



# Red Hat Satellite 6.12

## 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 6.12 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

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

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

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

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

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 概要

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

## 目次

RED HAT ドキュメントへのフィードバック (英語のみ)	5
第1章 SATELLITE におけるホストの概要	6
第2章 ホストの管理	7
2.1. RED HAT SATELLITE でのホストの作成	7
2.2. ホストのクローン作成	9
2.3. 仮想マシンのハイパーバイザーからの SATELLITE への関連付	9
2.4. ホストのシステム目的の編集	9
2.5. 複数ホストのシステム目的の編集	10
2.6. ホストのモジュールストリームの変更	11
2.7. ホストグループの作成	11
2.8. ライフサイクル環境ごとのホストグループの作成	13
2.9. ホストグループへのホストの追加	14
2.10. ホストのホストグループの変更	14
2.11. ホストコレクションへのホストの追加	15
2.12. ホストのコンテンツソースの変更	15
2.13. ホストの環境の変更	16
2.14. ホストの管理ステータスの変更	16
2.15. ホストでトレーサーの有効化	16
2.16. ホストでのアプリケーションの再起動	17
2.17. ホストの特定組織への割り当て	17
2.18. ホストの特定ロケーションへの割り当て	18
2.19. ホスト間の切り替え	18
2.20. SATELLITE からのホストの削除	19
第3章 SATELLITE へのホストの登録	20
3.1. ホストの登録	21
3.2. 登録テンプレートのカスタマイズ	24
3.3. ホスト登録時の PUPPET AGENT のインストールと設定	26
3.4. RED HAT SATELLITE への ATOMIC HOST の登録	27
3.5. ブートストラップスクリプトを使ったホストの RED HAT SATELLITE への登録	28
3.6. KATELLO エージェントのインストール	36
3.7. トレーサーのインストール	37
3.8. PUPPET AGENT の手動でのインストールと設定	38
3.9. KATELLO エージェントからリモート実行への移行	39
第4章 ネットワークインターフェ이스の追加	42
4.1. 物理インターフェ이스の追加	42
4.2. 仮想インターフェ이스の追加	43
4.3. ボンディングインターフェ이스の追加	44
4.4. SATELLITE で利用可能なボンディングモード	45
4.5. ベースボード管理コントローラー (BMC) インターフェ이스の追加	46
第5章 ホストを次の主要な RED HAT ENTERPRISE LINUX リリースにアップグレード	48
第6章 ホストの RED HAT ENTERPRISE LINUX への変換	50
6.1. 変換データを生成するための変数	51
第7章 RHEL WEB コンソールを使用したホストの管理と監視	54
7.1. SATELLITE での RHEL WEB コンソールの有効化	54
7.2. RHEL WEB コンソールを使用したホストの管理と監視	54
7.3. SATELLITE での RHEL WEB コンソールの無効化	55

<b>第8章 RED HAT INSIGHTS を使用したホストの監視</b>	<b>56</b>
8.1. SATELLITE のホストでの RED HAT INSIGHTS の使用	56
8.2. ホストの INSIGHTS プランの作成	57
<b>第9章 レポートテンプレートを使用したホストの監視</b>	<b>59</b>
9.1. ホスト監視レポートの生成	59
9.2. レポートテンプレートの作成	60
9.3. レポートテンプレートのエクスポート	61
9.4. SATELLITE API を使用したレポートテンプレートのエクスポート	62
9.5. レポートテンプレートのインポート	63
9.6. SATELLITE API を使用したレポートテンプレートのインポート	64
9.7. エンタイトルメントを監視するレポートテンプレートの作成	65
9.8. レポートテンプレートのセーフモード	66
<b>第10章 ホストコレクションの設定</b>	<b>68</b>
10.1. ホストコレクションの作成	68
10.2. ホストコレクションのクローン作成	68
10.3. ホストコレクションの削除	68
10.4. ホストコレクションへのホストの一括追加	69
10.5. ホストコレクションからのホストの削除	69
10.6. ホストコレクションへのコンテンツの追加	70
<b>第11章 リモートジョブの設定およびセットアップ</b>	<b>75</b>
11.1. ホストでのジョブの実行について	75
11.2. リモート実行のワークフロー	75
11.3. リモート実行用のパーミッション	76
11.4. リモート実行用のトランスポートモード	77
11.5. プル要求を使用するためのホストの設定	78
11.6. ジョブテンプレートの作成	79
11.7. 名前別での ANSIBLE PLAYBOOK のインポート	80
11.8. 利用可能なすべての ANSIBLE PLAYBOOK のインポート	80
11.9. SATELLITE での任意の CAPSULE へのフォールバックリモート実行の設定	81
11.10. SATELLITE でのグローバル CAPSULE リモート実行の設定	82
11.11. 代替ディレクトリーを使用してホストでリモートジョブを実行するための SATELLITE の設定	82
11.12. リモート実行のための SSH 鍵の配布	83
11.13. リモート実行用の SSH 鍵の手動での配布	83
11.14. SATELLITE API を使用したリモート実行用の SSH 鍵の取得	83
11.15. プロビジョニング中に SSH キーを配布するためのキックスタートテンプレートの設定	84
11.16. KERBEROS チケットを付与するための KEYTAB の設定	84
11.17. リモート実行用の KERBEROS 認証の設定	85
11.18. ジョブテンプレートのセットアップ	86
11.19. リモートジョブの実行	86
11.20. ホストの定期的な ANSIBLE ジョブのスケジュール	88
11.21. ホストグループの定期的な ANSIBLE ジョブのスケジュール	88
11.22. ジョブの監視	89
<b>第12章 SATELLITE のホストステータス</b>	<b>90</b>
12.1. ホストのグローバルステータスの概要	90
12.2. ホストサブステータスの概要	90
<b>第13章 テンプレートリポジトリの同期</b>	<b>93</b>
13.1. TEMPLATESYNC プラグインの有効化	93
13.2. TEMPLATESYNC プラグインの設定	93
13.3. テンプレートのインポートおよびエクスポート	95

---

13.4. 高度な GIT 設定	99
13.5. FOREMAN テンプレートプラグインのアンインストール	100
<b>第14章 パッケージの管理</b>	<b>101</b>
14.1. ホストへのパッケージのインストール	101
14.2. ホスト上のパッケージのアップグレード	101
14.3. ホストからのパッケージの削除	101
<b>付録A テンプレート作成の参照</b>	<b>103</b>
A.1. SATELLITE WEB UI のテンプレート作成の参照へのアクセス	103
A.2. テンプレートでのオートコンプリートの使用	103
A.3. ERB テンプレートの作成	104
A.4. ERB テンプレートのトラブルシューティング	106
A.5. 一般的な SATELLITE 固有のマクロ	106
A.6. テンプレートマクロ	107
A.7. ホスト固有の変数	109
A.8. キックスタート固有の変数	113
A.9. 条件付きステートメント	113
A.10. アレイの解析	114
A.11. テンプレートスニペットの例	116
<b>付録B ジョブテンプレートの例および拡張</b>	<b>118</b>
B.1. ジョブテンプレートのカスタマイズ	118
B.2. デフォルトのジョブテンプレートカテゴリー	118
B.3. RESTORECON テンプレートの例	119
B.4. RESTORECON テンプレートのレンダリング	119
B.5. 複数のホストでの RESTORECON テンプレートの実行	120
B.6. テンプレートにパワー操作を組み込む	120





## RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

- 特定の部分について簡単なコメントをお寄せいただく場合は、以下をご確認ください。
  1. ドキュメントの表示が **Multi-page HTML** 形式になっていることを確認してください。ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
  2. マウ斯卡ーソルを使用して、コメントを追加するテキストの部分を強調表示します。
  3. 強調表示されたテキストの下に表示される **Add Feedback** ポップアップをクリックします。
  4. 表示される手順に従ってください。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
  1. [Bugzilla](#) の Web サイトに移動します。
  2. Component として **Documentation** を使用します。
  3. **Description** フィールドに、ドキュメントの改善に向けたご提案を記入してください。ドキュメントの該当部分へのリンクも追加してください。
  4. **Submit Bug** をクリックします。

## 第1章 SATELLITE におけるホストの概要

ホストは、Red Hat Satellite が管理する Linux クライアントを指します。ホストは物理サーバーでも仮想サーバーでもかまいません。仮想ホストは、Amazon EC2、Google Compute Engine、KVM、libvirt、Microsoft Azure、OpenStack、Red Hat Virtualization、Rackspace Cloud Services、VMware vSphere など、Red Hat Satellite がサポートする任意のプラットフォームにデプロイできます。

Red Hat Satellite は、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など、大規模なホスト管理が可能です。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。

Satellite Web UI では、Satellite Server が認識する全ホストをタイプ別に分類して参照できます。

- **すべてのホスト**:-: Satellite Server が認識するすべてのホストの一覧です。
- **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する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す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、検索するホストを入力し、部分一致検索には、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 は、自己登録されていない場合でも、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。

## 第2章 ホストの管理

本章では、ホストの作成、登録、管理、および削除について説明します。

### 2.1. RED HAT SATELLITE でのホストの作成

以下の手順を使用して Red Hat Satellite でホストを作成します。Satellite Web UI の代わりに CLI を使用する場合は、[CLI 手順](#)を参照してください。

#### 手順

1. Satellite Web UI で、**ホスト > ホストの作成** の順にクリックします。
2. **ホスト** タブで、必要な詳細を入力します。
3. **Ansible ロール** タブをクリックして、**Ansible ロール** リストから、ホストに追加するロールを1つまたは複数選択します。矢印 アイコンを使用して、追加または削除するロールを管理します。
4. **Puppet クラス** タブで、追加する Puppet クラスを選択します。
5. **インターフェイス** タブで、以下を行います。
  - a. 各インターフェイスに対して、**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 インターフェイスに対して、**タイプ** リストで、インターフェイ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 リスト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。または、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。または、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イスを設定します。
    - **プライマリー**: このチェックボックスを選択すると、このインターフェイスの 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イスを使用します。つまり、このインターフェイスを使用して 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イスが使用されます。**anaconda** による RPM のダウンロードや、**%post** スクリプトの Puppet 設定などの多くのプロビジョニングタスクは、プライマリーインターフェイスを使用する点にご留意ください。
  - **仮想 NIC:** このインターフェイ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。
    - **割り当て先:** この仮想インターフェイスが割り当てられるインターフェイスのデバイス名を入力します。
- b. **OK** をクリックして、インターフェイス設定を保存します。
- c. オプションとして、**インターフェイスの追加** をクリックし、追加ネットワークインターフェイスを組み込みます。詳細は、[4章 ネットワークインターフェイスの追加](#) を参照してください。
- d. **送信** をクリックし、変更を適用して終了します。
6. **オペレーティングシステム** タブで、必要な情報を入力します。Red Hat オペレーティングシステムの場合は、**メディアの選択** で **同期したコンテンツ** を選択します。Red Hat 以外のオペレーティングシステムを使用する場合には、**すべてのメディア** を選択してから、**メディアの選択** リストからインストールメディアを選択します。このリストからパーティションテーブルを選択するか、**カスタムパーティションテーブル** フィールドでカスタムのパーティションテーブルを入力します。両方は指定できません。
7. **パラメーター** タブで **パラメーターの追加** をクリックして、ランタイム時にジョブテンプレートにわたすパラメーター変数を追加します。これには、ホストに関連付ける 全 Puppet クラス、Ansible Playbook パラメーター、ホストパラメーターが含まれます。Ansible のジョブテンプレートでパラメーター変数を使用するには、**ホストパラメーター** を追加する必要があります。
- Red Hat Enterprise Linux 8 ホストの作成時には、システム目的属性を設定できます。システム目的属性は、ホストの作成時に、どのサブスクリプションを自動的にアタッチするかを定義します。**ホストパラメーター** エリアで、適切な値を指定し、以下のパラメーターを入力します。値の一覧は、**標準の RHEL 8 インストールの実行の [システムの目的の概要](#)** を参照してください。
- **syspurpose\_role**
  - **syspurpose\_sla**
  - **syspurpose\_usage**
  - **syspurpose\_addons**
- リモートジョブ実行用のプルモードでホストを作成する場合は、タイプ **boolean** を **true** に設定して **enable-remote-execution-pull** パラメーターを追加します。詳細は、「[リモート実行用のトランスポートモード](#)」を参照してください。
8. **追加情報** タブに、ホストに関する追加情報を入力します。
9. **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。

## CLI 手順

- ホストをホストグループに関連付けて作成するには、次のコマンドを入力します。

```
# hammer host create \  
--name "My_Host_Name" \  
--hostgroup "My_Host_Group" \  
--interface="primary=true, \  
    provision=true, \  
    mac=mac_address, \  
    ip=ip_address" \  
--organization "My_Organization" \  
--location "My_Location" \  
--ask-root-password yes
```

上記のコマンドを実行すると、root パスワードを指定するように求められます。ホストの IP および MAC アドレスを指定する必要があります。プライマリーのネットワークインターフェイスの他のプロパティはホストグループから継承するか、**--subnet** および **--domain** パラメーターを使用して設定することができます。**--interface** オプションを使用して追加のインターフェイスを設定できます。このオプションはキーと値のペアの一覧を受け取ります。利用可能なインターフェイス設定の一覧については **hammer host create --help** コマンドを入力します。

## 2.2. ホストのクローン作成

既存ホストのクローンを作成できます。

### 手順

1. Satellite Web UI で、**Hosts > All Hosts**に移動します。
2. **Actions** メニューで **Clone** をクリックします。
3. **Host** タブで、元のホストとは異なる **Name** を指定します。
4. **Interfaces** タブで、別の IP アドレスを指定します。
5. **Submit** をクリックしてホストのクローンを作成します。

詳細は、「[Red Hat Satellite でのホストの作成](#)」を参照してください。

## 2.3. 仮想マシンのハイパーバイザーからの SATELLITE への関連付

### 手順

1. Satellite Web UI で、**Infrastructure > Compute Resources** に移動します。
2. コンピュートリソースを選択します。
3. **仮想マシン** タブで、**アクション** メニューから **仮想マシンの関連付** をクリックします。

## 2.4. ホストのシステム目的の編集

Red Hat Enterprise Linux ホストのシステム目的属性を編集できます。システム目的により、ネット

ワーク上でシステムの使用目的を設定し、Red Hat Hybrid Cloud Console のサブスクリプションサービスでレポートの精度を向上させることができます。システムの目的の詳細は、[標準の RHEL 8 インストールの実行の システムの目的の概要](#) を参照してください。

### 前提条件

- 編集するホストが subscription-manager で登録されている。

### 手順

- Satellite Web UI で、**Hosts** > **All Hosts** に移動します。
- 変更するホストの名前をクリックします。
- Overview** タブで、**System purpose** カードの **Edit** をクリックします。
- ホストのシステム目的属性を選択します。
- Save** をクリックします。

### CLI 手順

- ホストにログインして、必要なシステム目的属性を編集します。たとえば、使用タイプを **Production**、ロールを **Red Hat Enterprise Linux Server** に設定し、**addon** アドオンを追加します。値の一覧は、[標準の RHEL 8 インストールの実行の システムの目的の概要](#) を参照してください。

```
# subscription-manager syspurpose set usage 'Production'
# subscription-manager syspurpose set role 'Red Hat Enterprise Linux Server'
# subscription-manager syspurpose add addons 'your_addon'
```

- このホストのシステム目的属性を検証します。

```
# subscription-manager syspurpose
```

- このホストに自動的にサブスクリプションをアタッチします。

```
# subscription-manager attach --auto
```

- このホストのシステム目的のステータスを検証します。

```
# subscription-manager status
```

## 2.5. 複数ホストのシステム目的の編集

Red Hat Enterprise Linux ホストのシステム目的属性を編集できます。システム目的属性は、どのサブスクリプションを自動的にホストにアタッチするかを定義します。システムの目的の詳細は、[標準の RHEL 8 インストールの実行の システムの目的の概要](#) を参照してください。

### 前提条件

- 編集するホストが subscription-manager で登録されている。

## 手順

1. Satellite Web UI で、**ホスト > コンテンツホスト** に移動し、編集する Red Hat Enterprise Linux 8 ホストを選択します。
2. **Select Action** 一覧をクリックし、**Manage System Purpose** を選択します。
3. 選択したホストに割り当てるシステム目的属性を選択します。以下の値のいずれかを選択できます。
  - 選択した全ホストを設定するための特定の属性
  - **変更なし** (選択したホストに設定された属性を保持)
  - **なし (消去)** (選択したホストの属性を消去)
4. **割り当て** をクリックします。
5. Satellite Web UI で、**Hosts > Content Hosts** に移動し、同じ Red Hat Enterprise Linux 8 ホストを選択し、システム目的に基づいてサブスクリプションを自動的にアタッチします。
6. **アクションの選択** 一覧をクリックして **サブスクリプションの管理** を選択します。
7. **自動アタッチ** をクリックして、選択した全ホストに、システムロールをもとにサブスクリプションを自動的にアタッチします。

## 2.6. ホストのモジュールストリームの変更

Red Hat Enterprise Linux 8 を実行しているホストがある場合は、インストールするリポジトリのモジュールストリームを変更できます。

Satellite Web UI で、ホストからモジュールストリームを有効化、無効化、インストール、更新、および削除できます。

## 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Content** タブをクリックし、**Module streams** タブをクリックします。
4. モジュールの横にある縦リーダーをクリックして、実行するアクションを選択します。リモート実行ジョブが完了すると、REX ジョブ通知が表示されます。

## 2.7. ホストグループの作成

多数のホストを作成する場合には、ホストの多くに、共通の設定と属性を指定できます。新規ホストすべてにこれらの設定および属性を追加するのは時間がかかります。ホストグループを使用する場合には、作成するホストに対して、共通の属性を適用できます。

ホストグループは、共通するホスト設定のテンプレートとして機能し、ホストに指定する同じ情報が多数含まれます。ホストグループを指定して、ホストを作成する場合には、このホストは、ホストグループで定義した設定を継承します。その後に、追加の情報を指定して、ホストを個別化できます。

Satellite Web UI の代わりに CLI を使用する場合は、[CLI 手順](#) を参照してください。

## ホストグループの階層

ホストグループには、階層を作成できます。組織内の全ホストを表すベースレベルのホストグループを設定し、汎用的な設定を行い、その中のネストされたグループを指定して、固有の設定を指定するようにします。たとえば、以下のように、オペレーティングシステムを定義する Base レベルのホストグループ1つおよび、Base レベルのホストグループを継承するネスト化されたホストグループ2つを設定できます。

- **Hostgroup: Base** (Red Hat Enterprise Linux 7.6)
  - **Hostgroup: Webserver** (nginx Puppet クラスを適用)
    - **Host: webserver1.example.com** (Web サーバー)
    - **Host: webserver2.example.com** (Web サーバー)
  - **Hostgroup: Storage** (nfs Puppet クラスを適用)
    - **Host: storage1.example.com** (ストレージサーバー)
    - **Host: storage2.example.com** (ストレージサーバー)
  - **Host: custom.example.com** (カスタムホスト)

この例では、すべてのホストは **Base** ホストグループの継承により、Red Hat Enterprise Linux 7.6 をオペレーティングシステムとして使用します。2つの Web サーバーホストは **Webserver** ホストグループからの設定を継承します。これには、**nginx** Puppet クラスおよび **Base** ホストグループの設定が含まれます。2つのストレージサーバーは、**Storage** ホストグループからの設定を継承します。これには、**nfs** Puppet クラスおよび **Base** ホストグループからの設定が含まれます。カスタムホストは **Base** ホストグループからの設定のみを継承します。

## 手順

1. Satellite Web UI で **設定 > ホストグループ** に移動して、**ホストグループの作成** をクリックします。
2. 属性を継承する既存のホストグループがある場合には、**親** リストからホストグループを選択します。継承する属性がない場合には、このフィールドは空白のままにします。
3. 新規ホストグループの **名前** を入力します。
4. 新たに作成するホストに継承させる情報をさらに入力します。
5. **Ansible ロール** タブをクリックして、**Ansible ロール** リストから、ホストに追加するロールを1つまたは複数選択します。矢印 アイコンを使用して、追加または削除するロールを管理します。
6. 追加タブをクリックして、ホストグループに属性として指定する情報を追加します。





### 注記

Puppet は、**Production** 環境内に作成した Puppet 環境に関連付けられているホストグループにホストを登録すると、Puppet CA 証明書の取得に失敗します。

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、ディレクトリーを手動で作成します。

```
# mkdir /etc/puppetlabs/code/environments/example_environment
```

7. **送信** をクリックしてホストグループを保存します。

### CLI 手順

- **hammer hostgroup create** コマンドでホストグループを作成します。以下に例を示します。

```
# hammer hostgroup create --name "Base" \
--architecture "My_Architecture" \
--content-source-id _My_Content_Source_ID_ \
--content-view "_My_Content_View_" \
--domain "_My_Domain_" \
--lifecycle-environment "_My_Lifecycle_Environment_" \
--locations "_My_Location_" \
--medium-id _My_Installation_Medium_ID_ \
--operatingsystem "_My_Operating_System_" \
--organizations "_My_Organization_" \
--partition-table "_My_Partition_Table_" \
--puppet-ca-proxy-id _My_Puppet_CA_Proxy_ID_ \
--puppet-environment "_My_Puppet_Environment_" \
--puppet-proxy-id _My_Puppet_Proxy_ID_ \
--root-pass "My_Password" \
--subnet "_My_Subnet_"
```

## 2.8. ライフサイクル環境ごとのホストグループの作成

以下の手順を使用して、ライブラリーライフサイクル環境のホストグループを作成し、他のライフサイクル環境向けに、ネストされたホストグループを追加します。

### 手順

ライフサイクル環境ごとにホストグループを作成するには、以下の Bash スクリプトを実行します。

```
MAJOR="My_Major_OS_Version"
ARCH="My_Architecture"
ORG="My_Organization"
LOCATIONS="My_Location"
PTABLE_NAME="My_Partition_Table"
DOMAIN="My_Domain"
```

```
hammer --output csv --no-headers lifecycle-environment list --organization "${ORG}" | cut -d ',' -f 2 |
while read LC_ENV; do
  [[ "${LC_ENV}" == "Library" ]] && continue

  hammer hostgroup create --name "rhel-${MAJOR}server-${ARCH}-${LC_ENV}" \
```

```
--architecture "${ARCH}" \  
--partition-table "${PTABLE_NAME}" \  
--domain "${DOMAIN}" \  
--organizations "${ORG}" \  
--query-organization "${ORG}" \  
--locations "${LOCATIONS}" \  
--lifecycle-environment "${LC_ENV}"  
done
```

## 2.9. ホストグループへのホストの追加

Satellite Web UI でホストグループにホストを追加できます。

### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Edit** ボタンをクリックします。
4. **Host Group** リストからホストグループを選択します。
5. **Submit** をクリックします。

### 検証

- **Overview** タブの **Details** カードに、ホストが属するホストグループが表示されるようになります。

## 2.10. ホストのホストグループの変更

以下の手順を使用して、ホストのホストグループを変更します。

ホストグループを変更した後にホストを再プロビジョニングすると、ホストがホストグループから継承する新しい値が適用されます。

### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Edit** ボタンをクリックします。
4. **Host Group** リストから新規ホストグループを選択します。
5. **Submit** をクリックします。

### 検証

- **Overview** タブの **Details** カードに、ホストが属するホストグループが表示されるようになります。

## 2.11. ホストコレクションへのホストの追加

Satellite Web UI で、ホストをホストコレクションに追加できます。

### 前提条件

ホストコレクションに追加するために、ホストを Red Hat Satellite に登録しておく。ホストの登録に関する情報は、「[ホストの登録](#)」を参照してください。

ホストをホストコレクションに追加すると、Satellite 監査システムでは変更がログに記録されないので注意してください。

### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Host collections** カードで縦リーダーをクリックして **Add host to collections** を選択します。
4. ホストコレクションを選択します。
5. **Add** をクリックします。

### CLI 手順

- ホストコレクションにホストを追加するには、以下のコマンドを入力します。

```
# hammer host-collection add-host \  
--host-ids My_Host_ID_1 \  
--id My_Host_Collection_ID
```

## 2.12. ホストのコンテンツソースの変更

コンテンツソースは、ホストがコンテンツを消費する Capsule です。以下の手順を使用して、ホストのコンテンツソースを変更します。

### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Edit** ボタンの横にある縦リーダーアイコンをクリックし、**Change content source** を選択します。
4. 一覧から Environment、Content View、および Content Source を選択します。
5. **コンテンツソースの変更** をクリックします。

リモート実行を使用するか、手動でコンテンツソースの変更を完了することができます。リモート実行を使用してホストの設定を更新するには、**Run job invocation** をクリックします。リモート実行ジョブの実行の詳細については、[リモートジョブの設定と設定](#) を参照してください。コンテンツソースを手動で更新するには、ホストの **コンテンツソースの変更** から自動生成されたコマンドを実行します。

## 2.13. ホストの環境の変更

以下の手順を使用して、ホストの環境を変更します。

### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。
2. 変更するホストの名前をクリックします。
3. **コンテンツビューの詳細** カードの縦リーダーをクリックし、**Edit content view assignment** を選択します。
4. 環境を選択します。
5. コンテンツビューを選択します。
6. **Save** をクリックします。

## 2.14. ホストの管理ステータスの変更

デフォルトでは、Satellite がプロビジョニングするホストは管理対象となっています。ホストを管理対象に設定した場合には、Satellite Server からホストパラメーターを追加で設定できます。このように追加したパラメーターは、**オペレーティングシステム** タブに表示されます。**オペレーティングシステム** タブで設定を変更した場合には、ホストをビルドして再起動するように設定しない限り、これらの変更は適用されません。

Satellite のサポート対象外の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設定します。

### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。
2. 変更するホストの名前をクリックします。
3. **Edit** ボタンをクリックします。
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. **Submit** をクリックします。

## 2.15. ホストでトレーサーの有効化

以下の手順を使用して、Satellite でトレーサーを有効にし、トレースにアクセスします。トレーサーは、再起動する必要があるサービスおよびアプリケーションの一覧を表示します。トレースは、Satellite Web UI でトレーサーが生成する出力です。

### 前提条件

- Satellite Client 6 リポジトリの同期
- Satellite Client 6 リポジトリは、コンテンツビューおよびホストのライフサイクル環境で利用できます。

- Satellite Client 6 リポジトリがホストに対して有効になっている。
- リモート実行が有効になっている。

#### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Traces** タブをクリックし、**Enable Traces** ボタンをクリックします。
4. 一覧から **katello-host-tools-tracer** をインストールするプロバイダーを選択します。
5. **Enable Tracer** ボタンをクリックします。リモート実行ジョブが完了すると、REX ジョブ通知が表示されます。

## 2.16. ホストでのアプリケーションの再起動

以下の手順を使用して、Satellite Web UI からアプリケーションを再起動します。

#### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Traces** タブを選択します。
4. 再起動するアプリケーションを選択します。
5. **Restart app** リストから **Restart via remote execution** を選択します。リモート実行ジョブが完了すると、REX ジョブ通知が表示されます。

## 2.17. ホストの特定組織への割り当て

以下の手順を使用して、ホストを特定の組織に割り当てます。組織とその設定方法に関する一般的な情報については、**Red Hat Satellite の管理** の [組織の管理](#) を参照してください。



#### 注記

ホストがすでに別の組織に登録されている場合は、ホストを新しい組織に割り当てる前に、まずそのホストを登録解除する必要があります。ホストを登録解除するには、ホストで **subscription-manager unregister** を実行します。ホストを新しい組織に割り当てた後、ホストを再登録できます。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。
2. 変更する必要があるホストのチェックボックスを選択します。
3. **アクションの選択** 一覧から **組織の割り当て** を選択すると、新しいオプションウィンドウが開きます。

4. **組織の選択** リストから、ホストを割り当てる組織を選択します。 **Fix Organization on Mismatch** チェックボックスを選択します。



#### 注記

ドメインまたはサブネットなど、ホストに関連付けるリソースがあるにもかかわらず、こ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に、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)** オプションを使用すると、このようなリソースが組織に追加されるので、このオプションは推奨の選択肢になります。**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、実際には設定に不一致がない場合でも、別の組織にホストを再割り当てすると失敗します。

5. **Submit** をクリックします。

## 2.18. ホストの特定ロケーションへの割り当て

以下の手順を使用して、ホストを特定のロケーションに割り当てます。ロケーションに関する一般的な情報および設定方法は、**コンテンツの管理** の [ロケーションの作成](#) を参照してください。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。
2. 変更する必要があるホストのチェックボックスを選択します。
3. **アクションの選択** リストから **ロケーションの割り当て** を選択すると、新しい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** リストに移動して、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** チェックボックスを選択します。




#### 注記

ドメインまたはサブネットなど、ホストに関連付けるリソースがあるにもかかわらず、これらのリソースがホストの割り当て先のロケーションに割り当てられていない場合に、不一致が生じます。**Fix Location on Mismatch (ロケーションの不一致についての修正)** オプションを使用すると、このようなリソースがロケーションに追加されるので、このオプションは推奨の選択肢になります。**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、実際には設定に不一致がない場合でも、別のロケーションにホストを再割り当てすると失敗します。

5. **Submit** をクリックします。

## 2.19. ホスト間の切り替え

Satellite Web UI で特定のホストを使用している場合は、ホストスイッチャーを使用してページを離れることなくホスト間を移動できます。ホスト名の横にある  をクリックします。これにより、ホストのリストがアルファベット順に表示され、ページ区切りの矢印と、探しているホストを見つけるための検索バーが表示されます。

## 2.20. SATELLITE からのホストの削除

以下の手順を使用して Red Hat Satellite からホストを削除します。

### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** または **ホスト > コンテンツホスト** へ移動します。**すべてのホスト** または **コンテンツホスト** のどちらのページからホストを削除しても相違はないので注意してください。どちらの場合も、Satellite はホストを完全に削除します。
2. 削除するホストを選択します。
3. **アクションの選択** リストから **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 からホストを完全に削除します。



### 警告

デフォルトでは、**Destroy associated VM on host delete** は **no** に設定されます。仮想マシンに関連付けられたホストレコードが削除されると、仮想マシンはコンピュータリソースに残ります。

コンピュータリソースで仮想マシンを削除するには、**Administer > Settings** に移動して、**Provisioning** タブを選択します。**Destroy associated VM on host delete** を **yes** に設定すると、仮想マシンレコードが関連付けられている仮想マシンが削除されると、仮想マシンが削除されます。このような状況で仮想マシンが削除されないようにするには、コンピュータリソースから仮想マシンを削除せずに Satellite から仮想マシンの関連付けを解除するか、設定を変更します。

### 2.20.1. 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する方法

### 手順

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** リストから **ホストの関連付けを解除** ボタンを選択します。
3. オプションで、チェックボックスを選択して、今後のためにホストを保存します。
4. **Submit** をクリックします。

## 第3章 SATELLITE へのホストの登録

Satellite Server または Capsule Server にホストを登録する方法は、主に 3 つあります。

1. **(デフォルトの方法)** Satellite から **curl** コマンドを生成し、ホスト (数に制限なし) からこのコマンドを実行して、これらのホストをグローバル登録テンプレートを使用して登録します。この方法は、Satellite にまだ登録されていないホストに適しています。  
この方法を使うと、Satellite への登録時に Satellite SSH 鍵をホストにデプロイし、リモート実行ジョブのホストを有効にすることもできます。リモート実行ジョブの詳細は、[リモートジョブの設定とセットアップ](#) を参照してください。

この方法を使うと、Satellite への登録時に Red Hat Insights でホストを設定することもできます。Satellite ホストと Insights を使用する方法は、[Red Hat Insights を使用したホストの管理](#) を参照してください。

- 2. **(非推奨)** コンシューマー RPM ([satellite.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てから、Subscription Manager を実行します。この方法は、Satellite を介してプロビジョニングされていないホストに適しています。
- 3. **(非推奨)** ブートストラップスクリプト ([satellite.example.com/pub/bootstrap.py](#)) をダウンロードして実行します。この方法は、Satellite を介してプロビジョニングされていないホストに適しています。

Atomic Host を Satellite Server または Capsule Server に登録することもできます。

次のいずれかの手順を使用して、ホストを登録します。

- [「ホストの登録」](#)
- [「Red Hat Satellite への Atomic Host の登録」](#)
- [「ブートストラップスクリプトを使ったホストの Red Hat Satellite への登録」](#)

以下の手順を使用して、ホストツールをインストールして設定します。

- [「Katello エージェントのインストール」](#)
- [「トレーサーのインストール」](#)
- [「Puppet Agent の手動でのインストールと設定」](#)

### 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 バージョンのいずれかを使用する必要があります。

- 9.0 以降
- 8.0 以降
- 16.1 以降
- 6 ([ELS アドオン](#) 使用)

資格ベースのサブスクリプションモデルは非推奨となり、将来のリリースで削除される予定であることに注意してください。Red Hat では、代わりに [Simple Content Access](#) のアクセスベースのサブスクリプションモデルを使用することを推奨します。



## 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

## 3.1. ホストの登録

Satellite で **curl** コマンドを生成し、ホストでこのコマンドを実行することで、ホストを Satellite に登録できます。この方法では、**global registration** テンプレートと **host initial configuration** という2つのテンプレートを使用します。これにより、ホスト登録プロセスを完全に制御できます。**Administer > Settings** に移動し、**Provisioning** タブをクリックすると、デフォルトのテンプレートを設定できます。

### 前提条件

- **curl** コマンドを生成する Satellite ユーザーには、**create\_hosts** パーミッションが必要。
- 登録するホストで root 権限がある。
- Satellite Server、Capsule Server、およびすべてのホストを同じ NTP サーバーと同期し、時間同期ツールを有効にして実行しておく。
- **rhsmcertd** デーモンをホストで実行しておく。
- ホストのアクティベーションキーがある。詳細は、[コンテンツの管理](#) の [アクティベーションキーの管理](#) を参照してください。
- オプション: ホストを Red Hat Insights に登録する場合は、**rhel-8-for-x86\_64-baseos-rpms** リポジトリと **rhel-8-for-x86\_64-appstream-rpms** リポジトリを同期し、アクティベーションキーでできるようにする必要があります。これは、**insights-client** パッケージをインストールするために必要です。
- Satellite Server の代わりに Capsule Server を使用する場合は、Capsule Server が適切に設定されていることを確認してください。詳細は、[Capsule Server のインストール](#) の [ホスト登録およびプロビジョニングのための Capsule の設定](#) を参照してください。

### 手順

1. Satellite Web UI で、**Hosts > Register Host** に移動します。
2. オプション: 別の **Organization** を選択します。
3. オプション: 別の **Location** を選択します。
4. オプション: **Host Group** リストから、ホストと関連付けるホストグループを選択します。**Host group** からの値を継承するフィールド: **Operating system**、**Activation Keys**、**Lifecycle environment**。

5. オプション: **Operating system** リストから、登録するホストのオペレーティングシステムを選択します。
6. オプション: **Capsule** 一覧から、ホストの登録に使用する Capsule を選択します。

7. オプション: 最初の呼び出しを非セキュアにする場合は、**Insecure** オプションを選択します。この最初の呼び出し中に、ホストは Satellite から CA ファイルをダウンロードします。ホストは、この CA ファイルを使用して Satellite に接続し、今後のすべての呼び出しは安全になります。

Red Hat は、セキュアでない呼び出しを回避することを推奨します。

Satellite とホスト間のネットワークに存在する攻撃者が、初回の安全ではない呼び出しから CA ファイルをフェッチする場合、攻撃者は登録したホストと JSON Web Tokens (JWT) 間の API 呼び出しの内容にアクセスできます。そのため、登録中の SSH 鍵のデプロイを選択した場合、攻撃者は SSH 鍵を使用してホストにアクセスできます。

代わりに、ホストを登録する前に、各ホストに手動で CA ファイルをコピーしてインストールできます。

そのためには、**Administer > Settings > Authentication** に移動し、**SSL CA file** 設定を特定することで、Satellite が CA ファイルを保存する場所を検出します。

CA ファイルをホストの `/etc/pki/ca-trust/source/anchors/` ディレクトリーにコピーし、以下のコマンドを入力します。

```
# update-ca-trust enable
# update-ca-trust
```

次に、以下のようなセキュアな **curl** コマンドを使用してホストを登録します。

```
# curl -sS https://satellite.example.com/register ...
```

以下は、**--insecure** オプションを指定した **curl** コマンドの例です。

```
# curl -sS --insecure https://satellite.example.com/register ...
```

8. **Advanced** タブを選択します。
9. **Setup REX** リストから、Satellite SSH 鍵をホストにデプロイするかどうかを選択します。**Yes** に設定すると、パブリック SSH キーが登録済みホストにインストールされます。継承された値は **host\_registration\_remote\_execution** パラメーターに基づいています。たとえば、ホストグループ、オペレーティングシステム、または組織から継承できます。上書きされると、選択した値がホストパラメーターレベルに保存されます。
10. **Setup Insights** | 一覧から、**insights-client** をインストールして、ホストを Insights に登録したかどうかを選択します。  
Insights ツールは、Red Hat Enterprise Linux でのみ利用できます。他のオペレーティングシステムには影響ありません。

登録されたマシンで以下のリポジトリを有効にする必要があります。

- Red Hat Enterprise Linux 6: **rhel-6-server-rpms**
- Red Hat Enterprise Linux 7: **rhel-7-server-rpms**

- Red Hat Enterprise Linux 8: **rhel-8-for-x86\_64-appstream-rpms**

**Insights-client** パッケージは、Red Hat Enterprise Linux 8 が最小インストールオプションでデプロイされた環境を除き、デフォルトで Red Hat Enterprise Linux 8 にインストールされます。

- オプション: **Install packages** フィールドに、登録時にホストにインストールするパッケージを(スペースで区切って)リストします。これは、**host\_packages** パラメーターで設定できます。
- オプション: 登録時にホスト上のすべてのパッケージを更新するには、**Update packages** オプションを選択します。これは、**host\_update\_packages** パラメーターで設定できます。
- オプション: **Repository** フィールドに、登録を実行する前に追加するリポジトリを入力します。たとえば、登録の目的で **subscription-manager** パッケージを利用できるようにすると便利です。Red Hat ファミリーのディストリビューションの場合、リポジトリの URL を入力します (例: **:http://rpm.example.com/**)。
- オプション: **Repository GPG key URL** フィールドで、公開鍵を指定して、GPG 署名付きパッケージの署名を確認します。GPG 公開鍵ヘッダーを持つ ASCII 形式で指定する必要があります。
- オプション: **Token lifetime (hours)** フィールドで、Satellite が認証に使用する JSON Web トークン (JWT) の有効期間を変更します。このトークンの期間は、生成された **curl** コマンドが機能する期間を定義します。期間は、0 - 999 999 時間または無制限に設定できます。  
Satellite は、ホストの認証に **curl** コマンドを生成するユーザーのパーミッションを適用する点に注意してください。ユーザーが追加のパーミッションを失ったり取得したりすると、JWT のパーミッションも変わってきます。そのため、トークン期間中にユーザーのパーミッションを削除、ブロック、または変更しないでください。  
  
JWT の範囲は登録エンドポイントのみに制限されているため、その他の場所では使用できません。
- オプション: **リモート実行インターフェイス** フィールドに、ホストが SSH 接続に使用する必要があるネットワークインターフェイスの識別子を入力します。このフィールドが空の場合、Satellite はデフォルトのネットワークインターフェイスを使用します。
- REX プルモード** リストから、Satellite リモート実行プルクライアントをデプロイするかどうかを選択します。  
**Yes** に設定すると、リモート実行のプルクライアントは登録されたホストにインストールされます。継承された値は **host\_registration\_remote\_execution\_pull** パラメーターに基づいています。たとえば、ホストグループ、オペレーティングシステム、または組織から継承できます。上書きされると、選択した値はホストパラメーターレベルに保存されます。  
  
登録したホストは、Red Hat Satellite Client 6 リポジトリにアクセスする必要があります。  
  
プルモードの詳細は、[「リモート実行用のトランスポートモード」](#) を参照してください。
- Activation Keys** フィールドで、ホストに割り当てるアクティベーションキーを1つ以上入力します。
- オプション: **Lifecycle environment** を選択します。
- オプション: サブスクリプションマネージャーのエラーを無視する場合は、**Ignore errors** オプションを選択します。
- オプション: 登録前に **katello-ca-consumer rpm** を削除し、**--force** 引数で **subscription-manager** を実行する場合は、**Force** オプションを選択します。

22. **Generate** ボタンをクリックします。
23. 生成された **curl** コマンドをコピーします。
24. 登録するホストで、**curl** コマンドを **root** として実行します。

## 3.2. 登録テンプレートのカスタマイズ

登録プロセスをカスタマイズする場合は、本セクションの情報を使用します。

Satellite のデフォルトテンプレートはすべてロックされている点に注意してください。登録プロセスをカスタマイズする場合は、デフォルトのテンプレートのクローンを作成し、クローンを編集する必要があります。次に、**Administer > Settings > Provisioning** で、**デフォルトのグローバル登録テンプレート**と、カスタムテンプレートを参照するように**デフォルトの'Host initial configuration'テンプレート**を変更します。

### テンプレート

登録プロセスでは、以下の登録テンプレートを使用します。

- **グローバル登録** テンプレートには、ホストを Satellite に登録するための手順が含まれています。このテンプレートは、ホストが **/register** エンドポイントにアクセスするとレンダリングされます。
- **Linux host\_init\_config default**テンプレートには、登録後にホストの初期設定を行う手順が含まれます。

### グローバルパラメーター

以下のグローバルパラメーターを設定するには、**設定 > グローバルパラメーター** に移動します。

- **host\_registration\_remote\_execution** パラメーターは **remote\_execution\_ssh\_keys** スニペットで使用されます。デフォルト値は **true** です。
- **host\_registration\_insights** パラメーターは **insights** スニペットで使用され、デフォルト値は **true** です。パラメーター値をオーバーライドするには、パラメーターのタイプを **boolean** に設定します。このパラメーターが **false** に設定されていると、Satellite と Insights クライアントがインベントリレポートを Red Hat Hybrid Cloud Console にアップロードできなくなります。
- **host\_packages** パラメーターは、ホストにパッケージをインストールするためのものです。
- **remote\_execution\_ssh\_keys**、**remote\_execution\_ssh\_user**、**remote\_execution\_create\_user**、**remote\_execution\_effective\_user\_method** パラメーターは **remote\_execution\_ssh\_keys** で使用されます。詳細は、スニペットの詳細を参照してください。
- **encode\_grub** パラメーターは、ホストの暗号化されたブートローダーパスワードの設定を有効にするためのものであり、デフォルト値は **false** です。  
実際にパスワードを設定するには、テンプレートで **grub\_pass** マクロを使用します。

### スニペット

スニペットは、**Linux host\_init\_config default**テンプレートで使用されます。

- **remote\_execution\_ssh\_keys** スニペットは、**host\_registration\_remote\_execution** パラメーターが **true** の場合にのみ、SSH キーをホストにデプロイします。

- **insights** スニペットと、グローバルパラメーター **host\_registration\_insights** が true に設定されている場合、Red Hat Insights クライアントをダウンロードおよびインストールします。
- **puppetlabs\_repo** および **puppet\_setup** スニペットは、ホストに Puppet エージェントをダウンロードしてインストールします (Puppet サーバーが割り当てられている場合のみ)
- **host\_init\_config\_post** は、ホストの初期設定時のユーザーのカスタムアクションの空のスニペットです。

## Variables

この表は、**Global Registration** テンプレートで使用される変数について説明しています。

表3.1 グローバル登録テンプレート変数

変数	コマンド引数	説明
<b>@user</b>	<b>none</b>	現在の認証ユーザーオブジェクト
<b>@organization</b>	<b>organization_id</b>	<b>organization_id</b> が設定されていない場合は、ユーザーのデフォルトの組織が設定されているか、ユーザーの組織リストにある最初の組織が設定されます。
<b>@location</b>	<b>location_id</b>	<b>location_id</b> が設定されていない場合、ユーザーのデフォルトのロケーションが設定されます。または、ユーザーのロケーションリストにある最初のロケーションが設定されます。
<b>@hostgroup</b>	<b>hostgroup_id</b>	ホストのホストグループです。
<b>@operatingsystem</b>	<b>operatingsystem_id</b>	ホストのオペレーティングシステム
<b>@setup_insights</b>	<b>setup_insights</b>	登録したホストの <b>host_registration_insights</b> グローバルパラメーターの値を上書きし、Insights クライアントをインストールします。
<b>@setup_remote_execution</b>	<b>setup_remote_execution</b>	登録済みホストの <b>host_registration_remote_execution</b> グローバルパラメーターの値を上書きし、リモート実行用に SSH 鍵をデプロイします。
<b>@setup_remote_execution</b>	<b>setup_remote_execution</b>	リモート実行用に、ホストのデフォルトインターフェイスを設定します。

変数	コマンド引数	説明
@packages	packages	インストールするパッケージ
@repo	repo	ホストにリポジトリを追加します。
@repo_gpg_key_url	repo_gpg_key_url	リポジトリ GPG キー形式の URL を設定します。
@activation_keys	activation_keys	ホストのアクティベーションキー
@force	force	--force 引数を指定して、 <b>katello-ca-consumer*</b> rpm を削除し、 <b>subscription-manager register</b> コマンドを実行します。
@ignore_subman_errors	ignore_subman_errors	subscription-manager エラーを無視します。
@lifecycle_environment_id	lifecycle_environment_id	ライフサイクル環境
@registration_url	none	<b>/register</b> エンドポイントの URL。

### 3.3. ホスト登録時の PUPPET AGENT のインストールと設定

登録時に、ホストに Puppet エージェントをインストールして設定できます。Satellite と Puppet を統合するには、設定済みの Puppet エージェントがホストに必要です。Puppet の詳細は、[Red Hat Satellite での Puppet 統合を使用した設定の管理](#) を参照してください。

#### 前提条件

- Satellite で Puppet が有効になっている。詳細は、[Managing Configurations Using Puppet Integration in Red Hat Satellite](#) の [Enabling Puppet Integration with Satellite](#) を参照してください。
- **Satellite Client 6** リポジトリを有効化して Satellite に同期している。詳細は、[コンテンツの管理](#) の [コンテンツのインポート](#) を参照してください。
- ホストの **Satellite Client 6** リポジトリを有効化するアクティベーションキーを作成している。詳細は、[コンテンツの管理](#) の [アクティベーションキーの管理](#) を参照してください。

#### 手順

1. Satellite Web UI で、**Configure > Global Parameters** に移動して、ホストパラメーターをグローバルに追加します。あるいは、**Configure > Host Groups** に移動し、ホストグループを編集または作成して、ホストパラメーターをホストグループにのみ追加することもできます。

2. グローバルパラメーターまたはホストグループのホストパラメーターを使用して、Puppet エージェントを有効化します。**Enable-puppet7** という名前のホストパラメーターを追加し、**boolean** タイプを選択して、値を **true** に設定します。
3. グローバルパラメーターまたはホストグループで次のホストパラメーターを使用して、Puppet エージェントの設定を指定します。
  - **puppet\_server** という名前のホストパラメーターを追加し、**string** タイプを選択して、値を Puppet サーバーのホスト名 (**puppet.example.com** など) に設定します。
  - オプション: **puppet\_ca\_server** という名前のホストパラメーターを追加し、**string** タイプを選択して、値を Puppet CA サーバーのホスト名 (**puppet-ca.example.com** など) に設定します。**puppet\_ca\_server** が設定されていない場合、Puppet エージェントは **puppet\_server** と同じサーバーを使用します。
  - オプション: **puppet\_environment** という名前のホストパラメーターを追加し、**string** タイプを選択して、ホストで使用する Puppet 環境に値を設定します。

[BZ2177730](#) が解決されるまでは、Puppet サーバーが Capsule Server となっている統合セットアップでも、ホストパラメーターを使用して Puppet エージェント設定を指定する必要があります。

4. **Hosts > Register Host** に移動し、適切なアクティベーションキーを使用してホストを登録します。詳細は、**ホストの管理** の [ホストの登録](#) を参照してください。
5. **Infrastructure > Capsules** に移動します。
6. 必要な Capsule Server の **Actions** コラムの一覧から、**Certificates** を選択します。
7. 必要なホストの右にある **Sign** をクリックして、Puppet エージェントの SSL 証明書に署名します。

### 3.4. RED HAT SATELLITE への ATOMIC HOST の登録

以下の手順を使用して、Atomic Host を Red Hat Satellite に登録します。

#### 手順

1. **root** ユーザーで、Atomic Host にログインします。
2. Satellite Server から **katello-rhsm-consumer** を取得します。

```
# wget http://satellite.example.com/pub/katello-rhsm-consumer
```

3. **katello-rhsm-consumer** のモードを実行可能に変更します。

```
# chmod +x katello-rhsm-consumer
```

4. **katello-rhsm-consumer** を実行します。

```
# ./katello-rhsm-consumer
```

5. サブスクリプションマネージャーに登録します。

```
# subscription-manager register
```

**注記**

Katello エージェントは、Atomic Host ではサポートされません。

## 3.5. ブートストラップスクリプトを使ったホストの RED HAT SATELLITE への登録

**非推奨** グローバル登録機能を使用した登録を使用します。

ブートストラップスクリプトを使用して、コンテンツの登録と Puppet の設定を自動化します。ブートストラップスクリプトを使用して、新しいホストを登録したり、RHN、SAM、RHSM、または別の RedHatSatellite インスタンスから既存のホストを移行したりできます。

Satellite Server のベースオペレーティングシステムに、デフォルトで **katello-client-bootstrap** パッケージがインストールされています。**bootstrap.py** 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ており、**satellite.example.com/pub/bootstrap.py** でホストに公開されます。このスクリプトでは、`/usr/share/doc/katello-client-bootstrap-version/README.md` ファイルにドキュメントが含まれます。

ブートストラップスクリプトを使用するには、ホストにスクリプトをインストールする必要があります。スクリプトは1度しか必要ではなく、また、**root** ユーザー専用であるため、`/root` または `/usr/local/sbin` に配置して、使用後に削除できます。この手順では、`/root` を使用します。

### 前提条件

- Satellite ユーザーに、ブートストラップスクリプト実行に必要なパーミッションを割り当て済みである。この手順の例では、**admin** ユーザーを指定します。セキュリティポリシーの関係上、この要件を満たせない場合には、新しいロールを作成して最小限必要なパーミッションを割り当てて、スクリプトを実行するユーザーにこのロールを追加してください。詳細は、「[ブートストラップスクリプトのパーミッションの設定](#)」を参照してください。
- Satellite Client 6 リポジトリを有効にしたホストのアクティベーションキーを用意する。アクティベーションキーの設定方法は、[コンテンツの管理](#) 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ある。ホストグループの作成方法は「[ホストグループの作成](#)」を参照してください。

### Puppet の考慮事項

ホストグループを **Production** 環境内に作成した Puppet 環境に関連付けると、Puppet はホストグループからホストを登録する時に Puppet CA 証明書の取得に失敗します。

ホストグループに関連付けて、適切な Puppet 環境を作成するには、以下の手順を実行します。

1. ディレクトリーを手動で作成します。

```
# mkdir /etc/puppetlabs/code/environments/example_environment
```

2. Satellite Web UI で、**Configure > Environments** に移動し、**Import environment from** 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。

### 手順



1. **root** ユーザーで、ホストにログインします。

2. スクリプトをダウンロードします。

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

3. スクリプトを実行可能にします。

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

4. ヘルプテキストを表示して、スクリプトが実行可能であることを確認します。

- Red Hat Enterprise Linux 8 の場合:

```
# /usr/libexec/platform-python bootstrap.py -h
```

- 他の Red Hat Enterprise Linux バージョンの場合:

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

5. 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入力します。

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

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key"
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# ./bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key"
```

6. **--login** オプションで指定した Satellite ユーザーのパスワードを入力します。

スクリプトは、進捗の通知を **stdout** に送信します。

7. スクリプトでプロンプトが表示されたら、ホストの Puppet 証明書を承認します。Satellite Web UI で **Infrastructure > Capsules** に移動して、**--server** オプションで指定した Satellite または Capsule Server を検出します。

8. **アクション** コラムの一覧から、**証明書** を選択します。

9. **アクション** コラムで、**署名** をクリックして、ホストの Puppet 証明書を承認します。
10. 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。
11. Satellite 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。
12. オプション: ホストの登録が完了したら、スクリプトを削除します。

```
# rm bootstrap.py
```

### 3.5.1. ブートストラップスクリプトのパーミッションの設定

以下の手順を使用して、Satellite ユーザーにブートストラップスクリプトの実行に必要なパーミッションを設定します。Satellite Web UI の代わりに CLI を使用する場合は、[CLI 手順](#) を参照してください。

#### 手順

1. Satellite Web UI で、**Administer > Users** に移動します。
2. 必要な **ユーザー名** をクリックして既存のユーザーを選択すると、選択したユーザーの情報を変更するタブが含まれる、新しいペインが表示されます。または、このスクリプトの実行専用 to 新規ユーザーを作成します。
3. **ロール** タブをクリックします。
4. **ロール** リストから **ホストの編集** および **ビューワー** を選択します。

#### 重要

**ホストの編集** ロールを割り当てると、ユーザーは、ホストの編集や削除、ホストの追加が可能です。セキュリティポリシーの関係上、この方法を使用できない場合は、以下のパーミッションを割り当てた新しいロールを作成して、このロールをユーザーに割り当ててください。

- **view\_organizations**
- **view\_locations**
- **view\_domains**
- **view\_hostgroups**
- **view\_hosts**
- **view\_architectures**
- **view\_ptables**
- **view\_operatingsystems**
- **create\_hosts**

5. **Submit** をクリックします。

## CLI 手順

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。

```
# 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
```

または、新規ユーザーを作成して、新しいロールを新規ユーザーに割り当てることもできます。Hammer を使用したユーザーの作成の詳細については、[Red Hat Satellite の管理のユーザーとロールの管理](#) を参照してください。

### 3.5.2. 詳細なブートストラップスクリプトの設定

以下のセクションでは、ブートストラップスクリプトを使用してホストを登録したり、移行したりする例をさらに紹介します。



#### 警告

以下の例では、**admin** Satellite ユーザーを指定します。セキュリティポリシーの関係上、この要件を満たせない場合には、新しいロールを作成してブートストラップスクリプトで最小限必要なパーミッションを割り当ててください。詳細は、[「ブートストラップスクリプトのパーミッションの設定」](#) を参照してください。

#### 3.5.2.1. Satellite から別の Satellite へのホストの移行

**--force** を指定してこのスクリプトを使用し、以前の Satellite から **katello-ca-consumer-\*** パッケージを削除し、新しい Satellite で **katello-ca-consumer-\*** パッケージをインストールします。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
```

```
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--force
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--force
```

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

ブートストラップスクリプトは、システムがレガシープラットフォームに登録済みであることの指標として、`/etc/syconfig/rhn/systemid` が存在し、RHN の接続が有効であることを検出します。次にこのスクリプトは、**`rhn-classic-migrate-to-rhsm`** を呼び出して RHN からシステムを移行します。このスクリプトでは監査上の理由で、システムのレガシープロファイルはデフォルトで削除されません。レガシープロファイルを削除するには、**`--legacy-purge`** を使用してから、**`--legacy-login`** を使用して適切なパーミッションのあるユーザーアカウントを指定し、プロファイルを削除します。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。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--legacy-purge \
--legacy-login rhn-user
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--legacy-purge \
--legacy-login rhn-user
```

### 3.5.2.3. Puppet を使用しない Satellite へのホストの登録

デフォルトでは、ブートストラップスクリプトを使用して、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 をインストールしない場合は **--skip-puppet** を使用します。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--skip-puppet
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--skip-puppet
```

#### 3.5.2.4. コンテンツ管理専用としてホストを Satellite 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。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \
--server satellite.example.com \
--organization="My_Organization" \
--activationkey="My_Activation_Key" \
--skip-foreman
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --server satellite.example.com \
--organization="My_Organization" \
--activationkey="My_Activation_Key" \
--skip-foreman
```

#### 3.5.2.5. ブートストラップスクリプトによるコンシューマー RPM のダウンロード方法の変更

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 **http://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm** からコンシューマー 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 を許可する場合があります。

ます。--download-method を使用して、ダウンロードメソッドを HTTP から HTTPS へ変更します。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \  
--login=admin \  
--server satellite.example.com \  
--location="My_Location" \  
--organization="My_Organization" \  
--hostgroup="My_Host_Group" \  
--activationkey="My_Activation_Key" \  
--download-method https
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \  
--server satellite.example.com \  
--location="My_Location" \  
--organization="My_Organization" \  
--hostgroup="My_Host_Group" \  
--activationkey="My_Activation_Key" \  
--download-method https
```

#### 3.5.2.6. ホストの IP アドレスの Satellite への指定

インターフェイスが複数あるホスト、または1つのインターフェイスに IP アドレスが複数あるホストでは、IP アドレスの自動検出設定を無効にして、特定の IP アドレスを Satellite に指定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。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \  
--login=admin \  
--server satellite.example.com \  
--location="My_Location" \  
--organization="My_Organization" \  
--hostgroup="My_Host_Group" \  
--activationkey="My_Activation_Key" \  
--ip 192.x.x.x
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \  
--server satellite.example.com \  
--location="My_Location" \  
--organization="My_Organization" \  
--hostgroup="My_Host_Group" \  
--activationkey="My_Activation_Key" \  
--ip 192.x.x.x
```

### 3.5.2.7. ホストでのリモート実行の有効化

**--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 鍵を追加します。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--rex \
--rex-user root
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--rex \
--rex-user root
```

### 3.5.2.8. 登録時のホストのドメイン作成

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、**--add-domain** を使用して追加します。

#### 手順

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py \
--login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--add-domain
```

- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
```

```
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--add-domain
```

### 3.5.2.9. ホストの代替 FQDN の指定

ホストのホスト名が FQDN でない場合や、RFC に準拠していない場合 (アンダースコアなどの文字が含まれている) には、ホスト名の検証の段階で、スクリプトが失敗します。Satellite で使用可能な FQDN を使用するようにホストを更新できない場合は、ブートストラップスクリプトを使用して別の FQDN を指定してください。

#### 手順

1. Hammer を使用して **create\_new\_host\_when\_facts\_are\_uploaded** と **create\_new\_host\_when\_report\_is\_uploaded** を false に設定します。

```
# 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
```

2. **--fqdn** を使用して、Satellite にレポートする FQDN を指定します。

- Red Hat Enterprise Linux 8 の場合は、以下のコマンドを入力します。

```
# /usr/libexec/platform-python bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--fqdn node100.example.com
```

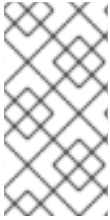
- Red Hat Enterprise Linux 6 または 7 では、次のコマンドを入力します。

```
# bootstrap.py --login=admin \
--server satellite.example.com \
--location="My_Location" \
--organization="My_Organization" \
--hostgroup="My_Host_Group" \
--activationkey="My_Activation_Key" \
--fqdn node100.example.com
```

## 3.6. KATELLO エージェントのインストール

Satellite クライアントをリモートで更新するには、Katello エージェントをインストールしてください。





## 注記

Katello エージェントは非推奨で、今後の Satellite のバージョンで削除されます。プロセスを移行し、リモート実行機能を使用してクライアントをリモートで更新してください。詳細は、[ホストの管理](#) の [Katello エージェントからリモート実行への移行](#) を参照してください。

**katello-agent** パッケージは、**goferd** service を提供する **gofer** パッケージによって異なります。

## 前提条件

- Satellite Server で、Satellite Client 6 リポジトリを有効にしておく。詳細は、[オンラインネットワークからの Satellite Server のインストール](#) の [Satellite Client 6 リポジトリの有効化](#) を参照してください。
- Satellite Server で、Satellite Client 6 リポジトリを同期しておく。詳細は、[オンラインネットワークからの Satellite Server のインストール](#) の [Satellite Client 6 リポジトリの同期](#) を参照してください。
- クライアントで Satellite Client 6 リポジトリを有効にしておく。

## 手順

1. **katello-agent** パッケージをインストールします。

```
# dnf install katello-agent
```

2. **goferd** サービスを開始します。

```
# systemctl start goferd
```

## 3.7. トレーサーのインストール

この手順を使用して、Red Hat Satellite にトレーサーをインストールし、トレースにアクセスします。トレーサーは、内容が古くなり、再起動が必要なサービスやアプリケーションの一覧を表示します。トレースは、Satellite Web UI でトレーサーが生成する出力です。

## 前提条件

- ホストが Red Hat Satellite に登録されている。
- Red Hat Satellite Client 6 リポジトリは、Satellite Server で有効化および同期されており、ストで有効化されている。

## 手順

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. 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。
4. **トレース** タブをクリックして、トレースを表示します。インストールされていない場合には、**Enable Traces** ボタンでそのパッケージをインストールするリモート実行ジョブを開始します。

### 3.8. PUPPET AGENT の手動でのインストールと設定

Puppet エージェントを手動でホストにインストールし、設定できます。Satellite と Puppet を統合するには、設定済みの Puppet エージェントがホストに必要です。Puppet の詳細は、[Red Hat Satellite での Puppet 統合を使用した設定の管理](#) を参照してください。

#### 前提条件

- Satellite で Puppet が有効になっている。詳細は、[Managing Configurations Using Puppet Integration in Red Hat Satellite](#) の [Enabling Puppet Integration with Satellite](#) を参照してください。
- ホストに Puppet 環境が割り当てられている。
- **Satellite Client 6** リポジトリを有効化して Satellite Server に同期し、ホスト上で有効化している必要がある。詳細は、[コンテンツの管理](#) の [コンテンツのインポート](#) を参照してください。

#### 手順

1. **root** ユーザーで、ホストにログインします。
2. Puppet エージェントパッケージをインストールします。
  - Red Hat Enterprise Linux 8 以降を実行しているホストの場合:
 

```
# dnf install puppet-agent
```
  - Red Hat Enterprise Linux 7 以前を実行しているホストの場合:
 

```
# yum install puppet-agent
```
3. 次のスクリプトを使用して、Puppet エージェントを現在のシェルの **PATH** に追加します。
 

```
./etc/profile.d/puppet-agent.sh
```
4. Puppet エージェントを設定します。ホストの所属先の Puppet 環境名に **environment** パラメーターを設定します。
 

```
# puppet config set server satellite.example.com --section agent
# puppet config set environment My_Puppet_Environment --section agent
```
5. Puppet エージェントサービスを開始します。
 

```
# puppet resource service puppet ensure=running enable=true
```
6. ホストの証明書を作成します。

```
# puppet ssl bootstrap
```

7. Satellite Web UI で、**Infrastructure > Capsules** に移動します。
8. 必要な Capsule Server の **Actions** コラムの一覧から、**Certificates** を選択します。
9. 必要なホストの右にある **Sign** をクリックして、Puppet エージェントの SSL 証明書に署名します。
10. ホスト上で、Puppet エージェントを再度実行します。

```
# puppet ssl bootstrap
```

### 3.9. KATELLO エージェントからリモート実行への移行

**Remote Execution** は、ホストでパッケージコンテンツを管理するのに推奨される方法です。Katello エージェントは非推奨で、今後の Satellite のバージョンで削除されます。以下の手順に従って、リモート実行に切り替えます。

#### 前提条件

- Satellite Server で、Satellite Client 6 リポジトリを有効にしておく。詳細は、[オンラインネットワークからの Satellite Server のインストール](#) の [Satellite Client 6 リポジトリの有効化](#) を参照してください。
- Satellite Server で、Satellite Client 6 リポジトリを同期しておく。詳細は、[オンラインネットワークからの Satellite Server のインストール](#) の [Satellite Client 6 リポジトリの同期](#) を参照してください。
- 以前は、コンテンツホストに **katello-agent** パッケージをインストールしている。

#### 手順

1. リモート実行が **ssh** モードを使用するように設定されている場合は、リモート実行 SSH 鍵をホストに配布します。詳細は、[「リモート実行のための SSH 鍵の配布」](#) を参照してください。
2. **pull-mqtt** モードを使用するようにリモート実行を設定している場合は、リモート実行プルクライアントをホストにデプロイします。詳細は、[「プル要求を使用するためのホストの設定」](#) を参照してください。
3. コンテンツホストの goferd サービスを停止します。

```
# systemctl stop goferd
```

4. コンテンツホストの goferd サービスを無効化します。

```
# systemctl disable goferd
```

5. コンテンツホスト上の Katello エージェントを削除します。

**警告**

お使いのホストが Red Hat Virtualization のバージョン 4.4 以下にインストールされている場合は、削除された依存関係によりホストが破損するので、**katello-agent** パッケージを削除しないでください。

```
# dnf remove katello-agent
```

6. Satellite Web UI で、**Administer** > **Settings** に移動します。
7. **コンテンツ** タブを選択します。
8. **Use remote execution by default** パラメーターを **Yes** に設定します。

Satellite Server は、Katello Agent の代わりにリモート実行によるホスト管理を使用するようになりました。

以下の表には、特定のパッケージ操作を実行するリモート実行と同等のコマンドをまとめています。ターゲットのホストまたはホストコレクションを判断するために、検索クエリーを指定する方法については、**hammer job-invocation create --help** を参照してください。

表3.2 Hammer コマンド

アクション	Katello Agent	リモート実行
パッケージのインストール	<b>hammer host package install</b>	<b>hammer job-invocation create --feature katello_package_install</b>
パッケージのインストール (ホストコレクション)	<b>hammer host-collection package install</b>	<b>hammer job-invocation create --feature katello_package_install</b>
パッケージの削除	<b>hammer host package remove</b>	<b>hammer job-invocation create --feature katello_package_remove</b>
パッケージ (ホストコレクション) の削除	<b>hammer host-collection package remove</b>	<b>hammer job-invocation create --feature katello_package_remove</b>
パッケージの更新	<b>hammer host package upgrade</b>	<b>hammer job-invocation create --feature katello_package_update</b>
パッケージ (ホストコレクション) の更新	<b>hammer host-collection package update</b>	<b>hammer job-invocation create --feature katello_package_update</b>

アクション	Katello Agent	リモート実行
すべてのパッケージの更新	<b>hammer host package update</b>	<b>hammer job-invocation create --feature katello_package_update</b>
エラータのインストール	<b>hammer host errata apply</b>	<b>hammer job-invocation create --feature katello_errata_install</b>
エラータのインストール (ホストコレクション)	<b>hammer host-collection errata install</b>	<b>hammer job-invocation create --feature katello_errata_install</b>
パッケージグループのインストール	<b>hammer host package-group install</b>	<b>hammer job-invocation create --feature katello_group_install</b>
パッケージグループのインストール (ホストコレクション)	<b>hammer host-collection package-group install</b>	<b>hammer job-invocation create --feature katello_group_install</b>
パッケージグループの削除	<b>hammer host package-group remove</b>	<b>hammer job-invocation create --feature katello_group_remove</b>
パッケージグループ (ホストコレクション) の削除	<b>hammer host-collection package-group remove</b>	<b>hammer job-invocation create --feature katello_group_remove</b>
パッケージグループを更新する	<b>hammer host package-group update</b>	<b>hammer job-invocation create --feature katello_group_update</b>
パッケージグループ (ホストコレクション) の更新	<b>hammer host-collection package-group update</b>	<b>hammer job-invocation create --feature katello_group_update</b>

## 第4章 ネットワークインターフェイスの追加

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イスを指定することをサポートします。「[Red Hat Satellite でのホストの作成](#)」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イスを設定することができます。

ホストに割り当てることのできるネットワークインターフェイスにはいくつかのタイプがあります。新規インターフェイ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。

- **インターフェイス**: 物理インターフェイスまたは仮想インターフェイスを追加で指定できます。作成できる仮想インターフェイスのタイプは 2 つあります。ホストが 1 つのインターフェイスを使用して複数の (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。既存のインターフェイスに別の IP アドレスを追加するには、**エイリアス** を使用します。  
物理インターフェイスの追加に関する情報は、「[物理インターフェイスの追加](#)」を参照してください。  
  
仮想インターフェイスの追加に関する情報は、「[仮想インターフェイスの追加](#)」を参照してください。
- **ボンド**: ボンディングインターフェイスを作成します。NIC ボンディングは、複数のネットワークインターフェイスを 1 つのインターフェイ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イスが 1 つのネットワークインターフェイスとして機能し、帯域幅の拡大と冗長性を提供します。詳細は、「[ボンディングインターフェイスの追加](#)」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および管理できます。BMC の詳細は、[オンラインネットワーク環境での Satellite Server のインストール](#) の [管理対象ホストでの電源管理の有効化](#) を参照してください。BMC インターフェイスの設定に関する詳細は、「[ベースボード管理コントローラー \(BMC\) インターフェイスの追加](#)」を参照してください。



### 注記

追加のインターフェイスには、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、新規インターフェイスが、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には、管理対象インターフェイ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イ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イスの設定はベアメタルホストでのみ機能します。

### 4.1. 物理インターフェイスの追加

この手順を使用して、別の物理インターフェイスをホストに追加します。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。

2. 編集するホストの横の **編集** をクリックします。
3. **インターフェイス タブ**で、**インターフェイスの追加** をクリックします。
4. **タイプ** リストで、**インターフェイス オプション**が選択されている状態にします。
5. **MAC アドレス** を指定します。この設定は必須です。
6. **eth0** などの **デバイス ID** を指定します。ボンディングインターフェイス、VLAN、エイリアスの作成時に、この ID を使用してこの物理インターフェイスを指定します。
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** リスト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** リスト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. **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イスでは、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対象インターフェイスでは、IP アドレスが自動補完されます。
11. インターフェイスが **管理対象** かどうかを選択します。インターフェイスが管理対象の場合は、プロビジョニング時に関連付けられた Capsule Server から設定がプルされ、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には、設定ファイルはインターフェイス用に自動的に作成されます。
12. ホストの **プライマリー** インターフェイスかどうかを選択します。プライマリーインターフェイスからの DNS 名を、FQDN のホストの部分として使用します。
13. ホストの **プロビジョニング** インターフェイスかどうかを選択します。TFTP ブートは、プロビジョニングインターフェイスを使用します。イメージベースのプロビジョニングの場合は、プロビジョニングを完了するスクリプトは、プロビジョニングインターフェイスを使用してプロビジョニングを完了します。
14. **リモート実行** のインターフェイスを使用するかどうかを選択します。
15. **仮想 NIC** チェックボックスのチェックを解除したままにします。
16. **OK** をクリックして、インターフェイス設定を保存します。
17. **送信** をクリックして、ホストへの変更を適用します。

## 4.2. 仮想インターフェイスの追加

以下の手順を使用して、ホストの仮想インターフェイスを設定します。仮想インターフェイスには、VLAN またはエイリアスインターフェイ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イスとは、既存のインターフェイスにアタッチされた追加の IP アドレスのことです。エイリアスインターフェイスは、自動的にアタッチ先のインターフェイスから MAC アドレスを継承するので、MAC アドレスを指定せずにエイリアスを作成できます。インターフェイスは、ブートモードを **static** に設定したサブネットに指定する必要があります。

## 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。
2. 編集するホストの横の **編集** をクリックします。
3. **インターフェイス** タブで、**インターフェイスの追加** をクリックします。
4. **タイプ** リストで、**インターフェイス** 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イス設定を指定します。適用できる設定オプションは、物理インターフェイスのオプションと同じです (「[物理インターフェイスの追加](#)」を参照)。  
管理対象の仮想インターフェイスの **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イスには不要です。  
  
VLAN を作成する場合、**デバイス ID** 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。仮想インターフェイ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**: オプションで VLAN タグを設定して、物理ネットワークから仮想インターフェイスにネットワークセグメントを分割します。タグを指定しない場合は、管理インターフェイスは、関連のあるサブネットの VLAN タグを継承します。このフィールドでユーザーが指定したエントリーは、エイリアスインターフェイスには適用されません。
  - **割り当て先**: **eth1** など、仮想インターフェイスの所属先となる物理インターフェイスの ID を指定します。この設定は必須です。
7. **OK** をクリックして、インターフェイス設定を保存します。
8. **送信** をクリックして、ホストへの変更を適用します。

## 4.3. ボンディングインターフェイスの追加

以下の手順を使用して、ホストのボンディングインターフェイスを設定します。Satellite Web UI の代わりに CLI を使用する場合は、[CLI 手順](#) を参照してください。

## 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。
2. 編集するホストの横の **編集** をクリックします。
3. **インターフェイス** タブで、**インターフェイスの追加** をクリックします。
4. **タイプ** リストから **ボンディング** を選択します。タイプ固有の設定オプションがフォームに追加されます。
5. 一般的なインターフェイス設定を指定します。適用できる設定オプションは、物理インターフェイスのオプションと同じです (「[物理インターフェイスの追加](#)」を参照)。  
ボンディングインターフェイスは、**デバイス ID** フィールドにある **bond0** 形式の ID を使用します。  
  
**MAC アドレス** 1つで十分です。



6. ボンディングインターフェイスに固有の設定オプションを指定します。

- **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、「[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 ID のコンマ区切りの一覧を指定します。物理インターフェイスまたは 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`)。ボンディングされたインターフェイスの設定オプションの詳細は、[Red Hat Enterprise Linux のネットワークの設定および管理 ガイド](#)の [ネットワークボンディングの設定](#) を参照してください。

7. **OK** をクリックして、インターフェイス設定を保存します。

8. **送信** をクリックして、ホストへの変更を適用します。

## CLI 手順

- ボンディングインターフェイスでホストを作成するには、以下のコマンドを入力します。

```
# hammer host create --name bonded_interface \
--hostgroup-id 1 \
--ip=192.168.100.123 \
--mac=52:54:00:14:92:2a \
--subnet-id=1 \
--managed true \
  --interface="identifier=eth1, \
    mac=52:54:00:62:43:06, \
    managed=true, \
    type=Nic::Managed, \
    domain_id=1, \
    subnet_id=1" \
  --interface="identifier=eth2, \
    mac=52:54:00:d3:87:8f, \
    managed=true, \
    type=Nic::Managed, \
    domain_id=1, \
    subnet_id=1" \
  --interface="identifier=bond0, \
    ip=172.25.18.123, \
    type=Nic::Bond, \
    mode=active-backup, \
    attached_devices=[eth1,eth2], \
    managed=true, \
    domain_id=1, \
    subnet_id=1" \
--organization "My_Organization" \
--location "My_Location" \
--ask-root-password yes
```

## 4.4. 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イスで順次行われます。
active-backup	ボンディングインターフェイ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イ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イスが使用されます。
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イスで送信されます。
broadcast	すべての送信はすべてのボンディングインターフェイスで行われます。
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イ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イ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。

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

以下の手順を使用して、ベースボード管理コントローラー (BMC) インターフェイスを、この機能をサポートするホストに設定します。

### 前提条件

- **ipmitool** パッケージがインストールされている。
- ホストの MAC アドレス、IP アドレス、BMC インターフェイスのその他の詳細、およびこのインターフェイスの適切な認証情報を確認している。



### 注記

BMC インターフェイスが管理対象の場合は、BMC インターフェイスの MAC アドレスのみが必要になります。これは DHCP 予約を作成するために必要です。

### 手順

1. Capsule Server で BMC が有効になっていない場合には、有効にします。

- a. 以下のオプションを指定して **satellite-installer** スクリプトを実行し、Capsule Server で BMC 電源管理を設定します。

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

- b. Satellite Web UI で、**Infrastructure > Capsules** に移動します。
  - c. **アクション** コラムの一覧から、**更新** をクリックします。**機能** コラムの一覧に BMC が追加されているはずです。
2. Satellite Web UI で、**ホスト > すべてのホスト** に移動します。
  3. 編集するホストの横の **編集** をクリックします。
  4. **インターフェイス** タブで、**インターフェイスの追加** をクリックします。
  5. **タイプ** リストから **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。
  6. 一般的なインターフェイス設定を指定します。適用できる設定オプションは、物理インターフェイスのオプションと同じです (「[物理インターフェイスの追加](#)」を参照)。
  7. BMC インターフェイスに固有の設定オプションを指定する方法:
    - **ユーザー名** および **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。
  8. **OK** をクリックして、インターフェイス設定を保存します。
  9. **送信** をクリックして、ホストへの変更を適用します。

## 第5章 ホストを次の主要な RED HAT ENTERPRISE LINUX リリースにアップグレード

ジョブテンプレートを使用して、Red Hat Enterprise Linux ホストを次のメジャーリリースにアップグレードできます。以下のアップグレードパスが可能です。

- Red Hat Enterprise Linux 7 から Red Hat Enterprise Linux 8 へ
- Red Hat Enterprise Linux 8 から Red Hat Enterprise Linux 9 へ

### 前提条件

- Red Hat Enterprise Linux ホストがアップグレードの要件を満たしていることを確認してください。
  - Red Hat Enterprise Linux 7 から Red Hat Enterprise Linux 8 へのアップグレードについては、**RHEL 7 から RHEL 8 へのアップグレードの [アップグレードの計画](#)** を参照してください。
  - Red Hat Enterprise Linux 8 から Red Hat Enterprise Linux 9 へのアップグレードは、**RHEL 8 から RHEL 9 へのアップグレードの [アップグレードの計画](#)** を参照してください。
- アップグレードのために Red Hat Enterprise Linux ホストを準備します。
  - Red Hat Enterprise Linux 7 から Red Hat Enterprise Linux 8 へのアップグレードは、**RHEL 7 から RHEL 8 へのアップグレードの [アップグレードのための RHEL 7 システムの準備](#)** を参照してください。
  - Red Hat Enterprise Linux 8 から Red Hat Enterprise Linux 9 へのアップグレードについては、**RHEL 8 から RHEL 9 へのアップグレードの [アップグレードに向けた RHEL 8 システムの準備](#)** を参照してください。
- Satellite でリモート実行機能を有効にする。詳細は、**11章 [リモートジョブの設定およびセットアップ](#)** を参照してください。
- アップグレードするホストに Satellite の SSH 鍵を配布します。詳細は、**「[リモート実行のための SSH 鍵の配布](#)」** を参照してください。

### 手順

1. Satellite で、Leapp プラグインを有効にします。

```
# satellite-installer --enable-foreman-plugin-leapp
```

2. Satellite Web UI で、**Hosts > All Hosts** に移動します。
3. 次の主要な Red Hat Enterprise Linux バージョンにアップグレードするホストを選択します。
4. Hosts ウィンドウの右上の **Select Action** リストから **Preupgrade check with Leapp** を選択します。
5. **Submit** をクリックして、アップグレード前のチェックを開始します。
6. チェックが完了したら **Leapp preupgrade report** タブをクリックして、Leapp がお使いのホストで問題を検出したかどうかを確認します。**Inhibitor** フラグの付いた問題は、重大であるとみなされ、アップグレードの手順に失敗する可能性が高くなります。**Has Remediation** フラグが

付けられた問題には、問題の修正に役立つ修正が含まれています。

a. **Has Remediation** フラグが付けられた問題をクリックして、展開します。

- 問題に修復 コマンド が含まれている場合は、リモート実行を使用して、Satellite から直接修正できます。問題を選択します。
- 問題に修復 ヒント のみが含まれている場合は、ヒントを使用して、ホストの問題を手動で修正します。

他の問題について、この手順を繰り返します。

b. 修復コマンドの問題を選択したら、**Fix Selected** をクリックして、ジョブを送信します。

c. 問題が修正されたら、**Rerun** ボタンをクリックし、**Submit** をクリックしてアップグレード前のチェックを再度実行し、アップグレードするホストに問題がなく、アップグレードの準備ができていることを確認します。

7. アップグレード前のチェックでホストに問題がないことが確認された場合は、**Run Upgrade** ボタンをクリックし、**Submit** をクリックしてアップグレードを開始します。

## 第6章 ホストの RED HAT ENTERPRISE LINUX への変換

Red Hat Enterprise Linux の派生ディストリビューションは、インストールされたアプリケーションや設定を保持しながら、ホスト上でサポート可能な Red Hat Enterprise Linux に変換できます。Satellite には **Convert2RHEL** ユーティリティがあり、変換プロセスを簡素化します。

Satellite の **Convert2RHEL** ユーティリティは、Ansible ロールと Ansible Playbook で設定されます。Ansible ロールを使用して Satellite Server で変換データを生成し、これには、必要なリポジトリの有効化、製品、アクティベーションキー、およびホストグループの作成が含まれます。次に、Ansible Playbook を使用してホストで実際に変換を行い、Convert2RHEL CLI ツールをホストにインストールして実行します。

Ansible ロールを使用して、以下の変換の変換データを生成できます。

- CentOS Linux 7 から Red Hat Enterprise Linux 7
- Oracle Linux 7 から Red Hat Enterprise Linux 7
- CentOS Linux 8 から Red Hat Enterprise Linux 8
- Oracle Linux 8 から Red Hat Enterprise Linux 8

これらの変換は、Red Hat によってサポートされています。

この変換プロセスは、システム上のすべての RPM パッケージが置き換えられる Red Hat Enterprise Linux のマイナーリリースアップグレードと似ています。サードパーティーパッケージと、Red Hat Enterprise Linux で利用できない Red Hat 以外のパッケージは保持されます。

Convert2RHEL ユーティリティは、変換中に問題を引き起こすことがわかっているロゴやパッケージなどの不要なパッケージを削除します。このユーティリティは、**CentOS-release** または **Oracle-release** パッケージを **rhel-release** パッケージに、CentOS または Oracle が署名したすべてのパッケージを Red Hat と同等のパッケージに置き換えます。また、このユーティリティは、ホストを Red Hat サブスクリプション管理にサブスクライブします。

変換プロセスの期間は、置き換えが必要なパッケージ数、ネットワークの速度、ストレージの速度、および同様の要因によって異なります。

### 前提条件

- RPM ベースの Linux ディストリビューションから RHEL への変換で、[サポートされている変換パス](#)を確認してください。
- RPM ベースの Linux ディストリビューションから RHEL への変換の [RHEL 変換の準備](#)の手順 1~5. を完了している必要があります。
- サブスクリプションマニフェストを Satellite にアップロードし、対象とする変換に十分な Red Hat Enterprise Linux エンタイトルメントが割り当てられていることを確認します。または、Ansible 変数を使用して、ディスクからマニフェストをインポートするようにロールに指示することもできます。マニフェストは、変換用にホストの登録先の組織にインポートする必要があります。  
割り当てを更新して、[Red Hat カスタマーポータル](#) から更新されたマニフェストをダウンロードできます。詳細は、Red Hat Subscription Management の [マニフェストの使用](#)を参照してください。
- ホストを変換する Red Hat Enterprise Linux のマイナーバージョン用に、Satellite で Red Hat リポジトリを有効にしてください。

## 変換手順の概要

1. **redhat.satellite.convert2rhel** Ansible ロールおよび変数をインポートします。詳細は、[Red Hat Satellite での Ansible 統合を使用した設定の管理](#) の [Ansible ロールと変数のインポート](#) を参照してください。
2. 変換データを生成するための Ansible 変数を設定します。詳細は、[「変換データを生成するための変数」](#) を参照してください。
3. **redhat.satellite.convert2rhel** ロールを Satellite Server を表すホストに割り当てます。詳細は、[Red Hat Satellite での Ansible Integration を使用した設定の管理](#) の [既存のホストへの Ansible ロールの割り当て](#) を参照してください。
4. Satellite Server で Ansible ロールを実行します。詳細は、[Red Hat Satellite での Ansible 統合を使用した設定の管理](#) の [Ansible ロールと変数の実行](#) を参照してください。  
Ansible ロールは、ホスト変換に必要なデータ、つまりリポジトリ、証明書、アクティベーションキー、およびホストグループを生成します。以前の手順で設定した変数に合わせて、このロールは、リリースが 7Server で、アーキテクチャーが x86\_64 の **rhel-7-server-rpms** リポジトリか、**rhel-8-for-x86\_64-baseos-rpms** および **rhel-8-for-x86\_64-appstream-rpms**、または両方を有効にします。
5. 生成されたホストグループを使用して、変換用にホストを登録します。  
グローバル登録テンプレートを使用して、変換前にホストを登録してサブスクライブします。ホストを CentOS 8 に変換する場合は、**CentOS 8 変換** などの変換用に生成されたホストグループを選択します。詳細は、[「ホストの登録」](#) を参照してください。
6. ホストで Convert2RHEL Playbook を実行します。次の設定でリモートジョブを実行します。
  - ジョブカテゴリ: **Convert 2 RHEL**
  - ジョブテンプレート: **Convert to RHEL**
  - アクティベーションキー: **convert2rhel\_rhel7** または **convert2rhel\_rhel8**
 詳細は、[「リモートジョブの実行」](#) を参照してください。

## 関連情報

- [How to perform an unsupported conversion from a RHEL-derived Linux distribution to RHEL](#)

## 6.1. 変換データを生成するための変数

Ansible ロールを実行して変換データを生成する前に、以下の必要な Ansible 変数の値を設定します。

Satellite は、必要なほとんどの Ansible 変数を **redhat.satellite.convert2rhel** ロールからインポートします。ただし、一部の変数はインポートされません。以下の表では、これらの変数にはアスタリスク \* が付いています。これらの追加変数を手動で作成し、**redhat.satellite.convert2rhel** ロールに割り当てる必要があります。

表6.1 変換に必要な変数

名前	タイプ	目的および値
<b>satellite_server_url</b> *	string	<b>https://satellite.example.com</b> などの Satellite Server の URL

名前	タイプ	目的および値
<b>satellite_username *</b>	string	ユーザー名
<b>satellite_password *</b>	string	パスワード
<b>satellite_organization *</b>	string	組織の名前
<b>satellite_content_rhel_wait_for_syncs *</b>	boolean	Satellite Server がリポジトリの同期が完了するまで待機せずに、データの生成を続行する場合は、 <b>false</b> に設定します。(デフォルト: <b>true</b> )
<b>satellite_validate_certs *</b>	boolean	Ansible で証明書チェックを有効にする場合は <b>true</b> に設定します (デフォルト: <b>true</b> )。
<b>satellite_convert2rhel_manage_subscription</b>	boolean	Satellite Server にマニフェストがすでに存在する場合は、 <b>false</b> に設定します。ディスクから新しいマニフェストをアップロードすると、現在のマニフェストが上書きされます (デフォルト: <b>true</b> )。
<b>satellite_content_rhel_enable_rhel7 *</b>	boolean	Red Hat Enterprise Linux 7 リポジトリを有効にします。ホストを Red Hat Enterprise Linux 7 に変換する予定がない場合は、 <b>false</b> に設定します (デフォルト: <b>true</b> )。
<b>satellite_convert2rhel_enable_oracle7</b>	boolean	Oracle Linux 7 の変換データを準備する場合は <b>true</b> に設定します。それ以外の場合は、値を <b>false</b> に設定する必要があります。
<b>satellite_content_rhel_enable_rhel8 *</b>	boolean	Red Hat Enterprise Linux 8 リポジトリを有効にします。ホストを Red Hat Enterprise Linux 8 に変換する予定がない場合は、 <b>false</b> に設定します (デフォルト: <b>true</b> )。
<b>satellite_convert2rhel_enable_oracle8</b>	boolean	Oracle Linux 8 の変換データを準備する場合は <b>true</b> に設定します。それ以外の場合は、値を <b>false</b> に設定する必要があります。

表6.2 変換用の任意の変数

名前	タイプ	目的および値
----	-----	--------



名前	タイプ	目的および値
<b>satellite_manifest_path</b> *	string	ディスクからアップロードするマニフェストへのパス (例: <b>~/manifest.zip</b> )。 <b>satellite_convert2rhel_manage_subscription</b> を使用してディスクから新しいマニフェストをアップロードする場合は、このパスを設定する必要があります。
<b>satellite_content_rhel_rhel8_releasever</b> *	string	<b>8.5</b> などのマイナーリリースバージョン。変換の問題を防ぐために、システムのマイナーリリースバージョンが最新の Red Hat Enterprise Linux リリースと異なる場合は、この変数を設定します (デフォルト: latest)。

## 第7章 RHEL WEB コンソールを使用したホストの管理と監視

RHEL Web コンソールは、アクションの実行および Red Hat Enterprise Linux ホストの監視に使用できるインタラクティブな Web インターフェイスです。remote-execution 機能を有効化し、Satellite を RHEL Web コンソールに統合できます。Satellite で管理するホストに RHEL Web コンソールをインストールすると、そのホストの RHEL Web コンソールダッシュボードを Satellite Web UI 内から表示できます。RHEL Web コンソールに統合されている機能 (Red Hat Image Builder など) を使用することもできます。

### 7.1. SATELLITE での RHEL WEB コンソールの有効化

デフォルトでは、Satellite では RHEL Web コンソール統合が無効になっています。Satellite 内からホストの RHEL Web コンソール機能にアクセスする場合は、最初に Satellite Server で RHEL Web コンソール統合を有効にする必要があります。

#### 手順

- Satellite Server で **--enable-foreman-plugin-remote-execution-cockpit** オプションを指定して **satellite-installer** を実行します。

```
# satellite-installer --enable-foreman-plugin-remote-execution-cockpit
```

### 7.2. RHEL WEB コンソールを使用したホストの管理と監視

Satellite Web UI を介して RHEL Web コンソール Web UI にアクセスし、その機能を使用して Satellite のホストを管理および監視できます。

#### 前提条件

- RHEL Web コンソールが Satellite で有効になっている。
- Red Hat Web コンソールが、表示するホストにインストールされている。
  - Red Hat Enterprise Linux 8 の場合は、**RHEL 8 で Web コンソールを使用したシステムの管理** の [Web コンソールのインストール](#) を参照してください。
  - Red Hat Enterprise Linux 7 の場合は、**RHEL 7 で Web コンソールを使用したシステムの管理** の [Web コンソールのインストール](#) を参照してください。
- Satellite または Capsule は、SSH 鍵を使用してホストを認証できる。詳細は、「[リモート実行のための SSH 鍵の配布](#)」を参照してください。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動し、RHEL Web コンソールを使用して管理および監視するホストを選択します。
2. ホストウィンドウの右上で **Web コンソール** をクリックします。

これで、RHEL Web コンソールを介して、ホストの監視と管理に使用できるすべての機能 (Red Hat Image Builder など) にアクセスできるようになりました。

Red Hat Web コンソールを使い始めるための詳細は、[RHEL 8 で Web コンソールを使用したシステムの管理](#) ガイド、または [RHEL 7 で Web コンソールを使用したシステムの管理](#) ガイドを参照してください。

RHEL Web コンソールでの Red Hat Image Builder の使用に関する詳細は、[RHEL 8 Web コンソールで Image Builder GUI へのアクセス](#) または [RHEL 7 Web コンソールで Image Builder GUI へのアクセス](#) を参照してください。

### 7.3. SATELLITE での RHEL WEB コンソールの無効化

Satellite で Red Hat Web コンソールを無効にする場合は、次の手順を実行します。

#### 手順

1. 次の **satellite-installer** コマンドを実行します。

```
satellite-installer --no-enable-foreman-plugin-remote-execution-cockpit
```

2. Satellite Web UI で、**Administer** > **Settings** に移動し、**Remote execution** タブをクリックします。
3. **Cockpit URL** 行で、**Value** の下の設定を消去し、**Submit** をクリックします。これにより、Satellite Web UI から **Web Console** ボタンが削除されます。
4. RHEL Web コンソールパッケージを Satellite からアンインストールします。

```
dnf remove rubygem-foreman_remote_execution-cockpit
```

## 第8章 RED HAT INSIGHTS を使用したホストの監視

本章では、ホスト監視レポートの作成、Red Hat Insights を使用したホストの監視、および Insights プランの作成について説明しています。

### 8.1. SATELLITE のホストでの RED HAT INSIGHTS の使用

Red Hat Insights を使用すると、セキュリティ違反、パフォーマンスの低下、および安定性の消失に関連するシステムとダウンタイムを診断できます。ダッシュボードを使用して、安定性、セキュリティ、およびパフォーマンスの主要なリスクを素早く特定できます。また、カテゴリー別に分類したり、影響度および解決方法の詳細を表示したり、影響を受けたシステムを調べたりすることができます。

Red Hat Insights を使用して Satellite で管理するホストを監視するには、Red Hat Insights をホストにインストールしてから、ホストを Red Hat Insights に登録する必要があります。

新しい Satellite ホストの場合、グローバル登録テンプレートを使用して、登録中に Insights で Satellite ホストをインストールおよび設定することができます。詳細は、[ホストの管理](#) の [グローバル登録テンプレートを使用した Red Hat Satellite へのホストの登録](#) を参照してください。

Puppet を使用して、または手動でホストをインストールおよび登録する方法については、[Red Hat Insights を使い始める](#) を参照してください。

#### ホストで利用可能な Red Hat Insights 情報

Red Hat Insights でホストに関する追加情報を利用できます。

この情報は、以下の 2 つの場所にあります。

- Satellite Web UI で、**Configure > Insights**に移動します。その **Remediate** ボタンの横にある縦リーダーに **Red Hat Insights のビュー**リンクがあり、一般的な推奨事項ページに表示されます。対象の推奨事項に対して **ナレッジベースの記事** がある場合には、各推奨事項の行にある縦リーダーから、**Red Hat Insights での表示**リンクがあり、推奨ルールに移動できます。
- 詳細は、**ホスト > すべてのホスト**に移動します。ホストに推奨事項が記載されている場合は、推奨事項の数字をクリックします。Insights タブの **Remediate** ボタンの横にある縦リーダーで、対象システムの情報への **Satellite Insights ページへの移動**リンクと、コンソール上のホスト情報への **Red Hat Insights での表示**リンクが提供されます。

#### rh-cloud および insights-client レポートからのホストの除外

**host\_registration\_insights** パラメーターを **False** に設定すると **rh-cloud** および **insights-client** レポートを省略できます。Satellite は、**rh-cloud** レポートからホストを除外し、**insights-client** がレポートをクラウドにアップロードしないようにします。

このパラメーターは、組織、ホストグループ、サブネット、およびドメインレベルで設定することもできます。新しいレポートがエンティティに関連付けられている限り、自動的にアップロードされないようにします。

[Red Hat Hybrid Cloud](#) ですでに報告されているホストでパラメーターを **false** に設定すると、インベントリから自動的に削除されます。ただし、このプロセスが完了するには時間がかかる場合があります。

#### Ansible ロールを使用した Red Hat Insights のデプロイ

**RedHatInsights.insights-client** Ansible ロールを使用して、Red Hat Insights でのホストのインストールと登録を自動化できます。このロールを Satellite に追加する方法の詳細は、[Red Hat Satellite での Ansible Integration を使用した設定の管理](#) の [Satellite での Ansible の使用開始](#) を参照してください。

1. **RedHatInsights.insights-client** ロールをホストに追加します。  
新しいホストについては、「[Red Hat Satellite でのホストの作成](#)」を参照してください。  
  
既存のホストについては、[Red Hat Satellite での Ansible Integration を使用した設定の管理の Ansible ロールを使用してクライアントの反復タスクを自動化する](#) を参照してください。
2. **RedHatInsights.insights-client** ロールをホストで実行するには、**ホスト > すべてのホスト** に移動して、使用するホスト名をクリックします。
3. **Ansible ロールの実行** ボタンをクリックします。

続行する前に、Insights の API トークンを設定する必要があります。詳細については、[Red Hat API Tokens](#) を参照してください。

次の手順を使用して、推奨事項を手動で同期できます。

1. Satellite Web UI で、**設定 > Insights** に移動します。
2. **Start Recommendations Sync** ボタンをクリックします。

API トークンを設定していない場合は、このページを使用する前に API トークンを作成するように求められます。

### 追加情報

- Red Hat Insights およびすべてのプラグインのログを確認するには、`/var/log/foreman/production.log` に移動します。
- Red Hat Insights との接続に問題がある場合は、証明書が最新のものであることを確認してください。サブスクリプションマニフェストをリフレッシュして証明書を更新します。
- ホスト上に **insights-client.timer** を設定することで、デフォルトの **insights-client** 実行スケジュールを変更することができます。詳細は、[Red Hat Insights のクライアント設定ガイドの insights-client スケジュールの変更](#) を参照してください。

## 8.2. ホストの INSIGHTS プランの作成

Satellite で、Red Hat Insights 修復プランを作成し、Satellite ホストでこのプランを実行します。

### 手順

1. Satellite Web UI で、**設定 > Insights** に移動します。
2. Red Hat Insights ページで、Insights プランに含める推奨事項の数を選択します。  
関連付けられた Playbook がある推奨事項のみを選択できます。
3. **Remediate** をクリックします。
4. **Remediation Summary** ウィンドウで、適用する **解決策** を選択します。**Filter** フィールドを使用して、特定のキーワードを検索します。
5. **Remediate** をクリックします。

6. **Job Invocation** ページで、事前完了フィールドの内容は変更しないでください。
7. オプション。リモート実行ジョブの詳細設定については、**Show Advanced Fields** をクリックします。
8. 必要な**Type of query**を選択します。
9. 必要な**Schedule**を選択します。
10. **Submit** をクリックします。

または、次のようになります。

1. Satellite Web UI で、**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. ホストの詳細ページで、**Recommendations** をクリックします。
4. Red Hat Insights ページで、Insights プランに含める推奨事項の数を選択し、同様に続行します。

ジョブウィンドウでは、プランの進捗を表示できます。

## 第9章 レポートテンプレートを使用したホストの監視

レポートテンプレートを使用して Satellite データをクエリーし、ホストのステータス、登録済みのホスト、適用可能なエラータ、適用済みのエラータ、サブスクリプションの詳細、ユーザーアクティビティなどの情報を取得できます。Satellite に同梱されるレポートテンプレートを使用するか、または要件に合わせて独自のカスタムレポートテンプレートを作成することができます。レポートエンジンは、Embedded Ruby (ERB) 構文を使用します。テンプレートの作成と ERB 構文の詳細は、[付録A テンプレート作成の参照](#)を参照してください。

テンプレートを作成するか、テンプレートのクローンを作成して、クローンを編集します。テンプレートの構文に関するヘルプは、テンプレートをクリックして、**ヘルプ** タブをクリックします。

### 9.1. ホスト監視レポートの生成

Satellite Web UI でレポートテンプレートを表示するには、**監視 > レポートテンプレート** に移動します。レポートをスケジュールするには、cron ジョブを設定するか、Satellite Web UI を使用します。

#### 手順

1. Satellite Web UI で、**監視 > レポートテンプレート** に移動します。
2. 使用するレポートテンプレートの右にある、**生成** をクリックします。
3. オプション: レポートをスケジュールするには、the **生成日時** フィールドの右側のアイコンをクリックして、レポートを生成する日時を選択します。
4. オプション: メールアドレスにレポートを送信するには、**メールでレポートを送信する** チェックボックスを選択して、**配信先のメールアドレス** フィールドで、必要なメールアドレスを入力します。
5. オプション: 検索クエリーフィルターを適用します。利用可能な結果すべてを表示するには、フィルターフィールドに何も値を投入しないでください。
6. **Submit** をクリックします。レポートが含まれる CSV ファイルをダウンロードします。**メールでレポートを送信する** チェックボックスを選択した場合は、ホストの監視レポートがメールアドレスに送信されます。

#### CLI 手順

1. 利用可能なレポートテンプレートすべてをリストします。

```
# hammer report-template list
```

2. レポートを生成します。

```
# hammer report-template generate --id My_Template_ID
```

このコマンドは、レポートが完全に生成されるまで待機してから完了します。レポートをバックグラウンドタスクとして生成する場合は、**hammer report-template schedule** コマンドを使用できます。



## 注記

サブスクリプションエンタイトルメントレポートを生成する場合は、**Days from Now** オプションを使用して、エンタイトルメントサブスクリプションの最新の有効期限を指定する必要があります。**no limit** 値を使用すると、すべてのエンタイトルメントを表示できます。

### すべてのエンタイトルメントを表示する

```
# hammer report-template generate \
--inputs "Days from Now=no limit" \
--name "Subscription - Entitlement Report"
```

### 60 日以内に有効期限が切れるすべてのエンタイトルメントを表示する

```
# hammer report-template generate \
--inputs "Days from Now=60" \
--name "Subscription - Entitlement Report"
```

## 9.2. レポートテンプレートの作成

Satellite では、レポートテンプレートを作成し、要件に合わせてテンプレートをカスタマイズできます。既存のレポートテンプレートをインポートして、スニペットとテンプレートマクロでさらにカスタマイズできます。

レポートテンプレートは Embedded Ruby (ERB) 構文を使用します。ERB 構文とマクロの使用に関する情報を表示するには、Satellite Web UI で、**監視 > レポートテンプレート** に移動し、**テンプレートの作成** をクリックしてから **ヘルプ** をクリックします。

Satellite でレポートテンプレートを作成すると、セーフモードがデフォルトで有効化されます。セーフモードの詳細は、**「レポートテンプレートのセーフモード」** を参照してください。

テンプレートの作成に関する詳細は、[\]](#) **を参照してください**。レポートテンプレートで利用できるマクロの詳細は、[xref:Template\\_Macros\\_managing-hosts](#) **[** **を参照してください**。

テンプレートへの入力 of 段階的な例を表示するには、**「エンタイトルメントを監視するレポートテンプレートの作成」** を参照してください。

### 手順

1. Satellite Web UI で、**監視 > レポートテンプレート** に移動して、**テンプレートの作成** をクリックします。
2. **名前** フィールドに、レポートテンプレートの一意名を入力します。
3. **デフォルト** を選択して、テンプレートをすべてのロケーションおよび組織で利用できるようにします。
4. テンプレートエディターで直接テンプレートを作成するか、**インポート** をクリックしてテキストファイルからテンプレートをインポートします。テンプレートのインポートに関する詳細は、**「レポートテンプレートのインポート」** を参照してください。
5. オプション: **監査コメント** フィールドで、このテンプレートに関する有用な情報を追加できます。



6. **入力** タブをクリックし、**名前** フィールドに、テンプレートで参照できる入力の名前を **input('name')** 形式で入力します。テンプレート本文でこの入力値を参照する前に、テンプレートを保存する必要がある点にご留意ください。
7. 入力値が必須かどうかを選択します。入力値が必須の場合は、**必須** チェックボックスをクリックします。
8. **値のタイプ** リストから、ユーザーが入力しなければならない入力値のタイプを選択します。
9. オプション: テンプレートの入力にファクトを使用する場合は、**詳細** チェックボックスをクリックします。
10. オプション: **オプション** フィールドで、ユーザーが選択できるオプションを定義します。このフィールドが未定義のままである場合、ユーザーは必要な値を入力できるフリーテキストフィールドを受け取ります。
11. オプション: **デフォルト** フィールドに、デフォルトのテンプレート入力として設定する値 (ホスト名など) を入力します。
12. オプション: **説明** フィールドに、レポートの生成時に入力に関するインラインヘルプとして表示する情報を入力できます。
13. オプション: **タイプ** タブをクリックして、このテンプレートが他のテンプレートに追加されるスニペットかどうかを選択します。
14. **ロケーション** タブをクリックして、テンプレートを使用するロケーションを追加します。
15. **組織** タブをクリックして、テンプレートを使用する組織を追加します。
16. **送信** をクリックして変更を保存します。

### 9.3. レポートテンプレートのエクスポート

Satellite で作成するレポートテンプレートをエクスポートできます。

#### 手順

1. Satellite Web UI で、**監視** > **レポートテンプレート** に移動します。
2. エクスポートするテンプレートを特定し、**アクション** コラムの一覧から **エクスポート** を選択します。
3. ダウンロードするすべてのレポートテンプレートに対して、この操作を繰り返します。

テンプレートのダウンロードを含む **.erb** ファイルです。

#### CLI 手順

1. エクスポートで利用可能なレポートテンプレートを表示するには、以下のコマンドを入力します。

```
# hammer report-template list
```

このコマンドの出力で、エクスポートするテンプレートのテンプレート ID をメモします。

2. レポートテンプレートをエクスポートするには、以下のコマンドを実行します。

```
# hammer report-template dump --id My_Template_ID > example_export.erb
```

## 9.4. SATELLITE API を使用したレポートテンプレートのエクスポート

Satellite **report\_templates** API を使用して、Satellite からレポートテンプレートをエクスポートできます。Satellite API の使用に関する詳細は、[API ガイド](#) を参照してください。

### 手順

1. 以下のリクエストを使用して、使用可能なレポートテンプレートの一覧を取得します。

#### 要求例:

```
$ curl --insecure --user admin:redhat \
--request GET \
--config https://satellite.example.com/api/report_templates \
| json_reformat
```

この例では、**json\_reformat** ツールを使用して JSON 出力をフォーマットしています。

#### 応答例:

```
{
  "total": 6,
  "subtotal": 6,
  "page": 1,
  "per_page": 20,
  "search": null,
  "sort": {
    "by": null,
    "order": null
  },
  "results": [
    {
      "created_at": "2019-11-20 17:49:52 UTC",
      "updated_at": "2019-11-20 17:49:52 UTC",
      "name": "Applicable errata",
      "id": 112
    },
    {
      "created_at": "2019-11-20 17:49:52 UTC",
      "updated_at": "2019-11-20 17:49:52 UTC",
      "name": "Applied Errata",
      "id": 113
    },
    {
      "created_at": "2019-11-30 16:15:24 UTC",
      "updated_at": "2019-11-30 16:15:24 UTC",
      "name": "Hosts - complete list",
      "id": 158
    },
    {
      "created_at": "2019-11-20 17:49:52 UTC",
      "updated_at": "2019-11-20 17:49:52 UTC",

```

```

    "name": "Host statuses",
    "id": 114
  },
  {
    "created_at": "2019-11-20 17:49:52 UTC",
    "updated_at": "2019-11-20 17:49:52 UTC",
    "name": "Registered hosts",
    "id": 115
  },
  {
    "created_at": "2019-11-20 17:49:52 UTC",
    "updated_at": "2019-11-20 17:49:52 UTC",
    "name": "Subscriptions",
    "id": 116
  }
]
}

```

2. エクスポートするテンプレートの **id** をメモし、以下のリクエストを使用してテンプレートをエクスポートします。

#### 要求例:

```

$ curl --insecure --output /tmp/_Example_Export_Template.erb \
--user admin:password --request GET --config \
https://satellite.example.com/api/report_templates/My_Template_ID/export

```

**158** は、エクスポートするテンプレートの ID の例である点にご留意ください。

この例では、エクスポートされたテンプレートは、**host\_complete\_list.erb** にリダイレクトされます。

## 9.5. レポートテンプレートのインポート

作成する新しいテンプレートの本文にレポートテンプレートをインポートできます。Satellite Web UI を使用すると、テンプレートのインポートは個別でしかできない点にご留意ください。一括操作には、Satellite API を使用します。詳細は、「[Satellite API を使用したレポートテンプレートのインポート](#)」を参照してください。

#### 前提条件

- 新しいテンプレートで使用するためにテンプレートをインポートするには、Satellite からテンプレートをエクスポートしておく必要がある。詳細は、「[レポートテンプレートのエクスポート](#)」を参照してください。

#### 手順

1. Satellite Web UI で、**監視 > レポートテンプレート** に移動します。
2. Report Templates ウィンドウの右上にある **Create Template** をクリックします。
3. **エディター** タブの右上にあるフォルダーアイコンをクリックし、インポートする **.erb** ファイルを選択します。
4. 要件に合わせてテンプレートを編集します。

## 5. Submit をクリックします。

新規テンプレートのカスタマイズに関する詳細は、[付録A テンプレート作成の参照](#)を参照してください。

## 9.6. SATELLITE API を使用したレポートテンプレートのインポート

Satellite API を使用して、レポートテンプレートを Satellite にインポートできます。Satellite API を使用してレポートテンプレートをインポートすると、レポートテンプレートのメタデータが自動的に解析され、組織とロケーションが割り当てられます。Satellite API の使用に関する詳細は、[API ガイド](#)を参照してください。

### 前提条件

- **.erb** 構文を使用してテンプレートを作成するか、別の Satellite からテンプレートをエクスポートしておく。

テンプレートの作成に関する詳細は、[付録A テンプレート作成の参照](#)を参照してください。

Satellite からのテンプレートのエクスポートに関する詳細は、「[Satellite API を使用したレポートテンプレートのエクスポート](#)」を参照してください。

### 手順

1. 以下の例を使用して、**.json** ファイルにインポートするテンプレートをフォーマットします。

```
# cat Example_Template.json
{
  "name": "Example Template Name",
  "template": "Enter ERB Code Here"
}
```

### ERB テンプレートを含む JSON ファイルの例:

```
{
  "name": "Hosts - complete list",
  "template": "<%#
name: Hosts - complete list
snippet: false
template_inputs:
- name: host
  required: false
  input_type: user
  advanced: false
  value_type: plain
  resource_type: Katello::ActivationKey
model: ReportTemplate
-%>
<% load_hosts(search: input('host')).each_record do |host| -%>
<%
  report_row(
    'Server FQND': host.name
```

```

    )
  -%>
<% end -%>
<%= report_render %>
"
}

```

2. 以下のリクエストを使用して、テンプレートをインポートします。

```

$ curl --insecure --user admin:redhat \
--data @Example_Template.json --header "Content-Type:application/json" \
--request POST --config https://satellite.example.com/api/report_templates/import

```

3. 以下のリクエストを使用して、レポートテンプレートの一覧を取得し、Satellite でテンプレートを表示できることを確認します。

```

$ curl --insecure --user admin:redhat \
--request GET --config https://satellite.example.com/api/report_templates | json_reformat

```

## 9.7. エンタイトルメントを監視するレポートテンプレートの作成

レポートテンプレートを使用して、特定のサブスクリプションを持つホストの一覧を返し、それらのホストのコア数を表示できます。テンプレートの作成に関する詳細は、[付録A テンプレート作成の参照](#)を参照してください。

### 手順

1. Satellite Web UI で、**監視** > **レポートテンプレート** に移動して、**テンプレートの作成** をクリックします。
2. オプション: **エディター** フィールドで **<%#>** タグを使用し、後の参照用として役立つと思われる情報を含むコメントを追加します。以下に例を示します。

```

<%#
name: Entitlements
snippet: false
model: ReportTemplate
require:
- plugin: katello
  version: 3.14.0
-%>

```

3. **load\_hosts()** マクロを使用して行を追加し、以下のメソッドと変数をマクロに入力します。

```

<%- load_hosts(includes: [:lifecycle_environment, :operatingsystem, :architecture,
:content_view, :organization, :reported_data, :subscription_facet, :pools =>
[:subscription]]).each_record do |host| -%>

```

使用できる変数の一覧を表示するには、**ヘルプ** タブをクリックし、**セーフモードのメソッドと変数** の表で、**Host::Managed** 行を探します。

4. **each** メソッドを使用して、**host.pools** 変数で行を追加します。以下に例を示します。

```
<%- host.pools.each do |pool| -%>
```

5. **report\_row()** メソッドで行を追加してレポートを作成し、レポートの一部としてターゲットにする変数を追加します。

```
<%-   report_row(
      'Name': host.name,
      'Organization': host.organization,
      'Lifecycle Environment': host.lifecycle_environment,
      'Content View': host.content_view,
      'Host Collections': host.host_collections,
      'Virtual': host.virtual,
      'Guest of Host': host.hypervisor_host,
      'OS': host.operatingsystem,
      'Arch': host.architecture,
      'Sockets': host.sockets,
      'RAM': host.ram,
      'Cores': host.cores,
      'SLA': host_sla(host),
      'Products': host_products(host),
      'Subscription Name': sub_name(pool),
      'Subscription Type': pool.type,
      'Subscription Quantity': pool.quantity,
      'Subscription SKU': sub_sku(pool),
      'Subscription Contract': pool.contract_number,
      'Subscription Account': pool.account_number,
      'Subscription Start': pool.start_date,
      'Subscription End': pool.end_date,
      'Subscription Guest': registered_through(host)
    ) -%>
```

6. テンプレートに終了ステートメントを追加します。

```
<%- end -%>
<%- end -%>
```

7. レポートを生成するには、**<%= report\_render -%>** マクロを追加する必要があります。

```
<%= report_render -%>
```

8. **送信** をクリックしてテンプレートを保存します。

## 9.8. レポートテンプレートのセーフモード

Satellite でレポートテンプレートを作成すると、セーフモードがデフォルトで有効化されます。セーフモードでは、レポートテンプレートで利用できるマクロと変数が制限されます。セーフモードは、レンダリングの問題を防ぎ、レポートテンプレートのベストプラクティスを実施します。サポートされているマクロと変数の一覧は、Satellite Web UI で利用可能です。

利用可能なマクロと変数を表示するには、Satellite Web UI で、**監視 > レポートテンプレート** に移動し、**テンプレートの作成** をクリックします。テンプレートの作成ウィンドウで、**ヘルプ** タブをクリックし、**セーフモードメソッド** を展開します。

セーフモードが有効な間に、セーフモードメソッドに一覧表示されていないマクロまたは変数を使用しようとすると、テンプレートエディターにエラーメッセージが表示されます。

Satellite のセーフモードのステータスを表示するには、Satellite Web UI で、**管理 > 設定** and click the **プロビジョニング** タブに移動します。セーフモードレンダリング 行を特定し、値を確認します。

## 第10章 ホストコレクションの設定

ホストコレクションは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。

### 10.1. 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を作成する方法を示しています。

#### 手順

1. Satellite Web UI で、**ホスト > ホストコレクション** に移動します。
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 **Name** 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 **Description** を追加します。
6. **Save** をクリックします。

#### CLI 手順

- ホストコレクションを作成するには、以下のコマンドを実行します。

```
# hammer host-collection create \  
--name "My_Host_Collection" \  
--organization "My_Organization"
```

### 10.2. 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。

#### 手順

1. Satellite Web UI で、**ホスト > ホストコレクション** に移動します。
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。

### 10.3. 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。



## 手順

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

Are you sure you want to remove host collection **Host Collection Name**?

4. **Remove** をクリックします。

## 10.4. ホストコレクションへのホストの一括追加

複数のホストをホストコレクションに追加できます。

### 前提条件

ホストコレクションに追加するために、ホストを Red Hat Satellite に登録しておく。ホストの登録に関する情報は、「[ホストの登録](#)」を参照してください。

ホストをホストコレクションに追加すると、Satellite 監査システムでは変更がログに記録されないので注意してください。

## 手順

1. Satellite Web UI で、**ホスト > ホストコレクション** に移動します。
2. ホストの追加先となるホストコレクションを選択します。
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。

## CLI 手順

- 複数のホストをホストコレクションに追加するには、以下のコマンドを実行します。

```
# hammer host-collection add-host \  
--host-ids My_Host_ID_1,My_Host_ID_2 \  
--id My_Host_Collection_ID
```

## 10.5. 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ホストを削除すると、データベースのホストコレクションのレコードは変更されないため、Satellite 監査システムでは変更はログに記録されないので注意してください。

## 手順

1. Satellite Web UI で、**ホスト > ホストコレクション** に移動します。
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。

## 10.6. 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。

### 10.6.1. 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

- この手順を実行する前に、追加予定のコンテンツを既存のリポジトリの1つで利用できるようにしておく。または追加しておく。
- ホストの割り当て先の環境にコンテンツをプロモートしておく。

#### 手順

1. Satellite Web UI で、**ホスト > ホストコレクション** に移動します。
2. パッケージの追加先となるホストコレクションを選択します。
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することができます。
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ます。次に、以下をクリックします。
  - **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することができます。
  - **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することができます。

### 10.6.2. インストール済みパッケージの表示

以下の手順を使用して、ホストのインストール済みパッケージを表示します。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト**に移動し、ホストの名前を選択します。
2. **コンテンツ** タブでは、**パッケージ**にインストール済みパッケージの一覧が表示されます。
3. パッケージの詳細を表示するには、そのパッケージを選択します。
  - **詳細** タブには、選択したパッケージの詳細が表示されます。
  - **ファイル**タブには、パッケージに含まれるファイルの一覧が表示されます。
  - **Dependencies** タブには、パッケージの依存関係が一覧表示されます。
  - **リポジトリ** タブには、選択したパッケージを含むリポジトリが一覧表示されます。
4. **ライブラリー** または **デフォルトの組織** でフィルターを設定できます。

### 10.6.3. パッケージのアップグレード

以下の手順を使用して、ホストのインストール済みパッケージを表示します。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト**に移動し、アップグレードするパッケージを含むホストの名前を選択します。
2. **コンテンツ** タブで **パッケージ** を選択します。
3. **Status** 列には、パッケージがアップグレード可能または最新の状態であるかが表示されます。最新のパッケージを更新することはできません。
4. パッケージの一覧から、アップグレードするパッケージを選択し、行の最後にある縦リーダーをクリックします。
5. Remote Execution を使用するには **Apply via Remote Execution** を選択するか、たとえば **リモート実行** をカスタマイズする場合はカスタムのリモート実行を適用します。たとえば、適用時に時間を設定します。
6. **送信** をクリックしてパッケージをアップグレードします。

### 10.6.4. ホストからのパッケージの削除

以下の手順を使用して、インストール済みパッケージをホストから削除します。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト**に移動し、削除するパッケージを含むホストを選択します。
2. **コンテンツ** タブで **パッケージ** を選択します。
3. 削除するパッケージの最後にある縦リーダーアイコンをクリックし、**Remove** オプションを選択します。
4. **Submit** をクリックします。

### 10.6.5. 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のいずれかで利用可能であるか、またはリポジトリにない場合は、この手順を開始する前にリポジトリに追加しておく必要がある。
- 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る。

### 手順

1. Satellite Web UI で、**ホスト > ホストコレクション** に移動します。
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。**via remote execution - customize first** メニューエントリーを選択すると、**Job invocation** 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。

#### 10.6.6. 単一ホストへのエラータの追加

以下の手順を使用して、エラータをホストに追加します。

### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動し、エラータを追加するホストを選択します。
2. **コンテンツ** ボタンをクリックして **エラータ** タブを選択します。
3. ホストに追加するエラータを選択するか、一覧の上部にあるチェックボックスを選択して、インストール可能な全エラータを追加します。完全なリストから削除するエラータの横にあるチェックボックスをクリックします。
4. ホストに追加するエラータの横にある縦リーダーアイコンを使用して、**Apply via Remote Execution** を使用するか、リモート実行をカスタマイズする場合はカスタマイズされた **Apply via customized remote execution** を選択します。SSH を使用してターゲットホストへの接続がない場合は、**Apply via Katello agent** を選択します。
5. **Submit** をクリックします。

#### 10.6.7. インストール可能なエラータの適用

以下の手順を使用して、インストール可能なエラータの一覧を表示し、インストールするエラータを選択します。

### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動し、必要なホストを選択します。

2. ホストに関連付けられているエラータがある場合は、新しいホストページの Installable Errata カードに表示されます。
3. **Content** タブでは、**Errata** には選択したホストのインストール可能なエラータが表示されます。
4. インストールするエラータのチェックボックスをクリックします。
5. リモート実行を使用するには、ホストに追加するエラータの横にある縦3つのドットのアイコンを使用して、**Apply via Remote Execution** を選択します。リモート実行をカスタマイズする場合は **Apply via custom remote execution** を選択します。SSH を使用してターゲットホストへの接続がない場合は、**Apply via Katello agent** を選択します。
6. **Submit** をクリックします。

### 10.6.8. タイプおよび重大度によるエラータのフィルター

以下の手順を使用して、エラータのタイプまたは重大度でエラータをフィルターリングします。

#### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動し、ホストの名前をクリックします。
2. **コンテンツ** タブで、**エラータ** は選択したホストに関連付けられているエラータを表示します。
3. **タイプ** ボタンをクリックして、エラータをタイプ別にフィルターします。
4. セキュリティー、バグ修正、または機能強化タイプのエラータをフィルターして表示できます。
5. **Severity** ボタンをクリックして重大度でフィルターします。
6. 重大度 N/A、低、中程度、重要、重大のエラータを表示するにはフィルターを設定できます。
7. 選択を解除するには、オプションの一覧に戻り、選択したオプションを再度クリックします。

ホストページで **Errata** カードを使用して、表示する前にタイプのエラータを事前フィルターすることもできます。

### 10.6.9. 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。

#### 手順

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。

#### 10.6.10. 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。

##### 手順

1. Satellite Web UI で、**ホスト > ホストコレクション** に移動します。
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. 一覧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11章 リモートジョブの設定およびセットアップ

このセクションでは、リモートホストでジョブを実行するように Satellite を設定する方法を説明します。

リモートホストで適用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。

### 11.1. ホストでのジョブの実行について

シェルスクリプトまたは Ansible タスクと Playbook を使用して、Capsules からリモートでホスト上でジョブを実行できます。これはリモート実行と呼ばれます。

作成したカスタムの Ansible ロール、またはダウンロードしたロールの場合、ロールを含むパッケージを Capsule のベースオペレーティングシステムにインストールする必要があります。Ansible ロールを使用する前に、ロールがインストールされている Capsule から Satellite にロールをインポートする必要があります。

通信は Capsule Server 経由で行われるので、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、スケーリングして多数のホストを管理できます。詳細は、「[リモート実行用のトランスポートモード](#)」を参照してください。

Satellite は、Embedded Ruby (ERB) 構文ジョブテンプレートを使用します。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。

シェルスクリプトおよび Ansible のジョブテンプレートが複数、デフォルトで含まれています。詳細は、[ホストの管理](#)の [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 のベースオペレーティングシステムは、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。

複数のホストでジョブを一度に実行でき、コマンドで変数を使用すると、実行するジョブをより詳細に制御できます。ホストファクトとパラメーターを使用して、変数の値を設定できます。

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。

詳細は、[ホストの管理](#)の [リモートジョブの実行](#)を参照してください。

### 11.2. リモート実行のワークフロー

ホストでリモートジョブを実行すると、すべてのホストについて、Satellite は以下のアクションを実行して、使用するリモート実行 Capsule を検出します。

Satellite は、リモート実行機能が有効になっている Capsules のみを検索します。

1. Satellite は、**Remote execution** のチェックボックスが選択されているホストのインターフェイスを検出します。
2. Satellite はこれらのインターフェイスのサブネットを検出します。
3. Satellite は、これらのサブネットに割り当てられたリモート実行 Capsule を検出します。

4. この Capsule のセットから、Satellite は実行中のジョブの数が最も少ない Capsule を選択します。こうすることで、Satellite はリモート実行 Capsule 間でのジョブの負荷を確実に分散させます。

**Prefer registered through Capsule for remote execution**を有効にした場合、Satellite はホストが登録されている Capsule を使用して REX ジョブを実行します。

デフォルトでは、**Prefer registered through Capsule for remote execution**は **No** に設定されています。これを有効にするには、Satellite Web UI で **Administer > Settings** に移動し、**Content** タブで、**Prefer registered through Capsule for remote execution** を **Yes** に設定します。これにより、Satellite が登録先の Capsule によってホスト上で REX ジョブを実行することが保証されます。

この段階で Satellite がリモート実行 Capsule を検出できず、**任意の Capsule へのフォールバック** 設定が有効になっている場合、Satellite は別の Capsule のセットを追加して、そこからリモート実行 Capsule を選択します。Satellite は、ホストに割り当てられている以下のタイプの Capsuleの中から最も負荷の少ない Capsule を選択します。

- ホストのサブネットに割り当てられた DHCP Capsule、DNS Capsule、および TFTP Capsule
- ホストのドメインに割り当てられた DNS Capsule
- ホストのレルムに割り当てられた Realm Capsule
- Puppet server Capsule
- Puppet CA Capsule
- OpenSCAP Capsule

この段階で Satellite がリモート実行 Capsule を検出せず、**グローバル Capsule の有効化** 設定が有効になっている場合、Satellite は、ホストの組織およびロケーションにあるすべての Capsule のセットから最も負荷の少ないリモート実行 Capsule を選択し、リモートジョブを実行します。

### 11.3. リモート実行用のパーミッション

インフラストラクチャー内でどのロールをどのジョブを実行できるか (ターゲットにするホストを含めて) 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**: すべてのリモート実行機能にアクセスできます。
- **リモート実行ユーザー**: ジョブの実行のみ可能です。

**Remote Execution User** 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。カスタマイズされたロールで **view\_job\_templates** パーミッションを使用して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くジョブのみを確認し、トリガーすることが可能です。**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。

**execute\_jobs\_on\_infrastructure\_hosts** パーミッションを使用して、Red Hat Satellite にホストとして登録された Red Hat Satellite および Capsule に対してリモート実行ジョブを実行できます。標準の **Manager** と **Site Manager** のロールには、デフォルトでこのパーミッションがあります。**Manager** ま



または **Site Manager** ロールのいずれかを使用する場合、または **execute\_jobs\_on\_infrastructure\_hosts** パーミッションを持つカスタムロールを使用する場合は、登録済みの Red Hat Satellite および Capsule ホストに対してリモートジョブを実行できます。

ロールおよびパーミッションの使用に関する詳細は、**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 = webserver
```

この例の最初の行を使用して、選択した1つのホストに **Reboot** テンプレートを適用します。2行目を使用して、名前の末尾が **.staging.example.com** となるように、ホストのプールを定義します。3行目を使用して、テンプレートをホストグループにバインドします。



#### 注記

これらのロールを持つユーザーに割り当てられるパーミッションは、時間の経過とともに変化する可能性があります。将来実行するジョブがすでにスケジュールされている場合に、パーミッションが変更されると、ジョブの実行直前にパーミッションがチェックされるため、実行が失敗する可能性があります。

## 11.4. リモート実行用のトランスポートモード

Satellite が、リモートジョブ実行に2つの異なるトランスポートモードを使用するように設定できます。

**ssh** モードの Capsule では、リモート実行は SSH サービスを使用してジョブの詳細を転送します。これは、デフォルトのトランスポートモードです。ターゲットホストで SSH サービスを有効にし、アクティブにする必要があります。リモート実行 Capsule は、ターゲットホストの SSH ポートにアクセスする必要があります。別の設定がない限り、標準の SSH ポートは 22 です。

**pull-mqtt** モードの Capsules では、リモート実行は Message Queueing Telemetry Transport (MQTT) を使用して Satellite Server から受信するジョブを公開します。ホストは、**yggdrasil** プルクライアントを使用してジョブ通知のために Capsule の MQTT ブローカーをサブスクライブします。ホストが通知を受信したら、HTTPS 経由で Capsule からジョブ詳細をプルし、ジョブを実行して結果を Capsule に報告します。

**pull-mqtt** モードを使用するには、Capsule Server でこれを有効にし、ターゲットホストでプルクライアントを設定する必要があります。

#### 関連情報

- Capsule Server でプルモードを有効にするには、**Capsule Server のインストール**の [プルクライアントのリモート実行の設定](#) を参照してください。
- 既存のホストでプルモードを有効にするには、「[プル要求を使用するためのホストの設定](#)」に進みます。
- Katello Agent からホストを移行するには、「[Katello エージェントからリモート実行への移行](#)」を参照してください。

- 新しいホストでプルモードを有効にするには、[ホストの作成](#) または [ホストの登録](#) に進みます。

## 11.5. プル要求を使用するためのホストの設定

**pull-mqtt** モードを使用するように設定された Capsule の場合には、ホストはリモート実行プルクライアントを使用してリモートジョブにサブスクライブできます。管理対象ホストでは、Capsule Server への SSH 接続は必要ありません。

### 前提条件

- ホストを Satellite に登録している。
- ホストの Capsule が **pull-mqtt** モードを使用するように設定されている。詳細は、[Capsule Server のインストール](#) の [プルクライアントのリモート実行の設定](#) を参照してください。
- Red Hat Satellite Client 6 リポジトリは、Satellite Server で有効化および同期され、ホストで有効化されている。
- ホストは、ポート **1883** を使用して MQTT 経由で Capsule と通信できる。
- ホストは、HTTPS 経由で Capsule と通信できる。



### 注記

**katello-pull-transport-migrate** パッケージが作成され、ユーザーはプルクライアントを使用して Katello Agent からリモート実行に移行できるようになりました。ただし、ホストに Katello Agent をインストールする必要はありません。Katello Agent がインストールされているかどうかに関係なく、**katello-pull-transport-migrate** を使用できます。

### 手順

1. ホストに **katello-pull-transport-migrate** パッケージをインストールします。
  - Red Hat Enterprise Linux 8 ホストおよび Red Hat Enterprise Linux 9 ホストの場合:

```
# dnf install katello-pull-transport-migrate
```

- Red Hat Enterprise Linux 7 ホストの場合:

```
# yum install katello-pull-transport-migrate
```

このパッケージは、依存関係として **foreman\_ygg\_worker** および **yggdrasil** をインストールし、ホストでプルモードを有効にします。ホストの **subscription-manager** 設定およびコンシューマー証明書は、ホストで **yggdrasil** クライアントを設定するために使用され、プルモードのクライアントワーカーを起動します。

2. オプション: プルクライアントが実行中であり、適切に設定されていることを確認するには、**yggdrasild** サービスのステータスを確認します。

```
# systemctl status yggdrasild
```

3. オプション: パッケージのインストール後に、ホストから **katello-agent** を削除できます。

**警告**

お使いのホストが Red Hat Virtualization のバージョン 4.4 以下にインストールされている場合は、削除された依存関係によりホストが破損するので、**katello-agent** パッケージを削除しないでください。

## 11.6. ジョブテンプレートの作成

以下の手順を使用してジョブテンプレートを作成します。Satellite Web UI の代わりに CLI を使用する場合は、[CLI 手順](#) を参照してください。

### 手順

1. Satellite Web UI で、**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。
3. **テンプレート** タブをクリックして、**名前** フィールドにジョブテンプレートの一意名を入力します。
4. **デフォルト** を選択して、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できるようにします。
5. テンプレートエディターで直接テンプレートを作成するか、**インポート** をクリックしてテキストファイルからテンプレートをアップロードします。
6. オプション: **監査コメント** フィールドで、変更に関する情報を追加します。
7. **Job** タブをクリックし、**Job category** フィールドに独自のカテゴリを入力するか、**ホストの管理** の **デフォルトのジョブテンプレートカテゴリ** にリストされているデフォルトカテゴリから選択します。
8. オプション: **説明形式** フィールドで説明テンプレートを入力します。(例: **Install package %  
{package\_name}**)。また、テンプレートでは **%{template\_name}** および **%{job\_category}** も使用できます。
9. **プロバイダータイプ** リストから、Shell スクリプトに **SSH** を、Ansible タスクまたは Playbook に **Ansible** を選択します。
10. オプション: **Timeout to kill** フィールドで、ジョブが完了しない場合に、ジョブを中断するタイムアウトの値を入力します。
11. オプション: **入力を追加** をクリックし、入力パラメーターを定義します。ジョブの実行時にパラメーターを要求し、テンプレートに定義する必要はありません。各種サンプルについては、**ヘルプ** タブを参照してください。
12. オプション: **外部入力セット** をクリックして、このジョブの他のテンプレートを追加します。
13. オプション: **実効ユーザー** エリアで、コマンドでデフォルトの **remote\_execution\_effective\_user** 設定を使用できない場合に、ユーザーを設定します。

14. **オプション**: このテンプレートをスニペットとして他のテンプレートに追加する場合は、**タイプ** タブをクリックして、**スニペット** を選択します。
15. **ロケーション** タブをクリックして、テンプレートを使用するロケーションを追加します。
16. **組織** タブをクリックして、テンプレートを使用する組織を追加します。
17. **送信** をクリックして変更を保存します。

テンプレート構文に他のテンプレートを追加して、ジョブテンプレートを拡張およびカスタマイズできます。詳細は、**ホストの管理** の [テンプレート作成リファレンス](#) および [ジョブテンプレートの例と拡張機能](#) を参照してください。

## CLI 手順

- テンプレート定義ファイルを使用してジョブテンプレートを作成するには、以下のコマンドを使用します。

```
# hammer job-template create \
--file "Path_to_My_Template_File" \
--job-category "My_Category_Name" \
--name "My_Template_Name" \
--provider-type SSH
```

## 11.7. 名前別での ANSIBLE PLAYBOOK のインポート

Capsule にインストールされているコレクションから Satellite に、名前を使用して Ansible Playbook をインポートできます。

### 前提条件

- Satellite の Ansible プラグインが有効になっている

### 手順

1. 以下の API 要求を使用して、利用可能な Ansible Playbook を取得します。

```
# curl -X GET 'Content-Type: application/json'
https://satellite.example.com/ansible/api/v2/ansible_playbooks/fetch?
proxy_id=My_capsule_ID
```

2. インポートする Ansible Playbook を選択し、その名前を書き留めます。
3. その名前を使用して Ansible Playbook をインポートします。

```
# curl -X PUT 'Content-Type: application/json' -d '{"playbook_names":
["My_Playbook_Name"]}'
https://satellite.example.com/ansible/api/v2/ansible_playbooks/sync?
proxy_id=My_capsule_ID
```

インポートが完了すると、Satellite Web UI で通知が表示されます。

## 11.8. 利用可能なすべての ANSIBLE PLAYBOOK のインポート

利用可能なすべての Ansible Playbook は、Capsule にインストールされているコレクションから Satellite にインポートできます。

### 前提条件

- Satellite の Ansible プラグインが有効になっている

### 手順

- 以下の API 要求を使用して Ansible Playbook をインポートします。

```
# curl -X PUT 'Content-Type: application/json'  
https://satellite.example.com/ansible/api/v2/ansible_playbooks/sync?proxy_id=My-  
capsule-ID
```

インポートが完了すると、Satellite Web UI で通知が表示されます。

## 11.9. SATELLITE での任意の CAPSULE へのフォールバックリモート実行の設定

任意の Capsule へのフォールバック設定を有効にして、ホストに割り当てられている Capsule の一覧からリモート実行 Capsule を検索するように Satellite を設定できます。これは、サブネットが設定されていないホストでリモートジョブを実行する必要がある場合、またはリモート実行機能が有効になっていない Capsule にホストのサブネットが割り当てられている場合に役立ちます。

任意の Capsule へのフォールバック設定が有効になっている場合、Satellite は別の Capsule のセットを追加して、そこからリモート実行 Capsule を選択します。また、以下のように Satellite は、ホストに割り当てられたすべての Capsule のセットから最も負荷の少ない Capsule を選択します。

- ホストのサブネットに割り当てられた DHCP Capsule、DNS Capsule、および TFTP Capsule
- ホストのドメインに割り当てられた DNS Capsule
- ホストのレルムに割り当てられた Realm Capsule
- Puppet server Capsule
- Puppet CA Capsule
- OpenSCAP Capsule

### 手順

1. Satellite Web UI で、**Administer > Settings** に移動します。
2. **Remote Execution** をクリックします。
3. **任意の Capsule へのフォールバック** を設定します。

### CLI 手順

- Satellite で **hammer settings set** コマンドを入力して、**任意の Capsule へのフォールバック** を設定します。値を **true** に設定するには、次のコマンドを入力します。

```
# hammer settings set \
--name=remote_execution_fallback_proxy \
--value=true
```

## 11.10. SATELLITE でのグローバル CAPSULE リモート実行の設定

デフォルトで Satellite は、Capsule がホストのサブネットに割り当てられているかどうかに関係なく、ホストの組織とロケーションでリモート実行 Capsule を検索します。ホストのサブネットに割り当てられている Capsule に検索を限定する場合は、**グローバル Capsule の有効化** 設定を無効化することができます。

**グローバル Capsule の有効化** 設定が有効になっている場合、Satellite は別の Capsule のセットを追加して、そこからリモート実行 Capsule を選択します。また、Satellite は、ホストの組織およびロケーションにあるすべての Capsule のセットから最も負荷の少ないリモート実行 Capsule を選択し、リモートジョブを実行します。

### 手順

1. Satellite Web UI で、**Administer** > **Settings** に移動します。
2. **Remote Execution** をクリックします。
3. **グローバル Capsule の有効化** を設定します。

### CLI 手順

- Satellite で **hammer settings set** コマンドを入力して、**Enable Global Capsule** を設定します。値を **true** に設定するには、次のコマンドを入力します。

```
# hammer settings set \
--name=remote_execution_global_proxy \
--value=true
```

## 11.11. 代替ディレクトリーを使用してホストでリモートジョブを実行するための SATELLITE の設定

デフォルトで Satellite は、クライアントシステムの **/var/tmp** ディレクトリーを使用して、リモート実行ジョブを実行します。クライアントシステムの **/var/** ボリュームまたはファイルシステムに **noexec** が設定されている場合は、代替ディレクトリーを使用するように Satellite を設定する必要があります。設定されない場合はスクリプトを実行できないので、リモート実行ジョブは失敗します。

### 手順

1. 新しいディレクトリーを作成します。

```
# mkdir /My_Remote_Working_Directory
```

2. デフォルトの **var** ディレクトリーから SELinux コンテキストをコピーします。

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

3. システムを設定します。

```
# satellite-installer \
--foreman-proxy-plugin-remote-execution-script-remote-working-dir
/My_Remote_Working_Directory
```

## 11.12. リモート実行のための SSH 鍵の配布

**ssh** モードの Capsule の場合、リモート実行接続は SSH を使用して認証されます。Capsule からの公開 SSH 鍵は、管理するアタッチ済みのホストに配布する必要があります。

ホストで SSH サービスが有効化され、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、ポート 22 へのアクセスを有効化します。

Capsule からターゲットホストに公開 SSH 鍵を配布するには、以下のいずれか1つの方法を使用します。

1. [「リモート実行用の SSH 鍵の手動での配布」](#)
2. [「Satellite API を使用したリモート実行用の SSH 鍵の取得」](#)
3. [「プロビジョニング中に SSH キーを配布するためのキックスタートテンプレートの設定」](#)
4. 新しい Satellite ホストの場合、グローバル登録テンプレートを使用して、登録中に SSH 鍵を Satellite ホストにデプロイすることができます。詳細は、[ホストの管理](#) の [グローバル登録テンプレートを使用した Red Hat Satellite へのホストの登録](#) を参照してください。

Satellite は、リモート実行機能の SSH 鍵を、デフォルトで Satellite からプロビジョニングされたホストに配布します。

ホストが Amazon Web Services で実行されている場合は、パスワード認証を有効にします。詳細は、[New User Accounts](#) を参照してください。

## 11.13. リモート実行用の SSH 鍵の手動での配布

SSH 鍵を手動で配布するには、以下の手順を実行します。

### 手順

- SSH 公開キーを Capsule からターゲットホストにコピーします。

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

管理するターゲットホストごとにこの手順を繰り返します。

### 検証

- ターゲットホストに鍵が正常に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入力します。

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

## 11.14. SATELLITE API を使用したリモート実行用の SSH 鍵の取得

Satellite API を使用して Capsule から公開鍵をダウンロードするには、各ターゲットホストでこの手順を実行します。

#### 手順

1. ターゲットホストで `~/.ssh` ディレクトリーを作成し、SSH 鍵を保存します。

```
# mkdir ~/.ssh
```

2. Capsule から SSH 鍵をダウンロードします。

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

3. `~/.ssh` ディレクトリーのパーミッションを設定します。

```
# chmod 700 ~/.ssh
```

4. `authorized_keys` ファイルのパーミッションを設定します。

```
# chmod 600 ~/.ssh/authorized_keys
```

## 11.15. プロビジョニング中に SSH キーを配布するためのキックスタートテンプレートの設定

`Remote_execution_ssh_keys` スニペットをカスタムキックスタートテンプレートに追加して、プロビジョニング中に SSH キーをホストにデプロイできます。Satellite に同梱されるキックスタートテンプレートには、デフォルトでこのスニペットが含まれています。Satellite は、プロビジョニング中にリモート実行用の SSH キーをシステムにコピーします。

#### 手順

- 新しくプロビジョニングされたホストに公開鍵を含めるには、使用するキックスタートテンプレートに以下のスニペットを追加します。

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

## 11.16. KERBEROS チケットを付与するための KEYTAB の設定

以下の手順を使用して、Satellite が Keytab を使用して Kerberos 付与チケットを取得するように設定します。Keytab が設定されていないと、手動でチケットを取得する必要があります。

#### 手順

1. `foreman-proxy` ユーザーの ID を特定します。

```
# id -u foreman-proxy
```

2. 新しいファイルのパーミッションが **600** になるように、`umask` の値を変更します。

```
# umask 077
```



3. Keytab のディレクトリーを作成します。

```
# mkdir -p "/var/kerberos/krb5/user/My_User_ID"
```

4. Keytab を作成するか、既存の Keytab をディレクトリーにコピーします。

```
# cp My_Client.keytab /var/kerberos/krb5/user/My_User_ID/client.keytab
```

5. ディレクトリーの所有者を **foreman-proxy** ユーザーに変更します。

```
# chown -R foreman-proxy:foreman-proxy "/var/kerberos/krb5/user/My_User_ID"
```

6. keytab ファイルが読み取り専用であることを確認します。

```
# chmod -wx "/var/kerberos/krb5/user/My_User_ID/client.keytab"
```

7. SELinux コンテキストを復元します。

```
# restorecon -RvF /var/kerberos/krb5
```

## 11.17. リモート実行用の KERBEROS 認証の設定

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

### 前提条件

- Kerberos サーバーに Satellite Server を登録する。
- Kerberos サーバーに Satellite ターゲットホストを登録する。
- リモート実行用に Kerberos ユーザーアカウントを設定して初期化する。
- 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する。

### 手順

1. 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-script-ssh-kerberos-auth true
```

2. 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 **Administer > Settings** に移動して、**Remote Execution** タブをクリックします。**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。
3. **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。

### 検証

- 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。詳細は、[ホストの管理](#) の [リモートジョブの実行](#) を参照してください。

## 11.18. ジョブテンプレートのセットアップ

Satellite は、ジョブ実行に使用可能なデフォルトのジョブテンプレートを提供します。ジョブテンプレートの一覧を表示するには、[ホスト > ジョブテンプレート](#) に移動します。テンプレートを変更せずに使用する場合は、[ホストの管理](#) の [リモートジョブの実行](#) 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ができます。デフォルトのジョブテンプレートは、編集できないようにロックされています。テンプレートのクローンを作成して、作成したクローンを編集します。

### 手順

1. テンプレートのクローンを作成するには、[アクション](#) コラムで、[クローン](#) を選択します。
2. クローンに一意名を指定して、[送信](#) をクリックして変更を保存します。

ジョブテンプレートは、Embedded Ruby (ERB) 構文を使用します。テンプレート作成の詳細は、[ホストの管理](#) ガイドの [テンプレート作成の参照](#) を参照してください。

### Ansible の考慮事項

Ansible ジョブテンプレートを作成するには、以下の手順を使用し、ERB 構文ではなく YAML 構文を使用します。テンプレートは `---` で開始します。Ansible playbook YAML ファイルをジョブテンプレートの本文に埋め込みます。また、ERB 構文を追加して、YAML Ansible テンプレートをカスタマイズすることも可能です。Satellite に Ansible Playbook をインポートすることも可能です。詳細は、[ホストの管理](#) の [レポジトリテンプレートの同期](#) を参照してください。

### パラメーター変数

ランタイム時に、ジョブテンプレートはホストに定義するパラメーター変数を受け取ることができます。ホストの編集ページの [パラメーター](#) タブで表示されるパラメーターのみが、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点にご留意ください。ランタイム時に Ansible ジョブテンプレートがパラメーター変数を受け取らない場合には、Satellite Web UI で [管理 > 設定](#) に移動して、[Ansible](#) タブをクリックします。トップレベルの [Ansible 変数](#) の行で、[Value](#) パラメーターを [No](#) に変更します。

## 11.19. リモートジョブの実行

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ことができます。

Satellite Web UI の代わりに CLI を使用する場合は、[CLI 手順](#) を参照してください。

### 手順

1. Satellite Web UI で、[ホスト > すべてのホスト](#) に移動し、リモートジョブを実行する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を絞り込むことができます。
2. [アクションの選択](#) リストから [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。
4. 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。

5. オプション: **ブックマーク** リスト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。
6. オプション: **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、クエリー変更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。
7. 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の詳細は、[ジョブテンプレートの作成](#) を参照してください。
8. オプション: ジョブの詳細設定を行うには、**詳細フィールドの表示** をクリックします。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** は一度に実行するジョブの最大数を定義します。この定義で、多数のホストでジョブを実行する時に、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。
  - **Timeout to kill** は、ジョブが終了しない場合に、ジョブの強制終了までの間隔 (秒単位) を定義します。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。
  - **実行順序** は、ジョブがホストで実行される順序を決定します (アルファベット順またはランダム)。  
**同時実行レベル** および **Timeout to kill** 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することができます。
9. 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 回限りのジョブを未来の日付で定義したり、定期的なジョブを設定することもできます。定期的なジョブについては、開始日と終了日、実行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。cron の詳細は、[cron での Linux システムタスクの自動化](#) を参照してください。
10. **Submit** をクリックします。同じページの **Recent Jobs** セクションでジョブのステータスを表示できます。

## CLI 手順

1. Satellite で以下のコマンドを入力します。

```
# hammer settings set \
  --name=remote_execution_global_proxy \
  --value=false
```

2. 使用するジョブテンプレートの ID を検出します。

```
# hammer job-template list
```

3. テンプレートの詳細を表示して、テンプレートに必要なパラメーターを確認します。

```
# hammer job-template info --id My_Template_ID
```

4. カスタムパラメーターでリモートジョブを実行します。

```
# hammer job-invocation create \
--inputs My_Key_1="My_Value_1",My_Key_2="My_Value_2",... \
--job-template "My_Template_Name" \
--search-query "My_Search_Query"
```

**My\_Search\_Query** は、ホストを定義するフィルター式 (例: "name ~ My\_Pattern ") に置き換えます。hammer を使用したリモートコマンド実行に関する詳細については、**hammer job-template --help** および **hammer job-invocation --help** を入力します。

## 11.20. ホストの定期的な ANSIBLE ジョブのスケジュール

ホストで Ansible ロールを実行する定期的なジョブをスケジュールできます。

### 前提条件

- **view\_foreman\_tasks**、**view\_job\_invocations**、および **view\_repeat\_logics** 権限があることを確認する。

### 手順

1. Satellite Web UI で、**ホスト > すべてのホスト** に移動し、リモートジョブを実行するターゲットホストを選択します。
2. **Ansible** タブで、**ジョブ** を選択します。
3. **Schedule recurring job** をクリックします。
4. **Create New Recurring Ansible Run** ウィンドウで、最初の実行の繰り返し頻度、開始時刻、日付を定義します。
5. **Submit** をクリックします。
6. オプション: ホストの概要で、または **Ansible > Jobs** に移動して、スケジュールされた Ansible ジョブを表示します。

## 11.21. ホストグループの定期的な ANSIBLE ジョブのスケジュール

ホストグループで Ansible ロールを実行する定期的なジョブをスケジュールできます。

### 手順

1. Satellite Web UI で、**Configure > Host groups** に移動します。
2. **Actions** 列で、Ansible ロールの実行をスケジュールするホストグループの **Configure Ansible Job** を選択します。
3. **Schedule recurring job** をクリックします。
4. **Create New Recurring Ansible Run** ウィンドウで、最初の実行の繰り返し頻度、開始時刻、日付を定義します。

5. **Submit** をクリックします。

## 11.22. ジョブの監視

ジョブの実行中に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。

Ansible ジョブは、ホスト 100 台で一括して実行するので、特定のホストで実行するジョブをキャンセルできません。Ansible Playbook を全ホスト上で一括して実行してからでないと、ジョブは完了しません。

### 手順

1. Satellite Web UI で、**Monitor > Jobs**に移動します。このページは、**Execute now** 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、**監視 > 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** コラムで、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監視できる **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。

### CLI 手順

1. ジョブの ID を検出します。

```
# hammer job-invocation list
```

2. ジョブの出力を監視します。

```
# hammer job-invocation output \  
--host "My_Host_Name" \  
--id My_Job_ID
```

3. オプション: ジョブをキャンセルするには、次のコマンドを入力します。

```
# hammer job-invocation cancel \  
--id My_Job_ID
```

## 第12章 SATELLITE のホストステータス

Satellite では、各ホストには、注意が必要なホストを示すグローバルステータスがあります。各ホストには、特定の機能のステータスを表すサブステータスもあります。サブステータスが変更されると、グローバルステータスが再計算され、すべてのサブステータスのステータスによって結果が決定されます。

### 12.1. ホストのグローバルステータスの概要

グローバルステータスは、特定のホストの全体的なステータスを表します。ステータスの値としては、**OK**、**Warning**、**Error** の 3 つがあります。グローバルステータスは、ホストの概要ページで確認できます。ステータスについては、ホスト名の横に小さなアイコンがステータスに対応する色で表示されます。アイコンにカーソルを合わせると、サブステータス情報を含むツールチップが表示され、詳細をすばやく確認できます。ホストのグローバルステータスを表示するには、Satellite Web UI で、**ホスト > すべてのホスト** に移動します。

#### OK

サブステータスで報告されているエラーはありません。このステータスは緑色で強調表示されます。

#### Warning

エラーは検出されませんでした。一部のサブステータスで警告が発生しました。たとえば、ホストがレポートを送信するように設定されているにもかかわらず、ホストの設定管理レポートがありません。警告を調査して、デプロイメントが正常に保たれていることを確認することをお勧めします。このステータスは黄色で強調表示されます。

#### Error

一部のサブステータスで失敗が報告されています。たとえば、実行に失敗したリソースが含まれています。このステータスは赤色で強調表示されます。

#### 検索構文

ステータスに基づきホストを検索する場合は、**Administering Satellite** ガイドの [検索およびブックマーク機能](#) の章で大まかに説明されている Satellite の検索構文を使用してから、次のステータス関連の例を使用して検索を構築します。

OK ステータスのホストを検索する場合:

```
global_status = ok
```

注意すべきホストを検索する場合:

```
global_status = error or global_status = warning
```

### 12.2. ホストサブステータスの概要

サブステータスは、ホスト機能の一部のみをモニターリングします。

現在、Satellite には **Build** と **Configuration** のサブステータスのみがあります。Satellite に追加するプラグインによっては、さらに多くのサブステータスが存在する可能性があります。

**build** は、マネージドホストと、Satellite が無人モードで実行されている場合のサブステータスです。

**configuration** は、Satellite が Ansible、Puppet、Salt などの設定管理システムを使用している場合のみのサブステータスです。

ホストのサブステータスを表示するには、Satellite Web UI で **Hosts > All Hosts** に移動し、完全なステータスを検査するホストをクリックします。各ホストのホバーヘルプでサブステータス情報を表示することもできます。

ホストの詳細ページの **Properties** テーブルで、グローバルホストステータスとすべてのサブステータスの両方を表示できます。

各サブステータスには、それぞれ 3 つのグローバルステータス値のいずれかにマッピングされた独自の値をセットで定義できます。

**Build** サブステータスには、**pending** と **built** の 2 つの値があり、どちらもグローバルステータスの **OK** にマッピングされています。

**Configuration** ステータスの値はそれよりも多く、次のようにグローバルステータスにマッピングされています。

### グローバルステータスの OK にマッピングされるサブステータス

#### Active

前回の実行中に、いくつかのリソースが適用されました。

#### Pending

前回の実行中に、いくつかのリソースが適用されましたが、設定管理インテグレーションは **noop** モードで実行するように設定されました。

#### No changes

前回の実行中、何も変更されませんでした。

#### No reports

このサブステータスは、**Warning** と **OK** のいずれかになります。これは、レポートはないが、たとえば関連する設定管理プロキシなどをホストが使用する場合、または **always\_show\_configuration\_status** 設定が **true** に設定されている場合に発生し、**Warning** にマッピングされます。

### グローバルステータスの Error にマッピングされるサブステータス

#### エラー

これは、設定中にエラーが発生したことを示します。たとえば、実行でパッケージのインストールに失敗した場合などです。

### グローバルステータスの Warning にマッピングされるサブステータス

#### Out of sync

**outofsync\_interval** に基づき、想定された間隔内に設定レポートを受信しませんでした。レポートは送信元によって識別され、それに基づいて異なる間隔を持つことができます。

#### No reports

ホストが設定管理システムを使用しているが、Satellite がレポートを受信しない場合 **Warning** にマッピングされます。それ以外の場合は、**OK** にマッピングされます。

### 検索構文

サブステータスに基づきホストを検索する場合は、**Satellite の管理** ガイドの [検索およびブックマーク](#) の章で大まかに説明されている Satellite の検索構文を使用してから、次のステータス関連の例を使用して検索を構築します。

最後に報告された状態に基づいて、ホストの設定サブステータスを検索します。

たとえば、保留中のリソースが少なくとも1つあるホストを検索する場合:

```
status.pending > 0
```

前回の実行中に一部のサービスを再起動したホストを検索する場合:

```
status.restarted > 0
```

最後の実行で何かが起きた可能性があることを示す、様子の異なるホストを検索する場合:

```
status.interesting = true
```



## 第13章 テンプレートリポジトリの同期

Satellite では、Satellite Server とバージョン管理システムまたはローカルディレクトリー間で、ジョブテンプレート、プロビジョニングテンプレート、レポートテンプレート、およびパーティションテーブルテンプレートのリポジトリを同期できます。本章では、Git リポジトリをデモ目的で使します。

このセクションでは、TemplateSync プラグインをインストールおよび設定し、エクスポートおよびインポートタスクを実行するためのワークフローについて詳しく説明します。

### 13.1. TEMPLATESYNC プラグインの有効化

#### 手順

1. Satellite Server でプラグインを有効化するには、以下のコマンドを入力します。

```
# satellite-installer --enable-foreman-plugin-templates
```

2. プラグインが適切にインストールされていることを確認するには、**管理 > 設定** に **TemplateSync** メニューがあることを確認します。

### 13.2. TEMPLATESYNC プラグインの設定

Satellite Web UI で、**管理 > 設定 > TemplateSync** に移動して、プラグインを設定します。以下の表は、属性の動作を説明しています。一部の属性は、タスクのインポートまたはエクスポートにのみ使用される点にご留意ください。

表13.1 テンプレートのプラグイン設定の同期

パラメーター	API パラメーター名	インポートの意味	エクスポートの意味
関連付け	<b>associate</b>  許可される値: <b>always</b> 、 <b>new</b> 、 <b>never</b>	OS、組織、およびロケーションベースのメタデータへのテンプレートの関連付け	該当なし
ブランチ	<b>branch</b>	Git リポジトリで、読み取るデフォルトブランチを指定します。	Git リポジトリで、書き込むデフォルトブランチを指定します。
ディレクトリー名	<b>dirname</b>	リポジトリ下で、読み込むサブディレクトリーを指定します。	リポジトリ下で、書き込むサブディレクトリーを指定します。
フィルター	<b>filter</b>	正規表現に一致する名前を持つテンプレートだけをインポートします。	正規表現に一致する名前を持つテンプレートだけをエクスポートします。

パラメーター	API パラメーター名	インポートの意味	エクスポートの意味
強制インポート	<b>force</b>	インポートしたテンプレートで、ロックされている同じ名前のテンプレートを上書きします。	該当なし
テンプレートのロック	<b>lock</b>	<b>強制インポート</b> が有効になっていない限り、同じ名前の新しいテンプレートをインポートするときに既存のテンプレートを上書きしないでください。	該当なし
メタデータエクスポートモード	<b>metadata_export_mode</b>  許可される値: <b>refresh</b> 、 <b>keep</b> 、 <b>remove</b>	該当なし	エクスポートする際にメタデータが処理される方法を定義します。  <ul style="list-style-type: none"> <li>● <b>リフレッシュ</b>: テンプレートコンテンツから既存のメタデータを削除して、現在の割り当ておよび属性をベースにしたメタデータを新たに生成します。</li> <li>● <b>維持</b>: 既存のメタデータを保持します。</li> <li>● <b>削除</b>: メタデータがないテンプレートをエクスポートします。メタデータを手動で追加する場合は便利です。</li> </ul>
否定	<b>negate</b>  許可される値: <b>true</b> 、 <b>false</b>	フィルター属性を無視するテンプレートをインポートします。	フィルター属性を無視するテンプレートをエクスポートします。
接頭辞	<b>prefix</b>	テンプレート名は接頭辞で開始しないため、指定した文字列をテンプレートの頭に追加します。	該当なし
リポジトリ	<b>repo</b>	同期するリポジトリへのパスを定義します。	エクスポートするリポジトリへのパスを定義します。

パラメーター	API パラメーター名	インポートの意味	エクスポートの意味
詳細	<b>verbose</b>  許可される値: <b>true、false</b>	このアクションについて、詳細なメッセージをログに記録します。	該当なし

## 13.3. テンプレートのインポートおよびエクスポート

Satellite Web UI、Hammer CLI、または Satellite API を使用して、テンプレートをインポートおよびエクスポートできます。Satellite API 呼び出しは、ロールベースのアクセス管理システムを使用して、任意のユーザーでタスクの実行が可能になります。Git などのバージョン管理システム、またはローカルディレクトリーとテンプレートを同期できます。

### 13.3.1. テンプレートのインポート

任意のリポジトリからテンプレートをインポートできます。`/tmp/dir`、`git://example.com`、`https://example.com`、`ssh://example.com` などの異なるプロトコルを使ってリポジトリにポイントさせることもできます。

#### 前提条件

- 各テンプレートに、テンプレートが属するロケーションおよび組織が含まれている必要があります。これは、すべてのタイプのテンプレートタイプに適用されます。テンプレートをインポートする前に、以下のセクションをテンプレートに追加します。

```
<%#
kind: provision
name: My_Provisioning_Template
oses:
- My_first_OS
- My_second_OS
locations:
- My_first_Location
- My_second_Location
organizations:
- My_first_Organization
- My_second_Organization
%>
```

#### 手順

- Satellite Web UI で、**ホスト > テンプレートの同期** に移動します。
- インポート** をクリックします。
- 各フィールドには、**管理 > 設定 > TemplateSync** で設定された値が入力されます。インポートするテンプレートに従って値を変更します。各フィールドの詳細は、[「TemplateSync プラグインの設定」](#) を参照してください。
- Submit** をクリックします。

Satellite Web UI はインポートのステータスを表示します。ステータスは永続的ではありません。ステータスページを離れると、ページに戻ることはできません。

## CLI 手順

- リポジトリからテンプレートをインポートするには、以下のコマンドを実行します。

```
$ hammer import-templates \
  --branch "My_Branch" \
  --filter '.*Template Name$' \
  --organization "My_Organization" \
  --prefix "[Custom Index]" \
  --repo "https://git.example.com/path/to/repository"
```

テンプレートのインデックス化と管理には、**--prefix** を使ってテンプレートにカテゴリーを設定することができます。大型リポジトリから特定のテンプレートを選択するには、**--filter** を使ってインポートするテンプレートのタイトルを定義します。たとえば、**--filter '.\*Ansible Default\$'** とすると、各種 Ansible Default テンプレートをインポートします。

### 13.3.2. テンプレートのエクスポート

テンプレートは、Git リポジトリなどのバージョン管理サーバーにエクスポートできます。

## 手順

- Satellite Web UI で、**ホスト > テンプレートの同期** に移動します。
- エクスポート** をクリックします。
- 各フィールドには、**管理 > 設定 > TemplateSync** で設定された値が入力されます。エクスポートするテンプレートに従って値を変更します。各フィールドの詳細は、[「TemplateSync プラグインの設定」](#) を参照してください。
- Submit** をクリックします。

Satellite Web UI はエクスポートのステータスを表示します。ステータスは永続的ではありません。ステータスページを離れると、ページに戻ることはできません。

## CLI 手順

- Git リポジトリのローカルコピーのクローンを作成します。

```
$ git clone https://github.com/foreman/community-templates /custom/templates
```

- 以下のコマンドで、ローカルディレクトリの所有者を **foreman** ユーザーに変更し、SELinux コンテキストを変更します。

```
# chown -R foreman:foreman /custom/templates
# chcon -R -t httpd_sys_rw_content_t /custom/templates
```

- テンプレートをローカルリポジトリにエクスポートするには、以下のコマンドを実行します。

```
hammer export-templates --organization 'Default Organization' --repo /custom/templates
```

テンプレートをエクスポートする際に、バックエンドサービスが `systemd` プライベートの一時ディレクトリで実行されるため、`/tmp` や `/var/tmp` などの一時ディレクトリは回避してください。

### 13.3.3. Satellite API を使用したテンプレートの同期

#### 前提条件

- 各テンプレートに、テンプレートが属するロケーションおよび組織が含まれている必要がある。これは、すべてのタイプのテンプレートタイプに適用されます。テンプレートをインポートする前に、以下のセクションをテンプレートに追加します。

```
<%#
kind: provision
name: My_Provisioning_Template
oses:
- My_first_OS
- My_second_OS
locations:
- My_first_Location
- My_second_Location
organizations:
- My_first_Organization
- My_second_Organization
%>
```

#### 手順

- SSH 認証を使用するバージョン管理システムを設定します (gitosis、gitolite、git デーモンなど)。
- TemplateSync** タブで **TemplateSync** プラグイン設定を設定します。
  - Branch** 設定を変更して、Git サーバーへのターゲットブランチに一致します。
  - Git リポジトリに一致するように、**Repo** 設定を変更します。たとえば、`git@git.example.com/templates.git` に置いたリポジトリに対して、設定を `ssh://git@git.example.com/templates.git` に設定します。
- Git SSH ホストキーを **foreman** ユーザーとして受け取ります。

```
# sudo -u foreman ssh git.example.com
```

SSH 接続が成功していないため、出力に **Permission denied, please try again.** メッセージが表示されることが想定されます。

- SSH 鍵ペアがない場合は作成します。パスフレーズは指定しないでください。

```
# sudo -u foreman ssh-keygen
```

- Satellite の公開鍵を使用してバージョン管理サーバーを設定します。公開鍵は、`/usr/share/foreman/.ssh/id_rsa.pub` にあります。

- Satellite Server から、**TemplateSync** メニューに指定したバージョン管理リポジトリにテンプレートをエクスポートします。

```
$ curl -H "Accept:application/json" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/export \
-X POST

{"message":"Success"}
```

- コンテンツを変更したら、テンプレートを Satellite Server にインポートします。

```
$ curl -H "Accept:application/json" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/import \
-X POST

{"message":"Success"}
```

Satellite が提供するテンプレートがロックされ、デフォルトではインポートできない点にご留意ください。この動作を上書きするには、**TemplateSync** メニューの **Force import** 設定を **yes** に変更するか、**force** パラメーター **-d '{ "force": "true" }'** を **import** コマンドに追加します。

### 13.3.4. Satellite API を使用したローカルディレクトリーとテンプレートの同期

ローカルディレクトリーで、バージョン管理リポジトリを設定した場合は、テンプレートをローカルディレクトリーと同期すると便利です。これにより、テンプレートを編集し、ディレクトリーで編集履歴を追跡できます。テンプレートの編集後に変更を Satellite Server に同期することも可能です。

#### 前提条件

- 各テンプレートに、テンプレートが属するロケーションおよび組織が含まれている必要があります。これは、すべてのタイプのテンプレートタイプに適用されます。テンプレートをインポートする前に、以下のセクションをテンプレートに追加します。

```
<%#
kind: provision
name: My_Provisioning_Template
oses:
- My_first_OS
- My_second_OS
locations:
- My_first_Location
- My_second_Location
organizations:
- My_first_Organization
- My_second_Organization
%>
```

#### 手順

1. テンプレートを保存するディレクトリーを作成し、適切なパーミッションおよび SELinux コンテキストを適用します。

```
# mkdir -p /usr/share/templates_dir/
# chown foreman /usr/share/templates_dir/
# chcon -t httpd_sys_rw_content_t /usr/share/templates_dir/ -R
```

2. **TemplateSync** タブで **Repo** 設定を変更し、エクスポートディレクトリー **/usr/share/templates\_dir/** に一致させます。
3. Satellite Server からローカルディレクトリーにテンプレートをエクスポートします。

```
$ curl -H "Accept:application/json" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/export \
-X POST \

{"message":"Success"}
```

4. コンテンツを変更したら、テンプレートを Satellite Server にインポートします。

```
$ curl -H "Accept:application/json" \
-H "Content-Type:application/json" \
-u login:password \
-k https://_satellite.example.com/api/v2/templates/import \
-X POST

{"message":"Success"}
```

Satellite が提供するテンプレートがロックされ、デフォルトではインポートできない点にご留意ください。この動作を上書きするには、**TemplateSync** メニューの **Force import** 設定を **yes** に変更するか、**force** パラメーター **-d '{"force": "true"}'** を **import** コマンドに追加します。

## 注記

**-d** パラメーターを使用して、リクエストでデフォルトの API 設定を上書きします。以下の例では、**git.example.com/templates** リポジトリにテンプレートをエクスポートします。

```
$ curl -H "Accept:application/json" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/export \
-X POST \
-d '{"repo":"git.example.com/templates"}'
```

## 13.4. 高度な GIT 設定

コマンドラインで、または **.gitconfig** ファイルを編集して、**TemplateSync** プラグインに追加の Git 設定を実行できます。

### 自己署名の Git 証明書の同意

Git サーバーで自己署名証明書の認証を使用している場合は、**git config http.sslCAPath** コマンドでその証明書を検証します。たとえば、以下のコマンドを実行して **/cert/cert.pem** に保存されている自己署名証明書を確認します。

```
# sudo -u foreman git config --global http.sslCAPath cert/cert.pem
```

詳細なオプションの一覧は、**git-config** の man ページを参照します。

## 13.5. FOREMAN テンプレートプラグインのアンインストール

foreman\_templates プラグインを削除した後のエラーを回避するには、以下を実行します。

### 手順

1. Satellite インストーラーを使用するプラグインを無効にします。

```
# satellite-installer --no-enable-foreman-plugin-templates
```

2. プラグインのカスタムデータを削除します。このコマンドは、作成したテンプレートには影響しません。

```
# foreman-rake templates:cleanup
```

3. プラグインをアンインストールします。

```
# satellite-maintain packages remove foreman-plugin-templates
```



## 第14章 パッケージの管理

Satellite を使用して、ホストでパッケージをインストール、アップグレード、および削除できます。

### 14.1. ホストへのパッケージのインストール

この手順を使用して、Satellite Web UI を使用してパッケージを確認し、ホストにインストールします。インストール可能なパッケージのリストは、ホストに割り当てられたコンテンツビューおよびライフサイクル環境によって異なります。

#### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動し、パッケージをインストールするホストを選択します。
2. **Content** タブで、**Packages** タブを選択します。
3. アップグレードボタンの横にある縦の省略記号アイコンで、**Install Packages** をクリックします。
4. **Install packages** ウィンドウで、ホストにインストールするパッケージを選択します。
5. **Install** をクリックします。

デフォルトでは、パッケージはリモート実行を使用してインストールされます。リモート実行ジョブの詳細は、**ホストの管理** の [リモートジョブの設定とセットアップ](#) を参照してください。

### 14.2. ホスト上のパッケージのアップグレード

Satellite Web UI では、ホスト上のパッケージを一括でアップグレードできます。

#### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。
2. 変更するホストの名前をクリックします。
3. **Content** タブをクリックし、**Packages** タブをクリックします。
4. **Status** 一覧から **Upgradable** を選択します。
5. アップグレードするパッケージを選択します。
6. **Upgrade** ボタンをクリックします。リモート実行ジョブが完了すると、REX ジョブ通知が表示されます。

### 14.3. ホストからのパッケージの削除

Satellite Web UI で、ホストからパッケージを削除できます。

#### 手順

1. Satellite Web UI で、**Hosts > All Hosts** に移動します。

2. 変更するホストの名前をクリックします。
3. **Content** タブをクリックし、**Packages** タブをクリックします。
4. 削除するパッケージの縦リーダーをクリックして、**Remove** を選択します。リモート実行ジョブが完了すると、REX ジョブ通知が表示されます。

## 付録A テンプレート作成の参照

Embedded Ruby (ERB) は、Ruby コードとプレーンテキストを統合するテンプレートをもとに、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、以下の場合に ERB 構文を使用します。

### プロビジョニングテンプレート

詳細は、[ホストのプロビジョニング](#) の [プロビジョニングテンプレートの作成](#) を参照してください。

### リモート実行ジョブのテンプレート

詳細は、[11章 リモートジョブの設定およびセットアップ](#) を参照してください。

### レポートテンプレート

詳細は、[9章 レポートテンプレートを使用したホストの監視](#) を参照してください。

### パーティションテーブルのテンプレート

詳細は、[ホストのプロビジョニング](#) の [パーティションテーブルの作成](#) を参照してください。

### スマートクラスパラメーター

詳細は [Red Hat Satellite での Puppet 統合を使用した設定の管理](#) の [Puppet Smart Class パラメーターの設定](#) を参照してください。

このセクションでは、ERB テンプレートで使用可能な Satellite 固有のマクロと変数を使用例と併せて概説します。Red Hat Satellite が提供するデフォルトのテンプレート ([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](#)、[監視 > レポートテンプレート](#)) には、ERB 構文の適切な例も含まれている点にご留意ください。

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. SATELLITE WEB UI のテンプレート作成の参照へのアクセス

Satellite Web UI でテンプレート作成の参照ドキュメントにアクセスできます。

### 手順

1. Satellite Web UI にログインします。
2. Satellite Web UI で、[管理 > About](#) に移動します。
3. サポートセクションの [Templates DSL](#) リンクをクリックします。

## A.2. テンプレートでのオートコンプリートの使用

オートコンプリートオプションを使用して、テンプレートエディターで使用可能なマクロと使用法情報のリストにアクセスできます。これは、Satellite 内のすべてのテンプレートで機能します。

### 手順

1. Satellite Web UI で、[Hosts > Partition tables](#)、[Hosts > Provisioning templates](#)、または [Hosts > Job templates](#) のいずれかに移動します。

2. テンプレートエディターの右上隅にある **settings** アイコンをクリックし、**Autocompletion** を選択します。
3. テンプレートエディターで **Ctrl + Space** を押して、使用可能なすべてのマクロのリストにアクセスします。探しているものに関する詳細情報を入力して、マクロのリストを絞り込むことができます。たとえば、ホストのコンテンツソースの ID を一覧表示する方法を探している場合は、**host** と入力して、コンテンツソースで使用可能なマクロの一覧を確認できます。
4. ドロップダウンの横にあるウィンドウには、マクロの説明、その使用方法、および返される値が表示されます。
5. 探しているメソッドが見つかったら、**Enter** を押してメソッドを入力します。

**settings** メニューで **Live Autocompletion** を有効にして、何かを入力するたびにパターンに一致するマクロのリストを表示することもできます。ただし、これにより、プロビジョニングテンプレートのパッケージ名など、意図しない場所にマクロが入力される可能性があります。

### A.3. ERB テンプレートの作成

以下のタグは最も重要であり、ERB テンプレートで一般的に使用されています。

<% %>

すべての Ruby コードは、ERB テンプレートの <% %> 内に囲まれています。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 固有のマクロおよび変数を含めることができます。以下に例を示します。

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

このテンプレートは、サービスを使用して警告なしにアクションを実行し、出力には何も返さないことに注意してください。

<%= %>

これは、<% %> と同じ機能を提供しますが、テンプレートが実行されると、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。以下に例を示します。

入力例:

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

レンダリング例:

```
host.example.com
```

入力例:

```
<% server_name = @host.fqdn %>
<%= server_name %>
```

レンダリング例:

```
host.example.com
```

誤った変数を入力した場合、出力は返されない点にご留意ください。ただし、誤った変数でメソッドを呼び出そうとすると、以下のエラーメッセージが返されます。

入力例:

```
<%= @example_incorrect_variable.fqdn -%>
```

レンダリング例:

```
undefined method `fqdn' for nil:NilClass
```

```
<% -%>, <%= -%>
```

デフォルトでは、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。

入力例:

```
<%= "line1" %>
<%= "line2" %>
```

レンダリング例:

```
line1
line2
```

デフォルトの動作を変更するには、`-%>` で囲みマークを変更します。

入力例:

```
<%= "line1" -%>
<%= "line2" %>
```

レンダリング例:

```
line1line2
```

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)。ERB タグの空白は無視されます。

これをレポートテンプレートで使用して、FQDN と IP アドレス間の不要な改行を削除する方法の例を以下に示します。

入力例:

```
<%= @host.fqdn -%>
<%= @host.ip -%>
```

レンダリング例:

```
host.example.com10.10.181.216
```

```
<%# %>
```

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みます。

入力例:

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

これは出力を生成しません。

## ERB テンプレートのインデント

ERB タグの長さが異なるため、ERB 構文にインデントを入れると見にくい場合があります。ERB 構文は空白を無視します。インデントを処理する方法の1つは、新しい行の各行頭に ERB タグを宣言し、ERB タグ内の空白を使用して構文内の関係を説明することです。以下に例を示します。

```
<%- load_hosts.each do |host| -%>
<%-   if host.build? %>
<%=     host.name %> build is in progress
<%-   end %>
<%- end %>
```

## A.4. ERB テンプレートのトラブルシューティング

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

- **テンプレートエディターによる直接的な方法** – (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 配下の) テンプレートの編集時に、テンプレート タブで **プレビュー** をクリックしてから、一覧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れ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一覧から **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.5. 一般的な SATELLITE 固有のマクロ

このセクションでは、ERB テンプレート用の Satellite 固有のマクロを一覧表示します。以下の表に記載されているマクロは、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 一般的なマクロ

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。

名前	説明
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 <b>http://HOST/unattended/provision</b> 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 <b>unattended/snippets/</b> ディレクトリーからこれをロードします。
snippet_if_exists(name)	指定された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。

## A.6. テンプレートマクロ

カスタムテンプレートを作成する場合は、以下のマクロをいくつか使用できます。テンプレートのタイプに応じて、以下のマクロの一部には異なる要件があります。

レポートテンプレートで利用可能なマクロに関する詳細は、Satellite Web UI で、**監視 > レポートテンプレート** に移動し、**テンプレートの作成** をクリックします。テンプレートの作成ウィンドウで、**ヘルプ** タブをクリックします。

ジョブテンプレートで使用可能なマクロに関する詳細は、Satellite Web UI で、**ホスト > ジョブテンプレート** に移動し、**新しいジョブテンプレート** をクリックします。新しいジョブテンプレートウィンドウで、**ヘルプ** タブをクリックします。

### input

**input** マクロを使用すると、テンプレートで使用できる入力データをカスタマイズできます。ユーザーが使用できる入力名、タイプ、およびオプションを定義できます。レポートテンプレートの場合、ユーザー入力のみを使用できます。新しい入力を定義してテンプレートを保存すると、テンプレート本文の ERB 構文で入力を参照できます。

```
<%= input('cpus') %>
```

これは、ユーザー入力 **cpus** から値をロードします。

### load\_hosts

**load\_hosts** マクロを使用すると、ホストの完全なリストを生成できます。

```
<%- load_hosts().each_record do |host| -%>
<%= host.name %>
```

**load\_hosts** マクロを **each\_record** マクロと共に使用して、1000 件のレコードを一括でロードし、メモリー消費を減らします。

レポートのホスト一覧をフィルターリングする場合は、オプション **search: input('Example\_Host')** を追加できます。

```
<% load_hosts(search: input('Example_Host')).each_record do |host| -%>
  <%= host.name %>
<% end -%>
```

この例では、最初に入力を作成し、次にそれを使用して、**load\_hosts** マクロが取得する検索条件を絞り込みます。

## report\_row

**report\_row** マクロを使用すると、分析を容易にするためにフォーマットされたレポートを作成できます。**report\_row** マクロは、出力を生成するために **report\_render** マクロを必要とします。

入力例:

```
<%- load_hosts(search: input('Example_Host')).each_record do |host| -%>
<%- report_row(
  'Server FQDN': host.name
) -%>
<%- end -%>
<%= report_render -%>
```

レンダリング例:

```
Server FQDN
host1.example.com
host2.example.com
host3.example.com
host4.example.com
host5.example.com
host6.example.com
```

別のヘッダーを追加することで、レポートにコラムを追加できます。以下の例では、レポートに IP アドレスを追加します。

入力例:

```
<%- load_hosts(search: input('host')).each_record do |host| -%>
<%- report_row(
  'Server FQDN': host.name,
  'IP': host.ip
) -%>
<%- end -%>
<%= report_render -%>
```

レンダリング例:

```
Server FQDN,IP
host1.example.com,10.8.30.228
```



```
host2.example.com,10.8.30.227
host3.example.com,10.8.30.226
host4.example.com,10.8.30.225
host5.example.com,10.8.30.224
host6.example.com,10.8.30.223
```

## report\_render

このマクロは、レポートテンプレートでのみ使用できます。

**report\_render** マクロを使用して、レポートの出力を作成します。テンプレートのレンダリングプロセス中に、レポートに使用する形式を選択できます。YAML、JSON、HTML、および CSV 形式がサポートされています。

```
<%= report_render -%>
```

## render\_template()

このマクロは、ジョブテンプレートでのみ使用できます。

このマクロを使用して、特定のテンプレートをレンダリングできます。また、テンプレートに渡す引数を有効化して定義することもできます。

## truthy

**truthy** マクロを使用すると、値が整数またはブール値であるかに関係なく、渡された値が true か false かどうかを宣言できます。

このマクロは、テンプレートに複数の値タイプが含まれる場合に混乱を避けるのに役立ちます。たとえば、ブール値 **true** は文字列値 **"true"** とは異なります。このマクロを使用して、テンプレートで値を解釈する方法を宣言し、混乱を回避することができます。

**truthy** を使用して、以下のように値を宣言できます。

```
truthy?("true") => true
truthy?(1) => true
truthy?("false") => false
truthy?(0) => false
```

## falsy

**falsy** マクロは、真理マクロと同じ目的として機能します。

**falsy** マクロを使用すると、値が整数またはブール値であるかに関係なく、渡された値が true または false かどうかを宣言できます。

**falsy** を使用して、以下のように値を宣言できます。

```
falsy?("true") => false
falsy?(1) => false
falsy?("false") => true
falsy?(0) => true
```

## A.7. ホスト固有の変数

以下の変数により、テンプレート内でホストデータを使用できます。ジョブテンプレートは **@host** 変数のみを受け入れる点にご留意ください。

表A.2 ホスト固有の変数およびマクロ

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イスの配列を返します。「 <a href="#">配列の解析</a> 」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。
@host.domain	ホストのドメインです。
@host.environment <b>非推奨</b> 代わりに <b>host_puppet_environment</b> 変数を使用してください。	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	ホストのブートローダー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_enc['parameters']	ホストパラメーターの情報が含まれる Ruby ハッシュを返します。たとえば、host_enc['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 <b>true</b> を返します。
@host.interfaces	プライマリーインターフェイスを含む利用可能なすべてのホストインターフェイスの配列が含まれます。「 <a href="#">配列の解析</a> 」を参照してください。
@host.interfaces_with_identifier('IDs')	指定された ID を持つインターフェイスの配列を返します。複数の ID の配列を入力として渡すことができます (例: @host.interfaces_with_identifier(['eth0', 'eth1'])). 「 <a href="#">配列の解析</a> 」を参照してください。

名前	説明
@host.ip	ホストの IP アドレスです。
@host.location	ホストの位置です。
@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イスの配列を返します (BMC およびボンディングインターフェイスを除く)。 <a href="#">「配列の解析」</a> 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。
@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(medium_provider)	カーネルおよび initrd への完全パスで、配列を返します。
@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('parameter_name')	指定したホストパラメーターの値を返します。
host_param_false?('parameter_name')	指定したホストパラメーターが false と評価されると、 <b>false</b> を返します。
host_param_true?('parameter_name')	指定したホストパラメーターが true と評価されると、 <b>true</b> を返します。
@host.primary_interface	ホストのプライマリインスタンスを返します。
@host.provider	コンピュータ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イスを返します。インターフェイスオブジェクトを返します。

名前	説明
@host.ptable	パーティションテーブル名です。
@host.puppet_ca_server <b>非推奨</b> 代わりに <b>host_puppet_ca_server</b> 変数を使用してください。	ホストが使用すべき Puppet CA サーバーです。
@host.puppetmaster <b>非推奨</b> 代わりに <b>host_puppet_server</b> 変数を使用してください。	ホストが使用すべき Puppet サーバー。
@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 <b>true</b> を返します。
@host.shortname	ホストの省略名です。
@host.sp_ip	BMC インターフェイスの IP アドレスです。
@host.sp_mac	BMC インターフェイスの MAC アドレスです。
@host.sp_name	BMC インターフェイスの名前です。
@host.sp_subnet	BMC ネットワークのサブネットです。
@host.subnet.dhcp	DHCP プロキシがこのホストに設定されている場合は <b>true</b> 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、 <b>true</b> 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。

名前	説明
grub_pass	--md5pass=#{@host.grub_pass} などの暗号化されたブートローダーパスワードを設定するためのブートローダー引数を返します。
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。

一般的な Ruby メソッドのほとんどは、ホスト固有の変数に適用できます。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 %>
```

## A.8. キックスタート固有の変数

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.3 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 (テーブルの最初の行に #Dynamic オプションがある) の場合、 <b>true</b> 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。

## A.9. 条件付きステートメント

テンプレートでは、存在する値に応じてさまざまなアクションを実行できます。これを実現するには、ERB 構文で条件付きステートメントを使用できます。

以下の例では、ERB 構文は特定のホスト名を検索し、見つかった値に応じて出力を返します。

## 入力の例

```
<% load_hosts().each_record do |host| -%>
<% if @host.name == "host1.example.com" -%>
<%   result="positive" -%>
<% else -%>
<%   result="negative" -%>
<% end -%>
<%= result -%>
```

## レンダリング例

```
host1.example.com
positive
```

## A.10. アレイの解析

テンプレートを作成または変更する際、アレイを返す変数が出てくる場合があります。たとえば、**@host.interfaces** または **@host.bond\_interfaces** などのネットワークインターフェイ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イスデータを返します。特定のインターフェイ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。

### アレイを解析する正しい方法を見つける

以下の手順は、テンプレート内のアレイの解析方法として関連するものを見つけるために使用できる例です。この例では、レポートテンプレートが使用されていますが、この手順は他のテンプレートにも適用できます。

1. この例では、コンテンツホストの NIC を取得するために **@host.interfaces** 変数を使用すると、アレイを解析する方法を見つけるために使用できるクラス値が返されます。

#### 入力例:

```
<%= @host.interfaces -%>
```

#### レンダリング例:

```
<Nic::Base::ActiveRecord_Associations_CollectionProxy:0x00007f734036fbe0>
```

2. テンプレートの作成ウィンドウで、ヘルプ タブをクリックし、**ActiveRecord\_Associations\_CollectionProxy** クラスおよび **Nic::Base** クラスを検索します。
3. **ActiveRecord\_Associations\_CollectionProxy** の場合、許可された方法またはメンバー コラムで、以下のメソッドを表示してアレイを解析できます。

```
[] each find_in_batches first map size to_a
```

4. **Nic::Base** の場合、許可された方法またはメンバー コラムで、以下の方法を表示してアレイを解析できます。

```
alias? attached_devices attached_devices_identifiers attached_to bond_options
children_mac_addresses domain fqdn identifier inheriting_mac ip ip6 link mac managed?
mode mtu nic_delay physical? primary provision shortname subnet subnet6 tag virtual?
vlanid
```

5. インターフェイスアレイを繰り返すには、関連する方法を ERB 構文に追加します。

#### 入力例:

```
<% load_hosts().each_record do |host| -%>
<%   host.interfaces.each do |iface| -%>
  iface.alias?: <%= iface.alias? %>
  iface.attached_to: <%= iface.attached_to %>
  iface.bond_options: <%= iface.bond_options %>
  iface.children_mac_addresses: <%= iface.children_mac_addresses %>
  iface.domain: <%= iface.domain %>
  iface.fqdn: <%= iface.fqdn %>
  iface.identifier: <%= iface.identifier %>
  iface.inheriting_mac: <%= iface.inheriting_mac %>
  iface.ip: <%= iface.ip %>
  iface.ip6: <%= iface.ip6 %>
  iface.link: <%= iface.link %>
  iface.mac: <%= iface.mac %>
  iface.managed?: <%= iface.managed? %>
  iface.mode: <%= iface.mode %>
  iface.mtu: <%= iface.mtu %>
  iface.physical?: <%= iface.physical? %>
  iface.primary: <%= iface.primary %>
  iface.provision: <%= iface.provision %>
  iface.shortname: <%= iface.shortname %>
  iface.subnet: <%= iface.subnet %>
  iface.subnet6: <%= iface.subnet6 %>
  iface.tag: <%= iface.tag %>
  iface.virtual?: <%= iface.virtual? %>
  iface.vlanid: <%= iface.vlanid %>
<%- end -%>
```

#### レンダリング例:

```
host1.example.com
iface.alias?: false
iface.attached_to:
iface.bond_options:
iface.children_mac_addresses: []
iface.domain:
iface.fqdn: host1.example.com
iface.identifier: ens192
iface.inheriting_mac: 00:50:56:8d:4c:cf
iface.ip: 10.10.181.13
iface.ip6:
iface.link: true
iface.mac: 00:50:56:8d:4c:cf
iface.managed?: true
iface.mode: balance-rr
iface.mtu:
```

```

iface.physical?: true
iface.primary: true
iface.provision: true
iface.shortname: host1.example.com
iface.subnet:
iface.subnet6:
iface.tag:
iface.virtual?: false
iface.vlanid:

```

## A.11. テンプレートスニペットの例

### ホストで Puppet および Puppetlabs が有効化されているかどうかの確認

以下の例では、ホストで Puppet および Puppetlabs リポジトリが有効化されているかどうかを確認します。

```

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

```

### ホストのオペレーティングシステムのメジャーバージョンとマイナーバージョンの取得

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 -%>

```

### テンプレートへのスニペットのインポート

以下の例は、**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 %>

```

### キックスタートスニペットの条件付きインポート

以下の例では、ホストのサブネットが 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 -%>

```



## ■ ホストのカスタムファクトからの値の解析

**host.facts** 変数を使用して、ホストファクトとカスタムファクトの値を解析できます。この例では、**luks\_stat** は、ホストファクトである **dmi::system::serial\_number** と同じ方法で解析できるカスタムファクトです。

```
'Serial': host.facts['dmi::system::serial_number'],  
'Encrypted': host.facts['luks_stat'],
```

この例では、適用可能なエラータレポートテンプレートをカスタマイズして、各ホストのカーネルバージョンに関するカスタム情報を解析できます。

```
<%-   report_row(  
      'Host': host.name,  
      'Operating System': host.operatingsystem,  
      'Kernel': host.facts['uname::release'],  
      'Environment': host.lifecycle_environment,  
      'Erratum': erratum.errata_id,  
      'Type': erratum.errata_type,  
      'Published': erratum.issued,  
      'Applicable since': erratum.created_at,  
      'Severity': erratum.severity,  
      'Packages': erratum.package_names,  
      'CVEs': erratum.cves,  
      'Reboot suggested': erratum.reboot_suggested,  
    ) -%>
```

## 付録B ジョブテンプレートの例および拡張

このセクションは、要件に合わせたジョブテンプレートの修正、カスタマイズ、および拡張に役立つ参照情報として使用できます。

### B.1. ジョブテンプレートのカスタマイズ

ジョブテンプレートの作成時に、テンプレートエディターフィールドで既存のテンプレートを追加できます。こうすることで、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。

次のテンプレートは、クライアントに **nginx** サービスをインストールして開始するためのデフォルトのテンプレートを組み合わせたものです。

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

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** タブに移動して **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** 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。

### B.2. 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、および削除アクションが含まれています。
Puppet	ターゲットホストで Puppet ホスト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、およびステータスアクションが含まれます。

ジョブテンプレートのカテゴリー	説明
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 (たとえば、コンテンツホストの一括操作のための UI など) で使用されますが、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。

### B.3. RESTORECON テンプレートの例

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。

#### 手順

1. Satellite Web UI で、**ホスト > ジョブテンプレート** に移動します。**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、**ジョブカテゴリー** を **Commands** に設定します。
4. **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。
5. **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。
6. **入力タイプ** リストから **ユーザー入力** を選択します。ジョブの呼び出し中に表示する説明を入力します (例: **Target directory for restorecon**)。
7. **Submit** をクリックします。詳細は、**ホストの管理** の [複数のホストでの restorecon テンプレートの実行](#) を参照してください。

### B.4. RESTORECON テンプレートのレンダリング

この例は、[restorecon テンプレートの例](#) 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。このテンプレートはジョブの実行時にユーザー入力を必要としません。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に SELinux コンテキストが復元されます。

[ジョブテンプレートの設定](#) に従って新しい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") %>
```

## B.5. 複数のホストでの RESTORECON テンプレートの実行

以下の例では、[restorecon テンプレートの例](#) 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/** ディレクトリー内にある全ファイルの SELinux コンテキストを復元します。

### 手順

1. Satellite Web UI で、**Hosts > All hosts** に移動して、ターゲットホストを選択します。**アクションの選択** リストで、**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** ジョブカテゴリを選択し、**Run Command - restorecon** ジョブテンプレートを選択します。
3. **ディレクトリー** フィールドに **/home** と入力します。
4. **スケジュール** を **Execute now** に設定します。
5. **Submit** 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。

## B.6. 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の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設定](#) に従って新しい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") %>
```