Red Hat Satellite 6.3

ホストの管理

Red Hat Satellite 6 環境におけるホストの管理ガイド

Last Updated: 2018-07-18

Red Hat Satellite 6.3 ホストの管理

Red Hat Satellite 6 環境におけるホストの管理ガイド

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

法律上の通知

概要

本書は、Red Hat Satellite 環境でホストを設定し、これを使用する方法を説明します。この作業を行う前に、Red Hat Satellite 6 Server と、必要なすべての Capsule Server が正常にインストールされている必要があります。

目次

第1章 はじめに

本章では、ホスト、ホストの種類、実行できるアクションを説明します。

1.1. ホストの理解

ホストは、Red Hat Satellite が管理する Red Hat Enterprise Linux システムを指します。ホストは物理システムまたは仮想システムになります。KVM、VMware vSphere、OpenStack、Amazon EC2、Rackspace Cloud Services、Google Compute Engine、または Docker コンテナーなどの仮想ホストは、Red Hat Satellite がサポートするプラットフォームにデプロイできます。

1.2. ホスト管理の理解

Red Hat Satellite は規模を拡大してホストを管理できます。たとえば、監視、プロビジョニング、リモート実行、設定管理、ソフトウェア管理、およびサブスクリプション管理を行います。ホストの管理は、Red Hat Satellite Web UI またはコマンドラインから行えます。ホストの管理は「3章 *ホストの管理*」を参照してください。

第2章 コンテナーの管理

本章では、コンテナーの管理方法を説明します。コンテナーのコンテンツを設定する方法は **『コンテンツ管理ガイド』**の「コンテナーイメージの管理」を参照してください。

2.1. コンテナーホストの準備

前提条件

Red Hat Satellite では、コンテナーを Docker プロバイダータイプのコンピュートリソースにのみデプロイできます。そのため、コンテナーの表示または作成を初めて試みる時に、Satellite によって、Docker コンピュートリソースの作成を求めるプロンプトが出されます。これを行うには、コンテナーホストを作成し、このホストをコンピュートリソースとして指定します。

コンテナーホストの準備

1. Red Hat カスタマーポータルの『コンテナーの使用ガイド』ガイドの**「RHEL 7 における Docker の取得」**で説明している通り、イメージをホストする Red Hat Enterprise Linux 7 サーバーを準備し、このサーバーで **docker** サービスを有効にします。コンテナーホストは、Satellite Server と同じマシンにデプロイすることも、独立してデプロイすることもできます。

注記

現在、Red Hat Enterprise Linux 7 はコンテナーホストにサポートされる唯一のシステムです。**docker** パッケージは rhel-7-server-extras-rpms リポジトリーで利用できます。Red Hat Enterprise Linux 6 システムは、現在のところコンテナーをホストするシステムとしてサポートされていません。

2. コンテナーホストで、以下のコマンドを実行して、Satellite Server の CA 証明書をインストールします。

```
rpm -Uvh https://satellite.example.com/pub/katello-ca-consumer-
latest.noarch.rpm
```

ここで、**satellite.example.com** は Satellite Server の完全修飾ドメイン名です。コンテナーホストが Satellite ホストとしてすでに登録されている場合はこの手順を省略します。

3. コンテナーホストの場所に応じて、以下のタスクを実行します。

   a. コンテナーホストが Satellite Server と同じマシンにある場合には、以下を実行します。

      i. docker ユーザーグループを作成し、foreman ユーザーをそのグループに追加します。

```
# groupadd docker
# usermod -aG docker foreman
```

      ii. **/etc/sysconfig/docker** ファイルで、OPTIONS 変数を以下のように修正します。

```
OPTIONS='--selinux-enabled -G docker'
```

      iii. 影響を受けるサービスを再起動して変更を適用します。

```
# systemctl restart docker.service
# katello-service restart
```

b. コンテナーホストが Satellite Server とは異なるマシンにある場合は、以下を実行します。

i. コンテナーホストのポートを開き、Satellite Server と通信します。**/etc/sysconfig/docker** ファイルの OPTIONS 変数を以下のように修正します。

```
OPTIONS='--selinux-enabled -H tcp://0.0.0.0:2375 -H
unix:///var/run/docker.sock'
```

TLS が有効な場合はポート **2376** を使用できます。

ii. 以下のように docker サービスを再開し、設定を検証します。

```
# systemctl restart docker.service
# systemctl status docker.service
```

Docker コンピュートリソースの作成

1. ポート 5000 が Satellite Server で有効になっていることを確認します。コンテナーホストは、このポートを使用して Satellite Server のコンテンツビューからイメージをプルします。
2. **『プロビジョニングガイド』**の「Atomic Host 接続の Satellite Server への追加」 の説明通りに、コンピュートリソースを作成します。コンテナーホストのロケーションに従って、リソースの **URL** を指定します。
   a. コンテナーホストが Satellite Server と同じマシンにある場合は、**unix://var/run/docker.sock** をリソース URL として設定します。
3. コンテナーホストが Satellite Server とは異なるマシンにある場合は、以下の形式で URL を指定します。

```
http://container_host_fqdn:2375
```

ここで、**container_host_fqdn** はコンテナーホストの完全修飾ドメイン名を表し、Satellite との通信用にコンテナーホストで開くポート番号は **2375** (TLS を使用している場合は **2376**) にできます。

4. **テスト接続** をクリックして、コンテナーホストが利用できるかどうかをテストします。
5. **送信** をクリックして、コンピュートリソースを作成します。

2.2. コンテナーの作成

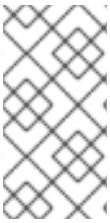
Satellite に Docker コンピュートリソースが 1 つ以上あると、コンテナーを作成することができます。新しいコンテナーを作成する方法は「コンテナーの作成」に記載される手順に従ってください。既存コンテナーをモニターする方法は「コンテナーのモニター」を参照してください。

コンテナーを作成するには、まずイメージをインポートする必要があります。イメージは、プラットフォームイメージまたは先に作成したレイヤー化したイメージである場合があります。Satellite は以下のイメージソースをサポートします。

- **ローカルコンテンツ**: コンテナーの作成時に **コンテンツビュー** オプションによって表されます。このオプションを使用すると、一部のコンテンツビューおよびライフサイクル環境で Capsule Server にすでに存在するリポジトリーからイメージをインポートできます。ローカル

レジストリーを作成し、これにデータを設定する方法は**『コンテンツ管理ガイド』**の「Red Hat Container Catalog からのコンテナーイメージのインポート」を参照してください。

- **Docker ハブ**: Docker ハブレジストリーを検索して、そこからイメージをプルできます。検証済みのコンテンツを含む、信頼されたイメージのみをプルするようにしてください。
- **外部レジストリー**: 先に作成した外部レジストリーからイメージをインポートできます。レジストリーを Red Hat Satellite で作成する方法は**『コンテンツ管理ガイド』**の「他のイメージレジストリーからのコンテナーイメージのインポート」を参照してください。



注記

既存コンテナーの設定を変更することはできません。設定を変更するには「コンテナーの作成」の説明に従って、変更した設定を含む交換用のコンテナーを作成する必要があります。そのため、実際のワークフローでコンテナーが交換可能であることを確認してください。

コンテナーの作成

1. **コンテナー** > **新規コンテナー** に移動するか、**コンテナー** > **すべてのコンテナー** に移動して **新規コンテナー**をクリックします。
2. コンテナー作成の **準備** 段階で、以下の設定を行います。
   - **コンピュートリソース** タブの **Deployed on** ドロップダウンメニューで、コンピュートリソースを選択します。コンピュートリソースの詳細は**『プロビジョニングガイド』**の「プロビジョニングワークフローの定義」を参照してください。
   - **ロケーション** タブで、新規コンテナーを利用可能にするロケーションを選択します。
   - **組織** タブで、新規コンテナーを利用可能にする組織を選択します。
     次へ をクリックして、先に進みます。
3. コンテナー作成の **イメージ** 段階では、コンテナーのベースとして機能するイメージをインポートします。これは、プラットフォームイメージの場合も、先に作成したレイヤー化されたイメージである場合もあります。以下のいずれかのオプションから選択してください。
   - **コンテンツビュー** タブを選択して、ライフサイクル環境からイメージをインポートします。ライフサイクル環境、コンテンツビュー、リポジトリー、タグ、および Capsule Server を指定します。
   - **Docker ハブ** タブを選択し、Docker ハブレジストリーからイメージをインポートします。**検索** フィールドにイメージ名を入力すると、Satellite はコンピュートリソースを自動的に検索します。眼鏡のアイコンをクリックして Docker ハブを検索します。検索結果の一覧からイメージを選択し、ドロップダウンリストからタグを選択します。
   - **外部レジストリー** タブを選択し、既存のレジストリーからイメージをインポートします。ドロップダウンメニューからレジストリーを選択し、イメージ名でこれを検索します。Satellite は **タグ** フィールドに、選択したイメージ名に使用できるタグを設定します。詳細は**『コンテンツ管理ガイド』**の「他のイメージレジストリーからのコンテナーイメージのインポート」を参照してください。
     次へ をクリックして、先に進みます。
4. コンテナー作成の **設定** 段階では、以下のパラメーターを設定します。
   - コンテナー名を指定します。

- コンテナー内で実行するコマンドを指定します。
- コンテナーが起動するとすぐに実行するコマンドのエントリーポイントを指定します。デフォルトのエントリーポイントは `/bin/sh -c` です。
- CPU をコンテナーに割り当てます。たとえば、`0-2,16` は CPU 0、1、2、および 16 を表します。
- コンテナーに使用できる CPU 時間の相対的配分を定義します。
- コンテナーのメモリー制限を指定します。たとえば、`512m` と指定すると、コンテナーのメモリー使用量が 512 MB に制限されます。
  次へ をクリックして、先に進みます。

5. **環境** というコンテナー作成の最終段階では、pseudo-tty を割り当てる必要があるかどうかを選択し、STDIN、STDOUT、および STDERR をコンテナーに割り当てます。**環境変数の追加** をクリックしてコンテナーのカスタム環境変数を作成します。**実行しますか?** チェックボックスを選択して、作成後のコンテナーを自動的に開始します。
6. **送信** をクリックして、コンテナーを作成します。

コンテナーの作成後に、Satellite はコンテナーのメタデータの概要を表示します。デフォルトで、新規コンテナーは無効になっています (コンテナーの作成時に **実行しますか?** チェックボックスを選択していない場合)。コンテナーの開始方法は「コンテナーの開始または停止」を参照してください。

> **例2.1 Satellite での Red Hat Enterprise Linux コンテナーの作成**
>
> Red Hat Enterprise Linux コンテナーを Red Hat Satellite で有効にするには、以下の操作を実行します。
>
> 1. **『コンテンツ管理ガイド』**の「Red Hat Container Catalog からのコンテナーイメージのインポート」に従ってカスタムレジストリーを作成します。**registry.access.redhat.com** をレジストリー URL として指定します。
> 2. 「コンテナーの作成」に従って新規コンテナーを作成します。コンテナー作成の **イメージ** 段階で、**外部レジストリー** タブに移動し、直前の手順で作成されたレジストリーを選択します。検索フィールドを使用して Red Hat Enterprise Linux イメージの必要なバージョンを検索します。**設定** および **環境** ステージに進み、コンテナーを完成します。

2.3. コンテナーのモニター

Red Hat Satellite では、コンテナーと、コンテナー内で実行するプロセスの状態をモニターすることができます。一部のコンテナーには **managed (管理)** というマークが付けられます。このマークは、そのコンテナーが Satellite 環境内で作成され、プロビジョニングされていることを意味します。

以下の手順では、選択した組織のコンテナーを一覧表示し、コンテナーのメタデータをモニターする方法を示します。

コンテナーの調査:

1. **コンテナー** > **すべてのコンテナー** に移動します。
2. **コンテナー** ページでは、すべての Docker コンピュートリソースに専用タブが付けられています。これらのタブのそれぞれには、各コンテナーの選択されたパラメーターと共に、利用可能なコンテナーの表が含まれています。検査するコンピュートリソースのタブを選択します。

3. コンテナーのメタデータを表示するには、検査するコンテナーの名前をクリックします。Satellite ではコンテナーのプロパティーの表が表示されます。

4. **プロセス** タブでは、コンテナーで現在実行しているプロセスを表示できます。プロセス名をクリックして、そのプロセスのメタデータを表示します。

5. コンテナーが実行している場合、その標準出力を **ログ** タブで確認できます。コンテナーの作成時に **pseudo-tty の割り当て** チェックボックスを選択している場合、コンソールは対話的になります。そうでない場合は、コンテナーの開始時に生成される初期の標準出力が表示されます。

2.4. コンテナーの開始、コミット、および削除

デフォルトで新規コンテナーは無効になります。コンテナーを有効にして、コンピュートリソースでコンテナー化されたアプリケーションのプロセスを開始します。次にホストは、Web アプリケーションの場合と同様にコンテナーと通信できます。以下の手順では、コンテナーを開始し、停止する方法を示します。

コンテナーの開始または停止

1. **コンテナー** > **すべてのコンテナー** に移動して、利用可能なコンテナーの一覧を表示します。

2. 開始するコンテナーの横にある **パワーオン** をクリックします。コンテナーが開始するとボタンが **パワーオフ** に切り替わり、これをクリックするとコンテナーが停止します。これらの操作は **`docker start`** コマンドおよび **`docker stop`** コマンドと同等です。

以下の手順では、コンテナーをコミットしてコンテナーの状態を保存する新規のイメージレイヤーを作成します。

コンテナーのコミット

1. **コンテナー** > **すべてのコンテナー** に移動して、利用可能なコンテナーの一覧を表示します。

2. コミットするコンテナーの名前をクリックします。

3. **コミット** をクリックします。Satellite は、以下を行うプロンプトを表示します。

   - リポジトリー名を指定します。これには 1 つの名前、または **user/my-rhel-image** などのようにユーザー名と組み合わせて指定できます。
   - タグをイメージに割り当てます。
   - 問い合わせ先情報を指定します。
   - イメージ情報のコメントを入力します。

4. **送信** をクリックします。

注記

コンテナーは元のイメージのリポジトリーにコミットされます。たとえば、コンテナーが Docker ハブからプルされたイメージをベースにしている場合、コミットされた変更は Docker ハブに戻されます。

コンテナーの削除

1. **コンテナー** > **すべてのコンテナー** に移動して、利用可能なコンテナーの一覧を表示します。
2. 削除するコンテナーの名前をクリックします。
3. **削除** をクリックします。
4. 警告ボックスで、**OK** をクリックしてコンテナーを削除します。

第3章 ホストの管理

本章は、ホストの作成、登録、管理、および削除を説明します。グループ、およびホストの環境を変更し、追加のネットワークインターフェースを設定し、特定の組織とロケーションにホストを割り当てます。

3.1. ホストの参照

Satellite Server Web UI では、Satellite Server が認識するすべてのホストを閲覧できます。画面上部の **ホスト** タブに移動し、以下の項目を含むドロップダウンメニューを開きます。

- **すべてのホスト**: Satellite Server が認識するホストの一覧です。
- **検出されたホスト**: Discovery プラグインによってプロビジョニングネットワークで検出されたベアメタルホストの一覧です。
- **コンテンツホスト**: コンテンツおよびサブスクリプションに関連付けられたタスクを管理するホストの一覧です。
- **ホストコレクション**: エラータのインストールなどの一括操作に使用されるユーザー定義のホストコレクションの一覧です。

ホストを検索する場合は、**検索** フィールドにそのホストを入力します。検索文字列の一部にアスタリスク (*) を使用できます。たとえば **dev-node.example.com** という名前のコンテンツホストを検索する場合は、**コンテンツホスト** ページをクリックし、**検索** フィールドに **dev-node*** と入力します。または、***node*** と入力してもコンテンツホスト **dev-node.example.com** が見つかります。

警告

Satellite Server が自己登録されない場合でも、ホストとして一覧に追加されます。ホストの一覧から Satellite Server を削除しないでください。

3.2. ホストのステータスタイプ

Satellite Server が認識する各ホストには、ホストに行われた最新のアクション、またはそのホストに適用される今後の変更に基づいて、状態のタイプが割り当てられます。**ホスト** → **すべてのホスト** に移動して、各ホストの状態を表示します。以下の表は、ホストに割り当てることができるステータスのタイプを示しています。

表3.1 ホストのステータスタイプ

アイコン	ステータス	説明
	エラー	ホストでエラーが検出されています。エラーアイコン上にマウスを置くと、エラーの原因を示すヒントが表示されます。ホストをクリックすると、問題の詳細なレポートを表示できます。

アイコン	ステータス	説明
	警告	ホストが設定されていますが、直近のレポート期間では、ホストのレポートが収集されませんでした。
	OK	ホストには保留中のアクションがなく、直近のレポート期間には保留中の変更やエラーがありません。

3.3. ホストの概要

ホストの概要ページには、指定されたホストと、ホストとインストーラー間の接続に関する情報が表示されます。ホストの概要ページを表示するには、**ホスト** → **すべてのホスト** を選択し、ホストの名前をクリックします。

詳細

詳細バーには、ホストの詳細情報へのショートカットを提供するボタンの列と、重要な詳細情報とイベントの概要を表示するタブが含まれます。

- **監査**: 現在のホストの監査エントリーが含まれるページです。
- **ファクト**: 現在のホストにおけるファクトの一覧が含まれるページです。このボタンは、インストーラーがホストのファクトを収集してからでないと選択できません。
- **レポート**: 現在のホストのレポートの一覧を含むページです。このボタンは、インストーラーがホストのレポートを収集してからでないと選択できません。
- **YAML**: ホストの IP アドレス、MAC アドレス、名前、ホストに適用されているパラメーターの値など、ホストに関する YAML 形式の詳細が記載されたページです。
- **プロパティー**: ホストの IP アドレス、MAC アドレス、ホストに適用されているオペレーティングシステムのエントリーなどのホストに関する一般的な詳細の一覧です。
- **メトリック**: ホストでレポートされている全イベントの概要を示す表です。
- **テンプレート**: ホストが現在アクセスできるプロビジョニングテンプレートの一覧です。この一覧に含まれるプロビジョニングテンプレートは、ホストに適用されるオペレーティングシステムのエントリーに基づいて自動的に設定されます。
- **NIC**: ホストに設定した NIC の詳細情報を示す表です。

ホストのアクション

以下のボタンのいずれかをクリックしてホストに対する共通のアクションを実行します。

- **リモートジョブのスケジュール**: ホストでのジョブの実行を許可します。ジョブの実行の詳細は「5章 *ホストでのリモートジョブの実行*」を参照してください。
- **ブートディスク**: ホストのブートディスクを選択できるメニューです。ホストのブート ISO を作成する方法は『**Red Hat Satellite プロビジョニングガイド**』の「PXE を使用しないプロビジョニングによる新規ホストの作成」を参照してください。

- **編集**: 設定するホストの詳細ページを開きます。インストーラーがすべての設定を自動的に行うため、通常は手動の設定が不要となることに注意してください。
- **ビルド**: 次回のホストの起動時にプロビジョニングするホストにフラグを設定します。インストーラーがプロビジョニングプロセスのすべての側面を管理するため、通常はホストを手動でプロビジョニングする必要がないことに注意してください。
- **削除**: ユーザーインターフェースからホストを削除します。

ホストのグラフ

ホストの概要ページには、ホストで実行した最新の Puppet 実行の状態を表示する 2 つのグラフが含まれます。

- **ランタイム**: **設定の取得** と **ランタイム** という 2 つのデータポイントを追跡します。 **設定の取得** データポイントは、指定した Puppet 実行時にホストの情報を収集するためにかかる時間を表し、**ランタイム** データポイントは、Puppet 実行にかかる時間を表します。これらのデータはどちらも秒単位で測定されます。
- **リソース**: Puppet 実行時にホストで実行するアクションの数を追跡します。このグラフに表示されるカテゴリーは **レポート** ページに表示されるカテゴリーと同じで、各カテゴリーのアクション数を使用して測定されます。

3.4. ホストの作成

以下の手順では、Red Hat Satellite でホストを作成する方法を説明します。

ホストの作成

1. **ホスト** > **ホストの作成** をクリックします。
2. **ホスト** タブで、必要な詳細を入力します。
3. **Puppet クラス** タブで、追加する Puppet クラスを選択します。
4. **インターフェース** タブで、以下を行います。
   a. 各インターフェースに対して、**アクション** コラムで **編集** をクリックし、必要に応じて以下を設定します。
      - **タイプ**: ボンドまたは BMC インターフェースに対して、**タイプ** リストで、インターフェースタイプを選択します。
      - **MAC アドレス**: MAC アドレスを入力します。
      - **DNS 名**: DNS サーバーに認識させる DNS 名を入力します。これは、完全修飾ドメイン名 (FQDN) のホスト部分に使用されます。
      - **ドメイン**: プロビジョニングネットワークのドメイン名を選択します。これにより、**サブネット** の一覧が自動的に更新され、適切なサブネットの選択肢が表示されます。
      - **IPv4 サブネット**: 一覧から、ホストの IPv4 サブネットを選択します。
      - **IPv6 サブネット**: 一覧から、ホストの IPv6 サブネットを選択します。
      - **IPv4 アドレス**: サブネットに対して IP アドレス管理 (IPAM) が有効な場合は、IP アドレスが自動的に提案されます。アドレスを入力することもできます。トークンのプロビ

ジョニングが有効な場合、ドメインが DNS を管理しない場合、サブネットが逆引き DNS を管理しない場合、またはサブネットが DHCP 予約を管理しない場合は、このアドレスを省略できます。

- **IPv6 アドレス**: サブネットに対して IP アドレス管理 (IPAM) を有効にした場合は、IP アドレスが自動的に提案されます。アドレスを入力することもできます。
- **管理**: このチェックボックスを選択すると、Capsule が提供する DHCP サービスおよび DNS サービスを使用してプロビジョニングを行う際にインターフェースを設定します。
- **プライマリー**: このチェックボックスを選択すると、このインターフェースの DNS 名を、FQDN のホスト部分に使用します。
- **プロビジョニング**: このチェックボックスを選択すると、プロビジョニングにこのインターフェースを使用します。つまり、このインターフェースを使用して TFTP ブートが行われ、そしてイメージをベースにしたプロビジョニングでは、プロビジョニングを実行するスクリプトにこのインターフェースが使用されます。**anaconda** による RPM のダウンロードなど、**%post** スクリプトの Puppet 設定は、プライマリーインターフェースを使用します。
- **仮想 NIC**: このインターフェースが物理デバイスではない場合は、このチェックボックスを選択します。この設定にはオプションが 2 つあります。
  - **タグ**: 任意で VLAN タグを設定します。設定していない場合はサブネットの VLAN ID となります。
  - **割当先**: この仮想インターフェースが割り当てられるインターフェースのデバイス名を入力します。

b. **OK** をクリックして、インターフェース設定を保存します。

c. オプションとして、**インターフェースの追加** をクリックし、追加ネットワークインターフェースを組み込みます。詳細は「追加のネットワークインターフェースの設定」を参照してください。

d. **送信** をクリックして変更を適用して、終了します。

5. **オペレーティングシステム** タブに必要な詳細を入力します。ドロップダウンリストからパーティションテーブルを選択するか、**ディスク (Custom partition table)** フィールドにカスタムパーティションテーブルを入力できます。両方を指定することはできません。

6. **パラメーター** タブで、**パラメーターの追加** をクリックし、必要なパラメーターを追加します。たとえば、すべての Puppet クラスパラメーターや、ホストに関連付けられているホストパラメーターが挙げられます。

7. **追加情報** タブに、ホストに関する追加情報を入力します。

8. **送信** をクリックして、プロビジョニングリクエストを完了します。

3.5. 登録

本セクションでは、Satellite Server または Capsule Server にホストを登録する方法を説明します。ホストを登録する方法は主に 2 つあります。

- コンシューマー RPM (**server.example.com**/pub/katello-ca-consumer-latest.noarch.rpm) をダウンロードしてインストールし、サブスクリプションマネージャーを実行します。この方法はホ

ストを新規インストールした時に適しています。詳細は「ホスト登録の設定」および「ホストの登録」を参照してください。

- ブートストラップスクリプト (**server.example.com**/pub/bootstrap.py) をダウンロードして実行します。この方法は、ホストを新規インストールした時と、たとえば Satellite 5 や別の Satellite 6 にすでに登録している場合に適しています。詳細は「ブートストラップスクリプトを使用したホストの Satellite 6 への登録」を参照してください。

キックスタートのインストール後のフェーズ、または端末から Red Hat Subscription Manager を介して Satellite Server に登録したホストは、 **ホスト** > **コンテンツホスト** からアクセスできる **コンテンツホスト** ページに表示されます。Satellite Server によってプロビジョニングされたホストは、**ホスト** > **すべてのホスト** からアクセスできる **ホスト** ページに表示されます。

3.5.1. ホスト登録の設定

Red Hat Enterprise Linux ホストは、デフォルトでカスタマーポータルのサブスクリプション管理に登録されます。それぞれのホストの設定を更新して、適切な Satellite Server または Capsule Server から更新を受け取れるようにする必要があります。

ホストでサポート対象のオペレーティングシステム

ホストは以下の Red Hat Enterprise Linux バージョンを使用している必要があります。

- 5.7 以降
- 6.1 以降*
- 7.0 以降

注記

Red Hat Enterprise Linux 6.1、6.2、および 6.3 では、Red Hat Subscription Manager の **`--auto-attach`** 機能をサポートしません。この問題の詳細は 「subscription-manager が "error: no such option: --auto-attach" を返す 」 を参照してください。

サポートされるアーキテクチャー

Red Hat Enterprise Linux のすべてのアーキテクチャーがサポートされます。

- i386
- x86_64
- s390x
- ppc_64

前提条件

- Satellite Server、任意の Capsule Server、およびすべてのホストが同じ NTP サーバーで同期されていることを確認します。
- Satellite Server、任意の Capsule Server、およびホストで時刻同期ツールが有効になっており、実行していることを確認します。
  - Red Hat Enterprise Linux 6 の場合:

```
# chkconfig ntpd on; service ntpd start
```

- Red Hat Enterprise Linux 7 の場合:

```
# systemctl start chronyd; systemctl enable chronyd
```

- **rhsmcertd** デーモンがホストで実行していることを確認します。
  - Red Hat Enterprise Linux 6 の場合:

```
# service rhsmcertd start
```

  - Red Hat Enterprise Linux 7 の場合:

```
# systemctl start rhsmcertd
```

以下の手順では、ホストを Red Hat Satellite に登録する設定方法を示します。

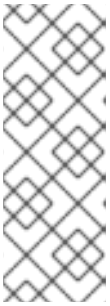
ホストを登録するための設定

1. Satellite Server または Capsule Server の完全修飾ドメイン名 (FQDN) をメモしておきます (例: **server.example.com**)。
2. 端末で、root ユーザーとしてホストに接続します。
3. コンシューマー RPM を、ホストを登録する Satellite Server または Capsule Server からインストールします。コンシューマー RPM は、ホストのコンテンツソースのロケーションを更新し、ホストが、Red Hat Satellite に指定したコンテンツソースからコンテンツをダウンロードできるようにします。

```
# rpm -Uvh http://server.example.com/pub/katello-ca-consumer-
latest.noarch.rpm
```

重要

実行中の Docker デーモンが再起動します。



注記

RPM パッケージが署名されていない場合は、必要に応じて、**--nosignature** オプションを使用してパッケージをインストールします。**katello-ca-consumer-hostname-1.0-1.noarch.rpm** では、サーバーのホスト名を含む追加の katello-ca-consumer RPM が利用できます。**katello-ca-consumer-latest.noarch.rpm** rpm は常に最新のバージョンを反映します。どちらを使用しても達成できる目的は同じになります。

3.5.2. ホストの登録

前提条件

- 「ホスト登録の設定」の手順をすべて完了します。

- ホストに対して、適切なコンテンツビュー環境に関連しているアクティベーションキーが存在していることを確認します。存在していない場合は、**『コンテンツ管理ガイド』**の「アクティベーションキーの管理」を参照してください。アクティベーションキーでは、デフォルトで自動アタッチが有効になっています。この機能は、一般的にハイパーバイザーとして使用されるホストで使用されます。

- インストールされている **subscription-manager** ユーティリティーのバージョンが 1.10 以上であることを確認します。パッケージは標準の Red Hat Enterprise Linux リポジトリーで利用できます。

ホストの登録

1. 端末で、root ユーザーとしてホストに接続します。

2. Red Hat Subscription Manager (RHSM) に関連するすべての古いホストデータをクリアします。

```
# subscription-manager clean
```

3. RHSM を使用してホストを登録します。

```
# subscription-manager register --org your_org_name \
--activationkey your_activation_key
```

例3.1 登録後のコマンド出力:

```
# subscription-manager register --org MyOrg --activationkey
TestKey-1
The system has been registered with id: 62edc0f8-855b-4184-b1b8-
72a9dc793b96
```

注記

アクティベーションキーで定義したコンテンツビューとライフサイクル環境を上書きするには、**--environment** オプションを使用します。たとえば、「開発」ライフサイクル環境のコンテンツビュー「MyView」にホストを登録するには、以下を実行します。

```
# subscription-manager register --org your_org_name \
--environment Development/MyView \
--activationkey your_activation_key
```

注記

Red Hat Enterprise Linux 6.3 ホストの場合、リリースバージョンはデフォルトで Red Hat Enterprise Linux 6 Server になり、これが 6.3 リポジトリーに設定されている必要があります。

Red Hat Enterprise Linux 6.3 がこのリポジトリーに設定されているようにするには、以下を実行します。

1. Red Hat Satellite で、**ホスト** > **コンテンツホスト** を選択します。
2. 変更するホストの名前をクリックします。
3. **コンテンツホストのコンテンツ** セクションで、**リリースバージョン** の右側にある編集アイコンをクリックします。
4. **リリースバージョン** ドロップダウンメニューで、"6.3" を選択します。
5. **保存** をクリックします。

3.5.3. Katello エージェントのインストール

以下の手順は、Satellite 6 に登録されているホストに Katello エージェントをインストールする方法を示しています。`katello-agent` パッケージは、**goferd** サービスを提供する `gofer` パッケージによって異なります。Red Hat Satellite Server または Capsule Server が、コンテンツホストに適用できるエラータに関する情報を提供できるように、このサービスを有効にしておく必要があります。

Katello パッケージのアップロードのトリガーとなって Satellite を更新する唯一のパッケージ管理コマンドは、 `yum` であることに注意してください。パッケージのインストールまたは更新に `rpm` コマンドは使用しないようにしてください。

前提条件

Satellite バージョン 6.1 以降では、**Red Hat Satellite Tools 6** リポジトリーを有効にする必要があります。**Red Hat Common** リポジトリーは使用されなくなっており、Satellite バージョン 6.1 以上との互換性はありません。

必要なパッケージを提供するため、**Red Hat Satellite Tools 6** リポジトリーを有効にし、Red Hat Satellite Server に同期してホストで利用できるようにする必要があります。

Red Hat Satellite Tools 6 リポジトリーが有効になっているのを確認

1. Satellite Web UI で、**コンテンツ** > **Red Hat リポジトリー** に移動し、**RPM** タブをクリックします。
2. **Red Hat Enterprise Linux Server** 項目を見つけ、展開します。
3. **Red Hat Satellite Tools 6 (for RHEL VERSION Server) (RPMs)** 項目を見つけ、展開します。**Red Hat Satellite Tools 6** 項目が表示されていない場合は、カスタマーポータルから取得したサブスクリプションマニフェストにその項目が含まれないことが原因として考えられます。この問題を修正するには、カスタマーポータルにログインし、これらのリポジトリーを追加し、サブスクリプションマニフェストをダウンロードして、Satellite にインポートします。
4. リポジトリーの名前の横にある **有効化** チェックボックスが選択されていることを確認します。選択されてない場合は選択します。

ホストで実行している、サポートされている全メジャーバージョンの Red Hat Enterprise Linux に対して、**Red Hat Satellite Tools 6** リポジトリーを有効にします。

Katello エージェントのインストール

1. ホストで **rhel-version-server-satellite-tools-6-rpms** リポジトリーが有効になっていることを確認します。自動アタッチが有効なアクティベーションキーを使用してホストを登録した場合は、リポジトリーが自動的に有効になっています。

```
# yum repolist enabled | grep -i rhel-version-server-satellite-
tools-6-rpms
```

rhel-version-server-satellite-tools-6-rpms が有効になっていない場合は、以下のコマンドで有効にします。

```
# subscription-manager repos --enable=rhel-version-server-satellite-
tools-6-rpms
```

2. 以下のコマンドを使用して、**katello-agent** RPM パッケージをインストールします。

```
# yum install katello-agent
```

3. **goferd** サービスが実行していることを確認します。
   - Red Hat Enterprise Linux 6 の場合は、以下のコマンドを実行します。

```
# service goferd start
```

   - Red Hat Enterprise Linux 7 の場合は、以下のコマンドを実行します。

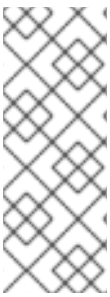
```
# systemctl start goferd
```

3.5.4. トレーサーのインストール

本セクションは、Red Hat Satellite 6.3 とトレーサーを統合する方法と、トレーサーをインストールしてトレースにアクセスする方法を説明します。

トレーサーは、システムで実行しているプロセスの再起動が必要かどうかを特定する監視ツールです。

トレースは、古くなって再起動が必要なアプリケーションの一覧を表示します。トレースには、Satellite Web UI からアクセスできます。



注記

トレーサーと Satellite Server の統合はテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat サービスレベルアグリーメント (SLA) では完全にサポートされていません。これらは、機能的に完全でない可能性があり、実稼働環境での使用を目的とはしていませんが、近々発表予定のプロダクトイノベーションをリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

前提条件

Red Hat Satellite Tools 6.3 リポジトリーを有効にし、Red Hat Satellite Server に同期してホストで利用できるようにする必要があります。

トレーサーのインストール:

トレーサーは任意のコンポーネントであるため、Katello の残りのホストツールとは別にインストールする必要があります。

1. コンテンツホストで **katello-host-tools-tracer** RPM パッケージをインストールします。

```
# yum install katello-host-tools-tracer
```

2. 以下のコマンドを入力します。

```
# katello-tracer-upload
```

3. トレーサーがインストールされていることを確認します。
   a. Satellite Web UI で **ホスト** > **すべてのホスト** に移動して、必要なホスト名をクリックします。
   b. **プロパティー** タブの **プロパティー** 表で、トレースの項目を確認します。**プロパティー** 表でトレース項目が見つからない場合は、トレースがインストールされていません。

トレースへのアクセス

トレースは、Satellite web UI のトレーサーによって生成されます。

1. Satellite Web UI で **ホスト** > **コンテンツホスト** に移動して、必要なホスト名をクリックします。
2. **トレース** タブをクリックして、表示されているトレースを表示します。

トレーサーツールの詳細は Red Hat ナレッジベースの記事「Satellite 6.3 Feature Overview: Tracer [Tech Preview]」を参照してください。

3.5.5. Puppet エージェントのインストールおよび設定

本セクションは、ホストに Puppet エージェントをインストールして設定する方法を説明します。Puppet エージェントを適切にインストールして設定している場合は、**ホスト** > **すべてのホスト** に移動して、Red Hat Satellite Server に表示されるホストの一覧を表示します。

前提条件

必要なパッケージを提供するため、**Red Hat Satellite Tools 6** リポジトリーを有効にし、Red Hat Satellite Server に同期してホストで利用できるようにする必要があります。

Red Hat Satellite Tools 6 リポジトリーが有効になっているのを確認

1. Satellite Web UI で、**コンテンツ** > **Red Hat リポジトリー** に移動し、**RPM** タブをクリックします。
2. **Red Hat Enterprise Linux Server** 項目を見つけ、展開します。
3. **Red Hat Satellite Tools 6 (for RHEL VERSION Server) (RPMs)** 項目を見つけ、展開します。**Red Hat Satellite Tools 6** 項目が表示されていない場合は、カスタマーポータルから取得した

サブスクリプションマニフェストにその項目が含まれないことが原因として考えられます。この問題を修正するには、カスタマーポータルにログインし、これらのリポジトリーを追加し、サブスクリプションマニフェストをダウンロードして、Satellite にインポートします。

4. リポジトリーの名前の横にある **有効化** チェックボックスが選択されていることを確認します。選択されてない場合は選択します。

Puppet エージェントのインストールおよに有効化

1. ホストで端末コンソールを開き、**root** ユーザーでログインします。
2. 以下のコマンドを使用して、**rhel-version-server-satellite-tools-6-rpms** リポジトリーが有効になっていることを確認します。

```
# yum repolist enabled | grep -i rhel-version-server-satellite-
tools-6-rpms
```

rhel-version-server-satellite-tools-6-rpms が有効になっていない場合は、以下のコマンドで有効にします。

```
# subscription-manager repos --enable=rhel-version-server-satellite-
tools-6-rpms
```

3. 以下のコマンドを使用して Puppet エージェントの RPM パッケージをインストールします。

```
# yum install puppet
```

4. 起動時に開始する Puppet エージェントを設定します。
   a. Red Hat Enterprise Linux 6 の場合:

```
# chkconfig puppet on
```

   b. Red Hat Enterprise Linux 7 の場合:

```
# systemctl enable puppet
```

前提条件

Puppet エージェントを設定するには、以下の条件を満たしている必要があります。

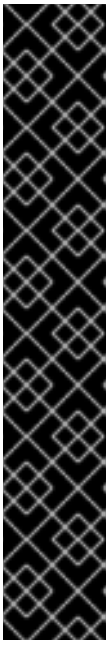
- ホストを Red Hat Satellite Server に登録しておく必要があります。
- Red Hat Satellite Tools 6 リポジトリーが有効になっている必要があります。
- Puppet パッケージがホストにインストールされている必要があります。

Puppet エージェントの設定

1. **/etc/puppet/puppet.conf** ファイルでサーバーおよび環境の設定を指定して、Puppet エージェントを設定します。

```
# vi /etc/puppet/puppet.conf
```

```
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
    ssldir = /var/lib/puppet/ssl

...

[agent]
    # The file in which puppetd stores a list of the classes
    # associated with the retrieved configuratiion.  Can be loaded
in
    # the separate ``puppet`` executable using the ``--loadclasses``
    # option.
    # The default value is '$confdir/classes.txt'.
    classfile = $vardir/classes.txt
    pluginsync = true
    report = true
    ignoreschedules = true
    daemon = false
    ca_server = satellite.example.com
    server = satellite.example.com
    environment = KT_Example_Org_Library_RHEL6Server_3

    # Where puppetd caches the local configuration.  An
    # extension indicating the cache format is added automatically.
    # The default value is '$confdir/localconfig'.
    localconfig = $vardir/localconfig

...
```



重要

environment パラメーターを、ホストが属する Puppet 環境の名前に設定します。Puppet 環境は、ホストまたはホストグループに関連付けられる Puppet モジュールのコレクションです。

- ホストの Puppet 環境を確認するには、**ホスト** > **すべてのホスト** に移動して、ホストテーブルで **Environment** コラムを確認します。
- Puppet 環境をホストに割り当てるには、**ホスト** > **すべてのホスト** に移動して、選択したホストの横にある **編集** をクリックします。
- Satellite Server で有効になっている Puppet 環境を一覧表示するには、**設定** > **環境** に移動します。さらに Satellite Server の **/etc/puppet/environments/** ディレクトリーで、Puppet 環境に関連付けられている Puppet モジュールおよびマニフェストを検索することもできます。

詳細は『Red Hat Satellite Puppet ガイド』を参照してください。

2. ホスト上で Puppet エージェントを実行します。

```
# puppet agent -t --server satellite.example.com
```

3. Satellite Server Web UI から、Puppet クライアントの SSL 証明書に署名します。
   a. Web UI から Satellite Server にログインします。
   b. **インフラストラクチャー** > **Capsule** に移動します。
   c. 必要な Capsule の右側にあるドロップダウンメニューから **証明書** を選択します。
   d. 必要なホストの右側にある **署名** をクリックします。
   e. **puppet agent** コマンドを再入力します。

```
# puppet agent -t --server satellite.example.com
```

注記

Puppet エージェントをホストに設定する場合に、ホストを組織やロケーションに割り当てられないため、**任意の組織** が選択されている場合にのみ、**すべてのホスト** の下に一覧表示されます。ホストへの組織の割り当ては「ホストの特定組織への割り当て」を参照し、ホストのロケーションへの割り当ては「ホストの特定ロケーションへの割り当て」を参照してください。

3.5.6. ブートストラップスクリプトを使用したホストの Satellite 6 への登録

6.2 以降に組み込まれているブートストラップを使用して、Satellite 6 に新規ホストを登録したり、既存ホストを移行したりできます。

ブートストラップスクリプトは、コンテンツの登録、製品の証明書、および Puppet 設定を処理します。ブートストラップスクリプトには、Red Hat Enterprise Linux システムの登録先 (RHN、Satellite 5、SAM、RHSM)、または登録されている場合でもシステムが Satellite 6 にサブスクライブしているかどうかにかかわらず、その Red Hat Enterprise Linux システムを対象とするという利点があります。

ブートストラップスクリプトのパッケージである **katello-client-bootstrap** は、デフォルトで Satellite Server のベースシステムにインストールされます。スクリプトは **/var/www/html/pub/** ディレクトリーにインストールされ、ホストで使用できるようになります。これには、以下の形式の URL からアクセスできます。

```
satellite6.example.com/pub/bootstrap.py
```

このスクリプトには readme ファイルがあります。Satellite CLI でこのファイルを表示するには、以下のコマンドを実行します。

```
$ less /usr/share/doc/katello-client-bootstrap-version/README.md
```

ホストへのブートストラップのインストール

このスクリプトが必要なのは一度だけで、さらには **root** ユーザー向けのものなので、これを **/root** に保存して使用後に削除するか、**/usr/local/sbin** に保存できます。この例では **/root** に保存しています。

root 権限で、以下のように、ブートストラップスクリプトをホストにインストールます。

1. 正しいディレクトリーにいることを確認します。たとえば、**/root** に変更するには、以下を実行します。

   ```
   # cd
   ```

2. スクリプトをダウンロードします。

   ```
   # curl -O http://satellite6.example.com/pub/bootstrap.py
   ```

   これにより、適切なディレクトリーにスクリプトをインストールします。

3. スクリプトを実行可能にします。

   ```
   # chmod +x bootstrap.py
   ```

4. スクリプトが実行できることを確認するには、以下のように使用ステートメントを表示します。

   ```
   # ./bootstrap.py -h
   ```

5. 移行プロセスが完了したら、スクリプトを削除することもできます。

   ```
   # cd
   # rm bootstrap.py
   ```

必要なパーミッション: このユーザーには、ブートストラップスクリプトを実行するパーミッションが必要です。パーミッションは、Web UI または Hammer コマンドラインツールを使用して設定できます。

Web UI を使用して、ブートストラップスクリプトのパーミッションを設定

1. **管理** > **ユーザー** に移動します。

2. このスクリプトを実行する既存のユーザーを選択するか、新しいユーザーを作成します。ユーザーは、**ユーザー名** フィールドにあります。選択したユーザーの情報を修正するタブが含まれる新しいペインが開きます。

3. **ロール** タブをクリックします。

4. **ロール** 一覧から、**Viewer** および **Edit hosts** を選択します。選択したロールは **選択された項目** 一覧に表示されるため、ロールが選択されていることを確認します。

警告

Edit hosts ロールでは、ホストの追加、編集、削除が可能になります。これが、セキュリティーポリシーを許可できない場合は、ブートストラップスクリプトで必要な最低限のパーミッションを持つ新しいロールを作成します。Hammer コマンドラインツールを使用して **bootstrap** ロールを作成するには「Hammer を使用してブートストラップスクリプトにパーティションを設定」を参照してください。このツールだけを、ブートストラップスクリプトを実行するユーザーに割り当てます。もしくは、Web UI で **管理** > **ロール** に移動して、適切なロールを作成します。その後、Web UI を使用してロールを作成し、適切なフィルターを設定します。

5. **送信** をクリックすると、ブートストラップスクリプトを実行するのに必要なパーミッションが、指定したユーザーに設定されます。

Hammer を使用してブートストラップスクリプトにパーティションを設定

1. ブートストラップスクリプトで最低限必要なパーミッションを持つロールを作成します。この例は、**Bootstrap** という名前のロールを作成します。必要に応じてこれを修正して、より具体的な名前を提供します。

```
$ ROLE='Bootstrap'
$ hammer role create --name "$ROLE"
$ hammer filter create --role "$ROLE" --permissions
view_organizations
$ hammer filter create --role "$ROLE" --permissions view_locations
$ hammer filter create --role "$ROLE" --permissions view_domains
$ hammer filter create --role "$ROLE" --permissions view_hostgroups
$ hammer filter create --role "$ROLE" --permissions view_hosts
$ hammer filter create --role "$ROLE" --permissions
view_architectures
$ hammer filter create --role "$ROLE" --permissions view_ptables
$ hammer filter create --role "$ROLE" --permissions
view_operatingsystems
$ hammer filter create --role "$ROLE" --permissions create_hosts
```

2. 既存のユーザーに新しいロールを割り当てます。

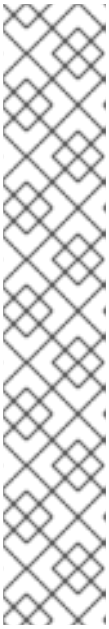
```
$ hammer user add-role --id user_id --role Bootstrap
```

新規ユーザーを作成して、新しいロールを割り当てるオプションもあります。Hammer からユーザーを作成する方法は『Red Hat Satellite Hammer CLI ガイド』の「ユーザーの作成」を参照してください。

ブートストラップスクリプトの実行

前提条件

- 前述のように、ブートストラップスクリプトがインストールされています。
- ホストのアクティベーションキーがあります。アクティベーションキーの設定方法は**『コンテンツ管理ガイド』**の「アクティベーションキーの管理」を参照してください。
- ホストグループを作成済みです。ホストグループの作成方法は**『プロビジョニングガイド』**の「Satellite Server でのホストグループの作成」を参照してください。



注記

Puppet が、**Production** 環境に作成した Puppet 環境に関連付けられているホストグループを持つホストの登録時に、Puppet CA 証明書を取得することができません。ホストグループに関連付ける適切な Puppet 環境を作成するには、以下の手順を行います。

1. 手動でディレクトリーを作成して、所有者を変更します。

```
# mkdir /etc/puppet/environments/example_environment
# chown apache
/etc/puppet/environments/example_environment
```

2. **設定** → **環境** へと移動し、**環境をインポート** をクリックします。ボタン名には、内部または外部の Capsule の FQDN が含まれます。
3. 作成したディレクトリーを選択し、**更新** をクリックします。

ブートストラップスクリプトの実行

1. ご使用の環境に適した値を使用して、ブートストラップコマンドを以下のように入力します。

警告

本セクションの例では、管理ユーザーを指定します。このユーザーがセキュリティーポリシーを許可できない場合は、適切なユーザーに追加できるブートストラップスクリプトで必要な最低限のパーティションを持つロールを新たに作成します。これは Web UI または Hammer で行います。詳細は「Web UI を使用して、ブートストラップスクリプトのパーミッションを設定」および「Hammer を使用してブートストラップスクリプトにパーティションを設定」を参照してください。

--server オプションの場合は、Satellite Server または Capsule Server の FQDN 名を指

定します。オプションが **--location**、**--organization**、および **--hostgroup** の場合は、オプションへの引数として、ラベルではなく引用符で囲まれた名前を使用します。高度なユースケースは「詳細なブートストラップスクリプトの設定」を参照してください。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key
```

スクリプトは、**--login** オプションを使用して入力した Satellite ユーザー名に対応するパスワードの入力を求めるプロンプトを出します。

2. スクリプトが実行し、**stdout** に進捗の通知が送信されます。証明書の承認を求めるプロンプトが表示されるのを確認します。以下は例になります。

```
[NOTIFICATION], [2016-04-26 10:16:00], [Visit the UI and approve
this certificate via Infrastructure->Capsules]
[NOTIFICATION], [2016-04-26 10:16:00], [if auto-signing is
disabled]
[RUNNING], [2016-04-26 10:16:00], [/usr/bin/puppet agent --test -
-noop --tags no_such_tag --waitforcert 10]
```

ホストは、管理者が Puppet 証明書を承認するまで無期限に待機します。

   a. Web UI で、**インフラストラクチャー** > **Capsule** に移動します。

   b. **--server** オプションで指定した FQDN に対応する Capsule 名の右側にある **証明書** を選択します。

   c. **アクション** コラムで、**署名** を選択して、ホストの Puppet 証明書を承認します。

   d. ホストに戻り、残りのブートストラップ処理が完了するのを確認します。

3. Web UI で **ホスト** > **すべてのホスト** に移動して、そのホストが、適切なホストグループに接続していることを確認します。

Katello エージェントがホストにインストールされていない場合は、「Katello エージェントのインストール」に進みます。

3.5.7. 詳細なブートストラップスクリプトの設定

ブートストラップスクリプトを使用する標準ワークフローについては、「ブートストラップスクリプトの実行」で要点を説明しています。本セクションでは、別の例を示します。

警告

本セクションの例では、管理ユーザーを指定します。このユーザーがセキュリティーポリシーを許可できない場合は、ブートストラップスクリプトで必要な最低限のパーティションを持つロールを新たに作成します。詳細は「Web UI を使用して、ブートストラップスクリプトのパーミッションを設定」および「Hammer を使用してブートストラップスクリプトにパーティションを設定」を参照してください。

Satellite 6 から別の Satellite 6 へのホストの移行

--force でこのスクリプトを使用します。スクリプトは、古い Satellite から katello-ca-consumer-* パッケージを削除して、新しい Satellite から katello-ca-consumer-* パッケージをインストールします。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--force
```

Red Hat Network (RHN) または Satellite 5 から Satellite 6 へのホストの移行

ブートストラップスクリプトは、**/etc/syconfig/rhn/systemid** と、RHN への有効な接続がシステムがレガシープラットフォームに登録されているインジケーターとしてあることを検出します。その後、スクリプトが **rhn-classic-migrate-to-rhsm** を呼び出して、RHN からシステムを移行します。デフォルトでは、スクリプトは、監査の理由により、システムのレガシープロファイルを削除しません。レガシープロファイルを削除するには、**--legacy-purge** および **--legacy-login** を使用して、プロファイルを削除するのに適切なパーミッションを持つユーザーアカウントを指定します。プロンプトが表示されたら、ユーザーアカウントのパスワードを入力します。以下は例になります。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--legacy-purge \
--legacy-login rhn-user
```

Satellite 6 にホストを登録し、Puppet 設定の省略

デフォルトでは、ブートストラップスクリプトは、コンテンツ管理および設定管理に対してホストを設定します。既存の設定管理システムがあり、ホストに puppet をインストールしたくない場合は **--skip-puppet** を使用します。以下は例になります。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
```

```
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--skip-puppet
```

ホストを Satellite 6 のコンテンツ管理のみに登録

システムをコンテンツホストとして登録し、プロビジョニングおよび設定管理機能を除外するには、**--skip-foreman** を使用します。以下は例になります。

```
# bootstrap.py --server satellite6.example.com \
--organization="Example Organization" \
--activationkey=activation_key \
--skip-foreman
```

ブートストラップスクリプトがコンシューマー RPM をダウンロードする方法を変更

デフォルトでは、ブートストラップスクリプトは HTTP を使用してコンシューマー RPM (**server.example.com**/pub/katello-ca-consumer-latest.noarch.rpm) をダウンロードします。環境によっては、ホストと Satellite との間のみ HTTPS を許可したい場合があります。**--download-method** を使用して、ダウンロード方法を HTTP から HTTPS へ変更します。以下は例になります。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--download-method https
```

ホストの IP アドレスを Satellite に提供

複数のインターフェースを持つホスト、または 1 つのインターフェースに複数の IP アドレスを持つホストでは、IP アドレスの自動検出を上書きし、特定の IP アドレスを Satellite に提供する必要がでてくる場合があります。**--ip** を使用してください。以下は例となります。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--ip 192.x.x.x
```

ホストでリモート実行を有効化

--rex および **--rex-user** を使用して、リモート実行を有効にし、指定したユーザーに必要な SSH キーを追加します。以下は例となります。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
```

```
--activationkey=activation_key \
--rex \
--rex-user root
```

登録時にホストにドメインの作成

ホストレコードを作成するには、スクリプトを実行する前に、ホストの DNS ドメインが Satellite に存在している必要があります。ドメインが存在しない場合は、以下のように **--add-domain** を使用して追加します。

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--add-domain
```

ホストの任意の完全修飾ドメイン名 (FQDN) の提供

ホストのホスト名が FQDN ではない場合、または RFC 準拠ではない (下線などの文字を含む) 場合、ホスト名の検証段階でスクリプトが失敗します。**--fqdn** を使用して、Satellite に報告される FQDN を指定します。指定するには、**hammer** を使用して **create_new_host_when_facts_are_uploaded** および **create_new_host_when_report_is_uploaded** を false に設定します。以下は例になります。

```
# hammer settings set \
--name  create_new_host_when_facts_are_uploaded \
--value false
# hammer settings set \
--name  create_new_host_when_report_is_uploaded \
--value false
```

```
# bootstrap.py --login=admin \
--server satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--fqdn node100.example.com
```

3.6. ホストのグループの変更

以下の手順では、ホストのグループを変更する方法を示します。

1. **ホスト** > **すべてのホスト** に移動します。
2. 変更するホストのチェックボックスを選択します。
3. 画面右上の **アクションの選択** メニューで、**グループの変更** を選択します。新規オプションのウィンドウが開きます。
4. **ホストグループ** メニューで、ホストに必要なグループを選択します。

5. **送信** をクリックします。

3.7. ホストの環境の変更

以下の手順は、ホストの環境を変更する方法を示します。

1. **ホスト** > **すべてのホスト** に移動します。
2. 変更するホストのチェックボックスを選択します。
3. 画面右上の **アクションの選択** メニューで、**環境の変更** を選択します。新規オプションのウィンドウが開きます。
4. **環境** メニューから、ホストの環境を選択します。
5. **送信** をクリックします。

3.8. ホストの管理

Satellite がプロビジョニングするホストは、デフォルトでは管理対象です。ホストを管理対象に設定した場合は、Satellite Server からホストの追加パラメーターを設定できるようになります。このような追加パラメーターの一覧は、**オペレーティングシステム** タブで確認できます。

オペレーティングシステム タブで変更した設定は、ビルドするホストを設定して再起動するまで有効になりません。

Satellite でサポートされていないオペレーティングシステムを使用するシステムの設定管理に関するレポートを取得する必要がある場合は、ホストを非管理対象にすることが推奨されます。

以下の手順では、ホストのステータスを管理または管理解除に切り替える方法を説明します。

1. **ホスト** > **すべてのホスト** に移動します。
2. ホストを選択します。
3. **編集** をクリックします。
4. **ホストの管理** または **ホストの管理解除** をクリックして、ホストのステータスを変更します。
5. **送信** をクリックして変更を保存します。

3.9. ホストの特定組織への割り当て

以下の手順では、ホストを特定の組織に割り当てる方法を説明します。組織に関する一般的な情報および設定方法は、**『コンテンツ管理ガイド』**の「組織の作成」を参照してください。

1. **ホスト** > **すべてのホスト** に移動します。
2. 変更するホストのチェックボックスを選択します。
3. 画面右上の **アクションの選択** メニューで、**組織の割り当て** を選択します。新規オプションのウィンドウが開きます。
4. **組織の選択** メニューに移動し、ホストに割り当てる組織を選択します。**Fix Organization on Mismatch (組織の不一致についての修正)** チェックボックスを選択します。

注記

ドメインやサブネットなど、ホストに関連付けられているリソースがある一方で、それらのリソースがホストの割り当て先の組織に割り当てられていない場合、不一致が生じます。**Fix Organization on Mismatch (組織の不一致についての修正)** オプションにより、このようなリソースが組織に追加されるため、このオプションは推奨される選択肢になります。一方、**Fail on Mismatch (不一致により失敗)** オプションを選択すると、常にエラーメッセージが生成されます。たとえば、別の組織へのホストの再割り当ては、設定に不一致がない場合でも失敗します。

5. **送信** をクリックします。

3.10. ホストの特定ロケーションへの割り当て

以下の手順では、特定のロケーションにホストを割り当てる方法を説明します。ロケーションの一般的な情報および設定方法は、**『プロビジョニングガイド』**の「ロケーションの作成」を参照してください。

1. **ホスト** > **すべてのホスト** に移動します。
2. 変更するホストのチェックボックスを選択します。
3. 画面右上の **アクションの選択** メニューで、**ロケーションの割り当て** を選択します。新規オプションウィンドウが開きます。
4. **ロケーションの選択** メニューに移動して、ホストに割り当てるロケーションを選択します。**Fix Location on Mismatch (ロケーションの不一致についての修正)** チェックボックスを選択します。

注記

ドメインやサブネットなど、ホストに関連付けられているリソースがある一方で、それらのリソースがホストの割り当て先の組織に割り当てられていない場合、不一致が生じます。**Fix Organization on Mismatch (組織の不一致についての修正)** オプションにより、このようなリソースが組織に追加されるため、このオプションは推奨される選択肢になります。一方、**Fail on Mismatch (不一致により失敗)** オプションを選択すると、常にエラーメッセージが生成されます。たとえば、別の組織へのホストの再割り当ては、設定に不一致がない場合でも失敗します。

5. **送信** をクリックして、ホストへのロケーションの割り当てを完了します。

3.11. 追加のネットワークインターフェースの設定

Red Hat Satellite は、1 台のホストに対して複数のネットワークインターフェースを指定することをサポートします。「ホストの作成」で説明されているように新規ホストを作成する場合や、既存ホストを編集する場合に、これらのインターフェースを設定することができます。

ホストに割り当てることができるネットワークインターフェースにはいくつかのタイプがあります。新規インターフェースを追加する場合は、以下のいずれかを選択してください。

- **インターフェース**: 追加の物理インターフェースまたは仮想インターフェースを指定できます。作成できる仮想インターフェースのタイプは 2 つあります。ホストが 1 つのインターフェース

で複数の (仮想) ネットワークと通信する必要がある場合は **VLAN** を使用します。これらのネットワークは互いにアクセスできません。もう 1 つのタイプの仮想インターフェースは **alias** です。これは既存のインターフェースに割り当てられる追加の IP アドレスです。詳細は「仮想インターフェースの追加」または「物理インターフェースの追加」を参照してください。

- **ボンド**: ボンディングインターフェースを作成します。NIC ボンドは複数のネットワークインターフェースを 1 つのインターフェースにバインディングして 1 つのデバイスと表示し、MAC アドレスを 1 つ持つ方法です。これにより、複数のネットワークインターフェースが 1 つのインターフェースとして機能し、帯域幅の拡大と冗長性を同時に提供します。詳細は「ボンディングインターフェースの追加」を参照してください。
- **BMC**: ベースボード管理コントローラー(BMC) により、マシンの物理的な状態をリモートで監視し、管理することができます。BMC の詳細は『**Red Hat Satellite インストールガイド**』の「管理対象ホスト上での電源管理の有効化」を参照し、 BMC インターフェースの設定方法は「ベースボード管理コントローラー (BMC) インターフェースの追加」を参照してください。

注記

追加のインターフェースには、デフォルトで **管理対象** フラグが有効になっています。これは、新規インターフェースが、選択したサブネットに関連付けられた DNS および DHCP Capsule Server によるプロビジョニング時に自動的に設定されることを意味します。これには、DNS および DHCP Capsule Server が適切に設定されたサブネットが必要です。ホストのプロビジョニングにキックスタートメソッドを使用する場合、管理対象インターフェースの設定ファイルはインストール後のフェーズで、**/etc/sysconfig/network-scripts/ifcfg-$interface_id** に自動的に作成されます。

注記

現在、仮想およびボンディングインターフェースには物理デバイスの MAC アドレスが必要です。そのため、これらのインターフェースの設定はベアメタルホストでのみ機能します。

3.11.1. 物理インターフェースの追加

以下の手順は、物理インターフェースをホストに追加する方法を示しています。

物理インターフェースの追加

1. **ホスト** > **すべてのホスト** に移動し、利用可能なホストを表示します。
2. 編集するホストの横の **編集** をクリックします。
3. **インターフェース** タブで、**インターフェースの追加** をクリックします。
4. **タイプ** メニューで、**インターフェース** オプションが選択されている状態にします。
5. 追加インターフェースの **MAC アドレス** を指定します。この設定は必須です。
6. **eth0**、**eth1.1** などのデバイスの **識別子** を指定します。識別子はボンディングインターフェース (**割り当て済みデバイス** フィールド (詳細は「ボンディングインターフェースの追加」を参照)、VLAN およびエイリアス (**割り当て先** フィールド (「仮想インターフェースの追加」を参照) に使用されます。
7. ホストの IP アドレスに関連付けられた **DNS 名** を指定します。Satellite は、選択したドメイン

(「DNS A」フィールド) に関連付けられた Capsule Server、および選択したサブネット (「DNS PTR」フィールド) に関連付けられた Capsule Server にこの名前を保存します。そのため、1 台のホストに複数の DNS エントリーを持たせることができます。

8. **ドメイン** ドロップダウンメニューからドメインを選択します。ドメインを作成して管理するには、**インフラストラクチャー** > **ドメイン** に移動します。

9. **サブネット** ドロップダウンメニューからサブネットを選択します。サブネットを作成して管理するには、**インフラストラクチャー** > **サブネット** に移動します。

10. インターフェースの **IP アドレス** を指定します。DHCP Capsule Server が割り当てられた管理対象インターフェースでは、DHCP リースを作成するためにこの設定が必要です。DHCP が有効になっている管理インターフェースでは、IP アドレスの自動補完が行われます。

11. インターフェースを管理するかどうかを決定します。**管理** チェックボックスが選択されていると、インターフェース設定はプロビジョニング時に関連付けられた Capsule Server からプルされ、DNS エントリーおよび DHCP エントリーが作成されます。キックスタートのプロビジョニングを使用している場合、設定ファイルはインターフェース用に自動的に作成されます。

12. **仮想 NIC** チェックボックスを選択して、仮想インターフェースを作成します。詳細は「仮想インターフェースの追加」を参照してください。

13. **OK** をクリックしてインターフェース設定を保存し、**送信** をクリックして、変更をホストに適用します。

3.11.2. 仮想インターフェースの追加

以下の手順は、ホストに追加の仮想インターフェースを設定する方法を示しています。これには、VLAN またはエイリアスインターフェースのいずれかを使用することができます。

エイリアスインターフェースは既存インターフェースに割り当てる追加の IP アドレスです。以下に注意してください。

- エイリアスインターフェースは割り当てられているインターフェースから MAC アドレスを自動的に継承するため、MAC アドレスを指定せずにエイリアスを作成できます。
- インターフェースは、ブートモードが **静的** に設定されているサブネットに指定する必要があります。

仮想インターフェースの追加

1. **ホスト** > **すべてのホスト** に移動し、利用可能なホストを表示します。

2. 編集するホストの横の **編集** をクリックします。

3. **インターフェース** タブで、**インターフェースの追加** をクリックします。

4. **タイプ** メニューで、**インターフェース** オプションが選択されている状態にします。

5. 一般的なインターフェース設定を指定します。適用できる設定オプションは、物理インターフェースのオプションと同じです (「物理インターフェースの追加」を参照)。
管理対象の仮想インターフェースの **MAC アドレス** を指定し、プロビジョニング用の設定ファイルが適切に生成されるようにします。ただし、**MAC アドレス** は、管理対象外の仮想インターフェースには不要です。

VLAN を作成する場合、**識別子** フィールドに **eth1.10** の形式で ID を指定します。エイリアスを作成する場合は、**eth1:10** の形式で ID を使用します。

6. **仮想 NIC** チェックボックスを選択します。仮想インターフェースに固有の追加設定オプションがその形式に追加されます。

   - **タグ**: ネットワークでより高いレベルのセグメント化を可能にするために、インターフェースごとにタグを指定できます。空白のままにすると、管理対象インターフェースは、このサブネットに VLAN ID が指定されている場合に、関連付けられたサブネットの VLAN ID からタグを継承します。このフィールドのユーザー定義のエントリーはエイリアスインターフェースでは適用されません。

   - **割当先**: 仮想インターフェースが属する物理インターフェースの識別子を指定します (例: eth1)。この設定は必須です。

7. **OK** をクリックしてインターフェース設定を保存し、**送信** をクリックして、変更をホストに適用します。

3.11.3. ボンディングインターフェースの追加

以下の手順は、ホストのボンディングインターフェースを設定する方法を示しています。

ボンディングインターフェースの追加

1. **ホスト** > **すべてのホスト** に移動し、利用可能なホストを表示します。

2. 編集するホストの横の **編集** をクリックします。

3. **インターフェース** タブで、**インターフェースの追加** をクリックします。

4. **タイプ** メニューから **Bond** を選択します。タイプ固有の設定オプションがその形式に追加されます。

5. 一般的なインターフェース設定を指定します。適用可能な設定オプションは、物理インターフェースのオプションと同じです (「物理インターフェースの追加」を参照)。ボンディングインターフェースは、**識別子** フィールドの **bond0** 形式のユーザー ID を使用します。**MAC アドレス** フィールドには MAC アドレスを 1 つ指定すれば十分です。

6. ボンディングインターフェースに固有の設定オプションを指定

   - **モード**: フォールトトレランスおよび負荷分散のポリシーを定義するボンドモードを選択します。各ボンドモードの簡単な説明は、表3.2「Red Hat Satellite で利用可能なボンディングモード」を参照してください。

   - **割り当て済みデバイス**: 割り当てられたデバイスの識別子のコンマ区切りの一覧を指定します。物理インターフェースまたは VLAN を指定できます。

   - **ボンドオプション**: 設定オプションのコンマ区切りの一覧を指定します (例: **miimon=100**)。ボンディングインターフェースに指定できるいくつかの設定オプションがあります。詳細は『Red Hat Enterprise Linux 7 ネットワークガイド』を参照してください。

7. **OK** をクリックしてインターフェース設定を保存し、**送信** をクリックして、変更をホストに適用します。

表3.2 Red Hat Satellite で利用可能なボンディングモード

ボンディングモード	説明
balance-rr	送受信は、ボンディングインターフェースで順次行われます。
active-backup	ボンディングインターフェースの中で最初に利用可能になったものから送受信が行われます。アクティブなボンディングインターフェースに障害がある場合に限り別のボンディングインターフェースが使用されます。
balance-xor	送信は選択されたハッシュポリシーに基づいて行われます。このモードでは、特定のピア用に宛先が指定されたトラフィックは、常に同じインターフェースで送信されます。
broadcast	すべての送信はすべてのボンディングインターフェースで行われます。
802.a3	同じ設定を共有するアグリゲーショングループを作成します。アクティブなグループのすべてのインターフェースで送受信が行われます。
balance-tlb	送信トラフィックが各ボンディングインターフェースの現在の負荷に応じて配分されます。
balance-alb	受信ロードバランシングは ARP (Address Resolution Protocol) ネゴシエーションにより実現されています。

3.11.4. ベースボード管理コントローラー (BMC) インターフェースの追加

本セクションでは、ベースボード管理コントローラー (BMC) インターフェースを、この機能をサポートするホストに設定する方法を説明します。

前提条件

次に進む前に、以下の前提条件を満たしていることを確認します。

- Capsule Server で BMC が有効になっています。必要に応じて「既存の Capsule Server で BMC パワー管理の有効化」を参照してください。
- `ipmitool` パッケージがインストールされています。
- ホストの MAC アドレス、IP アドレス、および BMC インターフェースのその他の詳細、およびこのインターフェースの適切な認証情報を確認します。

注記

BMC インターフェースが管理されている場合は BMC インターフェースの MAC アドレスのみが必要になります。これは DHCP 予約を作成するのに必要になります。

既存の Capsule Server で BMC パワー管理の有効化

1. **satellite-installer** ルーチンを使用し、以下のオプションを指定した以下のコマンドを実行して Capsule Server の BMC パワー管理を設定します。

```
# satellite-installer --foreman-proxy-bmc=true \ --foreman-proxy-
bmc-default-provider=ipmitool
```

2. Capsule Server の機能を更新します。

   a. Satellite Web UI にログインし、**インフラストラクチャー** > **Capsule** に移動します。

   b. 更新する必要がある機能がある Capsule Server を特定します。右側のドロップダウンリストで、**更新** をクリックします。**機能** コラムにある機能の一覧には BMC が含まれているはずです。

BMC インターフェースの追加

1. **ホスト** > **すべてのホスト** に移動し、利用可能なホストを表示します。
2. 編集するホストの横の **編集** をクリックします。
3. **インターフェース** タブで、**インターフェースの追加** をクリックします。
4. **タイプ** メニューから **BMC** を選択します。タイプ固有の設定オプションがその形式に追加されます。
5. 一般的なインターフェース設定を指定します。適用できる設定オプションは、物理インターフェースのオプションと同じです (「物理インターフェースの追加」を参照)。
6. BMC インターフェースに固有の設定オプションを指定します。
   - **ユーザー名**、**パスワード**: BMC で必要な認証情報を指定します。
   - **プロバイダー**: BMC プロバイダーを指定します。
7. **OK** をクリックしてインターフェース設定を保存し、**送信** をクリックして、変更をホストに適用します。

3.12. ホストの削除

以下の手順では、Red Hat Satellite からホストを削除する方法を説明します。

ホストの削除

1. **ホスト** > **すべてのホスト**、または **ホスト** > **コンテンツホスト** に移動します。
2. 削除するホストを選択します。
3. **アクションの選択** をクリックし、ドロップダウンメニューから **ホストの削除** を選択します。

4. **送信** をクリックして、Red Hat Satellite から永続的にホストを削除します。

警告

仮想マシンに関連付けられているホストのレコードが削除されている場合、仮想マシンも削除されます。このような状況で仮想マシンが削除されることを防ぐには、ハイパーバーザーから仮想マシンを削除せずに Satellite との関連付けを解除します。

仮想マシンをハイパーバイザーから削除せずに Satellite との関連付けを解除

1. Satellite Web UI で、**ホスト** > **すべてのホスト** に移動し、関連付けが解除されたホストの左側にあるチェックボックスを選択します。
2. **アクションの選択** ドロップダウンメニューから **ホストの関連付けを解除** ボタンを選択します。
3. 確認ウィンドウで、以下を行います。
   a. オプションで、チェックボックスを選択して、将来のためにホストを保持します。
   b. **送信** をクリックして変更を保存します。

第4章 ホストコレクションの設定

ホストコレクションは、複数のコンテンツホストのグループです。この機能により、一度に複数のホストで同じアクションを実行できます。この アクションにはパッケージおよびエラータのインストール、削除、更新や、割り当てているライフサイクル環境の変更、コンテンツビューの変更が含まれます。お客様の要件を満たすためにホストコレクションを作成できます。たとえば、職務、部署、事業単位でホストコレクションのホストを 1 つにまとめます。

4.1. ホストコレクションの作成

以下の手順では、ホストコレクションから作成する方法を示します。

ホストコレクションの作成

1. **ホスト** > **ホストコレクション** をクリックします。
2. **新規ホストコレクション** をクリックします。
3. ホストコレクションの名前を追加します。
4. **無制限のコンテンツホスト** を削除して、**制限** フィールドにホストの最大数を入力します。
5. ホストコレクションの説明を追加します。
6. **保存** をクリックします。

4.2. ホストコレクションのクローン作成

以下の手順では、ホストコレクションのクローンを作成する方法を示します。

ホストコレクションのクローンの作成

1. **ホスト** > **ホストコレクション** をクリックします。
2. 左側のパネルで、クローンを作成するホストコレクションをクリックします。
3. **コレクションのコピー** をクリックします。
4. クローン作成されたコレクションの名前を指定します。
5. **作成** をクリックします。

4.3. ホストコレクションの削除

以下の手順では、ホストコレクションを削除する方法を示します。

ホストコレクションの削除

1. **ホスト** > **ホストコレクション** をクリックします。
2. 削除するホストコレクションを選択します。
3. **削除** をクリックします。警告ボックスが表示されます。

> ホストコレクション **ホストコレクション名** を削除してもよろしいですか?

4. **削除** をクリックします。

4.4. ホストコレクションへのホストの追加

以下の手順では、ホストをホストコレクションに追加する方法を示します。

前提条件

ホストをホストコレクションに追加するためには、Red Hat Satellite に登録する必要があります。ホストを登録する方法は「登録」を参照してください。

ホストコレクションへのホストの追加

1. **ホスト** > **ホストコレクション** をクリックします。
2. ホストの追加先となるホストコレクションをクリックします。
3. **ホスト** タブで、**追加** サブタブを選択します。
4. テーブルから追加するホストを選択してから、**選択項目の追加** をクリックします。

4.5. ホストコレクションからのホストの削除

以下の手順では、ホストをホストコレクションから削除する方法を示します。

ホストコレクションからのホストの削除

1. **ホスト** > **ホストコレクション** をクリックします。
2. 必要なホストコレクションを選択します。
3. **ホスト** タブで、**一覧表示/削除** サブタブを選択します。
4. ホストコレクションから削除するホストを選択し、**選択項目の削除** をクリックします。

4.6. ホストコレクションへのコンテンツの追加

以下の手順は、Red Hat Satellite でコンテンツをホストコレクションに追加する方法を示しています。

4.6.1. パッケージのホストコレクションへの追加

以下の手順では、パッケージをホストコレクションに追加する方法を示します。

前提条件

- 追加するコンテンツが既存リポジトリーのいずれかで利用可能であるか、またはリポジトリーにない場合は、この手順を開始する前にリポジトリーに追加しておく必要があります。
- コンテンツはホストを割り当てる環境にプロモートする必要があります。

ホストコレクションにパッケージの追加

1. **ホスト** > **ホストコレクション** をクリックします。
2. パッケージの追加先となるホストコレクションをクリックします。
3. **コレクションの各種アクション** タブで、**パッケージのインストール、削除、および更新** をクリックします。
4. すべてのパッケージを更新するには、**すべてのパッケージの更新** ボタンをクリックして、デフォルトメソッドを使用します。または、ボタン右側のドロップダウンアイコンを選択して、使用するメソッドを選択します。**リモート実行 - 最初にカスタマイズ** メニューエントリーを選択すると、**ジョブ呼び出し** ページに移動します。ここでアクションをカスタマイズできます。
5. 必要に応じて、ラジオボタン **パッケージ** または **パッケージグループ** を選択します。
6. パッケージまたはパッケージグループの名前をフィールドに指定してから、以下のいずれかをクリックします。
   - **インストール**: デフォルトメソッドを使用して、新規パッケージをインストールします。または、ボタンの右側にあるドロップダウンアイコンを選択して、使用するメソッドを選択します。**リモート実行 - 最初にカスタマイズ** メニューエントリーを選択すると、 **ジョブ呼び出し** ページに移動します。ここでアクションをカスタマイズできます。
   - **更新**: デフォルトメソッドを使用して、ホストコレクションの既存のパッケージを更新します。または、ボタン右側のドロップダウンアイコンを選択して、使用するメソッドを選択します。**リモート実行 - 最初にカスタマイズ** メニューエントリーを選択すると、**ジョブ呼び出し** ページに移動します。ここでアクションをカスタマイズできます。

4.6.2. エラータのホストコレクションへの追加

以下の手順では、エラータをホストコレクションに追加する方法を示します。

前提条件

- 追加するエラータは既存リポジトリーのいずれかで利用可能であるか、またはリポジトリーにない場合は、この手順を開始する前にリポジトリーに追加しておく必要があります。
- エラータはホストの割り当てられる環境にプロモートする必要があります。

ホストコレクションへのエラータの追加

1. **ホスト** > **ホストコレクション** をクリックします。
2. エラータの追加先となるホストコレクションを選択します。
3. **コレクションの各種アクション** タブで、**エラータのインストール** をクリックします。
4. ホストコレクションに追加するエラータを選択し、**選択をインストール** ボタンをクリックして、デフォルトメソッドを使用します。または、ボタン右側のドロップダウンアイコンを選択して、使用するメソッドを選択します。**リモート実行 - 最初にカスタマイズ** メニューエントリーを選択すると、**ジョブ呼び出し** ページに移動します。ここでアクションをカスタマイズすることができます。

4.7. ホストコレクションからのコンテンツの削除

以下の手順では、パッケージをホストコレクションから削除する方法を示します。

コンテンツをホストコレクションから削除するには、以下を実行します。

1. **ホスト** > **ホストコレクション** をクリックします。
2. パッケージを削除するホストコレクションをクリックします。
3. **コレクションの各種アクション** タブで、**パッケージのインストール、削除、および更新** をクリックします。
4. 必要に応じて、ラジオボタン **パッケージ** または **パッケージグループ** を選択します。
5. パッケージまたはパッケージグループの名前をフィールドに指定します。
6. **削除** ボタンをクリックし、デフォルトメソッドを使用するパッケージまたはパッケージグループを選択します。または、ボタンの右側にあるドロップダウンアイコンを選択して、使用するメソッドを選択します。**リモート実行 - 最初にカスタマイズ** ニューエントリーを選択すると、**ジョブ呼び出し** ページに移動します。ここでアクションをカスタマイズできます。

4.8. ホストコレクションのライフサイクル環境またはコンテンツビューの変更

以下の手順では、ホストコレクションの割り当てられたライフサイクル環境またはコンテンツビューを変更する方法を示します。

ホストコレクションのライフサイクル環境またはコンテンツビューの変更

1. **ホスト** > **ホストコレクション** をクリックします。
2. ライフサイクル環境またはコンテンツビューを変更するホストコレクションを選択します。
3. **コレクションの各種アクション** タブで、**割り当て済みのライフサイクル環境またはコンテンツビューの変更** をクリックします。
4. ホストコレクションに割り当てるライフサイクル環境を選択します。
5. ドロップダウンリストから必要なコンテンツビューを選択します。
6. **割り当て** をクリックします。

注記

変更が反映するには約 4 時間かかります。ホストで変更を直ちに反映させるには、以下のコマンドを実行します。

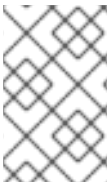
```
# subscription-manager refresh
```

リモート実行を使用すれば、複数のホストで同時にこのコマンドを実行できます。

第5章 ホストでのリモートジョブの実行

Red Hat Satellite は、任意のコマンドをホストで実行することをサポートします。これはリモート実行と呼ばれています。リモート実行は、Satellite Server ではデフォルトで有効になっていますが、必要な全 Capsule Server では手動で有効にする必要があります。通信は Capsule Server 経由で行われます。これは、Satellite Server にはターゲットホストへの直接のアクセスが不要であり、多数のホストを管理するために拡張可能であることを意味します。リモート実行は SSH サービスを使用するため、ターゲットホストで有効で、実行している必要があります。Capsule に、ターゲットホストのポート 22 にアクセスできることを確認してください。

コマンドはプロビジョニングテンプレートやパーティションテーブルと同様の方法でカスタマイズできます。デフォルトでいくつかのジョブテンプレートが含まれており、これらを使用してコマンドを実行できます。「ジョブテンプレートのセットアップ」を参照してください。



注記

Capsule Server のベースシステムは Satellite Server の内部 Capsule のクライアントであるため、このセクションは Capsule Server を含む Satellite Server に接続されるホストのすべてのタイプに適用されます。

コマンドは、一度に複数のホストに対して実行でき、デプロイメントに適した変数をコマンドで使用できます。変数の値は、ホストのファクト、Smart Class パラメーター、Smart 変数、またはホストパラメーターで設定できます。さらに、コマンドの実行時にテンプレートのカスタム値を指定できます。「ジョブの実行」を参照してください。

以下の一覧は、リモート実行の使用例を一部を示しています。

- ソフトウェアパッケージのインストール、更新、または削除
- 設定管理エージェントのブートストラップ
- Puppet、Salt、または Chef 実行のトリガー

デフォルトでは、それぞれの Capsule はリモート実行機能が無効にされた状態でインストールされます。Capsule Server でリモート実行を使用するには、これを有効にする必要があります。有効にするには、以下のコマンドを実行します。

```
# satellite-installer --scenario capsule \
--enable-foreman-proxy-plugin-remote-execution-ssh
```

Capsule Server でリモート実行が実行していることを確認するには、Web UI で **インフラストラクチャー** > **Capsule** に移動します。Capsule Server は、**SSH** が実行している **Features** コラムに一覧表示されます。

デフォルトで、Satellite Server は Katello エージェントではなくリモート実行を使用するように設定されています。必要な場合、これらの設定は、まずカスタムジョブテンプレートを作成し、次に Web UI の **管理** > **リモート実行機能** に移動してこれらの新規テンプレートを選択して変更できます。変更するそれぞれのアクションについて、ラベルを選択し、使用するジョブテンプレートを選択します。

5.1. リモートコマンドのセキュアな接続の確立

リモート実行に使用される SSH キーは Capsule のインストール時に自動的に作成され、設定は **/etc/foreman-proxy/settings.d/remote_execution_ssh.yml** ファイルにあります。設定には以下のオプションが含まれます。

ssh_identity_file

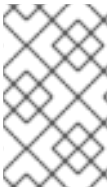
SSH キーをロードするファイルです。デフォルトでは **/usr/share/foreman-proxy/.ssh/id_rsa_foreman_proxy** に設定されています。

local_working_dir

リモート実行に必要なスクリプトを実行するために、Satellite または Capsule で使用されるディレクトリーです。デフォルトでは **/var/tmp** に設定されています。

remote_working_dir

リモート実行ジョブを実行するために使用されるクライアントシステムのディレクトリーです。デフォルトでは、**/var/tmp** に設定されています。



注記

クライアントシステムで **noexec** が **/var/** ボリュームまたはファイルシステムに設定されている場合は、 **`remote_working_dir`** を変更します。変更しないとスクリプトを実行できず、リモート実行ジョブが失敗します。

代替ディレクトリーの使用が必要な場合は、**new_place** などの新規ディレクトリーを作成し、デフォルトのディレクトリーから SELinux コンテキストをコピーします。以下は例になります。

```
# chcon --reference=/var new_place
```

SELinux ラベルの使用方法は、**『SELinux ユーザーおよび管理者のガイド』**の「SELinux ラベルの維持」を参照してください。

リモート実行のための SSH キーの配布

リモート実行を有効にするには、Capsule から管理するホストに、公開 SSH キーを配信します。ホストで SSH サービスが有効になっており、実行していることを確認します。ポート 22 にアクセスできるように、ネットワークまたはホストベースのファイアウォールを設定します。

Capsule からターゲットホストに公開鍵を配信する方法は 3 つあります。

- キーを手動で配布するには、Capsule で以下のコマンドを実行します。

  ```
  # ssh-copy-id -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub
  root@target.example.com
  ```

  ここで、**target.example.com** はターゲットホストのホスト名です。管理する各ターゲットホストに対して、これを繰り返し実行します。

  ターゲットホストにキーがコピーされたことを確認するには、Capsule で以下のコマンドを実行します。

  ```
  # ssh -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy
  root@target.example.com
  ```

- Satellite API を使用して公開鍵を Capsule から直接ダウンロードするには、それぞれのターゲットホストに対して以下のコマンドを実行します。

  ```
  # curl https://myproxy.example.com:9090/ssh/pubkey >>
  ~/.ssh/authorized_keys
  ```

  ここで、**myproxy.example.com** は Capsule のホスト名を表します。

- 公開鍵を新規にプロビジョニングされたホストに組み込むには、以下の行を含めるように **Kickstart default finish** テンプレートなどを変更します。

```
<%= snippet 'remote_execution_ssh_keys' %>
```

5.2. リモート実行に KERBEROS 認証の設定

Satellite 6.3 以降、Kerberos 認証を使用して、Satellite ホストでリモート実行に SSH 接続を確立できます。

前提条件

Red Hat Satellite でリモート実行に Kerberos 認証情報を使用する前に、ID 管理に Kerberos サーバーを設定し、以下の前提条件を完了していることを確認します。

- Kerberos サーバーの Satellite Server の登録
- Kerberos サーバーの Satellite ターゲットホストの登録
- リモート実行用に Kerberos ユーザーアカウントの設定および初期化
- Satellite の **foreman-proxy** ユーザーに、チケットを付与する有効な Kerberos チケットがあることを確認

ホストでリモート実行に Kerberos 認証情報を使用するように Satellite を設定するには、以下の手順を完了します。

1. Red Hat Bugzilla 1541481 が解決するまで、**tfm-rubygem-net-ssh-krb** パッケージをインストールする前に、一時的に SELinux を **permissive** に設定する必要があります。

```
# setenforce 0
```

2. **tfm-rubygem-net-ssh-krb** パッケージをインストールするには、以下のコマンドを入力します。

```
# yum install tfm-rubygem-net-ssh-krb
```

3. リモート実行に Kerberos 認証をインストールおよび有効にするには、以下のコマンドを入力します。

```
# satellite-installer --scenario satellite \
 --foreman-proxy-plugin-remote-execution-ssh-ssh-kerberos-auth true
```

4. SELinux を **enforcing** に設定します。

```
# setenforce 1
```

5. リモート実行のデフォルトユーザーを編集するには、Satellite Web UI で **管理** > **設定** に移動して、**リモート実行** タブをクリックします。**remote_execution_ssh_user** 行で 2 番目のコラムを編集し、Kerberos アカウントのユーザー名を追加します。

6. **remote_execution_effective_user** に移動し、2 番目のコラムを編集して、Kerberos アカウントのユーザー名を追加します。

Kerberos 認証が使用できることを確認するには、ホストでリモートジョブを実行します。

5.3. リモートコマンドの設定および実行

リモートホストで実行するコマンドはジョブテンプレートとして定義する必要があります。ジョブテンプレートを定義した後はこれを複数回使用することができます。

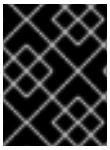
5.3.1. ジョブテンプレートのセットアップ

Satellite はジョブの実行に使用できる数多くのデフォルトジョブテンプレートを提供します。それらは、**ホスト** > **ジョブテンプレート** の下にあります。デフォルトテンプレートの中でそれぞれのニーズに合うテンプレートを見つけたら 「ジョブの実行」 に進みます。

デフォルトテンプレートは独自のテンプレートを作成するためのベースとして使用することもできます。デフォルトのジョブテンプレートは編集のためにロックされるため、テンプレートを変更するには最初にクローンを作成する必要があります。ジョブテンプレートは Embedded Ruby (ERB) 構文を使用します。詳細は「付録A *テンプレート作成の参照*」を参照してください。

ジョブテンプレートのクローンの作成

1. **ホスト** > **ジョブテンプレート** に移動します。
2. **新規ジョブテンプレート** をクリックします。または、**アクション** コラムのドロップダウンメニューで **クローン** を選択します。
3. ジョブテンプレートの設定
   a. **テンプレート** タブで、ジョブテンプレートの固有の名前を入力します。**デフォルト** を選択すると、テンプレートをすべての組織およびロケーションで利用可能にできます。**テンプレートエディター** を使用してテンプレートを手動で入力することも、**参照** をクリックしてテキストファイルからアップロードすることもできます。テンプレートは Embedded Ruby (ERB) テンプレート構文を使用します。詳細は「詳細テンプレートの作成」を参照してください。たとえば、パワー関連のアクションを実行するには詳細なテンプレートが必要になります。**Power Action - SSH Default** テンプレートをカスタムテンプレートに組み込む方法は例5.4「テンプレートにパワー操作を組み込む」を参照してください。
   b. **Job** タブで、ジョブカテゴリーを定義 (独自のカテゴリーを定義するか、表5.1「デフォルトのジョブテンプレートカテゴリー」 に一覧表示されているデフォルトカテゴリーから選択する) したり、実効ユーザーを定義したりできます。これらの設定は、ジョブの起動時にも設定できます (「リモートジョブの実行」を参照)。また、テンプレートコマンドの入力パラメーターを定義できます。これらのパラメーターはジョブの実行時に要求されます。
   c. 残りのタブでは、テンプレートのタイプ、組織、およびロケーションを設定したり、テンプレート履歴を表示したりできます。
4. **送信** をクリックします。ページの更新時に、新規のテンプレートがジョブテンプレートの一覧に表示されるはずです。



重要

ホストの編集ページの **パラメーター** タブでのみ表示できるパラメーターは、ジョブテンプレートの入力パラメーターとして使用できることに注意してください。

表5.1 デフォルトのジョブテンプレートカテゴリー

ジョブテンプレートのカテゴリー	説明
Packages	パッケージ関連のアクションを実行するためのテンプレートです。デフォルトで、インストール、更新、および削除のアクションが含まれています。
Puppet	ターゲットホストで Puppet を実行するためのテンプレートです。
Power	パワー関連のアクションを実行するためのテンプレートです。デフォルトで、再起動およびシャットダウンのアクションが含まれます。
Commands	リモートホストでカスタムコマンドを実行するためのテンプレートです。
Services	サービス関連のアクションを実行するためのテンプレートです。デフォルトで、開始、停止、再起動、およびステータスのアクションが含まれます。
Katello	コンテンツ関連のアクションを実行するためのテンプレートです。これらのテンプレートは主として Satellite Web UI の各種の場所で使用されます (たとえば、コンテンツホストの一括操作のための UI など) が、エラータのインストールなどの各種操作を実行するために個別に使用できます。

例5.1 restorecon テンプレートの作成

この例は、**Run Command - restorecon** というテンプレートを作成する方法を示します。これは、ターゲットホストで選択したディレクトリー内の全ファイルに対して、デフォルトの **SELinux** コンテキストを復元します。

1. **ホスト** > **ジョブテンプレート** に移動して、**新規ジョブテンプレート** をクリックします。
2. **名前** フィールドに **Run Command - restorecon** と入力します。**デフォルト** を選択して、テンプレートをすべての組織で利用できるようにします。以下のテキストを **テンプレートエディター** に追加します。

```
restorecon -RvF <%= input("directory") %>
```

<%= input("directory") %> の文字列は、ジョブの呼び出し時にユーザー定義のディレクトリーに置き換えられます。

3. **ジョブ** タブで、以下のアクションを実行します。
   a. **ジョブカテゴリー** を **Commands** に設定します。
   b. **入力を追加** をクリックして、ジョブのカスタマイズを可能にします。**名前** フィールドに **directory** と入力します。入力する名前は、**テンプレートエディター** で指定した値と一致している必要があります。

c. **必須** をクリックし、ユーザーがパラメーターを指定しなければコマンドが実行しないようにします。

d. **入力タイプ** ドロップダウンメニューから **ユーザー入力** を選択します。また、ジョブ呼び出しの際に表示される **説明** を指定します (例: **Target directory for restorecon**)。

4. **送信** をクリックします。

このテンプレートに基づいてジョブを実行する方法は、例5.2「複数のホストで restorecon テンプレートの実行」を参照してください。

5.3.2. ジョブの実行

このセクションでは、1 つ以上のホストに対してジョブテンプレートに基づくジョブを実行する方法を説明します。

リモートジョブの実行

1. **ホスト** > **すべてのホスト** に移動し、ジョブのターゲットホストを選択します。検索フィールドを使用してホストの一覧の範囲を制限することができます。

2. 画面右上の **アクションを選択** メニューから **リモートジョブのスケジュール** を選択し、**ジョブ呼び出し** ページに移動します。ターゲットがホスト 1 台である場合は、その名前をクリックし、ホスト情報ページで **リモートジョブのスケジュール** をクリックします。**実行** ボタンを使用して **ジョブテンプレート** ページからジョブを起動することもできます。

3. **ジョブ呼び出し** ページで、主なジョブ設定を定義します。

   a. 使用する **ジョブカテゴリー** および **ジョブテンプレート** を選択します。これらの設定は必須です。

   b. オプションとして、**ブックマーク** の一覧に保存された検索文字列を選択し、ターゲットホストを指定します。

   c. オプションとして、**検索クエリー** を入力し、ターゲットホストの範囲をさらに狭めることができます。**解決** 行には、クエリーの影響を受けるホストの数が表示されます。更新ボタンを押して、クエリーを変更した後の数を再計算します。プレビューアイコンにはターゲットホストが一覧表示されます。

   d. 残りの設定は、選択したジョブテンプレートによって異なります。カスタムパラメーターをテンプレートに追加する方法は「ジョブテンプレートのクローンの作成」を参照してください。

4. **詳細フィールドを表示** をクリックすると、ジョブの詳細設定が表示されます。一部の詳細設定はジョブテンプレートによって異なります。以下は一般的な設定です。

   - **実効ユーザー**: ジョブを実行するためにユーザーを定義します。デフォルトは SSH ユーザーです。

   - **同時実行レベル** 1 度に実行するジョブの最大数を定義します。これにより、大規模ホストでジョブを実行する際に、システムのリソースに負荷が過剰にかかるのを防ぐことができます。

- **タイムスパン** は、ジョブが終了していない場合に、強制終了するまでの間隔 (秒単位) で定義します。以前のタスクが終了するまで時間がかかりすぎているなど、定義した間隔で起動できなかったタスクはキャンセルされます。

- **クエリーのタイプ**: 検索クエリーが評価されるタイミングを定義します。これは、スケジュールされているタスクに対してクエリーが常に最新の状態に保つのに役立ちます。**同時実行レベル** 設定および **タイムスパン** 設定により、お使いのインフラストラクチャーハードウェアおよびニーズに合わせてジョブ実行を調整します。

5. ジョブをすぐに実行する場合は、**スケジュール** が **今すぐ実行** に設定されていることを確認します。さらに 1 回限りの将来のジョブを定義したり、再帰的に実行されるジョブを設定することもできます。再帰的に実行されるジョブについては、開始日と終了日、実行の回数と頻度を定義できます。また cron 構文を使用して繰り返しを定義することもできます。詳細は Red Hat Enterprise Linux 7 の『**システム管理者のガイド**』の「システムタスクの自動化」セクションを参照してください。

6. **送信** をクリックします。これにより **ジョブの概要** ページが表示され、ジョブの完了時にはジョブのステータスも表示されます。

例5.2 複数のホストで restorecon テンプレートの実行

以下の例では、例5.1「restorecon テンプレートの作成」 で作成されたテンプレートに基づいて、複数のホストでジョブを実行する方法を示します。このジョブは、**/home/** ディレクトリーの下にあるすべてのファイルで SELinux コンテキストを復元します。

1. **ホスト** > **すべてのホスト** に移動し、ターゲットホストを選択します。**アクションの選択** ドロップダウンメニューで、**リモートジョブのスケジュール** を選択します。

2. **ジョブ呼び出し** ページで、**Commands** ジョブカテゴリーを選択し、**Run Command - restorecon** ジョブテンプレートを選択します。

3. **directory** フィールドに、**/home** と入力します。

4. **スケジュール** を **今すぐ実行** に設定します。

5. **送信** をクリックします。**ジョブ呼び出し** ページに移動します。ここでジョブ実行のステータスを監視できます。

5.3.3. ジョブの監視

実行中のジョブの進捗を監視できます。これは、トラブルシューティングが必要になる場合に役立ちます。

ジョブを監視するには、以下を実行します。

1. ジョブのページに移動します。このページは、**今すぐ実行** が設定されているジョブをトリガーすると自動的に表示されます。スケジュールされたジョブを監視するには、**モニター** > **ジョブ** に移動して、検査するジョブ実行を選択します。

2. ジョブページで、**ホスト** タブをクリックします。これにより、ジョブが実行しているホストの一覧が表示されます。

3. **ホスト** 列で、検査するホストの名前をクリックします。これにより、ジョブの実行をリアルタイムでモニターできる **コマンドの詳細** ページが表示されます。

4. いつでも **ジョブに戻る** をクリックして、**ジョブの詳細** ページに戻ることができます。

5.3.4. 詳細テンプレートの作成

ジョブテンプレートの作成時に、既存テンプレートを **テンプレートエディター** フィールドにインポートできます。これはレンダリングと呼ばれています。これにより、テンプレートを組み合わせたり、一般的なテンプレートからより具体的なテンプレートを作成したりできます。

以下のテンプレートを使用してデフォルトのテンプレートを組み合わせ、Red Hat Enterprise Linux システムに **httpd** サービスをインストールして起動できます。

```
<%= render_template 'Package Action - SSH Default', :action => 'install',
:package => 'httpd' %>
<%= render_template 'Service Action - SSH Default', :action => 'start',
:service_name => 'httpd' %>
```

上記のテンプレートはレンダリングされるテンプレートのパラメーター値を直接指定します。ユーザーがジョブ実行時にレンダリングされたテンプレートへの入力を定義できるようにする **input()** メソッドを使用することもできます。たとえば、以下の構文を使用できます。

```
<%= render_template 'Package Action - SSH Default', :action => 'install',
:package => input("package") %>
```

上記のテンプレートを使用して、レンダリングされたテンプレートからパラメーター定義をインポートする必要があります。これを行うには、**ジョブ** タブに移動し、**外部入力セットを追加** をクリックし、**ターゲットテンプレート** ドロップダウンリストで、レンダリングされたテンプレートを選択します。すべてのパラメーターをインポートするか、コンマ区切りの一覧を指定することができます。

例5.3 restorecon テンプレートのレンダリング

この例は、例5.1「restorecon テンプレートの作成」で作成される **Run command - restorecon** テンプレートから派生するテンプレートを作成する方法を示しています。このテンプレートでは、ジョブ実行時にユーザーが入力する必要はなく、ターゲットホストの /**home**/ ディレクトリー下のすべてのファイルで SELinux コンテキストを復元します。

「ジョブテンプレートのセットアップ」に従って新規テンプレートを作成し、**テンプレートエディター** 画面で以下の文字列を指定します。

```
<%= render_template("Run Command - restorecon", :directory => "/home")
%>
```

例5.4 テンプレートにパワー操作を組み込む

以下の例では、再起動などのパワー操作を実行するためにジョブテンプレートをセットアップする方法を示します。この手順は、Satellite が再起動時に切断の例外をエラーとして解釈するのを防ぐため、ジョブのリモート実行が正常に機能します。

「ジョブテンプレートのセットアップ」に従って新規テンプレートを作成し、**テンプレートエディター** 画面で以下の文字列を指定します。

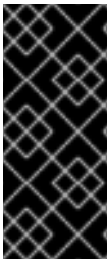
```
<%= render_template("Power Action - SSH Default", :action => "restart")
%>
```

5.4. グローバル設定

Satellite のリモート実行機能は、その動作を設定するために使用できる数多くのグローバル設定を提供しています。グローバル設定の一覧は表5.2「リモート実行用のグローバル設定」で紹介されています。これらの設定を確認し、更新するには、**管理** > **設定** に移動して、**リモート実行** タブをクリックします。

表5.2 リモート実行用のグローバル設定

パラメーター名	説明
remote_execution_effective_user	これは、任意のジョブに対するデフォルトの実効ユーザーです。ジョブが実行する際に、プロセスの実効ユーザーがそれに応じて変更されます (sudo による変更など)。このオプションはジョブテンプレートおよびジョブ呼び出し別にオーバーライドできます。
remote_execution_effective_user_method	ターゲットホストで実効ユーザーを設定するために使用するメソッドを指定します。現時点では su と sudo だけがサポートされています。
remote_execution_fallback_proxy	ホストでリモート実行が設定された Capsule を検索します。これはホストにサブネットがないか、またはサブネットにリモート実行が有効になっている Capsule がない場合に役立ちます。
remote_execution_global_proxy	ホストに割り当てられた Capsule 外にあるリモート実行 Capsule を検索します。ロケーションまたは組織が有効になっている場合、検索はホストの組織またはロケーションに限定されます。
remote_execution_ssh_user	Capsule が SSH でターゲットに接続する際に使用されるデフォルトユーザーです。`remote_execution_ssh_user` 変数を設定し、これをホスト別にオーバーライドできます。 この設定は、ホスト、ホストグループ、オペレーティングシステム、ドメイン、ロケーションまたは組織別に実行できます。さらに、`remote_execution_effective_user` とは異なるユーザーを指定することもできます。
remote_execution_sync_templates	データベースのシード処理時にジョブテンプレートをディスクと同期させるかどうかを定義します。



重要

/etc/foreman/settings.yaml 設定ファイルでグローバルパラメーターを設定できますが、このファイルに追加する手動の変更は、次に **`satellite-installer`** を実行したときに上書きされます。したがって、Red Hat では、このパラメーターを Web UI で変更することを推奨します。または、コンソールから **`foreman-rake config`** コマンドを使用することができます。

5.4.1. リモート実行用の Capsule の選択

リモート実行では、ホストに指定されたジョブを Capsule Server が実行することが必要になります。デフォルトで、**リモート実行プロバイダー** 機能が有効になったホストの組織およびロケーション内の Capsule は、これらのジョブを実行できると見なされています。**`remote_execution_global_proxy`** 変数を **`false`** に設定すると、この動作を無効にできます。これは、ネットワークの分離のためにすべての Capsule を使用できない場合など、より複雑な環境で必要になる場合があります。この設定では、Capsule のプールが各サブネットに割り当てられ、そのサブネット間でジョブの負荷が分散されます。

また、**`remote_execution_fallback_proxy`** 変数を **`true`** に設定してフォールバックモードを有効にすることもできます。この設定では、Capsule でもリモート実行が設定されている場合に、リモート実行に Puppet マスターなどのホストに関連付けられた Capsule が使用されます。

5.5. リモート実行用のパーミッションの委任

ターゲットにするホストを含め、インフラストラクチャー内で実行するジョブと、それを実行するホストを制御できます。リモート実行機能は 2 つの組み込みロールを提供します。

- **Remote Execution Manager (リモート実行マネージャー)**: このロールは、すべてのリモート実行機能および機能性へのアクセスを許可します。
- **Remote Execution User (リモート実行ユーザー)**: このロールは、ジョブの実行のみを許可します。ジョブテンプレートを変更するパーミッションは提供されません。

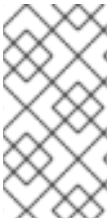
Remote Execution User (リモート実行ユーザー) ロールのクローンを作成し、そのフィルターをカスタマイズして詳細度を高めることができます。**`view_job_templates`** パーミッションでフィルターを調整する場合、ユーザーは一致するジョブテンプレートに基づいてジョブを確認し、トリガーすることのみが可能です。**`view_hosts`** パーミッションおよび **`view_smart_proxies`** パーミッションを使用すると、ロールに表示されるホストまたは Capsule を制限できます。

`execute_template_invocation` パーミッションはジョブの実行が開始する直前に確認される特殊なパーミッションです。このパーミッションは、特定のホストで実行できるジョブテンプレートを定義します。これにより、パーミッションの指定時に詳細度をさらに高めることができます。ロールおよびパーミッションの使用については、**『Red Hat Satellite の管理』**の「ロールの作成および管理」を参照してください。

以下の例は、**`execute_template_invocation`** パーミッションのフィルターを示しています。

```
name = Reboot and host.name = staging.example.com
name = Reboot and host.name ~ *.staging.example.com
name = "Restart service" and host_group.name = webservers
```

上記の例の最初の行により、選択したホストで **Reboot** テンプレートを実行できます。2 番目の行は、**.staging.example.com** で終わる名前を持つホストのプールを定義します。3 番目の行はテンプレートをホストグループにバインドします。



注記

ユーザーに割り当てるパーミッションは時間の経過と共に変更できます。ユーザーに将来実行するようにスケジュールされたジョブがあり、パーミッションが変更した場合、パーミッションがジョブ実行の直前にチェックされるため、これによる実行が失敗する可能性があります。

第6章 SATELLITE でのベアメタルホストの検出

Red Hat Satellite 6.3 には Discovery プラグインが同梱されています。Discovery プラグインはプロビジョニングネットワーク上の不明ホストのベアメタル検出を自動化します。これらの新規ホストは Satellite Server に登録され、クライアントの Puppet エージェントは、Facter が収集するシステムのファクト (シリアル ID、ネットワークインターフェース、メモリー、ディスク情報など) をアップロードします。登録後、ホストは Satellite Web UI の **検出されたホスト** ページに表示されます。事前に定義された検出ルールを使用して、プロビジョニングを手動 (Web UI、CLI、または API を使用) か、または自動で開始することができます。

Discovery プラグインは、プロビジョニングネットワークと Satellite Server インスタンスの両方に直接アクセスできる Satellite Capsule Server 経由で通信します。Satellite Server からホストを直接検出することができますが、Red Hat では以下の設定の使用を推奨しています。

```
Satellite Server (Satellite Server Discovery plug-in) <--> Satellite
Capsule (Satellite Capsule Discovery plug-in) <--> Discovered Host
(Satellite Discovery image)
```

Satellite Discovery プラグインは 3 つの異なるコンポーネントで構成されています。

Satellite Server Discovery プラグイン

これは Satellite Server で実行し、検出されたホストと使用できるように API および UI 機能を提供します。**`tfm-rubygem-foreman_discovery`** パッケージにはこのプラグインが含まれます。

The Satellite Capsule Server Discovery プラグイン

これは、プロビジョニングネットワークで検出されたホストと Satellite Server との間の通信プロキシーです。**`rubygem-smart_proxy_discovery`** パッケージにはこのプラグインが含まれます。

Satellite Discovery イメージ

これは、 Red Hat Enterprise Linux をベースとする最小オペレーティングシステムです。このオペレーティングシステムはホスト上で PXE で起動し、初期のハードウェア情報を取得し、Satellite Server にチェックインするために使用されます。検出されたホストは、Anaconda で再起動するまで Satellite Discovery イメージを実行し続けます。その後にプロビジョニングプロセスを開始します。**`foreman-discovery-image`** パッケージにはこのイメージが含まれます。これは TFTP サービスを提供する Satellite Capsule Server にインストールされる必要があります。

6.1. PXE ベースの検出に対するネットワーク設定

検出プロセスは PXE に基づいています。システムは、LAN または VLAN への 1 つの Ethernet 接続を使用して、ネットワークから起動できます。その他のすべてのネットワークインターフェース設定 (ボンディング、チーミング、ブリッジ、DSL、Wi-Fi など) はサポートされていません。

検出および PXE には、異なる LAN または VLAN が必要です。VLAN トランクを使用するようにシステムを設定することはできますが、プロビジョニング VLAN に正しい VLAN タグを持つプロビジョニングインターフェースも設定し、インストール後のスクリプトを使用して本番環境の VLAN へのタグを変更する必要があります。

検出したシステムが kexec を使用して、PXE 起動を完全に回避する Anaconda を使用して新しいカーネルをロードする 非 PXE モードで特別なネットワーク設定を使用することが技術的に可能であっても、検出イメージは現在このような設定を許可しません。検出の拡張やスクリプトを使用してネットワークの再設定は可能ですが、Satellite 6 検出プラグインは、このような設定では動作しません。

検出プロセスでは、ネットワークインターフェースを設定する可能性は現在制限されているため、そしてプロビジョニングインターフェースはプライマリーインターフェースでもあるため、設定を簡易にす

るには、プライマリーインターフェースと、本番環境で使用するインターフェースを分けます。Satellite 6 テンプレート機能は、必要に応じて、インターフェースを設定するインストール後のスクリプトをデプロイするのに使用できます。

6.2. SATELLITE DISCOVERY プラグインの設定

以下のセクションでは、Satellite Discovery プラグインを設定する方法と、Satellite Server で PXE 起動テンプレートを準備する方法を説明します。

6.2.1. Satellite Discovery イメージのデプロイメント

Satellite Discovery イメージを含むパッケージを、TFTP サービスを提供する Satellite Capsule Server (Satellite Server ではありません) にインストールします。

```
# yum install foreman-discovery-image
```

このパッケージには、Linux カーネルと、PXE が起動する検出されるホストに使用される起動可能な ISO ファイルとなる初期 RAM ディスクイメージが含まれます。以下のコマンドを実行してパッケージの内容を検査します。以下のような出力が生成されます。

```
$ rpm -ql foreman-discovery-image
/usr/share/foreman-discovery-image
/usr/share/foreman-discovery-image/fdi-image-rhel_7-2.1.0-20150212.1.iso
```

このパッケージのインストール時に、ISO ファイルのカーネルとイメージが TFTP ディレクトリーに抽出され、イメージおよびカーネルの最新バージョンへのシンボリックリンクが作成されます。PXE 起動のプロビジョニングテンプレートのシンボリックリンクを使用すると、**foreman-discovery-image** パッケージがアップグレードされるたびにテンプレートのバージョンを変更する必要がありません。以下は例になります。

```
$ find /var/lib/tftpboot/boot
/var/lib/tftpboot/boot
/var/lib/tftpboot/boot/fdi-image-rhel_7-2.1.0-20150212.1-img
/var/lib/tftpboot/boot/fdi-image-rhel_7-2.1.0-20150212.1-vmlinuz
/var/lib/tftpboot/boot/fdi-image-rhel_7-img
/var/lib/tftpboot/boot/fdi-image-rhel_7-vmlinuz
```

注記

現時点では、Red Hat Enterprise Linux 6 の Satellite 6 インストールの場合でも Red Hat Enterprise Linux 7 Discovery イメージのみが提供されます。**foreman-discovery-image** パッケージのアップグレード時に実行中の検出されたホストがある場合は、すぐにすべて再起動して更新されたバージョンのイメージをロードします。これは、Satellite 6 Web UI、CLI、API で実行できます。

6.2.2. PXE 起動の設定

不明のホストがプロビジョニングネットワークで起動する場合、Satellite Server は、ローカルハードドライブから起動するという 1 つのオプションを含む PXELinux ブートメニューを提供します。以下の手順では、ハードウェアの検出を有効にするために、Satellite に PXE テンプレートを構築します。

PXE-booting でホスト検出の設定

1. Satellite web UI で、**ホスト** > **プロビジョニングテンプレート** に移動します。
2. **プロビジョニングテンプレート** ページの右上で、**PXE デフォルトのビルド** をクリックして **OK** をクリックします。

テンプレートは、すべての TFTP サーバーでデフォルトのテンプレートになります。プロビジョニングサブネットにある不明な新しいホストはすべてこの設定を使用し、Foreman Discovery Image をデフォルトとして使用します。

6.2.3. グローバル Discovery 設定の確認

Satellite Web UI で Discovery プラグインに関連するグローバル設定を確認できます。**管理** > **設定** に移動して、**Discovered** タブを開きます。以下の設定に留意してください。

組織の検出、ロケーションの検出

これらの変数は、検出されたホストを配置する場所を指定します。デフォルトでは、検出されたホストは最初に作成された組織とロケーションの下に自動的に配置されます。

インターフェースのファクト

この変数は検出されたホストの MAC アドレスを判別するために使用される受信ファクトを指定します。デフォルトでは、PXELinux BOOTIF カーネルコマンドラインのオプションが使用されます。

ホスト名のファクト

この変数は、ホスト名に使用するファクトを一覧表示できます。ファクトはコンマで区切られ、最初のファクトが優先されます。

自動プロビジョニング

この変数は、指定したルールに従って自動プロビジョニングを有効にします。デフォルトでは false に設定されています。Red Hat は、自動プロビジョニングを有効にする前に手動プロビジョニングで行った設定をテストすることを推奨します。詳細は「検出されたホストのプロビジョニング」を参照してください。

再起動

この変数は、プロビジョニング時に PXE が検出したホストの自動再実行、もしくはローカルメディアから起動したホストに kexec を使用することを有効にします。デフォルトでは true に設定されています。

ホスト名の接頭辞

この変数は、ホスト名に使用するデフォルトの接頭辞を指定します。デフォルトでは "mac" に指定されます。この変数は文字で始める必要があります。

ファクトの列

この変数により、検出されたホストの一覧の追加列に Facter がレポートするファクトを追加できます。

強調表示したファクト

この変数は正規表現を使用して、強調表示したセクションにファクトを整理します。

ストレージのファクト

この変数は正規表現を使用して、ストレージセクションにファクトを整理します。

ハードウェアのファクト

この変数は正規表現を使用して、ハードウェアセクションにファクトを整理します。

ネットワークのファクト

この変数は正規表現を使用して、ネットワークセクションにファクトを整理します。

IPMI のファクト

この変数は正規表現を使用して、IPMI セクションにファクトを整理します。

6.3. SATELLITE CAPSULE SERVER DISCOVERY プラグインの設定

foreman_url 設定が Satellite Capsule Server 設定ファイルにあることを確認します。設定は以下のように表示されます。

```
# grep foreman_url /etc/foreman-proxy/settings.yml
:foreman_url: https://satellite.example.com
```

satellite-installer コマンドはこの変数を自動的に設定しますが、Red Hat ではホストが正常に反応し、通信をブロックするファイアウォールのルールがないことを確認することを推奨します。

6.3.1. Discovery 用サブネットの設定

検出されたホストを持つすべてのサブネットが Satellite Capsule Server で通信できるように設定する必要があります。Satellite Web UI で **インフラストラクチャー** > **サブネット** に移動し、ホスト検出を実行する必要がある各サブネットに必要な Capsule Server を選択し、Discovery Capsule Server に接続していることを確認します。

Capsule Server で Discovery プラグインが有効になっていることを確認するには、**インフラストラクチャー** > **Capsule** の順に移動します。Discovery プラグインが Capsule Server に関連付けられている機能の一覧に表示されるはずです。**更新** をクリックして、一覧が最新の状態であることを確認します。

6.3.2. Discovery プラグインでの Hammer の使用

Discovery プラグインで **hammer** コマンドを使用するには、以下のように **/etc/hammer/cli.modules.d/foreman_discovery.yml** で Discovery プラグインを有効にする必要があります。

```
:foreman_discovery:
  :enable_module: true
```

hammer が使用するファイルおよびディレクトリーの詳細は、「hammer configuration directories」 を参照してください。

6.3.3. ユーザーの各種パーミッションの確認

最初の起動時に、Satellite Capsule Server Discovery プラグインは **Discovery** というロールを作成します。このロールを管理者以外のユーザーに割り当て、それらのユーザーが Discovery プラグインを使用できるようにします。または、**perform_discovery** パーミッションを既存ロールに割り当てることもできます。ロールおよびパーミッションの詳細は『**Red Hat Satellite の管理**』の「ユーザーの作成および管理」を参照してください。

6.4. 検出されたホストのプロビジョニング

Satellite Server と Capsule Server の両方で Discovery プラグインを適切に設定すると、ベアメタルホストが自動的に検出できるようになります。これを行うには「PXE 起動の設定」の説明通りに、PXE 設

定テンプレートで設定されたプロビジョニングネットワークでマシンを起動します。マシンは Satellite Server に自動的に登録され、Satellite Web UI の **ホスト** > **検出されたホスト**の一覧に表示されます。

検出されたホストは手動でプロビジョニングすることも、自動プロビジョニングを設定することもできます。

6.4.1. ホストの手動プロビジョニング

以下の手順では、Satellite Web UI で検出されたホストを手動でプロビジョニングする方法を説明します。

検出されたホストの手動でのプロビジョニング

1. **ホスト** > **検出されたホスト** に移動します。
2. プロビジョニングするホストを選択し、**プロビジョニング** をクリックします。
3. ホストの **編集** ページに必要な詳細を入力し、**保存** をクリックします。

ホスト設定の保存時に、Satellite は TFTP サーバーのホストの PXELinux ファイルを変更し、検出されたホストを再起動します。次に選択したオペレーティングシステムのインストーラーを起動し、最終的にはインストールしたオペレーティングシステムを起動します。

検出された既存ホストのプロビジョニングを再実行する場合は、マシンからオペレーティングシステムを削除して再起動します。その後ホストは **検出されたホスト** ページに再度表示されます。

6.4.2. 検出されたホストの使用停止

Red Hat Satellite で特定のホストを管理する必要がなくなった場合は、ホストの使用を停止にして、検出されないようにする必要があります。

検出されたホストの使用停止

1. ホストをシャットダウンします。
2. **ホスト** > **検出されたホスト** に移動します。
3. **名前** コラムで、使用を停止するホストを検出し、**編集** ドロップダウンメニューから **削除** を選択します。

6.4.3. ホストの自動プロビジョニング

Satellite 6.3 では、ホストグループをプロビジョニングされたホストに割り当て、プロビジョニングを自動的にトリガーするプロビジョニングルールを定義できます。

プロビジョニングルールの作成

1. **設定** > **検出ルール** に移動します。
2. **新規ルール** をクリックします。プロビジョニングルールの以下のパラメーターを指定します。
   - **名前** はルールの一覧に表示されるルールの名前です。この名前には、英数字以外の文字やスペースを使用することはできません。

- **検索** は、特定のルールで検出されたホストを一致させる検索ステートメントです。スコープ指定の検索構文を使用してこれを定義できます。スコープ指定の検索例は「スコープ指定の検索構文」を参照してください。
- **ホストグループ** は、プロビジョニングプロセスを開始する前に一致するホストに割り当てられるホストグループです。選択したホストグループには必要なパラメーターがすべて設定されていることを確認します。必要なパラメーターにはアスタリスク (*) のマークが付けられます。
- **ホスト** は、人間の判読できるホスト名を、一致するホストに割り当てるパターンを定義します。これを空白のままにすると、割り当てられるホスト名はデフォルトで「macMACADDRESS」形式となります。プロビジョニングテンプレートに使用される構文と同じ構文が使用されます。詳細と例は「ホスト名のパターン」を参照してください。
- **ホストの制限** は、ルールに基づいてプロビジョニングされるホストの最大数です。制限に達すると、1 つ以上のホストが削除されるまでルールは有効になりません。通常のユースケースでは、ホスト名やホストグループなどのプロビジョニングパラメーターをエントリーごとに変更する必要がある場合に、サーバーラックまたは行ごとにルールを使用します。この値をゼロ (0) に設定すると制限なしに設定できます。
- **優先度** は、ルールの実行順序を指定します。値はゼロ以上である必要があります。値が低いほど優先度が高くなります。2 つのルールの優先度が同じ場合には、最初に検出されるルールが適用されます。
- **有効化** は、ルールを一時的に有効または無効にするオプションを提供します。

3. **送信** をクリックしてルールを保存します。

デフォルトで、Satellite はホストの自動検出を有効にしません。以下の手順では、自動プロビジョニング変数を有効にし、指定されたルールに基づいて自動プロビジョニングを行う方法について説明します。

自動プロビジョニングの有効化

1. Satellite Web UI で、**管理** > **設定** > **Discovered** に移動します。
2. **名前** コラムで自動プロビジョニングを探し、その値を **`true`** に設定します。
3. **保存** をクリックします。

Red Hat は、ルールを定義した後に、ホストに対して **自動検出** ボタンを使用してホストを検出し、ルールを適用することを推奨します。これにより、グローバルオプションを有効にせずに自動プロビジョンがトリガーされます。

6.4.4. スコープ指定の検索構文

このセクションでは、選択したパラメーターに応じて検出されたホストにフィルターを設定するスコープ指定の検索構文を使用する方法を説明します。これは自動プロビジョニングのルールを作成する際に便利です (「ホストの自動プロビジョニング」 を参照)。

Satellite Web UI の検索フィールドは自動補完に対応しているため、検索構文の作成が容易になります。たとえば、**ホスト** > **検出したホスト** ページで検索パターンをテストすることができます。以下は通常の検索クエリーの例になります。

- facts.architecture = x86_64

- facts.bios_vendor ~ 'Dell*'
- facts.macaddress = "aa:bb:cc:dd:ee:ff"
- facts.macaddress_eth0 = "aa:bb:cc:dd:ee:ff"
- facts.ipaddress_eth1 ~ "192.168.*"
- facts.architecture ^ (x86_64,i386)

注記

スコープ指定検索のキャレット記号 (^) は「in」(SQL と同じ用法) を意味し、正規表現に使用される「starts with」を意味しません。スコープ指定の検索演算子の完全一覧は https://github.com/wvanbergen/scoped_search/blob/master/lib/scoped_search/query_langua を参照してください。

Satellite 6.3 では、すべてのファクトは文字列のため、数値比較を実行することはできません。ただし、3 つの重要なファクトが抽出され、数字に変換されます。詳細は表6.1「数値比較を可能にするファクト」で説明されています。

表6.1 数値比較を可能にするファクト

検索パラメーター	説明	使用例
cpu_count	CPU の数	cpu_count >= 8
disk_count	割り当てられたディスクの数	disk_count < 10
disks_size	ディスク空き容量の合計 (MiB)	disks_size > 1000000

6.4.5. ホスト名のパターン

本セクションでは、自動プロビジョニングのルールを作成する際に使用できるホスト名のパターンを一覧表示します (「ホストの自動プロビジョニング」を参照)。

ターゲットホスト名のテンプレートパターンには、プロビジョニングテンプレート (ERB) と同じ構文が使用されます。ドメインは自動的に追加されます。**@host** 属性のほかに、ランダムな整数の **rand()** 関数が利用できます。以下は例になります。

- application-server-<%= rand(99999) %>
- load-balancer-<%= @host.facts['bios_vendor'] + '-' + rand(99999) %>
- wwwsrv-<%= @host.hostgroup.name %>
- minion-<%= @host.discovery_rule.name %>
- db-server-<%= @host.ip.gsub('.','-') + '-' + @host.hostgroup.subnet.name %>

重要

ホスト名のパターンを作成する際に、作成されるホスト名が固有の名前であることを確認してください。ホスト名は数字で始めることができません。Facter (MAC アドレス、BIOS、シリアル ID など) で提供される固有情報を使用するか、またはホスト名をランダム化することは適切な方法です。

6.4.6. コマンドラインでの Discovery プラグインの使用

hammer コマンドを使用して、検出に関連する特定のタスクを実行できます。**hammer -h** コマンドを実行して設定を確認します。

```
$ hammer -h | grep discovery
discovery                    Manipulate discovered hosts.
discovery_rule               Manipulate discovered rules.
```

hammer discovery -h コマンドを使用して利用可能なオプションを表示します。たとえば、以下のコマンドを使用して検出されるホストを再起動できます (以下は ID が 130 の場合)。

```
$ hammer discovery reboot -id 130
Host reboot started
```

6.5. DISCOVERY イメージの拡張

カスタムファクト、ソフトウェア、またはデバイスドライバーを使用して Satellite Discovery イメージを拡張することができます。イメージで使用できるように追加コードが含まれる圧縮されたアーカイブファイルを提供することもできます。

最初に、以下のディレクトリー構造を作成します。

```
.
├── autostart.d
│   └── 01_zip.sh
├── bin
│   └── ntpdate
├── facts
│   └── test.rb
└── lib
    ├── libcrypto.so.1.0.0
    └── ruby
        └── test.rb
```

説明:

- **autostart.d** ディレクトリーには、起動時にホストが Satellite に登録される前に、イメージによって POSIX の順序で実行するスクリプトが含まれています。
- **bin** ディレクトリーは $PATH 変数に追加されます。ここにバイナリーファイルを配置すると、autostart スクリプトで使用できます。
- **facts** ディレクトリーは FACTERLIB 変数に追加され、カスタムファクトを設定して Satellite に送信できるようになります。

- **lib** ディレクトリーを LD_LIBRARY_PATH 変数に追加し、**lib/ruby** を RUBYLIB 変数に追加して、**/bin** のバイナリーファイルを正常に実行できるようにします。

新規ディレクティブおよびオプションが既存の環境変数 (PATH、LD_LIBRARY_PATH、RUBYLIB、および FACTERLIB) に追加されます。スクリプトの内容へのパスを明示的に指定する必要がある場合は、zip コンテンツがイメージ上の **/opt/extension** ディレクトリーに抽出されます。

上記のディレクトリー構造を作成したら、以下のコマンドでこの構造を zip アーカイブにパッケージ化します。

```
zip -r my_extension.zip .
```

複数の zip ファイルを作成できますが、これらは Discovery イメージの同じ場所に抽出されるため、ファイル名が同じ場合に、後から抽出された zip のファイルが、先に抽出された zip のファイルを上書きします。

Discovery イメージで使用される拡張子が認識されるように、検出イメージと共に zip ファイルを TFTP サーバーに配置してから、PXELinux テンプレートの APPEND 行を、パスが TFTP ルートに相対する **fdi.zips** オプションで更新します。たとえば、 **$TFTP/zip1.zip** および **$TFTP/boot/zip2.zip** に 2 つのアーカイブがある場合は、以下の構文を使用します。

```
fdi.zips=zip1.zip,boot/zip2.zip
```

PXE テンプレートの更新方法は「PXE 起動の設定」を参照してください。

6.6. SATELLITE 検出のトラブルシューティング

マシンが、Satellite Web UI の **ホスト** > **検出されたホスト** に表示されない場合は、以下の設定領域を調べてエラーを切り分けます。

- **ホスト** > **プロビジョニングテンプレート** に移動し、**PXE デフォルトのビルド** ボタンを使用してデフォルトの PXELinux テンプレートを再デプロイします。
- TFTP Capsule Server で **pxelinux.cfg/default** 設定ファイルを確認します。
- ホスト、Capsule Server、および Satellite Server 間で適切なネットワーク接続があることを確認します。
- 使用している PXELinux テンプレートに含まれている PXE 検出スニペットを確認します。スニペットの名前は **pxelinux_discovery**、**pxegrub_discovery**、または **pxegrub2_discovery** です。PXE 検出スニペットの **proxy.url** オプションと **proxy.type** オプションを検証してください。
- 検出されたノードで DNS が適切に機能していることを確認するか、使用している PXE Linux テンプレートにある PXE 検出スニペットの **proxy.url** オプションにある IP アドレスを使用します。
- DHCP サーバーが、起動したイメージに IP アドレスを適切に送信していることを確認します。
- 検出されたホスト (または仮想マシン) に 1200 MB 以上のメモリーがあることを確認します。メモリーが 1200 MB より少なくなると、イメージがインメモリーで抽出される必要があるため、各種のカーネルパニックエラーがランダムに発生する可能性があります。

重要なシステムファクトを収集するには、**discovery-debug** コマンドを使用します。これにより、システムログ、ネットワーク設定、ファクトの一覧などの情報が 標準出力に出力されます。通常のユースケースでは、追加の調査のために、**scp** コマンドでこの出力をリダイレクトしてコピーします。

検出されたホストの最初の仮想コンソールは **systemd** ログのために予約されます。とくに役立つシステムログには以下のようにタグが付けられます。

- discover-host: 初回ファクトのアップロード
- foreman-discovery: ファクトの更新、リモート再起動のコマンド
- nm-prepare: NetworkManager を事前に定義する起動スクリプト
- NetworkManager: ネットワーク情報

TTY2 以上を使用して検出されたホストにログインします。root アカウントおよび SSH アクセスはデフォルトで無効になっていますが、以下のカーネルコマンドラインのオプションを使用して、デフォルト PXELinux テンプレートの APPEND 行に root パスワードを設定します。

```
fdi.ssh=1 fdi.rootpw=redhat
```

第7章 RED HAT SATELLITE と ANSIBLE TOWER の統合

Red Hat Satellite 6.3 と Ansible Tower を統合して、Ansible Tower の動的インベントリーソースとして Satellite Server を使用します。

また、ホストまたは Ansible Tower のいずれかから、Satellite が管理するホストで Playbook を実行するようにプロビジョニングコールバック機能を使用できます。Satellite Server から新しいプロビジョニングする際に、プロビジョニングコールバック機能により、Ansible Tower から Playbook を実行します。Playbook は、以下のキックスタートデプロイメントに従うようにホストを設定します。

7.1. SATELLITE SERVER を動的インベントリー項目として ANSIBLE TOWER に追加

Satellite Server をダイナミックインベントリー項目として Ansible Tower に追加するには、Ansible Tower に Satellite Server ユーザーの認証情報を作成し、 Ansible Tower ユーザーを認証情報に追加してから、インベントリーソースを設定する必要があります。

前提条件

Satellite Server と Ansible Tower が、証明書とコールバックを使用して通信します。

- 必要なパーミッションフィルターを持つ統合ロールが割り当てられている Satellite Server ユーザーが必要になります。ユーザー、ロール、およびパーミッションフィルターの管理方法は、**『Red Hat Satellite の管理』**の「ユーザーとロールの管理」および「ロールの作成および管理」を参照してください。
- 以下のパーミッションフィルターを指定して、ロールをユーザーに割り当てる必要があります。

表7.1 パーミッションフィルター

リソース	パーミッション	アクセスの説明
ホスト	`view_hosts`	Satellite Server ホストを表示します。
ホストグループ	`view_hostgroups`	Satellite Server ホストグループを表示します。
ファクト値	`view_facts`	Satellite Server ファクトを表示します。

Satellite Server を動的インベントリー項目として Ansible Tower に追加

1. Ansible Tower Web UI で、Satellite に対して認証情報を作成します。認証情報の作成方法は**『Ansible Tower ユーザーガイド』**の「新規認証情報の追加」および「Red Hat Satellite 6 認証情報」を参照してください。

表7.2 Satellite の認証情報

認証情報の種類:	**Red Hat Satellite 6**
Satellite 6 URL:	https://**satellite.example.com**

ユーザー名:	統合ロールを持つ Satellite ユーザーのユーザー名。
パスワード:	Satellite ユーザーのパスワード。

2. 新しい認証情報に Ansible Tower ユーザーを追加します。ユーザーを認証情報に追加する方法は**『Ansible Tower ユーザーガイド』**の「認証情報の使用開始」を参照してください。

3. Satellite Server を新しいインベントリーソースとして追加し、以下のインベントリーソースオプションを指定します。インベントリーを追加する方法は**『Anabilities Tower ユーザーガイド』**の「新規インベントリーの追加」を参照してください。

表7.3 インベントリーソースオプション

ソース:	**Red Hat Satellite 6**
認証情報:	Satellite Server に作成した認証情報。
上書き:	選択
起動時の更新:	選択
キャッシュのタイムアウト:	**90**

インベントリー管理の詳細は **『Ansible Tower ユーザーガイド』** の 「インベントリー」を参照してください。

7.2. ホストへのプロビジョニングコールバックの設定

Ansible Tower テンプレートにプロビジョニングコールバックを設定できます。設定すると、Ansible Tower サーバーで特定の URL を呼び出して、変数を渡し、呼び出しシステムで Playbook を実行できます。

新たにデプロイしたホストで Playbook を実行するのに、この機能を使用することもできます。プロビジョニングコールバックの詳細は**『Anabilities Tower ユーザーガイド』**の「プロビジョニングコールバック」を参照してください。

Satellite の **`Satellite Kickstart Default`** テンプレートおよび **`Satellite Kickstart Default Finish`** テンプレートには、以下の 3 つのスニペットが含まれます。

```
ansible_provisioning_callback
ansible_tower_callback_script
ansible_tower_callback_service
```

ホストまたはホストグループにプロビジョニングコールバックを設定するには、各スニペットにパラメーターを作成して定義する必要があります。

前提条件

- ホストにプロビジョニングコールバックを設定する前に、Red Hat Satellite 6.3 と Ansible Tower を統合する必要があります。詳細は「Satellite と Ansible Tower の統合」を参照してください。
- Ansible Tower Web UI で、プロビジョニングコールバックを有効にし、ホストの設定キーを生成し、ジョブテンプレートの **template_ID** を取得する必要があります。ジョブテンプレートの詳細は **『Ansible Tower ユーザーガイド』**の「ジョブテンプレート」を参照してください。

ホストのプロビジョニングコールバックの設定

1. Red Hat Satellite Web UI で、**ホスト** > **すべてのホスト** に移動します。
2. ホストページで、**ホスト** の一覧から、編集するホストを選択します。
3. ホストグループウィンドウで **パラメーター** タブをクリックします。
4. ホストパラメーターウィンドウで、**パラメーターの追加** をクリックします。
5. **名前** フィールドに、**`ansible_tower_provisioning`** と入力します。
6. **値** フィールドに、**`true`** と入力します。
7. ステップ 4 から 6 を繰り返して、以下のパラメーターを作成します。

表7.4 ホストパラメーター

名前	値	説明
`ansible_tower_provisioning`	true	プロビジョニングコールバックを有効にします。
`ansible_tower_fqdn`	**tower.example.com**	Ansible Tower の完全修飾ドメイン名 (FQDN)。
`ansible_job_template_id`	**template_ID**	テンプレートの URL で確認できるプロビジョニングテンプレートの ID (**`/templates/job_template/5`**)。
`ansible_host_config_key`	**config_KEY**	Ansible Tower のジョブテンプレートが作成したホスト設定キー。

8. 必要なパラメーターをすべて作成したら **送信** をクリックします。
9. プロビジョニングコールバックが正しく設定されていることを確認するには、**`ansible-callback`** サービスを開始し、サービスのステータスを確認します。
   a. コマンドラインで、以下のコマンドを実行して、**`ansible-callback`** サービスを開始します。

```
# systemctl start ansible-callback
```

b. コマンドラインで、以下のコマンドを実行して、**ansible-callback** サービスのステータスを出力します。

```
# systemctl status ansible-callback
```

プロビジョニングコールバックが正しく設定されていると、以下の出力が返ります。

```
SAT_host systemd[1]: Started Provisioning callback to Ansible
Tower...
```

ホストグループにプロビジョニングコールバックの設定

1. Red Hat Satellite Web UI で、**設定** > **ホストグループ** に移動します。
2. ホストグループページで、**ホストグループ** 一覧から、編集するホストグループを選択します。
3. ホストグループウィンドウで **パラメーター** タブをクリックします。
4. ホストグループウィンドウで、**パラメーターの追加** をクリックします。
5. **名前** フィールドに、**ansible_tower_provisioning** と入力します。
6. **値** フィールドに、**true** と入力します。
7. ステップ 4 から 6 を繰り返して、以下のパラメーターを作成します。

表7.5 ホストグループパラメーター

名前	値	説明
ansible_tower_provisioning	true	プロビジョニングコールバックを有効にします。
ansible_tower_fqdn	**tower.example.com**	Ansible Tower の完全修飾ドメイン名 (FQDN)。
ansible_job_template_id	**template_ID**	テンプレートの URL で確認できるプロビジョニングテンプレートの ID (**/templates/job_template/5**)。
ansible_host_config_key	**config_KEY**	Ansible Tower のジョブテンプレートが作成したホスト設定キー。

8. 必要なパラメーターをすべて作成したら **送信** をクリックします。
9. プロビジョニングコールバックが正しく設定されていることを確認するには、**ansible-callback** サービスを開始し、サービスのステータスを確認します。

a. コマンドラインで、以下のコマンドを実行して、**ansible-callback** サービスを開始します。

```
# systemctl start ansible-callback
```

b. コマンドラインで、以下のコマンドを実行して、**ansible-callback** サービスのステータスを出力します。

```
# systemctl status ansible-callback
```

プロビジョニングコールバックが正しく設定されていると、以下の出力が返ります。

```
SAT_host systemd[1]: Started Provisioning callback to Ansible
Tower...
```

プロビジョニングコールバック URL と、ホストの設定キーを使用して Ansible Tower を呼び出します。これにより、ホストに対してテンプレートで指定した Playbook が実行します。

プロビジョニングコールバック機能を使用して、プロビジョニングプロセスの一部として、Ansible Tower から Playbook を呼び出せます。Playbook は、キックスタートのデプロイメント後にホストを設定します。

第8章 サンプルシナリオ

8.1. 簡単なシナリオ

ここでは、ホストを 1 台追加して登録して設定し、ジョブを実行する方法を説明します。

8.1.1. ホストの作成

以下の手順では、Red Hat Satellite でホストを作成する方法を説明します。

ホストの作成

1. **ホスト** > **ホストの作成** をクリックします。
2. **ホスト** タブで、必要な詳細を入力します。
    a. **名前** フィールドに、ホストの名前 (例: **host1.example.com**) を入力します。
    b. **組織** フィールドに、組織名 (例: **MyOrg**) を入力します。
    c. **ロケーション** フィールドに、ロケーション名 (例: **MyLoc**) を入力します。
3. オプションで、**Puppet クラス** タブで、追加する Puppet クラスを選択します。
4. **インターフェース** タブで、プライマリーインターフェースを編集します。
    a. **アクション** コラムで、**編集** ボタンをクリックします。
    b. **タイプ** ドロップダウンメニューからタイプ (例: **Interface**) を選択します。
    c. **MAC アドレス** フィールドに、ホストの MAC アドレスを入力します。
    d. **デバイス ID** フィールドに、インターフェースのデバイス識別子 (例: **eth0**) を指定します。
    e. **DNS 名** フィールドに、ホストの DNS 名を指定します。プライマリーインターフェースの場合は、このホスト名が、FQDN を形成する際のドメイン名に使用されます。
    f. **ドメイン** ドロップダウンメニューからドメインを選択します (例: **satellite.example.com**)。これにより、**IPv4 サブネット** および **IPv6 サブネット** 一覧が自動的に更新され、利用可能なサブネットが選択できます。オプションで、サブネットを選択します。
    g. **IPv4 アドレス** フィールドに、ホストの IPv4 アドレスを入力します。
    h. **OK** をクリックします。
5. **オペレーティングシステム** タブに、必要な詳細を入力します。
    a. **アーキテクチャー** ドロップダウンメニューからアーキテクチャーを選択します (例: **x86_64**)。
    b. **オペレーティングシステム** ドロップダウンメニューからオペレーティングシステムを選択します (例: **RHEL Server 7.4**)。
    c. **パーティションテーブル** ドロップダウンメニューからパーティションテーブル を選択します (例: **Kickstart default**)。

d. **Root パスワード** フィールドに、ホストの root パスワードを入力します。

6. オプションで **パラメーター** タブで、Puppet マスターで、デフォルト値をオーバーライドするパラメーターを選択します。

7. オプションで、**追加情報** タブに、ホストに関する追加情報を入力します。

8. **送信** をクリックします。

8.1.2. ホストの登録

host1.example.com を作成したら、登録してアップデートを受け取れるようにします。以下の手順は、ホストが Red Hat Enterprise Linux 7 を実行していることを前提としています。

ホストの登録

1. 端末で、root ユーザーとしてホストに接続します。

2. 同期ツールが有効になっており、ホストで実行していることを確認します。

```
# systemctl start chronyd; systemctl enable chronyd
```

3. コンシューマー RPM を、ホストを登録する Satellite Server または Capsule Server からインストールします。コンシューマー RPM は、ホストのコンテンツソースのロケーションを更新し、ホストが、Red Hat Satellite に指定したコンテンツソースからコンテンツをダウンロードできるようにします。

```
# rpm -Uvh http://satellite.example.com/pub/katello-ca-consumer-
latest.noarch.rpm
```

4. ホストに対して、適切なコンテンツビューと環境に関連しているアクティベーションキーが存在していることを確認します。存在していない場合は、**『コンテンツ管理ガイド』**の「アクティベーションキーの管理」 を参照してください。

5. Red Hat Subscription Manager (RHSM) に関連するすべての古いホストデータをクリアします。

```
# subscription-manager clean
```

6. RHSM を使用してホストを登録します。

```
# subscription-manager register --org MyOrg \
--activationkey my_activation_key
```

登録後のコマンド出力:

```
# subscription-manager register --org MyOrg --activationkey
my_activation_key
The system has been registered with id: 62edc0f8-855b-4184-b1b8-
72a9dc793b96
```

7. **Red Hat Satellite Tools 6** リポジトリーを有効にします。

```
# subscription-manager repos --enable=rhel-version-server-satellite-
tools-6-rpms
```

8. **katello-agent** をインストールします。

```
# yum install katello-agent
```

9. **goferd** サービスが実行していることを確認します。

```
# systemctl start goferd
```

10. Puppet エージェントをインストールして設定します。

    a. Puppet エージェントをインストールします。

```
# yum install puppet
```

    b. システムの起動時に Puppet エージェントが起動するように設定します。

```
# systemctl enable puppet
```

    c. **/etc/puppet/puppet.conf** ファイルでサーバーおよび環境の設定を指定して、Puppet エージェントを設定します。

```
# vi /etc/puppet/puppet.conf
```

```
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
    ssldir = /var/lib/puppet/ssl

...

[agent]
    # The file in which puppetd stores a list of the classes
    # associated with the retrieved configuratiion.  Can be
loaded in
    # the separate ``puppet`` executable using the ``--
loadclasses``
    # option.
    # The default value is '$confdir/classes.txt'.
    classfile = $vardir/classes.txt
    pluginsync = true
    report = true
    ignoreschedules = true
    daemon = false
```

```
    ca_server = satellite.example.com
    server = satellite.example.com
    environment = KT_Example_Org_Library_RHEL7Server

    # Where puppetd caches the local configuration.  An
    # extension indicating the cache format is added
automatically.
    # The default value is '$confdir/localconfig'.
    localconfig = $vardir/localconfig

...
```

d. ホスト上で Puppet エージェントを実行します。

```
# puppet agent -t --waitforcert 10 --server satellite.example.com
```

e. Satellite Server Web UI から、Puppet クライアントの SSL 証明書に署名します。

i. Web UI から Satellite Server にログインします。

ii. **インフラストラクチャー** > **Capsule** に移動します。

iii. 必要な Capsule の右側にあるドロップダウンメニューから **証明書** を選択します。

iv. 必要なホストの右側にある **署名** をクリックします。

v. **puppet agent** コマンドを再入力します。

```
# puppet agent -t --server satellite.example.com
```

8.1.3. ホストでのジョブの実行

以下の手順では、作成して登録しておいたホスト **host1.example.com** でジョブテンプレートを実行する方法を説明します。

リモートジョブの実行

1. **ホスト** > **すべてのホスト** に移動し、ターゲットホストを選択します (この例では **host1.example.com** を使用)。
2. 画面右上の **アクションの選択** メニューから **リモートジョブのスケジュール** を選択します。
3. **ジョブ呼び出し** ページで、主なジョブ設定を定義します。
   a. **ジョブカテゴリー** ドロップダウンメニューから、ジョブカテゴリー (例: **Commands**) を選択します。
   b. **ジョブテンプレート** ドロップダウンメニューから、ジョブテンプレート (例: **Run Command - SSH Default**) を選択します。
   c. **command** フィールドに、ホストで実行するコマンドを入力します。たとえば、**timedatectl set-timezone Europe/Prague** と実行すると、タイムゾーンがヨーロッパのプラハに設定されます。
4. **送信** をクリックします。

付録A テンプレート作成の参照

Embedded Ruby (ERB) は、プレーンテキストと Ruby コードを組み合わせるテンプレートをベースにしてテキストファイルを生成するためのツールです。Red Hat Satellite は、**プロビジョニングテンプレート** (『**プロビジョニングガイド**』の「プロビジョニングテンプレートの作成」)、リモート実行の**ジョブテンプレート** (「5章 *ホストでのリモートジョブの実行*」)、**パーティションテーブル** のテンプレート (『**プロビジョニングガイド**』の「パーティションテーブルの作成」)、**スマート変数** (『**Puppet ガイド**』の「スマート変数の設定」)、および **スマートクラスパラメーター** (『**Puppet ガイド**』の「スマートクラスパラメーターの設定」) で ERB 構文を使用します。本セクションでは、ERB テンプレートで使用できる Satellite 固有の関数および変数の概要を説明し、使用例を挙げます。また、Red Hat Satellite で提供されるデフォルトテンプレート (**ホスト** > **プロビジョニングテンプレート**、**ホスト** > **ジョブテンプレート**) でも ERB 構文例が紹介されています。

ホストのプロビジョニング時またはリモートジョブの実行時に、ERB のコードが実行し、変数がホスト固有の値に置き換えられます。このプロセスは、**レンダリング** と呼ばれています。Satellite Server ではセーフモードのレンダリングオプションがデフォルトで有効になっており、これにより、有害なコードがテンプレートから実行されないようにすることができます。

A.1. ERB テンプレートの作成

以下は ERB 構文を要約したポイントになります。

- **<% %>**: Ruby コードを ERB テンプレート内で囲むマークです。コードはテンプレートのレンダリング時に実行されます。これには Ruby の制御フロー構造と、Satellite 固有の関数および変数を含めることができます。たとえば、以下のようになります。

```
<% if @host.operatingsystem.family == "Redhat" &&
@host.operatingsystem.major.to_i > 6 %>
systemctl <%= input("action") %> <%= input("service") %>
<% else %>
service <%= input("service") %> <%= input("action") %>
<% end -%>
```

- **<%= %>**: コード出力はテンプレートに挿入されます。これは変数の置き換えに便利です。たとえば、以下のようになります。

```
echo <%= @host.name %>
```

- **<% -%>**、**<%= -%>**: デフォルトで、行末で閉じられている場合に、改行文字が Ruby ブロックの後に挿入されます。この動作を抑制するには、囲みマークを変更します。たとえば、以下のようなテンプレートになります。

```
curl <%= @host.ip -%>
/mydir
```

  上記のテンプレートは以下のようにレンダリングされます。

```
curl <%= @host.ip %>/mydir
```

  これは、レンダリングされるテンプレートの行数を減らすために使用されます (Ruby 構文で許可される場合)。

- **<%# %>**: テンプレートのレンダリング時に無視されるコメントを囲むマークです。

```
<%# A comment %>
```

A.2. ERB テンプレートのトラブルシューティング

Satellite Web UI では、特定ホスト用のテンプレートのレンダリングを検証するための方法を 2 つ提供しています。

- **テンプレートエディターによる直接的な方法**: テンプレートの編集時に (**ホスト** > **パーティションテーブル**、**ホスト** > **プロビジョニングテンプレート**、または **ホスト** > **ジョブテンプレート** の) **テンプレート** タブで **プレビュー** をクリックしてから、ドロップダウンメニューでホストを選択します。次に、選択したホストのパラメーターを使用して、テキストフィールドでテンプレートをレンダリングします。プレビューが失敗した場合は、ここでテンプレートの問題を特定できます。

- **ホストの詳細ページを使用する方法**: **ホスト** > **すべてのホスト** でホストを選択し、**テンプレート** タブをクリックして、ホストに関連付けらえたテンプレートを一覧表示します。選択したテンプレートの横にあるドロップダウンメニューから **確認** を選択して、そのテンプレートをレンダリングします。

A.3. SATELLITE 固有の関数および変数

本セクションでは、ERB テンプレート用の Satellite 固有の関数および変数を一覧表示します。この中にはどのテンプレートでも使用できるものもあれば、使用が制限されるものもあります。たとえば、ジョブテンプレートは @host 変数のみを受け入れ、表A.4「キックスタート固有の変数」 の変数はキックスタートテンプレートでのみ適用できます。

以下の表に記載されている関数は、すべての種類のテンプレートで使用することができます。

表A.1 汎用的な関数

名前	説明
indent(n)	コードブロックを n スペース分インデントします。インデントされていないスニペットテンプレートの使用時に便利です。
foreman_url(kind)	完全な URL を、ホストでレンダリングされた指定タイプのテンプレートに返します。たとえば、「provision」タイプのテンプレートは通常 **http://HOST/unattended/provision** にあります。
snippet(name)	指定されたスニペットテンプレートをレンダリングします。プロビジョニングテンプレートをネスト化するのに便利です。
snippets(file)	Foreman データベースで、指定したスニペットをレンダリングします。データベースにない場合は **unattended/snippets/** ディレクトリーからこれをロードします。

名前	説明
snippet_if_exists(name)	指定してスニペットをレンダリングし、指定された名前を持つスニペットが見つからない場合は省略します。

例A.1 スニペットおよびインデント関数の使用

以下の構文は、**subscription_manager_registration** スニペットをテンプレートにインポートし、4 スペース分インデントします。

```
<%= indent 4 do
snippet 'subscription_manager_registration'
end %>
```

以下の関数はジョブテンプレートで使用できます。使用例は「詳細テンプレートの作成」を参照してください。

表A.2 ジョブテンプレート固有の関数

名前	説明
input(input_name)	ジョブ実行時に、指定した入力の値を返します。
render_template(name, parameters)	汎用的な snippet() 関数と同様に、指定されたテンプレートを返しますが、引数をテンプレートに渡すことができます。

以下の変数は、テンプレート内でのホストデータの使用を可能にします。

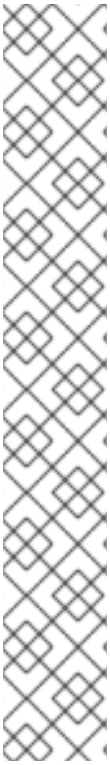
表A.3 ホスト固有の変数および関数

名前	説明
@host.architecture	ホストのアーキテクチャーです。
@host.bond_interfaces	すべてのボンディングインターフェースのアレイを返します。「注記」を参照してください。
@host.capabilities	システムプロビジョニングの方法には、ビルド (キックスタートなど) またはイメージのいずれかを使用できます。
@host.certname	ホストの SSL 証明書名です。
@host.diskLayout	ホストのディスクレイアウトです。オペレーティングシステムから継承できます。

名前	説明
@host.domain	ホストのドメインです。
@host.environment	ホストの Puppet 環境です。
@host.facts	Facter からファクトの Ruby ハッシュを返します。たとえば、出力の 'ipaddress' ファクトにアクセスするには、@host.facts['ipaddress'] を指定します。
@host.grub_pass	ホストの GRUB パスワードを返します。
@host.hostgroup	ホストのホストグループです。
@host.info['parameters']	ホストパラメーターに関する情報が含まれる Ruby ハッシュを返します。たとえば、@host.info['parameters']['lifecycle_environment'] を使用してホストのライフサイクル環境を取得します。
@host.image_build?	ホストがイメージを使用してプロビジョニングされる場合は **`true`** を返します。
@host.interfaces	プライマリーインターフェースを含む利用可能なすべてのホストインターフェースのアレイが含まれます。「注記」を参照してください。
@host.interfaces_with_identifier('IDs')	指定した ID を持つインターフェースのアレイを返します。複数の ID のアレイを渡すことができます (例: @host.interfaces_with_identifier(['eth0', 'eth1']))。「注記」を参照してください。
@host.ip	ホストの IP アドレスです。
@host.location	ホストの場所 (ロケーション) です。
@host.mac	ホストの MAC アドレスです。
@host.managed_interfaces	管理対象インターフェースのアレイを返します (BMC およびボンディングインターフェースを除く)。「注記」を参照してください。
@host.medium	割り当てられたオペレーティングシステムのインストールメディアです。
@host.name	ホストの完全名です。
@host.operatingsystem.family	オペレーティングシステムファミリーです。

名前	説明
@host.operatingsystem.major	割り当てられたオペレーティングシステムのメジャーバージョンの番号です。
@host.operatingsystem.minor	割り当てられたオペレーティングシステムのマイナーバージョンの番号です。
@host.operatingsystem.name	割り当てられたオペレーティングシステムの名前です。
@host.operatingsystem.boot_files_uri(@host.medium,@host.architecture)	カーネルおよび initrd への完全パスで、アレイを返します。
@host.os.medium_uri(@host)	プロビジョニングに使用される URI です (インストールメディアに設定されるパス)。
@host.param_false?(name)	指定した名前のホストパラメーターが false と評価される場合に **`false`** を返します。
@host.param_true?(name)	指定した名前のホストパラメーターが true と評価される場合に **`true`** を返します。
@host.params['parameter_name']	指定したパラメーターの値を返します。
@host.primary_interface	ホストのプライマリーインターフェースを返します。
@host.provider	コンピュートリソースプロバイダーです。
@host.provision_interface	ホストのプロビジョニングインターフェースを返します。インターフェースオブジェクトを返します。
@host.ptable	パーティションテーブル名です。
@host.puppetmaster	ホストが使用する必要のある Puppet マスターです。
@host.pxe_build?	ホストがネットワークまたは PXE を使用してプロビジョニングされる場合に **`true`** を返します。
@host.shortname	ホストの省略名です。
@host.sp_ip	BMC インターフェースの IP アドレスです。
@host.sp_mac	BMC インターフェースの MAC アドレスです。
@host.sp_name	BMC インターフェースの名前です。

名前	説明
@host.sp_subnet	BMC ネットワークのサブネットです。
@host.subnet.dhcp	DHCP プロキシーがこのホストに設定されている場合は `true` を返します。
@host.subnet.dns_primary	ホストのプライマリー DNS サーバーです。
@host.subnet.dns_secondary	ホストのセカンダリー DNS サーバーです。
@host.subnet.gateway	ホストのゲートウェイです。
@host.subnet.mask	ホストのサブネットマスクです。
@host.url_for_boot(:initrd)	このホストに関連付けられる initrd イメージへの完全パスです。変数を補間しないので推奨されません。
@host.url_for_boot(:kernel)	このホストに関連付けられたカーネルへの完全パスです。変数を補間しないので推奨されません。boot_files_uri が優先されます。
@provisioning_type	プロビジョニングのタイプに応じて「host」または「hostgroup」と等しくなります。
@static	ネットワーク設定が静的な場合、`true` を返します。
@template_name	レンダリングされるテンプレートの名前です。
grub_pass	md5pass 引数でラップされる GRUB パスワードを返します (例: **--md5pass=#{@host.grub_pass}**)。
ks_console	ポートを使用して組み立てられる文字列、およびカーネル行に追加できるボーレートを返します (例: **console=ttyS1,9600**)。
root_pass	システムに設定される root パスワードを返します。



注記

@host.interfaces、**@host.bond_interfaces** などのネットワークインターフェースに関連するホスト変数は、アレイで分類されるインターフェースデータを返します。特定インターフェースのパラメーター値を抽出するには、Ruby メソッドを使用してアレイを解析します。たとえば、アレイの最初のインターフェースに関する情報を取得し、これをキックスタートテンプレートで使用するには、以下を実行します。

```
<% myinterface = @host.interfaces.first %>
IPADDR="<%= myinterface.ip %>"
NETMASK="<%= myinterface.subnet.mask %>"
GATEWAY="<%= myinterface.subnet.gateway %>"
```

インターフェース名のアレイを抽出するなどの目的でインターフェースアレイを繰り返すことができます。以下は例になります。

```
<% ifnames = []
@host.interfaces.each do |i|
  ifnames.push(i.name)
end %>
```

例A.2 ホスト固有変数の使用

以下の例では、ホストで Puppet および Puppetlabs リポジトリーが有効になっているかどうかをチェックします。

```
<%
pm_set = @host.puppetmaster.empty? ? false : true
puppet_enabled = pm_set || @host.param_true?('force-puppet')
puppetlabs_enabled = @host.param_true?('enable-puppetlabs-repo')
%>
```

以下の例では、パッケージ関連の決定に使用できるホストのオペレーティングシステムのマイナーバージョンおよびメジャーバージョンを取得する方法を示します。

```
<%
os_major = @host.operatingsystem.major.to_i
os_minor = @host.operatingsystem.minor.to_i
%>

<% if ((os_minor < 2) && (os_major < 14)) -%>
...
<% end -%>
```

以下の例では、ホストのサブネットで DHCP ブートモードが有効な場合に 'kickstart_networking_setup' スニペットをインポートします。

```
<% subnet = @host.subnet %>
<% if subnet.respond_to?(:dhcp_boot_mode?) -%>
<%= snippet 'kickstart_networking_setup' %>
<% end -%>
```

> 一般的な Ruby メソッドのほとんどはホスト固有の変数で使用できます。たとえば、ホストの IP アドレスの最後のセグメントを抽出するには、以下を使用できます。
>
> ```
> <% @host.ip.split('.').last %>
> ```

以下の変数は、キックスタートプロビジョニングテンプレート内で使用されるように設計されています。

表A.4 キックスタート固有の変数

名前	説明
@arch	ホストのアーキテクチャー名です。@host.architecture.name と同じです。
@dynamic	使用されているパーティションテーブルが %pre スクリプト (表の最初の行に #Dynamic オプションがある) の場合 **true** を返します。
@epel	epel-release rpm の正しいバージョンを自動インストールするコマンドです。%post スクリプトで使用されます。
@mediapath	URL コマンドを提供する詳細なキックスタート行です。
@osver	オペレーティングシステムのメジャーバージョンの番号です。@host.operatingsystem.major と同じです。

付録B GOFERD を使用しないホスト管理

Satellite 6.2.11 以降、リモート実行によるエラータおよびパッケージの管理は、**yum** プラグインで利用できます。これにより、goferd サービスデーモンは無効になり、コンテンツホストのメモリーおよび CPU の負荷が減ります。

yum プラグインは **katello-host-tools** に含まれます。これは、Satellite 6.2.11 クライアントアップデートに同梱されています。

B.1. 前提条件

リモート実行でホスト管理ができるようにするには、すべてのコンテンツホストで以下を行う必要があります。

- 「Katello エージェントのインストール」 に従って、コンテンツホストに **katello-agent** がインストールされていることを確認します。
- goferd サービスの停止:

```
# systemctl stop goferd.service
```

- goferd サービスの無効化:

```
# systemctl disable goferd.service
```

- 「リモートコマンドのセキュアな接続の確立」 に従って、SSH キーをコンテンツホストに配布します。

B.2. GOFERD をシステムのデフォルトとして使用しないホスト管理の設定

以下の手順は、将来のパッケージデプロイメントに使用するために、リモート実行をシステムデフォルトとして使用するようにホスト管理を設定します。

Goferd をシステムのデフォルトとして使用しないホスト管理の設定:

1. Satellite Server Web UI にログインします。
2. **管理** > **設定** に移動します。
3. **コンテンツ** タブを選択します。
4. **Use remote execution by default** パラメーターを **Yes** に設定します。

これで、Satellite サーバーは、goferd ではなく、リモート実行を介してホスト管理を使用するようになりました。

B.3. HAMMER の制限

以下は、エラータのプッシュに **hammer** コマンドを使用している場合に適用されます。**hammer** コマンドは、goferd を使用してコンテンツホストのエラータを管理しています。回避策としては、Satellite のリモート実行機能を使用して、エラータを適用します。

Hammer リモート実行コマンドの使用:

たとえば、**host123.example.org** で **`yum -y update`** を実行します。

```
# hammer job-invocation create \
--job-template "Run Command - SSH Default" \
--inputs command="yum -y update" \
--search-query "name ~ host123"
Job invocation 24 created
[.....................................] [100%]
1 task(s), 1 success, 0 fail
```