Red Hat Enterprise Linux 8

RHEL の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: 2021-09-22

Red Hat Enterprise Linux 8 RHEL の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 8 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。Red Hat Enterprise Linux 8 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 (英語のみ)

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。改善点を報告する場合は、以下のように行います。

- 特定の文章に簡単なコメントを記入する場合は、以下の手順を行います。
  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端に **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。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
  1. Bugzilla の Web サイトにアクセスします。
  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

本セクションでは、RHEL システムロールの概要を説明します。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 8 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。

- kdump
- network
- selinux
- storage
- certificate
- kernel_settings
- logging
- metrics
- nbde_client and nbde_server
- timesync
- tlog

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** ドキュメント [1]
- 「SELinux システムロールの概要」
- 「ストレージロールの概要」
- RHEL の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

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。

システムロールの用語

システムロールの用語

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。

```
# yum install rhel-system-roles
```

- RHEL システムロールを使用する Playbook を実行するには、**ansible** パッケージが必要です。Ansible Engine リポジトリーが有効になり、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ことを確認します。
  - Red Hat Ansible Engine サブスクリプションをお持ちでない場合は、Red Hat Enterprise Linux サブスクリプションで提供される Red Hat Ansible Engine の限定サポートバージョンをご利用できます。この場合は、次の手順を実行します。
    1. RHEL Ansible Engine リポジトリーを有効にします。

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

    2. Ansible Engine をインストールします。

```
# yum install ansible
```

  - Red Hat Ansible Engine のサブスクリプションをお持ちの場合は、「Red Hat Ansible Engine のダウンロードおよびインストールの方法」に記載されている手順を行ってください。

- 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.timesync
```

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 は、**selinux**、**kdump**、**network**、**timesync**、または **storage** などの必要なロールの名前に置き換えます。

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** を使用するように Playbook を編集して、以下の

コマンドを実行します。

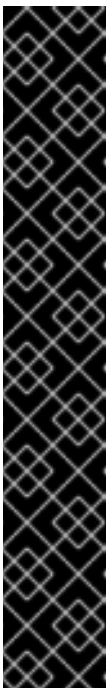
```
# ansible-playbook name.of.the.playbook
```

- 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.timesync
```

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

1.4. 関連情報

- Red Hat ナレッジベースの記事: 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 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

- Red Hat Ansible Engine のサブスクリプションを利用している。「Red Hat Ansible Engine のダウンロードおよびインストールの方法」 を参照してください。
- 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。

```
# yum install rhel-system-roles
```

Red Hat Ansible Engine サブスクリプションをお持ちでない場合は、Red Hat Enterprise Linux サブスクリプションで提供される Red Hat Ansible Engine の限定サポートバージョンを使用できます。この場合は、次の手順を実行します。

a. RHEL Ansible Engine リポジトリーを有効にします。

```
# subscription-manager refresh

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

b. Ansible Engine をインストールします。

```
# yum install ansible
```

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 コレクションのインストールおよび使用

3.1. 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 ロールの従来の 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、カーネルロールの **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確認できます。

3.2. 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 フィルタープラグイン
  - 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

3.3. 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

- Red Hat Ansible Engine バージョン 2.9 以降がインストールされている。
- **python3-jmespath** パッケージがインストールされている。
- 管理ノードが記載されているインベントリーファイルが存在する。

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 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

3.4. 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

- Red Hat Ansible Engine バージョン 2.9 以降がインストールされている。
- **python3-jmespath** パッケージがインストールされている。
- 管理ノードが記載されているインベントリーファイルが存在する。

手順

1. 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
```

2. **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。
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

3.5. コレクションを使用したローカルロギングシステムロールの適用

以下の例では、コレクションを使用して Red Hat Ansible Engine 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。

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. 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 YAML ファイルに挿入します。

   ```
   ---
   - name: Deploying basics input and implicit files output
     hosts: all
     roles:
       - redhat.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 logging-playbook.yml
   ```

   ここでは、

   - **inventory-file** は、インベントリーファイルの名前に置き換えます。
   - **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。

   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名前 (この場合は **root**) が表示されます。

第4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

Red Hat Ansible Engine の詳しい知識がある経験のあるユーザーは、**kernel_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。

4.1. 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Ansible Automation Platformのロールおよびモジュールの集合です。

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。

インベントリーファイルを使用して、Ansible Engine が Playbook に従って設定するシステムセットを定義することができます。

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

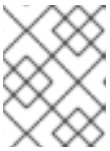
- **/usr/share/doc/rhel-system-roles/kernel_settings/** ディレクトリーの **README.md ファイルおよび README.html** ファイル
- Playbook の使用
- インベントリーの構築方法

4.2. カーネル設定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

- Red Hat Ansible Engine サブスクリプションが、**kernel_settings** ロールの実行元となる **コントロールマシン** とも呼ばれるシステムに割り当てられている。詳細は、「Red Hat Ansible Engine のダウンロードおよびインストールの方法」 を参照してください。
- コントロールマシンで Ansible Engine リポジトリーが有効になっている。
- Ansible Engine がコントロールマシンにインストールされている。



注記

Ansible Engine を、**管理ホスト** とも呼ばれるシステムにインストールする必要はありません。ここでは、カーネルパラメーターを設定する必要があります。

- **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。
- 管理ホストのインベントリーがコントロールマシンに存在し、Ansible Engine がそれらのホストに接続できる。

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、特定のシステムのコレクションに対して Ansible Engine をより効果的に実行できます。

2. Ansible Engine 操作のデフォルトおよび権限昇格を設定するための設定ファイルを作成します。

   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>/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。

   ```
   ---
   - name: Configure kernel settings
     hosts: testingservers

     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

     roles:
       - linux-system-roles.kernel_settings
   ```

   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 Engine 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**
- **インベントリーの使用**
- **Ansible の設定**
- **Playbook の使用**
- **変数の使用**
- **Roles**

第5章 システムロールを使用したネットワーク接続の設定

RHEL の network 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。

5.1. 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、RHEL システムロールを使用して、以下の設定で enp7s0 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

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 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。
- 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: linux-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

5.2. 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 サーバーはネットワークで使用できる。
- 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。

- 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: linux-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

5.3. システムロールを使用した VLAN タグの設定

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を使用する ID 10 の VLAN を追加する方法を説明します。親デバイスとして、VLAN 接続には IP、デフォルトゲートウェイ、および DNS 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 VLAN をポートとして使用するには、ip 属性を省略し、親設定で IP 設定を設定します。
- 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 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。
- 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: linux-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: vlan10
        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

5.4. 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 の 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 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。

- 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: linux-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

5.5. 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

ネットワーク 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 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 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。
- 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: linux-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

5.6. 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

RHEL システムロールを使用すると、802.1X 標準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** 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。
- 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: linux-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

5.7. 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

network の RHEL システムロールを使用するプレイを実行すると、設定がプレイで指定されたものにマッチしない場合に、システムロールは、既存の接続プロファイルを上書きして上書きします。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 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。
- 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: linux-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

5.8. 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

network の RHEL システムロールを使用するプレイを実行すると、設定がプレイで指定されたものにマッチしない場合に、システムロールは、既存の接続プロファイルを上書きして上書きします。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 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。

- 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: linux-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。

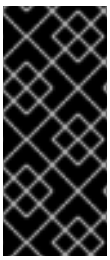
```
# ip -4 route
default via 198.51.100.254 dev enp7s0 proto static metric 100
192.0.2.0/24 via 198.51.100.1 dev enp7s0 proto static metric 100
203.0.113.0/24 via 198.51.100.2 dev enp7s0 proto static metric 100
...
```

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

5.9. 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

network の RHEL システムロールを使用するプレイを実行すると、設定がプレイで指定されたものにマッチしない場合に、システムロールは、既存の接続プロファイルを上書きして上書きします。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** 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。
- 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: linux-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:
            feature:
              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

5.10. システムロールを使用した ETHTOOL COALESCE 設定

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

重要

network の RHEL システムロールを使用するプレイを実行すると、設定がプレイで指定されたものにマッチしない場合に、システムロールは、既存の接続プロファイルを上書きして上書きします。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 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。
- 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: linux-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6章 システムロールを使用した SELINUX の設定

6.1. 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 システムロールは、以下のアクションを有効にします。

- **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6.1 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。**ansible-doc selinux** を参照してください。	**/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。**ansible-doc seboolean** を参照してください。	**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。**ansible-doc sefcontext** を参照してください。	**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。**ansible-doc seport** を参照してください。	**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。**ansible-doc selogin** を参照してください。	**semanage login**

ロール変数	説明	CLI の代替手段
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

6.2. SELINUX システムロールを使用した複数のシステムに SELINUX 設定を適用

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

- **Red Hat Ansible Engine のサブスクリプションがシステムに割り当てられている。詳細は、「Red Hat Ansible Engine のダウンロードおよびインストールの方法」を参照してください。**

手順

1. **RHEL Ansible リポジトリーを有効にします。以下に例を示します。**

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

2. **Ansible Engine をインストールします。**

```
# yum install ansible
```

3. **RHEL システムロールをインストールします。**

```
# yum install rhel-system-roles
```

4. **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
```

5. **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" }
```

6. **変更を保存し、テキストエディターを終了します。**

7. **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7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。

7.1. 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。

Ansible が 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るログ
- Elasticsearch に送信されるログ
- 別のロギングシステムに転送されるログ

logging システムロールを使用すると、必要に応じて入力と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。

7.2. 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に **imklog** を読み込みます。デフォルトは **false** です。*
    - ***use_imuxsock**: **imjournal** ではなく **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。*
- *状態: 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**)*

7.3. 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、Red Hat Ansible Engine 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。

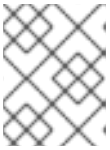
前提条件

- *Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。*

注記

ロギングソリューションをデプロイするシステムに Red Hat Ansible Engine 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** パッケージがある。*



注記

*デプロイ時に **rsyslog** がシステムロールによりインストールされるので、**rsyslog** はインストールする必要はありません。*

- *ロギングソリューションを設定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. *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:
    - linux-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
```

ここで、 **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。*

7.4. 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。*
- *Red Hat Ansible Engine が他のシステムを設定する* **コントロールノード** *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 Engine 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。*

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_input0
      type: files
      input_log_path: /var/log/containerA/*.log
    - name: files_input1
      type: files
      input_log_path: /var/log/containerB/*.log
  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_input0, files_input1]
      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)**

7.5. 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 Engine 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

- Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。

注記

ロギングソリューションをデプロイするシステムに Red Hat Ansible Engine 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** パッケージがある。

注記

デプロイ時に **rsyslog** がシステムロールによりインストールされるので、**rsyslog** はインストールする必要はありません。

- 2 つ以上のシステムがある。
  - 少なくとも 1 つはロギングサーバー
  - 少なくとも 1 つはロギングクライアント

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:
    - linux-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:
    - linux-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
   ```

   詳細は以下のようになります。

   - **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*
- ***/usr/share/ansible/roles/rhel-system-roles.logging/README.html*** の **rhel-system-roles** パッケージ
- *RHEL システムロール*

7.6. 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8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て任意数の RHEL システムに同時に同じ設定の SSH クライアントを Red Hat Ansible Automation Platform で設定できます。

8.1. SSHD システムロール変数

SSHD 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 ::
```

SSHD システムロールの変数

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。

SSHD 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_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 8 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。*

8.2. SSHD システムロールを使用した OPENSSH サーバーの設定

SSHD 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。

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。*
- *Red Hat Ansible Engine が他のシステムを設定する* **コントロールノード** *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 Engine 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。*

手順

1. *SSHD システムロールの Playbook の例をコピーします。*

```
# 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
```

詳細は以下のようになります。

- ***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*** *file.*
- ***ansible-playbook(1)*** *man ページ。*

8.3. SSH システムロール変数

SSH *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 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。*

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

SSH 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。

前提条件

- *1 つ以上の **管理対象ノード** (SSH 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。*
- *Red Hat Ansible Engine が他のシステムを設定する **コントロールノード** 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 Engine 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。*

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 Client 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
```

第9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、RHEL で Crypto Policies の システムロールを使用して、Red Hat Ansible Automation Platform でさまざまなシステムにカスタムの暗号化ポリシーを迅速かつ一貫して設定できます。

9.1. CRYPTO POLICIES システムロール変数およびファクト

Crypto 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。

Crypto 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レベルを決定します。異なる暗号化ポリシーレベルの詳細は、「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。

Crypto 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レベルを表示します。

crypto_policies_available_modules

システムで利用可能なサブポリシーモジュールをすべて表示します。

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

9.2. CRYPTO 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

Crypto 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。

- *Red Hat Ansible Engine が他のシステムを設定する* **コントロールノード** *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 Engine 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。*

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: linux-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* **crypto** *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: linux-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。

9.3. 関連情報

- **/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md** file.
- **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10章 CLEVIS および TANG のシステムロールの使用

10.1. 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。

RHEL 8.3 では、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 ロールが導入されました。**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。

nbde_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。**nbde_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。

nbde_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。

nbde_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。

nbde_server 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、「RHEL システムロールの概要」を参照してください。

10.2. 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-server 設定を含む Ansible Playbook を準備および適用します。

前提条件

- Red Hat Ansible Engine のサブスクリプションがシステムに割り当てられている。詳細は、「Red Hat Ansible Engine のダウンロードおよびインストールの方法」を参照してください。

手順

1. RHEL Ansible リポジトリーを有効にします。以下に例を示します。

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

2. Ansible Engine をインストールします。

```
# yum install ansible
```

3. RHEL システムロールをインストールします。

```
# yum install rhel-system-roles
```

4. 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
```

5. 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
```

6. 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。

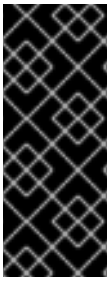
```
---
- hosts: all

  vars:
    nbde_server_rotate_keys: yes

  roles:
    - linux-system-roles.nbde_server
```

7. 終了した Playbook を適用します。

```
# ansible-playbook -i host1,host2,host3 my-tang-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。*

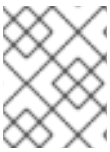
```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

関連情報

- *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。*

10.3. 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-client 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

- *Red Hat Ansible Engine のサブスクリプションがシステムに割り当てられている。詳細は、「Red Hat Ansible Engine のダウンロードおよびインストールの方法」を参照してください。*
- *ボリュームは、すでに LUKS で暗号化されています。*

手順

1. *RHEL Ansible リポジトリーを有効にします。以下に例を示します。*

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

2. *Ansible Engine をインストールします。*

```
# yum install ansible
```

3. *RHEL システムロールをインストールします。*

```
# yum install rhel-system-roles
```

4. *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
```

5. *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
```

6. *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。*

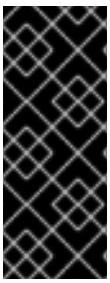
```
---
- hosts: all

  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:
    - linux-system-roles.nbde_client
```

7. *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** 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11章 RHEL システムロールを使用した証明書の要求

Certificate システムロールを使用すると、Red Hat Ansible Engine を使用して証明書の発行および管理ができます。

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

11.1. CERTIFICATE システムロール

Certificate システムロールを使用すると、Red Hat Ansible Engine を使用して TLS 証明書と SSL 証明書の発行および更新を管理することができます。

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 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。

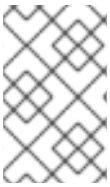
関連情報

- **certificate_requests** 変数で使用されるパラメーターの詳細と、**certificate** システムロールに関する追加情報は、**/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** ファイルを参照してください。
- 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。

11.2. 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すると、Red Hat Ansible Engine を使用して自己署名証明書を発行することができます。

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

- Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。

注記

***certificate** ソリューションをデプロイするシステムに **Ansible** 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。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。*

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

- ***certificate_requests** 変数で使用されるパラメーターの詳細と、**certificate** システムロールに関する追加情報は、**/usr/share/ansible/roles/rhel-system-roles.certificate/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。*

11.3. 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

Certificate システムロールを使用すると、統合認証局 (CA) がある IdM サーバーを使用しながら、Red Hat Ansible Engine 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

- *Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。*

注記

***certificate** ソリューションをデプロイするシステムに Ansible 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。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。*

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

- **certificate_requests** 変数で使用されるパラメーターの詳細と、**certificate** システムロールに関する追加情報は、**/usr/share/ansible/roles/rhel-system-roles.certificate/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。

11.4. CERTIFICATE システムロールを使用して、証明書の発行前または発行後に実行するコマンドの指定

Certificate システムロールを使用すると、Red Hat Ansible Engine 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することができます。

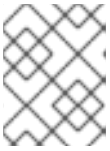
以下の例では、管理者が **www.example.com** の自己署名証明書を発行または更新する前に **httpd** サービスを停止し、後で再起動します。

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

- *Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。*



注記

***certificate** ソリューションをデプロイするシステムに Ansible 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。*
  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。

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:
  - linux-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

- **certificate_requests** 変数で使用されるパラメーターの詳細と、**certificate** システムロールに関する追加情報は、**/usr/share/ansible/roles/rhel-system-roles.certificate/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12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 **kdump** ロールを使用できます。

kdump 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。

12.1. KDUMP RHEL システムロール

kdump 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。

12.2. KDUMP ロールのパラメーター

kdump RHEL システムロールに使用するパラメーターは以下のとおりです。

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および **kdump** 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。

12.3. RHEL システムロールを使用した KDUMP の設定

Ansible Playbook を実行して **kdump** 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

kdump ロールは、**/etc/kdump.conf** ファイルを置き換えることで、管理対象ホストの kdump 設定全体を置き換えます。また、kdump 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で、ロール変数で指定されていない場合でも、以前の kdump の設定もすべて置き換えられます。

前提条件

- *Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。*

注記

***kdump** 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。*
- ***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

- *kdump 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。*
- *「ロールの適用」 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13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 8 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。

13.1. 「ストレージロールの概要」

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

13.2. 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。

13.3. 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13.1 /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 8 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。詳細は、「論理ボリュームを管理する Ansible Playbook の例」 を参照してください。*
  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。*

13.4. 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13.2 **/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。

13.5. 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13.3 **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。*

13.6. 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13.4 /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。*

13.7. 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13.5 /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。

13.8. 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13.6 **/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。*

13.9. STORAGE RHEL システムロールを使用して既存の 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13.7 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13.8 **/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。*

13.10. STORAGE RHEL システムロールを使用して、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13.9 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。

13.11. STORAGE RHEL システムロールを使用して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13.10 /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

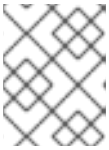
- /usr/share/ansible/roles/rhel-system-roles.storage/README.mdファイル。

13.12. 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

- *Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。*



注記

***storage** 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。*
- ***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

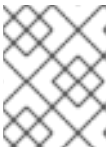
- *RAID の管理*
- */usr/share/ansible/roles/rhel-system-roles.storage/README.md* ファイル。

13.13. 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

- Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。



注記

storage 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。
- **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

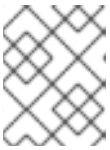
- RAID の管理
- /usr/share/ansible/roles/rhel-system-roles.storage/README.md ファイル。

13.14. STORAGE 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

- Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。



注記

ボリュームを作成するシステムに Red Hat Ansible Automation Platform をインストールする必要はありません。

- Ansible コントローラーに **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、LUKS 暗号化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_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

- *LUKS を使用したブロックデバイスの暗号化*
- ***/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル*

13.15. 関連情報

- ***/usr/share/doc/rhel-system-roles/storage/***
- ***/usr/share/ansible/roles/rhel-system-roles.storage/***

第14章 RHEL システムロールを使用した時刻同期の設定

timesync 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。

14.1. TIMESYNC システムロール

timesync 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、**timesync** 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。

timesync ロールを使用すると、システムが **ntp** または **chrony** を使用して NTP プロトコルを実装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。

14.2. サーバーの 1 つのプールへの TIMESYNC 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、**timesync** ロールを適用する方法を示しています。

警告

timesync 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

- Playbook を実行するシステムに Red Hat Ansible Engine がインストールされている。

注記

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。
- **timesync** 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
```

14.3. TIMESYNC システムロール変数

*以下の変数を **timesync** 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

関連情報

- ***timesync** 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** ファイルまたは **README.html** ファイル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パフォーマンスの監視

システム管理者は、Ansible Automation Platform コントロールノードでメトリック 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。

15.1. メトリック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。メトリック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。メトリック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15.1 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: false**
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: false**
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、メトリック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。*

15.2. メトリック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、メトリック ***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、****grafana*** *でデータ視覚化を同時にプロビジョニングする方法を説明します。*

前提条件

- *監視するマシンに* ***Red Hat Ansible Engine*** *が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。*

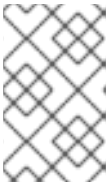
```
localhost ansible_connection=local
```

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 *ブール値は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****pcp*** *が追加されて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。*

15.3. メトリック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の集合の設定

この手順では、メトリックシステムロールを使用して、それ自体を監視するマシンの集合の設定方法を説明します。

前提条件

- *Playbook の実行に使用するマシンに Red Hat Ansible Engine が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. *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
```

15.4. メトリックシステムロールを使用したローカルマシンを介したマシンの集合の一元監視

*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、メトリックシステムロールを使用して、マシンの集合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

- *Playbook の実行に使用するマシンに Red Hat Ansible Engine が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。*

15.5. METRICS システムロールを使用したシステムの監視中の認証の設定

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。metric 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、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

- *Playbook の実行に使用するマシンに Red Hat Ansible Engine が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://127.0.0.1?username=your_username" disk.dev.read
```

```
Password:
disk.dev.read
inst [0 or "sda"] value 19540
```

15.6. 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および有効化

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

- 監視するマシンに Red Hat Ansible Engine が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。
- 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。

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_sql: 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16章 TLOG RHEL システムロールを使用したセッションの記録用のシステムの設定

tlog 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記録するようにシステムを設定できます。

16.1. TLOG システムロール

tlog RHEL システムロールを使用して、RHEL での端末セッションの記録用に RHEL システムを設定できます。**tlog** パッケージおよび関連する Web コンソールセッションプレーヤーを使用すると、ユーザーの端末セッションを記録および再生できます。

SSSD サービスを介して、ユーザーごと、またはユーザーグループごとに記録を行うように設定できます。端末の入出力はすべて収集され、テキスト形式でシステムジャーナルに保存されます。

関連情報

- RHEL でのセッションの記録に関する詳細は、『セッションの記録』を参照してください。

16.2. TLOG システムロールのコンポーネントおよびパラメーター

セッション記録ソリューションは、以下のコンポーネントで構成されています。

- 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

tlog RHEL システムロールに使用されるパラメーターは以下のとおりです。

ロール変数	説明
tlog_use_sssd (default: yes)	SSSD を使用してセッションの記録を設定します (記録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 記録スコープの設定: all / some / none
tlog_users_sssd (default: [])	記録するユーザーの YAML リスト
tlog_groups_sssd (default: [])	記録するグループの YAML リスト

- **tlog** で使用されるパラメーターの詳細と、tlog 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。

16.3. TLOG RHEL システムロールのデプロイ

以下の手順に従って、Ansible Playbook を準備して適用し、RHEL システムが **systemd** ジャーナルに記録データをロギング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 **tlog** 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。*
- *Ansible Engine が他のシステムを設定するシステムである 1 つのコントロールノードがある。*
- *Playbook を実行するコントロールノードに Red Hat Ansible Engine がインストールされている。*
- *Playbook を実行するコントロールノードに **rhel-system-roles** パッケージがインストールされている。*
- ***tlog** システムロールを設定するシステムが 1 つ以上ある。**tlog** 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。*

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:
    - 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記録することを指定します。*
   - ***tlog_users_sssd**:*
     - ***recordeduser** は、セッションを記録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** 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記録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。

16.4. グループまたはユーザーの一覧を除外するための TLOG RHEL システムロールのデプロイ

*SSSD セッションの記録設定オプション **exclude_users** および**exclude_groups** に対応するために、RHEL で **tlog** システムロールを使用することができます。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記録して **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 **tlog** 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。*
- *コントロールノードが **1** つある。このノードは、**Red Hat Ansible Engine** から他のシステムの設定に使用するシステムです。*
- *Playbook を実行するコントロールノードに **Red Hat Ansible Engine** が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ノードにインストールされている。*
- ***tlog** システムロールを設定するシステムが **1** つ以上ある。*
  ***tlog** 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。*

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記録するように指定します。
- **tlog_exclude_users_sssd**:
  - User name: セッションの記録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、セッション記録から除外するグループを指定します。

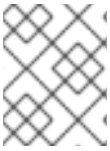
2. オプションで **Playbook** の構文を確認します。

```
# 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** パッケージをインストールします。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、シェルユーザーとして **tlog** セッションを冗長化します。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記録を表示および再生できるようにする **Cockpit** モジュールです。



注記

exclude_users 一覧に記載されているユーザー、または **exclude_groups** 一覧のグループに所属するユーザーの場合は、そのユーザーのセッションを記録できません。

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。*
- *CLI でデプロイされた tlog システムロールを使用したセッションの録画*

16.5. CLI でデプロイされた TLOG システムロールを使用したセッションの記録

指定したシステムに ***tlog*** *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記録できます。*

前提条件

- *ターゲットシステムに* ***tlog*** *システムロールをデプロイしている。*
- ***/etc/sssd/conf.d*** *ファイルに SSSD 設定ドロップファイルが作成されている。*

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 recordeduser
# passwd recordeduser
```

2. *上記で作成したユーザーとしてシステムにログインし直します。*

```
# ssh recordeduser@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. ***記録したユーザー*** *のパスワードを挿入します。*
   システムは、セッションを記録していることを示すメッセージを表示します。

```
ATTENTION! Your session is being recorded!
```

5. *セッションの記録が完了したら、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記録および保存され、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

ジャーナルで記録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. 記録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
```

16.6. CLI を使用した記録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記録を再生できます。

前提条件

- ユーザーセッションを記録している。「CLI でデプロイされた tlog システムロールを使用したセッションの録画」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記録を再生します。

```
# journalctl -o verbose -r
```

2. **tlog** 記録を検索します。

```
$ /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記録用のユーザー名
- **out_txt** フィールド (記録したセッションの 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 を使用して記録を再生します。

```
# tlog-play -r journal -M TLOG_REC=ID_number
```

これにより、ユーザーセッションの記録端末の出力が再生されることがわかります。

第17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

高可用性クラスター (HA クラスター) ロールはテクノロジープレビューとして利用できます。

現在、HA システムロールは制約をサポートしていません。制約を設定してからロールを実行すると、制約や、ロールでサポートされていない設定も削除されます。

現在、HA システムロールは SBD をサポートしていません。

17.1. 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** システムロールシステム Playbook の例については、「フェンスおよびリソースを使用した高可用性クラスターの設定」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** システムロールのシステムロール Playbook の例については、「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」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。

リソースのクローン設定を含む **ha_cluster** システムロールのシステムロール Playbook の例については、「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」を参照してください。

17.2. 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。

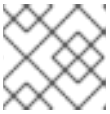
```
all:
  hosts:
    node1:
      ha_cluster:
        node_name: node-A
        pcs_address: node1-address
        corosync_addresses:
          - 192.168.1.11
          - 192.168.2.11
    node2:
      ha_cluster:
        node_name: node-B
        pcs_address: node2-address:2224
        corosync_addresses:
          - 192.168.1.12
          - 192.168.2.12
```

17.3. 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

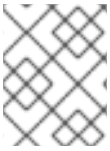
- Playbook を実行するノードに Red Hat Ansible Engine がインストールされている。



注記

Ansible 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。



注記

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. Playbook を実行します。

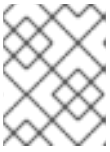
```
$ ansible-playbook -i inventory new-cluster.yml
```

17.4. 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 Red Hat Ansible Engine がインストールされている。



注記

Ansible Engine 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。

注記

***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. *Playbook を実行します。*

```
$ ansible-playbook -i inventory new-cluster.yml
```

17.5. 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 Red Hat Ansible Engine がインストールされている。*

注記

Ansible Engine 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。*
- *「Pacemaker クラスターで ext4 ファイルシステムを持つ LVM ボリュームの設定」で説明されているように、ext4 ファイルシステムを使用して LVM 論理ボリュームを設定している。*
- *「Apache HTTP サーバーの設定」で説明されているように、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。*

注記

***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. ***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していない場合は、**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します。*

```
[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):   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):   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、「現在のノードを優先するようにリソースを設定」を参照してください。

17.6. 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