.html  
  
- name: Team-Application is installed  
  win_feature:  
    name: Team-Application  
    state: present  
    ignore_errors: yes  
  
- name: Homepage index is set  
  win_uri:  
    url: http://win1.example.com  
    return_content: yes  
  register: index  
  failed_when: index.content != "Ansible is great"
```

- ▶ 5. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
- 5.1. Cliquez sur **File → Save** ou tapez **Ctrl+S** pour enregistrer le fichier.
  - 5.2. Accédez au volet **Source Control**, puis cliquez sur le **+** en regard du fichier **install-iis.yml** pour indexer les modifications.
  - 5.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 5.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- ▶ 6. À partir de **workstation**, accédez à votre instance Ansible Tower sur `http://tower.example.com`. Connectez-vous en tant que **student** avec **RedHat123@!** comme mot de passe.
- ▶ 7. Lancez une tâche à partir du modèle de tâche **Run control project**.
- 7.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 7.2. Cliquez sur le nom de **Run control project**.
  - 7.3. Sélectionnez le playbook **install-iis.yml** dans la liste **PLAYBOOK**.

## chapitre 6 | Mise en œuvre d'un contrôle de tâche

- 7.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
- 7.5. Étant donné que l'erreur est ignorée pour la tâche **Team-Application is installed**, elle est signalée comme un échec, mais l'exécution du playbook continue. Étant donné que le contenu de la page d'accueil n'est pas correctement défini, la tâche échoue sur la tâche **Homepage index is set**.

```

13
14 TASK [Website index.html created] *****
15 changed: [win1.example.com]
16
17 TASK [Team-Application is installed] *****
18 fatal: [win1.example.com]: FAILED! => {"changed": false, "msg": "Failed to install Windows Feature: ArgumentNotValid: The role, role service, or feature name is not valid: 'Team-Application'. The name was not found."}
19 ...ignoring
20
21 TASK [Homepage index is set] *****
22 fatal: [win1.example.com]: FAILED! => {"accept_ranges": "bytes", "changed": false, "character_set": "ISO-8859-1", "content": "Hello World!", "content_encoding": "", "content_length": "12", "content_type": "text/html", "cookies": [], "date": "Sat, 21 Sep 2019 17:22:25 GMT", "e_tag": "\"484ff1ba170d51:0\"", "elapsed": 0.1093835, "failed_when_result": true, "headers": ["Accept-Ranges", "Content-Length", "Content-Type", "Date", "ETag", "Last-Modified", "Server"], "is_from_cache": false, "is_mutually_authenticated": false, "last_modified": "Sat, 21 Sep 2019 17:22:12 GMT", "method": "GET", "protocol_version": {"Build": -1, "Major": 1, "MajorRevision": -1, "Minor": 1, "MinorRevision": -1, "Revision": -1}, "response_uri": "http://win1.example.com/", "server": "Microsoft-IIS/10.0", "status_code": 200, "status_description": "OK", "supports_headers": true, "url": "http://win1.example.com"}
23
24 PLAY RECAP *****
25 win1.example.com : ok=5 changed=2 unreachable=0 failed=1 skip
ped=0 rescued=0 ignored=1
26

```

► 8. Mettez à jour votre playbook pour corriger l'échec.

- 8.1. Dans Visual Studio Code, localisez la tâche existante nommée **Website index.html created** et mettez à jour le contenu pour qu'il définisse la page d'accueil sur le serveur **win1.example.com** sur **Ansible is great**.

```
- name: Website index.html created
  win_copy:
    content: "Ansible is great"
    dest: C:\Inetpub\wwwroot\index.html
```

- 8.2. Enregistrez, validez et synchronisez votre travail dans Visual Studio Code. Relancez la tâche dans Ansible Tower. Avec le jeu de contenu de la page d'accueil approprié, votre playbook s'exécute correctement.

```
Run control project
PLAYS 1 TASKS 6 HOSTS 1 ELAPSED 00:00:48
SEARCH Q KEY

-
3 PLAY [Install the IIS web service] *****
4
5 TASK [Gathering Facts] *****
6 ok: [win1.example.com]
7
8 TASK [IIS service installed] *****
9 ok: [win1.example.com]
10
11 TASK [IIS service started] *****
12 ok: [win1.example.com]
13
14 TASK [Website index.html created] *****
15 changed: [win1.example.com]
16
17 TASK [Team-Application is installed] *****
18 fatal: [win1.example.com]: FAILED! => {"changed": false, "msg": "Failed to install Windows Feature: ArgumentNotValid: The role, role service, or feature name is not valid: 'Team-Application'. The name was not found."}
19 ...ignoring
20
21 TASK [Homepage index is set] *****
22 ok: [win1.example.com]
23
24 PLAY RECAP *****
25 win1.example.com : ok=6    changed=1    unreachable=0    failed=0    skip
ped=0    rescued=0    ignored=1
26
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Mise en œuvre d'un contrôle de tâche

### Liste de contrôle des performances

Au cours de cet atelier, vous allez écrire un playbook pour déployer un service, et utiliser des tests conditionnels, des gestionnaires et des techniques pour gérer l'échec des tâches dans votre playbook.

### Résultats

Vous serez en mesure de gérer le flux des tâches à l'aide de tests conditionnels et de gestionnaires.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Red Hat Ansible Tower et GitLab à partir du poste de travail Windows.

Dans le cadre de cet atelier, utilisez le modèle de tâche **Run control project** pour exécuter votre playbook au sein d'Ansible Tower.



#### Note

Cette activité nécessite **win2.example.com** dans l'inventaire **Default inventory**. Vous avez ajouté l'hôte dans un exercice précédent. Si l'hôte n'est pas présent, suivez les instructions de l'exercice guidé *Mise en œuvre de plusieurs plays* pour le créer.

1. Connectez-vous à votre instance Red Hat Ansible Tower.
2. Dans le menu **Inventories**, assurez-vous que **win2.example.com** est membre de l'inventaire **Default**.
3. Lancez Visual Studio Code et clonez le référentiel <https://gitlab.example.com/student/control.git> [dans votre dossier Documents], puis ouvrez le projet.
4. Créez un nouveau playbook dans le référentiel **control** nommé **review.yml**.
5. Rédigez un playbook qui utilise l'inventaire **default** contenant à la fois les machines **win1.example.com** et **win2.example.com**.
6. Pour le système **win1**, installez la fonction **Web-Server** et assurez-vous que le service web est en cours d'exécution. Utilisez une boucle pour ajouter les fonctions **Web-Security**, **Web-Basic-Auth** et **Web-IP-Security** supplémentaires au système, et mettez à jour la page d'accueil du serveur pour afficher le nom d'hôte de la machine.

7. Écrivez un gestionnaire nommé **Machine is restarted** qui redémarre le serveur et ajoutez-le à chacune des tâches d'installation de fonctions.
8. Ajoutez une tâche conditionnelle pour le système **win2.example.com** afin d'installer toutes les mises à jour de sécurité Windows. Notifyez le gestionnaire **Machine is restarted**.
9. Ajoutez une instruction **when** pour chaque tâche du playbook qui utilise un fait Ansible pour correspondre au nom d'hôte. Cet ajout doit faire en sorte que le playbook n'exécute les tâches désignées que sur la machine cible spécifiée par les instructions de l'atelier.
10. Dans Visual Studio Code, enregistrez, validez et synchronisez votre fichier dans le référentiel **control**.

Dans Red Hat Ansible Tower, actualisez les fichiers pour le projet **control repository**.

Lancez une tâche à partir du modèle de tâche **Run control project**.

11. Vérifiez que les deux systèmes présentent la configuration attendue.

L'atelier est maintenant terminé.

## ► Solution

# Mise en œuvre d'un contrôle de tâche

### Liste de contrôle des performances

Au cours de cet atelier, vous allez écrire un playbook pour déployer un service, et utiliser des tests conditionnels, des gestionnaires et des techniques pour gérer l'échec des tâches dans votre playbook.

### Résultats

Vous serez en mesure de gérer le flux des tâches à l'aide de tests conditionnels et de gestionnaires.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Red Hat Ansible Tower et GitLab à partir du poste de travail Windows.

Dans le cadre de cet atelier, utilisez le modèle de tâche **Run control project** pour exécuter votre playbook au sein d'Ansible Tower.



#### Note

Cette activité nécessite **win2.example.com** dans l'inventaire **Default inventory**. Vous avez ajouté l'hôte dans un exercice précédent. Si l'hôte n'est pas présent, suivez les instructions de l'exercice guidé *Mise en œuvre de plusieurs plays* pour le créer.

1. Connectez-vous à votre instance Red Hat Ansible Tower.
  - 1.1. À partir de **workstation**, connectez-vous à Ansible Tower sur <https://tower.example.com> ou en double-cliquant sur l'icône du Bureau **Ansible Tower**. Authentifiez-vous avec le nom d'utilisateur **student** et le mot de passe **RedHat123@!**.
2. Dans le menu **Inventories**, assurez-vous que **win2.example.com** est membre de l'inventaire **Default**.
  - 2.1. Si vous avez besoin d'ajouter cette configuration, accédez à la page **Inventories** dans Ansible Tower, ajoutez l'hôte **win2.example.com**, puis placez cet hôte dans l'inventaire **default**.
3. Lancez Visual Studio Code et clonez le référentiel <https://gitlab.example.com/student/control.git> [dans votre dossier Documents], puis ouvrez le projet.

- 3.1. Sur **Workstation**, double-cliquez sur l'icône du Bureau nommée Visual Studio Code.  
Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.
  - 3.2. Saisissez l'URL de référentiel `https://gitlab.example.com/student/control.git` et choisissez le dossier **Documents** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier `C:\Users\student\Documents\control` sur **workstation**.
  - 3.3. Dans le coin inférieur droit, cliquez sur **Open** pour ouvrir le projet.
4. Créez un nouveau playbook dans le référentiel **control** nommé **review.yml**.
    - 4.1. Dans Visual Studio Code, cliquez sur l'icône **New File** en regard de **CONTROL** et nommez le fichier **review.yml**.
  5. Rédigez un playbook qui utilise l'inventaire **default** contenant à la fois les machines **win1.example.com** et **win2.example.com**.
  6. Pour le système **win1**, installez la fonction **Web-Server** et assurez-vous que le service web est en cours d'exécution. Utilisez une boucle pour ajouter les fonctions **Web-Security**, **Web-Basic-Auth** et **Web-IP-Security** supplémentaires au système, et mettez à jour la page d'accueil du serveur pour afficher le nom d'hôte de la machine.
    - 6.1. Mettez à jour le playbook pour qu'il corresponde à ce contenu :

```
---
- name: Control review lab
hosts: all

tasks:
  - name: Web server feature is installed
    win_feature:
      name: Web-Server
      state: present

  - name: Web server is started
    win_service:
      name: W3Svc
      state: started

  - name: Website index.html created
    win_copy:
      content: "win1.example.com"
      dest: C:\Inetpub\wwwroot\index.html

  - name: Web security features are installed
    win_feature:
      name: "{{ item }}"
      state: present
    loop:
      - Web-Security
```

**chapitre 6 |** Mise en œuvre d'un contrôle de tâche

- Web-Basic-Auth
- Web-IP-Security

7. Écrivez un gestionnaire nommé **Machine is restarted** qui redémarre le serveur et ajoutez-le à chacune des tâches d'installation de fonctions.

7.1. Le playbook doit à présent présenter le contenu suivant :

```
---
```

```
- name: Control review lab
  hosts: all

  tasks:

    - name: Web server feature is installed
      win_feature:
        name: Web-Server
        state: present
      notify: Machine is restarted

    - name: Web server is started
      win_service:
        name: W3Svc
        state: started

    - name: Website index.html created
      win_copy:
        content: "win1.example.com"
        dest: C:\Inetpub\wwwroot\index.html

    - name: Web security features are installed
      win_feature:
        name: "{{ item }}"
        state: present
      loop:
        - Web-Security
        - Web-Basic-Auth
        - Web-IP-Security
      notify: Machine is restarted

  handlers:

    - name: Machine is restarted
      win_reboot:
```

8. Ajoutez une tâche conditionnelle pour le système **win2.example.com** afin d'installer toutes les mises à jour de sécurité Windows. Notifyez le gestionnaire **Machine is restarted**.

8.1. Le playbook doit à présent contenir les éléments suivants :

```
---
```

```
- name: Control review lab
  hosts: all

  tasks:

    - name: Web server feature is installed on win1
      win_feature:
        name: Web-Server
        state: present
      notify: Machine is restarted

    - name: Web server is started on win1
      win_service:
        name: W3Svc
        state: started

    - name: Website index.html created on win1
      win_copy:
        content: "win1.example.com"
        dest: C:\Inetpub\wwwroot\index.html

    - name: Web security features are installed on win1
      win_feature:
        name: "{{ item }}"
        state: present
      loop:
        - Web-Security
        - Web-Basic-Auth
        - Web-IP-Security
      notify: Machine is restarted

    - name: Security updates applied on win2
      win_updates:
        category_names: SecurityUpdates
      notify: Machine is restarted

  handlers:

    - name: Machine is restarted
      win_reboot:
```

9. Ajoutez une instruction **when** pour chaque tâche du playbook qui utilise un fait Ansible pour correspondre au nom d'hôte. Cet ajout doit faire en sorte que le playbook n'exécute les tâches désignées que sur la machine cible spécifiée par les instructions de l'atelier.

9.1. Mettez à jour le playbook pour qu'il corresponde à ce contenu :

```
---
```

```
- name: Control review lab
  hosts: all
```

```
tasks:  
  
  - name: Web server feature is installed on win1  
    win_feature:  
      name: Web-Server  
      state: present  
    notify: Machine is restarted  
    when: "'win1' in inventory_hostname"  
  
  - name: Web server is started on win1  
    win_service:  
      name: W3Svc  
      state: started  
    when: "'win1' in inventory_hostname"  
  
  - name: Website index.html created on win1  
    win_copy:  
      content: "win1.example.com"  
      dest: C:\Inetpub\wwwroot\index.html  
    when: "'win1' in inventory_hostname"  
  
  - name: Web security features are installed on win1  
    win_feature:  
      name: "{{ item }}"  
      state: present  
    loop:  
      - Web-Security  
      - Web-Basic-Auth  
      - Web-IP-Security  
    notify: Machine is restarted  
    when: "'win1' in inventory_hostname"  
  
  - name: Security updates applied on win2  
    win_updates:  
      category_names: SecurityUpdates  
    notify: Machine is restarted  
    when: "'win2' in inventory_hostname"  
  
handlers:  
  
  - name: Machine is restarted  
    win_reboot:
```

10. Dans Visual Studio Code, enregistrez, validez et synchronisez votre fichier dans le référentiel **control**.

Dans Red Hat Ansible Tower, actualisez les fichiers pour le projet **control repository**.

Lancez une tâche à partir du modèle de tâche **Run control project**.

- 10.1. Dans Visual Studio Code, enregistrez votre fichier, ajoutez un bref message de validation, et synchronisez votre fichier dans le référentiel **control**.

- 10.2. Dans Red Hat Ansible Tower, cliquez sur **Get latest SCM revision** dans la page **Projects** en regard du projet **control repository**.
- 10.3. Sélectionnez le modèle de tâche **Run control project** dans la page **Templates** pour le modifier. Sélectionnez **review.yml** dans la liste **PLAYBOOK**, puis cliquez sur **Launch**.
11. Vérifiez que les deux systèmes présentent la configuration attendue.
  - 11.1. Examinez la sortie de tâche au sein d'Ansible Tower pour vous assurer que toutes les tâches aient été correctement exécutées.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Les boucles sont utilisées pour itérer des listes de valeurs, comme une liste simple de chaînes ou une liste complexe de dictionnaires.
- Les conditions sont utilisées pour exécuter des tâches uniquement lorsque certaines conditions sont remplies.
- Les gestionnaires sont des tâches spéciales qui s'exécutent à la fin d'une section s'ils sont notifiés par des tâches au sein de cette section.
- Les gestionnaires ne sont notifiés que lorsqu'une tâche indique avoir modifié quelque chose sur l'hôte géré.
- Les tâches peuvent ignorer les échecs, appeler des gestionnaires même en cas d'échec d'une tâche, marquer les tâches comme ayant échoué même si elles réussissent, ou remplacer le comportement qui entraîne le marquage d'une tâche comme modifiée.
- Les blocs permettent de regrouper des tâches en une unité et d'exécuter d'autres tâches si toutes les tâches du bloc aboutissent.

## chapitre 7

# Déploiement de fichiers sur des hôtes gérés

### Objectif

Déployer, modifier et gérer des fichiers sur vos hôtes gérés.

### Résultats

- Créer, installer, modifier et supprimer des fichiers sur des hôtes gérés, et configurer des permissions de fichiers et d'autres caractéristiques.
- Déployer des fichiers personnalisés sur des hôtes gérés à l'aide de modèles Jinja2.

### Sections

- Modification et transfert de fichiers sur des hôtes (et exercice guidé)
- Déploiement de fichiers avec Jinja2 (et exercice guidé)

### Atelier

Déploiement de fichiers sur des hôtes gérés

# Modification et transfert de fichiers sur des hôtes

---

## Résultats

Au terme de cette section, vous serez en mesure de créer, installer, modifier et supprimer des fichiers sur des hôtes gérés, et configurer des permissions de fichiers et d'autres caractéristiques.

## Description des modules de fichiers

Red Hat Ansible Engine est livré avec une grande collection de modules (également connue en tant que *bibliothèque de modules*) qui sont développés dans le cadre du projet Ansible en amont. Pour faciliter leur organisation, leur documentation et leur gestion, ils sont organisés en groupes selon leur fonction, à la fois dans la documentation et lorsqu'ils sont installés sur un système.

La bibliothèque de modules Windows inclut des modules qui vous permettent d'accomplir la plupart des tâches liées à la gestion de fichiers Windows, telles que la création, la copie, l'édition, la recherche et la suppression de fichiers, et la modification d'ACL et d'autres attributs de fichiers.

Le tableau suivant fournit une liste des modules de gestion de fichiers principaux :

Nom du module	Description du module
<b>win_acl</b>	Ajoute ou supprime des droits et des permissions dans le fichier, le dossier ou la clé de registre spécifiés pour un utilisateur ou un groupe donné.
<b>win_copy</b>	Copie des fichiers à partir de nœuds de contrôle locaux vers des nœuds gérés à distance.
<b>win_stat</b>	Affiche des informations sur les fichiers.
<b>win_file</b>	Crée des fichiers et met à jour les horodatages de modification sur les fichiers existants. Ce module ne modifie ni la propriété ni les permissions, et ne manipule pas non plus les liens.
<b>win_robocopy</b>	Synchronise le contenu des fichiers ou des répertoires depuis la source vers une destination sur les nœuds de contrôle Windows locaux. Ce module appelle la commande <b>robocopy</b> , qui est disponible sur les systèmes Microsoft Windows modernes.
<b>win_owner</b>	Définit le propriétaire des fichiers et des répertoires.
<b>win_file_version</b>	Récupère les versions de compilation des DLL ou des fichiers .EXE.
<b>win_find</b>	Recherche des fichiers sur des hôtes gérés basés sur Windows en fonction de critères spécifiques.
<b>win_tempfile</b>	Crée des fichiers et des répertoires temporaires.
<b>win_unzip</b>	Décomprime les fichiers compressés et les archives sur les hôtes gérés.

## Exemples d'automatisation avec des modules de fichiers

La création, la copie, la modification, la recherche et la suppression de fichiers, ainsi que la modification d'ACL sur des hôtes gérés, sont des tâches courantes qui peuvent être simplifiées à l'aide de modules de la bibliothèque de modules Windows. Les exemples suivants montrent comment utiliser ces modules pour automatiser les tâches courantes de gestion des fichiers.

### Vérification de l'existence d'un fichier ou d'un répertoire sur vos systèmes

Utilisez le module **win\_file** pour créer un fichier ou un répertoire sur des hôtes gérés, et pour mettre à jour l'heure de modification si le fichier existe déjà. Ce module ne modifie ni la propriété ni les permissions de fichiers, et ne manipule pas non plus les liens. Utilisez les modules **win\_acl** ou **win\_owner** pour ces tâches.

```
- name: Create a file or modify time stamp if already present
  win_file:
    path: C:\Temp\test.conf
    state: touch
```

L'exemple suivant montre comment créer un répertoire et des répertoires parents obligatoires, s'ils n'existent pas.

```
- name: Create a folder and its parent folders
  win_file:
    path: C:\Temp\folder\subfolder
    state: directory
```



#### Note

Utilisez le module **win\_tempfile** pour créer des fichiers et des répertoires temporaires.

Si vous ne spécifiez pas d'emplacement pour le fichier temporaire, le module utilise le répertoire temporaire du système par défaut défini par la variable d'environnement **%TEMP%**.

### Récupération de l'état d'un fichier sur des hôtes gérés

Le module **win\_stat** renvoie un hachage/dictionnaire de valeurs contenant les données d'état du fichier. Utilisez ces informations pour examiner les données de sortie pour des attributs et des valeurs spécifiques.

```
- name: Check the information on the file if the file exists
  win_stat:
    path: C:\Temp\test.conf
    register: testfile

- name: Print results of win_stat
  debug:
    var: testfile
```

L'exemple suivant illustre la sortie du module **win\_stat**.

```
"testfile": {
    "changed": false,
    "failed": false,
    "stat": {
        "attributes": "Archive",
        "checksum": "da39a3ee5e6b4b0d3255bfef95601890af80709",
        "creationtime": 1567356109.3531778,
        "exists": true,
        "extension": ".conf",
        "filename": "test.conf",
        "hlnk_targets": [],
        "isarchive": true,
        "isdir": false,
        "ishidden": false,
        "isjunction": false,
        "islnk": false,
        "isreadonly": false,
        "isreg": true,
        "isshared": false,
        "lastaccesstime": 1567356109.3531778,
        "lastwritetime": 1567356195.2883112,
        "nlink": 1,
        "owner": "BUILTIN\\Administrators",
        "path": "C:\\Temp\\test.conf",
        "size": 0
    }
}
```

Voici un ensemble de tâches plus pratique :

```
- name: Collect the information on C:\\Temp\\test.conf
  win_stat:
    path: C:\\Temp\\test.conf
    register: testfile

- name: Print C:\\Temp\\test.conf checksum
  debug:
    msg: The checksum of test.conf is {{ testfile['stat']['checksum'] }}
```

## Modification des attributs de fichiers

Utilisez le module **win\_owner** pour modifier la propriété d'un fichier ou d'un répertoire, et le module **win\_acl** pour gérer les permissions de fichier.

La tâche suivante illustre la modification de la propriété du répertoire **C:\\Temp\\folder**. Si vous souhaitez modifier le propriétaire de tous les fichiers et répertoires du répertoire **C:\\Temp\\folder**, utilisez l'option **recurse: true**, qui est définie sur **false** par défaut.

```
- name: Change the ownership of the directory
  win_owner:
    path: C:\Temp\folder
    user: SYSTEM
    recurse: false
```

Pour modifier l'ACL d'un fichier, utilisez le module **win\_acl**.

```
- name: Prevent user Robin from having any access to a directory
  win_acl:
    path: C:\Confidential
    user: Robin
    rights: Read,Write,Modify,FullControl,Delete
    type: deny
    state: present
```



### Note

Pour plus d'informations sur les autorisations du système de fichiers Windows, rendez-vous à la page *FileSystemRights Enum* accessible sur <https://msdn.microsoft.com/en-us/library/system.security.accesscontrol.filesystemrights.aspx>

## Copie de fichiers sur des hôtes gérés

Dans l'exemple suivant, le module **win\_copy** copie un fichier situé sur le nœud de contrôle Ansible (par exemple, le serveur Ansible Tower) vers les hôtes gérés.

Par défaut, ce module suppose que le paramètre **force: yes** est défini. Ce paramètre oblige le module à écraser le fichier distant s'il existe et qu'il présente un contenu différent du fichier en cours de copie. Si **force** est défini sur **no**, Ansible ne copie le fichier sur l'hôte géré que s'il n'existe pas encore.

Dans l'exemple suivant, Ansible copie le fichier **test.conf** à partir du répertoire **files** dans lequel se trouve le playbook, vers **C:\Temp\test.conf** sur les hôtes gérés par Microsoft Windows.

```
- name: Copy a single file
  win_copy:
    src: files/test.conf
    dest: C:\Temp\test.conf
```



### Note

Si vous copiez des fichiers volumineux, envisagez de les héberger sur un serveur Web et d'utiliser le module **win\_get\_url**, au lieu du module **win\_copy**. Le module **win\_copy** s'exécute au moyen du protocole WinRM, ce qui n'est pas un mécanisme de transport très efficace.

## Copier des fichiers sur des hôtes gérés

Le module **win\_robocopy** est un wrapper autour du mécanisme *Copie de fichiers robuste pour Windows* (Robocopy), disponible dans tous les environnements d'exploitation Microsoft Windows modernes. Utilisez ce module pour copier des fichiers d'un répertoire sur la machine distante vers un autre.

```
- name: Synchronize the contents of one directory with another
  win_robocopy:
    src: C:\Temp
    dest: D:\New
    recurse: yes
```

## Récupération des fichiers de version de compilation

L'exemple suivant illustre l'utilisation du module **win\_file\_version** pour collecter la version de compilation des DLL et des fichiers .exe.

```
- name: Checking the version of cmd.exe
  win_file_version:
    path: C:\Windows\System32\cmd.exe
    register: version

- debug:
  msg: '{{ version }}'
```

## Recherche de fichiers sur les hôtes gérés Windows

Utilisez le module **win\_find** pour rechercher des fichiers en fonction de critères spécifiques. Vous pouvez utiliser plusieurs critères pour rechercher un fichier.

L'exemple suivant utilise le module **win\_find** pour rechercher tous les fichiers dont les noms correspondent à un modèle spécifique. Ce modèle spécifique ordonne à Ansible de localiser tous les fichiers journaux, ainsi que la sortie de la commande ou de l'exécution. Tous les fichiers doivent avoir une taille de 1 Go ou plus, et les fichiers masqués sont compris.

```
- name: Search for a file
  win_find:
    paths: C:\Temp
    hidden: yes
    recurse: yes
    patterns: [ '*.log', '*.out' ]
    size: 1g
```

## Décompression des fichiers

Utilisez le module **win\_zip** pour décompresser les fichiers compressés et les archives. Ce module prend en charge les fichiers .zip en mode natif. Si le module **PowerShell Community Extensions (PSCX)** est installé, alors le module **win\_zip** prend en charge tous les fichiers que **7zip** prend en charge. De plus, l'installation du module **PSCX** est requise pour pouvoir utiliser l'option **recurse**.

L'exemple suivant utilise **win\_psmodule** pour installer le module **PSCX**:

```
- name: Install PSCX module
  win_psmodule:
    name: pscx
    state: present
```

L'exemple suivant appelle le module **win\_unzip** pour décompresser récursivement les fichiers compressés et supprimer l'archive ZIP lorsque vous avez terminé :

```
- name: Unzip gz log files
  win_unzip:
    src: C:\Logs\logs.gz
    dest: C:\Logs\extraced\applicationlogs
    delete_archive: true
    recurse: true
```

## Modification de fichiers sur des hôtes gérés

Pour vous assurer qu'une ligne de texte spécifique existe dans un fichier existant, utilisez le module **win\_lineinfile** :

```
- name: Modifying a line in the service file
  win_lineinfile:
    path: C:\Temp\service
    regexp: '^# port for rdp'
    insertbefore: '^www.*8080/tcp'
    line: '# line modified by ansible'
```

## Création de raccourcis sur des hôtes gérés Windows

Utilisez le module **win\_shortcut** pour créer des raccourcis sur vos systèmes, comme illustré dans l'exemple suivant :

```
- name: Create an application shortcut on the desktop
  win_shortcut:
    src: '%ProgramFiles(x86)%\Google\Chrome\Application\chrome.exe'
    dest: '%UserProfile%\Desktop\Ansible Tower.lnk'
    args: --new-window http://tower.example.com/
    directory: '%ProgramFiles(x86)%\Google\Chrome\Application'
    icon: '%ProgramFiles(x86)%\Google\Chrome\Application\chrome.exe, 0' ❶
```

❶ Indique l'icône à utiliser. Le nom de fichier est suivi d'une virgule et du numéro dans le fichier de bibliothèque (**.dll**), ou utilisez 0 pour un fichier image.



## Références

### **Windows modules — Ansible Documentation**

[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_windows\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html)

### **robocopy — Commandes Windows**

<https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/robocopy>

### **Pscx — PowerShell Gallery**

<https://www.powershellgallery.com/packages/Pscx/3.3.2>

## ► Exercice guidé

# Modification et transfert de fichiers sur des hôtes

Dans cet exercice, vous allez écrire des plays qui obtiennent des informations sur les fichiers, les créer, les copier, les modifier et les supprimer, ainsi que définir des permissions de propriété et de fichier sur ceux-ci.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Créer des répertoires sur des hôtes gérés.
- Copier un fichier zip sur un hôte géré et le décompresser.
- Copier des fichiers d'un répertoire vers un autre sur un hôte géré.
- Créer un raccourci.
- Modifier les droits de propriété et les permissions liés aux fichier.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Red Hat Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. Ouvrez l'éditeur Visual Studio Code et clonez le référentiel **managing-files** sur votre **workstation**.
  - 1.1. À partir de **workstation**, ouvrez l'éditeur Visual Studio Code à partir de l'icône sur le Bureau. Vous pouvez aussi cliquer sur **Recherche Windows**, rechercher **code**, puis ouvrir Visual Studio Code.
  - 1.2. Pour ouvrir la palette de commandes, accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P**.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.
  - 1.3. À l'invite, fournissez l'URL de référentiel, <https://gitlab.example.com/student/managing-files.git>, puis sélectionnez le dossier **Documents** dans le répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **managing-files** sur l'instance **workstation**.
  - 1.4. Lorsque vous êtes invité à ouvrir le projet, cliquez sur **Open**.

- 2. Ouvrez le playbook **unzip.yml**, puis ajoutez des tâches pour déployer le contenu d'un fichier zip sur l'hôte géré **win1**.
- 2.1. Ajoutez une tâche **Temp directory created** qui utilise le module **win\_file** pour vérifier que le répertoire **C:\Temp\DO417\payload** existe. Ansible crée le répertoire s'il n'existe pas.

```
- name: Temp directory created
  win_file:
    path: C:\Temp\DO417\payload
    state: directory
```

  - 2.2. Ajoutez une deuxième tâche qui transfère le fichier **files/example.zip** depuis le clone du référentiel Git **managing-files** sur Red Hat Ansible Tower, vers le répertoire **C:\Temp\DO417** sur l'hôte géré.

```
- name: Zip archive is copied
  win_copy:
    src: files/example.zip
    dest: C:\Temp\DO417\example.zip
```

  - 2.3. Ajoutez une tâche finale **Zip archive is unzipped** pour décompresser l'archive **example.zip** dans le répertoire **C:\Temp\DO417\payload**.

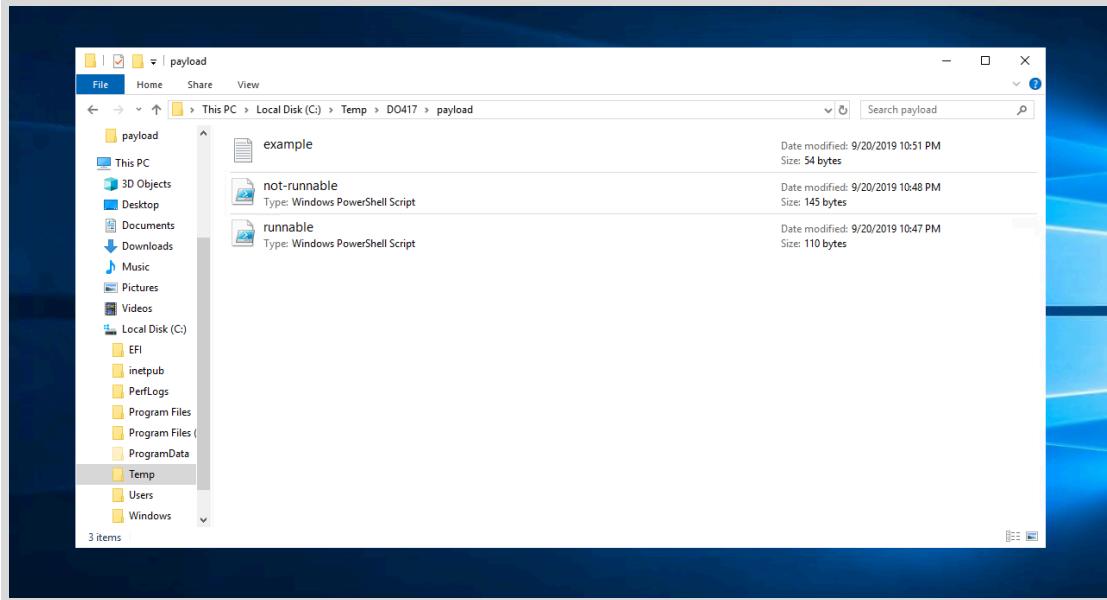
```
- name: Zip archive is unzipped
  win_unzip:
    src: C:\Temp\DO417\example.zip
    dest: C:\Temp\DO417\payload
    delete_archive: yes
```

  - 2.4. Examinez le fichier YAML et vérifiez que votre playbook correspond aux tâches ci-dessous.
- ```
---
- name: Example archive deployed and permissions set
  hosts: win1.example.com
  tasks:
    - name: Temp directory created
      win_file:
        path: C:\Temp\DO417\payload
        state: directory

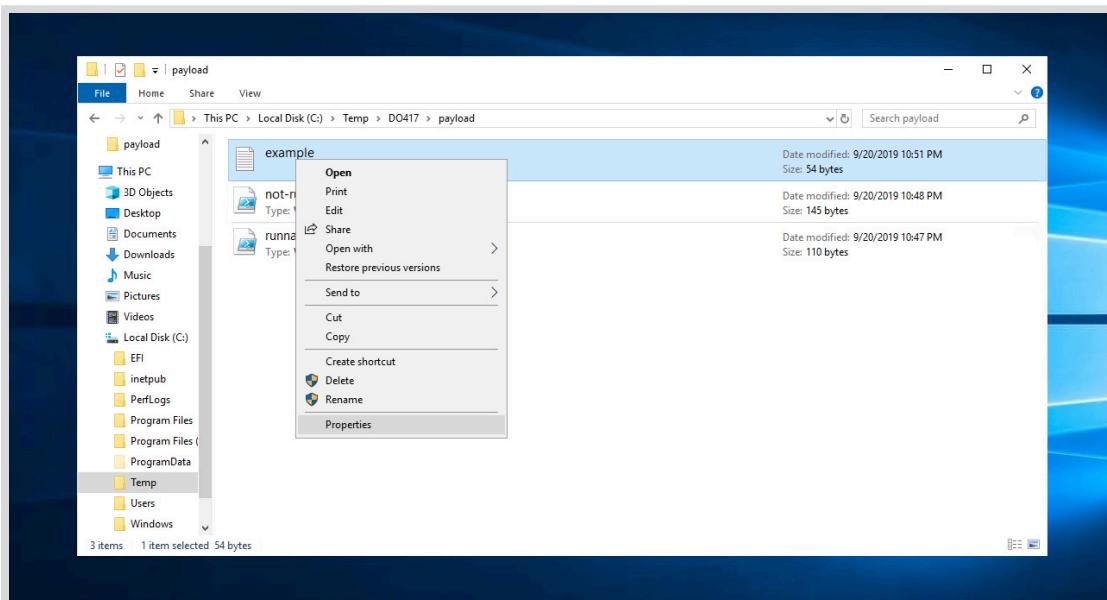
    - name: Zip archive is copied
      win_copy:
        src: files/example.zip
        dest: C:\Temp\DO417\example.zip

    - name: Zip archive is unzipped
      win_unzip:
        src: C:\Temp\DO417\example.zip
        dest: C:\Temp\DO417\payload
        delete_archive: yes
```

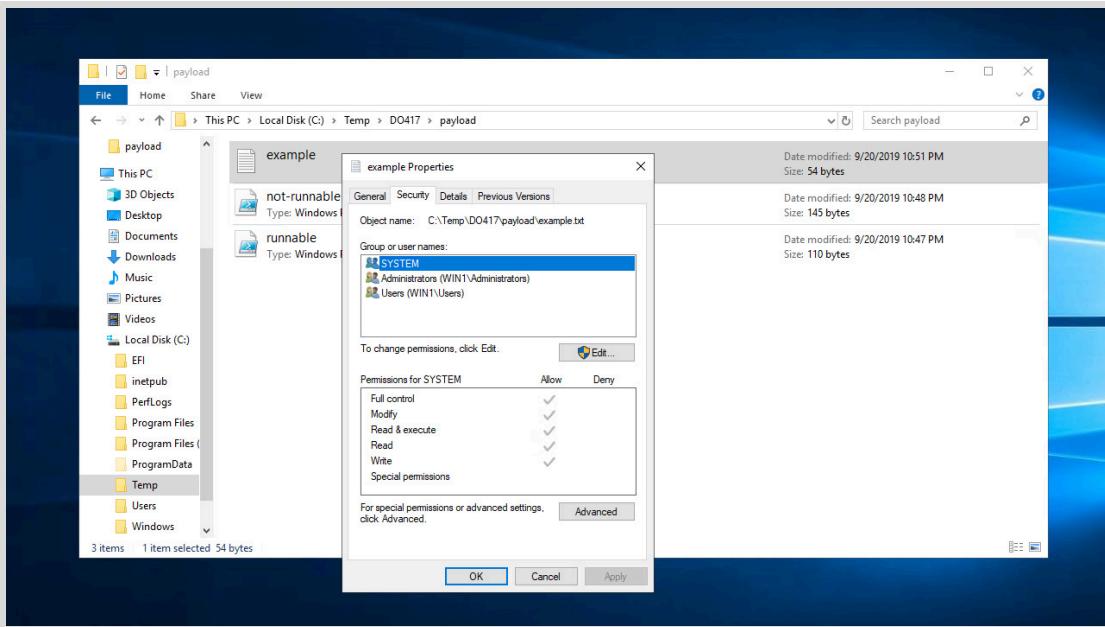
- ▶ 3. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 3.1. Pour enregistrer le fichier, accédez à **File** → **Save** ou appuyez sur **Ctrl+S**.
  - 3.2. Accédez au volet **Source Control** et indexez les modifications.
  - 3.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 3.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- ▶ 4. À partir de **workstation**, accédez à votre instance Red Hat Ansible Tower sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- ▶ 5. Exécutez votre playbook à l'aide du modèle de tâche **Run managing-files project**.
  - 5.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 5.2. Cliquez sur le modèle de tâche **Run managing-files project**.
  - 5.3. Sélectionnez le playbook **unzip.yml** dans la liste **PLAYBOOK**.
  - 5.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
  - 5.5. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**.
- ▶ 6. Connectez-vous à l'hôte géré **win1.example.com** pour inspecter les fichiers non archivés.
  - 6.1. Cliquez sur **Recherche Windows**, recherchez **distance**, puis ouvrez Connexion Bureau à distance.
  - 6.2. Saisissez **win1.example.com** pour l'ordinateur, **devops** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Notez que l'utilisateur **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.
  - 6.3. À l'invite, cliquez sur **Yes** pour accepter le certificat non sécurisé utilisé dans votre environnement de formation.
  - 6.4. Ouvrez **File Explorer** et accédez au répertoire **C:\Temp\DO417\payload**. Examinez le document texte et deux scripts PowerShell qu'Ansible a dézippés.



- 6.5. Cliquez avec le bouton droit de la souris sur le fichier **example.txt**, sélectionnez **Properties**, puis cliquez sur **Security**.



- 6.6. Examinez les permissions de fichier pour les différents utilisateurs et groupes. Notez que les groupes **SYSTEM** et **Administrators** disposent des permissions **Full control**.



Cliquez sur **Cancel** pour fermer la boîte de dialogue **example Properties**.

- 6.7. Fermez la connexion du bureau à distance **win1.example.com**.
  - ▶ 7. Modifiez le playbook **unzip.yml** pour copier les fichiers sur le Bureau, créez un raccourci pour le fichier texte et modifiez les permissions et la propriété.
- 7.1. Revenez dans **Visual Studio Code** et ouvrez le playbook **unzip.yml**.
  - 7.2. Ajoutez une nouvelle tâche, **Payload is copied to the Desktop**, afin de copier les fichiers du répertoire **C:\Temp\DO417\payload** dans le répertoire **C:\Users\devops\Desktop\example** à l'aide de RoboCopy.

```
- name: Payload is copied to the Desktop
win_robocopy:
  src: C:\Temp\DO417\payload
  dest: C:\Users\devops\Desktop\example
```

- 7.3. Ajoutez une tâche **Shortcut created** pour créer un raccourci sur le Bureau qui établit un lien vers le fichier texte dans le répertoire **example**.

```
- name: Shortcut created
win_shortcut:
  src: '%UserProfile%\Desktop\example\example.txt'
  dest: '%UserProfile%\Desktop\example-txt.lnk'
```

- 7.4. Ajoutez une tâche qui appelle le module **win\_acl** pour refuser l'accès en écriture au fichier texte **example.txt**.

```
- name: Example.txt write permissions are denied
  win_acl:
    path: C:\Users\devops\Desktop\example\example.txt
    user: devops
    rights: Write
    type: deny
    state: present
```

- 7.5. Utilisez le module **win\_acl** pour ajouter une règle de refus **ReadAndExecute**, qui refuse l'autorisation d'ouvrir et d'exécuter le script PowerShell **not-runnable.ps1**.

```
- name: Not-runnable.ps1 execution is denied
  win_acl:
    path: C:\Users\devops\Desktop\example\not-runnable.ps1
    user: devops
    rights: ReadAndExecute
    type: deny
    state: present
```

- 7.6. Enfin, ajoutez une tâche qui appelle le module **win\_owner** pour modifier le propriétaire du script PowerShell **runnable.ps1**.

```
- name: Runnable.ps1 is owned by SYSTEM
  win_owner:
    path: C:\Users\devops\Desktop\example\runnable.ps1
    user: SYSTEM
```

- 7.7. Examinez le fichier YAML terminé et vérifiez que votre playbook correspond aux tâches ci-dessous.

```
---
- name: Example archive deployed and permissions set
  hosts: win1.example.com
  tasks:
    - name: Temp directory created
      win_file:
        path: C:\Temp\DO417\payload
        state: directory

    - name: Zip archive is copied
      win_copy:
        src: files/example.zip
        dest: C:\Temp\DO417\example.zip

    - name: Zip archive is unzipped
      win_unzip:
        src: C:\Temp\DO417\example.zip
        dest: C:\Temp\DO417\payload
```

```
delete_archive: yes

- name: Payload is copied to the Desktop
  win_roboCopy:
    src: C:\Temp\DO417\payload
    dest: C:\Users\devops\Desktop\example

- name: Shortcut created
  win_shortcut:
    src: '%UserProfile%\Desktop\example\example.txt'
    dest: '%UserProfile%\Desktop\example-txt.lnk'

- name: Example.txt write permissions are denied
  win_acl:
    path: C:\Users\devops\Desktop\example\example.txt
    user: devops
    rights: Write
    type: deny
    state: present

- name: Not-runnable.ps1 execution is denied
  win_acl:
    path: C:\Users\devops\Desktop\example\not-runnable.ps1
    user: devops
    rights: ReadAndExecute
    type: deny
    state: present

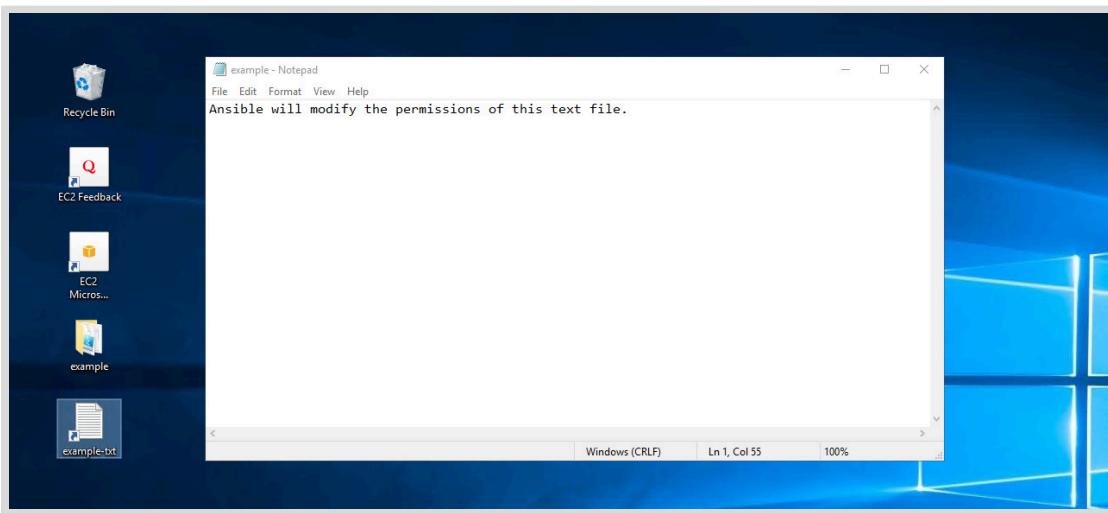
- name: Runnable.ps1 is owned by SYSTEM
  win_owner:
    path: C:\Users\devops\Desktop\example\runnable.ps1
    user: SYSTEM
```

- 8. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
- 8.1. Enregistrez le fichier.
  - 8.2. Accédez au volet **Source Control**, puis indexez les modifications.
  - 8.3. Saisissez un court message de validation, puis validez les modifications.
  - 8.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- 9. Retournez à Red Hat Ansible Tower et exécutez à nouveau votre playbook.
- 9.1. Dans le volet de navigation Ansible Tower, cliquez sur **Templates**.
  - 9.2. Cliquez sur l'icône de fusée dans la ligne du modèle de tâche **Run managing-files project** pour ré-exécuter la tâche avec le playbook mis à jour.

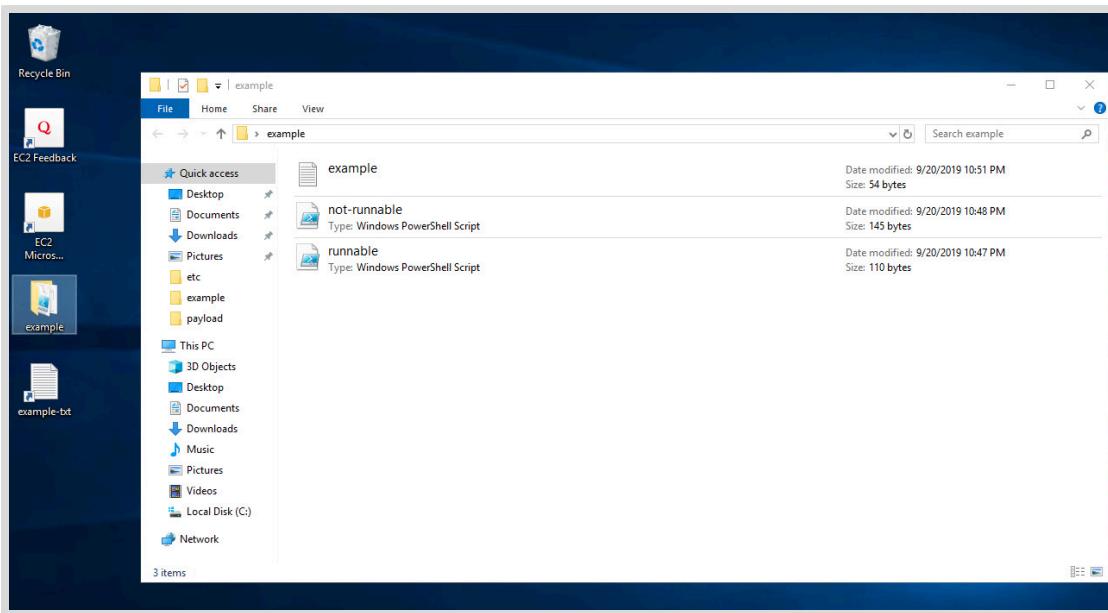
- 9.3. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**. Si la tâche ne s'est pas correctement déroulée, comparez votre travail avec le fichier de solution **solutions/unzip.sol** dans le référentiel Git **managing-files**.

- 10. Reconnectez-vous à l'hôte géré **win1.example.com** pour examiner les modifications apportées au fichier.
- 10.1. Cliquez sur **Recherche Windows**, recherchez **distance**, puis ouvrez Connexion Bureau à distance.
  - 10.2. Saisissez **win1.example.com** pour l'ordinateur, **devops** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Notez que l'utilisateur **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.
  - 10.3. À l'invite, cliquez sur **Yes** pour accepter le certificat non sécurisé utilisé dans votre environnement de formation.

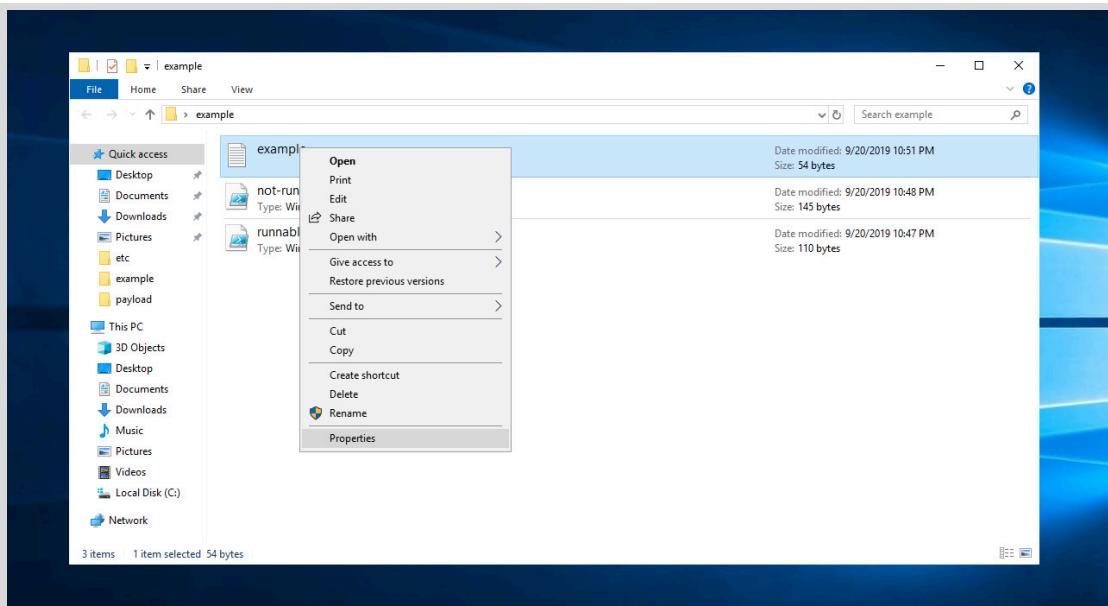
- 10.4. Double-cliquez sur le raccourci **example-txt** sur le Bureau pour ouvrir le fichier texte situé dans le dossier **example**. Fermez l'application Notepad après avoir vérifié que le fichier s'ouvre correctement.



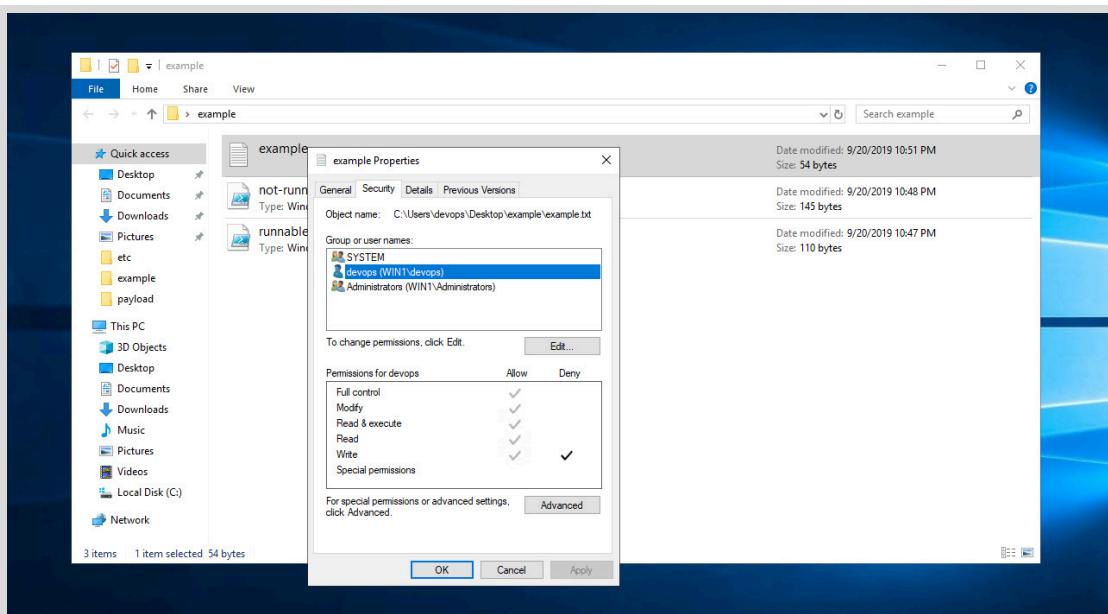
- 10.5. Ouvrez le dossier **example** sur le Bureau et observez que les fichiers de notre archive **example.zip** sont tous présents.



- 10.6. Cliquez avec le bouton droit sur le document **example.txt**, sélectionnez **Properties**, puis cliquez sur **Security** pour inspecter les permissions de fichiers.

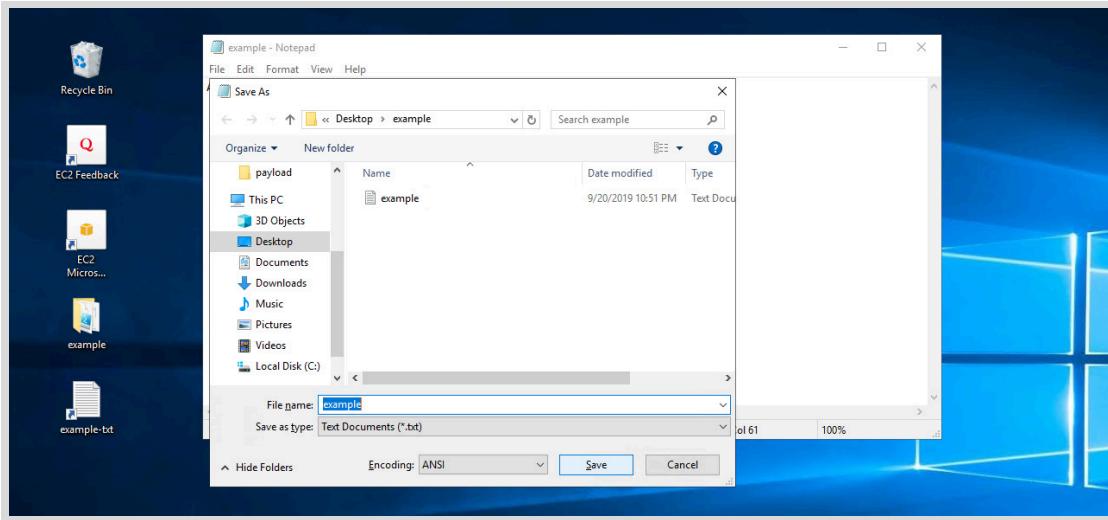


Sélectionnez l'utilisateur **devops** et notez que l'autorisation **Write** est refusée.

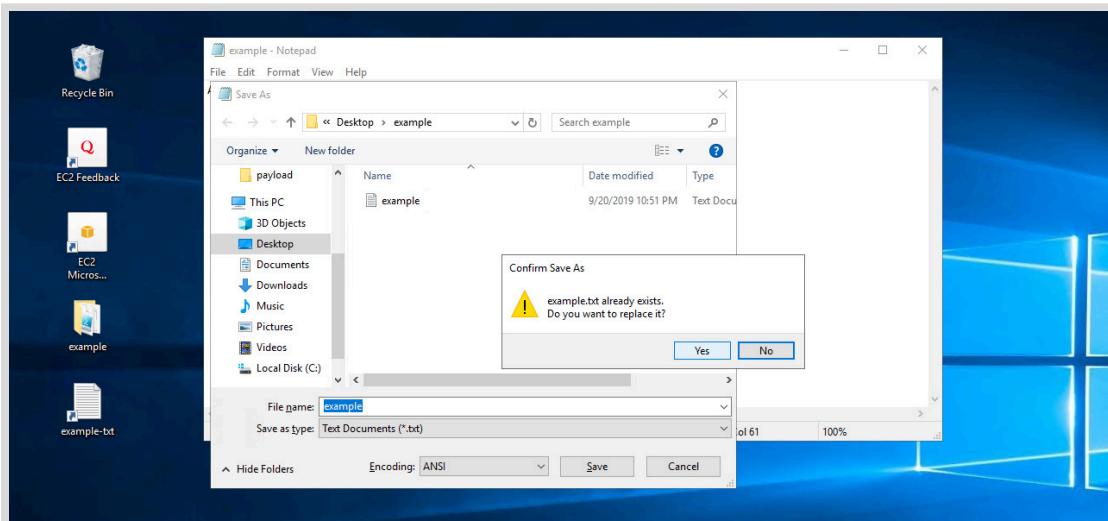


Cliquez sur **Cancel** pour fermer la boîte de dialogue **example Properties**.

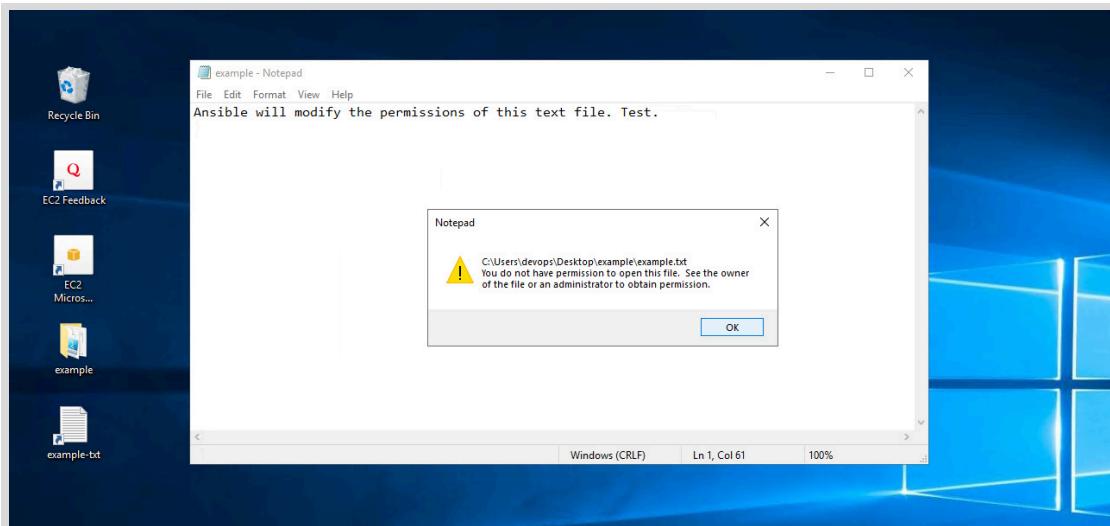
- 10.7. Double-cliquez sur le fichier **example.txt** pour l'ouvrir dans Notepad. Vérifiez que le fichier n'est pas accessible en écriture en essayant d'enregistrer le fichier. Dans Notepad, sélectionnez **File** → **Save**, puis cliquez sur **Save** pour enregistrer le fichier.



Cliquez sur **Yes** pour confirmer que vous avez l'intention de remplacer le fichier **example.txt**. Lorsque vous y êtes invité, cliquez sur **Yes**.



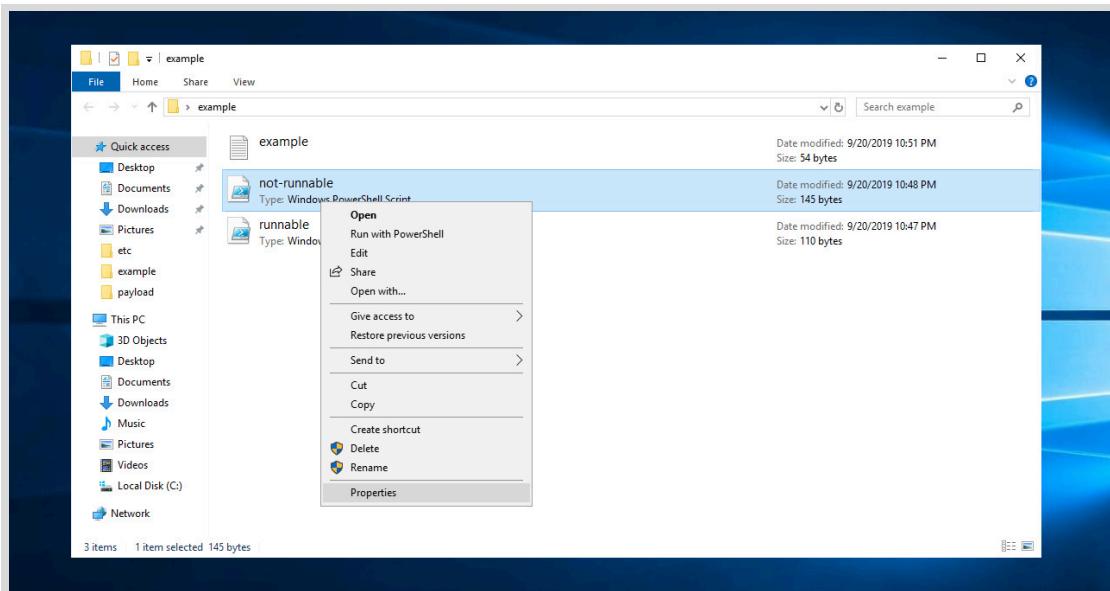
Une fenêtre affiche un message vous indiquant que vous n'êtes pas autorisé à écrire dans le fichier, ce qui correspond au comportement attendu.



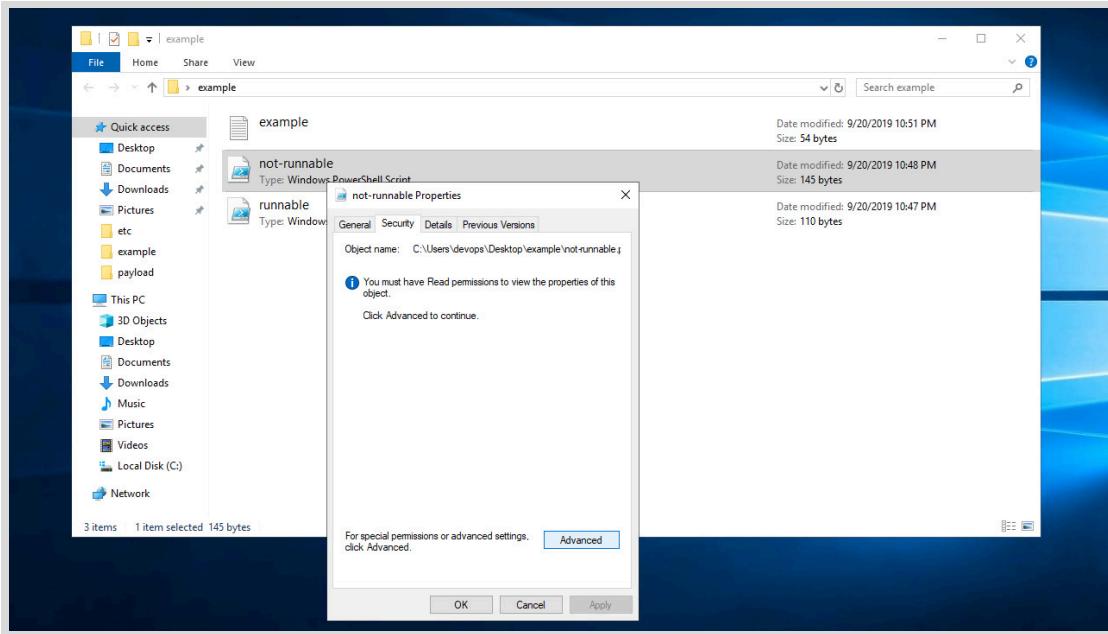
### Important

Ansible a demandé à l'hôte géré de refuser les permissions d'écriture sur le fichier **example.txt**. L'erreur indique que Windows a bloqué l'action d'écriture et que la tâche Ansible a réussi.

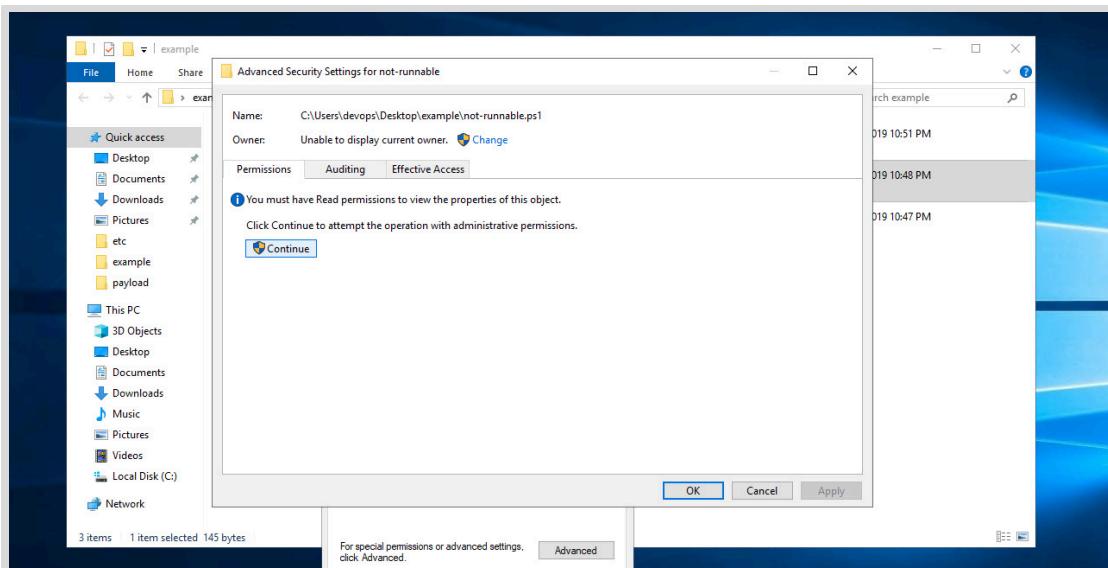
- 10.8. Inspectez les permissions de fichiers pour le script PowerShell **not-runnable.ps1**. Cliquez avec le bouton droit de la souris sur le fichier, sélectionnez **Properties**, puis cliquez sur **Security**.



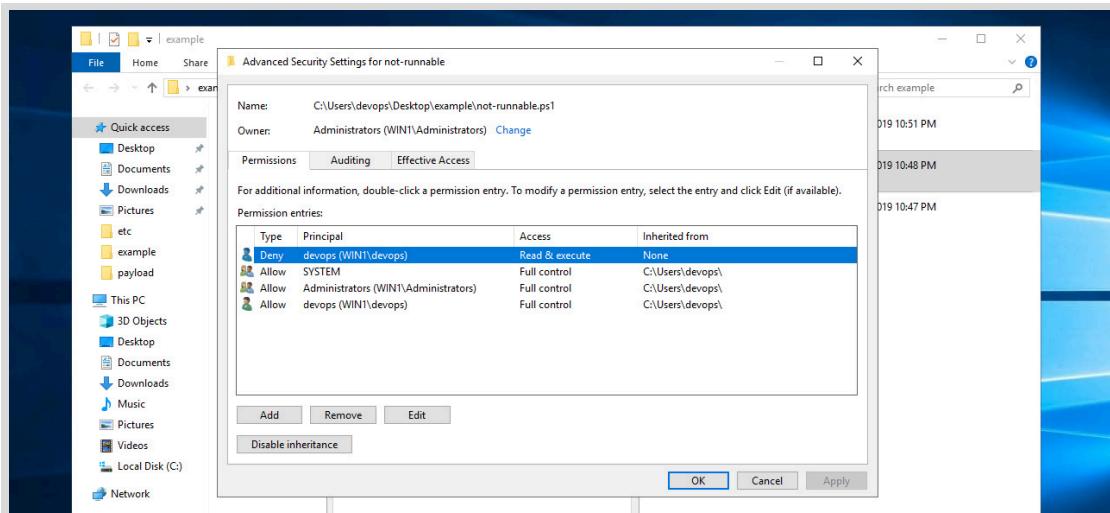
Notez l'avertissement d'information qui s'affiche : « You must have Read permissions to view the properties of this object ». Cet avertissement est attendu, car Ansible a refusé à l'utilisateur devops l'autorisation de lire et exécuter (**ReadAndExecute**) le fichier.



Cliquez sur **Advanced**, puis cliquez sur **Continue** pour afficher la liste des permissions appliquées.

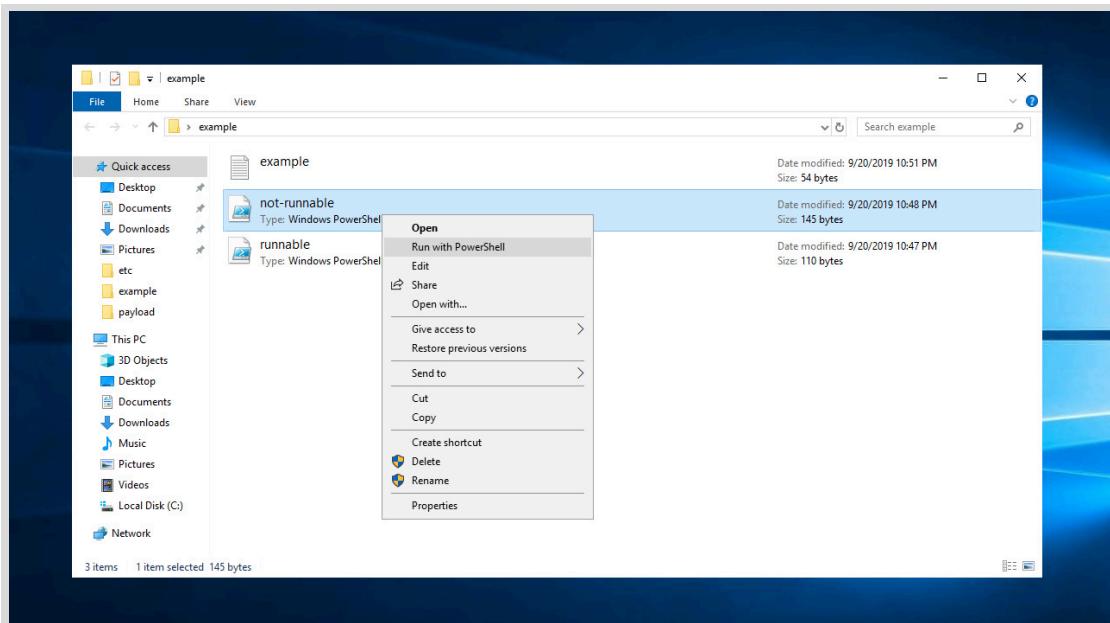


Notez que l'entrée d'autorisation pour l'utilisateur **devops** affiche une règle **deny** pour l'accès **Read & execute**.



Cliquez sur **Cancel** deux fois pour fermer la boîte de dialogue des propriétés.

- 10.9. Essayez d'exécuter le script PowerShell **not-runnable.ps1** pour vérifier que Windows refuse les demandes d'exécution de ce script. Cliquez avec le bouton droit de la souris sur le script **not-runnable.ps1**, puis sélectionnez **Run with PowerShell** dans le menu.



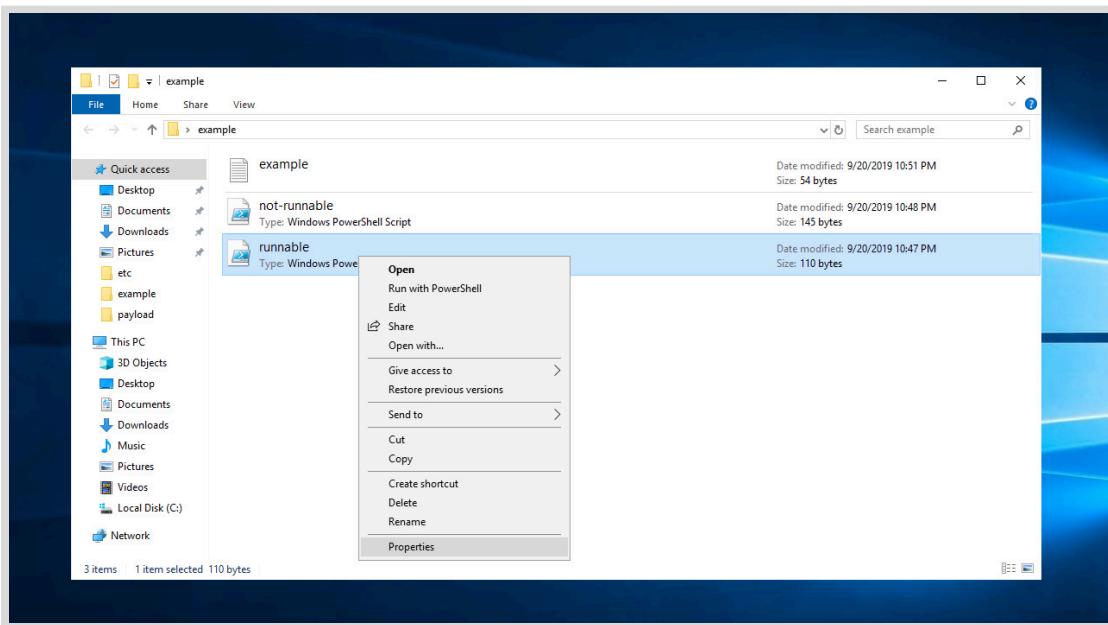
PowerShell démarre, puis se ferme sans action. Le comportement attendu est correct.



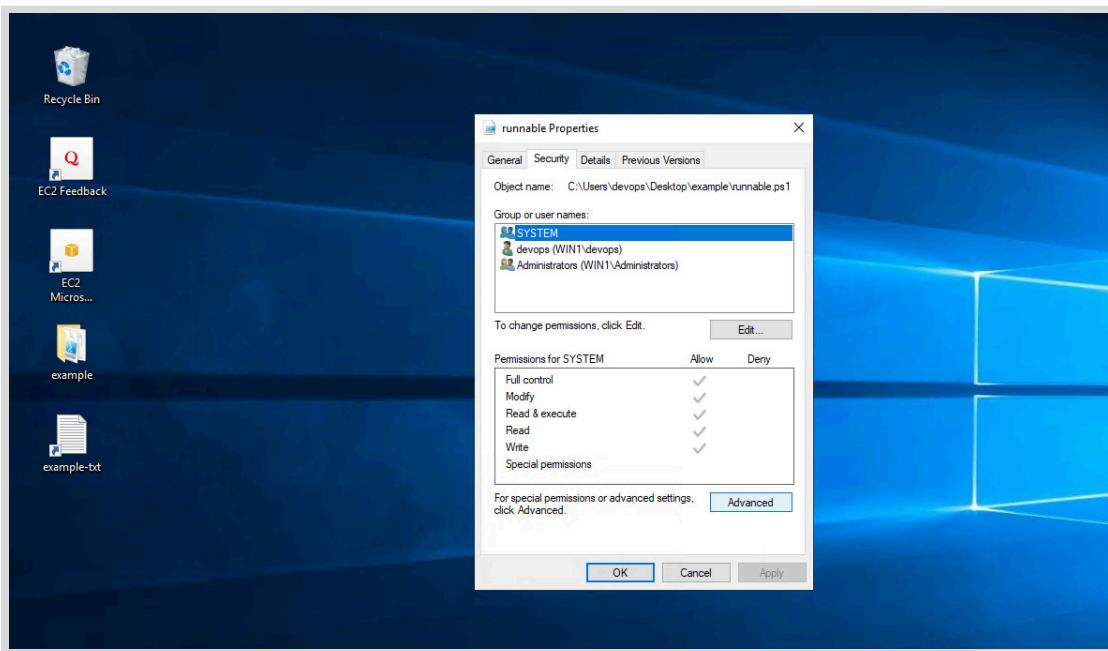
### Important

Ansible a demandé à l'hôte géré de refuser à l'utilisateur devops l'autorisation d'ouvrir et d'exécuter le script PowerShell **not-runnable.ps1**. Aucune action n'est attendue.

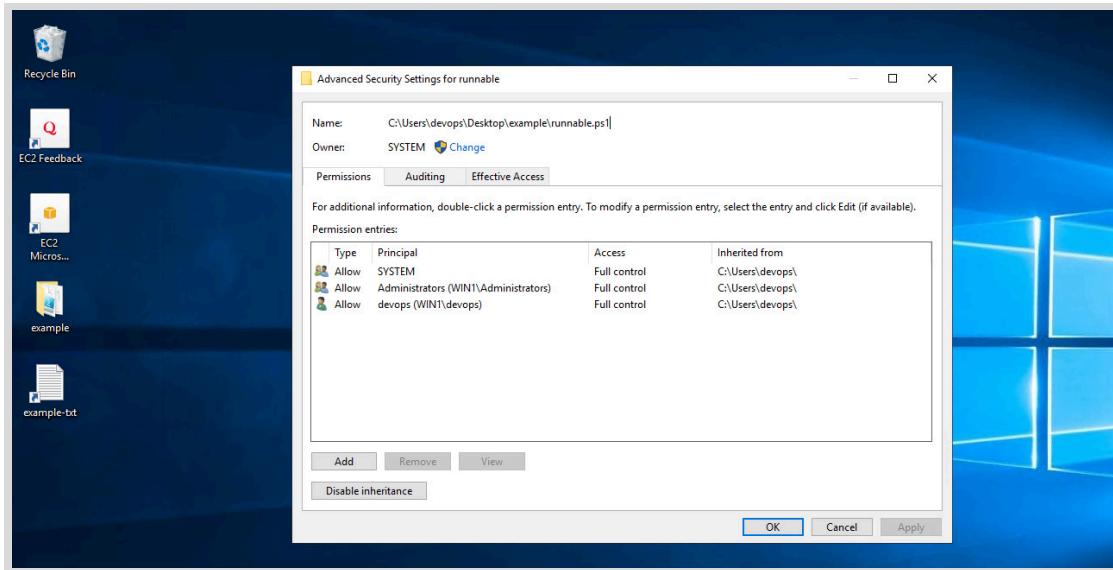
10.10. Validez la propriété de fichier du script PowerShell **runnable.ps1**. Cliquez avec le bouton droit de la souris sur le fichier, sélectionnez **Properties**, puis cliquez sur **Security**.



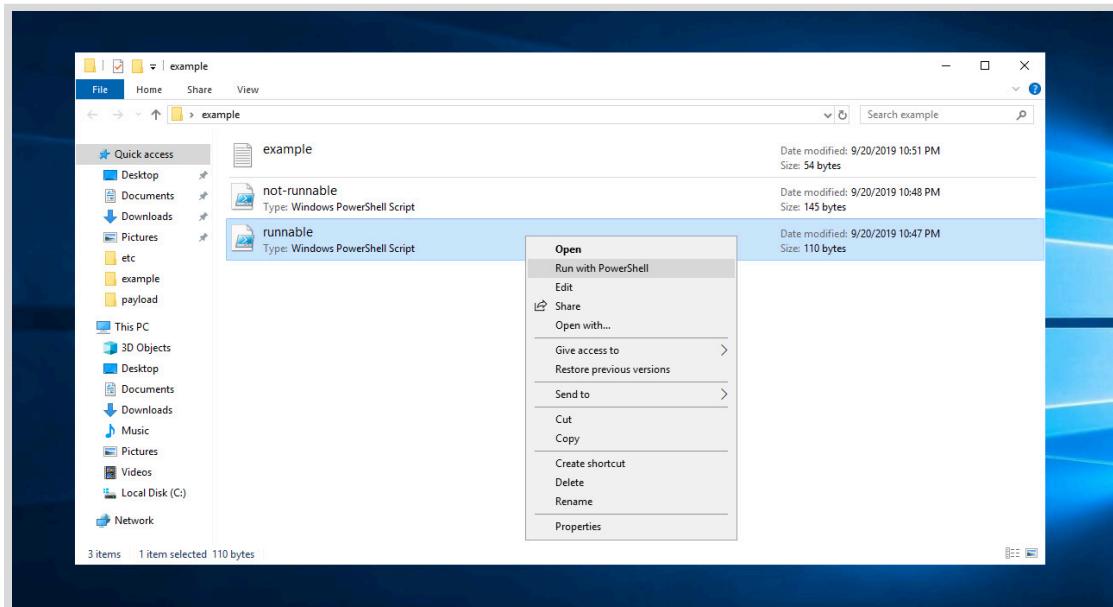
Cliquez sur le bouton **Advanced** pour afficher des informations plus détaillées sur la sécurité.



Notez que le propriétaire est le groupe d'utilisateurs **SYSTEM**.



- 10.11. Vérifiez que Windows exécute le script PowerShell **runnable.ps1**. Cliquez avec le bouton droit de la souris sur **runnable.ps1**, puis sélectionnez **Run with PowerShell** dans le menu.



Une nouvelle fenêtre annonce que l'exemple de script s'est exécuté avec succès.

- 10.12. Fermez la connexion du bureau à distance **win1.example.com**.
- 11. Nettoyez votre travail en exécutant le playbook **cleanup-unzip.yml** à l'aide du modèle de tâche **Run managing-files project**.
- 11.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 11.2. Cliquez sur le modèle de tâche **Run managing-files project** pour l'ouvrir.
  - 11.3. Sélectionnez le playbook **cleanup-unzip.yml** dans la liste **PLAYBOOK**.

- 11.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'exercice guidé est maintenant terminé.

# Création de modèles de fichiers à l'aide de Jinja2

---

## Résultats

Au terme de cette section, vous serez en mesure d'utiliser des modèles Jinja2 pour déployer des fichiers personnalisés sur des hôtes gérés.

## Création de modèles à partir de fichiers

Ansible dispose d'un certain nombre de modules qui peuvent être utilisés pour modifier des fichiers existants. Ceux-ci incluent `win_lineinfile` et `assemble`, entre autres. Cependant, ils ne sont pas toujours faciles à utiliser efficacement et correctement.

Un moyen beaucoup plus puissant de gérer les fichiers est de les *transformer en modèles*. Avec cette méthode, vous écrivez un fichier de configuration de modèle personnalisé automatiquement pour l'hôte géré lors du déploiement du fichier, en utilisant des variables et des faits Ansible. Cette approche est plus facile à gérer et moins sujette aux erreurs que la création manuelle de chaque fichier.

## Présentation de Jinja2

Ansible utilise le système de création de modèles Jinja2 pour les fichiers modèles. Ansible utilise également la syntaxe Jinja2 pour référencer les variables dans les playbooks, vous savez donc déjà un peu comment l'utiliser.

## Utilisation de délimiteurs

Les variables et les expressions logiques sont placées entre balises, ou délimiteurs. Par exemple, les modèles Jinja2 utilisent `{% EXPR %}` pour les expressions ou la logique (des boucles, par exemple), et ils utilisent `{{ EXPR }}` pour générer les résultats d'une expression ou d'une variable pour l'utilisateur final. Cette dernière balise, une fois le rendu effectué, est remplacée par une ou plusieurs valeurs et visualisée par l'utilisateur final. Utilisez la syntaxe `{# COMMENT #}` pour inclure des commentaires qui ne doivent pas apparaître dans le fichier final.

La première ligne de l'exemple suivant contient un commentaire qui est exclu du fichier final. Les valeurs des faits système référencés remplacent les références de variable dans la seconde ligne.

```
{# Generate a %SystemRoot%\System32\drivers\etc\hosts line #}
{{ ansible_facts['default_ipv4']['address'] }}    {{ ansible_facts['hostname'] }}
```

## Génération d'un modèle Jinja2

Un modèle Jinja2 se compose de plusieurs éléments : des données, des variables et des expressions. Les valeurs remplacent ces variables et expressions lors du rendu du modèle Jinja2. Les variables utilisées dans le modèle sont spécifiées dans la section `vars` du playbook. Les faits des hôtes gérés peuvent également être utilisés en tant que variables sur un modèle.

**Note**

Rappelez-vous que la sortie du module **setup** facilite l'examen des faits associés aux hôtes gérés, car ce module exécute et recueille les faits par défaut. Pour plus d'informations, consultez Gestion des faits.

L'exemple suivant illustre la création d'un modèle pour un fichier HTML avec des variables et des faits récupérés par Ansible auprès d'hôtes gérés. Lors de l'exécution du playbook associé, les faits sont remplacés par leurs valeurs dans l'hôte géré en cours de configuration.

**Note**

Aucune extension de fichier spécifique n'est nécessaire pour un fichier qui contient un modèle Jinja2 (.j2, par exemple). Cependant, fournir une telle extension de fichier peut vous aider à reconnaître plus facilement le fichier modèle.

```
<!doctype html>
<!-- {{ ansible_managed }} -->

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Info</title>
</head>
<body>
  <h1>{{ ansible_facts['hostname'] }}</h1>

  <ul>
    <li>IP: {{ ansible_facts['ip_addresses'][0] }}</li>
    <li>OS: {{ ansible_facts['os_name'] }}</li>
    <li>Data Center: {{ data_center }}</li>
  </ul>

</body>
</html>
```

## Déploiement de modèles Jinja2

Les modèles Jinja2 constituent un puissant outil pour la personnalisation de fichiers de configuration pour le déploiement sur les hôtes gérés. Après la création du modèle Jinja2 pour un fichier de configuration, déployez-le sur les hôtes gérés à l'aide du module **win\_template**, lequel prend en charge le transfert d'un fichier local du noeud de contrôle vers les hôtes gérés.

Utilisez la syntaxe suivante lorsque vous travaillez avec le module **win\_template**. La valeur associée à la clé **src** spécifie le modèle Jinja2 source ; dans cet exemple, la source est le répertoire **templates** relatif au playbook. La valeur associée à la clé **dest** spécifie le fichier à créer sur les hôtes de destination.

```
tasks:
  - name: template render
    win_template:
      src: templates/info.j2
      dest: C:\Inetpub\wwwroot\info.html
```

**Note**

Par défaut, le module **win\_template** s'assure automatiquement que la sortie du modèle traité respecte la séquence de sauts de ligne pour le retour chariot standard de Windows. Ansible convertit les fichiers de modèle Linux ou MacOS avec des séquences de sauts de ligne **\n** en fins de lignes Windows **\r\n**.

Pour conserver les nouvelles lignes Linux, définissez le paramètre **newline\_sequence win\_template** sur **\n**.

## Gestion de fichiers transformés en modèles

Pour empêcher les administrateurs système de modifier les fichiers déployés par Ansible, il est recommandé d'inclure un commentaire en haut du modèle pour indiquer que le fichier ne doit pas être modifié manuellement.

Une façon de procéder consiste à utiliser la chaîne « Ansible managed » définie dans la directive **ansible\_managed**. Il ne s'agit pas d'une variable normale, mais elle peut être utilisée comme telle dans un modèle. Définissez la directive **ansible\_managed** dans le fichier **ansible.cfg** :

```
ansible_managed = Ansible managed
```

Utilisez la syntaxe suivante pour inclure la chaîne **ansible\_managed** dans un modèle Jinja2 :

```
{{ ansible_managed }}
```

## Description des structures de contrôle

Utilisez les structures de contrôle Jinja2 dans les fichiers de modèle pour réduire les saisies répétitives, pour créer dynamiquement les entrées de chaque hôte dans un play ou pour insérer du texte de manière conditionnelle dans un fichier.

### Fournir un contrôle conditionnel

Jinja2 utilise l'instruction **if** pour fournir un contrôle conditionnel. Cela vous permet de mettre une ligne dans un fichier déployé, mais uniquement quand certaines conditions sont remplies.

Dans l'exemple suivant, la valeur de **result** variable est placée dans le fichier déployé uniquement si la valeur de **finished** variable est **Vrai** .

```
{% if finished %}
{{ result }}
{% endif %}
```

## Itération des listes avec des boucles

Jinja2 utilise l'instruction **for** pour fournir la fonctionnalité de bouclage. Dans l'exemple suivant, la variable **user** est remplacée par toutes les valeurs incluses dans la variable **users**, une valeur par ligne.

```
{% for user in users %}
    {{ user }}
{% endfor %}
```

L'exemple de modèle suivant utilise une instruction **for** pour parcourir toutes les valeurs de la variable **users**, en remplaçant **myuser** par chaque valeur, sauf lorsque la valeur est **Administrator**.

```
{# for statement #}
{% for myuser in users if not myuser == "Administrator" %}
User number {{ loop.index }} - {{ myuser }}
{% endfor %}
```

La variable **loop.index** s'étend jusqu'au numéro d'index sur lequel la boucle est actuellement en cours d'exécution. Sa valeur est égale à 1 lors de la première exécution de la boucle et augmente d'une unité à chaque itération.

Autre exemple : ce modèle utilise également une instruction **for** et part du principe qu'une variable **myhosts** est définie dans le fichier d'inventaire actuel. Cette variable contient une liste d'hôtes à gérer. Avec l'instruction **for** suivante, tous les hôtes du groupe **myhosts** de l'inventaire sont listés dans le fichier.

```
{% for myhost in groups['myhosts'] %}
{{ myhost }}
{% endfor %}
```

Si vous souhaitez un autre exemple, plus pratique, vous pouvez utiliser cette approche pour générer dynamiquement un fichier **C:\Windows\System32\drivers\etc\hosts** à partir de faits d'hôte. Supposons que vous ayez le playbook suivant :

```
- name: hosts file is up to date
hosts: all
gather_facts: yes
tasks:
  - name: Hosts file is updated
    win_template:
      src: templates/hosts.j2
      dest: C:\Windows\System32\drivers\etc\hosts
```

Le modèle **templates/hosts.j2** à trois lignes suivant construit le fichier à partir de tous les hôtes du groupe **all**. Dans cet exemple, la ligne du milieu dans le modèle est extrêmement longue, en raison de la longueur des noms de variables. Ansible itère sur chaque hôte du groupe pour obtenir trois faits pour le fichier **C:\Windows\System32\drivers\etc\hosts**.

```
{% for host in groups['all'] %}
{{ hostvars['host']['ansible_facts']['ip_addresses'][0] }} {{ hostvars['host']
['ansible_facts']['fqdn'] }} {{ hostvars['host']['ansible_facts']['hostname'] }}
{% endfor %}
```

**Important**

Vous pouvez utiliser des boucles et des conditions Jinja2 dans les modèles Ansible, mais pas dans les playbooks Ansible.

## Transformation de formats à l'aide de filtres

Jinja2 fournit des filtres qui modifient le format en sortie pour les expressions de modèle. Des filtres sont disponibles pour des langages tels que YAML et JSON. Le filtre **to\_json** forme la sortie de l'expression à l'aide de JSON. Le filtre **to\_yaml** forme la sortie de l'expression à l'aide de YAML.

```
{{ output | to_json }}
{{ output | to_yaml }}
```

D'autres filtres, tels que **to\_nice\_json** et **to\_nice\_yaml**, convertissent une sortie lisible par des humains au format JSON ou YAML.

```
{{ output | to_nice_json }}
{{ output | to_nice_yaml }}
```

Les filtres **from\_json** et **from\_yaml** attendent tous deux des chaînes au format JSON ou YAML, respectivement, pour les analyser.

```
{{ output | from_json }}
{{ output | from_yaml }}
```

**Références**

**win\_template - Templates a file out to a remote server &mdash; Ansible Documentation**

[https://docs.ansible.com/ansible/latest/modules/win\\_template\\_module.html](https://docs.ansible.com/ansible/latest/modules/win_template_module.html)

**Variables &mdash; Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_variables.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html)

**Filtres &mdash; Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_filters.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html)

## ► Exercice guidé

# Création de modèles de fichiers à l'aide de Jinja2

Dans cet exercice, vous allez créer un fichier de modèle simple à déployer sur chacun de vos hôtes gérés, et remplir le contenu personnalisé pour chaque hôte.

## Résultats

Vous serez en mesure d'utiliser des modèles Jinja2 pour déployer du contenu dynamique sur des hôtes gérés.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Red Hat Ansible Tower et GitLab à partir du poste de travail Windows.

Assurez-vous que l'hôte **win2.example.com** est listé dans **Default inventory**.

- ▶ 1. Ouvrez l'éditeur Visual Studio Code et clonez le référentiel **managing-files** sur votre **workstation**.
  - 1.1. À partir de **workstation**, double-cliquez sur l'icône de Bureau de l'éditeur Visual Studio Code. Vous pouvez aussi cliquer sur **Recherche Windows**, rechercher **code**, puis ouvrir Visual Studio Code.
  - 1.2. Pour ouvrir la palette de commandes, accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P**.  
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.
  - 1.3. À l'invite, fournissez l'URL de référentiel, <https://gitlab.example.com/student/managing-files.git>, puis sélectionnez le dossier **Documents** dans le répertoire **home/student** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **managing-files** sur l'instance **workstation**.
  - 1.4. Lorsque vous êtes invité à ouvrir le projet, cliquez sur **Open**.

- ▶ 2. Ouvrez le fichier de modèle Jinja2 **hosts.j2** dans le dossier **templates**.

Windows utilise le fichier d'hôtes pour mapper les noms d'hôtes sur les adresses IP. Le fichier de modèle **hosts.j2** indique à Windows de résoudre **www1.example.com** et **www2.example.com** sur l'adresse IP de localhost **127.0.0.1**.

De plus, notez les commentaires **Hosts** et **Update At** en haut du fichier. Les commentaires du fichiers d'hôtes sont précédés du symbole #.

- 3. Remplacez le texte **CHANGE\_ME** sur la première ligne par une expression de modèle qui correspond au nom d'hôte de l'hôte géré.

```
# Host: {{ ansible_facts['hostname'] }}
```

- 4. Remplacez le texte **CHANGE\_ME** sur la première ligne par une expression de modèle qui correspond à la date et l'heure actuelles.

```
# Updated At: {{ now() }}
```

- 5. Mettez à jour le modèle **hosts.j2** pour résoudre les noms d'hôte personnalisés en adresses IP des hôtes gérés **win1** et **win2**. Remplacez **127.0.0.1** sur chaque ligne par une expression de modèle placée entre accolades doubles, comme suit :

```
{{ hostvars['win1.example.com'].ansible_facts.ip_addresses[0] }} www1.example.com
{{ hostvars['win2.example.com'].ansible_facts.ip_addresses[0] }} www2.example.com
```

Ansible stocke les faits collectés pour chaque hôte géré dans le dictionnaire **hostvars['host'].ansible\_facts**.



#### Note

Le nom de ce dictionnaire peut également utiliser la notation de crochets partout, par exemple :

```
hostvars['host']['ansible_facts']
```

Cependant, pour éviter les retours à la ligne dans le texte de l'exemple, cette syntaxe n'est pas utilisée au cours de cet exercice.

- 6. Vérifiez que le fichier final **hosts.j2** correspond au modèle Jinja2 suivant, puis sélectionnez **File → Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.

```
# Host: {{ ansible_facts['hostname'] }}
# Updated At: {{ now() }}

{{ hostvars['win1.example.com'].ansible_facts.ip_addresses[0] }} www1.example.com
{{ hostvars['win2.example.com'].ansible_facts.ip_addresses[0] }} www2.example.com
```

- 7. Ouvrez le playbook **hosts.yml**. Ajoutez des tâches pour mettre à jour le contenu du serveur Web IIS, puis déployez le modèle **templates/hosts.j2** sur l'hôte géré Windows.

```
---
- name: Web server with custom hosts file deployed
  hosts: all
  tasks:
    - name: IIS Web Server started
      win_feature:
```

```
name: Web-Server
state: present

- name: Index file is created ①
  win_copy:
    content: "Hello from {{ ansible_hostname }}!"
    dest: C:\Inetpub\wwwroot\index.html

- name: Hosts file template deployed ②
  win_template:
    src: templates/hosts.j2 ③
    dest: C:\Windows\System32\drivers\etc\hosts ④
    backup: yes ⑤
```

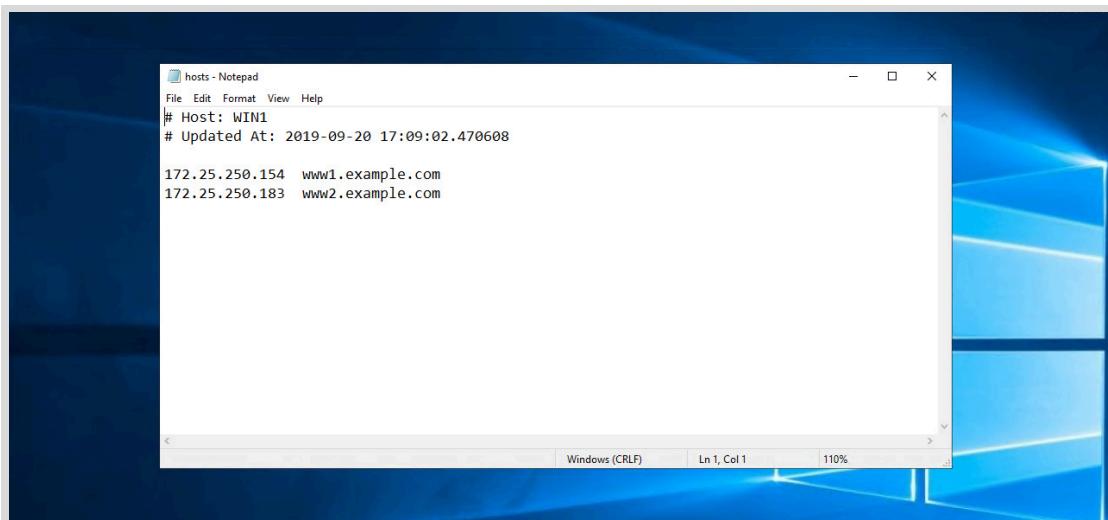
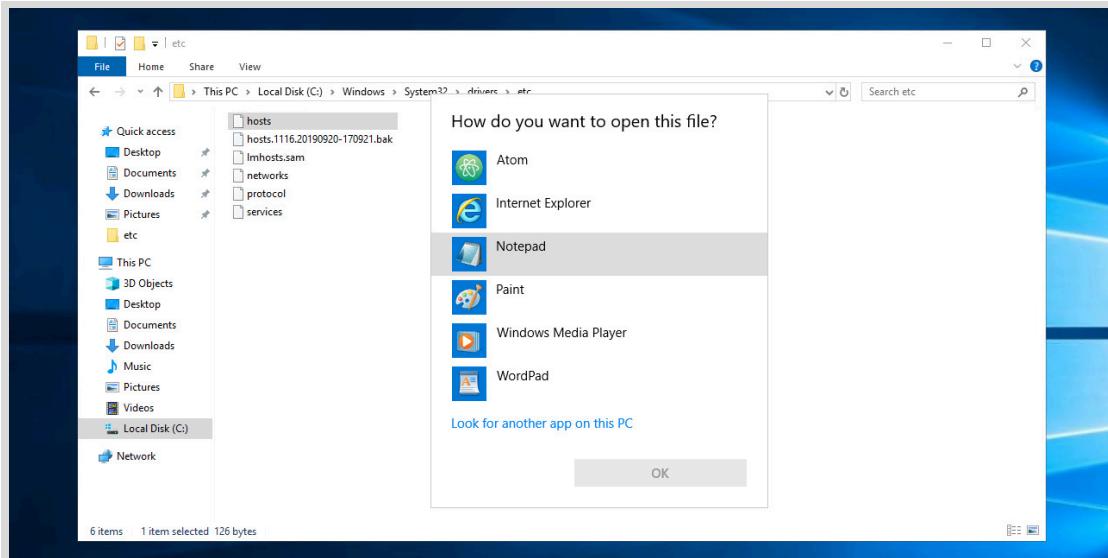
- ① Tâche qui invoque le module **win\_copy** pour créer un fichier **index.html** et écrire un message spécifique à l'hôte.
- ② Tâche qui invoque le module **win\_template** pour traiter le modèle Jinja2 et écrire la sortie dans un fichier.
- ③ Déclare l'emplacement du modèle à l'aide du paramètre **src**.
- ④ Ajoute un paramètre **dest** pour spécifier une destination sur les hôtes Windows gérés. Étant donné que la destination est un hôte géré Windows, utilisez un chemin avec les barres obliques inverses appropriées et tous les caractères d'échappement requis.
- ⑤ Ajoutez le paramètre **backup** pour enregistrer une sauvegarde horodatée du fichier.

- ▶ 8. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 8.1. Pour enregistrer le fichier, cliquez sur **File → Save** ou appuyez sur **Ctrl+S**.
  - 8.2. Accédez au volet **Source Control**, puis cliquez sur **+** pour indexer les modifications à la fois sur **hosts.yml** et **hosts.j2**.
  - 8.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 8.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- ▶ 9. À partir de **workstation**, accédez à votre instance Red Hat Ansible Tower sur <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- ▶ 10. Utilisez le modèle de tâche **Run managing-files project** pour exécuter votre playbook.
  - 10.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 10.2. Cliquez sur le modèle de tâche **Run managing-files project**.
  - 10.3. Sélectionnez le playbook **hosts.yml** dans la liste **PLAYBOOK**.

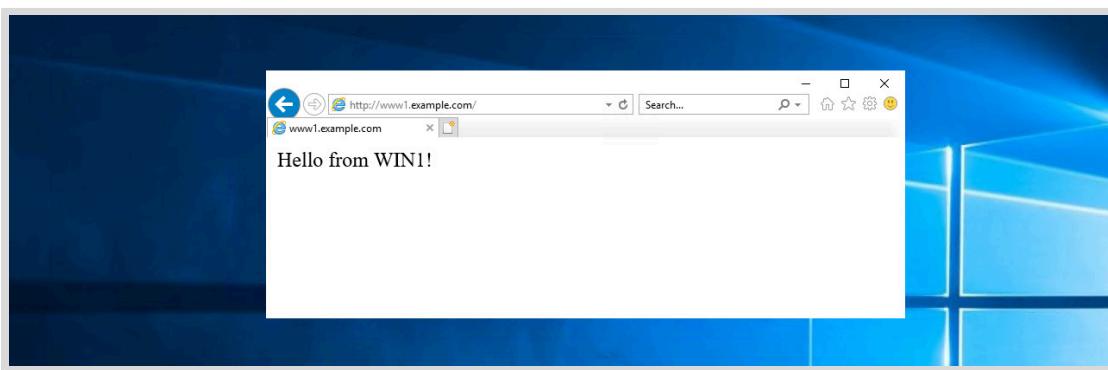
The screenshot shows the configuration for a new Ansible job template named "Run managing-files project". The "DETAILS" tab is active. The configuration includes:

- NAME:** Run managing-files project
- DESCRIPTION:** Use this job template to run your playbooks
- JOB TYPE:** Run
- INVENTORY:** Default inventory
- PROJECT:** managing-files repository
- PLAYBOOK:** hosts.yml
- CREDENTIAL:** DevOps
- FORKS:** 0
- LIMIT:** (empty)
- VERBOSITY:** 0 (Normal)
- JOB TAGS:** (empty)
- SKIP TAGS:** (empty)
- INSTANCE GROUPS:** (empty)
- JOB SLICING:** 1

- 10.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
- 11. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**. Si la tâche ne s'est pas correctement déroulée, comparez votre travail avec les fichiers de solution **solutions/hosts.sol** et **solutions/templates/hosts.j2.sol** dans le référentiel Git **managing-files**.
- 12. Connectez-vous à l'hôte géré **win1.example.com** pour vérifier le déploiement du fichier d'hôtes.
- 12.1. Cliquez sur **Recherche Windows**, recherchez **distance**, puis ouvrez Connexion Bureau à distance.
  - 12.2. Saisissez **win1.example.com** pour l'ordinateur, **devops** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Notez que **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.
  - 12.3. Lorsque vous y êtes invité, cliquez sur **Yes** pour accepter le certificat non sécurisé utilisés votre environnement de formation.
  - 12.4. Ouvrez **File Explorer** et accédez au document de texte **C:\Windows\System32\drivers\etc\hosts**. Cliquez avec le bouton droit de la souris sur le fichier pour l'ouvrir, puis sélectionnez Notepad lorsque vous êtes invité à choisir parmi une liste d'applications.



12.5. Ouvrez **Internet Explorer** et accédez à **www1.example.com**.



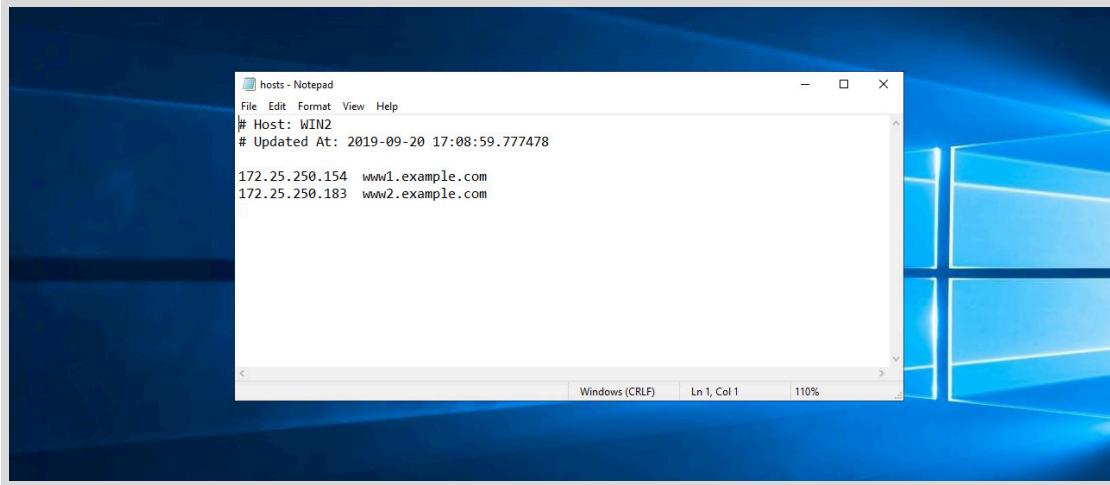
12.6. Accédez à **www2.example.com** et vérifiez que le domaine est correctement résolu sur l'hôte **win2**.



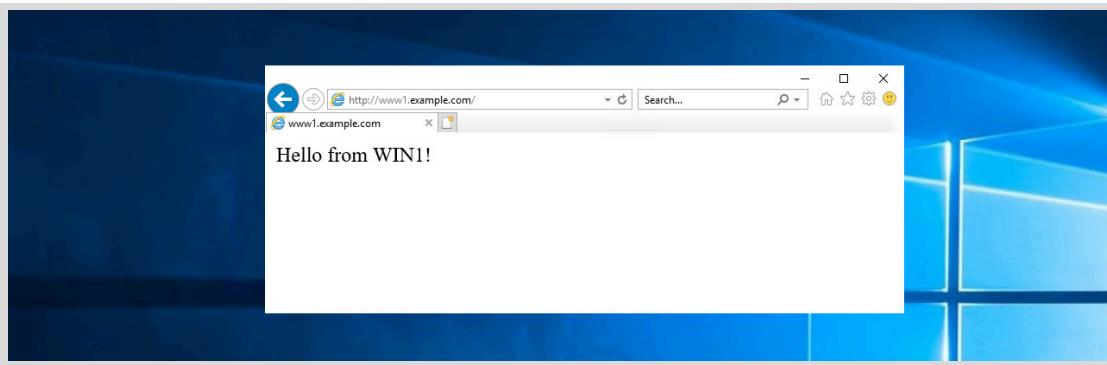
12.7. Fermez la connexion du bureau à distance **win1.example.com**.

- 13. Connectez-vous à l'hôte géré **win2.example.com**, puis vérifiez le déploiement du fichier d'hôtes sur l'hôte géré **win2**.

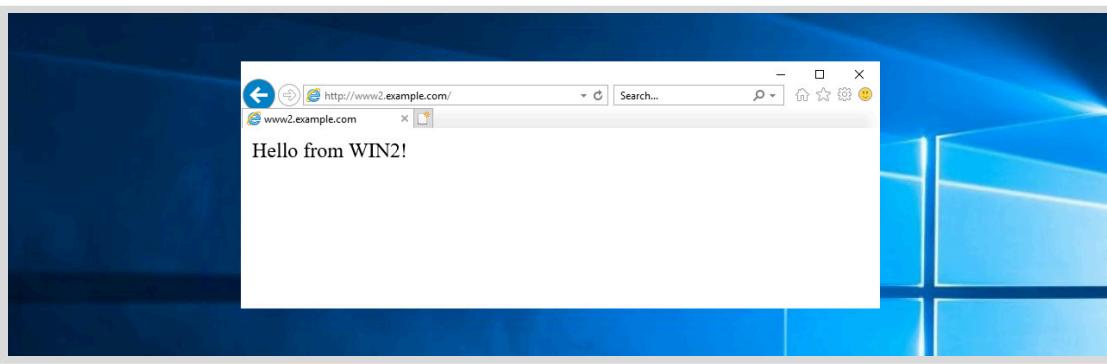
- 13.1. Cliquez sur **Recherche Windows**, recherchez **distance**, puis ouvrez Connexion Bureau à distance.
- 13.2. Saisissez **win2.example.com** pour l'ordinateur, **devops** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Notez que **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.
- 13.3. Lorsque vous y êtes invité, cliquez sur **Yes** pour accepter le certificat non sécurisé utilisés votre environnement de formation.
- 13.4. Ouvrez **File Explorer**, puis accédez au document de texte **C:\Windows\System32\drivers\etc\hosts**. Cliquez avec le bouton droit de la souris sur le fichier pour l'ouvrir, puis sélectionnez Notepad lorsque vous êtes invité à choisir parmi une liste d'applications.



13.5. Ouvrez **Internet Explorer** et accédez à **www1.example.com**.



- 13.6. Accédez à **www2.example.com** et vérifiez que le domaine est correctement résolu sur l'hôte **win2**.



- 13.7. Fermez la connexion du bureau à distance **win2.example.com**.

► 14. Exécutez le playbook **cleanup-hosts.yml** à partir du modèle de tâche **Run managing-files project** pour effacer votre travail.

- 14.1. Dans le volet de navigation, cliquez sur **Templates**.
- 14.2. Cliquez sur le modèle de tâche **Run managing-files project**.
- 14.3. Sélectionnez le playbook **cleanup-hosts.yml** dans la liste **PLAYBOOK**.

Run managing-files project

DETAILS		PERMISSIONS	NOTIFICATIONS	COMPLETED JOBS	SCHEDULES	ADD SURVEY
* NAME		DESCRIPTION		* JOB TYPE		PROMPT ON LAUNCH
Run managing-files project		Use this job template to run your playbooks		Run		<input type="checkbox"/>
* INVENTORY		* PROJECT		* PLAYBOOK		PROMPT ON LAUNCH
Default inventory		managing-files repository		cleanup-hosts.yml		<input type="checkbox"/>
CREDENTIAL		FORKS		LIMIT		PROMPT ON LAUNCH
DevOps		0				<input type="checkbox"/>
* VERBOSITY		JOB TAGS		SKIP TAGS		PROMPT ON LAUNCH
0 (Normal)						<input type="checkbox"/>
LABELS		INSTANCE GROUPS		JOB SLICING		
				1		

14.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Déploiement de fichiers sur des hôtes gérés

## Liste de contrôle des performances

Au cours de cet atelier, vous allez écrire un playbook qui crée un fichier sur vos hôtes gérés, qui est personnalisé à l'aide d'un modèle Jinja2 et configuré avec les permissions de fichier appropriées.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Utiliser des modèles Jinja2 pour déployer du contenu dynamique sur des hôtes gérés.
- Modifier les permissions de fichier pour un utilisateur ou un groupe spécifié.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. À partir du **poste de travail** Windows, ouvrez l'éditeur Visual Studio Code et clonez le référentiel **managing-files**, <https://gitlab.example.com/student/managing-files.git>, vers **C:\Users\student\Documents\managing-files**, si le dossier n'y figure pas déjà.
2. Mettez à jour le modèle Jinja2 **templates/index.html.j2** pour afficher des informations sur l'hôte géré en procédant comme suit :
  - Le commentaire HTML situé en haut du fichier insère la variable **ansible\_managed**.
  - L'élément **h1** affiche le nom d'hôte.
  - La balise de définition **OS** affiche le fait **os\_name**.
  - La balise de définition **Memory** affiche le fait **memtotal\_mb**.
  - La balise de définition **Processor vCPU** affiche le fait **processor\_vcpus**.
  - La balise de définition **Processor Cores** affiche le fait **processor\_cores**.
3. Définissez des tâches dans le fichier playbook **review.yml** sur :
  - Assurez-vous que la fonctionnalité de serveur Web IIS est présente.

- Déployez le modèle Jinja2 **templates/index.html.j2** sur **C:\Inetpub\wwwroot\index.html** sur l'hôte géré.
- Refusez à **Users** sur les hôtes gérés l'accès en écriture (**Write**) au fichier **index.html**.

Enregistrez le playbook **review.yml**.

4. Validez et transmettez par push vos modifications au référentiel **managing-files**.
5. À partir de **workstation**, accédez à votre instance Red Hat Ansible Tower sur **http://tower.example.com**. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.  
Ajoutez l'hôte **win2.example.com** à l'inventaire **Default inventory** dans Ansible Tower si cela n'a pas déjà été fait lors d'un exercice précédent.
6. Lancez le playbook **review.yml** à l'aide du modèle de tâche **Run managing-files project**.
7. Vérifiez que la tâche **Run managing-files project** est correctement exécutée.  
Accédez à **http://win1.example.com** et **http://win2.example.com** pour vérifier que la page Web est correctement rendue et fournie.
8. Connectez-vous à l'hôte géré **win1.example.com** pour vérifier que l'autorisation d'écriture sur **C:\Inetpub\wwwroot\index.html** est refusée.
9. Lancez le playbook **cleanup-review.yml** à l'aide du modèle de tâche **Run managing-files project**.

L'atelier est maintenant terminé.

## ► Solution

# Déploiement de fichiers sur des hôtes gérés

### Liste de contrôle des performances

Au cours de cet atelier, vous allez écrire un playbook qui crée un fichier sur vos hôtes gérés, qui est personnalisé à l'aide d'un modèle Jinja2 et configuré avec les permissions de fichier appropriées.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Utiliser des modèles Jinja2 pour déployer du contenu dynamique sur des hôtes gérés.
- Modifier les permissions de fichier pour un utilisateur ou un groupe spécifié.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide de **training** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. À partir du **poste de travail** Windows, ouvrez l'éditeur Visual Studio Code et clonez le référentiel **managing-files**, <https://gitlab.example.com/student/managing-files.git>, vers **C:\Users\student\Documents\managing-files**, si le dossier n'y figure pas déjà.



#### Note

Les instructions suivantes décrivent la façon de cloner le référentiel **managing-files**. Si vous avez cloné le référentiel lors d'un exercice précédent, vous pouvez ignorer cette étape.

- 1.1. À partir du **poste de travail** Windows, double-cliquez sur l'icône de Bureau pour ouvrir l'éditeur Visual Studio Code.
- 1.2. Pour cloner le référentiel **updates**, accédez à **View** → **Command Palette** ou appuyez sur **Ctrl+Shift+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
À l'invite, fournissez l'URL de référentiel <https://gitlab.example.com/student/managing-files.git>, puis sélectionnez le dossier **Documents** dans le répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **managing-files** sur l'instance **workstation**.

Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage.

2. Mettez à jour le modèle Jinja2 **templates/index.html.j2** pour afficher des informations sur l'hôte géré en procédant comme suit :

- Le commentaire HTML situé en haut du fichier insère la variable **ansible\_managed**.
- L'élément **h1** affiche le nom d'hôte.
- La balise de définition **OS** affiche le fait **os\_name**.
- La balise de définition **Memory** affiche le fait **memtotal\_mb**.
- La balise de définition **Processor vCPU** affiche le fait **processor\_vcpus**.
- La balise de définition **Processor Cores** affiche le fait **processor\_cores**.

Ouvrez le fichier de modèle Jinja2 **index.html.j2** dans le dossier **templates**, puis modifiez-le pour insérer des variables et des faits comme suit :

```
<!doctype html>
<!-- {{ ansible_managed }} -->
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Info</title>
  <style type="text/css">
    dt { font-weight: bold; }
  </style>
</head>
<body>
  <h1>{{ ansible_facts['hostname'] }}</h1>

  <dl>
    <dt>OS</dt>
    <dd>{{ ansible_facts['os_name'] }}</dd>

    <dt>Memory</dt>
    <dd>{{ ansible_facts['memtotal_mb'] }}</dd>

    <dt>Processor vCPUs</dt>
    <dd>{{ ansible_facts['processor_vcpus'] }}</dd>

    <dt>Processor Cores</dt>
    <dd>{{ ansible_facts['processor_cores'] }}</dd>
  </dl>
</body>
</html>
```

Cliquez sur **File → Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.

3. Définissez des tâches dans le fichier playbook **review.yml** sur :

- Assurez-vous que la fonctionnalité de serveur Web IIS est présente.
- Déployez le modèle Jinja2 **templates/index.html.j2** sur **C:\Inetpub\wwwroot\index.html** sur l'hôte géré.

- Refusez à **Users** sur les hôtes gérés l'accès en écriture (**Write**) au fichier **index.html**.

Enregistrez le playbook **review.yml**.

- 3.1. Ouvrez le fichier de playbook **review.yml** et modifiez-le pour insérer le contenu comme suit.

```
---
```

```
- name: Deploying Files to Managed Hosts Lab
  hosts: all
  tasks:
    - name: IIS Web Server started
      win_feature: ①
        name: Web-Server
        state: present

    - name: Hosts file template deployed
      win_template: ②
        src: templates/index.html.j2
        dest: C:\Inetpub\wwwroot\index.html
        backup: yes

    - name: User write permission is denied
      win_acl: ③
        path: C:\Inetpub\wwwroot\index.html
        user: Users
        rights: Write
        type: deny
        state: present
```

- ① Activez la fonctionnalité **IIS Web Server**, qui fournit la page Web **index.html**.
- ② Appelez le module **win\_template** pour déployer le modèle **index.html.j2** sur l'hôte géré. Utilisez l'argument **backup: yes** pour créer une sauvegarde horodatée.
- ③ Ajoutez une règle d'autorisation qui refuse au groupe **Users** l'accès en écriture au fichier **index.html**.

- 3.2. Cliquez sur **File → Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.

4. Validez et transmettez par push vos modifications au référentiel **managing-files**.

- 4.1. À partir de Visual Studio Code, enregistrez et indexez les fichiers. Accédez au panneau **Source Control**, puis cliquez sur **+** pour indexer toutes les modifications.
- 4.2. Saisissez **Deploys web page from template** comme message de validation, puis cliquez sur **Commit** ou appuyez sur **Ctrl+Entrée** pour valider vos modifications.
- 4.3. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
5. À partir de **workstation**, accédez à votre instance Red Hat Ansible Tower sur **http://tower.example.com**. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.

**chapitre 7 |** Déploiement de fichiers sur des hôtes gérés

Ajoutez l'hôte **win2.example.com** à l'inventaire **Default inventory** dans Ansible Tower si cela n'a pas déjà été fait lors d'un exercice précédent.

- 5.1. À partir de **workstation**, double-cliquez sur l'icône du Bureau Ansible Tower. Connectez-vous à Ansible Tower en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.
- 5.2. Cliquez sur **Inventories** dans le volet de navigation.
- 5.3. Cliquez sur **Default inventory** dans la page **INVENTORIES**.
- 5.4. Cliquez sur **HOSTS**.
- 5.5. Si **win2.example.com** n'est pas déjà listé dans la table hosts, cliquez sur **+** pour ajouter un nouvel hôte au groupe. Sélectionnez **New Host** dans la liste.
- 5.6. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
HOST NAME	win2.example.com

- 5.7. Cliquez sur **SAVE** pour ajouter le nouvel hôte.
6. Lancez le playbook **review.yml** à l'aide du modèle de tâche **Run managing-files project**.
  - 6.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 6.2. Cliquez sur le modèle de tâche **Run managing-files project**.
  - 6.3. Sélectionnez le playbook **review.yml** dans la liste **PLAYBOOK**.

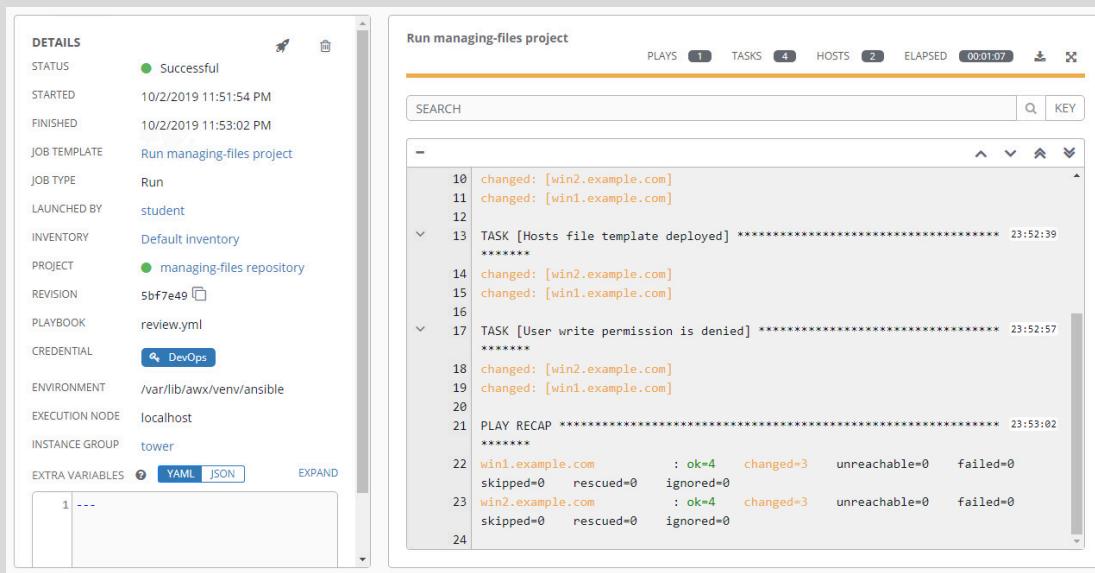
The screenshot shows the configuration interface for a 'Run managing-files project' template. The 'DETAILS' tab is selected. Key settings include:

- NAME:** Run managing-files project
- DESCRIPTION:** Use this job template to run your playbooks
- JOB TYPE:** Run
- INVENTORY:** Default inventory
- PROJECT:** managing-files repository
- PLAYBOOK:** review.yml
- CREDENTIAL:** DevOps
- FORKS:** 0
- LIMIT:** 1
- VERBOSITY:** 0 (Normal)
- SHOW CHANGES:** OFF
- OPTIONS:** Includes checkboxes for Enable Privilege Escalation, Allow Provisioning Callbacks, Enable Concurrent Jobs, and Use Fact Cache.

- 6.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
7. Vérifiez que la tâche **Run managing-files project** est correctement exécutée. Accédez à <http://win1.example.com> et <http://win2.example.com> pour vérifier que la page Web est correctement rendue et fournie.

**chapitre 7 |** Déploiement de fichiers sur des hôtes gérés

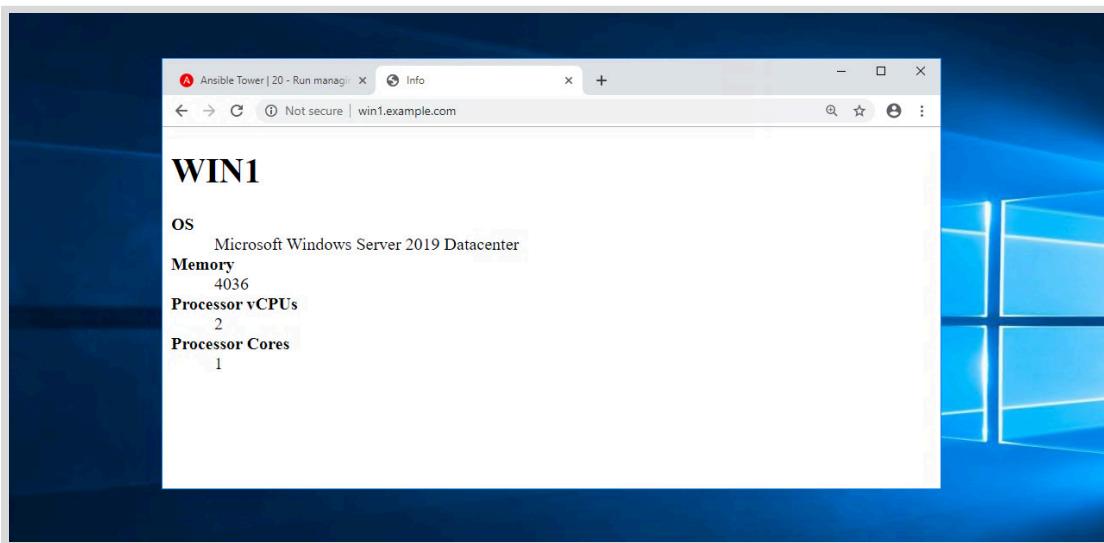
- 7.1. Vérifiez que le statut dans le volet **DETAILS** affiche **Successful**. Observez les tâches dans le journal qui indiquent que des modifications ont été apportées sur les hôtes gérés.



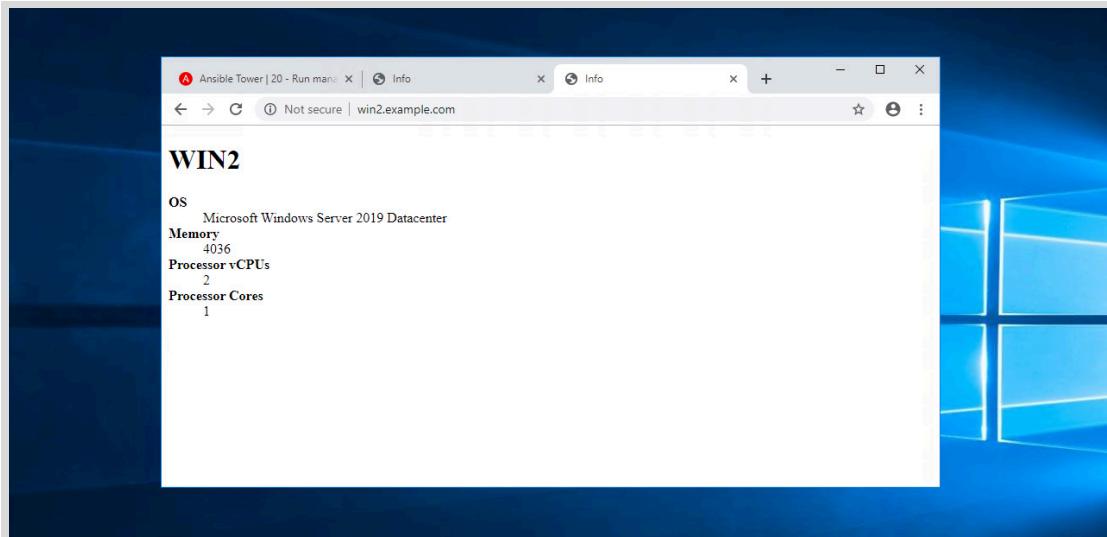
The screenshot shows the Ansible Tower interface. On the left, the 'DETAILS' panel displays job metadata: STATUS (Successful), STARTED (10/2/2019 11:51:54 PM), FINISHED (10/2/2019 11:53:02 PM), JOB TEMPLATE (Run managing-files project), JOB TYPE (Run), LAUNCHED BY (student), INVENTORY (Default inventory), PROJECT (managing-files repository), REVISION (5bf7e49), PLAYBOOK (review.yml), CREDENTIAL (DevOps), ENVIRONMENT (/var/lib/awx/venv/ansible), EXECUTION NODE (localhost), INSTANCE GROUP (tower), and EXTRA VARIABLES (YAML, JSON). On the right, the 'Run managing-files project' log window shows the command output:

```
PLAYS 1 TASKS 4 HOSTS 2 ELAPSED 00:01:07
SEARCH
10 changed: [win2.example.com]
11 changed: [win1.example.com]
12
13 TASK [Hosts file template deployed] ****
14 changed: [win2.example.com]
15 changed: [win1.example.com]
16
17 TASK [User write permission is denied] ****
18 changed: [win2.example.com]
19 changed: [win1.example.com]
20
21 PLAY RECAP ****
22 win1.example.com : ok=4 changed=3 unreachable=0 failed=0
skipped=0 rescued=0 ignored=0
23 win2.example.com : ok=4 changed=3 unreachable=0 failed=0
skipped=0 rescued=0 ignored=0
24
```

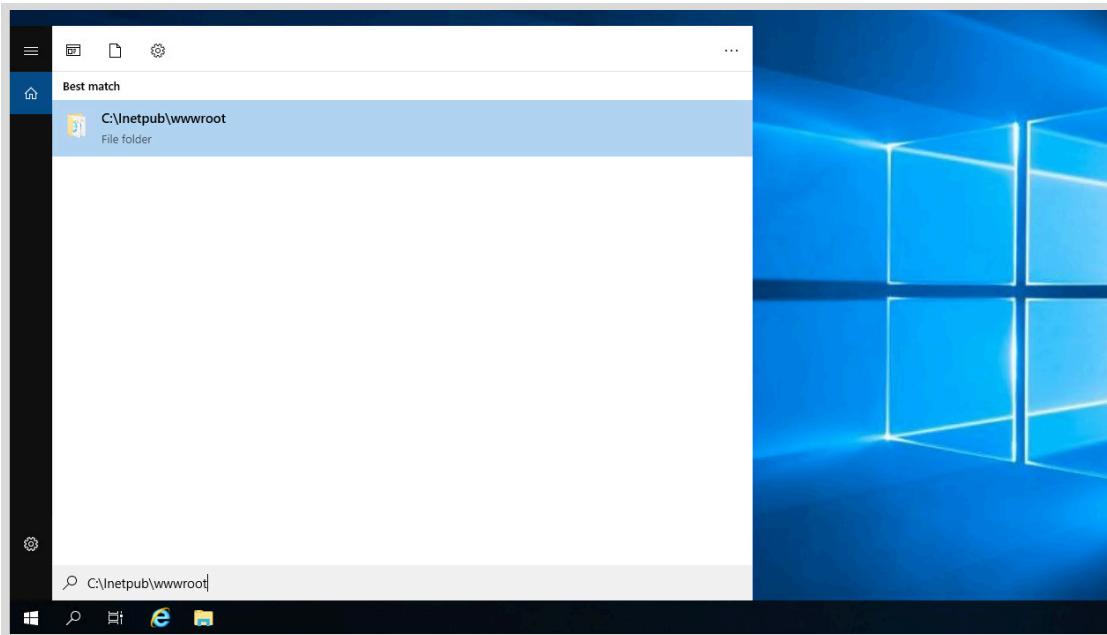
- 7.2. Dans Chrome, ouvrez un nouvel onglet et rendez-vous à l'adresse `http://win1.example.com/`. Le serveur Web sur l'hôte géré **win1** renvoie la page Web **index.html** générée.



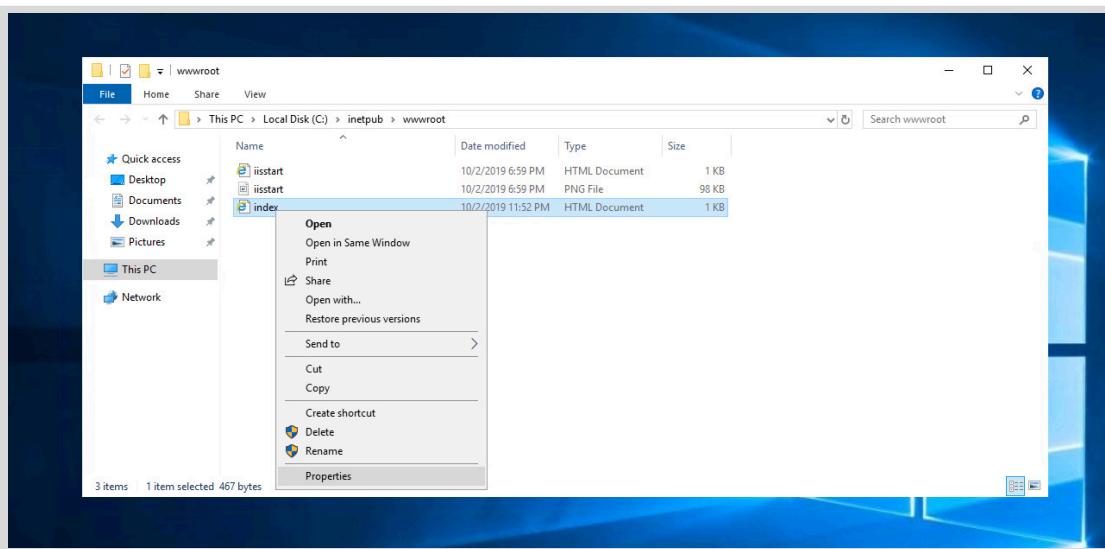
- 7.3. Dans Chrome, ouvrez un nouvel onglet et rendez-vous à l'adresse `http://win2.example.com/`. Le serveur Web sur l'hôte géré **win2** renvoie la page Web **index.html** générée.



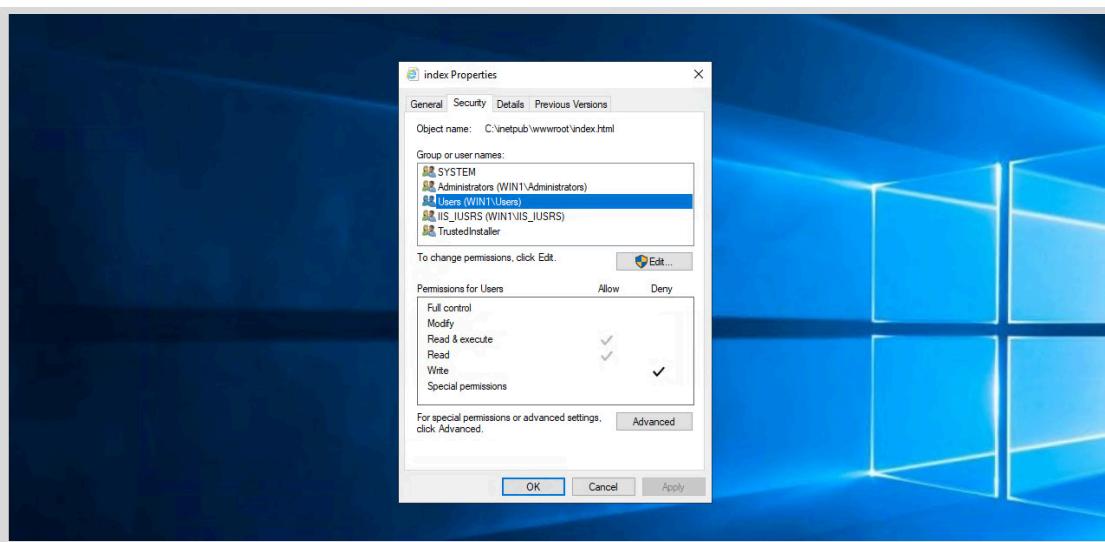
8. Connectez-vous à l'hôte géré **win1.example.com** pour vérifier que l'autorisation d'écriture sur **C:\Inetpub\wwwroot\index.html** est refusée.
  - 8.1. Cliquez sur **Recherche Windows**, recherchez **distance**, puis ouvrez Connexion Bureau à distance.
  - 8.2. Saisissez **win1.example.com** pour l'ordinateur, **devops** comme nom d'utilisateur et **RedHat123@!** comme mot de passe. Notez que **devops** est un utilisateur local, et non un utilisateur de domaine, n'incluez donc pas le domaine **EXAMPLE**.
  - 8.3. À l'invite, cliquez sur **Yes** pour accepter le certificat non sécurisé utilisé dans votre environnement de formation.
  - 8.4. Dans la session **win1.example.com** distante, cliquez sur **Search Windows**, puis tapez **C:\Inetpub\wwwroot** pour rechercher le répertoire racine du serveur Web. Cliquez sur le dossier du fichier pour l'ouvrir.



- 8.5. Cliquez avec le bouton droit sur le fichier **index.html**, sélectionnez **Properties**, puis cliquez sur l'onglet **Security**.



- 8.6. Sélectionnez **Users (WIN1\Users)** dans le panneau **Group or User Names**, puis vérifiez que l'option **Deny** est sélectionnée pour les autorisations **Write**.



- 8.7. Fermez la connexion du bureau à distance **win1.example.com**.
9. Lancez le playbook **cleanup-review.yml** à l'aide du modèle de tâche **Run managing-files project**.
- 9.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 9.2. Cliquez sur le modèle de tâche **Run managing-files project**.
  - 9.3. Sélectionnez le playbook **cleanup-review.yml** dans la liste **PLAYBOOK**.

Run managing-files project

**DETAILS**   **PERMISSIONS**   **NOTIFICATIONS**   **COMPLETED JOBS**   **SCHEDULES**   **ADD SURVEY**

* NAME Run managing-files project	DESCRIPTION Use this job template to run your playbooks	* JOB TYPE Run	PROMPT ON LAUNCH
* INVENTORY Default inventory	PROMPT ON LAUNCH	* PROJECT managing-files repository	PROMPT ON LAUNCH
CREDENTIAL DevOps	PROMPT ON LAUNCH	FORKS 0	LIMIT PROMPT ON LAUNCH
* VERBOSITY 0 (Normal)	PROMPT ON LAUNCH	JOB TAGS PROMPT ON LAUNCH	SKIP TAGS PROMPT ON LAUNCH
LABELS	INSTANCE GROUPS PROMPT ON LAUNCH	JOB SLICING 1	OPTIONS
TIMEOUT 0	SHOW CHANGES OFF	<input type="checkbox"/> ENABLE PRIVILEGE ESCALATION <input type="checkbox"/> ALLOW PROVISIONING CALLBACKS <input type="checkbox"/> ENABLE CONCURRENT JOBS <input type="checkbox"/> USE FACT CACHE	

9.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Ansible inclut un certain nombre de modules pour la gestion des fichiers Microsoft Windows, tels que la création, la copie et la modification de fichiers, ainsi que la gestion des ACL et la propriété des fichiers.
- Vous pouvez utiliser les modèles Jinja2 pour construire dynamiquement des fichiers à déployer.
- Un modèle Jinja2 se compose généralement de deux éléments : des variables et des expressions. Ces variables et expressions sont remplacées par des valeurs lors du rendu du modèle Jinja2.
- Les filtres Jinja2 transforment des expressions de modèle d'un type de données vers un autre.



## chapitre 8

# Interaction avec les utilisateurs et les domaines

### Objectif

Gérer les utilisateurs locaux et de domaine, gérer les domaines Active Directory et générer une liste d'hôtes gérés sur la base de l'appartenance à un domaine pour un inventaire dynamique Red Hat Ansible Tower.

### Résultats

- Automatiser la création et la gestion des comptes d'utilisateurs et de groupes locaux.
- Créer un domaine Active Directory le cas échéant et configurer l'appartenance des hôtes, les utilisateurs et les groupes dans ce domaine.
- Générez un inventaire Ansible de manière dynamique dans Red Hat Ansible Tower en fonction de l'appartenance à un domaine Active Directory.

### Sections

- Gestion des comptes d'utilisateur locaux (et exercice guidé)
- Gestion des domaines Active Directory (et exercice guidé)
- Génération d'inventaires dynamiques à partir d'Active Directory (et exercice guidé)

### Atelier

Atelier : Interaction avec les utilisateurs et les domaines

# Gestion des comptes d'utilisateur locaux

---

## Résultats

Au terme de cette section, vous serez en mesure d'automatiser la création et la gestion de comptes d'utilisateurs locaux et de groupe.

## Gestion de comptes d'utilisateurs sur les systèmes Windows

Le module Ansible **win\_user** vous permet de gérer des comptes d'utilisateurs locaux sur un hôte Windows distant. Vous pouvez gérer plusieurs paramètres, tels que la suppression de l'utilisateur, la gestion du mot de passe de l'utilisateur et la gestion des appartenances aux groupes de l'utilisateur.

### Création d'un nouveau compte d'utilisateur Windows local

L'exemple suivant montre comment ajouter un nouvel utilisateur local, et comment associer cet utilisateur aux groupes locaux appropriés.

```
- name: Add new user to the development machine and assign the appropriate groups.
  win_user:
    name: devops_user ①
    password: plain_text_password ②
    state: present ③
    groups: ④
      - Users
      - Developers
    groups_action: replace ⑤
```

- ① Le paramètre **name** est la seule condition requise pour le module.
- ② Le paramètre **password** permet éventuellement de définir le mot de passe de l'utilisateur. La valeur du mot de passe est écrite en texte clair.
- ③ Le paramètre **state** défini sur **present** indique à Ansible de créer ou de mettre à jour le compte d'utilisateur.
- ④ Le paramètre **groups** spécifie une liste de groupes (**Users** et **Developers**) auxquel l'utilisateur doit appartenir par défaut.
- ⑤ Le paramètre **groups\_action** contrôle la manière dont la liste des groupes est interprétée. Si elle ne figure pas dans la liste, l'action par défaut est **replace**, ce qui signifie que l'utilisateur sera membre exactement des groupes listés si l'utilisateur existe déjà et qu'il est membre de certains groupes. Dans l'exemple ci-dessus, cela est spécifié de manière explicite (une bonne pratique qui permet de s'assurer que ce qui se passera est évident).

### Suppression d'un compte d'utilisateur Windows local

L'exemple suivant montre comment supprimer un utilisateur local d'un système distant.

```
- name: Remove a local account
  win_user:
    name: devops_user ①
    state: absent ②
```

- ①** Le paramètre **name** est la seule condition requise dans le module utilisateur.
- ②** Le paramètre **state** indique à la machine de supprimer le compte d'utilisateur s'il existe sur le système.

## Paramètres couramment utilisés du module `win_user`

Le module `win_user` renvoie également des valeurs. Les modules Ansible peuvent prendre des valeurs de retour et les enregistrer dans une variable. Utilisez ce module pour interroger les détails du compte d'utilisateur sans apporter aucune modification au système.

Le tableau suivant décrit certains paramètres couramment utilisés.

### Paramètres couramment utilisés

Paramètre	Commentaires
Description	Définit, en option, la description d'un compte d'utilisateur.
groups	Ajoute ou supprime l'utilisateur d'une liste définie de groupes.
groups_action	Contrôle la manière dont le paramètre <b>groups</b> est interprété. S'il est défini sur <b>add</b> , l'utilisateur est ajouté à ces groupes, mais si l'utilisateur existe déjà et qu'il est membre d'autres groupes, l'utilisateur reste également dans ces groupes. S'il est défini sur <b>replace</b> , l'utilisateur serait fait membre de ces groupes exactement et supprimé de tous les autres groupes. S'il est défini sur <b>remove</b> , l'utilisateur sera supprimé des groupes listés.
account_locked	Accepte la valeur booléenne yes ou no. Lorsque ce paramètre est défini sur <b>no</b> , déverrouille un compte d'utilisateur s'il est verrouillé.
fullname	Nom complet de l'utilisateur.
password_never_expires	Accepte la valeur booléenne Yes ou No. Si ce paramètre est défini sur <b>Yes</b> , Ansible définit le mot de passe de sorte qu'il n'expire jamais, ou s'il est défini sur <b>no</b> , Ansible autorise l'expiration du mot de passe.
update_password	met <b>toujours</b> à jour le mot de passe si celui-ci est différent. <b>on-create</b> définit le mot de passe pour les utilisateurs nouvellement créés.
user_CANNOT_change_password	Accepte la valeur booléenne Yes ou No. Empêche l'utilisateur de modifier son mot de passe si la valeur est définie sur <b>yes</b> .

Paramètre	Commentaires
state	Accepte l'une de trois valeurs différentes. Lorsqu'il est défini sur <b>absent</b> , supprime un compte d'utilisateur le cas échéant. Lorsqu'il est défini sur <b>present</b> , crée ou met à jour le compte d'utilisateur. Lorsqu'il est défini sur <b>query</b> , récupère les détails du compte d'utilisateur sans apporter aucune modification.

## Gestion des profils d'utilisateurs locaux

Windows requiert un profil d'utilisateur pour chaque compte d'utilisateur sur un ordinateur. Normalement, le profil d'utilisateur local est créé pour chaque utilisateur lorsque celui-ci ouvre une session sur l'ordinateur pour la première fois. L'utilisation du module Ansible **win\_user\_profile** vous permet de créer ou de supprimer les profils d'utilisateurs locaux sur un hôte Windows. Vous pouvez créer ou supprimer un profil pour un compte local ou de domaine, créer un profil avant qu'un utilisateur n'ouvre une session, ou supprimer un profil lorsqu'un compte d'utilisateur n'existe pas.

### Création d'un nouveau profil d'utilisateur local

L'exemple suivant illustre l'utilisation du module **win\_user\_profile** pour créer un nouveau profil local pour un utilisateur local existant.

```
- name: Create a local profile for an account.
  win_user_profile:
    username: devops_user ①
    state: present ②
```

- ① Lorsque **state** est défini sur **present**, définissez le paramètre **username** sur un nom de compte valide.
- ② Lorsque le paramètre **state** est défini sur **present**, la machine crée ou met à jour le compte d'utilisateur.

### Suppression d'un profil d'utilisateur local

Supprimez un profil d'un compte existant, un compte supprimé, un compte supprimé basé sur le SID ou plusieurs profils partageant le même chemin d'accès de base à l'aide du module **win\_user\_profile**. L'exemple suivant montre comment supprimer un profil d'utilisateur local existant pour un compte existant.

```
- name: Remove a local profile for a still valid account.
  win_user_profile:
    username: devops_user ①
    state: absent ②
```

- ① Lorsque **state** est défini sur **present**, définissez le paramètre **username** sur un nom de compte valide.
- ② Lorsque le paramètre **state** est défini sur **absent**, la machine supprime le profil de l'utilisateur.

L'exemple suivant illustre la suppression d'un profil d'utilisateur local existant pour un compte existant.

```
- name: Remove a local profile for a deleted account.
  win_user_profile:
    name: devops_user ①
    state: absent ②
```

- ① Le paramètre **name** doit être défini sur un nom de profil valide. Gardez à l'esprit que lorsque **state** est défini sur **absent**, et que **username** n'est pas défini, le module supprime tous les profils contenus dans le chemin du profil pour la valeur **name**.
- ② Le paramètre **state** défini sur **absent** indique à la machine de supprimer le profil de l'utilisateur.



### Important

Souvenez-vous de la distinction importante entre le module **win\_user\_profile** et les paramètres **name** et **username**. Le paramètre **name** est le nom de base du chemin du profil. Le paramètre **username** est le nom de compte de l'identificateur de sécurité (SID) pour le profil.

## Groupes locaux

Le module **win\_group** vous permet de gérer les groupes locaux sur un hôte Windows. Vous pouvez créer ou supprimer un groupe local à l'aide de ce module.

### Création d'un nouveau groupe local

L'exemple suivant montre comment utiliser le module pour créer un nouveau groupe local.

```
- name: Create a new local group.
  win_group:
    name: developers ①
    description: Developers Group ②
    state: present ③
```

- ① Le paramètre **name** définit le nom du groupe.
- ② Le paramètre **description** décrit l'objet du groupe.
- ③ Le paramètre **state** défini sur **present** indique à la machine de créer le groupe local.

### Suppression d'un groupe local

L'exemple suivant montre comment supprimer un groupe local existant.

```
- name: Remove a local group.
  win_group:
    name: developers ①
    state: absent ②
```

- ① Le paramètre **name** définit le nom du groupe.
- ② Le paramètre **state** défini sur **absent** indique à la machine de supprimer le groupe local.

## Gestion de l'appartenance à un groupe local Windows

Utilisez le module **win\_group\_membership** pour ajouter et supprimer des utilisateurs locaux, des utilisateurs de services et de domaines, ainsi que des groupes de domaines d'un groupe local.

## Ajout de membres à un groupe local

L'exemple suivant illustre l'utilisation du module `win_group_membership` pour ajouter à la fois des utilisateurs locaux et des utilisateurs de domaine à un groupe local existant.

```
- name: Adding a local and domain user to a local group.
  win_group_membership:
    name: developers ①
    members: ②
      - devops
      - DOMAIN_NAME\DemoUser
    state: present ③
```

- ①** Le paramètre **name** définit le nom du groupe local.
- ②** Le paramètre **members** contient une liste des utilisateurs locaux, des utilisateurs de domaines ou des groupes de domaines possibles qui doivent être ajoutés ou supprimés du groupe local.
- ③** Le paramètre **state** définit l'état souhaité des membres du groupe local. Lorsqu'il est défini sur **pure**, seuls les membres spécifiés existent et tous les autres membres existants non spécifiés sont supprimés.

Le paramètre **members** peut accepter les éléments suivants :

- Les utilisateurs locaux en tant que `.\nomutilisateur` ou `NOMSERVEUR\inomutilisateur`.
- Les utilisateurs et les groupes de domaines en tant que `DOMAINE\nomutilisateur` et `nomutilisateur@DOMAINE`.
- Les utilisateurs de services en tant que `NT AUTHORITY\nomutilisateur`.
- Tous les types d'utilisateurs locaux, de domaines et de services comme `nomutilisateur`, favorisant les recherches de domaines si l'hôte se trouve dans un domaine.

## Suppression de membres d'un groupe local

L'exemple suivant montre comment utiliser le module pour supprimer un groupe de domaines d'un groupe local.

```
- name: Remove a domain group from a local group.
  win_group_membership:
    name: developers ①
    members: DOMAINNAME\DemoGroup ②
    state: absent ③
```

- ①** Le paramètre **name** définit le nom du groupe local à modifier.
- ②** Le paramètre **members** contient une liste des utilisateurs locaux, des utilisateurs de domaines ou des groupes de domaines possibles qui doivent être supprimés du groupe local.
- ③** Le paramètre **state** défini sur **absent** indique à la machine de supprimer les membres spécifiés du groupe local.



### Références

#### Modules Windows &mdash; Documentation Ansible

[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_windows\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html)

## ► Exercice guidé

# Gestion des comptes d'utilisateur locaux

Au cours de cet exercice, vous allez gérer des comptes d'utilisateurs locaux sur vos hôtes gérés.

## Résultats

Vous devez être en mesure de créer un nouvel utilisateur local, ainsi que de modifier l'appartenance à un groupe d'un utilisateur local existant.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.

- ▶ 1. Lancez l'éditeur Visual Studio Code et clonez le référentiel **ad** sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Cliquez sur **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/ad.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **ad**.  
Vous pouvez aussi cliquer sur **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers, puis passer à l'étape suivante.
- ▶ 2. Cliquez sur **File → Open Folder**, puis ouvrez le dossier **c:\Users\student\Documents\ad**.
- ▶ 3. Modifiez le fichier **localuser.yml**, puis validez vos modifications. Modifiez ce fichier de façon à ce qu'il cible tous les hôtes et qu'il utilise le module **win\_user** pour créer un utilisateur local **John Snow** avec **RedHat123@!** comme mot de passe.
  - 3.1. Sélectionnez le fichier **localuser.yml**, puis modifiez le nom de la tâche et les détails du nouvel utilisateur local que vous souhaitez créer.

```
...output omitted...
hosts: all

tasks:
  - name: John Snow user is created
    win_user:
      name: John Snow
      password: RedHat123@!
      state: present
      groups:
        - Users
```

- ▶ 4. Modifiez le fichier **localuser.yml**, puis validez vos modifications. Ajoutez une nouvelle tâche au playbook qui ajoute le nouvel utilisateur local au groupe **Print operators**. Pour cette tâche, utilisez le module **win\_group\_membership**.

- 4.1. Sélectionnez le fichier **localuser.yml**, puis modifiez le nom de la deuxième tâche et les détails du nouveau groupe auquel vous voulez ajouter l'utilisateur.

```
...output omitted...
  - name: John Snow added to the Print operators group
    win_group_membership:
      name: Print operators
      members:
        - John Snow
      state: present
```

- 4.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été effectuées.
- 4.3. Passez à l'affichage **Source Control**, puis cliquez sur **+** en regard de **localuser.yml** pour effectuer les modifications.
- 4.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 4.5. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.

À l'invite, cliquez sur **Yes** ou **Ask Me Later**.

- ▶ 5. Accédez à la console Web Ansible Tower. Utilisez le modèle de tâche **Run ad project** pour tester votre playbook Ansible modifié.
- 5.1. Double-cliquez sur le raccourci du Bureau nommé Ansible Tower et authentifiez-vous avec l'utilisateur **admin** et le mot de passe **RedHat123@!**.
  - 5.2. Dans le volet de navigation, cliquez sur **Templates**.
  - 5.3. Dans la liste des modèles, sélectionnez le modèle **Run ad project**.

- 5.4. Sélectionnez le playbook **localuser.yml** dans la liste **PLAYBOOK**.
- 5.5. Activez la case à cocher **PROMPT ON LAUNCH** près de l'option **LIMIT**.
- 5.6. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Sélectionnez **LAUNCH** pour commencer l'exécution.
- 5.7. Dans la fenêtre **RUN AD PROJECT**, saisissez le nom d'hôte **win1.example.com**.  
Cliquez sur **NEXT**, puis sur **LAUNCH**.
- 5.8. Observez la sortie en direct de la tâche en cours d'exécution.
- 5.9. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- 5.10. Cliquez sur **Log Out** pour quitter l'interface Web d'Ansible Tower.

L'exercice guidé est maintenant terminé.

# Gestion des domaines Active Directory

## Résultats

Au terme de cette section, vous devez pouvoir créer un domaine Active Directory le cas échéant et configurer l'appartenance des hôtes, les utilisateurs et les groupes dans ce domaine.

## Création de domaines Active Directory

Vous pouvez utiliser Ansible pour créer et gérer des domaines Active Directory. À l'aide du module **win\_domain** Ansible, vous pouvez créer un nouveau domaine et une nouvelle forêt. Lorsque vous créez un nouveau domaine et une nouvelle forêt, vous pouvez spécifier le domaine et le niveau fonctionnel de la forêt. Cependant, le niveau fonctionnel du domaine ne peut pas être inférieur au niveau fonctionnel de la forêt. Une fois que ce module a modifié votre système, vous devez redémarrer. Utilisez un gestionnaire qui appelle le module **win\_reboot** pour redémarrer une machine Windows et qui attend qu'elle s'arrête puis redémarre.

## Création d'un nouveau domaine Active Directory

L'exemple suivant illustre la création d'un nouveau domaine, notamment une nouvelle forêt et la configuration DNS.

```
- name: Creates a new Windows domain in a new forest with specific parameters
  win_domain:
    create_dns_delegation: yes 1
    database_path: C:\Windows\NTDS 2
    dns_domain_name: EXAMPLE.COM 3
    domain_mode: Win2012R2 4
    domain_netbios_name: EXAMPLE 5
    forest_mode: Win2012R2 6
    safe_mode_password: plain_text_password 7
    sysvol_path: C:\Windows\SYSVOL 8
    register: domain_install 9

- name: Rebooting the server
  win_reboot:
    msg: "Rebooting..."
    when: domain_install.reboot_required
```

- 1** Le paramètre **create\_dns\_delegation** contrôle la création d'une délégation DNS qui fait référence au nouveau serveur DNS que vous installez, ainsi que le contrôleur de domaine. Cette opération n'est valide que pour le DNS intégré à Active Directory.
- 2** Le paramètre **database\_path** définit le chemin d'accès à l'emplacement du répertoire dans lequel la base de données du domaine est créée.
- 3** Le paramètre **dns\_domain\_name** définit le nom DNS du domaine.
- 4** Le paramètre **domain\_mode** définit le niveau fonctionnel du domaine (les choix disponibles sont : **Win2003**, **Win2008**, **Win2008R2**, **Win2012**, **Win2012R2** et **WinThreshold**).
- 5** Le paramètre **domain\_netbios\_name** définit le nom NetBIOS du domaine root dans la nouvelle forêt.

- ➏ Le paramètre **forest\_mode** définit le niveau fonctionnel de la nouvelle forêt (les choix disponibles sont : **Win2003**, **Win2008**, **Win2008R2**, **Win2012**, **Win2012R2**, **WinThreshold**).
- ➐ Le paramètre **safe\_mode\_password** définit le mot de passe du mode sans échec. Il s'agit d'un paramètre obligatoire. La valeur du mot de passe est écrite en texte clair.
- ➑ Le paramètre **sysvol\_path** définit le chemin d'accès du répertoire dans lequel le fichier **Sysvol** est créé.
- ➒ Le paramètre **register** enregistre l'état du résultat ; en fonction de l'état, un gestionnaire peut être déclenché. Dans cet exemple, lorsque l'état change, le gestionnaire déclenche un redémarrage.

## Création de contrôleurs Active Directory supplémentaires

Vous pouvez utiliser Ansible pour créer et gérer des contrôleurs Active Directory. À l'aide du module **win\_domain\_controller** Ansible, vous pouvez configurer un hôte en tant que contrôleur de domaine ou le rétrograder à un serveur membre. Une fois que ce module a modifié votre système, vous devez redémarrer. Utilisez un gestionnaire qui appelle le module **win\_reboot** pour redémarrer une machine Windows et qui attend qu'elle s'arrête puis redémarre.

### Création d'un nouveau contrôleur Active Directory

L'exemple suivant montre comment promouvoir un serveur en un deuxième contrôleur de domaine.

```
- name: Promoting the server to a domain controller.
  win_domain_controller:
    dns_domain_name: example.com ①
    domain_admin_user: demo_user@example.com ②
    domain_admin_password: plain_text_password ③
    safe_mode_password: plain_text_password ④
    state: domain_controller ⑤
    register: domain_state ⑥

- name: Rebooting the server
  win_reboot:
    msg: "Rebooting..."
    when: domain_state.reboot_required
```

- ➑ Le paramètre **dns\_domain\_name** spécifie le nom DNS du domaine pour lequel l'hôte est promu contrôleur de domaine.
- ➒ Le paramètre **domain\_admin\_user** est obligatoire. Le paramètre spécifie le nom de l'utilisateur d'administration du domaine.
- ➓ Le paramètre **domain\_admin\_password** est obligatoire et spécifie le mot de passe de l'utilisateur dans le paramètre **domain\_admin\_user**. La valeur du mot de passe est écrite en texte clair.
- ➔ Le paramètre **safe\_mode\_password** est requis lors de la promotion d'un serveur en contrôleur de domaine. Ce paramètre définit le mot de passe du mode sans échec pour le contrôleur. La valeur du mot de passe est écrite en texte clair.
- ➕ Le paramètre **state** défini sur **domain\_controller** indique à Ansible de promouvoir le serveur en contrôleur de domaine.
- ➖ Le paramètre **register** est utilisé pour enregistrer l'état du résultat ; en fonction de l'état, un gestionnaire peut être déclenché. Dans cet exemple, lorsque l'état change, le gestionnaire déclenche un redémarrage.

En fonction de la configuration de votre environnement local, ce module Ansible configure le serveur en tant que contrôleur de domaine en lecture-écriture dans le domaine existant, ou il peut promouvoir le serveur en tant que réplica supplémentaire en lecture seule du domaine existant. Afin de créer un contrôleur de domaine en lecture seule, vous devez ajouter le paramètre **read\_only** au playbook et le définir sur **yes**.

## Installation des fonctions Windows d'Active Directory

L'exemple suivant montre comment installer les fonctions Windows nécessaires à la création et à la gestion des services Active Directory.

```
- name: Installing AD features.
  win_feature:
    name: AD-Domain-Services ①
    include_sub_features: yes ②
    include_management_tools: yes ③
    state: present
```

- ①** Le paramètre **name** est la seule condition requise pour le module. Ce paramètre spécifie les fonctions ou les rôles Windows disponibles à installer.
- ②** Si elle est définie sur **yes**, cette option ajoute toutes les sous-fonctions disponibles de la fonction spécifiée.
- ③** Lorsqu'elle est définie sur **yes**, cette option ajoute tous les outils de gestion de la fonction spécifiée correspondante.

## Rétrogradation d'un contrôleur de domaine à un membre de domaine

Vous pouvez également utiliser le module **win\_domain\_controller** pour rétrograder un serveur d'un contrôleur de domaine à un membre du domaine. L'exemple suivant montre comment utiliser le module pour rétrograder un contrôleur de domaine existant à un membre du domaine.

```
- name: Demoting a domain controller
  win_domain_controller:
    dns_domain_name: example.com ①
    domain_admin_user: demo_user@example.com ②
    domain_admin_password: plain_text_password ③
    local_admin_password: plain_text_password ④
    state: member_server ⑤
```

- ①** Le paramètre **dns\_domain\_name** spécifie le domaine à partir duquel le serveur sera rétrogradé.
- ②** Le paramètre **domain\_admin\_user** est obligatoire. Le paramètre spécifie le nom de l'utilisateur d'administration du domaine.
- ③** Le paramètre **domain\_admin\_password** est obligatoire. Ce paramètre spécifie le mot de passe de l'utilisateur dans le paramètre **domain\_admin\_user**. La valeur du mot de passe est écrite en texte clair.
- ④** Le paramètre **local\_admin\_password** est requis lors de la rétrogradation d'un serveur en membre de domaine. Ce paramètre définit le mot de passe de l'administrateur local pour le serveur. La valeur du mot de passe est écrite en texte clair.
- ⑤** Le paramètre **state** défini sur **member** indique à Ansible de rétrograder le serveur d'un contrôleur de domaine à un membre de domaine.

## Jonction et suppression d'un serveur d'un domaine Active Directory

Vous pouvez utiliser Ansible pour joindre de nouveaux hôtes à un domaine Active Directory existant. À l'aide du module **win\_domain\_membership** Ansible, vous pouvez joindre un hôte à un domaine. Lorsque que ce module modifie votre système, vous devez redémarrer. Utilisez un gestionnaire qui appelle le module **win\_reboot** et qui attend que la machine s'arrête puis redémarre.

### Jonction d'un domaine Active Directory

L'exemple suivant montre comment joindre un serveur à un domaine.

```
- name: Joining a domain
  win_domain_membership:
    dns_domain_name: EXAMPLE.COM1
    domain_admin_user: demo_admin@example.com2
    domain_admin_password: plain_text_password3
    state: domain4
    register: domain_state5

- name: Rebooting the server
  win_reboot:
    msg: "Rebooting..."
    when: domain_state.reboot_required
```

- <sup>1</sup> Le paramètre **dns\_domain\_name** définit le nom DNS du domaine.
- <sup>2</sup> Le paramètre **domain\_admin\_user** est obligatoire. Le paramètre spécifie le nom de l'utilisateur d'administration du domaine.
- <sup>3</sup> Le paramètre **domain\_admin\_password** est obligatoire. Ce paramètre spécifie le mot de passe de l'utilisateur dans le paramètre **domain\_admin\_user**. La valeur du mot de passe est écrite en texte clair.
- <sup>4</sup> Le paramètre **state** défini sur **domain** indique à Ansible de joindre le serveur à un domaine.
- <sup>5</sup> Le paramètre **register** est utilisé pour enregistrer l'état du résultat ; en fonction de l'état, un gestionnaire peut être déclenché. Dans cet exemple, lorsque l'état change, le gestionnaire déclenche un redémarrage.

### Suppression d'un serveur d'un domaine Active Directory

Vous pouvez utiliser le même module Ansible pour supprimer un serveur d'un domaine. L'exemple suivant montre comment supprimer un serveur d'un domaine.

```
- name: Removing the server from the domain
  win_domain_membership:
    workgroup_name: demo_workgroup1
    domain_admin_user: demo_admin@example.com2
    domain_admin_password: plain_text_password3
    state: workgroup4
```

- <sup>1</sup> Le paramètre **workgroup\_name** définit le nom du groupe de travail dont le serveur fera partie.
- <sup>2</sup> Le paramètre **domain\_admin\_user** est obligatoire. Le paramètre spécifie le nom de l'utilisateur d'administration du domaine.

- ❸ Le paramètre **domain\_admin\_password** est obligatoire. Ce paramètre spécifie le mot de passe de l'utilisateur dans le paramètre **domain\_admin\_user**. La valeur du mot de passe est écrite en texte clair.
- ❹ Le paramètre **state** défini sur **workgroup** indique à Ansible de supprimer le serveur d'un domaine.

## Création, modification ou suppression d'objets dans un domaine Active Directory

Vous pouvez utiliser Ansible pour manipuler des objets dans votre domaine Active Directory. Par exemple, vous pouvez créer, modifier ou supprimer des utilisateurs de domaine, des groupes et des comptes d'ordinateur de votre domaine.

### Gestion des comptes d'utilisateurs Active Directory

Vous pouvez utiliser le module **win\_domain\_user** pour gérer les comptes d'utilisateurs de domaine. L'exemple suivant montre comment créer ou supprimer un compte d'utilisateur de domaine

```
- name: Creating a new domain user account with details
  win_domain_user:
    name: robert❶
    firstname: Robert❷
    surname: Johnson❸
    company: Demo Inc.❹
    password: plain_text_password❺
    state: present❻
    groups:
      - Domain Users
    street: 123 14th St.❻
    city: Demotown
    state_province: IN
    postal_code: 12345
    country: US
    attributes:
      telephonenumber: 111-123456
```

- ❶ Le paramètre **name** est obligatoire. Le paramètre définit le nom de l'utilisateur à créer, à supprimer ou à modifier.
- ❷ Le paramètre **firstname** définit le prénom de l'utilisateur.
- ❸ Le paramètre **lastname** définit le nom de l'utilisateur.
- ❹ Le paramètre **company** définit le nom de l'entreprise de l'utilisateur.
- ❺ Le paramètre **password** spécifie le mot de passe de l'utilisateur. La valeur du mot de passe est écrite en texte clair.
- ❻ Le paramètre **state** définit le type d'action qu'Ansible doit effectuer. Lorsqu'il est défini sur **present**, Ansible crée ou met à jour l'utilisateur si le compte existe. Lorsqu'il est défini sur **absent**, Ansible supprime le compte d'utilisateur du domaine. Lorsqu'il est défini sur **query**, Ansible récupère les détails du compte d'utilisateur sans apporter aucune modification.
- ❼ Le paramètre **groups** ajoute ou supprime l'utilisateur d'une liste de groupes.
- ❽ Des informations supplémentaires sur l'utilisateur peuvent être spécifiées et stockées dans votre domaine.

Pour supprimer un compte d'utilisateur du domaine, le module n'a besoin que de deux paramètres. Ces paramètres sont le nom correct du compte à supprimer, et le paramètre **state** défini sur **absent**.

## Création d'un nouveau groupe de domaines

Utilisez le module **win\_domain\_group** pour gérer les groupes de domaines. L'exemple suivant montre comment créer un groupe de domaines ou s'assurer qu'il existe.

```
- name: Create a new domain group
  win_domain_group:
    name: developers ❶
    description: Developers Group ❷
    state: present ❸
```

- ❶** Le paramètre **name** définit le nom du groupe.
- ❷** Le paramètre **description** décrit l'objet du groupe.
- ❸** Le paramètre **state** défini sur **present** indique à Ansible de créer le groupe de domaines.

## Suppression d'un groupe de domaines

L'exemple suivant montre comment supprimer un groupe de domaines existant ou s'assurer qu'il n'existe pas.

```
- name: Remove a domain group
  win_domain_group:
    name: developers ❶
    state: absent ❷
```

- ❶** Le paramètre **name** définit le nom du groupe.
- ❷** Le paramètre **state** défini sur **absent** indique à Ansible de supprimer le groupe de domaines.

## Ajout d'utilisateurs de domaine à un groupe de domaines

À l'aide du module **win\_domain\_group\_membership**, vous pouvez ajouter et supprimer des utilisateurs de domaine, ou des groupes de domaines, d'un groupe de domaines.

L'exemple suivant illustre l'utilisation du module **win\_domain\_group\_membership** pour ajouter des utilisateurs de domaine à un groupe de domaines existant.

```
- name: Adding domain users to a domain group
  win_domain_group_membership:
    name: developers ❶
    members: ❷
      - DomainUser
      - DomainUser2
    state: present ❸
```

- ❶** Le paramètre **name** définit le nom du groupe de domaines.
- ❷** Le paramètre **members** contient une liste des utilisateurs de domaines ou des groupes de domaines possibles à ajouter ou supprimer dans le groupe de domaines.

- ③ Le paramètre **state** définit l'état souhaité des membres du groupe de domaines. Lorsqu'il est défini sur **pure**, seuls les membres spécifiés existent et Ansible supprime tous les autres membres existants qui ne sont pas spécifiés.

## Création d'un nouveau compte d'ordinateur

Vous pouvez utiliser Ansible pour ajouter de nouveaux comptes d'ordinateur à votre domaine Active Directory existant. L'exemple suivant montre comment utiliser le module **win\_domain\_computer** pour ajouter un nouveau compte d'ordinateur à votre domaine.

```
- name: Creating a new computer account in the domain
  win_domain_computer:
    name: WINDOWS11
    sam_acount_name: windows1$2
    dns_hostname: windows1.example.com3
    ou: "CN=Computers,DC=example,DC=com"4
    state: present5
```

- ① Le paramètre **name** définit le nom de l'objet LDAP.
- ② Le paramètre **sam\_account\_name** spécifie le nom de compte du Security Account Manager (SAM) de l'ordinateur. Tous les noms de compte SAM doivent se terminer par le signe \$.
- ③ Le paramètre **dns\_hostname** définit le nom de domaine complet (FQDN) DNS souhaité de l'ordinateur.
- ④ Le paramètre **ou** spécifie l'*unité organisationnelle* dans laquelle créer le nouvel objet ordinateur.
- ⑤ Le paramètre **state** définit l'état souhaité de l'ordinateur dans le domaine. Lorsqu'il est défini sur **present**, l'objet ordinateur est créé. Lorsqu'il est défini sur **absent**, l'objet est supprimé.



### Références

#### Documentation Ansible — Modules Windows

[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_windows\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html)

## ► Exercice guidé

# Gestion des domaines Active Directory

Dans cet exercice, vous allez ajouter un nouveau contrôleur de domaine Active Directory à votre domaine existant, et ajouter des hôtes, des utilisateurs et des groupes à ce domaine.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Associer un nouveau serveur à un domaine.
- Ajouter un deuxième contrôleur de domaine à votre domaine existant.
- Ajouter des utilisateurs et des groupes à un domaine.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. Lancez l'éditeur Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **ad**, clonez-le sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/ad.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **ad**. Vous pouvez éventuellement sélectionner **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers.
- ▶ 2. Accédez à **File → Open Folder**, puis sélectionnez le dossier **c:\Users\student\Documents\ad** pour ouvrir le répertoire **c:\Users\student\Documents\ad**.
- ▶ 3. Modifiez le fichier **join-ad.yml**, puis validez et poussez vos modifications.  
Modifiez le fichier de façon à ce qu'il cible tous les hôtes et qu'il utilise le module **win\_domain\_membership** pour joindre le serveur **win2.example.com** au domaine AD existant **example.com**.

- 3.1. Sélectionnez le fichier **join-ad.yml**, puis modifiez le nom de la tâche et les détails nécessaires pour qu'un serveur rejoigne le domaine **example.com**.

```
...output omitted...
hosts: all
vars:
  domainname: EXAMPLE.COM
  dnsname: example.com
  domain_admin: Administrator@example.com

tasks:
  - name: Joining the "{{ domainname }}" domain
    win_domain_membership:
      dns_domain_name: "{{ dnsname }}"
      domain_admin_user: "{{ domain_admin }}"
      domain_admin_password: "{{ password }}"
      state: domain
    register: domain_state

...output omitted...
```



#### Note

Notez que **domain\_admin\_password** utilise une variable qui n'est pas définie dans le playbook. Pour des raisons de sécurité, le mot de passe n'est pas stocké dans le playbook proprement dit. Au lieu de cela, vous allez créer un sondage dans Ansible Tower pour définir la valeur de cette variable à une étape ultérieure.

- 3.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, indiquant que des modifications ont été effectuées.
- 3.3. Passez à l'affichage **Source Control**, puis cliquez sur **+** en regard de **join-ad.yml** pour effectuer les modifications.
- 3.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 3.5. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.

À l'invite, cliquez sur **Yes** ou **Ask Me Later**.

- 4. Ajoutez un questionnaire au modèle de tâche **Run ad project**.



#### Note

Pour plus d'informations sur les sondages Ansible Tower, voir un chapitre ultérieur. Les sondages fournissent aux utilisateurs des questions personnalisées. Cette méthode prend en charge les invites de saisie de valeurs de variables supplémentaires qui soient plus conviviales que celles de la méthode **PROMPT ON LAUNCH**.

- 4.1. Connectez-vous à l'interface Ansible Tower. Cliquez sur **Templates** dans le volet de navigation.
- 4.2. Dans la liste des modèles disponibles, cliquez sur **Run ad project** pour modifier le modèle de tâche.
- 4.3. Cliquez sur **ADD SURVEY** pour ajouter un questionnaire.
- 4.4. Sur l'écran suivant, saisissez les informations suivantes dans le champ de texte **EXTRA VARIABLES** dans la partie inférieure de la fenêtre :

Champ	Valeur
PROMPT	Mot de passe de l'administrateur Active Directory
DESCRIPTION	Il s'agit du mot de passe d'administration AD.
ANSWER VARIABLE NAME	password
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40
REQUIRED	Sélectionné

- 4.5. Cliquez sur **+ADD** pour ajouter l'invite de sondage au sondage. Un aperçu de votre sondage s'affiche.

**Important**

Avant d'enregistrer, assurez-vous que le commutateur **ON/OFF** dans la fenêtre de l'éditeur du sondage est sur **ON**.

- 4.6. Cliquez sur **SAVE** pour ajouter le sondage au modèle de tâche.
  - 4.7. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.
- 5. Ajoutez le serveur **win2.example.com** à l'inventaire **Default Inventory** dans Ansible Tower.
- 5.1. Cliquez sur **Inventories** dans le volet de navigation.
  - 5.2. Cliquez sur **Default inventory** dans la page **INVENTORIES**.
  - 5.3. Cliquez sur l'onglet **GROUPS**, puis sélectionnez le groupe **Windows**.
  - 5.4. Cliquez sur l'onglet **HOSTS** pour accéder à la liste d'hôtes dans le groupe.
  - 5.5. Cliquez sur **+**, puis sélectionnez **New host** pour ajouter un nouvel hôte au groupe.
  - 5.6. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
HOST NAME	win2.example.com

- 5.7. Cliquez sur **SAVE** pour ajouter le nouvel hôte.
- 6. Utilisez le modèle de tâche **Run ad project** pour tester votre playbook Ansible modifié.
- 6.1. Cliquez sur **Templates** dans le volet de navigation.
  - 6.2. Dans la liste des modèles, sélectionnez le modèle **Run ad project**.

**Note**

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 6.3. Sélectionnez le playbook **join-ad.yml** dans la liste **PLAYBOOK**.
  - 6.4. Assurez-vous que la case **PROMPT ON LAUNCH** pour l'option **LIMIT** est cochée.
  - 6.5. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche, puis cliquez sur **LAUNCH** pour commencer l'exécution.
  - 6.6. Dans la fenêtre **RUN AD PROJECT**, saisissez le nom d'hôte **win2.example.com**. Cliquez sur **NEXT**.
  - 6.7. Dans le champ **ACTIVE DIRECTORY ADMINISTRATOR PASSWORD**, saisissez **RedHat123@!**, c'est-à-dire le mot de passe de l'administrateur Active Directory. Ce mot de passe est identique au compte administrateur de **windc**. Cliquez sur **NEXT** pour passer en revue les valeurs du questionnaire, puis cliquez sur **LAUNCH**.
  - 6.8. Observez la sortie en direct de la tâche en cours d'exécution. Pour rejoindre le domaine, vous devez redémarrer le serveur **win2.example.com**. Le playbook contient un gestionnaire qui est déclenché lorsque les paramètres de domaine d'hôte sont modifiés, provoquant ainsi le redémarrage du serveur par Ansible. L'état de la tâche change une fois que le serveur a redémarré avec succès.
  - 6.9. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- 7. Modifiez le fichier **second-ad-server.yml**, puis validez et poussez vos modifications. Modifiez ce fichier de façon à ce qu'il cible tous les hôtes et qu'il utilise le module **win\_domain\_controller** pour promouvoir le serveur **win2.example.com** en contrôleur de domaine. Vous pouvez utiliser ce même module pour créer un domaine Active Directory entièrement nouveau.
- 7.1. Sélectionnez le fichier **second-ad-server.yml**, puis modifiez le nom de la tâche et les détails nécessaires pour promouvoir le serveur **win2.example.com** au statut de contrôleur de domaine.

```
...output omitted...
hosts: all
vars:
  domainname: EXAMPLE.COM
  dnsname: example.com
```

```

domain_admin: Administrator@example.com
safe_password: RedHat123@!

tasks:
  - name: Promoting the server to an AD controller
    win_domain_controller:
      dns_domain_name: "{{ dnsname }}"
      domain_admin_user: "{{ domain_admin }}"
      domain_admin_password: "{{ password }}"
      safe_mode_password: "{{ safe_password }}"
      state: domain_controller
      register: domain_state
...output omitted...

```



### Note

Notez que **domain\_admin\_password** utilise une variable qui n'est pas définie dans le playbook. Pour des raisons de sécurité, le mot de passe n'est pas stocké dans le playbook proprement dit. Au lieu de cela, le sondage dans Ansible Tower permet de définir la valeur de cette variable.

- 7.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été effectuées.
- 7.3. Passez à l'affichage **Source Control**, puis sélectionnez + pour **second-ad-server.yml** afin d'effectuer les modifications.
- 7.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 7.5. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.  
À l'invite, cliquez sur **Yes** ou **Ask Me Later**.
- 8. Utilisez le modèle de tâche **Run ad project** pour tester votre playbook Ansible modifié.
  - 8.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 8.2. Dans la liste des modèles, sélectionnez le modèle **Run ad project**.



### Note

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 8.3. Sélectionnez le playbook **second-ad-server.yml** dans la liste **PLAYBOOK**.
- 8.4. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **LIMIT** est activée.
- 8.5. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.

Cliquez sur **LAUNCH** pour commencer l'exécution.

- 8.6. Dans la fenêtre **RUN AD PROJECT**, utilisez le nom d'hôte **win2.example.com**. Cliquez sur **NEXT**.
  - 8.7. Dans le champ **ACTIVE DIRECTORY ADMINISTRATOR PASSWORD**, saisissez **RedHat123@!**, c'est-à-dire le mot de passe de l'administrateur Active Directory. Ce mot de passe est identique au compte administrateur de **windc**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 8.8. Observez la sortie en direct de la tâche en cours d'exécution. Pour rejoindre le domaine, vous devez redémarrer le serveur **win2.example.com**. Le playbook contient un gestionnaire qui est déclenché lorsque les paramètres de domaine d'hôte sont modifiés, provoquant ainsi le redémarrage du serveur par Ansible. L'état de la tâche change une fois que le serveur a redémarré avec succès.
  - 8.9. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**. Il faut un certain temps pour que les services de domaine Active Directory achèvent la mise à jour de l'état.
- ▶ 9. Modifiez le fichier **domain-mod.yml**, puis validez et poussez vos modifications. Utilisez ce fichier pour créer un groupe de domaines, et un utilisateur de domaine qui est membre de ce nouveau groupe.
- 9.1. Sélectionnez le fichier **domain-mod.yml**, puis modifiez le nom de la tâche et les détails nécessaires à la création du nouveau groupe et du nouvel utilisateur.

```
...output omitted...
hosts: all
...output omitted...
user_password: RedHat123@!

tasks:
  - name: Creating a new group in the {{ domainname }} domain
    win_domain_group:
      name: Test Group
      scope: domainlocal
      category: security
      attributes:
        mail: helpdesk@example.com
        wwwHomePage: www.example.com
      ignore_protection: yes

  - name: Creating a new user
    win_domain_user:
      name: daniel
      firstname: Daniel
      surname: George
      password: "{{ user_password }}"
      state: present
      groups:
        - Domain Users
        - Test Group
      street: 125 42nd St.
      city: Sometown
```

```
state_province: CA
postal_code: 1234
country: US
attributes:
  telephoneNumber: 543-123456
```

- 9.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été effectuées.
  - 9.3. Passez à l'affichage **Source Control**, puis sélectionnez + pour **domain-mod.yml** afin d'effectuer les modifications.
  - 9.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 9.5. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- À l'invite, cliquez sur **Yes** ou **Ask Me Later**.
- ▶ 10. Ajoutez le serveur **windc.example.com** à l'inventaire **Default Inventory** dans Ansible Tower.
- 10.1. Cliquez sur **Inventories** dans le volet de navigation.
  - 10.2. Cliquez sur **Default inventory** dans la page **INVENTORIES**. Cliquez sur **GROUPS** pour accéder aux groupes d'inventaire et sélectionnez le groupe **Windows**.
  - 10.3. Cliquez sur **HOSTS** pour ajouter un hôte à ce groupe.
  - 10.4. Cliquez sur + pour ajouter un nouvel hôte au groupe. Sélectionnez **New Host** dans la liste qui s'affiche.
  - 10.5. Modifiez la méthode d'authentification utilisée par Ansible pour le serveur **windc.example.com** sur Kerberos pour configurer l'authentification centralisée. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
HOST NAME	windc.example.com
VARIABLES	ansible_winrm_transport: kerberos

- 10.6. Cliquez sur **SAVE**.

▶ 11. Créez de nouvelles information d'identification appelées **AD Administrator**.

  - 11.1. Cliquez sur **Credentials** dans le volet de navigation.
  - 11.2. Cliquez sur + pour ajouter de nouvelles informations d'identification.
  - 11.3. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
NAME	Administrateur AD
DESCRIPTION	Informations d'identification d'administrateur AD
ORGANIZATION	Valeur par défaut
CREDENTIAL TYPE	Machine
USERNAME	Administrateur
PASSWORD	<b>RedHat123@!</b>

- 11.4. Ne modifiez pas les autres champs et cliquez sur **SAVE** pour créer les nouvelles informations d'identification.
- ▶ 12. Utilisez le modèle de tâche **Run ad project** pour tester votre playbook Ansible modifié.
- 12.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 12.2. Dans la liste des modèles, sélectionnez le modèle **Run ad project**.
  - 12.3. Sélectionnez le playbook **domain-mod.yml** dans la liste **PLAYBOOK**.
  - 12.4. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **LIMIT** est activée.
  - 12.5. Sélectionnez l'option **PROMPT ON LAUNCH** du champ **CREDENTIAL**.
  - 12.6. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
  - 12.7. Dans la fenêtre **RUN AD PROJECT**, choisissez les informations d'identification **AD Administrator** et cliquez sur **NEXT**.
  - 12.8. Dans le champ de texte **LIMIT**, saisissez le nom d'hôte **windc.example.com**, puis cliquez sur **NEXT**.
  - 12.9. Dans le champ **ACTIVE DIRECTORY ADMINISTRATOR PASSWORD**, saisissez **RedHat123@!**, c'est-à-dire le mot de passe de l'administrateur Active Directory.  
Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 12.10. Observez la sortie en direct de la tâche en cours d'exécution.
  - 12.11. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- ▶ 13. Modifiez le modèle de tâche **Run ad Project** et désactivez l'enquête.
- 13.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 13.2. Dans la liste des modèles, sélectionnez **Run ad project**.



#### Note

Pour sélectionner un projet à modifier, cliquez sur son nom.

13.3. Cliquez sur **EDIT SURVEY**.

13.4. Dans la partie supérieure de la fenêtre de l'éditeur de sondage, basculez **ON/OFF** et définissez-le sur **OFF**.

13.5. Cliquez sur **SAVE** pour mettre à jour le sondage

► 14. Cliquez sur **Log Out** pour quitter l'interface Web d'Ansible Tower.

L'exercice guidé est maintenant terminé.

# Génération d'inventaires dynamiques à partir d'Active Directory

## Résultats

Au terme de cette section, vous serez en mesure de générer un inventaire Ansible de manière dynamique dans Red Hat Ansible Tower, en fonction de l'appartenance à un domaine Active Directory.

## Inventaires dynamiques

Lorsqu'Ansible exécute un playbook, il utilise un inventaire pour déterminer les hôtes sur lesquels les plays doivent s'exécuter. Ansible et Ansible Tower facilitent tous deux la définition d'un inventaire statique des hôtes que vous spécifiez explicitement dans un inventaire.

Cependant, ces listes statiques nécessitent une administration manuelle pour les maintenir à jour. Cela peut s'avérer peu pratique ou difficile, en particulier lorsqu'une organisation souhaite exécuter les playbooks sur des hôtes qui sont créés de manière dynamique dans un environnement de virtualisation ou de Cloud Computing.

Dans ce scénario, l'utilisation d'un *inventaire dynamique* est préférable. Les inventaires dynamiques sont des scripts qui, lorsqu'ils sont exécutés, déterminent de manière dynamique les hôtes et les groupes d'hôtes qui doivent figurer dans l'inventaire en fonction des informations provenant d'une source externe. Les sources externes peuvent inclure l'API pour les fournisseurs de cloud, Cobbler, les annuaires LDAP ou d'autres logiciels de base de données de gestion de la configuration (CMDB) tiers. L'utilisation d'un inventaire dynamique est une pratique recommandée dans un environnement informatique de grande envergure et qui évolue rapidement, où les systèmes sont fréquemment déployés, testés, puis supprimés.

Par défaut, Ansible Tower est doté d'un support d'inventaire dynamique intégré pour un certain nombre de sources d'inventaire externes (ou de sources d'inventaire cloud), notamment :

- Amazon Web Services EC2
- Google Compute Engine (GCE)
- Microsoft Azure Resource Manager
- VMware vCenter
- Red Hat Satellite 6
- Red Hat CloudForms
- Red Hat Virtualization
- OpenStack

De plus, les scripts d'inventaire dynamique personnalisés peuvent demander à Ansible Tower d'accéder à des informations d'inventaire à partir d'autres sources. Ansible Tower peut récupérer ces scripts à partir d'un projet qui utilise un référentiel tel que Git en tant que source.

Le reste de cette section examine deux exemples de configuration d'inventaire dynamique dans Ansible Tower. Le premier exemple illustre la prise en charge intégrée de Microsoft Azure

Resource Manager. Le deuxième exemple examine la manière dont vous pouvez utiliser des scripts d'inventaire dynamique personnalisés avec Active Directory.

## Inventaires dynamiques Microsoft Azure Resource Manager

Les technologies de cloud telles que Microsoft Azure apportent de nombreuses modifications au cycle de vie du serveur. Les hôtes changent au fil du temps, tandis que les applications externes créent et lancent de nouveaux hôtes. Maintenir un fichier d'inventaire statique précis est une véritable gageure, quelle que soit la durée. Par conséquent, une mise à jour dynamique de l'inventaire, basée sur les informations fournies directement à partir de Microsoft Azure Resource Manager, est très utile.

Le processus de base permettant de configurer des inventaires dynamiques est similaire lors de l'utilisation de l'une des sources de cloud intégrées :

1. Créez des informations d'identification pour vous authentifier auprès de la source de données cloud que vous envisagez d'utiliser ; le type d'informations d'identification doit correspondre à la source.
2. Créez un inventaire pour obtenir des informations sur l'inventaire dynamique.
3. Dans le nouvel inventaire, créez une **Source** avec l'une des sources d'inventaires dynamiques intégrées (au lieu de **Manual**). Utilisez les nouvelles informations d'identification pour vous authentifier auprès de cette source. Vous pouvez également définir d'autres options, comme la mise à jour automatique au lancement.
4. Mettez à jour la source de l'inventaire pour la première fois.

Ce processus fonctionne pour les inventaires dynamiques basés sur Microsoft Azure. Figure 8.1 montre comment créer des informations d'identification à utiliser par un inventaire dynamique Microsoft Azure Resource Manager, y compris les informations que vous devez indiquer :

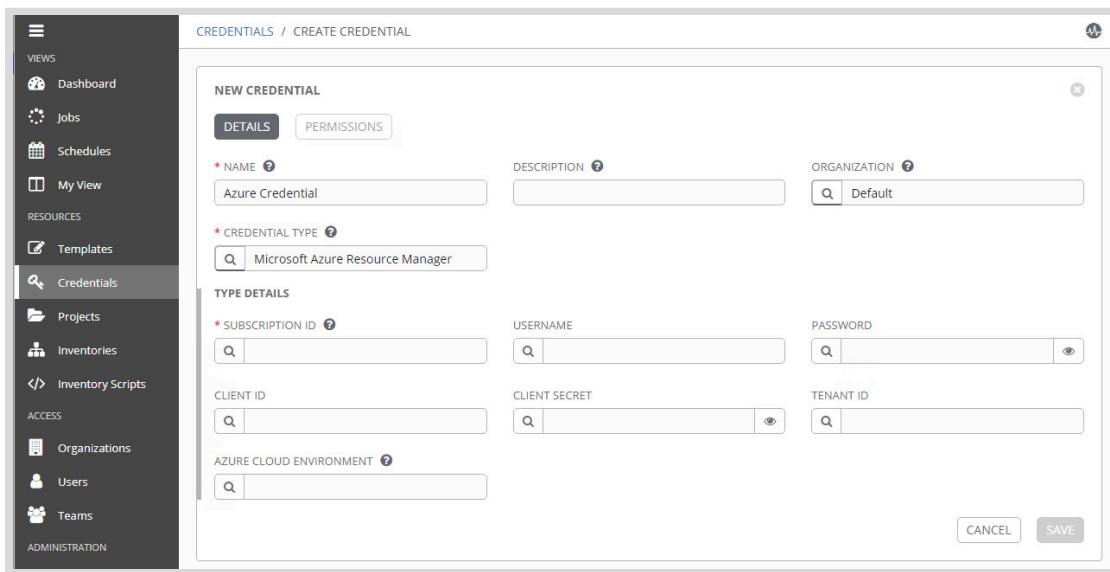


Figure 8.1: Création d'informations d'identification du cloud

Vous reconnaîtrez certains de ces éléments de votre utilisation d'autres objets dans Ansible Tower, y compris **Name**, **Description** et **Organization**. Le seul nouvel élément est **Credential Type**. Vous devez choisir le type d'informations d'identification approprié pour le produit que vous utilisez. Dans cet exemple, le produit est **Microsoft Azure Resource Manager**.

## chapitre 8 | Interaction avec les utilisateurs et les domaines

Les informations d'identification Microsoft Azure Resource Manager nécessitent des informations supplémentaires :

### Username

L'utilisateur qui peut se connecter au compte Microsoft Azure.

### Password

Le mot de passe de l'utilisateur.

### Subscription ID (required)

L'UUID d'abonnement dans une construction Azure, qui est mis en correspondance avec un nom d'utilisateur.

### Client ID

ID du client Azure.

### Client Secret

Le secret du client Azure.

### Tenant ID

L'ID du tenant client Azure.

### Azure Cloud Environment

La variable associée aux environnements cloud Azure.

Le mécanisme de synchronisation Ansible Tower utilise ces informations d'identification que vous venez de créer. Après avoir créé les informations d'identification, sélectionnez **Inventories** dans le volet de navigation. Comme illustré dans Figure 8.2, vous devez créer un nouvel inventaire :

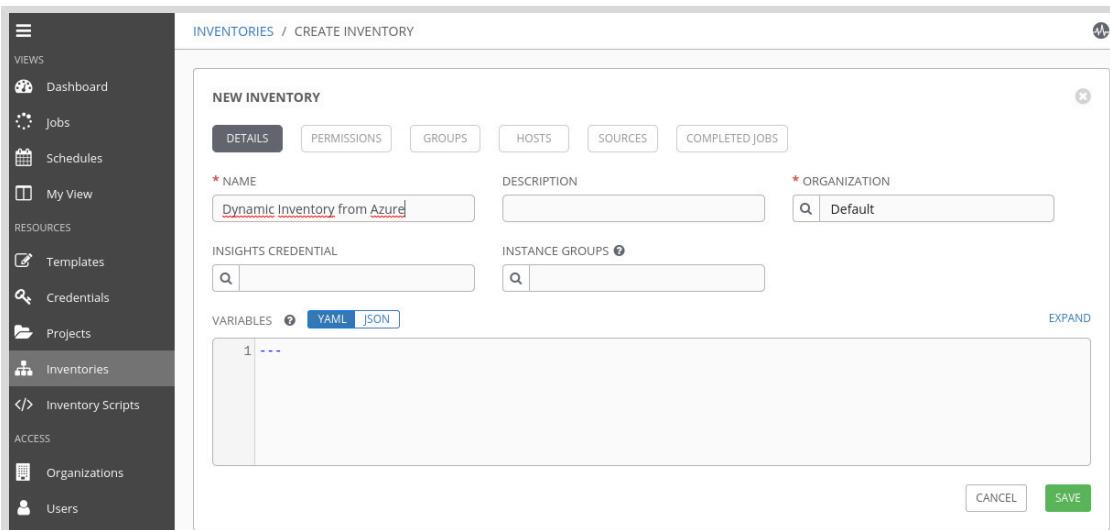


Figure 8.2: Création d'un inventaire cloud

Donnez au nouvel inventaire un nom unique et affectez-le à une organisation existante. Après avoir enregistré la configuration du nouvel inventaire, cliquez sur **SOURCES** pour créer une source pour l'inventaire. Ansible Tower utilise cette source en conjonction avec les scripts Microsoft Azure Resource Manager existants et les informations d'identification créées précédemment. Figure 8.3 montre un exemple de ce type de source :

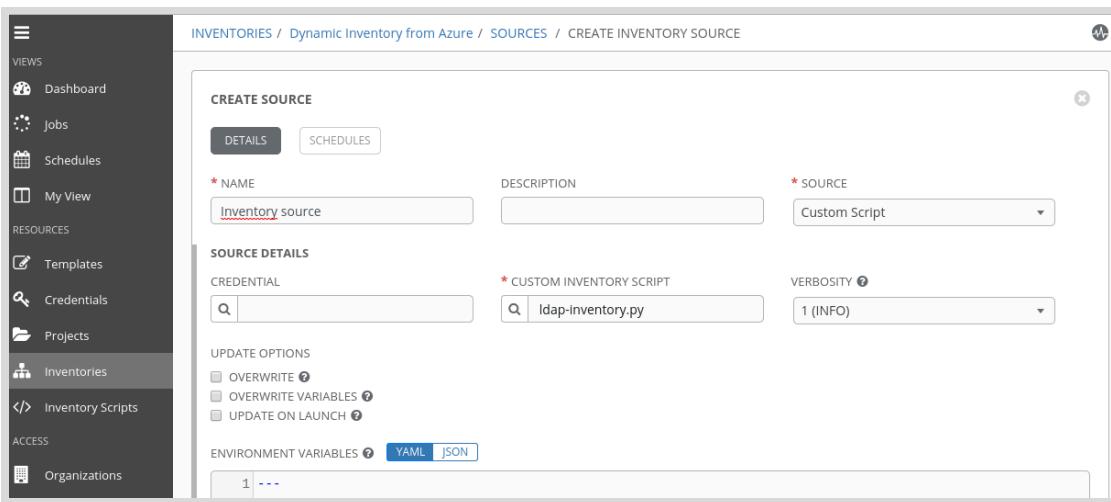


Figure 8.3: Création d'une source de cloud

Cette nouvelle source utilise la prise en charge d'Ansible Tower intégrée pour Microsoft Azure Resource Manager en tant que **SOURCE** et les nouvelles informations d'identification pour votre environnement Microsoft Auzre en tant que **CREDENTIAL**. Vous avez le choix entre trois options **UPDATE OPTIONS** :

#### Overwrite

Si cette option est sélectionnée, le processus de mise à jour de l'inventaire supprime tous les groupes enfants et les hôtes de l'inventaire local qui ne se trouvent pas dans la source externe. Par défaut, cette option n'est pas active, ce qui signifie que tous les hôtes et les groupes enfants qui ne se trouvent pas dans la source externe restent en l'état.

#### Overwrite Variables

Si cette option n'est pas sélectionnée, alors Ansible Tower fusionne les variables locales avec celles trouvées sur la source externe. Ansible Tower supprime toutes les variables introuvables sur la source externe.

#### Update on Launch

Si cette option est activée, alors chaque fois qu'une tâche est exécutée à l'aide de cet inventaire, Ansible Tower actualise dernier est actualisé à partir de la source externe, avant l'exécution des tâches du projet.

Dans la liste source de l'inventaire, la petite icône du Cloud située à gauche du nom de la source affiche l'état de la synchronisation de l'inventaire dynamique pour cette source. Lorsqu'elle est désactivée, aucun statut n'est disponible. Pour démarrer le processus de synchronisation, cliquez sur le bouton de synchronisation. Une fois la synchronisation terminée, l'icône de statut du cloud devient verte si la synchronisation s'est effectuée correctement ou rouge si elle a échoué.

Une fois la synchronisation avec la source externe correctement effectuée, examinez les groupes enfants et les hôtes créés dans Ansible Tower à l'aide des informations de la source externe.

Les groupes enfants contiennent les hôtes visibles dans les listes **HOSTS**. Cliquez sur le nom du groupe afin d'afficher le contenu de ce groupe, puis passez en revue chaque groupe enfant y compris une liste des hôtes associés. Cet inventaire est mis à jour chaque fois que vous le synchronisez avec la source externe. Vous pouvez effectuer la synchronisation manuellement, la programmer à l'aide des mécanismes d'Ansible Tower, ou la définir pour qu'elle se mette à jour automatiquement à chaque exécution d'une tâche à l'aide de cet inventaire.

## Scripts d'inventaire dynamique personnalisés

Ansible vous permet d'écrire des scripts personnalisés pour générer un inventaire dynamique. Bien qu'Ansible Tower offre une prise en charge intégrée d'un certain nombre de sources d'inventaire dynamique, Ansible Tower accepte également les scripts d'inventaire dynamique personnalisés.

### Écriture ou obtention de scripts d'inventaire personnalisés

Ansible Tower prend en charge les scripts d'inventaire personnalisés écrits en Bash ou Python. Ces scripts sont exécutés en tant qu'utilisateur **awx** et disposent d'un accès limité au serveur Ansible Tower. Le script doit commencer par une ligne Shebang appropriée (par exemple, **#!/usr/bin/python** pour un script Python).

De nombreux exemples de scripts d'inventaire personnalisés à utiliser avec diverses sources externes ont été fournis par la communauté, le référentiel Git pour Ansible à l'adresse <https://github.com/ansible/ansible/tree-devel/contrib/inventory/>.

Si vous souhaitez écrire votre propre script d'inventaire personnalisé, vous trouverez des informations sur Développement de sources d'inventaire dynamique [[http://docs.ansible.com/ansible/dev\\_guide/developing\\_inventory.html](http://docs.ansible.com/ansible/dev_guide/developing_inventory.html)] dans le manuel *Ansible Developer Guide*. Lorsque vousappelez le script d'inventaire dynamique avec l'option **--list**, il doit sortir l'inventaire au format JSON.

Voici un exemple de sortie provenant d'un script d'inventaire dynamique personnalisé :

```
{
  "databases" : {
    "vars" : {
      "example_db" : true
    },
    "hosts" : [
      "db1.demo.example.com",
      "db2.demo.example.com"
    ]
  },
  "webservers" : [
    "web1.demo.example.com",
    "web2.demo.example.com"
  ],
  "boston" : {
    "children" : [
      "backup",
      "ipa"
    ],
    "vars" : {
      "example_host" : false
    },
    "hosts" : [
      "server1.demo.example.com",
      "server2.demo.example.com",
      "server3.demo.example.com"
    ]
  },
  "backup" : [
    "server4.demo.example.com"
  ]
}
```

```

    ],
    "ipa" : [
        "server5.demo.example.com"
    ]
}

```

Comme illustré dans l'exemple précédent, chaque groupe contient une liste d'hôtes, de groupes enfants potentiels, de variables de groupe possibles ou une liste d'hôtes.



### Note

Lorsqu'il est appelé avec l'option `--host hostname`, le script imprime un hachage/dictionnaire JSON des variables pour l'hôte spécifié (potentiellement, un hachage ou un dictionnaire JSON vide si aucune variable n'est fournie).

Éventuellement, si l'option `--list` renvoie un élément de niveau supérieur appelé `_meta`, toutes les variables d'hôte peuvent être renvoyées dans un appel de script, ce qui améliore les performances des scripts. Dans ce cas, les appels `--host` ne se font pas.

Pour plus d'informations, reportez-vous à la documentation citée précédemment dans la section *Developing Dynamic Inventory Sources* du manuel *Ansible Developer Guide*.

## Utilisation de scripts d'inventaire personnalisés dans Ansible Tower

Après avoir créé ou téléchargé le script d'inventaire personnalisé approprié, vous devez l'importer dans Ansible Tower et configurer l'inventaire. Figure 8.4 montre comment cette tâche est accomplie :

Pour télécharger un script d'inventaire personnalisé dans Ansible Tower, sélectionnez **Inventories Scripts** dans le volet de navigation de l'interface web d'Ansible Tower. Cliquez sur le bouton **+** pour ajouter un script d'inventaire personnalisé.

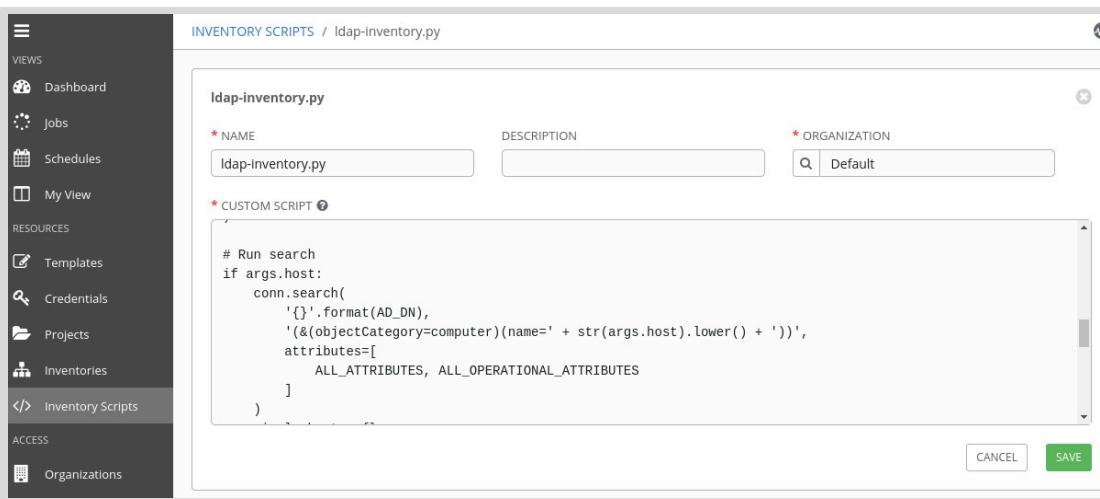


Figure 8.4: Script d'inventaire d'Ansible Tower

**chapitre 8 |** Interaction avec les utilisateurs et les domaines

Définissez un nouveau nom pour le script d'inventaire personnalisé dans le champ **NAME**, sélectionnez une organisation dans le champ **ORGANIZATION**, puis collez le script dans la zone de texte **CUSTOM SCRIPT**. Cliquez sur le bouton **SAVE**.

Après avoir importé et défini le script d'inventaire dynamique dans Ansible Tower, configuez-le exactement comme n'importe quel inventaire dynamique intégré :

1. Créez une source dans un inventaire existant pour l'inventaire dynamique. Définissez **SOURCE** sur **Custom Script** et indiquez le nom du script personnalisé dans le champ **CUSTOM INVENTORY SCRIPT**. Il s'agit du nom que vous avez donné au script lors de son importation dans Ansible Tower.
2. Synchronisez la source avec la source d'inventaire, par exemple à partir de votre serveur Active Directory.

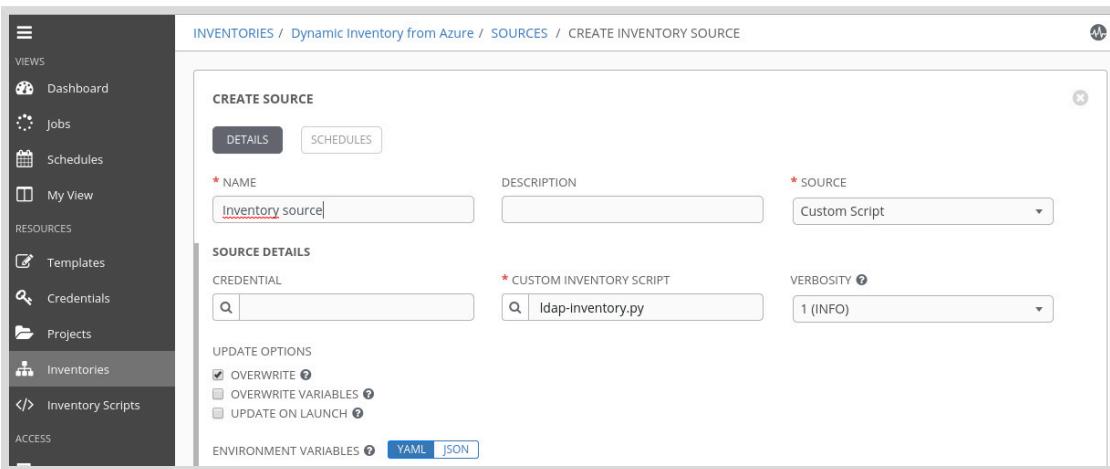


Figure 8.5: Ajout d'un nouveau script personnalisé en tant que source pour un inventaire dynamique



### Références

#### Guide de l'utilisateur d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/userguide/>

#### Guide d'administration d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/administration/>

## ► Exercice guidé

# Génération d'inventaires dynamiques à partir d'Active Directory

Dans cet exercice, vous allez configurer votre système Red Hat Ansible Tower avec un inventaire dynamique généré du fait de l'appartenance à un domaine Active Directory.

## Résultats

Vous devez être en mesure d'ajouter un script d'inventaire personnalisé et de l'utiliser pour remplir un inventaire dynamique dans Tower.

## Avant De Commencer



### Important

Cet exercice utilise des ressources d'Ansible Tower provenant de l'exercice guidé précédent. Avant de commencer cet exercice, vous devez effectuer l'exercice précédent.

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Utilisez le mot de passe unique affiché sous l'onglet **Online Lab** de l'interface ROL pour vous connecter à **workstation** via le protocole RDP.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. À partir de **workstation**, cliquez sur l'icône **Ansible Tower** sur votre Bureau pour accéder à Ansible Tower.  
Connectez-vous à la console Web en tant qu'**admin** avec le mot de passe **RedHat123@!**.
- ▶ 2. Ajoutez le script d'inventaire personnalisé **ldap-inventory.py** dans Ansible Tower.
  - 2.1. Cliquez sur **Inventory Scripts** dans le volet de navigation.
  - 2.2. Cliquez sur **+** pour ajouter un script d'inventaire personnalisé.
  - 2.3. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
NAME	<b>ldap-inventory.py</b>
DESCRIPTION	Script d'inventaire dynamique pour Active Directory
ORGANIZATION	<b>Valeur par défaut</b>

- 2.4. Lancez l'éditeur Visual Studio Code, accédez à **File → Open Folder**, puis sélectionnez le dossier **C:\Users\student\Documents\ad** pour ouvrir le référentiel **ad**.

Copiez le contenu du script **ldap-inventory.py** dans le champ **CUSTOM SCRIPT**.

- 2.5. Cliquez sur **SAVE** pour ajouter le script d'inventaire personnalisé.

► **3.** Créez l'inventaire **Dynamic inventory**.

- 3.1. Cliquez sur **Inventories** dans le volet de navigation.

- 3.2. Cliquez sur **+** pour ajouter un nouvel inventaire, puis sélectionnez **Inventory**.

- 3.3. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
NAME	<b>Dynamic inventory</b>
DESCRIPTION	Inventaire dynamique du serveur AD
ORGANIZATION	<b>Valeur par défaut</b>
VARIABLES	<ul style="list-style-type: none"> <li>• <b>ansible_port: 5986</b></li> <li>• <b>ansible_connection: winrm</b></li> <li>• <b>ansible_winrm_server_cert_validation: ignore</b></li> <li>• <b>ansible_winrm_transport: kerberos</b></li> </ul>

- 3.4. Cliquez sur **SAVE** pour créer l'inventaire.

► **4.** Ajoutez le script **ldap-inventory.py** en tant que nouvelle source de l'inventaire.

- 4.1. Cliquez sur **SOURCES**.

- 4.2. Cliquez sur **+** pour ajouter une nouvelle source.

- 4.3. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
NAME	<b>Custom Script</b>
DESCRIPTION	Script personnalisé pour l'inventaire dynamique
SOURCE	<b>Custom Script</b>
CUSTOM INVENTORY SCRIPT	<b>ldap-inventory.py</b>

- 4.4. Pour **UPDATE OPTIONS**, sélectionnez l'option **OVERWRITE**. Cette option indique à Ansible Tower d'écraser les hôtes s'ils figurent déjà dans l'inventaire.

- 4.5. Cliquez sur **SAVE** pour créer la source.

- ▶ 5. Mettez à jour l'inventaire dynamique.
- 5.1. Faites défiler vers le bas, cliquez sur le bouton de synchronisation de **Custom Script**, puis attendez que le cloud devienne vert et statique.
  - 5.2. Cliquez sur **HOSTS** pour vous assurer que l'inventaire contient maintenant quatre hôtes, tous associés à un nouveau groupe **windows**.
- ▶ 6. Revenez à Visual Studio Code, puis modifiez le fichier **add-computers.yml**. Modifiez ce fichier de façon à ce qu'il cible tous les hôtes et qu'il utilise le module **win\_domain\_membership** pour joindre le serveur **win2.example.com** au domaine AD existant **example.com**. Une fois que vous avez terminé, validez et poussez vos modifications.
- 6.1. Sélectionnez le fichier **add-computers.yml**, puis modifiez le nom de la tâche et les détails nécessaires pour qu'un serveur rejoigne le domaine **example.com**.

```
...output omitted...
- name: A new Linux host is added to the "{{ domainname }}" AD domain
  win_domain_computer:
    name: LINUX1
    dns_hostname: linux1.{{ dnsname }}
    ou: "CN=Computers,DC=example,DC=com"
    description: Linux server example
    enabled: yes
    state: present

- name: A new Windows host is added to the "{{ domainname }}" AD domain
  win_domain_computer:
    name: WIN_TEST
    sam_account_name: win_test$  

    dns_hostname: win_test.{{ dnsname }}
    ou: "CN=Computers,DC=example,DC=com"
    description: Windows server example
    state: present
```



#### Mise en garde

Assurez-vous que le nom de l'option **sam\_account\_name** : se termine par un signe **\$**. Les comptes SAM doivent toujours se terminer par le signe **\$**.

- 6.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **master**, ce qui indique que des modifications ont été apportées.
- 6.3. Passez à l'affichage **Source Control**, puis cliquez sur **+** en regard de **add-computers.yml** pour effectuer les modifications.
- 6.4. Saisissez le message de validation **Define configuration for joining Linux server to the AD domain**, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 6.5. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales

au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.

À l'invite, cliquez sur **Yes** ou **Ask Me Later**.

- 7. Basculez vers l'interface web d'Ansible Tower, puis utilisez le modèle de tâche **Run ad project** pour tester votre playbook Ansible modifié.

- 7.1. Dans le volet de navigation, cliquez sur **Templates**.
- 7.2. Dans la liste des modèles, sélectionnez **Run ad project**.



### Note

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 7.3. Sélectionnez le playbook **add-computers.yml** dans la liste **PLAYBOOK**.
- 7.4. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **LIMIT** est activée.
- 7.5. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **CREDENTIAL** est activée.
- 7.6. Activez la case à cocher **PROMPT ON LAUNCH** pour l'option **INVENTORY**.

- 7.7. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
- 7.8. Dans la fenêtre **RUN AD PROJECT**, sélectionnez **Dynamic inventory**. Cliquez sur **NEXT**.
- 7.9. Dans la liste des informations d'identification disponibles, choisissez les informations d'identification **AD Administrator**. Cliquez sur **NEXT**.

**Note**

Les informations d'identification **AD Administrator** ont été créées au cours de l'exercice précédent. Si vous ne disposez pas de ces informations d'identification, suivez les instructions de l'exercice guidé précédent (Étape 11) pour les créer.

- 7.10. Dans le champ **LIMIT**, saisissez le nom d'hôte **windc.example.com**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 7.11. Dans le champ **ACTIVE DIRECTORY ADMINISTRATOR PASSWORD**, saisissez **RedHat123@!**, c'est-à-dire le mot de passe de l'administrateur Active Directory. Ce mot de passe est identique au compte administrateur de **windc**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 7.12. Observez la sortie en direct de la tâche en cours d'exécution. L'état de la tâche change une fois que le serveur a redémarré avec succès.
  - 7.13. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- 8. Synchronisez la liste d'hôtes d'Active Directory à l'aide de l'inventaire dynamique et vérifiez que les deux nouveaux hôtes figurent dans l'inventaire.
- 8.1. Cliquez sur **Inventories** dans le volet de navigation.
  - 8.2. Cliquez sur **Dynamic inventory**.
  - 8.3. Cliquez sur **SOURCES**.
  - 8.4. Cliquez sur le bouton de synchronisation de **Custom Script**, puis attendez que le cloud devienne vert et statique.
  - 8.5. Cliquez sur **HOSTS**, puis vérifiez que les hôtes **linux1.example.com** et **win\_test.example.com** ajoutés dans Active Directory figurent désormais dans l'inventaire.

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Interaction avec les utilisateurs et les domaines

### Liste de contrôle des performances

Dans cet atelier, vous allez ajouter des utilisateurs et des groupes, qu'ils soient locaux ou basés sur un domaine.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Ajouter deux utilisateurs locaux.
- Ajouter un nouvel utilisateur à un groupe local existant.
- Ajouter un nouvel utilisateur de domaine.
- Ajouter un nouvel utilisateur à un groupe de domaines nouvellement créé.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. Modifiez le playbook **localuser-review.yml** dans le référentiel **ad** et procédez aux modifications indiquées. Validez et poussez vos modifications vers le référentiel Git, en utilisant le message de validation **Adds users and manage group membership**. Les commentaires figurant dans le playbook indiquent les modifications à apporter.
  - Modifiez le playbook pour qu'il cible tous les hôtes.  
Nommez la première tâche **Two local users are created**.
  - Renommez la deuxième tâche **The Review User1 user is added to the Print Operators group**.
  - Mettez à jour la première tâche avec le module Ansible correct et les options du module pour ajouter deux utilisateurs locaux.
  - Mettez à jour la deuxième tâche avec le module Ansible correct et les options du module pour ajouter l'utilisateur **Review User1** au groupe local **Print Operators**.
2. Mettez à jour le modèle de tâche **Run ad project** pour utiliser le playbook Ansible **localuser-review.yml** modifié. Mettez à jour le modèle pour indiquer à Ansible Tower

de vous demander l'hôte à cibler, ainsi que les informations d'identification et l'inventaire à utiliser.

Lancez une tâche à l'aide du modèle et, lorsque vous y êtes invité, utilisez **win1.example.com** comme hôte cible, **DevOps** comme informations d'identification et **Default inventory** comme inventaire.

3. À partir de **workstation**, modifiez le playbook **domain-mod-review.yml** dans le projet **ad** dans Visual Studio Code. Mettez à jour le playbook de façon à ce qu'Ansible crée un groupe de domaines, et un utilisateur de domaine qui est membre de ce nouveau groupe. Assurez-vous que le playbook utilise les modules Ansible et les options de module corrects pour créer le nouveau groupe et le nouvel utilisateur.  
Utilisez la première tâche pour créer un groupe dans le domaine **EXAMPLE..COM**. Utilisez la deuxième tâche pour créer un nouvel utilisateur dans le domaine. L'utilisateur doit être membre du groupe nouvellement créé, ainsi que du groupe **Domain Users**.  
Validez et poussez vos modifications vers le référentiel Git distant à l'aide du message de validation **Create new group in domain and user**.
4. Mettez à jour le modèle de tâche **Run ad project** dans Ansible Tower pour utiliser le playbook Ansible **domain-mod-review.yml**. Mettez à jour le modèle de tâche afin qu'il vous invite au lancement à fournir les informations d'identification, l'inventaire et l'hôte à cibler.  
Lancez une tâche à l'aide du modèle et, lorsque vous y êtes invité, sélectionnez **Dynamic inventory** comme inventaire, **AD Administrator** comme informations d'identification et **windc.example.com** comme hôte.



#### Note

Vous avez créé les informations d'identification **AD Administrator** au cours d'un exercice précédent. Si vous ne disposez pas de ces informations d'identification, suivez les instructions dans l'exercice guidé *Managing Active Directory Domains* pour les créer. Les étapes sont également affichées dans la solution pour cet atelier.

L'atelier est maintenant terminé.

## ► Solution

# Interaction avec les utilisateurs et les domaines

### Liste de contrôle des performances

Dans cet atelier, vous allez ajouter des utilisateurs et des groupes, qu'ils soient locaux ou basés sur un domaine.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Ajouter deux utilisateurs locaux.
- Ajouter un nouvel utilisateur à un groupe local existant.
- Ajouter un nouvel utilisateur de domaine.
- Ajouter un nouvel utilisateur à un groupe de domaines nouvellement créé.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. Modifiez le playbook **localuser-review.yml** dans le référentiel **ad** et procédez aux modifications indiquées. Validez et poussez vos modifications vers le référentiel Git, en utilisant le message de validation **Adds users and manage group membership**. Les commentaires figurant dans le playbook indiquent les modifications à apporter.
  - Modifiez le playbook pour qu'il cible tous les hôtes.  
Nommez la première tâche **Two local users are created**.
  - Renommez la deuxième tâche **The Review User1 user is added to the Print Operators group**.
  - Mettez à jour la première tâche avec le module Ansible correct et les options du module pour ajouter deux utilisateurs locaux.
  - Mettez à jour la deuxième tâche avec le module Ansible correct et les options du module pour ajouter l'utilisateur **Review User1** au groupe local **Print Operators**.
- 1.1. Lancez l'éditeur Visual Studio Code. Accédez à **File → Open Folder**, puis sélectionnez le dossier **c:\Users\student\Documents\ad** pour ouvrir le répertoire **c:\Users\student\Documents\ad**.

1.2. Modifiez le playbook **localuser-review.yml**, en le modifiant comme suit :

- Le playbook doit viser tous les hôtes.
- La première tâche doit utiliser le modèle **win\_user** pour créer les utilisateurs **Review User1** et **Review User2**. Définissez **RedHat123@!** comme mot de passe.
- La deuxième tâche doit utiliser le module **win\_group\_membership** pour ajouter **Review User1** au groupe **Print operators**.

Le playbook terminé doit se présenter comme suit :

```
---
hosts: all

tasks:
  - name: Two local users are created
    win_user:
      name: "{{ item }}"
      password: RedHat123@!
      state: present
      groups:
        - Users
    loop:
      - Review User1
      - Review User2

  - name: The Review User1 user is added to the Print operators group
    win_group_membership:
      name: Print operators
      members:
        - Review User1
      state: present
```

- 1.3. Enregistrez vos modifications. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master** pour indiquer que de nouvelles modifications ont été apportées
- 1.4. Passez à l'affichage **Source Control**, puis cliquez sur **+** en regard du playbook **localuser-review.yml** pour effectuer les modifications.
- 1.5. Entrez le message de validation **Adds users and manage group membership**, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 1.6. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
2. Mettez à jour le modèle de tâche **Run ad project** pour utiliser le playbook Ansible **localuser-review.yml** modifié. Mettez à jour le modèle pour indiquer à Ansible Tower de vous demander l'hôte à cibler, ainsi que les informations d'identification et l'inventaire à utiliser.

**chapitre 8 |** Interaction avec les utilisateurs et les domaines

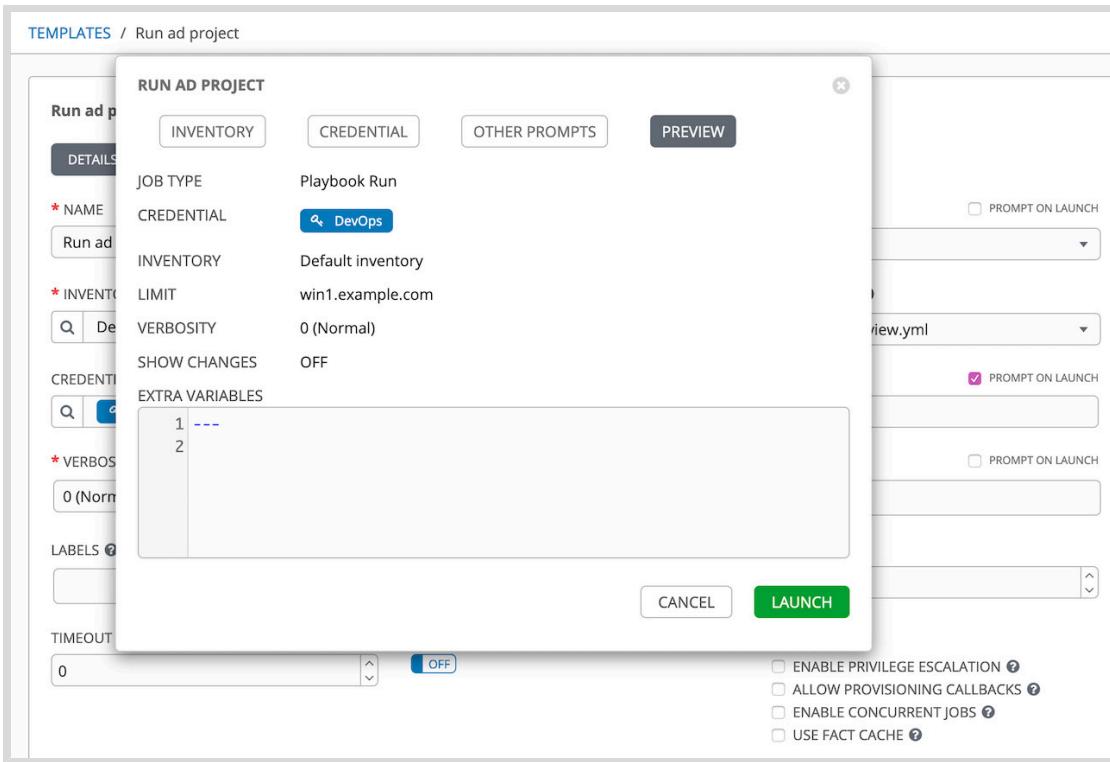
Lancez une tâche à l'aide du modèle et, lorsque vous y êtes invité, utilisez **win1.example.com** comme hôte cible, **DevOps** comme informations d'identification et **Default inventory** comme inventaire.

- 2.1. Double-cliquez sur l'icône de Bureau Ansible Tower pour accéder à Ansible Tower.  
Connectez-vous en utilisant **admin** avec le mot de passe **RedHat123@!**.
- 2.2. Dans le volet de navigation, cliquez sur **Templates**.
- 2.3. Dans la liste des modèles, sélectionnez le modèle **Run ad project**.

**Note**

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 2.4. Sélectionnez le playbook **localuser-review.yml** dans la liste **PLAYBOOK**.
- 2.5. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **LIMIT** est activée.
- 2.6. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **CREDENTIAL** est activée.
- 2.7. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **INVENTORY** est activée.
- 2.8. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
- 2.9. Dans la fenêtre **RUN AD PROJECT**, choisissez **Default inventory**, puis cliquez sur **NEXT**.
- 2.10. Dans la fenêtre **RUN AD PROJECT**, choisissez les informations d'identification **DevOps**, puis cliquez sur **NEXT**.
- 2.11. Dans le champ **LIMIT**, saisissez **win1.example.com** en tant que nom d'hôte, cliquez sur **NEXT**, puis cliquez sur **LAUNCH**.



2.12. Observez la sortie en direct de la tâche en cours d'exécution.

2.13. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.

3. À partir de **workstation**, modifiez le playbook **domain-mod-review.yml** dans le projet **ad** dans Visual Studio Code. Mettez à jour le playbook de façon à ce qu'Ansible crée un groupe de domaines, et un utilisateur de domaine qui est membre de ce nouveau groupe. Assurez-vous que le playbook utilise les modules Ansible et les options de module corrects pour créer le nouveau groupe et le nouvel utilisateur.

Utilisez la première tâche pour créer un groupe dans le domaine **EXAMPLE..COM**. Utilisez la deuxième tâche pour créer un nouvel utilisateur dans le domaine. L'utilisateur doit être membre du groupe nouvellement créé, ainsi que du groupe **Domain Users**.

Validez et poussez vos modifications vers le référentiel Git distant à l'aide du message de validation **Create new group in domain and user**.

- 3.1. À partir de Visual Studio Code, ouvrez le playbook **domain-mod-review.yml** dans le projet **ad** et mettez-le à jour. Utilisez le module **win\_domain\_group** pour la première tâche, afin de créer un groupe dans le domaine **EXAMPLE.COM**.

```
---
- name: Creating AD groups and users
  hosts: all
  vars:
    domainname: EXAMPLE.COM
    user_password: RedHat123@!
```

```
tasks:
  - name: New group in the "{{ domainname }}" is created
    win_domain_group:
      name: Review Group
      scope: global
```

- 3.2. Mettez à jour la deuxième tâche pour qu'elle utilise **win\_domain\_user** afin de créer un nouvel utilisateur dans le domaine. Le paramètre **groups** vous permet de gérer l'appartenance à un groupe pour l'utilisateur.

```
...output omitted...
- name: New user in the "{{ domainname }}" is created
  win_domain_user:
    name: oliver
    firstname: Oliver
    lastname: Stone
    password: "{{ user_password }}"
    state: present
  groups:
    - Domain Users
    - Review Group
```

- 3.3. Enregistrez vos modifications. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été apportées.
- 3.4. Passez à l'affichage **Source Control**, puis cliquez sur **+** en regard de **domain-mod-review.yml** pour effectuer les modifications.
- 3.5. Entrez le message de validation **Create new group in domain and user**, puis appuyez sur **Ctrl+Entrée** pour valider vos modifications.
- 3.6. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
4. Mettez à jour le modèle de tâche **Run ad project** dans Ansible Tower pour utiliser le playbook Ansible **domain-mod-review.yml**. Mettez à jour le modèle de tâche afin qu'il vous invite au lancement à fournir les informations d'identification, l'inventaire et l'hôte à cibler.
- Lancez une tâche à l'aide du modèle et, lorsque vous y êtes invité, sélectionnez **Dynamic inventory** comme inventaire, **AD Administrator** comme informations d'identification et **windc.example.com** comme hôte.



### Note

Vous avez créé les informations d'identification **AD Administrator** au cours d'un exercice précédent. Si vous ne disposez pas de ces informations d'identification, suivez les instructions dans l'exercice guidé *Managing Active Directory Domains* pour les créer. Les étapes sont également affichées dans la solution pour cet atelier.

- 4.1. À partir de **Workstation**, revenez à la console Web d'Ansible tour.

- 4.2. Si vous n'avez pas encore créé les informations d'identification pour **AD Administrator**, cliquez sur **Credentials** dans le volet de navigation.
- 4.3. Cliquez sur le bouton **+** pour ajouter de nouvelles informations d'identification.
- 4.4. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
NAME	Administrateur AD
DESCRIPTION	Informations d'identification d'administrateur AD
ORGANIZATION	Valeur par défaut
CREDENTIAL TYPE	Machine
USERNAME	Administrateur
PASSWORD	Saisissez <b>RedHat123@!</b> , c'est-à-dire le mot de passe de l'administrateur Active Directory.

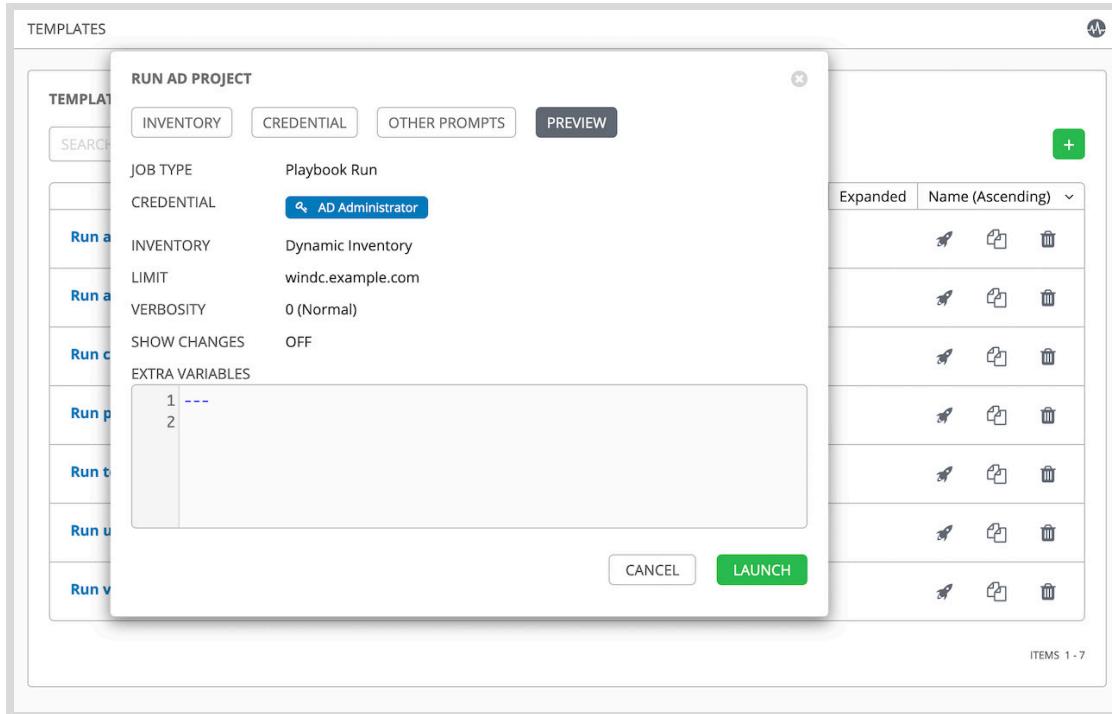
- 4.5. Ne modifiez pas les autres champs et cliquez sur **SAVE** pour créer les informations d'identification.
- 4.6. Dans le volet de navigation, cliquez sur **Templates**.
- 4.7. Dans la liste des modèles, sélectionnez le modèle **Run ad project**.



#### Note

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 4.8. Sélectionnez le playbook **domain-mod-review.yml** dans la liste **PLAYBOOK**.
- 4.9. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **LIMIT** est activée.
- 4.10. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **CREDENTIAL** est activée.
- 4.11. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **INVENTORY** est activée.
- 4.12. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
- 4.13. Dans la fenêtre **RUN AD PROJECT**, choisissez **Dynamic inventory**, puis cliquez sur **NEXT**.
- 4.14. Sélectionnez les informations d'identification **AD Administrator**, puis cliquez sur **NEXT**.
- 4.15. Dans le champ **LIMIT**, saisissez le nom d'hôte **windc.example.com**, puis cliquez sur **NEXT**.



- 4.16. Observez la sortie en direct de la tâche en cours d'exécution.
- 4.17. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- 4.18. Cliquez sur **Log Out** pour quitter l'interface Web d'Ansible Tower.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Vous pouvez utiliser le module Ansible **win\_user** pour gérer les comptes d'utilisateurs locaux, y compris les mots de passe et les groupes.
- Vous pouvez créer et modifier des groupes locaux et de domaine avec les modules **win\_group** et **win\_domain\_group**.
- Vous pouvez utiliser le module **win\_group\_membership** pour gérer les utilisateurs à partir de groupes locaux.
- Vous pouvez utiliser les modules **win\_domain**, **win\_domain\_controller** et **win\_domain\_membership** pour créer des domaines Active Directory et contrôler l'appartenance à un domaine.
- Un *inventaire dynamique* est un inventaire qui met automatiquement à jour sa liste de membres et de groupes à partir d'une source de données externe, telle que l'API d'un service cloud, une base de données ou un autre fournisseur CMDB.
- Red Hat Ansible Tower inclut une prise en charge intégrée de certaines sources d'inventaire dynamiques, et les scripts peuvent être importés pour prendre en charge d'autres.



## chapitre 9

# Automatisation des tâches d'administration Windows

### Objectif

Automatiser les tâches d'administration Windows Server.

### Résultats

- Exécuter des commandes PowerShell arbitraires, configurer des modules PowerShell et configurer des tâches planifiées sur des hôtes gérés avec des playbooks Ansible.
- Automatiser la configuration des périphériques de stockage sur les hôtes gérés.

### Sections

- Exécution des commandes et planification des tâches sur les hôtes (et exercice guidé)
- Configuration et gestion du stockage (et exercice guidé)

### Atelier

Automatisation des tâches d'administration Windows

# Exécution des commandes et planification des tâches sur les hôtes

---

## Résultats

À la fin de cette section, vous devez pouvoir exécuter des commandes PowerShell arbitraires, configurer des modules PowerShell et configurer des tâches planifiées sur des hôtes gérés avec des playbooks Ansible.

## Exécution de commandes directes avec des modules

Ansible fournit une variété de modules pour interagir avec les systèmes Windows. Vous pouvez utiliser des modules conçus spécifiquement pour accomplir une tâche particulière, ou vous pouvez utiliser des modules pour exécuter directement des commandes sur des hôtes gérés.

Par exemple, **win\_user** est un module conçu pour effectuer une tâche particulière : la gestion des utilisateurs locaux. Cependant, il existe un certain nombre de modules que vous pouvez utiliser pour exécuter des commandes à partir de la ligne de commande ou à l'aide de scripts. Dans cette section, nous allons examiner quatre de ces modules :

- **win\_command** exécute une commande, mais n'utilise pas le shell Windows.
- **win\_shell** exécute une commande à l'aide d'un shell Windows et prend en charge l'extension par variable et d'autres fonctions du shell.
- **win\_psexec** exécute une commande à l'aide de **psexec**.
- **script** transfère un script vers l'hôte géré et l'exécute.

## Exécution de commandes arbitraires

Utilisez le module **win\_command** pour exécuter des commandes arbitraires sur les nœuds. Cela revient à exécuter une commande dans l'invite de commande.

```
- name: Run the wbadmin command
  win_command: wbadmin -backupTarget:C:\backup\ ❶
  args:
    chdir: C:\somedir\ ❸
    creates: C:\backup\ ❹
```

- ❶ La commande à exécuter. Cet exemple montre comment appeler **wbadmin** pour créer une sauvegarde du système d'exploitation dans **C:\backup**.
- ❷ Le contexte de la commande, tel que le répertoire dans lequel Ansible exécute la commande.
- ❸ **chdir** ordonne à Ansible d'exécuter la commande à partir du répertoire spécifié. Cela revient à exécuter **cd** avant d'exécuter la commande.
- ❹ **creates** ordonne à Ansible de vérifier si la ressource spécifiée existe avant d'exécuter la commande. Dans cet exemple, Ansible exécute la commande uniquement si **C:\backup** elle n'est pas présente sur le système.

## Exécution de commandes par le biais d'un shell

Utilisez le module **win\_shell** pour exécuter des commandes par le biais d'un shell qui est par défaut PowerShell. Ce module vous permet de sélectionner le shell qui exécute la commande et un contexte pour celui-ci, tel que le répertoire à partir duquel la commande est exécutée.

```
- name: Test the availability of a server
  win_shell: |
    if (Test-Connection -ComputerName "server-1" -Count 2 -Quiet ) { ❶
      Write-Output "[OK]"
    } else {
      Write-Output "[FAIL]"
    }
```

- ❶ Cela démontre comment vous pouvez transmettre plusieurs lignes au module. Dans cet exemple, Ansible utilise la fonction PowerShell **Test-Connection** pour envoyer des paquets ICMP au serveur **server-1**.

## Exécution de commandes en tant qu'utilisateur avec des autorisations élevées

Utilisez le module **win\_psexec** pour exécuter des commandes distantes en tant qu'utilisateur disposant d'autorisations élevées. Le module appelle **psexec** sur les hôtes gérés et exécute les commandes que vous définissez dans le module. Ce module comporte de nombreuses options, notamment les options de commande **psexec**, les noms d'hôte à cibler, la priorité de la commande et les informations d'identification à utiliser lors de l'exécution de commandes.



### Important

Le module **win\_psexec** requiert que PsExec soit disponible sur les hôtes gérés Ansible sur lesquels la tâche s'exécute. Cependant, PsExec n'a pas besoin d'être présent sur les serveurs ciblés par PsExec.

De plus, l'utilisation de ce module avec un certain ensemble de privilèges (par exemple, des informations d'identification) suppose que vous avez configuré les privilèges sur les serveurs ciblés par PsExec.

Il s'agit d'un module utile lors de l'exécution d'un ensemble de commandes sur plusieurs hôtes gérés par Ansible, lorsque Ansible ne peut pas accéder directement aux hôtes. Par exemple, si les hôtes se trouvent sur un réseau protégé, ils ne seront peut-être pas directement accessibles. Vous pouvez configurer le logiciel ou mettre à jour la configuration réseau de votre environnement à l'aide de cette commande. Utilisez le paramètre **hostnames** pour définir la liste des serveurs auxquels se connecter.

```
- name: Restart IIS service firewall rule to the web servers
  win_psexec:
    command: > iisreset /noforce ❶
    hostnames: ❷
      - web-001
      - web-002
      - web-003
    elevated: yes ❸
```

```
username: demo
password: password
priority: realtime ④
```

- ➊ Ordonne à Ansible de se connecter à chaque serveur de la liste et d'exécuter la commande. Cet exemple est équivalent aux commandes suivantes :

```
psexec \\web-001 -u demo -p D3m0$ iisreset /noforce
psexec \\web-002 -u demo -p password iisreset /noforce
psexec \\web-003 -u demo -p password iisreset /noforce
```

Dans l'exemple, le mot de passe est défini dans la tâche, mais Ansible Tower peut inviter l'utilisateur à saisir une valeur pour la variable **password**.

- ➋ Définit une liste d'hôtes à cibler. Ansible se connecte à l'hôte géré, puis exécute PsExec sur les hôtes définis dans la liste. Vous pouvez utiliser un nom DNS ou une adresse IP pour vous connecter aux hôtes gérés. Si vous utilisez un nom DNS, l'hôte géré doit être en mesure de résoudre le nom. Pour qu'il s'exécute sur tous les ordinateurs du domaine en cours, utilisez \\*.  
➌ Lorsqu'elle est définie sur **no**, cette option vous permet d'exécuter la commande à distance en tant qu'utilisateur limité, en supprimant les privilèges d'administration du processus ou de la commande. Lorsqu'elle est définie sur **yes**, PsExec exécute la commande à distance en tant qu'utilisateur avec priviléges.  
➍ Utilisez ce paramètre pour définir un niveau de priorité de processus parmi les six niveaux suivants :

- **Realtime**
- **High**
- **Above normal**
- **Normal**
- **Below normal**
- **Low**

## Transfert et exécution de scripts

Transférez un script du nœud de contrôle vers l'hôte géré et exécutez ce script à l'aide du module **script**. Ansible traite le script via le shell distant par défaut, mais vous pouvez spécifier le chemin d'accès à un autre shell, tel que PowerShell.

Comme les autres modules, vous pouvez définir le contexte d'exécution du script, tel que le répertoire dans lequel Ansible doit exécuter le script.

Ce module est utile si vous disposez d'un script auquel le nœud de contrôle peut accéder, mais auquel les hôtes gérés ne peuvent pas. Cela est également utile si vous avez des scripts existants qui ne peuvent pas être convertis en modules Ansible.



### Important

Lorsque vous utilisez Ansible Tower, placez vos scripts dans le référentiel contenant les playbooks à exécuter, puis utilisez un chemin relatif pour accéder aux scripts. Cela permet à Ansible d'accéder aux scripts sans avoir besoin d'accéder à votre instance Ansible Tower et d'y apporter des modifications.

```

- name: Restart services
  script: files/scripts/Windows/restart_services.cmd ①
  args:
    creates: C:\Windows\logs\restarted_services.log ②
    executable: >
      %SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe ③

```

- ① Définit le *chemin d'accès local* à un script ; autrement dit, un chemin sur l'hôte qui exécute Ansible. Dans cet exemple, le chemin local est un répertoire Linux qui contient des scripts Windows dans le référentiel.
- ② Ordonne à Ansible d'annuler l'exécution du script si la ressource que vous définissez dans le paramètre **creates** existe sur le système distant.
- ③ Ordonne à Ansible d'utiliser le shell que vous définissez ici pour exécuter le script.

## Risques liés à l'exécution de commandes directes

Ces quatre modules sont pratiques à utiliser pour l'automatisation, mais utilisez-les avec précaution. Ces modules ne sont pas naturellement idempotents comme d'autres modules Ansible. Étant donné que ces modules peuvent exécuter n'importe quelle commande de manière flexible, ils n'ont pas de code intégré pour vérifier s'il est nécessaire ou sûr de réexécuter la commande.

De plus, Ansible renvoie toujours le statut **changed**, même si la commande n'apporte pas de modification au système. Une stratégie d'atténuation consiste à utiliser **changed\_when** qui vous permet d'implémenter votre propre logique pour décider de ce qui constitue une modification. Les opérations qui ne changent rien sur vos systèmes renvoient le statut **OK**.

En bref, ces modules peuvent avoir des effets inattendus et dangereux. Voici quelques situations possibles dans lesquelles vous pourriez rencontrer ces effets :

- Vous tentez de supprimer un fichier qui n'est plus présent.
- Vous avez apporté des modifications à un fichier auquel une commande fait référence.
- Un composant dépend d'un fichier qui est recréé chaque fois que vous exécutez un playbook, tel qu'un certificat TLS.
- La tâche restaure une base de données à chaque fois que vous exécutez un play.



### Note

Agissez avec précaution lorsque vous utilisez ces modules et choisissez des modules Ansible conçus pour des tâches spécifiques chaque fois que cela est possible. Cela vous permet de gagner du temps en réduisant les besoins en matière de vérification du système avant et après l'exécution des tâches, et rend vos plays plus robustes.

## Comparaison de modules

Le tableau suivant compare et expose les différences des différents modules.

	<b>win_command</b>	<b>win_shell</b>	<b>win_psexec</b>	<b>Script</b>
Les commandes sont exécutées par le biais d'un shell	Non	Oui	Oui	Oui
Peut transférer un script local et l'exécuter à distance	Non	Non	Non	Oui
Peut définir le contexte de la commande à exécuter	Oui	Oui	Oui	Oui
Peut utiliser des shells différents	Non	Oui	Non	Oui
Peut exécuter des commandes distantes en tant qu'un autre utilisateur	Non	Non	Oui	Non
Peut définir le <b>stdin</b> sur une valeur spécifiée	Oui	Oui	Non	Non
Peut définir une valeur de délai d'expiration	Non	Non	Oui	Non

## Description des cas d'utilisation des commandes directes

La section suivante décrit certains cas d'utilisation pour les modules mentionnés ci-dessus, à l'aide de codes de retour et d'instructions Ansible.

### Mise à jour des serveurs DNS

Utilisez le module **win\_shell** pour modifier le serveur DNS d'une interface, référencé par son index. Un gestionnaire redémarre le serveur après la modification.



#### Note

L'indicateur de style de bloc (>) vous permet d'utiliser des instructions multilignes, concaténées au moment de l'exécution. Il s'agit d'une syntaxe utile pour passer plusieurs paramètres à une commande.

```
- name: Update DNS configuration on NICs
hosts: all
vars:
  net_adapter: 'Ethernet 3' ❶

tasks:
  - name: Interface index is retrieved
    win_shell: >
      (Get-NetIpInterface "{{ net_adapter }}" |
       Select-Object -ExpandProperty InterfaceIndex -last 1 |
```

```

    Out-String) ②
register: net_index

- name: DNS server is set
  win_shell: >
    Set-DnsClientServerAddress
    -InterfaceIndex '"{{ net_index.stdout_lines[0] | trim }}"' ③
    -ServerAddresses ("10.0.0.1","10.0.0.2")
  notify:
    - restart_server ④

handlers:
  - name: restart_server
    win_reboot:

```

- ① Définit le nom de l'interface réseau à interroger.
- ② Liste tous les adaptateurs et les filtres de l'index et capture le résultat dans une variable. La **dernière** propriété de l'applet de commande **Select-Object** limite la sortie à un résultat. Cela empêche les sorties en double.
- ③ Le serveur DNS est défini à l'aide de l'index d'interface renvoyé par la tâche précédente.
- ④ Le filtre **trim** supprime la sortie des caractères, tels que les retours chariot.
- ⑤ Cette commande informe le gestionnaire **restart\_server** qui indique au serveur de redémarrer si la commande met à jour le périphérique réseau.



### Note

Dans cet exemple, le déclencheur s'exécute toujours et le serveur redémarre à chaque fois. Un moyen d'améliorer ce playbook consiste à ajouter une tâche supplémentaire qui vérifie si les entrées DNS sont déjà définies. Cela empêche le serveur de redémarrer si aucune modification n'est apportée aux entrées DNS.

## Exécution de sauvegardes à l'aide de Robocopy

Pour sauvegarder un dossier à l'aide de Robocopy, utilisez **win\_shell**. Ansible indique aux systèmes de créer des copies de dossiers en miroir, tout en préservant tous les fichiers et dossiers préexistants dans la destination.

Ansible ignore la sauvegarde si le fichier journal existe, tel qu'il est défini par la variable **creates**. L'instruction **failed\_when** indique à Ansible d'abandonner le play si Robocopy renvoie un code de sortie différent de 0.



### Note

Bien que vous puissiez utiliser le module Ansible **win\_robocopy**, il ne prend pas en charge toutes les options Robocopy.

```

- name: Data is backed up
  win_shell: >
    robocopy C:\sites\ C:\Storage\Backup\sites
    /MIR /XX
    /log:C:\Storage\Backup\backup_result.log
  args:

```

```

creates: C:\Storage\Backup\backup_result.log
register: backup_result
failed_when: backup_result.rc != 0

```

Vous pouvez éventuellement utiliser **win\_shell** pour envoyer un courrier électronique aux administrateurs à propos de l'échec.

```

- name: Data is backed up
  win_shell: >
    robocopy C:\sites\ C:\Storage\Backup\sites
    /MIR /XX
    /log:C:\Storage\Backup\backup_result.log
  args:
    creates: C:\Storage\Backup\backup_result.log
  register: backup_result
  failed_when: backup_result.rc != 0

- name: Email is sent to backup administrators
  win_shell: >
    Send-MailMessage
    -To backup-admins@acme.org
    -from backup_server@acme.org
    -Subject 'Backup did not complete'
    -SmtpServer 127.0.0.1
  when: backup_result.rc != 0

```

Pour une automatisation plus sophistiquée, vous pouvez exécuter le script PowerShell suivant en cas d'échec de la sauvegarde. Les scripts PowerShell prennent en charge l'utilisation de serveurs SMTP externes avec authentification.

```

# This PowerShell script
$Username = "Username";
$Password = "Password";

function Send-ToEmail([string]$email){
    $message = new-object Net.Mail.MailMessage;
    $message.From = "backup_server@acme.org";
    $message.To.Add($email);
    $message.Subject = "Backup did not complete";
    $message.Body = "Backup did not complete on the backup server. Please review the logs";

    $smtp = new-object Net.Mail.SmtpClient("smtp.acme.org", "587");
    $smtp.EnableSSL = $true;
    $smtp.Credentials = New-Object System.Net.NetworkCredential($Username,
$Password);
    $smtp.send($message);
    write-host "Mail Sent";
    $attachment.Dispose();
}

Send-ToEmail -email "backup-admins@acme.org";

```

```
- name: Email is sent to backup administrators
  script: files/send-email.ps1
  when: backup_result.rc != 0
```

## Génération de certificats à l'aide d'Ansible

L'exemple suivant montre une utilisation avancée du module `win_shell` visant à générer un certificat TLS pour un service.

Le playbook interroge le magasin de certificats et la configuration du site Web IIS par défaut, puis exécute des vérifications supplémentaires pour déterminer les actions à effectuer.

```
- name: Generate a self-signed certificate
  hosts: web.acme.org
  gather_facts: false
  vars:
    dns_name: 'acme.org'

  tasks:
    - name: Certificate is retrieved ①
      win_shell: dir cert:\localmachine\my\
      register: result_output

    - name: Certificate is generated ②
      win_shell: >
        New-SelfSignedCertificate
        -DnsName "{{ inventory_hostname }}"
        -CertStoreLocation cert:\LocalMachine\My
        -KeyExportPolicy Exportable
      failed_when: cert_creation.rc != 0 ③
      when: 'dns_name not in result_output.stdout' ④

    - name: IIS Binding is retrieved ⑤
      win_shell: Get-IISSiteBinding 'Default Web Site'
      changed_when: false
      register: binding_result

    - name: Certificate thumbprint is retrieved ⑥
      win_shell: >
        (Get-ChildItem -Path Cert:\LocalMachine\My |
         Where-Object {$_._Subject -match "{{ dns_name }}"}).Thumbprint;
      register: cert_thumbprint

    - name: SSL binding is configured ⑦
      win_iis_webbinding:
        name: Default Web Site
        protocol: https
        port: 443
        ip: "*"
        certificate_hash: "{{ cert_thumbprint.stdout | trim }}"
        state: present
      when: "'443' not in binding_result.stdout"
```

- ➊ La tâche vérifie la présence d'un certificat dans le magasin SSL, puis enregistre la sortie dans la variable **result\_output**.
- ➋ La tâche génère un certificat SSL dont le nom correspond à la variable **dns\_name**.
- ➌ La tâche abandonne le play en cas d'échec de la commande, c'est-à-dire si le code de retour est différent de 0.
- ➍ Ansible indique au système de créer le certificat uniquement s'il n'est pas présent dans le magasin. Pour cette vérification, la tâche inspecte la sortie de la variable **result\_output**.
- ➎ La tâche appelle l'applet de commande **Get-IISSiteBinding** pour vérifier si le site IIS par défaut dispose d'une liaison SSL.
- ➏ La tâche récupère l'empreinte du certificat SSL, puis enregistre la sortie dans la variable **cert\_thumbprint**.
- ➐ Enfin, le playbook crée une liaison qui associe le certificat SSL au site IIS. Ansible transmet la sortie standard de la variable **cert\_thumbprint** au paramètre **certificate\_hash**.

L'instruction **when** indique à Ansible d'ignorer la tâche si la liaison est présente.

## Exécution de commandes avec PsExec

PsExec est un utilitaire qui permet d'exécuter des commandes sur des hôtes distants ; il exécute des processus sur d'autres systèmes sans avoir à installer manuellement le programme, et remplace **telnet**.



### Important

Pour utiliser ce module, vous devez autoriser les connexions via le port TCP 45 et le port UDP 137 sur les serveurs auxquels PsExec se connecte.

Pour interagir avec PsExec, utilisez le module **win\_psexec**. Ce module prend en charge différentes options, telles que les informations d'identification à utiliser, la possibilité d'interagir avec le poste de travail sur les systèmes distants et de définir une priorité pour le processus.

L'exemple suivant montre comment configurer le pare-feu de deux serveurs Web et comment récupérer un fichier à partir de ces serveurs après l'exécution de la commande. Le code de retour permet à Ansible d'annuler le play si la commande PsExec échoue.



### Note

Le paramètre **hostname** définit une liste d'hôtes auxquels PsExec se connecte et sur lesquels les commandes sont exécutées. Ils peuvent être différents des hôtes gérés par Ansible.

```
- name: Configure firewall rule and retrieve CA certificate on web servers
hosts: all

tasks:
  - name: Firewall rule is added
    win_psexec:
      command: >-
        netsh advfirewall firewall
        add rule name="Open Port 80" dir=in
        action=allow protocol=TCP localport=80" ❷
      executable: C:\PsTools\PsExec.exe ❸
      hostnames: ❹
```

```
- web-001
- web-002
- web-003
elevated: yes
username: Administrator
password: 'D3m0$'
priority: realtime
register: psexec_command
failed_when: psexec_command.rc != 0

- name: CA certificates are retrieved 5
win_shell: >
    Invoke-WebRequest -Uri "https://{{ item }}.acme.org:80/CA.crt"
    -OutFile 'C:\temp\{{ item }}.crt'
loop:
- web-001
- web-002
- web-003

- name: CA certificates are imported 6
win_shell: >
    Import-Certificate -FilePath 'C:\temp\{{ item }}.crt'
    -CertStoreLocation Cert:\LocalMachine\Root
loop:
- web-001
- web-002
- web-003
```

- 1** Le signe **>-** indique à Ansible de supprimer la nouvelle ligne à la fin de la commande.
- 2** Ordonne à Ansible de se connecter à chaque serveur de la liste et d'exécuter la commande **netsh** pour ouvrir le port 80 du pare-feu. La variable **psexec\_command** stocke le résultat de la commande.
- 3** Définit l'emplacement de l'utilitaire PsExec. Il s'agit d'un paramètre utile, car PsExec peut l'exécuter à partir de n'importe quel emplacement.
- 4** Définit une liste d'hôtes à cibler. Ansible se connecte à l'hôte géré, comme défini par le paramètre **hosts**, puis exécute PsExec sur les hôtes définis dans la liste. Vous pouvez utiliser un nom DNS ou une adresse IP pour vous connecter aux hôtes gérés. L'hôte géré doit pouvoir résoudre le serveur ou y accéder.



### Note

Pour qu'il s'exécute sur tous les ordinateurs du domaine en cours, utilisez **\\*\\***.

- 5** Récupère un fichier à partir des serveurs Web, une fois que PsExec a configuré le pare-feu sur ces serveurs.
- Le mot-clé **item** indique à Ansible de faire une boucle sur les trois serveurs Web.
- 6** Ajoute l'autorité de certification (CA) au magasin de certificats de la machine locale.

## Gestion des modules PowerShell

Les modules PowerShell sont un ensemble de fonctionnalités Windows PowerShell présentées sous la forme d'une offre groupée pratique avec l'extension **.psm1**. Un module contient de manière minimale un script PowerShell qui effectue un ensemble d'opérations via PowerShell.

Utilisez Ansible pour télécharger de nouveaux modules à partir de galeries de modules, telles que PowerShell Gallery disponible à l'adresse <https://www.powershellgallery.com/>.

Les *référentiels* PowerShell décrivent les points d'accès qui hébergent ces modules ; après avoir ajouté un référentiel, utilisez l'applet de commande **Install-Module** pour installer ce référentiel.

Ansible propose deux modules pour la gestion de vos référentiels et modules PowerShell :

- **win\_psmodule**

Prend en charge l'installation ou la suppression d'un module PowerShell de la liste des référentiels configurés ou d'une galerie personnalisée.

- **win\_psrepository**

Prend en charge la gestion de vos référentiels de modules PowerShell. Il prend un nom et une source comme paramètres afin d'ajouter ou de supprimer un référentiel.

Cela revient à utiliser l'applet de commande **Register-PSRepository**.

L'exemple suivant montre comment ajouter un référentiel personnalisé, et installer un module PowerShell à partir de ce référentiel, à l'aide d'Ansible.

```
- name: Install PowerShell modules
  hosts: all

  tasks:
    - name: Repository is added ❶
      win_psrepository:
        source: https://www.powershellgallery.com/api/v2/ ❷
        name: PowerShell Gallery
        state: present

    - name: Repository is queried ❸
      win_shell: Get-PSRepository "PowerShell Gallery"
      register: repository_output
      failed_when: repo_status.rc != 0
      changed_when: false

    - name: Modules are installed ❹
      win_psmodule:
        name: "{{ item }}"
        state: present
      loop:
        - Carbon
        - PowerShellGet
        - xCertificate
```

- ❶ Cette tâche crée le référentiel **PowerShell Gallery**. Elle enregistre ensuite la sortie dans la variable **ps\_gallery**. Utilisez **state: absent** pour supprimer le référentiel, ce que fait l'applet de commande PowerShell **Unregister-PSRepository**.
- ❷ Il s'agit du point d'accès de l'API du référentiel.
- ❸ Cette tâche appelle l'applet de commande **Get-PSRepository** pour s'assurer que le référentiel est présent. Si ce n'est pas le cas, le play échoue.

- ④ Cette tâche indique au système d'installer un ensemble de modules à partir du référentiel. Pour supprimer ces modules, attribuez la valeur **absent** au paramètre **state**. Cela revient à utiliser l'applet de commande PowerShell **Uninstall-Module**.

## Planification de tâches sous Windows

Les systèmes Windows sont livrés avec un planificateur de tâches qui vous permet d'effectuer des tâches de routine sur vos systèmes. Définissez un ensemble de critères pour chaque tâche, appelés *déclencheurs*, qui indique au système d'exécuter la tâche, également appelée *actions*.

Vous pouvez exécuter de nombreuses tâches, telles que l'exécution d'un programme ou d'un script PowerShell, l'envoi d'un courrier électronique, l'installation d'un service, etc. Vous pouvez créer des tâches pour les cas suivants :

- Lorsqu'un événement système se produit, tel que le changement de nom de l'hôte.
- À un moment précis, tel qu'un jour de la semaine ou du mois.
- Lors du redémarrage du système.
- Lorsqu'un utilisateur se connecte.

Il existe de nombreuses façons d'interagir avec le planificateur de tâches, par le biais de l'interface ou par programme via PowerShell, à l'aide de C++ ou du code C# pour vos applications Windows.

Un moyen efficace d'utiliser le planificateur de tâches est l'appel de PowerShell en tant qu'action qui exécute ensuite un script via l'option **-F**. Cela vous permet de créer des scripts avancés pour la gestion de vos systèmes, et d'utiliser le planificateur de tâches pour les exécuter aux intervalles que vous définissez.

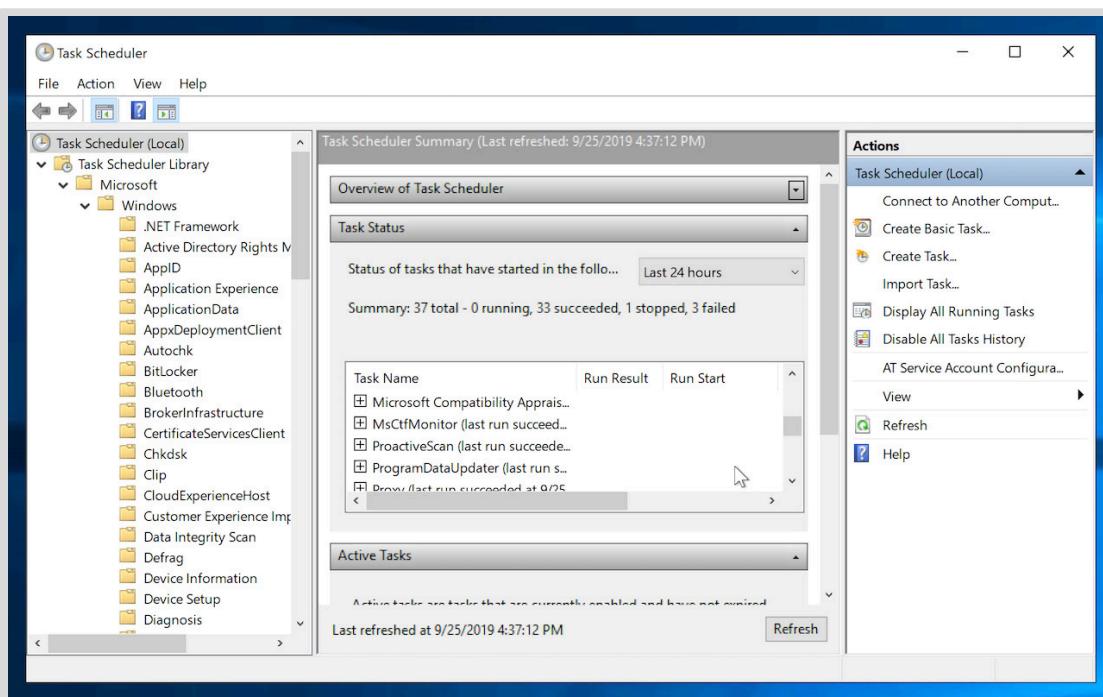


Figure 9.1: Planificateur de tâches Windows

## Planification de tâches à l'aide d'Ansible

Ansible est fourni avec deux modules pour la planification des tâches et la gestion des planifications : **win\_scheduled\_task** et **win\_scheduled\_task\_stat**. Utilisez

**win\_scheduled\_task** pour gérer les tâches planifiées et **win\_scheduled\_task\_stat** pour récupérer les informations relatives à une tâche planifiée ou la liste des tâches planifiées d'un dossier.

**win\_scheduled\_task** comporte de nombreuses options, telles que la liste des actions à configurer pour la tâche (arguments, répertoire de travail et chemin de l'exécutable), si la tâche doit être exécutée à l'aide de la commande **Run** ou du menu **Context**, de la planification de la tâche et de la priorité de la tâche.



### Note

Pour obtenir la liste de tous les paramètres, consultez [win\\_scheduled\\_task – Documentation Ansible](#) listée dans la section Références.

L'exemple suivant montre comment créer une tâche planifiée qui exécute un script PowerShell lors du démarrage du serveur et lorsque l'utilisateur auquel appartient la tâche se connecte. Le script itère au sein d'une liste de serveurs et tente de les atteindre.

```
- name: Create a scheduled task
hosts: all

tasks:
  - name: A scheduled task is created
    win_scheduled_task:
      name: QueryHosts
      description: >
        Run a PowerShell script that uses ping to reach the servers
        in the environment
      author: sylvia@acme.org
      enabled: true
      hidden: true 1
      actions:
        - path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe 2
          arguments: >
            -ExecutionPolicy Unrestricted
            -NonInteractive -File C:\Scripts\pingHosts.ps1 3
      triggers: 4
        - type: boot
        - type: logon
      priority: 1 5
      state: present
```

- 1** Lorsque **hidden** est défini sur **true**, la tâche n'apparaît pas dans la console du gestionnaire des tâches. Par défaut, **hidden** est défini sur **false**.
- 2** Crée une **action** ; dans cet exemple, cette action est une tâche qui appelle PowerShell.
- 3** Le paramètre **arguments** définit une liste d'arguments facultatifs. Le planificateur de tâches transmet ces arguments à la commande.
- 4** Définit la liste des déclencheurs ; dans cet exemple, la tâche planifiée est exécutée lors du démarrage du système, et lorsque le propriétaire de la tâche se connecte.
- 5** Définit une priorité pour la tâche, qui accepte une valeur comprise entre 0 et 10. Voir [Propriété TaskSettings.Priority – Documentation Windows](#) listée dans la section Références pour obtenir une liste des priorités possibles.

Après avoir exécuté le play, la tâche est visible dans l'interface du planificateur de tâches.

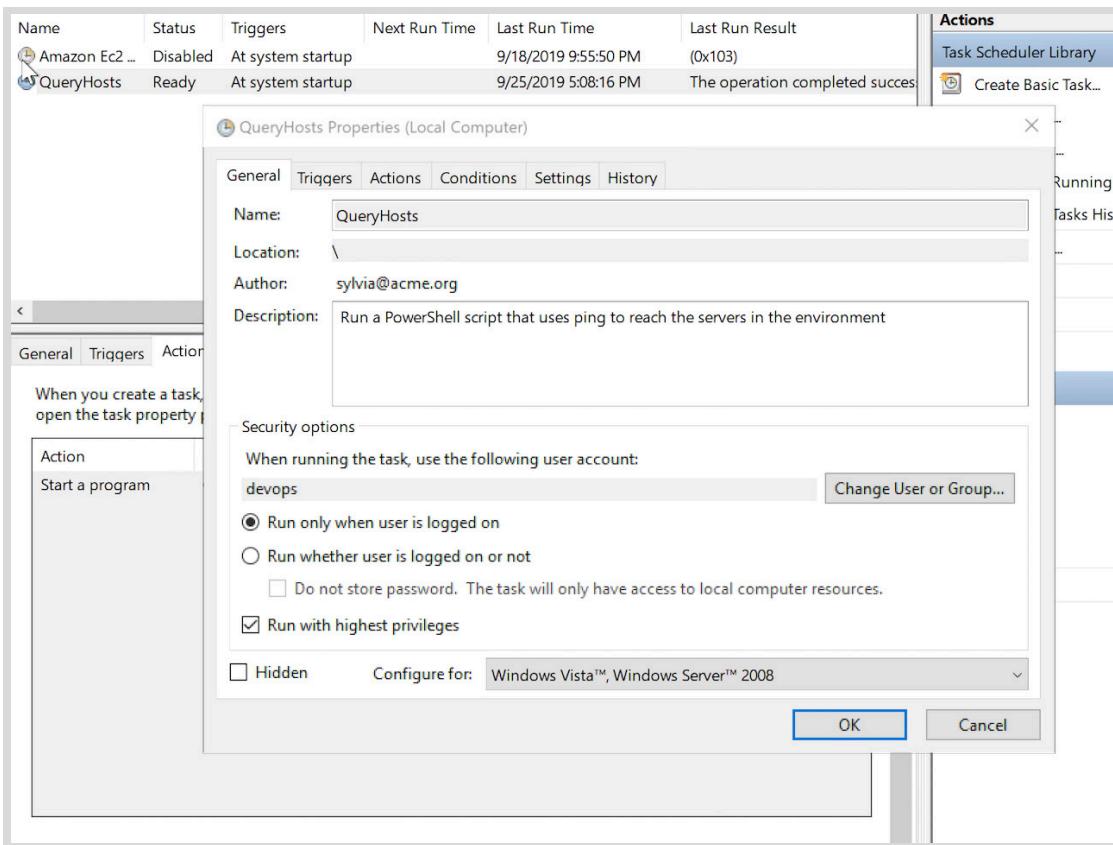


Figure 9.2: Planificateur de tâches Windows

Utilisez le module **win\_scheduled\_task\_stat** pour récupérer des informations sur les tâches planifiées sur le système. Le module renvoie des informations telles que les actions, le propriétaire de la tâche ou les déclencheurs.

Le playbook suivant utilise **win\_scheduled\_task\_stat** pour récupérer des informations sur la tâche créée dans l'exemple précédent. L'utilisation du paramètre **path** facultatif vous permet d'extraire des informations sur les tâches qui sont stockées en dehors du dossier root.

```
- name: Retrieve information about task
hosts: all

tasks:
  - name: Information about QueryHosts is retrieved
    win_scheduled_task_stat:
      name: QueryHosts
```

Le résultat est un dictionnaire JSON qui contient des informations détaillées sur la tâche :

Le dictionnaire JSON **state** contient des informations sur l'exécution de la tâche planifiée. Vous pouvez ensuite analyser ces données pour déterminer la réussite ou l'échec de la tâche.

```
{
...output omitted...
"principal": {
  "display_name": null,
  "logon_type": "TASK_LOGON_INTERACTIVE_TOKEN",
```

```

"user_id": "WIN1\\devops",
"group_id": null,
"id": "Author",
"run_level": "TASK_RUNLEVEL_HIGHEST"
},
"task_exists": true,
...output omitted...
"state": {
    "status": "TASK_STATE_READY",
    "last_task_result": 0,
    "next_run_time": "1899-12-30T00:00:00",
    "last_run_time": "2019-09-25T17:08:16",
    "number_of_missed_runs": 0
},
{
    "hide_app_window": false,
    "arguments": "-ExecutionPolicy Unrestricted -NonInteractive -File C:\\\\Scripts\\\\pingHosts.ps1",
    "path": "C:\\Windows\\\\System32\\\\WindowsPowerShell\\\\v1.0\\\\powershell.exe",
    "working_directory": null,
    "type": "TASK_ACTION_EXEC",
    "id": null
},
],
"registration_info": {
    "description": "Run a PowerShell script that uses ping to reach the servers in the environment",
    "author": "sylvia@acme.org",
    "documentation": null,
    "uri": "\\\\QueryHosts",
    "source": null,
    "version": null,
    "date": null,
    "security_descriptor": null
},
"settings": {
    "run_only_if_network_available": false,
    "execution_time_limit": "PT72H",
    "idle_settings": {
        "wait_timeout": "PT1H",
        "idle_duration": "PT10M",
        "stop_on_idle_end": true,
        "restart_on_idle": false
...output omitted...
}

```



### Note

Utilisez ce module pour obtenir des informations sur votre tâche planifiée et sur la manière de la modifier légèrement.



## Références

### **win\_scheduled\_task — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/modules/win\\_scheduled\\_task\\_module.html](https://docs.ansible.com/ansible/latest/modules/win_scheduled_task_module.html)

### **Psexec v2.2 — Documentation Windows**

<https://docs.microsoft.com/en-us/sysinternals/downloads/psexec>

### **PowerShell Gallery**

<https://www.powershellgallery.com/>

### **Utilisation du planificateur de tâches — Documentation Windows**

<https://www.powershellgallery.com/https://docs.microsoft.com/en-us/windows/win32/taskschd/using-the-task-scheduler>

### **Propriété TaskSettings.Priority — Documentation Windows**

<https://docs.microsoft.com/en-us/windows/win32/taskschd/tasksettings-priority>

## ► Exercice guidé

# Exécution des commandes et planification des tâches sur les hôtes

Au cours de cet exercice, vous allez écrire des playbooks qui gèrent des modules PowerShell, exécuter des commandes arbitraires de manière idempotente et configurer une tâche planifiée.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Créer des informations d'identification Ansible Tower.
- Écrire un playbook Ansible pour gérer les référentiels et les modules PowerShell.
- Écrire un playbook Ansible pour générer un certificat SSL et configurer un site IIS pour l'utiliser.
- Écrire un playbook Ansible pour planifier une tâche.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

- ▶ 1. Ouvrez Visual Studio Code, puis clonez le référentiel **windows** disponible à l'adresse <https://gitlab.example.com/student/windows.git>.
  - 1.1. Cliquez sur l'icône Visual Studio Code sur le Bureau pour ouvrir l'éditeur.
  - 1.2. Accédez à **View** → **Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel. Utilisez l'URL de référentiel <https://gitlab.example.com/student/windows.git>.  
À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training**, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **windows**.  
Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage pour ajouter le dossier à l'espace de travail.
- ▶ 2. Ouvrez le playbook Ansible **powershell-modules.yml**. Les étapes suivantes décrivent les modifications à apporter au playbook afin de configurer un référentiel PowerShell et d'installer un ensemble de modules.

- 2.1. Ouvrez le fichier **powershell-modules.yml** au niveau supérieur du projet.

Le playbook contient les espaces réservés que vous devez modifier. Ne mettez pas à jour la première tâche qui efface tous les référentiels existants.

- 2.2. Créez deux variables dans la section **vars** du playbook comme suit :

- **powershell\_repo\_endpoint** pointe vers un point d'accès de l'API du référentiel PowerShell qui est accessible à l'adresse <https://www.powershellgallery.com/api/v2/>.
- La variable **powershell\_modules** est une liste YAML qui définit la liste des modules PowerShell à installer. Définissez les deux modules suivants :
  - **PSWindowsUpdate**
  - **Get-PSGoodFirstIssue**

Le début du playbook doit se présenter comme suit :

```
- name: Install PowerShell modules
hosts: all
vars:
  powershell_repo_endpoint: https://www.powershellgallery.com/api/v2/
  powershell_modules:
    - PSWindowsUpdate
    - Get-PSGoodFirstIssue
```

- 2.3. Ensuite, créez une tâche qui enregistre un nouveau référentiel PowerShell. Utilisez le module **win\_psrepository** pour effectuer cette tâche. Le module requiert les éléments suivants : *source*, *name* et *state*. Utilisez **powershell\_repo\_endpoint** en tant que paramètre **source** du module.

```
...output omitted...
tasks:
  - name: Current repositories are unregistered
    win_shell: Unregister-PSRepository "{{ item }}"
    ignore_errors: true
    loop:
      - PowerShell Gallery
      - PSGallery

  - name: Repository is added
    win_psrepository:
      source: "{{ powershell_repo_endpoint }}"
      name: "PowerShell Gallery"
      state: present
```

- 2.4. Ajoutez une tâche pour vous assurer que le référentiel PowerShell est présent. La tâche capture la sortie dans la variable **repo\_status**, puis échoue si la commande ne renvoie pas 0.

L'instruction **changed\_when: false** indique à Ansible de renvoyer l'état **OK**, car la tâche n'apporte aucune modification au système.

**Note**

Notez la mise en retrait. Étant donné que **register**, **failed\_when** et **changed\_when** ne font pas partie des paramètres du module, vous devez les mettre en retrait au même niveau que le nom du module.

```
...output omitted...
- name: Repository is queried
  win_shell: Get-PSRepository "PowerShell Gallery"
  register: repo_status
  failed_when: repo_status.rc != 0
  changed_when: false
```

- 2.5. Ajoutez une tâche pour installer les modules PowerShell figurant dans la liste **powershell\_modules**. Utilisez le module **win\_psmodule** pour effectuer cette tâche. Dans la mesure où il y a plusieurs paquetages, créez une boucle pour effectuer une itération sur les éléments de la liste.

**Note**

Étant donné que la variable est une liste, vous pouvez la transmettre dans la boucle **loop** telle quelle.

```
...output omitted...
- name: PowerShell modules are installed
  win_psmodule:
    name: "{{ item }}"
    state: present
  loop: "{{ powershell_modules }}"
```

Une fois l'opération terminée, le playbook doit s'afficher comme suit :

```
- name: Install PowerShell modules
  hosts: all
  vars:
    powershell_repo_endpoint: https://www.powershellgallery.com/api/v2/
    powershell_modules:
      - PSWindowsUpdate
      - Get-PSGoodFirstIssue

  tasks:
    - name: Current repositories are unregistered
      win_shell: Unregister-PSRepository "{{ item }}"
      ignore_errors: true
      loop:
        - PowerShell Gallery
        - PSGallery

    - name: Repository is added
```

```

win_psrepository:
  source: "{{ powershell_repo_endpoint }}"
  name: PowerShell Gallery
  state: present

  - name: Repository is queried
    win_shell: Get-PSRepository "PowerShell Gallery"
    register: repo_status
    failed_when: repo_status.rc != 0
    changed_when: false

  - name: PowerShell modules are installed
    win_psmodule:
      name: "{{ item }}"
      state: present
    loop: "{{ powershell_modules }}"

```

**Note**

Le playbook complet est disponible dans le répertoire **solutions** du projet.

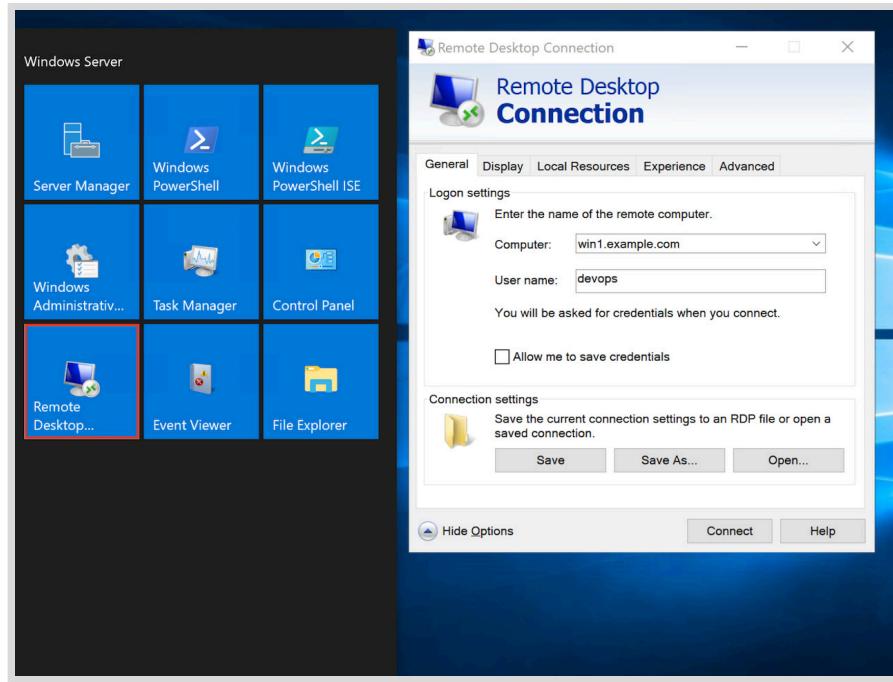
- ▶ 3. Enregistrez et validez vos modifications, puis synchronisez le référentiel.
  - 3.1. Cliquez sur **Source Control**.
  - 3.2. Cliquez sur **+** pour que le fichier **powershell-modules.yml** indexe vos modifications.
  - 3.3. Dans le champ commit message, saisissez le message de validation **Register PowerShell repository and install modules**, puis cliquez sur **Commit** au-dessus du champ message, ou appuyez sur **Ctrl+Entrée**.
  - 3.4. Cliquez sur Synchronize changes dans le coin inférieur gauche de l'éditeur. Cette action synchronise vos modifications avec GitLab.
- ▶ 4. Accédez à Ansible Tower et mettez à jour le modèle **Run windows project**. Utilisez le playbook Ansible **powershell-modules.yml**.
  - 4.1. À partir de **workstation**, double-cliquez sur Ansible Tower pour accéder à la console Web.  
Connectez-vous à Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe
  - 4.2. Cliquez sur **Templates** pour accéder à la liste des modèles, puis sélectionnez **Run windows project** pour le modifier.
  - 4.3. Sélectionnez **powershell-modules.yml** dans le champ **PLAYBOOK**. Conservez les autres valeurs telles quelles, puis cliquez sur **Save**.

The screenshot shows the 'Run windows project' configuration page in Ansible Tower. The 'PLAYBOOK' field is set to 'powershell-modules.yml' and is highlighted with a red box. Other fields include 'NAME' (Run windows project), 'DESCRIPTION' (Use this job template to run your playbooks), 'JOB TYPE' (Run), 'INVENTORY' (Default inventory), 'PROJECT' (windows repository), 'CREDENTIAL' (DevOps), 'VERBOSITY' (0 (Normal)), 'LABELS' (empty), 'TIMEOUT' (0), 'INSTANCE GROUPS' (empty), 'SHOW CHANGES' (OFF), 'LIMIT' (0), 'SKIP TAGS' (empty), 'JOB SLICING' (1), and 'OPTIONS' (checkboxes for privilege escalation, provisioning callbacks, concurrent jobs, and fact cache).

- 4.4. Cliquez sur **LAUNCH** pour exécuter le modèle de tâche.
- 4.5. Observez la sortie tandis qu'Ansible Tower efface tous les référentiels existants, enregistre le référentiel que vous avez défini et installe les deux modules PowerShell.

The screenshot shows the Ansible Tower interface with two main panes. The left pane displays the 'DETAILS' of the 'Run windows project' job, including its status (Running), start time (9/30/2019 3:01:49 PM), and configuration details like job template, project, and playbook. The right pane shows the 'Run windows project' log output, which includes tasks for registering repositories, adding the PowerShell Gallery, querying the repository, and installing PowerShell modules. The log output is timestamped and shows 'ok' and 'changed' states for various hosts (win1.example.com, win2.example.com).

5. À partir de **workstation**, ouvrez le client du bureau à distance pour vous connecter à **win1.example.com**. Assurez-vous que les modules sont installés sur le serveur.
  - 5.1. Accédez au client **Remote Desktop Connection** disponible dans le menu **Démarrer**. Le client est accessible à partir de la liste des applications **Windows Server**. Accédez à **win1** en utilisant **devops** comme nom d'utilisateur. Cliquez sur **Connect** pour lancer la connexion. Utilisez le mot de passe **RedHat123@!**, puis cliquez sur **OK**. À l'invite, cliquez sur **Yes** pour accepter le certificat auto-signé.



- 5.2. Cliquez sur **Windows PowerShell** dans le menu **Démarrer** pour ouvrir une console PowerShell.
- 5.3. Tapez la commande suivante pour importer le module **PSWindowsUpdate** :

```
PS C:\Users\devops> import-Module PSWindowsUpdate
```

- 5.4. Interrogez les options disponibles du module.

```
PS C:\Users\devops> Get-Command -module PSWindowsUpdate
```

CommandType	Name	Version	Source
-----	-----	-----	-----
Alias	Clear-WUJob	2.1.1.2	PSWindowsUpdate
Alias	Download-WindowsUpdate	2.1.1.2	PSWindowsUpdate
Alias	Get-WUIInstall	2.1.1.2	PSWindowsUpdate
Alias	Get-WUList	2.1.1.2	PSWindowsUpdate
Alias	Hide-WindowsUpdate	2.1.1.2	PSWindowsUpdate
Alias	Install-WindowsUpdate	2.1.1.2	PSWindowsUpdate
Alias	Show-WindowsUpdate	2.1.1.2	PSWindowsUpdate
Alias	UnHide-WindowsUpdate	2.1.1.2	PSWindowsUpdate
<i>...output omitted...</i>			

- 5.5. Répétez les deux étapes précédentes pour importer le module **adPSGoodFirstIssue** et interrogez les options disponibles. Quittez la connexion RDP lorsque vous avez terminé.

```
PS C:\Users\devops> import-Module Get-PSGoodFirstIssue
PS C:\Users\devops> Get-Command -module Get-PSGoodFirstIssue
 CommandType      Name          Version   Source
 -----          ----          0.0.7     Get-PSGoodFirstIssue
 Function        Get-PSGoodFirstIssue    0.0.7     Get-PSGoodFirstIssue
 Function        Get-PSHacktoberFestIssue
```

- ▶ 6. Revenez dans Visual Studio Code et ouvrez le playbook Ansible **IIS-ssl.yml**. Le playbook contient des tâches initiales qui configurent le site IIS.

Les étapes suivantes décrivent la liste des modifications à apporter au playbook afin de configurer un site Web IIS sur **win1.example.com** avec un certificat SSL auto-généré.

- Ajoutez une tâche qui vérifie si le site IIS dispose d'une liaison SSL sur le port 8888. Au cours d'une étape ultérieure, vous allez configurer ce port pour servir le site Web via TLS.

La tâche utilise l'applet de commande **Get-IISSiteBinding** pour récupérer la liaison SSL du site IIS **D0417-windows**, puis enregistre la sortie dans la variable **iis\_ssl\_stat**. Étant donné que la tâche n'apporte aucune modification au système, ajoutez l'instruction **changed\_when: false**.

```
...output omitted...
## DO NOT EDIT ABOVE THIS LINE ##

- name: IIS binding is retrieved
  win_shell: Get-IISSiteBinding "D0417-windows"
  changed_when: false
  register: iis_ssl_stat
```

- Ajoutez une tâche qui vérifie la présence d'un certificat SSL pour **win1.example.com**. Enregistrez le résultat dans une variable et utilisez l'instruction **ignore\_errors: true** pour vous assurer que le play s'exécute même si le certificat n'existe pas.

Étant donné que la commande n'apporte aucune modification au système, ajoutez **changed\_when: False**.

```
...output omitted...
- name: SSL is queried
  win_shell: dir cert:\localmachine\my\
  ignore_errors: True
  changed_when: False
  register: ssl_status
```

- Si la tâche précédente ne renvoie aucune valeur correspondante pour un certificat, ajoutez une tâche qui génère le certificat SSL.

La tâche appelle l'applet de commande **New-SelfSignedCertificate** pour générer un certificat auto-signé avec le nom d'hôte **win1.example.com**; Ansible remplace **inventory\_hostname** par le nom d'hôte du serveur.

```
...output omitted...
- name: SSL certificate is generated
  win_shell: >
    New-SelfSignedCertificate
    -DnsName "{{ inventory_hostname }}"
    -CertStoreLocation cert:\LocalMachine\My
    -KeyExportPolicy Exportable
  when: "'win1' not in ssl_status.stdout"
```

- 6.4. Ajoutez une clause **block** qui indique à Ansible de configurer les services IIS avec un certificat SSL si la liaison SSL est absente. Elle effectue les tâches suivantes :

- récupère l'empreinte de certificat et la stocke dans une variable ;
- crée une liaison SSL pour le site IIS **D0417-windows** ;
- redémarre le site IIS.

Ajoutez une instruction **when** pour ordonner à Ansible d'exécuter le bloc si la liaison SSL n'est pas présente.

```
...output omitted...
- name: IIS site is configured
  block
  when: "'8888' not in iis_ssl_stat.stdout"
```

- 6.5. Dans l'instruction **block**, créez une tâche nommée **Certificate thumbprint is retrieved**. La tâche récupère l'empreinte de certificat SSL et la stocke dans **cert\_thumbprint**.

```
...output omitted...
- name: IIS site is configured
  block:
    - name: Certificate thumbprint is retrieved
      win_shell: >
        (Get-ChildItem -Path Cert:\LocalMachine\My | Where-Object
         {$_.Subject -match "{{ inventory_hostname }}"}).Thumbprint;
      register: cert_thumbprint
```

- 6.6. Ajoutez une deuxième tâche dans le bloc qui configure la liaison SSL du site IIS **D0417-windows**. Nommez la tâche **SSL binding is added**. Cette tâche réutilise la variable **cert\_thumbprint** dans le paramètre **certificate\_hash**. Le filtre **trim** supprime toute continuation de ligne.

```
...output omitted...
- name: SSL binding is added
  win_iis_webbinding:
    name: D0417-windows
    protocol: https
    port: 8888
    ip: "*"
    certificate_hash: "{{ cert_thumbprint.stdout | trim }}"
    state: present
```

- 6.7. Enfin, ajoutez une tâche nommée **IIS site is restarted** pour redémarrer le site IIS. Il utilise le module **win\_iis\_website** pour redémarrer le site, puis définit une nouvelle entrée pour IIS.

```
...output omitted...
- name: IIS site is restarted
  win_iis_website:
    name: D0417-windows
    state: started
```

Le playbook mis à jour s'affiche comme suit : les tâches qui sont déjà incluses dans le playbook sont omises de la sortie.

```
- name: IIS site with SSL certificate is Installed and Configured
hosts: win1.example.com

tasks:
...output omitted...
- name: IIS binding is retrieved
  win_shell: Get-IISSiteBinding "D0417-windows"
  changed_when: false
  register: iis_ssl_stat

- name: SSL is queried
  win_shell: dir cert:\localmachine\my\
  ignore_errors: True
  changed_when: False
  register: ssl_status

- name: SSL certificate is generated
  win_shell: >
    New-SelfSignedCertificate
    -DnsName "{{ inventory_hostname }}"
    -CertStoreLocation cert:\LocalMachine\My
    -KeyExportPolicy Exportable
  when: "'win1' not in ssl_status.stdout"

- name: IIS site is configured
  block:
    - name: Certificate thumbprint is retrieved
      win_shell: >
        (Get-ChildItem -Path Cert:\LocalMachine\My | Where-Object
        {$_.Subject -match "{{ inventory_hostname }}"}).Thumbprint;
      register: cert_thumbprint

    - name: SSL binding is added
      win_iis_webbinding:
        name: D0417-windows
        protocol: https
        port: 8888
        ip: "*"
        certificate_hash: "{{ cert_thumbprint.stdout | trim }}"
        state: present
```

```

- name: IIS site is restarted
  win_iis_website:
    name: D0417-windows
    state: started

when: "'8888' not in iis_ssl_stat.stdout"

```



### Note

Le playbook complet est disponible dans le répertoire **solutions** du projet.

- ▶ **7.** Enregistrez le playbook et indexez-le. Cliquez sur **Source Control**. Cliquez sur **+** pour que le fichier **IIS-ssl.yml** indexe vos modifications.
- ▶ **8.** Dans le champ commit message, saisissez un message de validation **Configure an SSL certificate for IIS**, puis cliquez sur le bouton **Commit** au-dessus du champ message ou tapez **Ctrl+Entrée**.
- ▶ **9.** Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- ▶ **10.** Accédez à Ansible Tower et mettez à jour le modèle **Run windows project**. Sélectionnez **IIS-ssl.yml** comme playbook Ansible pour le modèle de tâche.
  - 10.1. À partir de **workstation**, accédez à Ansible Tower. Accédez à **Templates** dans le menu latéral pour atteindre les modèles. Cliquez sur **Run windows project** pour modifier le modèle de tâche.
  - 10.2. Sélectionnez **IIS-ssl.yml** dans le champ **PLAYBOOK**, puis cliquez sur **SAVE**.

The screenshot shows the 'Run windows project' template configuration in Ansible Tower. The 'PLAYBOOK' field is highlighted with a red box. Other fields visible include 'NAME' (Run windows project), 'DESCRIPTION' (Use this job template to run your playbooks), 'JOB TYPE' (Run), 'INVENTORY' (Default inventory), 'PROJECT' (windows repository), 'CREDENTIAL' (DevOps), 'FORKS' (0), 'VERBOSITY' (0 (Normal)), 'LIMIT' (1), 'SHOW CHANGES' (OFF), and various 'OPTIONS' checkboxes.

- 10.3. Cliquez sur **LAUNCH** pour exécuter la tâche. Ansible vous redirige vers la sortie de la tâche. Le résultat indique qu'Ansible configure le site IIS, génère le certificat SSL et met à jour le site IIS.

**DETAILS**

- STATUS: Successful
- STARTED: 10/1/2019 10:41:19 AM
- FINISHED: 10/1/2019 10:42:58 AM
- JOB TEMPLATE: Run windows project
- JOB TYPE: Run
- LAUNCHED BY: admin
- INVENTORY: Default inventory
- PROJECT: windows repository
- REVISION: 364ddd2
- PLAYBOOK: dummy.yml
- CREDENTIAL: DevOps
- ENVIRONMENT: /var/lib/awx/venv/ansible
- EXECUTION NODE: localhost
- INSTANCE GROUP: tower
- EXTRA VARIABLES: YAML JSON EXPAND

**Run windows project**

PLAYS	TASKS	HOSTS	ELAPSED
1	14	1	00:01:39

SEARCH  Q KEY

```

36 TASK [SSL certificate is generated]
*****
37 changed: [win1.example.com]
38
39 TASK [Certificate thumbprint is retrieved]
*****
40 changed: [win1.example.com]
41
42 TASK [SSL binding is added]
*****
43 changed: [win1.example.com]
44
45 TASK [IIS site is restarted]
*****
46 ok: [win1.example.com]
47
48 PLAY RECAP
*****
49 win1.example.com : ok=14  changed=7    unreachable=0    failed=0
d=0      skipped=0   rescued=0    ignored=0
50

```

- 11. À partir de **workstation**, ouvrez un nouvel onglet Google Chrome et accédez à <https://win1.example.com:8888>. Cliquez sur le bouton **Not Secure** dans la partie supérieure pour passer en revue les informations de certificat, ce qui confirme que le site est correctement exécuté avec un certificat de sécurité auto-signé.

This is the default page for your IIS site.

This page has been published by the administrator.

To make changes to this page, edit the configuration file.

List of the modules used to create this page:

- win\_feature
- win\_file
- win\_copy
- win\_iis\_website

**Certificate**

General Details Certification Path

**Certificate Information**

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: win1.example.com

Issued by: win1.example.com

Valid from 10/1/2019 to 10/1/2020

OK

Used to configure the IIS website

- 12. Revenez dans Visual Studio Code et ouvrez le playbook Ansible **scheduled-tasks.yml**. Ce playbook crée une tâche planifiée qui exécute un script PowerShell. Le script itère au sein d'une liste de deux serveurs et tente de les atteindre.

Les étapes suivantes décrivent la liste des modifications à apporter au playbook afin de planifier une tâche dans le planificateur Windows.

- 12.1. Définissez une variable **ps\_script** dans la section **vars** du playbook. Attribuez-lui la valeur **pingHosts.ps1**

```
- name: Create a scheduled task
hosts: win1.example.com
vars:
  ps_script: 'pingHosts.ps1'
```

- 12.2. Définissez la première tâche qui utilise le module **win\_file**. La tâche crée le répertoire **scripts** sur l'hôte géré.

```
- name: Create a scheduled task
hosts: win1.example.com
vars:
  ps_script: 'pingHosts.ps1'

tasks:
  - name: Script directory is created
    win_file:
      path: C:\Scripts\
      state: directory
```

- 12.3. Créez une tâche qui utilise le module **win\_copy** pour transférer le script PowerShell vers l'hôte géré dans **C:\Scripts**. Réutilisez la variable **ps\_script** pour le paramètre **src**.



#### Note

Assurez-vous d'utiliser des barres obliques et non des barres obliques inverses, car le fichier se trouve sur l'hôte Ansible Tower.

```
...output omitted...
- name: PowerShell script is transferred
  win_copy:
    src: "files/{{ ps_script }}"
    dest: C:\Scripts\
```

- 12.4. Créez une tâche qui utilise le module **win\_scheduled\_task** pour créer la tâche planifiée. Utilisez les informations suivantes :

- Attribuez le nom **QueryHosts** à la tâche.
- Donnez à la tâche la description **Run a PowerShell script that uses ping to reach servers in the environment**.
- Appelez PowerShell en mode non interactif et utilisez l'option **-File** pour passer le script PowerShell **pingHosts.ps1** en tant qu'argument.
- Définissez deux déclencheurs pour la tâche : un qui indique à la tâche planifiée de s'exécuter lors du démarrage du système, et un autre qui indique que la tâche doit être exécutée lorsque l'utilisateur se connecte.
- Définissez la priorité 1.

**Note**

Certaines valeurs sont déjà préremplies pour aider à la réalisation de la tâche.

```
...output omitted...
- name: A scheduled task is created
  win_scheduled_task:
    name: QueryHosts
    description: >
      Run a PowerShell script that uses ping to reach a servers
      in the environment
    author: student@example.com
    enabled: true
    actions:
      - path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        arguments: >
          -ExecutionPolicy Unrestricted
          '-NonInteractive -File C:\Scripts\{{ ps_script }}'
    triggers:
      - type: boot
      - type: logon
    priority: 1
    state: present
```

- 12.5. Utilisez le module **win\_scheduled\_task\_stat** pour récupérer des informations sur la tâche planifiée. Le module vide un dictionnaire JSON qui est accessible à partir de la sortie de tâche d'Ansible Tower. Attribuez le nom de la tâche planifiée au paramètre **name**.

Utilisez l'instruction **changed\_when: False**, car la tâche n'apporte aucune modification au système.

```
...output omitted...
- name: Status of the task is retrieved
  win_scheduled_task_stat:
    name: QueryHosts
    changed_when: False
```

Le playbook terminé doit s'afficher comme suit :

```
- name: Create a scheduled task
hosts: win1.example.com
vars:
  ps_script: 'pingHosts.ps1'

tasks:
  - name: Script directory is created
    win_file:
      path: C:\Scripts\
      state: directory
```

```
- name: PowerShell script is transferred
  win_copy:
    src: "files/{{ ps_script }}"
    dest: C:\Scripts\

- name: A scheduled task is created
  win_scheduled_task:
    name: QueryHosts
    description: >
      Run a PowerShell script that uses ping to reach a servers
      in the environment
    author: student@example.com
    enabled: true
    actions:
      - path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        arguments: >
          -ExecutionPolicy Unrestricted
          '-NonInteractive -File C:\Scripts\{{ ps_script }}'
    triggers:
      - type: boot
      - type: logon
    priority: 1
    state: present

- name: Status of the task is retrieved
  win_scheduled_task_stat:
    name: QueryHosts
    changed_when: False
```



### Note

Le playbook complet est disponible dans le dossier **solutions** du projet.

- 13. Enregistrez et validez vos modifications, puis synchronisez le référentiel.
- 13.1. Cliquez sur l'icône **Source Control** dans la barre latérale.
  - 13.2. Cliquez sur le symbole + pour que le fichier **scheduled-tasks.yml** indexe vos modifications.
  - 13.3. Dans le champ commit message, saisissez un message de validation de **Configure a scheduled task**, puis cliquez sur **commit** ou appuyez sur **Ctrl+Entrée**.
  - 13.4. Cliquez sur Synchronize changes dans le coin inférieur gauche de l'éditeur. Cette action synchronise vos modifications avec GitLab.
- 14. Accédez à Ansible Tower et mettez à jour le modèle **Run windows project**. Utilisez le playbook Ansible **scheduled-tasks.yml**.
- 14.1. À partir de **workstation**, accédez à Ansible Tower. Accédez à **Templates** pour accéder aux modèles.  
Cliquez sur **Run windows project** pour modifier le modèle de tâche.
  - 14.2. Sélectionnez **scheduled-tasks.yml** dans le champ **PLAYBOOK**, puis cliquez sur **SAVE**.

- 14.3. Cliquez sur **LAUNCH** pour exécuter la tâche. Ansible vous redirige vers la sortie de la tâche. Le résultat indique qu'Ansible installe le script PowerShell dans le répertoire **scripts** et crée une tâche planifiée.

- 14.4. Dans la sortie de tâche, cliquez sur l'entrée **ok: [win1.example.com]** de la tâche **Status of the task is retrieved** afin de lire les informations relatives à la tâche planifiée. Examinez le dictionnaire JSON qui contient des informations sur le niveau d'exécution de la tâche planifiée, les arguments de la tâche, ainsi que d'autres informations sur la tâche.



The screenshot shows a terminal window with the following details:

- win1.example.com**
- CREATED**: 10/1/2019 1:34:31 PM
- ID**: 4702
- PLAY**: Create a scheduled task
- TASK**: Status of the task is retrieved
- MODULE**: win\_scheduled\_task\_stat

A **JSON** button is visible. Below it, the JSON output is displayed:

```

34 "principal": {
35   "display_name": null,
36   "logon_type": "TASK_LOGON_INTERACTIVE_TOKEN",
37   "user_id": "WIN1\\devops",
38   "group_id": null,
39   "id": "Author",
40   "run_level": "TASK_RUNLEVEL_LUA"
41 },
42 "task_exists": true,

```

A **CLOSE** button is at the bottom right.

► 15. À partir de **workstation**, ouvrez le client du bureau à distance pour vous connecter à **win1.example.com**. Lancez le planificateur de tâches pour confirmer la présence de la tâche planifiée.

- 15.1. Accédez au client **Remote Desktop Connection** disponible dans le menu **Démarrer**. Le client est accessible à partir de la liste des applications **Windows Server**. Accédez à **win1** en utilisant **devops** comme nom d'utilisateur. Cliquez sur **Connect** pour lancer la connexion. Utilisez le mot de passe **RedHat123@!**, puis cliquez sur **OK**. À l'invite, cliquez sur **Yes** pour accepter le certificat auto-signé.
- 15.2. Cliquez sur l'icône de loupe et saisissez **Task Scheduler** pour accéder au planificateur de tâches.
- 15.3. Dans le panneau **Active Tasks**, double-cliquez sur l'entrée **QueryHosts**.



#### Note

Si vous ne voyez pas la tâche, accédez à **Action → Refresh** pour mettre à jour la liste des tâches planifiées.

- 15.4. Cliquez sur les différents onglets pour passer en revue les différentes options utilisées par la tâche. Quittez la connexion RDP lorsque vous avez terminé.

The screenshot shows the Windows Task Scheduler interface. At the top, there is a table with columns: Name, Status, Triggers, Next Run Time, Last Run Time, Last Run Result, and Author. Two tasks are listed:

Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author
Amazon Ec2 ...	Disabled	At system startup	9/18/2019 9:55:50 PM		(0x103)	
QueryHosts	Ready	Multiple triggers defined		11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	student

Below the table, there is a navigation bar with tabs: General, Triggers, Actions, Conditions, Settings, and History. The 'Actions' tab is selected. A note below the tabs states: "When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages command." Under the 'Actions' tab, there is a table with columns: Action and Details. One action is listed:

Action	Details
Start a program	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPol...

L'exercice guidé est maintenant terminé.

# Configuration et gestion du stockage

---

## Résultats

À la fin de cette section, vous devez pouvoir automatiser la configuration des périphériques de stockage sur les hôtes gérés.

## Gestion du stockage sous Windows

Ansible inclut des modules pour la gestion des partitions et des systèmes de fichiers sur des périphériques de stockage basés sur des blocs, et pour la gestion du stockage réseau, tels que les partages et les lecteurs mappés. Les opérations, telles que l'obtention d'informations sur le stockage à partir d'un hôte géré ou la défragmentation d'un volume existant, sont également prises en charge. La liste suivante fournit des informations détaillées sur certaines des fonctionnalités fournies.

### Modules de stockage Windows

#### `win_defrag`

Lorsqu'il y a beaucoup d'activité sur un volume, les fichiers peuvent être fragmentés, ce qui signifie qu'ils sont répartis sur différentes zones du système de fichiers. Les fichiers fragmentés peuvent avoir des vitesses de lecture/écriture plus lentes en raison de l'augmentation du nombre de recherches requises. Une recherche a lieu lorsque le lecteur déplace la tête vers le début des données à lire. Lorsqu'un fichier est fragmenté, le lecteur doit chercher chaque morceau avant de procéder à la lecture ou à l'écriture.

Une fois qu'un système de fichiers a été défragmenté, tous les morceaux de chaque fichier sont contigus, ce qui signifie qu'ils sont situés les uns à côté des autres sur le disque ; Cela se traduit par un nombre réduit de recherches. Les systèmes de fichiers sur des disques SSD ne nécessitent pas de défragmentation, car il n'y a pas de têtes physiques à déplacer.

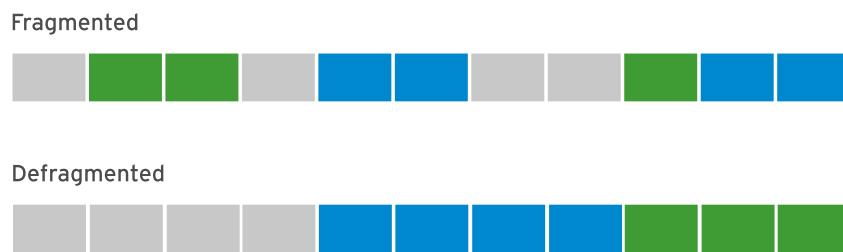


Figure 9.13: Fragmentation du système de fichiers

À l'aide du module `win_defrag`, les administrateurs peuvent déterminer exactement à quel moment la défragmentation doit se produire, car les performances du disque seront affectées lors de l'opération. Le module a plusieurs paramètres utiles, tels que `parallel`, qui permettent à tous les lecteurs d'être défragmentés en même temps. `exclude_volumes` est utilisé pour exclure un lecteur de l'opération de défragmentation. Vous pouvez également spécifier `priority` pour réduire l'impact sur les performances, et `freespace_consolidation`, ce qui peut s'avérer utile lorsqu'un disque doit être réduit.

#### win\_disk\_facts

Avant d'apporter des modifications aux partitions de disque et aux systèmes de fichiers, vous devez vous assurer que la configuration de stockage sur l'hôte géré convient. Ce module récupère les informations sur le système de stockage à partir de l'hôte géré en tant que faits qui sont ensuite intégrés à la variable **ansible\_facts** sous la matrice **disks**.

La sortie inclut un large éventail d'informations, y compris le type de bus, le fabricant, si la partition est un style MBR ou GPT, ainsi que des attributs de disque physique et virtuel. Le test des champs de la matrice **ansible\_facts['disks']** permet d'exécuter des tâches de manière conditionnelle.

#### win\_disk\_image

Offre la possibilité de monter une image de disque, qui peut également être classée comme tâche de stockage. Les images au format ISO, VHD ou VHDX sont prises en charge, ce qui autorise des opérations telles que l'installation de logiciels à partir d'une image ISO ou la manipulation de disques de machines virtuelles.

#### win\_format

Formate un volume Windows nouveau ou existant, avec un choix de systèmes de fichiers FAT, FAT32, EXFAT, NTFS ou ReFS. Spécifiez uniquement l'un des paramètres **drive\_letter**, **path** ou **label** lors de l'utilisation de ce module. Les autres paramètres disponibles incluent **compress** pour activer la compression et **full** pour exécuter un format complet.

#### win\_mapped\_drive

Prend en charge la mise en correspondance de partages Windows avec des lettres de lecteur.



##### Note

Les paramètres **username** et **password** de ce module ne sont utiles que pour tester une connexion de lecteur mappé ; vous devez utiliser le module **win\_credential** pour stocker les informations d'identification de manière persistante.

#### win\_partition

Définit les partitions de style MBR de type FAT12, FAT16, extended, huge, IFS ou FAT32. Les partitions de style GPT peuvent être n'importe laquelle des types suivants : system\_partition, microsoft\_reserved, basic\_data ou microsoft\_recovery. D'autres paramètres incluent **active** pour rendre une partition MBR amorçable et **hidden** pour empêcher le système d'exploitation de voir la partition.

#### win\_share

Définit les dossiers partagés et l'accès autorisé ; ce module est un partenaire de **win\_mapped\_drive**, les paramètres du contrôle d'accès **change**, **deny**, **full**, **list** et **read**.

L'exemple suivant montre le module **win\_defrag** en cours d'utilisation. Le lecteur système est exclu, et tous les autres lecteurs sont défragmentés en parallèle, mais avec une priorité faible.

```
- name: Data drives defragmented
  win_defrag:
    parallel: yes
    exclude_volumes: C
    priority: low
```

Cet exemple montre comment utiliser le module **win\_disk\_facts** pour extraire des informations relatives à un disque spécifique.

```
---
```

- name: System disk firmware versions are printed
  - hosts: demo.example.com

tasks:

- name: Populate disk facts
  - win\_disk\_facts:
- name: Show firmware version for drive C
  - debug:
    - var: ansible\_facts['disks'][0]['firmware\_version']

L'exemple suivant illustre le module **win\_disk\_image** montant une image ISO. Le module **debug** est utilisé pour afficher la lettre de lecteur sur lequel l'image ISO est montée.

- name: MyApp ISO is mounted
  - win\_disk\_image:
    - image\_path: C:\myapp\_installer.iso
    - state: present
  - register: myapp\_iso
- name: MyApp ISO drive letter is shown
  - debug:
    - msg: "MyApp ISO mounted at: {{ myapp\_iso['mount\_paths'][0] }}"

L'exemple suivant utilise le module **win\_format** pour effectuer un formatage complet du lecteur E::

- name: Log volume (E:) is formatted as NTFS
  - win\_format:
    - drive\_letter: E
    - file\_system: NTFS
    - new\_label: Logs
    - full: true

L'exemple suivant illustre l'utilisation du module **win\_mapped\_drive** pour mapper un lecteur avec des informations d'identification de manière persistante.

- name: Data drive for SQL Server is mapped
  - block:
  - name: Service account credentials are saved
    - win\_credential:
      - name: '\*.example.com'
      - type: domain\_password
      - username: "{{ sql\_svc\_acct\_name }}@EXAMPLE"
      - secret: "{{ sql\_svc\_acct\_pass }}"
    - state: present
  - name: Data drive is mapped

```

win_mapped_drive:
  letter: S
  path: \\NAS01\DB_DATA$
  state: present
vars:
  ansible_become: yes
  ansible_become_method: runas
  ansible_become_user: '{{ ansible_user }}'
  ansible_become_pass: '{{ ansible_password }}'

```

**Note**

Notez l'utilisation de **win\_credential** pour stocker de manière persistante les informations d'identification du compte de service. Notez également l'utilisation de variables \***\_become** qui permettent l'accès au magasin d'informations d'identification de l'utilisateur.

Dans cet exemple, le module **win\_partition** crée une partition de 50 Gio.

```

- name: Scratch disk is created
  win_partition:
    drive_letter: F
    partition_size: 50 GiB
    disk_number: 1

```

L'exemple suivant utilise le module **win\_share** pour partager un dossier de formulaires d'une société.

```

- name: Forms share is added
  win_share:
    name: Forms
    description: Company forms
    path: D:\shares\Forms
    list: yes
    full: Domain Admins,HR
    read: Domain Users

```

Les cas d'utilisation courants nécessiteront probablement plusieurs de ces modules fonctionnant ensemble. Par exemple, l'utilisation de **win\_disk\_facts** pour déterminer la configuration de stockage exacte, suivie de **win\_partition** et de **win\_format** pour créer un lecteur formaté. Un autre exemple consiste à utiliser **win\_share** pour créer un dossier partagé, suivi de **win\_credential** et **win\_mapped\_drive** pour mapper de manière persistante une lettre de lecteur sur le partage, sur les machines clientes.

**Références****Modules Windows — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_windows\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html)

**Filtre de requête JSON — Documentation Ansible**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_filters.html#json-query-filter](https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html#json-query-filter)

## ► Exercice guidé

# Configuration et gestion du stockage

Dans cet exercice, vous allez écrire un playbook qui permet de s'assurer qu'un périphérique de stockage spécifique est configuré et formaté, et qu'un partage Windows est créé sur le nouveau volume.

## Résultats

Vous devez pouvoir créer et formater des partitions, et créer des partages Windows.

## Avant De Commencer

Vérifiez que **workstation** et toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab dans un navigateur.

Ouvrez votre client RDP et connectez-vous à **workstation** en utilisant **example.com** comme domaine, **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Vous pouvez trouver les informations de connexion à vos systèmes dans le courrier électronique que vous avez reçu.

- ▶ 1. Lancez l'éditeur Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **windows**, clonez-le sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/windows.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training**, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **windows**.  
Pour afficher les fichiers, sélectionnez **Open** dans la fenêtre qui s'affiche après le clonage, puis passez à l'étape suivante.
- ▶ 2. Pour accéder au répertoire **C:\Users\student\Documents\windows**, cliquez sur **File → Open Folder**, puis sélectionnez le dossier **C:\Users\student\Documents\windows**.
- ▶ 3. Ouvrez le playbook **storage.yml**, puis effectuez les tâches.

**Note**

La tâche **Second disk extracted as standalone fact** utilise une méthode de filtre avancée pour obtenir les faits pour le deuxième disque. Cette méthode n'est pas abordée dans ce cours, mais elle est requise pour cet exercice. Pour plus d'informations, reportez-vous à la section Références de la section *Configuration et gestion du stockage*.

- 3.1. Terminez la tâche **Second disk size printed** pour afficher la valeur de la variable `second_disk[0]['size']`.

```
- name: Second disk size printed
  debug:
    var: second_disk[0]['size']
```

- 3.2. Modifiez la tâche **Partition is created on second disk** pour définir les valeurs suivantes :

- Numéro de disque 1
- Lecteur F
- Dimensionné à 1 Gio

```
- name: Partition is created on second disk
  win_partition:
    disk_number: 1
    drive_letter: F
    partition_size: 1 GiB
    register: partition_task
```

- 3.3. Exécutez la tâche **New partition is formatted** pour créer un volume NTFS nommé **data** sur le lecteur F:. La tâche est exécutée de manière conditionnelle, selon si la tâche précédente dont le résultat est apparu dans **changed** a été résolue avec la valeur **true**.

```
- name: New partition is formatted
  win_format:
    drive_letter: F
    file_system: NTFS
    new_label: data
    full: true
  when: partition_task['changed']
```

- 3.4. Exécutez la tâche **Share folder is created** pour créer le dossier **F:\shares\forms**.

```
- name: Share folder is created
  win_file:
    path: F:\shares\forms
    state: directory
```

- 3.5. Complétez la tâche **Content added to forms folder** pour créer le fichier **F:\shares\forms\leave\_requests.txt**. Cette tâche utilise le module **win\_copy**.

```
- name: Content added to forms folder
  win_copy:
    content: 'All leave requests are now handled by our Intranet site:
https://intranet.example.com/forms'
    dest: F:\shares\forms\leave_requests.txt
```

- 3.6. Exécutez la tâche nommée **Share folder is created** pour partager le dossier **F:\shares\forms**. Les administrateurs ont besoin d'un accès complet, les utilisateurs du domaine ont besoin d'un accès en lecture, et la liste doit être activée. Cette tâche utilise le module **win\_copy**.

```
- name: Forms folder is shared
  win_share:
    name: forms
    description: Company forms share
    path: F:\shares\forms
    list: yes
    full: Administrators
    read: Domain Users
```

- 3.7. Le playbook terminé doit s'afficher comme suit.

```
---
- name: Configure local storage
  hosts: win1.example.com

  tasks:
    - name: Disk facts are populated
      win_disk_facts:

    - name: Second disk extracted as standalone fact
      set_fact:
        second_disk: "{{ ansible_facts['disks'] | json_query('?[number==`1`]') }}"

    - name: Second disk size printed
      debug:
        var: second_disk[0]['size']

    - name: Initialize second disk if required
      win_shell: "Initialize-Disk -Number 1"
      when: second_disk[0]['guid'] is none

    - name: Partition is created on second disk
      win_partition:
        disk_number: 1
        drive_letter: F
        partition_size: 1 GiB
        register: partition_task

    - name: New partition is formatted
```

```

win_format:
  drive_letter: F
  file_system: NTFS
  new_label: data
  full: true
when: partition_task['changed']

- name: Share folder is created
  win_file:
    path: F:\shares\forms
    state: directory

- name: Content added to forms folder
  win_copy:
    content: 'All leave requests are now handled by our Intranet site:
https://intranet.example.com/forms'
    dest: F:\shares\forms\leave_requests.txt

- name: Forms folder is shared
  win_share:
    name: forms
    description: Company forms share
    path: F:\shares\forms
    list: yes
    full: Administrators
    read: Domain Users

```

- ▶ 4. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 4.1. Pour enregistrer le fichier, cliquez sur **File** → **Save** ou appuyez sur **Ctrl+S**.
  - 4.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard du fichier **storage.yml** pour indexer les modifications.
  - 4.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 4.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- ▶ 5. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- ▶ 6. Testez votre playbook à l'aide du modèle de tâche **Run windows project**.
  - 6.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 6.2. Cliquez sur le modèle de tâche **Run windows project**.
  - 6.3. Sélectionnez le playbook **storage.yml** dans la liste **PLAYBOOK**.

- 6.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
- ▶ 7. Une fois toutes les tâches terminées, vérifiez que le statut dans le volet **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec le fichier de solution **solutions/storage.sol** dans le référentiel Git **windows**.
- ▶ 8. Accédez au nouveau partage pour vérifier que le playbook a été correctement utilisé.
- 8.1. Cliquez sur **Recherche Windows**, recherchez **run**, puis ouvrez la fenêtre Run.
  - 8.2. Tapez **\win1** et appuyez ensuite sur Entrée.
  - 8.3. Le partage **forms** s'affiche. Ouvrez le partage, puis ouvrez le fichier **leave\_requests.txt** pour vérifier que le contenu correspond au playbook.
  - 8.4. Fermez le fichier **leave\_requests.txt** et le partage **forms**.
- ▶ 9. Revenez à l'interface utilisateur Web d'Ansible Tower. Modifiez le modèle **Run windows project** pour annuler vos modifications et utilisez le playbook **revert\_storage.yml**, puis lancez une tâche.
- 9.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 9.2. Cliquez sur le modèle de tâche **Run windows project**.
  - 9.3. Sélectionnez le playbook **revert\_storage.yml** dans la liste **PLAYBOOK**.
  - 9.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Automatisation des tâches d'administration Windows

Au cours de cet exercice, vous allez créer et formater une partition, créer une tâche planifiée et installer un module PowerShell.

## Résultats

Vous devez pouvoir créer et formater une partition de disque et la mapper sur une lettre de lecteur, planifier une tâche pour défragmenter le système de fichiers sur ce périphérique, ajouter un référentiel PowerShell Gallery et installer un module PowerShell sur des hôtes gérés à l'aide d'Ansible.

## Avant De Commencer

Vérifiez que **workstation** et toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab dans un navigateur.

Ouvrez votre client RDP et connectez-vous à **workstation** en utilisant **example.com** comme domaine, **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Recherchez les informations de connexion à vos systèmes dans le courrier électronique que vous avez reçu.

1. Lancez l'éditeur Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **windows**, clonez-le sur votre instance **workstation**.
2. Sélectionnez **File** → **Open Folder**, puis le dossier **C:\Users\student\Documents\windows** dans lequel afficher le contenu.
3. Ouvrez le playbook **review.yml**, puis effectuez les tâches de façon à ce qu'elles génèrent les résultats suivants.
  - Créez et formatez une partition NTFS de 2 Gio sur le deuxième disque en tant que lecteur S:.
  - Planifiez une tâche nommée DefragDBDrive pour défragmenter le nouveau lecteur chaque dimanche.
  - Ajoutez le référentiel PowerShell Gallery.
  - Installez le module SqIServer PowerShell.
4. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
5. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
6. Pour tester votre playbook, modifiez le modèle de tâche **Run windows project** pour utiliser le playbook **review.yml**, puis lancez une tâche.

7. Une fois toutes les tâches terminées, vérifiez que le statut dans le volet **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec le fichier de solution **solutions/review.sol** dans le référentiel Git **windows**.
8. Inspectez **win1** pour vérifier la réussite du playbook.
  - Vérifiez que le lecteur S: existe.
  - Vérifiez que la tâche DefragDBDrive est planifiée.
  - Vérifiez que le module SqlServer peut être importé.
9. Revenez à l'interface utilisateur Web d'Ansible Tower. Pour annuler les modifications, modifiez le modèle **Run windows project** pour utiliser le playbook **revert\_review.yml**, puis lancez une tâche.



**Note**

Si vous n'avez pas quitté la console PowerShell à l'étape précédente, l'exécution de ce playbook va échouer.

L'atelier est maintenant terminé.

## ► Solution

# Automatisation des tâches d'administration Windows

Au cours de cet exercice, vous allez créer et formater une partition, créer une tâche planifiée et installer un module PowerShell.

## Résultats

Vous devez pouvoir créer et formater une partition de disque et la mapper sur une lettre de lecteur, planifier une tâche pour défragmenter le système de fichiers sur ce périphérique, ajouter un référentiel PowerShell Gallery et installer un module PowerShell sur des hôtes gérés à l'aide d'Ansible.

## Avant De Commencer

Vérifiez que **workstation** et toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab dans un navigateur.

Ouvrez votre client RDP et connectez-vous à **workstation** en utilisant **example.com** comme domaine, **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Recherchez les informations de connexion à vos systèmes dans le courrier électronique que vous avez reçu.

1. Lancez l'éditeur Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **windows**, clonez-le sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/windows.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training**, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **windows**.  
Pour afficher les fichiers, sélectionnez **Open** dans la fenêtre qui s'affiche après le clonage, puis passez à l'étape suivante.
2. Sélectionnez **File → Open Folder**, puis le dossier **C:\Users\student\Documents\windows** dans lequel afficher le contenu.
3. Ouvrez le playbook **review.yml**, puis effectuez les tâches de façon à ce qu'elles génèrent les résultats suivants.
  - Créez et formatez une partition NTFS de 2 Gio sur le deuxième disque en tant que lecteur S:.
  - Planifiez une tâche nommée DefragDBDrive pour défragmenter le nouveau lecteur chaque dimanche.

- Ajoutez le référentiel PowerShell Gallery.
- Installez le module SqlServer PowerShell.

3.1. Modifiez la tâche **Partition is created on second disk** pour définir les valeurs suivantes.

- Numéro de disque 1
- Lettre de lecteur S
- Dimensionné à 2 Gio

```
- name: Partition is created on second disk
  win_partition:
    disk_number: 1
    drive_letter: "{{ db_drive_letter }}"
    partition_size: 2 GiB
  register: partition_task
```

3.2. Exécutez la tâche **New partition is formatted** pour créer un volume NTFS nommé **db** sur le lecteur S:. La tâche est exécutée de manière conditionnelle, selon si la tâche précédente dont le résultat est apparu dans **changed** a été résolue avec la valeur **true**.

```
- name: New partition is formatted
  win_format:
    drive_letter: "{{ db_drive_letter }}"
    file_system: NTFS
    new_label: db
    full: true
  when: partition_task['changed']
```

3.3. Effectuez la tâche qui crée la tâche de défragmentation planifiée.

```
- name: Defrag scheduled task is created
  win_scheduled_task:
    name: DefragDBDrive
    description: Degragment the database drive
    actions:
      - path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        arguments: >
          -ExecutionPolicy Unrestricted
          -NonInteractive
          -Command {Optimize-Volume -DriveLetter {{ db_drive_letter }} -Defrag -Verbose}
    triggers:
      - type: weekly
        days_of_week: sunday
        start_boundary: '2019-09-29T03:05:00'
    username: SYSTEM
    run_level: highest
    state: present
```

3.4. Terminez la tâche qui enregistre un nouveau référentiel PowerShell. Utilisez la variable **powershell\_gallery\_endpoint** en tant que paramètre **source** du module.

```

...output omitted...
- name: Current repositories are unregistered
  win_shell: Unregister-PSRepository "{{ item }}"
  ignore_errors: true
  loop:
    - PowerShell Gallery
    - PSGallery

- name: PowerShell Gallery repository is added
  win_psrepository:
    source: "{{ powershell_gallery_endpoint }}"
    name: "PowerShell Gallery"
    state: present

```

- 3.5. Effectuez la tâche qui interroge le référentiel PowerShell Gallery. Cette tâche capture la sortie dans la variable **repo\_status**, puis échoue si la commande ne renvoie pas 0. L'instruction **changed\_when: false** indique à Ansible de renvoyer l'état **OK**, car la tâche n'apporte aucune modification au système.

```

- name: PowerShell Gallery repository is queried
  win_shell: Get-PSRepository "PowerShell Gallery"
  register: repo_status
  failed_when: repo_status.rc != 0
  changed_when: false

```

- 3.6. Effectuez la tâche permettant d'installer les modules PowerShell listés dans la liste **powershell\_modules**. Utilisez le module **win\_psmodule** pour effectuer cette tâche.

```

- name: PowerShell modules are installed
  win_psmodule:
    name: "{{ item }}"
    state: present
  loop: "{{ powershell_modules }}"

```

- 3.7. Le playbook terminé doit s'afficher comme suit :

```

---
- name: Configure database host
  hosts: win1.example.com
  vars:
    db_drive_letter: S
    powershell_gallery_endpoint: https://www.powershellgallery.com/api/v2/
    powershell_modules:
      -SqlServer

  tasks:
    - name: Disk facts are populated
      win_disk_facts:

    - name: Second disk extracted as standalone fact

```

```

set_fact:
    second_disk: "{{ ansible_facts['disks'] | json_query('?[?number=='`1`']') }}"

- name: Initialize second disk if required
  win_shell: "Initialize-Disk -Number 1"
  when: second_disk[0]['guid'] is none

- name: Partition is created on second disk
  win_partition:
    disk_number: 1
    drive_letter: "{{ db_drive_letter }}"
    partition_size: 2 GiB
  register: partition_task

- name: New partition is formatted
  win_format:
    drive_letter: "{{ db_drive_letter }}"
    file_system: NTFS
    new_label: db
    full: true
  when: partition_task['changed']

- name: Defrag scheduled task is created
  win_scheduled_task:
    name: DefragDBDrive
    description: Degragment the database drive
    actions:
      - path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        arguments: >
          -ExecutionPolicy Unrestricted
          -NonInteractive
          -Command {Optimize-Volume -DriveLetter {{ db_drive_letter }} -Defrag -Verbose}
    triggers:
      - type: weekly
        days_of_week: sunday
        start_boundary: '2019-09-29T03:05:00'
    username: SYSTEM
    run_level: highest
    state: present

- name: Current PowerShell repositories are unregistered
  win_shell: Unregister-PSRepository "{{ item }}"
  ignore_errors: true
  loop:
    - PowerShell Gallery
    - PSGallery

- name: PowerShell Gallery repository is added
  win_psrepository:
    source: "{{ powershell_gallery_endpoint }}"
    name: PowerShell Gallery
    state: present

- name: PowerShell Gallery repository is queried

```

```

win_shell: Get-PSRepository "PowerShell Gallery"
register: repo_status
failed_when: repo_status.rc != 0
changed_when: false

- name: PowerShell modules for database support are installed
  win_psmodule:
    name: "{{ item }}"
    state: present
  loop: "{{ powershell_modules }}"

```

4. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 4.1. Cliquez sur **File** → **Save** ou appuyez sur **Ctrl+S**.
  - 4.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard du fichier **time.yml** pour indexer les modifications.
  - 4.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 4.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
5. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
6. Pour tester votre playbook, modifiez le modèle de tâche **Run windows project** pour utiliser le playbook **review.yml**, puis lancez une tâche.
  - 6.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 6.2. Cliquez sur le modèle de tâche **Run windows project**.
  - 6.3. Sélectionnez le playbook **review.yml** dans la liste **PLAYBOOK**.
  - 6.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
7. Une fois toutes les tâches terminées, vérifiez que le statut dans le volet **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec le fichier de solution **solutions/review.sol** dans le référentiel Git **windows**.
8. Inspectez **win1** pour vérifier la réussite du playbook.
  - Vérifiez que le lecteur S: existe.
  - Vérifiez que la tâche DefragDBDrive est planifiée.
  - Vérifiez que le module SqlServer peut être importé.
  - 8.1. Connectez-vous à **win1** à l'aide de la connexion Bureau à distance.
  - 8.2. Lancez Explorer, puis vérifiez que le lecteur S: existe.
  - 8.3. Ouvrez Task Scheduler, cliquez sur Task Scheduler Library, puis vérifiez que la tâche DefragDBDrive est listée avec les paramètres corrects.

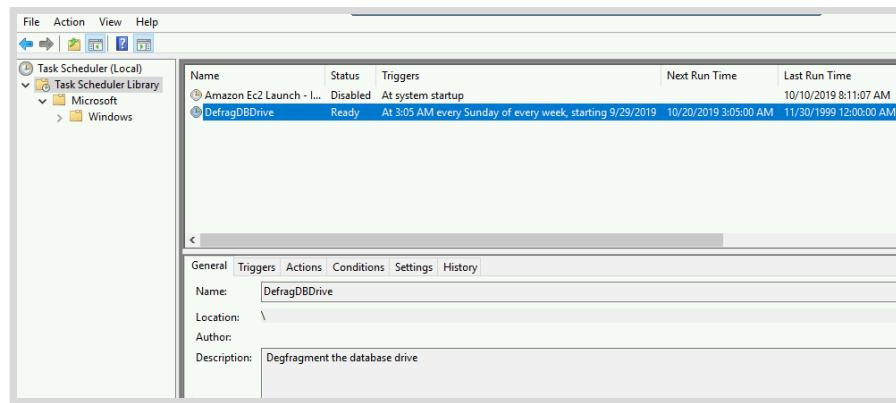


Figure Error1: Liste des tâches du planificateur de tâches

- 8.4. Lancez une console PowerShell, importez le module SqlServer, puis affichez ses commandes. Quittez la console PowerShell lorsque vous avez terminé.

```
PS C:\Users\devops> Import-Module -Name SqlServer
PS C:\Users\devops> Get-Command -Module SqlServer
...output omitted...
Cmdlet      Set-SqlSmartAdmin           21.1.18179 SqlServer
Cmdlet      Start-SqlInstance          21.1.18179 SqlServer
Cmdlet      Stop-SqlInstance          21.1.18179 SqlServer
Cmdlet      Suspend-SqlAvailabilityDatabase 21.1.18179 SqlServer
Cmdlet      Switch-SqlAvailabilityGroup 21.1.18179 SqlServer
Cmdlet      Test-SqlAvailabilityGroup   21.1.18179 SqlServer
Cmdlet      Test-SqlAvailabilityReplica 21.1.18179 SqlServer
Cmdlet      Test-SqlDatabaseReplicaState 21.1.18179 SqlServer
Cmdlet      Test-SqlSmartAdmin         21.1.18179 SqlServer
Cmdlet      Write-SqlTableData        21.1.18179 SqlServer

PS C:\Users\devops>
```

9. Revenez à l'interface utilisateur Web d'Ansible Tower. Pour annuler les modifications, modifiez le modèle **Run windows project** pour utiliser le playbook **revert\_review.yml**, puis lancez une tâche.

**Note**

Si vous n'avez pas quitté la console PowerShell à l'étape précédente, l'exécution de ce playbook va échouer.

- 9.1. Dans le volet de navigation, cliquez sur **Templates**.
- 9.2. Cliquez sur le modèle de tâche **Run windows project**.
- 9.3. Sélectionnez le playbook **revert\_review.yml** dans la liste **PLAYBOOK**.
- 9.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Il existe plusieurs modules pour l'exécution de commandes sous Windows, telles que les modules **win\_command**, **win\_shell** et **win\_psexec**.
- Le module **win\_psexec** est utile lorsque Ansible n'a pas un accès direct aux hôtes cibles. Pour gérer des hôtes sur un réseau isolé, utilisez le module **win\_psexec** sur un hôte de saut.
- Pour des commandes plus complexes, le module **script** vous permet de transférer et d'exécuter un script.
- Ansible fournit le module **win\_scheduled\_task** pour créer et gérer des tâches planifiées sur des noeuds gérés.
- Il existe un large éventail de modules Ansible pouvant être utilisés pour obtenir des informations sur les périphériques de stockage, pour partitionner, formater et défragmenter des disques, ainsi que pour mapper des périphériques sur des lettres de lecteur.

## chapitre 10

# Gestion de projets volumineux

### Objectif

Écrire des playbooks optimisés pour des projets plus vastes et plus complexes, et qui réutilisent le code d'automatisation existant.

### Résultats

- Gérer des playbooks volumineux en important ou en incluant d'autres playbooks ou tâches à partir de fichiers, de manière inconditionnelle ou sur la base d'un test conditionnel.
- Créer un rôle afin de permettre la réutilisation du code par différents projets Ansible et l'exécuter dans le cadre de l'un des plays du playbook.
- Sélectionner et récupérer des rôles à partir d'Ansible Galaxy ou d'autres sources, telles qu'un référentiel Git, et de les utiliser dans des playbooks.
- Automatiser des tâches en configurant et en exécutant les ressources de configuration de l'état souhaité de PowerShell (DSC, Desired State Configuration) à partir de votre playbook Ansible.

### Sections

- Inclusion et importation de fichiers (et exercice guidé)
- Création de rôles (et exercice guidé)
- Déploiement de rôles avec Ansible Galaxy (et exercice guidé)
- Intégration d'Ansible avec des ressources de configuration de l'état souhaité (et exercice guidé)

### Atelier

Gestion de projets volumineux

# Inclusion et importation de fichiers

---

## Résultats

À la fin de cette section, vous devez pouvoir gérer des playbooks volumineux en important ou en incluant d'autres playbooks ou tâches à partir de fichiers externes, de manière inconditionnelle ou sur la base d'un test conditionnel.

## Gestion de projets volumineux

Lorsqu'un playbook est long ou complexe, vous pouvez le diviser en fichiers plus petits pour faciliter la gestion. Vous pouvez combiner plusieurs playbooks en un playbook principal de manière modulaire ou insérer des listes de tâches d'un fichier dans un play. Cela peut faciliter la réutilisation de plays ou de séquences de tâches dans différents projets.

## Inclusion ou importation de fichiers

Ansible peut utiliser deux opérations pour importer du contenu dans un playbook. Vous pouvez *include* du contenu, ou vous pouvez *importer* du contenu.

Lorsque vous incluez du contenu, c'est une opération *dynamique*. Ansible traite le contenu inclus lors de l'exécution du playbook.

Lorsque vous importez du contenu, c'est une opération *statique*. Ansible prétraite le contenu importé lors de l'analyse initiale du playbook, avant le début de l'exécution.

## Importation de playbooks

La directive **import\_playbook** vous permet d'importer des fichiers externes contenant des listes de plays dans un playbook. En d'autres termes, il peut y avoir un playbook maître qui importe un ou plusieurs playbooks supplémentaires.

Le contenu importé étant un playbook complet, vous ne pouvez utiliser la fonction **import\_playbook** qu'au niveau supérieur d'un playbook et pas dans un play. Si vous importez plusieurs playbooks, ils sont alors importés et exécutés dans l'ordre.

L'exemple suivant montre comment un playbook master importe deux playbooks supplémentaires :

```
- name: Create users in windows server
  import_playbook: win-create-users.yml

- name: Install msi packages
  import_playbook: install-msi-packages.yml
```

Vous pouvez également entrelacer des plays dans votre playbook maître avec des playbooks importés. Dans l'exemple suivant, le play **Play to install IIS Web Server** s'exécute en premier, suivi par les plays du playbook Ansible **deploy-site.yml** importé.

```
- name: Play to install IIS Web Server
  hosts: winhost1
  tasks:
    - name: Ensure IIS is installed
      win_feature:
        name: "Web-Server"
        state: present
        restart: yes
        include_sub_features: yes
        include_management_tools: yes

    - name: Import playbook plays to deploy website
      import_playbook: deploy-site.yml
```

## Importation et inclusion de tâches

Vous pouvez importer ou inclure des tâches d'un fichier de tâches dans un play. Un fichier de tâches est un fichier contenant une liste plate de tâches.

L'extrait suivant montre le contenu d'un fichier de tâches **windows\_tasks.yml**:

```
- name: Add User
  win_user:
    name: Demo
    password: "S3cr3tP@ssw0rd"
    state: present

- name: Run PowerShell script
  script: files/helloworld.ps1
```

## Importation de fichiers de tâches

Vous pouvez importer de manière statique un fichier de tâches dans un play à l'aide de la fonction **import\_tasks**. L'emplacement de **import\_tasks** dans le playbook détermine l'endroit où les tâches sont insérées et l'ordre dans lequel plusieurs importations sont exécutées.

Voici un exemple de tâches d'importation à partir du fichier de tâches **windows\_tasks.yml** dans un play :

```
- name: Running tasks from imported files
  hosts: winhost1

  tasks:
    - name: Importing external task file
      import_tasks: windows_tasks.yml
```

Lorsque vous importez un fichier de tâches, les tâches de ce fichier sont directement insérées lors de l'analyse du playbook. Étant donné qu'**import\_tasks** importe les tâches de manière statique lorsque le playbook est analysé, cela engendre quelques contraintes :

- Lorsque vous utilisez la fonction **import\_tasks**, des instructions conditionnelles définies lors de l'importation, telles que **when**, sont appliquées à chacune des tâches importées.

- Vous ne pouvez pas utiliser de boucles avec la fonction **import\_tasks**.
- Si vous utilisez une variable pour spécifier le nom du fichier à importer, vous ne pouvez pas utiliser une variable d'inventaire d'hôte ou de groupe.

## Inclusion de fichiers de tâches

Vous pouvez inclure de manière dynamique un fichier de tâches dans un playbook à l'aide de la fonction **include\_tasks**.

Voici un exemple d'inclusion de tâches à partir du fichier de tâches **windows\_tasks.yml** dans un play :

```
- name: Running tasks from included files
  hosts: winhost1
  tasks:
    - name: Importing external task file
      include_tasks: windows_tasks.yml
```

La fonction **include\_tasks** ne traite pas le contenu du playbook tant que le play n'est pas en cours d'exécution et que cette partie du play n'a pas été atteinte. L'ordre dans lequel le contenu du playbook est traité a une incidence sur le comportement de la fonction d'inclusion des tâches :

- Vous pouvez utiliser des boucles avec l'instruction **include\_tasks**. Les tâches ou rôles inclus sont exécutés une fois pour chaque élément de la boucle.
- Lorsque vous utilisez la fonction **include\_tasks**, des instructions conditionnelles telles que **when** définies sur l'inclusion déterminent si les tâches sont incluses dans le play.
- Vous ne pouvez pas utiliser une instruction **notify** pour déclencher un nom de gestionnaire figurant dans un fichier de tâches inclus. Vous pouvez déclencher un gestionnaire dans le playbook principal contenant un fichier de tâches complet. Dans ce cas, toutes les tâches du fichier inclus sont exécutées.



### Note

Vous trouverez une discussion plus détaillée sur les différences de comportement entre **import\_tasks** et **include\_tasks** lorsque des conditions sont utilisées dans le chapitre *Conditions* du *Guide de l'utilisateur Ansible* à l'adresse [https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_conditionals.html#applying-when-to-roles-imports-and-includes](https://docs.ansible.com/ansible/latest/user_guide/playbooks_conditionals.html#applying-when-to-roles-imports-and-includes).

## Cas d'utilisation pour les fichiers de tâches

Dans les exemples suivants, il peut être utile de gérer des ensembles de tâches en tant que fichiers distincts du playbook :

- Si de nouveaux serveurs nécessitent une configuration complète, les administrateurs peuvent créer plusieurs ensembles de tâches pour créer des utilisateurs, installer des fonctions, configurer des services et des priviléges, définir les accès à un système de fichiers partagé, renforcer les serveurs et installer des mises à jour de sécurité et un agent de surveillance. Chaque ensemble de tâches peut être géré à l'aide d'un fichier de tâches distinct ayant son propre conteneur.

- Si des développeurs gèrent collectivement des serveurs, les administrateurs système et les administrateurs de base de données, chaque entité peut alors écrire son propre fichier de tâches qui peut ensuite être examiné et intégré par le gestionnaire système.
- Si un serveur nécessite une configuration particulière, celle-ci peut être intégrée sous la forme d'un ensemble de tâches qui s'exécutent sur la base d'une condition. En d'autres termes, les tâches ne sont incluses et exécutées que si des critères définis sont remplis.
- Si un groupe de serveurs doit exécuter une tâche ou un ensemble de tâches spécifique, il est possible de les exécuter sur un serveur à la seule condition que le serveur fasse partie d'un groupe d'hôtes particulier.

## Gestion des fichiers de tâches

Vous pouvez créer un répertoire dédié pour les fichiers de tâches, et enregistrer tous les fichiers de tâches dans ce répertoire. Ensuite, votre playbook peut simplement inclure ou importer des fichiers de tâches à partir de ce répertoire. Cela permet de construire un playbook complexe dont la structure et les composants sont faciles à gérer.

## Définition de variables pour les plays et les tâches externes

L'intégration de plays ou de tâches à partir de fichiers externes dans les playbooks à l'aide des fonctions d'importation et d'inclusion d'Ansible améliore considérablement la réutilisation des tâches et des playbooks dans un environnement Ansible. Pour maximiser les possibilités de réutilisation, ces fichiers de tâches et de plays doivent être aussi génériques que possible. Les variables peuvent être utilisées pour paramétriser les éléments de play et de tâche afin d'élargir le champ d'application des tâches et des plays.

Par exemple, la tâche suivante installe la fonction requise pour un service Web, puis démarre le service nécessaire.

```
- name: IIS service installed
  win_feature:
    name: Web-Server
    state: present

- name: IIS service started
  win_service:
    name: W3Svc
    state: started
```

Si vous paramétrez les éléments de fonction et de service comme indiqué dans l'exemple suivant, vous pouvez alors utiliser le fichier de tâches pour l'installation et l'administration d'autres fonctions et leurs services, plutôt que pour le service Web exclusivement.

```
- name: "{{ package }}" service installed
  win_feature:
    name: "{{ package }}"
    state: present

- name: "{{ svc }}" service started
```

```
win_service:  
  name: "{{ svc }}"  
  state: started
```

Par la suite, lors de l'intégration du fichier de tâches dans un playbook, vous définirez les variables à utiliser pour l'exécution de la tâche comme suit :

```
...output omitted...  
tasks:  
- name: Import task file and set variables  
  import_tasks: task.yml  
vars:  
  package: Web-Server  
  svc: W3Svc
```

Ansible rend les variables transmises disponibles pour les tâches importées à partir du fichier externe.

Vous pouvez utiliser la même technique pour rendre les fichiers de plays plus réutilisables. Lors de l'intégration d'un fichier de play dans un playbook, vous transmettez les variables à utiliser pour l'exécution du play comme suit :

```
...output omitted...  
- name: Import play file and set the variable  
  import_playbook: play.yml  
vars:  
  package: Web-Server
```



### Important

Les versions antérieures d'Ansible utilisaient une fonction **include** pour inclure les playbooks et les fichiers de tâches, en fonction du contexte. Cette fonctionnalité est encore disponible, mais considérée comme étant obsolète pour un certain nombre de raisons.

Avant Ansible 2.0, **include** fonctionnait comme une importation statique. Dans Ansible 2.0, elle a été modifiée pour fonctionner de manière dynamique, mais cela a engendré quelques limitations. Dans Ansible 2.1, il est devenu possible pour la fonction **include** d'être dynamique ou statique selon les paramètres de la tâche, une approche déroutante et susceptible de générer des erreurs. Il y avait également des problèmes pour s'assurer qu'**include** fonctionnait correctement dans tous les contextes.

Ainsi, la fonction **include** a donc été remplacée dans Ansible 2.4 par de nouvelles directives telles que **include\_tasks**, **import\_tasks** et **import\_playbook**. Vous trouverez peut-être des exemples de la fonction **include** dans des playbooks plus anciens, mais évitez de les utiliser dans les nouveaux.



### Références

#### &mdash; Documentation Ansible

[https://docs.ansible.com/ansible/latest/user\\_guide/index.html](https://docs.ansible.com/ansible/latest/user_guide/index.html)

## ► Exercice guidé

# Inclusion et importation de fichiers

Dans cet exercice, vous allez inclure et importer des playbooks et des tâches dans un playbook Ansible de niveau supérieur.

## Résultats

Vous devez pouvoir inclure et importer des playbooks et des tâches.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



### Important

Cet exercice utilise des ressources créées dans le chapitre *Interaction avec les utilisateurs et les domaines*. Vous devez terminer les exercices de ce chapitre avant de commencer cet exercice.

- 1. Lancez l'éditeur Visual Studio Code et clonez le référentiel **projects** sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel `https://gitlab.example.com/student/projects.git`. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training**, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **tower**.  
Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers.
- 2. Passez en revue et corrigez le contenu des trois fichiers dans le sous-répertoire **tasks**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.

- 2.1. Accédez au répertoire **tasks** situé au niveau supérieur. Sélectionnez le fichier **environment.yml**, ce fichier contient une tâche pour l'installation du paquetage. Modifiez le fichier comme suit :

```
---
- name: IIS and .Net 4.5 are installed
  win_feature:
    name: "{{ iis_package_names }}"
    include_management_tools: True
    state: present
```

- 2.2. Cliquez sur **Save**. Notez les icônes de statut situées dans le coin inférieur gauche de l'éditeur. Un astérisque s'affiche désormais en regard de **Master**, ce qui indique que des modifications ont été apportées.
- 2.3. Dans le même sous-répertoire, sélectionnez le fichier **firewall.yml** qui contient une tâche pour l'ajout d'une nouvelle règle au pare-feu. Modifiez le contenu du fichier comme suit :

```
---
- name: Firewall rule is enabled
  win_firewall_rule:
    name: HTTP
    localport: "{{ iis_site_port }}"
    action: allow
    direction: in
    protocol: tcp
    state: present
    enabled: yes
```

- 2.4. Dans le même sous-répertoire, sélectionnez le fichier **site.yml**. Ce fichier contient des tâches permettant de créer le site et les répertoires IIS, et de copier le fichier **index.html** dans le répertoire approprié. Parcourez le contenu du fichier :

```
---
- name: Logs directory is created
  win_file:
    path: C:\sites\logs
    state: directory

- name: Site directory is created
  win_file:
    path: "{{ iis_site_path }}"
    state: directory

- name: Index page for site is installed
  win_copy:
    src: files/index.html
    dest: '{{ iis_site_path }}\index.html'

- name: IIS site is created
  win_iis_website:
    name: "{{ iis_site_name }}"
```

```
state: started
port: "{{ iis_site_port }}"
ip: "*"
hostname: "{{ inventory_hostname }}"
application_pool: DefaultAppPool
physical_path: "{{ iis_site_path }}"
parameters: logfile.directory:C:\sites\logs
```

- 2.5. Passez à la vue **Source Control**, puis cliquez sur **+** en regard de tous les fichiers que vous avez modifiés pour indexer les modifications.
  - 2.6. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 2.7. L'icône de statut située dans le coin inférieur gauche de la fenêtre indique qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronize Changes** pour envoyer vos modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.
- 3. Passez en revue et corrigez le contenu du fichier **test.yml** dans le sous-répertoire **plays**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.
- 3.1. Accédez au répertoire **plays** situé au niveau supérieur. Sélectionnez le fichier **test.yml** qui contient un play qui teste les connexions à un service Web. Modifiez le fichier comme suit :

```
---
- name: Test web service
  hosts: win1.example.com
  gather_facts: false

  tasks:
    - name: Connect to internet web server
      win_uri:
        url: "{{ url }}"
      status_code: 200
```

- 3.2. Cliquez sur **Save**. Notez les icônes de statut situées dans le coin inférieur gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été effectuées.
- 3.3. Passez à la vue **Source Control**, puis cliquez sur **+** en regard du fichier **test.yml** pour indexer les modifications.
- 3.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 3.5. L'icône de statut située en bas à gauche indique qu'une modification a été apportée au référentiel distant. Cliquez sur **Synchronize Changes** pour envoyer les modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.

- 4. Passez en revue et corrigez le contenu du fichier **deploy.yml** dans le répertoire de niveau supérieur. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.

- 4.1. Accédez au répertoire de niveau supérieur. Sélectionnez le fichier **deploy.yml** qui contient un ensemble de tâches comprenant et important des tâches et des plays de différents fichiers situés dans ce répertoire de projet. Modifiez le fichier comme suit :

```
---
```

```
- name: Configure web server
  hosts: all
  gather_facts: false

  tasks:
    - name: Include the environment task file and set the variables
      include_tasks: tasks/environment.yml
      vars:
        iis_package_names:
          - Web-Server
          - NET-Framework-Core
          - Web-Asp-Net45

    - name: Import the site task file and set the variables
      import_tasks: tasks/site.yml
      vars:
        iis_site_path: C:\sites\projects
        iis_site_name: "D0417-projects"
        iis_site_port: 8080

    - name: Import the firewall task file and set the variable
      import_tasks: tasks/firewall.yml
      vars:
        iis_site_port: 8080

  - name: Import test play file and set the variable
    import_playbook: plays/test.yml
    vars:
      url: 'http://win1.example.com:8080'
```

- 4.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été apportées.
- 4.3. Passez à la vue **Source Control**, puis cliquez sur **+** en regard du fichier **deploy.yml** pour indexer les modifications.
- 4.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 4.5. L'icône de statut située dans le coin inférieur gauche de la fenêtre indique qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronize Changes** pour envoyer les modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.

- 5. Dans les activités précédentes, vous avez créé des sites Web IIS et des règles de pare-feu HTTP. Exécutez le playbook **remove-iis.yml** de la tâche **Run projects project** pour supprimer les sites Web et les règles de pare-feu.
- 5.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 5.2. Dans la liste des modèles, sélectionnez le modèle **Run projects project**.
  - 5.3. Sélectionnez le playbook **remove-iis.yml** dans la liste **PLAYBOOK**.
  - 5.4. Si nécessaire, cochez la case **PROMPT ON LAUNCH** de l'option **LIMIT**.
  - 5.5. Si nécessaire, décochez la case **PROMPT ON LAUNCH** des options **CREDENTIAL** et **INVENTORY**.
  - 5.6. Dans l'option **INVENTORY**, sélectionnez **Dynamic Inventory**.
  - 5.7. Dans l'option **CREDENTIAL**, sélectionnez **AD Administrator**.
  - 5.8. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
  - 5.9. Dans la fenêtre **RUN PROJECTS PROJECT**, saisissez le nom d'hôte **win1.example.com, win2.example.com**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 5.10. Observez la sortie en direct de la tâche en cours d'exécution.
  - 5.11. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- 6. Exécutez la tâche **Run projects project** pour tester votre playbook à l'aide des tâches import et include.
- 6.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 6.2. Dans la liste des modèles, sélectionnez le modèle **Run projects project**.
  - 6.3. Sélectionnez le playbook **deploy.yml** dans la liste **PLAYBOOK**.
  - 6.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
  - 6.5. Dans la fenêtre **RUN PROJECTS PROJECT**, saisissez le nom d'hôte **win1.example.com**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 6.6. Observez la sortie en direct de la tâche en cours d'exécution.
  - 6.7. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- 7. Le dernier play exécute un test à l'aide de l'hôte géré. Si l'état de la tâche réussit, la page Web est alors accessible. En option, vous pouvez ouvrir Chrome à partir de **workstation** et accédez à **http://win1.example.com:8080**. La nouvelle page Web sera disponible.
- 8. Cliquez sur **Log Out** pour vous déconnecter d'Ansible Tower.

L'exercice guidé est maintenant terminé.

# Création de rôles

---

## Résultats

À la fin de cette section, vous devez pouvoir créer un rôle afin de permettre la réutilisation du code par différents projets Ansible et l'exécuter dans le cadre de l'un des plays du playbook.

## Définition d'une structure pour les playbooks Ansible à l'aide de rôles

Au fur et à mesure que vous développerez plus de playbooks, vous découvrirez que vous avez de nombreuses possibilités de réutiliser le code des playbooks que vous avez déjà écrits. Par exemple, un play pour configurer une base de données Microsoft SQL pour une application pourrait être réaffecté, avec des noms d'hôte, des mots de passe et des utilisateurs différents, pour configurer une base de données Microsoft SQL pour une autre application.

Cependant, ce play peut être long et complexe, avec de nombreux fichiers inclus ou importés, et avec des tâches et des gestionnaires permettant de gérer diverses situations. Copier tout ce code dans un autre playbook peut représenter une tâche non négligeable.

Les rôles Ansible vous permettent de réutiliser plus facilement le code Ansible de manière générique. Vous pouvez conditionner, dans une structure de répertoire normalisée, l'ensemble des tâches, variables, fichiers, modèles et autres ressources nécessaires pour déployer l'infrastructure ou des applications. Copiez ce rôle d'un projet à un autre en copiant le répertoire. Appelez ensuite ce rôle depuis un play pour l'exécuter.

Un rôle bien écrit vous permet de transmettre des variables au rôle à partir du playbook qui ajustent son comportement, en définissant tous les noms d'hôte, adresses IP, noms d'utilisateur, secrets spécifiques au site ou autres détails spécifiques. Par exemple, un rôle permettant de déployer un serveur de base de données peut prendre en charge des variables définissant le nom d'hôte, l'utilisateur et le mot de passe de l'administrateur de base de données, ainsi que d'autres paramètres nécessitant une personnalisation pour votre installation. L'auteur du rôle peut également s'assurer que des valeurs par défaut raisonnables sont définies pour ces variables si vous choisissez de ne pas les définir dans le play.

Les rôles Ansible présentent les avantages suivants :

- Les rôles permettent de grouper du contenu et facilitent ainsi le partage de code avec autrui.
- Vous pouvez écrire les rôles afin de définir les éléments essentiels d'un type de système : serveur Web, serveur de base de données, référentiel Git et plus encore.
- Les rôles facilitent la gestion des projets volumineux.
- Différents rôles d'administrateurs et de développeurs en parallèle.

En plus d'écrire, d'utiliser, de réutiliser et de partager vos propres rôles, vous pouvez obtenir de nombreux rôles pris en charge par la communauté à partir du site Web Ansible Galaxy. Vous en apprendrez plus sur ces rôles dans la suite de ce chapitre.

## Discussion du processus de création de rôle

Aucun outil de développement spécial n'est requis pour créer des rôles dans Ansible. Créer et utiliser un rôle est une procédure en trois étapes :

1. Créer la structure du répertoire de rôles.
2. Définir le contenu du rôle.
3. Utiliser le rôle dans un playbook.

### Création de la structure du répertoire de rôles

Par défaut, Ansible recherche des rôles dans le répertoire contenant vos playbooks Ansible, dans le sous-répertoire **roles**. Cela vous permet de stocker des rôles avec le playbook et d'autres fichiers de prise en charge.

Chaque rôle a son propre répertoire avec une structure de répertoire normalisée. Le répertoire de niveau supérieur définit le nom du rôle proprement dit. Les fichiers sont organisés en sous-répertoires nommés en fonction de l'objectif de chaque fichier dans le rôle, tels que **tasks** et **handlers**. Les sous-répertoires **files** et **templates** contiennent des fichiers auxquels les tâches d'autres fichiers YAML font référence.

### Création d'un squelette de rôle

Vous devez créer tous les sous-répertoires et les fichiers manuellement. Cependant, si vous n'utilisez pas un sous-répertoire particulier dans votre rôle, vous n'avez pas besoin de le créer.

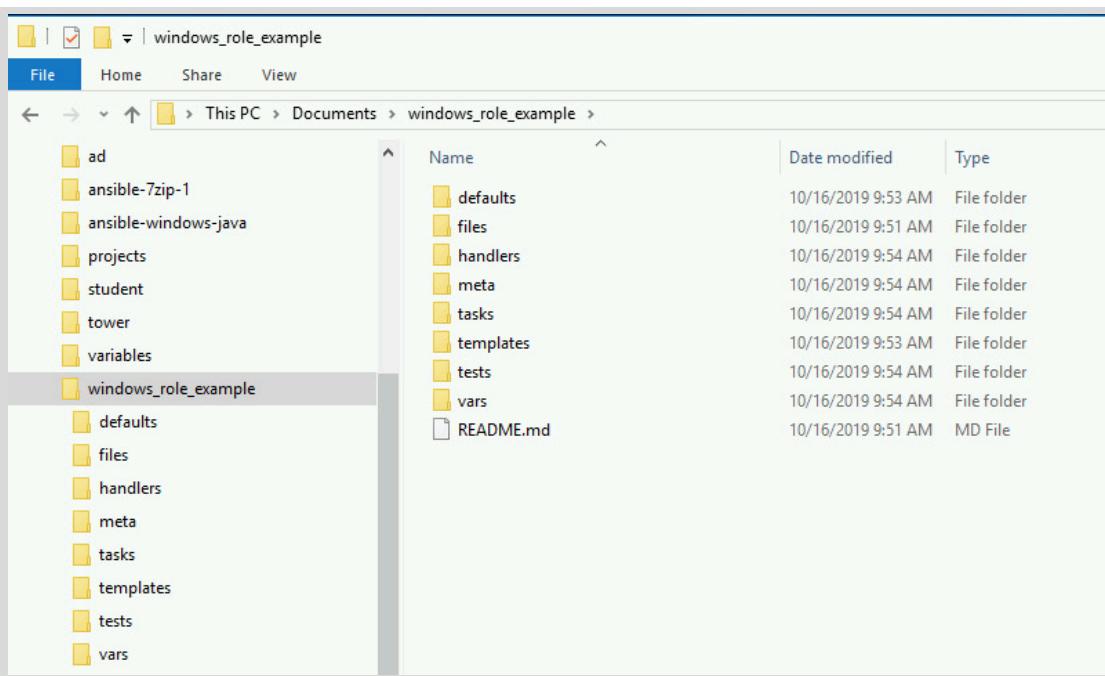


Figure 10.1: Structure du répertoire de rôles

### Examen de la structure du répertoire de rôles

Comme indiqué précédemment, un rôle Ansible est défini par une structure normalisée de sous-répertoires et de fichiers. Les éléments suivants affichent la structure de répertoire classique d'un rôle :

Les éléments suivants affichent la structure de répertoire classique d'un rôle :

```
win_package.example
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Le tableau suivant décrit les sous-répertoires de rôle Ansible :

Répertoire	Fonction
<b>defaults</b>	Le fichier <b>main.yml</b> de ce répertoire contient les valeurs par défaut des variables de rôle. Celles-ci peuvent être remplacées lorsque le rôle est utilisé. Ces variables ont une faible priorité et sont destinées à être modifiées et personnalisées dans les plays.
<b>files</b>	Le répertoire contient des fichiers statiques auxquels les tâches du rôle font référence.
<b>handlers</b>	Le fichier <b>main.yml</b> de ce répertoire contient les définitions du gestionnaire de rôle.
<b>meta</b>	Le fichier <b>main.yml</b> de ce répertoire contient des informations sur le rôle, parmi lesquelles l'auteur, la licence, les plateformes et les dépendances en option du rôle.
<b>tâches</b>	Le fichier <b>main.yml</b> de ce répertoire contient les définitions de tâche du rôle.
<b>Modèles</b>	Le répertoire contient des modèles Jinja2 auxquels les tâches du rôle font référence.
<b>tests</b>	Ce répertoire peut contenir un inventaire et un playbook <b>test.yml</b> qui peuvent être utilisés pour tester le rôle.
<b>vars</b>	Le fichier <b>main.yml</b> de ce répertoire définit les valeurs de variable du rôle. Ces variables sont souvent utilisées à des fins internes au sein du rôle. Ces variables ont une priorité élevée et ne sont pas destinées à être modifiées lorsqu'elles sont utilisées dans un playbook.

Tous les rôles n'auront pas tous ces répertoires. Si un sous-répertoire existe mais qu'il est vide, il est ignoré. Si un rôle n'utilise pas de fonctionnalité, le sous-répertoire peut être complètement omis.

Généralement, un fichier **README.md** à la racine du répertoire de rôle fournit une description, simple et lisible par un humain, du rôle, de la documentation et des exemples d'utilisation, ainsi que toute exigence non Ansible qui doit être respectée.

## Définition du contenu du rôle

Une fois que vous avez créé la structure de répertoire, vous devez écrire le contenu du rôle. Un bon endroit pour commencer est le fichier de tâches **role-name\tasks\main.yml** qui est la liste principale des tâches exécutées par le rôle.

Par exemple, si vous convertissez un play en rôle, copiez toutes les tâches du play dans le fichier **tasks\main.yml**.

Le fichier **tasks\main.yml** suivant vérifie que la fonction **Web-Server** est installée, que le service **W3Svc** est en cours d'exécution et que le fichier **index.html** est basé sur un modèle à l'aide du module **win\_template**.

Étant donné que le module **win\_template** est appelé par une tâche dans un rôle, le modèle **index.j2** est extrait du sous-répertoire **templates**:

```
---
# tasks file for win_package.example role

- name: IIS service installed
  win_feature:
    name: Web-Server
    state: present

- name: IIS service started
  win_service:
    name: W3Svc
    state: started

- name: Website index.html created
  win_template:
    src: index.j2
    dest: C:\Inetpub\wwwroot\index.html
```

La commande suivante affiche le contenu du modèle **index.j2** du rôle, situé dans le répertoire **templates**. Elle fait référence à des faits Ansible et à une variable **system\_owner**.

```
Hello World!
Welcome to {{ ansible_facts['hostname'] }}.

You can contact {{ system_owner }} for more details.
```

Le rôle définit une valeur par défaut pour la variable **system\_owner**. Définissez cette valeur dans le fichier **defaults\main.yml** dans la structure du répertoire du rôle. Le fichier **defaults\main.yml** suivant définit la variable **system\_owner** sur **user@host.example.com**. Il s'agit de

l'adresse électronique inscrite dans le fichier **index.html** des hôtes gérés auxquels ce rôle est appliqué.

```
---
system_owner: user@host.example.com
```

## Pratiques recommandées pour le développement de contenu de rôle

Les rôles permettent aux playbooks d'être écrits de manière modulaire. Pour optimiser l'efficacité des rôles nouvellement développés, pensez à mettre en œuvre les pratiques recommandées suivantes :

- Conservez chaque rôle dans son propre référentiel de contrôle de version. Ansible fonctionne bien avec les référentiels basés sur Git.
- Ne stockez pas les informations sensibles du référentiel de rôles, telles que les mots de passe ou les clés SSH. Paramétrez les valeurs sensibles en tant que variables avec des valeurs par défaut non sensibles. Les playbooks qui utilisent le rôle sont responsables de la définition des variables sensibles par le biais des fichiers de variables Ansible Vault, des questionnaires Ansible Tower, de la fonction « prompt on launch » lorsque vous avez démarré avec un modèle de tâche Ansible Tower, ou d'autres moyens appropriés.
- Créez uniquement les répertoires et les fichiers dont vous avez besoin.
- Créez et maintenez **README.md** et des fichiers **meta/main.yml** pour documenter votre rôle, qui l'a écrit et comment l'utiliser.
- Gardez votre rôle concentré sur un but ou une fonction spécifique. Au lieu de créer un rôle qui accomplit plusieurs choses, écrivez plusieurs rôles.
- Réutilisez et refactorisez les rôles souvent. Résistez à la création de nouveaux rôles pour les configurations de périphérie. Si un rôle existant remplit la majeure partie de la configuration requise, refactorisez-le pour intégrer le nouveau scénario de configuration. Utilisez des techniques de test d'intégration et de régression pour vous assurer que le rôle fournit les nouvelles fonctionnalités requises et ne pose pas de problèmes pour les playbooks existants.

## Définition de variables de rôle et de valeurs par défaut

Pour définir les *variables de rôle*, créez un fichier **vars/main.yml** contenant des paires clé/valeur dans la hiérarchie du répertoire de rôle. Comme pour toute autre variable, des références à ces paires de valeurs sont contenues dans le fichier YAML du rôle : **{{ VAR\_NAME }}**. Ces variables ont une priorité élevée ; il est impossible de les remplacer par des variables d'inventaire. L'intention de ces variables est qu'elles soient utilisées par le fonctionnement interne du rôle.

Les *variables par défaut* autorisent la définition de valeurs par défaut pour les variables pouvant être utilisées dans un play pour configurer le rôle ou personnaliser son comportement. Pour définir ces variables, créez un fichier **defaults/main.yml** contenant des paires clé/valeur dans la hiérarchie du répertoire de rôle. De toutes les variables disponibles, les variables par défaut ont la priorité la plus faible. Elles peuvent être facilement remplacées par d'autres variables, y compris des variables d'inventaire. Ces variables fournissent à la personne qui écrit un play utilisant ce rôle un moyen de personnaliser ou de contrôler exactement la façon dont ce rôle fonctionnera. Elles fournissent également des informations sur le rôle nécessaires pour configurer ou déployer quelque chose correctement.

Définissez une variable spécifique dans **vars\main.yml** ou **defaults\main.yml**, mais pas simultanément. Utilisez des variables par défaut lorsque les valeurs correspondantes doivent être remplacées.



### Important

Les rôles ne devraient pas contenir de données spécifiques aux sites. Ils ne doivent jamais contenir de secrets tels que des mots de passe ou des clés privées.

Les rôles sont supposés être génériques, réutilisables et pouvoir être partagés librement. Les détails spécifiques aux sites ne doivent pas être codés en dur.

Les secrets doivent être fournis au rôle par d'autres moyens. C'est l'une des raisons pour lesquelles vous souhaiterez peut-être définir des variables de rôle en appelant un rôle. Les variables de rôle définies dans le play peuvent fournir le secret, ou pointer vers un fichier chiffré par Ansible Vault contenant le secret.

## Utilisation du rôle dans un playbook

Pour appeler un rôle dans un play, faites-y référence dans la section **roles** : . Le playbook suivant fait référence au rôle **win\_package.example**. Étant donné qu'aucune variable n'est spécifiée, le rôle est appliqué avec ses valeurs de variable par défaut.

```
---
- name: use win_package.example role in play
  hosts: winhost1
  roles:
    - win_package.example
```

Lors de l'exécution du playbook, vous pouvez identifier les tâches résultant d'un rôle par le nom de rôle précédant le nom de la tâche.

Par exemple, la sortie du play ci-dessus peut ressembler à ceci :

```
PLAY [use win_package.example role in play] ****
TASK [setup] ****
ok: [winhost1]

TASK [win_package.example: IIS service installed] ****
changed: [winhost1]

TASK [win_package.example: IIS service started] ****
changed: [winhost1]

TASK [win_package.example: Website index.html created] ****
changed: [winhost1]

PLAY RECAP ****
winhost1 : ok=4     changed=3    unreachable=0   failed=0
```

## Modification du comportement d'un rôle avec des variables

Un rôle bien écrit utilise des variables par défaut pour modifier le comportement d'un rôle, afin qu'il corresponde à un scénario de configuration associé. Cela contribue à rendre le rôle plus générique et réutilisable dans divers contextes.

La valeur de toute variable définie dans le répertoire `defaults` d'un rôle est remplacée si cette même variable est définie :

- dans un fichier d'inventaire, en tant que variable d'hôte ou variable de groupe ;
- dans un fichier YAML sous les répertoires `group_vars` ou `host_vars` d'un projet playbook ;
- en tant que variable, imbriquée dans le mot-clé `vars` d'un play ;
- en tant que variable, lors de l'inclusion du rôle dans le mot-clé `roles` d'un play.

L'exemple suivant illustre l'utilisation du rôle `win_package.example` avec une valeur différente pour la variable du rôle `system_owner`. La valeur spécifiée, `someone@host.example.com`, remplace la référence de variable lors de l'application du rôle à un hôte géré.

```
---
- name: use win_package.example role in play with a variable setting
  hosts: winhost1
  vars:
    system_owner: someone@host.example.com
  roles:
    - win_package.example
```

Lorsqu'elle est définie de cette façon, la valeur de `system_owner` déclarée dans `vars` remplace la valeur de la variable du même nom définie dans le répertoire `defaults`. Cependant, la valeur de `system_owner` ne remplace aucune déclaration dans le répertoire `vars`.

L'exemple suivant illustre l'utilisation du rôle `win_package.example` avec une valeur différente pour la variable de rôle `system_owner`. La valeur spécifiée, `someone@host.example.com`, remplace la référence à la variable même si elle est définie dans le répertoire `vars` ou `defaults` du rôle.

```
---
- name: use windows role in playbook
  hosts: winhost1
  roles:
    - win_package.example
      system_owner: someone@host.example.com
```

**Important**

La priorité des variables peut être source de confusion lorsque vous travaillez avec des variables de rôle dans un play.

- Presque toutes les autres variables remplacent les variables par défaut d'un rôle : variables d'inventaire, play **vars**, paramètres de rôle en ligne, et plus encore.
- Un nombre moins important de variables peut remplacer les variables de rôle définies dans le répertoire **vars**. Les faits, les variables chargées avec **include\_vars**, les variables enregistrées et les paramètres de rôle sont quelques-unes des variables qui peuvent avoir la priorité. Les variables d'inventaire et les **vars** de play ne le peuvent pas. Il s'agit d'un point important, car cela permet d'éviter que votre play ne modifie accidentellement le fonctionnement interne du rôle.
- Toutefois, les variables déclarées en ligne en tant que paramètres de rôle, comme dans le dernier des exemples précédents, présentent une priorité très élevée. Elles peuvent remplacer les variables de rôle définies dans le répertoire **vars**. Si un paramètre de rôle a le même nom qu'une variable définie dans le **vars** d'un play, le **vars** d'un rôle, ou une variable d'inventaire ou de playbook, le paramètre de rôle remplace les autres variables.

## Appel de plusieurs rôles

L'utilisation de plusieurs rôles dans un play est simple :

```
---
- hosts: remote.example.com
  roles:
    - role1
    - role2
```

L'exemple suivant définit les valeurs de deux variables de rôle de **role2**, **var1** et **var2**. Toutes les variables **defaults** et **vars** sont remplacées lorsque **role2** est utilisé.

```
---
- hosts: remote.example.com
  roles:
    - role1
    - role: role2
      vars:
        var1: val1
        var2: val2
```

La syntaxe YAML équivalente que vous pouvez voir utilisée est la suivante :

```
---
- hosts: remote.example.com
  roles:
    - role1
    - { role: role2, vars: { var1: val1, var2: val2 } }
```

Il y a des situations où cette syntaxe peut être plus difficile à lire, même si elle est plus compacte.



### Important

Les variables de rôle définies en ligne (les paramètres de rôle), comme dans les exemples précédents, présentent une priorité très élevée. Elles remplacent la plupart des autres variables.

Veillez à ne pas réutiliser les noms des variables de rôle que vous définissez en ligne ailleurs dans votre play, car les valeurs des variables de rôle remplacent les variables d'inventaire et toute variable **vars** de play.

## Contrôle de l'ordre d'exécution de tâches

Pour chaque play d'un playbook, les tâches s'exécutent dans l'ordre indiqué dans la liste des tâches. Une fois toutes les tâches exécutées, tous les gestionnaires notifiés sont exécutés.

Lorsqu'un rôle est ajouté à un play à l'aide de la directive **roles**, les tâches de ce rôle sont ajoutées au début de la liste des tâches. Si un second rôle est inclus dans un play, la liste de tâches est ajoutée après la liste du premier rôle. L'ordre des directives de **tasks** et de **roles** n'a pas d'importance. Cependant, dans un souci de clarté, il est utile de placer la section **roles** en premier dans le play.

```
---
- hosts: remote.example.com
  roles:
    - role: role1
    - role: role2
  tasks:
    - task1
```

Les gestionnaires de rôles sont ajoutés aux plays de la même manière que les tâches de rôles. Chaque play définit une liste de gestionnaires. Les gestionnaires de rôles sont d'abord ajoutés à la liste des gestionnaires, suivis de tous les gestionnaires définis dans la section **handlers** du play.

Dans certains scénarios, il peut être nécessaire d'exécuter certaines tâches play avant les rôles. Pour prendre en charge de tels scénarios, les plays peuvent être configurés avec une section **pre\_tasks**. Toute tâche listée dans cette section s'exécute avant que des rôles ne soient exécutés. Si l'une de ces tâches notifie un gestionnaire, ces tâches de gestionnaire s'exécutent alors avant les rôles ou les tâches normales.

Les plays prennent également en charge un mot-clé **post\_tasks**. Ces tâches s'exécutent après l'exécution des tâches normales du play et de tous les gestionnaires qu'ils ont notifiés.

Le play suivant illustre les éléments **pre\_tasks**, **roles**, **tasks**, **post\_tasks** et **handlers**.

Dans un scénario de production, il est inhabituel qu'un play contienne toutes ces sections. Bien que la plupart des tâches de cet exemple ne soient pas nommées pour économiser de l'espace, l'omission des noms de tâches n'est pas recommandée lorsque vous travaillez en dehors de cet environnement de test.

```
- name: Play to illustrate order of execution
  hosts: remote.example.com
  pre_tasks:
    - debug:
```

```
    msg: 'pre-task'
    notify: my handler
roles:
  - role1
tasks:
  - debug:
      msg: 'first task'
      notify: my handler
post_tasks:
  - debug:
      msg: 'post-task'
      notify: my handler
handlers:
  - name: my handler
debug:
  msg: Running my handler
```

Dans l'exemple ci-dessus, une tâche **debug** est exécutée dans chaque section pour notifier le gestionnaire **my handler**. La tâche **my handler** est exécutée trois fois :

- après l'exécution de toutes les tâches **pre\_tasks** ;
- après l'exécution de toutes les tâches de rôle et des tâches de la section **tasks** ;
- après l'exécution de toutes les tâches **post\_tasks**.

## Inclusion et importation de rôles

Outre l'ajout de rôles à un play à l'aide de la section **roles**, vous pouvez ajouter des rôles à un play à l'aide d'une tâche ordinaire. Utilisez le module **include\_role** pour inclure dynamiquement un rôle et utilisez le module **import\_role** pour en importer un de manière statique.

Le playbook suivant montre comment un rôle peut être inclus à l'aide d'une tâche avec le module **include\_role**.

```
- name: Execute a role as a task
hosts: remote.example.com
tasks:
  - name: A normal task
    debug:
      msg: 'first task'
  - name: A task to include role2 here
    include_role: role2
```



### Note

Le module **include\_role** a été ajouté dans Ansible 2.3, et le module **import\_role** a été ajouté dans Ansible 2.4.



### Références

**Guide utilisateur Ansible &mdash; Ansible Documentation**

[https://docs.ansible.com/ansible/latest/user\\_guide/index.html](https://docs.ansible.com/ansible/latest/user_guide/index.html)

## ► Exercice guidé

# Création de rôles

Au cours de cet exercice, vous allez créer un rôle Ansible, l'ajouter à un playbook et exécuter le playbook pour tester le nouveau rôle.

## Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Créer un rôle Ansible.
- Utiliser le rôle dans un playbook.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

Tous les exercices guidés Chapitre 8, *Interaction avec les utilisateurs et les domaines* doivent être terminés avant de commencer cet exercice, en raison des modifications de l'inventaire et de l'authentification.

- ▶ 1. Ouvrez l'éditeur Visual Studio Code et clonez le référentiel **projects** sur votre **workstation**.
  - 1.1. À partir de **workstation**, double-cliquez sur l'icône sur le bureau pour ouvrir l'éditeur Visual Studio Code. Vous pouvez aussi cliquer sur **Recherche Windows**, rechercher **code**, puis ouvrir Visual Studio Code.
  - 1.2. Accédez à **View** → **Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.
  - 1.3. À l'invite, fournissez l'URL de référentiel <https://gitlab.example.com/student/projects.git>, puis sélectionnez le dossier **Documents** dans le répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **projects** sur l'instance **workstation**.
  - 1.4. Lorsqu'une invite d'ouverture du projet s'affiche dans le coin inférieur droit de l'éditeur, cliquez sur **Open**.
- ▶ 2. Ouvrez le fichier de playbook **roles-example.yml** pour passer en revue son contenu. Le playbook déclare quelques variables pour insérer des chaînes dans un modèle et une déclaration booléenne **debug\_tools: True**. Les tâches assurent le serveur Web

IIS est démarré, puis déploient des fichiers statiques et un modèle Jinja2. La variable **debug\_tools** est utilisée comme condition pour installer la fonction HTTP Tracing IIS.

- 3. Déplacez toutes les tâches à l'exception de la première et de la dernière dans le fichier **roles-example.yml** dans le fichier de playbook du rôle Web **roles\web\tasks\main.yml**.
- 3.1. Dans le fichier **roles-example.yml**, sélectionnez les quatre tâches commençant par la tâche nommée **IIS Web Server started**.
  - 3.2. Sélectionnez **Edit → Cut** ou appuyez sur **Ctrl+X** pour coller le texte dans le playbook **roles\web\tasks\main.yml** vide.
  - 3.3. Accédez à **Edit → Paste** ou appuyez sur **Ctrl+V** pour coller le texte dans le playbook **roles\web\tasks\main.yml** vide.
  - 3.4. Sélectionnez tout le texte et appuyez deux fois sur **Maj+Tab** pour supprimer la mise en retrait de la tâche.
  - 3.5. Étant donné que les rôles suivent une structure de répertoire prédéterminée, Ansible trouve des modèles et des fichiers dans leurs répertoires respectifs. Supprimez les sous-répertoires **templates/** et **files/** des paramètres **src win\_template** et **win\_copy**.
  - 3.6. Vérifiez que votre **roles\web\tasks\main.yml** correspond à ce qui suit :

```
---
```

```
- name: IIS Web Server started
  win_feature:
    name: Web-Server
    state: present

- name: HTTP Tracing is installed
  win_feature:
    name: Web-Http-Tracing
    state: present
  when: debug_tools

- name: Index page is deployed
  win_template:
    src: index.html.j2
    dest: C:\Inetpub\wwwroot\index.html

- name: Info page is deployed
  win_copy:
    src: info.html
    dest: C:\Inetpub\wwwroot\info.html
```

- 3.7. Enregistrez les modifications dans les fichiers **roles\web\tasks\main.yml** et **roles-example.yml**.

► 4. Déplacez la variable **deployed\_by: Ansible** du playbook **roles-example.yml** vers **roles\web\vars\main.yml**. Les vars de rôle Ansible sont utilisées pour les variables internes au rôle.

  - 4.1. Supprimez la ligne **deployed\_by: Ansible** du playbook **roles-example.yml**.

- 4.2. Ajoutez la déclaration de variable au fichier **roles\web\vars\main.yml**, de sorte qu'il corresponde à ce qui suit :

```
---  
deployed_by: Ansible
```

- 4.3. Enregistrez les fichiers **roles\web\vars\main.yml** et **roles-example.yml** mis à jour.
- ▶ 5. Utilisez les valeurs par défaut du rôle Ansible pour remplacer les variables dans le playbook invocation.

- 5.1. Supprimez les lignes **site\_title: Default Site** et **site\_subtitle: Example Web Site** du playbook **roles-example.yml**.

- 5.2. Ajoutez la déclaration de variable au fichier **roles\web\defaults\main.yml**, de sorte qu'il affiche :

```
---  
site_title: Default Site  
site_subtitle: Welcome!  
debug_tools: False
```

Notez la spécification d'un sous-titre par défaut de **Welcome!**, et la désactivation de **debug\_tools**.

- 5.3. Enregistrez les fichiers **roles\web\defaults\main.yml** et **roles-example.yml**.
- ▶ 6. Cliquez sur le fichier **templates\index.html.j2** dans le volet **PROJECTS**, puis faites-le glisser dans le répertoire **roles\web\templates**.
- ▶ 7. Cliquez sur le fichier **files\info.html** dans le volet **PROJECTS**, puis faites-le glisser dans le répertoire **roles\web\files**.
- ▶ 8. Mettez à jour **roles-example.yml** pour utiliser les rôles, pre\_tasks et post\_tasks comme suit :

```
---
```

```
- name: Deploy website
  hosts: all
  vars:
    debug_tools: True

  pre_tasks: ①
    - debug:
        msg: Installing with debug tools enabled.
        when: debug_tools

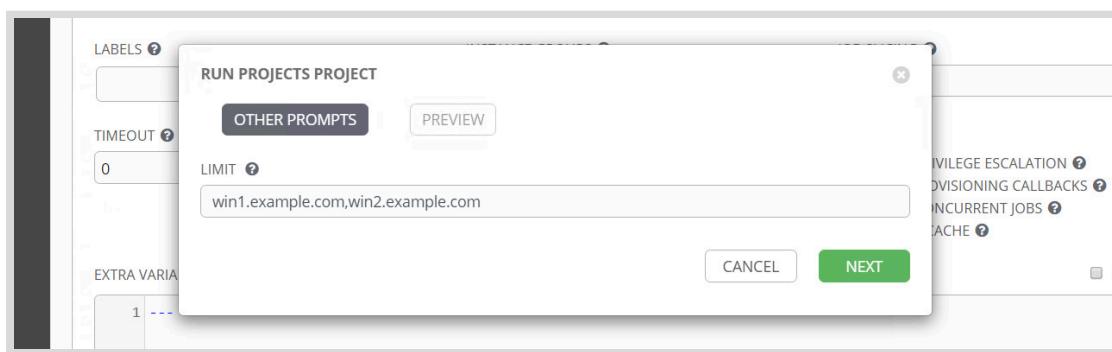
  roles: ②
    - role: web
      site_title: Example Web Site ③

  post_tasks: ④
    - name: Custom page is added
      win_copy:
        content: "Hello!"
        dest: C:\Inetpub\wwwroot\example.html
```

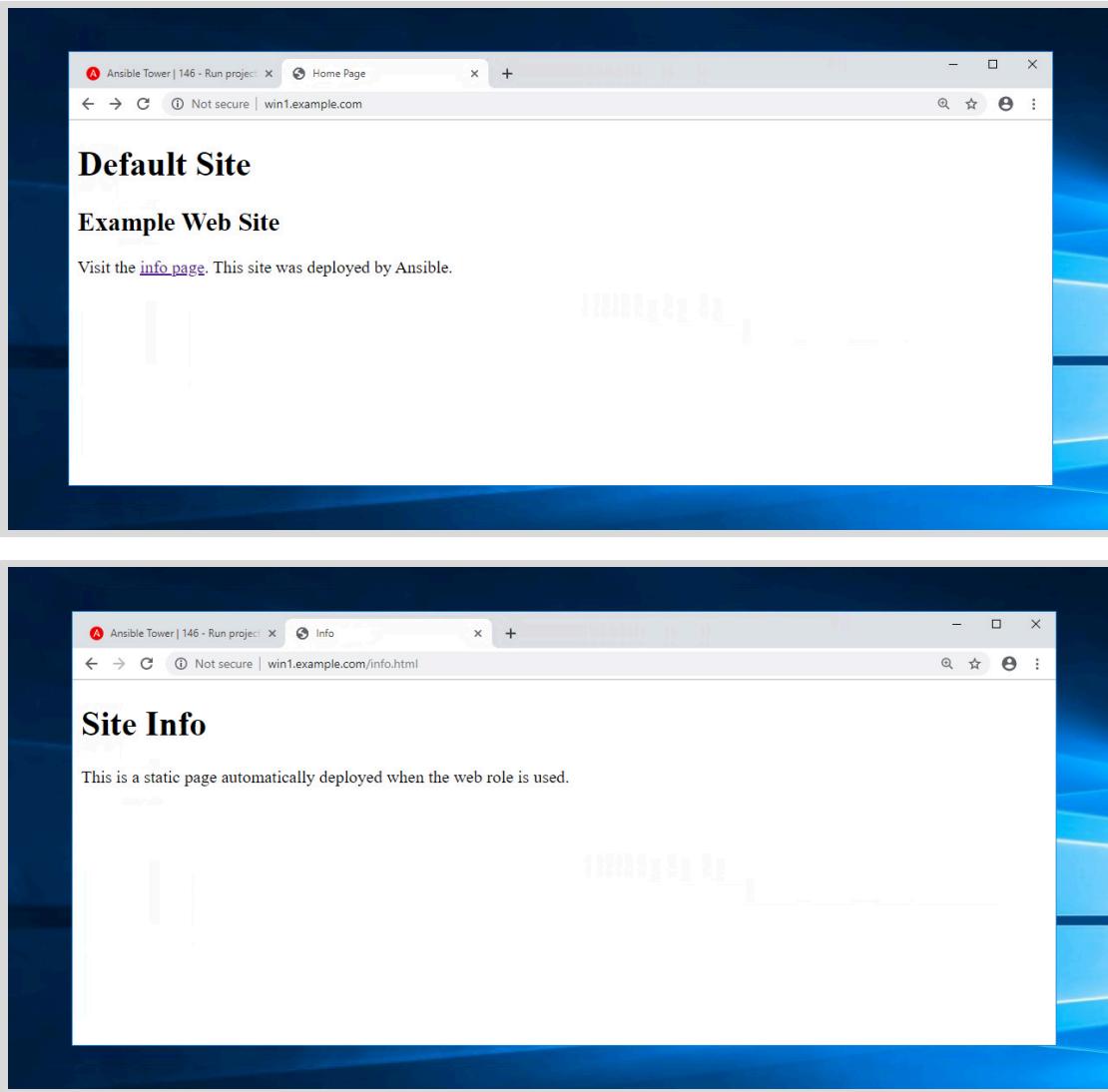
- ➊ Déplacez la tâche de message debug vers **pre\_task**, de sorte qu'elle soit exécutée avant les tâches de rôle.
  - ➋ Ajoutez une liste de rôles avec un seul élément **role: web** appelant le rôle Web.
  - ➌ Remplacez le **site\_title** par défaut pour insérer un titre personnalisé sur le site Web.
  - ➍ Déplacez la tâche **Custom page is added** finale à **post\_task** qu'Ansible appelle une fois que les tâches de rôle sont terminées.
- ▶ 9. Validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
- 9.1. Accédez au volet **Source Control**, puis indexez les modifications.
  - 9.2. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 9.3. Identifiez les icônes de statut situées dans le coin inférieur gauche de l'éditeur et notez qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications.
- ▶ 10. À partir de **workstation**, accédez à votre instance Red Hat Ansible Tower à l'adresse <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- ▶ 11. Exécutez votre playbook à l'aide du modèle de tâche **Run projects project**.
- 11.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 11.2. Cliquez sur le modèle de tâche **Run projects project**.
  - 11.3. Cliquez sur l'icône de loupe correspondant au champ de saisie **INVENTORY**. Cliquez sur **Dynamic inventory**, puis sur **SELECT**.

- 11.4. Cliquez sur l'icône de loupe correspondant au champ de saisie **CREDENTIAL**. Cliquez sur **AD Administrator**, puis sur **SELECT**.
- 11.5. Sélectionnez l'option **PROMPT ON LAUNCH** de la fonction **LIMIT**.
- 11.6. Sélectionnez le playbook **roles-example.yml** dans la liste **PLAYBOOK**.

- 11.7. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
- 11.8. Tapez **win1.example.com,win2.example.com** dans le champ **LIMIT**. Cliquez sur **NEXT**, puis sur **LAUNCH**.



- 11.9. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**.
- ▶ 12. Dans Chrome, ouvrez un nouvel onglet et rendez-vous à l'adresse `http://win1.example.com/`. Cliquez sur le lien « info page » pour vérifier le déploiement du fichier **info.html** statique.



- ▶ 13. Dans Chrome, accédez à `http://win2.example.com/` pour vérifier qu'Ansible a déployé le site Web sur les hôtes gérés `win1` et `win2`.
- ▶ 14. Exécutez votre playbook `cleanup-roles.yml` à partir du modèle de tâche **Run projects project** pour procéder au nettoyage.
  - 14.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 14.2. Cliquez sur le modèle de tâche **Run projects project**.
  - 14.3. Assurez-vous que le champ **INVENTORY** est défini sur **Dynamic inventory**.
  - 14.4. Assurez-vous que le champ **CREDENTIAL** est défini sur **AD Administrator**.
  - 14.5. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **LIMIT** est activée.
  - 14.6. Sélectionnez le playbook `cleanup-roles.yml` dans la liste **PLAYBOOK**.
  - 14.7. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.

L'exercice guidé est maintenant terminé.

# Déploiement de rôles avec Ansible Galaxy

## Résultats

À la fin de cette section, vous devez pouvoir sélectionner et récupérer des rôles à partir d'Ansible Galaxy ou d'autres sources, telles qu'un référentiel Git, et de les utiliser dans des playbooks.

## Présentation d'Ansible Galaxy

Ansible Galaxy [<https://galaxy.ansible.com>] est une bibliothèque publique de contenu Ansible, élaborée par une communauté d'administrateurs et d'utilisateurs Ansible. Elle contient des milliers de rôles Ansible et intègre une base de données interrogable pour aider les utilisateurs d'Ansible à identifier les rôles susceptibles de les aider à effectuer une tâche administrative précise. Ansible Galaxy comprend des liens vers de la documentation et des vidéos à l'intention des nouveaux utilisateurs et des développeurs de rôles Ansible.

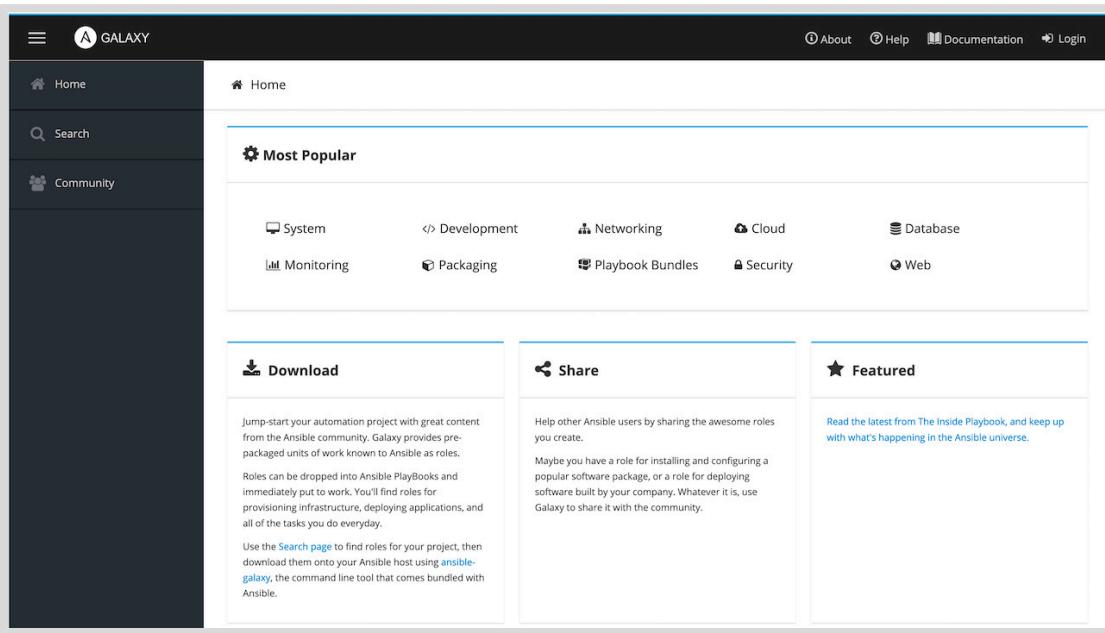


Figure 10.6: Ansible Galaxy

## Aide sur Ansible Galaxy

L'onglet Documentation disponible sur la page d'accueil du site Web Ansible Galaxy décrit l'utilisation d'Ansible Galaxy. Cette page fournit des informations sur le téléchargement et l'utilisation des rôles à partir d'Ansible Galaxy. Elle contient également des instructions relatives au développement de rôles et à leur téléchargement sur Ansible Galaxy.

## Rechercher les rôles dans Ansible Galaxy

L'onglet Search sur la page d'accueil du site Web Ansible Galaxy permet aux utilisateurs d'accéder à des informations relatives aux rôles publiés sur Ansible Galaxy. Vous pouvez rechercher un rôle Ansible en fonction de son nom, en utilisant des balises ou d'autres attributs de rôle.

**chapitre 10 |** Gestion de projets volumineux

De nombreux rôles sur Ansible Galaxy sont conçus pour d'autres systèmes d'exploitation ou périphériques réseau. Utilisez l'onglet **Search** pour rechercher des rôles compatibles avec Microsoft Windows. Par exemple, la recherche du mot-clé ou de la balise « Windows » renvoie un grand nombre de résultats.

Les résultats sont présentés par ordre décroissant selon le score **Best Match**, qui est un score calculé à partir de la qualité du rôle, de la popularité du rôle et de critères de recherche.

**Note**

Consultez la section Évaluation du contenu [[https://galaxy.ansible.com/docs/contributing/content\\_scoring.html](https://galaxy.ansible.com/docs/contributing/content_scoring.html)] de la documentation Ansible Galaxy pour plus d'informations sur la façon dont Ansible Galaxy note les rôles.

Figure 10.7 montre comment utiliser la fonction de recherche permettant de parcourir les rôles. Le menu **Filters** vous permet d'effectuer des recherches par mots-clés, ID d'auteur, plateforme et balises. Les valeurs possibles de la plateforme sont **Windows** pour Microsoft Windows, entre autres.

Les balises sont des chaînes arbitraires d'un seul mot. Définies par l'auteur du rôle, elles décrivent et classent le rôle. Les utilisateurs peuvent utiliser des balises pour trouver des rôles pertinents. Les valeurs possibles pour les balises sont system, development, web, monitoring, etc. Un rôle peut avoir jusqu'à 20 balises dans Ansible Galaxy.

The screenshot shows the Ansible Galaxy search interface. The search bar at the top contains 'mssql'. Below it, there are two main sections: 'Collections' and 'Roles'.

- Collections:** Shows one collection named 'windows' by 'ygo74'. It has 0 Modules, 1 Roles, and 0 Plugins. The role 'mssql' is listed under it.
- Roles:** Shows four roles:
  - 'mssql' by 'lifeofguenter': 4.7 / 5 Score, build: passing, 1570 Downloads, Last Imported: 2 days ago. Tags: development, sql, system.
  - 'mssql' by 'robertdebock': 5 / 5 Score, build: failing, 603 Downloads, Last Imported: 2 days ago. Tags: centos, mssql, opensuse, ubuntu.
  - 'odbc\_driver\_for\_mssql...' by 'amestsantim': 4.3 / 5 Score, 161 Downloads, Last Imported: 9 months ago. Tags: (empty).

On the right side, there are two sidebar sections:
 

- Popular Tags:** A list of tags and their counts:
 

Tag	Count
system	6,252
development	2,997
web	2,539
monitoring	1,366
networking	1,170
database	1,051
cloud	986
ubuntu	834
packaging	824
docker	809
- Popular Platforms:** A list of platforms and their counts:
 

Platform	Count
Ubuntu	88,737
EL	17,305
Debian	34,514

Figure 10.7: Recherche de rôles dans Ansible Galaxy

Ansible Galaxy indique le nombre de fois que chaque rôle a été téléchargé à partir d'Ansible Galaxy. En outre, Ansible Galaxy indique également le nombre d'observateurs, de fourches et d'étoiles du référentiel GitHub qui stocke le rôle. Utilisez ces informations pour déterminer le niveau de développement actif d'un rôle et sa popularité dans la communauté.

Figure 10.8 montre une page de rôle, qui contient des informations telles que les balises, la version Ansible requise par le rôle ou la préparation de la production du rôle.

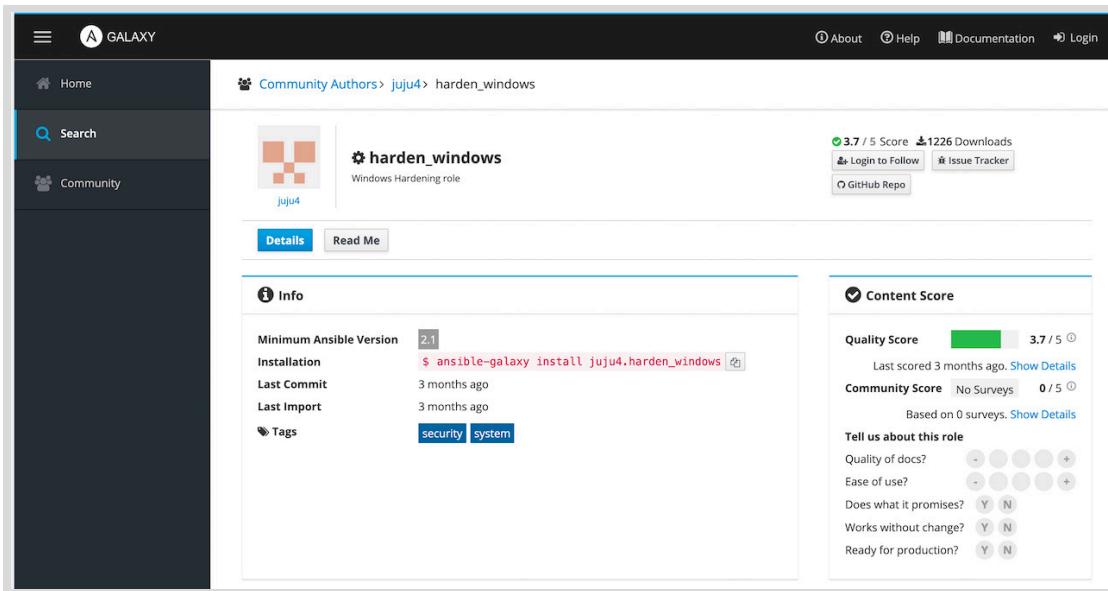


Figure 10.8: Examen des rôles dans Ansible Galaxy

**Important**

Dans l'interface de recherche Ansible Galaxy, les recherches par mot-clé correspondent à des mots ou des expressions dans le fichier **README**, au nom du contenu ou à la description du contenu. Les recherches de balises, en revanche, correspondent spécifiquement aux valeurs de balise définies pour le rôle par l'auteur.

## Installation de rôles à l'aide d'un fichier d'exigences

Ansible Tower permet d'installer une liste de rôles sur la base de définitions dans un fichier texte. Par exemple, si vous avez un playbook qui nécessitent l'installation de rôles spécifiques, vous pouvez créer un fichier **roles/requirements.yml** dans le répertoire du projet qui spécifie les rôles nécessaires.

Utilisez le mot-clé **name** pour remplacer le nom local du rôle. Utilisez le mot-clé **version** pour spécifier la version du rôle. Vous pouvez utiliser le mot-clé **version** avec toute valeur correspondant à une branche, une balise ou un hachage de validation du référentiel logiciel du rôle. L'attribut **src** spécifie la source du rôle, par exemple le rôle **arillso.chocolatey** d'Ansible Galaxy.

L'extrait suivant montre un exemple d'un fichier **roles/requirements.yml** qui installe un rôle spécifique à partir d'Ansible Galaxy :

```
- src: arillso.chocolatey
  name: test.chocolatey
  version: 1.3.2
```

**Important**

Spécifiez la version du rôle dans votre fichier **requirements.yml**, en particulier pour les playbooks en production. Si vous ne spécifiez pas de version, la version la plus récente du rôle est utilisée. Si l'auteur en amont apporte des modifications au rôle incompatibles avec votre playbook, cela peut entraîner une défaillance de l'automatisation ou d'autres problèmes.

Spécifiez les rôles qui ne figurent pas dans Ansible Galaxy dans votre fichier **roles/requirements.yml**, et Ansible Tower peut les récupérer automatiquement. Vous pouvez héberger vos propres rôles internes ou propriétaires dans un référentiel Git privé ou sur un serveur Web. L'exemple suivant montre comment configurer un fichier de configuration à l'aide de diverses sources distantes.

```
# from Ansible Galaxy, using the latest version
- src: geerlingguy.redis

# from Ansible Galaxy, overriding the name and using a specific version
- src: geerlingguy.redis
  version: "1.5.0"
  name: redis_prod

# from any Git-based repository, using HTTPS and a specific commit hash
- src: https://gitlab.com/guardianproject-ops/ansible-nginx-acme.git
  scm: git
  version: 56e00a54
  name: nginx-acme

# from any Git-based repository, using SSH and a specific Git branch
- src: git@gitlab.com:guardianproject-ops/ansible-nginx-acme.git
  scm: git
  version: master
  name: nginx-acme-ssh
```

Le mot-clé **src** spécifie le nom du rôle Ansible Galaxy. Si le rôle n'est pas hébergé sur Ansible Galaxy, le mot-clé **src** clé indique l'URL du rôle.

Si le rôle est hébergé dans un référentiel de contrôle de source, l'attribut **scm** est requis. Ansible Tower est capable de télécharger et d'installer des rôles depuis des référentiels logiciels basé sur Git ou Mercurial. Un référentiel basé sur Git nécessite une valeur **scm** correspondant à **git**, alors qu'un rôle hébergé sur un référentiel Mercurial requiert une valeur **hg**. Si le rôle est hébergé sur Ansible Galaxy ou en tant qu'archive tar sur un serveur Web, le mot-clé **scm** est omis.

**Références**

[Documentation Ansible](#) &mdash; [Documentation Ansible](#)

<https://galaxy.ansible.com/docs/>

## ► Exercice guidé

# Déploiement de rôles avec Ansible Galaxy

Au cours de cet exercice, vous allez utiliser Ansible Galaxy pour télécharger et installer un rôle Ansible à partir de votre serveur Git.

## Résultats

Vous devez pouvoir utiliser Ansible Tower pour télécharger un rôle à partir d'Ansible Galaxy et l'utiliser dans votre environnement.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



### Important

Cet exercice utilise des ressources créées dans Chapitre 8, *Interaction avec les utilisateurs et les domaines*. Vous devez effectuer les exercices de ce chapitre avant de tenter l'activité suivante.

- ▶ 1. Lancez l'éditeur Visual Studio Code et clonez le référentiel **projects** sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/projects.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire **/home/student**, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **projects**.  
Dans la fenêtre qui s'affiche après le clonage des fichiers, cliquez sur **Open**.
- ▶ 2. Examinez et modifiez le contenu du fichier **requirements.yml** dans le sous-répertoire **roles**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.
  - 2.1. Accédez au répertoire **roles** situé au niveau supérieur. Sélectionnez le fichier **\_requirements.yml** et appuyez sur **F2** pour renommer le fichier en

**requirements.yml**. Le trait de soulignement a empêché Ansible de tenter d'analyser le fichier incomplet lors de la synchronisation du projet.

- 2.2. Ouvrez le fichier **requirements.yml**. Ce fichier contient tout ce dont vous avez besoin pour qu'Ansible Tower télécharge le rôle correct. Modifiez le fichier comme suit :

```
- src: https://github.com/arillso/ansible.chocolatey.git
version: 1.3.2
name: ansible-chocolatey-role
```

- 2.3. Cliquez sur **Save**. Notez les icônes de statut situées dans le coin inférieur gauche. L'astérisque sur **Master** indique que des modifications ont été effectuées.
- 3. Passez en revue et corrigez le contenu du fichier **install\_chocolatey.yml** dans le répertoire de niveau supérieur. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.
- 3.1. Accédez au répertoire de niveau supérieur. Sélectionnez le fichier **install\_chocolatey.yml**; ce playbook utilise le rôle du fichier **requirements.yml** pour installer Chocolatey avec des options personnalisées. Vous pouvez trouver ce rôle particulier sur <https://galaxy.ansible.com/arillso/chocolatey>. Modifiez le fichier comme suit :

```
---
- name: Installing chocolatey using a role
hosts: all
tasks:
  - name: Install chocolatey
    vars:
      chocolatey_config:
        - commandExecutionTimeoutSeconds: 2300
      chocolatey_feature:
        - virusCheck: true
        - usePackageRepositoryOptimizations: true
  import_role:
    name: ansible-chocolatey-role
```



### Important

Généralement, un rôle possède de nombreuses variables qui sont définies sur certaines valeurs par défaut. Utilisez l'interface Galaxy pour accéder au fichier **README.md**, qui décrit l'utilisation du rôle, y compris les variables qui personnalisent le comportement du rôle.

- 3.2. Cliquez sur **Save**.
- 3.3. Passez à la vue **Source Control**, puis cliquez sur **+** pour indexer les modifications.
- 3.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 3.5. L'icône de statut située en bas à gauche indique qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronize Changes** pour

envoyer les modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.

- ▶ 4. Démarrez une mise à jour du projet **projects repository** pour synchroniser les modifications que vous avez apportées et pour télécharger automatiquement le rôle défini dans le fichier **requirements.yml** à partir d'Ansible Galaxy.
- 4.1. Dans le volet de navigation, cliquez sur **Projects**.
  - 4.2. Dans la liste des projets, cliquez sur l'icône de synchronisation des modifications du projet **projects repository** pour démarrer le processus de synchronisation.
  - 4.3. Cliquez sur l'icône de statut et observez la mise à jour SCM automatique du projet **projects repository**.



#### Note

Remarquez la tâche **fetch galaxy roles from requirements.yml**. Cette tâche est automatiquement déclenchée à la fin de la synchronisation du projet lorsque Ansible Tower trouve le fichier **roles/requirements.yml**.

- ▶ 5. Exécutez la tâche **Run projects project** pour tester votre playbook et le nouveau rôle.
- 5.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 5.2. Dans la liste des modèles, sélectionnez le modèle **Run projects project**.



#### Note

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 5.3. Sélectionnez le playbook **install\_chocolatey.yml** dans la liste **PLAYBOOK**.
- 5.4. Si nécessaire, cochez la case **PROMPT ON LAUNCH** de l'option **LIMIT**.
- 5.5. Si nécessaire, décochez la case **PROMPT ON LAUNCH** des options **CREDENTIAL** et **INVENTORY**.
- 5.6. Dans l'option **INVENTORY**, sélectionnez **Dynamic inventory**.
- 5.7. Dans l'option **CREDENTIAL**, sélectionnez **AD Administrator**.
- 5.8. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
- 5.9. Dans la fenêtre **RUN PROJECTS PROJECT**, saisissez le nom d'hôte **win1.example.com**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
- 5.10. Observez la sortie en direct de la tâche en cours d'exécution.
- 5.11. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.  
L'installation de Chocolatey avec des options personnalisées à l'aide d'un rôle Ansible Galaxy est terminée.

- ▶ 6. Ajoutez un rôle stocké dans votre référentiel Git local au fichier **requirements.yml**.

- 6.1. Modifiez le contenu du fichier **requirements.yml** dans le sous-répertoire **roles**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.
- 6.2. Accédez au répertoire **roles** situé au niveau supérieur. Sélectionnez le fichier **requirements.yml** qui contient tout ce dont vous avez besoin pour qu'Ansible Tower télécharge le rôle correct. Ajoutez un rôle au fichier ; le rôle a été téléchargé à partir d'Ansible Galaxy et stocké dans votre référentiel Git local. Vous pouvez trouver ce rôle particulier sur <https://galaxy.ansible.com/meyerf99/7zip>. Ajoutez le rôle suivant à la fin du fichier :

```
...output omitted...
- name: Ansible 7-zip Galaxy role
  src: https://gitlab.example.com/student/ansible-7zip.git
  scm: git
  name: ansible-7zip
```

- 6.3. Cliquez sur **Save**. Notez les icônes de statut situées dans le coin inférieur gauche. Un astérisque s'affiche pour **Master**, ce qui indique que des modifications ont été effectuées.
  - 6.4. Passez à la vue **Source Control**, puis cliquez sur **+** pour **requirements.yml** pour indexer les modifications.
  - 6.5. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 6.6. L'icône de statut située en bas à gauche indique qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronize Changes** pour envoyer vos modifications au référentiel distant et pour récupérer toutes les nouvelles modifications à partir du référentiel distant.
- ▶ 7. Passez en revue le contenu du fichier **install\_7zip.yml** dans le répertoire de niveau supérieur.
- 7.1. Accédez au répertoire de niveau supérieur. Sélectionnez le fichier **install\_7zip.yml** ; ce playbook utilise le rôle du fichier **requirements.yml** pour installer 7zip. Il s'affiche comme suit :

```
---
- name: Installing 7zip using a local role
  hosts: all
  tasks:
    - name: Install 7zip
      import_role:
        name: ansible-7zip
```

- ▶ 8. Démarrez une mise à jour du projet **projects repository** pour synchroniser les modifications que vous avez apportées et pour télécharger automatiquement le rôle défini dans le fichier **requirements.yml** à partir d'Ansible Galaxy.
- 8.1. Dans le volet de navigation, cliquez sur **Projects**.
  - 8.2. Dans la liste des projets, cliquez sur l'icône de synchronisation des modifications du projet **projects repository** pour démarrer le processus de synchronisation.

- 8.3. Cliquez sur l'icône de statut et observez la mise à jour SCM automatique du projet **projects repository**.



### Note

Remarquez la tâche **fetch galaxy roles from requirements.yml**. Cette tâche est automatiquement déclenchée à la fin de la synchronisation du projet lorsque Ansible Tower trouve le fichier **roles/requirements.yml**.

- 9. Le rôle 7zip que vous avez synchronisé utilise le module **win\_msi** obsolète, ainsi qu'une version antérieure de 7zip. Dans les étapes suivantes, vous modifiez le rôle pour remplacer le module obsolète et la version changéthe de 7zip par une plus récente.
- 9.1. Lancez l'éditeur Visual Studio Code et clonez le référentiel **ansible-7zip** sur votre instance **workstation**.
- 9.2. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
- 9.3. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/ansible-7zip.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire **/home/student**, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **ansible-7zip**.  
Dans la fenêtre qui s'affiche après le clonage, cliquez sur **Open** pour afficher les fichiers.
- 9.4. Passez en revue et corrigez le contenu du fichier **main.yml** dans le répertoire **tasks**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.
- 9.5. Accédez au répertoire **tasks**. Sélectionnez le fichier **main.yml**; ce fichier contient un ensemble de tâches nécessaires au téléchargement et à l'installation de 7zip.  
Remplacez toutes les occurrences du module **win\_msi** par le module **win\_package**. Remplacez la version 7zip de **1604** par **1900**. Le fichier **main.yml** avec toutes les modifications s'affiche comme suit :

```
---  
# tasks file for 7zip  
- name: create directory on localhost  
  file:  
    path: /tmp/ansible  
    state: directory  
  delegate_to: localhost  
  
- name: download 7zip from URL  
  get_url:  
    url: http://7-zip.org/a/7z1900-x64.msi  
    dest: /tmp/ansible  
  delegate_to: localhost  
  
- name: create directory
```

```

win_file:
  path: C:\ansible
  state: directory

- name: copy 7zip from localhost to windows server
  win_copy:
    src: /tmp/ansible/7z1900-x64.msi
    dest: C:\ansible\7z1900-x64.msi

- name: install 7 zip
  win_package:
    path: 'C:\ansible\7z1900-x64.msi'
    state: present
  when: not sevenzip_uninstall

- name: uninstall 7 zip
  win_package:
    path: 'C:\ansible\7z1900-x64.msi'
    state: absent
  when: sevenzip_uninstall

- name: remove 7zip install folder
  win_file:
    path: C:\ansible
    state: absent

- name: remove 7zip temp destination
  file:
    path: /tmp/ansible
    state: absent
  delegate_to: localhost

```

- 9.6. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, indiquant que des modifications ont été effectuées.
  - 9.7. Passez à la vue **Source Control**, puis cliquez sur **+** en regard du fichier **main.yml** pour indexer les modifications.
  - 9.8. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 9.9. Les icônes de statut situées en bas à gauche indiquent qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronize Changes** pour envoyer les modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.
- 10. Étant donné que vous avez modifié le rôle, Ansible Tower doit télécharger la version correcte du rôle à partir du référentiel. Les étapes suivantes décrivent comment modifier le fichier **requirements.yml**, de sorte qu'AnsibleTower utilise la version correcte du rôle.
- 10.1. Le numéro de validation est à présent le numéro de version du rôle. Pour trouver le numéro de validation correct dans l'éditeur de code de Visual Studio Code, cliquez sur **master** dans la barre de statut.

- 10.2. Dans la fenêtre récemment ouverte, notez le numéro en regard de **master**. Il s'agit du numéro de la dernière validation avec votre modification. Par exemple, le numéro peut être **b967f3e4**.
- 10.3. Modifiez le répertoire de travail dans Visual Studio Code. Cliquez sur **Fichier**, puis sur **Open Folder** et sélectionnez le répertoire **projects**.
- 10.4. Modifiez le contenu du fichier **requirements.yml** dans le sous-répertoire **roles**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.
- 10.5. Accédez au répertoire **roles** situé au niveau supérieur. Sélectionnez le fichier **requirements.yml**. Ajoutez le numéro de version de validation en tant que paramètre pour la version du rôle. Modifiez le fichier comme suit :

```
...output omitted...
- name: Ansible 7-zip Galaxy role
  src: https://gitlab.example.com/student/ansible-7zip.git
  scm: git
version: b967f3e4 <--INSERT THE COMMIT VERSION NUMBER HERE
  name: ansible-7zip
```

- 10.6. Cliquez sur **Save**. Notez les icônes de statut situées dans le coin inférieur gauche. Un astérisque s'affiche en regard de **master**, ce qui indique que des modifications ont été apportées.
  - 10.7. Passez à la vue **Source Control**, puis cliquez sur **+** en regard de **requirements.yml** pour indexer les modifications.
  - 10.8. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 10.9. L'icône de statut située dans le coin inférieur gauche indique qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronise Changes** pour envoyer vos modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.
- 11. Démarrez une mise à jour du projet **projects repository** pour synchroniser les modifications que vous avez apportées et pour télécharger automatiquement le rôle défini dans le fichier **requirements.yml** à partir d'Ansible Galaxy.
- 11.1. Dans le volet de navigation, cliquez sur **Projects**.
  - 11.2. Dans la liste des projets, cliquez sur l'icône de synchronisation des modifications du projet **projects repository** pour démarrer le processus de synchronisation.
  - 11.3. Cliquez sur l'icône de statut et observez la mise à jour SCM automatique du projet **projects repository**.



#### Note

Remarquez la tâche **fetch galaxy roles from requirements.yml**. Cette tâche est automatiquement déclenchée à la fin de la synchronisation du projet lorsque Ansible Tower trouve le fichier **roles/requirements.yml**.

- 12. Exécutez la tâche **Run projects project** pour tester votre rôle mis à jour.

- 12.1. Dans le volet de navigation, cliquez sur **Templates**.
- 12.2. Dans la liste des modèles, sélectionnez le modèle **Run projects project**.



**Note**

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 12.3. Sélectionnez le playbook **install\_7zip.yml** dans la liste **PLAYBOOK**.
- 12.4. Si nécessaire, cochez la case **PROMPT ON LAUNCH** de l'option **LIMIT**.
- 12.5. Si nécessaire, décochez la case **PROMPT ON LAUNCH** des options **CREDENTIAL** et **INVENTORY**.
- 12.6. Dans l'option **INVENTORY**, sélectionnez **Dynamic inventory**.
- 12.7. Dans l'option **CREDENTIAL**, sélectionnez **AD Administrator**.
- 12.8. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
- 12.9. Dans la fenêtre **RUN PROJECTS PROJECT**, saisissez le nom d'hôte **win1.example.com**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
- 12.10. Observez la sortie en direct de la tâche en cours d'exécution.
- 12.11. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.

- 13. Vous avez modifié un rôle et utilisé ce rôle pour installer 7zip sur l'un des hôtes gérés.  
Cliquez sur l'icône **Log Out** pour vous déconnecter d'Ansible Tower.

L'exercice guidé est maintenant terminé.

# Intégration d'Ansible avec des ressources de configuration de l'état souhaité

## Résultats

À la fin de cette section, vous devez pouvoir configurer et exécuter les ressources de configuration de l'état souhaité de PowerShell (DSC, Desired State Configuration) à partir de votre playbook Ansible afin d'automatiser les tâches.

## Présentation de la configuration de l'état souhaité

La plateforme DSC (*Desired State Configuration*) permet la gestion de la configuration du système et est intégrée à PowerShell qui utilise un modèle déclaratif. Elle utilise une exécution en mode push pour envoyer des configurations aux hôtes cibles par le biais du code. Cette plateforme de gestion de la configuration est exécutée différemment d'Ansible et est spécifique à la plateforme Windows. DSC utilise un gestionnaire de configuration local qui s'exécute sur tous les nœuds distants en tant que moteur d'exécution DSC.

Microsoft encourage les efforts de la communauté en matière de création et de maintenance des ressources DSC pour de nombreuses technologies. Ces ressources sont publiées chaque mois dans PowerShell Gallery en tant que kit de ressources DSC, et sont disponibles à partir du site Github : <https://github.com/PowerShell/DscResources>.

Le module Ansible **win\_dsc** a été introduit dans Ansible 2.4 et permet à Ansible d'utiliser les ressources DSC existantes pour les hôtes Windows. La configuration minimale requise pour exécuter ce module sur les hôtes est PowerShell v5.0 ou version ultérieure.

## Choix des modules Ansible ou des ressources DSC

Les ressources DSC et les modules Ansible partagent le même objectif : définir et garantir l'état correct d'une ressource système. En tant que tels, ils peuvent être des technologies complémentaires. Étant donné qu'Ansible peut exécuter des ressources DSC à partir d'un playbook, vous pouvez utiliser l'outil Ansible multiplateforme pour gérer l'automatisation de manière centralisée et standardisée, tout en tirant parti des ressources DSC spécifiques à Windows dans l'automatisation de PowerShell Gallery pour Windows.

Les ressources DSC prennent en charge la réutilisation du code, à l'instar des rôles Ansible, mais fournissent une bibliothèque plus importante de code d'automatisation existant et pris en charge par la communauté que l'utilisation de modules Ansible.

Dans de nombreux cas, vous pouvez choisir entre un module Ansible ou une ressource DSC pour automatiser une tâche particulière. Les listes suivantes décrivent certains critères que vous pouvez utiliser lorsque vous décidez de l'approche à utiliser :

### Raisons d'utiliser des modules Ansible

- L'hôte ne prend pas en charge PowerShell v5.0, ou vous ne pouvez pas le mettre à niveau facilement.
- Le module Ansible que vous utilisez possède une fonction dont est dépourvue la ressource DSC.

- Les vérifications d'idempotence du module Ansible sont mieux adaptées à vos besoins que la prise en charge limitée du mode de vérification DSC.
- Le module Ansible en question prend en charge le mode diff, et les ressources DSC ne le font pas.
- Il existe des bogues dans une ressource DSC, mais le module Ansible fonctionne.

### Raisons d'utiliser des ressources DSC via Ansible

- Vous connaissez mieux les ressources DSC et vous êtes plus à l'aise avec la manière dont elles fonctionnent.
- Vous souhaitez utiliser des ressources DSC et des modules Ansible dans les mêmes exécutions d'automatisation ou dans des opérations d'automatisation multiplateforme.
- Le module Ansible ne prend pas en charge une fonction présente dans une ressource DSC.
- Aucun module Ansible n'est disponible.
- Il existe des bogues dans le module Ansible, mais la ressource DSC fonctionne.

## Liste des ressources DSC

Voici des exemples de ressources DSC intégrées disponibles pour la gestion de Windows :

- Archive
- Fichier
- Groupe
- Paquetage
- WindowsFeature

Vous pouvez trouver la liste de toutes les ressources DSC à l'adresse <https://docs.microsoft.com/en-us/powershell/scripting/dsc/resources/resources?view=powershell-6#built-in-resources>.

Microsoft et la Communauté via le kit de ressources DSC, contiennent de nombreuses ressources DSC personnalisées, notamment :

- Contrôleur de domaine
- Site Web IIS
- Cluster SQL Server
- Cluster de basculement
- DNS

Des ressources personnalisées supplémentaires sont disponibles à l'adresse <https://www.powershellgallery.com>.

## Exécution des ressources DSC dans un playbook Ansible

Il est facile d'utiliser le module **win\_dsc** pour piloter un fournisseur de ressource DSC unique, tel qu'un module Ansible.

Dans le premier exemple, cet extrait de la section **tasks** d'un play utilise des modules Ansible pour installer IIS (Internet Information Services) et créer une page Web :

```
- name: Install IIS Web-Server
  win_feature:
    name: Web-Server
    state: present
    restart: True
    include_sub_features: True
    include_management_tools: True

- name: Create IIS site
  win_iis_website:
    name: Ansible
    state: started
    physical_path: C:\website\DO417

- name: Add HTTP webbinding to IIS
  win_iis_webbinding:
    name: Ansible
    protocol: http
    port: 8080
    ip: '*'
    state: present
```

Utilisez le module **win\_dsc** pour accomplir les mêmes tâches à l'aide des ressources DSC :

```
- name: Install required DSC module:
  win_psmodule:
    name: xWebAdministration①
    state: present

- name: Install IIS Web-Server
  win_dsc:
    resource_name: windowsfeature②
    name: Web-Server

- name: Create IIS site and add HTTP webbinding
  win_dsc:
    resource_name: xWebsite③
    Ensure: Present
    Name: Ansible
    State: Started
    PhysicalPath: C:\website\DO417
    BindInfo:
      - Protocol: http
        Port: 8080
        IPAddress: '*'
```

Dans le code précédent, Ansible installe le module de ressources DSC **xWebAdministration** à l'aide du module **win\_psmodule**. Avec les ressources DSC correctes présentes désormais, la ressource **windowsfeature** est appelée pour installer la fonctionnalité **Web-Server**. La

ressource **xWebsite** est appelée dans la dernière tâche pour créer le site Web IIS et pour ajouter une liaison HTTP au serveur Web IIS.

- ➊ Le module **win\_psmodule** d'Ansible installe le module de ressources DSC **xWebAdministration**.
- ➋ La ressource **windowsfeature** est appelée pour installer la fonction **Web-Server**.
- ➌ La ressource **xWebsite** est appelée dans la dernière tâche, pour créer le site Web IIS et ajouter une liaison HTTP à IIS.

## Exécution de DSC à partir d'Ansible en tant qu'utilisateur non-SYSTEM

Par défaut, DSC exécute chaque ressource à l'aide du compte **SYSTEM**, et pas du compte qu'Ansible utilise pour exécuter le module. Cela signifie que les ressources chargées de manière dynamique, basées sur un profil utilisateur, telles que la ruche de Registre **HKEY\_CURRENT\_USER**, sont chargées sous le profil **SYSTEM**.

Le paramètre **PsDscRunAsCredential** peut être défini pour chaque ressource DSC, forçant ainsi le moteur DSC à s'exécuter sous un compte différent. **PsDscRunAsCredential** a un type de **PSCredential** que vous configurez avec les suffixes **username** et **password**.

En utilisant le type de ressource du Registre comme exemple, l'exemple suivant illustre la définition d'une tâche pour accéder à la ruche **HKEY\_CURRENT\_USER** de l'utilisateur Ansible :

```
- name: use win_dsc with PsDscRunAsCredential to run as a different user
  win_dsc:
    resource_name: Registry
    Ensure: Present
    Key: HKEY_CURRENT_USER\ExampleKey
    ValueName: TestValue
    ValueData: TestData
    PsDscRunAsCredential_username: '{{ansible_user}}'
    PsDscRunAsCredential_password: '{{ansible_password}}'
    no_log: true
```



### Références

**Documentation Ansible – Configuration de l'état souhaité**

[https://docs.ansible.com/ansible/latest/user\\_guide/windows\\_dsc.html](https://docs.ansible.com/ansible/latest/user_guide/windows_dsc.html)

## ► Exercice guidé

# Intégration d'Ansible avec des ressources de configuration de l'état souhaité

Au cours de cet exercice, vous allez configurer et utiliser une ressource DSC à partir d'un playbook Ansible.

## Résultats

Vous devez pouvoir utiliser le module **win\_dsc** pour créer des fichiers et des répertoires.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



### Important

Cet exercice utilise des ressources créées dans Chapitre 8, *Interaction avec les utilisateurs et les domaines*. Vous devez effectuer les exercices de ce chapitre avant de tenter l'activité suivante.

- ▶ 1. Lancez l'éditeur Visual Studio Code et clonez le référentiel **projects** sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel. Utilisez l'URL de référentiel `https://gitlab.example.com/student/projects.git`. À l'invite, sélectionnez le dossier **Documents** du répertoire `/home/student`, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **projects**.  
Dans la fenêtre qui s'affiche après le clonage, cliquez sur **Open** pour afficher les fichiers.
- ▶ 2. Passez en revue et corrigez le contenu du fichier **dsc\_usage.yml**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.

- 2.1. Sélectionnez le fichier **dsc\_usage.yml**; ce fichier contient des tâches de création d'un répertoire, de création d'un fichier et d'extraction d'une archive zip. Modifiez le fichier comme suit :

```
...output omitted...
- name: Create a new directory
  win_dsc:
    resource_name: File
    DestinationPath: C:\D0417\dsc-usage-example
    Ensure: Present
    Type: Directory

- name: Create a new file with text
  win_dsc:
    resource_name: File
    DestinationPath: C:\D0417\dsc-usage-example\new-file.txt
    Contents: |
      Hello
      This is the D0417 course.
    Ensure: Present
    Type: File

- name: Copy a zip file to the managed host
  win_copy:
    src: do417-example.zip
    dest: C:\D0417\dsc-usage-example\do417-example.zip

- name: Extract a zip file in the new directory
  win_dsc:
    resource_name: Archive
    Destination: C:\D0417\dsc-usage-example\extracted-zip
    Path: C:\D0417\dsc-usage-example\do417-example.zip
    Ensure: Present
```

- 2.2. Cliquez sur **Save**. Notez les icônes de statut situées en bas à gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été effectuées.
- 2.3. Passez à la vue **Source Control**, puis cliquez sur **+** en regard du fichier modifié pour indexer les modifications.
- 2.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 2.5. L'icône de statut située dans le coin inférieur gauche indique qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronize Changes** pour envoyer vos modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.
- 3. Exécutez la tâche **Run projects project** pour tester votre playbook à l'aide des modules **win\_dsc**.

- 3.1. Dans le volet de navigation, cliquez sur **Templates**.

- 3.2. Dans la liste des modèles, sélectionnez le modèle **Run projects project**.

**Note**

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 3.3. Sélectionnez le playbook **dsc\_usage.yml** dans la liste **PLAYBOOK**.
  - 3.4. Si nécessaire, cochez la case **PROMPT ON LAUNCH** de l'option **LIMIT**.
  - 3.5. Si nécessaire, décochez la case **PROMPT ON LAUNCH** des options **CREDENTIAL** et **INVENTORY**.
  - 3.6. Dans l'option **INVENTORY**, sélectionnez **Dynamic inventory**.
  - 3.7. Dans l'option **CREDENTIAL**, sélectionnez **AD Administrator**.
  - 3.8. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
  - 3.9. Dans la fenêtre **RUN PROJECTS PROJECT**, saisissez le nom d'hôte **win1.example.com**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 3.10. Observez la sortie en direct de la tâche en cours d'exécution.
  - 3.11. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- ▶ 4. Cliquez sur l'icône **Log Out** pour vous déconnecter d'Ansible Tower.
- ▶ 5. Connectez-vous à l'hôte **win1.example.com** en tant qu'utilisateur **devops** avec le mot de passe **RedHat123@!**. Vérifiez que vous disposez d'un répertoire **C:\DO417\dsc-usage-example** contenant un fichier **new-file.txt**, ainsi qu'un sous-répertoire **extracted-zip** contenant le fichier **Example-of-a-extracted-file.txt** dézippé.

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Gestion de projets volumineux

### Liste de contrôle des performances

Au cours de cet atelier, vous allez créer des rôles et des playbooks Ansible qui utilisent des rôles Ansible et des ressources DSC pour automatiser les tâches reproductibles.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Modifier un playbook pour utiliser les ressources DSC.
- Convertir le playbook en rôle.
- Créer un playbook qui utilise le rôle pour créer des répertoires et installer une fonction Windows.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

Tous les exercices guidés Chapitre 8, *Interaction avec les utilisateurs et les domaines* doivent être terminés avant de commencer cet exercice, en raison des modifications de l'inventaire et de l'authentification.

1. Modifiez le playbook **projects-review.yml** dans le référentiel **projects** et procédez aux modifications indiquées. Validez et transmettez par push les modifications au référentiel Git. Les commentaires figurant dans le playbook indiquent les modifications à apporter.
  - Utilisez DSC pour créer la structure de répertoire appropriée et pour décompresser l'archive fournie.

Utilisez DSC pour installer la fonctionnalité Windows **telnet-client**.

  - Mettez à jour la première tâche avec le module Ansible correct et les options du module nécessaires pour créer le répertoire défini.
  - Mettez à jour la troisième tâche avec le module Ansible correct et les options du module pour installer la fonction Windows **telnet-client**.
2. Convertissez le playbook **projects-review.yml** que vous avez modifié en rôle nommé **projects-review**. Les répertoires et fichiers nécessaires situés dans le répertoire du rôle **roles\projects-review** sont déjà remplis. Copiez et collez le code du playbook d'origine dans les fichiers **main.** **yml** appropriés dans la nouvelle structure de rôle. Lorsque

vous copiez le code aux emplacements appropriés, assurez-vous que la mise en retrait est correcte.

Les commentaires de chacun des fichiers **main.yml** vous aident à identifier quelle partie du playbook d'origine doit être collée dans quel répertoire ou fichier dans la nouvelle structure de rôle.

Validez et transmettez par push les modifications au référentiel Git.

- Placez la première variable du playbook dans le fichier **roles\projects-review\defaults\main.yml**.
  - Placez toutes les autres variables du playbook dans le fichier **roles\projects-review\var\main.yml**.
  - Placez toutes les tâches du playbook dans le fichier **roles\projects-review\tasks\main.yml**.
  - Copiez le fichier **do417-example.zip** du répertoire **projects\files** dans le répertoire **roles\projects-review\files**.
3. Créez un playbook **projects\_review\_role.yml** qui utilise votre nouveau rôle pour créer la structure de répertoire et installe la fonction Windows. Remplacez la variable par défaut de ce rôle en la faisant pointer vers le répertoire **C:\D0417\projects-review-role\**.  
Validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
4. Démarrez une mise à jour du projet **projects repository** pour synchroniser vos modifications. Exécutez le modèle de tâche **Run projects project** pour utiliser nouveau rôle **projects-review**. Mettez à jour le modèle pour indiquer à Ansible Tower d'utiliser le playbook **projects\_review\_role.yml** afin de vous inviter à cibler l'hôte et définir les informations d'identification correctes et l'inventaire à utiliser.  
Lancez une tâche à l'aide du modèle et, à l'invite, utilisez **win2.example.com** comme hôte cible, **AD Administrator** comme informations d'identification et **Dynamic inventory** comme inventaire.
5. Vous avez modifié un playbook et l'avez converti en rôle. Vous avez utilisé ce rôle pour installer une fonction Windows, et pour créer des répertoires et des fichiers sur l'un des hôtes gérés. Si nécessaire, vous pouvez vous connecter à **win2.example.com** pour vérifier que le nouveau répertoire **C:\D0417\projects-review-role\** a été créé, ainsi que tous les sous-répertoires et fichiers définis dans le rôle. Utilisez le nom d'utilisateur **EXAMPLE\Administrator** et le mot de passe **AD Administrator** pour accéder à **win2.example.com**.
6. Cliquez sur **Log Out** pour quitter l'interface Web d'Ansible Tower.

L'atelier est maintenant terminé.

## ► Solution

# Gestion de projets volumineux

### Liste de contrôle des performances

Au cours de cet atelier, vous allez créer des rôles et des playbooks Ansible qui utilisent des rôles Ansible et des ressources DSC pour automatiser les tâches reproductibles.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Modifier un playbook pour utiliser les ressources DSC.
- Convertir le playbook en rôle.
- Créer un playbook qui utilise le rôle pour créer des répertoires et installer une fonction Windows.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

Tous les exercices guidés Chapitre 8, *Interaction avec les utilisateurs et les domaines* doivent être terminés avant de commencer cet exercice, en raison des modifications de l'inventaire et de l'authentification.

1. Modifiez le playbook **projects-review.yml** dans le référentiel **projects** et procédez aux modifications indiquées. Validez et transmettez par push les modifications au référentiel Git. Les commentaires figurant dans le playbook indiquent les modifications à apporter.
  - Utilisez DSC pour créer la structure de répertoire appropriée et pour décompresser l'archive fournie.

Utilisez DSC pour installer la fonctionnalité Windows **telnet-client**.

  - Mettez à jour la première tâche avec le module Ansible correct et les options du module nécessaires pour créer le répertoire défini.
  - Mettez à jour la troisième tâche avec le module Ansible correct et les options du module pour installer la fonction Windows **telnet-client**.
  1. Lancez l'éditeur Visual Studio Code. Accédez à **File → Open Folder**, puis sélectionnez le dossier **c:\Users\student\Documents\projects** pour ouvrir le répertoire **c:\Users\student\Documents\projects**.
  2. Modifiez le playbook **projects-review.yml**, en le modifiant comme suit :

- Utilisez DSC pour créer la structure de répertoire appropriée et pour décompresser l'archive fournie.

Utilisez DSC pour installer la fonctionnalité Windows **telnet-client**.

- Mettez à jour la première tâche avec le module Ansible correct et les options du module pour créer le répertoire défini.
- Mettez à jour la troisième tâche avec le module Ansible correct et les options du module pour installer la fonction Windows **telnet-client**.

Le playbook terminé doit s'afficher comme suit :

```
---
- name: Configure a host using DSC
hosts: all
vars:
  directory_location: C:\D0417\projects-review\
  file_name: "{{ directory_location }}new-file.txt"
  windows_resource: WindowsFeature
  feature_name: telnet-client
  zip_file_location: files/do417-example.zip
  zip_remote_location: "{{ directory_location }}do417-example.zip"
  zip_extracted_location: "{{ directory_location }}extracted-zip"

tasks:
  - name: Create a new directory
    win_dsc:
      resource_name: File
      DestinationPath: "{{ directory_location }}"
      Ensure: Present
      Type: Directory

  - name: Create a new file with text
    win_dsc:
      resource_name: File
      DestinationPath: "{{ file_name }}"
      Contents: |
        Hello
        This is the D0417 course.
      Ensure: Present
      Type: File

  - name: Install a Windows feature
    win_dsc:
      resource_name: "{{ windows_resource }}"
      Name: "{{ feature_name }}"

  - name: Copy a zip file to the managed host
    win_copy:
      src: "{{ zip_file_location }}"
      dest: "{{ zip_remote_location }}"
```

```
- name: Extract a zip file in the new directory
  win_dsc:
    resource_name: Archive
    Destination: "{{ zip_extracted_location }}"
    Path: "{{ zip_remote_location }}"
    Ensure: Present
```

- 1.3. Enregistrez vos modifications. Notez les icônes de statut situées dans le coin inférieur gauche. L'astérisque en regard de **Master** indique que de nouvelles modifications ont été apportées.
  - 1.4. Passez à la vue **Source Control**, puis cliquez sur **+** en regard du playbook **projects-review.yml** pour indexer les modifications.
  - 1.5. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 1.6. L'icône de statut située dans le coin inférieur gauche indique qu'il y a des modifications à transmettre par push au référentiel distant. Cliquez sur **Synchronize Changes** pour envoyer vos modifications au référentiel distant et pour recevoir toutes les nouvelles modifications à partir du référentiel distant.
2. Convertissez le playbook **projects-review.yml** que vous avez modifié en rôle nommé **projects-review**. Les répertoires et fichiers nécessaires situés dans le répertoire du rôle **roles\projects-review** sont déjà remplis. Copiez et collez le code du playbook d'origine dans les fichiers **main.yml** appropriés dans la nouvelle structure de rôle. Lorsque vous copiez le code aux emplacements appropriés, assurez-vous que la mise en retrait est correcte.
- Les commentaires de chacun des fichiers **main.yml** vous aident à identifier quelle partie du playbook d'origine doit être collée dans quel répertoire ou fichier dans la nouvelle structure de rôle.
- Validez et transmettez par push les modifications au référentiel Git.
- Placez la première variable du playbook dans le fichier **roles\projects-review\defaults\main.yml**.
  - Placez toutes les autres variables du playbook dans le fichier **roles\projects-review\var\main.yml**.
  - Placez toutes les tâches du playbook dans le fichier **roles\projects-review\tasks\main.yml**.
  - Copiez le fichier **do417-example.zip** du répertoire **projects\files** dans le répertoire **roles\projects-review\files**.
- 2.1. Ouvrez le fichier de playbook **projects-review.yml**.
  - 2.2. Copiez les tâches dans le fichier **projects-review.yml** dans le fichier de playbook du rôle **roles\projects-review\tasks\main.yml**.  
Dans le fichier **projects-review.yml**, sélectionnez les tâches commençant par la tâche nommée **Create a new directory**.
  - 2.3. Sélectionnez **Edit → Copy** ou appuyez sur **Ctrl+C** pour couper les tâches du fichier.
  - 2.4. Accédez à **Edit → Paste** ou appuyez sur **Ctrl+V** pour coller le texte dans le playbook **roles\projects-review\tasks\main.yml** vide.

- 2.5. Sélectionnez tout le texte et appuyez deux fois sur **MAJ+Tab** pour supprimer la mise en retrait de la tâche.
- 2.6. Vérifiez que votre fichier **roles\projects-review\tasks\main.yml** s'affiche comme suit :

```
---
```

```
- name: Create a new directory
  win_dsc:
    resource_name: File
    DestinationPath: "{{ directory_location }}"
    Ensure: Present
    Type: Directory

- name: Create a new file with text
  win_dsc:
    resource_name: File
    DestinationPath: "{{ file_name }}"
    Contents: |
      Hello
      This is the DO417 course.
    Ensure: Present
    Type: File

- name: Install a Windows feature
  win_dsc:
    resource_name: "{{ windows_resource }}"
    Name: "{{ feature_name }}"

- name: Copy a zip file to the managed host
  win_copy:
    src: "{{ zip_file_location }}"
    dest: "{{ zip_remote_location }}"

- name: Extract a zip file in the new directory
  win_dsc:
    resource_name: Archive
    Destination: "{{ zip_extracted_location }}"
    Path: "{{ zip_remote_location }}"
    Ensure: Present
```

- 2.7. Enregistrez les modifications dans le fichier **roles\projects-review\tasks\main.yml**.
- 2.8. Copiez toutes les variables, à l'exception de la variable **directory\_location**, du playbook **projects-review.yml** dans le fichier **roles\projects-review\vars\main.yml**.
- 2.9. Ajoutez la déclaration de variable au fichier **roles\projects-review\vars\main.yml**, de sorte qu'il affiche :

```
---
```

```
file_name: "{{ directory_location }}new-file.txt"
windows_resource: WindowsFeature
feature_name: telnet-client
zip_file_location: files/do417-example.zip
zip_remote_location: "{{ directory_location }}do417-example.zip"
zip_extracted_location: "{{ directory_location }}extracted-zip"
```

- 2.10. Enregistrez le fichier **roles\projects-review\vars\main.yml** mis à jour.
- 2.11. Copiez la ligne **directory\_location: C:\D0417\projects-review\** du playbook **projects-review.yml**. Ajoutez la déclaration de variable au fichier **roles\projects-review\defaults\main.yml**, de sorte qu'il affiche :

```
---
```

```
directory_location: C:\D0417\projects-review\
```

- 2.12. Enregistrez le fichier **roles\projects-review\defaults\main.yml**.
  - 2.13. Copiez le fichier **files\do417-example.zip** dans le répertoire **roles\projects-review\files**.
  - 2.14. Validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 2.15. Accédez au volet **Source Control**, puis indexez les modifications.
  - 2.16. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 2.17. Identifiez les icônes de statut situées dans le coin inférieur gauche et notez qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications.
3. Créez un playbook **projects\_review\_role.yml** qui utilise votre nouveau rôle pour créer la structure de répertoire et installe la fonction Windows. Remplacez la variable par défaut de ce rôle en la faisant pointer vers le répertoire **C:\D0417\projects-review-role\**. Validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 3.1. Créez le fichier **projects\projects\_review\_role.yml** et utilisez le nouveau rôle comme suit :

```
---
```

```
- name: Configure a host using a role
hosts: all
roles:
  - role: projects-review
    directory_location: C:\D0417\projects-review-role\
```

- 3.2. Validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
- Accédez au volet **Source Control**, puis indexez les modifications.

- 3.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 3.4. Identifiez les icônes de statut situées dans le coin inférieur gauche et notez qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur Synchronize Changes pour transmettre par push vos modifications.
4. Démarrez une mise à jour du projet **projects repository** pour synchroniser vos modifications. Exécutez le modèle de tâche **Run projects project** pour utiliser nouveau rôle **projects-review**. Mettez à jour le modèle pour indiquer à Ansible Tower d'utiliser le playbook **projects\_review\_role.yml** afin de vous inviter à cibler l'hôte et définir les informations d'identification correctes et l'inventaire à utiliser.  
Lancez une tâche à l'aide du modèle et, à l'invite, utilisez **win2.example.com** comme hôte cible, **AD Administrator** comme informations d'identification et **Dynamic inventory** comme inventaire.
  - 4.1. Double-cliquez sur l'icône de Bureau Ansible Tower pour accéder à Ansible Tower. Connectez-vous en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.
  - 4.2. Dans le volet de navigation, cliquez sur **Projects**.
  - 4.3. Dans la liste des projets, cliquez sur l'icône de synchronisation du projet **projects repository** pour démarrer le processus de synchronisation.
  - 4.4. Cliquez sur l'icône de statut et observez la mise à jour SCM automatique du projet **projects repository**.
  - 4.5. Une fois le projet correctement synchronisé, cliquez sur **Templates**.
  - 4.6. Cliquez sur le modèle de tâche **Run projects project** pour le modifier.
  - 4.7. Cliquez sur l'icône de loupe du champ **INVENTORY**. Cliquez sur **Dynamic Inventory**, puis sur **SELECT**.
  - 4.8. Cliquez sur l'icône de loupe du champ de saisie **CREDENTIAL**. Cliquez sur le bouton **AD Administrator**, puis sur **SELECT**.
  - 4.9. Cochez la case **PROMPT ON LAUNCH** de l'option **LIMIT**.
  - 4.10. Sélectionnez le playbook **projects\_review\_role.yml** dans la liste **PLAYBOOK**.
  - 4.11. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
  - 4.12. Tapez **win2.example.com** dans le champ **LIMIT**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 4.13. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**.
5. Vous avez modifié un playbook et l'avez converti en rôle. Vous avez utilisé ce rôle pour installer une fonction Windows, et pour créer des répertoires et des fichiers sur l'un des hôtes gérés. Si nécessaire, vous pouvez vous connecter à **win2.example.com** pour vérifier que le nouveau répertoire **C:\D0417\projects-review-role\** a été créé, ainsi que tous les sous-répertoires et fichiers définis dans le rôle. Utilisez le nom d'utilisateur

**EXAMPLE\Administrator** et le mot de passe **AD Administrator** pour accéder à **win2.example.com**.

6. Cliquez sur **Log Out** pour quitter l'interface Web d'Ansible Tower.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Vous pouvez importer des playbooks dans des playbooks de niveau supérieur pour gérer des projets volumineux avec de nombreux plays de manière modulaire.
- Vous pouvez inclure de manière dynamique ou importer de manière statique des tâches dans un playbook.
- Les rôles Ansible vous permettent de partager et de réutiliser plus facilement le code Ansible.
- Ansible Galaxy est une bibliothèque publique de contenu Ansible, élaborée et gérée par une communauté d'administrateurs et d'utilisateurs Ansible.
- Vous pouvez utiliser les ressources DSC PowerShell comme tâches de vos plays Ansible pour exécuter des tâches, à l'aide de l'automatisation existante dans PowerShell Gallery, gérée par Microsoft et la communauté.



## chapitre 11

# Construction de workflows Ansible Tower

### Objectif

Simplifiez la gestion des tâches et lancez des tâches complexes à l'aide de Red Hat Ansible Tower.

### Résultats

- Décrire les utilisateurs et les équipes de Red Hat Ansible Tower, les rôles auxquels ils peuvent être affectés, ainsi que la manière de les créer et de les gérer.
- Créer un questionnaire pour que les utilisateurs puissent facilement lancer des tâches avec Red Hat Ansible Tower.
- Planifier l'exécution automatique d'un modèle de tâche dans Red Hat Ansible Tower.
- Créer une séquence de modèles de tâches à exécuter à l'aide de workflows et de modèles de tâche de workflow dans Red Hat Ansible Tower.

### Sections

- Gestion des utilisateurs et des équipes (et exercice guidé)
- Crédit et utilisation de questionnaires (et exercice guidé)
- Planification de lancements de tâches (et exercice guidé)
- Exécution de workflows complexes (et exercice guidé)

### Atelier

Construction de workflows Ansible Tower

# Création et gestion d'utilisateurs Ansible Tower

---

## Résultats

À la fin de cette section, vous devez pouvoir décrire les utilisateurs et les équipes dans Red Hat Ansible Tower, les rôles auxquels ils peuvent affecter, et la façon de les créer et de les gérer.

## Contrôles d'accès basés sur les rôles

Dans Ansible Tower, les utilisateurs, les organisations, les ressources, telles que les projets, les inventaires ou les modèles de tâche, sont tous gérés en tant qu'objets au niveau Tower. Les utilisateurs se voient attribuer des *rôles* qui contrôlent des autorisations déterminant qui peut voir, modifier ou supprimer un objet dans Ansible Tower. Les rôles sont appliqués non seulement à des utilisateurs individuels, mais également à des organisations et des équipes, afin de contrôler l'accès aux ressources.

Plusieurs personnes utilisant une installation Ansible Tower nécessitent différents niveaux d'accès. Certains utilisateurs doivent disposer d'un accès pour exécuter des modèles de tâches existants par rapport à un inventaire préconfiguré de machines. Les autres doivent pouvoir modifier des inventaires, des modèles de tâche et des playbooks particuliers. Les autres utilisateurs doivent disposer d'un accès pour changer quoi que ce soit dans l'installation Ansible Tower.

La configuration de comptes d'utilisateur pour chaque personne facilite la gestion de l'accès individuel aux inventaires, aux informations d'identification, aux projets et aux modèles de tâche.

Les utilisateurs se voient attribuer des *rôles* qui contrôlent des autorisations déterminant qui peut voir, modifier ou supprimer un objet dans Ansible Tower. Les rôles sont gérés à l'aide de contrôles d'accès basés sur les rôles (RBAC). Vous pouvez gérer les rôles collectivement en les accordant à une équipe, qui est un ensemble d'utilisateurs. Tous les utilisateurs d'une équipe héritent des rôles de l'équipe. Les rôles peuvent également être accordés à une organisation qui est un regroupement logique d'équipes et de ressources. Chaque utilisateur doit être affecté à une organisation, et les utilisateurs héritent des rôles de l'organisation, en fonction de leur type d'utilisateur.



### Note

Les rôles déterminent si les utilisateurs et les équipes peuvent voir, utiliser, modifier ou supprimer des objets tels que des inventaires et des projets.

## Gestion des comptes d'utilisateur

Par défaut, le programme d'installation d'Ansible Tower crée un compte d'utilisateur **admin** avec le contrôle total de l'installation d'Ansible Tower. À l'aide du compte **admin** spécial, un administrateur Ansible Tower peut se connecter à l'interface Web et créer des utilisateurs supplémentaires.

Les trois types d'utilisateurs dans Ansible Tower sont les suivants :

### System Administrator

Le type d'utilisateur **System Administrator** (également appelé Superuser) fournit un accès illimité permettant d'effectuer n'importe quelle action dans l'ensemble de l'environnement d'Ansible Tower. **System Administrator** est un rôle singleton spécial

qui dispose d'une autorisation en lecture et en écriture sur tous les objets de toutes les organisations dans Ansible Tower.

L'utilisateur **admin** créé par le programme d'installation possède également le rôle singleton de l'administrateur système. Par conséquent, un administrateur Ansible Tower doit être connecté à Ansible Tower en tant qu'utilisateur administrateur.

#### Auditeur système

Le type d'utilisateur **System Auditor** possède également un rôle singleton spécial qui dispose d'un accès en lecture seule à la totalité de l'environnement Ansible Tower.

#### Normal User

**Normal User** est le type d'utilisateur standard. À l'origine, aucun rôle spécial n'est affecté à ce type d'utilisateur et il commence par un accès minimal. **Normal User** n'est affecté à aucun rôle singleton, et il ne reçoit que les rôles associés à l'organisation dont l'utilisateur est membre.

## Création d'utilisateurs

Les étapes suivantes montrent comment un utilisateur, qui est connecté à Ansible Tower en tant qu'utilisateur **admin**, peut créer des comptes d'utilisateur supplémentaires.

- Cliquez sur **Users** dans le volet de navigation.
- Cliquez sur **+** pour créer un utilisateur.
- Saisissez respectivement le prénom, le nom et l'adresse électronique du nouvel utilisateur dans les champs **FIRST NAME**, **LAST NAME** et **EMAIL**.
- Dans le champ **USERNAME**, spécifiez un nom d'utilisateur unique pour le nouvel utilisateur.
- Cliquez sur l'icône de loupe de la zone **ORGANIZATION** pour afficher la liste des organisations disponibles au sein d'Ansible Tower. Sélectionnez une organisation dans la liste, puis cliquez sur **SAVE**.
- Saisissez le mot de passe souhaité pour le nouvel utilisateur dans les champs **PASSWORD** et **CONFIRM PASSWORD**.
- Sélectionnez un type d'utilisateur.
- Cliquez sur **SAVE** pour terminer.

The screenshot shows the 'CREATE USER' interface in Ansible Tower. The 'DETAILS' tab is active. The user has entered the following information:

- FIRST NAME:** New
- LAST NAME:** User
- ORGANIZATION:** Default
- EMAIL:** email@address.com
- USERNAME:** newuser
- PASSWORD:** (Redacted)
- CONFIRM PASSWORD:** (Redacted)
- USER TYPE:** Normal User

At the bottom right are the 'CANCEL' and 'SAVE' buttons.

Figure 11.1: Renseignez les informations relatives au nouvel utilisateur

## Modification des utilisateurs

Utilisez la fenêtre **Edit User** pour modifier les propriétés de l'utilisateur récemment créé, comme décrit dans les étapes suivantes :

- Cliquez sur **Users** dans le volet de navigation.
- Cliquez sur l'utilisateur à modifier.
- Modifiez les champs souhaités.
- Cliquez sur **SAVE** pour terminer.

## Gestion d'Ansible Tower avec des organisations

Une *organisation* est une collection logique d'équipes, de projets et d'inventaires. Tous les utilisateurs Ansible Tower doivent appartenir à au moins une organisation.

L'un des avantages de l'utilisation d'Ansible Tower est la possibilité de partager des rôles et des playbooks Ansible au sein de limites départementales ou fonctionnelles dans une entreprise. Par exemple, le groupe Opérations d'une organisation peut déjà disposer de rôles Ansible pour la mise en service des serveurs de production, Web, de base de données et de serveurs d'applications. Les développeurs peuvent utiliser ces mêmes rôles pour approvisionner des serveurs lors de la préparation de leur environnement de développement. Ansible Tower permet à différents utilisateurs et groupes d'utiliser des rôles et des playbooks existants.

Cependant, pour les déploiements de très grande envergure, il peut être utile de catégoriser un grand nombre d'utilisateurs, d'équipes, de projets et d'inventaires au sein d'une organisation générique. Certains services ne peuvent pas se déployer sur certains inventaires d'hôtes, ni exécuter certains playbooks. En utilisant des organisations, vous pouvez configurer un ensemble d'utilisateurs et d'équipes pour fonctionner avec uniquement les ressources Ansible Tower qu'elles sont supposées utiliser.

## Création d'organisations supplémentaires

Dans le cadre de l'installation d'Ansible Tower, une organisation par défaut est créée. Les étapes suivantes décrivent la création des organisations supplémentaires à l'aide de l'interface Web d'Ansible Tower.

- Connectez-vous à l'interface web d'Ansible Tower en tant qu'utilisateur **admin**.
- Cliquez sur **Organizations** dans le volet de navigation.
- Cliquez sur **+** pour créer une organisation.
- Dans les champs, saisissez un nom pour la nouvelle organisation et, si vous le souhaitez, une description facultative.

The screenshot shows the 'New Organization' dialog box. At the top, there are three tabs: 'DETAILS' (which is selected and highlighted in dark grey), 'USERS', and 'PERMISSIONS'. Below the tabs, there are four input fields: a required field for 'NAME' (marked with a red asterisk), an optional 'DESCRIPTION' field, an 'INSTANCE GROUPS' field with a search icon, and a 'MAX HOSTS' field set to 0. At the bottom right of the dialog are two buttons: 'CANCEL' and 'SAVE'.

Figure 11.2: Informations complètes sur la nouvelle organisation

- Cliquez sur **SAVE** pour terminer.

## Rôles de l'organisation

Les utilisateurs nouvellement créés héritent des rôles spécifiques de leur organisation, en fonction de leur type d'utilisateur. Vous pouvez attribuer des rôles supplémentaires à un utilisateur après sa création pour accorder des permissions d'affichage, d'utilisation ou de modification à d'autres objets d'Ansible Tower. Une organisation est elle-même l'un de ces objets. Il existe quatre rôles pouvant être affectés aux utilisateurs dans une organisation :

### Organizational Admin

Un utilisateur avec le rôle **Admin** organisationnel peut gérer tous les aspects de cette organisation, y compris la lecture et la modification de celle-ci, ainsi que l'ajout et la suppression d'utilisateurs et d'équipes.

**Note**

Un certain nombre de rôles d'administration accordent un accès plus limité qu'**Admin** :

**Project Admin**

Un utilisateur disposant de ce rôle peut créer, lire, mettre à jour et supprimer n'importe quel projet dans l'organisation. En association avec le rôle

**Inventory Admin**, ce rôle permet aux utilisateurs de créer des modèles de tâche.

**Inventory Admin**

Un utilisateur disposant de ce rôle peut créer, lire, mettre à jour et supprimer n'importe quel inventaire dans l'organisation. En conjonction avec les rôles **Job**

**Template Admin** et **Project Admin**, cela permet à l'utilisateur d'avoir un contrôle total sur les modèles de tâches au sein de l'organisation.

**Credential Admin**

Un utilisateur disposant de ce rôle peut créer, lire, mettre à jour et supprimer des informations d'identification partagées.

**Notification Admin**

Un utilisateur disposant de ce rôle peut recevoir des notifications.

**Workflow Admin**

Un utilisateur disposant de ce rôle peut créer des workflows au sein de l'organisation.

**Job Template Admin**

Un utilisateur disposant de ce rôle peut apporter des modifications à des champs non sensibles dans des modèles de tâche. Pour apporter des modifications aux champs ayant un impact sur l'exécution d'une tâche, l'utilisateur doit également disposer du rôle **Admin** sur le modèle de tâche, du rôle **Use** sur le projet apparenté et du rôle **Use** dans l'inventaire associé.

**Auditor**

Un utilisateur disposant de ce rôle dispose d'un accès en lecture seule à l'organisation.

**Member**

Un utilisateur disposant de ce rôle dispose d'un accès en lecture seule à l'organisation. Le rôle **Member** de l'organisation fournit uniquement à l'utilisateur la possibilité d'afficher la liste des utilisateurs qui sont membres de l'organisation et les rôles d'organisation qui leur sont affectés.

Contrairement aux rôles **Admin** et **Auditor** de l'organisation, le rôle **Member** ne fournit aux utilisateurs d'autorisation sur aucune des ressources que l'organisation contient, telles que les équipes, les informations d'identification, les projets, les inventaires, les modèles de tâche, les modèles de travail ou les notifications.

**Read**

Un utilisateur disposant de ce rôle bénéficie d'un accès en lecture seule à l'objet organization uniquement. Le rôle **Read** permet à l'utilisateur d'afficher la liste des utilisateurs qui sont membres de l'organisation et les rôles d'organisation qui leur sont affectés.

Toutefois, contrairement aux rôles **Admin** et **Auditor** de l'organisation, le rôle **Read** n'hérite d'aucun rôle sur aucune des ressources que l'organisation contient, telles que les équipes, les

informations d'identification, les projets, les inventaires, les modèles de tâche, les modèles de travail et les notifications.

Vous ne pouvez affecter aucun rôle à l'objet organization sur les ressources Ansible Tower. Par conséquent, un utilisateur qui dispose du rôle **Member** ou du rôle **Read** pour une organisation n'a accès qu'à l'objet organization et n'hérite d'aucune autre autorisation en raison de ces rôles.

#### Execute

Un utilisateur disposant de ce rôle peut exécuter des modèles de tâche et des modèles de tâche de workflow dans l'organisation.



#### Important

Les utilisateurs disposant du rôle singleton **System Administrator** héritent du rôle **Admin** sur chaque organisation au sein d'Ansible Tower.

Les utilisateurs disposant du rôle singleton **System Auditor** héritent du rôle **Auditor** sur chaque organisation au sein d'Ansible Tower.

Un utilisateur créé avec le type d'utilisateur **Normal User** est affecté à un rôle **Member** sur l'organisation à laquelle il a été affecté au moment de la création de l'utilisateur dans Ansible Tower. Vous pouvez ajouter d'autres rôles ultérieurement, y compris des rôles **Member** supplémentaires sur d'autres organisations.

## Gestion des rôles d'organisation d'utilisateurs

L'écran **Edit User** n'autorise que les modifications apportées au type d'utilisateur et à l'organisation à laquelle un utilisateur appartient. Étant donné que les types d'utilisateurs désignés pour les utilisateurs déterminent leurs rôles d'organisation hérités, la modification de l'organisation et du type d'utilisateur d'un utilisateur peut avoir une incidence sur son rôle d'organisation.

L'administration complète du rôle d'un utilisateur dans une organisation nécessite un accès en tant qu'administrateur. Les étapes suivantes présentent une vue d'ensemble de l'administration des rôles utilisateur dans Ansible Tower.

- Connectez-vous à l'interface Web d'Ansible Tower en tant qu'**admin** ou n'importe quel utilisateur disposant du rôle **Admin** sur l'organisation en cours de modification.
- Cliquez sur **Organizations** sur le volet de navigation.
- Cliquez sur **Permissions** pour l'organisation en cours de gestion. Une liste des utilisateurs auxquels des rôles ont été attribués s'affiche.
- Dans le volet **ADD USERS/TEAMS**, cochez la case correspondant à l'utilisateur souhaité.
- Cliquez sur **SELECT ROLES** et sélectionnez le rôle d'organisation souhaité pour l'utilisateur. Cette étape peut être répétée plusieurs fois pour ajouter plusieurs rôles à un utilisateur.



#### Note

Pour obtenir la liste de la définition de chaque rôle, cliquez sur le bouton **KEY**.

The screenshot shows the 'PERMISSIONS' tab for the 'Default' organization. At the top, there are tabs for 'DETAILS', 'USERS', 'PERMISSIONS' (which is selected), and 'NOTIFICATIONS'. Below the tabs is a search bar and a green '+' button. The main area displays a table with columns 'USER', 'ROLE', and 'TEAM ROLES'. The data is as follows:

USER	ROLE	TEAM ROLES
admin	SYSTEM ADMINISTRATOR	
daniel	PROJECT ADMIN	
david	MEMBER	
donnie	MEMBER	

**Figure 11.3: Définir les rôles à affecter à l'organisation**

- Cliquez sur **SAVE** pour affecter des rôles à l'utilisateur pour l'organisation.
- Cliquez sur le **X** précédent le rôle pour supprimer les rôles existants d'un utilisateur.

## Gestion des groupes d'utilisateurs avec des équipes

Les équipes sont des groupes d'utilisateurs qui facilitent la gestion des rôles sur les objets Ansible Tower tels que les inventaires, les projets et les modèles de tâches, ce qui est plus efficace que de les gérer séparément pour chaque utilisateur. Plutôt que d'affecter les mêmes rôles à plusieurs utilisateurs, vous pouvez affecter des rôles à l'équipe représentant un groupe d'utilisateurs. Les utilisateurs, qui sont membres d'une équipe, héritent des rôles affectés à celle-ci.

Dans Ansible Tower, les utilisateurs existent en tant qu'objets au niveau de Tower. Un utilisateur peut avoir des rôles dans plusieurs organisations. Une équipe appartient à une seule organisation, mais un **System Administrator** Ansible Tower peut affecter les rôles de l'équipe à des ressources appartenant à d'autres organisations.



### Important

Bien qu'une équipe appartienne à une organisation particulière, vous ne pouvez pas affecter de rôles organisationnels à une équipe, comme vous pouvez le faire avec des utilisateurs individuels. Pour gérer un objet organization, vous devez affecter directement les rôles à des utilisateurs spécifiques.

Toutefois, vous pouvez affecter des rôles aux équipes qui donnent accès aux ressources appartenant à une organisation particulière, telles que des projets, des inventaires ou des modèles de tâche.

## Création d'équipes

Les étapes suivantes décrivent comment créer des équipes dans une organisation.

- Connectez-vous à l'interface Web d'Ansible Tower en tant qu'utilisateur **admin** ou en tant qu'utilisateur ayant reçu le rôle **admin** pour l'organisation à laquelle la nouvelle équipe appartient.
- Cliquez sur **Teams** dans le volet de navigation gauche.
- Cliquez sur **+**.
- Dans le volet **New Team**, saisissez un nom pour la nouvelle équipe dans le champ **NAME**.
- Saisissez une description dans le champ **DESCRIPTION**, si vous le souhaitez.
- Pour compléter le champ **ORGANIZATION**, cliquez sur l'icône de loupe pour afficher la liste. Dans le volet **SELECT ORGANIZATION**, sélectionnez l'organisation à laquelle appartient l'équipe.
- Cliquez sur **SAVE** pour créer l'équipe.

## Ajout d'utilisateurs à une équipe

Une fois la création de l'équipe terminée, vous pouvez ajouter des utilisateurs à celle-ci. Les étapes suivantes résument la manière d'ajouter des utilisateurs, affectés au rôle de **Member**, à une équipe d'une organisation :

- Connectez-vous à l'interface Web d'Ansible Tower en tant qu'utilisateur **admin** ou en tant qu'utilisateur ayant reçu le rôle **admin** pour l'organisation à laquelle l'équipe appartient.
- Cliquez sur **Organizations** sur le volet de navigation gauche.
- Cliquez sur le lien **TEAMS** sous l'organisation à laquelle l'équipe appartient.
- Dans le volet **Teams**, cliquez sur le nom de l'équipe à laquelle vous souhaitez ajouter l'utilisateur.
- Dans le volet de détails de l'équipe, cliquez sur **USERS**.
- Cliquez sur **+**.

USER	FIRST NAME	LAST NAME	ROLE
admin			SYSTEM ADMINISTRATOR
daniel	Daniel	George	ADMIN
david	David	Jackobs	MEMBER
donnie	Donnie	Jameson	READ

Figure 11.4: Ajouter un utilisateur à une équipe avec le rôle Member

- Dans le volet **ADD USERS**, sélectionnez un ou plusieurs utilisateurs à ajouter à l'équipe. Ensuite, cliquez sur **SAVE** pour ajouter les utilisateurs sélectionnés à l'équipe.

## Intégration d'Ansible Tower à Active Directory

Vous pouvez intégrer Ansible Tower à votre serveur Active Directory local. Ansible Tower utilise le serveur LDAP pour authentifier les utilisateurs. Ansible Tower crée automatiquement un compte pour chaque utilisateur qui se connecte à l'aide d'un nom d'utilisateur et d'un mot de passe LDAP. Les utilisateurs créés de cette manière ne peuvent pas changer leur nom d'utilisateur, prénom, nom de famille ni définir un mot de passe local pour eux-mêmes.

Les étapes suivantes décrivent comment intégrer Ansible Tower à Active Directory :

- Dans Active Directory, créez un utilisateur disposant d'autorisations en lecture pour l'ensemble de la structure LDAP.
- Connectez-vous à l'interface Web d'Ansible Tower en tant qu'utilisateur **admin** ou en tant qu'utilisateur ayant reçu le rôle **admin**.
- Cliquez sur **Settings** dans le volet de navigation gauche.
- Cliquez sur **Authentication** dans la page **SETTINGS**.
- Cliquez sur **LDAP**.
- Dans le champ **LDAP SERVER URI**, saisissez l'adresse du serveur LDAP auquel Ansible Tower se connectera. Par exemple, `ldap://demo.example.com:389`.
- Dans le champ **LDAP BIND DN**, saisissez le nom unique de la liaison de l'utilisateur qu'Ansible Tower doit utiliser pour se connecter au serveur LDAP. Par exemple, `CN=demo_user,CN=Users,DC=example,DC=com`.
- Dans le champ **LDAP BIND PASSWORD**, saisissez le mot de passe de l'utilisateur de la liaison.
- Dans la liste **LDAP GROUP TYPE**, choisissez le type **ActiveDirectoryGroupType**.
- Dans le champ **LDAP USER SEARCH**, saisissez le modèle de la requête de recherche d'utilisateur LDAP. Pour Active Directory, utilisez le modèle `sAMAccountName=%(user)s`. Par exemple :

```
[  
  "CN=Users, DC=example, DC=com",  
  "SCOPE_SUBTREE",  
  "(sAMAccountName=%(user)s")  
]
```

- Dans le champ **LDAP GROUP SEARCH**, saisissez le modèle de requête de recherche de groupes LDAP. Par exemple :

```
[  
  "dc=example, dc=com",  
  "SCOPE_SUBTREE",  
  "(objectClass=group)"  
]
```

- Dans le champ **LDAP USER ATTRIBUTE MAP**, saisissez le modèle de mise en correspondance des utilisateurs LDAP avec les attributs utilisateur de l'API Ansible Tower. Par exemple :

```
{  
  "first_name": "givenName",  
  "last_name": "sn",  
  "email": "mail"  
}
```

- Cette étape est facultative, mais en fonction de la configuration de l'environnement, elle peut présenter une utilité. Avec ce paramètre, vous pouvez mapper des utilisateurs aux organisations d'Ansible Tower, par rapport à leur appartenance à un groupe de LDAP. Dans le champ **LDAP ORGANIZATION MAP**, saisissez le modèle de mise en correspondance des groupes LDAP avec les organisations Ansible Tower. Dans l'exemple ci-dessous, les membres du groupe **Domain Admins** sont mappés sur **Demo Organization** avec le rôle **admin**, tandis que les membres du groupe **Users** sont mappés sur l'organisation avec le rôle de **Member** :

```
{  
  "Demo Organization": {  
    "admins": "CN=Domain Admins,CN=Users,DC=example,DC=com",  
    "users": [  
      "CN=Users,DC=example,DC=com"  
    ],  
    "remove_users": true,  
    "remove_admins": true  
  }  
}
```

- Cette étape est facultative, mais en fonction de la configuration de l'environnement, elle peut présenter une utilité. Avec ce paramètre, vous pouvez mapper tous les utilisateurs Active Directory en tant que membres d'une équipe particulière. Dans le champ **LDAP TEAM MAP**, saisissez le modèle de mise en correspondance des utilisateurs LDAP avec les équipes Ansible Tower. Dans l'exemple ci-dessous, tous les membres de l'annuaire Active Directory qui se connectent correctement à Ansible Tower sont automatiquement ajoutés à l'équipe **My Demo Team** avec un rôle **Member** :

```
{  
  "My Demo Team": {  
    "organization": "Demo Organization",  
    "users": true  
  }  
}
```

- Cliquez sur **SAVE** pour enregistrer les nouveaux paramètres.

Vous devez maintenant pouvoir vous connecter à Ansible Tower à l'aide d'un nom d'utilisateur et d'un mot de passe stockés dans Active Directory.



## Références

### Guide de l'utilisateur d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/userguide>

## ► Exercice guidé

# Gestion de l'accès des utilisateurs et des équipes

Au cours de cet exercice, vous allez créer et gérer des utilisateurs et des équipes dans Red Hat Ansible Tower.

## Résultats

Vous devez pouvoir créer des comptes d'utilisateur et des équipes, et décrire l'accès fourni par différents types de comptes.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide **d'example.com** en tant que domaine et **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.



### Important

Cet exercice utilise des ressources créées dans le *Chapitre 8 : Interaction avec les utilisateurs et les domaines* ; vous devez effectuer ces exercices avant de commencer celui-ci.

- ▶ 1. Accédez à Ansible Tower en double-cliquant sur le raccourci Ansible Tower sur votre Bureau, ou en accédant à <https://tower.example.com>. Connectez-vous en utilisant le compte **admin** et **RedHat123@!** comme mot de passe.
- ▶ 2. Modifiez le modèle de tâche **Run tower project** pour exécuter une nouvelle tâche qui utilise le playbook **domain-users.yml**. Le playbook remplit Active Directory avec des comptes d'utilisateur supplémentaires. Vous allez utiliser certains de ces comptes pour créer des utilisateurs dans Ansible Tower.
  - 2.1. Dans le volet de navigation gauche, cliquez sur **Templates**.
  - 2.2. Dans la liste des modèles, sélectionnez le modèle **Run tower project**.
  - 2.3. Sélectionnez le playbook **domain-users.yml** dans la liste **PLAYBOOK**.
  - 2.4. Cochez la case **PROMPT ON LAUNCH** de l'option **LIMIT**.
  - 2.5. Activez la case à cocher **PROMPT ON LAUNCH** pour l'option **INVENTORY**.
  - 2.6. Cochez la case **PROMPT ON LAUNCH** pour l'option **CREDENTIAL**.

- 2.7. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
- 2.8. Dans la fenêtre **RUN TOWER PROJECT**, choisissez **Dynamic Inventory**, puis cliquez sur **NEXT**.
- 2.9. Sélectionnez les informations d'identification **AD Administrator**, puis cliquez sur **NEXT**.
- 2.10. Dans le champ **LIMIT**, saisissez le nom d'hôte **windc.example.com**. Cliquez sur **NEXT**, puis sur **LAUNCH**.
- 2.11. Observez la sortie en direct de la tâche en cours d'exécution.
- 2.12. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.

► 3. Créez l'utilisateur **sam** avec le privilège **Normal User**.

- 3.1. Cliquez sur **Users** dans le volet de navigation.
- 3.2. Cliquez sur **+** pour ajouter un nouvel utilisateur.
- 3.3. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
FIRST NAME	Sam
LAST NAME	Simons
ORGANIZATION	Valeur par défaut
EMAIL	sam@example.com
USERNAME	sam
PASSWORD	RedHat123@!
CONFIRM PASSWORD	RedHat123@!
USER TYPE	Normal User

- 3.4. Cliquez sur **SAVE** pour créer l'utilisateur.
- 4. Vérifiez les permissions de l'utilisateur **sam** que vous venez de créer.
  - 4.1. Cliquez sur **PERMISSIONS** pour accéder aux autorisations de l'utilisateur.
  - 4.2. Comme illustré, **sam** est membre de l'organisation **Default**.
  - 4.3. Cliquez sur **Log Out** dans le coin supérieur droit pour vous déconnecter, puis reconnectez-vous en tant qu'utilisateur **sam** avec le mot de passe **RedHat123!**.
  - 4.4. Dans le volet de navigation, vous pouvez voir que l'accès de l'utilisateur est limité, car certaines options ne sont pas disponibles.
  - 4.5. Cliquez sur **Users** dans le volet de navigation pour gérer les autorisations de l'utilisateur. L'utilisateur **sam** ne doit pas pouvoir créer des utilisateurs.

- 4.6. Cliquez sur l'icône **Log Out** située dans le coin supérieur droit pour vous déconnecter.

► **5.** Créez l'utilisatrice **sylvia** en tant qu'auditrice de système.

- 5.1. Reconnectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
- 5.2. Cliquez sur **Users** dans le volet de navigation pour gérer les utilisateurs.
- 5.3. Cliquez sur le bouton **+** pour ajouter un nouvel utilisateur.
- 5.4. Dans la fenêtre suivante, renseignez les informations comme suit :

<b>Champ</b>	<b>Valeur</b>
FIRST NAME	Sylvia
LAST NAME	Simons
ORGANIZATION	Valeur par défaut
EMAIL	sylvia@example.com
USERNAME	sylvia
PASSWORD	RedHat123@!
CONFIRM PASSWORD	RedHat123@!
USER TYPE	System Auditor

- 5.5. Cliquez sur **SAVE** pour créer le nouvel utilisateur.

► **6.** Vérifiez les autorisations pour **sylvia**.

- 6.1. Cliquez sur l'onglet **PERMISSIONS** pour afficher les autorisations de l'utilisateur.
- 6.2. Comme vous pouvez le voir, **sylvia** a le rôle System Auditor.
- 6.3. Déconnectez-vous, puis reconnectez-vous en tant que **sylvia** avec le mot de passe **RedHat123@!**.
- 6.4. Étant donné que **sylvia** est un auditrice de système, elle a accès à tous les éléments de l'interface d'Ansible Tower.
- 6.5. Cliquez sur **Users** dans le volet de navigation pour gérer les utilisateurs. L'utilisatrice **sylvia** ne doit pas pouvoir ajouter de nouveaux utilisateurs.
- 6.6. Cliquez sur l'icône **Log Out** pour vous déconnecter de l'interface Web d'Ansible Tower.

► **7.** Créez l'utilisateur **simon** avec le privilège **System Administrator**.

- 7.1. Connectez-vous à l'interface Web Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

- 7.2. Cliquez sur **Users** dans le volet de navigation pour gérer les utilisateurs.
- 7.3. Cliquez sur **+** pour ajouter un nouvel utilisateur.
- 7.4. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
FIRST NAME	Simon
LAST NAME	Stephens
ORGANIZATION	Valeur par défaut
EMAIL	simon@example.com
USERNAME	simon
PASSWORD	RedHat123@!
CONFIRM PASSWORD	RedHat123@!
USER TYPE	System Administrator

- 7.5. Cliquez sur **SAVE** pour créer le nouvel utilisateur.
- 8. Vérifiez les autorisations pour **simon**.
- 8.1. Cliquez sur **PERMISSIONS** pour afficher les autorisations de l'utilisateur. Cette fois, le cadre d'autorisations indique explicitement « **SYSTEM ADMINISTRATORS HAVE ACCESS TO ALL PERMISSIONS** ».
  - 8.2. Déconnectez-vous, puis reconnectez-vous en tant qu'utilisateur **simon** avec le mot de passe **RedHat123@!**.
  - 8.3. Vous pouvez voir que l'utilisateur a accès à tous les éléments dans le volet de navigation.
  - 8.4. Cliquez sur **Users** dans le volet de navigation pour gérer les utilisateurs. Comme vous pouvez le voir, **simon** dispose des droits permettant de créer de nouveaux utilisateurs.
  - 8.5. Cliquez sur l'icône **Log Out** pour vous déconnecter de l'interface Web d'Ansible Tower.
- 9. Les étapes suivantes décrivent comment intégrer Ansible Tower à Active Directory.



#### Important

Dans votre environnement de formation, certaines options sont déjà prédéfinies. Normalement, tous les champs sont vides et vous devez les remplir avec les paramètres appropriés selon votre configuration d'environnement.

Connectez-vous à l'interface Web d'Ansible Tower en tant qu'utilisateur **admin** ou en tant qu'utilisateur ayant reçu le rôle **admin**.

- 9.1. Cliquez sur **Settings** dans le volet de navigation.
- 9.2. Cliquez sur **Authentication** dans la page **SETTINGS**.
- 9.3. Cliquez sur **LDAP**.
- 9.4. Dans le champ **LDAP SERVER URI**, saisissez `ldap://windc.example.com:389`.
- 9.5. Dans le champ **LDAP BIND DN**, saisissez  
`CN=admin, CN=Users, DC=example, DC=com`.
- 9.6. Dans le champ **LDAP BIND PASSWORD**, saisissez le mot de passe `RedHat123@!` de l'utilisateur de la liaison.
- 9.7. Dans la liste **LDAP GROUP TYPE**, sélectionnez le type **ActiveDirectoryGroupType**.
- 9.8. Dans le champ **LDAP USER SEARCH**, saisissez le modèle de la requête de recherche d'utilisateur LDAP.

```
[  
  "CN=Users,DC=example,DC=com",  
  "SCOPE_SUBTREE",  
  "(sAMAccountName=%(user)s")  
]
```

- 9.9. Dans le champ **LDAP GROUP SEARCH**, saisissez le modèle de requête de recherche de groupes LDAP.

```
[  
  "CN=Builtin,dc=example,dc=com",  
  "SCOPE_SUBTREE",  
  "(objectClass=group)"  
]
```

- 9.10. Dans le champ **LDAP USER ATTRIBUTE MAP**, saisissez le modèle de mise en correspondance des utilisateurs LDAP avec les attributs utilisateur de l'API Ansible Tower.

```
{  
  "first_name": "givenName",  
  "last_name": "sn",  
  "email": "mail"  
}
```

- 9.11. Effacez le champ **LDAP GROUP TYPE PARAMETERS**, afin qu'il ne contienne que `{}`.

```
{}
```

- 9.12. Dans le champ **LDAP ORGANIZATION MAP**, saisissez le modèle de mise en correspondance des groupes LDAP avec les organisations Ansible Tower.

```
{
  "Default": {
    "admins": "CN=Administrators,CN=BuiltIn,DC=example,DC=com",
    "users": false,
    "remove_admins": true
  }
}
```

- 9.13. Dans le champ **LDAP TEAM MAP**, saisissez le modèle de mise en correspondance des utilisateurs LDAP avec les équipes Ansible Tower.

```
{
  "Developers": {
    "organization": "Default",
    "users": true
  }
}
```

- 9.14. Cliquez sur **SAVE** pour enregistrer les nouveaux paramètres.

Vous pouvez maintenant vous connecter à Ansible Tower à l'aide d'un nom d'utilisateur et d'un mot de passe stockés dans Active Directory.

- 9.15. Cliquez sur l'icône **Log Out** pour vous déconnecter de l'interface Web d'Ansible Tower.

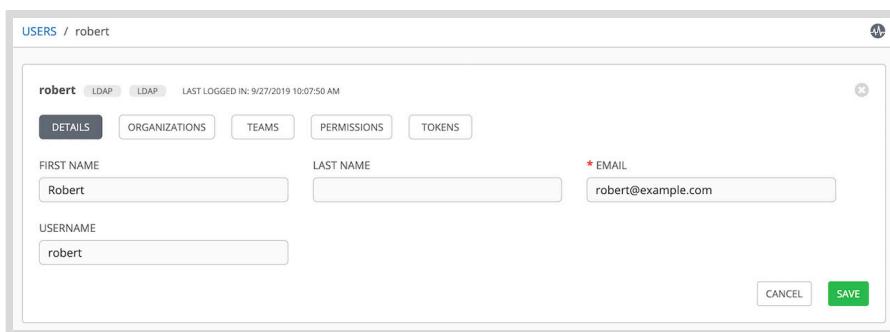
- 10. Connectez-vous à Ansible Tower en tant qu'utilisateur Active Directory **robert** et avec le mot de passe **RedHat123@!**. Avec l'intégration LDAP que vous avez configurée, Ansible Tower attribue automatiquement à cet utilisateur le rôle **admin**, en fonction de son appartenance au groupe Active Directory.

- 10.1. Connectez-vous à l'interface Web Ansible Tower en utilisant **robert** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Vous pouvez voir que l'utilisateur a accès à tous les éléments dans le volet de navigation.

- 10.2. Cliquez sur l'utilisateur **robert** pour accéder aux détails du compte pour cet utilisateur.

Notez qu'Ansible crée automatiquement l'utilisateur lors de sa première connexion, en fonction de l'intégration Active Directory et de la mise en correspondance de l'appartenance au groupe LDAP approprié.



- 10.3. Cliquez sur **PERMISSIONS** pour afficher les autorisations de l'utilisateur.

The screenshot shows the 'PERMISSIONS' tab selected for the user 'robert'. The interface includes tabs for DETAILS, ORGANIZATIONS, TEAMS, PERMISSIONS (selected), and TOKENS. A search bar and a 'KEY' button are also present. The main table lists organizations and teams with their respective types and roles. The 'Default' organization is listed as an Organization with an Admin role. The 'Developers' team is listed as a Team with a Member role.

NAME	TYPE	ROLE	ACTIONS
Default	Organization	Admin	x
Developers	Team	Member	x

Notez que l'utilisateur a le rôle **admin** pour l'organisation **Default**, car cet utilisateur est membre du groupe **Administrators** dans Active Directory. Lorsque l'utilisateur **robert** s'est connecté pour la première fois, Ansible Tower a créé une équipe appelée **Developers**. Cet utilisateur a été ajouté à l'équipe et le rôle de **Member** lui a été affecté. La création d'équipes et l'affectation d'utilisateurs sont basées sur l'appartenance de l'utilisateur au groupe Active Directory **Domain Users**, dont les membres sont automatiquement mappés en tant que membres de l'équipe **Developers**.

- 10.4. Cliquez sur **Users** dans le volet de navigation pour gérer les utilisateurs. Comme vous pouvez le voir, **robert** dispose des droits permettant de créer des utilisateurs.

► 11. Créez une équipe **Devops**.

- 11.1. Cliquez sur **Teams** dans le volet de navigation pour gérer les utilisateurs.
- 11.2. Cliquez sur **+** pour ajouter une nouvelle équipe.
- 11.3. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
NAME	Devops
DESCRIPTION	Devops Team
ORGANIZATION	Valeur par défaut

- 11.4. Cliquez sur **SAVE** pour créer l'équipe.

► 12. Ajoutez un utilisateur existant en tant que membre à l'équipe **Devops**.

- 12.1. Dans le volet de détails de l'équipe **Devops**, cliquez sur **USERS**.
- 12.2. Cliquez sur **+** pour ajouter de nouveaux utilisateurs à l'équipe.
- 12.3. Sélectionnez **sylvia**.
- 12.4. Cliquez sur **SAVE**.  
Notez que **sylvia** a le nouveau rôle **Member**. Cela signifie qu'elle est maintenant **System Auditor**, et qu'elle est également membre de l'équipe **Devops**.
- 12.5. Cliquez sur l'icône **Log Out** pour vous déconnecter.

- 13. Connectez-vous à Ansible Tower en tant qu'utilisateur Active Directory **david** et avec le mot de passe **RedHat123@!**. Ansible Tower attribue automatiquement à cet utilisateur le rôle **normal user**, en fonction de son appartenance au groupe Active Directory.

- 13.1. Connectez-vous à l'interface web Ansible Tower en utilisant **david** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

Vous pouvez voir que cet utilisateur a un accès limité à tous les éléments Ansible Tower dans la barre de navigation.

- 13.2. Cliquez sur **david** pour accéder aux détails du compte d'utilisateur.

Notez qu'Ansible Tower crée automatiquement l'utilisateur lors de sa première connexion, en fonction de l'intégration Active Directory et de la mise en correspondance de l'appartenance au groupe LDAP approprié.

- 13.3. Cliquez sur **PERMISSIONS** pour afficher les autorisations de l'utilisateur.

Notez que l'utilisateur possède uniquement le rôle **Member** dans l'équipe **Developers**, en fonction de l'appartenance au groupe Active Directory **Domain Users**.

- 13.4. Cliquez sur **Users** dans le volet de navigation pour gérer les utilisateurs. L'utilisateur **david** ne doit pas pouvoir créer des utilisateurs.

- 14. Cliquez sur l'icône **Log Out** située dans le coin supérieur droit pour vous déconnecter

L'exercice guidé est maintenant terminé.

# Création et utilisation de questionnaires

## Résultats

À la fin de cette section, vous devez pouvoir créer un questionnaire pour aider les utilisateurs à lancer simplement des tâches avec Red Hat Ansible Tower.

## Gestion des variables

Il est recommandé d'écrire des playbooks que vous pouvez réutiliser dans différentes situations ou pour divers systèmes. Ces systèmes peuvent nécessiter des comportements différents, des configurations ou un travail dans des environnements divers. Une façon simple de traiter cette variation consiste à utiliser des variables que vous pouvez remplacer lors de l'exécution de vos plays.

Vous pouvez définir des valeurs pour les variables de plusieurs façons dans Ansible, et vous pouvez remplacer des valeurs si elles sont correctement configurées. Par exemple, un rôle fournit une valeur par défaut pour une variable qui peut à son tour être remplacée par des valeurs définies pour cette variable par un inventaire ou un playbook. En général, il est préférable de définir une valeur d'une variable à un seul emplacement pour éviter les problèmes de priorité des variables.

Dans Ansible Tower, les modèles de tâche peuvent définir des variables supplémentaires et inviter des utilisateurs à saisir une valeur lors du lancement d'un modèle de tâche, ou elles peuvent être définies automatiquement en réexécutant à nouveau une tâche qui a été lancée avec des variables supplémentaires. Ansible Tower ne prend pas en charge les playbooks avec des questions **vars\_prompt**; le remplacement le plus proche est *surveys* qui est abordé en détail plus loin dans cette section.



### Important

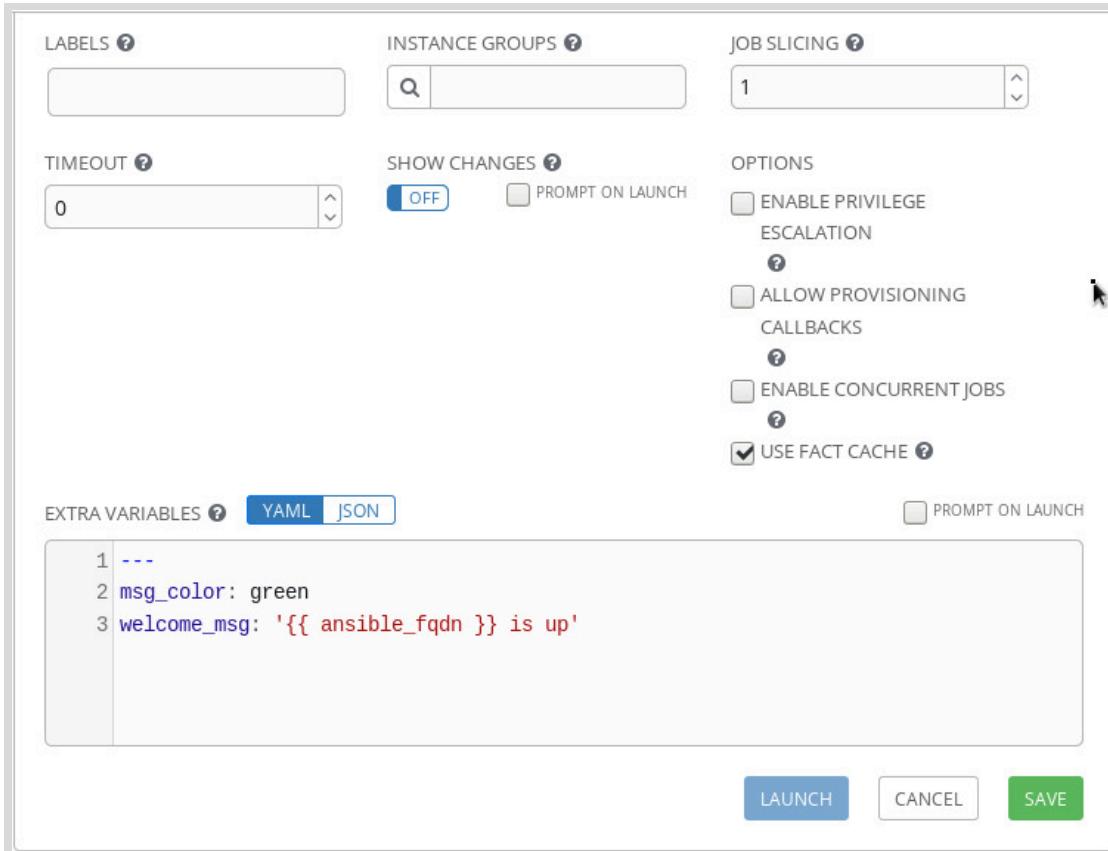
Ansible Tower ne prend pas en charge les playbooks qui utilisent **vars\_prompt** pour définir des variables de manière interactive.

## Définition de variables supplémentaires

Dans Ansible Tower, vous pouvez utiliser des modèles de tâche pour définir directement des variables supplémentaires de l'une des deux façons :

- Définissez des variables supplémentaires en les saisissant au format YAML ou JSON dans le champ **EXTRA VARIABLES** du modèle de tâche.
- Sélectionnez l'option **PROMPT ON LAUNCH** pour le champ **EXTRA VARIABLES**; Ansible Tower invite les utilisateurs à modifier de manière interactive la liste des variables supplémentaires utilisées avant d'exécuter une tâche.

L'exemple suivant illustre la définition de variables supplémentaires dans le modèle de tâche :



**Figure 11.7: Ajout de variables supplémentaires à un modèle de tâche**

Si vous sélectionnez l'option **PROMPT ON LAUNCH** pour le champ **EXTRA VARIABLES**, une fenêtre s'affiche pour modifier les variables lorsque vous lancez une tâche à partir d'un modèle de tâche.

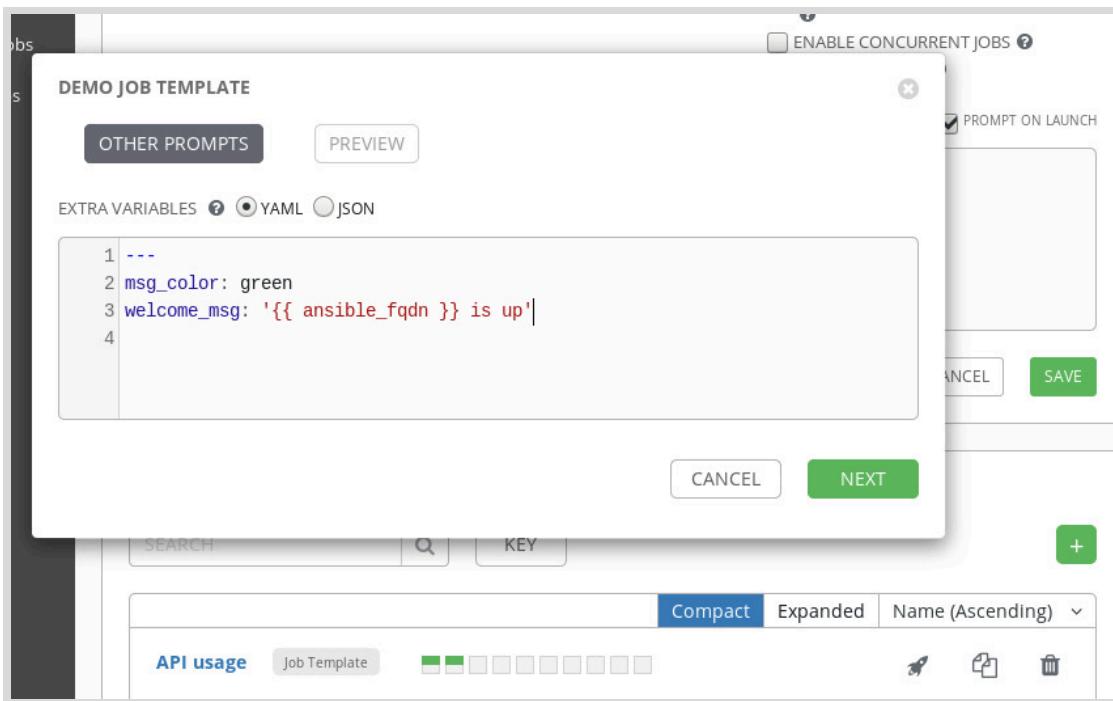


Figure 11.8: Ajustement des variables supplémentaires au lancement d'une tâche

Si vous relancez la tâche qui en résulte, la tâche réutilise les mêmes variables supplémentaires. Vous ne pouvez pas modifier les variables supplémentaires d'une tâche lorsqu'elle est réexécutée. Pour modifier les variables supplémentaires, lancez la tâche à partir des modèles de tâche d'origine et utilisez des valeurs différentes pour les variables.

## Définition de variables supplémentaires avec des questionnaires

Il peut s'avérer difficile d'utiliser des variables supplémentaires, car l'utilisateur qui lance la tâche doit comprendre les variables disponibles et la manière dont les playbooks associés au modèle de tâche les utilisent. Les **questionnaires** configurent le modèle de tâche de sorte qu'il affiche une forme abrégée avant l'exécution, qui vous invite à saisir des informations déterminant la valeur des variables supplémentaires.

Inviter l'utilisateur à saisir des informations offre plusieurs avantages par rapport à d'autres moyens de définir des variables supplémentaires. Aucune connaissance approfondie du fonctionnement des variables supplémentaires, ou du nom des variables supplémentaires utilisées par les playbooks, n'est nécessaire. L'utilisateur peut même ignorer que des variables supplémentaires sont utilisées dans le modèle de tâche.

Étant donné que les invites peuvent contenir du texte arbitraire, elles peuvent être formulées dans un langage convivial et facile à comprendre, même si les utilisateurs manquent de connaissances approfondies sur Ansible.



### Important

Les questionnaires définissent des variables supplémentaires. En fait, les valeurs définies à l'aide d'un questionnaire remplacent les valeurs définies sur des variables du même nom de toute autre manière. Cela inclut le champ **EXTRA VARIABLES** et l'option **PROMPT ON LAUNCH** définie sur le modèle de tâche.

Parmi les éléments que vous pouvez configurer dans un questionnaire, citons :

#### Questions conviviales

Les questionnaires peuvent inviter les utilisateurs à effectuer une saisie à l'aide de questions personnalisées. Cela permet d'avoir des invites plus conviviales de saisie de valeurs de variables supplémentaires par l'utilisateur que la méthode **PROMPT ON LAUNCH**.

#### Type de réponse

Outre l'utilisation d'une invite conviviale, les questionnaires peuvent également définir des règles pour les saisies de l'utilisateur et les valider. Les réponses des utilisateurs aux questions peuvent être limitées à l'un des sept types de réponses suivants :

Type	Description
Text	Texte à une seule ligne
Textarea	Texte multiligne
Password	Les données sont traitées comme des informations sensibles.
Multiple Choice (single select)	Une liste d'options parmi lesquelles les utilisateurs peuvent choisir une seule option comme réponse.
Multiple Choice (multiple select)	Une liste d'options parmi lesquelles les utilisateurs peuvent choisir une ou plusieurs options comme réponse.
Integer	Nombre entier
Float	Nombre décimal à virgule flottante

#### Longueur de la réponse

Vous pouvez également définir des règles concernant la taille des réponses des utilisateurs aux questions du questionnaire. Pour les types de réponses suivants, un questionnaire peut définir la longueur de caractères minimale et maximale autorisée pour les réponses des utilisateurs : **Text**, **Textarea**, **Password**, **Integer** et **Float**.

#### Réponse par défaut

Vous pouvez fournir une réponse par défaut à une question. La question peut également indiquer **REQUIRED**, ce qui signifie que les utilisateurs doivent fournir une réponse à la question.

## Création d'un questionnaire

Vous ne pouvez pas ajouter de questionnaire lors de la création d'un modèle de tâche. Ajoutez un questionnaire à un modèle de tâche après avoir créé ce dernier.

Figure 11.9 montre comment ajouter un questionnaire à un modèle de tâche existant.

1. Cliquez sur **Templates** dans la barre de navigation pour afficher la liste des modèles de tâche existants.
2. Cliquez sur le modèle de tâche souhaité pour le modifier.
3. Cliquez sur **ADD SURVEY** pour accéder à l'interface de création du questionnaire.

4. Ajoutez une question au questionnaire.
  - 4.1. Dans le champ **PROMPT**, saisissez la question à afficher à l'utilisateur.
  - 4.2. Dans le champ **ANSWER VARIABLE NAME**, saisissez le nom de la variable supplémentaire à laquelle la réponse de l'utilisateur est affectée.
  - 4.3. Dans la liste **ANSWER TYPE**, sélectionnez le type de réponse souhaité pour la réponse de l'utilisateur.
  - 4.4. Si vous utilisez un type de réponse à choix multiple, définissez la liste en saisissant un élément de liste par ligne dans le champ **MULTIPLE CHOICE OPTIONS**.
  - 4.5. Si vous utilisez un type de réponse qui ne soit pas à choix multiple, spécifiez éventuellement la longueur de caractère minimale et maximale de la réponse de l'utilisateur dans les champs **MINIMUM LENGTH** et **MAXIMUM LENGTH**, respectivement.
  - 4.6. Si vous le souhaitez, définissez éventuellement une valeur par défaut pour la variable supplémentaire interrogée dans le champ **DEFAULT ANSWER**. Ansible Tower utilise cette valeur si l'utilisateur ne fournit aucune réponse.
  - 4.7. Sélectionnez **REQUIRED** pour indiquer qu'une réponse est obligatoire. Désactivez ce champ si une réponse n'est pas obligatoire.
  - 4.8. Cliquez sur **+ADD** pour ajouter la question au questionnaire. Un aperçu de la nouvelle question s'affiche dans le panneau **PREVIEW**.
5. Répétez les étapes précédentes pour ajouter d'autres questions au questionnaire. Après avoir ajouté toutes vos questions, allez en haut de la fenêtre. Un commutateur dans l'angle supérieur gauche, en regard de **Survey**, détermine si l'enquête est activée ou désactivée. Par défaut, les questionnaires sont activés lors de la création, de sorte que le commutateur est en position **ON**. Pour désactiver le questionnaire, faites passer le commutateur en position **OFF**.
6. Cliquez sur **SAVE** pour enregistrer le questionnaire.

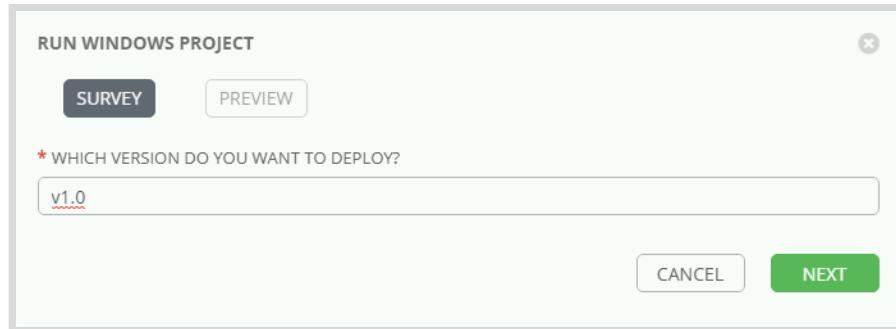
The screenshot shows a modal dialog box titled "Run windows project | SURVEY ON". The main area is labeled "ADD SURVEY PROMPT" and contains the following fields:

- \* PROMPT: A text input field containing "Which version do you want to deploy?"
- DESCRIPTION: An empty text input field.
- \* ANSWER VARIABLE NAME: A text input field containing "version".
- \* ANSWER TYPE: A dropdown menu set to "Text".
- MINIMUM LENGTH: A numeric input field set to 0.
- MAXIMUM LENGTH: A numeric input field set to 1024.
- DEFAULT ANSWER: An empty text input field.
- A checked checkbox labeled "REQUIRED".

On the right side of the dialog, there is a "PREVIEW" panel with the message "PLEASE ADD A SURVEY PROMPT.". At the bottom of the dialog are three buttons: "CLEAR", "+ ADD" (which is highlighted in green), and "CANCEL" and "SAVE".

Figure 11.9: Ajout de questionnaires à un modèle de tâche

Après avoir ajouté un questionnaire à un modèle de tâche, les utilisateurs sont invités à saisir des réponses aux questions du questionnaire lorsqu'ils lancent des tâches avec ce modèle de tâche. Si le modèle de tâche contient des variables supplémentaires ou est configuré pour inviter l'utilisateur à définir des variables supplémentaires, ces valeurs supplémentaires sont alors définies avant que le questionnaire ne soit affiché pour l'utilisateur. Les réponses aux questions du questionnaire remplacent les variables supplémentaires qui ont déjà été définies.



**Figure 11.10: Invite du questionnaire**



## Références

### Guide de l'utilisateur d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/userguide>

## ► Exercice guidé

# Création et utilisation de questionnaires

Dans cet exercice, vous allez créer un questionnaire pour un modèle de tâche dans Red Hat Ansible Tower, puis l'utiliser pour lancer des tâches.

## Résultats

Vous devez pouvoir ajouter un questionnaire à un modèle de tâche existant et lancer une tâche à l'aide d'un questionnaire à partir de l'interface utilisateur Web d'Ansible Tower.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous au système **workstation** Windows à l'aide **d'example.com** en tant que domaine et **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.



### Important

Cet exercice utilise des ressources créées dans le *Chapitre 8 : Interaction avec les utilisateurs et les domaines* ; vous devez effectuer ces exercices avant de commencer celui-ci.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir de la station de travail Windows.

- ▶ 1. Lancez l'éditeur Visual Studio Code et clonez le référentiel **tower** sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone** et sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/tower.git>. À l'invite, sélectionnez le dossier **Documents** dans le répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel. Cette opération clone le référentiel distant dans le dossier **tower**.  
Pour afficher les fichiers, cliquez sur **Open** dans la boîte de dialogue qui s'affiche après le clonage.
- ▶ 2. Modifiez le fichier **index.html.j2** dans le dossier **templates**, de sorte qu'il utilise des variables pour générer le fichier **index.html**. Après avoir effectué les modifications, validez-les et transmettez-les par push à la branche master.
  - 2.1. Accédez au répertoire **templates** situé au niveau supérieur. Sélectionnez le fichier **index.html.j2**, puis modifiez le fichier comme suit :

```
...output omitted...
<h2 style="color: #2e6c80;">This page has been provisioned by {{ student_name }}  

  using Ansible on the {{ ansible_facts['hostname'] }} host</h2>
<h2 style="color: #2e6c80;">This page version is {{ version }}</h2>
```

- 2.2. Cliquez sur **Save**. Notez les icônes de statut situées dans le coin inférieur gauche. Un astérisque s'affiche en regard de **Master**, ce qui indique que des modifications ont été apportées.
  - 2.3. Passez à la vue **Source Control**, puis cliquez sur **+** en regard de **index.html.j2** pour indexer les modifications.
  - 2.4. Saisissez un message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 2.5. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
- ▶ 3. Ajoutez un questionnaire au modèle de tâche **Run tower project**. Ce questionnaire collecte des valeurs pour les variables nécessaires à la génération du fichier **index.html**.
- 3.1. Connectez-vous à Ansible Tower. Cliquez sur **Templates** dans le volet de navigation.
  - 3.2. Dans la liste des modèles disponibles, cliquez sur **Run tower project** pour modifier le modèle de tâche.
  - 3.3. Cliquez sur **ADD SURVEY** pour ajouter un questionnaire.
  - 3.4. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
PROMPT	What version are you deploying?
DESCRIPTION	This is the page version
ANSWER VARIABLE NAME	version
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40
REQUIRED	Coché

- 3.5. Cliquez sur **+ADD** pour ajouter l'invite au questionnaire, puis affichez un aperçu du questionnaire.
- 3.6. Ajoutez une autre question au questionnaire, en remplissant les détails comme suit :

Champ	Valeur
PROMPT	What is your name?
DESCRIPTION	This is the page owner
ANSWER VARIABLE NAME	student_name
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40
REQUIRED	Coché

- 3.7. Cliquez sur **+ADD** pour ajouter l'invite au questionnaire, puis affichez un aperçu du questionnaire.



#### Important

Avant d'enregistrer, assurez-vous que le commutateur du questionnaire est sur **ON** en haut de la fenêtre de l'éiteur du questionnaire.

- 3.8. Cliquez sur **SAVE** pour ajouter le questionnaire au modèle de tâche.
- 3.9. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.
- ▶ 4. Exécutez la tâche de **Run tower project** pour tester votre modèle Jinja et le questionnaire.
- 4.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 4.2. Dans la liste des modèles, sélectionnez le modèle **Run tower project**.



#### Note

Pour sélectionner un projet à modifier, cliquez sur son nom.

- 4.3. Sélectionnez le playbook **manage.yml** dans la liste **PLAYBOOK**.
- 4.4. Cochez la case **PROMPT ON LAUNCH** de l'option **LIMIT** si elle n'est pas cochée.
- 4.5. Si nécessaire, décochez la case **PROMPT ON LAUNCH** des options **CREDENTIAL** et **INVENTORY**.
- 4.6. Dans l'option **INVENTORY**, sélectionnez **Dynamic inventory**.
- 4.7. Dans l'option **CREDENTIAL**, sélectionnez **AD Administrator**.
- 4.8. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.  
Cliquez sur **LAUNCH** pour commencer l'exécution.
- 4.9. Dans la fenêtre **RUN TOWER PROJECT**, saisissez le nom d'hôte **win2.example.com**. Cliquez sur **NEXT**.

- 4.10. Dans le champ **WHAT VERSION ARE YOU DEPLOYING?**, saisissez **1.0** comme valeur. Dans le champ **WHAT IS YOUR NAME?**, saisissez votre nom. Cliquez sur **NEXT**, puis sur **Launch**.
  - 4.11. Observez la sortie en direct de la tâche en cours d'exécution. L'état de la tâche change une fois que le serveur a redémarré avec succès.
  - 4.12. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.
- 5. À partir de **workstation**, ouvrez Chrome et accédez à `http://win2.example.com:8080`. Assurez-vous que la page d'accueil indique ce qui suit :

This page was created using variables from an Ansible Tower survey

This page has been provisioned by John using Ansible on the WIN1 host

This page version is 1.0

To make changes to this page, edit the `templates/index.html.j2` file and rerun the job in Ansible Tower.

List of the modules used to create this site:

<code>win_feature</code>	Used to install IIS features
<code>win_file</code>	Used to create the directory structure
<code>win_template</code>	Used to install that file
<code>win_iis_website</code>	Used to configure the IIS website
<code>win_firewall_rule</code>	Used to open the firewall

Notez le numéro de version de la page et votre nom en tant que créateur de la page.

- 6. Exécutez à nouveau la tâche de **Run tower project**, cette fois avec une valeur de numéro de version différente.
- 6.1. Dans Ansible Tower, cliquez sur **Templates** du volet de navigation.
  - 6.2. Dans la liste des modèles, sélectionnez l'icône de fusée du modèle **Run tower project**.
  - 6.3. Dans la fenêtre **RUN TOWER PROJECT**, saisissez le nom d'hôte `win2.example.com`. Cliquez sur **NEXT**.
  - 6.4. Dans le champ **WHAT VERSION ARE YOU DEPLOYING?**, saisissez **1.5** comme valeur. Dans le champ **WHAT IS YOUR NAME?**, saisissez votre nom. Cliquez sur **NEXT**, puis sur **LAUNCH**.
  - 6.5. Observez la sortie en direct de la tâche en cours d'exécution. L'état de la tâche change une fois que le serveur a redémarré avec succès.
  - 6.6. Vérifiez que le statut de la tâche dans le volet **DETAILS** affiche **Successful**.

- 7. Ouvrez Chrome et accédez à <http://win2.example.com:8080>. Assurez-vous que la page d'accueil indique ce qui suit :

**This page was created using variables from an Ansible Tower survey**

**This page has been provisioned by John using Ansible on the WIN1 host**

**This page version is 1.5**

To make changes to this page, edit the **templates/index.html.j2** file and rerun the job in Ansible Tower.

List of the modules used to create this site:

<b>win_feature</b>	Used to install IIS features
<b>win_file</b>	Used to create the directory structure
<b>win_template</b>	Used to install that file
<b>win_iis_website</b>	Used to configure the IIS website
<b>win_firewall_rule</b>	Used to open the firewall

Notez la nouvelle valeur de la version de la page.

- 8. Modifiez le modèle de tâche **Run tower project** et désactivez le questionnaire.

- 8.1. Dans Ansible Tower, cliquez sur les modèles de tâche du volet de navigation.
- 8.2. Dans la liste des modèles, sélectionnez **Run tower project**.
- 8.3. Cliquez sur **EDIT SURVEY**.
- 8.4. En haut de la fenêtre de l'éditeur de questionnaire, activez le commutateur et positionnez-le sur **OFF**.
- 8.5. Cliquez sur **SAVE** pour mettre à jour le sondage

- 9. Cliquez sur **Log Out** pour vous déconnecter d'Ansible Tower.

L'exercice guidé est maintenant terminé.

# Planification de lancement de tâches

---

## Résultats

À la fin de cette section, vous devez pouvoir planifier l'exécution automatique d'un modèle de tâche dans Red Hat Ansible Tower.

## Planification de l'exécution d'une tâche

Il se peut que vous souhaitiez lancer un modèle de tâche automatiquement à un moment donné ou selon un calendrier particulier. Red Hat Ansible Tower vous permet de configurer des *tâches planifiées* qui lancent des modèles de tâche selon un calendrier personnalisable.

Si vous disposez du rôle **Execute** sur un modèle de tâche, vous pouvez lancer des tâches à partir de ce modèle à l'aide d'un calendrier. Pour configurer des tâches planifiées, commencez par sélectionner **Templates** dans le volet de navigation. Cliquez sur le modèle de tâche que vous souhaitez planifier, puis cliquez sur **SCHEDULES**. Cliquez sur **+** pour créer un calendrier de ce modèle de tâche.

The screenshot shows the 'Prune Old Backups' schedule configuration page. It includes the following fields:

- NAME:** Prune Old Backups
- START DATE:** 9/26/2019
- START TIME (HH24:MM:SS):** 00:00:00
- LOCAL TIME ZONE:** America/Phoenix
- REPEAT FREQUENCY:** Month
- FREQUENCY DETAILS:**
  - EVERY:** 1 MONTH
  - ON DAY:** 1
  - ON THE:** first Sunday
- END:** On Date (12/31/2021)
- END TIME (HH24:MM:SS):** 00:00:00
- SCHEDULE DESCRIPTION:** every month on Sunday until December 30, 2021
- OCCURRENCES (Limited to first 10):** 10
- DATE FORMAT:** UTC
- LOCAL TIME ZONE:** UTC
- UTC:** checked

Figure 11.13: Planification de l'exécution d'une tâche

Saisissez les informations requises :

- **NAME** : nom du calendrier.
- **START DATE** : date de début du calendrier de la tâche.
- **START TIME** : heure à laquelle la tâche associée sera lancée.
- **LOCAL TIME ZONE** : exécutez la commande PowerShell **Get-TimeZone** pour déterminer votre fuseau horaire local.

- **REPEAT FREQUENCY** : fréquence de répétition de la tâche gérée par le calendrier. Vous pouvez choisir **None (run once)**, **Minute**, **Hour**, **Day**, **Week**, **Month** ou **Year**.

En fonction de la fréquence choisie, vous devrez peut-être saisir des informations supplémentaires (par exemple, le lancement d'une tâche tous les deux jours ou le premier dimanche de chaque mois).

Lorsque vous avez terminé, cliquez sur **SAVE** pour enregistrer le calendrier. Vous pouvez désactiver ou activer un calendrier à l'aide du commutateur **ON/OFF** situé en regard du nom du calendrier.

## Désactivation temporaire d'un calendrier

Cliquez sur **Schedules** dans le volet de navigation pour afficher le volet **Scheduled Jobs**. Ce volet liste tous les calendriers établis. À gauche du nom de chaque calendrier se trouve un commutateur **ON/OFF**. Placez le commutateur sur **ON** ou **OFF** pour activer ou désactiver le calendrier, respectivement.

Vous pouvez également modifier ou supprimer n'importe quel calendrier dans ce volet, à condition de disposer de privilèges suffisants pour le faire.

## Tâches de gestion planifiées

Red Hat Ansible Tower inclut deux tâches planifiées spécifiques par défaut. Ces deux calendriers sont destinés aux tâches de gestion intégrées qui effectuent une maintenance périodique sur le serveur Ansible Tower lui-même ; ces tâches nettoient les anciennes informations du journal du flux d'activités et l'exécution de tâches historiques.

**Cleanup Job Schedule** supprime les détails des tâches historiques pour économiser de l'espace. Cette tâche s'exécute une fois par semaine le dimanche pour supprimer les informations relatives aux tâches datant de plus de 120 jours par défaut. Vous pouvez modifier le calendrier pour ajuster le jour ou la fréquence de la tâche, ainsi que les données qu'elle conserve ou supprime.

**Cleanup Activity Schedule** s'exécute une fois par semaine le mardi, afin de supprimer les informations du flux d'activités qui datent de plus de 355 jours. Vous pouvez également modifier le calendrier afin de changer le moment d'exécution de la tâche et la quantité de données qu'il conserve.



### Références

#### Guide de l'utilisateur d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/userguide>

## ► Exercice guidé

# Planification de lancement de tâches

Dans cet exercice, vous allez planifier un modèle de tâche pour qu'il se lance automatiquement selon un calendrier récurrent.

## Résultats

Vous devez pouvoir programmer un modèle de tâche pour qu'il se lance automatiquement.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



### Important

Cet exercice utilise des ressources créées dans le *Chapitre 8 : Interaction avec les utilisateurs et les domaines* ; vous devez effectuer ces exercices avant de commencer celui-ci.

- ▶ 1. Ajoutez un calendrier au modèle de tâche de **Run tower project** qui le configure pour qu'il s'exécute trois minutes après l'heure actuelle. Utilisez l'hôte **win1.example.com**.  
Accédez à Ansible Tower en utilisant le raccourci Ansible Tower sur votre Bureau, ou en accédant à <https://tower.example.com>. Connectez-vous en utilisant le compte **admin** et **RedHat123@!** comme mot de passe.
- ▶ 2. Dans le volet de navigation, cliquez sur **Templates**.
- ▶ 3. Dans la liste des modèles, sélectionnez **Run tower project**.
- ▶ 4. Sélectionnez le playbook **facts.yml** dans la liste **PLAYBOOK**.
- ▶ 5. Assurez-vous que l'option **PROMPT ON LAUNCH** est décochée pour les champs **LIMIT**, **INVENTORY** et **CREDENTIAL**.  
Sélectionnez **Default inventory** dans le champ **INVENTORY**. Dans le champ **LIMIT**, saisissez **win1.example.com** comme nom d'hôte. Assurez-vous que les informations d'identification **DevOps** sont définies dans le champ **Credentials**.
- ▶ 6. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.

- 7. Cliquez sur **SCHEDULES** pour gérer l'exécution automatique d'une tâche à partir de ce modèle de tâche.
- 8. Cliquez sur **+** pour ajouter un calendrier.
- 9. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
NAME	<b>Automatic job run</b>
START DATE	Observez la date actuelle sur <b>workstation</b> et définissez l'exécution de la tâche à la date d'aujourd'hui.
START TIME	Regardez l'heure actuelle sur <b>workstation</b> , puis réglez l'exécution de la tâche sur +3 minutes à partir de l'heure actuelle.
REPEAT FREQUENCY	Choisir une <b>heure</b>
FREQUENCY DETAILS	<ul style="list-style-type: none"> <li>Chaque heure</li> <li>Se termine après cinq occurrences</li> </ul>



### Note

L'heure de début utilise une notation d'horloge de 24 heures, par exemple, saisissez 18:00 pour indiquer 18 heures.

TEMPLATES / Run tower project / SCHEDULES / Automatic job run

**Automatic job run**

* NAME	* START DATE	* START TIME (HH24:MM:SS)
Automatic job run	09/26/2019	19:55:00
* LOCAL TIME ZONE	* REPEAT FREQUENCY	
America/Los_Angeles	Hour	

**FREQUENCY DETAILS**

* EVERY	* END	* OCCURRENCES
1	After	5

**SCHEDULE DESCRIPTION**

every hour for 5 times

OCCURRENCES (Limited to first 10)   DATE FORMAT    LOCAL TIME ZONE    UTC

09-26-2019 19:55:00  
09-26-2019 20:55:00  
09-26-2019 21:55:00  
09-26-2019 22:55:00  
09-26-2019 23:55:00

**CANCEL** **SAVE**

- 10. Cliquez sur **SAVE** pour créer le calendrier.



**Note**

Pour enregistrer le calendrier, vous devez désactiver le questionnaire. Si vous n'avez pas désactivé le questionnaire dans l'exercice précédent, revenez en arrière et suivez les étapes pour le désactiver.

- 11. Cliquez sur **Jobs** dans la barre de navigation et attendez que les tâches planifiées soient exécutées.

L'exercice guidé est maintenant terminé.

# Exécution de workflows complexes

## Résultats

À la fin de cette section, vous devez pouvoir créer une séquence de modèles de tâches à exécuter à l'aide de workflows et de modèles de tâche de workflow dans Red Hat Ansible Tower.

## Modèles de tâche de flux de travail

Dans une section précédente, vous avez appris à utiliser des modèles de tâche pour exécuter un playbook Ansible simple comme tâche. Plus une organisation utilise Ansible, plus le nombre de playbooks Ansible augmente. Pour gérer la complexité au fur et à mesure que le nombre d'utilisateurs et de playbooks augmentent, les playbooks effectuent généralement un ensemble de tâches associées à une fonction particulière. Au lieu d'écrire un grand playbook pour automatiser une opération complexe, envisagez d'exécuter plusieurs playbooks de manière séquentielle.

Au lieu d'écrire un grand playbook pour automatiser une opération complexe, envisagez d'exécuter plusieurs playbooks de manière séquentielle. Par exemple, pour approvisionner un serveur, il se peut que vous souhaitiez utiliser un playbook de l'équipe chargée de la mise en réseau pour allouer une adresse IP au serveur et pour configurer un enregistrement DNS, puis utiliser un autre playbook de l'équipe des opérations pour installer et configurer le système d'exploitation du serveur. Enfin, vous pouvez utiliser un playbook de l'équipe de développement pour déployer une application. En d'autres termes, vous devez suivre un *workflow* particulier pour que le processus réussisse.

Pour gérer ce workflow dans Ansible Tower, les utilisateurs peuvent lancer manuellement plusieurs tâches de manière séquentielle. Toutefois, les tâches doivent être exécutées dans le bon ordre afin que tout fonctionne correctement, tel qu'il est défini par votre workflow.

1. Les tâches relatives à la mise en réseau doivent être exécutées en premier.
2. Les tâches relatives aux opérations sont exécutées si les tâches relatives à la mise en réseau ont été correctement effectuées.
3. De même, les tâches de développement de l'application sont exécutées si les tâches relatives à la mise en réseau et aux opérations ont bien été effectuées.

Enfin, si l'un de ces playbooks échoue, les utilisateurs devront peut-être exécuter d'autres playbooks pour les récupérer.

Pour faciliter la gestion de ce workflow, Red Hat Ansible Tower prend en charge les *modèles de tâche de workflow*. Un modèle de tâche de workflow connecte plusieurs modèles de tâche dans un workflow. Lorsqu'il est lancé, le modèle de tâche de workflow lance une tâche à l'aide du premier modèle de tâche, et selon qu'elle réussit ou échoue, détermine le modèle de tâche à lancer ensuite. Cela permet d'exécuter une séquence de tâches et d'exécuter automatiquement des étapes de récupération en cas d'échec d'une tâche.

Vous pouvez démarrer des modèles de tâche de workflow de différentes façons : manuellement, à partir de l'interface utilisateur Web d'Ansible Tower ; en tant que tâche planifiée ; ou à l'aide de l'API Ansible Tower par un programme externe.

Les modèles de tâche de workflow n'exécutent pas seulement les modèles de tâche en série. À l'aide de l'éditeur de workflow graphique, les modèles de tâche de workflow enchaînent plusieurs modèles de tâche et exécutent différents modèles de tâche selon que le précédent a réussi ou échoué.

## Création de modèles de tâche de flux de travail

Vous devez créer un modèle de tâche de workflow avant de pouvoir définir un workflow ou de l'associer au modèle de tâche de workflow. Les modèles de tâche de workflow peuvent être créés avec ou sans organisation. La création d'un modèle de tâche de workflow dans le contexte d'une organisation nécessite que l'utilisateur ait le rôle **admin** pour cette organisation. La création d'un modèle de tâche de workflow qui ne fait pas partie d'une organisation nécessite un utilisateur doté du type d'utilisateur **System Administrator** singleton.

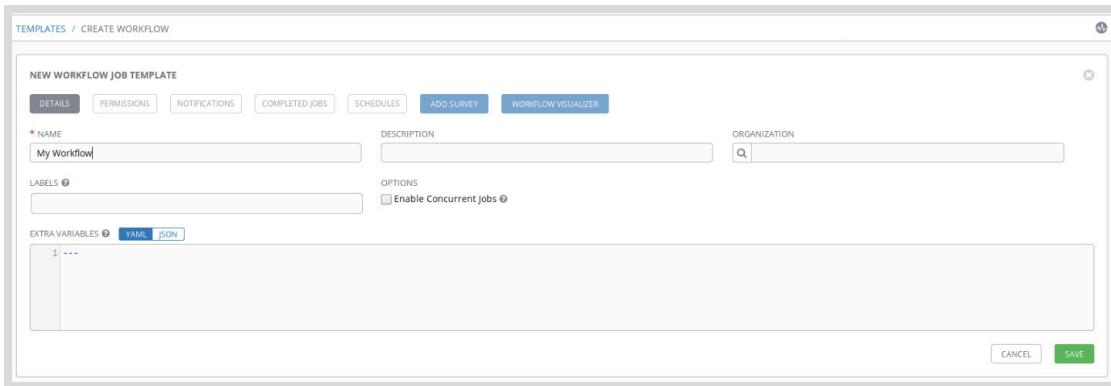


Figure 11.15: Création de modèles de tâche dans Ansible Tower

La création de modèles de tâche de workflow est similaire à la création de modèles de tâche.

1. Cliquez sur **Templates** dans le volet de navigation pour accéder à l'interface de gestion des modèles.
2. Cliquez sur **+**, puis sélectionnez **Workflow Template**.
3. Saisissez un nom unique pour le modèle de tâche de workflow dans le champ **NAME**. Saisissez éventuellement les paires clé/valeur souhaitées dans le champ **EXTRA VARIABLES**.
4. Cliquez sur **SAVE** pour créer le modèle de tâche de workflow. Une fois qu'un modèle de tâche de workflow a été créé, vous pouvez utiliser le *visualiseur de workflow* pour définir et associer un workflow.

## Utilisation du visualiseur de flux de travail

Une fois que vous avez créé un modèle de tâche de workflow, **WORKFLOW VISUALIZER** devient actif dans l'écran de modification du modèle de tâche de workflow. L'éditeur de workflow s'ouvre dans une nouvelle fenêtre. Le visualiseur de workflow est une interface graphique qui permet de définir les modèles de tâche à incorporer dans un workflow, ainsi que l'arborescence de décision, qui doit être utilisée pour enchaîner les modèles de tâche.

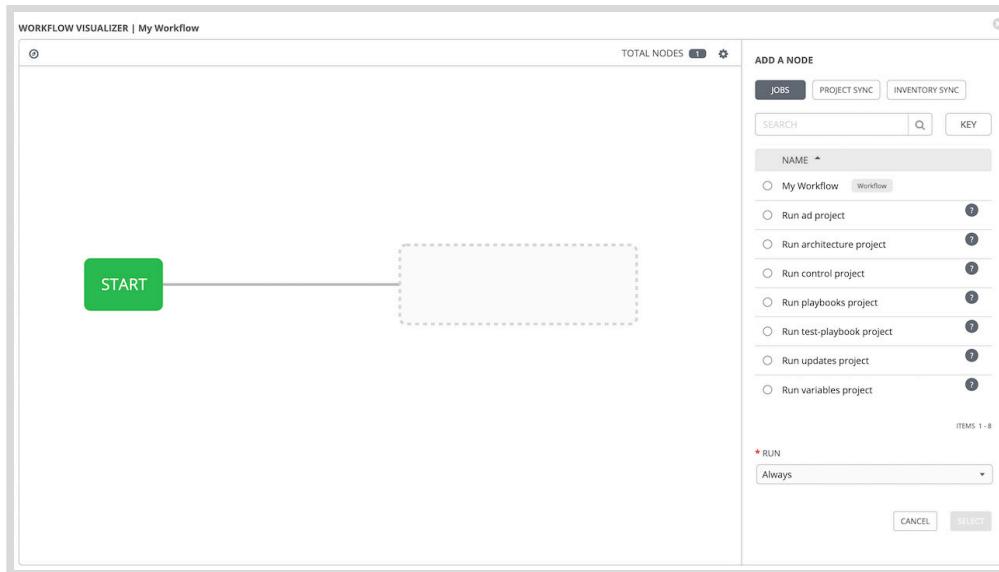


Figure 11.16: Démarrage d'un workflow dans le visualiseur de workflow

Lorsque vous lancez le visualiseur de workflow, il contient un seul nœud **START** qui représente le point de départ de l'exécution du workflow. Cliquez sur **START** pour lancer le processus de modification du workflow ; le visualiseur de workflow affiche une liste de ressources Ansible Tower que vous pouvez ajouter comme première étape du workflow. Sélectionnez le type de ressources souhaité, la ressource spécifique, puis cliquez sur **SELECT** pour ajouter une ressource Ansible Tower sélectionnée comme premier nœud.

En plus des modèles de tâche, vous pouvez également intégrer des tâches qui synchronisent des projets ou des inventaires dans votre workflow. Cela est utile pour s'assurer qu'Ansible Tower met à jour les ressources de projet et d'inventaire avant le démarrage des modèles de tâche qui en dépendent. Pour faciliter l'identification de ces tâches dans le workflow, les nœuds de synchronisation de projet et d'inventaire sont indiqués par un symbole **P** ou **I** respectivement, qui s'affiche dans le coin inférieur gauche du nœud. Cette notation est expliquée par la touche située en haut de l'écran de l'éditeur de workflow. Les nœuds de modèle de tâche ne sont pas signalés par une notation spéciale, car il s'agit du type de nœud principal dans un workflow.

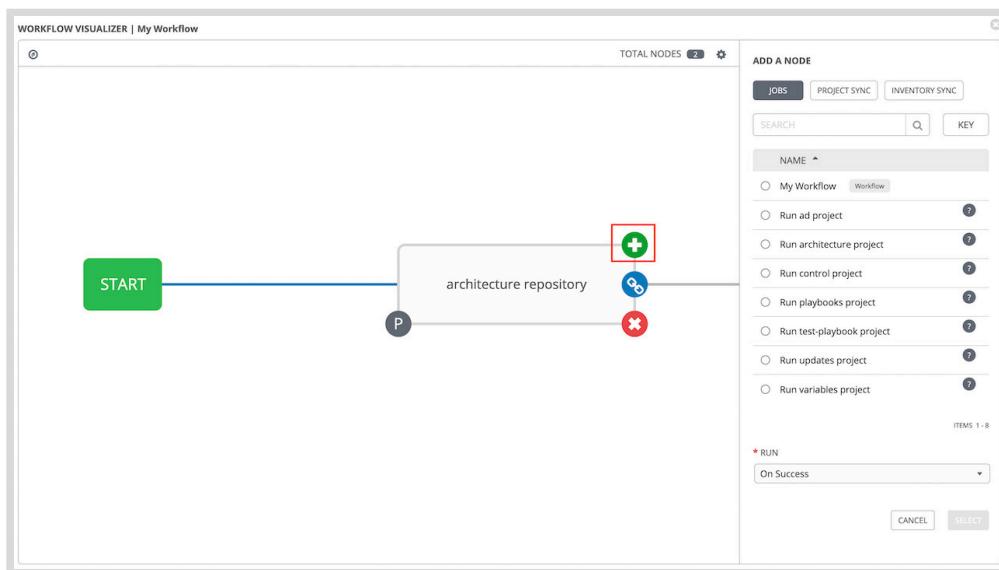


Figure 11.17: Ajout d'un nœud « On Success » au flux de travail

Après l'ajout d'une ressource en tant que premier nœud de workflow, si vous placez le curseur dessus, deux boutons apparaissent. Le bouton rouge - supprime le nœud et le bouton vert + ajoute un autre nœud. Lors de l'ajout de nœuds, l'invite **RUN** s'affiche dans le panneau de sélection des ressources et vous demande de saisir des informations supplémentaires lorsque vous sélectionnez une ressource. Cette invite propose les trois choix suivants pour désigner la relation entre le nouveau nœud et le nœud précédent.

<b>RUN</b>	<b>Relation entre les nœuds</b>
<b>On Success</b>	Ansible Tower exécute cette ressource de nœud une fois les actions associées au nœud précédent correctement effectuées.
<b>On Failure</b>	Ansible Tower exécute cette ressource de nœud après l'échec des actions associées à ce nœud précédent.
<b>Always</b>	Ansible Tower exécute cette ressource de nœud indépendamment du résultat des actions associées au nœud précédent.

Un nœud peut avoir plusieurs nœuds enfants. Vous pouvez ajouter un nœud enfant avec un type d'association de nœud parent **On Success** et un autre nœud enfant avec un type d'association **On Failure**. Cette opération crée une branche dans la structure du workflow de telle sorte qu'une certaine mesure soit prise en cas de réussite d'une action et une autre mesure en cas d'échec.

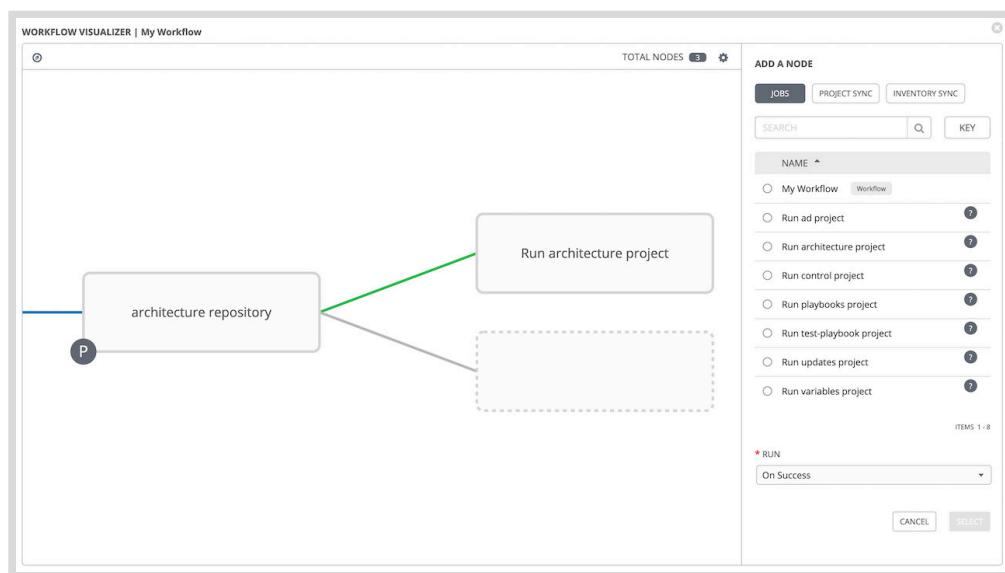
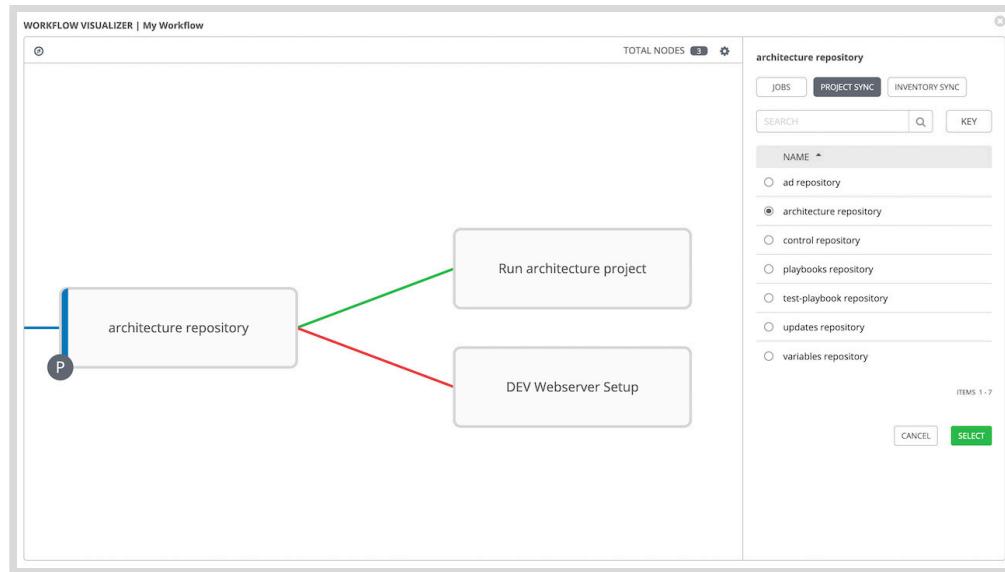


Figure 11.18: Ajout de plusieurs nœuds enfants au flux de travail

Lorsque vous ajoutez des nœuds à un workflow, les lignes de couleur différente qui connectent les nœuds dans l'éditeur de workflow indiquent les relations entre les nœuds parents et enfants. Une ligne verte indique une relation de type **On Success** entre un nœud parent et un nœud enfant, et une ligne rouge indique une relation de type **On Failure**. Une ligne bleue indique une relation de type **Always**.



**Figure 11.19: Relations entre les noeuds du flux de travail**

Une fois que vous avez créé l’arborescence de décision dans l’éditeur de workflow, cliquez sur **SAVE** pour enregistrer le workflow.

## Questionnaires

Les modèles de tâche de workflow ont accès à de nombreuses fonctions qui ont été abordées dans les chapitres précédents pour les modèles de tâche. Comme pour les modèles de tâche, vous pouvez ajouter des questionnaires aux modèles de tâche de workflow pour permettre aux utilisateurs de définir des variables supplémentaires de manière interactive.



### Note

Lorsque des questionnaires sont ajoutés à un modèle de tâche de workflow, chaque tâche exécutée par le workflow peut accéder aux variables supplémentaires qui en résultent.

## Lancement de tâches de flux de travail

Comme pour les modèles de tâche, les utilisateurs ont besoin du rôle **execute** sur le modèle de tâche de workflow pour l'exécuter. Une fois le rôle **execute** assigné, un utilisateur peut lancer une tâche via un modèle de tâche de workflow même s'il n'a pas les autorisations de lancer de manière indépendante les modèles de tâche qu'il utilise.

Le lancement d'un modèle de tâche de workflow est similaire au lancement d'un modèle de tâche, comme illustré dans ce qui suit :

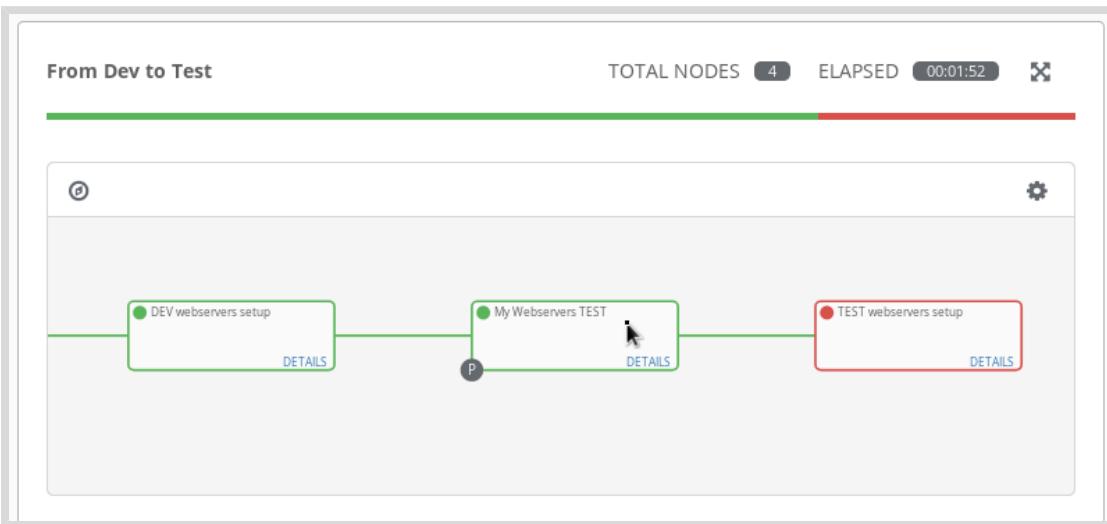
1. Cliquez sur **Templates** dans le volet de navigation pour accéder à l'interface de gestion des modèles.
2. Cliquez sur l'icône de fusée du modèle de tâche de workflow pour lancer la tâche.

## Évaluation de l'exécution d'une tâche de flux de travail

Après le lancement d'une tâche de workflow, Ansible Tower affiche la page Job Details de la tâche en cours d'exécution. Cette page comporte deux volets : le volet **DETAILS** affiche les détails de

l'exécution de la tâche de workflow, et le volet de progression du workflow affiche la progression de la tâche via les étapes du workflow.

À la fin de chaque étape, son nœud apparaît en vert ou en rouge, selon que les actions associées à cette étape dans le flux de travail ont réussi ou échoué. Les progressions d'une étape à l'autre sont représentées par des lignes colorées indiquant la décision responsable de la progression. Une ligne verte indique une progression **On Success** et une ligne rouge indique une progression **On Failure**. Une ligne bleue indique une progression **Always**.



**Figure 11.20: Progression de la tâche de flux de travail**

Les détails de l'exécution de la tâche de workflow peuvent être affichés pendant ou après l'exécution. Cliquez sur le lien **DETAILS** pour chaque nœud du diagramme de workflow, afin d'afficher les détails relatifs à un travail en cours d'exécution ou à un travail terminé. Cliquez sur ce lien pour afficher les résultats et la sortie standard de l'exécution de la tâche.



## Références

### Guide de l'utilisateur d'Ansible Tower

<https://docs.ansible.com/ansible-tower/latest/html/userguide/>

## ► Exercice guidé

# Exécution de workflows complexes

Dans cet exercice, vous allez créer un modèle de tâche de workflow qui exécutera plusieurs playbooks, selon que les playbooks précédents sont terminés ou qu'ils ont signalé des échecs.

## Résultats

Vous devez pouvoir créer un modèle de tâche de workflow et lancer un workflow à partir de l'interface Web d'Ansible Tower.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.



### Important

Cet exercice utilise des ressources créées dans le *Chapitre 8 : Interaction avec les utilisateurs et les domaines* ; vous devez effectuer ces exercices avant de commencer celui-ci.

- ▶ 1. Accédez à Ansible Tower en utilisant le raccourci Ansible Tower sur votre Bureau, ou en accédant à <https://tower.example.com>. Connectez-vous en utilisant le compte **admin** et **RedHat123@!** comme mot de passe.
- ▶ 2. Exécutez votre playbook **prepare.yml** à partir du modèle de tâche **Run tower project**.
  - 2.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 2.2. Cliquez sur le modèle de tâche **Run tower project** pour l'ouvrir.
  - 2.3. Sélectionnez le playbook **prepare.yml** dans la liste **PLAYBOOK**.
  - 2.4. Assurez-vous que la case **PROMPT ON LAUNCH** pour l'option **LIMIT** est cochée et que le champ de texte est vide.
  - 2.5. Assurez-vous que la case **PROMPT ON LAUNCH** pour l'option **CREDENTIAL** est décochée. Dans la liste des informations d'identification disponibles, choisissez **AD Administrator**.

- 2.6. Assurez-vous que la case à cocher **PROMPT ON LAUNCH** pour l'option **INVENTORY** est activée. Dans la liste des inventaires disponibles, choisissez **Dynamic Inventory**.
- 2.7. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
- ▶ 3. Modifiez le modèle de tâche **Run tower project** existant. Limitez le modèle de tâche pour qu'il s'exécute uniquement sur **win2.example.com**, en utilisant les informations d'identification **AD Administrator**, **Dynamic inventory** et le playbook **manage.yml**.
- 3.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 3.2. Dans la liste des modèles, sélectionnez **Run tower project**.
  - 3.3. Sélectionnez le playbook **manage.yml** dans la liste **PLAYBOOK**.
  - 3.4. Dans le champ **LIMIT**, saisissez le nom d'hôte **win2.example.com**.
  - 3.5. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.
- ▶ 4. Créez un modèle de tâche de workflow **Deploy the web page**.
- 4.1. Cliquez sur **Templates** dans le volet de navigation.
  - 4.2. Cliquez sur le bouton **+** pour ajouter un nouveau modèle de tâche de workflow.
  - 4.3. Sélectionnez **Workflow Template** dans la liste.
  - 4.4. Sur l'écran suivant, renseignez les informations comme suit :
- | Champ        | Valeur                       |
|--------------|------------------------------|
| NAME         | Deploy the web page          |
| DESCRIPTION  | Deploy the web page workflow |
| ORGANIZATION | Valeur par défaut            |
- 4.5. Cliquez sur **SAVE** pour créer le modèle de tâche de workflow.
- ▶ 5. Configurez le workflow du modèle de tâche de workflow **Deploy the web page**.
- 5.1. La fenêtre **WORKFLOW VISUALIZER** s'ouvre automatiquement.
  - 5.2. Cliquez sur **START** pour ajouter la première action. Cette opération affiche la liste des actions disponibles dans le panneau **ADD A NODE**.
  - 5.3. Cliquez sur **PROJECT SYNC** pour afficher la liste des projets disponibles.
  - 5.4. Sélectionnez **tower repository**, puis cliquez sur **SELECT**. Le nœud **START** est alors relié par une ligne bleue (always perform) au nœud du projet **tower repository** dans la fenêtre Workflow Visualizer.
  - 5.5. Déplacez votre souris sur le nouveau nœud et cliquez sur **+** pour ajouter une action à la suite de la synchronisation du projet **tower repository**. Cette opération affiche la liste des actions disponibles dans le panneau **ADD A NODE**.
  - 5.6. Cliquez sur **PROJECT SYNC** pour afficher la liste des projets disponibles.

- 5.7. Sélectionnez **broken repository**, puis cliquez sur **SELECT**. Cela lie le nœud du projet **tower repository** à un nouveau nœud pour la synchronisation de projet **broken repository** dans la fenêtre Workflow Visualizer. Le lien vert indique que la progression n'a lieu qu'après l'accomplissement du nœud précédent.
  - 5.8. Déplacez votre souris sur le nouveau nœud et cliquez sur **+** pour ajouter une action à la suite de la synchronisation du projet **broken repository**. Une liste des actions à effectuer s'affiche alors.
  - 5.9. Cliquez sur **JOBS** si nécessaire, puis sélectionnez le modèle de tâche **Run tower project**.
  - 5.10. Sélectionnez **On Success** dans la liste **RUN**, puis cliquez sur **SELECT**. Cela lie le nœud du projet **broken repository** à un nouveau nœud pour le modèle de tâche **Run tower project** dans la fenêtre Workflow Visualizer. Le lien vert indique que la progression n'a lieu qu'après l'accomplissement du nœud précédent.
  - 5.11. Déplacez votre souris sur le nœud **broken repository** et cliquez sur **+** pour ajouter une action à la suite de la synchronisation du projet **broken repository**. Une liste des actions à effectuer s'affiche alors. Cette fois, vous allez ajouter un nœud à exécuter en cas d'échec du nœud de synchronisation du projet **broken repository**.
  - 5.12. Cliquez sur **JOBS** s'il n'est pas encore affiché, puis sélectionnez le modèle de tâche **Clean win2**.
  - 5.13. Sélectionnez **On Failure** dans la liste **RUN**, puis cliquez sur **SELECT**. Cela lie le nœud du projet **broken repository** à un nouveau nœud pour le modèle de tâche **Clean win2** dans la fenêtre Workflow Visualizer. Le lien rouge indique que la progression n'a lieu qu'après l'échec du nœud précédent.
  - 5.14. Cliquez sur **SAVE** pour enregistrer le workflow.  
Une fois l'éditeur fermé, cliquez sur **SAVE** encore une fois pour enregistrer le modèle de tâche.
- ▶ 6. Lancez une tâche à l'aide du modèle de tâche de workflow **Deploy the web page**.
- 6.1. Cliquez sur **Templates** dans le volet de navigation.
  - 6.2. Cliquez sur l'icône de fusée du modèle de tâche de workflow **Deploy the web page** pour lancer la tâche. Cela vous redirige vers une page d'état détaillée du workflow en cours d'exécution.
  - 6.3. Observez les tâches en cours d'exécution dans le workflow. Cliquez sur **Details** pour chaque tâche en cours d'exécution pour afficher une sortie plus détaillée.
  - 6.4. Notez que le nœud **broken repository** est marqué comme étant red, et que le nœud **Clean win2** s'exécute. La bordure rouge autour du nœud **broken repository** indique que la tâche associée à ce nœud a échoué ; en fonction de la logique de workflow, le nœud **Clean win2** est exécuté et correctement terminé.
  - 6.5. Vérifiez que le statut du workflow dans le volet **DETAILS** affiche **Successful**.
- ▶ 7. Corrigez le problème dans le projet **broken repository**, puis exécutez à nouveau le workflow. Observez comment la logique de workflow exécute une sélection de nœuds différente en fonction de l'exécution réussie de chaque nœud du workflow.

- 7.1. Cliquez sur **Projects** dans le volet de navigation.
  - 7.2. Cliquez sur le projet **broken repository**.
  - 7.3. Dans le champ **SCM URL**, saisissez l'URL de synchronisation du référentiel, <https://gitlab.example.com/student/windows.git>.
  - 7.4. Cliquez sur **SAVE** pour enregistrer les modifications.
- **8.** Les workflows peuvent utiliser des questionnaires pour affecter des valeurs à des variables existantes. Ajoutez un questionnaire au workflow **Deploy the web page**.
- 8.1. Ajoutez un questionnaire au workflow **Deploy the web page**. Ce questionnaire collecte des valeurs pour les variables nécessaires à la génération du fichier **index.html**.
  - 8.2. Cliquez sur **Templates** dans le volet de navigation.
  - 8.3. Dans la liste des modèles disponibles, cliquez sur **Deploy the web page** pour modifier le modèle de workflow.
  - 8.4. Cliquez sur **ADD SURVEY** pour ajouter un questionnaire.
  - 8.5. Sur l'écran suivant, renseignez les informations comme suit :

Champ	Valeur
PROMPT	What version are you deploying?
DESCRIPTION	This is the page version
ANSWER VARIABLE NAME	version
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40
REQUIRED	Sélectionné

- 8.6. Cliquez sur **+ADD** pour ajouter l'invite de questionnaire au questionnaire. Un aperçu du questionnaire s'affiche alors.
- 8.7. Ajoutez une autre question au questionnaire, en renseignant les informations comme suit :

Champ	Valeur
PROMPT	What is your name?
DESCRIPTION	This is the page owner
ANSWER VARIABLE NAME	student_name
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40
REQUIRED	Sélectionné

- 8.8. Cliquez sur **+ADD** pour ajouter cette invite au questionnaire. Un aperçu du questionnaire s'affiche alors.



### Important

Avant d'enregistrer, assurez-vous que le commutateur **ON/OFF** est sur **ON**.

- 8.9. Cliquez sur **SAVE** pour ajouter le sondage au modèle de tâche.
- 8.10. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.
- ▶ 9. Lancez le workflow modifié avec le questionnaire et vérifiez que chacun des nœuds est correctement exécuté. Cliquez sur **Templates** dans le volet de navigation.
- 9.1. Cliquez sur l'icône de fusée du modèle de workflow **Deploy the web page** pour lancer le workflow.
- 9.2. La fenêtre **DEPLOY THE WEB PAGE** s'affiche. Dans le champ **WHAT VERSION ARE YOU DEPLOYING?**, saisissez **3.0**. Dans le champ **WHAT IS YOUR NAME?**, saisissez votre nom. Cliquez sur **NEXT**, puis sur **LAUNCH**.
- 9.3. Observez les tâches en cours d'exécution du workflow. Cliquez sur **DETAILS** pour chaque tâche en cours d'exécution pour afficher une sortie plus détaillée. Tous les nœuds de workflow doivent terminer correctement leurs tâches.
- 9.4. Vérifiez que le statut du workflow dans le volet **DETAILS** est **Successful**.
- ▶ 10. Ouvrez Chrome, puis accédez à <http://win2.example.com:8080>. La nouvelle page Web est déployée à l'aide de votre workflow, y compris toutes les données fournies par le questionnaire.
- ▶ 11. Cliquez sur l'icône **Log Out** pour vous déconnecter d'Ansible Tower.

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Construction de workflows Ansible Tower

### Liste de contrôle des performances

Au cours de cet atelier, vous allez ajouter des utilisateurs et des équipes, et créer et utiliser des questionnaires associés à un nouveau workflow.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Ajouter deux nouveaux utilisateurs.
- Créer une nouvelle équipe.
- Créer un workflow.
- Ajouter un questionnaire au workflow et exécuter le workflow.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. Ajoutez deux utilisateurs en tant que **Normal Users** à l'organisation **Default** à l'aide des informations suivantes :

	<b>User 1</b>	<b>User 2</b>
FIRST NAME	Adam	Ophelia
LAST NAME	Brilliant	Dunham
ORGANIZATION	Valeur par défaut	Valeur par défaut
EMAIL	adam@example.com	ophelia@example.com
USERNAME	adam	ophelia
PASSWORD	RedHat123@!	RedHat123@!
CONFIRM PASSWORD	RedHat123@!	RedHat123@!

2. Créez une équipe appelée **Operations** dans l'organisation **Default**.
3. Ajoutez **adam** et **ophelia** à l'équipe **Operations**.

4. Copiez le modèle de tâche de **Run tower project** existant et nommez-le **TEST webservers setup**.

Utilisez les informations suivantes pour modifier le nouveau modèle de tâche :

Champ	Valeur
NAME	<b>TEST webservers setup</b>
INVENTORY	<b>Dynamic inventory</b>
CREDENTIAL	<b>AD Administrator</b>
PROJECT	<b>tower repository</b>
PLAYBOOK	<b>manage.yml</b>
LIMIT	<b>win2.example.com</b>

5. Copiez le modèle de tâche de **Run tower project** existant et nommez-le **PROD webservers setup**.

Utilisez les informations suivantes pour modifier le modèle de tâche copié :

Champ	Valeur
NAME	<b>PROD webservers setup</b>
INVENTORY	<b>Dynamic inventory</b>
CREDENTIAL	<b>AD Administrator</b>
PROJECT	<b>tower repository</b>
PLAYBOOK	<b>manage.yml</b>
LIMIT	<b>windc.example.com</b>

6. Créez un modèle de tâche de workflow **From Test to Prod** à l'aide des informations suivantes :

Champ	Valeur
NAME	From Test to Prod
DESCRIPTION	Deploy to Test and on success deploy to Prod
ORGANIZATION	Valeur par défaut

7. Configurez le modèle de tâche de workflow **From Test to Prod** de sorte qu'il contienne les étapes suivantes :
- Synchronisez le projet **tower repository**.
  - Une fois l'étape précédente terminée, lancez le modèle de tâche **TEST webservers setup**.
  - Une fois l'étape précédente terminée, lancez le modèle de tâche **PROD webservers setup**.
8. Ajoutez un questionnaire contenant les informations suivantes au modèle de tâche de workflow **From Test to Prod**. Assurez-vous qu'il s'agit d'un questionnaire obligatoire.

Champ	Valeur
PROMPT	What version are you deploying?
DESCRIPTION	This version number will be displayed at the bottom of the index page.
ANSWER VARIABLE NAME	version
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40

Ajoutez une deuxième question au questionnaire avec les informations suivantes :

Champ	Valeur
PROMPT	What is your name?
DESCRIPTION	This is the page owner
ANSWER VARIABLE NAME	student_name
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40

9. Lancez un workflow à l'aide du modèle de tâche de workflow **From Test to Prod**. Lorsque le questionnaire vous y invite, saisissez **v1.3** pour la version de page et votre nom.
10. Vérifiez que les serveurs Web ont été mis à jour sur **win2.example.com** et **windc.example.com**. Ouvrez Chrome et accédez à <http://win2.example.com:8080> et <http://windc.example.com:8080>.

L'atelier est maintenant terminé.

## ► Solution

# Construction de workflows Ansible Tower

### Liste de contrôle des performances

Au cours de cet atelier, vous allez ajouter des utilisateurs et des équipes, et créer et utiliser des questionnaires associés à un nouveau workflow.

### Résultats

Vous serez en mesure de réaliser les tâches suivantes :

- Ajouter deux nouveaux utilisateurs.
- Créer une nouvelle équipe.
- Créer un workflow.
- Ajouter un questionnaire au workflow et exécuter le workflow.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

1. Ajoutez deux utilisateurs en tant que **Normal Users** à l'organisation **Default** à l'aide des informations suivantes :

	<b>User 1</b>	<b>User 2</b>
FIRST NAME	Adam	Ophelia
LAST NAME	Brilliant	Dunham
ORGANIZATION	Valeur par défaut	Valeur par défaut
EMAIL	adam@example.com	ophelia@example.com
USERNAME	adam	ophelia
PASSWORD	RedHat123@!	RedHat123@!
CONFIRM PASSWORD	RedHat123@!	RedHat123@!

- 1.1. Connectez-vous à l'interface Web Ansible Tower en utilisant **admin** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.

- 1.2. Cliquez sur **Users** dans le volet de navigation pour gérer les utilisateurs.
- 1.3. Cliquez sur **+** pour ajouter un nouvel utilisateur.
- 1.4. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
FIRST NAME	Adam
LAST NAME	Brilliant
ORGANIZATION	Valeur par défaut
EMAIL	adam@example.com
USERNAME	adam
PASSWORD	RedHat123@!
CONFIRM PASSWORD	RedHat123@!
USER TYPE	Normal User

- 1.5. Cliquez sur **SAVE** pour créer le nouvel utilisateur.
- 1.6. Faites défiler le volet suivant et cliquez sur le bouton **+** pour ajouter un autre utilisateur.
- 1.7. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
FIRST NAME	Ophelia
LAST NAME	Dunham
ORGANIZATION	Valeur par défaut
EMAIL	ophelia@example.com
USERNAME	ophelia
PASSWORD	RedHat123@!
CONFIRM PASSWORD	RedHat123@!
USER TYPE	Normal User

- 1.8. Cliquez sur **SAVE** pour créer le nouvel utilisateur.
2. Créez une équipe appelée **Operations** dans l'organisation **Default**.
  - 2.1. Cliquez sur **Teams** dans la fenêtre de navigation pour gérer les équipes.
  - 2.2. Cliquez sur **+** pour ajouter une nouvelle équipe.
  - 2.3. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
NAME	Opérations
DESCRIPTION	Ops Team
ORGANIZATION	Valeur par défaut

- 2.4. Cliquez sur **SAVE** pour créer l'équipe **Operations**.
3. Ajoutez **adam** et **ophelia** à l'équipe **Operations**.
- 3.1. Dans la fenêtre de l'éditeur de l'équipe **Operations**, cliquez sur **USERS** pour gérer les utilisateurs de l'équipe.
  - 3.2. Cliquez sur **+** pour ajouter un nouvel utilisateur à l'équipe.
  - 3.3. Cochez la case en regard de **adam** pour sélectionner cet utilisateur.
  - 3.4. Répétez la même étape pour **ophelia**.
  - 3.5. Cliquez sur **SAVE**.
  - 3.6. Dans la fenêtre suivante, vérifiez qu'**adam** et **ophelia** ont le rôle de **Member** dans l'équipe **Operations**.
4. Copiez le modèle de tâche de **Run tower project** existant et nommez-le **TEST webserver setup**. Utilisez les informations suivantes pour modifier le nouveau modèle de tâche :

Champ	Valeur
NAME	<b>TEST webserver setup</b>
INVENTORY	<b>Dynamic inventory</b>
CREDENTIAL	<b>AD Administrator</b>
PROJECT	<b>tower repository</b>
PLAYBOOK	<b>manage.yml</b>
LIMIT	<b>win2.example.com</b>

- 4.1. Cliquez sur **Templates** dans le volet de navigation.
- 4.2. Cliquez sur l'icône de copie du modèle de tâche **Run tower project**.
- 4.3. Cliquez sur le modèle **Run tower project@DATE**.
- 4.4. Utilisez les informations suivantes pour modifier le nouveau modèle de tâche :

Champ	Valeur
NAME	<b>TEST webservers setup</b>
INVENTORY	<b>Dynamic inventory</b>
CREDENTIAL	<b>AD Administrator</b>
PROJECT	<b>tower repository</b>
PLAYBOOK	<b>manage.yml</b>
LIMIT	<b>win2.example.com</b>

- 4.5. Cliquez sur **SAVE**.
5. Copiez le modèle de tâche de **Run tower project** existant et nommez-le **PROD webservers setup**.

Utilisez les informations suivantes pour modifier le modèle de tâche copié :

Champ	Valeur
NAME	<b>PROD webservers setup</b>
INVENTORY	<b>Dynamic inventory</b>
CREDENTIAL	<b>AD Administrator</b>
PROJECT	<b>tower repository</b>
PLAYBOOK	<b>manage.yml</b>
LIMIT	<b>windc.example.com</b>

- 5.1. Cliquez sur **Templates** dans le volet de navigation.
- 5.2. Cliquez sur l'icône de copie du modèle de tâche **Run tower project**.
- 5.3. Cliquez sur le modèle **Run tower project@DATE**.

5.4. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
NAME	<b>PROD webservers setup</b>
INVENTORY	<b>Dynamic inventory</b>
CREDENTIAL	<b>AD Administrator</b>
PROJECT	<b>tower repository</b>
PLAYBOOK	<b>manage.yml</b>
LIMIT	<b>windc.example.com</b>

- 5.5. Cliquez sur **SAVE**.
6. Créez un modèle de tâche de workflow **From Test to Prod** à l'aide des informations suivantes :

Champ	Valeur
NAME	From Test to Prod
DESCRIPTION	Deploy to Test and on success deploy to Prod
ORGANIZATION	Valeur par défaut

- 6.1. Cliquez sur **Templates** dans le volet de navigation.
- 6.2. Cliquez sur le bouton **+** pour ajouter un nouveau modèle de tâche de workflow.
- 6.3. Sélectionnez **Workflow Template** dans la liste.
- 6.4. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
NAME	<b>From Test to Prod</b>
DESCRIPTION	<b>Deploy to Test and on success deploy to Prod</b>
ORGANIZATION	<b>Valeur par défaut</b>

- 6.5. Cliquez sur **SAVE** pour créer le modèle de tâche de workflow.
7. Configurez le modèle de tâche de workflow **From Test to Prod** de sorte qu'il contienne les étapes suivantes :
- Synchronisez le projet **tower repository**.
  - Une fois l'étape précédente terminée, lancez le modèle de tâche **TEST webservers setup**.
  - Une fois l'étape précédente terminée, lancez le modèle de tâche **PROD webservers setup**.
- 7.1. Cliquez sur **WORKFLOW VISUALIZER** pour l'ouvrir.
- 7.2. Cliquez sur **START** pour ajouter la première action dans le workflow. La liste des actions disponibles à effectuer s'affiche dans le panneau **ADD A NODE**.
- 7.3. Cliquez sur **PROJECT SYNC** pour afficher la liste des projets disponibles. Sélectionnez **tower repository**, puis cliquez sur **SELECT**. Dans la fenêtre Workflow Visualizer, le nœud **START** est alors relié par une ligne bleue au nœud du projet **tower repository**, ce qui indique que cette étape sera toujours effectuée.
- 7.4. Déplacez votre souris sur le nouveau nœud, puis cliquez sur **+** pour ajouter une action à la suite de la synchronisation du projet **tower repository**. La liste des actions disponibles à effectuer s'affiche.

- 7.5. Dans le volet **ADD A NODE**, cliquez sur **JOBS** s'il n'est pas déjà sélectionné. Sélectionnez le modèle de tâche **TEST webservers setup**. Selon la taille de la fenêtre de votre navigateur Web, vous devrez peut-être consulter la page suivante pour trouver le modèle de tâche. Sélectionnez **On Success** dans la liste **RUN**, puis cliquez sur **SELECT**. La fenêtre du visualiseur de workflow affiche le nœud du projet **tower repository** relié par une ligne verte au modèle de tâche **TEST webservers setup**. Cela indique que si la synchronisation du projet **tower repository** est bien effectuée, le modèle de tâche **TEST webservers setup** sera lancé.
- 7.6. Déplacez votre souris sur le nouveau nœud et cliquez sur **+** pour ajouter une action après le modèle de tâche **TEST webservers setup**.
- 7.7. Cliquez sur **JOBS** s'il n'est pas encore affiché, puis sélectionnez le modèle de tâche **PROD webservers setup**. Sélectionnez **On Success** dans la liste **RUN**, puis cliquez sur **SELECT**.
- 7.8. Cliquez sur **SAVE** pour enregistrer le modèle de tâche de workflow.
- 8.** Ajoutez un questionnaire contenant les informations suivantes au modèle de tâche de workflow **From Test to Prod**. Assurez-vous qu'il s'agit d'un questionnaire obligatoire.

Champ	Valeur
PROMPT	What version are you deploying?
DESCRIPTION	This version number will be displayed at the bottom of the index page.
ANSWER VARIABLE NAME	version
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40

Ajoutez une deuxième question au questionnaire avec les informations suivantes :

Champ	Valeur
PROMPT	What is your name?
DESCRIPTION	This is the page owner
ANSWER VARIABLE NAME	student_name
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40

- 8.1. Cliquez sur **ADD SURVEY** pour ajouter un questionnaire.
- 8.2. Dans la fenêtre suivante, renseignez les informations comme suit :

Champ	Valeur
PROMPT	What version are you deploying?
DESCRIPTION	This version number will be displayed at the bottom of the index page.
ANSWER VARIABLE NAME	deployment_version
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40
REQUIRED	Sélectionné

- 8.3. Cliquez sur **ADD** pour ajouter l'invite de questionnaire au questionnaire. Un aperçu du questionnaire s'affiche alors.
- 8.4. Ajoutez une autre question au questionnaire, en renseignant les informations comme suit :

Champ	Valeur
PROMPT	What is your name?
DESCRIPTION	This is the page owner
ANSWER VARIABLE NAME	student_name
ANSWER TYPE	Text
MINIMUM LENGTH	1
MAXIMUM LENGTH	40
REQUIRED	Coché

- 8.5. Cliquez sur **+ADD** pour ajouter l'invite de questionnaire au questionnaire. Un aperçu du questionnaire s'affiche alors.



### Important

Avant d'enregistrer, assurez-vous que le commutateur **ON/OFF** est sur **ON** dans la fenêtre de l'éiteur du questionnaire.

- 8.6. Cliquez sur **SAVE** pour ajouter le questionnaire au modèle de tâche.
- 8.7. Cliquez sur **SAVE** pour mettre à jour le modèle de tâche.
9. Lancez un workflow à l'aide du modèle de tâche de workflow **From Test to Prod**. Lorsque le questionnaire vous y invite, saisissez **v1.3** pour la version de page et votre nom.
- 9.1. Faites défiler vers le bas jusqu'à la section **TEMPLATES**.

- 9.2. Cliquez sur l'icône de fusée du modèle de tâche de workflow **From Test to Prod** pour lancer le workflow. Le questionnaire que vous venez de créer s'ouvre et vous invite à effectuer une saisie.
  - 9.3. Dans le champ **WHAT VERSION ARE YOU DEPLOYING?**, saisissez **1.3**. Dans le champ **WHAT IS YOUR NAME?**, saisissez votre nom. Cliquez sur **NEXT**, puis sur **LAUNCH**. Cela vous redirige vers une page d'état détaillée du workflow en cours d'exécution.
  - 9.4. Observez les tâches en cours d'exécution du workflow. Cliquez sur **DETAILS** pour chaque tâche en cours d'exécution pour afficher une sortie plus détaillée de la tâche.
- 10.** Vérifiez que les serveurs Web ont été mis à jour sur **win2.example.com** et **windc.example.com**. Ouvrez Chrome et accédez à **http://win2.example.com:8080** et **http://windc.example.com:8080**.
- 10.1. Ouvrez un navigateur Web et accédez à **http://win2.example.com:8080** et **http://windc.example.com:8080** dans des onglets séparés. Vous devez voir la nouvelle page avec le nom et la version corrects.
  - 10.2. Lorsque vous êtes prêt, déconnectez-vous de l'interface utilisateur Web d'Ansible Tower.

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Les organisations sont des collections logiques d'utilisateurs, d'équipes, de projets et d'inventaires.
- Il existe trois types d'utilisateur suivants : **System Administrator**, **System Auditor** et **Normal User**.
- **System Administrator** et **System Auditor** sont des rôles singleton qui accordent respectivement l'accès en lecture/écriture et en lecture seule à toutes les ressources Ansible Tower.
- Une équipe est un groupe d'utilisateurs, et vous pouvez utiliser une équipe pour faciliter l'affectation de rôles particuliers sur des ressources Ansible Tower à un ensemble d'utilisateurs.
- Utilisez des questionnaires pour activer une automatisation en une seule étape. Vous pouvez inviter les utilisateurs à saisir les valeurs des variables supplémentaires que les playbooks utilisent.
- Un modèle de tâche de workflow vous permet de lancer une tâche qui exécute plusieurs playbooks en séquence.



## chapitre 12

# Révision complète

### Objectif

Tâches de révision depuis *Microsoft Windows Automation with Red Hat Ansible*

### Résultats

- Tâches de révision depuis *Microsoft Windows Automation with Red Hat Ansible*

### Sections

- Révision complète

### Atelier

- Atelier : Examen général de l'atelier 1
- Atelier : Examen général de l'atelier 2
- Atelier : Examen général de l'atelier 3
- Atelier : Examen général de l'atelier 4

# Révision complète

---

## Résultats

Après avoir terminé cette section, vous devez avoir révisé et actualisé les connaissances et les compétences acquises dans *Microsoft Windows Automation with Red Hat Ansible*.

## Révision de Microsoft Windows Automation with Red Hat Ansible

Avant de commencer la révision complète de ce cours, vous devez être familiarisé avec les rubriques abordées dans chaque chapitre.

Vous pouvez vous référer aux précédentes sections du manuel pour en savoir plus.

### Chapitre 1, Présentation de Red Hat Ansible Automation

Décrire l'objectif et les avantages de l'automatisation des tâches d'administration de Windows Server, ainsi que l'architecture de base d'une solution basée sur Red Hat Ansible Automation.

- Décrire les concepts fondamentaux de l'automatisation avec Red Hat Ansible Automation, sa conception de base et les cas d'utilisation courants.
- Décrire l'architecture d'une mise en œuvre d'automatisation Windows à l'aide de la solution Red Hat Ansible Tower et d'un référentiel Git comme point central de contrôle.
- Récupérer, gérer, modifier et stocker des fichiers dans un système de contrôle de version Git distant, tel que GitHub ou GitLab, à l'aide de Visual Studio Code.

### Chapitre 2, Exécution de commandes simples d'automatisation

Préparer des hôtes Microsoft Windows en vue de l'automatisation et Red Hat Ansible Tower en tant que système de contrôle d'automatisation central, et exécuter des tâches d'automatisation uniques sur ces hôtes à partir d'Ansible Tower.

- Configurer les systèmes Microsoft Windows pour fournir les conditions préalables et l'accès nécessaires à la gestion par Red Hat Ansible Automation.
- Configurer Red Hat Ansible Tower avec un inventaire des hôtes Windows à gérer, ainsi que les informations d'identification nécessaires pour vous authentifier et vous connecter à ces hôtes.
- Décrire ce qu'est un module Ansible et exécuter une seule opération sur les hôtes gérés sous la forme d'une commande ad hoc à l'aide de Red Hat Ansible Tower.

### Chapitre 3, Mise en œuvre de playbooks Ansible

Rédiger un simple playbook pour automatiser les tâches sur plusieurs hôtes basés sur Microsoft Windows, puis utiliser Red Hat Ansible pour l'exécuter.

- Rédiger un playbook de base et le stocker dans un référentiel Git.

**chapitre 12 |** Révision complète

- Configurer un modèle de tâche pour un playbook dans Red Hat Ansible Tower, puis l'utiliser pour exécuter le playbook sur les hôtes gérés.
- Rédiger un playbook qui inclut plusieurs plays, puis utiliser l'augmentation des priviléges par play pour effectuer des tâches en tant qu'utilisateurs particuliers.

## **Chapitre 4, Gestion des variables et des faits**

Rédiger des playbooks qui utilisent des variables pour simplifier la gestion du playbook, et des faits afin de référencer des informations sur les hôtes gérés.

- Créer et référencer des variables qui affectent des hôtes ou groupes d'hôtes spécifiques, le play ou l'environnement global, et décrire le fonctionnement de la priorité des variables.
- Chiffrer les variables sensibles à l'aide d'Ansible Vault et exécuter des playbooks faisant référence à des fichiers de variables chiffrées par Vault.
- Référencer des données spécifiques à des hôtes gérés particuliers à l'aide de faits Ansible.

## **Chapitre 5, Installation et mise à jour de logiciels**

Installer, gérer et s'assurer que le logiciel est à jour à l'aide de playbooks Ansible.

- S'assurer que le logiciel est installé et à jour sur les systèmes Microsoft Windows.
- Inspecter le Registre Windows et s'assurer que les clés de Registre sont correctement configurées.

## **Chapitre 6, Mise en œuvre d'un contrôle de tâche**

Gérer l'exécution des tâches à l'aide de boucles, de tests conditionnels et de gestionnaires, et les récupérer en cas d'échec des tâches.

- Rédiger des plays de manière efficace à l'aide de boucles et utiliser des conditions pour contrôler le moment où les tâches sont exécutées.
- Implémenter des tâches qui s'exécutent uniquement lorsqu'une autre tâche apporte une modification à l'hôte géré.
- Contrôler ce qui se passe lorsqu'une tâche échoue, récupérer après l'échec d'une tâche et contrôler les conditions amenant une tâche à signaler un échec.

## **Chapitre 7, Déploiement de fichiers sur des hôtes gérés**

Déployer, modifier et gérer des fichiers sur vos hôtes gérés.

- Créer, installer, modifier et supprimer des fichiers sur des hôtes gérés, et configurer des permissions de fichiers et d'autres caractéristiques.
- Déployer des fichiers personnalisés sur des hôtes gérés à l'aide de modèles Jinja2.

## **Chapitre 8, Interaction avec les utilisateurs et les domaines**

Gérer les utilisateurs locaux et de domaine, gérer les domaines Active Directory et générer une liste d'hôtes gérés sur la base de l'appartenance à un domaine pour un inventaire dynamique Red Hat Ansible Tower.

- Automatiser la création et la gestion des comptes d'utilisateurs et de groupes locaux.

- Créer un domaine Active Directory le cas échéant et configurer l'appartenance des hôtes, les utilisateurs et les groupes dans ce domaine.
- Générez un inventaire Ansible de manière dynamique dans Red Hat Ansible Tower en fonction de l'appartenance à un domaine Active Directory.

## **Chapitre 9, Automatisation des tâches d'administration Windows**

Automatiser les tâches d'administration Windows Server.

- Exécuter des commandes PowerShell arbitraires, configurer des modules PowerShell et configurer des tâches planifiées sur des hôtes gérés avec des playbooks Ansible.
- Automatiser la configuration des périphériques de stockage sur les hôtes gérés.

## **Chapitre 10, Gestion de projets volumineux**

Écrire des playbooks optimisés pour des projets plus vastes et plus complexes, et qui réutilisent le code d'automatisation existant.

- Gérer des playbooks volumineux en important ou en incluant d'autres playbooks ou tâches à partir de fichiers, de manière inconditionnelle ou sur la base d'un test conditionnel.
- Créer un rôle afin de permettre la réutilisation du code par différents projets Ansible et l'exécuter dans le cadre de l'un des plays du playbook.
- Sélectionner et récupérer des rôles à partir d'Ansible Galaxy ou d'autres sources, telles qu'un référentiel Git, et de les utiliser dans des playbooks.
- Automatiser des tâches en configurant et en exécutant les ressources de configuration de l'état souhaité de PowerShell (DSC, Desired State Configuration) à partir de votre playbook Ansible.

## **Chapitre 11, Construction de workflows Ansible Tower**

Simplifiez la gestion des tâches et lancez des tâches complexes à l'aide de Red Hat Ansible Tower.

- Décrire les utilisateurs et les équipes de Red Hat Ansible Tower, les rôles auxquels ils peuvent être affectés, ainsi que la manière de les créer et de les gérer.
- Créer un questionnaire pour que les utilisateurs puissent facilement lancer des tâches avec Red Hat Ansible Tower.
- Planifier l'exécution automatique d'un modèle de tâche dans Red Hat Ansible Tower.
- Créer une séquence de modèles de tâches à exécuter à l'aide de workflows et de modèles de tâche de workflow dans Red Hat Ansible Tower.

## ► Open Lab

# Examen général de l'atelier 1

Dans cet exercice, vous allez examiner les faits du système et vous assurer que certaines clés de Registre spécifiques sont correctement définies.

## Résultats

Vous devez pouvoir exécuter des commandes ad hoc et configurer des clés de Registre Windows.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

## Instructions

Dans cet exercice, vous allez configurer un message de connexion rempli avec certains faits du système.

- À partir d'Ansible Tower, utilisez la fonction Run Commands pour exécuter le module **setup** sur **win1.example.com**. Notez les faits qui sont disponibles.
- À partir de **workstation**, ouvrez Visual Studio Code et clonez le référentiel **comprevi**w si vous ne l'avez pas déjà fait. Le référentiel est accessible à l'adresse <https://gitlab.example.com/student/comprevi.git>.
- Ouvrez le playbook **lab1.yml**, puis effectuez les tâches de façon à ce qu'elles incluent les éléments suivants. Après la mise à jour du playbook Ansible, le message doit afficher, dans l'ordre, le nom du système, son adresse IP, la version du système d'exploitation, le nombre de processeur et la mémoire totale.
  - **legalnoticecaption** définit la légende du message de connexion.
  - **legalnoticetext** définit le texte du message de connexion.
  - **ansible\_facts["fqdn"]**
  - **ansible\_facts["ip\_addresses"][0]**
  - **ansible\_facts["distribution"]**
  - **ansible\_facts["processor\_count"]**
  - **ansible\_facts["memtotal\_mb"]**
- Enregistrez, validez et transmettez par push vos modifications à GitLab.
- Dans l'interface utilisateur Web d'Ansible Tower, modifiez **Run comprevi project** pour utiliser le playbook **lab1.yml**, puis lancez une tâche.
- Connectez-vous à **win1** en tant que **student** pour vérifier que le message de connexion s'affiche.



**Note**

Si vous avez déjà effectué une connexion à **win1**, puis une déconnexion, utilisez l'option *Sign Out* avant de vous reconnecter. La déconnexion laisse la session utilisateur ouverte, de sorte que le message de connexion ne s'affiche pas jusqu'à ce que vous vous déconnectiez et que vous vous reconnectiez à nouveau.

Vous avez maintenant terminé la révision complète.

## ► Solution

# Examen général de l'atelier 1

Dans cet exercice, vous allez examiner les faits du système et vous assurer que certaines clés de Registre spécifiques sont correctement définies.

## Résultats

Vous devez pouvoir exécuter des commandes ad hoc et configurer des clés de Registre Windows.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

## Instructions

Dans cet exercice, vous allez configurer un message de connexion rempli avec certains faits du système.

- À partir d'Ansible Tower, utilisez la fonction Run Commands pour exécuter le module **setup** sur **win1.example.com**. Notez les faits qui sont disponibles.
- À partir de **workstation**, ouvrez Visual Studio Code et clonez le référentiel **comprevi**w si vous ne l'avez pas déjà fait. Le référentiel est accessible à l'adresse <https://gitlab.example.com/student/comprevi.git>.
- Ouvrez le playbook **lab1.yml**, puis effectuez les tâches de façon à ce qu'elles incluent les éléments suivants. Après la mise à jour du playbook Ansible, le message doit afficher, dans l'ordre, le nom du système, son adresse IP, la version du système d'exploitation, le nombre de processeur et la mémoire totale.
  - **legalnoticecaption** définit la légende du message de connexion.
  - **legalnoticetext** définit le texte du message de connexion.
  - **ansible\_facts["fqdn"]**
  - **ansible\_facts["ip\_addresses"][0]**
  - **ansible\_facts["distribution"]**
  - **ansible\_facts["processor\_count"]**
  - **ansible\_facts["memtotal\_mb"]**
- Enregistrez, validez et transmettez par push vos modifications à GitLab.
- Dans l'interface utilisateur Web d'Ansible Tower, modifiez **Run comprevi project** pour utiliser le playbook **lab1.yml**, puis lancez une tâche.
- Connectez-vous à **win1** en tant que **student** pour vérifier que le message de connexion s'affiche.

**Note**

Si vous avez déjà effectué une connexion à **win1**, puis une déconnexion, utilisez l'option **Sign Out** avant de vous reconnecter. La déconnexion laisse la session utilisateur ouverte, de sorte que le message de connexion ne s'affiche pas jusqu'à ce que vous vous déconnectiez et que vous vous reconnectiez à nouveau.

1. À partir de **workstation**, accédez à l'instance Ansible Tower disponible sur <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
 

Cliquez sur **Inventories** du volet de configuration Ansible Tower, accédez à la page **Default inventory**, puis ouvrez le volet d'exécution de la commande ad hoc de l'hôte **win1.example.com**.

  - 1.1. À partir de **workstation**, double-cliquez sur l'icône de raccourci **Ansible Tower** sur votre Bureau pour accéder à votre instance Ansible Tower située sur <http://tower.example.com>.
  - 1.2. Connectez-vous à Ansible Tower en utilisant **student** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.
  - 1.3. Dans la page principale du tableau de bord, sélectionnez **Inventories** dans le volet de navigation et sélectionnez **Default inventory** dans les inventaires listés.
  - 1.4. Dans la page **Default inventory**, cliquez sur **HOSTS**, puis cochez la case en regard de l'entrée d'hôte **win1.example.com**.
  - 1.5. Après avoir sélectionné **win1.example.com**, cliquez sur **RUN COMMANDS** pour accéder au volet d'exécution de la commande ad hoc.
  - 1.6. Sur la page **RUN COMMAND**, sélectionnez le module **setup** dans la liste **MODULE**. Sélectionnez les informations d'identification de la machine **DevOps**, puis cliquez sur **LAUNCH** pour exécuter le module.
  - 1.7. Examinez la sortie pour vérifier que le module a bien été exécuté et prenez note des différents faits disponibles.
2. Lancez l'éditeur Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **compreviwe**, clonez-le sur votre instance **workstation**.
  - 2.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 2.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P**.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/compreviwe.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **compreviwe**.  
Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers, puis passez à l'étape suivante.

3. Accédez à **File** → **Open Folder**, puis sélectionnez le dossier **C:\Users\student\Documents\comprevie** pour ouvrir ce répertoire.
4. Ouvrez le playbook **lab1.yml**, puis effectuez les tâches suivantes.
  - 4.1. Modifiez la tâche **Configure login message Caption** pour configurer la légende du message de connexion. Cette opération modifie la valeur **legalnoticecaption**.

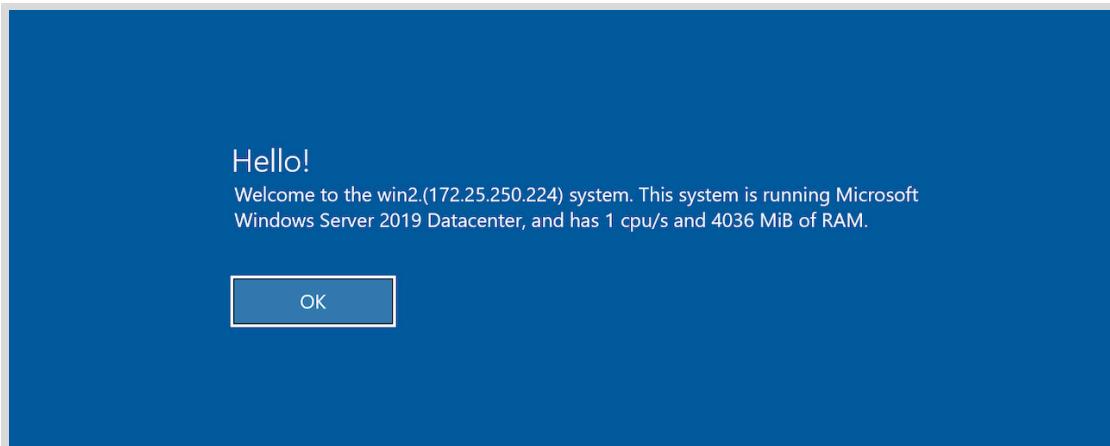
```
- name: Configure login message caption
  win_regedit:
    path: '{{ message_key }}'
    name: legalnoticecaption
    data: 'Hello!'
    type: string
```

- 4.2. Modifiez la tâche **Configure login message text**. Cette opération modifie la valeur **legalnoticetext**.

```
- name: Configure login message text
  win_regedit:
    path: '{{ message_key }}'
    name: legalnoticetext
    data: >
      'Welcome to the {{ ansible_facts['fqdn'] }}
      ({{ ansible_facts['ip_addresses'][0] }}) system.
      This system is running {{ ansible_facts['distribution'] }},
      and has {{ ansible_facts['processor_count'] }} cpu/s
      and {{ ansible_facts['memtotal_mb'] }} MiB of RAM.'
    type: string
```

5. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 5.1. Cliquez sur **File** → **Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.
  - 5.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard du fichier **lab1.yml** pour indexer les modifications.
  - 5.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 5.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
6. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur <http://tower.example.com>. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
7. Testez votre playbook à l'aide du modèle de tâche **Run comprevie project**.
  - 7.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 7.2. Cliquez sur le modèle de tâche **Run comprevie project** pour le sélectionner.
  - 7.3. Sélectionnez le playbook **lab1.yml** dans la liste **PLAYBOOK**.

- 7.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
8. Une fois toutes les tâches terminées, vérifiez que le statut de la section **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec le fichier de solution **solutions/lab1.sol** du projet **comprevieW**.
9. À partir de **workstation**, ouvrez votre client RDP et connectez-vous à **win1.example.com** en utilisant **devops** comme nom d'utilisateur avec le mot de passe **RedHat123@!**. Acceptez le certificat non sécurisé, puis vérifiez que le message de connexion s'affiche.



Cliquez sur **OK** pour continuer.

Déconnectez-vous de **win1.example.com**.

Vous avez maintenant terminé la révision complète.

## ► Open Lab

# Examen général de l'atelier 2

Au cours de cet exercice, vous allez gérer des services et du contenu de fichier à l'aide d'un modèle.

## Résultats

Vous devez pouvoir utiliser les services Microsoft Windows et utiliser un modèle pour générer un fichier sur un hôte géré.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

## Instructions

Au cours de cet exercice, vous allez désactiver des services inutiles pour réduire la surface d'attaque du système et alimenter le fichier hosts avec les noms et les adresses IP de tous les hôtes du groupe Windows.



### Note

Notez que le playbook contient deux plays. Le premier play rassemble des faits pour tous les hôtes du groupe **Windows**, de sorte que le modèle du second play puisse accéder aux variables d'hôte (**hostvars**) pour tous les membres du groupe.

- À partir de **workstation**, lancez l'éditeur Visual Studio Code et clonez le référentiel **comprevieW** si vous ne l'avez pas déjà fait. Le référentiel est accessible à l'adresse <https://gitlab.example.com/student/comprevieW.git>.
- Ouvrez le playbook **lab2.yml**, puis effectuez les tâches. La liste suivante décrit les services à désactiver.
  - Application Host Helper Service (**AppHostSvc**)
  - Service de géolocalisation (**lfsvc**)
  - Spouleur d'impression (**Spooler**)
  - Découverte SSDP (**SSDPSVC**)
- Dans le playbook, utilisez le module correct pour envoyer à l'hôte géré le fichier de modèle **hosts.j2**. Réutilisez la variable qui pointe vers le répertoire contenant le fichier **hosts**.
- Mettez à jour la seconde tâche afin d'utiliser le module pour la gestion des services. Désactivez et arrêtez les services listés dans la boucle **disabled\_services**.
- Modifiez le fichier de modèle **templates/hosts.j2**.

Ajoutez les variables d'hôte (**hostvars**) suivantes à la boucle **for** au bas du fichier. Le format doit être une liste séparée par des espaces contenant l'adresse IP, le nom de domaine complet (FQDN) et le nom d'hôte dans cet ordre.

- {{ hostvars[host]['ansible\_ip\_addresses'][0] }}
  - {{ hostvars[host]['ansible\_fqdn'] }}
  - {{ hostvars[host]['inventory\_hostname\_short'] }}
- Enregistrez, validez et transmettez par push vos modifications à GitLab.
  - Dans l'interface utilisateur Web d'Ansible Tower, modifiez le modèle **Run compreview project** pour utiliser le playbook **lab2.yml**, puis lancez une tâche.
  - Connectez-vous à **win2** en tant qu'**EXAMPLE\Administrator** avec le mot de passe **AD Administrator** pour vérifier que le fichier **C:\Windows\System32\drivers\etc\hosts** contient une entrée pour chaque hôte du **groupe Windows**, et que les services appropriés sont désactivés.
  - Dans l'interface utilisateur Web d'Ansible Tower, modifiez le modèle **Run compreview project** pour utiliser le playbook **revert\_lab2.yml**, puis lancez une tâche.

Vous avez maintenant terminé la révision complète.

## ► Solution

# Examen général de l'atelier 2

Au cours de cet exercice, vous allez gérer des services et du contenu de fichier à l'aide d'un modèle.

## Résultats

Vous devez pouvoir utiliser les services Microsoft Windows et utiliser un modèle pour générer un fichier sur un hôte géré.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

## Instructions

Au cours de cet exercice, vous allez désactiver des services inutiles pour réduire la surface d'attaque du système et alimenter le fichier hosts avec les noms et les adresses IP de tous les hôtes du groupe Windows.



### Note

Notez que le playbook contient deux plays. Le premier play rassemble des faits pour tous les hôtes du groupe **Windows**, de sorte que le modèle du second play puisse accéder aux variables d'hôte (**hostvars**) pour tous les membres du groupe.

- À partir de **workstation**, lancez l'éditeur Visual Studio Code et clonez le référentiel **comprevieW** si vous ne l'avez pas déjà fait. Le référentiel est accessible à l'adresse <https://gitlab.example.com/student/comprevieW.git>.
- Ouvrez le playbook **lab2.yml**, puis effectuez les tâches. La liste suivante décrit les services à désactiver.
  - Application Host Helper Service (**AppHostSvc**)
  - Service de géolocalisation (**lfsvc**)
  - Spouleur d'impression (**Spooler**)
  - Découverte SSDP (**SSDPSVC**)
- Dans le playbook, utilisez le module correct pour envoyer à l'hôte géré le fichier de modèle **hosts.j2**. Réutilisez la variable qui pointe vers le répertoire contenant le fichier **hosts**.
- Mettez à jour la seconde tâche afin d'utiliser le module pour la gestion des services. Désactivez et arrêtez les services listés dans la boucle **disabled\_services**.
- Modifiez le fichier de modèle **templates/hosts.j2**.

Ajoutez les variables d'hôte (**hostvars**) suivantes à la boucle **for** au bas du fichier. Le format doit être une liste séparée par des espaces contenant l'adresse IP, le nom de domaine complet (FQDN) et le nom d'hôte dans cet ordre.

- {{ hostvars[host]['ansible\_ip\_addresses'][0] }}
  - {{ hostvars[host]['ansible\_fqdn'] }}
  - {{ hostvars[host]['inventory\_hostname\_short'] }}
- Enregistrez, validez et transmettez par push vos modifications à GitLab.
  - Dans l'interface utilisateur Web d'Ansible Tower, modifiez le modèle **Run comprevie project** pour utiliser le playbook **lab2.yml**, puis lancez une tâche.
  - Connectez-vous à **win2** en tant qu'**EXAMPLE\Administrator** avec le mot de passe **AD Administrator** pour vérifier que le fichier **C:\Windows\System32\drivers\etc\hosts** contient une entrée pour chaque hôte du **groupe Windows**, et que les services appropriés sont désactivés.
  - Dans l'interface utilisateur Web d'Ansible Tower, modifiez le modèle **Run comprevie project** pour utiliser le playbook **revert\_lab2.yml**, puis lancez une tâche.

1. À partir de **workstation**, ouvrez Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **comprevie**, clonez-le sur votre instance **workstation**.

- 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.

- 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.

Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.

Utilisez l'URL de référentiel <https://gitlab.example.com/student/comprevie.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training**, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **comprevie**.

Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers.



#### Note

Exécutez cette étape uniquement si vous avez déjà cloné le référentiel **comprevie**.

Accédez à **File → Open Folder**, puis sélectionnez le dossier **C:\Users\student\Documents\comprevie** pour ouvrir le répertoire.

3. Ouvrez le playbook **lab2.yml**, puis effectuez les tâches suivantes.

- 3.1. Renseignez la liste des services de la variable **disabled\_services**.

```
disabled_services:  
  - AppHostSvc  
  - lfsvc  
  - Spooler  
  - SSDPSRV
```

- 3.2. Renseignez la tâche **Configure hosts file from template**. Cette opération remplit le fichier **C:\Windows\System32\drivers\etc\hosts**.

```
- name: Configure hosts file from template
  win_template:
    src: templates/hosts.j2
    dest: '{{ etc_dir }}\hosts'
```

- 3.3. Modifiez la tâche **Reduce server footprint**. Le paramètre **ignore\_errors** est utilisé de sorte que les services absents n'entraînent pas l'échec de l'ensemble de la tâche.

```
- name: Reduce server footprint
  win_service:
    name: "{{ item }}"
    start_mode: disabled
    state: stopped
  loop: "{{ disabled_services }}"
  ignore_errors: yes
```

4. Complétez le fichier **templates/hosts.j2**.

Ajoutez les entrées **hostvars** appropriées au corps de la boucle. Formatez l'entrée hosts comme '**IP\_ADDRESS FQDN HOSTNAME**'.

```
{% for host in groups['Windows'] %}
{{ hostvars[host]['ansible_ip_addresses'][0] }} {{ hostvars[host]
['ansible_fqdn'] }} {{ hostvars[host].inventory_hostname_short }}
{% endfor %}
```

5. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.

- 5.1. Cliquez sur **File → Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.
- 5.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard du fichier **Lab2.yml** pour indexer ce fichier.
- 5.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
- 5.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
6. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur **http://tower.example.com**. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
7. Utilisez le modèle de tâche **Run comppreview project** pour tester votre playbook.
  - 7.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 7.2. Cliquez sur le modèle de tâche **Run comppreview project** pour y accéder.

- 7.3. Sélectionnez le playbook **lab2.yml** dans la liste **PLAYBOOK**.
- 7.4. Sélectionnez **Default inventory** dans la liste **INVENTORY**.
- 7.5. Si vous avez terminé le *Chapitre 8 : Interaction avec les utilisateurs et les domaines*, sélectionnez les informations d'identification **AD Administrator** dans la liste **CREDENTIAL** et mettez à jour le champ de texte **EXTRA VARIABLES** pour qu'il corresponde à ce qui suit :

```
---  
ansible_winrm_transport: kerberos
```

- 7.6. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
8. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec les fichiers de solution **solutions/lab2.sol** et **solutions/hosts.sol** dans le référentiel Git **comprevie**.
9. Vérifiez que toutes les modifications ont été exécutées.

- 9.1. À partir de **workstation**, connectez-vous à **win2.example.com** avec le client du bureau à distance. Utilisez le nom d'utilisateur **EXAMPLE\Administrator** et le mot de passe **AD Administrator**.
- 9.2. Utilisez un éditeur de texte pour vérifier que le fichier **C:\Windows\System32\drivers\etc\hosts** contient une entrée pour chaque hôte du groupe **Windows**.

```
# localhost name resolution is handled within DNS itself.  
#      127.0.0.1      localhost  
#      ::1            localhost  
  
172.25.250.224 win2.example.com win2  
172.25.250.70 win1.example.com win1
```

- 9.3. Ouvrez l'applet du panneau de configuration **Services** et vérifiez que les services appropriés sont désactivés.
- 9.4. Déconnectez-vous de **win2.example.com** une fois le test terminé.
10. Dans l'interface utilisateur Web d'Ansible Tower, modifiez le modèle **Run comprevie project** pour utiliser le playbook **revert\_lab2.yml**, puis lancez une tâche.

Vous avez maintenant terminé la révision complète.

## ► Open Lab

# Examen général de l'atelier 3

Au cours de cet exercice, vous allez copier des fichiers et installer des logiciels.

### Résultats

Vous devez pouvoir utiliser les paquetages d'installation et utiliser les blocs dans les playbooks pour contrôler les ensembles de tâches.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

### Instructions

Dans cet exercice, vous allez installer le paquetage **7zip**.

- À partir de **workstation**, lancez Visual Studio Code et clonez le référentiel **comprevew** si vous ne l'avez pas déjà fait.
- Ouvrez le playbook **lab3.yml**, puis effectuez les tâches suivantes :
  - Ajoutez une clause initiale pour gérer les défaillances dans le playbook.
  - Utilisez le module Ansible pour transférer le fichier **7z.exe**.
  - Remplacez les deux espaces réservés du module **win\_package** par les deux paramètres qui indiquent l'ID de produit et les arguments d'installation à transmettre.
  - Ajoutez une deuxième clause dans le playbook pour indiquer à Ansible d'afficher le message de débogage et tenter de supprimer 7-Zip.
  - Ajoutez une clause finale dans le playbook pour indiquer à Ansible de toujours exécuter les tâches.
  - Utilisez le module Ansible approprié, avec un paramètre, pour demander la suppression du paquetage d'installation.
- Enregistrez, validez et transmettez par push vos modifications à GitLab.
- Dans l'interface utilisateur Web d'Ansible Tower, modifiez le modèle **Run comprevew project** pour utiliser le playbook **lab3.yml**, puis lancez une tâche.
- Connectez-vous à **win1** en tant que **student** pour vérifier que **7zip** est installé.

Vous avez maintenant terminé la révision complète.

## ► Solution

# Examen général de l'atelier 3

Au cours de cet exercice, vous allez copier des fichiers et installer des logiciels.

### Résultats

Vous devez pouvoir utiliser les paquetages d'installation et utiliser les blocs dans les playbooks pour contrôler les ensembles de tâches.

### Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

### Instructions

Dans cet exercice, vous allez installer le paquetage **7zip**.

- À partir de **workstation**, lancez Visual Studio Code et clonez le référentiel **comprevue** si vous ne l'avez pas déjà fait.
- Ouvrez le playbook **lab3.yml**, puis effectuez les tâches suivantes :
  - Ajoutez une clause initiale pour gérer les défaillances dans le playbook.
  - Utilisez le module Ansible pour transférer le fichier **7z.exe**.
  - Remplacez les deux espaces réservés du module **win\_package** par les deux paramètres qui indiquent l'ID de produit et les arguments d'installation à transmettre.
  - Ajoutez une deuxième clause dans le playbook pour indiquer à Ansible d'afficher le message de débogage et tenter de supprimer 7-Zip.
  - Ajoutez une clause finale dans le playbook pour indiquer à Ansible de toujours exécuter les tâches.
  - Utilisez le module Ansible approprié, avec un paramètre, pour demander la suppression du paquetage d'installation.
  - Enregistrez, validez et transmettez par push vos modifications à GitLab.
  - Dans l'interface utilisateur Web d'Ansible Tower, modifiez le modèle **Run comprevue project** pour utiliser le playbook **lab3.yml**, puis lancez une tâche.
  - Connectez-vous à **win1** en tant que **student** pour vérifier que **7zip** est installé.

1. À partir de **workstation**, lancez Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **comprevew**, clonez-le sur votre instance **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/comprevew.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training**, puis cliquez sur **Select Repository Location**.  
Cette opération clone le référentiel distant dans le dossier **comprevew**.  
Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers.

**Note**

Ignorez cette étape si vous avez précédemment cloné le référentiel.

3. Ouvrez le playbook **lab3.yml**, puis effectuez les tâches suivantes :
  - 3.1. Corrigez la structure de bloc de sorte que la désinstallation soit effectuée dans la section **rescue**, et que le paquetage d'installation soit toujours nettoyé.

```
- name: Handle package installation
  block:
    - name: Copy package to target host
  ...output omitted...

  rescue:
    - debug:
        msg: "Sorry, it didn't work out :("
  ...output omitted...

  always:
    - name: Clean up installation package
```
  - 3.2. Assurez-vous que la tâche **Copy package to target host** utilise le module correct.

```
- name: Copy package to target host
  win_copy:
    src: files/7z.exe
    dest: C:\Users\student\Downloads\
```
  - 3.3. Exécutez la tâche **Install package**.

```
- name: Install package
  win_package:
    path: C:\Users\student\Downloads\7z.exe
    product_id: 7-Zip
    arguments: /S
    state: present
    ignore_errors: true
```

3.4. Exécutez la tâche **clean up installation package**.

```
- name: Clean up installation package
  win_file:
    path: C:\Users\student\Downloads\7z.exe
    state: absent
```

4. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.
  - 4.1. Cliquez sur **File → Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.
  - 4.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard de **lab3.yml** pour indexer les modifications.
  - 4.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.
  - 4.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
5. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
6. Utilisez le modèle de tâche **Run comppreview project** pour tester votre playbook.
  - 6.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 6.2. Cliquez sur le modèle de tâche **Run comppreview project** pour y accéder.
  - 6.3. Sélectionnez le playbook **lab3.yml** dans la liste **PLAYBOOK**.
  - 6.4. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
7. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec le fichier **solutions/lab3.sol** dans le référentiel Git **comppreview**.
8. Vérifiez que toutes les modifications sont terminées.
  - 8.1. Connectez-vous à **win1.example.com** à l'aide du client du bureau à distance. Utilisez **devops** comme nom d'utilisateur et **RedHat123@!** comme mot de passe.
  - 8.2. Cliquez sur **Recherche Windows**, recherchez **7-zip**, puis notez que le gestionnaire de fichiers 7-Zip s'affiche dans les résultats de la recherche.

8.3. Déconnectez-vous de **win1.example.com** lorsque le test est terminé.

Vous avez maintenant terminé la révision complète.

## ► Open Lab

# Examen général de l'atelier 4

Dans cet exercice, vous allez utiliser un rôle pour configurer un service et installer un paquetage à partir du référentiel Chocolatey.

## Résultats

Vous devez pouvoir importer des rôles pour gérer des fonctions et installer des logiciels à l'aide du module **win\_chocolatey**.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

## Instructions

Au cours de cet exercice, vous allez configurer le service SNMP sur **win2** et installer un client SNMP sur **win1**. Ensuite, utilisez le client SNMP sur **win1** pour parcourir l'arborescence de la MIB sur **win2**.

- Lancez l'éditeur Visual Studio Code et clonez le référentiel **comprevieW**, si vous ne l'avez pas déjà fait.
- Ouvrez le playbook **lab4.yml**, puis remplissez les variables et effectuez les tâches du play **SNMP configured on server**.

Les valeurs ont besoin des valeurs suivantes :

### Variables et valeurs du rôle `ansible.snmp`

```
snmp_access_address
  {{ hostvars['win1.example.com']['ansible_ip_addresses'][0] }}

snmp_community
  publique

snmp_contact
  Adam Admin

snmp_location
  Arizona

snmp_monitoring_server
  {{ hostvars['win1.example.com']['ansible_ip_addresses'][0] }}

snmp_password
  Password1!
```

```
snmp_user  
admin
```

#### Variable et valeur du play de configuration du client SNMP

```
package  
snmpb
```

- Dans le play **SNMP configured on server**, utilisez l'instruction d'importation Ansible pour importer le rôle **ansible.snmp**.
- Dans le play **SNMP configured on client**, utilisez le module Ansible approprié pour installer le paquetage *chocolatey*.
- Dans la dernière tâche du play **SNMP configured on client**, utilisez le module Ansible approprié pour installer le paquetage défini par la variable **package**.
- Enregistrez, validez et transmettez par push vos modifications à GitLab.
- Dans l'interface utilisateur Web d'Ansible Tower, modifiez **Run comprevew project** pour utiliser le playbook **lab4.yml**, puis lancez une tâche.
- À partir de **workstation**, utilisez le client RDP pour vous connecter à **win1** en tant que **student**, puis utilisez l'outil **snmpb** pour parcourir MIB de **win2**.

Vous avez maintenant terminé la révision complète.

## ► Solution

# Examen général de l'atelier 4

Dans cet exercice, vous allez utiliser un rôle pour configurer un service et installer un paquetage à partir du référentiel Chocolatey.

## Résultats

Vous devez pouvoir importer des rôles pour gérer des fonctions et installer des logiciels à l'aide du module **win\_chocolatey**.

## Avant De Commencer

Ouvrez votre client RDP et connectez-vous à **workstation** Windows à l'aide de **student** comme nom d'utilisateur. Votre mot de passe unique s'affiche dans l'onglet **Online Lab** de l'interface ROL.

Cet onglet affiche également les informations de connexion pour tous vos systèmes.

Vérifiez que toutes les machines d'infrastructure sont en cours d'exécution et que vous pouvez accéder à Ansible Tower et GitLab à partir du poste de travail Windows.

## Instructions

Au cours de cet exercice, vous allez configurer le service SNMP sur **win2** et installer un client SNMP sur **win1**. Ensuite, utilisez le client SNMP sur **win1** pour parcourir l'arborescence de la MIB sur **win2**.

- Lancez l'éditeur Visual Studio Code et clonez le référentiel **comprevieW**, si vous ne l'avez pas déjà fait.
- Ouvrez le playbook **lab4.yml**, puis remplissez les variables et effectuez les tâches du play **SNMP configured on server**.

Les valeurs ont besoin des valeurs suivantes :

### Variables et valeurs du rôle `ansible.snmp`

```
snmp_access_address
  {{ hostvars['win1.example.com']['ansible_ip_addresses'][0] }}

snmp_community
  publique

snmp_contact
  Adam Admin

snmp_location
  Arizona

snmp_monitoring_server
  {{ hostvars['win1.example.com']['ansible_ip_addresses'][0] }}

snmp_password
  Password1!
```

```
snmp_user
admin
```

### Variable et valeur du play de configuration du client SNMP

package  
**snmpb**

- Dans le play **SNMP configured on server**, utilisez l'instruction d'importation Ansible pour importer le rôle **ansible.snmp**.
- Dans le play **SNMP configured on client**, utilisez le module Ansible approprié pour installer le paquetage *chocolatey*.
- Dans la dernière tâche du play **SNMP configured on client**, utilisez le module Ansible approprié pour installer le paquetage défini par la variable **package**.
- Enregistrez, validez et transmettez par push vos modifications à GitLab.
- Dans l'interface utilisateur Web d'Ansible Tower, modifiez **Run comprevue project** pour utiliser le playbook **lab4.yml**, puis lancez une tâche.
- À partir de **workstation**, utilisez le client RDP pour vous connecter à **win1** en tant que **student**, puis utilisez l'outil **snmpb** pour parcourir MIB de **win2**.

1. À partir de **workstation**, lancez Visual Studio Code. Si vous n'avez pas cloné auparavant le référentiel **comprevue**, clonez-le sur votre système **workstation**.
  - 1.1. Cliquez sur **Recherche Windows**, recherchez le **code**, puis ouvrez Visual Studio Code.
  - 1.2. Accédez à **View → Command Palette** ou appuyez sur **Ctrl+Maj+P** pour ouvrir la palette de commandes.  
Saisissez **clone**, puis sélectionnez **Git: Clone** pour cloner un référentiel.  
Utilisez l'URL de référentiel <https://gitlab.example.com/student/comprevue.git>. À l'invite, sélectionnez le dossier **Documents** du répertoire personnel de l'utilisateur **training** en tant qu'emplacement de référentiel, puis cliquez sur **Select Repository Location**. Cette opération clone le référentiel distant dans le dossier **comprevue**.  
Cliquez sur **Open** dans la fenêtre qui s'affiche après le clonage pour afficher les fichiers, puis passez à l'étape suivante.
2. Accédez au répertoire **c:\Users\student\Documents\comprevue** et cliquez sur **File → Open Folder**, pour ouvrir le dossier **C:\Users\student\Documents\comprevue**.
3. Ouvrez le playbook **lab4.yml**, puis effectuez les tâches suivantes :
  - 3.1. Renseignez les variables du rôle **ansible.snmp**, comme indiqué précédemment dans cet atelier.

```
vars:
  snmp_access_address: >-
    {{ hostvars['win1.example.com']['ansible_ip_addresses'][0] }}
  snmp_community: public
```

```
snmp_contact: Adam Admin
snmp_location: Arizona
snmp_monitoring_server: >-
    {{ hostvars['win1.example.com']['ansible_ip_addresses'][0] }}
snmp_password: Password1!
snmp_user: admin
```

**Note**

Le signe **>-** indique à Ansible de supprimer la nouvelle ligne à la fin de la chaîne.

- 3.2. Importez le rôle `ansible.snmp`.

```
...output omitted...
    snmp_user: admin

import_role:
    name: ansible.snmp
```

- 3.3. Ajoutez la variable **package** du second play, comme indiqué dans le tableau. Attribuez-lui la valeur **snmpb**.

```
vars:
    package: snmpb
```

- 3.4. Exécutez la tâche **Chocolatey is installed**. Cette tâche utilise le module `win_chocolatey` pour installer le paquetage *chocolatey*.

```
- name: Chocolatey is installed
  win_chocolatey:
    name: chocolatey
    state: present
```

- 3.5. Exécutez la tâche **Packages are installed**, à l'aide de la variable configurée précédemment.

```
- name: Packages are installed
  win_chocolatey:
    name: "{{ package }}"
    state: present
    pinned: true
```

4. Enregistrez et validez les modifications sur votre référentiel Git local, puis poussez-les vers le référentiel Git distant.

- 4.1. Cliquez sur **File → Save** ou appuyez sur **Ctrl+S** pour enregistrer le fichier.
- 4.2. Accédez au volet **Source Control**, puis cliquez sur **+** en regard de **lab4.yml** pour indexer les modifications.
- 4.3. Saisissez un court message de validation, puis appuyez sur **Ctrl+Entrée** pour valider les modifications.

- 4.4. Les icônes de statut affichées dans le coin inférieur gauche de l'éditeur indiquent qu'il y a des modifications à transmettre par push au référentiel Git distant. Cliquez sur **Synchronize Changes** pour transmettre par push vos modifications locales au référentiel distant et récupérer toutes les nouvelles modifications à partir du référentiel.
5. À partir de **workstation**, accédez à votre instance Ansible Tower disponible sur `http://tower.example.com`. Connectez-vous en tant qu'utilisateur **student** avec le mot de passe **RedHat123@!**.
6. Utilisez le modèle de tâche **Run comppreview project** pour tester votre playbook.
  - 6.1. Dans le volet de navigation, cliquez sur **Templates**.
  - 6.2. Cliquez sur le modèle de tâche **Run comppreview project**.
  - 6.3. Sélectionnez le playbook **lab4.yml** dans la liste **PLAYBOOK**.
  - 6.4. Sélectionnez **Default inventory** dans la liste **INVENTORY**.
  - 6.5. Si vous avez terminé le *Chapitre 8 : Interaction avec les utilisateurs et les domaines*, sélectionnez les informations d'identification **AD Administrator** dans la liste **CREDENTIAL** et mettez à jour le champ de texte **EXTRA VARIABLES** pour qu'il corresponde à ce qui suit :

```
---  
ansible_winrm_transport: kerberos
```

- 6.6. Cliquez sur **SAVE** pour mettre à jour le modèle, puis cliquez sur **LAUNCH** pour exécuter la tâche.
7. Une fois toutes les tâches terminées, vérifiez que le statut du volet **DETAILS** affiche **Successful**. En cas d'échec de la tâche, comparez votre playbook avec le fichier de solution **solutions/lab4.sol** dans le référentiel Git **comppreview**.
8. À partir de **workstation**, utilisez le client du bureau à distance pour vous connecter à **win1.example.com**. Utilisez **devops** comme utilisateur et **RedHat123@!** comme mot de passe. Utilisez l'outil **snmpb** pour parcourir la MIB de **win2**.
  - 8.1. L'outil **snmpb** ne crée pas d'entrée de menu ; Accédez à **C:\Program Files (x86)\SnmpB** à l'aide de l'Explorateur Windows, puis démarrez l'outil **snmpb**.
  - 8.2. Cliquez sur **Options → Manage Agent Profiles...**, puis cliquez avec le bouton droit dans le volet gauche et sélectionnez **New agent profile**. Définissez le nom sur **win2** et **Agent Address/Name** sur **win2.example.com**. Vérifiez toutes les versions de SNMP, puis cliquez sur **OK**.
  - 8.3. À partir de la **liste Remote SNMP Agent**, faites basculer l'hôte de **localhost** vers **win2**.  
Sélectionnez **SNMPv2c** comme version SNMP.
  - 8.4. Cliquez avec le bouton droit sur le dossier de niveau supérieur **MIB Tree**, puis sélectionnez **Walk**. Les résultats de la requête s'affichent dans le volet droit.  
Faites défiler jusqu'au début des résultats pour voir les valeurs définies dans le playbook.

```
-----SNMP query started-----
1: sysDescr.0 Hardware: Intel64 Family 6 Model 85 Stepping 4 AT/AT COMPATIBLE -
   Software: Windows Version 6.3 (Build 17763 Multiprocessor Free)
2: sysObjectID.0 enterprises.311.1.1.3.1.2
3: sysUpTime.0 0:06:43.47
4: sysContact.0 Adam Admin
5: sysName.0 win2
6: sysLocation.0 Arizona
7: sysServices.0 79
...output omitted...
```



### Note

Si la requête SNMP affiche un message **Timeout**, un redémarrage de l'hôte **win2.example.com** peut s'avérer nécessaire.

8.5. Déconnectez-vous de **win1.example.com** lorsque le test est terminé.

Vous avez maintenant terminé la révision complète.



## annexe A

# Sujets supplémentaires

### Objectif

Enquêter sur des sujets supplémentaires non inclus dans le cours officiel.

# Gestion des rôles utilisateur Red Hat Ansible Tower

---

## Résultats

Après avoir terminé cette section, vous serez en mesure d'appliquer des rôles utilisateur pour contrôler l'accès aux informations d'identification, aux projets et aux modèles de tâche dans Red Hat Ansible Tower.

## Gestion des rôles d'inventaire

Les utilisateurs et les équipes peuvent se voir attribuer la possibilité de lire, d'utiliser ou de gérer un inventaire en attribuant des rôles appropriés pour cet inventaire. Dans certains cas, au lieu d'affecter directement un rôle à l'utilisateur ou à l'équipe, il est possible d'hériter d'un rôle qui octroie indirectement des permissions d'utilisation d'un inventaire.

La liste suivante répertorie les rôles disponibles pour un inventaire :

### Tâches

Le rôle **Admin** d'inventaire accorde aux utilisateurs des permissions complètes sur un inventaire. Ces permissions incluent la suppression et la modification de l'inventaire. Ce rôle octroie, en outre, les autorisations associées aux rôles d'inventaire **Use**, **Ad Hoc** et **Update**.

### Use

Le rôle d'inventaire **Use** permet aux utilisateurs d'utiliser un inventaire dans une ressource de modèle de tâche. Cela permet de contrôler l'inventaire utilisé pour lancer des tâches à l'aide du playbook du modèle de tâche.

### Ad Hoc

Le rôle d'inventaire **Ad Hoc** permet aux utilisateurs d'utiliser l'inventaire pour exécuter des commandes ad hoc. L'utilisation d'Ansible Tower pour l'exécution de commandes ad hoc est décrite en détails dans le Guide de l'utilisateur d'Ansible Tower.

### Update

Le rôle d'inventaire **Update** permet aux utilisateurs de mettre à jour un inventaire dynamique à partir de sa source de données externe.

### Read

Le rôle d'inventaire **Read** donne aux utilisateurs la possibilité d'afficher le contenu d'un inventaire.

## Configuration de l'accès pour les inventaires

Lorsqu'un inventaire est créé pour la première fois, il est uniquement accessible aux utilisateurs qui ont les rôles **Admin**, **Inventory Admin** ou **Auditor** pour l'organisation à laquelle l'inventaire appartient. Tous les autres accès doivent être spécifiquement configurés.

Pour ce faire, affectez les rôles appropriés aux utilisateurs et aux équipes, comme indiqué ci-dessus. L'inventaire doit être créé et enregistré pour que les rôles puissent être affectés aux utilisateurs et aux équipes.

Les rôles sont affectés par le biais de la section **PERMISSIONS** de l'écran de l'éditeur de l'inventaire (accessible par le biais de l'icône de crayon en regard de son nom). La procédure

La liste suivante détaille les étapes à suivre pour affecter des rôles d'utilisateur et d'équipe à un inventaire :

1. Connectez-vous en tant qu'utilisateur avec le rôle **Admin** sur l'organisation dans laquelle l'inventaire a été créé.
2. Cliquez sur **Inventories** dans le volet de navigation pour afficher la liste des inventaires de l'organisation.
3. Cliquez sur l'icône de crayon de l'inventaire pour accéder à l'écran de l'éditeur de l'inventaire.
4. Dans l'écran de l'éditeur de l'inventaire, cliquez sur le bouton **PERMISSIONS** pour accéder à l'éditeur des autorisations.  
Une liste s'affiche, indiquant les utilisateurs et les équipes, avec les rôles qui leur sont affectés. Si un **X** apparaît en regard d'un rôle, cliquez dessus pour retirer ce rôle à l'utilisateur.
5. Cliquez sur le bouton **+** pour ajouter des autorisations.
6. Dans l'écran de sélection **ADD USERS / TEAMS**, cliquez sur **USERS** ou **TEAMS**. Activez les cases à cocher en regard des utilisateurs ou des équipes auxquels vous souhaitez affecter de nouveaux rôles.
7. Sous **Please assign roles to the selected users/teams**, cliquez sur la liste **SELECT ROLES**. Sélectionnez le rôle d'inventaire souhaité pour chaque utilisateur ou équipe.  
Cliquez sur **KEY** pour obtenir la liste des rôles d'inventaire et de leurs définitions.
8. Cliquez sur **SAVE** pour finaliser les nouveaux rôles.

## Gestion des rôles d'informations d'identification de machine

Les informations d'identification de machine (informations d'identification qui ne sont pas affectées à une organisation) sont uniquement accessibles par leurs créateurs ou par les utilisateurs du type **System Administrator** ou **System Auditor**. Les autres utilisateurs ne peuvent pas se voir attribuer de rôles sur des informations d'identification de machine privées.

Pour affecter des rôles à des informations d'identification de machine, ces informations doivent être affectées à une organisation Ansible Tower. Ensuite, les utilisateurs et les équipes de cette organisation peuvent partager ces informations d'identification par le biais d'affectations de rôles.

La liste suivante répertorie les rôles d'informations d'identification disponibles.

### Tâches

Le rôle **Admin** accorde aux utilisateurs des autorisations complètes pour des informations d'identification de machine. Ces autorisations comprennent la suppression et la modification des informations d'identification, ainsi que la possibilité d'utiliser les informations d'identification dans un modèle de tâche.

### Use

Le rôle **Use** octroie aux utilisateurs la possibilité d'utiliser des informations d'identification de machine dans un modèle de tâche.

### Read

Le rôle **Read** octroie aux utilisateurs la possibilité de consulter les détails des informations d'identification de machine. Cela ne leur permet toutefois pas de déchiffrer les secrets qui appartiennent à ces informations d'identification via l'interface Web.

## Configuration de l'accès pour les informations d'identification de machine

Lorsque les informations d'identification de machine affectées à une organisation sont créées, seuls peuvent y accéder le propriétaire et les utilisateurs disposant du rôle **Admin** ou **Auditor** dans l'organisation dans laquelle ces informations d'identification ont été créées. Tout accès supplémentaire souhaité doit être spécifiquement configuré.

Les rôles supplémentaires ne peuvent pas être affectés tant qu'elles n'ont pas été enregistrées d'abord, après quoi ces rôles peuvent être définis en modifiant les informations d'identification de machine.

Les rôles sont attribués dans la section **PERMISSIONS** de l'écran de l'éditeur des informations d'identification de machine.

La procédure suivante décrit les étapes à suivre pour accorder des autorisations aux informations d'identification de machine appartenant à une organisation après leur création.

1. Connectez-vous en tant qu'utilisateur avec le rôle **Admin** dans l'organisation dans laquelle les informations d'identification de machine ont été créées.
2. Cliquez sur **Credentials** pour accéder à l'interface de gestion des informations d'identification.
3. Cliquez sur le nom des informations d'identification de machine à modifier, afin d'accéder à l'écran de l'éditeur des informations d'identification.
4. Sur l'écran de l'éditeur des informations d'identification, cliquez sur le bouton **PERMISSIONS** afin d'accéder à l'éditeur d'autorisations.
5. Cliquez sur le bouton **+** pour ajouter des autorisations.
6. Dans l'écran de sélection des utilisateurs et des équipes, cliquez sur **USERS** ou sur **TEAMS**, puis cochez les cases relatives aux utilisateurs ou équipes auxquels accorder des rôles.
7. Sous **Please assign roles for the selected users/teams**, cliquez sur **KEY** pour afficher la liste des rôles d'informations d'identification et les définitions associées.
8. Cliquez sur la liste **SELECT ROLES** et sélectionnez le rôle d'informations d'identification souhaité pour chaque utilisateur ou équipe.
9. Cliquez sur **SAVE** pour finaliser les modifications apportées aux autorisations.



### Important

Vous pouvez également ajouter des autorisations pour des informations d'identification au moyen des écrans de gestion des utilisateurs ou des équipes.

## Gestion des rôles d'informations d'identification SCM

À l'instar des informations d'identification de machine, les informations d'identification SCM privées ne peuvent être utilisées que par leurs créateurs et par les utilisateurs **System Administrator** et **System Auditor**. Les informations d'identification SCM attribuées à une organisation peuvent être partagées avec d'autres utilisateurs en attribuant à ces derniers ou aux équipes les rôles appropriés pour ces informations d'identification.

Vous trouverez, ci-dessous, la liste des rôles disponibles pour permettre aux utilisateurs d'accéder aux informations d'identification SCM :

**annexe A |** Sujets supplémentaires**Admin**

Le rôle **Admin** vous accorde des autorisations complètes pour des informations d'identification SCM. Ces autorisations incluent la suppression et la modification des informations d'identification. Ce rôle accorde également à l'utilisateur les autorisations associées aux rôles **Use** et **Read** des informations d'identification.

**Use**

Le rôle **Use** vous donne la possibilité d'associer des informations d'identification SCM à un projet. Ce rôle vous accorde également les autorisations associées au rôle **Use** des informations d'identification.

Le rôle **Use** ne contrôle pas si un utilisateur peut lui-même utiliser les informations d'identification SCM pour mettre à jour un projet, mais uniquement s'il peut attribuer ces informations d'identification SCM afin qu'elles puissent ensuite être utilisées par une personne disposant du rôle **Update** pour un projet.

Par exemple, si vous associez des informations d'identification SCM à un projet, tout utilisateur disposant du rôle **Update** sur le projet peut utiliser les informations associées sans avoir le rôle **Use** sur ces informations d'identification.

**Read**

Le rôle **Read** vous donne la possibilité de consulter les détails des informations d'identification SCM.

## Configuration de l'accès pour les informations d'identification SCM

Lorsque vous créez des informations d'identification SCM, seuls les utilisateurs disposant du rôle **Admin** ou **Auditor** dans l'organisation à laquelle appartiennent ces informations d'identification peuvent y accéder. Vous devez configurer explicitement tout accès supplémentaire pour les autres utilisateurs.

L'attribution de rôles d'informations d'identification SCM aux utilisateurs ou aux équipes détermine les personnes disposant d'autorisations sur ces informations d'identification. Vous ne pouvez pas affecter ces autorisations la première fois que vous créez les informations d'identification SCM. Vous pouvez gérer les autorisations en modifiant les informations d'identification après leur création.

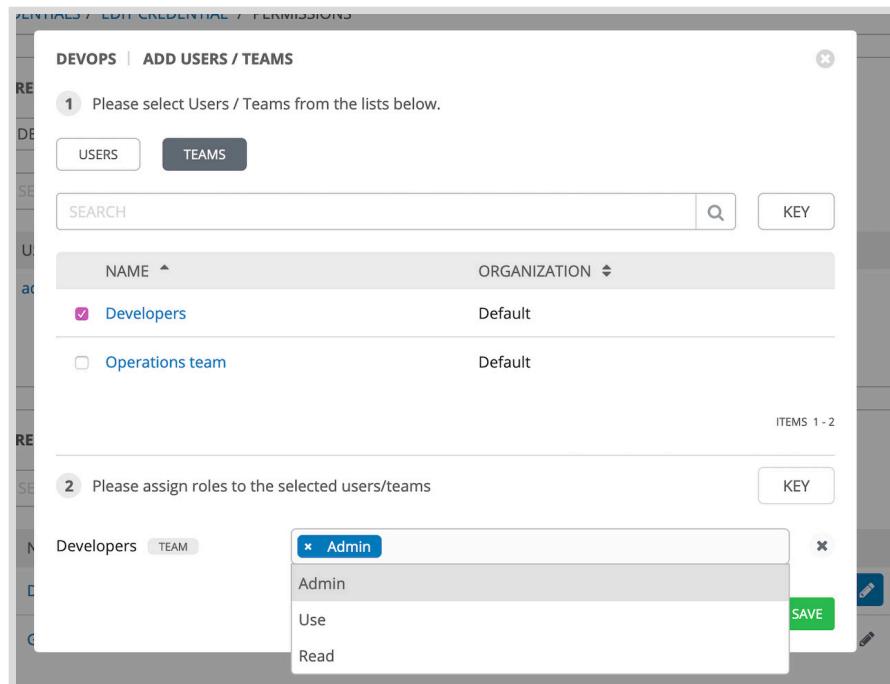
Les rôles sont attribués dans la section **PERMISSIONS** de l'écran de l'éditeur des informations d'identification. La liste suivante décrit les étapes à suivre pour accorder des autorisations aux informations d'identification SCM appartenant à une organisation.

1. Connectez-vous en tant qu'utilisateur ayant le rôle **Admin** dans l'organisation à laquelle appartiennent les informations d'identification SCM.
2. Cliquez sur **Credentials** dans le volet de navigation pour accéder à l'interface de gestion des informations d'identification.
3. Sélectionnez les informations d'identification SCM pour les modifier.
4. Sélectionnez **PERMISSIONS** pour gérer les autorisations relatives aux informations d'identification.
5. Cliquez sur **+** pour ajouter une nouvelle autorisation.
6. Dans l'écran de sélection de l'utilisateur et de l'équipe, sélectionnez soit une équipe soit un utilisateur pour lequel vous souhaitez créer une autorisation.

**annexe A |** Sujets supplémentaires

7. Cliquez sur **KEY** pour afficher la liste des rôles d'informations d'identification et les définitions associées.

Sélectionnez un rôle dans la liste **SELECT ROLES**, puis choisissez le rôle souhaité pour l'utilisateur ou l'équipe.



**Figure A.1: Gestion des rôles d'informations d'identification dans Ansible Tower**



### Important

Vous pouvez également gérer des autorisations pour des informations d'identification SCM au moyen des écrans de gestion des utilisateurs ou des équipes disponibles dans l'interface **Settings** d'Ansible Tower.

## Gestion des rôles de projet

Vous pouvez affecter des rôles de projet aux utilisateurs ou à l'équipe à laquelle ils appartiennent. Par exemple, si un utilisateur requiert des autorisations pour un projet spécifique, vous pouvez configurer l'autorisation du projet pour cet utilisateur de manière individuelle ou pour son équipe.

Voici la liste des rôles de projet disponibles :

### Tâches

Le rôle **Admin** vous octroie un accès complet au projet. Grâce à ce rôle, les utilisateurs peuvent supprimer le projet ou modifier ses propriétés, y compris ses autorisations. En outre, ce rôle vous octroie les rôles **Use**, **Update** et **Read** qui sont abordés plus loin dans cette section.

### Use

Le rôle **Use** vous donne la possibilité d'utiliser un projet avec un modèle de tâche. Ce rôle vous accorde également les autorisations associées au rôle **Read** du projet.

#### Update

Le rôle **Update** vous donne la possibilité de mettre à jour ou de planifier manuellement une mise à jour des supports du projet à partir de sa source SCM. Ce rôle vous accorde également les autorisations associées au rôle **Read** du projet.

#### Read

Le rôle **Read** vous offre également la possibilité d'afficher les détails, les autorisations et les notifications associés au projet.

## Configuration de l'accès pour les projets

Une fois le projet créé, il est accessible uniquement aux utilisateurs ayant le rôle **Admin** ou **Auditor** dans l'organisation à laquelle il appartient.

Vous devez configurer explicitement tout autre accès pour les autres utilisateurs. Vous ne pouvez pas affecter de rôles la première fois que vous définissez le projet ; cela n'est possible qu'en modifiant le projet après sa création.

Vous pouvez passer en revue les rôles en accédant à la section **PERMISSIONS** du projet. La procédure suivante décrit les étapes à suivre pour affecter des rôles pour un projet :

1. Veillez à vous connecter en tant qu'utilisateur avec le rôle **Admin** pour le projet.
2. Cliquez sur **Projects** dans le volet de navigation pour afficher la liste des projets.  
Cliquez sur l'icône du crayon correspondant à la ligne du projet que vous souhaitez mettre à jour.
3. Dans l'écran de l'éditeur du projet, sélectionnez **PERMISSIONS** pour gérer les autorisations du projet.
4. Cliquez sur **+** pour créer une autorisation.
5. Dans l'écran de sélection de l'utilisateur et de l'équipe, sélectionnez soit une équipe soit un utilisateur.  
Sélectionnez **KEY** pour afficher la liste des rôles de projet et leur définition.
6. Cliquez sur la liste **SELECT ROLES** et sélectionnez le rôle de projet souhaité pour chaque utilisateur ou équipe.

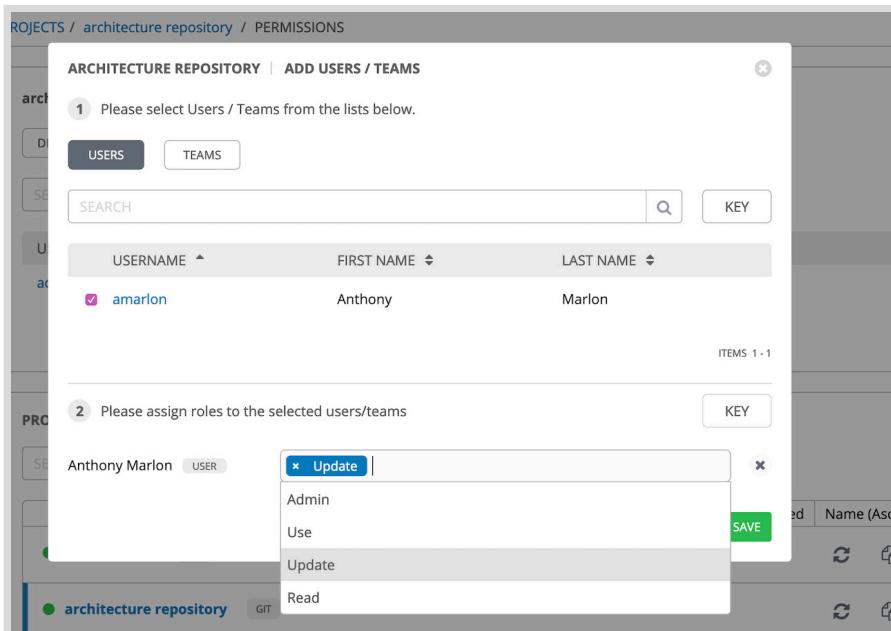


Figure A.2: Gestion des rôles de projet

**Note**

Vous pouvez également gérer des rôles pour les projets à partir des écrans de gestion des utilisateurs ou des équipes.

## Gestion des rôles de modèle de tâche

Ansible Tower utilise trois rôles pour contrôler les personnes autorisées à accéder à des modèles de tâche.

### Tâches

Le rôle **Admin** vous autorise à supprimer un modèle de tâche ou à en modifier les propriétés, y compris les autorisations associées. Ce rôle vous accorde également les autorisations associées aux rôles **Execute** et **Read** du modèle de tâche.

### Execute

Le rôle **Execute** vous autorise à exécuter une tâche à l'aide du modèle de tâche. Il vous autorise également à planifier une tâche à partir d'un modèle de tâche, mais il vous accorde aussi les autorisations associées au rôle **Read** du modèle de tâche.

**Important**

Pour pouvoir exécuter une tâche à partir d'un modèle de tâche, vous n'avez besoin que du rôle **Execute** de ce modèle ; vous n'avez besoin du rôle **Use** pour aucune des ressources associées au modèle de tâche.

### Read

Le rôle **Read** vous accorde un accès en lecture seule pour afficher les propriétés d'un modèle de tâche. Il autorise également l'affichage d'autres informations relatives au modèle de tâche, telles que la liste des tâches exécutées à l'aide du modèle de tâche, ainsi que les autorisations et notifications associées.

## Configuration de l'accès pour les modèles de tâche

Lorsque vous créez un modèle de tâche, vous seul pouvez y accéder ou bien les utilisateurs disposant du rôle **Admin** ou **Auditor** dans l'organisation à laquelle appartient le projet. Vous devez configurer explicitement tout accès supplémentaire.

L'attribution, aux utilisateurs ou aux équipes, des rôles de modèles de tâche décrits précédemment détermine les autorisations attribuées sur un modèle de tâche. Vous ne pouvez pas affecter ces autorisations lorsque vous créez le modèle de tâche ; vous pouvez configurer ces options en modifiant le modèle de tâche après sa création.

Les rôles sont disponibles dans la section **PERMISSIONS** de l'écran de l'éditeur des modèles de tâche. La procédure suivante décrit les étapes nécessaires pour attribuer des autorisations à un modèle de tâche après sa création.

1. Connectez-vous en tant qu'utilisateur avec le rôle **Admin** dans l'organisation à laquelle appartient le modèle de tâche ou en tant qu'utilisateur l'ayant créé.
2. Cliquez sur **TEMPLATES** dans le volet de navigation pour afficher la liste des modèles. Cliquez sur l'icône du crayon correspondant au modèle de tâche à modifier.
3. Sélectionnez **PERMISSIONS** pour gérer les autorisations relatives au modèle de tâche. Cliquez sur **+** pour ajouter des autorisations.
4. Dans l'écran de sélection de l'utilisateur et de l'équipe, sélectionnez un utilisateur ou une équipe qui requiert des autorisations pour le modèle de tâche.
5. Cliquez sur **KEY** pour afficher la liste des rôles de modèle de tâche et les définitions associées.
6. Sélectionnez le rôle de l'utilisateur ou de l'équipe dans la liste **SELECT ROLES**.

USER	ROLE	TEAM ROLES
admin	<b>ADMIN</b> <b>SYSTEM ADMINISTRATOR</b>	
amarlon	<b>EXECUTE</b>	

Figure A.3: Gestion des rôles pour les modèles de tâche



### Références

#### Guide de l'utilisateur d'Ansible Tower

<http://docs.ansible.com/ansible-tower/latest/html/userguide/index.html>



## annexe B

# Mentions légales

## Licence du rôle arillso.chocolatey

---

Le rôle **arillso.chocolatey** d'Ansible Galaxy (à l'adresse <https://galaxy.ansible.com/arillso/chocolatey>) a été utilisé dans un ou plusieurs exercices de ce cours. Le support d'origine de ce projet est disponible sur <https://github.com/arillso/ansible.chocolatey> sous la licence MIT suivante :

Copyright (c) 2019 Arillso

Par la présente, la permission est accordée gratuitement à toute personne obtenant une copie de ce logiciel et des fichiers de documentation associés (le « Logiciel »), d'utiliser le Logiciel sans restriction, même sans limitation des droits d'utiliser, de copier, de modifier, de fusionner, de publier, de distribuer, de concéder en sous-licence et/ou de vendre des copies du Logiciel et de permettre aux personnes auxquelles le Logiciel est fourni de le faire, sous réserve des conditions suivantes :

L'avis de copyright ci-dessus et cet avis d'autorisation doivent être inclus dans toutes les copies ou parties substantielles du Logiciel.

LE LOGICIEL EST FOURNI « EN L'ÉTAT », SANS GARANTIE D'AUCUNE SORTE, EXPRESSE OU IMPLICITE, Y COMPRIS, MAIS SANS S'Y LIMITER, LES GARANTIES DE QUALITÉ MARCHANDE, D'ADÉQUATION À UN USAGE PARTICULIER ET DE NON-CONTREFAÇON. EN AUCUN CAS, LES AUTEURS OU LES DÉTENTEURS DE COPYRIGHT NE POURRONT ÊTRE TENUS RESPONSABLES D'UNE RÉCLAMATION, DE DOMMAGES OU D'AUTRES RESPONSABILITÉS, QUE CE SOIT DANS LE CADRE D'UN CONTRAT, D'UN DÉLIT OU AUTRE, DÉCOULANT DE OU EN RELATION AVEC LE LOGICIEL OU L'UTILISATION OU AUTRES MANIPULATIONS DE CE LOGICIEL.