Red Hat Enterprise Linux 9

Automatiser l'administration du système en utilisant les rôles système RHEL

Configuration cohérente et reproductible des déploiements RHEL sur plusieurs hôtes à l'aide des playbooks de Red Hat Ansible Automation Platform

Last Updated: 2024-02-18

Red Hat Enterprise Linux 9 Automatiser l'administration du système en utilisant les rôles système RHEL

Configuration cohérente et reproductible des déploiements RHEL sur plusieurs hôtes à l'aide des playbooks de Red Hat Ansible Automation Platform

Notice légale

Résumé

Les rôles système Red Hat Enterprise Linux (RHEL) sont une collection de rôles, de modules et de playbooks Ansible qui permettent d'automatiser l'administration cohérente et reproductible des systèmes RHEL. Avec les rôles système RHEL, vous pouvez gérer efficacement de vastes inventaires de systèmes en exécutant des playbooks de configuration à partir d'un seul système.

Table des matières

RENDRE L'OPEN SOURCE PLUS INCLUSIF

Red Hat s'engage à remplacer les termes problématiques dans son code, sa documentation et ses propriétés Web. Nous commençons par ces quatre termes : master, slave, blacklist et whitelist. En raison de l'ampleur de cette entreprise, ces changements seront mis en œuvre progressivement au cours de plusieurs versions à venir. Pour plus de détails, voir le message de notre directeur technique Chris Wright.

FOURNIR UN RETOUR D'INFORMATION SUR LA DOCUMENTATION DE RED HAT

Nous apprécions vos commentaires sur notre documentation. Faites-nous savoir comment nous pouvons l'améliorer.

Soumettre des commentaires sur des passages spécifiques

1. Consultez la documentation au format **Multi-page HTML** et assurez-vous que le bouton **Feedback** apparaît dans le coin supérieur droit après le chargement complet de la page.
2. Utilisez votre curseur pour mettre en évidence la partie du texte que vous souhaitez commenter.
3. Cliquez sur le bouton **Add Feedback** qui apparaît près du texte en surbrillance.
4. Ajoutez vos commentaires et cliquez sur **Submit**.

Soumettre des commentaires via Bugzilla (compte requis)

1. Connectez-vous au site Web de Bugzilla.
2. Sélectionnez la version correcte dans le menu **Version**.
3. Saisissez un titre descriptif dans le champ **Summary**.
4. Saisissez votre suggestion d'amélioration dans le champ **Description**. Incluez des liens vers les parties pertinentes de la documentation.
5. Cliquez sur **Submit Bug**.

CHAPITRE 1. INTRODUCTION AUX RÔLES DU SYSTÈME RHEL

En utilisant RHEL System Roles, vous pouvez gérer à distance les configurations système de plusieurs systèmes RHEL dans les principales versions de RHEL. RHEL System Roles est un ensemble de rôles et de modules Ansible. Pour l'utiliser afin de configurer des systèmes, vous devez utiliser les composants suivants :

Nœud de contrôle

Un nœud de contrôle est le système à partir duquel vous exécutez les commandes et les playbooks Ansible. Votre nœud de contrôle peut être une Ansible Automation Platform, Red Hat Satellite ou un hôte RHEL 9, 8 ou 7. Pour plus d'informations, voir Préparation d'un nœud de contrôle sur RHEL 9 .

Nœud géré

Les nœuds gérés sont les serveurs et les périphériques réseau que vous gérez avec Ansible. Les nœuds gérés sont aussi parfois appelés hôtes. Il n'est pas nécessaire d'installer Ansible sur les nœuds gérés. Pour plus d'informations, voir Préparation d'un nœud géré .

Playbook Ansible

Dans un cahier de jeu, vous définissez la configuration que vous souhaitez obtenir sur vos nœuds gérés ou un ensemble d'étapes à exécuter par le système sur le nœud géré. Les playbooks sont le langage de configuration, de déploiement et d'orchestration d'Ansible.

Inventaire

Dans un fichier d'inventaire, vous dressez la liste des nœuds gérés et spécifiez des informations telles que l'adresse IP de chaque nœud géré. Dans un inventaire, vous pouvez également organiser les nœuds gérés, en créant et en imbriquant des groupes pour faciliter la mise à l'échelle. Un fichier d'inventaire est aussi parfois appelé fichier d'hôte.

Sur Red Hat Enterprise Linux 9, vous pouvez utiliser les rôles suivants fournis par le paquetage **rhel-system-roles**, qui est disponible dans le référentiel **AppStream**:

Nom du rôle	Description du rôle	Titre du chapitre
certificate	Délivrance et renouvellement du certificat	Demande de certificats à l'aide des rôles système RHEL
cockpit	Console web	Installation et configuration de la console web avec le cockpit RHEL System Role
crypto_policies	Politiques cryptographiques à l'échelle du système	Définition d'une politique cryptographique personnalisée pour l'ensemble des systèmes
firewall	Firewalld	Configuration de firewalld à l'aide des rôles système
ha_cluster	Cluster HA	Configuration d'un cluster à haute disponibilité à l'aide des rôles système
kdump	Fiches du noyau	Configuration de kdump à l'aide des rôles système RHEL

Nom du rôle	Description du rôle	Titre du chapitre
kernel_settings	Paramètres du noyau	Utiliser les rôles Ansible pour configurer de façon permanente les paramètres du noyau
logging	Enregistrement	Utilisation du rôle de système de journalisation
metrics	Métriques (PCP)	Surveillance des performances à l'aide des rôles système RHEL
network	Mise en réseau	Utilisation du rôle de système RHEL de réseau pour gérer les connexions InfiniBand
nbde_client	Client Network Bound Disk Encryption	Utilisation des rôles système nbde_client et nbde_server
nbde_server	Serveur de cryptage de disque lié au réseau	Utilisation des rôles système nbde_client et nbde_server
postfix	Postfixe	Variables du rôle postfixe dans Rôles du système
selinux	SELinux	Configuration de SELinux à l'aide des rôles système
ssh	Client SSH	Configuration de la communication sécurisée avec les rôles du système ssh
sshd	Serveur SSH	Configuration de la communication sécurisée avec les rôles du système ssh
storage	Stockage	Gestion du stockage local à l'aide des rôles système RHEL
tlog	Enregistrement de la session du terminal	Configuration d'un système pour l'enregistrement de sessions à l'aide du rôle de système RHEL tlog
timesync	Synchronisation du temps	Configuration de la synchronisation temporelle à l'aide des rôles système RHEL

Nom du rôle	Description du rôle	Titre du chapitre
vpn	VPN	Configurer des connexions VPN avec IPsec en utilisant le rôle système RHEL vpn

Ressources supplémentaires

- Rôles du système Red Hat Enterprise Linux (RHEL)
- **/usr/share/doc/rhel-system-roles/** fournie par le paquet **rhel-system-roles**

CHAPITRE 2. PRÉPARATION D'UN NŒUD DE CONTRÔLE ET DE NŒUDS GÉRÉS À L'UTILISATION DES RÔLES SYSTÈME RHEL

Avant de pouvoir utiliser des rôles système RHEL individuels pour gérer des services et des paramètres, vous devez préparer le nœud de contrôle et les nœuds gérés.

2.1. PRÉPARATION D'UN NŒUD DE CONTRÔLE SUR RHEL 9

Avant d'utiliser RHEL System Roles, vous devez configurer un nœud de contrôle. Ce système configure ensuite les hôtes gérés à partir de l'inventaire conformément aux playbooks.

Conditions préalables

- RHEL 8.6 ou une version ultérieure est installée. Pour plus d'informations sur l'installation de RHEL, voir Effectuer une installation standard de RHEL 9 .
- Le système est enregistré sur le portail client.
- Un abonnement **Red Hat Enterprise Linux Server** est attaché au système.
- S'il est disponible dans votre compte Portail Client, un abonnement **Ansible Automation Platform** est attaché au système.

Procédure

1. Installez le paquetage **rhel-system-roles**:

```
[root@control-node]# dnf install rhel-system-roles
```

   Cette commande installe le paquet **ansible-core** en tant que dépendance.

NOTE

Dans RHEL 8.5 et les versions antérieures, les packages Ansible étaient fournis via Ansible Engine au lieu d'Ansible Core, et avec un niveau de support différent. N'utilisez pas Ansible Engine car les packages peuvent ne pas être compatibles avec le contenu d'automatisation Ansible dans RHEL 8.6 et les versions ultérieures. Pour plus d'informations, voir Étendue de la prise en charge du package Ansible Core inclus dans les référentiels AppStream RHEL 9 et RHEL 8.6 et versions ultérieures.

2. Créez un utilisateur nommé **ansible** pour gérer et exécuter les playbooks :

```
[root@control-node]# useradd ansible
```

3. Passez à l'utilisateur nouvellement créé **ansible**:

```
[root@control-node]# su - ansible
```

   Effectuez le reste de la procédure en tant qu'utilisateur.

4. Créez une clé publique et une clé privée SSH :

```
[ansible@control-node]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa): <password>
...
```

Utilisez l'emplacement par défaut proposé pour le fichier clé.

5. Facultatif : Pour éviter qu'Ansible ne vous demande le mot de passe de la clé SSH à chaque fois que vous établissez une connexion, configurez un agent SSH.

6. Créez le fichier **~/.ansible.cfg** avec le contenu suivant :

```
[defaults]
inventory = /home/ansible/inventory
remote_user = ansible

[privilege_escalation]
become = True
become_method = sudo
become_user = root
become_ask_pass = True
```

NOTE

Les paramètres du fichier **~/.ansible.cfg** ont une priorité plus élevée et remplacent les paramètres du fichier global **/etc/ansible/ansible.cfg**.

Avec ces paramètres, Ansible effectue les actions suivantes :

- Gère les hôtes dans le fichier d'inventaire spécifié.
- Utilise le compte défini dans le paramètre **remote_user** lorsqu'il établit des connexions SSH avec les nœuds gérés.
- Utilise l'utilitaire **sudo** pour exécuter des tâches sur les nœuds gérés en tant qu'utilisateur **root**.
- Demande le mot de passe root de l'utilisateur distant à chaque fois que vous appliquez un playbook. Ceci est recommandé pour des raisons de sécurité.

7. Créez un fichier **~/inventory** au format INI ou YAML qui répertorie les noms d'hôtes gérés. Vous pouvez également définir des groupes d'hôtes dans le fichier d'inventaire. Par exemple, voici un fichier d'inventaire au format INI avec trois hôtes et un groupe d'hôtes nommé **US**:

```
managed-node-01.example.com

[US]
managed-node-02.example.com ansible_host=192.0.2.100
managed-node-03.example.com
```

Notez que le nœud de contrôle doit être en mesure de résoudre les noms d'hôte. Si le serveur DNS ne peut pas résoudre certains noms d'hôtes, ajoutez le paramètre **ansible_host** à côté de l'entrée de l'hôte pour spécifier son adresse IP.

Prochaines étapes

- Préparez les nœuds gérés. Pour plus d'informations, voir Préparation d'un nœud géré .

Ressources supplémentaires

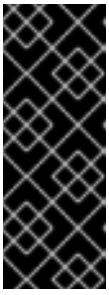
- Étendue de la prise en charge du package Ansible Core inclus dans les référentiels AppStream RHEL 9 et RHEL 8.6 et versions ultérieures
- Comment enregistrer et abonner un système au portail client de Red Hat à l'aide du gestionnaire d'abonnements ?
- La page de manuel **ssh-keygen(1)**
- Se connecter à des machines distantes avec des clés SSH en utilisant ssh-agent
- Paramètres de configuration d'Ansible
- Comment constituer votre inventaire

2.2. PRÉPARATION D'UN NŒUD GÉRÉ

Les nœuds gérés sont les systèmes répertoriés dans l'inventaire et qui seront configurés par le nœud de contrôle conformément au cahier de jeu. Il n'est pas nécessaire d'installer Ansible sur les hôtes gérés.

Conditions préalables

- Vous avez préparé le nœud de contrôle. Pour plus d'informations, voir Préparation d'un nœud de contrôle sur RHEL 9.
- Vous disposez d'un accès SSH à partir du nœud de contrôle.



IMPORTANT

L'accès SSH direct en tant qu'utilisateur **root** présente un risque pour la sécurité. Pour réduire ce risque, vous créerez un utilisateur local sur ce nœud et configurerez une politique **sudo** lors de la préparation d'un nœud géré. Ansible sur le nœud de contrôle peut alors utiliser le compte d'utilisateur local pour se connecter au nœud géré et exécuter des playbooks en tant qu'utilisateurs différents, tels que **root**.

Procédure

1. Créez un utilisateur nommé **ansible**:

```
[root@managed-node-01]# useradd ansible
```

   Le nœud de contrôle utilise ensuite cet utilisateur pour établir une connexion SSH avec cet hôte.

2. Définir un mot de passe pour l'utilisateur **ansible**:

```
[root@managed-node-01]# passwd ansible
Changing password for user ansible.
New password: <password>
```

```
Retype new password: <password>
passwd: all authentication tokens updated successfully.
```

Vous devez saisir ce mot de passe lorsque Ansible utilise **sudo** pour effectuer des tâches en tant qu'utilisateur **root**.

3. Installez la clé publique SSH de l'utilisateur **ansible** sur le nœud géré :

   a. Connectez-vous au nœud de contrôle en tant qu'utilisateur **ansible** et copiez la clé publique SSH sur le nœud géré :

   ```
   [ansible@control-node]$ ssh-copy-id managed-node-01.example.com
   /usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
   "/home/ansible/.ssh/id_rsa.pub"
   The authenticity of host 'managed-node-01.example.com (192.0.2.100)' can't be
   established.
   ECDSA key fingerprint is
   SHA256:9bZ33GJNODK3zbNhybokN/6Mq7hu3vpBXDrCxe7NAvo.
   ```

   b. Lorsque vous y êtes invité, connectez-vous en entrant **yes**:

   ```
   Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
   /usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
   are already installed
   /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
   to install the new keys
   ```

   c. Lorsque vous y êtes invité, saisissez le mot de passe :

   ```
   ansible@managed-node-01.example.com's password: <password>

   Number of key(s) added: 1

   Now try logging into the machine, with:   "ssh '<managed-node-01.example.com>'"
   and check to make sure that only the key(s) you wanted were added.
   ```

   d. Vérifiez la connexion SSH en exécutant à distance une commande sur le nœud de contrôle :

   ```
   [ansible@control-node]$ ssh <managed-node-01.example.com> whoami
   ansible
   ```

4. Créer une configuration **sudo** pour l'utilisateur **ansible**:

   a. Créez et modifiez le fichier **/etc/sudoers.d/ansible** à l'aide de la commande **visudo**:

   ```
   [root@managed-node-01]# visudo /etc/sudoers.d/ansible
   ```

   L'avantage d'utiliser **visudo** plutôt qu'un éditeur normal est que cet utilitaire fournit des contrôles de base et vérifie les erreurs d'analyse avant d'installer le fichier.

   b. Configurez une politique **sudoers** dans le fichier **/etc/sudoers.d/ansible** qui réponde à vos besoins, par exemple :

- Pour autoriser l'utilisateur **ansible** à exécuter toutes les commandes en tant qu'utilisateur et groupe sur cet hôte après avoir saisi le mot de passe de l'utilisateur **ansible**, utilisez l'option suivante :

  ```
  ansible ALL=(ALL) ALL
  ```

- Pour autoriser l'utilisateur **ansible** à exécuter toutes les commandes en tant qu'utilisateur et groupe sur cet hôte sans saisir le mot de passe de l'utilisateur **ansible**, utilisez la commande suivante

  ```
  ansible ALL=(ALL) NOPASSWD: ALL
  ```

Vous pouvez également configurer une politique plus fine qui correspond à vos exigences en matière de sécurité. Pour plus d'informations sur les politiques de **sudoers**, voir la page de manuel **sudoers(5)**.

Vérification

1. Vérifiez que vous pouvez exécuter des commandes à partir du nœud de contrôle sur tous les nœuds gérés :

   ```
   [ansible@control-node]$ ansible all -m ping
   BECOME password: <password>
   managed-node-01.example.com | SUCCESS => {
       "ansible_facts": {
           "discovered_interpreter_python": "/usr/bin/python3"
       },
       "changed": false,
       "ping": "pong"
   }
   ...
   ```

   Le groupe all codé en dur contient dynamiquement tous les hôtes répertoriés dans le fichier d'inventaire.

2. Vérifiez que l'escalade des privilèges fonctionne correctement en exécutant l'utilitaire **whoami** sur un hôte géré à l'aide du module Ansible **command**:

   ```
   [ansible@control-node]$ ansible managed-node-01.example.com -m command -a whoami
   BECOME password: <password>
   managed-node-01.example.com | CHANGED | rc=0 >>
   root
   ```

   Si la commande renvoie la valeur root, vous avez configuré correctement **sudo** sur les nœuds gérés.

Ressources supplémentaires

- Préparation d'un nœud de contrôle sur RHEL 9 .
- La page de manuel **sudoers(5)**

CHAPITRE 3. INSTALLATION ET UTILISATION DES COLLECTIONS

3.1. INTRODUCTION AUX COLLECTIONS ANSIBLE

Les collections Ansible sont la nouvelle façon de distribuer, de maintenir et de consommer l'automatisation. En combinant plusieurs types de contenu Ansible tels que des playbooks, des rôles, des modules et des plugins, vous pouvez bénéficier d'améliorations en termes de flexibilité et d'évolutivité.

Les collections Ansible sont une option au format traditionnel des rôles système RHEL. L'utilisation des Rôles système RHEL dans le format Ansible Collection est presque la même que dans le format traditionnel des Rôles système RHEL. La différence est que les collections Ansible utilisent le concept de **fully qualified collection name** (FQCN), qui se compose d'un **namespace** et d'un **collection name**. Le **namespace** que nous utilisons est le **redhat** et le **collection name** est le **rhel_system_roles**. Ainsi, alors que le format traditionnel RHEL System Roles pour le rôle **kernel_settings** est présenté comme **rhel-system-roles.kernel_settings** (avec des tirets), l'utilisation de la collection **fully qualified collection name** pour le rôle **kernel_settings** serait présentée comme **redhat.rhel_system_roles.kernel_settings** (avec des traits de soulignement).

La combinaison d'un **namespace** et d'un **collection name** garantit l'unicité des objets. Elle garantit également que les objets sont partagés entre les collections Ansible et les espaces de noms sans aucun conflit.

Ressources supplémentaires

- Pour utiliser les collections certifiées Red Hat en accédant à Automation Hub, vous devez disposer d'un abonnement à Ansible Automation Platform (AAP).

3.2. STRUCTURE DES COLLECTIONS

Les collections sont un format de paquetage pour le contenu Ansible. La structure des données est la suivante :

- docs/ : documentation locale pour la collection, avec des exemples, si le rôle fournit la documentation
- galaxy.yml : données sources pour le fichier MANIFEST.json qui fera partie du paquetage Ansible Collection
- playbooks/ : les playbooks sont disponibles ici
  - tasks/ : contient les "fichiers de listes de tâches" pour l'utilisation de include_tasks/import_tasks
- plugins/ : tous les plugins et modules Ansible sont disponibles ici, chacun dans son sous-répertoire
  - modules/ : Modules Ansible
  - modules_utils/ : code commun pour le développement de modules
  - lookup/ : recherche d'un plugin
  - filter/ : Plugin de filtre Jinja2

    - connection/ : plugins de connexion requis si vous n'utilisez pas le plugin par défaut
- roles/ : répertoire des rôles Ansible
- tests/ : tests pour le contenu de la collection

3.3. INSTALLATION DES COLLECTIONS À L'AIDE DE L'INTERFACE DE PROGRAMMATION

Les collections sont un format de distribution pour le contenu Ansible qui peut inclure des playbooks, des rôles, des modules et des plugins.

Vous pouvez installer Collections par l'intermédiaire d'Ansible Galaxy, du navigateur ou de la ligne de commande.

Conditions préalables

- Accès et autorisations à une ou plusieurs *managed nodes*.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
    - Les paquets **ansible-core** et **rhel-system-roles** sont installés.
    - Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

- Installez la collection via un paquetage RPM :

```
# dnf install rhel-system-roles
```

Une fois l'installation terminée, les rôles sont disponibles à l'adresse **redhat.rhel_system_roles.<role_name>**. En outre, vous pouvez trouver la documentation de chaque rôle à l'adresse **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md**.

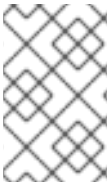
Verification steps

Pour vérifier l'installation, exécutez le rôle **kernel_settings** avec le mode **check** sur votre hôte local. Vous devez également utiliser le paramètre **--become** car il est nécessaire pour le module Ansible **package**. Cependant, ce paramètre ne modifiera pas votre système :

1. Exécutez la commande suivante :

```
$ ansible-playbook -c local -i localhost, --check --become
/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
ests_default.yml
```

La dernière ligne de la sortie de la commande doit contenir la valeur **failed=0**.



NOTE

La virgule après **localhost** est obligatoire. Vous devez l'ajouter même s'il n'y a qu'un seul hôte dans la liste. Sans elle, **ansible-playbook** identifierait **localhost** comme un fichier ou un répertoire.

Ressources supplémentaires

- La page de manuel **ansible-playbook**.
- L'option **-i** de la commande **ansible-playbook**

3.4. INSTALLATION DES COLLECTIONS À PARTIR D'AUTOMATION HUB

Si vous utilisez le Hub d'automatisation, vous pouvez installer la collection de rôles système RHEL hébergée sur le Hub d'automatisation.

Conditions préalables

- Accès et autorisations à une ou plusieurs *managed nodes*.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.
  - Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Définissez Red Hat Automation Hub comme source de contenu par défaut dans le fichier de configuration **ansible.cfg**. Voir Configurer Red Hat Automation Hub en tant que source primaire de contenu .
2. Installez la collection **redhat.rhel_system_roles** à partir du Hub d'automatisation :

   ```
   # ansible-galaxy collection install redhat.rhel_system_roles
   ```

   Une fois l'installation terminée, les rôles sont disponibles à l'adresse **redhat.rhel_system_roles.<role_name>**. En outre, vous pouvez trouver la documentation de chaque rôle à l'adresse **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md**.

Verification steps

Pour vérifier l'installation, exécutez le rôle **kernel_settings** avec le mode **check** sur votre hôte local. Vous devez également utiliser le paramètre **--become** car il est nécessaire pour le module Ansible **package**. Cependant, ce paramètre ne modifiera pas votre système :

1. Exécutez la commande suivante :

```
$ ansible-playbook -c local -i localhost, --check --become
/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
ests_default.yml
```

La dernière ligne de la sortie de la commande doit contenir la valeur **failed=0**.

NOTE

La virgule après **localhost** est obligatoire. Vous devez l'ajouter même s'il n'y a qu'un seul hôte dans la liste. Sans elle, **ansible-playbook** identifierait **localhost** comme un fichier ou un répertoire.

Ressources supplémentaires

- La page de manuel **ansible-playbook**.
- L'option **-i** de la commande **ansible-playbook**

3.5. DÉPLOIEMENT DU RÔLE DE SYSTÈME TLOG RHEL À L'AIDE DE COLLECTIONS

Voici un exemple utilisant Collections pour préparer et appliquer un playbook afin de déployer une solution de journalisation sur un ensemble de machines distinctes.

Conditions préalables

- Une collection Galaxy est installée.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploy session recording
  hosts: all
  vars:
    tlog_scope_sssd: some
    tlog_users_sssd:
      - recordeduser

  roles:
    - redhat.rhel-system-roles.tlog
```

Où ?

- **tlog_scope_sssd**:
  - **some** spécifie que vous souhaitez enregistrer uniquement certains utilisateurs et groupes, pas **all** ou **none**.
- **tlog_users_sssd**:

- **recordeduser** spécifie l'utilisateur à partir duquel vous souhaitez enregistrer une session. Notez que cette fonction n'ajoute pas l'utilisateur à votre place. Vous devez définir l'utilisateur vous-même.

2. Optionnellement, vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

Par conséquent, le playbook installe le rôle **tlog** sur le système que vous avez spécifié. Il crée également un fichier de dépôt de configuration SSSD qui peut être utilisé par les utilisateurs et les groupes que vous définissez. SSSD analyse et lit ces utilisateurs et groupes pour superposer la session **tlog** en tant qu'utilisateur shell. En outre, si le paquet **cockpit** est installé sur le système, le playbook installe également le paquet **cockpit-session-recording**, qui est un module **Cockpit** vous permettant de visualiser et de lire des enregistrements dans l'interface de la console Web.

Verification steps

1. Tester la syntaxe du fichier /**etc**/**rsyslog.conf**:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. Vérifiez que le système envoie des messages au journal :

Pour vérifier que le fichier de dépôt de configuration SSSD est créé dans le système, procédez comme suit :

1. Naviguez jusqu'au dossier dans lequel le fichier de dépôt de la configuration SSSD a été créé :

```
# cd /etc/sssd/conf.d
```

2. Vérifier le contenu du fichier :

```
# cat sssd-session-recording.conf
```

Vous pouvez voir que le fichier contient les paramètres que vous avez définis dans le playbook.

CHAPITRE 4. MODULES ANSIBLE IPMI DANS RHEL

4.1. LA COLLECTION RHEL_MGMT

L'interface de gestion de plate-forme intelligente (IPMI) est une spécification d'un ensemble de protocoles standard pour communiquer avec les dispositifs de contrôleur de gestion de carte de base (BMC). Les modules **IPMI** vous permettent d'activer et de prendre en charge l'automatisation de la gestion du matériel. Les modules **IPMI** sont disponibles en :

- La collection **rhel_mgmt**. Le nom du paquet est **ansible-collection-redhat-rhel_mgmt**.
- L'AppStream RHEL 8, qui fait partie du nouveau paquet **ansible-collection-redhat-rhel_mgmt**.

Les modules IPMI suivants sont disponibles dans la collection rhel_mgmt :

- **ipmi_boot**: Gestion de l'ordre des périphériques de démarrage
- **ipmi_power**: Gestion de l'énergie pour les machines

Les paramètres obligatoires utilisés pour les modules IPMI sont les suivants :

- **ipmi_boot** paramètres :

Nom du module	Description
nom	Nom d'hôte ou adresse IP du BMC
password	Mot de passe pour se connecter au BMC
bootdev	Dispositif à utiliser lors du prochain démarrage * réseau * disquette * hd * sûr * optique * configuration * par défaut
User	Nom d'utilisateur pour se connecter au BMC

- **ipmi_power** paramètres :

Nom du module	Description
nom	Nom d'hôte ou adresse IP du BMC

Nom du module	Description
password	Mot de passe pour se connecter au BMC
utilisateur	Nom d'utilisateur pour se connecter au BMC
État	Vérifier si la machine est dans l'état souhaité * sur * éteint * arrêt * réinitialiser * botte

4.2. INSTALLATION DE LA COLLECTION RHEL MGMT À L'AIDE DU CLI

Vous pouvez installer la collection **rhel_mgmt** à l'aide de la ligne de commande.

Conditions préalables

- Le paquet **ansible-core** est installé.

Procédure

- Installez la collection via un paquetage RPM :

```
# yum install ansible-collection-redhat-rhel_mgmt
```

  Une fois l'installation terminée, les modules IPMI sont disponibles dans la collection Ansible **redhat.rhel_mgmt**.

Ressources supplémentaires

- La page de manuel **ansible-playbook**.

4.3. EXEMPLE D'UTILISATION DU MODULE IPMI_BOOT

L'exemple suivant montre comment utiliser le module **ipmi_boot** dans un playbook pour définir un périphérique de démarrage pour le prochain démarrage. Pour des raisons de simplicité, les exemples utilisent le même hôte que l'hôte de contrôle Ansible et l'hôte géré, ce qui permet d'exécuter les modules sur le même hôte que celui où le playbook est exécuté.

Conditions préalables

- La collection rhel_mgmt est installée.
- La bibliothèque **pyghmi** du paquet **python3-pyghmi** est installée dans l'un des emplacements suivants :

- L'hôte sur lequel vous exécutez le manuel de jeu.
- L'hôte géré. Si vous utilisez localhost comme hôte géré, installez le paquetage **python3-pyghmi** sur l'hôte où vous exécutez le playbook à la place.

- La BMC IPMI que vous souhaitez contrôler est accessible via le réseau depuis l'hôte sur lequel vous exécutez le playbook, ou l'hôte géré (si vous n'utilisez pas localhost comme hôte géré). Notez que l'hôte dont la BMC est configurée par le module est généralement différent de l'hôte où le module est exécuté (l'hôte géré par Ansible), car le module contacte la BMC sur le réseau en utilisant le protocole IPMI.
- Vous disposez d'informations d'identification vous permettant d'accéder à BMC avec un niveau d'accès approprié.

Procédure

1. Créez un nouveau fichier *playbook.yml* avec le contenu suivant :

```
---
- name: Sets which boot device will be used on next boot
  hosts: localhost
    tasks:
    - redhat.rhel_mgmt.ipmi_boot:
        name: bmc.host.example.com
          user: admin_user
          password: basics
          bootdev: hd
```

2. Exécuter le playbook contre localhost :

```
# ansible-playbook playbook.yml
```

En conséquence, la sortie renvoie la valeur "success".

4.4. EXEMPLE AVEC LE MODULE IPMI_POWER

Cet exemple montre comment utiliser le module **ipmi_boot** dans un playbook pour vérifier si le système est allumé. Pour des raisons de simplicité, les exemples utilisent le même hôte que l'hôte de contrôle Ansible et l'hôte géré, ce qui permet d'exécuter les modules sur le même hôte que celui où le playbook est exécuté.

Conditions préalables

- La collection rhel_mgmt est installée.
- La bibliothèque **pyghmi** du paquet **python3-pyghmi** est installée dans l'un des emplacements suivants :
  - L'hôte sur lequel vous exécutez le manuel de jeu.
  - L'hôte géré. Si vous utilisez localhost comme hôte géré, installez le paquetage **python3-pyghmi** sur l'hôte où vous exécutez le playbook à la place.
- La BMC IPMI que vous souhaitez contrôler est accessible via le réseau depuis l'hôte sur lequel vous exécutez le playbook, ou l'hôte géré (si vous n'utilisez pas localhost comme hôte géré).

Notez que l'hôte dont la BMC est configurée par le module est généralement différent de l'hôte où le module est exécuté (l'hôte géré par Ansible), car le module contacte la BMC sur le réseau en utilisant le protocole IPMI.

- Vous disposez d'informations d'identification vous permettant d'accéder à BMC avec un niveau d'accès approprié.

Procédure

1. Créez un nouveau fichier *playbook.yml* avec le contenu suivant :

```
---
- name: Turn the host on
  hosts: localhost
    tasks:
    - redhat.rhel_mgmt.ipmi_power:
        name: bmc.host.example.com
          user: admin_user
          password: basics
          state: on
```

2. Exécuter le cahier des charges :

```
# ansible-playbook playbook.yml
```

La sortie renvoie la valeur "true".

CHAPITRE 5. LES MODULES REDFISH DANS RHEL

Les modules Redfish pour la gestion à distance de périphériques font désormais partie de la collection Ansible de **redhat.rhel_mgmt**. Avec les modules Redfish, vous pouvez facilement utiliser l'automatisation de la gestion sur les serveurs bare-metal et le matériel de plate-forme en obtenant des informations sur les serveurs ou en les contrôlant via un contrôleur hors bande (OOB), en utilisant le transport HTTPS standard et le format JSON.

5.1. LES MODULES REDFISH

La collection Ansible **redhat.rhel_mgmt** fournit les modules Redfish pour prendre en charge la gestion du matériel dans Ansible via Redfish. La collection **redhat.rhel_mgmt** est disponible dans le paquetage **ansible-collection-redhat-rhel_mgmt**. Pour l'installer, voir Installation de la collection redhat.rhel_mgmt à l'aide de l'interface de programmation.

Les modules Redfish suivants sont disponibles dans la collection **redhat.rhel_mgmt**:

1. **redfish_info**: Le module **redfish_info** récupère des informations sur le contrôleur hors bande (OOB) distant, telles que l'inventaire des systèmes.
2. **redfish_command**: Le module **redfish_command** effectue des opérations de contrôle hors bande (OOB) telles que la gestion des journaux et des utilisateurs, ainsi que des opérations d'alimentation telles que le redémarrage du système, la mise sous tension et la mise hors tension.
3. **redfish_config**: Le module **redfish_config** effectue les opérations du contrôleur OOB telles que la modification de la configuration OOB ou le réglage de la configuration BIOS.

5.2. PARAMÈTRES DES MODULES REDFISH

Les paramètres utilisés pour les modules Redfish sont les suivants :

redfish_info paramètres :	Description
baseuri	(Obligatoire) - URI de base du contrôleur OOB.
category	(Obligatoire) - Liste des catégories à exécuter sur le contrôleur OOB. La valeur par défaut est ["Systems"].
command	(Obligatoire) - Liste des commandes à exécuter sur le contrôleur OOB.
username	Nom d'utilisateur pour l'authentification auprès du contrôleur OOB.
password	Mot de passe pour l'authentification au contrôleur OOB.

redfish_command paramètres :	Description
baseuri	(Obligatoire) - URI de base du contrôleur OOB.
category	(Obligatoire) - Liste des catégories à exécuter sur le contrôleur OOB. La valeur par défaut est ["Systems"].
command	(Obligatoire) - Liste des commandes à exécuter sur le contrôleur OOB.
username	Nom d'utilisateur pour l'authentification auprès du contrôleur OOB.
password	Mot de passe pour l'authentification au contrôleur OOB.

redfish_config paramètres :	Description
baseuri	(Obligatoire) - URI de base du contrôleur OOB.
category	(Obligatoire) - Liste des catégories à exécuter sur le contrôleur OOB. La valeur par défaut est ["Systems"].
command	(Obligatoire) - Liste des commandes à exécuter sur le contrôleur OOB.
username	Nom d'utilisateur pour l'authentification auprès du contrôleur OOB.
password	Mot de passe pour l'authentification au contrôleur OOB.
bios_attributes	Attributs du BIOS à mettre à jour.

5.3. UTILISATION DU MODULE REDFISH_INFO

L'exemple suivant montre comment utiliser le module **redfish_info** dans un playbook pour obtenir des informations sur l'inventaire des CPU. Pour des raisons de simplicité, l'exemple utilise le même hôte que l'hôte de contrôle Ansible et l'hôte géré, ce qui permet d'exécuter les modules sur le même hôte que celui où le playbook est exécuté.

Conditions préalables

- La collection **redhat.rhel_mgmt** est installée.

- La bibliothèque **pyghmi** du paquetage **python3-pyghmi** est installée sur l'hôte géré. Si vous utilisez localhost comme hôte géré, installez le paquetage **python3-pyghmi** sur l'hôte où vous exécutez le playbook.
- Détails de l'accès au contrôleur OOB.

Procédure

1. Créez un nouveau fichier *playbook.yml* avec le contenu suivant :

```
---
- name: Get CPU inventory
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.redfish_info:
        baseuri: "{{ baseuri }}"
        username: "{{ username }}"
        password: "{{ password }}"
        category: Systems
        command: GetCpuInventory
      register: result
```

2. Exécuter le playbook contre localhost :

```
# ansible-playbook playbook.yml
```

En conséquence, la sortie renvoie les détails de l'inventaire de l'unité centrale.

5.4. UTILISATION DU MODULE REDFISH_COMMAND

L'exemple suivant montre comment utiliser le module **redfish_command** dans un playbook pour allumer un système. Pour des raisons de simplicité, l'exemple utilise le même hôte que l'hôte de contrôle Ansible et l'hôte géré, ce qui permet d'exécuter les modules sur le même hôte que celui où le playbook est exécuté.

Conditions préalables

- La collection **redhat.rhel_mgmt** est installée.
- La bibliothèque **pyghmi** du paquetage **python3-pyghmi** est installée sur l'hôte géré. Si vous utilisez localhost comme hôte géré, installez le paquetage **python3-pyghmi** sur l'hôte où vous exécutez le playbook.
- Détails de l'accès au contrôleur OOB.

Procédure

1. Créez un nouveau fichier *playbook.yml* avec le contenu suivant :

```
---
- name: Power on system
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.redfish_command:
```

```
        baseuri: "{{ baseuri }}"
        username: "{{ username }}"
        password: "{{ password }}"
        category: Systems
        command: PowerOn
```

2. Exécuter le playbook contre localhost :

```
# ansible-playbook playbook.yml
```

Le système se met alors en marche.

5.5. UTILISATION DU MODULE REDFISH_CONFIG

L'exemple suivant montre comment utiliser le module **redfish_config** dans un playbook pour configurer un système afin qu'il démarre avec l'UEFI. Pour des raisons de simplicité, l'exemple utilise le même hôte que l'hôte de contrôle Ansible et l'hôte géré, ce qui permet d'exécuter les modules sur le même hôte que celui où le playbook est exécuté.

Conditions préalables

- La collection **redhat.rhel_mgmt** est installée.
- La bibliothèque **pyghmi** du paquetage **python3-pyghmi** est installée sur l'hôte géré. Si vous utilisez localhost comme hôte géré, installez le paquetage **python3-pyghmi** sur l'hôte où vous exécutez le playbook.
- Détails de l'accès au contrôleur OOB.

Procédure

1. Créez un nouveau fichier *playbook.yml* avec le contenu suivant :

```
---
- name: "Set BootMode to UEFI"
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.redfish_config:
        baseuri: "{{ baseuri }}"
        username: "{{ username }}"
        password: "{{ password }}"
        category: Systems
        command: SetBiosAttributes
        bios_attributes:
          BootMode: Uefi
```

2. Exécuter le playbook contre localhost :

```
# ansible-playbook playbook.yml
```

Par conséquent, le mode de démarrage du système est réglé sur UEFI.

CHAPITRE 6. CONFIGURATION PERMANENTE DES PARAMÈTRES DU NOYAU À L'AIDE DE KERNEL_SETTINGS RHEL SYSTEM ROLE

Vous pouvez utiliser le rôle **kernel_settings** pour configurer les paramètres du noyau sur plusieurs clients à la fois. Cette solution :

- Fournit une interface conviviale avec des paramètres d'entrée efficaces.
- Conserve tous les paramètres du noyau en un seul endroit.

Après avoir exécuté le rôle **kernel_settings** à partir de la machine de contrôle, les paramètres du noyau sont appliqués immédiatement aux systèmes gérés et persistent à travers les redémarrages.

IMPORTANT

Notez que les rôles système RHEL livrés sur les canaux RHEL sont disponibles pour les clients RHEL sous forme de paquetage RPM dans le référentiel AppStream par défaut. Les rôles système RHEL sont également disponibles sous forme de collection pour les clients ayant des abonnements Ansible via Ansible Automation Hub.

6.1. INTRODUCTION AU RÔLE KERNEL_SETTINGS

RHEL System Roles est un ensemble de rôles qui fournit une interface de configuration cohérente pour gérer à distance plusieurs systèmes.

Les rôles système RHEL ont été introduits pour les configurations automatisées du noyau à l'aide du rôle système **kernel_settings**. Le paquet **rhel-system-roles** contient ce rôle système, ainsi que la documentation de référence.

Pour appliquer les paramètres du noyau sur un ou plusieurs systèmes de manière automatisée, utilisez le rôle **kernel_settings** avec une ou plusieurs de ses variables de rôle de votre choix dans un playbook. Un playbook est une liste d'un ou plusieurs jeux lisibles par l'homme et écrits au format YAML.

Le rôle **kernel_settings** vous permet de configurer :

- Les paramètres du noyau en utilisant la variable de rôle **kernel_settings_sysctl**
- Divers sous-systèmes du noyau, périphériques matériels et pilotes de périphériques utilisant la variable de rôle **kernel_settings_sysfs**
- L'affinité du processeur pour le gestionnaire de service **systemd** et les processus qui en découlent en utilisant la variable de rôle **kernel_settings_systemd_cpu_affinity**
- Le sous-système de mémoire du noyau utilise les variables de rôle **kernel_settings_transparent_hugepages** et **kernel_settings_transparent_hugepages_defrag** pour rendre les hugepages transparentes

Ressources supplémentaires

- **README.md** et **README.html** dans le répertoire **/usr/share/doc/rhel-system-roles/kernel_settings/**
- Travailler avec des playbooks

- Comment constituer votre inventaire

6.2. APPLICATION DES PARAMÈTRES SÉLECTIONNÉS DU NOYAU À L'AIDE DU RÔLE KERNEL_SETTINGS

Suivez ces étapes pour préparer et appliquer un playbook Ansible afin de configurer à distance les paramètres du noyau avec un effet persistant sur plusieurs systèmes d'exploitation gérés.

Conditions préalables

- Vous avez les autorisations **root**.
- En vertu de votre abonnement RHEL, vous avez installé les paquets **ansible-core** et **rhel-system-roles** sur la machine de contrôle.
- Un inventaire des hôtes gérés est présent sur la machine de contrôle et Ansible peut s'y connecter.

IMPORTANT

RHEL 8.0 - 8.5 donne accès à un dépôt Ansible séparé qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**; des connecteurs tels que **docker** et **podman**; et tout l'univers des plugins et des modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, reportez-vous à la section Comment télécharger et installer Red Hat Ansible Engine ?

RHEL 8.6 et 9.0 a introduit Ansible Core (fourni en tant que **ansible-core** RPM), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. Le dépôt AppStream fournit **ansible-core**, qui a une portée de support limitée. Pour en savoir plus, consultez le document Scope of support for the ansible-core package included in the RHEL 9 AppStream.

Procédure

1. Si vous le souhaitez, vous pouvez consulter le fichier **inventory** à des fins d'illustration :

```
#  cat /home/jdoe/<ansible_project_name>/inventory
[testingservers]
pdoe@192.168.122.98
fdoe@192.168.122.226

[db-servers]
db1.example.com
db2.example.com

[webservers]
web1.example.com
web2.example.com
192.0.2.42
```

Ce fichier définit le groupe **[testingservers]** et d'autres groupes. Il vous permet d'exécuter Ansible plus efficacement sur un ensemble spécifique de systèmes.

2. Créer un fichier de configuration pour définir les valeurs par défaut et l'escalade des privilèges pour les opérations Ansible.

   a. Créez un nouveau fichier YAML et ouvrez-le dans un éditeur de texte, par exemple :

   ```
   # vi /home/jdoe/<ansible_project_name>/ansible.cfg
   ```

   b. Insérez le contenu suivant dans le fichier :

   ```
   [defaults]
   inventory = ./inventory

   [privilege_escalation]
   become = true
   become_method = sudo
   become_user = root
   become_ask_pass = true
   ```

   La section **[defaults]** indique un chemin d'accès au fichier d'inventaire des hôtes gérés. La section **[privilege_escalation]** définit que les privilèges de l'utilisateur doivent passer à **root** sur les hôtes gérés spécifiés. Ceci est nécessaire pour une configuration réussie des paramètres du noyau. Lorsque le playbook Ansible est exécuté, vous serez invité à saisir le mot de passe de l'utilisateur. L'utilisateur passe automatiquement à **root** par le biais de **sudo** après s'être connecté à un hôte géré.

3. Créer un playbook Ansible qui utilise le rôle **kernel_settings**.

   a. Créez un nouveau fichier YAML et ouvrez-le dans un éditeur de texte, par exemple :

   ```
   # vi /home/jdoe/<ansible_project_name>/kernel-roles.yml
   ```

   Ce fichier représente un playbook et contient généralement une liste ordonnée de tâches, également appelées *plays*, qui sont exécutées contre des hôtes gérés spécifiques sélectionnés dans votre fichier **inventory**.

   b. Insérez le contenu suivant dans le fichier :

   ```
   ---
   -
     hosts: testingservers
     name: "Configure kernel settings"
     roles:
       - rhel-system-roles.kernel_settings
     vars:
       kernel_settings_sysctl:
         - name: fs.file-max
           value: 400000
         - name: kernel.threads-max
           value: 65536
       kernel_settings_sysfs:
         - name: /sys/class/net/lo/mtu
           value: 65000
       kernel_settings_transparent_hugepages: madvise
   ```

   La clé **name** est facultative. Elle associe une chaîne arbitraire à la pièce en tant qu'étiquette

et identifie l'objet de la pièce. La clé **hosts** de la pièce spécifie les hôtes contre lesquels la pièce est exécutée. La ou les valeurs de cette clé peuvent être fournies sous forme de noms individuels d'hôtes gérés ou de groupes d'hôtes tels que définis dans le fichier **inventory**.

La section **vars** représente une liste de variables contenant les noms des paramètres du noyau sélectionnés et les valeurs auxquelles ils doivent être définis.

La clé **roles** spécifie le rôle du système qui va configurer les paramètres et les valeurs mentionnés dans la section **vars**.

NOTE

Vous pouvez modifier les paramètres du noyau et leurs valeurs dans le playbook en fonction de vos besoins.

4. En option, vérifiez que la syntaxe de votre pièce est correcte.

```
# ansible-playbook --syntax-check kernel-roles.yml

playbook: kernel-roles.yml
```

Cet exemple montre la vérification réussie d'un playbook.

5. Exécutez votre cahier des charges.

```
# ansible-playbook kernel-roles.yml

...

BECOME password:

PLAY [Configure kernel settings]
*********************************************************************************



PLAY RECAP
*********************************************************************************************************************
fdoe@192.168.122.226       : ok=10   changed=4    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
pdoe@192.168.122.98        : ok=10   changed=4    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
```

Avant qu'Ansible n'exécute votre playbook, vous allez être invité à saisir votre mot de passe afin qu'un utilisateur sur les hôtes gérés puisse être basculé sur **root**, ce qui est nécessaire pour configurer les paramètres du noyau.

La section récapitulative montre que la lecture s'est terminée avec succès (**failed=0**) pour tous les hôtes gérés, et que 4 paramètres du noyau ont été appliqués (**changed=4**).

6. Redémarrez les hôtes gérés et vérifiez les paramètres du noyau concernés pour vous assurer que les changements ont été appliqués et qu'ils persistent après les redémarrages.

Ressources supplémentaires

- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL
- **README.html** et **README.md** dans le répertoire **/usr/share/doc/rhel-system-roles/kernel_settings/**
- Constituez votre inventaire
- Configuration d'Ansible
- Travailler avec des Playbooks
- Utilisation de variables
- Rôles

CHAPITRE 7. UTILISATION DU RÔLE DE SYSTÈME RHC POUR ENREGISTRER LE SYSTÈME

Le rôle de système RHEL **rhc** permet aux administrateurs d'automatiser l'enregistrement de plusieurs systèmes auprès de Red Hat Subscription Management (RHSM) et des serveurs Satellite. Le rôle prend également en charge les tâches de configuration et de gestion liées à Insights en utilisant Ansible.

7.1. INTRODUCTION AU RÔLE DU SYSTÈME RHC

Le rôle système RHEL est un ensemble de rôles qui fournit une interface de configuration cohérente pour gérer à distance plusieurs systèmes. Le rôle de système de configuration d'hôte à distance (**rhc**) permet aux administrateurs d'enregistrer facilement les systèmes RHEL sur les serveurs RHSM (Red Hat Subscription Management) et Satellite. Par défaut, lorsque vous enregistrez un système à l'aide du rôle système **rhc**, le système est connecté à Insights. De plus, avec le rôle système **rhc**, vous pouvez :

- Configurer les connexions à Red Hat Insights
- Activer et désactiver les référentiels
- Configurer le proxy à utiliser pour la connexion
- Configurer les insights remédiations et les mises à jour automatiques
- Définir le déblocage du système
- Configurer les balises insights

7.2. ENREGISTRER UN SYSTÈME EN UTILISANT LE RÔLE DE SYSTÈME RHC

Vous pouvez enregistrer votre système auprès de Red Hat en utilisant le rôle de système **rhc** RHEL. Par défaut, le rôle de système **rhc** RHEL connecte le système à Red Hat Insights lorsque vous l'enregistrez.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un coffre-fort pour sauvegarder les informations sensibles :

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. La commande **ansible-vault create** crée un fichier crypté de l'espace de stockage et l'ouvre dans un éditeur. Saisissez les données sensibles que vous souhaitez enregistrer dans le coffre-fort, par exemple :

```
activationKey: activation_key
username: username
password: password
```

3. Enregistrez les modifications et fermez l'éditeur. Ansible crypte les données dans le coffre-fort. Vous pouvez ensuite modifier les données dans l'espace de stockage à l'aide de la commande **ansible-vault edit *secrets.yml*** pour modifier les données du coffre-fort.

4. Optionnel : Afficher le contenu du coffre-fort :

```
$ ansible-vault view secrets.yml
```

5. Créez un fichier playbook, par exemple ~/**registration.yml**, et utilisez l'une des options suivantes en fonction de l'action que vous souhaitez effectuer :

   a. Pour s'enregistrer à l'aide d'une clé d'activation et d'un identifiant d'organisation (recommandé), utilisez la procédure suivante :

```
---
- name: Registering system using activation key and organization ID
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      activation_keys:
        keys:
          - "{{ activationKey }}"
     rhc_organization: organizationID
  roles:
    - role: rhel-system-roles.rhc
```

   b. Pour s'enregistrer en utilisant un nom d'utilisateur et un mot de passe, utilisez la procédure suivante :

```
---
- name: Registering system with username and password
  hosts:  managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
  roles:
    - role: rhel-system-roles.rhc
```

6. Exécutez le manuel de jeu :

```
# ansible-playbook ~/registration.yml --ask-vault-pass
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.3. ENREGISTRER UN SYSTÈME AVEC SATELLITE EN UTILISANT LE RÔLE DE SYSTÈME RHC

Lorsque les organisations utilisent Satellite pour gérer les systèmes, il est nécessaire d'enregistrer le système via Satellite. Vous pouvez enregistrer votre système à distance avec Satellite en utilisant le rôle de système RHEL **rhc**.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un coffre-fort pour sauvegarder les informations sensibles :

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. La commande **ansible-vault create** crée un fichier crypté et l'ouvre dans un éditeur. Saisissez les données sensibles que vous souhaitez enregistrer dans le coffre-fort, par exemple :

```
activationKey : activation_key
```

3. Enregistrez les modifications et fermez l'éditeur. Ansible crypte les données dans le coffre-fort. Vous pouvez ensuite modifier les données dans l'espace de stockage à l'aide de la commande **ansible-vault edit *secrets.yml*** pour modifier les données du coffre-fort.
4. Optionnel : Afficher le contenu du coffre-fort :

```
$ ansible-vault view secrets.yml
```

5. Créez un fichier playbook, par exemple **~/registration-sat.yml**.
6. Utilisez le texte suivant à l'adresse **~/registration-sat.yml** pour enregistrer le système à l'aide d'une clé d'activation et d'un identifiant d'organisation :

```
---
- name: Register to the custom registration server and CDN
```

```
hosts: managed-node-01.example.com
vars_files:
  - secrets.yml
vars:
  rhc_auth:
    login:
      activation_keys:
        keys:
          - "{{ activationKey }}"
      rhc_organization: organizationID
  rhc_server:
    hostname: example.com
      port: 443
      prefix: /rhsm
  rhc_baseurl: http://example.com/pulp/content
 roles:
   - role: rhel-system-roles.rhc
```

7. Exécutez le manuel de jeu :

```
# ansible-playbook ~/registration-sat.yml --ask-vault-pass
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.4. DÉSACTIVER LA CONNEXION À INSIGHTS APRÈS L'ENREGISTREMENT EN UTILISANT LE RÔLE DE SYSTÈME RHC

Lorsque vous enregistrez un système à l'aide du rôle de système **rhc** RHEL, le rôle active par défaut la connexion à Red Hat Insights. Vous pouvez la désactiver en utilisant le rôle de système **rhc**, si cela n'est pas nécessaire.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Le système est déjà enregistré.

Procédure

1. Créez un fichier playbook, par exemple **~/dis-insights.yml** et ajoutez-y le contenu suivant :

```
---
- name: Disable Insights connection
```

```
hosts: managed-node-01.example.com
vars:
  rhc_insights:
    state: absent
roles:
  - role: rhel-system-roles.rhc
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/dis-insights.yml
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.5. ACTIVATION DES RÉFÉRENTIELS EN UTILISANT LE RÔLE SYSTÈME RHC

Vous pouvez activer ou désactiver à distance les référentiels sur les nœuds gérés en utilisant le rôle système **rhc** RHEL.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Vous disposez des détails des référentiels que vous souhaitez activer ou désactiver sur les nœuds gérés.
- Vous avez enregistré le système.

Procédure

1. Créez un fichier playbook, par exemple **~/configure-repos.yml**:
   a. Pour activer un référentiel :

```
---
- name: Enable repository
  hosts: managed-node-01.example.com
  vars:
    rhc_repositories:
      - {name: "RepositoryName", state: enabled}
  roles:
    - role: rhel-system-roles.rhc
```

b. Pour désactiver un référentiel :

```
---
- name: Disable repository
  hosts: managed-node-01.example.com
  vars:
    rhc_repositories:
      - {name: "RepositoryName", state: disabled}
  roles:
    - role: rhel-system-roles.rhc
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/configure-repos.yml
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.6. DÉFINIR LES VERSIONS EN UTILISANT LE RÔLE SYSTÈME RHC

Vous pouvez limiter le système à l'utilisation des dépôts d'une version mineure particulière de RHEL au lieu de la dernière. Vous pouvez ainsi verrouiller votre système sur une version mineure spécifique de RHEL.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Vous connaissez la version mineure de RHEL à laquelle vous souhaitez verrouiller le système. Notez que vous ne pouvez verrouiller le système que sur la version mineure de RHEL que l'hôte exécute actuellement ou sur une version mineure ultérieure.
- Vous avez enregistré le système.

Procédure

1. Créez un fichier playbook, par exemple **~/release.yml**:

```
---
- name: Set Release
  hosts: managed-node-01.example.com
  vars:
```

```
    rhc_release: "8.6"
  roles:
    - role: rhel-system-roles.rhc
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/release.yml
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.7. UTILISATION D'UN SERVEUR PROXY LORS DE L'ENREGISTREMENT DE L'HÔTE À L'AIDE DU RÔLE SYSTÈME RHC

Si vos restrictions de sécurité n'autorisent l'accès à Internet que par l'intermédiaire d'un serveur proxy, vous pouvez spécifier les paramètres du proxy dans le cahier de jeu lorsque vous enregistrez le système à l'aide du rôle de système RHEL **rhc**.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un coffre-fort pour sauvegarder les informations sensibles :

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. La commande **ansible-vault create** crée un fichier crypté et l'ouvre dans un éditeur. Saisissez les données sensibles que vous souhaitez enregistrer dans le coffre-fort, par exemple :

```
username: username
password: password
proxy_username: proxyusernme
proxy_password: proxypassword
```

3. Enregistrez les modifications et fermez l'éditeur. Ansible crypte les données dans le coffre-fort. Vous pouvez ensuite modifier les données dans l'espace de stockage à l'aide de la commande **ansible-vault edit *secrets.yml*** pour modifier les données du coffre-fort.

4. Optionnel : Afficher le contenu du coffre-fort :

```
$ ansible-vault view secrets.yml
```

5. Créez un fichier playbook, par exemple **~/configure-proxy.yml**:

   a. Pour s'inscrire au portail client RHEL en utilisant un proxy :

```
---
- name: Register using proxy
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
    rhc_proxy:
      hostname: proxy.example.com
      port: 3128
      username: "{{ proxy_username }}"
      password: "{{ proxy_password }}"
  roles:
    - role: rhel-system-roles.rhc
```

   b. Pour supprimer le serveur proxy de la configuration du service Red Hat Subscription Manager :

```
---
- name: To stop using proxy server for registration
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
     rhc_proxy: {"state":"absent"}
  roles:
    - role: rhel-system-roles.rhc
```

6. Exécutez le manuel de jeu :

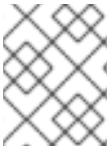
```
# ansible-playbook ~/configure-proxy.yml --ask-vault-pass
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.8. DÉSACTIVER LES MISES À JOUR AUTOMATIQUES DES RÈGLES INSIGHTS EN UTILISANT LE RÔLE SYSTÈME RHC

Vous pouvez désactiver les mises à jour automatiques des règles de collecte pour Red Hat Insights en utilisant le rôle de système **rhc** RHEL. Par défaut, lorsque vous connectez votre système à Red Hat Insights, cette option est activée. Vous pouvez la désactiver en utilisant le rôle de système RHEL **rhc**.



NOTE

Si vous désactivez cette fonction, vous risquez d'utiliser des fichiers de définition de règles obsolètes et de ne pas recevoir les mises à jour de validation les plus récentes.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Vous avez enregistré le système.

Procédure

1. Créez un coffre-fort pour sauvegarder les informations sensibles :

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. La commande **ansible-vault create** crée un fichier crypté et l'ouvre dans un éditeur. Saisissez les données sensibles que vous souhaitez enregistrer dans le coffre-fort, par exemple :

```
username: username
password: password
```

3. Enregistrez les modifications et fermez l'éditeur. Ansible crypte les données dans le coffre-fort. Vous pouvez ensuite modifier les données dans l'espace de stockage à l'aide de la commande **ansible-vault edit *secrets.yml*** pour modifier les données du coffre-fort.

4. Optionnel : Afficher le contenu du coffre-fort :

```
$ ansible-vault view secrets.yml
```

5. Créez un fichier playbook, par exemple **~/auto-update.yml** et ajoutez-y le contenu suivant :

```
---
 - name: Disable Red Hat Insights autoupdates
   hosts: managed-node-01.example.com
   vars_files:
     - secrets.yml
   vars:
```

```
rhc_auth:
  login:
    username: "{{ username }}"
    password: "{{ password }}"
rhc_insights:
  autoupdate: false
  state: present
roles:
  - role: rhel-system-roles.rhc
```

6. Exécutez le manuel de jeu :

```
# ansible-playbook ~/auto-update.yml --ask-vault-pass
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.9. DÉSACTIVATION DES REMÉDIATIONS INSIGHTS À L'AIDE DU RÔLE SYSTÈME RHEL RHC

Vous pouvez configurer les systèmes pour qu'ils mettent automatiquement à jour la configuration dynamique en utilisant le rôle de système RHEL **rhc**. Lorsque vous connectez votre système à Red Hat Insights, ce rôle est activé par défaut. Vous pouvez le désactiver s'il n'est pas nécessaire.

NOTE

L'activation de la remédiation avec le rôle de système **rhc** garantit que votre système est prêt à être remédié lorsqu'il est connecté directement à Red Hat. Pour les systèmes connectés à un Satellite ou à une Capsule, l'activation de la remédiation doit être réalisée différemment. Pour plus d'informations sur les remédiations de Red Hat Insights, consultez le Guide de remédiation de Red Hat Insights .

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Les remédiations Insights sont activées.
- Vous avez enregistré le système.

Procédure

1. Pour activer la remédiation, créez un fichier playbook, par exemple **~/remediation.yml**:

```
---
- name: Disable remediation
  hosts: managed-node-01.example.com
  vars:
    rhc_insights:
      remediation: absent
      state: present
  roles:
    - role: rhel-system-roles.rhc
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/remediation.yml
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

7.10. CONFIGURATION DES BALISES INSIGHTS À L'AIDE DU RÔLE SYSTÈME RHC

Vous pouvez utiliser les balises pour le filtrage et le regroupement du système. Vous pouvez également personnaliser les balises en fonction de vos besoins.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un coffre-fort pour sauvegarder les informations sensibles :

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. La commande **ansible-vault create** crée un fichier crypté et l'ouvre dans un éditeur. Saisissez les données sensibles que vous souhaitez enregistrer dans le coffre-fort, par exemple :

```
username: username
password: password
```

3. Enregistrez les modifications et fermez l'éditeur. Ansible crypte les données dans le coffre-fort.

Vous pouvez ensuite modifier les données dans l'espace de stockage à l'aide de la commande **ansible-vault edit *secrets.yml*** pour modifier les données du coffre-fort.

4. Optionnel : Afficher le contenu du coffre-fort :

```
$ ansible-vault view secrets.yml
```

5. Créez un fichier playbook, par exemple **~/tags.yml**, et ajoutez-y le contenu suivant :

```
---
- name: Creating tags
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
    rhc_insights:
      tags:
        group: group-name-value
          location: location-name-value
          description:
            - RHEL8
            - SAP
           sample_key:value
        state: present
  roles:
    - role: rhel-system-roles.rhc
```

6. Exécutez le manuel de jeu :

```
# ansible-playbook ~/remediation.yml --ask-vault-pass
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**
- Pour plus d'informations, reportez-vous à Filtrage du système et groupes Red Hat Insights .

7.11. DÉSENREGISTREMENT D'UN SYSTÈME À L'AIDE DU RÔLE DE SYSTÈME RHC

Vous pouvez désenregistrer le système auprès de Red Hat si vous n'avez plus besoin du service d'abonnement.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.

- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Le système est déjà enregistré.

Procédure

1. Pour annuler l'enregistrement, créez un fichier playbook, par exemple **~/unregister.yml**, et ajoutez-y le contenu suivant :

```
---
- name: Unregister the system
  hosts: managed-node-01.example.com
  vars:
    rhc_state: absent
  roles:
    - role: rhel-system-roles.rhc
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/unregister.yml
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md**

CHAPITRE 8. CONFIGURATION DES PARAMÈTRES RÉSEAU À L'AIDE DES RÔLES SYSTÈME RHEL

Les administrateurs peuvent automatiser les tâches de configuration et de gestion liées au réseau en utilisant le rôle de système RHEL **network**.

8.1. CONFIGURATION D'UNE CONNEXION ETHERNET AVEC UNE ADRESSE IP STATIQUE EN UTILISANT LE RÔLE DE SYSTÈME RHEL AVEC UN NOM D'INTERFACE

Vous pouvez configurer à distance une connexion Ethernet à l'aide de **network** RHEL System Role.

Par exemple, la procédure ci-dessous crée un profil de connexion NetworkManager pour le périphérique **enp7s0** avec les paramètres suivants :

- Une adresse IPv4 statique - **192.0.2.1** avec un masque de sous-réseau **/24**
- Une adresse IPv6 statique - **2001:db8:1::1** avec un masque de sous-réseau **/64**
- Une passerelle par défaut IPv4 - **192.0.2.254**
- Une passerelle par défaut IPv6 - **2001:db8:1::fffe**
- Un serveur DNS IPv4 - **192.0.2.200**
- Un serveur DNS IPv6 - **2001:db8:1::ffbb**
- Un domaine de recherche DNS - **example.com**

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Un périphérique Ethernet physique ou virtuel existe dans la configuration du serveur.
- Les nœuds gérés utilisent NetworkManager pour configurer le réseau.

Procédure

1. Créez un fichier playbook, par exemple ***~/ethernet-static-IP.yml***avec le contenu suivant :

```
---
- name: Configure the network
```

```
hosts: managed-node-01.example.com
tasks:
- name: Configure an Ethernet connection with static IP
  include_role:
    name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp7s0
        interface_name: enp7s0
        type: ethernet
        autoconnect: yes
        ip:
          address:
            - 192.0.2.1/24
            - 2001:db8:1::1/64
          gateway4: 192.0.2.254
          gateway6: 2001:db8:1::fffe
          dns:
            - 192.0.2.200
            - 2001:db8:1::ffbb
          dns_search:
            - example.com
        state: up
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/ethernet-static-IP.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.2. CONFIGURATION D'UNE CONNEXION ETHERNET AVEC UNE ADRESSE IP STATIQUE EN UTILISANT LE RÔLE DE SYSTÈME RHEL AVEC UN CHEMIN D'ACCÈS AUX PÉRIPHÉRIQUES

Vous pouvez configurer à distance une connexion Ethernet à l'aide de **network** RHEL System Role.

Vous pouvez identifier le chemin d'accès à l'appareil à l'aide de la commande suivante :

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

Par exemple, la procédure ci-dessous crée un profil de connexion NetworkManager avec les paramètres suivants pour le périphérique qui correspond à l'expression PCI ID **0000:00:0[1-3].0**, mais pas **0000:00:02.0**:

- Une adresse IPv4 statique - **192.0.2.1** avec un masque de sous-réseau **/24**
- Une adresse IPv6 statique - **2001:db8:1::1** avec un masque de sous-réseau **/64**
- Une passerelle par défaut IPv4 - **192.0.2.254**
- Une passerelle par défaut IPv6 - **2001:db8:1::fffe**

- Un serveur DNS IPv4 - **192.0.2.200**
- Un serveur DNS IPv6 - **2001:db8:1::ffbb**
- Un domaine de recherche DNS - **example.com**

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Un périphérique Ethernet physique ou virtuel existe dans la configuration du serveur.
- Les nœuds gérés utilisent NetworkManager pour configurer le réseau.

Procédure

1. Créez un fichier playbook, par exemple ***~/ethernet-static-IP.yml***avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with static IP
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: example
          match:
            path:
              - pci-0000:00:0[1-3].0
              - &!pci-0000:00:02.0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 192.0.2.1/24
              - 2001:db8:1::1/64
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
```

```
      dns_search:
        - example.com
      state: up
```

Le paramètre **match** dans cet exemple définit qu'Ansible applique la pièce aux périphériques qui correspondent à l'ID PCI **0000:00:0[1-3].0**, mais pas à **0000:00:02.0**. Pour plus de détails sur les modificateurs spéciaux et les jokers que vous pouvez utiliser, consultez la description du paramètre **match** dans le fichier **/usr/share/ansible/roles/rhel-system-roles.network/README.md**.

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/ethernet-static-IP.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.3. CONFIGURATION D'UNE CONNEXION ETHERNET AVEC UNE ADRESSE IP DYNAMIQUE EN UTILISANT LE RÔLE DE SYSTÈME RHEL AVEC UN NOM D'INTERFACE

Vous pouvez configurer à distance une connexion Ethernet à l'aide du rôle de système RHEL **network**. Pour les connexions avec des paramètres d'adresse IP dynamiques, NetworkManager demande les paramètres IP pour la connexion à partir d'un serveur DHCP.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Un périphérique Ethernet physique ou virtuel existe dans la configuration du serveur.
- Un serveur DHCP est disponible dans le réseau
- Les nœuds gérés utilisent NetworkManager pour configurer le réseau.

Procédure

1. Créez un fichier playbook, par exemple ***~/ethernet-dynamic-IP.yml***avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
```

```
tasks:
- name: Configure an Ethernet connection with dynamic IP
  include_role:
    name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp7s0
        interface_name: enp7s0
        type: ethernet
        autoconnect: yes
        ip:
          dhcp4: yes
          auto6: yes
        state: up
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/ethernet-dynamic-IP.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.4. CONFIGURATION D'UNE CONNEXION ETHERNET AVEC UNE ADRESSE IP DYNAMIQUE EN UTILISANT LE RÔLE DE SYSTÈME RHEL AVEC UN CHEMIN D'ACCÈS DE PÉRIPHÉRIQUE

Vous pouvez configurer à distance une connexion Ethernet à l'aide du rôle de système RHEL **network**. Pour les connexions avec des paramètres d'adresse IP dynamiques, NetworkManager demande les paramètres IP pour la connexion à partir d'un serveur DHCP.

Vous pouvez identifier le chemin d'accès à l'appareil à l'aide de la commande suivante :

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Un périphérique Ethernet physique ou virtuel existe dans la configuration du serveur.
- Un serveur DHCP est disponible dans le réseau.

- Les hôtes gérés utilisent NetworkManager pour configurer le réseau.

Procédure

1. Créez un fichier playbook, par exemple ***~/ethernet-dynamic-IP.yml***avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with dynamic IP
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: example
          match:
            path:
              - pci-0000:00:0[1-3].0
              - &!pci-0000:00:02.0
          type: ethernet
          autoconnect: yes
          ip:
            dhcp4: yes
            auto6: yes
          state: up
```

   Le paramètre **match** dans cet exemple définit qu'Ansible applique la pièce aux périphériques qui correspondent à l'ID PCI **0000:00:0[1-3].0**, mais pas à **0000:00:02.0**. Pour plus de détails sur les modificateurs spéciaux et les jokers que vous pouvez utiliser, consultez la description du paramètre **match** dans le fichier **/usr/share/ansible/roles/rhel-system-roles.network/README.md**.

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/ethernet-dynamic-IP.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.5. CONFIGURATION DU MARQUAGE VLAN À L'AIDE DU RÔLE DE SYSTÈME RHEL DU RÉSEAU

Vous pouvez utiliser le rôle de système **network** RHEL pour configurer le marquage VLAN. Cet exemple ajoute une connexion Ethernet et un VLAN avec l'ID **10** au-dessus de cette connexion Ethernet. En tant que périphérique enfant, la connexion VLAN contient l'IP, la passerelle par défaut et les configurations DNS.

En fonction de votre environnement, ajustez le jeu en conséquence. Par exemple :

- Pour utiliser le VLAN comme port dans d'autres connexions, telles qu'un lien, omettre l'attribut **ip** et définir la configuration IP dans la configuration enfant.

- Pour utiliser des périphériques team, bridge ou bond dans le VLAN, adaptez les attributs **interface_name** et **type** des ports que vous utilisez dans le VLAN.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/vlan-ethernet.yml*** avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure a VLAN that uses an Ethernet connection
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        # Add an Ethernet profile for the underlying device of the VLAN
        - name: enp1s0
          type: ethernet
          interface_name: enp1s0
          autoconnect: yes
          state: up
          ip:
            dhcp4: no
            auto6: no

        # Define the VLAN profile
        - name: enp1s0.10
          type: vlan
          ip:
            address:
              - "192.0.2.1/24"
              - "2001:db8:1::1/64"
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
```

```
      vlan_id: 10
      parent: enp1s0
      state: up
```

L'attribut **parent** du profil VLAN configure le VLAN pour qu'il fonctionne au-dessus du dispositif **enp1s0**.

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/vlan-ethernet.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.6. CONFIGURATION D'UN PONT RÉSEAU À L'AIDE DU RÔLE DE SYSTÈME RHEL DE RÉSEAU

Vous pouvez utiliser le rôle système **network** RHEL pour configurer un pont Linux. Par exemple, vous pouvez l'utiliser pour configurer un pont réseau qui utilise deux périphériques Ethernet et définit les adresses IPv4 et IPv6, les passerelles par défaut et la configuration DNS.

NOTE

Définir la configuration IP sur le pont et non sur les ports du pont Linux.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Deux ou plusieurs périphériques réseau physiques ou virtuels sont installés sur le serveur.

Procédure

1. Créez un fichier playbook, par exemple ***~/bridge-ethernet.yml***avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure a network bridge that uses two Ethernet ports
    include_role:
```

```
    name: rhel-system-roles.network

  vars:
    network_connections:
      # Define the bridge profile
      - name: bridge0
        type: bridge
        interface_name: bridge0
        ip:
          address:
            - "192.0.2.1/24"
            - "2001:db8:1::1/64"
          gateway4: 192.0.2.254
          gateway6: 2001:db8:1::fffe
          dns:
            - 192.0.2.200
            - 2001:db8:1::ffbb
          dns_search:
            - example.com
        state: up

      # Add an Ethernet profile to the bridge
      - name: bridge0-port1
        interface_name: enp7s0
        type: ethernet
        controller: bridge0
        port_type: bridge
        state: up

      # Add a second Ethernet profile to the bridge
      - name: bridge0-port2
        interface_name: enp8s0
        type: ethernet
        controller: bridge0
        port_type: bridge
        state: up
```

2. Exécutez le manuel de jeu :

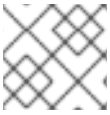
```
# ansible-playbook ~/bridge-ethernet.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.7. CONFIGURATION D'UNE LIAISON RÉSEAU À L'AIDE DU RÔLE DE SYSTÈME RHEL RÉSEAU

Vous pouvez utiliser le site **network** RHEL System Roles pour configurer un lien Linux. Par exemple, vous pouvez l'utiliser pour configurer un lien réseau en mode de sauvegarde active qui utilise deux périphériques Ethernet et définit des adresses IPv4 et IPv6, des passerelles par défaut et une configuration DNS.



NOTE

Définir la configuration IP sur le lien et non sur les ports du lien Linux.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Deux ou plusieurs périphériques réseau physiques ou virtuels sont installés sur le serveur.

Procédure

1. Créez un fichier playbook, par exemple ***~/bond-ethernet.yml***avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure a network bond that uses two Ethernet ports
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        # Define the bond profile
        - name: bond0
          type: bond
          interface_name: bond0
          ip:
            address:
              - "192.0.2.1/24"
              - "2001:db8:1::1/64"
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
          bond:
            mode: active-backup
          state: up

        # Add an Ethernet profile to the bond
        - name: bond0-port1
```

```
    interface_name: enp7s0
    type: ethernet
    controller: bond0
    state: up

  # Add a second Ethernet profile to the bond
  - name: bond0-port2
    interface_name: enp8s0
    type: ethernet
    controller: bond0
    state: up
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/bond-ethernet.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.8. CONFIGURATION D'UNE CONNEXION IPOIB À L'AIDE DU RÔLE DE RÉSEAU RHEL SYSTEM ROLE

Vous pouvez utiliser le rôle système **network** RHEL pour créer à distance des profils de connexion NetworkManager pour les périphériques IP over InfiniBand (IPoIB). Par exemple, ajoutez à distance une connexion InfiniBand pour l'interface **mlx4_ib0** avec les paramètres suivants en exécutant un playbook Ansible :

- Un dispositif IPoIB - **mlx4_ib0.8002**
- Une clé de partition **p_key** - **0x8002**
- Une adresse statique **IPv4** - **192.0.2.1** avec un masque de sous-réseau **/24**
- Une adresse statique **IPv6** - **2001:db8:1::1** avec un masque de sous-réseau **/64**

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des sélections sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Un périphérique InfiniBand nommé **mlx4_ib0** est installé dans les nœuds gérés.
- Les nœuds gérés utilisent NetworkManager pour configurer le réseau.

Procédure

1. Créez un fichier playbook, par exemple *~/IPoIB.yml*avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure IPoIB
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:

        # InfiniBand connection mlx4_ib0
        - name: mlx4_ib0
          interface_name: mlx4_ib0
          type: infiniband

        # IPoIB device mlx4_ib0.8002 on top of mlx4_ib0
        - name: mlx4_ib0.8002
          type: infiniband
          autoconnect: yes
          infiniband:
            p_key: 0x8002
            transport_mode: datagram
          parent: mlx4_ib0
          ip:
            address:
              - 192.0.2.1/24
              - 2001:db8:1::1/64
          state: up
```

Si vous définissez un paramètre **p_key** comme dans cet exemple, ne définissez pas de paramètre **interface_name** sur le périphérique IPoIB.

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/IPoIB.yml
```

Vérification

1. Sur l'hôte **managed-node-01.example.com**, affichez les paramètres IP du périphérique **mlx4_ib0.8002**:

```
# ip address show mlx4_ib0.8002
...
inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute ib0.8002
   valid_lft forever preferred_lft forever
inet6 2001:db8:1::1/64 scope link tentative noprefixroute
   valid_lft forever preferred_lft forever
```

2. Affichez la clé de partition (P_Key) de l'appareil **mlx4_ib0.8002**:

```
# cat /sys/class/net/mlx4_ib0.8002/pkey
0x8002
```

3. Affiche le mode de l'appareil **mlx4_ib0.8002**:

```
# cat /sys/class/net/mlx4_ib0.8002/mode
datagram
```

Ressources supplémentaires

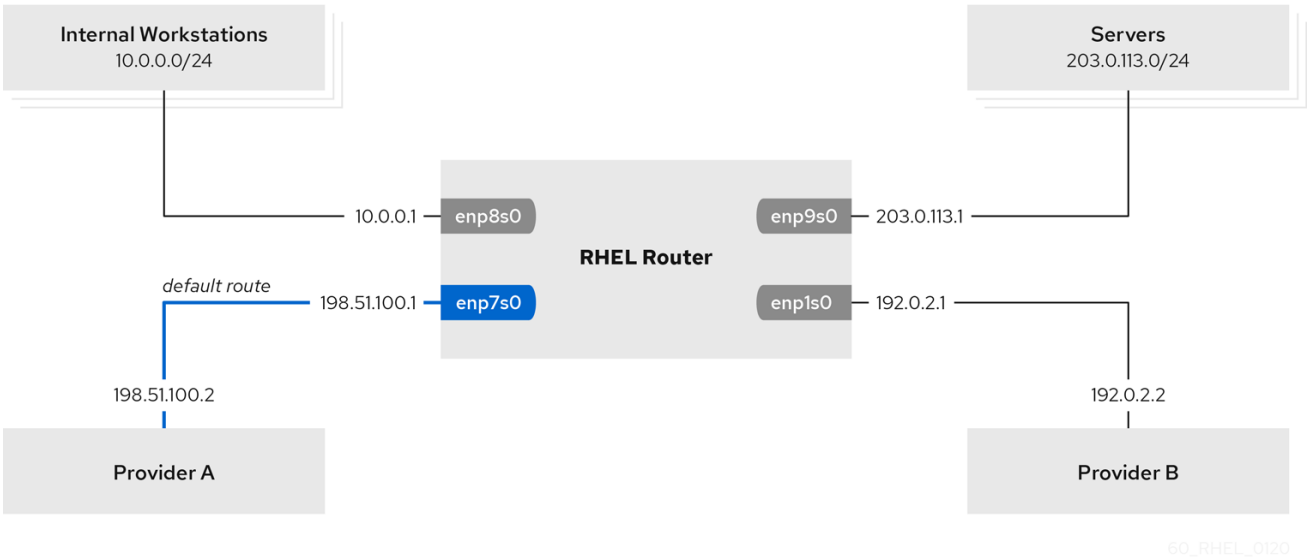
- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.9. ROUTAGE DU TRAFIC D'UN SOUS-RÉSEAU SPÉCIFIQUE VERS UNE PASSERELLE PAR DÉFAUT DIFFÉRENTE EN UTILISANT LE RÔLE DE SYSTÈME RHEL DU RÉSEAU

Vous pouvez utiliser le routage basé sur des règles pour configurer une passerelle par défaut différente pour le trafic provenant de certains sous-réseaux. Par exemple, vous pouvez configurer RHEL comme un routeur qui, par défaut, achemine tout le trafic vers le fournisseur d'accès Internet A à l'aide de la route par défaut. Toutefois, le trafic reçu du sous-réseau des postes de travail internes est acheminé vers le fournisseur B.

Pour configurer le routage basé sur des stratégies à distance et sur plusieurs nœuds, vous pouvez utiliser le rôle système RHEL **network**. Effectuez cette procédure sur le nœud de contrôle Ansible.

Cette procédure suppose la topologie de réseau suivante :



Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.

- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Les nœuds gérés utilisent les services **NetworkManager** et **firewalld**.
- Les nœuds gérés que vous souhaitez configurer possèdent quatre interfaces réseau :
  - L'interface **enp7s0** est connectée au réseau du fournisseur A. L'IP de la passerelle dans le réseau du fournisseur est **198.51.100.2**, et le réseau utilise un masque de réseau **/30**.
  - L'interface **enp1s0** est connectée au réseau du fournisseur B. L'IP de la passerelle dans le réseau du fournisseur est **192.0.2.2**, et le réseau utilise un masque de réseau **/30**.
  - L'interface **enp8s0** est connectée au sous-réseau **10.0.0.0/24** avec des postes de travail internes.
  - L'interface **enp9s0** est connectée au sous-réseau **203.0.113.0/24** où se trouvent les serveurs de l'entreprise.
- Les hôtes du sous-réseau des postes de travail internes utilisent **10.0.0.1** comme passerelle par défaut. Dans la procédure, vous attribuez cette adresse IP à l'interface réseau **enp8s0** du routeur.
- Les hôtes du sous-réseau du serveur utilisent **203.0.113.1** comme passerelle par défaut. Dans la procédure, vous attribuez cette adresse IP à l'interface réseau **enp9s0** du routeur.

Procédure

1. Créez un fichier playbook, par exemple **~/pbr.yml**, avec le contenu suivant :

```
---
- name: Configuring policy-based routing
  hosts: managed-node-01.example.com
  tasks:
  - name: Routing traffic from a specific subnet to a different default gateway
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: Provider-A
          interface_name: enp7s0
          type: ethernet
          autoconnect: True
          ip:
            address:
              - 198.51.100.1/30
            gateway4: 198.51.100.2
            dns:
              - 198.51.100.200
          state: up
          zone: external

        - name: Provider-B
```

```
    interface_name: enp1s0
    type: ethernet
    autoconnect: True
    ip:
      address:
        - 192.0.2.1/30
      route:
        - network: 0.0.0.0
          prefix: 0
          gateway: 192.0.2.2
          table: 5000
    state: up
    zone: external

  - name: Internal-Workstations
    interface_name: enp8s0
    type: ethernet
    autoconnect: True
    ip:
      address:
        - 10.0.0.1/24
      route:
        - network: 10.0.0.0
          prefix: 24
          table: 5000
      routing_rule:
        - priority: 5
          from: 10.0.0.0/24
          table: 5000
    state: up
    zone: trusted

  - name: Servers
    interface_name: enp9s0
    type: ethernet
    autoconnect: True
    ip:
      address:
        - 203.0.113.1/24
    state: up
    zone: trusted
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/pbr.yml
```

Vérification

1. Sur un hôte RHEL dans le sous-réseau des stations de travail internes :
   a. Installez le paquetage **traceroute**:

```
# dnf install traceroute
```

   b. Utilisez l'utilitaire **traceroute** pour afficher l'itinéraire vers un hôte sur Internet :

```
# traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)     0.337 ms  0.260 ms  0.223 ms
 2  192.0.2.1 (192.0.2.1)   0.884 ms  1.066 ms  1.248 ms
 ...
```

La sortie de la commande indique que le routeur envoie des paquets sur **192.0.2.1**, qui est le réseau du fournisseur B.

2. Sur un hôte RHEL dans le sous-réseau du serveur :

   a. Installez le paquetage **traceroute**:

   ```
   # dnf install traceroute
   ```

   b. Utilisez l'utilitaire **traceroute** pour afficher l'itinéraire vers un hôte sur Internet :

   ```
   # traceroute redhat.com
   traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
    1  203.0.113.1 (203.0.113.1)    2.179 ms  2.073 ms  1.944 ms
    2  198.51.100.2 (198.51.100.2)  1.868 ms  1.798 ms  1.549 ms
    ...
   ```

   La sortie de la commande indique que le routeur envoie des paquets sur **198.51.100.2**, qui est le réseau du fournisseur A.

3. Sur le routeur RHEL que vous avez configuré à l'aide du rôle de système RHEL :

   a. Affichez la liste des règles :

   ```
   # ip rule list
   0:      from all lookup local
   5:    from 10.0.0.0/24 lookup 5000
   32766:  from all lookup main
   32767:  from all lookup default
   ```

   Par défaut, RHEL contient des règles pour les tables **local**, **main**, et **default**.

   b. Afficher les itinéraires dans la table **5000**:

   ```
   # ip route list table 5000
   0.0.0.0/0 via 192.0.2.2 dev enp1s0 proto static metric 100
   10.0.0.0/24 dev enp8s0 proto static scope link src 192.0.2.1 metric 102
   ```

   c. Affichez les interfaces et les zones de pare-feu :

   ```
   # firewall-cmd --get-active-zones
   external
     interfaces: enp1s0 enp7s0
   trusted
     interfaces: enp8s0 enp9s0
   ```

   d. Vérifiez que le masquage est activé dans la zone **external**:

```
# firewall-cmd --info-zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0 enp7s0
  sources:
  services: ssh
  ports:
  protocols:
  masquerade: yes
  ...
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.10. CONFIGURATION D'UNE CONNEXION ETHERNET STATIQUE AVEC AUTHENTIFICATION RÉSEAU 802.1X À L'AIDE DU RÔLE DE SYSTÈME RHEL RÉSEAU

À l'aide du rôle système **network** RHEL, vous pouvez automatiser la création d'une connexion Ethernet qui utilise la norme 802.1X pour authentifier le client. Par exemple, ajoutez à distance une connexion Ethernet pour l'interface **enp1s0** avec les paramètres suivants en exécutant un script Ansible :

- Une adresse IPv4 statique - **192.0.2.1** avec un masque de sous-réseau **/24**
- Une adresse IPv6 statique - **2001:db8:1::1** avec un masque de sous-réseau **/64**
- Une passerelle par défaut IPv4 - **192.0.2.254**
- Une passerelle par défaut IPv6 - **2001:db8:1::fffe**
- Un serveur DNS IPv4 - **192.0.2.200**
- Un serveur DNS IPv6 - **2001:db8:1::ffbb**
- Un domaine de recherche DNS - **example.com**
- 802.1X authentification réseau utilisant le protocole d'authentification extensible (EAP) **TLS**

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible

- Le réseau prend en charge l'authentification réseau 802.1X.
- Les nœuds gérés utilisent NetworkManager.
- Les fichiers suivants, nécessaires à l'authentification TLS, existent sur le nœud de contrôle :
  - La clé du client est stockée dans le fichier **/srv/data/client.key**.
  - Le certificat du client est stocké dans le fichier **/srv/data/client.crt**.
  - Le certificat de l'autorité de certification (CA) est stocké dans le fichier **/srv/data/ca.crt**.

Procédure

1. Créez un fichier playbook, par exemple ***~/enable-802.1x.yml***avec le contenu suivant :

```
---
- name: Configure an Ethernet connection with 802.1X authentication
  hosts: managed-node-01.example.com
  tasks:
    - name: Copy client key for 802.1X authentication
      copy:
        src: "/srv/data/client.key"
        dest: "/etc/pki/tls/private/client.key"
        mode: 0600

    - name: Copy client certificate for 802.1X authentication
      copy:
        src: "/srv/data/client.crt"
        dest: "/etc/pki/tls/certs/client.crt"

    - name: Copy CA certificate for 802.1X authentication
      copy:
        src: "/srv/data/ca.crt"
        dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

    - include_role:
        name: rhel-system-roles.network

      vars:
        network_connections:
          - name: enp1s0
            type: ethernet
            autoconnect: yes
            ip:
              address:
                - 192.0.2.1/24
                - 2001:db8:1::1/64
              gateway4: 192.0.2.254
              gateway6: 2001:db8:1::fffe
              dns:
                - 192.0.2.200
                - 2001:db8:1::ffbb
              dns_search:
                - example.com
            ieee802_1x:
```

```
      identity: user_name
      eap: tls
      private_key: "/etc/pki/tls/private/client.key"
      private_key_password: "password"
      client_cert: "/etc/pki/tls/certs/client.crt"
      ca_cert: "/etc/pki/ca-trust/source/anchors/ca.crt"
      domain_suffix_match: example.com
    state: up
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/enable-802.1x.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.11. CONFIGURER UNE CONNEXION WIFI AVEC L'AUTHENTIFICATION RÉSEAU 802.1X EN UTILISANT LE RÔLE RÉSEAU RHEL SYSTEM ROLE

À l'aide des rôles système RHEL, vous pouvez automatiser la création d'une connexion wifi. Par exemple, vous pouvez ajouter à distance un profil de connexion sans fil pour l'interface **wlp1s0** à l'aide d'un playbook Ansible. Le profil créé utilise la norme 802.1X pour authentifier le client sur un réseau wifi. Le playbook configure le profil de connexion pour utiliser DHCP. Pour configurer des paramètres IP statiques, adaptez les paramètres du dictionnaire **ip** en conséquence.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.
- Le réseau prend en charge l'authentification réseau 802.1X.
- Vous avez installé le paquetage **wpa_supplicant** sur le nœud géré.
- DHCP est disponible dans le réseau du nœud géré.
- Les fichiers suivants, nécessaires à l'authentification TLS, existent sur le nœud de contrôle :
  - La clé du client est stockée dans le fichier **/srv/data/client.key**.
  - Le certificat du client est stocké dans le fichier **/srv/data/client.crt**.
  - Le certificat d'autorité de certification est stocké dans le fichier **/srv/data/ca.crt**.

Procédure

1. Créez un fichier playbook, par exemple *~/**enable-802.1x.yml*** avec le contenu suivant :

```
---
- name: Configure a wifi connection with 802.1X authentication
  hosts: managed-node-01.example.com
  tasks:
    - name: Copy client key for 802.1X authentication
      copy:
        src: "/srv/data/client.key"
        dest: "/etc/pki/tls/private/client.key"
        mode: 0400

    - name: Copy client certificate for 802.1X authentication
      copy:
        src: "/srv/data/client.crt"
        dest: "/etc/pki/tls/certs/client.crt"

    - name: Copy CA certificate for 802.1X authentication
      copy:
        src: "/srv/data/ca.crt"
        dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

    - block:
        - import_role:
            name: linux-system-roles.network
          vars:
            network_connections:
              - name: Configure the Example-wifi profile
                interface_name: wlp1s0
                state: up
                type: wireless
                autoconnect: yes
                ip:
                  dhcp4: true
                  auto6: true
                wireless:
                  ssid: "Example-wifi"
                  key_mgmt: "wpa-eap"
                ieee802_1x:
                  identity: "user_name"
                  eap: tls
                  private_key: "/etc/pki/tls/client.key"
                  private_key_password: "password"
                  private_key_password_flags: none
                  client_cert: "/etc/pki/tls/client.pem"
                  ca_cert: "/etc/pki/tls/cacert.pem"
                  domain_suffix_match: "example.com"
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/enable-802.1x.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.12. DÉFINITION DE LA PASSERELLE PAR DÉFAUT SUR UNE CONNEXION EXISTANTE À L'AIDE DU RÔLE DE RÉSEAU RHEL SYSTEM ROLE

Vous pouvez utiliser le rôle de système **network** RHEL pour définir la passerelle par défaut.

IMPORTANT

Lorsque vous exécutez une séquence qui utilise le rôle système **network** RHEL, le rôle système remplace un profil de connexion existant portant le même nom si la valeur des paramètres ne correspond pas à ceux spécifiés dans la séquence. Par conséquent, indiquez toujours la configuration complète du profil de connexion réseau dans la pièce, même si, par exemple, la configuration IP existe déjà. Dans le cas contraire, le rôle rétablit les valeurs par défaut.

Selon qu'il existe déjà ou non, la procédure crée ou met à jour le profil de connexion **enp1s0** avec les paramètres suivants :

- Une adresse IPv4 statique - **198.51.100.20** avec un masque de sous-réseau **/24**
- Une adresse IPv6 statique - **2001:db8:1::1** avec un masque de sous-réseau **/64**
- Une passerelle par défaut IPv4 - **198.51.100.254**
- Une passerelle par défaut IPv6 - **2001:db8:1::fffe**
- Un serveur DNS IPv4 - **198.51.100.200**
- Un serveur DNS IPv6 - **2001:db8:1::ffbb**
- Un domaine de recherche DNS - **example.com**

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/ethernet-connection.yml***avec le contenu suivant :

```
---
```

```
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with static IP and default gateway
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp1s0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 198.51.100.20/24
              - 2001:db8:1::1/64
            gateway4: 198.51.100.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 198.51.100.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
          state: up
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/ethernet-connection.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.13. CONFIGURATION D'UNE ROUTE STATIQUE À L'AIDE DU RÔLE RÉSEAU RHEL SYSTEM ROLE

Vous pouvez utiliser le rôle de système **network** RHEL pour configurer des itinéraires statiques.

IMPORTANT

Lorsque vous exécutez une séquence qui utilise le rôle système **network** RHEL, le rôle système remplace un profil de connexion existant portant le même nom si la valeur des paramètres ne correspond pas à ceux spécifiés dans la séquence. Par conséquent, indiquez toujours la configuration complète du profil de connexion réseau dans la pièce, même si, par exemple, la configuration IP existe déjà. Dans le cas contraire, le rôle rétablit les valeurs par défaut.

Selon qu'il existe déjà ou non, la procédure crée ou met à jour le profil de connexion **enp7s0** avec les paramètres suivants :

- Une adresse IPv4 statique - **192.0.2.1** avec un masque de sous-réseau **/24**
- Une adresse IPv6 statique - **2001:db8:1::1** avec un masque de sous-réseau **/64**

- Une passerelle par défaut IPv4 - **192.0.2.254**
- Une passerelle par défaut IPv6 - **2001:db8:1::fffe**
- Un serveur DNS IPv4 - **192.0.2.200**
- Un serveur DNS IPv6 - **2001:db8:1::ffbb**
- Un domaine de recherche DNS - **example.com**
- Routes statiques :
  - **198.51.100.0/24** avec passerelle **192.0.2.10**
  - **2001:db8:2::/64** avec passerelle **2001:db8:1::10**

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/add-static-routes.yml***avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with static IP and additional routes
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp7s0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 192.0.2.1/24
              - 2001:db8:1::1/64
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
```

```
      - 2001:db8:1::ffbb
    dns_search:
      - example.com
    route:
      - network: 198.51.100.0
        prefix: 24
        gateway: 192.0.2.10
      - network: 2001:db8:2::
        prefix: 64
        gateway: 2001:db8:1::10
  state: up
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/add-static-routes.yml
```

Vérification

1. Sur les nœuds gérés :
   a. Afficher les itinéraires IPv4 :

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp7s0
```

   b. Afficher les itinéraires IPv6 :

```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp7s0 metric 1024 pref medium
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.14. CONFIGURATION D'UNE FONCTION DE DÉLESTAGE ETHTOOL À L'AIDE DU RÔLE DE SYSTÈME RHEL DE RÉSEAU

Vous pouvez utiliser le rôle de système **network** RHEL pour configurer les fonctionnalités **ethtool** d'une connexion NetworkManager.

IMPORTANT

Lorsque vous exécutez une séquence qui utilise le rôle système **network** RHEL, le rôle système remplace un profil de connexion existant portant le même nom si la valeur des paramètres ne correspond pas à ceux spécifiés dans la séquence. Par conséquent, indiquez toujours la configuration complète du profil de connexion réseau dans la pièce, même si, par exemple, la configuration IP existe déjà. Dans le cas contraire, le rôle rétablit les valeurs par défaut.

Selon qu'il existe déjà ou non, la procédure crée ou met à jour le profil de connexion **enp1s0** avec les paramètres suivants :

- Une adresse statique **IPv4** - **198.51.100.20** avec un masque de sous-réseau **/24**
- Une adresse statique **IPv6** - **2001:db8:1::1** avec un masque de sous-réseau **/64**
- Une passerelle par défaut **IPv4** - **198.51.100.254**
- Une passerelle par défaut **IPv6** - **2001:db8:1::fffe**
- Un serveur DNS **IPv4** - **198.51.100.200**
- Un serveur DNS **IPv6** - **2001:db8:1::ffbb**
- Un domaine de recherche DNS - **example.com**
- **ethtool** caractéristiques :
  - Déchargement générique de la réception (GRO) : désactivé
  - Délestage générique de segmentation (GSO) : activé
  - Segmentation du protocole de transmission de contrôle de flux TX (SCTP) : désactivée

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/configure-ethernet-device-with-ethtool-features.yml***avec le contenu suivant :

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with ethtool features
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp1s0
          type: ethernet
          autoconnect: yes
          ip:
            address:
```

```
      - 198.51.100.20/24
      - 2001:db8:1::1/64
    gateway4: 198.51.100.254
    gateway6: 2001:db8:1::fffe
    dns:
      - 198.51.100.200
      - 2001:db8:1::ffbb
    dns_search:
      - example.com
  ethtool:
    features:
      gro: "no"
      gso: "yes"
      tx_sctp_segmentation: "no"
  state: up
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/configure-ethernet-device-with-ethtool-features.yml
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

8.15. ÉTATS DU RÉSEAU POUR LE RÔLE DE SYSTÈME RHEL

Le rôle système **network** RHEL prend en charge les configurations d'état dans les playbooks pour configurer les périphériques. Pour ce faire, utilisez la variable **network_state** suivie des configurations d'état.

Avantages de l'utilisation de la variable **network_state** dans un playbook :

- En utilisant la méthode déclarative avec les configurations d'état, vous pouvez configurer des interfaces et le NetworkManager crée un profil pour ces interfaces en arrière-plan.
- Avec la variable **network_state**, vous pouvez spécifier les options que vous souhaitez modifier, et toutes les autres options resteront inchangées. En revanche, avec la variable **network_connections**, vous devez spécifier tous les paramètres pour modifier le profil de connexion au réseau.

Par exemple, pour créer une connexion Ethernet avec des paramètres d'adresse IP dynamiques, utilisez le bloc **vars** suivant dans votre playbook :

Playbook avec configurations d'état	Manuel de jeu régulier

```
vars:
  network_state:
    interfaces:
    - name: enp7s0
      type: ethernet
      state: up
      ipv4:
        enabled: true
        auto-dns: true
        auto-gateway: true
        auto-routes: true
        dhcp: true
      ipv6:
        enabled: true
        auto-dns: true
        auto-gateway: true
        auto-routes: true
        autoconf: true
        dhcp: true
```

```
vars:
  network_connections:
    - name: enp7s0
      interface_name: enp7s0
      type: ethernet
      autoconnect: yes
      ip:
        dhcp4: yes
        auto6: yes
      state: up
```

Par exemple, pour modifier uniquement l'état de connexion des paramètres d'adresse IP dynamique que vous avez créés comme indiqué ci-dessus, utilisez le bloc **vars** suivant dans votre playbook :

Playbook avec configurations d'état

```
vars:
  network_state:
    interfaces:
    - name: enp7s0
      type: ethernet
      state: down
```

Manuel de jeu régulier

```
vars:
  network_connections:
    - name: enp7s0
      interface_name: enp7s0
      type: ethernet
      autoconnect: yes
      ip:
        dhcp4: yes
        auto6: yes
      state: down
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** fichier

CHAPITRE 9. CONFIGURATION DE FIREWALLD À L'AIDE DES RÔLES DE SYSTÈME

Vous pouvez utiliser le rôle système **firewall** pour configurer les paramètres du service **firewalld** sur plusieurs clients à la fois. Cette solution :

- Fournit une interface avec des paramètres d'entrée efficaces.
- Conserve tous les paramètres de **firewalld** en un seul endroit.

Après avoir exécuté le rôle **firewall** sur le nœud de contrôle, le rôle système applique immédiatement les paramètres **firewalld** au nœud géré et les rend persistants lors des redémarrages.

9.1. INTRODUCTION AU RÔLE DU SYSTÈME RHEL FIREWALL

RHEL System Roles est un ensemble de contenus pour l'utilitaire d'automatisation Ansible. Ce contenu, associé à l'utilitaire d'automatisation Ansible, fournit une interface de configuration cohérente pour gérer à distance plusieurs systèmes.

Le rôle **rhel-system-roles.firewall** des rôles système RHEL a été introduit pour les configurations automatisées du service **firewalld**. Le paquetage **rhel-system-roles** contient ce rôle système, ainsi que la documentation de référence.

Pour appliquer les paramètres **firewalld** à un ou plusieurs systèmes de manière automatisée, utilisez la variable **firewall** System Role dans un cahier de jeu. Un cahier de jeu est une liste d'un ou plusieurs jeux écrits au format YAML basé sur le texte.

Vous pouvez utiliser un fichier d'inventaire pour définir un ensemble de systèmes qu'Ansible doit configurer.

Le rôle **firewall** vous permet de configurer de nombreux paramètres **firewalld**, par exemple :

- Zones.
- Les services pour lesquels les paquets doivent être autorisés.
- Octroi, rejet ou abandon de l'accès du trafic aux ports.
- Transfert de ports ou de plages de ports pour une zone.

Ressources supplémentaires

- **README.md** et **README.html** dans le répertoire **/usr/share/doc/rhel-system-roles/firewall/**
- Travailler avec des playbooks
- Comment constituer votre inventaire

9.2. RÉINITIALISATION DES PARAMÈTRES DE FIREWALLD À L'AIDE DU RÔLE DE SYSTÈME RHEL DU PARE-FEU

Avec le rôle système **firewall** RHEL, vous pouvez réinitialiser les paramètres **firewalld** à leur état par défaut. Si vous ajoutez le paramètre **previous:replaced** à la liste des variables, le rôle système supprime tous les paramètres existants définis par l'utilisateur et rétablit les valeurs par défaut de **firewalld**. Si

vous combinez le paramètre **previous:replaced** avec d'autres paramètres, le rôle **firewall** supprime tous les paramètres existants avant d'en appliquer de nouveaux.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/reset-firewalld.yml***avec le contenu suivant :

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
  - name: Reset firewalld
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - previous: replaced
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/configuring-a-dmz.yml
```

Vérification

- Exécutez cette commande en tant que **root** sur le nœud géré pour vérifier toutes les zones :

```
# firewall-cmd --list-all-zones
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**
- **ansible-playbook(1)**
- **firewalld(1)**

9.3. TRANSFÉRER LE TRAFIC ENTRANT D'UN PORT LOCAL VERS UN AUTRE PORT LOCAL

Le rôle **firewall** vous permet de configurer à distance les paramètres **firewalld** avec un effet persistant sur plusieurs hôtes gérés.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/port_forwarding.yml***avec le contenu suivant :

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Forward incoming traffic on port 8080 to 443
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/port_forwarding.yml
```

Vérification

- Sur l'hôte géré, affichez les paramètres **firewalld**:

```
# firewall-cmd --list-forward-ports
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

9.4. CONFIGURATION DES PORTS À L'AIDE DES RÔLES DE SYSTÈME

Vous pouvez utiliser le rôle système RHEL **firewall** pour ouvrir ou fermer les ports du pare-feu local pour le trafic entrant et faire en sorte que la nouvelle configuration persiste lors des redémarrages. Par exemple, vous pouvez configurer la zone par défaut pour autoriser le trafic entrant pour le service HTTPS.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/opening-a-port.yml***avec le contenu suivant :

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Allow incoming HTTPS traffic to the local host
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - port: 443/tcp
          service: http
          state: enabled
          runtime: true
          permanent: true
```

   L'option **permanent: true** rend les nouveaux paramètres persistants à travers les redémarrages.

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/opening-a-port.yml
```

Vérification

- Sur le nœud géré, vérifiez que le port **443/tcp** associé au service **HTTPS** est ouvert :

```
# firewall-cmd --list-ports
443/tcp
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

9.5. CONFIGURATION D'UNE ZONE DMZ FIREWALLD À L'AIDE DU RÔLE DE SYSTÈME FIREWALLD RHEL

En tant qu'administrateur système, vous pouvez utiliser le rôle système **firewall** pour configurer une zone **dmz** sur l'interface **enp1s0** afin d'autoriser le trafic **HTTPS** vers la zone. Vous permettez ainsi à des utilisateurs externes d'accéder à vos serveurs web.

Effectuez cette procédure sur le nœud de contrôle Ansible.

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo** sur ces nœuds.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un fichier playbook, par exemple ***~/configuring-a-dmz.yml***avec le contenu suivant :

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - zone: dmz
          interface: enp1s0
          service: https
          state: enabled
          runtime: true
          permanent: true
```

2. Exécutez le manuel de jeu :

```
# ansible-playbook ~/configuring-a-dmz.yml
```

Vérification

- Sur le nœud géré, affichez des informations détaillées sur la zone **dmz**:

```
# firewall-cmd --zone=dmz --list-all
```

```
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: https ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
```

Ressources supplémentaires

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

CHAPITRE 10. VARIABLES DU RÔLE POSTFIX DANS RÔLES DU SYSTÈME

Les variables de rôle **postfix** permettent à l'utilisateur d'installer, de configurer et de démarrer l'agent de transfert de courrier (MTA) **postfix**.

Les variables de rôle suivantes sont définies dans cette section :

- **postfix_conf**: Il comprend des paires clé/valeur de tous les paramètres de configuration pris en charge par **postfix**. Par défaut, l'adresse **postfix_conf** n'a pas de valeur.

```
postfix_conf:
  relayhost: example.com
```

Si votre scénario nécessite la suppression de toute configuration existante et l'application de la configuration souhaitée sur une installation propre de **postfix**, spécifiez l'option **previous: replaced** dans le dictionnaire **postfix_conf**:

Exemple avec l'option **previous: replaced**:

```
postfix_conf:
  relayhost: example.com
  previous: replaced
```

- **postfix_check**: Il détermine si un contrôle a été exécuté avant le démarrage de **postfix** pour vérifier les changements de configuration. La valeur par défaut est true.

Par exemple :

```
postfix_check: true
```

- **postfix_backup**: Il détermine si une seule copie de sauvegarde de la configuration est créée. Par défaut, la valeur de **postfix_backup** est false.

Pour écraser toute sauvegarde précédente, exécutez la commande suivante :

```
# *cp /etc/postfix/main.cf /etc/postfix/main.cf.backup*
```

Si la valeur **postfix_backup** est modifiée en **true**, vous devez également attribuer la valeur false à la valeur **postfix_backup_multiple**.

Par exemple :

```
postfix_backup: true
postfix_backup_multiple: false
```

- **postfix_backup_multiple**: Il détermine si le rôle effectuera une copie de sauvegarde horodatée de la configuration.

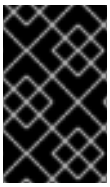
Pour conserver plusieurs copies de sauvegarde, exécutez la commande suivante :

```
# *cp /etc/postfix/main.cf /etc/postfix/main.cf.$(date -Isec)*
```

Par défaut, la valeur de **postfix_backup_multiple** est vraie. Le paramètre **postfix_backup_multiple:true** est prioritaire sur **postfix_backup**. Si vous souhaitez utiliser **postfix_backup**, vous devez définir le paramètre **postfix_backup_multiple:false**.

- **postfix_manage_firewall**: Intègre le rôle **postfix** au rôle **firewall** pour gérer l'accès aux ports. Par défaut, la variable est définie à **false**. Si vous souhaitez gérer automatiquement l'accès aux ports à partir du rôle **postfix**, définissez la variable à **true**.

- **postfix_manage_selinux**: Intègre le rôle **postfix** au rôle **selinux** pour gérer l'accès aux ports. Par défaut, la variable est définie à **false**. Si vous souhaitez gérer automatiquement l'accès aux ports à partir du rôle **postfix**, définissez la variable à **true**.



IMPORTANT

Les paramètres de configuration ne peuvent pas être supprimés. Avant d'exécuter le rôle **postfix**, définissez tous les paramètres de configuration requis pour **postfix_conf** et utilisez le module de fichier pour supprimer **/etc/postfix/main.cf**.

10.1. RESSOURCES SUPPLÉMENTAIRES

- **/usr/share/doc/rhel-system-roles/postfix/README.md**

CHAPITRE 11. CONFIGURATION DE SELINUX À L'AIDE DES RÔLES SYSTÈME

11.1. INTRODUCTION AU RÔLE DU SYSTÈME SELINUX

RHEL System Roles est une collection de rôles et de modules Ansible qui fournissent une interface de configuration cohérente pour gérer à distance plusieurs systèmes RHEL. Le rôle de système **selinux** permet les actions suivantes :

- Nettoyage des modifications de politiques locales liées aux booléens SELinux, aux contextes de fichiers, aux ports et aux connexions.
- Définition des booléens de la politique SELinux, des contextes de fichiers, des ports et des connexions.
- Restauration des contextes de fichiers sur les fichiers ou répertoires spécifiés.
- Gestion des modules SELinux.

Le tableau suivant donne un aperçu des variables d'entrée disponibles dans le rôle de système **selinux**.

Tableau 11.1. selinux Variables de rôle du système

Variable de rôle	Description	Alternative à l'interface de programmation
politique_sélinux	Choix d'une politique de protection des processus ciblés ou d'une protection de sécurité multi-niveaux.	**SELINUXTYPE** en **/etc/selinux/config**
état_sélinux	Change les modes SELinux.	**setenforce** et **SELINUX** dans **/etc/selinux/config**.
booléens_sélinux	Active et désactive les booléens SELinux.	**setsebool**
selinux_fcontexts	Ajoute ou supprime une correspondance de contexte de fichier SELinux.	**semanage fcontext**
selinux_restore_dirs	Rétablit les étiquettes SELinux dans l'arborescence du système de fichiers.	**restorecon -R**
ports_sélinux	Définit les étiquettes SELinux sur les ports.	**semanage port**
selinux_logins	Définit les utilisateurs en fonction du mappage des utilisateurs SELinux.	**semanage login**

Variable de rôle	Description	Alternative à l'interface de programmation
selinux_modules	Installe, active, désactive ou supprime les modules SELinux.	**semodule**

L'exemple d'exécution **/usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml** installé par le paquetage **rhel-system-roles** montre comment définir la stratégie ciblée en mode d'exécution. Il applique également plusieurs modifications de la stratégie locale et restaure les contextes de fichiers dans le répertoire **/tmp/test_dir/**.

Pour une référence détaillée sur les variables de rôle **selinux**, installez le paquetage **rhel-system-roles** et consultez les fichiers **README.md** ou **README.html** dans le répertoire **/usr/share/doc/rhel-system-roles/selinux/**.

Ressources supplémentaires

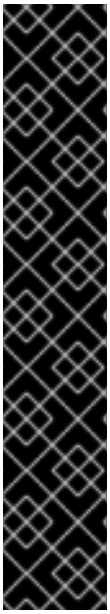
- Introduction aux rôles du système RHEL

11.2. UTILISATION DU RÔLE DE SYSTÈME SELINUX POUR APPLIQUER LES PARAMÈTRES SELINUX À PLUSIEURS SYSTÈMES

Suivez les étapes pour préparer et appliquer un playbook Ansible avec vos paramètres SELinux vérifiés.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **selinux**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.
  - Un fichier d'inventaire qui répertorie les nœuds gérés.



IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Préparez votre playbook. Vous pouvez partir de zéro ou modifier le playbook d'exemple installé avec le paquetage **rhel-system-roles**:

```
# cp /usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml my-selinux-playbook.yml
# vi my-selinux-playbook.yml
```

2. Modifiez le contenu du playbook pour l'adapter à votre scénario. Par exemple, la partie suivante garantit que le système installe et active le module SELinux **selinux-local-1.pp**:

```
selinux_modules:
- { path: "selinux-local-1.pp", priority: "400" }
```

3. Enregistrez les modifications et quittez l'éditeur de texte.

4. Exécutez votre manuel de jeu sur les systèmes *host1*, *host2* et *host3*:

```
# ansible-playbook -i host1,host2,host3 my-selinux-playbook.yml
```

Ressources supplémentaires

- Pour plus d'informations, installez le paquetage **rhel-system-roles** et consultez les répertoires **/usr/share/doc/rhel-system-roles/selinux/** et **/usr/share/ansible/roles/rhel-system-roles.selinux/**.

CHAPITRE 12. CONFIGURATION DE LA JOURNALISATION À L'AIDE DES RÔLES SYSTÈME RHEL

En tant qu'administrateur système, vous pouvez utiliser le rôle système **logging** pour configurer un hôte RHEL en tant que serveur de journalisation afin de collecter les journaux de nombreux systèmes clients.

12.1. LE RÔLE DU SYSTÈME LOGGING

Avec le rôle de système **logging**, vous pouvez déployer des configurations de journalisation sur des hôtes locaux et distants.

Pour appliquer un rôle de système **logging** à un ou plusieurs systèmes, vous définissez la configuration de la journalisation dans un fichier *playbook*. Un cahier de lecture est une liste d'un ou plusieurs jeux. Les carnets de lecture sont lisibles par l'homme et sont écrits au format YAML. Pour plus d'informations sur les carnets de lecture, voir Travailler avec des carnets de lecture dans la documentation Ansible.

L'ensemble des systèmes que vous souhaitez configurer conformément au livre de jeu est défini dans une page *inventory file*. Pour plus d'informations sur la création et l'utilisation d'inventaires, voir Comment construire votre inventaire dans la documentation Ansible.

Les solutions de journalisation offrent plusieurs façons de lire les journaux et plusieurs sorties de journalisation.

Par exemple, un système d'enregistrement peut recevoir les données suivantes :

- les fichiers locaux,
- **systemd/journal**,
- un autre système d'enregistrement sur le réseau.

En outre, un système d'enregistrement peut avoir les résultats suivants :

- les journaux stockés dans les fichiers locaux du répertoire /**var**/**log**,
- les journaux envoyés à Elasticsearch,
- les journaux transmis à un autre système de journalisation.

Avec le rôle de système **logging**, vous pouvez combiner les entrées et les sorties en fonction de votre scénario. Par exemple, vous pouvez configurer une solution de journalisation qui stocke les entrées provenant de **journal** dans un fichier local, tandis que les entrées lues dans les fichiers sont à la fois transmises à un autre système de journalisation et stockées dans les fichiers journaux locaux.

12.2. LOGGING PARAMÈTRES DU RÔLE DU SYSTÈME

Dans un playbook de rôle système **logging**, vous définissez les entrées dans le paramètre **logging_inputs**, les sorties dans le paramètre **logging_outputs** et les relations entre les entrées et les sorties dans le paramètre **logging_flows**. Le rôle de système **logging** traite ces variables avec des options supplémentaires pour configurer le système de journalisation. Vous pouvez également activer le cryptage ou une gestion automatique des ports.

NOTE

Actuellement, le seul système de journalisation disponible dans le rôle de système **logging** est **Rsyslog**.

- **logging_inputs**: Liste des entrées pour la solution d'enregistrement.
  - **name**: Nom unique de l'entrée. Utilisé dans la liste des entrées **logging_flows**: et dans le nom du fichier généré **config**.
  - **type**: Type de l'élément d'entrée. Le type spécifie un type de tâche qui correspond à un nom de répertoire dans **roles/rsyslog/{tasks,vars}/inputs/**.
    - **basics**: Entrées configurant les entrées à partir du journal **systemd** ou de la prise **unix**.
      - **kernel_message**: Charger **imklog** si la valeur est **true**. La valeur par défaut est **false**.
      - **use_imuxsock**: Utilisez **imuxsock** au lieu de **imjournal**. **false** est la valeur par défaut.
      - **ratelimit_burst**: Nombre maximal de messages pouvant être émis dans le cadre de **ratelimit_interval**. La valeur par défaut est **20000** si **use_imuxsock** est faux. La valeur par défaut est **200** si **use_imuxsock** est vrai.
      - **ratelimit_interval**: Intervalle pour évaluer **ratelimit_burst**. La valeur par défaut est de 600 secondes si **use_imuxsock** est faux. La valeur par défaut est 0 si **use_imuxsock** est vrai. 0 indique que la limitation de débit est désactivée.
      - **persist_state_interval**: L'état du journal est conservé tous les **value** messages. La valeur par défaut est **10** et n'est effective que si **use_imuxsock** est faux.
    - **files**: Entrées configuration des entrées à partir de fichiers locaux.
    - **remote**: Entrées configurant les entrées de l'autre système d'enregistrement sur le réseau.
  - **state**: État du fichier de configuration. **present** ou **absent**. La valeur par défaut est **present**.
- **logging_outputs**: Liste des sorties pour la solution d'enregistrement.
  - **files**: Sorties configurer les sorties vers des fichiers locaux.
  - **forwards**: Sorties Configuration des sorties vers un autre système d'enregistrement.
  - **remote_files**: Sorties configurer les sorties d'un autre système de journalisation vers des fichiers locaux.
- **logging_flows**: Liste des flux qui définissent les relations entre **logging_inputs** et **logging_outputs**. La variable **logging_flows** a les clés suivantes :
  - **name**: Nom unique du flux
  - **inputs**: Liste des valeurs du nom **logging_inputs**
  - **outputs**: Liste des valeurs du nom **logging_outputs**.

- **logging_manage_firewall**: Si la valeur est **true**, la variable utilise le rôle **firewall** pour gérer automatiquement l'accès au port à partir du rôle **logging**.
- **logging_manage_selinux**: Si la valeur est **true**, la variable utilise le rôle **selinux** pour gérer automatiquement l'accès au port à partir du rôle **logging**.

Ressources supplémentaires

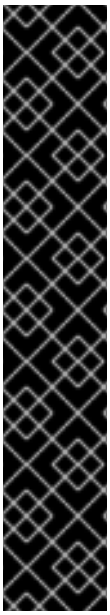
- Documentation installée avec le paquetage **rhel-system-roles** dans **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**

12.3. APPLICATION D'UN RÔLE DE SYSTÈME LOCAL LOGGING

Préparer et appliquer un playbook Ansible pour configurer une solution de journalisation sur un ensemble de machines distinctes. Chaque machine enregistre les journaux localement.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **logging**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

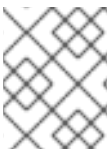


IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.



NOTE

Il n'est pas nécessaire que le paquet **rsyslog** soit installé, car le rôle de système installe **rsyslog** lorsqu'il est déployé.

Procédure

1. Créez un cahier de jeu qui définit le rôle requis :

   a. Créez un nouveau fichier YAML et ouvrez-le dans un éditeur de texte, par exemple :

   ```
   # vi logging-playbook.yml
   ```

   b. Insérer le contenu suivant :

   ```
   ---
   - name: Deploying basics input and implicit files output
     hosts: all
     roles:
       - rhel-system-roles.logging
     vars:
       logging_inputs:
         - name: system_input
           type: basics
       logging_outputs:
         - name: files_output
           type: files
       logging_flows:
         - name: flow1
           inputs: [system_input]
           outputs: [files_output]
   ```

2. Exécuter le manuel de jeu sur un inventaire spécifique :

   ```
   # ansible-playbook -i inventory-file /path/to/file/logging-playbook.yml
   ```

   Où ?

   - ***inventory-file*** est le fichier d'inventaire.
   - ***logging-playbook.yml*** est le manuel de jeu que vous utilisez.

Vérification

1. Tester la syntaxe du fichier **/etc/rsyslog.conf**:

   ```
   # rsyslogd -N 1
   rsyslogd: version 8.1911.0-6.el8, config validation run...
   rsyslogd: End of config validation run. Bye.
   ```

2. Vérifiez que le système envoie des messages au journal :

   a. Envoyer un message test :

   ```
   # logger test
   ```

   b. Consultez le journal **/var/log/messages**, par exemple :

   ```
   # cat /var/log/messages
   Aug  5 13:48:31 hostname root[6778]: test
   ```

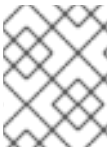
Où `*hostname*` est le nom d'hôte du système client. Notez que le journal contient le nom de l'utilisateur qui a entré la commande de l'enregistreur, dans ce cas **root**.

12.4. FILTRAGE DES JOURNAUX DANS UN SYSTÈME LOCAL LOGGING RÔLE DU SYSTÈME

Vous pouvez déployer une solution de journalisation qui filtre les journaux en fonction du filtre basé sur les propriétés **rsyslog**.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **logging**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Red Hat Ansible Core est installé
  - Le paquet **rhel-system-roles** est installé
  - Un fichier d'inventaire qui répertorie les nœuds gérés.



NOTE

Il n'est pas nécessaire que le paquet **rsyslog** soit installé, car le rôle de système installe **rsyslog** lorsqu'il est déployé.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploying files input and configured files output
  hosts: all
  roles:
    - linux-system-roles.logging
  vars:
    logging_inputs:
      - name: files_input
        type: basics
    logging_outputs:
      - name: files_output0
        type: files
        property: msg
        property_op: contains
        property_value: error
        path: /var/log/errors.log
      - name: files_output1
        type: files
        property: msg
        property_op: "!contains"
        property_value: error
```

```
      path: /var/log/others.log
  logging_flows:
    - name: flow0
      inputs: [files_input]
      outputs: [files_output0, files_output1]
```

Avec cette configuration, tous les messages contenant la chaîne ***error*** sont enregistrés dans **/var/log/errors.log**, et tous les autres messages sont enregistrés dans **/var/log/others.log**.

Vous pouvez remplacer la valeur de la propriété ***error*** par la chaîne de caractères par laquelle vous souhaitez filtrer.

Vous pouvez modifier les variables selon vos préférences.

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

Vérification

1. Tester la syntaxe du fichier **/etc/rsyslog.conf**:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. Vérifiez que le système envoie au journal les messages contenant la chaîne **error**:

   a. Envoyer un message test :

   ```
   # logger error
   ```

   b. Consultez le journal **/var/log/errors.log**, par exemple :

   ```
   # cat /var/log/errors.log
   Aug  5 13:48:31 hostname root[6778]: error
   ```

   Où ***hostname*** est le nom d'hôte du système client. Notez que le journal contient le nom de l'utilisateur qui a entré la commande logger, dans ce cas **root**.

Ressources supplémentaires

- Documentation installée avec le paquetage **rhel-system-roles** dans **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**

12.5. APPLICATION D'UNE SOLUTION DE JOURNALISATION À DISTANCE À L'AIDE DU RÔLE DE SYSTÈME LOGGING

Suivez ces étapes pour préparer et appliquer un playbook Red Hat Ansible Core afin de configurer une

solution de journalisation à distance. Dans ce livre de jeu, un ou plusieurs clients récupèrent les journaux de **systemd-journal** et les transmettent à un serveur distant. Le serveur reçoit des entrées distantes de **remote_rsyslog** et **remote_files** et émet les journaux vers des fichiers locaux dans des répertoires nommés par des noms d'hôtes distants.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **logging**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.
  - Un fichier d'inventaire qui répertorie les nœuds gérés.

NOTE

Il n'est pas nécessaire que le paquet **rsyslog** soit installé, car le rôle de système installe **rsyslog** lorsqu'il est déployé.

Procédure

1. Créez un cahier de jeu qui définit le rôle requis :
   a. Créez un nouveau fichier YAML et ouvrez-le dans un éditeur de texte, par exemple :

   ```
   # vi logging-playbook.yml
   ```

   b. Insérez le contenu suivant dans le fichier :

   ```
   ---
   - name: Deploying remote input and remote_files output
     hosts: server
     roles:
       - rhel-system-roles.logging
     vars:
       logging_inputs:
         - name: remote_udp_input
           type: remote
           udp_ports: [ 601 ]
         - name: remote_tcp_input
           type: remote
           tcp_ports: [ 601 ]
       logging_outputs:
         - name: remote_files_output
           type: remote_files
       logging_flows:
         - name: flow_0
           inputs: [remote_udp_input, remote_tcp_input]
           outputs: [remote_files_output]

   - name: Deploying basics input and forwards output
   ```

```
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: forward_output0
        type: forwards
        severity: info
        target: _host1.example.com_
        udp_port: 601
      - name: forward_output1
        type: forwards
        facility: mail
        target: _host1.example.com_
        tcp_port: 601
    logging_flows:
      - name: flows0
        inputs: [basic_input]
        outputs: [forward_output0, forward_output1]

[basic_input]
[forward_output0, forward_output1]
```

Où ***host1.example.com*** est le serveur de journalisation.

NOTE

Vous pouvez modifier les paramètres du cahier de jeu en fonction de vos besoins.

AVERTISSEMENT

La solution de journalisation ne fonctionne qu'avec les ports définis dans la politique SELinux du serveur ou du système client et ouverts dans le pare-feu. La stratégie SELinux par défaut inclut les ports 601, 514, 6514, 10514 et 20514. Pour utiliser un autre port, modifiez la stratégie SELinux sur les systèmes client et serveur.

2. Créez un fichier d'inventaire qui répertorie vos serveurs et vos clients :

   a. Créez un nouveau fichier et ouvrez-le dans un éditeur de texte, par exemple :

   ```
   # vi inventory.ini
   ```

   b. Insérez le contenu suivant dans le fichier d'inventaire :

   ```
   [servers]
   ```

```
server ansible_host=host1.example.com
[clients]
client ansible_host=host2.example.com
```

Où ?

- ***host1.example.com*** est le serveur de journalisation.
- ***host2.example.com*** est le client de journalisation.

3. Exécutez le manuel de jeu sur votre inventaire.

```
# ansible-playbook -i /path/to/file/inventory.ini /path/to/file/_logging-playbook.yml
```

Où ?

- ***inventory.ini*** est le fichier d'inventaire.
- ***logging-playbook.yml*** est le cahier de jeu que vous avez créé.

Vérification

1. Sur le système client et le système serveur, testez la syntaxe du fichier **/etc/rsyslog.conf**:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. Vérifiez que le système client envoie des messages au serveur :
   a. Sur le système client, envoyez un message de test :

```
# logger test
```

   b. Sur le système serveur, affichez le journal **/var/log/messages**, par exemple :

```
# cat /var/log/messages
Aug  5 13:48:31 host2.example.com root[6778]: test
```

   Où ***host2.example.com*** est le nom d'hôte du système client. Notez que le journal contient le nom de l'utilisateur qui a entré la commande logger, dans ce cas **root**.

Ressources supplémentaires

- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL
- Documentation installée avec le paquetage **rhel-system-roles** dans **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**
- Article de la base de données RHEL System Roles

12.6. UTILISATION DU RÔLE DE SYSTÈME LOGGING AVEC TLS

Transport Layer Security (TLS) est un protocole cryptographique conçu pour sécuriser les communications sur le réseau informatique.

En tant qu'administrateur, vous pouvez utiliser le rôle de système **logging** RHEL pour configurer le transfert sécurisé des journaux à l'aide de Red Hat Ansible Automation Platform.

12.6.1. Configuration de la journalisation des clients avec TLS

Vous pouvez utiliser le rôle de système **logging** pour configurer la journalisation dans les systèmes RHEL qui sont journalisés sur une machine locale et peuvent transférer les journaux vers le système de journalisation distant avec TLS en exécutant un livre de jeu Ansible.

Cette procédure configure TLS sur tous les hôtes du groupe clients dans l'inventaire Ansible. Le protocole TLS crypte la transmission des messages pour un transfert sécurisé des journaux sur le réseau.

Conditions préalables

- Vous disposez d'autorisations pour exécuter des playbooks sur les nœuds gérés sur lesquels vous souhaitez configurer TLS.
- Les nœuds gérés sont répertoriés dans le fichier d'inventaire du nœud de contrôle.
- Les paquets **ansible** et **rhel-system-roles** sont installés sur le nœud de contrôle.

Procédure

1. Créer un fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploying files input and forwards output with certs
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
    logging_inputs:
      - name: input_name
        type: files
        input_log_path: /var/log/containers/*.log
    logging_outputs:
      - name: output_name
        type: forwards
        target: your_target_host
        tcp_port: 514
        tls: true
        pki_authmode: x509/name
        permitted_server: 'server.example.com'
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

Le playbook utilise les paramètres suivants :

logging_pki_files

Ce paramètre permet de configurer TLS et de passer les paramètres **ca_cert_src**, **cert_src** et **private_key_src**.

ca_cert

Représente le chemin d'accès au certificat de l'autorité de certification. Le chemin par défaut est **/etc/pki/tls/certs/ca.pem** et le nom du fichier est défini par l'utilisateur.

cert

Représente le chemin d'accès au certificat. Le chemin par défaut est **/etc/pki/tls/certs/server-cert.pem** et le nom du fichier est défini par l'utilisateur.

private_key

Représente le chemin d'accès à la clé privée. Le chemin par défaut est **/etc/pki/tls/private/server-key.pem** et le nom du fichier est défini par l'utilisateur.

ca_cert_src

Représente le chemin d'accès au fichier de certification de l'autorité de certification locale qui est copié sur l'hôte cible. Si **ca_cert** est spécifié, il est copié à l'emplacement.

cert_src

Représente le chemin d'accès au fichier cert local qui est copié sur l'hôte cible. Si **cert** est spécifié, il est copié à l'emplacement.

private_key_src

Représente le chemin d'accès au fichier clé local qui est copié sur l'hôte cible. Si **private_key** est spécifié, il est copié à l'emplacement.

tls

L'utilisation de ce paramètre garantit un transfert sécurisé des journaux sur le réseau. Si vous ne voulez pas d'enveloppe sécurisée, vous pouvez définir **tls: true**.

2. Vérifier la syntaxe du playbook :

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file playbook.yml
```

12.6.2. Configuration de la journalisation du serveur avec TLS

Vous pouvez utiliser le rôle de système **logging** pour configurer la journalisation dans les systèmes RHEL en tant que serveur et recevoir les journaux du système de journalisation distant avec TLS en exécutant un livre de jeu Ansible.

Cette procédure configure TLS sur tous les hôtes du groupe de serveurs dans l'inventaire Ansible.

Conditions préalables

- Vous disposez d'autorisations pour exécuter des playbooks sur les nœuds gérés sur lesquels vous souhaitez configurer TLS.
- Les nœuds gérés sont répertoriés dans le fichier d'inventaire du nœud de contrôle.

- Les paquets **ansible** et **rhel-system-roles** sont installés sur le nœud de contrôle.

Procédure

1. Créer un fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
    logging_inputs:
      - name: input_name
        type: remote
        tcp_ports: 514
        tls: true
        permitted_clients: ['clients.example.com']
    logging_outputs:
      - name: output_name
        type: remote_files
        remote_log_path: /var/log/remote/%FROMHOST%/%PROGRAMNAME:::secpath-
replace%.log
        async_writing: true
        client_count: 20
        io_buffer_size: 8192
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

Le playbook utilise les paramètres suivants :

logging_pki_files

Ce paramètre permet de configurer TLS et de passer les paramètres **ca_cert_src**, **cert_src** et **private_key_src**.

ca_cert

Représente le chemin d'accès au certificat de l'autorité de certification. Le chemin par défaut est **/etc/pki/tls/certs/ca.pem** et le nom du fichier est défini par l'utilisateur.

cert

Représente le chemin d'accès au certificat. Le chemin par défaut est **/etc/pki/tls/certs/server-cert.pem** et le nom du fichier est défini par l'utilisateur.

private_key

Représente le chemin d'accès à la clé privée. Le chemin par défaut est **/etc/pki/tls/private/server-key.pem** et le nom du fichier est défini par l'utilisateur.

ca_cert_src

Représente le chemin d'accès au fichier de certification de l'autorité de certification locale qui est copié sur l'hôte cible. Si **ca_cert** est spécifié, il est copié à l'emplacement.

cert_src

Représente le chemin d'accès au fichier cert local qui est copié sur l'hôte cible. Si **cert** est spécifié, il est copié à l'emplacement.

private_key_src

Représente le chemin d'accès au fichier clé local qui est copié sur l'hôte cible. Si **private_key** est spécifié, il est copié à l'emplacement.

tls

L'utilisation de ce paramètre garantit un transfert sécurisé des journaux sur le réseau. Si vous ne voulez pas d'enveloppe sécurisée, vous pouvez définir **tls: true**.

2. Vérifier la syntaxe du playbook :

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file playbook.yml
```

12.7. UTILISATION DES RÔLES DU SYSTÈME LOGGING AVEC RELP

Reliable Event Logging Protocol (RELP) est un protocole de réseau pour l'enregistrement de données et de messages sur le réseau TCP. Il garantit une livraison fiable des messages d'événements et vous pouvez l'utiliser dans des environnements qui ne tolèrent aucune perte de message.

L'émetteur de RELP transfère les entrées de journal sous forme de commandes et le récepteur les accuse réception une fois qu'elles ont été traitées. Pour garantir la cohérence, le RELP enregistre le numéro de transaction de chaque commande transférée afin de permettre la récupération de tout type de message.

Vous pouvez envisager un système de journalisation à distance entre le client RELP et le serveur RELP. Le client RELP transfère les journaux vers le système de journalisation distant et le serveur RELP reçoit tous les journaux envoyés par le système de journalisation distant.

Les administrateurs peuvent utiliser le rôle de système **logging** pour configurer le système de journalisation afin qu'il envoie et reçoive des entrées de journal de manière fiable.

12.7.1. Configuration de la journalisation des clients avec RELP

Vous pouvez utiliser le rôle de système **logging** pour configurer la journalisation dans les systèmes RHEL qui sont journalisés sur une machine locale et peuvent transférer les journaux vers le système de journalisation distant avec RELP en exécutant un livre de jeu Ansible.

Cette procédure configure RELP sur tous les hôtes du groupe **clients** dans l'inventaire Ansible. La configuration de RELP utilise Transport Layer Security (TLS) pour crypter la transmission des messages afin de sécuriser le transfert des journaux sur le réseau.

Conditions préalables

- Vous avez le droit d'exécuter des playbooks sur les nœuds gérés sur lesquels vous souhaitez configurer RELP.
- Les nœuds gérés sont répertoriés dans le fichier d'inventaire du nœud de contrôle.

- Les paquets **ansible** et **rhel-system-roles** sont installés sur le nœud de contrôle.

Procédure

1. Créer un fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploying basic input and relp output
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: relp_client
        type: relp
        target: _logging.server.com_
        port: 20514
        tls: true
        ca_cert: _/etc/pki/tls/certs/ca.pem_
        cert: _/etc/pki/tls/certs/client-cert.pem_
        private_key: _/etc/pki/tls/private/client-key.pem_
        pki_authmode: name
        permitted_servers:
          - '*.server.example.com'
    logging_flows:
      - name: _example_flow_
        inputs: [basic_input]
        outputs: [relp_client]
```

Les playbooks utilisent les paramètres suivants :

- **target**: Il s'agit d'un paramètre obligatoire qui spécifie le nom de l'hôte où le système d'enregistrement à distance est exécuté.

- **port**: Numéro de port que le système d'enregistrement à distance écoute.

- **tls**: Assure un transfert sécurisé des journaux sur le réseau. Si vous ne voulez pas d'enveloppe sécurisée, vous pouvez fixer la variable **tls** à **false**. Par défaut, le paramètre **tls** est fixé à true lorsque vous travaillez avec RELP et nécessite des clés/certificats et des triplets {**ca_cert**, **cert**, **private_key**} et/ou {**ca_cert_src**, **cert_src**, **private_key_src**}.

  - Si le triplet {**ca_cert_src**, **cert_src**, **private_key_src**} est défini, les emplacements par défaut **/etc/pki/tls/certs** et **/etc/pki/tls/private** sont utilisés comme destination sur le nœud géré pour transférer des fichiers depuis le nœud de contrôle. Dans ce cas, les noms de fichiers sont identiques aux noms originaux dans le triplet

  - Si le triplet {**ca_cert**, **cert**, **private_key**} est défini, les fichiers sont censés se trouver sur le chemin par défaut avant la configuration de l'enregistrement.

  - Si les deux triplets sont activés, les fichiers sont transférés du chemin local du nœud de contrôle au chemin spécifique du nœud géré.

- **ca_cert**: Représente le chemin d'accès au certificat de l'autorité de certification. Le chemin par défaut est **/etc/pki/tls/certs/ca.pem** et le nom du fichier est défini par l'utilisateur.
- **cert**: Représente le chemin d'accès au certificat. Le chemin par défaut est **/etc/pki/tls/certs/server-cert.pem** et le nom du fichier est défini par l'utilisateur.
- **private_key**: Représente le chemin d'accès à la clé privée. Le chemin par défaut est **/etc/pki/tls/private/server-key.pem** et le nom du fichier est défini par l'utilisateur.
- **ca_cert_src**: Représente le chemin d'accès au fichier CA cert local qui est copié sur l'hôte cible. Si ca_cert est spécifié, il est copié à cet emplacement.
- **cert_src**: Représente le chemin d'accès au fichier local cert qui est copié sur l'hôte cible. Si cert est spécifié, il est copié à l'emplacement.
- **private_key_src**: Représente le chemin d'accès au fichier de la clé locale qui est copié sur l'hôte cible. Si la clé privée est spécifiée, elle est copiée à cet emplacement.
- **pki_authmode**: Accepte le mode d'authentification **name** ou **fingerprint**.
- **permitted_servers**: Liste des serveurs qui seront autorisés par le client de journalisation à se connecter et à envoyer des journaux via TLS.
- **inputs**: Liste des dictionnaires d'entrée de la journalisation.
- **outputs**: Liste des dictionnaires de sortie de la journalisation.

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le manuel de jeu :

```
# ansible-playbook -i inventory_file playbook.yml
```

12.7.2. Configuration de la journalisation du serveur avec RELP

Vous pouvez utiliser le rôle de système **logging** pour configurer la journalisation dans les systèmes RHEL en tant que serveur et recevoir les journaux du système de journalisation distant avec RELP en exécutant un livre de jeu Ansible.

Cette procédure configure RELP sur tous les hôtes du groupe **server** dans l'inventaire Ansible. La configuration de RELP utilise TLS pour chiffrer la transmission des messages afin de sécuriser le transfert des journaux sur le réseau.

Conditions préalables

- Vous avez le droit d'exécuter des playbooks sur les nœuds gérés sur lesquels vous souhaitez configurer RELP.
- Les nœuds gérés sont répertoriés dans le fichier d'inventaire du nœud de contrôle.
- Les paquets **ansible** et **rhel-system-roles** sont installés sur le nœud de contrôle.

Procédure

1. Créer un fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploying remote input and remote_files output
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: relp_server
        type: relp
        port: 20514
        tls: true
        ca_cert: _/etc/pki/tls/certs/ca.pem_
        cert: _/etc/pki/tls/certs/server-cert.pem_
        private_key: _/etc/pki/tls/private/server-key.pem_
        pki_authmode: name
        permitted_clients:
          - '_*example.client.com_'
    logging_outputs:
      - name: _remote_files_output_
        type: _remote_files_
    logging_flows:
      - name: _example_flow_
        inputs: _relp_server_
        outputs: _remote_files_output_
```

Les playbooks utilisent les paramètres suivants :

- **port**: Numéro de port que le système d'enregistrement à distance écoute.
- **tls**: Assure un transfert sécurisé des journaux sur le réseau. Si vous ne voulez pas d'enveloppe sécurisée, vous pouvez fixer la variable **tls** à **false**. Par défaut, le paramètre **tls** est fixé à true lorsque vous travaillez avec RELP et nécessite des clés/certificats et des triplets {**ca_cert**, **cert**, **private_key**} et/ou {**ca_cert_src**, **cert_src**, **private_key_src**}.
  - Si le triplet {**ca_cert_src**, **cert_src**, **private_key_src**} est défini, les emplacements par défaut **/etc/pki/tls/certs** et **/etc/pki/tls/private** sont utilisés comme destination sur le nœud géré pour transférer des fichiers depuis le nœud de contrôle. Dans ce cas, les noms de fichiers sont identiques aux noms originaux dans le triplet
  - Si le triplet {**ca_cert**, **cert**, **private_key**} est défini, les fichiers sont censés se trouver sur le chemin par défaut avant la configuration de l'enregistrement.
  - Si les deux triplets sont activés, les fichiers sont transférés du chemin local du nœud de contrôle au chemin spécifique du nœud géré.
- **ca_cert**: Représente le chemin d'accès au certificat de l'autorité de certification. Le chemin par défaut est **/etc/pki/tls/certs/ca.pem** et le nom du fichier est défini par l'utilisateur.
- **cert**: Représente le chemin d'accès au certificat. Le chemin par défaut est **/etc/pki/tls/certs/server-cert.pem** et le nom du fichier est défini par l'utilisateur.
- **private_key**: Représente le chemin d'accès à la clé privée. Le chemin par défaut est **/etc/pki/tls/private/server-key.pem** et le nom du fichier est défini par l'utilisateur.

- **ca_cert_src**: Représente le chemin d'accès au fichier CA cert local qui est copié sur l'hôte cible. Si ca_cert est spécifié, il est copié à cet emplacement.
- **cert_src**: Représente le chemin d'accès au fichier local cert qui est copié sur l'hôte cible. Si cert est spécifié, il est copié à l'emplacement.
- **private_key_src**: Représente le chemin d'accès au fichier de la clé locale qui est copié sur l'hôte cible. Si la clé privée est spécifiée, elle est copiée à cet emplacement.
- **pki_authmode**: Accepte le mode d'authentification **name** ou **fingerprint**.
- **permitted_clients**: Liste des clients qui seront autorisés par le serveur de journalisation à se connecter et à envoyer des journaux via TLS.
- **inputs**: Liste des dictionnaires d'entrée de la journalisation.
- **outputs**: Liste des dictionnaires de sortie de la journalisation.

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le manuel de jeu :

```
# ansible-playbook -i inventory_file playbook.yml
```

12.8. RESSOURCES SUPPLÉMENTAIRES

- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL
- Documentation installée avec le paquetage **rhel-system-roles** dans **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**.
- Rôles du système RHEL
- **ansible-playbook(1)** page de manuel.

CHAPITRE 13. CONFIGURATION DU JOURNAL SYSTEMD À L'AIDE DU RÔLE DE SYSTÈME RHEL JOURNALD

Avec le rôle de système **journald**, vous pouvez automatiser le journal **systemd** et configurer la journalisation persistante en utilisant la plate-forme d'automatisation Red Hat Ansible.

13.1. VARIABLES POUR LE RÔLE DE SYSTÈME JOURNALD RHEL

Le rôle système **journald** fournit un ensemble de variables permettant de personnaliser le comportement du service de journalisation **journald**. Le rôle comprend les variables suivantes :

Variable de rôle	Description
journald_persistent	Utilisez cette variable booléenne pour configurer **journald** de manière à ce que les fichiers journaux soient stockés sur le disque dans le répertoire **/var/log/journal/**. Si vous donnez à cette variable la valeur **true**, les journaux sont stockés sur le disque ; sinon, ils sont stockés dans la mémoire volatile. La valeur par défaut est **false**.
journald_max_disk_size	Utilisez cette variable pour spécifier la taille maximale, en mégaoctets, que les fichiers journaux peuvent occuper sur le disque. Reportez-vous au calcul de la taille par défaut décrit dans la page de manuel **journald.conf(5)**.
journald_max_files	Utilisez cette variable pour spécifier le nombre maximum de fichiers journaux que vous souhaitez conserver tout en respectant le paramètre **journal_max_disk_size** pour le journal.
journald_max_file_size	Utilisez cette variable pour spécifier la taille maximale, en mégaoctets, d'un seul fichier journal.
journald_per_user	Utilisez cette variable booléenne pour configurer **journald** afin que les données de journalisation soient séparées pour chaque utilisateur. La valeur par défaut est **true** et les utilisateurs non privilégiés peuvent lire les journaux du système à partir de leurs propres services. Notez que les fichiers journaux par utilisateur ne sont disponibles que lorsque la variable **journald_persistent** est définie sur **true**.
journald_compression	Utilisez cette variable booléenne pour appliquer la compression aux objets de données **journald** dont la taille est supérieure à la valeur par défaut de 512 octets. La valeur par défaut est **true**.

Variable de rôle	Description
journald_sync_interval	Utilisez cette variable pour spécifier le temps, en minutes, après lequel **journald** synchronise le fichier journal actuellement utilisé sur le disque. Par défaut, le rôle ne modifie pas la valeur actuelle.

Ressources supplémentaires

- La page de manuel **journald.conf(5)**.

13.2. CONFIGURATION DE LA JOURNALISATION PERSISTANTE À L'AIDE DU RÔLE DE SYSTÈME JOURNALD

En tant qu'administrateur système, vous pouvez configurer la journalisation persistante en utilisant le rôle système **journald**. L'exemple suivant montre comment configurer les variables du rôle système **journald** dans un playbook pour atteindre les objectifs suivants :

- Configuration de la journalisation persistante
- Spécification de la taille maximale de l'espace disque pour les fichiers journaux
- Configurer **journald** pour que les données du journal soient séparées pour chaque utilisateur
- Définition de l'intervalle de synchronisation

Conditions préalables

- Vous avez préparé le nœud de contrôle et les nœuds gérés .
- Vous êtes connecté au nœud de contrôle en tant qu'utilisateur pouvant exécuter des séquences sur les nœuds gérés.
- Le compte que vous utilisez pour vous connecter aux nœuds gérés dispose des autorisations **sudo**.
- Les nœuds gérés ou les groupes de nœuds gérés sur lesquels vous souhaitez exécuter cette séquence sont répertoriés dans le fichier d'inventaire Ansible.

Procédure

1. Créez un nouveau fichier **playbook.yml** avec le contenu suivant :

```
---
- hosts: all
  vars:
    journald_persistent: true
    journald_max_disk_size: 2048
    journald_per_user: true
    journald_sync_interval: 1
```

```
  roles:
    - linux-system-roles.journald
---
```

Par conséquent, le service **journald** stocke vos journaux de manière persistante sur un disque d'une taille maximale de 2048 Mo, et conserve les données des journaux séparément pour chaque utilisateur. La synchronisation a lieu toutes les minutes.

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml -i inventory_file
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

13.3. RESSOURCES SUPPLÉMENTAIRES

- La page de manuel **journald.conf(5)**
- La page de manuel **ansible-playbook(1)**

CHAPITRE 14. CONFIGURATION DE LA COMMUNICATION SÉCURISÉE À L'AIDE DES RÔLES DE SYSTÈME RHEL SSH ET SSHD

En tant qu'administrateur, vous pouvez utiliser le rôle de système **sshd** pour configurer les serveurs SSH et le rôle de système **ssh** pour configurer les clients SSH de manière cohérente sur un nombre quelconque de systèmes RHEL en même temps à l'aide du paquetage Ansible Core.

14.1. SSH VARIABLES DE RÔLE DU SYSTÈME DE SERVEUR

Dans un playbook de rôle de système **sshd**, vous pouvez définir les paramètres du fichier de configuration SSH en fonction de vos préférences et de vos limites.

Si vous ne configurez pas ces variables, le rôle de système produit un fichier **sshd_config** qui correspond aux valeurs par défaut de RHEL.

Dans tous les cas, les booléens s'affichent correctement sous la forme **yes** et **no** dans la configuration **sshd**. Vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes. Par exemple, vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes :

```
sshd_ListenAddress:
  - 0.0.0.0
  - '::'
```

se traduit par :

```
ListenAddress 0.0.0.0
ListenAddress ::
```

Variables pour le rôle de systèmesshd

sshd_enable

Si la valeur est **False**, le rôle est complètement désactivé. La valeur par défaut est **True**.

sshd_skip_defaults

S'il est défini sur **True**, le rôle de système n'applique pas de valeurs par défaut. Au lieu de cela, vous devez spécifier l'ensemble des valeurs par défaut de la configuration en utilisant les variables **sshd** dict ou **sshd_Key**. La valeur par défaut est **False**.

sshd_manage_service

S'il vaut **False**, le service n'est pas géré, ce qui signifie qu'il n'est pas activé au démarrage et qu'il ne démarre pas ou ne se recharge pas. La valeur par défaut est **True**, sauf en cas d'exécution dans un conteneur ou sous AIX, car le module de service Ansible ne prend pas actuellement en charge **enabled** pour AIX.

sshd_allow_reload

S'il est réglé sur **False**, **sshd** ne se recharge pas après un changement de configuration. Cela peut faciliter le dépannage. Pour appliquer la configuration modifiée, rechargez manuellement **sshd**. La valeur par défaut est la même que celle de **sshd_manage_service**, sauf sous AIX, où **sshd_manage_service** a pour valeur par défaut **False** mais **sshd_allow_reload** a pour valeur par défaut **True**.

sshd_install_service

S'il est défini sur **True**, le rôle installe les fichiers de service pour le service **sshd**. Ces fichiers sont prioritaires sur ceux fournis par le système d'exploitation. Ne définissez pas **True** à moins que vous ne configuriez une deuxième instance et que vous ne modifiiez également la variable **sshd_service**. La valeur par défaut est **False**.
Le rôle utilise comme modèles les fichiers désignés par les variables suivantes :

```
sshd_service_template_service (default: templates/sshd.service.j2)
sshd_service_template_at_service (default: templates/sshd@.service.j2)
sshd_service_template_socket (default: templates/sshd.socket.j2)
```

sshd_service

Cette variable modifie le nom du service **sshd**, ce qui est utile pour configurer une deuxième instance du service **sshd**.

sshd

Un dict qui contient la configuration. Par exemple :

```
sshd:
  Compression: yes
  ListenAddress:
    - 0.0.0.0
```

sshd_*OptionName*

Vous pouvez définir des options en utilisant des variables simples composées du préfixe **sshd_** et du nom de l'option au lieu d'un dict. Les variables simples remplacent les valeurs du dict **sshd**. Par exemple :

```
sshd_Compression: no
```

sshd_match et **sshd_match_1** à **sshd_match_9**

Une liste de dicts ou juste un dict pour une section Match. Notez que ces variables ne remplacent pas les blocs de correspondance tels qu'ils sont définis dans le dict **sshd**. Toutes les sources seront reflétées dans le fichier de configuration résultant.

Variables secondaires pour le rôle du système sshd

Vous pouvez utiliser ces variables pour remplacer les valeurs par défaut correspondant à chaque plate-forme prise en charge.

sshd_packages

Vous pouvez remplacer la liste par défaut des paquets installés en utilisant cette variable.

sshd_config_owner, sshd_config_group, et **sshd_config_mode**

Vous pouvez définir la propriété et les autorisations du fichier de configuration **openssh** que ce rôle produit à l'aide de ces variables.

sshd_config_file

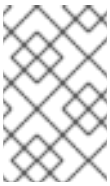
Le chemin où ce rôle enregistre la configuration du serveur **openssh** produite.

sshd_config_namespace

La valeur par défaut de cette variable est null, ce qui signifie que le rôle définit l'intégralité du contenu du fichier de configuration, y compris les valeurs par défaut du système. Vous pouvez également utiliser cette variable pour invoquer ce rôle à partir d'autres rôles ou à partir de plusieurs endroits

dans un seul playbook sur des systèmes qui ne prennent pas en charge le répertoire drop-in. La variable **sshd_skip_defaults** est ignorée et aucune valeur par défaut du système n'est utilisée dans ce cas.

Lorsque cette variable est définie, le rôle place la configuration que vous spécifiez aux extraits de configuration dans un fichier de configuration existant sous l'espace de noms donné. Si votre scénario nécessite d'appliquer le rôle plusieurs fois, vous devez sélectionner un espace de noms différent pour chaque application.



NOTE

Les limitations du fichier de configuration **openssh** restent d'application. Par exemple, seule la première option spécifiée dans un fichier de configuration est effective pour la plupart des options de configuration.

Techniquement, le rôle place les extraits dans les blocs "Match all", à moins qu'ils ne contiennent d'autres blocs de correspondance, afin de s'assurer qu'ils sont appliqués indépendamment des blocs de correspondance précédents dans le fichier de configuration existant. Cela permet de configurer des options non contradictoires à partir de différentes invocations de rôles.

sshd_binary

Chemin d'accès à l'exécutable **sshd** de **openssh**.

sshd_service

Le nom du service **sshd**. Par défaut, cette variable contient le nom du service **sshd** utilisé par la plate-forme cible. Vous pouvez également l'utiliser pour définir le nom du service personnalisé **sshd** lorsque le rôle utilise la variable **sshd_install_service**.

sshd_verify_hostkeys

La valeur par défaut est **auto**. Si la valeur est **auto**, toutes les clés d'hôte présentes dans le fichier de configuration produit sont répertoriées et tous les chemins d'accès non présents sont générés. De plus, les permissions et les propriétaires de fichiers sont définis sur des valeurs par défaut. Ceci est utile si le rôle est utilisé dans la phase de déploiement pour s'assurer que le service est capable de démarrer à la première tentative. Pour désactiver cette vérification, donnez à cette variable la valeur d'une liste vide **[]**.

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

Utilisez ces variables pour définir la propriété et les autorisations des clés de l'hôte à partir de **sshd_verify_hostkeys**.

sshd_sysconfig

Sur les systèmes basés sur RHEL, cette variable configure des détails supplémentaires du service **sshd**. S'il est défini sur **true**, ce rôle gère également le fichier de configuration **/etc/sysconfig/sshd** sur la base de la configuration suivante. La valeur par défaut est **false**.

sshd_sysconfig_override_crypto_policy

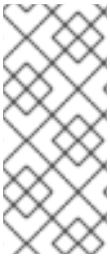
Dans RHEL, lorsqu'elle est définie sur **true**, cette variable remplace la politique cryptographique du système. La valeur par défaut est **false**.

sshd_sysconfig_use_strong_rng

Sur les systèmes basés sur RHEL, cette variable peut forcer **sshd** à réalimenter le générateur de nombres aléatoires **openssl** avec le nombre d'octets donné en argument. La valeur par défaut est **0**, qui désactive cette fonctionnalité. Ne l'activez pas si le système ne dispose pas d'un générateur de nombres aléatoires matériel.

14.2. CONFIGURATION DES SERVEURS OPENSSH À L'AIDE DU RÔLE DE SYSTÈME SSHD

Vous pouvez utiliser le rôle de système **sshd** pour configurer plusieurs serveurs SSH en exécutant un script Ansible.

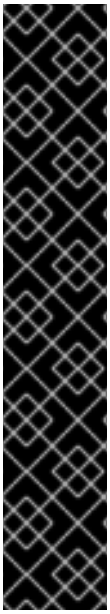


NOTE

Vous pouvez utiliser le rôle système **sshd** avec d'autres rôles système qui modifient la configuration de SSH et de SSHD, par exemple les rôles système RHEL de gestion des identités. Pour éviter que la configuration ne soit écrasée, assurez-vous que le rôle **sshd** utilise des espaces de noms (RHEL 8 et versions antérieures) ou un répertoire de dépôt (RHEL 9).

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **sshd**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.



IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Copiez l'exemple de playbook pour le rôle de système **sshd**:

```
# cp /usr/share/doc/rhel-system-roles/sshd/example-root-login-playbook.yml path/custom-playbook.yml
```

2. Ouvrez le playbook copié en utilisant un éditeur de texte, par exemple :

```
# vim path/custom-playbook.yml

---
- hosts: all
  tasks:
  - name: Configure sshd to prevent root and password login except from particular subnet
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd:
        # root login and password login is enabled only from a particular subnet
        PermitRootLogin: no
        PasswordAuthentication: no
        Match:
        - Condition: "Address 192.0.2.0/24"
          PermitRootLogin: yes
          PasswordAuthentication: yes
```

Le playbook configure le nœud géré en tant que serveur SSH configuré de telle sorte que :

- et le login de l'utilisateur **root** est désactivé
- et **root** n'est possible qu'à partir du sous-réseau **192.0.2.0/24**

Vous pouvez modifier les variables en fonction de vos préférences. Pour plus de détails, voir Variables de rôle du système du serveur SSH .

3. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

4. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file path/custom-playbook.yml

...

PLAY RECAP
**************************************************

localhost : ok=12 changed=2 unreachable=0 failed=0
skipped=10 rescued=0 ignored=0
```

Vérification

1. Connectez-vous au serveur SSH :

```
$ ssh user1@10.1.1.1
```

Où ?

- ***user1*** est un utilisateur du serveur SSH.
- ***10.1.1.1*** est l'adresse IP du serveur SSH.

2. Vérifiez le contenu du fichier **sshd_config** sur le serveur SSH :

```
$ cat /etc/ssh/sshd_config.d/00-ansible_system_role.conf
#
# Ansible managed
#
PasswordAuthentication no
PermitRootLogin no
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
```

3. Vérifiez que vous pouvez vous connecter au serveur en tant que root depuis le sous-réseau **192.0.2.0/24**:

   a. Déterminez votre adresse IP :

   ```
   $ hostname -I
   192.0.2.1
   ```

   Si l'adresse IP se situe dans la plage **192.0.2.1** - **192.0.2.254**, vous pouvez vous connecter au serveur.

   b. Se connecter au serveur en tant que **root**:

   ```
   $ ssh root@10.1.1.1
   ```

Ressources supplémentaires

- **/usr/share/doc/rhel-system-roles/sshd/README.md** fichier.
- **ansible-playbook(1)** page de manuel.

14.3. SSH VARIABLES DE RÔLE DU SYSTÈME

Dans un playbook **ssh** System Role, vous pouvez définir les paramètres du fichier de configuration SSH du client en fonction de vos préférences et de vos limites.

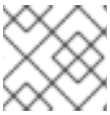
Si vous ne configurez pas ces variables, le rôle de système produit un fichier global **ssh_config** qui correspond aux valeurs par défaut de RHEL.

Dans tous les cas, les booléens s'affichent correctement sous la forme **yes** ou **no** dans la configuration **ssh**. Vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes. Par exemple, vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes :

```
LocalForward:
  - 22 localhost:2222
  - 403 localhost:4003
```

se traduit par :

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```



NOTE

Les options de configuration sont sensibles à la casse.

Variables pour le rôle de systèmessh

ssh_user

Vous pouvez définir un nom d'utilisateur existant pour lequel le rôle système modifie la configuration spécifique à l'utilisateur. La configuration spécifique à l'utilisateur est sauvegardée dans **~/.ssh/config** de l'utilisateur donné. La valeur par défaut est null, ce qui modifie la configuration globale pour tous les utilisateurs.

ssh_skip_defaults

La valeur par défaut est **auto**. Si la valeur est **auto**, le rôle système écrit le fichier de configuration de l'ensemble du système **/etc/ssh/ssh_config** et conserve les valeurs par défaut RHEL qui y sont définies. La création d'un fichier de configuration d'insertion, par exemple en définissant la variable **ssh_drop_in_name**, désactive automatiquement la variable **ssh_skip_defaults**.

ssh_drop_in_name

Définit le nom du fichier de configuration d'insertion, qui est placé dans le répertoire d'insertion de l'ensemble du système. Ce nom est utilisé dans le modèle **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** pour référencer le fichier de configuration à modifier. Si le système ne prend pas en charge les répertoires de dépôt, la valeur par défaut est null. Si le système prend en charge les répertoires de dépôt, la valeur par défaut est **00-ansible**.

AVERTISSEMENT

Si le système ne prend pas en charge les répertoires d'insertion, la définition de cette option entraînera l'échec de la lecture.

Le format suggéré est **NN-name**, où **NN** est un numéro à deux chiffres utilisé pour classer les fichiers de configuration et **name** est un nom descriptif du contenu ou du propriétaire du fichier.

ssh

Un dict qui contient les options de configuration et leurs valeurs respectives.

ssh_*OptionName*

Vous pouvez définir des options en utilisant des variables simples composées du préfixe **ssh_** et du nom de l'option au lieu d'un dict. Les variables simples remplacent les valeurs du dict **ssh**.

ssh_additional_packages

Ce rôle installe automatiquement les paquets **openssh** et **openssh-clients**, qui sont nécessaires pour les cas d'utilisation les plus courants. Si vous devez installer des paquets supplémentaires, par exemple **openssh-keysign** pour l'authentification basée sur l'hôte, vous pouvez les spécifier dans cette variable.

ssh_config_file

Chemin dans lequel le rôle enregistre le fichier de configuration produit. Valeur par défaut :

- Si le système dispose d'un répertoire de dépôt, la valeur par défaut est définie par le modèle **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf**.

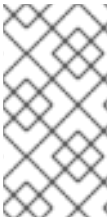
- Si le système ne dispose pas d'un répertoire d'accueil, la valeur par défaut est **/etc/ssh/ssh_config**.
- si la variable **ssh_user** est définie, la valeur par défaut est **~/.ssh/config**.

ssh_config_owner, ssh_config_group, ssh_config_mode

Le propriétaire, le groupe et les modes du fichier de configuration créé. Par défaut, le propriétaire du fichier est **root:root**, et le mode est **0644**. Si **ssh_user** est défini, le mode est **0600**, et le propriétaire et le groupe sont dérivés du nom d'utilisateur spécifié dans la variable **ssh_user**.

14.4. CONFIGURATION DES CLIENTS OPENSSH À L'AIDE DU RÔLE DE SYSTÈME SSH

Vous pouvez utiliser le rôle de système **ssh** pour configurer plusieurs clients SSH en exécutant un script Ansible.



NOTE

Vous pouvez utiliser le rôle de système **ssh** avec d'autres rôles de système qui modifient la configuration de SSH et de SSHD, par exemple les rôles de système RHEL de gestion des identités. Pour éviter que la configuration ne soit écrasée, assurez-vous que le rôle **ssh** utilise un répertoire de dépôt (par défaut à partir de RHEL 8).

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **ssh**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- hosts: all
  tasks:
  - name: "Configure ssh clients"
    include_role:
      name: rhel-system-roles.ssh
    vars:
      ssh_user: root
      ssh:
        Compression: true
        GSSAPIAuthentication: no
        ControlMaster: auto
        ControlPath: ~/.ssh/.cm%C
        Host:
          - Condition: example
            Hostname: example.com
            User: user1
      ssh_ForwardX11: no
```

   Ce playbook configure les préférences du client SSH de l'utilisateur **root** sur les nœuds gérés avec les configurations suivantes :

   - La compression est activée.
   - Le multiplexage ControlMaster est réglé sur **auto**.
   - L'alias ***example*** pour se connecter à l'hôte ***example.com*** est ***user1***.
   - L'alias ***example*** l'alias de l'hôte est créé, ce qui représente une connexion à l'hôte ***example.com*** hôte avec le ***user1*** nom d'utilisateur.
   - Le transfert X11 est désactivé.

   En option, vous pouvez modifier ces variables selon vos préférences. Pour plus de détails, voir **ssh** System Role variables.

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

Vérification

- Vérifiez que le nœud géré a la bonne configuration en ouvrant le fichier de configuration SSH dans un éditeur de texte, par exemple :

```
# vi ~root/.ssh/config
```

Après l'application de l'exemple de playbook présenté ci-dessus, le fichier de configuration devrait avoir le contenu suivant :

```
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

14.5. UTILISATION DU RÔLE DE SYSTÈME SSHD POUR UNE CONFIGURATION NON EXCLUSIVE

Normalement, l'application du rôle de système **sshd** écrase toute la configuration. Cela peut être problématique si vous avez précédemment ajusté la configuration, par exemple avec un rôle de système ou un livre de jeu différent. Pour appliquer le rôle système **sshd** uniquement aux options de configuration sélectionnées tout en conservant les autres options, vous pouvez utiliser la configuration non exclusive.

Dans RHEL 8 et les versions antérieures, vous pouvez appliquer la configuration non exclusive à l'aide d'un extrait de configuration. Pour plus d'informations, voir Utilisation du rôle de système du serveur SSH pour la configuration non exclusive dans la documentation de RHEL 8.

Dans RHEL 9, vous pouvez appliquer la configuration non exclusive en utilisant des fichiers dans un répertoire de dépôt. Le fichier de configuration par défaut est déjà placé dans le répertoire de dépôt en tant que **/etc/ssh/sshd_config.d/00-ansible_system_role.conf**.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **sshd**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Le paquet **ansible-core** est installé.
  - Un fichier d'inventaire qui répertorie les nœuds gérés.
  - Un playbook pour un rôle de système RHEL différent.

Procédure

1. Ajoutez un extrait de configuration avec la variable **sshd_config_file** au playbook :

```
---
- hosts: all
  tasks:
  - name: <Configure sshd to accept some useful environment variables>
    include_role:
```

```
    name: rhel-system-roles.sshd
  vars:
    sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
    sshd:
      # Environment variables to accept
      AcceptEnv:
        LANG
        LS_COLORS
        EDITOR
```

Dans la variable **sshd_config_file**, définissez le fichier **.conf** dans lequel le rôle de système **sshd** écrit les options de configuration.

Utilisez un préfixe à deux chiffres, par exemple ***42-*** pour spécifier l'ordre dans lequel les fichiers de configuration seront appliqués.

Lorsque vous appliquez le playbook à l'inventaire, le rôle ajoute les options de configuration suivantes au fichier défini par la variable **sshd_config_file**.

```
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

Vérification

- Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml -i inventory_file
```

Ressources supplémentaires

- **/usr/share/doc/rhel-system-roles/sshd/README.md** fichier.
- **ansible-playbook(1)** page de manuel.

CHAPITRE 15. CONFIGURATION DES CONNEXIONS VPN AVEC IPSEC À L'AIDE DU RÔLE SYSTÈME VPN RHEL

Avec le rôle de système **vpn**, vous pouvez configurer des connexions VPN sur des systèmes RHEL en utilisant Red Hat Ansible Automation Platform. Vous pouvez l'utiliser pour configurer des configurations d'hôte à hôte, de réseau à réseau, de serveur d'accès à distance VPN et de maillage.

Pour les connexions d'hôte à hôte, le rôle établit un tunnel VPN entre chaque paire d'hôtes dans la liste de **vpn_connections** en utilisant les paramètres par défaut, y compris la génération de clés si nécessaire. Vous pouvez également le configurer pour créer une configuration de maillage opportuniste entre tous les hôtes de la liste. Le rôle suppose que les noms des hôtes sous **hosts** sont les mêmes que les noms des hôtes utilisés dans l'inventaire Ansible, et que vous pouvez utiliser ces noms pour configurer les tunnels.

NOTE

Le rôle de système RHEL **vpn** ne prend actuellement en charge que Libreswan, qui est une implémentation IPsec, en tant que fournisseur de VPN.

15.1. CRÉATION D'UN VPN D'HÔTE À HÔTE AVEC IPSEC À L'AIDE DU RÔLE DE SYSTÈME VPN

Vous pouvez utiliser le rôle de système **vpn** pour configurer les connexions d'hôte à hôte en exécutant un manuel de jeu Ansible sur le nœud de contrôle, qui configurera tous les nœuds gérés répertoriés dans un fichier d'inventaire.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **vpn**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

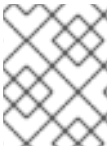
```
- name: Host to host VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

   Ce playbook configure la connexion ***managed_node1*-to-*managed_node2*** en utilisant l'authentification par clé pré-partagée avec des clés générées automatiquement par le rôle système. Puisque **vpn_manage_firewall** et **vpn_manage_selinux** sont tous deux définis sur true, le rôle **vpn** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **vpn**.

2. Facultatif : Configurez les connexions entre les hôtes gérés et les hôtes externes qui ne sont pas répertoriés dans le fichier d'inventaire en ajoutant la section suivante à la liste d'hôtes **vpn_connections**:

```
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
          external_node:
            hostname: 192.0.2.2
```

   Cela permet de configurer deux connexions supplémentaires : ***managed_node1*-to-*external_node*** et ***managed_node2*-to-*external_node***.



NOTE

Les connexions sont configurées uniquement sur les nœuds gérés et non sur le nœud externe.

1. Facultatif : vous pouvez spécifier plusieurs connexions VPN pour les nœuds gérés en utilisant des sections supplémentaires dans **vpn_connections**, par exemple un plan de contrôle et un plan de données :

```
- name: Multiple VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed_node1:
            hostname: 192.0.2.0 # IP for the control plane
          managed_node2:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed_node1:
            hostname: 10.0.0.1 # IP for the data plane
          managed_node2:
            hostname: 10.0.0.2
```

2. Facultatif : vous pouvez modifier les variables selon vos préférences. Pour plus de détails, voir le fichier **/usr/share/doc/rhel-system-roles/vpn/README.md**.

3. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check /path/to/file/playbook.yml -i /path/to/file/inventory_file
```

4. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i /path/to/file/inventory_file /path/to/file/playbook.yml
```

Vérification

1. Sur les nœuds gérés, confirmez que la connexion est chargée avec succès :

```
# ipsec status | grep connection.name
```

Remplacez *connection.name* par le nom de la connexion de ce nœud, par exemple **managed_node1-to-managed_node2**.

NOTE

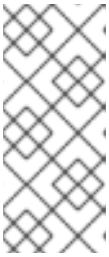
Par défaut, le rôle génère un nom descriptif pour chaque connexion qu'il crée du point de vue de chaque système. Par exemple, lors de la création d'une connexion entre **managed_node1** et **managed_node2**, le nom descriptif de cette connexion sur **managed_node1** est **managed_node1-to-managed_node2** mais sur **managed_node2** la connexion est nommée **managed_node2-to-managed_node1**.

1. Sur les nœuds gérés, confirmez que la connexion a été lancée avec succès :

```
# ipsec trafficstatus | grep connection.name
```

2. Facultatif : si une connexion n'a pas été chargée avec succès, ajoutez-la manuellement en entrant la commande suivante. Vous obtiendrez ainsi des informations plus précises sur la raison pour laquelle la connexion n'a pas pu être établie :

```
# ipsec auto --add connection.name
```



NOTE

Toutes les erreurs qui ont pu se produire au cours du processus de chargement et de démarrage de la connexion sont signalées dans les journaux, qui se trouvent à l'adresse **/var/log/pluto.log**. Comme ces journaux sont difficiles à analyser, essayez d'ajouter manuellement la connexion afin d'obtenir les messages de journaux à partir de la sortie standard à la place.

15.2. CRÉATION D'UNE CONNEXION VPN MAILLÉE OPPORTUNISTE AVEC IPSEC EN UTILISANT LE RÔLE DE SYSTÈME VPN

Vous pouvez utiliser le rôle système **vpn** pour configurer une connexion VPN maillée opportuniste qui utilise des certificats pour l'authentification en exécutant un livre de jeu Ansible sur le nœud de contrôle, qui configurera tous les nœuds gérés répertoriés dans un fichier d'inventaire.

L'authentification par certificat est configurée en définissant le paramètre **auth_method: cert** dans le playbook. Le rôle de système **vpn** suppose que la bibliothèque cryptographique IPsec Network Security Services (NSS), définie dans le répertoire **/etc/ipsec.d**, contient les certificats nécessaires. Par défaut, le nom du nœud est utilisé comme pseudonyme du certificat. Dans cet exemple, il s'agit de **managed_node1**. Vous pouvez définir d'autres noms de certificats en utilisant l'attribut **cert_name** dans votre inventaire.

Dans l'exemple de procédure suivant, le nœud de contrôle, qui est le système à partir duquel vous exécuterez le playbook Ansible, partage le même numéro de routage interdomaine sans classe (CIDR) que les deux nœuds gérés (192.0.2.0/24) et possède l'adresse IP 192.0.2.7. Par conséquent, le nœud de contrôle relève de la stratégie privée qui est automatiquement créée pour le CIDR 192.0.2.0/24.

Pour éviter la perte de connexion SSH pendant la pièce, une politique claire pour le nœud de contrôle est incluse dans la liste des politiques. Notez qu'il y a également un élément dans la liste des stratégies où le CIDR est égal à la valeur par défaut. Cela s'explique par le fait que ce playbook remplace la règle de la stratégie par défaut pour la rendre privée au lieu de private-or-clear.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **vpn**.
  - Sur tous les nœuds gérés, la base de données NSS dans le répertoire **/etc/ipsec.d** contient tous les certificats nécessaires à l'authentification des pairs. Par défaut, le nom du nœud est utilisé comme pseudonyme du certificat.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
- name: Mesh VPN
  hosts: managed_node1, managed_node2, managed_node3
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - opportunistic: true
        auth_method: cert
        policies:
          - policy: private
            cidr: default
          - policy: private-or-clear
            cidr: 198.51.100.0/24
          - policy: private
            cidr: 192.0.2.0/24
          - policy: clear
```

```
        cidr: 192.0.2.7/32
vpn_manage_firewall: true
vpn_manage_selinux: true
```

NOTE

Puisque **vpn_manage_firewall** et **vpn_manage_selinux** sont tous deux définis sur true, le rôle **vpn** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **vpn**.

2. Facultatif : vous pouvez modifier les variables selon vos préférences. Pour plus de détails, voir le fichier **/usr/share/doc/rhel-system-roles/vpn/README.md**.

3. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

4. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

15.3. RESSOURCES SUPPLÉMENTAIRES

- Pour plus de détails sur les paramètres utilisés dans le rôle de système **vpn** et des informations supplémentaires sur le rôle, voir le fichier **/usr/share/doc/rhel-system-roles/vpn/README.md**.
- Pour plus de détails sur la commande **ansible-playbook**, voir la page de manuel **ansible-playbook(1)**.

CHAPITRE 16. DÉFINITION D'UNE POLITIQUE CRYPTOGRAPHIQUE PERSONNALISÉE À L'AIDE DU RÔLE DE SYSTÈME RHEL CRYPTO-POLICIES

En tant qu'administrateur, vous pouvez utiliser le rôle système **crypto_policies** RHEL pour configurer rapidement et de manière cohérente des politiques cryptographiques personnalisées sur de nombreux systèmes différents à l'aide du package Ansible Core.

16.1. CRYPTO_POLICIES VARIABLES ET FAITS RELATIFS AU RÔLE DU SYSTÈME

Dans un playbook **crypto_policies** System Role, vous pouvez définir les paramètres du fichier de configuration **crypto_policies** en fonction de vos préférences et de vos limites.

Si vous ne configurez aucune variable, le rôle système ne configure pas le système et se contente de signaler les faits.

Variables sélectionnées pour le rôle du systèmecrypto_policies

crypto_policies_policy

Détermine la stratégie cryptographique que le rôle système applique aux nœuds gérés. Pour plus d'informations sur les différentes stratégies cryptographiques, voir Stratégies cryptographiques à l'échelle du système.

crypto_policies_reload

S'il est défini sur **yes**, les services concernés, actuellement les services **ipsec**, **bind** et **sshd**, se rechargent après l'application d'une stratégie cryptographique. La valeur par défaut est **yes**.

crypto_policies_reboot_ok

S'il est défini sur **yes** et qu'un redémarrage est nécessaire après que le rôle système a modifié la politique de cryptage, il définit **crypto_policies_reboot_required** sur **yes**. La valeur par défaut est **no**.

Faits définis par le systèmecrypto_policies Rôle

crypto_policies_active

Liste la politique actuellement sélectionnée.

crypto_policies_available_policies

Répertorie toutes les politiques disponibles sur le système.

crypto_policies_available_subpolicies

Répertorie toutes les sous-politiques disponibles sur le système.

Ressources supplémentaires

- Création et définition d'une politique cryptographique personnalisée à l'échelle du système .

16.2. DÉFINITION D'UNE POLITIQUE CRYPTOGRAPHIQUE PERSONNALISÉE À L'AIDE DU RÔLE DE SYSTÈME CRYPTO_POLICIES

Vous pouvez utiliser le rôle de système **crypto_policies** pour configurer un grand nombre de nœuds gérés de manière cohérente à partir d'un seul nœud de contrôle.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **crypto_policies**.
- Accès et autorisations à un nœud de contrôle, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- hosts: all
  tasks:
  - name: Configure crypto policies
    include_role:
      name: rhel-system-roles.crypto_policies
    vars:
      - crypto_policies_policy: FUTURE
      - crypto_policies_reboot_ok: true
```

Vous pouvez remplacer la valeur *FUTURE* par votre politique cryptographique préférée, par exemple : **DEFAULT**, **LEGACY**, et **FIPS:OSPP**.

La variable **crypto_policies_reboot_ok: true** provoque le redémarrage du système après que le rôle système a modifié la stratégie cryptographique.

Pour plus de détails, voir crypto_policies Variables et faits relatifs au rôle du système .

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file playbook.yml
```

Vérification

1. Sur le nœud de contrôle, créez un autre playbook nommé, par exemple, ***verify_playbook.yml***:

```
- hosts: all
  tasks:
 - name: Verify active crypto policy
   include_role:
     name: rhel-system-roles.crypto_policies

 - debug:
     var: crypto_policies_active
```

Ce playbook ne modifie aucune configuration sur le système, mais signale uniquement la politique active sur les nœuds gérés.

2. Exécutez le playbook sur le même fichier d'inventaire :

```
# ansible-playbook -i inventory_file verify_playbook.yml

TASK [debug] ***************************
ok: [host] => {
    "crypto_policies_active": "FUTURE"
}
```

La variable **"crypto_policies_active":** indique la politique active sur le nœud géré.

16.3. RESSOURCES SUPPLÉMENTAIRES

- **/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md** fichier.
- **ansible-playbook(1)** page de manuel.
- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL .

CHAPITRE 17. CONFIGURATION DE L'EDNB À L'AIDE DES RÔLES SYSTÈME RHEL

17.1. INTRODUCTION AUX RÔLES DES SYSTÈMES NBDE_CLIENT ET NBDE_SERVER (CLEVIS ET TANG)

RHEL System Roles est une collection de rôles et de modules Ansible qui fournissent une interface de configuration cohérente pour gérer à distance plusieurs systèmes RHEL.

Vous pouvez utiliser les rôles Ansible pour les déploiements automatisés de solutions de décryptage basé sur des politiques (PBD) à l'aide de Clevis et Tang. Le paquetage **rhel-system-roles** contient ces rôles système, les exemples associés ainsi que la documentation de référence.

Le rôle de système **nbde_client** vous permet de déployer plusieurs clients Clevis de manière automatisée. Notez que le rôle **nbde_client** ne prend en charge que les liaisons Tang, et que vous ne pouvez pas l'utiliser pour les liaisons TPM2 pour le moment.

Le rôle **nbde_client** nécessite des volumes qui sont déjà chiffrés à l'aide de LUKS. Ce rôle permet de lier un volume chiffré à l'aide de LUKS à un ou plusieurs serveurs liés au réseau (NBDE) - serveurs Tang. Vous pouvez soit préserver le chiffrement existant du volume à l'aide d'une phrase d'authentification, soit le supprimer. Après avoir supprimé la phrase d'authentification, vous pouvez déverrouiller le volume uniquement à l'aide de l'EDNB. Cette option est utile lorsqu'un volume est initialement crypté à l'aide d'une clé ou d'un mot de passe temporaire que vous devez supprimer après avoir approvisionné le système.

Si vous fournissez à la fois une phrase d'authentification et un fichier clé, le rôle utilise d'abord ce que vous avez fourni. S'il ne trouve aucun fichier valide, il tente de récupérer une phrase d'authentification à partir d'une liaison existante.

La PBD définit une liaison comme une correspondance entre un appareil et un emplacement. Cela signifie que vous pouvez avoir plusieurs liaisons pour le même appareil. L'emplacement par défaut est l'emplacement 1.

Le rôle **nbde_client** fournit également la variable **state**. Utilisez la valeur **present** pour créer une nouvelle liaison ou mettre à jour une liaison existante. Contrairement à la commande **clevis luks bind**, vous pouvez également utiliser **state: present** pour écraser une liaison existante dans son emplacement. La valeur **absent** supprime une liaison spécifiée.

Le rôle de système **nbde_client** vous permet de déployer et de gérer un serveur Tang dans le cadre d'une solution de chiffrement de disque automatisée. Ce rôle prend en charge les fonctionnalités suivantes :

- Touches Tang rotatives
- Déploiement et sauvegarde des clés Tang

Ressources supplémentaires

- Pour une référence détaillée sur les variables de rôle NBDE (Network-Bound Disk Encryption), installez le paquetage **rhel-system-roles** et consultez les fichiers **README.md** et **README.html** dans les répertoires **/usr/share/doc/rhel-system-roles/nbde_client/** et **/usr/share/doc/rhel-system-roles/nbde_server/**.

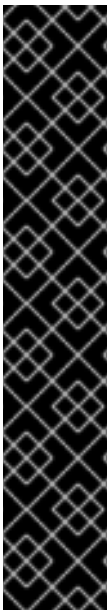
- Par exemple, les playbooks system-roles, installez le paquetage **rhel-system-roles** et consultez les répertoires **/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/**.
- Pour plus d'informations sur les rôles système RHEL, voir Préparation d'un nœud de contrôle et des nœuds gérés à l'utilisation des rôles système RHEL.

17.2. UTILISATION DU RÔLE DE SYSTÈME NBDE_SERVER POUR CONFIGURER PLUSIEURS SERVEURS TANG

Suivez les étapes pour préparer et appliquer un playbook Ansible contenant les paramètres de votre serveur Tang.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **nbde_server**.
- Accès et autorisations à un nœud de contrôle, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.



IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Préparez votre playbook contenant les paramètres des serveurs Tang. Vous pouvez partir de zéro ou utiliser l'un des playbooks d'exemple du répertoire **/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/**.

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/simple_deploy.yml
./my-tang-playbook.yml
```

2. Modifiez le playbook dans un éditeur de texte de votre choix, par exemple :

```
# vi my-tang-playbook.yml
```

3. Ajoutez les paramètres requis. L'exemple de playbook suivant assure le déploiement de votre serveur Tang et une rotation des clés :

```
---
- hosts: all

  vars:
    nbde_server_rotate_keys: yes
    nbde_server_manage_firewall: true
    nbde_server_manage_selinux: true

  roles:
    - rhel-system-roles.nbde_server
```

NOTE

Puisque **nbde_server_manage_firewall** et **nbde_server_manage_selinux** sont tous deux définis sur true, le rôle **nbde_server** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **nbde_server**.

4. Appliquer le manuel de jeu terminé :

```
# ansible-playbook -i inventory-file my-tang-playbook.yml
```

Where: * ***inventory-file*** is the inventory file. * ***logging-playbook.yml*** is the playbook you use.

IMPORTANT

Pour s'assurer que la mise en réseau d'une broche Tang est disponible lors du démarrage anticipé, utilisez l'outil **grubby** sur les systèmes où Clevis est installé :

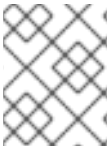
```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

Ressources supplémentaires

- Pour plus d'informations, installez le paquetage **rhel-system-roles** et consultez les répertoires **/usr/share/doc/rhel-system-roles/nbde_server/** et **usr/share/ansible/roles/rhel-system-roles.nbde_server/**.

17.3. UTILISATION DU RÔLE DE SYSTÈME NBDE_CLIENT POUR CONFIGURER PLUSIEURS CLIENTS CLEVIS

Suivez les étapes pour préparer et appliquer un playbook Ansible contenant les paramètres de votre client Clevis.



NOTE

Le rôle de système **nbde_client** ne prend en charge que les liaisons Tang. Cela signifie que vous ne pouvez pas l'utiliser pour les liaisons TPM2 pour le moment.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **nbde_client**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le guide de lecture.

Procédure

1. Préparez votre playbook contenant les paramètres pour les clients Clevis. Vous pouvez partir de zéro ou utiliser l'un des playbooks d'exemple du répertoire **/usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/**.

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/high_availability.yml
./my-clevis-playbook.yml
```

2. Modifiez le playbook dans un éditeur de texte de votre choix, par exemple :

```
# vi my-clevis-playbook.yml
```

3. Ajoutez les paramètres requis. L'exemple suivant configure les clients Clevis pour le déverrouillage automatique de deux volumes cryptés LUKS lorsqu'au moins l'un des deux serveurs Tang est disponible :

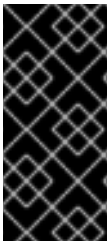
```
---
- hosts: all

  vars:
    nbde_client_bindings:
      - device: /dev/rhel/root
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com
      - device: /dev/rhel/swap
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com

  roles:
    - rhel-system-roles.nbde_client
```

4. Appliquer le manuel de jeu terminé :

```
# ansible-playbook -i host1,host2,host3 my-clevis-playbook.yml
```



IMPORTANT

Pour s'assurer que la mise en réseau d'une broche Tang est disponible lors du démarrage anticipé, utilisez l'outil **grubby** sur le système où Clevis est installé :

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

Ressources supplémentaires

- Pour plus de détails sur les paramètres et des informations supplémentaires sur le rôle du système client NBDE, installez le paquetage **rhel-system-roles** et consultez les répertoires **/usr/share/doc/rhel-system-roles/nbde_client/** et **/usr/share/ansible/roles/rhel-system-roles.nbde_client/**.

CHAPITRE 18. DEMANDE DE CERTIFICATS À L'AIDE DES RÔLES SYSTÈME RHEL

Vous pouvez utiliser le rôle de système **certificate** pour émettre et gérer des certificats.

Ce chapitre couvre les sujets suivants :

- Le rôle du système **certificate**
- Demande d'un nouveau certificat auto-signé à l'aide du rôle de système **certificate**
- Demande d'un nouveau certificat à l'autorité de certification IdM à l'aide du rôle de système **certificate**

18.1. LE RÔLE DU SYSTÈME CERTIFICATE

En utilisant le rôle de système **certificate**, vous pouvez gérer l'émission et le renouvellement des certificats TLS et SSL à l'aide d'Ansible Core.

Ce rôle utilise **certmonger** comme fournisseur de certificats et prend actuellement en charge l'émission et le renouvellement de certificats auto-signés et l'utilisation de l'autorité de certification (AC) intégrée à IdM.

Vous pouvez utiliser les variables suivantes dans votre playbook Ansible avec le rôle de système **certificate**:

certificate_wait

pour indiquer si la tâche doit attendre que le certificat soit délivré.

certificate_requests

pour représenter chaque certificat à délivrer et ses paramètres.

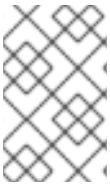
Ressources supplémentaires

- Voir le fichier **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md**.
- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL

18.2. DEMANDE D'UN NOUVEAU CERTIFICAT AUTO-SIGNÉ À L'AIDE DU RÔLE DE SYSTÈME CERTIFICATE

Avec le rôle de système **certificate**, vous pouvez utiliser Ansible Core pour émettre des certificats auto-signés.

Ce processus utilise le fournisseur **certmonger** et demande le certificat par le biais de la commande **getcert**.



NOTE

Par défaut, **certmonger** essaie automatiquement de renouveler le certificat avant qu'il n'expire. Vous pouvez désactiver cette fonction en définissant le paramètre **auto_renew** dans le manuel de jeu Ansible sur **no**.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.

Procédure

1. *Optional:* Créer un fichier d'inventaire, par exemple **inventory.file**:

```
$ *touch inventory.file* (toucher le fichier d'inventaire)
```

2. Ouvrez votre fichier d'inventaire et définissez les hôtes sur lesquels vous souhaitez demander le certificat, par exemple :

```
[webserver]
server.idm.example.com
```

3. Créez un fichier playbook, par exemple **request-certificate.yml**:
   - Définissez **hosts** pour inclure les hôtes sur lesquels vous souhaitez demander le certificat, par exemple **webserver**.
   - Définissez la variable **certificate_requests** de manière à ce qu'elle contienne les éléments suivants :
     - Attribuez au paramètre **name** le nom souhaité pour le certificat, par exemple **mycert**.
     - Le paramètre **dns** correspond au domaine à inclure dans le certificat, par exemple ***.example.com**.
     - Réglez le paramètre **ca** sur **self-sign**.
   - Définir le rôle de **rhel-system-roles.certificate** sous **roles**.
     Il s'agit du fichier du playbook pour cet exemple :

```
---
- hosts: webserver

  vars:
    certificate_requests:
      - name: mycert
        dns: "*.example.com"
        ca: self-sign

  roles:
    - rhel-system-roles.certificate
```

4. Enregistrer le fichier.
5. Exécutez le manuel de jeu :

```
*ansible-playbook -i inventory.file request-certificate.yml* $ *ansible-playbook -i inventory.file request-certificate.yml* $ *ansible-playbook -i inventory.file
```

Ressources supplémentaires

- Voir le fichier **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md**.
- Voir la page de manuel **ansible-playbook(1)**.

18.3. DEMANDE D'UN NOUVEAU CERTIFICAT À L'AUTORITÉ DE CERTIFICATION IDM À L'AIDE DU RÔLE DE SYSTÈME CERTIFICATE

Avec le rôle de système **certificate**, vous pouvez utiliser **anible-core** pour émettre des certificats tout en utilisant un serveur IdM avec une autorité de certification (CA) intégrée. Vous pouvez donc gérer efficacement et de manière cohérente la chaîne de confiance des certificats pour plusieurs systèmes lorsque vous utilisez IdM comme autorité de certification.

Ce processus utilise le fournisseur **certmonger** et demande le certificat par le biais de la commande **getcert**.

NOTE

Par défaut, **certmonger** essaie automatiquement de renouveler le certificat avant qu'il n'expire. Vous pouvez désactiver cette fonction en définissant le paramètre **auto_renew** dans le manuel de jeu Ansible sur **no**.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.

Procédure

1. *Optional:* Créer un fichier d'inventaire, par exemple **inventory.file**:

```
$ *touch inventory.file* (toucher le fichier d'inventaire)
```

2. Ouvrez votre fichier d'inventaire et définissez les hôtes sur lesquels vous souhaitez demander le certificat, par exemple :

```
[webserver]
server.idm.example.com
```

3. Créez un fichier playbook, par exemple **request-certificate.yml**:
   - Définissez **hosts** pour inclure les hôtes sur lesquels vous souhaitez demander le certificat, par exemple **webserver**.
   - Définissez la variable **certificate_requests** de manière à ce qu'elle contienne les éléments suivants :
     - Attribuez au paramètre **name** le nom souhaité pour le certificat, par exemple **mycert**.
     - Le paramètre **dns** correspond au domaine à inclure dans le certificat, par exemple **www.example.com**.

- Le paramètre **principal** indique le principal Kerberos, par exemple **HTTP/www.example.com@EXAMPLE.COM**.
- Réglez le paramètre **ca** sur **ipa**.

- Définir le rôle de **rhel-system-roles.certificate** sous **roles**.
  Il s'agit du fichier du playbook pour cet exemple :

```
---
- hosts: webserver
  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
        principal: HTTP/www.example.com@EXAMPLE.COM
        ca: ipa

  roles:
    - rhel-system-roles.certificate
```

4. Enregistrer le fichier.
5. Exécutez le manuel de jeu :

```
*ansible-playbook -i inventory.file request-certificate.yml* $ *ansible-playbook -i inventory.file request-certificate.yml* $ *ansible-playbook -i inventory.file
```

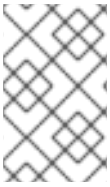
Ressources supplémentaires

- Voir le fichier **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md**.
- Voir la page de manuel **ansible-playbook(1)**.

18.4. SPÉCIFICATION DES COMMANDES À EXÉCUTER AVANT OU APRÈS L'ÉMISSION D'UN CERTIFICAT À L'AIDE DU RÔLE DE SYSTÈME CERTIFICATE

Avec le rôle **certificate**, vous pouvez utiliser Ansible Core pour exécuter une commande avant et après l'émission ou le renouvellement d'un certificat.

Dans l'exemple suivant, l'administrateur veille à arrêter le service **httpd** avant l'émission ou le renouvellement d'un certificat auto-signé pour **www.example.com**, et à le redémarrer ensuite.



NOTE

Par défaut, **certmonger** essaie automatiquement de renouveler le certificat avant qu'il n'expire. Vous pouvez désactiver cette fonction en définissant le paramètre **auto_renew** dans le manuel de jeu Ansible sur **no**.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.

- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.

Procédure

1. *Optional:* Créer un fichier d'inventaire, par exemple **inventory.file**:

```
$ *touch inventory.file* (toucher le fichier d'inventaire)
```

2. Ouvrez votre fichier d'inventaire et définissez les hôtes sur lesquels vous souhaitez demander le certificat, par exemple :

```
[webserver]
server.idm.example.com
```

3. Créez un fichier playbook, par exemple **request-certificate.yml**:

   - Définissez **hosts** pour inclure les hôtes sur lesquels vous souhaitez demander le certificat, par exemple **webserver**.
   - Définissez la variable **certificate_requests** de manière à ce qu'elle contienne les éléments suivants :
     - Attribuez au paramètre **name** le nom souhaité pour le certificat, par exemple **mycert**.
     - Le paramètre **dns** correspond au domaine à inclure dans le certificat, par exemple **www.example.com**.
     - Définissez le paramètre **ca** en fonction de l'autorité de certification que vous souhaitez utiliser pour émettre le certificat, par exemple **self-sign**.
     - Attribuez au paramètre **run_before** la valeur de la commande que vous souhaitez exécuter avant l'émission ou le renouvellement de ce certificat, par exemple **systemctl stop httpd.service**.
     - Attribuez au paramètre **run_after** la commande que vous souhaitez exécuter après l'émission ou le renouvellement de ce certificat, par exemple **systemctl start httpd.service**.
   - Définir le rôle de **rhel-system-roles.certificate** sous **roles**.
     Il s'agit du fichier du playbook pour cet exemple :

```
---
- hosts: webserver
  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
        ca: self-sign
        run_before: systemctl stop httpd.service
        run_after: systemctl start httpd.service

  roles:
    - rhel-system-roles.certificate
```

4. Enregistrer le fichier.

5. Exécutez le manuel de jeu :

```
*ansible-playbook -i inventory.file request-certificate.yml* $ *ansible-playbook -i inventory.file request-certificate.yml* $ *ansible-playbook -i inventory.file
```

Ressources supplémentaires

- Voir le fichier **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md**.
- Voir la page de manuel **ansible-playbook(1)**.

CHAPITRE 19. CONFIGURATION DES CRASH DUMPS AUTOMATIQUES À L'AIDE DU RÔLE DE SYSTÈME RHEL KDUMP

Pour gérer kdump à l'aide d'Ansible, vous pouvez utiliser le rôle **kdump**, qui est l'un des rôles système RHEL disponibles dans RHEL 8.

Le rôle **kdump** vous permet de spécifier l'endroit où enregistrer le contenu de la mémoire du système en vue d'une analyse ultérieure.

Pour plus d'informations sur les rôles système RHEL et leur application, voir Introduction aux rôles système RHEL.

19.1. LE RÔLE DU SYSTÈME RHEL KDUMP

Le rôle de système **kdump** vous permet de définir les paramètres de base du vidage du noyau sur plusieurs systèmes.

19.2. KDUMP PARAMÈTRES DE RÔLE

Les paramètres utilisés pour **kdump** RHEL System Roles sont les suivants :

Variable de rôle	Description
kdump_path	Le chemin dans lequel **vmcore** est écrit. Si **kdump_target** n'est pas nul, le chemin est relatif à cette cible de vidage. Sinon, il doit s'agir d'un chemin absolu dans le système de fichiers racine.

Ressources supplémentaires

- La page de manuel makedumpfile(8).
- Pour plus de détails sur les paramètres utilisés dans **kdump** et des informations supplémentaires sur le rôle de système **kdump**, voir le fichier **/usr/share/ansible/roles/rhel-system-roles.tlog/README.md**.

19.3. CONFIGURATION DE KDUMP À L'AIDE DES RÔLES SYSTÈME RHEL

Vous pouvez définir les paramètres de base du vidage du noyau sur plusieurs systèmes à l'aide du rôle de système **kdump** en exécutant un livre de jeu Ansible.

AVERTISSEMENT

Le rôle **kdump** remplace entièrement la configuration kdump des hôtes gérés en remplaçant le fichier **/etc/kdump.conf**. De plus, si le rôle **kdump** est appliqué, tous les paramètres précédents de **kdump** sont également remplacés, même s'ils ne sont pas spécifiés par les variables de rôle, en remplaçant le fichier **/etc/sysconfig/kdump**.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Vous disposez d'un fichier d'inventaire qui répertorie les systèmes sur lesquels vous souhaitez déployer **kdump**.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- hosts: kdump-test
  vars:
    kdump_path: /var/crash
  roles:
    - rhel-system-roles.kdump
```

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

Ressources supplémentaires

- Pour une référence détaillée sur les variables de rôle **kdump**, voir les fichiers README.md ou README.html dans le répertoire /usr/share/doc/rhel-system-roles/kdump.
- Voir Préparation du nœud de contrôle et des nœuds gérés à l'utilisation des rôles système RHEL
- Documentation installée avec le paquetage **rhel-system-roles /usr/share/ansible/roles/rhel-system-roles.kdump/README.html**

CHAPITRE 20. GESTION DU STOCKAGE LOCAL À L'AIDE DES RÔLES SYSTÈME RHEL

Pour gérer LVM et les systèmes de fichiers locaux (FS) à l'aide d'Ansible, vous pouvez utiliser le rôle **storage**, qui est l'un des rôles système RHEL disponibles dans RHEL 9.

L'utilisation du rôle **storage** vous permet d'automatiser l'administration des systèmes de fichiers sur les disques et les volumes logiques sur plusieurs machines et sur toutes les versions de RHEL à partir de RHEL 7.7.

Pour plus d'informations sur les rôles système RHEL et leur application, voir Introduction aux rôles système RHEL.

20.1. INTRODUCTION AU RÔLE DU SYSTÈME RHEL STORAGE

Le rôle de **storage** peut être géré :

- Systèmes de fichiers sur des disques qui n'ont pas été partitionnés
- Groupes de volumes LVM complets, y compris leurs volumes logiques et leurs systèmes de fichiers
- Volumes RAID MD et leurs systèmes de fichiers

Le rôle **storage** vous permet d'effectuer les tâches suivantes :

- Créer un système de fichiers
- Supprimer un système de fichiers
- Monter un système de fichiers
- Démonter un système de fichiers
- Créer des groupes de volumes LVM
- Supprimer les groupes de volumes LVM
- Créer des volumes logiques
- Supprimer des volumes logiques
- Créer des volumes RAID
- Supprimer des volumes RAID
- Créer des groupes de volumes LVM avec RAID
- Supprimer les groupes de volumes LVM avec RAID
- Créer des groupes de volumes LVM cryptés
- Créer des volumes logiques LVM avec RAID

20.2. PARAMÈTRES QUI IDENTIFIENT UN PÉRIPHÉRIQUE DE STOCKAGE DANS LE RÔLE DE SYSTÈME RHEL STORAGE

Votre configuration du rôle **storage** n'affecte que les systèmes de fichiers, les volumes et les pools que vous avez répertoriés dans les variables suivantes.

storage_volumes

Liste des systèmes de fichiers sur tous les disques non partitionnés à gérer.
storage_volumes peut également inclure des volumes **raid**.

Les partitions ne sont actuellement pas prises en charge.

storage_pools

Liste des pools à gérer.
Actuellement, le seul type de pool pris en charge est LVM. Avec LVM, les pools représentent des groupes de volumes (VG). Sous chaque pool se trouve une liste de volumes à gérer par le rôle. Avec LVM, chaque volume correspond à un volume logique (LV) avec un système de fichiers.

20.3. EXEMPLE DE SCRIPT ANSIBLE POUR CRÉER UN SYSTÈME DE FICHIERS XFS SUR UN PÉRIPHÉRIQUE BLOC

Cette section fournit un exemple de script Ansible. Ce playbook applique le rôle **storage** pour créer un système de fichiers XFS sur un périphérique bloc à l'aide des paramètres par défaut.

AVERTISSEMENT

Le rôle **storage** peut créer un système de fichiers uniquement sur un disque entier non partitionné ou sur un volume logique (LV). Il ne peut pas créer le système de fichiers sur une partition.

Exemple 20.1. Un playbook qui crée XFS sur /dev/sdb

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
  roles:
    - rhel-system-roles.storage
```

- Le nom du volume (***barefs*** dans l'exemple) est actuellement arbitraire. Le rôle **storage** identifie le volume par l'unité de disque répertoriée sous l'attribut **disks:**.

- Vous pouvez omettre la ligne **fs_type: xfs** car XFS est le système de fichiers par défaut dans RHEL 9.
- Pour créer le système de fichiers sur un LV, fournissez la configuration LVM sous l'attribut **disks:**, y compris le groupe de volumes qui l'entoure. Pour plus de détails, voir Example Ansible playbook to manage logical volumes.
  Ne pas fournir le chemin d'accès au dispositif LV.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.4. EXEMPLE DE PLAYBOOK ANSIBLE POUR MONTER UN SYSTÈME DE FICHIERS DE MANIÈRE PERSISTANTE

Cette section fournit un exemple de plan de jeu Ansible. Ce playbook applique le rôle **storage** pour monter immédiatement et de manière persistante un système de fichiers XFS.

Exemple 20.2. Un playbook qui monte un système de fichiers sur /dev/sdb vers /mnt/data

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- Cette procédure ajoute le système de fichiers au fichier **/etc/fstab** et monte immédiatement le système de fichiers.
- Si le système de fichiers sur le périphérique **/dev/sdb** ou le répertoire du point de montage n'existent pas, la séquence les crée.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.5. EXEMPLE DE SCRIPT ANSIBLE POUR LA GESTION DES VOLUMES LOGIQUES

Cette section fournit un exemple de manuel de jeu Ansible. Ce playbook applique le rôle **storage** pour créer un volume logique LVM dans un groupe de volumes.

Exemple 20.3. Un playbook qui crée un volume logique mylv dans le groupe de volumes myvg

```
- hosts: all
  vars:
    storage_pools:
      - name: myvg
        disks:
          - sda
          - sdb
          - sdc
        volumes:
          - name: mylv
            size: 2G
            fs_type: ext4
            mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- Le groupe de volumes **myvg** se compose des disques suivants :
  - /**dev**/**sda**
  - /**dev**/**sdb**
  - /**dev**/**sdc**
- Si le groupe de volumes **myvg** existe déjà, la procédure ajoute le volume logique au groupe de volumes.
- Si le groupe de volumes **myvg** n'existe pas, le playbook le crée.
- La procédure crée un système de fichiers Ext4 sur le volume logique **mylv** et monte de manière persistante le système de fichiers à l'adresse /**mnt**.

Ressources supplémentaires

- Le fichier /**usr**/**share**/**ansible**/**roles**/**rhel-system-roles.storage**/**README.md**.

20.6. EXEMPLE DE SCRIPT ANSIBLE POUR ACTIVER L'ÉLIMINATION DES BLOCS EN LIGNE

Cette section fournit un exemple de manuel de jeu Ansible. Ce playbook applique le rôle **storage** pour monter un système de fichiers XFS avec l'option d'élimination des blocs en ligne activée.

Exemple 20.4. Un playbook qui active l'élimination des blocs en ligne sur /mnt/data/

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
```

```
      mount_options: discard
roles:
  - rhel-system-roles.storage
```

Ressources supplémentaires

- Exemple de playbook Ansible pour monter un système de fichiers de manière persistante
- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.7. EXEMPLE DE SCRIPT ANSIBLE POUR CRÉER ET MONTER UN SYSTÈME DE FICHIERS EXT4

Cette section fournit un exemple de script Ansible. Ce playbook applique le rôle **storage** pour créer et monter un système de fichiers Ext4.

Exemple 20.5. Un playbook qui crée Ext4 sur /dev/sdb et le monte dans /mnt/data

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext4
        fs_label: label-name
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- Le playbook crée le système de fichiers sur le disque **/dev/sdb**.
- Le playbook monte de manière persistante le système de fichiers dans le répertoire ***/mnt/data*** répertoire.
- L'étiquette du système de fichiers est ***label-name***.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.8. EXEMPLE DE SCRIPT ANSIBLE POUR CRÉER ET MONTER UN SYSTÈME DE FICHIERS EXT3

Cette section fournit un exemple de script Ansible. Ce playbook applique le rôle **storage** pour créer et monter un système de fichiers Ext3.

Exemple 20.6. Un playbook qui crée Ext3 sur/dev/sdb et le monte à l'adresse/mnt/data

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext3
        fs_label: label-name
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- Le playbook crée le système de fichiers sur le disque **/dev/sdb**.
- Le playbook monte de manière persistante le système de fichiers dans le répertoire ***/mnt/data*** répertoire.
- L'étiquette du système de fichiers est ***label-name***.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.9. EXEMPLE DE PLAYBOOK ANSIBLE POUR REDIMENSIONNER UN SYSTÈME DE FICHIERS EXT4 OU EXT3 EXISTANT À L'AIDE DU RÔLE DE SYSTÈME RHEL STORAGE

Cette section fournit un exemple de plan de jeu Ansible. Ce playbook applique le rôle **storage** pour redimensionner un système de fichiers Ext4 ou Ext3 existant sur un périphérique bloc.

Exemple 20.7. Un playbook qui configure un seul volume sur un disque

```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
        size: 12 GiB
        fs_type: ext4
        mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

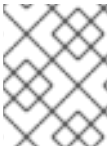
- Si le volume de l'exemple précédent existe déjà, pour redimensionner le volume, vous devez exécuter le même playbook, mais avec une valeur différente pour le paramètre **size**. Par

exemple, vous devez exécuter le même playbook, mais avec une valeur différente pour le paramètre :

Exemple 20.8. Un playbook qui redimensionne ext4 sur /dev/sdb

```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
        size: 10 GiB
        fs_type: ext4
        mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

- Le nom du volume (barefs dans l'exemple) est actuellement arbitraire. Le rôle Stockage identifie le volume par l'unité de disque listée dans l'attribut disks :.



NOTE

L'utilisation de l'action **Resizing** dans d'autres systèmes de fichiers peut détruire les données de l'appareil sur lequel vous travaillez.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.10. EXEMPLE DE PLAYBOOK ANSIBLE POUR REDIMENSIONNER UN SYSTÈME DE FICHIERS EXISTANT SUR LVM À L'AIDE DU RÔLE SYSTÈME STORAGE RHEL

Cette section fournit un exemple de script Ansible. Ce playbook applique le rôle système **storage** RHEL pour redimensionner un volume logique LVM avec un système de fichiers.

AVERTISSEMENT

L'utilisation de l'action **Resizing** dans d'autres systèmes de fichiers peut détruire les données de l'appareil sur lequel vous travaillez.

Exemple 20.9. Un playbook qui redimensionne les volumes logiques mylv1 et myvl2 existants dans le groupe de volumes myvg

```
---

- hosts: all
  vars:
   storage_pools:
     - name: myvg
       disks:
         - /dev/sda
         - /dev/sdb
         - /dev/sdc
       volumes:
          - name: mylv1
            size: 10 GiB
            fs_type: ext4
            mount_point: /opt/mount1
          - name: mylv2
            size: 50 GiB
            fs_type: ext4
            mount_point: /opt/mount2

- name: Create LVM pool over three disks
  include_role:
    name: rhel-system-roles.storage
```

- Cette procédure redimensionne les systèmes de fichiers existants suivants :
  - Le système de fichiers Ext4 sur le volume **mylv1**, qui est monté sur **/opt/mount1**, est redimensionné à 10 GiB.
  - Le système de fichiers Ext4 sur le volume **mylv2**, qui est monté sur **/opt/mount2**, est redimensionné à 50 GiB.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.11. EXEMPLE DE PLAYBOOK ANSIBLE POUR CRÉER UN VOLUME D'ÉCHANGE À L'AIDE DU RÔLE DE SYSTÈME RHEL STORAGE

Cette section fournit un exemple de script Ansible. Ce playbook applique le rôle **storage** pour créer un volume d'échange, s'il n'existe pas, ou pour modifier le volume d'échange, s'il existe déjà, sur un périphérique de bloc en utilisant les paramètres par défaut.

Exemple 20.10. Un playbook qui crée ou modifie un XFS existant sur /dev/sdb

```
---
- name: Create a disk device with swap
- hosts: all
  vars:
    storage_volumes:
      - name: swap_fs
        type: disk
        disks:
```

```
      - /dev/sdb
    size: 15 GiB
      fs_type: swap
  roles:
    - rhel-system-roles.storage
```

- Le nom du volume (***swap_fs*** dans l'exemple) est actuellement arbitraire. Le rôle **storage** identifie le volume par l'unité de disque répertoriée sous l'attribut **disks:**.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.12. CONFIGURATION D'UN VOLUME RAID À L'AIDE DU RÔLE DE SYSTÈME DE STOCKAGE

Avec le rôle de système **storage**, vous pouvez configurer un volume RAID sur RHEL en utilisant Red Hat Ansible Automation Platform et Ansible-Core. Créez un playbook Ansible avec les paramètres pour configurer un volume RAID en fonction de vos besoins.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Vous disposez d'un fichier d'inventaire détaillant les systèmes sur lesquels vous souhaitez déployer un volume RAID à l'aide du rôle de système **storage**.

Procédure

1. Créez un nouveau fichier *playbook.yml* avec le contenu suivant :

```
---
- name: Configure the storage
  hosts: managed-node-01.example.com
  tasks:
  - name: Create a RAID on sdd, sde, sdf, and sdg
    include_role:
      name: rhel-system-roles.storage
    vars:
    storage_safe_mode: false
    storage_volumes:
      - name: data
        type: raid
        disks: [sdd, sde, sdf, sdg]
        raid_level: raid0
        raid_chunk_size: 32 KiB
        mount_point: /mnt/data
        state: present
```

AVERTISSEMENT

Les noms de périphériques peuvent changer dans certaines circonstances, par exemple lorsque vous ajoutez un nouveau disque à un système. Par conséquent, pour éviter toute perte de données, n'utilisez pas de noms de disques spécifiques dans le guide de lecture.

2. Facultatif : Vérifiez la syntaxe du playbook :

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le manuel de jeu :

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**
- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL

20.13. CONFIGURATION D'UN POOL LVM AVEC RAID À L'AIDE DU RÔLE SYSTÈME STORAGE RHEL

Avec le rôle de système **storage**, vous pouvez configurer un pool LVM avec RAID sur RHEL à l'aide de Red Hat Ansible Automation Platform. Dans cette section, vous apprendrez à configurer un playbook Ansible avec les paramètres disponibles pour configurer un pool LVM avec RAID.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Vous disposez d'un fichier d'inventaire détaillant les systèmes sur lesquels vous souhaitez configurer un pool LVM avec RAID à l'aide du rôle de système **storage**.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
- hosts: all
  vars:
    storage_safe_mode: false
    storage_pools:
      - name: my_pool
        type: lvm
        disks: [sdh, sdi]
```

```
      raid_level: raid1
      volumes:
        - name: my_pool
          size: "1 GiB"
          mount_point: "/mnt/app/shared"
          fs_type: xfs
          state: present
  roles:
    - name: rhel-system-roles.storage
```

NOTE

Pour créer un pool LVM avec RAID, vous devez spécifier le type de RAID à l'aide du paramètre **raid_level**.

2. Facultatif. Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.storage/README.md**.

20.14. EXEMPLE DE PLAYBOOK ANSIBLE POUR COMPRESSER ET DÉDUPLIQUER UN VOLUME VDO SUR LVM À L'AIDE DU RÔLE SYSTÈME STORAGE RHEL

Cette section fournit un exemple de plan de jeu Ansible. Ce playbook applique le rôle système **storage** RHEL pour activer la compression et la déduplication des volumes logiques (LVM) à l'aide de Virtual Data Optimizer (VDO).

Exemple 20.11. Un playbook qui crée un volume mylv1 LVM VDO dans le groupe de volumes myvg

```
---
- name: Create LVM VDO volume under volume group 'myvg'
  hosts: all
  roles:
    -rhel-system-roles.storage
  vars:
    storage_pools:
      - name: myvg
        disks:
          - /dev/sdb
        volumes:
          - name: mylv1
            compression: true
            deduplication: true
```

```
    vdo_pool_size: 10 GiB
    size: 30 GiB
    mount_point: /mnt/app/shared
```

Dans cet exemple, les pools **compression** et **deduplication** sont réglés sur true, ce qui spécifie que le VDO est utilisé. Les paragraphes suivants décrivent l'utilisation de ces paramètres :

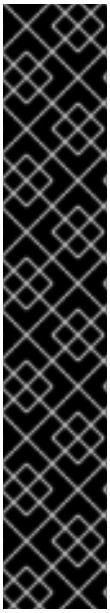
- Le site **deduplication** est utilisé pour dédupliquer les données dupliquées stockées sur le volume de stockage.
- La compression est utilisée pour comprimer les données stockées sur le volume de stockage, ce qui permet d'augmenter la capacité de stockage.
- Le paramètre vdo_pool_size indique la taille réelle du volume sur le périphérique. La taille virtuelle du volume VDO est définie par le paramètre **size**. NOTE : En raison de l'utilisation du rôle de stockage de LVM VDO, un seul volume par pool peut utiliser la compression et la déduplication.

20.15. CRÉATION D'UN VOLUME CHIFFRÉ LUKS2 À L'AIDE DU RÔLE SYSTÈME STORAGE RHEL

Vous pouvez utiliser le rôle **storage** pour créer et configurer un volume chiffré avec LUKS en exécutant un manuel de jeu Ansible.

Conditions préalables

- Accès et autorisations à un ou plusieurs nœuds gérés, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **crypto_policies**.
- Un fichier d'inventaire, qui répertorie les nœuds gérés.
- Accès et permissions à un nœud de contrôle, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes. Sur le nœud de contrôle, les paquets **ansible-core** et **rhel-system-roles** sont installés.



IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
         - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: your-password
  roles:
   - rhel-system-roles.storage
```

   Vous pouvez également ajouter les autres paramètres de cryptage tels que **encryption_key**, **encryption_cipher**, **encryption_key_size**, et **encryption_luks** version dans le fichier *playbook.yml*.

2. Facultatif : Vérifier la syntaxe du playbook :

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

Vérification

1. Visualiser l'état du cryptage :

```
# cryptsetup status sdb

/dev/mapper/sdb is active and is in use.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/sdb
[...]
```

2. Vérifiez le volume crypté LUKS créé :

```
# cryptsetup luksDump /dev/sdb

Version:        2
Epoch:          6
Metadata area:  16384 [bytes]
Keyslots area:  33521664 [bytes]
UUID:           a4c6be82-7347-4a91-a8ad-9479b72c9426
```

```
Label:          (no label)
Subsystem:      (no subsystem)
Flags:        allow-discards

Data segments:
  0: crypt
 offset: 33554432 [bytes]
 length: (whole device)
 cipher: aes-xts-plain64
 sector: 4096 [bytes]
[...]
```

3. Consultez les paramètres **cryptsetup** dans le fichier ***playbook.yml*** que le rôle **storage** prend en charge :

```
# cat ~/playbook.yml

    - hosts: all
      vars:
        storage_volumes:
          - name: foo
            type: disk
            disks:
             - nvme0n1
            fs_type: xfs
            fs_label: label-name
            mount_point: /mnt/data
            encryption: true
            #encryption_password: passwdpasswd
            encryption_key: /home/passwd_key
            encryption_cipher: aes-xts-plain64
            encryption_key_size: 512
            encryption_luks_version: luks2

      roles:
       - rhel-system-roles.storage
```

Ressources supplémentaires

- Chiffrement des blocs de données à l'aide de LUKS
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** fichier

20.16. EXEMPLE DE PLAYBOOK ANSIBLE POUR EXPRIMER LES TAILLES DE VOLUME DE POOL EN POURCENTAGE À L'AIDE DU RÔLE DE SYSTÈME RHEL STORAGE

Cette section fournit un exemple de manuel de jeu Ansible. Ce manuel applique le rôle de système **storage** pour vous permettre d'exprimer la taille des volumes LVM (Logical Manager Volumes) en pourcentage de la taille totale du pool.

Exemple 20.12. Un playbook qui exprime la taille des volumes en pourcentage de la taille totale du pool

```
---
- name: Express volume sizes as a percentage of the pool's total size
  hosts: all
  roles
    - rhel-system-roles.storage
  vars:
    storage_pools:
    - name: myvg
      disks:
        - /dev/sdb
      volumes:
        - name: data
          size: 60%
          mount_point: /opt/mount/data
        - name: web
          size: 30%
          mount_point: /opt/mount/web
        - name: cache
          size: 10%
          mount_point: /opt/cache/mount
```

Cet exemple spécifie la taille des volumes LVM en pourcentage de la taille du pool, par exemple : "60%". En outre, vous pouvez également spécifier la taille des volumes LVM en pourcentage de la taille du pool dans une taille lisible par l'homme du système de fichiers, par exemple : "10g" ou "50 GiB".

20.17. RESSOURCES SUPPLÉMENTAIRES

- **/usr/share/doc/rhel-system-roles/storage/**
- **/usr/share/ansible/roles/rhel-system-roles.storage/**

CHAPITRE 21. CONFIGURATION DE LA SYNCHRONISATION TEMPORELLE À L'AIDE DU RÔLE DE SYSTÈME TIMESYNC RHEL

Avec le rôle de système **timesync** RHEL, vous pouvez gérer la synchronisation temporelle sur plusieurs machines cibles sur RHEL à l'aide de Red Hat Ansible Automation Platform.

21.1. LE RÔLE DU SYSTÈME RHEL TIMESYNC

Vous pouvez gérer la synchronisation temporelle sur plusieurs machines cibles à l'aide du rôle de système RHEL **timesync**.

Le rôle **timesync** installe et configure une implémentation NTP ou PTP pour fonctionner en tant que client NTP ou réplique PTP afin de synchroniser l'horloge du système avec les serveurs NTP ou les grands maîtres des domaines PTP.

Notez que l'utilisation du rôle **timesync** facilite également la migration vers chrony, car vous pouvez utiliser le même playbook sur toutes les versions de Red Hat Enterprise Linux à partir de RHEL 6, que le système utilise **ntp** ou **chrony** pour implémenter le protocole NTP.

21.2. APPLICATION DU RÔLE DE SYSTÈME TIMESYNC POUR UN SEUL GROUPE DE SERVEURS

L'exemple suivant montre comment appliquer le rôle **timesync** dans une situation où il n'y a qu'un seul pool de serveurs.

AVERTISSEMENT

Le rôle **timesync** remplace la configuration du service fournisseur donné ou détecté sur l'hôte géré. Les paramètres précédents sont perdus, même s'ils ne sont pas spécifiés dans les variables du rôle. Le seul paramètre préservé est le choix du fournisseur si la variable **timesync_ntp_provider** n'est pas définie.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Vous disposez d'un fichier d'inventaire qui répertorie les systèmes sur lesquels vous souhaitez déployer **timesync** System Role.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- hosts: timesync-test
```

```
vars:
  timesync_ntp_servers:
    - hostname: 2.rhel.pool.ntp.org
      pool: yes
      iburst: yes
roles:
  - rhel-system-roles.timesync
```

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

21.3. APPLICATION DU RÔLE DE SYSTÈME TIMESYNC SUR LES SERVEURS CLIENTS

Vous pouvez utiliser le rôle **timesync** pour activer la sécurité du temps réseau (NTS) sur les clients NTP. Network Time Security (NTS) est un mécanisme d'authentification spécifié pour le protocole NTP (Network Time Protocol). Il vérifie que les paquets NTP échangés entre le serveur et le client ne sont pas modifiés.

AVERTISSEMENT

Le rôle **timesync** remplace la configuration du service fournisseur donné ou détecté sur l'hôte géré. Les paramètres précédents sont perdus même s'ils ne sont pas spécifiés dans les variables du rôle. Le seul paramètre préservé est le choix du fournisseur si la variable **timesync_ntp_provider** n'est pas définie.

Conditions préalables

- Il n'est pas nécessaire que Red Hat Ansible Automation Platform soit installé sur les systèmes sur lesquels vous souhaitez déployer la solution **timesync**.
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Vous disposez d'un fichier d'inventaire qui répertorie les systèmes sur lesquels vous souhaitez déployer le rôle de système **timesync**.
- La version du fournisseur NTP de **chrony** est 4.0 ou ultérieure.

Procédure

1. Créer un fichier ***playbook.yml*** avec le contenu suivant :

```
---
```

```
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: ptbtime1.ptb.de
        iburst: yes
        nts: yes
  roles:
    - rhel-system-roles.timesync
```

ptbtime1.ptb.de est un exemple de serveur public. Vous pouvez utiliser un autre serveur public ou votre propre serveur.

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

Vérification

1. Effectuez un test sur la machine cliente :

```
# chronyc -N authdata

Name/IP address         Mode KeyID Type KLen Last Atmp  NAK Cook CLen
==============================================================================
ptbtime1.ptb.de         NTS     1   15  256  157    0    0    8  100
```

2. Vérifiez que le nombre de cookies signalés est supérieur à zéro.

Ressources supplémentaires

- **chrony.conf(5)** page de manuel

21.4. TIMESYNC VARIABLES DES RÔLES DU SYSTÈME

Vous pouvez transmettre la variable suivante au rôle **timesync**:

- **timesync_ntp_servers**:

Paramètres des variables de rôle	Description
nom d'hôte : host.example.com	Nom d'hôte ou adresse du serveur
minpoll : *number*	Intervalle d'interrogation minimum. Valeur par défaut : 6
maxpoll : *number*	Intervalle d'interrogation maximal. Valeur par défaut : 10

Paramètres des variables de rôle	Description
iburst : oui	Indicateur permettant une synchronisation initiale rapide. Valeur par défaut : non
piscine : oui	Drapeau indiquant que chaque adresse résolue du nom d'hôte est un serveur NTP distinct. Valeur par défaut : no
nts : oui	Indicateur permettant d'activer la sécurité temporelle du réseau (NTS). Valeur par défaut : no. Pris en charge uniquement avec chrony >= 4.0.

Ressources supplémentaires

- Pour une référence détaillée sur les variables de rôle **timesync**, installez le paquet rhel-system-roles et consultez les fichiers README.md ou README.html dans le répertoire **/usr/share/doc/rhel-system-roles/timesync**.

CHAPITRE 22. CONTRÔLE DES PERFORMANCES À L'AIDE DE METRICS RHEL SYSTEM ROLE

En tant qu'administrateur système, vous pouvez utiliser le rôle système **metrics** RHEL pour surveiller les performances d'un système.

22.1. INTRODUCTION AU RÔLE DU SYSTÈME METRICS

RHEL System Roles est une collection de rôles et de modules Ansible qui fournissent une interface de configuration cohérente pour gérer à distance plusieurs systèmes RHEL. Le rôle système **metrics** configure les services d'analyse des performances pour le système local et, en option, inclut une liste de systèmes distants à surveiller par le système local. Le rôle système **metrics** vous permet d'utiliser **pcp** pour surveiller les performances de vos systèmes sans avoir à configurer **pcp** séparément, car la configuration et le déploiement de **pcp** sont gérés par le playbook.

Tableau 22.1. metrics variables de rôle du système

Variable de rôle	Description	Exemple d'utilisation
métriques_hôtes_surveillés	Liste des hôtes distants à analyser par l'hôte cible. Ces hôtes auront des métriques enregistrées sur l'hôte cible, il faut donc s'assurer qu'il y a suffisamment d'espace disque sous **/var/log** pour chaque hôte.	**metrics_monitored_hosts: ["*webserver.example.com*", "*database.example.com*"]**
jours_de_métriques_de_rétention	Configure le nombre de jours de rétention des données de performance avant leur suppression.	**metrics_retention_days: 14**
service_graphique_métriques	Indicateur booléen qui permet à l'hôte d'être configuré avec des services de visualisation de données de performance via **pcp** et **grafana**. La valeur par défaut est false.	**metrics_graph_service: no**
metrics_query_service	Indicateur booléen qui permet à l'hôte d'être configuré avec des services d'interrogation de séries temporelles pour l'interrogation des mesures enregistrées sur **pcp** via **redis**. La valeur par défaut est false.	**metrics_query_service: no**
fournisseur de métriques	Spécifie le collecteur de métriques à utiliser pour fournir des métriques. Actuellement, **pcp** est le seul fournisseur de métriques pris en charge.	**metrics_provider: "pcp"**

Variable de rôle	Description	Exemple d'utilisation
metrics_manage_firewall	Utilise le rôle **firewall** pour gérer l'accès aux ports directement à partir du rôle **metrics**. La valeur par défaut est false.	**metrics_manage_firewall: true**
metrics_manage_selinux	Utilise le rôle **selinux** pour gérer l'accès aux ports directement à partir du rôle **metrics**. La valeur par défaut est false.	**metrics_manage_selinux: true**

NOTE

Pour plus de détails sur les paramètres utilisés dans **metrics_connections** et des informations supplémentaires sur le rôle de système **metrics**, voir le fichier **/usr/share/ansible/roles/rhel-system-roles.metrics/README.md**.

22.2. UTILISATION DU RÔLE DE SYSTÈME METRICS POUR SURVEILLER VOTRE SYSTÈME LOCAL AVEC VISUALISATION

Cette procédure décrit comment utiliser le rôle de système RHEL **metrics** pour surveiller votre système local tout en fournissant une visualisation des données via **Grafana**.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur la machine que vous voulez surveiller.

Procédure

1. Configurez **localhost** dans l'inventaire Ansible **/etc/ansible/hosts** en ajoutant le contenu suivant à l'inventaire :

```
localhost ansible_connection=local
```

2. Créez un playbook Ansible avec le contenu suivant :

```
---
- name: Manage metrics
  hosts: localhost
  vars:
    metrics_graph_service: yes
    metrics_manage_firewall: true
    metrics_manage_selinux: true
  roles:
    - rhel-system-roles.metrics
```

3. Exécutez le playbook Ansible :

```
# ansible-playbook name_of_your_playbook.yml
```

NOTE

Comme le booléen **metrics_graph_service** est défini sur la valeur="yes", **Grafana** est automatiquement installé et provisionné avec **pcp** ajouté en tant que source de données. Comme metrics_manage_firewall et metrics_manage_selinux sont tous deux définis sur true, le rôle metrics utilisera les rôles système firewall et selinux pour gérer les ports utilisés par le rôle metrics.

4. Pour visualiser les métriques collectées sur votre machine, accédez à l'interface web **grafana** comme décrit dans Accéder à l'interface web Grafana .

22.3. L'UTILISATION DU RÔLE DE SYSTÈME METRICS PERMET DE CONFIGURER UN PARC DE SYSTÈMES INDIVIDUELS POUR QU'ILS SE SURVEILLENT EUX-MÊMES

Cette procédure décrit comment utiliser le rôle de système **metrics** pour configurer une flotte de machines afin qu'elles se surveillent elles-mêmes.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur la machine que vous souhaitez utiliser pour exécuter le playbook.
- La connexion SSH est établie.

Procédure

1. Ajoutez le nom ou l'IP des machines que vous souhaitez surveiller via le playbook au fichier d'inventaire Ansible **/etc/ansible/hosts** sous un nom de groupe d'identification entre parenthèses :

```
[remotes]
webserver.example.com
database.example.com
```

2. Créez un playbook Ansible avec le contenu suivant :

```
---
- hosts: remotes
  vars:
    metrics_retention_days: 0
    metrics_manage_firewall: true
    metrics_manage_selinux: true
  roles:
    - rhel-system-roles.metrics
```

NOTE

Puisque **metrics_manage_firewall** et **metrics_manage_selinux** sont tous deux définis sur true, le rôle metrics utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **metrics**.

3. Exécutez le playbook Ansible :

```
# ansible-playbook name_of_your_playbook.yml -k
```

Lorsque le site **-k** demande un mot de passe pour se connecter au système distant.

22.4. UTILISATION DU RÔLE DE SYSTÈME METRICS POUR SURVEILLER UN PARC DE MACHINES DE MANIÈRE CENTRALISÉE VIA VOTRE MACHINE LOCALE

Cette procédure décrit comment utiliser le rôle de système **metrics** pour configurer votre machine locale afin de surveiller de manière centralisée un parc de machines, tout en prévoyant la visualisation des données via **grafana** et l'interrogation des données via **redis**.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur la machine que vous souhaitez utiliser pour exécuter le playbook.

Procédure

1. Créez un playbook Ansible avec le contenu suivant :

```
---
- hosts: localhost
  vars:
    metrics_graph_service: yes
    metrics_query_service: yes
    metrics_retention_days: 10
    metrics_monitored_hosts: ["database.example.com", "webserver.example.com"]
    metrics_manage_firewall: yes
    metrics_manage_selinux: yes
  roles:
    - rhel-system-roles.metrics
```

2. Exécutez le playbook Ansible :

```
# ansible-playbook name_of_your_playbook.yml
```

NOTE

Étant donné que les booléens **metrics_graph_service** et **metrics_query_service** ont la valeur "yes", **grafana** est automatiquement installé et approvisionné avec **pcp** ajouté en tant que source de données, l'enregistrement des données **pcp** étant indexé dans **redis**, ce qui permet d'utiliser le langage d'interrogation **pcp** pour effectuer des requêtes complexes sur les données. Étant donné que **metrics_manage_firewall** et **metrics_manage_selinux** sont tous deux définis comme vrais, le rôle **metrics** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **metrics**.

3. Pour afficher une représentation graphique des métriques collectées de manière centralisée par votre machine et pour interroger les données, accédez à l'interface web **grafana** comme décrit dans Accéder à l'interface web Grafana .

22.5. CONFIGURATION DE L'AUTHENTIFICATION LORS DE LA SURVEILLANCE D'UN SYSTÈME À L'AIDE DU RÔLE DE SYSTÈME METRICS

PCP prend en charge le mécanisme d'authentification **scram-sha-256** par le biais du cadre SASL (Simple Authentication Security Layer). Le rôle système **metrics** RHEL automatise les étapes de configuration de l'authentification à l'aide du mécanisme d'authentification **scram-sha-256**. Cette procédure décrit comment configurer l'authentification à l'aide du rôle système **metrics** RHEL.

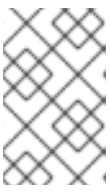
Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur la machine que vous souhaitez utiliser pour exécuter le playbook.

Procédure

1. Incluez les variables suivantes dans le playbook Ansible pour lequel vous souhaitez configurer l'authentification :

```
---
  vars:
    metrics_username: your_username
    metrics_password: your_password
    metrics_manage_firewall: true
    metrics_manage_selinux: true
```



NOTE

Puisque **metrics_manage_firewall** et **metrics_manage_selinux** sont tous deux définis sur true, le rôle **metrics** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **metrics**.

2. Exécutez le playbook Ansible :

```
# ansible-playbook name_of_your_playbook.yml
```

-

Verification steps

- Vérifiez la configuration de **sasl**:

```
# pminfo -f -h "pcp://ip_adress?username=your_username" disk.dev.read
Password:
disk.dev.read
inst [0 or "sda"] value 19540
```

ip_adress doit être remplacé par l'adresse IP de l'hôte.

22.6. UTILISATION DU RÔLE DE SYSTÈME METRICS POUR CONFIGURER ET ACTIVER LA COLLECTE DE MÉTRIQUES POUR SQL SERVER

Cette procédure décrit comment utiliser le rôle système **metrics** RHEL pour automatiser la configuration et l'activation de la collecte de métriques pour Microsoft SQL Server via **pcp** sur votre système local.

Conditions préalables

- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquetage **rhel-system-roles** est installé sur la machine que vous voulez surveiller.
- Vous avez installé Microsoft SQL Server pour Red Hat Enterprise Linux et établi une connexion "fiable" à un serveur SQL. Voir Installer SQL Server et créer une base de données sur Red Hat .
- Vous avez installé le pilote Microsoft ODBC pour SQL Server pour Red Hat Enterprise Linux. Voir Red Hat Enterprise Server et Oracle Linux .

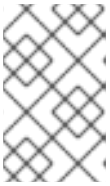
Procédure

1. Configurez **localhost** dans l'inventaire Ansible **/etc/ansible/hosts** en ajoutant le contenu suivant à l'inventaire :

```
localhost ansible_connection=local
```

2. Créez un playbook Ansible contenant les éléments suivants :

```
---
- hosts: localhost
  vars:
   metrics_from_mssql: true
   metrics_manage_firewall: true
   metrics_manage_selinux: true
  roles:
   - role: rhel-system-roles.metrics
```



NOTE

Puisque **metrics_manage_firewall** et **metrics_manage_selinux** sont tous deux définis sur true, le rôle **metrics** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **metrics**.

3. Exécutez le playbook Ansible :

```
# ansible-playbook name_of_your_playbook.yml
```

Verification steps

- Utilisez la commande **pcp** pour vérifier que l'agent PMDA du serveur SQL (mssql) est chargé et en cours d'exécution :

```
# pcp
platform: Linux rhel82-2.local 4.18.0-167.el8.x86_64 #1 SMP Sun Dec 15 01:24:23 UTC
2019 x86_64
 hardware: 2 cpus, 1 disk, 1 node, 2770MB RAM
 timezone: PDT+7
 services: pmcd pmproxy
    pmcd: Version 5.0.2-1, 12 agents, 4 clients
    pmda: root pmcd proc pmproxy xfs linux nfsclient mmv kvm mssql
          jbd2 dm
 pmlogger: primary logger: /var/log/pcp/pmlogger/rhel82-2.local/20200326.16.31
    pmie: primary engine: /var/log/pcp/pmie/rhel82-2.local/pmie.log
```

Ressources supplémentaires

- Pour plus d'informations sur l'utilisation de Performance Co-Pilot pour Microsoft SQL Server, consultez ce billet du Red Hat Developers Blog.

CHAPITRE 23. CONFIGURATION D'UN SYSTÈME POUR L'ENREGISTREMENT DE SESSIONS À L'AIDE DU RÔLE DE SYSTÈME TLOG RHEL

Avec le rôle de système **tlog** RHEL, vous pouvez configurer un système pour l'enregistrement de sessions de terminal sur RHEL à l'aide de Red Hat Ansible Automation Platform.

23.1. LE RÔLE DU SYSTÈME TLOG

Vous pouvez configurer un système RHEL pour l'enregistrement de sessions de terminal sur RHEL à l'aide du rôle de système RHEL **tlog**.

Vous pouvez configurer l'enregistrement par utilisateur ou par groupe d'utilisateurs au moyen du service **SSSD**.

Ressources supplémentaires

- Pour plus de détails sur l'enregistrement de sessions dans RHEL, voir Enregistrement de sessions.

23.2. COMPOSANTS ET PARAMÈTRES DU SYSTÈME TLOG RÔLE

La solution d'enregistrement de session se compose des éléments suivants :

- L'utilitaire **tlog**
- Démon des services de sécurité du système (SSSD)
- En option : L'interface de la console web

Les paramètres utilisés pour le rôle de système **tlog** RHEL sont les suivants :

Variable de rôle	Description
tlog_use_sssd (par défaut : yes)	Configurer l'enregistrement des sessions avec SSSD, la méthode préférée pour gérer les utilisateurs ou les groupes enregistrés
tlog_scope_sssd (par défaut : aucun)	Configurer l'étendue de l'enregistrement SSSD - tous / certains / aucun
tlog_users_sssd (par défaut : [])	Liste YAML des utilisateurs à enregistrer
tlog_groups_sssd (par défaut : [])	Liste YAML des groupes à enregistrer

- Pour plus de détails sur les paramètres utilisés dans **tlog** et des informations supplémentaires sur le rôle de système **tlog**, voir le fichier **/usr/share/ansible/roles/rhel-system-roles.tlog/README.md**.

23.3. DÉPLOIEMENT DU RÔLE DE SYSTÈME TLOG RHEL

Suivez ces étapes pour préparer et appliquer un playbook Ansible afin de configurer un système RHEL pour consigner les données d'enregistrement de session dans le journal systemd.

Conditions préalables

- Vous avez défini des clés SSH pour l'accès du nœud de contrôle au système cible où le rôle de système **tlog** sera configuré.
- Vous avez au moins un système pour lequel vous souhaitez configurer le rôle de système **tlog**.
- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquet **rhel-system-roles** est installé sur la machine de contrôle.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploy session recording
  hosts: all
  vars:
    tlog_scope_sssd: some
    tlog_users_sssd:
      - recorded-user

  roles:
    - rhel-system-roles.tlog
```

   Où ?

   - **tlog_scope_sssd**:
     - **some** spécifie que vous souhaitez enregistrer uniquement certains utilisateurs et groupes, pas **all** ou **none**.
   - **tlog_users_sssd**:
     - **recorded-user** spécifie l'utilisateur à partir duquel vous souhaitez enregistrer une session. Notez que cette fonction n'ajoute pas l'utilisateur à votre place. Vous devez définir l'utilisateur vous-même.

2. Optionnellement, vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

Par conséquent, le playbook installe le rôle système **tlog** RHEL sur le système que vous avez spécifié. Le rôle inclut **tlog-rec-session**, un programme de journalisation des entrées/sorties de session de terminal, qui agit en tant que shell de connexion pour un utilisateur. Il crée également un fichier de dépôt de

configuration SSSD qui peut être utilisé par les utilisateurs et les groupes que vous définissez. SSSD analyse et lit ces utilisateurs et groupes, et remplace leur shell utilisateur par **tlog-rec-session**. En outre, si le paquetage **cockpit** est installé sur le système, le playbook installe également le paquetage **cockpit-session-recording**, qui est un module **Cockpit** vous permettant de visualiser et de lire des enregistrements dans l'interface de la console Web.

Verification steps

Pour vérifier que le fichier de dépôt de configuration SSSD est créé dans le système, procédez comme suit :

1. Naviguez jusqu'au dossier dans lequel le fichier de dépôt de la configuration SSSD a été créé :

```
# cd /etc/sssd/conf.d
```

2. Vérifier le contenu du fichier :

```
# cat /etc/sssd/conf.d/sssd-session-recording.conf
```

Vous pouvez voir que le fichier contient les paramètres que vous avez définis dans le playbook.

23.4. DÉPLOIEMENT DU RÔLE DE SYSTÈME TLOG RHEL POUR L'EXCLUSION DE LISTES DE GROUPES OU D'UTILISATEURS

Vous pouvez utiliser le rôle système **tlog** pour prendre en charge les options de configuration d'enregistrement de session SSSD **exclude_users** et **exclude_groups**. Suivez ces étapes pour préparer et appliquer un playbook Ansible afin de configurer un système RHEL de manière à exclure les utilisateurs ou les groupes dont les sessions sont enregistrées et consignées dans le journal systemd.

Conditions préalables

- Vous avez défini des clés SSH pour l'accès du nœud de contrôle au système cible sur lequel vous souhaitez configurer le rôle de système **tlog**.
- Vous avez au moins un système sur lequel vous voulez configurer le rôle de système **tlog**.
- Le paquetage Ansible Core est installé sur la machine de contrôle.
- Le paquet **rhel-system-roles** est installé sur la machine de contrôle.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- name: Deploy session recording excluding users and groups
  hosts: all
  vars:
    tlog_scope_sssd: all
    tlog_exclude_users_sssd:
      - jeff
      - james
    tlog_exclude_groups_sssd:
      - admins
```

```
roles:
  - rhel-system-roles.tlog
```

Où ?

- **tlog_scope_sssd**:
  - **all**: spécifie que vous voulez enregistrer tous les utilisateurs et tous les groupes.
- **tlog_exclude_users_sssd**:
  - noms d'utilisateur : spécifie les noms d'utilisateur des utilisateurs que vous souhaitez exclure de l'enregistrement de la session.
- **tlog_exclude_groups_sssd**:
  - **admins** spécifie le groupe que vous souhaitez exclure de l'enregistrement de la session.

2. Optionnellement, vérifier la syntaxe du playbook ;

```
# ansible-playbook --syntax-check playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

Par conséquent, le playbook installe le rôle système **tlog** RHEL sur le système que vous avez spécifié. Le rôle inclut **tlog-rec-session**, un programme de journalisation des entrées/sorties de session de terminal, qui agit comme shell de connexion pour un utilisateur. Il crée également un fichier de dépôt de configuration SSSD **/etc/sssd/conf.d/sssd-session-recording.conf** qui peut être utilisé par les utilisateurs et les groupes, à l'exception de ceux que vous avez définis comme exclus. SSSD analyse et lit ces utilisateurs et groupes, et remplace leur shell utilisateur par **tlog-rec-session**. En outre, si le paquet **cockpit** est installé sur le système, le playbook installe également le paquet **cockpit-session-recording**, qui est un module **Cockpit** qui vous permet de visualiser et de lire des enregistrements dans l'interface de la console Web.

Verification steps

Pour vérifier que le fichier de dépôt de configuration SSSD est créé dans le système, procédez comme suit :

1. Naviguez jusqu'au dossier dans lequel le fichier de dépôt de la configuration SSSD a été créé :

```
# cd /etc/sssd/conf.d
```

2. Vérifier le contenu du fichier :

```
# cat sssd-session-recording.conf
```

Vous pouvez voir que le fichier contient les paramètres que vous avez définis dans le playbook.

Ressources supplémentaires

- Voir les répertoires **/usr/share/doc/rhel-system-roles/tlog/** et **/usr/share/ansible/roles/rhel-system-roles.tlog/**.

- L'enregistrement d'une session à l'aide du rôle de système d'enregistrement de session terminal déployé dans le CLI.

23.5. ENREGISTREMENT D'UNE SESSION À L'AIDE DU RÔLE DE SYSTÈME DÉPLOYÉ TLOG DANS L'INTERFACE DE LIGNE DE COMMANDE (CLI)

Après avoir déployé le rôle de système **tlog** dans le système que vous avez spécifié, vous pouvez enregistrer une session de terminal utilisateur à l'aide de l'interface de ligne de commande (CLI).

Conditions préalables

- Vous avez déployé le rôle de système **tlog** dans le système cible.
- Le fichier de dépôt de la configuration SSSD a été créé dans le répertoire **/etc/sssd/conf.d**. Voir Déploiement du rôle de système RHEL d'enregistrement de session terminal .

Procédure

1. Créez un utilisateur et attribuez-lui un mot de passe :

```
# useradd recorded-user
# passwd recorded-user
```

2. Connectez-vous au système en tant qu'utilisateur que vous venez de créer :

```
# ssh recorded-user@localhost
```

3. Tapez "oui" lorsque le système vous demande de taper "oui" ou "non" pour vous authentifier.
4. Insérez le mot de passe *recorded-user's*.
   Le système affiche un message indiquant que votre session est en cours d'enregistrement.

```
ATTENTION! Your session is being recorded!
```

5. Une fois l'enregistrement de la session terminé, tapez :

```
# exit
```

   Le système déconnecte l'utilisateur et ferme la connexion avec le serveur local.

En conséquence, la session de l'utilisateur est enregistrée, stockée et vous pouvez la lire à l'aide d'un journal.

Verification steps

Pour visualiser votre session enregistrée dans le journal, procédez comme suit :

1. Exécutez la commande ci-dessous :

```
# journalctl -o verbose -r
```

2. Recherchez le champ **MESSAGE** de l'écriture enregistrée **tlog-rec**.

```
# journalctl -xel _EXE=/usr/bin/tlog-rec-session
```

23.6. VISIONNER UNE SESSION ENREGISTRÉE À L'AIDE DE L'INTERFACE DE PROGRAMMATION

Vous pouvez lire l'enregistrement d'une session utilisateur à partir d'un journal à l'aide de l'interface de ligne de commande (CLI).

Conditions préalables

- Vous avez enregistré une session utilisateur. Voir Enregistrement d'une session à l'aide du rôle système tlog déployé dans l'interface de programmation.

Procédure

1. Sur le terminal CLI, lisez l'enregistrement de la session utilisateur :

```
# journalctl -o verbose -r
```

2. Recherchez l'enregistrement **tlog**:

```
$ /tlog-rec
```

   Vous pouvez voir des détails tels que

   - Le nom d'utilisateur pour l'enregistrement de la session utilisateur
   - Le champ **out_txt**, un code de sortie brut de la session enregistrée
   - Le numéro d'identification TLOG_REC=*ID_number*

3. Copier le numéro d'identifiant TLOG_REC=*ID_number*.
4. Lire l'enregistrement en utilisant le numéro d'identification TLOG_REC=*ID_number*.

```
# tlog-play -r journal -M TLOG_REC=ID_number
```

Par conséquent, vous pouvez voir la sortie du terminal d'enregistrement de la session de l'utilisateur en cours de lecture.

CHAPITRE 24. CONFIGURATION D'UN CLUSTER À HAUTE DISPONIBILITÉ À L'AIDE DU RÔLE DE SYSTÈME RHEL HA_CLUSTER

Avec le rôle de système **ha_cluster**, vous pouvez configurer et gérer un cluster de haute disponibilité qui utilise le gestionnaire de ressources de cluster de haute disponibilité Pacemaker.

24.1. HA_CLUSTER VARIABLES DE RÔLE DU SYSTÈME

Dans un playbook **ha_cluster** System Role, vous définissez les variables d'un cluster de haute disponibilité en fonction des exigences de votre déploiement de cluster.

Les variables que vous pouvez définir pour un rôle de système **ha_cluster** sont les suivantes.

ha_cluster_enable_repos

Indicateur booléen qui active les référentiels contenant les paquets nécessaires au rôle de système **ha_cluster**. Lorsque cette variable a la valeur **true**(valeur par défaut), vous devez disposer d'une couverture d'abonnement active pour RHEL et le module complémentaire de haute disponibilité RHEL sur les systèmes que vous utiliserez en tant que membres de votre cluster, faute de quoi le rôle de système échouera.

ha_cluster_manage_firewall

(RHEL 9.2 et versions ultérieures) Indicateur booléen qui détermine si le rôle système **ha_cluster** gère le pare-feu. Lorsque **ha_cluster_manage_firewall** vaut **true**, le service de haute disponibilité du pare-feu et le port **fence-virt** sont activés. Lorsque **ha_cluster_manage_firewall** a pour valeur **false**, le rôle système **ha_cluster** ne gère pas le pare-feu. Si votre système exécute le service **firewalld**, vous devez définir le paramètre sur **true** dans votre playbook.
Vous pouvez utiliser le paramètre **ha_cluster_manage_firewall** pour ajouter des ports, mais vous ne pouvez pas l'utiliser pour en supprimer. Pour supprimer des ports, utilisez directement le rôle de système **firewall**.

Depuis RHEL 9.2, le pare-feu n'est plus configuré par défaut, car il n'est configuré que lorsque **ha_cluster_manage_firewall** est défini sur **true**.

ha_cluster_manage_selinux

(RHEL 9.2 et versions ultérieures) Indicateur booléen qui détermine si le rôle système **ha_cluster** gère les ports appartenant au service de haute disponibilité du pare-feu à l'aide du rôle système **selinux**. Lorsque **ha_cluster_manage_selinux** a la valeur **true**, les ports appartenant au service de haute disponibilité du pare-feu sont associés au type de port SELinux **cluster_port_t**. Lorsque **ha_cluster_manage_selinux** est défini sur **false**, le rôle système **ha_cluster** ne gère pas SELinux. Si votre système exécute le service **selinux**, vous devez définir ce paramètre sur **true** dans votre playbook. La configuration du pare-feu est une condition préalable à la gestion de SELinux. Si le pare-feu n'est pas installé, la politique de gestion de SELinux est ignorée.

Vous pouvez utiliser le paramètre **ha_cluster_manage_selinux** pour ajouter une stratégie, mais vous ne pouvez pas l'utiliser pour la supprimer. Pour supprimer une stratégie, utilisez directement le rôle de système **selinux**.

ha_cluster_cluster_present

Un drapeau booléen qui, s'il vaut **true**, détermine que le cluster HA sera configuré sur les hôtes selon les variables passées au rôle. Toute configuration de cluster non spécifiée dans le rôle et non supportée par le rôle sera perdue.

Si **ha_cluster_cluster_present** est défini sur **false**, toute la configuration du cluster HA sera supprimée des hôtes cibles.

La valeur par défaut de cette variable est **true**.

L'exemple suivant supprime toute la configuration du cluster sur **node1** et **node2**

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_present: false

  roles:
    - rhel-system-roles.ha_cluster
```

ha_cluster_start_on_boot

Indicateur booléen qui détermine si les services de cluster seront configurés pour démarrer au démarrage. La valeur par défaut de cette variable est **true**.

ha_cluster_fence_agent_packages

Liste des paquets d'agents de clôture à installer. La valeur par défaut de cette variable est **fence-agents-all**, **fence-virt**.

ha_cluster_extra_packages

Liste des paquets supplémentaires à installer. La valeur par défaut de cette variable est aucun paquet.
Cette variable peut être utilisée pour installer des paquets supplémentaires qui ne sont pas installés automatiquement par le rôle, par exemple des agents de ressources personnalisés.

Il est possible de spécifier des agents de clôture comme membres de cette liste. Toutefois, **ha_cluster_fence_agent_packages** est la variable de rôle qu'il est recommandé d'utiliser pour spécifier les agents de clôture, de sorte que sa valeur par défaut est remplacée.

ha_cluster_hacluster_password

Chaîne de caractères spécifiant le mot de passe de l'utilisateur **hacluster**. L'utilisateur **hacluster** a un accès complet à un cluster. Il est recommandé de crypter le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault . Il n'y a pas de valeur de mot de passe par défaut, et cette variable doit être spécifiée.

ha_cluster_corosync_key_src

Le chemin d'accès au fichier Corosync **authkey**, qui est la clé d'authentification et de cryptage pour la communication Corosync. Il est fortement recommandé d'avoir une valeur **authkey** unique pour chaque cluster. La clé doit être composée de 256 octets de données aléatoires.
Si vous spécifiez une clé pour cette variable, il est recommandé de chiffrer la clé dans la chambre forte, comme décrit dans Chiffrer le contenu avec Ansible Vault .

Si aucune clé n'est spécifiée, une clé déjà présente sur les nœuds sera utilisée. Si les nœuds n'ont pas la même clé, une clé d'un nœud sera distribuée aux autres nœuds afin que tous les nœuds aient la même clé. Si aucun nœud ne possède de clé, une nouvelle clé sera générée et distribuée aux nœuds.

Si cette variable est définie, **ha_cluster_regenerate_keys** est ignoré pour cette touche.

La valeur par défaut de cette variable est null.

ha_cluster_pacemaker_key_src

Le chemin d'accès au fichier Pacemaker **authkey**, qui est la clé d'authentification et de cryptage pour la communication Pacemaker. Il est fortement recommandé d'avoir une valeur **authkey** unique pour chaque cluster. La clé doit être constituée de 256 octets de données aléatoires.
Si vous spécifiez une clé pour cette variable, il est recommandé de chiffrer la clé dans la chambre forte, comme décrit dans Chiffrer le contenu avec Ansible Vault .

Si aucune clé n'est spécifiée, une clé déjà présente sur les nœuds sera utilisée. Si les nœuds n'ont pas la même clé, une clé d'un nœud sera distribuée aux autres nœuds afin que tous les nœuds aient la même clé. Si aucun nœud ne possède de clé, une nouvelle clé sera générée et distribuée aux nœuds.

Si cette variable est définie, **ha_cluster_regenerate_keys** est ignoré pour cette touche.

La valeur par défaut de cette variable est null.

ha_cluster_fence_virt_key_src

Chemin d'accès au fichier de clé prépartagée **fence-virt** ou **fence-xvm**, qui est l'emplacement de la clé d'authentification pour l'agent de clôture **fence-virt** ou **fence-xvm**.
Si vous spécifiez une clé pour cette variable, il est recommandé de chiffrer la clé dans la chambre forte, comme décrit dans Chiffrer le contenu avec Ansible Vault .

Si aucune clé n'est spécifiée, une clé déjà présente sur les nœuds sera utilisée. Si les nœuds n'ont pas la même clé, une clé d'un nœud sera distribuée aux autres nœuds afin que tous les nœuds aient la même clé. Si aucun nœud ne possède de clé, une nouvelle clé est générée et distribuée aux nœuds. Si le rôle de système **ha_cluster** génère une nouvelle clé de cette manière, vous devez copier la clé dans l'hyperviseur de vos nœuds afin de garantir le bon fonctionnement de la clôture.

Si cette variable est définie, **ha_cluster_regenerate_keys** est ignoré pour cette touche.

La valeur par défaut de cette variable est null.

ha_cluster_pcsd_public_key_srcr, ha_cluster_pcsd_private_key_src

Le chemin d'accès au certificat TLS et à la clé privée de **pcsd**. Si ce chemin n'est pas spécifié, une paire certificat-clé déjà présente sur les nœuds sera utilisée. Si ce n'est pas le cas, une nouvelle paire de clés sera générée de manière aléatoire.
Si vous spécifiez une valeur de clé privée pour cette variable, il est recommandé de chiffrer la clé dans la chambre forte, comme décrit dans Chiffrer le contenu avec Ansible Vault .

Si ces variables sont définies, **ha_cluster_regenerate_keys** est ignoré pour cette paire certificat-clé.

La valeur par défaut de ces variables est null.

ha_cluster_pcsd_certificates

(RHEL 9.2 et versions ultérieures) Crée une clé privée et un certificat **pcsd** à l'aide du rôle système **certificate**.
Si votre système n'est pas configuré avec une clé privée et un certificat **pcsd**, vous pouvez les créer de deux manières :

- Définissez la variable **ha_cluster_pcsd_certificates**. Lorsque vous définissez la variable **ha_cluster_pcsd_certificates**, le rôle système **certificate** est utilisé en interne et crée la clé privée et le certificat pour **pcsd** comme défini.
- Ne définissez pas les variables **ha_cluster_pcsd_public_key_src**, **ha_cluster_pcsd_private_key_src** ou **ha_cluster_pcsd_certificates**. Si vous ne définissez aucune de ces variables, le rôle de système **ha_cluster** créera des certificats **pcsd** au moyen

de **pcsd** lui-même. La valeur de **ha_cluster_pcsd_certificates** est fixée à la valeur de la variable **certificate_requests** telle qu'elle est spécifiée dans le rôle système **certificate**. Pour plus d'informations sur le rôle système **certificate**, voir Demande de certificats à l'aide des rôles système RHEL.

Les considérations opérationnelles suivantes s'appliquent à l'utilisation de la variable **ha_cluster_pcsd_certificate**:

- À moins que vous n'utilisiez IPA et que vous ne joigniez les systèmes à un domaine IPA, le rôle système **certificate** crée des certificats auto-signés. Dans ce cas, vous devez configurer explicitement les paramètres de confiance en dehors du contexte des rôles système RHEL. Les rôles système ne prennent pas en charge la configuration des paramètres de confiance.
- Lorsque vous définissez la variable **ha_cluster_pcsd_certificates**, ne définissez pas les variables **ha_cluster_pcsd_public_key_src** et **ha_cluster_pcsd_private_key_src**.
- Lorsque vous définissez la variable **ha_cluster_pcsd_certificates**, **ha_cluster_regenerate_keys** est ignoré pour cette paire certificat-clé.

La valeur par défaut de cette variable est **[]**.
Pour un exemple de playbook **ha_cluster** System Role qui crée des certificats TLS et des fichiers clés dans un cluster à haute disponibilité, voir Création de certificats TLS pcsd et de fichiers clés pour un cluster à haute disponibilité.

ha_cluster_regenerate_keys

Indicateur booléen qui, lorsqu'il vaut **true**, détermine que les clés pré-partagées et les certificats TLS seront régénérés. Pour plus d'informations sur le moment où les clés et les certificats seront régénérés, voir les descriptions des variables **ha_cluster_corosync_key_src**, **ha_cluster_pacemaker_key_src**, **ha_cluster_fence_virt_key_src**, **ha_cluster_pcsd_public_key_src**, et **ha_cluster_pcsd_private_key_src**.

La valeur par défaut de cette variable est **false**.

ha_cluster_pcs_permission_list

Configure les permissions pour gérer un cluster en utilisant **pcsd**. Les éléments que vous configurez avec cette variable sont les suivants :

- **type** - **user** ou **group**
- **name** - nom de l'utilisateur ou du groupe
- **allow_list** - Actions autorisées pour l'utilisateur ou le groupe spécifié :
  - **read** - Visualiser l'état et les paramètres du cluster
  - **write** - Modifier les paramètres du cluster, à l'exception des autorisations et des ACL
  - **grant** - Modifier les permissions et les ACL des clusters
  - **full** - Accès illimité à une grappe, y compris l'ajout et la suppression de nœuds et l'accès aux clés et aux certificats

La structure de la variable **ha_cluster_pcs_permission_list** et ses valeurs par défaut sont les suivantes :

```
ha_cluster_pcs_permission_list:
```

```
- type: group
  name: hacluster
  allow_list:
    - grant
    - read
    - write
```

ha_cluster_cluster_name

Le nom du cluster. Il s'agit d'une chaîne de caractères dont la valeur par défaut est **my-cluster**.

ha_cluster_transport

(RHEL 9.1 et ultérieur) Définit la méthode de transport du cluster. Les éléments que vous configurez avec cette variable sont les suivants :

- **type** (facultatif) - Type de transport : **knet udp udpu** les types de transport **udp** et **udpu** ne supportent qu'un seul lien. Le cryptage est toujours désactivé pour **udp** et **udpu**. La valeur par défaut est **knet** si elle n'est pas spécifiée.
- **options** (facultatif) - Liste de dictionnaires nom-valeur contenant des options de transport.
- **links** (facultatif) - Liste de dictionnaires nom-valeur. Chaque liste de dictionnaires nom-valeur contient des options pour un lien Corosync. Il est recommandé de définir la valeur **linknumber** pour chaque lien. Sinon, la première liste de dictionnaires est attribuée par défaut au premier lien, la deuxième au deuxième lien, et ainsi de suite.
- **compression** (facultatif) - Liste de dictionnaires nom-valeur configurant la compression du transport. Pris en charge uniquement avec le type de transport **knet**.
- **crypto** (facultatif) - Liste de dictionnaires nom-valeur configurant le cryptage du transport. Par défaut, le cryptage est activé. Pris en charge uniquement avec le type de transport **knet**. Pour une liste des options autorisées, voir la page d'aide **pcs -h cluster setup** ou la description **setup** dans la section **cluster** de la page de manuel **pcs**(8). Pour des descriptions plus détaillées, voir la page de manuel **corosync.conf**(5).

  La structure de la variable **ha_cluster_transport** est la suivante :

```
ha_cluster_transport:
  type: knet
  options:
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
  links:
    -
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
    -
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
  compression:
    - name: option1_name
```

```
      value: option1_value
    - name: option2_name
      value: option2_value
  crypto:
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
```

Pour un exemple de playbook **ha_cluster** System Role qui configure une méthode de transport, voir Configuration des valeurs Corosync dans un cluster à haute disponibilité .

ha_cluster_totem

(RHEL 9.1 et ultérieur) Configure le totem Corosync. Pour une liste des options autorisées, voir la page d'aide **pcs -h cluster setup** ou la description **setup** dans la section **cluster** de la page de manuel **pcs**(8). Pour une description plus détaillée, voir la page de manuel **corosync.conf**(5). La structure de la variable **ha_cluster_totem** est la suivante :

```
ha_cluster_totem:
  options:
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
```

Pour un exemple de playbook **ha_cluster** System Role qui configure un totem Corosync, voir Configuration des valeurs Corosync dans un cluster à haute disponibilité .

ha_cluster_quorum

(RHEL 9.1 et versions ultérieures) Configure le quorum du cluster. Vous pouvez configurer les éléments suivants pour le quorum de cluster :

- **options** (facultatif) - Liste de dictionnaires nom-valeur configurant le quorum. Les options autorisées sont : **auto_tie_breaker**, **last_man_standing**, **last_man_standing_window**, et **wait_for_all**. Pour plus d'informations sur les options de quorum, consultez la page de manuel **votequorum**(5).

- **device** (facultatif) - (RHEL 9.2 et versions ultérieures) Configure le cluster pour qu'il utilise un périphérique quorum. Par défaut, aucun périphérique quorum n'est utilisé.

  - **model** (obligatoire) - Spécifie un modèle de périphérique quorum. Seul **net** est pris en charge

  - **model_options** (facultatif) - Liste de dictionnaires nom-valeur configurant le modèle de périphérique quorum spécifié. Pour le modèle **net**, vous devez spécifier les options **host** et **algorithm**.
    Utilisez l'option **pcs-address** pour définir une adresse et un port **pcsd** personnalisés afin de vous connecter à l'hôte **qnetd**. Si vous ne spécifiez pas cette option, le rôle se connecte au port par défaut **pcsd** sur l'hôte **host**.

  - **generic_options** (facultatif) - Liste de dictionnaires nom-valeur définissant les options du périphérique quorum qui ne sont pas spécifiques à un modèle.

  - **heuristics_options** (facultatif) - Liste de dictionnaires nom-valeur configurant l'heuristique des périphériques quorum.

Pour plus d'informations sur les options des périphériques quorum, voir la page de manuel **corosync-qdevice**(8). Les options génériques sont **sync_timeout** et **timeout**. Pour les options du modèle **net**, voir la section **quorum.device.net**. Pour les options heuristiques, voir la section **quorum.device.heuristics**.

Pour régénérer le certificat TLS d'un périphérique quorum, définissez la variable **ha_cluster_regenerate_keys** à **true**.

La structure de la variable **ha_cluster_quorum** est la suivante :

```
ha_cluster_quorum:
  options:
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
  device:
    model: string
    model_options:
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
    generic_options:
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
    heuristics_options:
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
```

Pour un exemple **ha_cluster** System Role playbook qui configure le quorum du cluster, voir Configuration des valeurs Corosync dans un cluster à haute disponibilité . Pour un exemple de manuel de rôle système **ha_cluster** qui configure un cluster à l'aide d'un dispositif de quorum, voir Configuration d'un cluster haute disponibilité à l'aide d'un dispositif de quorum .

ha_cluster_sbd_enabled

(RHEL 9.1 et versions ultérieures) Indicateur booléen qui détermine si le cluster peut utiliser le mécanisme de clôture des nœuds SBD. La valeur par défaut de cette variable est **false**.
Pour un exemple de playbook de rôle de système **ha_cluster** qui active le SBD, voir Configuration d'un cluster à haute disponibilité avec des clôtures de nœuds SBD.

ha_cluster_sbd_options

(RHEL 9.1 et versions ultérieures) Liste de dictionnaires nom-valeur spécifiant les options SBD. Les options prises en charge sont les suivantes :

- **delay-start** - la valeur par défaut est **no**
- **startmode** - la valeur par défaut est **always**
- **timeout-action** - la valeur par défaut est **flush,reboot**

- **watchdog-timeout** - la valeur par défaut est **5**
  Pour plus d'informations sur ces options, voir la section **Configuration via environment** de la page de manuel **sbd**(8).

Pour un exemple de playbook **ha_cluster** System Role qui configure les options SBD, voir Configuration d'un cluster à haute disponibilité avec des clôtures de nœuds SBD .
Lorsque vous utilisez le SBD, vous pouvez éventuellement configurer des dispositifs de chien de garde et de SBD pour chaque nœud d'un inventaire. Pour plus d'informations sur la configuration des dispositifs de surveillance et de SBD dans un fichier d'inventaire, voir Spécification d'un inventaire pour le rôle de système ha_cluster.

ha_cluster_cluster_properties

Liste des ensembles de propriétés de cluster pour la configuration de Pacemaker à l'échelle du cluster. Un seul ensemble de propriétés de cluster est pris en charge.
La structure d'un ensemble de propriétés de grappes est la suivante :

```
ha_cluster_cluster_properties:
  - attrs:
      - name: property1_name
        value: property1_value
      - name: property2_name
        value: property2_value
```

Par défaut, aucune propriété n'est définie.

L'exemple suivant configure un cluster composé de **node1** et **node2** et définit les propriétés des clusters **stonith-enabled** et **no-quorum-policy**.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_cluster_properties:
      - attrs:
          - name: stonith-enabled
            value: 'true'
          - name: no-quorum-policy
            value: stop

  roles:
    - rhel-system-roles.ha_cluster
```

ha_cluster_resource_primitives

Cette variable définit les ressources du pacemaker configurées par le rôle système, y compris les ressources stonith, y compris les ressources stonith. Vous pouvez configurer les éléments suivants pour chaque ressource :

- **id** (obligatoire) - ID d'une ressource.

- **agent** (obligatoire) - Nom d'une ressource ou d'un agent stonith, par exemple **ocf:pacemaker:Dummy** ou **stonith:fence_xvm**. Il est obligatoire de spécifier **stonith:** pour les agents stonith. Pour les agents de ressource, il est possible d'utiliser un nom court, par exemple **Dummy**, au lieu de **ocf:pacemaker:Dummy**. Toutefois, si plusieurs agents portant

le même nom court sont installés, le rôle échouera car il sera incapable de décider quel agent doit être utilisé. Il est donc recommandé d'utiliser des noms complets lors de la spécification d'un agent de ressource.

- **instance_attrs** (facultatif) - Liste d'ensembles d'attributs d'instance de la ressource. Actuellement, un seul ensemble est pris en charge. Les noms et valeurs exacts des attributs, ainsi que leur caractère obligatoire ou non, dépendent de la ressource ou de l'agent stonith.
- **meta_attrs** (facultatif) - Liste d'ensembles d'attributs méta de la ressource. Actuellement, un seul ensemble est pris en charge.
- **operations** (facultatif) - Liste des opérations de la ressource.
  - **action** (obligatoire) - Action d'opération telle que définie par le pacemaker et l'agent ressource ou stonith.
  - **attrs** (obligatoire) - Options d'opération, au moins une option doit être spécifiée.

La structure de la définition des ressources que vous configurez avec le rôle de système **ha_cluster** est la suivante.

```
- id: resource-id
  agent: resource-agent
  instance_attrs:
    - attrs:
        - name: attribute1_name
          value: attribute1_value
        - name: attribute2_name
          value: attribute2_value
  meta_attrs:
    - attrs:
        - name: meta_attribute1_name
          value: meta_attribute1_value
        - name: meta_attribute2_name
          value: meta_attribute2_value
  operations:
    - action: operation1-action
      attrs:
        - name: operation1_attribute1_name
          value: operation1_attribute1_value
        - name: operation1_attribute2_name
          value: operation1_attribute2_value
    - action: operation2-action
      attrs:
        - name: operation2_attribute1_name
          value: operation2_attribute1_value
        - name: operation2_attribute2_name
          value: operation2_attribute2_value
```

Par défaut, aucune ressource n'est définie.

Pour un exemple de playbook **ha_cluster** System Role qui inclut la configuration des ressources, voir Configuration d'un cluster à haute disponibilité avec des clôtures et des ressources .

ha_cluster_resource_groups

Cette variable définit les groupes de ressources de pacemaker configurés par le rôle de système. Vous pouvez configurer les éléments suivants pour chaque groupe de ressources :

- **id** (obligatoire) - ID d'un groupe.
- **resources** (obligatoire) - Liste des ressources du groupe. Chaque ressource est référencée par son ID et les ressources doivent être définies dans la variable **ha_cluster_resource_primitives**. Au moins une ressource doit être listée.
- **meta_attrs** (facultatif) - Liste d'ensembles d'attributs méta du groupe. Actuellement, un seul ensemble est pris en charge.

La structure de la définition du groupe de ressources que vous configurez avec le rôle de système **ha_cluster** est la suivante.

```
ha_cluster_resource_groups:
  - id: group-id
    resource_ids:
      - resource1-id
      - resource2-id
    meta_attrs:
      - attrs:
          - name: group_meta_attribute1_name
            value: group_meta_attribute1_value
          - name: group_meta_attribute2_name
            value: group_meta_attribute2_value
```

Par défaut, aucun groupe de ressources n'est défini.

Pour un exemple de playbook **ha_cluster** System Role qui inclut la configuration des groupes de ressources, voir Configuration d'un cluster à haute disponibilité avec des clôtures et des ressources .

ha_cluster_resource_clones

Cette variable définit les clones de ressources de pacemaker configurés par le rôle de système. Vous pouvez configurer les éléments suivants pour un clone de ressources :

- **resource_id** (obligatoire) - Ressource à cloner. La ressource doit être définie dans la variable **ha_cluster_resource_primitives** ou **ha_cluster_resource_groups**.
- **promotable** (facultatif) - Indique si le clone de ressource à créer est un clone promouvable, indiqué par **true** ou **false**.
- **id** (facultatif) - ID personnalisé du clone. Si aucun identifiant n'est spécifié, il sera généré. Un avertissement s'affiche si cette option n'est pas prise en charge par le cluster.
- **meta_attrs** (facultatif) - Liste d'ensembles d'attributs méta du clone. Actuellement, un seul ensemble est pris en charge.

La structure de la définition du clone de ressources que vous configurez avec le rôle de système **ha_cluster** est la suivante.

```
ha_cluster_resource_clones:
  - resource_id: resource-to-be-cloned
    promotable: true
    id: custom-clone-id
```

```
meta_attrs:
  - attrs:
      - name: clone_meta_attribute1_name
        value: clone_meta_attribute1_value
      - name: clone_meta_attribute2_name
        value: clone_meta_attribute2_value
```

Par défaut, aucun clone de ressources n'est défini.

Pour un exemple de playbook **ha_cluster** System Role qui inclut la configuration des clones de ressources, voir Configuration d'un cluster à haute disponibilité avec des clôtures et des ressources .

ha_cluster_constraints_location

Cette variable définit les contraintes de localisation des ressources. Les contraintes d'emplacement des ressources indiquent les nœuds sur lesquels une ressource peut s'exécuter. Vous pouvez spécifier une ressource par un identifiant de ressource ou par un modèle, qui peut correspondre à plusieurs ressources. Vous pouvez spécifier un nœud par un nom de nœud ou par une règle.
Vous pouvez configurer les éléments suivants pour une contrainte de localisation des ressources :

- **resource** (obligatoire) - Spécification d'une ressource à laquelle la contrainte s'applique.
- **node** (obligatoire) - Nom d'un nœud que la ressource doit préférer ou éviter.
- **id** (facultatif) - ID de la contrainte. S'il n'est pas spécifié, il sera généré automatiquement.
- **options** (facultatif) - Liste de dictionnaires nom-valeur.
  - **score** - Définit le poids de la contrainte.
    - Une valeur positive de **score** signifie que la ressource préfère s'exécuter sur le nœud.
    - Une valeur négative de **score** signifie que la ressource ne doit pas être exécutée sur le nœud.
    - Une valeur de **score** à **-INFINITY** signifie que la ressource ne doit pas être exécutée sur le nœud.
    - Si **score** n'est pas spécifié, la valeur du score est par défaut **INFINITY**.

Par défaut, aucune contrainte de localisation des ressources n'est définie.
La structure d'une contrainte de localisation de ressource spécifiant un identifiant de ressource et un nom de nœud est la suivante :

```
ha_cluster_constraints_location:
  - resource:
      id: resource-id
    node: node-name
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

Les éléments que vous configurez pour une contrainte d'emplacement de ressource qui spécifie un modèle de ressource sont les mêmes que ceux que vous configurez pour une contrainte d'emplacement de ressource qui spécifie un identifiant de ressource, à l'exception de la spécification de ressource elle-même. L'élément que vous spécifiez pour la spécification de ressource est le suivant :

- **pattern** (obligatoire) - Expression régulière POSIX étendue permettant de comparer les identifiants de ressources.

La structure d'une contrainte d'emplacement de ressource spécifiant un modèle de ressource et un nom de nœud est la suivante :

```
ha_cluster_constraints_location:
  - resource:
      pattern: resource-pattern
    node: node-name
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: resource-discovery
        value: resource-discovery-value
```

Vous pouvez configurer les éléments suivants pour une contrainte d'emplacement de ressource qui spécifie un identifiant de ressource et une règle :

- **resource** (obligatoire) - Spécification d'une ressource à laquelle la contrainte s'applique.
  - **id** (obligatoire) - ID de la ressource.
  - **role** (facultatif) - Le rôle de la ressource auquel la contrainte est limitée : **Started**, **Unpromoted**, **Promoted**.
- **rule** (obligatoire) - Règle de contrainte écrite en utilisant la syntaxe **pcs**. Pour plus d'informations, voir la section **constraint location** de la page de manuel **pcs**(8).
- Les autres éléments à spécifier ont la même signification que pour une contrainte de ressource qui ne spécifie pas de règle.

La structure d'une contrainte de localisation des ressources qui spécifie un identifiant de ressource et une règle est la suivante :

```
ha_cluster_constraints_location:
  - resource:
      id: resource-id
      role: resource-role
    rule: rule-string
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: resource-discovery
        value: resource-discovery-value
```

Les éléments que vous configurez pour une contrainte d'emplacement de ressource qui spécifie un modèle de ressource et une règle sont les mêmes que ceux que vous configurez pour une contrainte

d'emplacement de ressource qui spécifie un identifiant de ressource et une règle, à l'exception de la spécification de ressource elle-même. L'élément que vous spécifiez pour la spécification de ressource est le suivant :

- **pattern** (obligatoire) - Expression régulière POSIX étendue permettant de comparer les identifiants de ressources.

La structure d'une contrainte de localisation des ressources qui spécifie un modèle de ressource et une règle est la suivante :

```
ha_cluster_constraints_location:
  - resource:
      pattern: resource-pattern
      role: resource-role
    rule: rule-string
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: resource-discovery
        value: resource-discovery-value
```

Pour un exemple de playbook de rôle système **ha_cluster** qui crée un cluster avec des contraintes de ressources, voir Configuration d'un cluster de haute disponibilité avec des contraintes de ressources.

ha_cluster_constraints_colocation

Cette variable définit les contraintes de colocalisation des ressources. Les contraintes de colocalisation des ressources indiquent que l'emplacement d'une ressource dépend de l'emplacement d'une autre ressource. Il existe deux types de contraintes de colocalisation : une contrainte de colocalisation simple pour deux ressources et une contrainte de colocalisation définie pour plusieurs ressources.
Vous pouvez configurer les éléments suivants pour une contrainte simple de colocation des ressources :

- **resource_follower** (obligatoire) - Une ressource qui doit être située par rapport à **resource_leader**.
  - **id** (obligatoire) - ID de la ressource.
  - **role** (facultatif) - Le rôle de la ressource auquel la contrainte est limitée : **Started**, **Unpromoted**, **Promoted**.
- **resource_leader** (obligatoire) - Le cluster décidera d'abord où placer cette ressource et ensuite où placer **resource_follower**.
  - **id** (obligatoire) - ID de la ressource.
  - **role** (facultatif) - Le rôle de la ressource auquel la contrainte est limitée : **Started**, **Unpromoted**, **Promoted**.
- **id** (facultatif) - ID de la contrainte. S'il n'est pas spécifié, il sera généré automatiquement.
- **options** (facultatif) - Liste de dictionnaires nom-valeur.
  - **score** - Définit le poids de la contrainte.

- Les valeurs positives de **score** indiquent que les ressources doivent être exécutées sur le même nœud.
- Les valeurs négatives de **score** indiquent que les ressources doivent être exécutées sur des nœuds différents.
- Une valeur de **score** à **INFINITY** indique que les ressources doivent être exécutées sur le même nœud.
- Une valeur **score** de **-INFINITY** indique que les ressources doivent être exécutées sur des nœuds différents.
- Si **score** n'est pas spécifié, la valeur du score est par défaut **INFINITY**.

Par défaut, aucune contrainte de colocation des ressources n'est définie.
La structure d'une contrainte simple de colocation des ressources est la suivante :

```
ha_cluster_constraints_colocation:
  - resource_follower:
      id: resource-id1
      role: resource-role1
    resource_leader:
      id: resource-id2
      role: resource-role2
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

Vous pouvez configurer les éléments suivants pour une contrainte de colocalisation d'un ensemble de ressources :

- **resource_sets** (obligatoire) - Liste des ensembles de ressources.
  - **resource_ids** (obligatoire) - Liste des ressources d'un ensemble.
  - **options** (facultatif) - Liste de dictionnaires nom-valeur affinant la manière dont les ressources des ensembles sont traitées par la contrainte.
- **id** (facultatif) - Mêmes valeurs que pour une contrainte de colocalisation simple.
- **options** (facultatif) - Mêmes valeurs que pour une contrainte de colocalisation simple.

La structure d'une contrainte de colocalisation d'un ensemble de ressources est la suivante :

```
ha_cluster_constraints_colocation:
  - resource_sets:
      - resource_ids:
          - resource-id1
          - resource-id2
        options:
          - name: option-name
            value: option-value
    id: constraint-id
```

```
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

Pour un exemple de playbook de rôle système **ha_cluster** qui crée un cluster avec des contraintes de ressources, voir Configuration d'un cluster de haute disponibilité avec des contraintes de ressources.

ha_cluster_constraints_order

Cette variable définit les contraintes d'ordre des ressources. Les contraintes d'ordre des ressources indiquent l'ordre dans lequel certaines actions des ressources doivent se produire. Il existe deux types de contraintes d'ordre pour les ressources : une contrainte d'ordre simple pour deux ressources et une contrainte d'ordre défini pour plusieurs ressources.
Vous pouvez configurer les éléments suivants pour une contrainte d'ordre de ressources simple :

- **resource_first** (obligatoire) - Ressource dont dépend la ressource **resource_then**.
  - **id** (obligatoire) - ID de la ressource.
  - **action** (facultatif) - L'action qui doit être terminée avant qu'une action puisse être lancée pour la ressource **resource_then**. Valeurs autorisées : **start**, **stop**, **promote**, **demote**.
- **resource_then** (obligatoire) - La ressource dépendante.
  - **id** (obligatoire) - ID de la ressource.
  - **action** (facultatif) - Action que la ressource ne peut exécuter qu'une fois l'action sur la ressource **resource_first** terminée. Valeurs autorisées : **start**, **stop**, **promote**, **demote**.
- **id** (facultatif) - ID de la contrainte. S'il n'est pas spécifié, il sera généré automatiquement.
- **options** (facultatif) - Liste de dictionnaires nom-valeur.

Par défaut, aucune contrainte d'ordre de ressources n'est définie.
La structure d'une simple contrainte d'ordre de ressources est la suivante :

```
ha_cluster_constraints_order:
  - resource_first:
      id: resource-id1
      action: resource-action1
    resource_then:
      id: resource-id2
      action: resource-action2
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

Vous pouvez configurer les éléments suivants pour une contrainte de commande d'ensemble de ressources :

- **resource_sets** (obligatoire) - Liste des ensembles de ressources.
  - **resource_ids** (obligatoire) - Liste des ressources d'un ensemble.
  - **options** (facultatif) - Liste de dictionnaires nom-valeur affinant la manière dont les ressources des ensembles sont traitées par la contrainte.
- **id** (facultatif) - Mêmes valeurs que pour une contrainte d'ordre simple.
- **options** (facultatif) - Mêmes valeurs que pour une contrainte d'ordre simple.

La structure d'une contrainte de commande d'un ensemble de ressources est la suivante :

```
ha_cluster_constraints_order:
  - resource_sets:
      - resource_ids:
          - resource-id1
          - resource-id2
        options:
          - name: option-name
            value: option-value
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

Pour un exemple de playbook de rôle système **ha_cluster** qui crée un cluster avec des contraintes de ressources, voir Configuration d'un cluster de haute disponibilité avec des contraintes de ressources.

ha_cluster_constraints_ticket

Cette variable définit les contraintes des tickets de ressources. Les contraintes de ticket de ressource indiquent les ressources qui dépendent d'un certain ticket. Il existe deux types de contraintes de ticket de ressource : une contrainte de ticket simple pour une ressource et une contrainte d'ordre de ticket pour plusieurs ressources.
Vous pouvez configurer les éléments suivants pour une simple contrainte de ticket de ressource :

- **resource** (obligatoire) - Spécification d'une ressource à laquelle la contrainte s'applique.
  - **id** (obligatoire) - ID de la ressource.
  - **role** (facultatif) - Le rôle de la ressource auquel la contrainte est limitée : **Started**, **Unpromoted**, **Promoted**.
- **ticket** (obligatoire) - Nom du ticket dont dépend la ressource.
- **id** (facultatif) - ID de la contrainte. S'il n'est pas spécifié, il sera généré automatiquement.
- **options** (facultatif) - Liste de dictionnaires nom-valeur.
  - **loss-policy** (facultatif) - Action à effectuer sur la ressource si le ticket est révoqué.

Par défaut, aucune contrainte de ticket de ressource n'est définie.
La structure d'une simple contrainte de ticket de ressources est la suivante :

```
ha_cluster_constraints_ticket:
  - resource:
      id: resource-id
      role: resource-role
    ticket: ticket-name
    id: constraint-id
    options:
      - name: loss-policy
        value: loss-policy-value
      - name: option-name
        value: option-value
```

Vous pouvez configurer les éléments suivants pour une contrainte de ticket d'ensemble de ressources :

- **resource_sets** (obligatoire) - Liste des ensembles de ressources.
  - **resource_ids** (obligatoire) - Liste des ressources d'un ensemble.
  - **options** (facultatif) - Liste de dictionnaires nom-valeur affinant la manière dont les ressources des ensembles sont traitées par la contrainte.
- **ticket** (obligatoire) - Même valeur que pour une simple contrainte de ticket.
- **id** (facultatif) - Même valeur que pour une contrainte de ticket simple.
- **options** (facultatif) - Mêmes valeurs que pour une contrainte de ticket simple.

La structure d'une contrainte de ticket d'ensemble de ressources est la suivante :

```
ha_cluster_constraints_ticket:
  - resource_sets:
      - resource_ids:
          - resource-id1
          - resource-id2
        options:
          - name: option-name
            value: option-value
    ticket: ticket-name
    id: constraint-id
    options:
      - name: option-name
        value: option-value
```

Pour un exemple de playbook de rôle système **ha_cluster** qui crée un cluster avec des contraintes de ressources, voir Configuration d'un cluster de haute disponibilité avec des contraintes de ressources.

ha_cluster_qnetd

(RHEL 9.1 et versions ultérieures) Cette variable configure un hôte **qnetd** qui peut ensuite servir de périphérique quorum externe pour les clusters.
Vous pouvez configurer les éléments suivants pour un hôte **qnetd**:

- **present** (optionnel) - Si **true**, configurer une instance **qnetd** sur l'hôte. Si **false**, supprimer la configuration de **qnetd** sur l'hôte. La valeur par défaut est **false**. Si vous définissez **true**, vous devez définir **ha_cluster_cluster_present** à **false**.
- **start_on_boot** (facultatif) - Indique si l'instance **qnetd** doit démarrer automatiquement au démarrage. La valeur par défaut est **true**.
- **regenerate_keys** (facultatif) - Définissez cette variable sur **true** pour régénérer le certificat TLS **qnetd**. Si vous régénérez le certificat, vous devez soit réexécuter le rôle pour chaque cluster afin de le connecter à nouveau à l'hôte **qnetd**, soit exécuter **pcs** manuellement.

Vous ne pouvez pas exécuter **qnetd** sur un nœud de cluster car la clôture perturberait le fonctionnement de **qnetd**.
Pour un exemple de playbook **ha_cluster** System Role qui configure un cluster à l'aide d'un dispositif de quorum, voir Configuration d'un cluster à l'aide d'un dispositif de quorum .

24.2. SPÉCIFIER UN INVENTAIRE POUR LE RÔLE DE SYSTÈME HA_CLUSTER

Lors de la configuration d'un cluster HA à l'aide du playbook **ha_cluster** System Role, vous configurez les noms et adresses des nœuds du cluster dans un inventaire.

24.2.1. Configuration des noms et adresses des nœuds dans un inventaire

Pour chaque nœud d'un inventaire, vous pouvez éventuellement spécifier les éléments suivants :

- **node_name** - le nom d'un nœud dans un cluster.
- **pcs_address** - une adresse utilisée par **pcs** pour communiquer avec le nœud. Il peut s'agir d'un nom, d'un FQDN ou d'une adresse IP, ainsi que d'un numéro de port.
- **corosync_addresses** - la liste des adresses utilisées par Corosync. Tous les nœuds qui forment une grappe particulière doivent avoir le même nombre d'adresses et l'ordre des adresses a de l'importance.

L'exemple suivant montre un inventaire avec les cibles **node1** et **node2**. **node1** et **node2** doivent être des noms de domaine pleinement qualifiés ou doivent pouvoir se connecter aux nœuds comme lorsque, par exemple, les noms peuvent être résolus par le fichier **/etc/hosts**.

```
all:
  hosts:
    node1:
      ha_cluster:
        node_name: node-A
        pcs_address: node1-address
        corosync_addresses:
          - 192.168.1.11
          - 192.168.2.11
    node2:
      ha_cluster:
        node_name: node-B
        pcs_address: node2-address:2224
```

```
    corosync_addresses:
      - 192.168.1.12
      - 192.168.2.12
```

24.2.2. Configuration des périphériques watchdog et SBD dans un inventaire (RHEL 9.1 et suivantes)

Lorsque vous utilisez le SMD, vous pouvez configurer des dispositifs de chien de garde et de SMD pour chaque nœud d'un inventaire. Même si tous les dispositifs SBD doivent être partagés et accessibles depuis tous les nœuds, chaque nœud peut utiliser des noms différents pour les dispositifs. Les dispositifs de surveillance peuvent également être différents pour chaque nœud. Pour plus d'informations sur les variables SBD que vous pouvez définir dans un livre de simulation des rôles système, consultez les entrées **ha_cluster_sbd_enabled** et **ha_cluster_sbd_options** dans **ha_cluster** Variables de rôle système.

Pour chaque nœud d'un inventaire, vous pouvez éventuellement spécifier les éléments suivants :

- **sbd_watchdog** - Périphérique de surveillance à utiliser par SBD. La valeur par défaut est **/dev/watchdog** si elle n'est pas définie.
- **sbd_devices** - Périphériques à utiliser pour l'échange de messages SBD et pour la surveillance. La liste est vide par défaut si elle n'est pas définie.

L'exemple suivant montre un inventaire qui configure les périphériques watchdog et SBD pour les cibles **node1** et **node2**.

```
all:
  hosts:
    node1:
      ha_cluster:
        sbd_watchdog: /dev/watchdog2
        sbd_devices:
          - /dev/vdx
          - /dev/vdy
    node2:
      ha_cluster:
        sbd_watchdog: /dev/watchdog1
        sbd_devices:
          - /dev/vdw
          - /dev/vdz
```

24.3. CRÉATION DE CERTIFICATS TLS PCSD ET DE FICHIERS DE CLÉS POUR UN CLUSTER À HAUTE DISPONIBILITÉ (RHEL 9.2 ET VERSIONS ULTÉRIEURES)

Vous pouvez utiliser le rôle système **ha_cluster** pour créer des certificats TLS et des fichiers clés dans un cluster à haute disponibilité. Lorsque vous exécutez ce playbook, le rôle système **ha_cluster** utilise le rôle système **certificate** en interne pour gérer les certificats TLS.

Conditions préalables

- Les paquets **ansible-core** et **rhel-system-roles** sont installés sur le nœud à partir duquel vous souhaitez exécuter le playbook.

NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.

AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle de système ha_cluster .
2. Créez un fichier playbook, par exemple **new-cluster.yml**.

NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un cluster exécutant les services **firewalld** et **selinux** et crée un certificat **pcsd** auto-signé et des fichiers de clés privées dans **/var/lib/pcsd**. Le certificat **pcsd** porte le nom de fichier **FILENAME.crt** et le fichier de clés est nommé **FILENAME.key**.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_pcsd_certificates:
      - name: FILENAME
        common_name: "{{ ansible_hostname }}"
        ca: self-sign
  roles:
    - linux-system-roles.ha_cluster
```

3. Enregistrer le fichier.
4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

```
# ansible-playbook -i inventory new-cluster.yml
```

Ressources supplémentaires

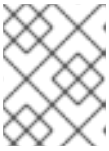
- Demande de certificats à l'aide des rôles système RHEL

24.4. CONFIGURATION D'UN CLUSTER DE HAUTE DISPONIBILITÉ SANS RESSOURCES

La procédure suivante utilise le rôle de système **ha_cluster** pour créer un cluster à haute disponibilité sans configuration de clôture et qui n'utilise aucune ressource.

Conditions préalables

- **ansible-core** est installé sur le nœud à partir duquel vous souhaitez exécuter le playbook.



NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

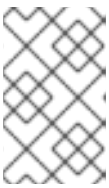
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.

AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle de système ha_cluster .
2. Créez un fichier playbook, par exemple **new-cluster.yml**.



NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un cluster exécutant les services **firewalld** et **selinux** sans clôture configurée et qui n'exécute aucune ressource.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true

  roles:
    - rhel-system-roles.ha_cluster
```

3. Enregistrer le fichier.

4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

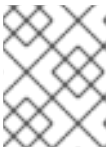
```
# ansible-playbook -i inventory new-cluster.yml
```

24.5. CONFIGURATION D'UN CLUSTER À HAUTE DISPONIBILITÉ AVEC CLÔTURES ET RESSOURCES

La procédure suivante utilise le rôle de système **ha_cluster** pour créer un cluster à haute disponibilité qui comprend un dispositif de clôture, des ressources de cluster, des groupes de ressources et une ressource clonée.

Conditions préalables

- **ansible-core** est installé sur le nœud à partir duquel vous souhaitez exécuter le playbook.



NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.

AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle de système ha_cluster .

2. Créez un fichier playbook, par exemple **new-cluster.yml**.

NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un cluster exécutant les services **firewalld** et **selinux**. Le cluster comprend des clôtures, plusieurs ressources et un groupe de ressources. Il comprend également un clone de ressources pour le groupe de ressources.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_resource_primitives:
      - id: xvm-fencing
        agent: 'stonith:fence_xvm'
        instance_attrs:
          - attrs:
              - name: pcmk_host_list
                value: node1 node2
      - id: simple-resource
        agent: 'ocf:pacemaker:Dummy'
      - id: resource-with-options
        agent: 'ocf:pacemaker:Dummy'
        instance_attrs:
          - attrs:
              - name: fake
                value: fake-value
              - name: passwd
                value: passwd-value
        meta_attrs:
          - attrs:
              - name: target-role
                value: Started
              - name: is-managed
                value: 'true'
        operations:
          - action: start
            attrs:
              - name: timeout
                value: '30s'
          - action: monitor
            attrs:
              - name: timeout
                value: '5'
              - name: interval
                value: '1min'
```

```
    - id: dummy-1
      agent: 'ocf:pacemaker:Dummy'
    - id: dummy-2
      agent: 'ocf:pacemaker:Dummy'
    - id: dummy-3
      agent: 'ocf:pacemaker:Dummy'
    - id: simple-clone
      agent: 'ocf:pacemaker:Dummy'
    - id: clone-with-options
      agent: 'ocf:pacemaker:Dummy'
  ha_cluster_resource_groups:
    - id: simple-group
      resource_ids:
        - dummy-1
        - dummy-2
      meta_attrs:
        - attrs:
            - name: target-role
              value: Started
            - name: is-managed
              value: 'true'
    - id: cloned-group
      resource_ids:
        - dummy-3
  ha_cluster_resource_clones:
    - resource_id: simple-clone
    - resource_id: clone-with-options
      promotable: yes
      id: custom-clone-id
      meta_attrs:
        - attrs:
            - name: clone-max
              value: '2'
            - name: clone-node-max
              value: '1'
    - resource_id: cloned-group
      promotable: yes

roles:
  - rhel-system-roles.ha_cluster
```

3. Enregistrer le fichier.

4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

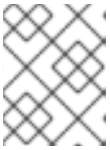
```
# ansible-playbook -i inventory new-cluster.yml
```

24.6. CONFIGURATION D'UN CLUSTER DE HAUTE DISPONIBILITÉ AVEC DES CONTRAINTES DE RESSOURCES

La procédure suivante utilise le rôle système **ha_cluster** pour créer un cluster de haute disponibilité qui inclut des contraintes d'emplacement des ressources, des contraintes de colocation des ressources, des contraintes d'ordre des ressources et des contraintes de ticket des ressources.

Conditions préalables

- **ansible-core** est installé sur le nœud à partir duquel vous souhaitez exécuter le playbook.



NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

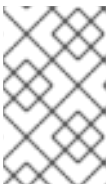
- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.

AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle de système ha_cluster .
2. Créez un fichier playbook, par exemple **new-cluster.yml**.



NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un cluster exécutant les services **firewalld** et **selinux**. Le cluster comprend des contraintes de localisation des ressources, des contraintes de colocation des ressources, des contraintes d'ordre des ressources et des contraintes de ticket des ressources.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    # In order to use constraints, we need resources the constraints will apply
    # to.
    ha_cluster_resource_primitives:
      - id: xvm-fencing
        agent: 'stonith:fence_xvm'
```

```
    instance_attrs:
      - attrs:
          - name: pcmk_host_list
            value: node1 node2
  - id: dummy-1
    agent: 'ocf:pacemaker:Dummy'
  - id: dummy-2
    agent: 'ocf:pacemaker:Dummy'
  - id: dummy-3
    agent: 'ocf:pacemaker:Dummy'
  - id: dummy-4
    agent: 'ocf:pacemaker:Dummy'
  - id: dummy-5
    agent: 'ocf:pacemaker:Dummy'
  - id: dummy-6
    agent: 'ocf:pacemaker:Dummy'
# location constraints
ha_cluster_constraints_location:
  # resource ID and node name
  - resource:
      id: dummy-1
    node: node1
    options:
      - name: score
        value: 20
  # resource pattern and node name
  - resource:
      pattern: dummy-\d+
    node: node1
    options:
      - name: score
        value: 10
  # resource ID and rule
  - resource:
      id: dummy-2
    rule: '#uname eq node2 and date in_range 2022-01-01 to 2022-02-28'
  # resource pattern and rule
  - resource:
      pattern: dummy-\d+
    rule: node-type eq weekend and date-spec weekdays=6-7
# colocation constraints
ha_cluster_constraints_colocation:
  # simple constraint
  - resource_leader:
      id: dummy-3
    resource_follower:
      id: dummy-4
    options:
      - name: score
        value: -5
  # set constraint
  - resource_sets:
      - resource_ids:
          - dummy-1
          - dummy-2
      - resource_ids:
```

```
          - dummy-5
          - dummy-6
        options:
          - name: sequential
            value: "false"
    options:
      - name: score
        value: 20
# order constraints
ha_cluster_constraints_order:
  # simple constraint
  - resource_first:
      id: dummy-1
    resource_then:
      id: dummy-6
    options:
      - name: symmetrical
        value: "false"
  # set constraint
  - resource_sets:
      - resource_ids:
          - dummy-1
          - dummy-2
        options:
          - name: require-all
            value: "false"
          - name: sequential
            value: "false"
      - resource_ids:
          - dummy-3
      - resource_ids:
          - dummy-4
          - dummy-5
        options:
          - name: sequential
            value: "false"
# ticket constraints
ha_cluster_constraints_ticket:
  # simple constraint
  - resource:
      id: dummy-1
    ticket: ticket1
    options:
      - name: loss-policy
        value: stop
  # set constraint
  - resource_sets:
      - resource_ids:
          - dummy-3
          - dummy-4
          - dummy-5
    ticket: ticket2
    options:
      - name: loss-policy
        value: fence
```

```
roles:
  - linux-system-roles.ha_cluster
```

3. Enregistrer le fichier.

4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

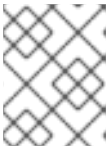
```
# ansible-playbook -i inventory new-cluster.yml
```

24.7. CONFIGURATION DES VALEURS COROSYNC DANS UN CLUSTER À HAUTE DISPONIBILITÉ

(RHEL 9.1 et versions ultérieures) La procédure suivante utilise le rôle de système **ha_cluster** pour créer un cluster à haute disponibilité qui configure les valeurs Corosync.

Conditions préalables

- **ansible-core** est installé sur le nœud à partir duquel vous souhaitez exécuter le playbook.



NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.

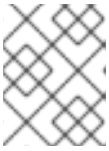
AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle de système ha_cluster .

2. Créez un fichier playbook, par exemple **new-cluster.yml**.



NOTE

Lors de la création de votre fichier playbook pour la production, Vault crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault .

L'exemple de fichier playbook suivant configure un cluster exécutant les services **firewalld** et **selinux** qui configure les propriétés Corosync.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_transport:
      type: knet
      options:
        - name: ip_version
          value: ipv4-6
        - name: link_mode
          value: active
      links:
        -
          - name: linknumber
            value: 1
          - name: link_priority
            value: 5
        -
          - name: linknumber
            value: 0
          - name: link_priority
            value: 10
    compression:
      - name: level
        value: 5
      - name: model
        value: zlib
    crypto:
      - name: cipher
        value: none
      - name: hash
        value: none
    ha_cluster_totem:
      options:
        - name: block_unlisted_ips
          value: 'yes'
        - name: send_join
          value: 0
    ha_cluster_quorum:
      options:
        - name: auto_tie_breaker
          value: 1
        - name: wait_for_all
          value: 1
```

```
roles:
  - linux-system-roles.ha_cluster
```

3. Enregistrer le fichier.

4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

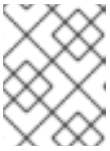
```
# ansible-playbook -i inventory new-cluster.yml
```

24.8. CONFIGURATION D'UN CLUSTER À HAUTE DISPONIBILITÉ AVEC DES CLÔTURES DE NŒUDS SBD

(RHEL 9.1 et versions ultérieures) La procédure suivante utilise le rôle de système **ha_cluster** pour créer un cluster à haute disponibilité qui utilise la clôture des nœuds SBD.

Conditions préalables

- **ansible-core** est installé sur le nœud à partir duquel vous souhaitez exécuter le playbook.



NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.

AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle système ha_cluster . Vous pouvez éventuellement configurer des périphériques watchdog et SBD pour chaque nœud de la grappe dans un fichier d'inventaire.

2. Créez un fichier playbook, par exemple **new-cluster.yml**.

NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un cluster exécutant les services **firewalld** et **selinux** qui utilise la clôture SBD.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_sbd_enabled: yes
    ha_cluster_sbd_options:
      - name: delay-start
        value: 'no'
      - name: startmode
        value: always
      - name: timeout-action
        value: 'flush,reboot'
      - name: watchdog-timeout
        value: 5

  roles:
    - linux-system-roles.ha_cluster
```

3. Enregistrer le fichier.

4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

```
# ansible-playbook -i inventory new-cluster.yml
```

24.9. CONFIGURATION D'UN CLUSTER DE HAUTE DISPONIBILITÉ À L'AIDE D'UN DISPOSITIF QUORUM (RHEL 9.2 ET VERSIONS ULTÉRIEURES)

Pour configurer un cluster de haute disponibilité avec un dispositif de quorum séparé en utilisant le rôle de système **ha_cluster**, vous devez d'abord configurer le dispositif de quorum. Une fois le dispositif quorum configuré, vous pouvez l'utiliser dans un nombre illimité de clusters.

24.9.1. Configuration d'un dispositif de quorum

Pour configurer un dispositif quorum à l'aide du rôle système **ha_cluster**, procédez comme suit. Notez que vous ne pouvez pas exécuter un dispositif quorum sur un nœud de cluster.

Conditions préalables

- Les paquets **ansible-core** et **rhel-system-roles** sont installés sur le nœud à partir duquel vous souhaitez exécuter le playbook.

NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

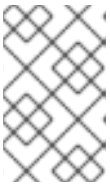
- Le système que vous utiliserez pour exécuter le dispositif de quorum dispose d'une couverture d'abonnement active pour RHEL et le module complémentaire de haute disponibilité RHEL.

AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier playbook, par exemple **qdev-playbook.yml**.



NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un dispositif de quorum sur un système exécutant les services **firewalld** et **selinux**.

```
- hosts: nodeQ
  vars:
    ha_cluster_cluster_present: false
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_qnetd:
      present: true

  roles:
    - linux-system-roles.ha_cluster
```

2. Enregistrer le fichier.

3. Exécutez le manuel de jeu, en spécifiant le nœud hôte pour le dispositif quorum.

```
# ansible-playbook -i nodeQ, qdev-playbook.yml
```

24.9.2. Configuration d'une grappe pour l'utilisation d'un dispositif quorum

Pour configurer une grappe afin d'utiliser un dispositif de quorum, procédez comme suit.

Conditions préalables

- **ansible-core** est installé sur le nœud à partir duquel vous souhaitez exécuter le playbook.

NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.
- Vous avez configuré un dispositif quorum.

AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle de système ha_cluster .
2. Créez un fichier playbook, par exemple **new-cluster.yml**.

NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un cluster exécutant les services **firewalld** et **selinux** qui utilise un périphérique quorum.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_quorum:
      device:
        model: net
        model_options:
          - name: host
```

```
      value: nodeQ
    - name: algorithm
      value: lms

roles:
  - linux-system-roles.ha_cluster
```

3. Enregistrer le fichier.

4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

```
# ansible-playbook -i inventory new-cluster.yml
```

24.10. CONFIGURATION D'UN SERVEUR HTTP APACHE DANS UN CLUSTER À HAUTE DISPONIBILITÉ AVEC LE RÔLE DE SYSTÈME HA_CLUSTER

Cette procédure configure un serveur HTTP Apache actif/passif dans un cluster Red Hat Enterprise Linux High Availability Add-On à deux nœuds à l'aide du rôle de système **ha_cluster**.

Conditions préalables

- **ansible-core** est installé sur le nœud à partir duquel vous souhaitez exécuter le playbook.

NOTE

Il n'est pas nécessaire que **ansible-core** soit installé sur les nœuds membres du cluster.

- Le paquetage **rhel-system-roles** est installé sur le système à partir duquel vous souhaitez exécuter le playbook.
- Les systèmes que vous utiliserez comme membres de votre cluster disposent d'une couverture d'abonnement active pour RHEL et RHEL High Availability Add-On.
- Votre système comprend une adresse IP virtuelle publique, nécessaire pour Apache.
- Votre système comprend un stockage partagé pour les nœuds de la grappe, utilisant iSCSI, Fibre Channel ou un autre périphérique de bloc réseau partagé.
- Vous avez configuré un volume logique LVM avec un système de fichiers XFS, comme décrit dans Configuration d'un volume LVM avec un système de fichiers XFS dans un cluster Pacemaker.
- Vous avez configuré un serveur HTTP Apache, comme décrit dans la section Configuration d'un serveur HTTP Apache.
- Votre système comprend un interrupteur d'alimentation APC qui sera utilisé pour clôturer les nœuds du cluster.

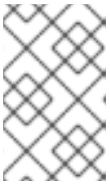
AVERTISSEMENT

Le rôle de système **ha_cluster** remplace toute configuration de cluster existante sur les nœuds spécifiés. Tous les paramètres non spécifiés dans le rôle seront perdus.

Procédure

1. Créez un fichier d'inventaire spécifiant les nœuds de la grappe, comme décrit dans la section Spécification d'un inventaire pour le rôle de système ha_cluster .

2. Créez un fichier playbook, par exemple **http-cluster.yml**.



NOTE

Lors de la création de votre fichier playbook pour la production, le coffre-fort crypte le mot de passe, comme décrit dans Cryptage du contenu avec Ansible Vault.

L'exemple de fichier playbook suivant configure un serveur HTTP Apache créé précédemment dans un cluster HA actif/passif à deux nœuds exécutant les services **firewalld** et **selinux**.

Cet exemple utilise un commutateur d'alimentation APC dont le nom d'hôte est **zapc.example.com**. Si le cluster n'utilise pas d'autres agents de clôture, vous pouvez éventuellement lister uniquement les agents de clôture dont votre cluster a besoin lors de la définition de la variable **ha_cluster_fence_agent_packages**, comme dans cet exemple.

```
- hosts: z1.example.com z2.example.com
  roles:
    - rhel-system-roles.ha_cluster
  vars:
    ha_cluster_hacluster_password: password
    ha_cluster_cluster_name: my_cluster
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_fence_agent_packages:
      - fence-agents-apc-snmp
    ha_cluster_resource_primitives:
      - id: myapc
        agent: stonith:fence_apc_snmp
        instance_attrs:
          - attrs:
              - name: ipaddr
                value: zapc.example.com
              - name: pcmk_host_map
                value: z1.example.com:1;z2.example.com:2
              - name: login
                value: apc
              - name: passwd
                value: apc
      - id: my_lvm
```

```
      agent: ocf:heartbeat:LVM-activate
      instance_attrs:
        - attrs:
            - name: vgname
              value: my_vg
            - name: vg_access_mode
              value: system_id
    - id: my_fs
      agent: Filesystem
      instance_attrs:
        - attrs:
            - name: device
              value: /dev/my_vg/my_lv
            - name: directory
              value: /var/www
            - name: fstype
              value: xfs
    - id: VirtualIP
      agent: IPaddr2
      instance_attrs:
        - attrs:
            - name: ip
              value: 198.51.100.3
            - name: cidr_netmask
              value: 24
    - id: Website
      agent: apache
      instance_attrs:
        - attrs:
            - name: configfile
              value: /etc/httpd/conf/httpd.conf
            - name: statusurl
              value: http://127.0.0.1/server-status
  ha_cluster_resource_groups:
    - id: apachegroup
      resource_ids:
        - my_lvm
        - my_fs
        - VirtualIP
        - Website
```

3. Enregistrer le fichier.

4. Exécutez le playbook en indiquant le chemin d'accès au fichier d'inventaire *inventory* que vous avez créé à l'étape 1.

```
# ansible-playbook -i inventory http-cluster.yml
```

5. Lorsque vous utilisez l'agent de ressources **apache** pour gérer Apache, il n'utilise pas **systemd**. Pour cette raison, vous devez modifier le script **logrotate** fourni avec Apache afin qu'il n'utilise pas **systemctl** pour recharger Apache.
Supprimez la ligne suivante dans le fichier **/etc/logrotate.d/httpd** sur chaque nœud du cluster.

```
/bin/systemctl reload httpd.service > /dev/null 2>/dev/null || true
```

Remplacez la ligne que vous avez supprimée par les trois lignes suivantes, en spécifiant **/var/run/httpd-*website*.pid** comme chemin d'accès au fichier PID où *website* est le nom de la ressource Apache. Dans cet exemple, le nom de la ressource Apache est **Website**.

```
/usr/bin/test -f /var/run/httpd-Website.pid >/dev/null 2>/dev/null &&
/usr/bin/ps -q $(/usr/bin/cat /var/run/httpd-Website.pid) >/dev/null 2>/dev/null &&
/usr/sbin/httpd -f /etc/httpd/conf/httpd.conf -c "PidFile /var/run/httpd-Website.pid" -k graceful >
/dev/null 2>/dev/null || true
```

Verification steps

1. Depuis l'un des nœuds de la grappe, vérifiez l'état de la grappe. Notez que les quatre ressources sont exécutées sur le même nœud, **z1.example.com**.
   Si vous constatez que les ressources que vous avez configurées ne fonctionnent pas, vous pouvez exécuter la commande **pcs resource debug-start *resource*** pour tester la configuration des ressources.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 16:38:51 2013
Last change: Wed Jul 31 16:42:14 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: apachegroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z1.example.com
     my_fs      (ocf::heartbeat:Filesystem):    Started z1.example.com
     VirtualIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
     Website    (ocf::heartbeat:apache):        Started z1.example.com
```

2. Une fois que le cluster est opérationnel, vous pouvez diriger un navigateur vers l'adresse IP que vous avez définie comme ressource **IPaddr2** pour voir l'exemple d'affichage, qui consiste en un simple mot "Hello".

```
Bonjour
```

3. Pour vérifier si le groupe de ressources s'exécutant sur **z1.example.com** est transféré au nœud **z2.example.com**, mettez le nœud **z1.example.com** en mode **standby**, après quoi le nœud ne sera plus en mesure d'héberger des ressources.

```
[root@z1 ~]# pcs node standby z1.example.com
```

4. Après avoir mis le nœud **z1** en mode **standby**, vérifiez l'état de la grappe à partir de l'un des nœuds de la grappe. Notez que les ressources devraient maintenant toutes être exécutées sur **z2**.

```
[root@z1 ~]# pcs status
```

```
Cluster name: my_cluster
Last updated: Wed Jul 31 17:16:17 2013
Last change: Wed Jul 31 17:18:34 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured

Node z1.example.com (1): standby
Online: [ z2.example.com ]

Full list of resources:

 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: apachegroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z2.example.com
     my_fs      (ocf::heartbeat:Filesystem):    Started z2.example.com
     VirtualIP  (ocf::heartbeat:IPaddr2):       Started z2.example.com
     Website    (ocf::heartbeat:apache):        Started z2.example.com
```

Le site web à l'adresse IP définie doit s'afficher sans interruption.

5. Pour supprimer **z1** du mode **standby**, entrez la commande suivante.

```
[root@z1 ~]# pcs node unstandby z1.example.com
```

NOTE

Le retrait d'un nœud du mode **standby** n'entraîne pas en soi le basculement des ressources vers ce nœud. Cela dépend de la valeur de **resource-stickiness** pour les ressources. Pour plus d'informations sur le méta-attribut **resource-stickiness**, voir Configurer une ressource pour qu'elle préfère son nœud actuel .

24.11. RESSOURCES SUPPLÉMENTAIRES

- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL .
- Documentation installée avec le paquetage **rhel-system-roles** dans **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**
- Article de la base de données RHEL System Roles
- La page de manuel **ansible-playbook(1)**.

CHAPITRE 25. INSTALLATION ET CONFIGURATION DE LA CONSOLE WEB AVEC LE COCKPIT RHEL SYSTEM ROLE

Avec le rôle de système **cockpit** RHEL, vous pouvez installer et configurer la console web dans votre système.

25.1. LE RÔLE DU SYSTÈME COCKPIT

Vous pouvez utiliser le rôle de système **cockpit** pour déployer et activer automatiquement la console web et ainsi pouvoir gérer vos systèmes RHEL à partir d'un navigateur web.

25.2. VARIABLES POUR LE RÔLE DE SYSTÈME COCKPIT RHEL

Les paramètres utilisés pour **cockpit** RHEL System Roles sont les suivants :

Variable de rôle	Description
cockpit_packages : (default : default)	Définit l'un des ensembles de paquets prédéfinis : default, minimal ou full. * cockpit_packages : (default : default) - pages les plus courantes et interface d'installation à la demande * cockpit_packages : (default : minimal) - seulement les pages Overview, Terminal, Logs, Accounts et Metrics ; dépendances minimales * cockpit_packages : (par défaut : full) - toutes les pages disponibles En option, spécifiez votre propre sélection de paquets de cockpit que vous souhaitez installer.
cockpit_enabled : (default:true)	Configure si le serveur web de la console web est activé pour démarrer automatiquement au démarrage
cockpit_started : (par défaut:true)	Configure si la console web doit être démarrée
cockpit_config : (par défaut : rien)	Vous pouvez appliquer les paramètres du fichier **/etc/cockpit/cockpit.conf**. REMARQUE : Le fichier de paramètres précédent sera perdu.
cockpit_port : (par défaut : 9090)	La console web fonctionne par défaut sur le port 9090. Vous pouvez modifier le port à l'aide de cette option.

Variable de rôle	Description
cockpit_manage_firewall : (default : false)	Permet au rôle **cockpit** de contrôler le rôle **firewall** pour ajouter des ports. Il ne peut pas être utilisé pour supprimer des ports. Si vous souhaitez supprimer des ports, vous devez utiliser directement le rôle de système de pare-feu.
cockpit_manage_selinux : (par défaut : false)	Permet au rôle **cockpit** de configurer SELinux en utilisant le rôle **selinux**. La politique SELinux par défaut n'autorise pas Cockpit à écouter sur un autre port que le port 9090. Si vous changez le port, mettez cette option à **true** pour que le rôle **selinux** puisse définir les permissions correctes sur le port (**websm_port_t**).
cockpit_certificates : (par défaut : rien)	Permet au rôle **cockpit** de générer de nouveaux certificats en utilisant le rôle **certificate**. La valeur de **cockpit_certificates** est transmise à la variable **certificate_requests** du rôle **certificate**. Ce rôle est appelé en interne par le rôle **cockpit** et génère la clé privée et le certificat.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.cockpit/README.md**.
- La page de manuel du fichier de configuration du Cockpit .

25.3. INSTALLATION DE LA CONSOLE WEB À L'AIDE DU RÔLE SYSTÈME COCKPIT RHEL

Vous pouvez utiliser le rôle de système **cockpit** pour installer et activer la console web RHEL.

Par défaut, la console Web RHEL utilise un certificat auto-signé. Pour des raisons de sécurité, vous pouvez spécifier un certificat émis par une autorité de certification de confiance.

Dans cet exemple, vous utilisez le rôle de système **cockpit** pour :

- Installez la console web RHEL.
- Autoriser la console web à gérer **firewalld**.
- Configurez la console web pour qu'elle utilise un certificat de l'autorité de certification approuvée **ipa** au lieu d'un certificat auto-signé.
- Configurer la console web pour qu'elle utilise un port personnalisé 9050.

NOTE

Vous n'avez pas besoin d'appeler les rôles système **firewall** ou **certificate** dans le livre de jeu pour gérer le pare-feu ou créer le certificat. Le rôle système **cockpit** les appelle automatiquement si nécessaire.

Conditions préalables

- Accès et autorisations à une ou plusieurs *managed nodes*.
- Accès et autorisations à *control node*.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.
  - Il existe un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- hosts: all
  tasks:
    - name: Install RHEL web console
      include_role:
        name: rhel-system-roles.cockpit
      vars:
        cockpit_packages: default
        #cockpit_packages: minimal
        #cockpit_packages: full
        cockpit_port:9050
        cockpit_manage_selinux: true
        cockpit_manage_firewall: true
        cockpit_certificates:
          - name: /etc/cockpit/ws-certs.d/01-certificate
            dns: ['localhost', 'www.example.com']
            ca: ipa
            group: cockpit-ws
```

2. Facultatif : Vérifiez la syntaxe du playbook :

```
# ansible-playbook --syntax-check -i inventory_file playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

Ressources supplémentaires

- Installation et activation de la console web .
- Demande de certificats à l'aide des rôles système RHEL .

CHAPITRE 26. GÉRER LES CONTENEURS EN UTILISANT LE RÔLE DE SYSTÈME RHEL PODMAN

Avec le rôle système **podman** RHEL, vous pouvez gérer la configuration de Podman, les conteneurs et les services **systemd** qui exécutent les conteneurs Podman.

26.1. LE RÔLE DE SYSTÈME RHEL PODMAN

Vous pouvez utiliser le rôle système **podman** RHEL pour gérer la configuration de Podman, les conteneurs et les services systemd qui exécutent les conteneurs Podman.

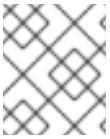
Ressources supplémentaires

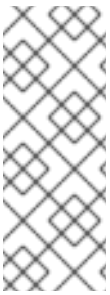
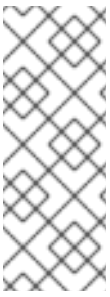
- Installation des rôles système RHEL
- Pour plus de détails sur les paramètres utilisés dans **podman** et des informations supplémentaires sur le rôle de système RHEL **podman**, voir le fichier **/usr/share/ansible/roles/rhel-system-roles.podman/README.md**.

26.2. VARIABLES POUR LE RÔLE DE SYSTÈME RHEL PODMAN

Les paramètres utilisés pour le rôle de système **podman** RHEL sont les suivants :

Variable	Description

podman_kube_spec	Décrit un pod podman et l'unité systemd correspondante à gérer.
	• **state**: (default : **created**) - indique une opération à exécuter avec les services et pods **systemd**:
	  ◦ **created**: créer les pods et le service **systemd**, mais ne pas les exécuter
	  ◦ **started**: créer les services pods et **systemd** et les démarrer
	  ◦ **absent**: supprimer les pods et les services **systemd**
	• **run_as_user**: (default : **podman_run_as_user**) - un utilisateur par pod. S'il n'est pas spécifié, c'est l'utilisateur root qui est utilisé.
	 
	NOTE
	L'utilisateur doit déjà exister.
	• **run_as_group** (par défaut : **podman_run_as_group**) - un groupe par pod. S'il n'est pas spécifié, c'est la racine qui est utilisée.

Variable	Description
	NOTE Le groupe doit déjà exister. • **systemd_unit_scope** (par défaut : **podman_systemd_unit_scope**) - champ d'application à utiliser pour l'unité **systemd**. S'il n'est pas spécifié, **system** est utilisé pour les conteneurs racine et **user** pour les conteneurs utilisateur. • **kube_file_src** - nom d'un fichier YAML Kubernetes sur le nœud contrôleur qui sera copié sur **kube_file** sur le nœud géré  NOTE Ne spécifiez pas la variable **kube_file_src** si vous spécifiez la variable **kube_file_content**. La variable **kube_file_content** est prioritaire sur la variable **kube_file_src**. • **kube_file_content** - chaîne de caractères au format YAML de Kubernetes ou un dict au format YAML de Kubernetes. Il spécifie le contenu de **kube_file** sur le nœud géré.  NOTE Ne spécifiez pas la variable **kube_file_content** si vous spécifiez la variable **kube_file_src**. La variable **kube_file_content** est prioritaire sur la variable **kube_file_src**. • **kube_file** - un nom de fichier sur le nœud géré qui contient la spécification Kubernetes du conteneur ou du pod. Vous n'avez généralement pas besoin de spécifier la variable **kube_file**, sauf si vous devez copier le fichier **kube_file** sur le nœud géré en dehors du rôle. Si vous spécifiez **kube_file_src** ou **kube_file_content**, vous n'avez pas besoin de spécifier cette variable.  NOTE Il est fortement recommandé d'omettre **kube_file** et de spécifier à la place **kube_file_src** ou **kube_file_content** et de laisser le rôle gérer le chemin et le nom du fichier.

Variable	Description
	○ Le nom de base du fichier sera la valeur metadata.name du yaml K8s, avec un suffixe **.yml**. ○ Le répertoire est **/etc/containers/ansible-kubernetes.d** pour les services du système. ○ Le répertoire est **$HOME/.config/containers/ansible-kubernetes.d** pour les services aux utilisateurs. ○ Il sera copié dans le fichier **/etc/containers/ansible-kubernetes.d/*<application_name>*.yml** sur le nœud géré.
podman_create_host_directories	Si true, le rôle garantit les répertoires d'hôtes spécifiés dans les montages d'hôtes dans **volumes.hostPath** spécifications dans le YAML Kubernetes donné dans **podman_kube_specs**. La valeur par défaut est false.  NOTE Les répertoires doivent être spécifiés en tant que chemins absolus (pour les conteneurs racine), ou chemins relatifs au répertoire d'origine (pour les conteneurs non racine), afin que le rôle puisse les gérer. Tout autre chemin est ignoré. Le rôle applique sa propriété ou ses autorisations par défaut aux répertoires. Si vous devez définir la propriété ou les autorisations, consultez **podman_host_directories**.
podman_host_directories	Il s'agit d'un dict. Si vous utilisez **podman_create_host_directories** pour indiquer au rôle de créer des répertoires hôtes pour les montages de volumes, et que vous devez spécifier les permissions ou la propriété qui s'appliquent à ces répertoires hôtes créés, utilisez **podman_host_directories**. Chaque clé est le chemin absolu du répertoire hôte à gérer. La valeur est au format des paramètres du module de fichier. Si vous ne spécifiez pas de valeur, le rôle utilisera les valeurs par défaut intégrées. Si vous souhaitez spécifier une valeur à utiliser pour tous les répertoires hôtes, utilisez la clé spéciale **DEFAULT**.

Variable	Description
podman_firewall	Il s'agit d'une liste de dict. Spécifie les ports que vous voulez que le rôle gère dans le pare-feu. Le format utilisé est le même que celui utilisé par le rôle système RHEL du pare-feu.
podman_selinux_ports	Il s'agit d'une liste de dict. Spécifie les ports pour lesquels vous souhaitez que le rôle gère la politique SELinux pour les ports utilisés par le rôle. Le format utilisé est le même que celui du rôle système RHEL selinux.
podman_run_as_user	Spécifie le nom de l'utilisateur à utiliser pour tous les conteneurs sans racine. Vous pouvez également spécifier un nom d'utilisateur par conteneur avec **run_as_user** dans **podman_kube_specs**.  NOTE L'utilisateur doit déjà exister.
podman_run_as_group	Spécifie le nom du groupe à utiliser pour tous les conteneurs sans racine. Vous pouvez également spécifier un nom de groupe par conteneur avec **run_as_group** dans **podman_kube_specs**.  NOTE Le groupe doit déjà exister.
podman_systemd_unit_scope	Définit la portée de **systemd** à utiliser par défaut pour toutes les unités **systemd**. Vous pouvez également spécifier la portée par conteneur avec **systemd_unit_scope** dans **podman_kube_specs**. Par défaut, les conteneurs sans racine utilisent **user** et les conteneurs racine utilisent **system**.
podman_containers_conf	Définit les paramètres de **containers.conf(5)** sous forme de dict. Les paramètres sont fournis dans un fichier drop-in dans le répertoire **containers.conf.d**. Si l'utilisateur est root (voir **podman_run_as_user**), les paramètres de **system** sont gérés. Sinon, ce sont les paramètres de **user** qui sont gérés. Voir la page de manuel **containers.conf** pour connaître l'emplacement des répertoires.

Variable	Description
podman_registries_conf	Définit les paramètres de **containers-registries.conf(5)** sous forme de dict. Les paramètres sont fournis dans un fichier drop-in dans le répertoire **registries.conf.d**. Si l'utilisateur est root (voir **podman_run_as_user**), les paramètres de **system** sont gérés. Sinon, ce sont les paramètres de **user** qui sont gérés. Voir la page de manuel **registries.conf** pour connaître l'emplacement des répertoires.
podman_storage_conf	Définit les paramètres de **containers-storage.conf(5)** sous forme de dict. Si l'utilisateur est root (voir **podman_run_as_user**), les paramètres de **system** sont gérés. Sinon, ce sont les paramètres de **user** qui sont gérés. Voir la page de manuel **storage.conf** pour connaître l'emplacement des répertoires.
podman_policy_json	Définit les paramètres de **containers-policy.conf(5)** sous forme de dict. Si l'utilisateur est root (voir **podman_run_as_user**), les paramètres de **system** sont gérés. Sinon, ce sont les paramètres de **user** qui sont gérés. Voir la page de manuel **policy.json** pour connaître l'emplacement des répertoires.

Ressources supplémentaires

- Installation des rôles système RHEL
- Pour plus de détails sur les paramètres utilisés dans **podman** et des informations supplémentaires sur le rôle de système RHEL **podman**, voir le fichier **/usr/share/ansible/roles/rhel-system-roles.podman/README.md**.

26.3. RESSOURCES SUPPLÉMENTAIRES

- Pour plus de détails sur les paramètres utilisés dans **podman** et des informations supplémentaires sur le rôle de système RHEL **podman**, voir le fichier **/usr/share/ansible/roles/rhel-system-roles.podman/README.md**.
- Pour plus de détails sur la commande **ansible-playbook**, voir la page de manuel **ansible-playbook(1)**.

CHAPITRE 27. INTÉGRATION DIRECTE DES SYSTÈMES RHEL À AD À L'AIDE DES RÔLES SYSTÈME RHEL

Avec le rôle de système **ad_integration**, vous pouvez automatiser une intégration directe d'un système RHEL avec Active Directory (AD) à l'aide de Red Hat Ansible Automation Platform.

Ce chapitre couvre les sujets suivants :

- Le rôle du système **ad_integration**
- Variables pour le rôle de système RHEL ad_integration
- Connexion directe d'un système RHEL à AD à l'aide du rôle de système **ad_integration**

27.1. LE RÔLE DU SYSTÈME AD_INTEGRATION

Le rôle de système **ad_integration** permet de connecter directement un système RHEL à Active Directory (AD).

Le rôle utilise les éléments suivants :

- SSSD pour interagir avec la source centrale d'identité et d'authentification
- **realmd** pour détecter les domaines AD disponibles et configurer les services système RHEL sous-jacents, dans ce cas SSSD, pour se connecter au domaine AD sélectionné

NOTE

Le rôle **ad_integration** est destiné aux déploiements utilisant l'intégration directe d'AD sans environnement de gestion des identités (IdM). Pour les environnements IdM, utilisez les rôles **ansible-freeipa**.

Ressources supplémentaires

- Connexion directe des systèmes RHEL à AD à l'aide de SSSD .

27.2. VARIABLES POUR LE RÔLE DE SYSTÈME AD_INTEGRATION RHEL

Le rôle de système **ad_integration** RHEL utilise les paramètres suivants :

Variable de rôle	Description
ad_integration_realm	Active Directory realm, ou nom de domaine à rejoindre.
ad_integration_password	Le mot de passe de l'utilisateur utilisé pour s'authentifier lors de la connexion de la machine à la zone. N'utilisez pas de texte brut. Au lieu de cela, utilisez Ansible Vault pour crypter la valeur.

Variable de rôle	Description
ad_integration_manage_crypto_policies	Si **true**, le rôle **ad_integration** utilisera **fedora.linux_system_roles.crypto_policies** si nécessaire. Par défaut : **false**
ad_integration_allow_rc4_crypto	Si **true**, le rôle **ad_integration** définira la politique de cryptage pour autoriser le cryptage RC4. Le fait d'indiquer cette variable permet de remplacer automatiquement **ad_integration_manage_crypto_policies** par **true**. Par défaut : **false**
ad_integration_timesync_source	Nom d'hôte ou adresse IP de la source de temps avec laquelle synchroniser l'horloge du système. Le fait d'indiquer cette variable permet de remplacer automatiquement **ad_integration_manage_timesync** par **true**.

Ressources supplémentaires

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.ad_integration/README.md**.

27.3. CONNEXION DIRECTE D'UN SYSTÈME RHEL À AD À L'AIDE DU RÔLE DE SYSTÈME AD_INTEGRATION

Vous pouvez utiliser le rôle de système **ad_integration** pour configurer une intégration directe entre un système RHEL et un domaine AD en exécutant un playbook Ansible.

NOTE

À partir de RHEL8, RHEL ne prend plus en charge le chiffrement RC4 par défaut. S'il n'est pas possible d'activer AES dans le domaine AD, vous devez activer la stratégie cryptographique **AD-SUPPORT** et autoriser le chiffrement RC4 dans le manuel de jeu.

IMPORTANT

L'heure entre le serveur RHEL et AD doit être synchronisée. Vous pouvez vous en assurer en utilisant le rôle de système **timesync** dans le playbook.

Dans cet exemple, le système RHEL rejoint le domaine AD **domain.example.com**, en utilisant l'utilisateur AD **Administrator** et le mot de passe de cet utilisateur stocké dans le coffre-fort Ansible. Le playbook définit également la stratégie cryptographique **AD-SUPPORT** et autorise le chiffrement RC4. Pour assurer la synchronisation temporelle entre le système RHEL et AD, le livre de jeu définit le serveur **adserver.domain.example.com** comme source **timesync**.

Conditions préalables

- Accès et autorisations à une ou plusieurs *managed nodes*.
- Accès et autorisations à *control node*.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.
  - Un fichier d'inventaire qui répertorie les nœuds gérés.
- Les ports suivants des contrôleurs de domaine AD sont ouverts et accessibles depuis le serveur RHEL :

Tableau 27.1. Ports requis pour l'intégration directe de systèmes Linux dans AD à l'aide du rôle de système ad_integration

Port source	Port de destination	Protocol	Service
1024:65535	53	UDP et TCP	DNS
1024:65535	389	UDP et TCP	LDAP
1024:65535	636	TCP	LDAPS
1024:65535	88	UDP et TCP	Kerberos
1024:65535	464	UDP et TCP	Changement/réinitialisation du mot de passe Kerberos (**kadmin**)
1024:65535	3268	TCP	Catalogue global LDAP
1024:65535	3269	TCP	Catalogue global LDAP SSL/TLS
1024:65535	123	UDP	NTP/Chronie (optionnel)
1024:65535	323	UDP	NTP/Chronie (optionnel)

Procédure

1. Créez un nouveau fichier **ad_integration.yml** avec le contenu suivant :

```
---
- hosts: all
  vars:
    ad_integration_realm: "domain.example.com"
    ad_integration_password: !vault | vault encrypted password
    ad_integration_manage_crypto_policies: true
```

```
    ad_integration_allow_rc4_crypto: true
    ad_integration_timesync_source: "adserver.domain.example.com"
  roles:
    - linux-system-roles.ad_integration
---
```

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check ad_integration.yml -i inventory_file
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/ad_integration.yml
```

Vérification

- Afficher les détails d'un utilisateur AD, tel que l'utilisateur **administrator**:

```
getent passwd administrator@ad.example.com
administrator@ad.example.com:*:1450400500:1450400513:Administrator:/home/administrator
@ad.example.com:/bin/bash
```

27.4. RESSOURCES SUPPLÉMENTAIRES

- Le fichier **/usr/share/ansible/roles/rhel-system-roles.ad_integration/README.md**.
- **man ansible-playbook(1)**