Red Hat Satellite 6.6

ホストの管理

Red Hat Satellite 6 環境におけるホストの管理ガイド

Last Updated: 2020-04-27

Red Hat Satellite 6.6 ホストの管理

Red Hat Satellite 6 環境におけるホストの管理ガイド

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

法律上の通知

概要

本書は、Red Hat Satellite 環境でホストを設定して使用する方法を説明します。この作業を行う前に、Red Hat Satellite 6 Server と、必要な 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

第1章 RED HAT SATELLITE 6 ホストの概要

ホストは、Red Hat Satellite が管理する Red Hat Enterprise Linux クライアントを指します。ホストは、物理システムまたは仮想システムのいずれかに設定でき、KVM、VMware vSphere、OpenStack、Amazon EC2、Rackspace Cloud Services、Google Compute Engine などの仮想ホストは、Red Hat Satellite がサポートするプラットフォームにデプロイできます。

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ます。また、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、自己登録ではなくても、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。

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. **インターフェース** タブで、以下を行います。
    a. 各インターフェースに対して、**アクション** コラムで **編集** をクリックし、必要に応じて以下を設定します。
        - **タイプ**: ボンドまたは BMC インターフェースに対して、**タイプ** リストで、インターフェースタイプを選択します。
        - **MAC アドレス**: MAC アドレスを入力します。
        - **DNS 名**: DNS サーバーに認識させる DNS 名を入力します。これは、完全修飾ドメイン名 (FQDN) のホスト部分に使用されます。
        - **ドメイン**: プロビジョニングネットワークのドメイン名を選択します。これにより、**サブネット** 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。アドレスを入力することもできます。トークンのプロビジョニングが有効な場合、ドメインが DNS を管理しない場合、サブネットが逆引き DNS を管理しない場合、またはサブネットが DHCP 予約を管理しない場合は、このアドレスを省略できます。
        - **IPv6 アドレス**: サブネットに対して IP アドレス管理 (IPAM) を有効にした場合は、IP アドレスが自動的に提案されます。アドレスを入力することもできます。
        - **管理**: このチェックボックスを選択すると、Capsule が提供する DHCP サービスおよび DNS サービスを使用してプロビジョニングを行う際にインターフェースを設定します。
        - **プライマリー**: このチェックボックスを選択すると、このインターフェースの DNS 名を、FQDN のホスト部分に使用します。

- **プロビジョニング**: このチェックボックスを選択すると、プロビジョニングにこのインターフェースを使用します。つまり、このインターフェースを使用して TFTP ブートが行われ、そしてイメージをベースにしたプロビジョニングでは、プロビジョニングを実行するスクリプトにこのインターフェースが使用されます。**anaconda** による RPM のダウンロードや、**%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ースを使用する点にご留意ください。

- **仮想 NIC**: このインターフェースが物理デバイスではない場合は、このチェックボックスを選択します。この設定にはオプションが 2 つあります。

  - **タグ**: 任意で VLAN タグを設定します。設定していない場合はサブネットの VLAN ID となります。

  - **割り当て先**: この仮想インターフェースが割り当てられるインターフェースのデバイス名を入力します。

b. **OK** をクリックして、インターフェース設定を保存します。

c. オプションとして、**インターフェースの追加** をクリックし、追加ネットワークインターフェースを組み込みます。詳細は*4章 ネットワークインターフェースの追加*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、ホストの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。システム目的の詳細は、『**標準的な RHEL インストールの実行**』の「システム目的の設定」を参照してください。

   - **syspurpose_role**

   - **syspurpose_sla**

   - **syspurpose_usage**

   - **syspurpose_addons**

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 をご利用の場合

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 "host_name" \
--hostgroup "hostgroup_name" \
--interface="primary=true, \
          provision=true, \
          mac=mac_address, \
          ip=ip_address" \
--organization "Your_Organization" \
--location "Your_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ースの他のプロパティー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ースを設定できます。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ー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. ホストのシステム目的の編集

Red Hat Enterprise Linux 8 ホストのシステム目的属性を設定できます。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アタッチするかを定義します。システム目的の詳細は、『**標準的な RHEL インストールの実行**』の「システム目的」を参照してください。

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 ホストの名前をクリックします。
2. **システム目的** エリアで、編集、追加または削除するシステム目的属性の **編集** または **削除** アイコンをクリックします。
3. **保存**をクリックします。
4. **サブスクリプション** タブをクリックして、**サブスクリプション** を選択します。
5. **自動アタッチの実行** をクリックして、自動的にホストにサブスクリプションを割り当てます。
6. ページを更新して、サブスクリプションリストに正しいサブスクリプションが含まれていることを確認します。

CLI をご利用の場合

1. 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、**addon** アドオンを追加します。値の一覧については、『**標準的な RHEL インストールの実行**』の「システム目的」を参照してください。

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

2. このホストのシステム目的属性を検証します。

```
# syspurpose show
```

3. 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
```

4. 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
```

2.3. ホストのモジュールストリーム変更

Red Hat Enterprise Linux 8 ホストを使用する場合には、インストールするリポジトリーのモジュールストリームを変更できます。たとえば、ホストの作成後に、Satellite Web UI を使用し、ホストからのモジュールストリームの有効化、無効化、インストール、更新、削除が可能になります。

手順

1. Satellite Web UI で、**ホスト** > **コンテンツホスト** に移動し、変更するモジュールを含むホストの名前をクリックします。
2. **モジュールストリーム** タブをクリックします。
3. **利用可能なモジュールストリーム** リストから、変更するモジュールの場所を特定します。**フィルター** フィールドを使用して、リストエントリーを絞り込みます。または、**フィルターステータス** リストを使用して、特定のステータスのモジュールを検索できます。
4. **アクション** リストから、モジュールに加える変更を選択します。
5. **ジョブ呼び出し** ウィンドウで、ジョブの情報が適切であることを確認します。必要に応じて詳細を変更し、**送信** をクリックします。

2.4. 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ます。これには、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。

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** (**httpd** 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、**httpd** 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て、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て、所有者を変更します。

```
# mkdir /etc/puppetlabs/code/environments/example_environment
# chown apache /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 をご利用の場合

hammer hostgroup create コマンドでホストグループを作成します。以下は例になります。

```
# hammer hostgroup create --name "Base" \
--lifecycle-environment "Production" --content-view "Base" \
--puppet-environment "production" --content-source-id 1 \
--puppet-ca-proxy-id 1 --puppet-proxy-id 1 --domain "example.com" \
```

```
--subnet `ACME's Internal Network` --architecture "x86_64" \
--operatingsystem "RedHat 7.2" --medium-id 9 \
--partition-table "Kickstart default" --root-pass "p@55w0rd!" \
--locations "New York" --organizations "ACME"
```

2.5. 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="7"
ARCH="x86_64"
ORG="Your Organization"
LOCATIONS="Your Location"
PTABLE_NAME="Kickstart default"
DOMAIN="example.com"

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
```

2.6. ホストのグループの変更

以下の手順を使用して、ホストのグループを変更します。

手順

1. Satellite Web UI で、**ホスト** > **すべてのホスト** に移動します。
2. 変更する必要の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 **グループの変更** を選択すると、新規オプションのウィンドウが開きます。
4. **ホストグループ** リストから、ホストに必要なグループを選択します。
5. **送信** をクリックします。

2.7. 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. 変更する必要の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 **環境の変更** を選択すると、新規オプションのウィンドウが開きます。
4. **環境** リストから、ホストの新しい環境を選択します。
5. **送信** をクリックします。

2.8. 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には、ホストを非管理対象に設定します。

以下の手順を使用して、ホストの管理対象および非管理対象のステータスを切り替えます。

手順

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

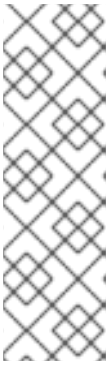
2.9. ホストの特定組織への割り当て

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

手順

1. Satellite Web UI で、**ホスト** > **すべてのホスト** に移動します。
2. 変更する必要のあるホストのチェックボックスを選択します。
3. 画面右上の **アクションの選択** 一覧から **組織の割り当て** を選択すると、新規オプションの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 (組織の不一致についての修正)** チェックボックスを選択します。



注記

ドメイン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. **送信** をクリックします。

2.10. 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、特定のロケーションにホストを割り当てます。ロケーションの一般的な情報および設定方法は、『**プロビジョニングガイド**』の「ロケーションの作成」を参照してください。

手順

1. Satellite Web UI で、**ホスト** > **すべてのホスト** に移動します。
2. 変更する必要のあるホストのチェックボックスを選択します。
3. 画面右上の **アクションの選択** 一覧から **ロケーションの割り当て** を選択すると、新規オプションの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 (ロケーションの不一致についての修正)** チェックボックスを選択します。

注記

ドメイン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. **送信** をクリックします。

2.11. RED HAT 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** に移動します。
2. 削除するホストを選択します。
3. **アクションの選択** 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

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

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

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. **送信** をクリックします。

第3章 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 2 つあります。

- コンシューマー RPM (**server.example.com**/pub/katello-ca-consumer-latest.noarch.rpm) をダウンロードしてインストールしてから、サブスクリプションマネージャーを実行します。この方法は、新規インストールしたホストに適しています。
- ブートストラップスクリプト (**server.example.com**/pub/bootstrap.py) をダウンロードして実行します。この方法は、新規インストールしたホストにも、Satellite 5 や別の Satellite 6 に登録済みのホストにも適しています。

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、クライアントを登録します。

- 「Red Hat Satellite へのホストの登録」
- 「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」

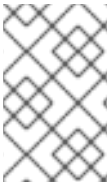
以下の手順を使用して、ホストツールをインストールして設定します。

- 「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 エージェントのインストールおよび設定」

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している必要があります。

- 5.7 以降
- 6.1 以降*
- 7.0 以上
- 8.0 以降



注記

Red Hat Enterprise Linux バージョン 6.1、6.2 および 6.3 では、**subscription-manager** と関連のパッケージを手動で更新する必要があります。詳細は、「https://access.redhat.com/solutions/1256763」を参照してください。

サポートされるアーキテクチャー

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

- i386
- x86_64
- s390x

- ppc_64

3.1. RED HAT SATELLITE へのホストの登録

以下の手順を使用して、ホストを Red Hat Satellite 6 に登録します。

前提条件

- Satellite Server、Capsule Server、およびすべてのホストは、同じ NTP サーバーと同期して時間同期ツールを有効にし、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、『**コンテンツ管理ガイド**』の「アクティベーションキーの管理」を参照してください。
- バージョンが 1.10 以降のサブスクリプションマネージャーを使用している。パッケージは標準の Red Hat Enterprise Linux リポジトリーで利用できます。

手順

Red Hat Enterprise Linux ホストは、デフォルトで Red Hat コンテンツ配信ネットワーク (CDN) に登録されます。

各ホスト設定を更新して、正しい Satellite Server または Capsule Server から更新を受け取るようにします。

1. Satellite Server または Capsule Server の完全修飾ドメイン名 (FQDN) をメモしておきます (例: **server.example.com**)。
2. **root** ユーザーとしてホストにログインして、ホストを登録する Satellite Server または Capsule Server から **katello-ca-consumer-latest.noarch.rpm** パッケージをダウンロードします。コンシューマー RPM はホストを設定して、Red Hat Satellite で指定されたコンテンツソースからコンテンツをダウンロードします。

```
# curl --insecure --output katello-ca-consumer-latest.noarch.rpm
https://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

3. **katello-ca-consumer-latest.noarch.rpm** パッケージをインストールします。

```
# yum localinstall katello-ca-consumer-latest.noarch.rpm
```

注記

RPM パッケージは署名されていません。必要に応じて、**--nosignature** オプションを使用してパッケージをインストールします。**katello-ca-consumer-hostname-1.0-1.noarch.rpm** パッケージは、追加の **katello-ca-consumer** RPM で、この中にサーバーのホスト名が含まれます。**katello-ca-consumer-latest.noarch.rpm** パッケージは常に最新のバージョンを反映します。どちらを使用しても達成できる目的は同じです。

4. Red Hat Subscription Manager (RHSM) に関連するすべての以前のホストデータを消去します。

```
# subscription-manager clean
```

5. RHSM を使用してホストを登録します。

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

例3.1 登録後のコマンド出力:

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

注記

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

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

注記

Red Hat Enterprise Linux 6.3 ホストの場合には、リリースバージョンが Red Hat Enterprise Linux 6 Server にデフォルト設定されており、6.3 リポジトリーを指定する必要があります。

1. Satellite Web UI で、**ホスト** > **コンテンツホスト** に移動します。
2. 変更が必要なホストの横にある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 **リリースバージョンの設定** を選択します。
4. **リリースバージョン** リストから **6.3** を選択します。
5. **完了** をクリックします。

3.2. 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 6 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
```

Red Hat サブスクリプションマネージャーで登録します。

```
# 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。

3.3. 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

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。新しいホストの登録や、既存のホストの Satellite 5、RHN、SAM または RHSM からの Red Hat Satellite 6 への移行には、ブートストラップスクリプトを使用できます。

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、**satellite6.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。セキュリティーポリシーの関係上、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、このロールをスクリプトを実行するユーザーに追加してください。詳細情報は、「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 Tools リポジトリー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『**コンテンツ管理ガイド**』の「アクティベーションキーの管理」を参照してください。
- ホストグループを作成済みである。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. 手動でディレクトリーを作成して、所有者を変更します。

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

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

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://satellite6.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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key
```

   - Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

```
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=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 **インフラストラクチャー** > **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。

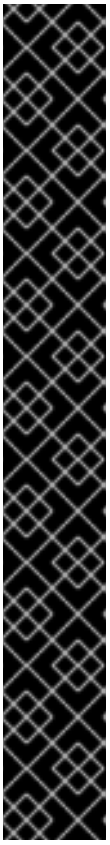
```
# rm bootstrap.py
```

3.3.1. 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指定します。

手順

1. Satellite Web UI で、**管理** > **ユーザー** 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するためだけに新しい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。セキュリティーポリシーの関係上、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. **送信** をクリックします。

CLI をご利用の場合

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方法は『**Hammer CLI ガイド**』の「ユーザーの作成」を参照してください。

3.3.2. 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。セキュリティーポリシーの関係上、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。

Satellite 6 から別の Satellite 6 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。

- 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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--force
```

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

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

ブートストラップスクリプトは、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の理由上、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。レガシーの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。

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

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

Satellite 6 にホストを登録して Puppet 設定を省略する手順

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。

- 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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--skip-puppet
```

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

コンテンツ管理専用としてホストを Satellite 6 に登録する手順

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。

- 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 satellite6.example.com \
```

```
--organization="Example Organization" \
--activationkey=activation_key \
--skip-foreman
```

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

ブートストラップスクリプトがコンシューマー RPM のダウンロードに使用する方法の変更

デフォルトでは、ブートストラップスクリプトは HTTP を使用してコンシューマー RPM (**server.example.com**/pub/katello-ca-consumer-latest.noarch.rpm) をダウンロードします。環境によっては、ホストと Satellite との間のみ HTTPS を許可する場合があります。**--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。

- 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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--download-method https
```

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

ホストの IP アドレスの Satellite への指定

インターフェースが複数あるホスト、または 1 つのインターフェースに 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。

- 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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--ip 192.x.x.x
```

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

ホストで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。

- 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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--rex \
--rex-user root
```

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

登録時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。

- 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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--add-domain
```

- Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

ホストへの別の 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。

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 satellite6.example.com \
--location="Example Location" \
--organization="Example Organization" \
--hostgroup="Example Host Group" \
--activationkey=activation_key \
--fqdn node100.example.com
```

   - Red Hat Enterprise Linux 5、6、7 の場合は、以下のコマンドを入力します。

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

3.4. 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する必要があります。

katello-agent パッケージは、**goferd** サービスを提供する **gofer** パッケージに依存します。Satellite Server または Capsule Server が、コンテンツホストに適用可能なエラータの情報を提供できるようにするには、このサービスを有効化する必要があります。

前提条件

Katello エージェントのインストール前に、以下の条件が満たされていることを確認してください。

- Satellite Server で、Satellite Tools リポジトリーを有効化しておく。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 Tools リポジトリーのインストール」を参照してください。
- Satellite Server で Satellite Tools リポジトリーを同期しておく。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 Tools リポジトリーの同期」を参照してください。
- クライアントで Satellite Tools リポジトリーを有効化しておく。たとえば、Red Hat Enterprise Linux 7 クライアントでリポジトリーが有効化されているかを確認するには、以下のコマンドをクライアントで実行してください:

```
# subscription-manager repos --enable rhel-7-server-satellite-tools-6.6-rpms
```

手順

Katello エージェントをインストールするには、以下の手順を実行します。

1. **libvirt-client** パッケージをインストールします。

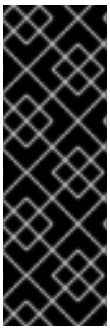
```
# yum install katello-agent
```

2. **goferd** サービスを開始します。

```
# systemctl start goferd
```

3.5. トレーサーのインストール

以下の手順を使用して、Red Hat Satellite 6.6 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、反対にトレースは、Satellite Web UI でトレーサーが生成する出力です。



重要

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

前提条件

- ホストが Red Hat Satellite Server に登録されている。

- Red Hat Satellite Tools 6.6 リポジトリーを Satellite Server で有効化して同期し、ホストで有効化しておく。

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. 以下のコマンドを実行します。

```
# 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. **プロパティー** タブの **プロパティー** のテーブルで、トレースの項目を確認します。**プロパティー** のテーブルでトレース項目が見つからない場合は、トレースがインストールされていません。
5. **ホスト** > **コンテンツホスト** に移動してから、必要なホスト名をクリックします。
6. **トレース** タブをクリックして、トレースを表示します。

3.6. PUPPET エージェントのインストールおよび設定

以下の手順を使用して、ホストに Puppet エージェントをインストールして設定します。Puppet の情報は、『Puppet ガイド』を参照してください。

前提条件

- ホストが Red Hat Satellite Server に登録されている。
- ホストに Puppet 環境が割り当てられている。
- Red Hat Satellite Tools 6.6 リポジトリーを Satellite Server で有効化して同期し、ホストで有効化しておく。

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。

```
# yum install puppet-agent
```

3. 起動時に Puppet エージェントが起動するように設定します。
   - Red Hat Enterprise Linux 6 の場合:

```
# chkconfig puppet on
```

   - Red Hat Enterprise Linux 7 の場合:

```
# systemctl enable puppet
```

4. 以下のサーバーと環境の設定を **/etc/puppetlabs/puppet/puppet.conf** ファイルに追加します。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。

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

5. ホスト上で Puppet エージェントを実行します。

```
# puppet agent -t
```

6. Satellite Web UI で、**インフラストラクチャー** > **Capsules (スマートプロキシー)** に移動します。

7. 必要な Capsule Server の **アクション** コラムの一覧から、**証明書** を選択します。

8. 必要なホストの右にある **署名** をクリックして、Puppet クライアントの SSL 証明書に署名します。

9. **puppet agent** コマンドを再入力します。

```
# puppet agent -t
```

第4章 ネットワークインターフェースの追加

Red Hat Satellite は、1 台のホストに対して複数のネットワークインターフェースを指定することをサポートします。「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ースを設定することができます。

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

- **インターフェース**: 物理インターフェースまたは仮想インターフェースを追加で指定できます。作成できる仮想インターフェースのタイプは 2 つあります。ホストが 1 つのインターフェースを使用して複数の (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ースに別の IP アドレスを追加するには、**alias** を使用します。
  物理インターフェースの追加に関する情報は、「物理インターフェースの追加」を参照してください。

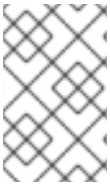
  仮想インターフェースの追加に関する情報は、「仮想インターフェースの追加」を参照してください。

- **ボンド**: ボンディングインターフェースを作成します。NIC ボンドは複数のネットワークインターフェースを 1 つのインターフェースにバインディングして 1 つのデバイスと表示し、MAC アドレスを 1 つ持つ方法です。これにより、複数のネットワークインターフェースが 1 つのインターフェースとして機能し、帯域幅の拡大と冗長性を提供します。詳細は「ボンディングインターフェースの追加」を参照してください。

- **BMC**: ベースボード管理コントローラー (BMC) により、マシンの物理的な状態をリモートで監視し、管理できます。BMC の詳細は、『**オンラインネットワークからの Satellite Server のインストール**』の「管理対象ホスト上での電源管理の有効化」を参照してください。BMC インターフェー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) インターフェースの追加」 を参照してください。

注記

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



注記

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

4.1. 物理インターフェースの追加

この手順を使用して、別の物理インターフェー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. **インターフェース** タブで、**インターフェースの追加** をクリックします。
4. **タイプ** リストで、**インターフェース** オプションが選択されている状態にします。
5. **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ース、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ースを指定します。
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. インターフェースの **IP アドレス** を指定します。DHCP Capsule Server が割り当てられた管理対象インターフェースでは、DHCP リースを作成するためにこの設定が必要です。DHCP が有効になっている管理インターフェースでは、IP アドレスが自動補完されます。
11. インターフェースが **管理対象** かどうかを決定します。インターフェー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ース用に自動的に作成されます。
12. ホストの **プライマリー** インターフェースかどうかを選択します。プライマリーインターフェースからの 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** インターフェー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ースを使用します。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ー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** のインターフェー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、インターフェー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. 仮想インターフェースの追加

以下の手順を使用して、ホストの仮想インターフェースを設定します。仮想インターフェースには、VLAN またはエイリアスインターフェースのいずれかを使用することができます。

エイリアスインターフェースとは、既存のインターフェー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ースは、自動的にアタッチ先のインターフェースから 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。インターフェースは、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. **インターフェース** タブで、**インターフェースの追加** をクリックします。
4. **タイプ** リストで、**インターフェース** オプションが選択されている状態にします。
5. 一般的なインターフェース設定を指定します。適用できる設定オプションは、物理インターフェースのオプションと同じです (「物理インターフェースの追加」を参照)。
管理対象の仮想インターフェースの **MAC アドレス** を指定し、プロビジョニング用の設定ファイルが適切に生成されるようにします。ただし、**MAC アドレス** は、管理対象外の仮想インターフェースには不要です。

   VLAN を作成する場合、**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。仮想インターフェースに固有の追加設定オプションがその形式に追加されます。
   - **タグ**: 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ー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、管理インターフェースは、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ースには適用されません。
   - **割り当て先**: **eth1** など、仮想インターフェースの所属先となる物理インターフェースの 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、インターフェー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ースの追加

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ースを設定します。

手順

1. Satellite Web UI で、**ホスト** > **すべてのホスト** に移動します。
2. 編集するホストの横の **編集** をクリックします。
3. **インターフェース** タブで、**インターフェースの追加** をクリックします。
4. **タイプ** 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ース設定を指定します。適用できる設定オプションは、物理インターフェースのオプションと同じです (「物理インターフェースの追加」を参照)。
ボンディングインターフェースは、**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ー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、表4.1「Red Hat Satellite で利用可能なボンディングモード」を参照してください。
- **割り当て済みデバイス**: 割り当てられたデバイスの ID のコンマ区切りの一覧を指定します。物理インターフェースまたは VLAN を指定できます。
- **ボンドオプション**: 設定オプションのコンマ区切りの一覧を指定します (例: **miimon=100**)。詳細は『Red Hat Enterprise Linux 7 ネットワークガイド』を参照してください。

7. **OK** をクリックして、インターフェース設定を保存します。
8. **送信** をクリックして、ホストへの変更を適用します。

CLI をご利用の場合

ボンディングインターフェー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 "Your_Organization" \
--location "Your_Location" \
--ask-root-password yes
```

表4.1 Red Hat Satellite で利用可能なボンディングモード

ボンディングモード	説明
balance-rr	送受信は、ボンディングインターフェースで順次行われます。

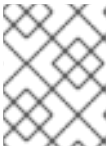
ボンディングモード	説明
active-backup	ボンディングインターフェースの中で最初に利用可能になったものから送受信が行われます。アクティブなボンディングインターフェースに障害がある場合に限り別のボンディングインターフェースが使用されます。
balance-xor	送信は選択されたハッシュポリシーに基づいて行われます。このモードでは、特定のピア用に宛先が指定されたトラフィックは常に同じインターフェースで送信されます。
broadcast	すべての送信はすべてのボンディングインターフェースで行われます。
802.a3	同じ設定を共有するアグリゲーショングループを作成します。アクティブなグ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

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

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) インターフェースを、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 インターフェースのその他の詳細、およびこのインターフェースの適切な認証情報を確認している。



注記

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

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て、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、**インフラストラクチャー** > **Capsules (スマートプロキシー)** 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. **インターフェース** タブで、**インターフェースの追加** 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ース設定を指定します。適用できる設定オプションは、物理インターフェースのオプションと同じです (「物理インターフェースの追加」を参照)。
7. BMC インターフェー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、インターフェー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 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。

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

Red Hat Insights を使用すると、セキュリティー違反、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、セキュリティー、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。

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。

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については、「Ansible ロールの管理」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、*8章Ansible ロールの使用* 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。

ロール実行が完了したら、Satellite Web UI の **Insights** > **概要** ページで追加したホストの表示と作業が可能になります。

追加情報

- Red Hat Insights プラグインにシステム更新を適用するには、更新後に **httpd restart** を入力します。
- 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。

5.2. ホストの INSIGHTS プランの作成

Satellite 6 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

プランを作成するには、以下の手順を行います。

1. Satellite Web UI で **Insights** > **インベントリー** に移動して、Insights プランに追加するホストを選択します。
2. **アクション** リストから **新規プラン/Playbook の作成** を選択します。
3. プラン/Playbook ビルダーウィンドウから、**新規プランの作成** を選択して、プランの名前を入力します。
4. 特定のシステム、グループまたは全システムに、ルールを適用するかどうかを選択します。
5. プランに追加するルールを 1 つまたは複数選択します。**フィルター** フィールドを使用して、特定のキーワードを検索します。
6. **保存**をクリックします。
7. プランナーウィンドウで、**Playbook の実行** を選択します。

Playbook を実行しながら、ジョブウィンドウでプランの進捗を表示できます。

Insights > **プランナー** に移動して、Insights プランを表示できます。

第6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ー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。

6.1. 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. **送信** をクリックします。レポートが含まれれる 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には、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 をご利用の場合

レポートを生成するには、次の手順を実行します。

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 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。

6.2. 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、付録A *テンプレート作成の参照* を参照してください。

レポートテンプレートで使用できるマクロの詳細は、「テンプレートマクロ」 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。

6.3. 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 をご利用の場合

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 template_ID > example_export.erb
```

6.4. 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 Guide』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。

Example request:

```
$ 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。

Example response:


```
{
  "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。

Example request:

```
$ curl --insecure --output /tmp/_Example_Export_Template.erb_ \
--user admin:password --request GET --config \
https://satellite.example.com/api/report_templates/158/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。

6.5. 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。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、「レポートテンプレート」ウィンドウの右上で **テンプレートの作成** をクリックします。
2. **Template** エリアで、**Import** をクリックしてインポートする **.erb** ファイルを選択します。
3. 要件に合わせてテンプレートを編集し、**送信** をクリックします。

新しい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。

6.6. 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 Guide』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
"
}
```

Example JSON File with ERB Template:

```
{
   "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
```

6.7. 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使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7章 ホストコレクションの設定

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

7.1. ホストコレクションの作成

以下の手順では、ホストコレクションから作成する方法を示します。

ホストコレクションの作成方法:

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

CLI をご利用の場合

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 \
--organization "Your_Organization" \
--name hc_name
```

7.2. ホストコレクションのクローン作成

以下の手順では、ホストコレクションのクローンを作成する方法を示します。

ホストコレクションのクローン作成方法:

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

7.3. ホストコレクションの削除

以下の手順では、ホストコレクションを削除する方法を示します。

ホストコレクションの削除方法:

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

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

4. **削除** をクリックします。

7.4. ホストコレクションへのホストの追加

以下の手順では、ホストをホストコレクションに追加する方法を示します。

前提条件

ホスト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、3章*ホストの登録* を参照してください。

ホストコレクションへのホストの追加方法:

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

CLI をご利用の場合

ホストコレクションにホストを追加するには、以下のコマンドを実行します。

```
# hammer host-collection add-host \
--id hc_ID \
--host-ids host_ID1,host_ID2...
```

7.5. ホストコレクションからのホストの削除

以下の手順では、ホストをホストコレクションから削除する方法を示します。

ホストコレクションからのホストの削除方法:

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

7.6. 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。

7.6.1. 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ーの 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。

ホストコレクションへのパッケージの追加方法:

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

7.6.2. エラータのホストコレクションへの追加

以下の手順では、エラータをホストコレクションに追加する方法を示します。

前提条件

- 追加するエラータは既存リポジトリーのいずれかで利用可能であるか、またはリポジトリーにない場合は、この手順を開始する前にリポジトリーに追加しておく必要がある。
- 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。

ホストコレクションへのエラータの追加方法:

1. **ホスト** > **ホストコレクション** をクリックします。

2. エラータの追加先となるホストコレクションを選択します。

3. **コレクションの各種アクション** タブで、**エラータのインストール** をクリックします。

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

7.7. ホストコレクションからのコンテンツの削除

以下の手順では、パッケージをホストコレクションから削除する方法を示します。

ホストコレクションからのコンテンツの削除方法:

1. **ホスト** > **ホストコレクション** をクリックします。

2. パッケージを削除するホストコレクションをクリックします。

3. **コレクションの各種アクション** タブで、**パッケージのインストール、削除、および更新** をクリックします。

4. 必要に応じて、ラジオボタン **パッケージ** または **パッケージグループ** を選択します。

5. パッケージまたはパッケージグループの名前をフィールドに指定します。

6. **削除** ボタンをクリックし、デフォルトメソッドを使用するパッケージまたはパッケージグループを選択します。または、ボタンの右側にあるドロップダウンアイコンを選択して、使用するメソッドを選択します。**リモート実行 - 最初にカスタマイズ** ニューエントリーを選択すると、**ジョブ呼び出し** ページに移動します。ここでアクションをカスタマイズできます。

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

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

ホストコレクションのライフサイクル環境またはコンテンツビューを変更する方法:

1. **ホスト** > **ホストコレクション** をクリックします。

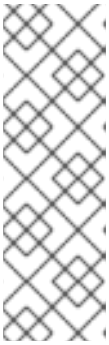
2. ライフサイクル環境またはコンテンツビューを変更するホストコレクションを選択します。

3. **コレクションの各種アクション** タブで、**割り当て済みのライフサイクル環境またはコンテンツビューの変更** をクリックします。

4. ホストコレクションに割り当てるライフサイクル環境を選択します。

5. 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8章 ANSIBLE ロールの使用

8.1. 既存ホストへの ANSIBLE ロールの割り当て

Red Hat Enterprise Linux バージョン 8、7、6.9 以降では、リモート管理に、Ansible ロールを使用できます。

前提条件

ホストにロールを割り当てる前に、Satellite に Ansible のロールをインポートする。詳細は、『**Red Hat Satellite の管理**』の「Red Hat Enterprise Linux System ロールの管理 」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. Ansible ロールを割り当てるホストで、**編集** をクリックします。
3. **Ansible ロール** タブを選択して、**すべての項目** リストで、追加するロールを検索します。
4. 追加するロールを選択して、矢印アイコンをクリックし、**選択された項目** リストにロールを移動させます。
5. **送信** をクリックします。

ホストに Ansible ロールを割り当てると、リモート実行用に Ansible を使用できるようになります。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。

パラメーター変数の上書き

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 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。

8.2. ホストでの ANSIBLE ロールの実行

Satellite Web UI を使用して、ホストで Ansible ロールを実行できます。

前提条件

- Ansible ロールを Satellite にインポートしておく。
- Ansible ロールをホストに割り当てておく。

手順

1. Satellite Web UI で、**ホスト** > **すべてのホスト** に移動します。
2. 実行する Ansible ロールが含まれ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 **Ansible ロールのプレイ** を選択します。

Ansible ロールの実行ページで、Ansible ジョブのステータスを表示できます。ジョブを返すには、**再実行** ボタンをクリックしてください。

8.3. ホストグループへの ANSIBLE ロールの割り当て

Red Hat Enterprise Linux バージョン 8、7、6.9 以降では、リモート管理に、Ansible ロールを使用できます。

前提条件

ホストグループにロールを割り当てる前に、Satellite に Ansible のロールをインポートしておく。詳細情報は、『**Red Hat Satellite の管理**』の「Red Hat Enterprise Linux System ロールの管理 」を参照してください。

手順

1. Satellite Web UI で、**設定** > **ホストグループ** に移動します。
2. ホストグループの一覧から、Ansible ロールを追加するホストグループ名をクリックします。
3. **Ansible ロール** タブを選択して、**すべての項目** リストで、追加するロールを検索します。
4. 追加するロールを選択して、矢印アイコンをクリックし、**選択された項目** リストにロールを移動させます。
5. **送信** をクリックします。

8.4. ホストグループでの ANSIBLE ロールの実行

Satellite Web UI を使用して、ホストグループで Ansible ロールを実行できます。

前提条件

- Ansible ロールを Satellite にインポートしておく。
- Ansible ロールをホストグループに割り当てておく。
- ホストグループに最低でもホストを 1 台以上設定する必要がある。

手順

1. Satellite Web UI で、**設定** > **ホストグループ** に移動します。
2. ホストグループの **アクション** コラムの一覧から、**ロールのプレイ** を選択します。

Ansible ロールの実行ページで、Ansible ジョブのステータスを表示できます。ジョブを返すには、**再実行** ボタンをクリックしてください。

第9章 ホストでのジョブの実行

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 経由で行われます。これは、Satellite Server にはターゲットホストへの直接のアクセスが不要であり、多数のホストを管理するためにスケーリング可能であることを意味します。リモート実行は SSH サービスを使用します。これは、ターゲットホストで有効化され、実行されている必要があります。リモート実行 Capsule が、ターゲットホストのポート 22 にアクセスできることを確認してください。

Satellite は、ERB 構文の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。

Shell 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、「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。

デフォルトでは、Satellite Server はリモート実行ではなく Katello エージェントを使用するように設定されています。この設定を変更するには、**管理** > **設定** に移動し、**コンテンツ** をクリックして、**Use remote execution by default** の設定を変更します。

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。

一度に複数のホストでジョブを実行でき、コマンドで変数を使用して、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、スマートクラスパラメーター、スマート変数、およびホストパラメーターを使用して、変数値を入力できます。Ansible 変数の詳細は、「Satellite での Ansible 変数のオーバーライド」を参照してください。さらに、コマンドを実行するときにテンプレートのカスタム値を指定できます。詳細は、「ジョブの実行」 を参照してください。

9.1. SATELLITE がリモート実行用に CAPSULE を選択する方法

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 Capsule のみを検出します。

1. Satellite は、**リモート実行** 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ースを検出します。
2. Satellite はこれらのインターフェー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。

5. 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 Master Capsule
    - Puppet CA Capsule
    - OpenSCAP Capsule
6. 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。

9.1.1. 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 Master Capsule
- Puppet CA Capsule
- OpenSCAP Capsule

手順

1. Satellite Web UI で、**管理** > **設定**に移動します。
2. **リモート実行** 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 をご利用の場合

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。たとえば、値を **true** に設定するには、以下のコマンドを入力します。

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

9.1.2. 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、**管理** > **設定**に移動します。
2. **リモート実行** 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 をご利用の場合

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

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

9.2. 代替ディレクトリーを使用してクライアン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

オプション: 代替ディレクトリーを使用するには、この手順を実行します。

1. 新しいディレクトリーを作成します (例: **new_place**)。

```
# mkdir /remote_working_dir
```

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 /remote_working_dir
```

3. **/etc/foreman-proxy/settings.d/remote_execution_ssh.yml** ファイルの **remote_working_dir** 設定を編集して、必要なディレクトリーにポイントさせます。以下に例を示します。

```
:remote_working_dir: /remote_working_dir
```

9.3. リモート実行のための SSH 鍵の配布

リモート実行接続の認証に SSH 鍵を使用するには、Capsule から管理するアタッチ済みのホストに、公開 SSH 鍵を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 鍵を配布するキックスタートテンプレートの設定」.

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。

9.3.1. 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

1. Capsule で以下のコマンドを入力します。管理するターゲットホストごとに繰り返します。

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

2. 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@target.example.com
```

9.3.2. 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て、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
```

9.3.3. プロビジョニング中に SSH 鍵を配布する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 鍵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。したがって、Satellite はプロビジョニング中にリモート実行用の SSH 鍵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' %>
```

9.4. 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

以下の手順を実行して、Satellite の **foreman-proxy** ユーザーが Kerberos 付与チケットを取得できるようにします。

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/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 your_client.keytab /var/kerberos/krb5/user/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/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/USER_ID/client.keytab"
```

7. SELinux コンテキストを復元します。

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

9.5. リモート実行用の KERBEROS 認証の設定

Satellite 6.6 以降、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

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

- Kerberos サーバーに 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

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

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

```
# setenforce 0
```

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

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

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

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

4. SELinux を **enforcing** に設定します。

```
# setenforce 1
```

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

6. **remote_execution_effective_user** に移動し、2 番目のコラムを編集して、Kerberos アカウントのユーザー名を追加します。
   BZ#1728612 が解決されるまで、**sudo** のみが、Ansible で機能する **become_method** となっています。

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

9.6. リモートジョブの設定および実行

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。

9.6.1. 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。変更せずにテンプレートを使用する場合には、「ジョブの実行」に進んでください。

デフォルトテンプレートは独自のテンプレートを作成するためのベースとして使用することもできます。デフォルトのジョブテンプレートは、編集できないようにロックされているので、テンプレートのクローンを作成して、このクローンを編集してください。

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。

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、1 行目に **- hosts: all** を追加する必要があります。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、12章*テンプレートリポジトリーの同期* を参照してください。

Ansible become_method の制約

BZ#1728612 が解決されるまで、**remote_execution_effective_user** の設定では、**sudo** のみが、Ansible で機能する **become_method** となっています。

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** タブでのみ表示できるパラメーターは、ジョブテンプレートの入力パラメーターとして使用できる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でパラメーター変数を受け取れない場合には、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。

ジョブテンプレートの作成方法:

1. **ホスト** > **ジョブテンプレート** に移動します。
2. **新規ジョブテンプレート** をクリックします。

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. **ジョブ** タブをクリックして、**ジョブカテゴリー** フィールドに、独自のカテゴリーを入力するか、表9.1「デフォルトのジョブテンプレートカテゴリー」に記載のデフォルトカテゴリーから選択します。
8. オプション: **Description Format** フィールドで、**Install package %{package_name}** など、記述テンプレートを入力します。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を追加する場合には、**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。

テンプレートの構文に他のテンプレートを追加して、詳細なテンプレートを作成できます。詳細は、「詳細テンプレートの作成」 を参照してください。

たとえば、電源関連のアクションを実行するには詳細なテンプレートが必要になります。**Power Action - SSH Default** テンプレートをカスタムテンプレートに組み込む方法は、例9.4「テンプレートにパワー操作を組み込む」 を参照してください。

CLI をご利用の場合

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_template_file" \
--name "template_name" \
```

```
--provider-type SSH \
--job-category "category_name"
```

表9.1 デフォルトのジョブテンプレートカテゴリー

ジョブテンプレートのカテゴリー	説明
Packages	パッケージ関連のアクションを実行するためのテンプレートです。デフォルトで、インストール、更新、および削除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。
Katello	コンテンツ関連のアクションを実行するためのテンプレートです。これらのテンプレートは主として Satellite Web UI の各種の場所で使用されます (たとえば、コンテンツホストの一括操作のための UI など)が、エラータのインストールなどの各種操作を実行するために個別に使用できます。

例9.1 restorecon テンプレートの作成

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

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

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

   <%= input("directory") %> の文字列は、ジョブの呼び出し時にユーザー定義のディレクトリーに置き換えられます。
3. **ジョブ** タブで、**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. **送信** をクリックします。

このテンプレートに基づいてジョブを実行する方法は、例9.2「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。

9.6.2. ジョブの実行

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

リモートジョブを実行する方法:

1. **ホスト** > **すべてのホスト** に移動し、ジョブのターゲットホストを選択します。検索フィールドを使用してホストの一覧を絞り込むことができます。

2. **アクションの選択** 一覧から **リモートジョブのスケジュール** を選択します。

3. **ジョブ呼び出し** ページで、主なジョブ設定を定義します。

   a. 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。

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

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

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

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

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

   - **同時実行レベル** 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、大規模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。

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

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

5. ジョブをすぐに実行する場合は、**スケジュール** が **今すぐ実行** に設定されていることを確認します。さらに、1回限りのジョブを未来の日付で定義したり、再帰的に実行するジョブを設定することもできます。再帰的に実行するジョブについては、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。詳細は Red Hat Enterprise Linux 7 の『**システム管理者のガイド**』の「システムタスクの自動化」セクションを参照してください。

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

CLI をご利用の場合

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
```

カスタムパラメーターを使用してリモートジョブを実行するには、以下の手順を実行します。

1. 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
```

2. 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 template_ID
```

3. 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 \
--job-template "template_name" \
--inputs key1="value",key2="value",... \
--search-query "query"
```

query は、ホストを定義するフィルター式に置き換えます (**"name ~ rex01"** など)。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。

例9.2 複数のホストでの restorecon テンプレートの実行

以下の例では、例9.1「restorecon テンプレートの作成」 で作成されたテンプレートに基づいて、複数のホストでジョブを実行する方法を示します。このジョブは、**/home/** ディレクトリー配下のすべてのファイルで SELinux コンテキストを復元します。

1. **ホスト** > **すべてのホスト** に移動し、ターゲットホストを選択します。**アクションの選択** 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** ジョブカテゴリーを選択し、**Run Command - restorecon** ジョブテンプレートを選択します。

3. **directory** フィールドに、**/home** と入力します。
4. **スケジュール** を **Execute now** に設定します。
5. **送信** をクリックします。**ジョブ呼び出し** ページに移動します。ここでジョブ実行のステータスを監視できます。

9.6.3. ジョブの監視

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

ジョブを監視する方法:

1. ジョブのページに移動します。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 をご利用の場合

実行中のジョブの進捗を監視するには、次の手順を実行します。

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 \
--id job_ID \
--host host_name
```

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 job_ID
```

9.6.4. 詳細テンプレートの作成

ジョブテンプレートの作成時に、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。

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

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

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

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

上記のテンプレートを使用して、レンダリングされたテンプレートからパラメーター定義をインポートする必要があります。これを行うには、**ジョブ** タブに移動して **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。

例9.3 restorecon テンプレートのレンダリング

この例は、例9.1「restorecon テンプレートの作成」で作成される **Run command - restorecon** テンプレートから派生するテンプレートを作成する方法を示しています。このテンプレートでは、ジョブ実行時にユーザーが入力する必要はなく、ターゲットホストの **/home/** ディレクトリー下のすべてのファイルで SELinux コンテキストを復元します。

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

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

例9.4 テンプレートにパワー操作を組み込む

以下の例では、再起動などのパワー操作を実行するため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セットアップ」に従って新規テンプレートを作成し、テンプレートエディター画面で以下の文字列を指定します。

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

9.7. リモート実行用のパーミッションの委任

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

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

Remote Execution User (リモート実行ユーザー) ロールのクローンを作成し、そのフィルターをカスタマイズして詳細度を高めることができます。**view_job_templates** パーミッションでフィルターを調整

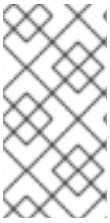
する場合、ユーザーは一致するジョブテンプレートに基づいてジョブを確認し、トリガーすることのみが可能です。**view_hosts** パーミッションおよび **view_smart_proxies** パーミッションを使用すると、ロールに表示されるホストまたは Capsule を制限できます。

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

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

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

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



注記

ユーザーに割り当てるパーミッションは時間の経過と共に変更されます。ユーザーに将来実行するスケジュールされたジョブがあり、パーミッションが変更された場合、パーミッションがジョブ実行の直前にチェックされるため、これによる実行の失敗が生じる可能性があります。

9.8. ANSIBLE RUNNER の設定

Ansible Runner を使用すると、Ansible Playbook を一度に複数のホストで実行する時のパフォーマンスを向上できます。

Ansible Runner は、ホスト 100 台で一括して Ansible Playbook を自動実行するので、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。

前提条件

Satellite では、リモート実行を有効にする必要がある。詳細は、*9章 ホストでのジョブの実行* を参照してください。

手順

Satellite をインストールおよび設定して Ansible Runner を使用するには、以下の手順を行います。

1. Satellite Server で次のコマンドを入力して、統合 Capsule に **ansible-runner** パッケージをインストールします。

```
# foreman-maintain packages install ansible-runner
```

2. Satellite Web UI で、**管理** > **設定** に移動して、**Ansible** タブをクリックします。
3. **Ansible 実行の実装** の行で、**Value** パラメーターを **ansible-runner** に変更します。
4. オプション: Ansible Runner を外部の Capsule にインストールする必要がある場合は、Capsule Server で以下のコマンドを入力します。

```
# yum install ansible-runner
```

第10章 SATELLITE でのベアメタルホストの検出

Red Hat Satellite 6.6 には Discovery プラグインが同梱されています。Discovery プラグインを使用すると、プロビジョニングネットワーク上にある不明ホストで、自動的にベアメタルを検出できます。これらの新規ホストは Satellite Server に登録され、クライアントの Puppet エージェントは、Facter が収集するシステムのファクト (シリアル ID、ネットワークインターフェース、メモリー、ディスク情報など) をアップロードします。登録後、ホストは Satellite Web UI の **検出されたホスト** ページに表示されます。事前に定義された検出ルールを使用して、プロビジョニングを手動 (Web UI、CLI、または API を使用) か、または自動で開始することができます。

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

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

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

Satellite Server Discovery プラグイン

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

The Satellite Capsule Server Discovery プラグイン

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

Satellite Discovery イメージ

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

10.1. PXE ベースの検出に対するネットワーク設定

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

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

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

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

Satellite 6 テンプレート機能は、必要に応じて、インターフェースを設定するインストール後のスクリプトをデプロイするのに使用できます。

10.2. SATELLITE DISCOVERY プラグインの設定

以下のセクションでは、Satellite Discovery プラグインを設定する方法および Satellite Server で PXE 起動テンプレートを準備する方法について説明します。

10.2.1. Satellite Discovery イメージのデプロイ

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

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

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

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

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

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

注記

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

10.2.2. PXE 起動の設定

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

PXE-booting でホスト検出の設定

1. Satellite Web UI で、**ホスト** > **プロビジョニングテンプレート** に移動します。

2. **プロビジョニングテンプレート** ページの右上で、**PXE デフォルトのビルド** をクリックして **OK** をクリックします。

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

10.2.3. グローバル Discovery 設定の確認

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

組織の検出、ロケーションの検出

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

インターフェースのファクト

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

ホスト名のファクト

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

自動プロビジョニング

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

再起動

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

ホスト名の接頭辞

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

ファクトの列

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

強調表示したファクト

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

ストレージのファクト

この変数は正規表現を使用して、ストレージセクションにファクトを整理します。

ハードウェアのファクト

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

ネットワークのファクト

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

IPMI のファクト

この変数は正規表現を使用して、IPMI セクションにファクトを整理します。

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

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

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

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

10.3.1. Discovery 用サブネットの設定

検出されたホストを持つすべてのサブネットが Satellite Server または Capsule Server に通信して、ホストを検出およびプロビジョニングできるように設定する必要があります。まず、Capsule を有効にして、サブネット用のテンプレートをプロビジョニングするプロキシーサーバーを提供できるようにする必要があります。サブネットのテンプレート Capsule は、TFTP Capsule として設定した Capsule と同じにする必要はありません。

新規サブネットを設定する場合には、『**プロビジョニングガイド**』の「Satellite Server へのサブネットの追加」を参照してください。

テンプレート Capsule の有効化

Satellite の統合 Capsule と外部の Capsule Server を有効化して、以下のコマンドでテンプレートをプロビジョニングするプロキシーサーバーを提供します。

```
# satellite-installer --foreman-proxy-templates=1
```

Capsule Server がテンプレートのプロキシーサーバーを提供していることを確認する方法:

1. **インフラストラクチャー** > **Capsules (スマートプロキシー)** に移動します。
2. **アクション** 一覧から **更新** を選択して、リストが最新であることを確認します。
3. **機能** 一覧で、「テンプレート」の用語を検索します。

TFTP Capsule、テンプレート Capsule、Discovery Capsule でサブネットを設定する方法:

1. Satellite Web UI で、**インフラストラクチャー** > **Capsule** に移動します。
2. 設定するサブネットを選択します。
3. **Capsules** タブで、このサブネットに対して **TFTP Capsule**、**Template Capsule** および **Discovery Capsule** を選択します。

10.3.2. Discovery プラグインでの Hammer の使用

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

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

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

10.3.3. ユーザーの各種パーミッションの確認

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

10.4. 検出されたホストのプロビジョニング

Satellite Server と Capsule Server の両方で Discovery プラグインを適切に設定すると、ベアメタルホストが自動的に検出できるようになります。これを行うには「PXE 起動の設定」の説明通りに、PXE 設定テンプレートで設定されたプロビジョニングネットワークでマシンを起動します。マシンは Satellite Server に自動的に登録され、Satellite Web UI の **ホスト** > **検出されたホスト**の一覧に表示されます。

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

10.4.1. ホストの手動プロビジョニング

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

検出されたホストを手動でプロビジョニングする方法:

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

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

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

10.4.2. 検出されたホストの使用停止

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

検出されたホストを使用停止にする方法:

1. ホストをシャットダウンします。
2. **ホスト** > **検出されたホスト** に移動します。
3. **名前** コラムで、使用を停止するホストを検出し、**編集** 一覧から **削除** を選択します。

10.4.3. ホストの自動プロビジョニング

Satellite 6.6 では、プロビジョニングされたホストに、ホストグループを割り当ててプロビジョニングを自動的にトリガーするプロビジョニングルールを定義できます。

プロビジョニングルールを作成する方法:

1. **設定** > **検出ルール** に移動します。
2. **新規ルール** をクリックします。プロビジョニングルールの以下のパラメーターを指定します。
   - **名前** はルールの一覧に表示されるルールの名前です。この名前には、英数字以外の文字やスペースを使用することはできません。
   - **検索** は、特定のルールで検出されたホストを一致させる検索ステートメントです。スコープ指定の検索構文を使用してこれを定義できます。スコープ指定の検索例は「スコープ指定の検索構文」を参照してください。
   - **ホストグループ** は、プロビジョニングプロセスを開始する前に一致するホストに割り当てられるホストグループです。選択したホストグループには必要なパラメーターがすべて設定されていることを確認します。必要なパラメーターにはアスタリスク (*) のマークが付けられます。
   - **ホスト** は、人間の判読できるホスト名を、一致するホストに割り当てるパターンを定義します。これを空白のままにすると、割り当てられるホスト名はデフォルトで「macMACADDRESS」形式となります。プロビジョニングテンプレートに使用される構文と同じ構文が使用されます。詳細と例は「ホスト名のパターン」を参照してください。
   - **ホストの制限** は、ルールに基づいてプロビジョニングされるホストの最大数です。制限に達すると、1 つ以上のホストが削除されるまでルールは有効になりません。通常のユースケースでは、ホスト名やホストグループなどのプロビジョニングパラメーターをエントリーごとに変更する必要がある場合に、サーバーラックまたは行ごとにルールを使用します。この値をゼロ (0) に設定すると制限なしに設定できます。
   - **優先度** は、ルールの実行順序を指定します。値はゼロ以上である必要があります。値が低いほど優先度が高くなります。2 つのルールの優先度が同じ場合には、最初に検出されるルールが適用されます。
   - **有効化** は、ルールを一時的に有効または無効にするオプションを提供します。
3. **送信** をクリックしてルールを保存します。

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

自動プロビジョニングを有効にする方法:

1. Satellite Web UI で、**管理** > **設定** > **Discovered** に移動します。

2. **名前** コラムで自動プロビジョニングを探し、その値を **true** に設定します。
3. **保存**をクリックします。

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

10.4.4. スコープ指定の検索構文

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

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

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

注記

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

Satellite 6.6 では、すべてのファクトは文字列のため、数値比較を実行することはできません。ただし、3 つの重要なファクトが抽出され、数字に変換されます。詳細は表10.1「数値比較を可能にするファクト」で説明されています。

表10.1 数値比較を可能にするファクト

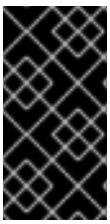
検索パラメーター	説明	使用法の例
cpu_count	CPU の数	cpu_count >= 8
disk_count	割り当てられたディスクの数	disk_count < 10
disks_size	ディスク空き容量の合計 (MiB 単位)	disks_size > 1000000

10.4.5. ホスト名のパターン

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

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

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



重要

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

10.4.6. コマンドラインでの Discovery プラグインの使用

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

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

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

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

10.5. DISCOVERY イメージの拡張

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

最初に、以下のディレクトリー構造を作成します。

```
.
├── autostart.d
│   └── 01_zip.sh
├── bin
│   └── ntpdate
├── facts
│   └── test.rb
```

```
└── lib
    ├── libcrypto.so.1.0.0
    └── ruby
        └── test.rb
```

ここで、

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

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

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

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

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

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

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

PXE テンプレートの更新方法は「PXE 起動の設定」を参照してください。

10.6. SATELLITE 検出のトラブルシューティング

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

- **ホスト** > **プロビジョニングテンプレート** に移動し、**PXE デフォルトのビルド** ボタンを使用してデフォルトの PXELinux テンプレートを再デプロイします。
- TFTP Capsule Server で **pxelinux.cfg/default** 設定ファイルを確認します。
- ホスト、Capsule Server、および Satellite Server 間で適切なネットワーク接続があることを確認します。
- 使用している PXELinux テンプレートに含まれている PXE 検出スニペットを確認します。スニペットの名前は **pxelinux_discovery**、**pxegrub_discovery**、または **pxegrub2_discovery** で

す。PXE 検出スニペットの **proxy.url** オプションと **proxy.type** オプションを検証してください。

- 検出されたノードで DNS が適切に機能していることを確認するか、使用している PXE Linux テンプレートにある PXE 検出スニペットの **proxy.url** オプションにある IP アドレスを使用します。
- DHCP サーバーが IP アドレスを起動したイメージに適切に送信していることを確認します。
- 検出されたホスト (または仮想マシン) に 1200 MB 以上のメモリーがあることを確認します。メモリーが 1200 MB より少なくなると、イメージがインメモリーで抽出される必要があるため、各種のカーネルパニックエラーがランダムに発生する可能性があります。

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

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

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

TTY2 以上を使用して、検出されたホストにログインします。root アカウントおよび SSH アクセスはデフォルトで無効にされますが、以下のカーネルコマンドラインのオプションを使って、デフォルト PXELinux テンプレートの APPEND 行で、SSH の有効化および root パスワードの設定ができます。

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

第11章 RED HAT SATELLITE と ANSIBLE TOWER の統合

Red Hat Satellite 6.6 と Ansible Tower を統合して、Ansible Tower の動的インベントリーソースとして Satellite Server を使用します。

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

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

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

前提条件

- 大規模なデプロイメントでは、管理ユーザーの使用を検討すること。たとえば、数万台のホストを管理するなど、Satellite のデプロイメントが大規模な場合、管理者以外のユーザーを使用すると、認証の確認中にタイムペナルティーが発生するため、パフォーマンスに悪影響を及ぼす可能性があります。
- 管理者以外のユーザーの場合、Satellite Server ユーザーに **Ansible Tower Inventory Reader** ロールを割り当てる。ユーザー、ロール、パーミッションフィルターの管理に関する詳細は、『**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。
- Satellite Server と Ansible Tower を同じネットワークまたはサブネット上にホストする。

手順

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

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

   表11.1 Satellite の認証情報

認証情報の種類	Red Hat Satellite 6
Satellite 6 URL	https://**satellite.example.com**
ユーザー名	統合ロールを持つ Satellite ユーザーのユーザー名
パスワード	Satellite ユーザーのパスワード

2. 新しい認証情報に Ansible Tower ユーザーを追加します。ユーザーを認証情報に追加する方法は『**Ansible Tower ユーザーガイド**』の「認証情報の使用開始」を参照してください。
3. 新しいインベントリーを追加します。詳細は『**Ansible Tower ユーザーガイド**』の「新規インベントリーの追加」を参照してください。

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

表11.2 インベントリーソースオプション

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

5. 追加したソースを同期するようにしてください。

11.2. ホストへのプロビジョニングコールバックの設定

Satellite でホストを作成すると、Ansible Tower を使用して Playbook を実行して新たに作成したホストを設定できます。これは、Ansible Tower では、**プロビジョニングコールバック** と呼ばれます。

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

プロビジョニングコールバックの詳細は『**Ansible Tower ユーザーガイド**』の「プロビジョニングコールバック」を参照してください。

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

1. **ansible_provisioning_callback**
2. **ansible_tower_callback_script**
3. **ansible_tower_callback_service**

ホストまたはホストグループにパラメーターを追加して、新規作成されたホストでの Ansible Playbook の実行にスニペットが使用可能な認証情報を指定できます。

前提条件

プロビジョニングコールバックを設定する前に、Satellite を動的インベントリーとして Ansible Tower に追加する必要がある。詳細は、「Red Hat Satellite と Ansible Tower の統合 」を参照してください。

Ansible Tower Web UI で、以下のタスクを実行する必要があります。

1. 新規ホスト向けに、マシンの認証情報を作成します。Satellite で作成したホストに割り当てる予定の認証情報に、同じパスワードを入力するようにしてください。詳細は、『**Ansible Tower ユーザーガイド**』の「新規認証情報の追加」を参照してください。

2. プロジェクトを作成します。詳細は、『**Ansible Tower ユーザーガイド**』の「プロジェクト」を参照してください。

3. ジョブテンプレートをプロジェクトに追加します。詳細情報は、『**Ansible Tower ユーザーガイド**』の「ジョブテンプレート」を参照してください。

4. ジョブテンプレートで、プロビジョニングコールバックを有効にし、ホストの設定キーを生成して、ジョブテンプレートの **template_ID** をメモする必要があります。ジョブテンプレートの詳細は 『**Ansible Tower ユーザーガイド**』の「ジョブテンプレート」を参照してください。

手順

Satellite の新規ホストにプロビジョニングコールバックを設定するには、以下の手順を実行します。

1. Red Hat Satellite Web UI で、**設定** > **ホストグループ** に移動します。

2. ホストグループを作成するか、既存のホストグループを編集します。

3. ホストグループウィンドウで **パラメーター** タブをクリックします。

4. **Add Parameter** をクリックします。

5. 新規パラメーターごとに、以下の情報を入力します。

表11.3 ホストパラメーター

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

6. **送信** をクリックします。

7. ホストグループを使用してホストを作成します。

8. 新規ホストで、以下のコマンドを入力して、**ansible-callback** サービスを開始します。

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

9. 新規ホストで、以下のコマンドを入力して、**ansible-callback** サービスのステータスを出力します。

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

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

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

手動でのプロビジョニングコールバック

プロビジョニングコールバック URL およびホストからのホスト設定キーを使用して Ansible Tower を呼び出します。以下に例を示します。

```
# curl -k -s --data curl --insecure --data host_config_key=my_config_key \
 https://tower.example.com/api/v2/job_templates/8/callback/
```

プロビジョニングコールバック URL の入力時には、**https** を使用するようにしてください。

これにより、ホストに対して、テンプレートで指定した Playbook の実行がトリガーされます。

第12章 テンプレートリポジトリーの同期

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ーを同期できます。本章では、Git リポジトリーをデモ目的で使用します。

このセクションでは、以下のワークフローを説明します。

- TemplateSync プラグインのインストールおよび設定
- タスクのエクスポートおよびインポートの実行

12.1. TEMPLATESYNC プラグインの有効化

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。

12.2. 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12.1 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	**associate** 許可される値: **always**、**new**、**never**	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	**branch**	Git リポジトリーで、読み取るデフォルトブランチを指定します。	Git リポジトリーで、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	**dirname**	リポジトリー下で、読み込むサブディレクトリーを指定します。	リポジトリー下で、書き込むサブディレクトリーを指定します。
フィルター	**filter**	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	**force**	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
メタデータエクスポートモード	**metadata_export_mode** 許可される値: **refresh**、**keep**、**remove**	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。 ● **リフレッシュ** – 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。 ● **維持**: 既存のメタデータを持続します。 ● **削除**: 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。
否定	**negate** 許可される値: **true**、**false**	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	**prefix**	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
リポジトリー	**repo**	同期するリポジトリーへのパスを定義します。	エクスポートするリポジトリーへのパスを定義します。
詳細	**verbose** 許可される値: **true**、**false**	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

12.3. テンプレートのインポートおよびエクスポート

Satellite API または Hammer CLI 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。

前提条件

インポートしたテンプレートが Satellite Web UI に表示されるようにするには、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 Kickstart File
oses:
- RedHat 7
- RedHat 6
locations:
- First Location
- Second Location
organizations:
- Default Organization
- Extra Organization
%>
```

12.4. SATELLITE API を使用したテンプレートの同期

1. 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)。
2. **TemplateSync** タブで TemplateSync プラグイン設定を設定します。
   a. **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。
   b. Git リポジトリーに一致するように、**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ーに対して、設定を **ssh://git@git.example.com/templates.git** に設定します。
3. 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期待されます。
4. 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
   ```
5. 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。
6. 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ーにテンプレートをエクスポートします。

```
$ curl -H "Accept:application/json,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/export \
-X POST

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

7. 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,version=2" \
-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。

12.5. 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ーを設定した場合は、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。

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,version=2" \
-H "Content-Type:application/json" \
-u login:password \
-k https://satellite.example.com/api/v2/templates/export \
-X POST \

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

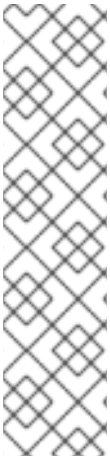
4. コンテンツを変更したら、テンプレートを Satellite Server にインポートします。

```
$ curl -H "Accept:application/json,version=2" \
-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** リポジトリーにテンプレートをエクスポートします。

```
$ curl -H "Accept:application/json,version=2" \
-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\"}"
```

12.6. HAMMER CLI を使用したテンプレートのインポート

hammer import-templates コマンドを使うと任意のリポジトリー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ーにポイントさせることもできます。

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。大型リポジトリー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。

手順

リポジトリー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 \
    --prefix '[Custom Index] ' \
    --filter '.*Template Name$' \
    --repo https://github.com/examplerepo/exampledirectory \
    --branch my_branch \
    --organization 'Default Organization'
```

12.7. HAMMER CLI を使用したテンプレートのエクスポート

hammer export-templates コマンドを使うとテンプレートを Git リポジトリー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

1. Git リポジトリーのローカルコピーをクローンします。
    -

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

2. 以下のコマンドで、ローカルディレクトリーの所有者を **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
```

3. テンプレートをローカルリポジトリー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
```

12.8. 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
```

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

12.9. 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。

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。

```
# yum remove tfm-rubygem-foreman_templates
```

第13章 サンプルシナリオ

13.1. 簡単なシナリオ

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

13.1.1. ホストの作成

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

ホストの作成方法:

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

6. オプションで **パラメーター** タブで、Puppet マスターで、デフォルト値をオーバーライドするパラメーターを選択します。
7. オプションで、**追加情報** タブに、ホストに関する追加情報を入力します。
8. **送信** をクリックします。

13.1.2. ホストの登録

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

ホストの登録方法:

1. 端末で、root ユーザーとしてホストに接続します。
2. 同期ツールが有効になっており、ホストで実行していることを確認します。

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

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

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

4. ホストに対して、適切なコンテンツビューと環境に関連しているアクティベーションキーが存在していることを確認します。存在していない場合は、『**コンテンツ管理ガイド**』の「アクティベーションキーの管理」を参照してください。
5. Red Hat Subscription Manager (RHSM) に関連するすべての以前のホストデータを消去します。

```
# subscription-manager clean
```

6. RHSM を使用してホストを登録します。

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

登録後のコマンド出力:

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

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

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

8. **katello-agent** をインストールします。

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

9. **goferd** サービスが実行していることを確認します。

```
# systemctl start goferd
```

10. Puppet エージェントをインストールして設定します。
    a. Puppet エージェントをインストールします。

```
# yum install puppet
```

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

```
# systemctl enable puppet
```

11. 以下のサーバーと環境の設定を **/etc/puppetlabs/puppet/puppet.conf** ファイルに追加します。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。

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

12. ホスト上で Puppet エージェントを実行します。

```
# puppet agent -t
```

13. Satellite Server Web UI で、Puppet クライアントの SSL 証明書に署名します。
    a. Satellite Server Web UI にログインします。
    b. **インフラストラクチャー** > **Capsule** に移動します。
    c. 必要な Capsule の右側にある一覧から **証明書** を選択します。
    d. 必要なホストの右側にある **署名** をクリックします。
    e. **puppet agent** コマンドを再入力します。

```
# puppet agent -t
```

13.1.3. ホストでのジョブの実行

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

リモートジョブを実行する方法:

1. **ホスト** > **すべてのホスト** に移動し、ターゲットホストを選択します (この例では **host1.example.com** を使用)。
2. 画面右上の **アクションの選択** リストから **リモートジョブのスケジュール** を選択します。
3. **ジョブ呼び出し** ページで、主なジョブ設定を定義します。
    a. **ジョブカテゴリー** リストから、ジョブカテゴリー (例: **Commands**) を選択します。

b. **ジョブテンプレート** リストから、ジョブテンプレート (例: **Run Command - SSH Default**) を選択します。

c. **command** フィールドに、ホストで実行するコマンドを入力します。たとえば、**timedatectl set-timezone Europe/Prague** と実行すると、タイムゾーンがヨーロッパのプラハに設定されます。

4. **送信** をクリックします。

付録A テンプレート作成の参照

Embedded Ruby (ERB) は、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、『**プロビジョニングガイド**』の「プロビジョニングテンプレートの作成」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、*9章 ホストでのジョブの実行*を参照してください。

レポートテンプレート

詳細は、*6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、『**プロビジョニングガイド**』の「パーティションテーブルの作成」を参照してください。

スマート変数

詳細は、『**Puppet ガイド**』の「スマート変数の設定」を参照してください。

スマートクラスパラメーター

Satellite Server に Puppet モジュールを追加する方法の詳細は、『**Puppet ガイド**』の「Configuring Smart Class Parameters」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. 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 -%>
```

テンプレートがレンダリングされるときに出力がない点にご留意ください。

<%= %>

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.2. ERB テンプレートのトラブルシューティング

Satellite Web UI では、特定ホスト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えたテンプレートを一覧表示します。選択したテンプレートの横にある一覧から **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.3. 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。
foreman_url(kind)	完全な URL を、ホストでレンダリングされた指定タイプのテンプレートに返します。たとえば、「provision」タイプのテンプレートは通常 http://HOST/unattended/provision にあります。
snippet(name)	指定されたスニペットテンプレートをレンダリングします。プロビジョニングテンプレートをネスト化するのに便利です。
snippets(file)	Foreman データベースで、指定したスニペットをレンダリングします。データベースにない場合は **unattended/snippets/** ディレクトリーからこれをロードします。
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.4. 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 および CSV 形式がサポートされています。

```
<%= report_render format: :yaml -%>
```

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。

A.5. 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ースのアレイを返します。「アレイの解析」 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。

名前	説明
@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	ホストの Puppet 環境です。
@host.facts	Facter からファクトの Ruby ハッシュを返します。たとえば、出力の 'ipaddress' ファクトにアクセスするには、@host.facts['ipaddress'] を指定します。
@host.grub_pass	ホストの GRUB パスワードを返します。
@host.hostgroup	ホストのホストグループです。
host_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 **true** を返します。
@host.interfaces	プライマリーインターフェースを含む利用可能なすべてのホストインターフェースのアレイが含まれます。「アレイの解析」 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ースのアレイを返します。複数の ID のアレイを入力として渡すことができます (例: @host.interfaces_with_identifier(['eth0', 'eth1']))。「アレイの解析」 を参照してください。
@host.ip	ホストの IP アドレスです。
@host.location	ホストの位置です。
@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ースのアレイを返します (BMC およびボンディングインターフェースを除く)。「アレイの解析」 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。

名前	説明
@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、アレイを返します。
@host.os.medium_uri(@host)	プロビジョニングに使用される URI です (インストールメディアに設定されるパス)。
host_param('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、**false** 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、**true** を返します。
@host.primary_interface	ホストのプライマリーインスタンスを返します。
@host.provider	コンピュートリソースプロバイダーです。
@host.provision_interface	ホストのプロビジョニングインターフェースを返します。インターフェースオブジェクトを返します。
@host.ptable	パーティションテーブル名です。
@host.puppet_ca_server	ホストが使用すべき Puppet CA サーバーです。
@host.puppetmaster	ホストが使用すべき Puppet マスターです。
@host.pxe_build?	ホストがネットワークまたは PXE を使用してプロビジョニングされる場合に **true** を返します。
@host.shortname	ホストの省略名です。
@host.sp_ip	BMC インターフェースの IP アドレスです。
@host.sp_mac	BMC インターフェースの MAC アドレスです。

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

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.6. 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、**true** を返します。
@epel	epel-release rpm の正しいバージョンを自動インストールするコマンドです。%post スクリプトで使用されます。
@mediapath	URL コマンドを提供する詳細なキックスタート行です。
@osver	オペレーティングシステムのメジャーバージョンの番号です。@host.operatingsystem.major と同じです。

A.7. 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.8. 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ースに関連す

るホスト変数は、アレイで分類されるインターフェースデータを返します。特定のインターフェースのパラメーター値を抽出するには、Ruby メソッドを使用してアレイを解析します。

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. インターフェー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.9. 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 リポジトリー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で DHCP ブートモードが有効な場合に **kickstart_networking_setup** スニペットをインポートします。

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

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 GOFERD を使用しないホスト管理

エラータとパッケージの管理をリモート実行で管理する場合は、**goferd** サービスを無効化して、コンテンツホストのメモリーと CPU の負荷を減らすことができます。

Satellite Tools リポジトリーでは、**katello-host-tools** を提供します。これは、エラータを管理するための通信サービスを提供します。

B.1. 前提条件

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

- **katello-host-tools** パッケージがインストールされていることを確認する。

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

- goferd サービスを停止する。

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

- goferd サービスを無効化する。

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

- 「リモート実行のための SSH 鍵の配布」 に従って、SSH 鍵をコンテンツホストに配布する。

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

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

Goferd をシステムのデフォルトとして使用しないホスト管理の設定方法

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

Satellite Server は、goferd 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。

B.3. HAMMER の制限

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

Hammer リモート実行コマンドの使用方法

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

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