Red Hat Enterprise Linux 9

RHEL システムロールを使用した管理タスクおよび設定タスク

Red Hat Ansible Automation Platform Playbook を使用した RHEL システムロールの適用およびシステム管理タスクの実行

Last Updated: 2022-08-15

Red Hat Enterprise Linux 9 RHEL システムロールを使用した管理タスクおよび設定タスク

Red Hat Ansible Automation Platform Playbook を使用した RHEL システムロールの適用およびシステム管理タスクの実行

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書は、Red Hat Enterprise Linux 9 で Ansible を使用してシステムロールを設定する方法を説明します。RHEL システムロールは、Red Hat Enterprise Linux を管理および設定する安定した設定インターフェースを提供する Ansible ロール、モジュール、および Playbook のコレクションです。RHEL 9 の複数のメジャーリリースバージョンと前方互換性があるように設計されています。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバックの提供

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
    1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
    2. マウスカーソルで、コメントを追加する部分を強調表示します。
    3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
    4. 表示される手順に従ってください。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
    1. Bugzilla の Web サイトに移動します。
    2. Component で **Documentation** を選択します。
    3. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも記入してください。
    4. **Submit Bug** をクリックします。

第1章 RHEL システムロールの使用

このセクションでは、RHEL システムロールについて説明します。また、Ansible Playbook を使用して特定のロールを適用し、さまざまなシステム管理タスクを実行する方法を説明します。

1.1. RHEL システムロールの概要

RHEL システムロールは、Ansible ロールおよびモジュールのコレクションです。RHEL システムロールは、複数の RHEL システムをリモートで管理するための設定インターフェースを提供します。このインターフェースは、RHEL の複数のバージョンにわたるシステム設定の管理と、新しいメジャーリリースの導入を可能にします。

Red Hat Enterprise Linux 9 のインターフェースは、現在、以下のロールから構成されます。

- 証明書の発行および更新
- カーネル設定
- メトリクス
- Network Bound Disk Encryption クライアントおよび Network Bound Disk Encryption サーバー
- ネットワーキング
- postfix
- SSH クライアント
- SSH サーバー
- システム全体の暗号化ポリシー
- ターミナルセッションの録画

これらのロールはすべて、**AppStream** リポジトリーで利用可能な **rhel-system-roles** パッケージで提供されます。

関連情報

- Red Hat Enterprise Linux (RHEL) System Roles
- **/usr/share/doc/rhel-system-roles/** ディレクトリーのドキュメント [1]

1.2. RHEL システムロールの用語

このドキュメントでは、以下の用語を確認できます。

Ansible Playbook

Playbook は、Ansible の設定、デプロイメント、およびオーケストレーションの言語です。リモートシステムを強制するポリシーや、一般的な IT プロセスで一連の手順を説明することができます。

コントロールノード

Ansible がインストールされているマシン。コマンドおよび Playbook を実行でき、すべてのコントロールノードから /usr/bin/ansible または /usr/bin/ansible-playbook を起動します。Python がインストールされているすべてのコンピューターをコントロールノードとして使用できます。ラップ

トップ、共有デスクトップ、およびサーバーですべての Ansible を実行できます。ただし、Windows マシンをコントロールノードとして使用することはできません。複数のコントロールノードを使用できます。

インベントリー

管理対象ノードの一覧。インベントリーファイルは「ホストファイル」とも呼ばれます。インベントリーでは、各管理対象ノードに対して IP アドレスなどの情報を指定できます。また、インベントリーは管理ノードを編成し、簡単にスケーリングできるようにグループの作成およびネスト化が可能です。インベントリーについての詳細は、「インベントリーの操作」セクションを参照してください。

管理ノード

Ansible で管理するネットワークデバイス、サーバー、またはその両方。管理対象ノードは、「ホスト」と呼ばれることもあります。Ansible が管理ノードにはインストールされません。

1.3. ロールの適用

以下の手順では、特定のロールを適用する方法を説明します。

前提条件

- **rhel-system-roles** パッケージが、コントロールノードとして使用するシステムにインストールされていることを確認します。

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

1. Ansible Core パッケージをインストールします。

```
# dnf install ansible-core
```

Ansible Core パッケージは、**ansible-playbook** CLI、Ansible Vault 機能、および RHEL Ansible コンテンツに必要な基本的なモジュールとフィルターを提供します。

- Ansible インベントリーを作成できることを確認します。
インベントリーは、ホスト、ホストグループ、および Ansible Playbook で使用されるいくつかの設定パラメーターを表します。

  Playbook は通常人が判読でき、**ini**、**yaml**、**json**、およびその他のファイル形式で定義されます。

- Ansible Playbook を作成できることを確認します。
Playbook は、Ansible の設定、デプロイメント、およびオーケストレーションの言語を表します。Playbook を使用すると、リモートマシンの設定を宣言して管理したり、複数のリモートマシンをデプロイしたり、手動で順番を付けたプロセスの手順をまとめたりできます。

  Playbook は、1 つ以上の **play** の一覧です。すべての **play** には、Ansible の変数、タスク、またはロールが含まれます。

  Playbook は人が判読でき、**yaml** 形式で定義されます。

手順

1. 管理するホストおよびグループを含む必要な Ansible インベントリーを作成します。以下の例は、**webservers** と呼ばれるホストのグループの **inventory.ini** というファイルを使用します。

```
[webservers]
host1
host2
host3
```

2. 必要なロールを含む Ansible Playbook を作成します。以下の例では、Playbook の **roles:** オプションを使用してロールを使用する方法を示しています。
以下の例は、特定の **play** の **roles:** オプションを使用してロールを使用する方法を示しています。

```
---
- hosts: webservers
  roles:

    - rhel-system-roles.network
    - rhel-system-roles.postfix
```

注記

すべてのロールには README ファイルが含まれます。このファイルには、ロールや、サポートされるパラメーター値の使用方法が記載されています。ロールのドキュメントディレクトリーで、特定ロール用の Playbook のサンプルを見つけることもできます。このようなドキュメンテーションディレクトリーは、**rhel-system-roles** パッケージでデフォルトで提供され、以下の場所に置かれます。

```
/usr/share/doc/rhel-system-roles/SUBSYSTEM/
```

SUBSYSTEM を、必要なロールの名前 (**postfix**、**metrics**、**network**、**tlog**、または **ssh** など) に置き換えます。

3. 特定のホストで Playbook を実行するには、以下のいずれかを実行する必要があります。

- **hosts: host1[,host2,...]** または **hosts: all** Playbook を編集し、次のコマンドを実行します。

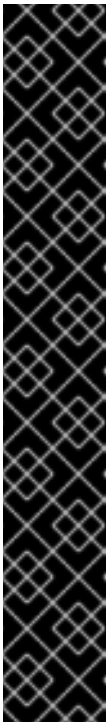
```
# ansible-playbook name.of.the.playbook
```

- インベントリーを編集して、使用するホストがグループで定義されていることを確認し、以下のコマンドを実行します。

```
# ansible-playbook -i name.of.the.inventory name.of.the.playbook
```

- **ansible-playbook** コマンドの実行時にすべてのホストを指定します。

```
# ansible-playbook -i host1,host2,... name.of.the.playbook
```



重要

-i フラグは、利用可能なすべてのホストのインベントリーを指定することに注意してください。ターゲットホストが複数あるが、Playbook を実行するホストを選択する場合は、Playbook に変数を追加して、ホストを選択できるようにすることができます。以下に例を示します。

```
Ansible Playbook | example-playbook.yml:


- hosts: "{{ target_host }}"
  roles:
    - rhel-system-roles.network
    - rhel-system-roles.postfix
```

Playbook 実行コマンド:

```
# ansible-playbook -i host1,..hostn -e target_host=host5 example-
playbook.yml
```

関連情報

- Ansible Playbook
- Using roles in Ansible playbook
- Examples of Ansible playbooks
- How to create and work with inventory?
- ansible-playbook tool

1.4. 関連情報

- Red Hat Enterprise Linux (RHEL) System Roles
- RHEL システムロールを使用したローカルストレージの管理
- 「複数のシステムへの同じ SELinux 設定のデプロイメント」

[1] 本書は **rhel-system-roles** パッケージで自動的にインストールされます。

第2章 RHEL システムロールのインストール

システムロールの使用を開始する前に、システムにインストールする必要があります。

2.1. システムへの RHEL システムロールのインストール

RHEL システムロールを使用するには、システムに必要なパッケージをインストールします。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- コントロールノードとして使用するシステムに Ansible パッケージがインストールされている。

手順

1. **rhel-system-roles** パッケージを、コントロールノードとして使用するシステムにインストールします。

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

2. Ansible Core パッケージをインストールします。

```
# dnf install ansible-core
```

Ansible Core パッケージは、**ansible-playbook** CLI、Ansible Vault 機能、および RHEL Ansible コンテンツに必要な基本的なモジュールとフィルターを提供します。

結果として、Ansible の Playbook を作成できます。

関連情報

- The Red Hat Enterprise Linux (RHEL) システムロール
- **ansible-playbook** の man ページ

第3章 RHEL システムロールの自動化を有効にするためのパッケージの更新

RHEL 9.0 GA リリース以降、Ansible Engine はサポートされなくなりました。代わりに、今回以降の RHEL リリースには Ansible Core が含まれます。

RHEL 9.0 GA の Ansible Core を使用して、Red Hat 製品によって作成または生成された Ansible 自動化コンテンツを有効化できます。

Ansible Core には、**ansible-playbook** や **ansible** コマンドなどの Ansible コマンドラインツールと、ビルトイン Ansible プラグイン のセットが含まれています。

3.1. ANSIBLE ENGINE と ANSIBLE CORE の違い

RHEL 8.5 以前のバージョンでは、Ansible Engine 2.9 を含む別の Ansible リポジトリーにアクセスして、Red Hat システムに対して Ansible に基づき自動化を有効にしていました。

Ansible サブスクリプションなしで Ansible Engine を使用する場合、サポート範囲は RHEL システムロール、Insights 修復 Playbook、OpenSCAP Ansible 修復 Playbook などの Red Hat 製品によって作成または生成された AnsiblePlaybook の実行に限定されます。

RHEL 8.6 以降のバージョンでは、Ansible Core が Ansible Engine に置き換わります。**ansible-core** パッケージは RHEL 9 AppStream リポジトリーに含まれており、Red Hat が提供する自動化コンテンツを有効にします。RHEL での Ansible Core のサポート範囲は、以前の RHEL バージョンと同じです。

- サポートは、RHEL システムロールなど、Red Hat 製品に含まれているか Red Hat 製品によって生成された Ansible Playbook、ロール、モジュール、または Insights によって生成された修復 Playbook に限定されます。
- Ansible Core を使用すると、RHEL システムロールや Insights 修復 Playbook など、サポートされている RHEL Ansible コンテンツのすべての機能を利用できます。

Ansible Engine リポジトリーは RHEL 8.6 でも引き続き使用できます。ただし、セキュリティーまたはバグ修正の更新は受信せず、RHEL 8.6 以降に含まれる Ansible 自動化コンテンツと互換性がない可能性があります。

基盤となるプラットフォームと Core で維持されるモジュールの追加サポートには、Ansible Automation Platform のサブスクリプションが必要です。

関連情報

- Scope of support for Ansible Core in RHEL

3.2. ANSIBLE ENGINE から ANSIBLE CORE への移行

前提条件

- 1 つ以上の **管理対象ノード** (RHEL システムロールで設定するシステム) へのアクセスおよびパーミッション。
  - 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. Ansible Engine をインストールします。

```
# dnf remove ansible
```

2. **ansible-2-for-rhel-8-x86_64-rpms** リポジトリーを無効にします。

```
# subscription-manager repos --disable ansible-2-for-rhel-8-x86_64-rpms
```

3. RHEL 8 AppStream リポジトリーで利用可能な Ansible Core をインストールします。

```
# dnf install ansible-core
```

検証

- **ansible-core** パッケージがシステムに存在することを確認します。

```
# dnf info ansible-core
```

ansible-core パッケージが実際にシステムに存在する場合、コマンド出力には、パッケージ名、バージョン、リリース、サイズなどに関する情報が表示されます。

```
Available Packages
Name         : ansible-core
Version      : 2.12.2
Release      : 1.fc34
Architecture : noarch
Size         : 2.4 M
Source       : ansible-core-2.12.2-1.fc34.src.rpm
Repository   : updates
Summary      : A radically simple IT automation system
URL          : http://ansible.com
```

関連情報

- Using Ansible in RHEL 9 .

第4章 コレクションのインストールおよび使用

4.1. ANSIBLE コレクションの概要

Ansible コレクションは、新たな方法で自動化を配布、メンテナンス、および使用します。Playbook、ロール、モジュール、プラグインなど、複数のタイプの Ansible コンテンツを組み合わせることで、柔軟性とスケーラビリティーが向上します。

Ansible Collections は、従来の RHEL システムロール形式に対するオプションです。Ansible Collection 形式で RHEL システムロールを使用するのは、従来の RHEL システムロール形式での使用とほぼ同じです。相違点は、Ansible Collections は **完全修飾コレクション名** (FQCN) という概念を使用する点です。このコレクション名は、**namespace** と **コレクション名** で構成されます。使用する **namespace** は **redhat** で、**コレクション名** は **rhel_system_roles** です。したがって、Kernel Settings の従来の RHEL システムロール形式は **rhel-system-roles.kernel_settings** として表示されますが、Kernel Settings ロールの **完全修飾コレクション名** というコレクションを使用すると **redhat.rhel_system_roles.kernel_settings** として表示されます。

namespace と **コレクション名** を組み合わせると、確実にオブジェクトが一意になります。また、オブジェクトが競合せずに Ansible Collections および namespace 間で共有されます。

関連情報

- Automation Hub にアクセスして Red Hat 認定コレクションを使用するには、Ansible Automation Platform (AAP サブスクリプション) が必要です。

4.2. コレクション構造

コレクションは、Ansible コンテンツのパッケージ形式です。データ構造は以下のようになります。

- docs/: 例も含めてコレクションについてまとめたローカルドキュメント。(ロールがドキュメントを提供する場合)
- galaxy.yml: Ansible Collection パッケージに含まれる MANIFEST.json のソースデータ
- Playbook/: Playbook はこちらで利用できます。
  - tasks/: include_tasks/import_tasks の使用状況に関する「task list files」を保管します。
- plugins/: Ansible プラグインおよびモジュールはすべてこちらの各サブディレクトリーから入手できます。
  - modules/: Ansible モジュール
  - modules_utils/: モジュール開発用の共通コード
  - lookup/: プラグインの検索
  - filter/: Jinja2 filter プラグイン
  - connection/: 接続プラグインはデフォルトを使用していない場合に必要です。
- roles/: Ansible ロール用ディレクトリー
- tests/: コレクションの内容のテスト

4.3. CLI を使用したコレクションのインストール

コレクションは、Playbook、ロール、モジュール、およびプラグインなど、Ansible コンテンツのディストリビューション形式です。

コレクションは、Ansible Galaxy、ブラウザーまたはコマンドラインを使用してインストールできます。

前提条件

- 1 つ以上の **管理対象ノード** へのアクセスとアクセス権。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。

手順

- RPM パッケージからコレクションをインストールします。

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

インストールが完了すると、ロールは **redhat.rhel_system_roles.<role_name>** として利用できます。また、各ロールのドキュメントは **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md** で確認できます。

検証手順

コレクションが正常にインストールされていることを確認するには、ローカルホストに kernel_settings を適用してください。

1. **tests_default.yml** のいずれかを作業ディレクトリーにコピーします。

```
$ cp
/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
ests_default.yml .
```

2. ファイルを編集し、"hosts: all" を "hosts: localhost" に置き換え、Playbook がローカルシステムでのみ実行されるようにします。
3. ansible-playbook をチェックモードで実行します。このモードでは、システムの設定は変更されません。

```
$ ansible-playbook --check tests_default.yml
```

このコマンドは、**failed=0** の値を返します。

関連情報

- **ansible-playbook** の man ページ

4.4. AUTOMATION HUB からのコレクションのインストール

Automation Hub を使用している場合は、Automation Hub でホストされている RHEL システムロールコレクションをインストールできます。

前提条件

- 1 つ以上の **管理対象ノード** へのアクセスとアクセス権。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. **ansible.cfg** 設定ファイルでコンテンツのデフォルトソースとして Red Hat Automation Hub を定義します。コンテンツについては、「プライマリーソースとしての Red Hat Automation Hub の設定」を参照してください。
2. Automation Hub から **redhat.rhel_system_roles** コレクションをインストールします。

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

   インストールが完了すると、ロールは **redhat.rhel_system_roles.<role_name>** として利用できます。また、各ロールのドキュメントは **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md** で確認できます。

検証手順

コレクションが正常にインストールされたことを確認するため、ローカルホストに **kernel_settings** を適用できます。

1. **tests_default.yml** のいずれかを作業ディレクトリーにコピーします。

   ```
   $ cp
   /usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
   ests_default.yml .
   ```

2. ファイルを編集し、"hosts: all" を "hosts: localhost" に置き換え、Playbook がローカルシステムでのみ実行されるようにします。
3. ansible-playbook をチェックモードで実行します。このモードでは、システムの設定は変更されません。

   ```
   $ ansible-playbook --check tests_default.yml
   ```

   このコマンドから **failed=0** の値が返されたことが分かります。

関連情報

- **ansible-playbook** の man ページ

4.5. コレクションを使用した TERMINAL SESSION RECORDING RHEL システムロールのデプロイ

以下の例では、コレクションを使用して Playbook を準備および適用し、一連の個別のマシンにロギングソリューションをデプロイしています。

前提条件

- Galaxy コレクションがインストールされている。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

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

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

詳細は以下のようになります。

- **tlog_scope_sssd**:
  - **some** は、**all** または **none** ではなく、特定のユーザーおよびグループのみを録画することを指定します。
- **tlog_users_sssd**:
  - **recordeduser** は、セッションを録画するユーザーを指定します。ただし、ユーザーは追加されない点に留意してください。ユーザーを独自に設定する必要があります。

2. オプション: Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

これにより、Playbook は指定したシステムに Terminal Session Recording ロールをインストールします。また、定義したユーザーおよびグループで使用できる SSSD 設定ドロップファイルを作成します。SSSD は、これらのユーザーおよびグループを解析して読み取り、シェルユーザーとして **tlog** セッショ

ンをオーバーレイします。さらに、**cockpit** パッケージがシステムにインストールされている場合、Playbook は **cockpit-session-recording** パッケージもインストールします。これは、Web コンソールインターフェースで録画を表示および再生できるようにする **Cockpit** モジュールです。

検証手順

1. **/etc/rsyslog.conf** ファイルの構文をテストします。

```
# 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. システムがログにメッセージを送信していることを確認します。

システムで SSSD 設定ドロップファイルが作成されることを確認するには、以下の手順を実行します。

1. SSSD 設定ドロップファイルが作成されるフォルダーに移動します。

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

2. ファイルの内容を確認します。

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

Playbook に設定したパラメーターがファイルに含まれていることが確認できます。

第5章 RHEL の ANSIBLE IPMI モジュール

5.1. RHEL_MGMT コレクション

Intelligent Platform Management Interface (IPMI) は、ベースボード管理コントローラー (BMC) デバイスと通信するための一連の標準プロトコルの仕様です。**IPMI** モジュールを使用すると、ハードウェア管理の自動化を有効にしてサポートできます。**IPMI** モジュールは次の場所で使用できます。

- **rhel_mgmt** コレクション。パッケージ名は **ansible-collection-redhat-rhel_mgmt** です。
- 新しい **ansible-collection-redhat-rhel_mgmt** パッケージの一部である RHEL 8 AppStream。

次の IPMI モジュールが rhel_mgmt コレクションで使用可能です。

- **ipmi_boot**:起動デバイスの順序管理
- **ipmi_power**:マシンの電源管理

IPMI モジュールに使用される必須パラメーターは次のとおりです。

- **ipmi_boot** パラメーター:

モジュール名	説明
name	BMC のホスト名または IP アドレス。
password	BMC に接続するためのパスワード
bootdev	次回起動時に使用するデバイス * network * floppy * hd * safe * optical * setup * default
User	BMC に接続するためのユーザー名

- **ipmi_power** パラメーター:

モジュール名	説明
name	BMC ホスト名または IP アドレス

モジュール名	説明
password	BMC に接続するためのパスワード
user	BMC に接続するためのユーザー名
状態	マシンが目的のステータスにあるかどうかを確認します * on * off * shutdown * reset * boot

5.2. CLI を使用した RHEL MGMT コレクションのインストール

コマンドラインを使用して **rhel_mgmt** コレクションをインストールできます。

前提条件

- **ansible-core** パッケージがインストールされている。

手順

- RPM パッケージからコレクションをインストールします。

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

インストールが完了すると、IPMI モジュールは **redhat.rhel_mgmt** Ansible コレクションで使用できるようになります。

関連情報

- **ansible-playbook** の man ページ

5.3. IPMI_BOOT モジュールの使用例

次の例は、Playbook で **ipmi_boot** モジュールを使用して、次回の起動用に起動デバイスを設定する方法を示しています。わかりやすくするために、ここに示す例では Ansible コントロールホストおよび管理対象ホストと同じホストを使用しているため、Playbook が実行されるのと同じホストでモジュールを実行します。

前提条件

- rhel_mgmt コレクションがインストールされている。

- **python3-pyghmi** パッケージの **pyghmi** ライブラリーが、次のいずれかの場所にインストールされている。
  - Playbook を実行するホスト。
  - 管理対象ホスト。localhost を管理対象ホストとして使用する場合は、代わりに、Playbook を実行するホストに **python3-pyghmi** パッケージをインストールします。
- 制御する IPMI BMC は、Playbook を実行するホスト、または管理対象ホスト (localhost を管理対象ホストとして使用していない場合) からネットワーク経由でアクセスできます。モジュールが IPMI プロトコルを使用してネットワーク経由で BMC に接続するため、通常、モジュールによって BMC が設定されているホストはモジュールが実行されているホスト (Ansible 管理対象ホスト) とは異なることに注意してください。
- 適切なレベルのアクセスで BMC にアクセスするためのクレデンシャルがあります。

手順

1. 以下のコンテンツを含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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. localhost に対して Playbook を実行します。

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

その結果、出力は値 success を返します。

5.4. IPMI_POWER モジュールの使用例

この例は、Playbook で **ipmi_boot** モジュールを使用して、システムがオンになっているかどうかを確認する方法を示しています。わかりやすくするために、ここに示す例では Ansible コントロールホストおよび管理対象ホストと同じホストを使用しているため、Playbook が実行されるのと同じホストでモジュールを実行します。

前提条件

- rhel_mgmt コレクションがインストールされている。
- **python3-pyghmi** パッケージの **pyghmi** ライブラリーが、次のいずれかの場所にインストールされている。
  - Playbook を実行するホスト。
  - 管理対象ホスト。localhost を管理対象ホストとして使用する場合は、代わりに、Playbook を実行するホストに **python3-pyghmi** パッケージをインストールします。

- 制御する IPMI BMC は、Playbook を実行するホスト、または管理対象ホスト (localhost を管理対象ホストとして使用していない場合) からネットワーク経由でアクセスできます。モジュールが IPMI プロトコルを使用してネットワーク経由で BMC に接続するため、通常、モジュールによって BMC が設定されているホストはモジュールが実行されているホスト (Ansible 管理対象ホスト) とは異なることに注意してください。
- 適切なレベルのアクセスで BMC にアクセスするためのクレデンシャルがあります。

手順

1. 以下のコンテンツを含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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. Playbook を実行します。

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

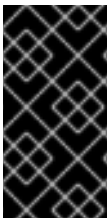
出力は値 true を返します。

第6章 ANSIBLE ロールを使用したカーネルパラメーターの永続的な設定

Kernet　Settings ロールを使用して、複数のクライアントにカーネルパラメーターを一度に設定することができます。この解決策は以下のとおりです。

- 効率的な入力設定を持つ使いやすいインターフェースを提供します。
- すべてのカーネルパラメーターを 1 か所で保持します。

コントロールマシンから Kernel Settings ロールを実行すると、カーネルパラメーターはすぐに管理システムに適用され、再起動後も維持されます。



重要

RHEL チャネルで提供される RHEL システムロールは、デフォルトの App Stream リポジトリーの RPM パッケージとして RHEL のお客様が利用できることに注意してください。RHEL システムロールは、Ansible Automation Hub を介して Ansible サブスクリプションを使用しているお客様のコレクションとしても利用できます。

6.1. カーネル設定ロールの概要

RHEL システムロールは、複数のシステムをリモートで管理する、一貫した構成インターフェースを提供する一連のロールです。

Kernel Settings システムロールを使用して、カーネルの自動設定に RHEL システムロールが導入されました。**rhel-system-roles** パッケージには、このシステムロールと参考ドキュメントも含まれます。

カーネルパラメーターを自動的に 1 つ以上のシステムに適用するには、Playbook で選択したロール変数を 1 つ以上使用して、Kernel Settings ロールを使用します。Playbook は人間が判読でき、YAML 形式で記述される 1 つ以上のプレイの一覧です。

Kernel Settings ロールを使用して、以下を設定できます。

- **kernel_settings_sysctl** ロールを使用したカーネルパラメーター
- **kernel_settings_sysfs** ロールを使用したさまざまなカーネルサブシステム、ハードウェアデバイス、およびデバイスドライバー
- **systemd** サービスマネージャーの CPU アフィニティー
  を、**kernel_settings_systemd_cpu_affinity** ロール変数を使用してフォーク処理します。
- **kernel_settings_transparent_hugepages** および
  kernel_settings_transparent_hugepages_defrag のロール変数を使用したカーネルメモリーサブシステムの Transparent Huge Page

関連情報

- **/usr/share/doc/rhel-system-roles/kernel_settings/** ディレクトリーの **README.md ファイル**および **README.html** ファイル
- Playbook の使用
- インベントリーの構築方法

6.2. KERNEL SETTINGS ロールを使用した選択したカーネルパラメーターの適用

以下の手順に従って、Ansible Playbook を準備および適用し、複数の管理システムで永続化の影響でカーネルパラメーターをリモートに設定します。

前提条件

- **root** 権限がある。
- RHEL サブスクリプションの資格を取得して、 **ansible-core**および**rhel-system-roles**パッケージをコントロールマシンにインストールしている。
- 管理対象ホストのインベントリーが制御マシンに存在し、Ansible から接続できる。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、 **ansible**、**ansible-playbook**などのコマンドラインユーティリティー、**docker**や**podman**などのコネクター、プラグインとモジュールすべてが含まれています。Ansible Engine を入手してインストールする方法については、Red Hat Ansible Engine をダウンロードしてインストールする方法を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core**RPM として提供)が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、および組み込みの Ansible プラグインセットが少し含まれています。App Stream リポジトリーには、**ansible-core** が含まれていますが、サポートの範囲が限定されています。詳細は、RHEL 9App Stream に含まれている ansible-core パッケージのサポート範囲 を確認してください。

手順

1. 必要に応じて、図の目的で **inventory** ファイルを確認します。

```
#  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
```

   ファイルは **[testingservers]** グループと他のグループを定義します。これにより、特定のシステムセットに対して Ansible をより効果的に実行できます。

2. 構成ファイルを作成して、Ansible 操作のデフォルトと特権昇格を設定します。

   a. 新しい YAML ファイルを作成し、これをテキストエディターで開きます。以下に例を示します

ます。

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

b. 以下の内容をファイルに挿入します。

```
[defaults]
inventory = ./inventory

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

[defaults] セクションは、管理対象ホストのインベントリーファイルへのパスを指定します。**[privilege_escalation]** セクションでは、指定した管理対象ホストのユーザー権限が **root** に移行されることを定義します。これは、カーネルパラメーターを正常に設定するために必要です。Ansible Playbook を実行すると、ユーザーパスワードの入力が求められます。管理対象ホストへの接続後に、**sudo** により **root** に自動的に切り替わります。

3. Kernel Settings ロールを使用する Ansible Playbook を作成します。

a. 新しい YAML ファイルを作成し、これをテキストエディターで開きます。以下に例を示します。

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

このファイルは Playbook を表し、通常は、**inventory** ファイルから選択した特定の管理対象ホストに対して実行される、**プレイ** とも呼ばれるタスクの順序付きリストが含まれます。

b. 以下の内容をファイルに挿入します。

```
---
-
  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
```

name キーは任意です。任意の文字列をラベルとしてプレイに関連付け、プレイの対象を特定します。プレイの **hosts** キーは、プレイを実行するホストを指定します。このキーの値または値は、管理対象ホストの個別名または **inventory** ファイルで定義されているホストのグループとして指定できます。

vars セクションは、設定する必要がある、選択したカーネルパラメーター名および値が含まれる変数の一覧を表します。

roles キーは、**vars** セクションで説明されているパラメーターおよび値を設定するシステムロールを指定します。

注記

必要に応じて、Playbook のカーネルパラメーターとその値を変更することができます。

4. 必要に応じて、プレイ内の構文が正しいことを確認します。

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

playbook: kernel-roles.yml
```

以下の例では、Playbook の検証が成功したことを示しています。

5. Playbook を実行します。

```
# 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
```

Ansible が Playbook を実行する前に、パスワードの入力を求められます。これにより、管理対象ホストのユーザーが **root** に切り替わります。これは、カーネルパラメーターの設定に必要です。

recap セクションは、すべての管理対象ホストのプレイが正常に終了したこと (**failed=0**)、および 4 つのカーネルパラメーターが適用されたこと (**changed=4**) を示しています。

6. 管理対象ホストを再起動して、影響を受けるカーネルパラメーターをチェックし、変更が適用され、再起動後も維持されていることを確認します。

関連情報

- RHEL システムロールの使用
- **/usr/share/doc/rhel-system-roles/kernel_settings/** ディレクトリーの **README.html ファイル**および **README.md** ファイル

- インベントリーの構築
- Ansible の設定
- Playbook の使用
- 変数の使用
- ロール

第7章 RHEL システムロールを使用したネットワーク接続の設定

Networking RHEL システムロールを使用すると、管理者は Ansible を使用してネットワーク関連の設定および管理タスクを自動化できます。

7.1. インターフェース名で RHEL システムロールを使用した静的イーサネット接続の設定

この手順では、Networking の RHEL システムロールを使用し、Ansible Playbook を実行して以下の設定で**enp7s0** インターフェースにイーサネット接続をリモートで追加する方法を説明します。

- 静的 IPv4 アドレス: **/24** サブネットマスクを持つ **192.0.2.1**
- 静的 IPv6 アドレス: **2001:db8:1::1** （**/64** サブネットマスクあり）
- IPv4 デフォルトゲートウェイ: **192.0.2.254**
- IPv6 デフォルトゲートウェイ: **2001:db8:1::fffe**
- IPv4 DNS サーバー: **192.0.2.200**
- IPv6 DNS サーバー: **2001:db8:1::ffbb**
- DNS 検索ドメイン: **example.com**

Ansible コントロールノードで以下の手順を実行します。

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。
- ホストは NetworkManager を使用してネットワークを設定している。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/ethernet-static-IP.yml** ファイルを以下の内容で作成します。

```
---
- name: Configure an Ethernet connection with static IP
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/ethernet-static-IP.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

-u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md**
- **ansible-playbook(1)** の man ページ

7.2. デバイスパスで RHEL システムロールを使用した静的イーサネット接続の設定

この手順では、RHEL システムロール を使用して、Ansible Playbook を実行することにより、特定のデバイスパスに一致するデバイスの静的 IP アドレスでイーサネット接続をリモートで追加する方法を説明します。

デバイスパスは、次のコマンドで識別できます。

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

この手順では、PCI ID **0000:00:0[1-3].0** 式に一致するが、**0000:00:02.0** には一致しないデバイスーに、次の設定を設定します。

- 静的 IPv4 アドレス: **/24** サブネットマスクを持つ **192.0.2.1**
- 静的 IPv6 アドレス: **2001:db8:1::1** （**/64** サブネットマスクあり）
- IPv4 デフォルトゲートウェイ: **192.0.2.254**
- IPv6 デフォルトゲートウェイ: **2001:db8:1::fffe**
- IPv4 DNS サーバー: **192.0.2.200**
- IPv6 DNS サーバー: **2001:db8:1::ffbb**
- DNS 検索ドメイン: **example.com**

Ansible コントロールノードで以下の手順を実行します。

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。
- ホストは NetworkManager を使用してネットワークを設定している。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/ethernet-dynamic-IP.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

この例の **match** パラメーターは、Ansible が PCI ID **0000:00:0[1-3].0** に一致するデバイスに再生を適用するが、**0000:00:02.0** には適用しないことを定義します。使用できる特殊な修飾子およびワイルドカードの詳細は、**/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイルーの **match** パラメーターの説明を参照してください。

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

-u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.3. インターフェース名で RHEL システムロールを使用した動的イーサネット接続の設定

この手順では、RHEL システムロールを使用して、Ansible Playbook を実行して **enp7s0** インターフェースの動的イーサネット接続をリモートに追加する方法を説明します。この設定により、ネットワーク接続は、DHCP サーバーからこの接続の IP 設定を要求するようになりました。Ansible コントロールノードで以下の手順を実行します。

前提条件

- DHCP サーバーをネットワークで使用できる。
- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。
- ホストは NetworkManager を使用してネットワークを設定している。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/ethernet-dynamic-IP.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.4. デバイスパスでの RHEL システムロールを使用した動的イーサネット接続の設定

この手順では、RHEL システムロール を使用して、Ansible Playbook を実行して特定のデバイスパスに

一致するデバイスに、動的イーサネット接続をリモートで追加する方法を説明します。動的 IP 設定を使用すると、ネットワーク接続は、DHCP サーバーからこの接続の IP 設定を要求します。Ansible コントロールノードで以下の手順を実行します。

デバイスパスは、次のコマンドで識別できます。

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

前提条件

- DHCP サーバーをネットワークで使用できる。
- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。
- ホストは NetworkManager を使用してネットワークを設定している。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

   ```
   node.example.com
   ```

2. **~/ethernet-dynamic-IP.yml** Playbook を以下の内容で作成します。

   ```
   ---
   - name: Configure an Ethernet connection with dynamic IP
     hosts: node.example.com
     become: true
     tasks:
     - 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
   ```

   この例の **match** パラメーターは、Ansible が PCI ID **0000:00:0[1-3].0** に一致するデバイスに再生を適用するが、**0000:00:02.0** には適用しないことを定義します。使用できる特殊な修飾子およびワイルドカードの詳細は、**/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイルーの **match** パラメーターの説明を参照してください。

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.5. RHEL システムロールを使用した VLAN タギングの設定

Networking の RHEL システムロールを使用して、VLAN タグ付けを設定できます。この手順では、イーサネット接続と、このイーサネット接続の上に ID **10** の VLANを追加する方法について説明します。子デバイスの VLAN 接続には、IP、デフォルトゲートウェイ、および DNS の設定が含まれます。

環境に応じて、play を適宜調整します。以下に例を示します。

- ボンディングなどの他の接続でポートとして VLAN を使用する場合は、**ip** 属性を省略し、子 設定で IP 設定を行います。
- VLAN でチーム、ブリッジ、またはボンディングデバイスを使用するには、**interface_name** と VLAN で使用するポートの **type** 属性を調整します。

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

   ```
   node.example.com
   ```

2. **~/vlan-ethernet.yml** Playbook を以下の内容で作成します。

   ```
   ---
   ```

```
- name: Configure a VLAN that uses an Ethernet connection
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

VLAN プロファイルの **parent** 属性は、**enp1s0** デバイス上で動作する VLAN を設定します。

3. Playbook を実行します。

- **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

- 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

```
# ansible-playbook -u user_name --ask-become-pass ~/vlan-ethernet.yml
```

--ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

-u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.6. RHEL システムロールを使用したネットワークブリッジの設定

Networking RHEL システムロールを使用して、Linux ブリッジを設定できます。この手順では、2 つのイーサネットデバイスを使用するネットワークブリッジを設定し、IPv4 アドレスおよび IPv6 アドレス、デフォルトゲートウェイ、および DNS 設定を設定する方法を説明します。

注記

Linux ブリッジのポートではなく、ブリッジで IP 設定を設定します。

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。
- サーバーに、2 つ以上の物理ネットワークデバイスまたは仮想ネットワークデバイスがインストールされている。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/bridge-ethernet.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure a network bridge that uses two Ethernet ports
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/bridge-ethernet.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.7. RHEL システムロールを使用したネットワークボンディングの設定

Networking の RHEL システムロールを使用して、ネットワークボンディングを設定できます。この手順では、2 つのイーサネットデバイスを使用するアクティブバックアップモードでボンディングを設定し、IPv4 アドレスおよび IPv6 アドレス、デフォルトゲートウェイ、および DNS 設定を設定する方法を説明します。

注記

Linux ボンディングのポートではなく、ボンディングに IP 設定を設定します。

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。
- サーバーに、2 つ以上の物理ネットワークデバイスまたは仮想ネットワークデバイスがインストールされている。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. ~/**bond-ethernet.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure a network bond that uses two Ethernet ports
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/bond-ethernet.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.8. RHEL システムロールを使用した 802.1X ネットワーク認証による静的イーサネット接続の設定

Networking RHEL システムロールを使用して、802.1X 規格を使用してクライアントを認証するイーサネット接続の作成を自動化できます。この手順では、以下の設定で **enp1s0** インターフェースのイーサネット接続をリモートに追加する方法を説明します。これには、Ansible Playbook を実行します。

- 静的 IPv4 アドレス: **/24** サブネットマスクを持つ **192.0.2.1**
- 静的 IPv6 アドレス: **2001:db8:1::1** （**/64** サブネットマスクあり）
- IPv4 デフォルトゲートウェイ: **192.0.2.254**
- IPv6 デフォルトゲートウェイ: **2001:db8:1::fffe**
- IPv4 DNS サーバー: **192.0.2.200**
- IPv6 DNS サーバー: **2001:db8:1::ffbb**
- DNS 検索ドメイン: **example.com**

- **TLS** Extensible Authentication Protocol (EAP) を使用した 802.1X ネットワーク認証

Ansible コントロールノードで以下の手順を実行します。

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが必要。
- ネットワークは 802.1X ネットワーク認証をサポートしている。
- 管理ノードは NetworkManager を使用している。
- TLS 認証に必要な以下のファイルがコントロールノードにある。
  - クライアントキーは、**/srv/data/client.key** ファイルに保存されます。
  - クライアント証明書は **/srv/data/client.crt** ファイルに保存されます。
  - 認証局 (CA) 証明書は、**/srv/data/ca.crt** ファイルに保存されます。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/enable-802.1x.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure an Ethernet connection with 802.1X authentication
  hosts: node.example.com
  become: true
  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
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/ethernet-static-IP.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.9. システムロールを使用して、既存の接続でデフォルトのゲートウェイを設定

Networking の RHEL システムロールを使用して、デフォルトゲートウェイを設定できます。

重要

Networking の RHEL システムロールを使用する再生を実行すると、設定値が再生で指定されたものと一致しない場合に、システムロールが同じ名前の既存の接続プロファイルを上書きします。したがって、IP 設定がすでに存在している場合でも、再生でネットワーク接続プロファイルの設定全体を指定する必要があります。それ以外の場合は、ロールはこれらの値をデフォルト値にリセットします。

この手順では、すでに存在するかどうかに応じて、以下の設定で **enp1s0** 接続プロファイルを作成または更新します。

- 静的 IPv4 アドレス: **/24** サブネットマスクを持つ **198.51.100.20**
- 静的 IPv6 アドレス: **2001:db8:1::1** （**/64** サブネットマスクあり）
- IPv4 デフォルトゲートウェイ: **198.51.100.254**
- IPv6 デフォルトゲートウェイ: **2001:db8:1::fffe**
- IPv4 DNS サーバー: **198.51.100.200**
- IPv6 DNS サーバー: **2001:db8:1::ffbb**
- DNS 検索ドメイン: **example.com**

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に **root** 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/ethernet-connection.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure an Ethernet connection with static IP and default gateway
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/ethernet-connection.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md**
- **ansible-playbook(1)** の man ページ

7.10. RHEL システムロールを使用した静的ルートの設定

Networking の RHEL システムロールを使用して、静的ルートを設定できます。

重要

Networking の RHEL システムロールを使用する再生を実行すると、設定値が再生で指定されたものと一致しない場合に、システムロールが同じ名前の既存の接続プロファイルを上書きします。したがって、IP 設定がすでに存在している場合でも、再生でネットワーク接続プロファイルの設定全体を指定する必要があります。それ以外の場合は、ロールはこれらの値をデフォルト値にリセットします。

この手順では、すでに存在するかどうかに応じて、以下の設定で **enp7s0** 接続プロファイルを作成または更新します。

- 静的 IPv4 アドレス: **/24** サブネットマスクを持つ **198.51.100.20**
- 静的 IPv6 アドレス: **2001:db8:1::1** （**/64** サブネットマスクあり）

- IPv4 デフォルトゲートウェイ: **198.51.100.254**
- IPv6 デフォルトゲートウェイ: **2001:db8:1::fffe**
- IPv4 DNS サーバー: **198.51.100.200**
- IPv6 DNS サーバー: **2001:db8:1::ffbb**
- DNS 検索ドメイン: **example.com**
- 静的ルート:
  - ゲートウェイ **198.51.100.1** の **192.0.2.0/24**
  - **203.0.113.0/24** のゲートウェイ **198.51.100.2**

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に root 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/add-static-routes.yml** Playbook を以下の内容で作成します。

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

    vars:
      network_connections:
        - name: enp7s0
          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
    route:
      - network: 192.0.2.0
        prefix: 24
        gateway: 198.51.100.1
      - network: 203.0.113.0
        prefix: 24
        gateway: 198.51.100.2
  state: up
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/add-static-routes.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

検証手順

- ルーティングテーブルを表示します。

```
# ip -4 route
default via 198.51.100.254 dev enp7s0 proto static metric 100
192.0.2.0/24 via 198.51.100.1 dev enp7s0 proto static metric 100
203.0.113.0/24 via 198.51.100.2 dev enp7s0 proto static metric 100
...
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.11. RHEL システムロールを使用した ETHTOOL 機能の設定

Networking の RHEL システムロールを使用して、NetworkManager 接続の **ethtool** 機能を設定できます。

重要

Networking の RHEL システムロールを使用する再生を実行すると、設定値が再生で指定されたものと一致しない場合に、システムロールが同じ名前の既存の接続プロファイルを上書きします。したがって、IP 設定などがすでに存在している場合でも、常にネットワーク接続プロファイルの設定全体が指定されます。それ以外の場合は、ロールはこれらの値をデフォルト値にリセットします。

この手順では、すでに存在するかどうかに応じて、以下の設定で **enp1s0** 接続プロファイルを作成または更新します。

- 静的 **IPv4** アドレス: **/24** サブネットマスクを持つ **198.51.100.20**
- 静的 **IPv6** アドレス: **2001:db8:1::1** （**/64** サブネットマスクあり）
- **IPv4** デフォルトゲートウェイ: **198.51.100.254**
- **IPv6** デフォルトゲートウェイ: **2001:db8:1::fffe**
- **IPv4** DNS サーバー: **198.51.100.200**
- **IPv6** DNS サーバー: **2001:db8:1::ffbb**
- DNS 検索ドメイン: **example.com**
- **ethtool** 機能:
  - 汎用受信オフロード(GRO): 無効
  - Generic segmentation offload(GSO): 有効化
  - TX stream control transmission protocol (SCTP) segmentation: 無効

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に root 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/configure-ethernet-device-with-ethtool-features.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure an Ethernet connection with ethtool features
  hosts: node.example.com
  become: true
  tasks:
  - 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
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

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

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/configure-ethernet-device-with-ethtool-features.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

-u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **ansible-playbook(1)** の man ページ

7.12. RHEL システムロールを使用した ETHTOOL COALESCE の設定

Networking の RHEL システムロールを使用して、NetworkManager 接続の **ethtool** を設定できます。

重要

Networking の RHEL システムロールを使用する再生を実行すると、設定値が再生で指定されたものと一致しない場合に、システムロールが同じ名前の既存の接続プロファイルを上書きします。したがって、IP 設定などがすでに存在している場合でも、常にネットワーク接続プロファイルの設定全体が指定されます。それ以外の場合は、ロールはこれらの値をデフォルト値にリセットします。

この手順では、すでに存在するかどうかに応じて、以下の設定で **enp1s0** 接続プロファイルを作成または更新します。

- 静的 IPv4 アドレス: **/24** サブネットマスクを持つ **198.51.100.20**
- 静的 IPv6 アドレス: **2001:db8:1::1** （**/64** サブネットマスクあり）
- IPv4 デフォルトゲートウェイ: **198.51.100.254**
- IPv6 デフォルトゲートウェイ: **2001:db8:1::fffe**
- IPv4 DNS サーバー: **198.51.100.200**
- IPv6 DNS サーバー: **2001:db8:1::ffbb**
- DNS 検索ドメイン: **example.com**
- **ethtool** coalesce の設定：
  - RX フレーム: **128**
  - TX フレーム: **128**

前提条件

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージが制御ノードにインストールされている。
- Playbook の実行時に root 以外のリモートユーザーを使用する場合は、管理ノードで適切な **sudo** パーミッションが付与される。

手順

1. Playbook の命令を実行するホストのインベントリーがまだ指定されていない場合は、そのホストの IP または名前を Ansible インベントリーファイル **/etc/ansible/hosts** に追加します。

```
node.example.com
```

2. **~/configure-ethernet-device-with-ipareplicacoalesce-settings.yml** Playbook を以下の内容で作成します。

```
---
- name: Configure an Ethernet connection with ethtool coalesce settings
  hosts: node.example.com
  become: true
  tasks:
  - 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:
    coalesce:
      rx_frames: 128
      tx_frames: 128
  state: up
```

3. Playbook を実行します。

   - **root** ユーザーとして管理対象ホストに接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u root ~/configure-ethernet-device-with-ethtoolcoalesce-settings.yml
     ```

   - 管理ホストにユーザーとして接続するには、次のコマンドを実行します。

     ```
     # ansible-playbook -u user_name --ask-become-pass ~/configure-ethernet-device-with-ethtoolcoalesce-settings.yml
     ```

     --ask-become-pass オプションは、**ansible-playbook** コマンドが **-u user_name** オプションで定義したユーザーの **sudo** パスワードを要求するようにします。

   -u user_name オプションを指定しないと、**ansible-playbook** は、コントロールノードに現在ログインしているユーザーとして管理ホストに接続します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md**
- **ansible-playbook(1)** の man ページ

第8章 システムロールの POSTFIX ロール変数

Postfix ロール変数により、ユーザーは Postfix Mail Transfer Agent (MTA) をインストール、設定、および起動できます。

以下のロール変数がこのセクションで定義されています。

- **postfix_conf**:対応している Postfix 設定パラメーターすべてのキー/値のペアが含まれます。初期設定では、この**postfix_conf**には値が設定されていません。

```
For example: `postfix_conf`:
        relayhost: "example.com"
```

- **postfix_check**:設定変更を検証するために、Postfix の起動時に確認が実行されたかどうかを判断します。デフォルト値は true です。

```
For example: `postfix_check: true`
```

- **postfix_backup**:設定のバックアップコピーを 1 つ作成するかどうかを指定します。デフォルトでは、**postfix_backup** の値は false です。

以前のバックアップを上書きする場合は、以下のコマンドを実行します。

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

postfix_backup の値を **true** に変更した場合は、**postfix_backup_multiple** の値も false に設定する必要があります。

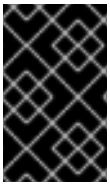
```
For example: postfix_backup: true
        postfix_backup_multiple: false
```

- **postfix_backup_multiple**:ロールが設定のタイムスタンプ付きバックアップコピーを作成するかどうかを決定します。

複数のバックアップコピーを保持するには、次のコマンドを実行します。

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

デフォルトでは、**postfix_backup_multiple** の値は true です。**postfix_backup_multiple:true** 設定は **postfix_backup** をオーバーライドします。**postfix_backup** を使用する場合は、**postfix_backup_multiple:false** を設定する必要があります。



重要

設定パラメーターは削除できません。Postfix ロールを実行する前に、**postfix_conf** を必要な設定パラメーターすべてに設定し、ファイルモジュールを使用して **/etc/postfix/main.cf** を削除します。

8.1. 関連情報

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

第9章 システムロールを使用した SELINUX の設定

9.1. SELINUX システムロールの概要

RHEL システムロールは、複数の RHEL システムをリモートで管理する一貫した構成インターフェースを提供する Ansible ロールおよびモジュールの集合です。SELinux システムロールでは、以下の操作が可能になります。

- SELinux ブール値、ファイルコンテキスト、ポート、およびログインに関連するローカルポリシーの変更を消去します。
- SELinux ポリシーブール値、ファイルコンテキスト、ポート、およびログインの設定
- 指定されたファイルまたはディレクトリーでファイルコンテキストを復元します。
- SELinux モジュールの管理

以下の表は、SELinux システムロールで利用可能な入力変数の概要を示しています。

表9.1 SELinux システムロール変数

ロール変数	説明	CLI の代替手段
selinux_policy	ターゲットプロセスまたは複数レベルのセキュリティー保護を保護するポリシーを選択します。	**/etc/selinux/config** の **SELINUXTYPE**
selinux_state	SELinux モードを切り替えます。	**/etc/selinux/config** の **setenforce** and **SELINUX**
selinux_booleans	SELinux ブール値を有効または無効にします。	**setsebool**
selinux_fcontexts	SELinux ファイルコンテキストマッピングを追加または削除します。	**semanage fcontext**
selinux_restore_dirs	ファイルシステムツリー内の SELinux ラベルを復元します。	**restorecon -R**
selinux_ports	ポートに SELinux ラベルを設定します。	**semanage port**
selinux_logins	ユーザーを SELinux ユーザーマッピングに設定します。	**semanage login**
selinux_modules	SELinux モジュールのインストール、有効化、無効化、または削除を行います。	**semodule**

rhel-system-roles パッケージによりインストールされる **/usr/share/doc/rhel-system-**

roles/selinux/example-selinux-playbook.yml のサンプル Playbook は、Enforcing モードでターゲットポリシーを設定する方法を示しています。Playbook は、複数のローカルポリシーの変更を適用し、**tmp/test_dir/** ディレクトリーのファイルコンテキストを復元します。

SELinux ロール変数の詳細は、**rhel-system-roles** パッケージをインストールし、**/usr/share/doc/rhel-system-roles/selinux/** ディレクトリーの **README.md** または **README.html** ファイルを参照してください。

関連情報

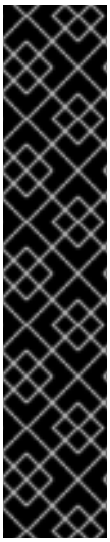
- RHEL システムロールの概要

9.2. SELINUX システムロールを使用した、複数のシステムでの SELINUX 設定の適用

以下の手順に従って、検証した SELinux 設定を使用して Ansible Playbook を準備し、適用します。

前提条件

- 1 つまたは複数の **管理対象ノード** へのアクセスとパーミッション (SELinux システムロールで設定するシステム)。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。



重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. Playbook を準備します。ゼロから開始するか、**rhel-system-roles** パッケージの一部としてインストールされたサンプル Playbook を変更してください。

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

2. シナリオに合わせて Playbook の内容を変更します。たとえば、次の部分では、システムが SELinux モジュール **selinux-local-1.pp** をインストールして有効にします。

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

3. 変更を保存し、テキストエディターを終了します。

4. **host1**、**host2** および **host3** システムで Playbook を実行します。

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

関連情報

- 詳細は、**rhel-system-roles** パッケージをインストールして、**/usr/share/doc/rhel-system-roles/selinux/** ディレクトリーおよび **/usr/share/ansible/roles/rhel-system-roles.selinux/** ディレクトリーを参照してください。

第10章 LOGGING システムロールの使用

システム管理者は、Logging システムロールを使用して、RHEL ホストをロギングサーバーとして設定し、多くのクライアントシステムからログを収集できます。

10.1. LOGGING システムロール

Logging システムロールを使用すると、ローカルおよびリモートホストにロギング設定をデプロイできます。

Logging システムロールを 1 つ以上のシステムに適用するには、**Playbook** でロギング設定を定義します。Playbook は、1 つ以上の play の一覧です。Playbook は YAML 形式で表現され、人が判読できるようになっています。Playbook の詳細は、Ansible ドキュメントの 「Working with playbooks」 を参照してください。

Playbook に従って設定するシステムのセットは、**インベントリーファイル** で定義されます。インベントリーの作成および使用に関する詳細は、Ansible ドキュメントの 「How to build your inventory」 を参照してください。

ロギングソリューションは、ログと複数のログ出力を読み取る複数の方法を提供します。

たとえば、ロギングシステムは以下の入力を受け取ることができます。

- ローカルファイル
- **systemd/journal**
- ネットワーク上で別のロギングシステム

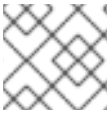
さらに、ロギングシステムでは以下を出力できます。

- **/var/log**ディレクトリーのローカルファイルに保存されているログ
- Elasticsearch に送信されたログ
- 別のロギングシステムに転送されたログ

Logging システムロールでは、シナリオに合わせて入出力を組み合わせることができます。たとえば、**journal** からの入力をローカルのファイルに保存しつつも、複数のファイルから読み込んだ入力を別のロギングシステムに転送してそのローカルのログファイルに保存するようにロギングソリューションを設定できます。

10.2. LOGGING システムロールのパラメーター

Logging システムロール Playbook では、**logging_inputs** パラメーターで入力を、**logging_outputs** パラメーターで出力を、そして **logging_flows** パラメーターで入力と出力の関係を定義します。Logging システムロールは、ロギングシステムの追加設定オプションで、上記の変数を処理します。暗号化を有効にすることもできます。



注記

現在、Logging システムロールで利用可能な唯一のロギングシステムは **Rsyslog** です。

- **logging_inputs**:ロギングソリューションの入力一覧。

- **name**:入力の一意の名前。**logging_flows** での使用: 入力一覧および生成された **config** ファイル名の一部で使用されます。
- **type**:入力要素のタイプ。type は、**roles/rsyslog/{tasks,vars}/inputs/** のディレクトリー名に対応するタスクタイプを指定します。
  - **basics**:**systemd** ジャーナルまたは **unix** ソケットからの入力を設定する入力。
    - **kernel_message**:**true** に設定されている場合は、**imklog** をロードします。デフォルトは **false** です。
    - **use_imuxsock**:**imjournal** の代わりに **imuxsock** を使用します。デフォルトは **false** です。
    - **ratelimit_burst**:**ratelimit_interval** 内に出力できるメッセージの最大数。**use_imuxsock** が false の場合、デフォルトで **20000** に設定されます。**use_imuxsock** が true の場合、デフォルトで **200** に設定されます。
    - **ratelimit_interval**:**ratelimit_burst** を評価する間隔。**use_imuxsock** が false の場合、デフォルトで 600 秒に設定されます。**use_imuxsock** が true の場合、デフォルトで 0 に設定されます。0 はレート制限がオフであることを示します。
    - **persist_state_interval**:ジャーナルの状態は、**value** メッセージごとに永続化されます。デフォルトは **10** です。**use_imuxsock** が false の場合のみ、有効です。
  - **files**:ローカルファイルからの入力を設定する入力。
  - **remote**:ネットワークを介して他のロギングシステムからの入力を設定する入力。
- **state**:設定ファイルの状態。**present** または **absent**。デフォルトは **present** です。

- **logging_outputs**:ロギングソリューションの出力一覧。
  - **files**:ローカルファイルへの出力を設定する出力。
  - **forwards**:別のロギングシステムへの出力を設定する出力。
  - **remote_files**:別のロギングシステムからの出力をローカルファイルに設定する出力。
- **logging_flows**:**logging_inputs** および **logging_outputs** の関係を定義するフローの一覧。**logging_flows** 変数には以下が含まれます。
  - **name**:フローの一意の名前。
  - **inputs**:**logging_inputs** 名の値の一覧。
  - **outputs**:**logging_outputs** 名の値の一覧。

関連情報

- **rhel-system-roles** パッケージでインストールされたドキュメント (**/usr/share/ansible/roles/rhel-system-roles.logging/README.html**)

10.3. ローカルの LOGGING システムロールの適用

以下の手順に従って、Ansible Playbook を準備および適用し、別個のマシンにロギングソリューションを設定します。各マシンはログをローカルに記録します。

前提条件

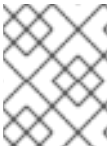
- Logging システムロールを設定する **管理対象ノード** 1 つ以上へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。



注記

デプロイメント時にシステムロールが**rsyslog**をインストールするため、**rsyslog**パッケージをインストールする必要はありません。

手順

1. 必要なロールを定義する Playbook を作成します。
   a. 新しい YAML ファイルを作成し、これをテキストエディターで開きます。以下に例を示します。

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

   b. 以下の内容を挿入します。

   ```
   ---
   - 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. 特定のインベントリーで Playbook を実行します。

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

詳細は以下のようになります。

- **inventory-file** はインベントリーファイルに置き換えます。
- **logging-playbook.yml** は、使用する Playbook に置き換えます。

検証

1. **/etc/rsyslog.conf** ファイルの構文をテストします。

```
# 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. システムがログにメッセージを送信していることを確認します。
   a. テストメッセージを送信します。

```
# logger test
```

   b. **/var/log/messages ログ** を表示します。以下に例を示します。

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

`hostname` はクライアントシステムのホスト名です。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

10.4. ローカルの LOGGING システムロールでのログのフィルタリング

rsyslog プロパティーベースのフィルターをもとにログをフィルターするロギングソリューションをデプロイできます。

前提条件

- Logging システムロールを設定する **管理対象ノード** 1 つ以上へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、

- Red Hat Ansible Core がインストールされている。
- **rhel-system-roles** パッケージがインストールされている。
- 管理対象ノードが記載されているインベントリーファイルがある。

注記

デプロイメント時にシステムロールが **rsyslog** をインストールするため、**rsyslog** パッケージをインストールする必要はありません。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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]
```

この設定を使用すると、**error** 文字列を含むメッセージはすべて **/var/log/errors.log** に記録され、その他のメッセージはすべて **/var/log/others.log** に記録されます。

error プロパティーの値はフィルタリングする文字列に置き換えることができます。

設定に合わせて変数を変更できます。

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

検証

1. **/etc/rsyslog.conf** ファイルの構文をテストします。

```
# 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. システムが **error** 文字列を含むメッセージをログに送信していることを確認します。

   a. テストメッセージを送信します。

```
# logger error
```

   b. 以下のように **/var/log/errors.log** ログを表示します。

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

   hostname はクライアントシステムのホスト名に置き換えます。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **rhel-system-roles** パッケージでインストールされたドキュメント (**/usr/share/ansible/roles/rhel-system-roles.logging/README.html**)

10.5. LOGGING システムロールを使用したリモートロギングソリューションの適用

以下の手順に従って、Red Hat Ansible Core Playbook を準備および適用し、リモートロギングソリューションを設定します。この Playbook では、1 つ以上のクライアントが **systemd-journal** からログを取得し、リモートサーバーに転送します。サーバーは、**remote_rsyslog** および **remote_files** からリモート入力を受信し、リモートホスト名によって名付けられたディレクトリーのローカルファイルにログを出力します。

前提条件

- Logging システムロールを設定する **管理対象ノード** 1 つ以上へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。

注記

デプロイメント時にシステムロールが **rsyslog** をインストールするため、**rsyslog** パッケージをインストールする必要はありません。

手順

1. 必要なロールを定義する Playbook を作成します。

   a. 新しい YAML ファイルを作成し、これをテキストエディターで開きます。以下に例を示します。

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

   b. 以下の内容をファイルに挿入します。

```
---
- 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]
```

host1.example.com はロギングサーバーに置き換えます。

注記

必要に応じて、Playbook のパラメーターを変更することができます。

警告

ロギングソリューションは、サーバーまたはクライアントシステムの SELinux ポリシーで定義され、ファイアウォールで開放されたポートでしか機能しません。デフォルトの SELinux ポリシーには、ポート 601、514、6514、10514、および 20514 が含まれます。別のポートを使用するには、クライアントシステムおよびサーバーシステムで SELinux ポリシーを変更 します。システムロールを使用したファイアウォールの設定は、まだサポートされていません。

2. サーバーおよびクライアントを一覧表示するインベントリーファイルを作成します。
   a. 新しいファイルを作成してテキストエディターで開きます。以下に例を示します。

```
# vi inventory.ini
```

   b. 以下のコンテンツをインベントリーファイルに挿入します。

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

   ここで、

   - **host1.example.com** はロギングサーバーです。
   - **host2.example.com** はロギングクライアントです。

3. インベントリーで Playbook を実行します。

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

   ここで、

   - **inventory.ini** はインベントリーファイルに置き換えます。

- **logging-playbook.yml** は作成した Playbook に置き換えます。

検証

1. クライアントとサーバーシステムの両方で、**/etc/rsyslog.conf** ファイルの構文をテストします。

```
# 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. クライアントシステムがサーバーにメッセージを送信することを確認します。

   a. クライアントシステムで、テストメッセージを送信します。

```
# logger test
```

   b. サーバーシステムで、**/var/log/messages** ログを表示します。以下に例を示します。

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

   host2.example.com は、クライアントシステムのホスト名です。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- RHEL システムロールの使用
- **rhel-system-roles** パッケージでインストールされたドキュメント (**/usr/share/ansible/roles/rhel-system-roles.logging/README.html**)
- RHEL システムロール のナレッジベース記事

10.6. TLS でのロギングシステムロールの使用

Transport Layer Security (TLS) は、コンピューターネットワーク上で安全に通信するために設計された暗号プロトコルです。

管理者は、Logging RHEL システムロールを使用し、Red Hat Ansible Automation Platform を使用したセキュアなログ転送を設定できます。

10.6.1. TLS を使用したクライアントロギングの設定

Logging システムロールを使用して、ローカルマシンにログインしている RHEL システムでロギングを設定し、Ansible Playbook を実行して、ログを TLS でリモートロギングシステムに転送できます。

この手順では、Ansible インベントリーの client グループ内の全ホストに TLS を設定します。TLS プロトコルは、メッセージ送信を暗号化し、ネットワーク経由でログを安全に転送します。

前提条件

- TLS を設定する管理ノードで Playbook の実行権限がある。
- 管理対象ノードがコントロールノードのインベントリーファイルに記載されている。
- **ansible** パッケージおよび **rhel-system-roles** パッケージがコントロールノードにインストールされている。

手順

1. 以下の内容を含む **playbook.yml** ファイルを作成します。

```
---
- 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]
```

Playbook は以下のパラメーターを使用します。

logging_pki_files

このパラメーターを使用して、TLS を設定し、**ca_cert_src**、**cert_src** および **private_key_src** パラメーターを指定する必要があります。

ca_cert

CA 証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。

cert

証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。

private_key

秘密鍵へのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。

ca_cert_src

ローカルの CA 証明書ファイルパスを表します。これはターゲットホストにコピーされます。**ca_cert** を指定している場合は、その場所にコピーされます。

cert_src

ローカルの証明書ファイルパスを表します。これはターゲットホストにコピーされます。**cert** を指定している場合には、その証明書が場所にコピーされます。

private_key_src

ローカルキーファイルのパスを表します。これはターゲットホストにコピーされます。**private_key** を指定している場合は、その場所にコピーされます。

tls

このパラメーターを使用すると、ネットワーク経由でログを安全に転送できるようになります。セキュアなラッパーが必要ない場合は、**tls: true** と設定できます。

2. Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

10.6.2. TLS を使用したサーバーロギングの設定

Logging システムロールを使用して、RHEL システムのログインをサーバーとして設定し、Ansible Playbook を実行して TLS でリモートロギングシステムからログを受信できます。

以下の手順では、Ansible インベントリーの server グループ内の全ホストに TLS を設定します。

前提条件

- TLS を設定する管理ノードで Playbook の実行権限がある。
- 管理対象ノードがコントロールノードのインベントリーファイルに記載されている。
- **ansible** パッケージおよび **rhel-system-roles** パッケージがコントロールノードにインストールされている。

手順

1. 以下の内容を含む **playbook.yml** ファイルを作成します。

```
---
- 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]
```

Playbook は以下のパラメーターを使用します。

logging_pki_files

このパラメーターを使用して、TLS を設定し、**ca_cert_src**、**cert_src** および **private_key_src** パラメーターを指定する必要があります。

ca_cert

CA 証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。

cert

証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。

private_key

秘密鍵へのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。

ca_cert_src

ローカルの CA 証明書ファイルパスを表します。これはターゲットホストにコピーされます。**ca_cert** を指定している場合は、その場所にコピーされます。

cert_src

ローカルの証明書ファイルパスを表します。これはターゲットホストにコピーされます。**cert** を指定している場合には、その証明書が場所にコピーされます。

private_key_src

ローカルキーファイルのパスを表します。これはターゲットホストにコピーされます。**private_key** を指定している場合は、その場所にコピーされます。

tls

このパラメーターを使用すると、ネットワーク経由でログを安全に転送できるようになります。セキュアなラッパーが必要ない場合は、**tls: true** と設定できます。

2. Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

10.7. RELP でのロギングシステムロールの使用

Reliable Event Logging Protocol (RELP) とは、TCP ネットワークを使用する、データとメッセージロギング用のネットワーキングプロトコルのことです。イベントメッセージを確実に配信するので、メッセージの損失が許されない環境で使用できます。

RELP の送信側はコマンド形式でログエントリーを転送し、受信側は処理後に確認応答します。RELP は、一貫性を保つために、転送されたコマンドごとにトランザクション番号を保存し、各種メッセージの復旧します。

RELP Client と RELP Server の間に、リモートロギングシステムを検討することができます。RELP Client はリモートロギングシステムにログを転送し、RELP Server はリモートロギングシステムから送信されたすべてのログを受け取ります。

管理者は Logging システムロールを使用して、ログエントリーが確実に送受信されるようにロギングシステムを設定することができます。

10.7.1. RELP を使用したクライアントロギングの設定

Logging システムロールを使用して、ローカルマシンにログインしている RHEL システムでロギングを設定し、Ansible Playbook を実行して、ログを RELP でリモートロギングシステムに転送できます。

この手順では、Ansible インベントリーの **client** グループ内の全ホストに RELP を設定します。RELP 設定は Transport Layer Security (TLS) を使用して、メッセージ送信を暗号化し、ネットワーク経由でログを安全に転送します。

前提条件

- RELP を設定する管理ノードで Playbook の実行権限がある。
- 管理対象ノードがコントロールノードのインベントリーファイルに記載されている。
- **ansible** パッケージおよび **rhel-system-roles** パッケージがコントロールノードにインストールされている。

手順

1. 以下の内容を含む **playbook.yml** ファイルを作成します。

```
---
- 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]
```

Playbook は、以下の設定を使用します。

- **target**:リモートロギングシステムが稼働しているホスト名を指定する必須パラメーターです。
- **port**: リモートロギングシステムがリッスンしているポート番号です。
- **tls**:ネットワーク上でログをセキュアに転送します。セキュアなラッパーが必要ない場合は、**tls** 変数を **false** に設定します。デフォルトでは **tls** パラメーターは true に設定されますが、RELP を使用する場合には鍵/証明書およびトリプレット {**ca_cert**、**cert**、**private_key**} や {**ca_cert_src**、**cert_src**、**private_key_src**} が必要です。
  - {**ca_cert_src**、**cert_src**、**private_key_src**} のトリプレットを設定すると、デフォルトの場所 (**/etc/pki/tls/certs** と **/etc/pki/tls/private**) を、コントロールノードから転送する管理対象ノードの宛先として使用します。この場合、ファイル名はトリプレットの元の名前と同じです。
  - {**ca_cert**、**cert**、**private_key**} トリプレットが設定されている場合には、ファイルはロギング設定の前にデフォルトのパスを配置する必要があります。
  - トリプレットの両方が設定されている場合には、ファイルはコントロールノードのローカルのパスから管理対象ノードの特定のパスへ転送されます。
- **ca_cert**:CA 証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。
- **cert**:証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。
- **private_key**:秘密鍵へのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。
- **ca_cert_src**:ローカルの CA 証明書ファイルパスを表します。これはターゲットホストにコピーされます。ca_cert が指定している場合は、その場所にコピーされます。
- **cert_src**:ローカルの証明書ファイルパスを表します。これはターゲットホストにコピーされます。cert を指定している場合には、その証明書が場所にコピーされます。
- **private_key_src**:ローカルキーファイルのパスを表します。これはターゲットホストにコピーされます。private_key を指定している場合には、その場所にコピーされます。
- **pki_authmode**:**name** または **fingerprint** の認証モードを使用できます。

- **permitted_servers**:ロギングクライアントが、TLS 経由での接続およびログ送信を許可するサーバーの一覧。
- **inputs**:ロギング入力ディクショナリーの一覧。
- **outputs**:ロギング出力ディクショナリーの一覧。

2. オプション:Playbook の構文を確認します。

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

3. Playbook を実行します。

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

10.7.2. RELP を使用したサーバーログの設定

Logging システムロールを使用して、RHEL システムのログインをサーバーとして設定し、Ansible Playbook を実行して RELP でリモートロギングシステムからログを受信できます。

以下の手順では、Ansible インベントリーの **server** グループ内の全ホストに RELP を設定します。RELP 設定は TLS を使用して、メッセージ送信を暗号化し、ネットワーク経由でログを安全に転送します。

前提条件

- RELP を設定する管理ノードで Playbook の実行権限がある。
- 管理対象ノードがコントロールノードのインベントリーファイルに記載されている。
- **ansible** パッケージおよび **rhel-system-roles** パッケージがコントロールノードにインストールされている。

手順

1. 以下の内容を含む **playbook.yml** ファイルを作成します。

```
---
- 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_
```

Playbook は、以下の設定を使用します。

- **port**: リモートロギングシステムがリッスンしているポート番号です。
- **tls**:ネットワーク上でログをセキュアに転送します。セキュアなラッパーが必要ない場合は、**tls** 変数を **false** に設定します。デフォルトでは **tls** パラメーターは true に設定されますが、RELP を使用する場合には鍵/証明書およびトリプレット {**ca_cert**、**cert**、**private_key**} や {**ca_cert_src**、**cert_src**、**private_key_src**} が必要です。
  - {**ca_cert_src**、**cert_src**、**private_key_src**} のトリプレットを設定すると、デフォルトの場所 (**/etc/pki/tls/certs** と **/etc/pki/tls/private**) を、コントロールノードから転送する管理対象ノードの宛先として使用します。この場合、ファイル名はトリプレットの元の名前と同じです。
  - {**ca_cert**、**cert**、**private_key**} トリプレットが設定されている場合には、ファイルはロギング設定の前にデフォルトのパスを配置する必要があります。
  - トリプレットの両方が設定されている場合には、ファイルはコントロールノードのローカルのパスから管理対象ノードの特定のパスへ転送されます。
- **ca_cert**:CA 証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。
- **cert**:証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。
- **private_key**:秘密鍵へのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。
- **ca_cert_src**:ローカルの CA 証明書ファイルパスを表します。これはターゲットホストにコピーされます。ca_cert が指定している場合は、その場所にコピーされます。
- **cert_src**:ローカルの証明書ファイルパスを表します。これはターゲットホストにコピーされます。cert を指定している場合には、その証明書が場所にコピーされます。
- **private_key_src**:ローカルキーファイルのパスを表します。これはターゲットホストにコピーされます。private_key を指定している場合には、その場所にコピーされます。
- **pki_authmode**:**name** または **fingerprint** の認証モードを使用できます。
- **permitted_clients**:ロギングサーバーが TLS 経由での接続およびログ送信を許可するクライアントの一覧。
- **inputs**:ロギング入力ディクショナリーの一覧。
- **outputs**:ロギング出力ディクショナリーの一覧。

2. オプション:Playbook の構文を確認します。

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

3. Playbook を実行します。

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

10.8. 関連情報

- RHEL システムロールの使用
- **rhel-system-roles** パッケージでインストールされたドキュメントは、**/usr/share/ansible/roles/rhel-system-roles.logging/README.html** にあります。
- RHEL システムロール
- **ansible-playbook(1)** man ページ。

第11章 SSH システムロールを使用した安全な通信の設定

管理者は、SSHD システムロールを使用して SSH サーバーを設定し、SSH システムロールを使用し、Ansible Core パッケージを使用して同時に任意の数の RHEL システムで SSH クライアントを一貫して設定できます。

11.1. SSH SERVER システムロール変数

SSH Server システムロール Playbook では、設定と制限に応じて、SSH 設定ファイルのパラメーターを定義できます。

これらの変数が設定されていない場合には、システムロールは RHEL のデフォルト値と同じ **sshd_config** ファイルを作成します。

どのような場合でも、ブール値は **sshd** 設定で適切に **yes** と **no** としてレンダリングされます。一覧を使用して複数行の設定項目を定義できます。以下に例を示します。

```
sshd_ListenAddress:
  - 0.0.0.0
  - '::'
```

レンダリングは以下のようになります。

```
ListenAddress 0.0.0.0
ListenAddress ::
```

SSH Server システムのロールの変数

sshd_enable

False に設定すると、ロールは完全に無効になります。デフォルトは **True** です。

sshd_skip_defaults

True に設定すると、システムロールではデフォルト値が適用されません。代わりに、**sshd** dict または **sshd_Key** 変数のいずれかを使用して、設定のデフォルト値をすべて指定します。デフォルトは **False** です。

sshd_manage_service

False に設定すると、サービスは管理対象ではなくなるので、起動時に有効化されず、起動または再読み込みされません。Ansible サービスモジュールが現在 AIX で **enabled** になっていないため、コンテナーまたは AIX 内で実行する時以外はデフォルトで **True** に設定されます。

sshd_allow_reload

False に設定すると、設定の変更後に **sshd** は再読み込みされません。これはトラブルシューティングで役立ちます。変更した設定を適用するには、**sshd** を手動で再読み込みします。AIX を除き、**sshd_manage_service** と同じ値にデフォルト設定されます。ここで、**sshd_manage_service** はデフォルトで **False** に設定されますが、**sshd_allow_reload** はデフォルトで **True** に設定されます。

sshd_install_service

True に設定すると、ロールは **sshd** サービスのサービスファイルをインストールします。これにより、オペレーティングシステムで提供されるファイルが上書きされます。2 つ目のインスタンスを設定し、**sshd_service** 変数も変更しない限り、**True** に設定しないでください。デフォルトは **False** です。
ロールは、以下の変数でテンプレートとして参照するファイルを使用します。

-

```
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

この変数により **sshd** サービス名が変更されます。これは、2 つ目の **sshd** サービスインスタンスを設定するのに役立ちます。

sshd

設定が含まれる dict。以下に例を示します。

```
sshd:
  Compression: yes
  ListenAddress:
    - 0.0.0.0
```

sshd_OptionName

dict の代わりに、**sshd_** プレフィックスとオプション名で構成される単純な変数を使用してオプションを定義できます。簡単な変数は、**sshd** dict の値を上書きします。以下に例を示します。

```
sshd_Compression: no
```

sshd_match and **sshd_match_1** to **sshd_match_9**

dict のリスト、または Match セクションの dict のみ。これらの変数は、**sshd** dict で定義されている一致するブロックを上書きしないことに注意してください。すべてのソースは作成された設定ファイルに反映されます。

SSH Server システムロールのセカンダリー変数

これらの変数を使用して、サポートされている各プラットフォームに対応するデフォルトを上書きすることができます。

sshd_packages

この変数を使用して、インストール済みパッケージのデフォルト一覧を上書きできます。

sshd_config_owner、sshd_config_group、sshd_config_mode

このロールは、これらの変数を使用して生成する **openssh** 設定ファイルの所有権およびパーミッションを設定できます。

sshd_config_file

このロールが作成した **openssh** サーバー設定を保存するパス。

sshd_config_namespace

この変数のデフォルト値は null です。これは、ロールがシステムのデフォルトを含む設定ファイルの内容全体を定義することを意味します。または、この変数を使用して、他のロールから、またはドロップインディレクトリーをサポートしないシステムの 1 つの Playbook 内の複数の場所から、このロールを呼び出すことができます。**sshd_skip_defaults** 変数は無視され、この場合、システムのデフォルトは使用されません。

この変数が設定されている場合、ロールは指定された namespace の下の既存の設定ファイルの設定スニペットに指定する設定を配置します。シナリオにロールを複数回適用する必要がある場合は、アプリケーションごとに異なる namespace を選択する必要があります。

注記

openssh 設定ファイルの制限は引き続き適用されます。たとえば、設定ファイルで指定した最初のオプションだけが、ほとんどの設定オプションで有効です。

技術的には、ロールは他の一致ブロックが含まれていない限り、スニペットを "Match all" ブロックに配置し、既存の設定ファイル内の以前の一致ブロックに関係なく適用されるようにします。これにより、異なるロール呼び出しから競合しないオプションを設定できます。

sshd_binary

openssh の **sshd** 実行可能ファイルへのパス。

sshd_service

sshd サービスの名前。デフォルトでは、この変数には、ターゲットプラットフォームが使用する **sshd** サービスの名前が含まれます。ロールが **sshd_install_service** 変数を使用する場合は、これを使用してカスタムの **sshd** サービスの名前を設定することもできます。

sshd_verify_hostkeys

デフォルトは **auto** です。**auto** に設定すると、生成された設定ファイルに存在するホストキーがすべて一覧表示され、存在しないパスが生成されます。また、パーミッションおよびファイルの所有者はデフォルト値に設定されます。これは、ロールがデプロイメント段階で使用され、サービスが最初の試行で起動できるようにする場合に便利です。このチェックを無効にするには、この変数を空のリスト **[]** に設定します。

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

これらの変数を使用して、**sshd_verify_hostkeys** からホストキーの所有権とパーミッションを設定します。

sshd_sysconfig

RHEL ベースのシステムでは、この変数は **sshd** サービスに関する追加情報を設定します。**true** に設定すると、このロールは以下の設定に基づいて **/etc/sysconfig/sshd** 設定ファイルも管理します。デフォルトは **false** です。

sshd_sysconfig_override_crypto_policy

RHEL では、**true** に設定すると、この変数はシステム全体の暗号化ポリシーを上書きします。デフォルトは **false** です。

sshd_sysconfig_use_strong_rng

RHEL ベースのシステムでは、この変数により、**sshd** は、引数として指定されたバイト数を使用して、**openssl** 乱数ジェネレーターを強制的に再シードすることができます。デフォルトは **0** で、この機能を無効にします。システムにハードウェア乱数ジェネレーターがない場合は、この機能を有効にしないでください。

11.2. SSH SERVER システムロールを使用した OPENSSH サーバーの設定

SSH Server システムロールを使用して、Ansible Playbook を実行することで複数の SSH サーバーを設定できます。

注記

SSH Server システムロールは、SSH および SSHD 設定を変更する他のシステムロール (ID 管理 RHEL システムロールなど) とともに使用できます。設定が上書きされないようにするには、SSH Server ロールがネームスペース (RHEL 8 以前のバージョン) またはドロップインディレクトリー (RHEL 9) を使用していることを確認してください。

前提条件

- 1 つ以上の **管理対象ノード** (SSHD システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. SSH Server システムロールのサンプル Playbook をコピーします。

```
# cp /usr/share/doc/rhel-system-roles/sshd/example-root-login-playbook.yml path/custom-playbook.yml
```

2. 以下の例のように、テキストエディターでコピーした Playbook を開きます。

```
# 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
```

Playbook は、以下のように、管理対象ノードを SSH サーバーとして設定します。

- パスワードと **root** ユーザーのログインが無効である
- **192.0.2.0/24** のサブネットからのパスワードおよび **root** ユーザーのログインのみが有効である

設定に合わせて変数を変更できます。詳細は、SSHD Server System Role variables を参照してください。

3. オプション:Playbook の構文を確認します。

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

4. インベントリーファイルで Playbook を実行します。

```
# 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
```

検証

1. SSH サーバーにログインします。

```
$ ssh user1@10.1.1.1
```

ここで、

- **user1** は、SSH サーバーのユーザーです。
- **10.1.1.1** は、SSH サーバーの IP アドレスです。

2. SSH サーバーの **sshd_config** ファイルの内容を確認します。

```
$ vim /etc/ssh/sshd_config

# Ansible managed
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
AuthorizedKeysFile .ssh/authorized_keys
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
```

```
PasswordAuthentication no
PermitRootLogin no
PrintMotd no
Subsystem sftp /usr/libexec/openssh/sftp-server
SyslogFacility AUTHPRIV
UsePAM yes
X11Forwarding yes
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
```

3. **192.0.2.0/24** サブネットから root としてサーバーに接続できることを確認します。

   a. IP アドレスを確認します。

   ```
   $ hostname -I
   192.0.2.1
   ```

   IP アドレスが **192.0.2.1** - **192.0.2.254** 範囲にある場合は、サーバーに接続できます。

   b. **root** でサーバーに接続します。

   ```
   $ ssh root@10.1.1.1
   ```

関連情報

- **/usr/share/doc/rhel-system-roles/sshd/README.md** ファイル。
- **ansible-playbook(1)** man ページ。

11.3. SSH CLIENT システムロール変数

SSH Client システムロール Playbook では、設定および制限に応じて、クライアント SSH 設定ファイルのパラメーターを定義できます。

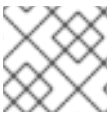
これらの変数が設定されていない場合には、システムロールは RHEL のデフォルト値と同じグローバル **ssh_config** ファイルを作成します。

どのような場合でも、ブール値は **ssh** 設定で適切に **yes** または **no** とレンダリングされます。一覧を使用して複数行の設定項目を定義できます。以下に例を示します。

```
LocalForward:
  - 22 localhost:2222
  - 403 localhost:4003
```

レンダリングは以下のようになります。

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```



注記

設定オプションでは、大文字と小文字が区別されます。

SSH Client システムロールの変数

ssh_user

システムロールでユーザー固有の設定を変更するように、既存のユーザー名を定義できます。ユーザー固有の設定は、指定したユーザーの **~/.ssh/config** に保存されます。デフォルト値は null で、すべてのユーザーに対するグローバル設定を変更します。

ssh_skip_defaults

デフォルトは **auto** です。**auto** に設定すると、システムロールはシステム全体の設定ファイル **/etc/ssh/ssh_config** を読み取り、そこで定義した RHEL のデフォルトを保持します。**ssh_drop_in_name** 変数を定義してドロップイン設定ファイルを作成すると、**ssh_skip_defaults** 変数が自動的に無効化されます。

ssh_drop_in_name

システム全体のドロップインディレクトリーに置かれたドロップイン設定ファイルの名前を定義します。この名前は、変更する設定ファイルを参照するテンプレート **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** で使用されます。システムがドロップインディレクトリーに対応していない場合、デフォルト値は null です。システムがドロップインディレクトリーに対応している場合、デフォルト値は **00-ansible** です。

警告

システムがドロップインディレクトリーに対応していない場合は、このオプションを設定すると、プレイに失敗します。

推奨される形式は **NN-name** です。**NN** は、設定ファイルの指定に使用する 2 桁の番号で、 **name** はコンテンツまたはファイルの所有者を示す名前になります。

ssh

設定オプションとその値が含まれる dict。

ssh_OptionName

dict の代わりに、**ssh_** プレフィックスとオプション名で構成される単純な変数を使用してオプションを定義できます。簡単な変数は、**ssh** dict の値を上書きします。

ssh_additional_packages

このロールは、一般的なユースケースに必要な **openssh パッケージ および openssh-clients** パッケージを自動的にインストールします。ホストベースの認証用に **openssh-keysign** などの追加のパッケージをインストールする必要がある場合は、この変数で指定できます。

ssh_config_file

ロールが生成した設定ファイルを保存するパス。デフォルト値:

- システムにドロップインディレクトリーがある場合、デフォルト値は **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** テンプレートで定義されます。
- システムにドロップインディレクトリーがない場合、デフォルト値は **/etc/ssh/ssh_config** になります。
- **ssh_user** 変数が定義されている場合、デフォルト値は **~/.ssh/config** になります。

ssh_config_owner, ssh_config_group, ssh_config_mode

作成した設定ファイルの所有者、グループ、およびモード。デフォルトでは、ファイルの所有者は **root:root** で、モードは **0644** です。**ssh_user** が定義されている場合、モードは **0600** で、owner と group は **ssh_user** 変数で指定したユーザー名から派生します。

11.4. SSH CLIENT システムロールを使用した OPENSSH クライアントの設定

SSH Client システムロールを使用して、Ansible Playbook を実行して複数の SSH クライアントを設定できます。

注記

SSH Client システムロールは、SSH および SSHD 設定を変更する他のシステムロール (ID 管理 RHEL システムロールなど) とともに使用できます。設定が上書きされないようにするには、SSH Client ロールがドロップインディレクトリー (RHEL 8 から デフォルト) を使用していることを確認してください。

前提条件

- 1 つ以上の **管理対象ノード** (SSH Client システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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
```

この Playbook は、以下の設定を使用して、管理対象ノードで **root** ユーザーの SSH クライアント設定を行います。

- 圧縮が有効になっている。
- ControlMaster multiplexing が **auto** に設定されている。
- **example.com** ホストに接続するための **example** エイリアスが **user1** である。
- ホストエイリアスの **example** が作成されている。(これはユーザー名が **user1** の **example.com** ホストへの接続を表します。)
- X11 転送が無効化されている。

必要に応じて、これらの変数は設定に合わせて変更できます。詳細は、SSH System Role variables を参照してください。

2. オプション:Playbook の構文を確認します。

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

3. インベントリーファイルで Playbook を実行します。

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

検証

- テキストエディターで SSH 設定ファイルを開いて、管理対象ノードが正しく設定されていることを確認します。以下に例を示します。

```
# vi ~root/.ssh/config
```

上記の Playbook の例の適用後に、設定ファイルの内容は以下のようになるはずです。

```
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
```

```
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

11.5. 非排他的設定での SSH サーバーシステムロールの使用

通常、SSH Server システムロールを適用すると、設定全体が上書きされます。これは、たとえば別のシステムロールや Playbook などを使用して、以前に設定を調整している場合に問題が発生する可能性があります。他のオプションをそのまま維持しながら、選択した設定オプションのみに SSH Server システムロールを適用するには、非排他的設定を使用できます。

RHEL 8 以前では、設定スニペットを使用して非排他的設定を適用することができます。詳細は、RHEL 8 ドキュメントの Using the SSH Server System Role for non-exclusive configuration を参照してください。

RHEL 9 では、ドロップインディレクトリーのファイルを使用して、非排他的設定を適用することができます。デフォルトの設定ファイルは、**/etc/ssh/sshd_config.d/00-ansible_system_role.conf** としてドロップインディレクトリーにすでに配置されています。

前提条件

- 1 つ以上の **管理対象ノード** (SSH Server システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。
  - 別の RHEL システムロールの Playbook。詳細は、Applying a role を参照してください。

手順

1. **sshd_config_file** 変数を含む設定スニペットを Playbook に追加します。

```
---
- 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
```

sshd_config_file 変数で、SSH Server システムロールが設定オプションを書き込む **.conf** ファイルを定義します。

設定ファイルが適用される順序を指定するには、2 桁のプレフィックス（例: **42-**）を使用します。

Playbook をインベントリーに適用すると、ロールは **sshd_config_file** 変数で定義されるファイルに次の設定オプションを追加します。

```
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

検証

- オプション:Playbook の構文を確認します。

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

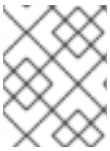
関連情報

- **/usr/share/doc/rhel-system-roles/sshd/README.md** ファイル。
- **ansible-playbook(1)** man ページ。

第12章 VPN RHEL システムロールを使用した IPSEC との VPN 接続の設定

VPNシステムロールを使用すると、Red Hat Ansible Automation Platformを使用してRHELシステムでVPN接続を構成できます。これを使用して、ホスト間、ネットワーク間、VPNリモートアクセスサーバー、およびメッシュ構成をセットアップできます。

ホスト間接続の場合、ロールは、必要に応じてキーを生成するなど、デフォルトのパラメーターを使用して、**vpn_connections**のリスト内のホストの各ペア間にVPNトンネルを設定します。または、リストされているすべてのホスト間に日和見メッシュ構成を作成するように構成することもできます。この役割は、**hosts**の下にあるホストの名前がAnsibleインベントリで使用されているホストの名前と同じであり、それらの名前を使用してトンネルを構成できることを前提としています。



注記

VPN RHEL システムロールは現在、VPN プロバイダーとして IPsec 実装である Libreswan のみをサポートしています。

12.1. VPNシステムロールを使用してIPSECでホスト間VPNの作成

VPNシステムロールを使用して、コントロールノードでAnsible Playbookを実行することにより、ホスト間接続を構成できます。これにより、インベントリファイルにリストされているすべての管理対象ノードが構成されます。

前提条件

- 1 つ以上の **管理対象ノード** (VPN システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

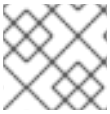
```
- name: Host to host VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
```

このPlaybookは、システムロールによって自動生成されたキーを使用した事前共有キー認証を使用して、**managed_node1からmanaged_node2**への接続を設定します。

2. オプション:ホストの **vpn_connections** リストに次のセクションを追加して、管理対象ホストから、インベントリファイルに記述されていない外部ホストへの接続を設定します。

```
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
          external_node:
            hostname: 192.0.2.2
```

これは、追加の接続（**managed_node1からexternal_node**）へと（**managed_node2からexternal_node**）を設定します。



注記

接続は管理対象ノードでのみ設定され、外部ノードでは設定されません。

1. オプション:**vpn_connections** 内の追加セクション (コントロールプレーンやデータプレーンなど) を使用して、管理対象ノードに複数の VPN 接続を指定できます。

```
- 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. オプション:設定に合わせて変数を変更できます。詳細は、**/usr/share/doc/rhel-system-roles/vpn/README.md**ファイルを参照してください。

3. オプション:Playbook の構文を確認します。

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

4. インベントリーファイルで Playbook を実行します。

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

検証

1. 管理対象ノードで、接続が正常にロードされていることを確認します。

```
# ipsec status | grep connection.name
```

connection.nameを、このノードからの接続の名前（たとえば、**managed_node1-to-managed_node2**）に置き換えます。

注記

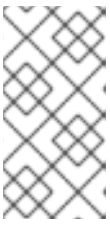
デフォルトでは、ロールは、各システムの観点から作成する接続ごとにわかりやすい名前を生成します。たとえば、**managed_node1**と**managed_node2**との間の接続を作成するときに、**managed_node1**上のこの接続のわかりやすい名前は**managed_node1-to-managed_node2**ですが、**managed_node2** では、この接続の名前は**managed_node2-to-managed_node1**となります。

1. 管理対象ノードで、接続が正常に開始されたことを確認します。

```
# ipsec trafficstatus | grep connection.name
```

2. オプション:接続が正常に読み込まれなかった場合は、次のコマンドを入力して手動で接続を追加します。これにより、接続の確立に失敗した理由を示す、より具体的な情報が提供されます。

```
# ipsec auto --add connection.name
```



注記

接続の読み込みおよび開始のプロセス中に発生した可能性のあるエラーは、ログに報告されます。ログは、**/var/log/pluto.log**にあります。これらのログは解析が難しいため、代わりに接続を手動で追加して、標準出力からログメッセージを取得してみてください。

12.2. VPNシステムロールを使用してIPSECで日和見メッシュVPN接続の作成

VPNシステムロールを使用して、コントロールノードでAnsible Playbookを実行することにより、認証に証明書を使用する日和見メッシュVPN接続を構成できます。これにより、インベントリファイルにリストされているすべての管理対象ノードが構成されます。

証明書による認証は、Playbookで**auth_method: cert**パラメーターを定義することによって設定されます。VPNシステムロールは、**/etc/ipsec.d**ディレクトリで定義されているIPsecネットワークセキュリティサービス（NSS）暗号ライブラリに必要な証明書が含まれていることを前提としています。デフォルトでは、ノード名が証明書のニックネームとして使用されます。この例では、これは**managed_node1**です。インベントリで**cert_name**属性を使用して、さまざまな証明書名を定義できます。

次の手順例では、Ansible Playbook を実行するシステムであるコントロールノードは、両方の管理対象ノード（192.0.2.0/24）と同じクラスレスドメイン間ルーティング（CIDR）番号を共有し、IPアドレスは192.0.2.7になります。したがって、コントロールノードは、CIDR 192.0.2.0/24用に自動的に作成されるプライベートポリシーに該当します。

再生中のSSH接続の損失を防ぐために、コントロールノードの明確なポリシーがポリシーのリストに含まれています。ポリシーリストには、CIDRがデフォルトと等しい項目もあることに注意してください。これは、このPlaybookがデフォルトポリシーのルールを上書きして、private-or-clearではなくprivateにするためです。

前提条件

- 1 つ以上の **管理対象ノード** (VPN システムロールで設定するシステム) へのアクセスおよびパーミッション。
  - すべての管理対象ノードで、**/etc/ipsec.d**ディレクトリのNSSデータベースには、ピア認証に必要なすべての証明書が含まれています。デフォルトでは、ノード名が証明書のニックネームとして使用されます。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
- 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
```

2. オプション:設定に合わせて変数を変更できます。詳細は、**/usr/share/doc/rhel-system-roles/vpn/README.md**ファイルを参照してください。

3. オプション:Playbook の構文を確認します。

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

4. インベントリーファイルで Playbook を実行します。

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

12.3. 関連情報

- VPN システムロールで使用するパラメーターの詳細と、VPN システムロールに関する追加情報は、**/usr/share/doc/rhel-system-roles/vpn/README.md** ファイルを参照してください。
- **ansible-playbook** コマンドの詳細は、**ansible-playbook(1)** の man ページを参照してください。

第13章 システム間でのカスタム暗号化ポリシーの設定

管理者は、Cryptographic Policies RHEL システムロール を使用して、Ansible Core パッケージを使用し、多くの異なるシステムでカスタム暗号化ポリシーを迅速かつ一貫して設定できます。

13.1. CRYPTOGRAPHIC POLICIES システムロール変数とファクト

Cryptographic Policies システムロール Playbook では、設定および制限に合わせて、暗号化ポリシー設定ファイルのパラメーターを定義できます。

変数を設定しない場合には、システムロールではシステムが設定されず、ファクトのみが報告されます。

Cryptographic Policies システムロールの一部の変数

crypto_policies_policy

管理対象ノードにシステムロールを適用する暗号化ポリシーを決定します。異なる暗号化ポリシーの詳細は、「システム全体の暗号化ポリシー」を参照してください。

crypto_policies_reload

yes に設定すると、暗号化ポリシーの適用後に、影響を受けるサービス (現在 **ipsec**、**バインド**、および **sshd** サービス) でリロードされます。デフォルトは **yes** です。

crypto_policies_reboot_ok

yes に設定されており、システムロールで暗号化ポリシーを変更した後に再起動が必要な場合には、**crypto_policies_reboot_required** を **yes** に設定します。デフォルトは **no** です。

Cryptographic Policies システムロールにより設定されるファクト

crypto_policies_active

現在選択されているポリシーを一覧表示します。

crypto_policies_available_policies

システムで利用可能なすべてのポリシーを表示します。

crypto_policies_available_subpolicies

システムで利用可能なすべてのサブポリシーを表示します。

関連情報

- システム全体のカスタム暗号化ポリシーの作成および設定

13.2. CRYPTOGRAPHIC POLICIES システムロールを使用したカスタム暗号化ポリシーの設定

Cryptographic Policies システムロールを使用すると、単一のコントロールノードから多数の管理対象ノードを設定できます。

前提条件

- Crypto Policies システムロールを設定する **管理対象ノード** 1 つ以上へのアクセスおよびパーミッション。

- コントロールノード (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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
```

FUTURE の値は、任意の暗号化ポリシー (例: **DEFAULT**、**LEGACY**、および **FIPS:OSPP**) に置き換えることができます。

crypto_policies_reboot_ok: true 変数を設定すると、システムロールで暗号化ポリシーを変更した後にシステムが再起動されます。

詳細は「Crypto Policies システムロール変数およびファクト」を参照してください。

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

検証

1. コントロールノードで (例: **verify_playbook.yml**) という名前の別の Playbook を作成します。

```
- hosts: all
  tasks:
 - name: Verify active crypto policy
   include_role:
     name: rhel-system-roles.crypto_policies

 - debug:
     var: crypto_policies_active
```

この Playbook では、システムの設定は変更されず、管理対象ノードのアクティブなポリシーだけを報告します。

2. 同じインベントリーファイルで Playbook を実行します。

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

TASK [debug] ***************************
ok: [host] => {
    "crypto_policies_active": "FUTURE"
}
```

"crypto_policies_active": 変数は、管理対象ノードでアクティブなポリシーを表示します。

13.3. 関連情報

- **/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md** ファイル
- **ansible-playbook(1)** man ページ。
- RHEL システムロールのインストール
- システムロールの適用

第14章 CLEVIS および TANG のシステムロールの使用

14.1. CLEVIS および TANG のシステムロールの概要

RHEL システムロールは、複数の RHEL システムをリモートで管理する一貫した構成インターフェースを提供する Ansible ロールおよびモジュールの集合です。

Clevis および Tang を使用した PBD (Policy-Based Decryption) ソリューションの自動デプロイメント用 Ansible ロールを使用することができます。**rhel-system-roles** パッケージには、これらのシステムロール、関連する例、リファレンスドキュメントが含まれます。

Network Bound Disk Encryption Client システムロールにより、複数の Clevis クライアントを自動的にデプロイできます。Network Bound Disk Encryption Client ロールは、Tang バインディングのみをサポートしており、現時点では TPM2 バインディングには使用できない点に留意してください。

Network Bound Disk Encryption Client ロールには、LUKS を使用して暗号化されているボリュームが必要です。このロールは、LUKS 暗号化ボリュームを 1 つまたは複数の NBDE (Network-Bound) サーバー (Tang サーバー) にバインドすることに対応します。既存のボリューム暗号をパスフレーズで保持するか、削除できます。パスフレーズを削除したら、NBDE のみを使用してボリュームのロックを解除できます。これは、システムのプロビジョニング後に削除する必要がある一時鍵またはパスワードを使用して、ボリュームが最初に暗号化されている場合に役立ちます。

パスフレーズと鍵ファイルの両方を指定する場合、ロールは最初に指定したものを使用します。有効なバインディングが見つからないと、既存のバインディングからパスフレーズを取得しようとします。

PBD では、デバイスをスロットにマッピングするものとしてバインディングを定義します。つまり、同じデバイスに複数のバインディングを設定できます。デフォルトのスロットは slot 1 です。

Network Bound Disk Encryption Client ロールは、**state** 変数も提供します。新しいバインディングを作成するか、既存のバインディングを更新する場合は、**present** を使用します。**clevis luks bind** とは異なり、**state: present** を使用してデバイススロットにある既存のバインディングを上書きすることもできます。**absent** に設定すると、指定したバインディングが削除されます。

Network Bound Disk Encryption Server システムロールを使用すると、自動ディスク暗号化ソリューションの一部として、Tang サーバーをデプロイして管理できます。このロールは以下の機能をサポートします。

- Tang 鍵のローテーション
- Tang 鍵のデプロイおよびバックアップ

関連情報

- NBDE (Network-Bound Disk Encryption) ロール変数の詳細は、**rhel-system-roles** パッケージをインストールし、**/usr/share/doc/rhel-system-roles/nbde_client/** と **/usr/share/doc/rhel-system-roles/nbde_server/** ディレクトリーの **README.md** と **README.html** ファイルを参照してください。
- たとえば、system-roles Playbook の場合は、**rhel-system-roles** パッケージをインストールし、**/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** ディレクトリーを参照してください。
- RHEL システムロールの詳細は、Introduction to RHEL System Roles を参照してください。

14.2. 複数の TANG サーバーを設定する NBDE SERVER システムロールの使用

以下の手順に従って、Tangサーバー設定を含む Ansible Playbook を準備および適用します。

前提条件

- 1 つ以上の **管理対象ノード** (NBDE Server システムロールで設定するシステム) へのアクセスおよびパーミッション。
- コントロールノード (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. Tang サーバーの設定が含まれる Playbook を準備します。ゼロから開始するか、または **/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** ディレクトリーにある Playbook のいずれかのサンプルを使用することができます。

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/simple_deploy.yml
./my-tang-playbook.yml
```

2. 選択したテキストエディターで Playbook を編集します。以下に例を示します。

```
# vi my-tang-playbook.yml
```

3. 必要なパラメーターを追加します。以下の Playbook の例では、Tang サーバーのデプロイとキーローテーションを確実に実行します。

```
---
- hosts: all
```

```
vars:
  nbde_server_rotate_keys: yes

roles:
  - rhel-system-roles.nbde_server
```

4. 終了した Playbook を適用します。

```
# ansible-playbook -i inventory-file my-tang-playbook.yml
```

ここで * **inventory-file** はインベントリーファイル、* **logging-playbook.yml** は Playbook も置き換えます。

重要

Clevis がインストールされているシステムで **grubby** ツールを使用して、システム起動時の早い段階で Tang ピンのネットワークを利用できるようにするには、次のコマンドを実行します。

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

関連情報

- 詳細は、**rhel-system-roles** パッケージをインストールして、**/usr/share/doc/rhel-system-roles/nbde_server/** ディレクトリーおよび **usr/share/ansible/roles/rhel-system-roles.nbde_server/** ディレクトリーを参照してください。

14.3. 複数の CLEVIS クライアントを設定する NBDE CLIENT システムロールの使用

手順に従って、Clevisクライアント設定を含む Ansible Playbook を準備および適用します。

注記

NBDE Client システムロールは、Tang バインディングのみをサポートします。これは、現時点では TPM2 バインディングに使用できないことを意味します。

前提条件

- 1 つ以上の **管理対象ノード** (NBDE Client システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
- Ansible Core パッケージがコントロールマシンにインストールされている。
- **rhel-system-roles** パッケージが、Playbook を実行するシステムにインストールされている。

手順

1. Clevis クライアントの設定が含まれる Playbook を準備します。ゼロから開始するか、または **/usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/** ディレクトリーにある Playbook のいずれかのサンプルを使用することができます。

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/high_availability.yml
./my-clevis-playbook.yml
```

2. 選択したテキストエディターで Playbook を編集します。以下に例を示します。

```
# vi my-clevis-playbook.yml
```

3. 必要なパラメーターを追加します。以下の Playbook の例では、2 つの Tang サーバーのうち少なくとも 1 台が利用可能な場合に、LUKS で暗号化した 2 つのボリュームを自動的にアンロックするように Clevis クライアントを設定します。

```
---
- 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. 終了した Playbook を適用します。

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

重要

Clevis がインストールされているシステムで **grubby** ツールを使用して、システム起動時の早い段階で Tang ピンのネットワークを利用できるようにするには、次のコマンドを実行します。

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

関連情報

- パラメーターの詳細と、NBDE Client システムロールに関する追加情報は、**rhel-system-roles** パッケージをインストールし、**/usr/share/doc/rhel-system-roles/nbde_client/** および **/usr/share/ansible/roles/rhel-system-roles.nbde_client/** ディレクトリーを参照してください。

第15章 RHEL システムロールを使用した証明書の要求

Certificate Issuance and Renewal システムロールを使用して、証明書を発行および管理できます。

本章では、以下のトピックについて説明します。

- Certificate システムロール
- Certificate システムロールを使用した新しい自己署名証明書の要求
- Certificate システムロールを使用した IdM CA からの新しい証明書の要求

15.1. CERTIFICATE ISSUANCE AND RENEWAL システムロール

Certificate Issuance and Renewal システムロールを使用して、Ansible Core を使用し、TLS および SSL の証明書の発行および更新を管理できます。

ロールは **certmonger** を証明書プロバイダーとして使用し、自己署名証明書の発行と更新、および IdM 統合認証局 (CA) の使用を現時点でサポートしています。

Certificate Issuance and Renewal システムロールを使用すると、Ansible Playbook で以下の変数を使用できます。

certificate_wait

タスクが証明書を発行するまで待機するかどうかを指定します。

certificate_requests

発行する各証明書とそのパラメーターを表すには、次のコマンドを実行します。

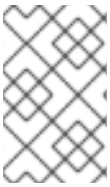
関連情報

- **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** ファイルを参照してください。
- Getting started with RHEL System Roles を参照してください。

15.2. CERTIFICATE ISSUANCE AND RENEWAL システムロールを使用した新しい自己署名証明書の要求

Certificate Issuance and Renewal システムロールでは、Ansible Core を使用して自己署名の証明書を発行できます。

このプロセスは、**certmonger** プロバイダーを使用し、**getcert** コマンドで証明書を要求します。



注記

デフォルトでは、**certmonger** は有効期限が切れる前に証明書の更新を自動的に試行します。これは、Ansible Playbook の **auto_renew** パラメーターを **no** に設定すると無効にできます。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。

- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。
  RHEL システムロールと、その適用方法の詳細は、Getting started with RHEL System Roles を参照してください。

手順

1. **オプション:inventory.file** などのインベントリーファイルを作成します。

```
$ touch inventory.file
```

2. インベントリーファイルを開き、証明書を要求するホストを定義します。以下に例を示します。

```
[webserver]
server.idm.example.com
```

3. Playbook ファイルを作成します (例: **request-certificate.yml**)。
   - **webserver** など、証明書を要求するホストを含むように **hosts** を設定します。
   - **certificate_requests** 変数を設定して以下を追加します。
     - **name** パラメーターを希望する証明書の名前 (**mycert** など) に設定します。
     - **dns** パラメーターを ***.example.com** などの証明書に含むドメインに設定します。
     - **ca** パラメーターを **self-sign** に設定します。
   - **roles** の下に **rhel-system-roles.certificate** ロールを設定します。
     以下は、この例の Playbook ファイルです。

```
---
- hosts: webserver

  vars:
    certificate_requests:
      - name: mycert
        dns: "*.example.com"
        ca: self-sign

  roles:
    - rhel-system-roles.certificate
```

4. ファイルを保存します。
5. Playbook を実行します。

```
$ ansible-playbook -i inventory.file request-certificate.yml
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** ファイルを参照してください。

- man ページの **ansible-playbook(1)** を参照してください。

15.3. CERTIFICATE ISSUANCE AND RENEWAL システムロールを使用した IDM CA からの新しい証明書の要求

Certificate Issuance and Renewal システムロールでは、統合認証局 (CA) で IdM サーバーを使用しているときに、**ansible-core** を使用して証明書を発行できます。したがって、IdM を CA として使用する場合に、複数のシステムの証明書トラストチェーンを効率的かつ一貫して管理できます。

このプロセスは、**certmonger** プロバイダーを使用し、**getcert** コマンドで証明書を要求します。

注記

デフォルトでは、**certmonger** は有効期限が切れる前に証明書の更新を自動的に試行します。これは、Ansible Playbook の **auto_renew** パラメーターを **no** に設定すると無効にできます。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。RHEL システムロールと、その適用方法の詳細は、Getting started with RHEL System Roles を参照してください。

手順

1. **オプション:inventory.file** などのインベントリーファイルを作成します。

```
$ touch inventory.file
```

2. インベントリーファイルを開き、証明書を要求するホストを定義します。以下に例を示します。

```
[webserver]
server.idm.example.com
```

3. Playbook ファイルを作成します (例: **request-certificate.yml**)。
   - **webserver** など、証明書を要求するホストを含むように **hosts** を設定します。
   - **certificate_requests** 変数を設定して以下を追加します。
     - **name** パラメーターを希望する証明書の名前 (**mycert** など) に設定します。
     - **dns** パラメーターをドメインに設定し、証明書に追加します (例: **www.example.com**)。
     - **principal** パラメーターを設定し、Kerberos プリンシパルを指定します (例: **HTTP/www.example.com@EXAMPLE.COM**)。
     - **ca** パラメーターを **ipa** に設定します。
   - **roles** の下に **rhel-system-roles.certificate** ロールを設定します。

以下は、この例の Playbook ファイルです。

```
---
- 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. ファイルを保存します。
5. Playbook を実行します。

```
$ ansible-playbook -i inventory.file request-certificate.yml
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** ファイルを参照してください。
- man ページの **ansible-playbook(1)** を参照してください。

15.4. CERTIFICATE ISSUANCE AND RENEWAL システムロールを使用して証明書の発行前後に実行するコマンドを指定

Certificate Issuance and Renewal システムロールでは、Ansible Core を使用して、証明書の発行または更新の前後にコマンドを実行できます。

以下の例では、管理者が **www.example.com** の自己署名証明書を発行または更新する前に **httpd** サービスを停止し、後で再起動します。

注記

デフォルトでは、**certmonger** は有効期限が切れる前に証明書の更新を自動的に試行します。これは、Ansible Playbook の **auto_renew** パラメーターを **no** に設定すると無効にできます。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。RHEL システムロールと、その適用方法の詳細は、Getting started with RHEL System Roles を参照してください。

手順

1. **オプション:inventory.file** などのインベントリーファイルを作成します。

-

```
$ touch inventory.file
```

2. インベントリーファイルを開き、証明書を要求するホストを定義します。以下に例を示します。

```
[webserver]
server.idm.example.com
```

3. Playbook ファイルを作成します (例: **request-certificate.yml**)。

- **webserver** など、証明書を要求するホストを含むように **hosts** を設定します。
- **certificate_requests** 変数を設定して以下を追加します。
  - **name** パラメーターを希望する証明書の名前 (**mycert** など) に設定します。
  - **dns** パラメーターをドメインに設定し、証明書に追加します (例: **www.example.com**)。
  - **ca** パラメーターを証明書を発行する際に使用する CA に設定します (例: **self-sign**)。
  - この証明書を発行または更新する前に、**run_before** パラメーターを実行するコマンドに設定します (例: **systemctl stop httpd.service**)。
  - この証明書を発行または更新した後に、**run_after** パラメーターを実行するコマンドに設定します (例: **systemctl start httpd.service**)。
- **roles** の下に **rhel-system-roles.certificate** ロールを設定します。
  以下は、この例の Playbook ファイルです。

```
---
- 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. ファイルを保存します。

5. Playbook を実行します。

```
$ ansible-playbook -i inventory.file request-certificate.yml
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** ファイルを参照してください。
- man ページの **ansible-playbook(1)** を参照してください。

第16章 RHEL システムロールを使用した KDUMP の設定

Ansible を使用して kdump を管理するには、RHEL 8 で利用可能な RHEL システムロールの 1 つである Kernel Dumps ロールを使用できます。

Kernel Dumps を使用すると、システムのメモリー内容を保存する場所を指定して後で分析できます。

RHEL システムロールと、その適用方法の詳細は、RHEL システムロールの概要 を参照してください。

16.1. KERNEL DUMPS RHEL システムロール

Kernel Dumps システムロールを使用すると、複数のシステムに基本的なカーネルダンプパラメーターを設定できます。

16.2. KERNEL DUMPS のロールパラメーター

Kernel Dumps RHEL システムロールに使用されるパラメーターは次のとおりです。

ロール変数	説明
kdump_path	**vmcore** が書き込まれるパス。**kdump_target** が null ではない場合、パスはそのダンプターゲットとの相対パスになります。そうでない場合は、root ファイルシステムの絶対パスである必要があります。

関連情報

- makedumpfile(8) の man ページ。
- **kdump** で使用されるパラメーターの詳細と Kernel Dumps システムロールの追加情報については、**/usr/share/ansible/roles/rhel-system-roles.tlog/README.md** を参照してください。

16.3. RHEL システムロールを使用した KDUMP の設定

Ansible Playbook を実行して Kernel Dumps システムロールを使用し、複数のシステムに基本的なカーネルダンプパラメーターを設定できます。

警告

Kernel Dumps ロールは、**/etc/kdump.conf** ファイルを置き換えて、管理対象ホストの kdump 設定をすべて置き換えます。また、Kernel Dumps ロールが適用されると、**/etc/sysconfig/kdump** ファイルを置き換えることにより、ロール変数で指定されていない場合も以前の kdump 設定がすべて置き換えられます。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。

- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。
- **kdump** をデプロイするシステムを一覧表示するインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- hosts: kdump-test
  vars:
    kdump_path: /var/crash
  roles:
    - rhel-system-roles.kdump
```

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

関連情報

- Kernel Dumps ロール変数の詳細は、/usr/share/doc/rhel-system-roles/kdump ディレクトリーの README.md ファイルまたは README.html ファイルを参照してください。
- Applying a system role を参照してください。
- **rhel-system-roles** パッケージをインストールし、**/usr/share/ansible/roles/rhel-system-roles.kdump/README.html** のドキュメントを参照してください。

第17章 RHEL システムロールを使用したローカルストレージの管理

Ansible を使用して LVM とローカルファイルシステム (FS) を管理するには、RHEL 9 で利用可能な RHEL システムロールの 1 つである Storage ロールを使用できます。

Storage ロールを使用すると、ディスク上のファイルシステム、複数のマシンにある論理ボリューム、および RHEL 7.7 以降の全バージョンでのファイルシステムの管理を自動化できます。

RHEL システムロールと、その適用方法の詳細は、RHEL システムロールの概要 を参照してください。

17.1. STORAGE ロールの概要

Storage ロールは以下を管理できます。

- パーティションが分割されていないディスクのファイルシステム
- 論理ボリュームとファイルシステムを含む完全な LVM ボリュームグループ

Storage ロールを使用すると、次のタスクを実行できます。

- ファイルシステムを作成する
- ファイルシステムを削除する
- ファイルシステムをマウントする
- ファイルシステムをアンマウントする
- LVM ボリュームグループを作成する
- LVM ボリュームグループを削除する
- 論理ボリュームを作成する
- 論理ボリュームを削除する
- RAID ボリュームを作成する
- RAID ボリュームを削除する
- RAID を使用して LVM プールを作成する
- RAID を使用して LVM プールを削除する

17.2. STORAGE システムロールでストレージデバイスを識別するパラメーター

Storage ロールの設定は、以下の変数に記載されているファイルシステム、ボリューム、およびプールにのみ影響します。

storage_volumes

管理対象のパーティションが分割されていない全ディスク上のファイルシステムの一覧
現在、パーティションはサポートされていません。

storage_pools

管理するプールの一覧
現在、サポートされている唯一のプールタイプは LVM です。LVM では、プールはボリュームグループ (VG) を表します。各プールの下には、ロールで管理されるボリュームの一覧があります。
LVM では、各ボリュームは、ファイルシステムを持つ論理ボリューム (LV) に対応します。

17.3. ブロックデバイスに XFS ファイルシステムを作成する ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook では、Storage ロールを適用し、デフォルトパラメーターを使用してブロックデバイスに XFS ファイルシステムを作成します。

警告

Storage ロールは、パーティションが分割されていないディスク全体または論理ボリューム (LV) でのみファイルシステムを作成できます。パーティションにファイルシステムを作成することはできません。

例17.1 /dev/sdb に XFS を作成する Playbook

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
  roles:
    - rhel-system-roles.storage
```

- 現在、ボリューム名 (この例では **barefs**) は任意です。Storage ロールは、**disks:** 属性に一覧表示されているディスクデバイスでボリュームを特定します。
- XFS は RHEL 9 のデフォルトファイルシステムであるため、**fs_type: xfs** 行を省略することができます。
- 論理ボリュームにファイルシステムを作成するには、エンクロージングボリュームグループを含む **disks:** 属性の下に LVM 設定を指定します。
  LV デバイスへのパスを指定しないでください。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.4. ファイルシステムを永続的にマウントする ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、Storage ロールをすぐに適用して、XFS ファイルシステムを永続的にマウントします。

例17.2 /dev/sdb のファイルシステムを /mnt/data にマウントする Playbook

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- この Playbook では、ファイルシステムが **/etc/fstab** ファイルに追加され、すぐにファイルシステムをマウントします。
- **/dev/sdb** デバイス上のファイルシステム、またはマウントポイントのディレクトリーが存在しない場合は、Playbook により作成されます。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.5. 論理ボリュームを管理する ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、Storage ロールを適用して、ボリュームグループに LVM 論理ボリュームを作成します。

例17.3 myvg ボリュームグループに mylv 論理ボリュームを作成する Playbook

```
- hosts: all
  vars:
    storage_pools:
      - name: myvg
        disks:
          - sda
          - sdb
          - sdc
        volumes:
          - name: mylv
            size: 2G
            fs_type: ext4
            mount_point: /mnt
  roles:
    - rhel-system-roles.storage
```

- **myvg** ボリュームグループは、次のディスクで構成されます。
  - **/dev/sda**
  - **/dev/sdb**
  - **/dev/sdc**
- **myvg** ボリュームグループがすでに存在する場合は、Playbook により論理ボリュームがボリュームグループに追加されます。
- **myvg** ボリュームグループが存在しない場合は、Playbook により作成されます。
- Playbook は、**mylv** 論理ボリューム上に Ext4 ファイルシステムを作成し、**/mnt** ファイルシステムを永続的にマウントします。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.6. オンラインのブロック破棄を有効にする ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook では、Storage ロールを適用して、オンラインのブロック破棄を有効にして XFS ファイルシステムをマウントします。

例17.4 /mnt/data/ でのオンラインのブロック破棄を有効にする Playbook

```
---
- 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
```

関連情報

- ファイルシステムを永続的にマウントする Ansible Playbook の例
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.7. EXT4 ファイルシステムを作成してマウントする ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、Storage ロールを適用して、Ext4 ファイルシステムを作成してマウントします。

例17.5 /dev/sdb に Ext4 を作成し、/mnt/data にマウントする Playbook

```
---
- 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
```

- Playbook は、**/dev/sdb** ディスクにファイルシステムを作成します。
- Playbook は、**/mnt/data** ディレクトリーにファイルシステムを永続的にマウントします。
- ファイルシステムのラベルは **label-name** です。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.8. EXT3 ファイルシステムを作成してマウントする ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、Storage ロールを適用して、Ext3 ファイルシステムを作成してマウントします。

例17.6 /dev/sdb に Ext3 を作成し、/mnt/data にマウントする Playbook

```
---
- 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
```

- Playbook は、**/dev/sdb** ディスクにファイルシステムを作成します。

- Playbook は、**/mnt/data** ディレクトリーにファイルシステムを永続的にマウントします。
- ファイルシステムのラベルは **label-name** です。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.9. STORAGE RHEL システムロールを使用し、既存の EXT4 または EXT3 ファイルシステムのサイズを変更する ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、Storage ロールを適用して、ブロックデバイスにある既存の Ext4 または Ext3 ファイルシステムのサイズを変更します。

例17.7 ディスクに単一ボリュームを設定する Playbook

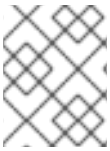
```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
 size: 12 GiB
        fs_type: ext4
        mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

- 直前の例のボリュームがすでに存在する場合に、ボリュームサイズを変更するには、異なるパラメーター **size** の値で、同じ Playbook を実行する必要があります。以下に例を示します。

例17.8 /dev/sdbで ext4 のサイズを変更する Playbook

```
---
- 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
```

- 現在、ボリューム名 (この例では barefs) は任意です。Storage ロールは、disks: 属性に一覧表示されているディスクデバイスでボリュームを特定します。



注記

他のファイルシステムで **Resizing** アクションを使用すると、作業しているデバイスのデータを破棄する可能性があります。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.10. STORAGE RHEL システムロールを使用し、LVM 上の既存のファイルシステムのサイズを変更する ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、Storage RHEL システムロールを適用して、ファイルシステムで LVM 論理ボリュームのサイズを変更します。

警告

他のファイルシステムで **Resizing** アクションを使用すると、作業しているデバイスのデータを破棄する可能性があります。

例17.9 myvg ボリュームグループの既存の mylv1 および myvl2 論理ボリュームのサイズを変更する Playbook

```
---

- 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
  incude_role:
    name: rhel-system-roles.storage
```

- この Playbook は、以下の既存のファイルシステムのサイズを変更します。
  - **/opt/mount1** にマウントされる **mylv1** ボリュームの Ext4 ファイルシステムは、そのサイズを 10 GiB に変更します。
  - **/opt/mount2** にマウントされる **mylv2** ボリュームの Ext4 ファイルシステムは、そのサイズを 50 GiB に変更します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.11. STORAGE RHEL システムロールを使用し、SWAP パーティションを作成する ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は Storage ロールを適用し、デフォルトパラメーターを使用してブロックデバイスにスワップパーティションを作成するか (存在しない場合)、または swap パーティションを変更します (すでに存在する場合)。

例17.10 /dev/sdb で既存の XFS を作成または変更する Playbook

```
---
- 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
```

- 現在、ボリューム名 (この例では **swap_fs**) は任意です。Storage ロールは、**disks:** 属性に一覧表示されているディスクデバイスでボリュームを特定します。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.12. STORAGE システムロールを使用した RAID ボリュームの設定

Storage システムロールを使用すると、Red Hat Ansible Automation Platform を使用して RHEL に RAID ボリュームを設定できます。本セクションでは、要件に合わせて RAID ボリュームを設定するために、利用可能なパラメーターを使用して Ansible Playbook を設定する方法を説明します。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。
- Storage システムロールを使用して、RAID ボリュームをデプロイするシステムの詳細を記録したインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
- hosts: all
  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
  roles:
    - name: rhel-system-roles.storage
```

警告

特定の状況でデバイス名が変更する場合があります。たとえば、新しいディスクをシステムに追加するときなどです。したがって、データの損失を防ぐために、Playbook で特定のディスク名を使用することは推奨していません。

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.13. STORAGE システムロールを使用した LVM POOL WITH RAID の設定

Storage システムロールを使用すると、Red Hat Ansible Automation Platform を使用して RHEL に LVM pool with RAID を設定できます。本セクションでは、利用可能なパラメーターを使用して Ansible Playbook を設定し、LVM pool with RAID を設定する方法を説明します。

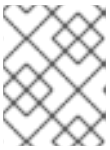
前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。
- Storage システムロールを使用して、LVM pool with RAID を設定するシステムの詳細を記録したインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
- 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
```



注記

LVM pool with RAID を作成するには、**raid_level** パラメーターを使用して RAID タイプを指定する必要があります。

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル。

17.14. STORAGE RHEL システムロールを使用し、LVM 上の VDO ボリュームを圧縮および重複排除する ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook では、Storage RHEL システムロールを適用し、VDO (Virtual Data Optimizer) ボリュームを使用して、論理マネージャーボリューム (LVM) への圧縮および重複排除を有効にします。

例17.11 myvg ボリュームグループに、mylv1 LVM VDO ボリュームを作成する Playbook

```
---
- 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
```

この例では、**compression** プールおよび **deduplication** プールを true に設定します。これは、VDO が使用されることを指定します。以下では、このパラメーターの使用方法を説明します。

- **deduplication** は、ストレージボリュームに保存されている重複データの重複排除に使用されます。
- 圧縮は、ストレージボリュームに保存されているデータを圧縮するために使用されます。これにより、より大きなストレージ容量が得られます。
- vdo_pool_size は、ボリュームがデバイスで使用する実際のサイズを指定します。VDO ボリュームの仮想サイズは、**size** パラメーターで設定します。注記:LVM VDO は Storage ロールで使用されるため、プールごとに圧縮と重複排除を使用できるボリュームは 1 つだけです。

17.15. STORAGE システムロールを使用した LUKS 暗号化ボリュームの作成

Storage ロールを使用し、Ansible Playbook を実行して、LUKS で暗号化されたボリュームを作成および設定できます。

前提条件

- Crypto Policies システムロールを設定する **管理対象ノード** 1 つ以上へのアクセスおよびパーミッション。
- コントロールノード (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、

- **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
- 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
```

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル

17.16. STORAGE RHEL システムロールを使用し、プールのボリュームサイズをパーセンテージで表すANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook では、Storage システムロールを適用して、論理マネージャーボリューム (LVM) ボリュームのサイズをプールの合計サイズのパーセンテージで表現できるようにします。

例17.12 ボリュームのサイズをプールの合計サイズのパーセンテージで表現する Playbook

```
---
- 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
```

この例では、LVM ボリュームのサイズを、プールサイズのパーセンテージで指定します (例: "60%")。また、LVM ボリュームのサイズを、人間が判読できるファイルシステムのサイズ ("10g" や "50 GiB" など) で、プールサイズのパーセンテージで指定することもできます。

17.17. 関連情報

- **/usr/share/doc/rhel-system-roles/storage/**
- **/usr/share/ansible/roles/rhel-system-roles.storage/**

第18章 RHEL システムロールを使用した時刻同期の設定

Time Synchronization RHEL システムロールを使用すると、Red Hat Ansible Automation Platform を使用して RHEL の複数のターゲットマシンで時刻同期を管理できます。

18.1. TIME SYNCHRONIZATION システムロール

Time Synchronization RHEL システムロールを使用して、複数のターゲットマシンで時刻同期を管理できます。

システムクロックが NTP サーバーまたは PTP ドメインのグランドマスターに同期するように、Time Synchronization ロールが NTP 実装または PTP 実装をインストールし、NTP クライアントまたは PTP レプリカとして動作するように設定します。

Time Synchronization ロールを使用すると、NTP プロトコルの実装にシステムが **ntp** または **chrony** を使用するかどうかにかかわらず、RHEL 6 以降のすべてのバージョンの Red Hat Enterprise Linux で同じ Playbook を使用できるため、chrony への移行 が容易になります。

18.2. サーバーの単一プールに対する TIME SYNCHRONIZATION システムロールの適用

以下の例は、サーバーにプールが 1 つしかない場合に、Time Synchronization ロールを適用する方法を示しています。

警告

Time Synchronization ロールは、管理対象ホストで指定または検出されたプロバイダーサービスの設定を置き換えます。以前の設定は、ロール変数で指定されていなくても失われます。**timesync_ntp_provider** 変数が定義されていない場合は、プロバイダーの唯一の設定が適用されます。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。
- Time Synchronization システムロールをデプロイするシステムを一覧表示するインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: 2.rhel.pool.ntp.org
```

```
      pool: yes
      iburst: yes
  roles:
    - rhel-system-roles.timesync
```

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

18.3. クライアント・サーバーでの TIME SYNCHRONIZATION システムロールの適用

Time Synchronization ロールを使用すると、NTP クライアントで Network Time Security (NTS) を有効にできます。Network Time Security (NTS) は、Network Time Protocol (NTP) で指定される認証メカニズムです。サーバーとクライアント間で交換される NTP パケットが変更されていないことを確認します。

警告

Time Synchronization ロールは、管理対象ホストで指定または検出されたプロバイダーサービスの設定を置き換えます。以前の設定は、ロール変数で指定されていなくても失われます。**timesync_ntp_provider** 変数が定義されていない場合は、プロバイダーの唯一の設定が適用されます。

前提条件

- **timesync** ソリューションをデプロイするシステムに Red Hat Ansible Automation Platform をインストールする必要はありません。
- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。
- Time Synchronization システムロールをデプロイするシステムを一覧表示するインベントリーファイルがある。
- **chrony** の NTP プロバイダーバージョンは 4.0 以降。

手順

1. 以下の内容を含む **playbook.yml** ファイルを作成します。

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: ptbtime1.ptb.de
```

```
      iburst: yes
      nts: yes
  roles:
    - rhel-system-roles.timesync
```

ptbtime1.ptb.de は、公開サーバーの例です。別のパブリックサーバーまたは独自のサーバーを使用できます。

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

検証

1. クライアントマシンでテストを実行します。

```
# 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. 報告された cookie の数がゼロよりも多いことを確認します。

関連情報

- **chrony.conf(5)** の man ページ

18.4. TIME SYNCHRONIZATION システムロール変数

以下の変数を Time Synchronization ロールに渡すことができます。

- **timesync_ntp_servers**:

ロール変数の設定	説明
hostname: host.example.com	サーバーのホスト名またはアドレス。
minpoll: **number**	最小ポーリング間隔。デフォルト:6
maxpoll: **number**	最大ポーリングの間隔。デフォルト:10
iburst: yes	高速な初期同期を有効にするフラグ。デフォルト: no
pool: yes	ホスト名の解決された各アドレスが別の NTP サーバーであることを示すフラグ。デフォルト: no

ロール変数の設定	説明
nts: yes	Network Time Security (NTS) を有効にするフラグ。デフォルト: no. Supported only with chrony >= 4.0.

関連情報

- Time Synchronization ロール変数の詳細は、rhel-system-roles パッケージをインストールし、**/usr/share/doc/rhel-system-roles/timesync** ディレクトリーの README.md または README.html ファイルを参照してください。

第19章 RHEL システムロールを使用したパフォーマンスの監視

システム管理者は、Metrics RHEL システムロールを使用して、システムのパフォーマンスを監視できます。

19.1. METRICS システムロールの概要

RHEL システムロールは、複数の RHEL システムをリモートで管理する一貫した構成インターフェースを提供する Ansible ロールおよびモジュールの集合です。Metrics システムロールはローカルシステムのパフォーマンス分析サービスを設定します。これには、オプションでローカルシステムによって監視されるリモートシステムの一覧が含まれます。Metrics システムロールを使用すると、**pcp** の設定とデプロイメントが Playbook によって処理されるため、**pcp** を個別に設定せずに、**pcp** を使用してシステムパフォーマンスを監視できます。

表19.1 メトリックシステムロール変数

ロール変数	説明	使用例
metrics_monitored_hosts	ターゲットホストが分析するリモートホストの一覧。これらのホストにはターゲットホストにメトリックが記録されるため、各ホストの **/var/log** の下に十分なディスク領域があることを確認してください。	**metrics_monitored_hosts: ["webserver.example.com", "database.example.com"]**
metrics_retention_days	削除前のパフォーマンスデータの保持日数を設定します。	**metrics_retention_days:14**
metrics_graph_service	**pcp** および **grafana** を介してパフォーマンスデータの視覚化のためにホストをサービスで設定できるようにするブール値フラグ。デフォルトでは false に設定されます。	**metrics_graph_service: no**
metrics_query_service	**redis** 経由で記録された **pcp** メトリックをクエリーするための時系列クエリーサービスでのホストの設定を可能にするブール値フラグ。デフォルトでは false に設定されます。	**metrics_query_service: no**
metrics_provider	メトリックを提供するために使用するメトリックコレクターを指定します。現在、サポートされている唯一のメトリックプロバイダーは **pcp** です。	**metrics_provider: "pcp"**

注記

metrics_connections で使用されるパラメーターの詳細と、Metrics システムロールに関する追加情報は、**/usr/share/ansible/roles/rhel-system-roles.metrics/README.md** ファイルを参照してください。

19.2. METRICS システムロールを使用した視覚化によるローカルシステムの監視

この手順では、Metrics RHEL システムロール を使用してローカルシステムを監視しながら、**Grafana** でデータ視覚化をプロビジョニングする方法を説明します。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- 監視するマシンに **rhel-system-roles** パッケージがインストールされている。

手順

1. 以下のコンテンツをインベントリーに追加して、**/etc/ansible/hosts** Ansible インベントリーの **localhost** を設定します。

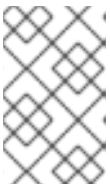
```
localhost ansible_connection=local
```

2. 以下の内容を含む Ansible Playbook を作成します。

```
---
- hosts: localhost
  vars:
    metrics_graph_service: yes
  roles:
    - rhel-system-roles.metrics
```

3. Ansible Playbook の実行:

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



注記

metrics_graph_service のブール値が value="yes" に設定されているため、**Grafana** は自動的にインストールされ、データソースとして追加された**pcp**でプロビジョニングされます。

4. マシンで収集されるメトリックを視覚化するには、「Grafana Web UI へのアクセス」 の説明どおりに **grafana** Web インターフェースにアクセスします。

19.3. METRICS システムロールを使用した自己監視のための個別システムフリートの設定

この手順では、Metrics システムロールを使用して、それ自体を監視するマシンフリートの設定方法を説明します。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook の実行に使用するマシンに **rhel-system-roles** パッケージがインストールされている。
- SSH 接続が確立している。

手順

1. Playbook 経由で監視するマシンの名前または IP を、括弧で囲まれた識別グループ名で **/etc/ansible/hosts** Ansible インベントリーファイルに追加します。

```
[remotes]
webserver.example.com
database.example.com
```

2. 以下の内容を含む Ansible Playbook を作成します。

```
---
- hosts: remotes
  vars:
    metrics_retention_days: 0
  roles:
    - rhel-system-roles.metrics
```

3. Ansible Playbook の実行:

```
# ansible-playbook name_of_your_playbook.yml -k
```

リモートシステムに接続するためのパスワードを求められる**-k** です。

19.4. METRICS システムロールを使用したローカルマシン経由でのマシンフリートの一元監視

この手順では、**grafana** を介したデータの視覚化のプロビジョニングおよび **redis** 経由でのデータのクエリーをしながら、Metrics システムロールを使用して、マシンフリートを一元管理するローカルマシンの設定方法を説明します。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook の実行に使用するマシンに **rhel-system-roles** パッケージがインストールされている。

手順

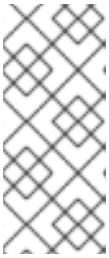
1. 以下の内容を含む Ansible Playbook を作成します。

```
---
- hosts: localhost
  vars:
```

```
    metrics_graph_service: yes
    metrics_query_service: yes
    metrics_retention_days: 10
    metrics_monitored_hosts: ["database.example.com", "webserver.example.com"]
  roles:
    - rhel-system-roles.metrics
```

2. Ansible Playbook の実行:

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



注記

metrics_graph_service および **metrics_query_service** のブール値は value="yes" に設定されているため、**grafana** は、**redis** にインデックス化された **pcp** データの記録のあるデータソースとして追加された **pcp** で自動的にインストールおよびプロビジョニングされます。これにより、**pcp** クエリー言語をデータの複雑なクエリーに使用できます。

3. マシンによって一元的に収集されるメトリックのグラフィック表示とデータのクエリーを行うには、「Grafana Web UI へのアクセス」 で説明されているように、**grafana** Web インターフェースにアクセスします。

19.5. METRICS システムロールを使用したシステム監視中の認証設定

PCP は、Simple Authentication Security Layer (SASL) フレームワークを介して **scram-sha-256** 認証メカニズムに対応します。Metrics RHEL システムロールは、**scram-sha-256** 認証メカニズムを使用して認証を設定する手順を自動化します。この手順では、Metrics RHEL システムロールを使用して、認証を設定する方法を説明します。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- Playbook の実行に使用するマシンに **rhel-system-roles** パッケージがインストールされている。

手順

1. 認証を設定する Ansible Playbook に、以下の変数を追加します。

```
---
  vars:
    metrics_username: your_username
    metrics_password: your_password
```

2. Ansible Playbook の実行:

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

検証手順

- **sasl** 設定を確認します。

```
# pminfo -f -h "pcp://ip_adress?username=your_username" disk.dev.read
Password:
disk.dev.read
inst [0 or "sda"] value 19540
```

ip_adress は、ホストの IP アドレスに置き換える必要があります。

19.6. METRICS システムロールを使用した SQL サーバーのメトリクスコレクションの設定と有効化

この手順では、Metrics RHEL システムロールを使用して、ローカルシステムの **pcp** を使用して Microsoft SQL Server のメトリック収集の設定と有効化を自動化する方法を説明します。

前提条件

- Ansible Core パッケージがコントロールマシンにインストールされている。
- 監視するマシンに **rhel-system-roles** パッケージがインストールされている。
- Red Hat Enterprise Linux に Microsoft SQL Server をインストールし、SQL Server への「信頼できる」接続を確立している。「Installing SQL Server」および「create a database on Red Hat 」を参照してください。
- Red Hat Enterprise Linux 用の SQL Server の Microsoft ODBC ドライバーがインストールされている。Red Hat Enterprise Server および Oracle Linux を参照してください。

手順

1. 以下のコンテンツをインベントリーに追加して、**/etc/ansible/hosts** Ansible インベントリーの **localhost** を設定します。

```
localhost ansible_connection=local
```

2. 以下の内容が含まれる Ansible Playbook を作成します。

```
---
- hosts: localhost
  roles:
    - role: rhel-system-roles.metrics
      vars:
        metrics_from_mssql: yes
```

3. Ansible Playbook の実行:

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

検証手順

- **pcp** コマンドを使用して、SQL Server PMDA エージェント (mssql) が読み込まれ、実行されていることを確認します。

```
# 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
```

関連情報

- Microsoft SQL Server での Performance Co-Pilot の使用に関する詳細は、Red Hat Developers Blog を参照してください。

第20章 MICROSOFT SQL SERVER ANSIBLE ロールを使用した MICROSOFT SQL SERVER の設定

管理者は、Microsoft SQL Server Ansible ロールを使用して、Microsoft SQL Server (SQL Server) をインストール、設定、および起動できます。Microsoft SQL Server Ansible ロールは、オペレーティングシステムを最適化して、SQL Server のパフォーマンスとスループットを向上させます。このロールは、SQL Server ワークロードを実行するために推奨される設定を使用し、RHEL ホストの設定を簡素化して自動化します。

20.1. 前提条件

- 2GB の RAM
- SQL Server を設定する管理対象ノードへの **root** アクセス
- 事前設定済みファイアウォール
  mssql_tcp_port 変数で設定した SQL Server の TCP ポートで接続を有効にする必要があります。この変数を定義しないと、ロールのデフォルトは TCP ポート番号 **1443** になります。

  新しいポートを追加するには、次のコマンドを実行します。

  ```
  # firewall-cmd --add-port=xxxx/tcp --permanent
  # firewall-cmd --reload
  ```

  xxx を TCP ポート番号に置き換え、ファイアウォールルールを再読み込みします。
- **オプション**:SQL ステートメントと、それを SQL Server に入力するプロシージャーを含む、**.sql** 拡張子を持つファイルを作成します。

20.2. MICROSOFT SQL SERVER ANSIBLE ロールのインストール

Microsoft SQL Server Ansible ロールは、**ansible-collection-microsoft-sql** パッケージに含まれています。

前提条件

- **root** アクセス

手順

1. RHEL 8 AppStream リポジトリーで利用可能な Ansible Core をインストールします。

   ```
   # dnf install ansible-core
   ```

2. Microsoft SQL Server Ansible ロールをインストールします。

   ```
   # dnf install ansible-collection-microsoft-sql
   ```

20.3. MICROSOFT SQL SERVER ANSIBLE ロールを使用した SQL サーバーのインストールと設定

Microsoft SQL Server Ansible ロールを使用して、SQL サーバーをインストールおよび設定できます。

前提条件

- Ansible インベントリーが作成されます。

手順

1. 拡張子が **.yml** のファイルを作成します。例: **mssql-server.yml**
2. 以下の内容を **.yml** に追加します。

```
---
- hosts: all
  vars:
    mssql_accept_microsoft_odbc_driver_17_for_sql_server_eula: true
    mssql_accept_microsoft_cli_utilities_for_sql_server_eula: true
    mssql_accept_microsoft_sql_server_standard_eula: true
    mssql_password: <password>
    mssql_edition: Developer
    mssql_tcp_port: 1443
  roles:
    - microsoft.sql.server
```

<password> を、使用している SQL Server のパスワードに置き換えます。

3. **mssql-server.yml** Ansible Playbook を実行します。

```
# ansible-playbook mssql-server.yml
```

20.4. TLS 変数

以下の変数は、TLS (Transport Level Security) の設定に使用できます。

表20.1 TLS ロール変数

ロール変数	説明

ロール変数	説明
mssql_tls_enable	この変数は、TLS 暗号化を有効または無効にします。 Microsoft SQL Server Ansible ロールは、変数が **true** に設定されていると、以下のタスクを実行します。 ● SQL Server の **/etc/pki/tls/certs/** に TLS 証明書をコピーします。 ● 秘密鍵を SQL Server 上の **/etc/pki/tls/private/** にコピーします。 ● TLS 証明書および秘密鍵を使用して接続を暗号化するように SQL Server を設定します。  **注記** Ansible コントロールノードに TLS 証明書と秘密鍵が必要です。 **false** に設定すると、TLS 暗号化が無効になります。このロールは、既存の証明書および秘密鍵ファイルを削除しません。
mssql_tls_cert	この変数を定義するには、TLS 証明書ファイルのパスを入力します。
mssql_tls_private_key	この変数を定義するには、秘密鍵ファイルのパスを入力します。
mssql_tls_version	この変数を定義して、使用する TSL バージョンを選択します。 既定値は **1.2** です。
mssql_tls_force	この変数を **true** に設定すると、ホストコンピューターの証明書および秘密鍵のファイルが置き換えられます。ファイルは、**/etc/pki/tls/certs/** ディレクトリーおよび **/etc/pki/tls/private/** ディレクトリーの下に存在している必要があります。 デフォルトは **false** です。

20.5. MLSERVICES の EULA への同意

必要な SQL Server Machine Learning Services (MLServices) をインストールするには、Python パッケージおよび R パッケージのオープンソースディストリビューション用のすべての EULA に同意する必要があります。

使用許諾条件は、**/usr/share/doc/mssql-server** を参照してください。

表20.2 SQL Server Machine Learning Services の EULA 変数

ロール変数	説明
mssql_accept_microsoft_sql_server_standard_eula	この変数は、**mssql-conf** パッケージのインストールに関する条件に同意するかどうかを決定します。 条件に同意する場合は、この変数を **true** に設定します。 デフォルトは **false** です。

20.6. MICROSOFT ODBC 17 向け EULA への同意

すべての EULA に同意し、Microsoft Open Database Connectivity (ODBC) ドライバーをインストールする必要があります。

使用許諾条件については、**/usr/share/doc/msodbcsql17/LICENSE.txt** および **/usr/share/doc/mssql-tools/LICENSE.txt** を参照してください。

表20.3 Microsoft ODBC 17 EULA 変数

ロール変数	説明
mssql_accept_microsoft_odbc_driver_17_for_sql_server_eula	この変数は、**msodbcsql17** パッケージのインストールに関する条件に同意するかどうかを決定します。 条件に同意する場合は、この変数を **true** に設定します。 デフォルトは **false** です。
mssql_accept_microsoft_cli_utilities_for_sql_server_eula	この変数は、**mssql-tools** パッケージのインストールに関する条件に同意するかどうかを決定します。 条件に同意する場合は、この変数を **true** に設定します。 デフォルトは **false** です。

第21章 TERMINAL SESSION RECORDING RHEL システムロールを使用したセッション記録用システムの設定

Terminal Session Recording RHEL システムロールを使用すると、Red Hat Ansible Automation Platform を使用して RHEL で端末セッションを録画するようにシステムを設定できます。

21.1. TERMINAL SESSION RECORDING システムロール

Terminal Session Recording RHEL システムロールを使用して、RHEL での端末セッションの録画用に RHEL システムを設定できます。

SSSD サービスを使用して、ユーザーごと、またはユーザーグループごとに録画を行うように設定できます。

関連情報

- RHEL でのセッションの録画に関する詳細は、Recording Sessions を参照してください。

21.2. TERMINAL SESSION RECORDING システムロールのコンポーネントとパラメーター

セッションの録画ソリューションには、以下のコンポーネントがあります。

- **tlog** ユーティリティー
- System Security Services Daemon (SSSD)
- オプション:Web コンソールインターフェース

Terminal Session Recording RHEL システムロールに使用されるパラメーターは次のとおりです。

ロール変数	説明
tlog_use_sssd (default: yes)	SSSD を使用してセッションの録画を設定します (録画したユーザーまたはグループの管理方法として推奨)。
tlog_scope_sssd (default: none)	SSSD 録画スコープの設定: all / some / none
tlog_users_sssd (default: [])	録画するユーザーの YAML リスト
tlog_groups_sssd (default: [])	録画するグループの YAML リスト

- **tlog** で使用されるパラメーターの詳細と、Terminal Session Recording システムロールに関する追加情報は、**/usr/share/ansible/roles/rhel-system-roles.tlog/README.md** ファイルを参照してください。

21.3. TERMINAL SESSION RECORDING RHEL システムロールのデプロイ

以下の手順に従って、Ansible Playbook を準備および適用し、RHEL システムが systemd ジャーナルにセッションの録画データをログに記録するように設定します。

前提条件

- コントロールノードから Terminal Session Recording システムロールが設定されるターゲットシステムへアクセスするための SSH キーを設定している。
- Terminal Session Recording システムロールを設定するシステムが 1 つ以上ある。
- Ansible Core パッケージがコントロールマシンにインストールされている。
- **rhel-system-roles** パッケージがコントロールマシンにインストールされている。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

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

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

詳細は以下のようになります。

- **tlog_scope_sssd**:
  - **some** は、**all** または **none** ではなく、特定のユーザーおよびグループのみを録画することを指定します。
- **tlog_users_sssd**:
  - **recorded-user** は、セッションを録画するユーザーを指定します。ただし、ユーザーは追加されない点に留意してください。ユーザーを独自に設定する必要があります。

2. オプション: Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

これにより、Playbook は指定したシステムに **tlog** RHEL システムロールをインストールします。ロールには、ユーザーのログインシェルとして動作する端末セッション I/O ロギングプログラムである **tlog-rec-session** が含まれます。また、定義したユーザーおよびグループで使用できる SSSD 設定ドロップファイルを作成します。SSSD は、これらのユーザーおよびグループを解析して読み取り、ユーザーシェルを **tlog-rec-session** に置き換えます。さらに、**cockpit** パッケージがシステムにインストー

ルされている場合、Playbook は **cockpit-session-recording** パッケージもインストールします。これは、Web コンソールインターフェースで録画を表示および再生できるようにする **Cockpit** モジュールです。

検証手順

システムで SSSD 設定ドロップファイルが作成されることを確認するには、以下の手順を実行します。

1. SSSD 設定ドロップファイルが作成されるフォルダーに移動します。

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

2. ファイルの内容を確認します。

```
# cat /etc/sssd/conf.d/sssd-session-recording.conf
```

Playbook に設定したパラメーターがファイルに含まれていることが確認できます。

21.4. グループまたはユーザーの一覧を除外するための TERMINAL SESSION RECORDING RHEL システムロールのデプロイ

Terminal Session Recording システムロールを使用すると、SSSD セッションの録画設定オプション **exclude_users** および **exclude_groups** をサポートできます。以下の手順に従って、Ansible Playbook を準備および適用し、ユーザーまたはグループがセッションを録画して systemd ジャーナルにロギングしないように RHEL システムを設定します。

前提条件

- コントロールノードから Terminal Session Recording システムロールを設定するターゲットシステムへアクセスするための SSH キーを設定している。
- Terminal Session Recording システムのロールを設定するシステムが少なくとも 1 つある。
- Ansible Core パッケージがコントロールマシンにインストールされている。
- **rhel-system-roles** パッケージがコントロールマシンにインストールされている。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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
```

詳細は以下のようになります。

- **tlog_scope_sssd**:
  - **all**: ユーザーおよびグループをすべて録画するように指定します。
- **tlog_exclude_users_sssd**:
  - User name: セッションの録画から除外するユーザーのユーザー名を指定します。
- **tlog_exclude_groups_sssd**:
  - **admins** は、セッションの録画から除外するグループを指定します。

2. オプションで Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

これにより、Playbook は指定したシステムに **tlog** RHEL システムロールをインストールします。ロールには、ユーザーのログインシェルとして動作する端末セッション I/O ロギングプログラムである **tlog-rec-session** が含まれます。また、除外対象外のユーザーおよびグループが使用できる **/etc/sssd/conf.d/sssd-session-recording.conf** SSSD 設定ドロップファイルを作成します。SSSD は、これらのユーザーおよびグループを解析して読み取り、ユーザーシェルを **tlog-rec-session** に置き換えます。さらに、**cockpit** パッケージがシステムにインストールされている場合、Playbook は **cockpit-session-recording** パッケージもインストールします。これは、Web コンソールインターフェースで録画を表示および再生できるようにする **Cockpit** モジュールです。

検証手順

システムで SSSD 設定ドロップファイルが作成されることを確認するには、以下の手順を実行します。

1. SSSD 設定ドロップファイルが作成されるフォルダーに移動します。

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

2. ファイルの内容を確認します。

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

Playbook に設定したパラメーターがファイルに含まれていることが確認できます。

関連情報

- **/usr/share/doc/rhel-system-roles/tlog/** ディレクトリーおよび **/usr/share/ansible/roles/rhel-system-roles.tlog/** ディレクトリーを参照してください。
- Recording a session using the deployed Terminal Session Recording System Role in the CLI 。

21.5. CLI でデプロイされた TERMINAL SESSION RECORDING システムロールを使用したセッションの録画

指定したシステムにターミナルセッション録画システムロールをデプロイしたら、コマンドラインインターフェース(CLI)を使用してユーザー端末セッションを録画できます。

前提条件

- ターゲットシステムに Terminal Session Recording システムロールをデプロイしている。
- SSSD 設定ドロップファイルが **/etc/sssd/conf.d** ディレクトリーに作成されていること。Deploying the Terminal Session Recording RHEL System Role を参照してください。

手順

1. ユーザーを作成し、このユーザーにパスワードを割り当てます。

```
# useradd recorded-user
# passwd recorded-user
```

2. 作成したユーザーとしてシステムにログインします。

```
# ssh recorded-user@localhost
```

3. 認証用に yes または no を入力するようにシステムが求めたら、「yes」を入力します。
4. **recorded-user の** パスワードを挿入します。
システムは、録画しているセッションに関するメッセージを表示します。

```
ATTENTION! Your session is being recorded!
```

5. セッションの録画が完了したら、以下を入力します。

```
# exit
```

システムはユーザーからログアウトし、ローカルホストとの接続を閉じます。

これにより、ユーザーセッションは録画および保存され、ジャーナルを使用して再生することができます。

検証手順

ジャーナルで録画したセッションを表示するには、以下の手順を実施します。

1. 以下のコマンドを実行します。

```
# journalctl -o verbose -r
```

2. 録画したジャーナルエントリー **tlog-rec** の **MESSAGE** フィールドを検索します。

```
# journalctl -xel _EXE=/usr/bin/tlog-rec-session
```

21.6. CLI を使用した録画したセッションの表示

コマンドラインインターフェース (CLI) を使用して、ジャーナルからユーザーセッションの録画を再生できます。

前提条件

- ユーザーセッションを録画している。Recording a session using the deployed Terminal Session Recording System Role in the CLI を参照してください。

手順

1. CLI 端末で、ユーザーセッションの録画を再生します。

```
# journalctl -o verbose -r
```

2. **tlog** 録画を検索します。

```
$ /tlog-rec
```

以下のような詳細が表示されます。

- ユーザーセッションの録画用のユーザー名
- **out_txt** フィールド (録画したセッションの raw 出力エンコード)
- 識別子番号 TLOG_REC=**ID_number**

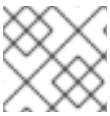
3. 識別子番号 TLOG_REC=**ID_number** をコピーします。
4. 識別子番号 TLOG_REC=**ID_number** を使用して録画を再生します。

```
# tlog-play -r journal -M TLOG_REC=ID_number
```

これにより、ユーザーセッションの録画端末の出力が再生されることがわかります。

第22章 システムロールを使用した高可用性クラスターの設定

HA Cluster システムロールを使用すると、Pacemaker の高可用性クラスターリソースマネージャーを使用する高可用性クラスターを設定し、管理できます。



注記

現在、HA システムロールは SBD をサポートしていません。

22.1. HA CLUSTER システムロール変数

HA Cluster システムロール Playbook では、クラスターデプロイメントの要件に従って、高可用性クラスターの変数を定義します。

HA Cluster システムロールに設定できる変数は以下のとおりです。

ha_cluster_enable_repos

HA Cluster システムロールで必要なパッケージを含むリポジトリーを有効にするブール値フラグ。これがこの変数のデフォルト値である **yes** に設定されている場合は、クラスターメンバーとして使用するシステムで RHEL および RHEL 高可用性アドオンのアクティブなサブスクリプションカバレッジが必要です。そうでない場合、システムロールは失敗します。

ha_cluster_cluster_present

yes に設定されている場合には、ロールに渡される変数に従って、HA クラスターがホストで設定されるブール値フラグを決定します。ロールで指定されておらず、ロールによってサポートされないクラスター設定は失われます。
ha_cluster_cluster_present を **no** に設定すると、すべての HA クラスター設定がターゲットホストから削除されます。

この変数のデフォルト値は **yes** です。

以下の Playbook の例では、**node1** および **node2** のすべてのクラスター設定を削除します。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_present: no

  roles:
    - rhel-system-roles.ha_cluster
```

ha_cluster_start_on_boot

起動時にクラスターサービスが起動するように設定されるかどうかを決定するブール値フラグ。この変数のデフォルト値は **yes** です。

ha_cluster_fence_agent_packages

インストールするフェンスエージェントパッケージの一覧この変数のデフォルト値は **fence-agents-all**, **fence-virt** です。

ha_cluster_extra_packages

インストールする追加パッケージの一覧この変数のデフォルト値はパッケージではありません。
この変数は、ロールによって自動的にインストールされていない追加パッケージをインストールするために使用できます (例: カスタムリソースエージェント)。

フェンスエージェントをこのリストのメンバーとして追加できます。ただし、**ha_cluster_fence_agent_packages** は、フェンスエージェントの指定に使用する推奨されるロール変数であるため、デフォルト値が上書きされます。

ha_cluster_hacluster_password

hacluster ユーザーのパスワードを指定する文字列の値。**hacluster** ユーザーには、クラスターへの完全アクセスがあります。「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、パスワードの暗号化を行うことが推奨されます。デフォルトのパスワード値がないため、この変数を指定する必要があります。

ha_cluster_corosync_key_src

Corosync **authkey** ファイルへのパス。これは、Corosync 通信の認証および暗号鍵です。各クラスターに一意の **authkey** 値を指定することが強く推奨されます。キーは、ランダムなデータの 256 バイトでなければなりません。
この変数の鍵を指定する場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、鍵を vault 暗号化することが推奨されます。

鍵が指定されていない場合は、ノードにすでに存在するキーが使用されます。ノードに同じ鍵がない場合、あるノードの鍵が他のノードに分散され、すべてのノードが同じキーを持つようにします。ノードに鍵がない場合は、新しい鍵が生成され、ノードに分散されます。

この変数が設定されている場合は、このキーで **ha_cluster_regenerate_keys** が無視されます。

この変数のデフォルト値は null です。

ha_cluster_pacemaker_key_src

Pacemaker の **authkey** ファイルへのパスです。これは、Pacemaker 通信の認証および暗号鍵です。各クラスターに一意の **authkey** 値を指定することが強く推奨されます。キーは、ランダムなデータの 256 バイトでなければなりません。
この変数の鍵を指定する場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、鍵を vault 暗号化することが推奨されます。

鍵が指定されていない場合は、ノードにすでに存在するキーが使用されます。ノードに同じ鍵がない場合、あるノードの鍵が他のノードに分散され、すべてのノードが同じキーを持つようにします。ノードに鍵がない場合は、新しい鍵が生成され、ノードに分散されます。

この変数が設定されている場合は、このキーで **ha_cluster_regenerate_keys** が無視されます。

この変数のデフォルト値は null です。

ha_cluster_fence_virt_key_src

fence-virt または **fence-xvm** の事前共有鍵ファイルへのパス。これは、**fence-virt** または **fence-xvm** フェンスエージェントの認証キーの場所になります。
この変数の鍵を指定する場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、鍵を vault 暗号化することが推奨されます。

鍵が指定されていない場合は、ノードにすでに存在するキーが使用されます。ノードに同じ鍵がない場合、あるノードの鍵が他のノードに分散され、すべてのノードが同じキーを持つようにします。ノードに鍵がない場合は、新しい鍵が生成され、ノードに分散されます。この方法で HA Cluster システムロールが新しいキーを生成する場合は、鍵をノードのハイパーバイザーにコピーして、フェンシングが機能するようにする必要があります。

この変数が設定されている場合は、このキーで **ha_cluster_regenerate_keys** が無視されます。

この変数のデフォルト値は null です。

ha_cluster_pcsd_public_key_srcr, ha_cluster_pcsd_private_key_src

pcsd TLS 証明書および秘密鍵へのパス。これが指定されていない場合は、ノード上にすでに証明書キーのペアが使用されます。証明書キーペアが存在しない場合は、無作為に新しいキーが生成されます。
この変数に秘密鍵の値を指定した場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、鍵を暗号化することが推奨されます。

これらの変数が設定されている場合は、この証明書と鍵のペアで **ha_cluster_regenerate_keys** は無視されます。

これらの変数のデフォルト値は null です。

ha_cluster_regenerate_keys

yes に設定されるブール値フラグは、共有前の鍵と TLS 証明書を再生成することを決定します。
キーおよび証明書が再生成された場合の詳細は、変数 **ha_cluster_corosync_key_src**、**ha_cluster_pacemaker_key_src**、**ha_cluster_fence_virt_key_src**、**ha_cluster_pcsd_public_key_src**、および **ha_cluster_pcsd_private_key_src** の説明を参照してください。
この変数のデフォルト値は **no** です。

ha_cluster_pcs_permission_list

pcsd を使用してクラスターを管理するパーミッションを設定します。この変数を使用して設定するアイテムは以下のとおりです。

- **type** - **user** または **group**
- **name** - ユーザーまたはグループの名前
- **allow_list** - 指定されたユーザーまたはグループの許可されるアクション:
  - **read** - クラスターのステータスおよび設定の表示
  - **write** - パーミッションおよび ACL を除くクラスター設定の変更
  - **grant** - クラスターパーミッションおよび ACL の変更
  - **full** - ノードの追加および削除、キーおよび証明書へのアクセスなど、クラスターへの無制限アクセス
    ::**ha_cluster_pcs_permission_list** 変数の構造とデフォルト値は以下のとおりです。

```
ha_cluster_pcs_permission_list:
  - type: group
    name: hacluster
    allow_list:
      - grant
      - read
      - write
```

ha_cluster_cluster_name

クラスターの名前。これは、デフォルトが **my-cluster** の文字列値です。

ha_cluster_cluster_properties

Pacemaker クラスター全体の設定のクラスタープロパティーのセットの一覧。クラスタープロパティーのセットは 1 つだけサポートされます。

クラスタープロパティーのセットの構造は次のとおりです。

```
ha_cluster_cluster_properties:
  - attrs:
      - name: property1_name
        value: property1_value
      - name: property2_name
        value: property2_value
```

デフォルトでは、プロパティーは設定されません。

以下の Playbook の例では、**node1** および **node2** で構成されるクラスターを設定し、**stonith-enabled** および **no-quorum-policy** クラスタープロパティーを設定します。

```
- 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

この変数は、stonith リソースなど、システムロールにより設定された Pacemaker リソースを定義します。各リソースに設定できるアイテムは次のとおりです。

- **id** (必須) - リソースの ID。
- **agent** (必須) - リソースまたは stonith エージェントの名前 (例: **ocf:pacemaker:Dummy** または **stonith:fence_xvm**)。stonith エージェントには、**stonith:** を指定する必要があります。リソースエージェントの場合は、**ocf:pacemaker:Dummy** ではなく、**Dummy** などの短縮名を使用することができます。ただし、同じ名前の複数のエージェントがインストールされていると、使用するエージェントを決定できないため、ロールは失敗します。そのため、リソースエージェントを指定する場合はフルネームを使用することが推奨されます。
- **instance_attrs** (オプション): リソースのインスタンス属性のセットの一覧。現在、1 つのセットのみがサポートされています。属性の名前と値、必須かどうか、およびリソースまたは stonith エージェントによって異なります。
- **meta_attrs** (オプション): リソースのメタ属性のセットの一覧。現在、1 つのセットのみがサポートされています。
- **operations** (任意): リソースの操作のリスト。
  - **action** (必須): pacemaker と、リソースまたは stonith エージェントで定義されている操作アクション。
  - **attrs** (必須): 少なくとも 1 つのオプションを指定する必要があります。
    ::HA Cluster システムロールで設定するリソース定義の構造は以下のとおりです。

```
- 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
```

デフォルトでは、リソースは定義されません。

リソース設定を含む HA Cluster システムロール Playbook の例については、Configuring a high availability cluster with fencing and resources を参照してください。

ha_cluster_resource_groups

この変数は、システムロールによって設定される Pacemaker リソースグループを定義します。各リソースグループに設定可能な項目は、以下のとおりです。

- **id** (必須): グループの ID。
- **resources** (必須): グループのリソースの一覧。各リソースは ID によって参照され、リソースは **ha_cluster_resource_primitives** 変数に定義する必要があります。1 つ以上のリソースを一覧表示する必要があります。
- **meta_attrs** (オプション): グループのメタ属性のセットの一覧。現在、1 つのセットのみがサポートされています。
  ::HA Cluster システムロールで設定するリソースグループ定義の構造は以下のとおりです。

```
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
```

デフォルトでは、リソースグループが定義されていません。

リソースグループ設定を含む HA Cluster システムロール Playbook の例については、Configuring a high availability cluster with fencing and resources を参照してください。

ha_cluster_resource_clones

この変数は、システムロールによって設定された Pacemaker リソースクローンを定義します。リソースクローンに設定できるアイテムは次のとおりです。

- **resource_id** (必須): クローンを作成するリソース。リソースは **ha_cluster_resource_primitives** 変数または **ha_cluster_resource_groups** 変数に定義する必要があります。
- **promotable** (任意): 作成するリソースクローンが昇格可能なクローンであるかどうかを示します。これは、**yes** または **no** と示されます。
- **id** (任意): クローンのカスタム ID。ID が指定されていない場合は、生成されます。このオプションがクラスターでサポートされない場合は、警告が表示されます。
- **meta_attrs** (任意): クローンのメタ属性のセットの一覧。現在、1 つのセットのみがサポートされています。
  ::HA Cluster システムロールで設定するリソースクローン定義の構造は次のとおりです。

```
ha_cluster_resource_clones:
  - resource_id: resource-to-be-cloned
    promotable: yes
    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
```

デフォルトでは、リソースのクローンが定義されていません。

リソースクローン設定を含む HA Cluster システムロール Playbook の例については、Configuring a high availability cluster with fencing and resources を参照してください。

ha_cluster_constraints_location

この変数は、リソースの場所の制約を定義します。リソースの場所の制約は、リソースを実行できるノードを示します。複数のリソースに一致する可能性のあるリソース ID またはパターンで指定されたリソースを指定できます。ノード名またはルールでノードを指定できます。
リソースの場所の制約に対して設定できる項目は次のとおりです。

- **resource** (必須) - 制約が適用されるリソースの仕様。
- **node** (必須) - リソースが優先または回避する必要があるノードの名前。
- **id** (オプション) - 制約の ID。指定しない場合、自動生成されます。

- **options** (オプション) - 名前と値のディクショナリーリスト。
  - **score** - 制約の重みを設定します。
    - 正の **score** 値は、リソースがノードでの実行を優先することを意味します。
    - 負の **score** 値は、リソースがノードで実行されないようにする必要があることを意味します。
    - **-INFINITY** の **score** 値は、リソースがノードで実行されないようにする必要があることを意味します。
    - **score** が指定されていない場合、スコア値はデフォルトで **INFINITY** になります。 ::デフォルトでは、リソースの場所の制約は定義されていません。

      リソース ID とノード名を指定するリソースの場所に対する制約の構造は次のとおりです。

      ```
      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
      ```

      リソースパターンを指定するリソースの場所の制約に対して設定する項目は、リソース ID を指定するリソースの場所の制約に対して設定する項目と同じです。ただし、リソース仕様そのものは除きます。リソース仕様に指定する項目は次のとおりです。

- **pattern** (必須) - POSIX 拡張正規表現リソース ID が照合されます。 ::リソースパターンとノード名を指定するリソースの場所に対する制約の構造は次のとおりです。

  ```
  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
  ```

  リソース ID とルールを指定するリソースの場所の制約に対して設定できる項目は次のとおりです。

- **resource** (必須) - 制約が適用されるリソースの仕様。
  - **id** (必須) - リソース ID。

    - **role** (オプション) - 制約が制限されているリソースロール:**Started**、**Unpromoted**、**Promoted**。

- **rule** (必須) - **pcs** 構文を使用して記述された制約ルール。詳細については、**pcs** (8) の man ページで **constraint location** セクションを参照してください。

- 指定するその他の項目は、ルールを指定しないリソース制約と同じ意味を持ちます。
::リソース ID とルールを指定するリソースの場所に対する制約の構造は次のとおりです。

```
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
```

リソースパターンとルールを指定するリソースの場所の制約に対して設定する項目は、リソース ID とルールを指定するリソースの場所の制約に対して設定する項目と同じです。ただし、リソース仕様そのものは除きます。リソース仕様に指定する項目は次のとおりです。

- **pattern** (必須) - POSIX 拡張正規表現リソース ID が照合されます。
::リソースパターンとルールを指定するリソースの場所に対する制約の構造は次のとおりです。

```
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
```

リソース制約のあるクラスターを作成する **ha_cluster** システムロール Playbook の例については、Configuring a high availability cluster with resource constraints を参照してください。

ha_cluster_constraints_colocation

この変数は、リソースコロケーションの制約を定義します。リソースコロケーションの制約は、あるリソースの場所が別のリソースの場所に依存することを示しています。コロケーション制約には、2 つのリソースに対する単純なコロケーション制約と、複数のリソースに対するセットコロケーション制約の 2 種類があります。
単純なリソースコロケーション制約に対して設定できる項目は次のとおりです。

- **resource_follower** (必須) - **resource_leader** に関連して配置する必要があるリソース。

    - **id** (必須) - リソース ID。

- **role** (オプション) - 制約が制限されているリソースロール:**Started**、**Unpromoted**、**Promoted**。

- **resource_leader** (必須) - クラスターは、最初にこのリソースを配置する場所を決定してから、**resource_follower** を配置する場所を決定します。
  - **id** (必須) - リソース ID。
  - **role** (オプション) - 制約が制限されているリソースロール:**Started**、**Unpromoted**、**Promoted**。

- **id** (オプション) - 制約の ID。指定しない場合、自動生成されます。

- **options** (オプション) - 名前と値のディクショナリーリスト。
  - **score** - 制約の重みを設定します。
    - 正の **score** 値は、リソースが同じノードで実行される必要があることを示します。
    - 負の **score** 値は、リソースが異なるノードで実行される必要があることを示します。
    - **+ INFINITY** の **score** 値は、リソースが同じノードで実行される必要があることを示します。
    - **-INFINITY** の **score** 値は、リソースが異なるノードで実行される必要があることを示します。
    - **score** が指定されていない場合、スコア値はデフォルトで **INFINITY** になります。::デフォルトでは、リソースコロケーション制約は定義されていません。

      単純なリソースコロケーション制約の構造は次のとおりです。

```
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
```

      リソースセットのコロケーション制約に対して設定できる項目は次のとおりです。

- **resource_sets** (必須) - リソースセットのリスト。
  - **resource_ids** (必須) - セット内のリソースのリスト。
  - **options** (オプション) - セット内のリソースが制約によってどのように扱われるかを微調整する名前と値のディクショナリーリスト。

- **id** (オプション) - 単純なコロケーション制約の場合と同じ値。

- **options** (オプション) - 単純なコロケーション制約の場合と同じ値。
::リソースセットに対するコロケーション制約の構造は次のとおりです。

```
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
```

リソース制約のあるクラスターを作成する **ha_cluster** システムロール Playbook の例については、Configuring a high availability cluster with resource constraints を参照してください。

ha_cluster_constraints_order

この変数は、リソースの順序に対する制約を定義します。リソース順序の制約は、特定のリソースアクションが発生する順序を示します。リソース順序の制約には、2 つのリソースに対する単純な順序制約と、複数のリソースに対するセット順序制約の 2 種類があります。
単純なリソース順序制約に対して設定できる項目は次のとおりです。

- **resource_first** (必須) - **resource_then** リソースが依存するリソース。
  - **id** (必須) - リソース ID。
  - **action** (オプション) - **resource_then** リソースに対してアクションを開始する前に完了する必要のあるアクション。許可される値: **start**、**stop**、**promote**、**demode**。
- **resource_then** (必須) - 依存リソース。
  - **id** (必須) - リソース ID。
  - **action** (オプション) - **resource_first** リソースに対するアクションが完了した後にのみリソースが実行できるアクション。許可される値: **start**、**stop**、**promote**、**demode**。
- **id** (オプション) - 制約の ID。指定しない場合、自動生成されます。
- **options** (オプション) - 名前と値のディクショナリーリスト。
::デフォルトでは、リソース順序の制約は定義されていません。

  単純なリソース順序の制約の構造は次のとおりです。

```
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
```

リソースセットの順序制約に対して設定できる項目は次のとおりです。

- **resource_sets** (必須) - リソースセットのリスト。
  - **resource_ids** (必須) - セット内のリソースのリスト。
  - **options** (オプション) - セット内のリソースが制約によってどのように扱われるかを微調整する名前と値のディクショナリーリスト。
- **id** (オプション) - 単純な順序制約の場合と同じ値。
- **options** (オプション) - 単純な順序制約の場合と同じ値。
::リソースセットの順序制約の構造は次のとおりです。

```
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
```

リソース制約のあるクラスターを作成する **ha_cluster** システムロール Playbook の例については、Configuring a high availability cluster with resource constraints を参照してください。

ha_cluster_constraints_ticket

この変数は、リソースチケットの制約を定義します。リソースチケットの制約は、特定のチケットに依存するリソースを示します。リソースチケット制約には、1 つのリソースに対する単純なチケット制約と、複数のリソースに対するチケット順序の制約の 2 種類があります。
単純なリソースチケット制約に対して設定できる項目は次のとおりです。

- **resource** (必須) - 制約が適用されるリソースの仕様。
  - **id** (必須) - リソース ID。
  - **role** (オプション) - 制約が制限されているリソースロール:**Started**、**Unpromoted**、**Promoted**。
- **ticket** (必須) - リソースが依存するチケットの名前。
- **id** (オプション) - 制約の ID。指定しない場合、自動生成されます。

- **options** (オプション) - 名前と値のディクショナリーリスト。
  - **loss-policy** (オプション) - チケットが取り消された場合にリソースに対して実行するアクション。
    ::デフォルトでは、リソースチケットの制約は定義されていません。

    単純なリソースチケット制約の構造は次のとおりです。

```
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
```

    リソースセットのチケット制約に対して設定できる項目は次のとおりです。

- **resource_sets** (必須) - リソースセットのリスト。
  - **resource_ids** (必須) - セット内のリソースのリスト。
  - **options** (オプション) - セット内のリソースが制約によってどのように扱われるかを微調整する名前と値のディクショナリーリスト。
- **ticket** (必須) - 単純なチケット制約の場合と同じ値。
- **id** (オプション) - 単純なチケット制約の場合と同じ値。
- **options** (オプション) - 単純なチケット制約の場合と同じ値。
  ::リソースセットのチケット制約の構造は次のとおりです。

```
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
```

リソース制約のあるクラスターを作成する **ha_cluster** システムロール Playbook の例については、Configuring a high availability cluster with resource constraints を参照してください。

22.2. HA CLUSTER システムロールのインベントリーの指定

HA Cluster システムロール Playbook を使用して HA クラスターを設定する場合は、インベントリー内のクラスターの名前とアドレスを設定します。

インベントリー内の各ノードには、必要に応じて以下の項目を指定することができます。

- **node_name** - クラスター内のノードの名前。
- **pcs_address** - ノードと通信するために **pcs** が使用するアドレス。名前、FQDN、または IP アドレスを指定でき、ポート番号を含めることができます。
- **corosync_addresses**: Corosync が使用するアドレスの一覧。特定のクラスターを形成するすべてのノードは、同じ数のアドレスが必要で、アドレスの順序が重要です。

以下の例は、ターゲット **node1** および **node2** を持つインベントリーを示しています。**node1** および **node2** は完全修飾ドメイン名のいずれかである必要があります。そうでないと、たとえば、名前が **/etc/hosts** ファイルで解決可能である場合などに、ノードに接続できる必要があります。

```
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
```

22.3. リソースを実行していない高可用性クラスターの設定

以下の手順では、HA Cluster システムロールを使用して、フェンシングが設定されていない高可用性クラスターと、リソースを実行しない高可用性クラスターを作成します。

前提条件

- Playbook を実行するノードに **ansible-core** がインストールされている。

注記

ansible-core をクラスターメンバーノードにインストールする必要はありません。

- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。RHEL システムロールと、その適用方法の詳細は、RHEL システムロールの使用 を参照してください。
- クラスターメンバーとして使用する RHEL を実行しているシステムに、RHEL および RHEL High Availability Add-On の有効なサブスクリプション範囲が必要です。

警告

HA Cluster システムロールは、指定されたノードの既存のクラスター設定を置き換えます。ロールで指定されていない設定は失われます。

手順

1. HA Cluster システムロールのインベントリーの指定 で説明されているように、クラスター内のノードを指定するインベントリーファイルを作成します。
2. Playbook ファイルを作成します (例: **new-cluster.yml**)。
   以下の Playbook ファイルの例では、フェンシングが設定されていないクラスターと、リソースを実行しないクラスターを設定します。実稼働環境の Playbook ファイルを作成する場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、パスワード vault を暗号化することが推奨されます。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password

  roles:
    - rhel-system-roles.ha_cluster
```

3. ファイルを保存します。
4. 手順 1 で作成したインベントリーファイル **inventory** へのパスを指定して、Playbook を実行します。

```
# ansible-playbook -i inventory new-cluster.yml
```

22.4. フェンシングおよびリソースを使用した高可用性クラスターの設定

以下の手順では、HA Cluster システムロールを使用して、フェンスデバイス、クラスターリソース、リソースグループ、およびクローンされたリソースを含む高可用性クラスターを作成します。

前提条件

- Playbook を実行するノードに **ansible-core** がインストールされている。

注記

ansible-core をクラスターメンバーノードにインストールする必要はありません。

- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。
  RHEL システムロールと、その適用方法の詳細は、RHEL システムロールの使用 を参照してください。

- クラスターメンバーとして使用する RHEL を実行しているシステムに、RHEL および RHEL High Availability Add-On の有効なサブスクリプション範囲が必要です。

警告

HA Cluster システムロールは、指定されたノードの既存のクラスター設定を置き換えます。ロールで指定されていない設定は失われます。

手順

1. HA Cluster システムロールのインベントリーの指定 で説明されているように、クラスター内のノードを指定するインベントリーファイルを作成します。

2. Playbook ファイルを作成します (例: **new-cluster.yml**)。
以下の Playbook ファイルの例では、フェンシング、複数のリソース、およびリソースグループを含むクラスターを設定します。また、リソースグループのリソースクローンも含まれます。
実稼働環境の Playbook ファイルを作成する場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、パスワード vault を暗号化することが推奨されます。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    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. ファイルを保存します。

4. 手順 1 で作成したインベントリーファイル **inventory** へのパスを指定して、Playbook を実行します。

```
# ansible-playbook -i inventory new-cluster.yml
```

22.5. リソースに制約のある高可用性クラスターの設定

次の手順では、**ha_cluster** システムロールを使用して、リソースの場所の制約、リソースコロケーションの制約、リソース順序の制約、およびリソースチケットの制約を含む高可用性クラスターを作成します。

前提条件

- Playbook を実行するノードに **ansible-core** がインストールされている。

注記

ansible-core をクラスターメンバーノードにインストールする必要はありません。

- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。RHEL システムロールと、その適用方法の詳細は、「RHEL システムロールの使用」を参照してください。
- クラスターメンバーとして使用する RHEL を実行しているシステムに、RHEL および RHEL High Availability Add-On の有効なサブスクリプション範囲が必要です。

警告

ha_cluster システムロールは、指定されたノードの既存のクラスター設定を置き換えます。ロールで指定されていない設定は失われます。

手順

1. 「ha_cluster システムロールのインベントリーの指定」で説明されているように、クラスター内のノードを指定するインベントリーファイルを作成します。
2. Playbook ファイルを作成します (例: **new-cluster.yml**)。
   次の Playbook ファイルの例は、リソースの場所の制約、リソースコロケーションの制約、リソース順序の制約、およびリソースチケットの制約を含むクラスターを設定します。実稼働環境の Playbook ファイルを作成する場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、パスワード vault を暗号化することが推奨されます。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    # 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. ファイルを保存します。

4. 手順 1 で作成したインベントリーファイル **inventory** へのパスを指定して、Playbook を実行します。

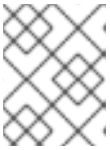
```
# ansible-playbook -i inventory new-cluster.yml
```

22.6. HA CLUSTER システムロールを使用した高可用性クラスターでの APACHE HTTP サーバーの設定

この手順では、HA Cluster システムロールを使用して、2 ノードの Red Hat Enterprise Linux High Availability Add-On クラスターでアクティブ/パッシブな Apache HTTP サーバーを設定します。

前提条件

- Playbook を実行するノードに **ansible-core** がインストールされている。



注記

ansible-core をクラスターメンバーノードにインストールする必要はありません。

- Playbook を実行するシステムに **rhel-system-roles** パッケージがインストールされている。RHEL システムロールと、その適用方法の詳細は、RHEL システムロールの使用 を参照してください。
- クラスターメンバーとして使用する RHEL を実行しているシステムに、RHEL および RHEL High Availability Add-On の有効なサブスクリプション範囲が必要です。
- システムに Apache に必要なパブリック仮想 IP アドレスが含まれている。
- システムに、iSCSI、ファイバーチャネル、またはその他の共有ネットワークブロックデバイスを使用する、クラスターのノードに対する共有ストレージが含まれます。
- Configuring an LVM volume with an ext4 file system in a Pacemaker cluster の説明に従って、ext4 ファイルシステムで LVM 論理ボリュームを設定している。
- Configuring an Apache HTTP Server の説明に従って、Apache HTTP サーバーを設定している。
- システムに、クラスターノードをフェンスするのに使用される APC 電源スイッチが含まれます。

警告

HA Cluster システムロールは、指定されたノードの既存のクラスター設定を置き換えます。ロールで指定されていない設定は失われます。

手順

1. HA Cluster システムロールのインベントリーの指定 で説明されているように、クラスター内のノードを指定するインベントリーファイルを作成します。

2. Playbook ファイルを作成します (例: **http-cluster.yml**)。
以下の Playbook ファイルの例では、アクティブ/パッシブの 2 ノード HA クラスターで事前に作成した Apache HTTP サーバーを設定します。

   この例では、ホスト名が **zapc.example.com** の APC 電源スイッチを使用します。クラスターが他のフェンスエージェントを使用しない場合は、以下の例のように、**ha_cluster_fence_agent_packages** 変数を定義するときに、クラスターが必要とするフェンスエージェントのみを任意で一覧表示できます。

   実稼働環境の Playbook ファイルを作成する場合は、「Ansible Vault を使用したコンテンツの暗号化」で説明されているように、パスワード vault を暗号化することが推奨されます。

```
- 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_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: ext4
      - 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. ファイルを保存します。

4. 手順 1 で作成したインベントリーファイル **inventory** へのパスを指定して、Playbook を実行します。

```
# ansible-playbook -i inventory http-cluster.yml
```

検証手順

1. クラスター内のノードのいずれかから、クラスターのステータスを確認します。4 つのリソースがすべて同じノード (**z1.example.com**) で実行されていることに注意してください。
設定したリソースが実行されていない場合は、**pcs resource debug-start resource** コマンドを実行してリソースの設定をテストできます。

```
[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. クラスターが稼働したら、ブラウザーで、**IPaddr2** リソースとして定義した IP アドレスを指定して、「Hello」と単語が表示されるサンプル表示を確認します。

```
Hello
```

3. **z1.example.com** で実行しているリソースグループが **z2.example.com** ノードにフェールオーバーするかどうかをテストするには、ノード **z1.example.com** を **standby** にすると、ノードがリソースをホストできなくなります。

```
[root@z1 ~]# pcs node standby z1.example.com
```

4. ノード **z1** を **standby** モードにしたら、クラスター内のノードのいずれかからクラスターのステータスを確認します。リソースはすべて **z2** で実行しているはずです。

```
[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
```

定義している IP アドレスの Web サイトは、中断せず表示されているはずです。

5. **スタンバイ** モードから **z1** を削除するには、以下のコマンドを実行します。

```
[root@z1 ~]# pcs node unstandby z1.example.com
```

注記

ノードを **スタンバイ** モードから削除しても、リソースはそのノードにフェイルオーバーしません。これは、リソースの **resource-stickiness** 値により異なります。**resource-stickiness** メタ属性の詳細は、Configuring a resource to prefer its current node を参照してください。

22.7. 関連情報

- RHEL システムロールの使用

- **rhel-system-roles** パッケージでインストールされたドキュメント (**/usr/share/ansible/roles/rhel-system-roles.logging/README.html**)
- RHEL システムロール のナレッジベース記事
- **ansible-playbook(1)** の man ページ

第23章 COCKPIT RHEL システムロールを使用した WEB コンソールのインストールと設定

Cockpit RHEL システムロールを使用すると、システムに Web コンソールをインストールして設定できます。

23.1. COCKPIT システムロール

Cockpit システムロールを使用して、Web コンソールを自動的にデプロイして有効にできます。その結果、Web ブラウザーから RHEL システムを管理できるようになります。

23.2. COCKPIT RHEL システムロールの変数

Cockpit RHEL システムロールに使用されるパラメーターは次のとおりです。

ロール変数	説明
cockpit_packages: (default: default)	事前定義されたパッケージセット (default、minimal、full) の 1 つを設定します。 * cockpit_packages: (default: default) - 最も一般的なページとオンデマンドインストール UI * cockpit_packages: (default: minimal) - 概要、ターミナル、ログ、アカウント、およびメトリックのページのみ。最小限の依存関係。 * cockpit_packages: (default: full) - 利用可能なすべてのページ。 必要に応じて、インストールするコックピットパッケージを独自に選択します。
cockpit_enabled: (default:yes)	Web コンソール Web サーバーが起動時に自動起動できるかどうかを設定します。
cockpit_started: (default:yes)	Web コンソールを起動する必要があるかどうかを設定します。
cockpit_config: (default: nothing)	**/etc/cockpit/cockpit.conf** ファイルで設定を適用できます。注記:以前の設定ファイルは失われます。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.cockpit/README.md** ファイル。
- Cockpit configuration file の man ページ。

23.3. COCKPIT RHEL システムロールを使用した WEB コンソールのインストール

以下の手順に従って、システムに Web コンソールをインストールし、システム内でサービスにアクセスできるようにします。

前提条件

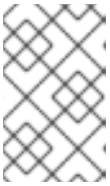
- 1 つ以上の **管理対象ノード** (VPN システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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

    - name: Configure Firewall for web console
      include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          service: cockpit
          state: enabled
```



注記

デフォルトでコックピットポートは firewalld で開いているため、「Web コンソールのファイアウォール設定」タスクは、システム管理者がこれをカスタマイズした場合にのみ適用されます。

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

関連情報

- Installing and enabling the web console 。

23.4. CERTIFICATE RHEL システムロールを使用した新しい証明書の設定

デフォルトでは、Web コンソールは最初の起動時に自己署名証明書を作成します。セキュリティー上の理由から、自己署名証明書をカスタマイズできます。新しい証明書を生成するには、Certificate ロールを使用できます。そのためには、以下の手順に従います。

前提条件

- 1 つ以上の **管理対象ノード** (VPN システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook2.yml** ファイルを作成します。

```
---
- hosts: all
  tasks:
    - name: Generate Cockpit web server certificate
      include_role:
        name: rhel-system-roles.certificate
      vars:
        certificate_requests:
          - name: /etc/cockpit/ws-certs.d/01-certificate
            dns: ['localhost', 'www.example.com']
            ca: ipa
            group: cockpit-ws
```

2. オプション:Playbook の構文を確認します。

```
# ansible-playbook --syntax-check -i inventory_file playbook2.yml
```

3. インベントリーファイルで Playbook を実行します。

```
# ansible-playbook -i inventory_file /path/to/file/playbook2.yml
```

関連情報

- Requesting certificates using RHEL System Roles 。