Red Hat Satellite 6.2 Hammer CLI ガイド

Satellite の CLI ツール、Hammer の使用

Red Hat Satellite Documentation Team

Red Hat Satellite 6.2 Hammer CLI ガイド

Satellite の CLI ツール、Hammer の使用

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

法律上の通知

概要

本ガイドは、Hammer CLI ツールを使用して Red Hat Satellite を設定/管理する方法を説明します。

目次

第1章 HAMMER の概要

Hammer は、Red Hat Satellite 6 で提供される強力なコマンドラインツールです。Hammer を使用して、CLI コマンドまたはシェルスクリプトの自動化により Red Hat Satellite Server を設定/管理することができます。Hammer は対話式のシェルも提供します。

Hammer と Satellite Web UI の比較

Web UI を使用した移動と比較すると、Hammer を使用する場合には環境変数やエイリアスなどのシェル機能が自由に使えるので Satellite Server との対話がはるかに早くなります。また、Hammer のコマンドを再利用可能なスクリプトに組み込み、あらゆるレベルの複雑性を持つタスクを自動化することもできます。Hammer コマンドからの出力を他のツールにリダイレクトして、既存の環境と統合することができます。Hammer コマンドは、Red Hat Satellite を実行するベースのオペレーティングシステムに直接発行することができます。

Hammer コマンドを発行するには、Satellite Server のベースのオペレーティングシステムにアクセスする必要があるので、Web UI と比較すると、潜在的なユーザー数が限定されてしまいます。Hammer と Web UI の違いはほぼありませんが、Web UI の開発の優先度は高く、特に新しく導入される機能についてはこちら優勢となっています。

Hammer と Satellite API の比較

多くのタスクで、Hammer も Satellite API も同等に利用可能です。Hammer は、スクリプトに適用する前に API の呼び出しの応答をテストするなど、Satellite API よりも使いやすいインターフェースとして利用できます (**`hammer -d organization list`** など、Hammer で発行した API の呼び出しを検査するには **`-d`** オプションを使用します)。API での変更は自動的に Hammer に適用されますが、API を直接使用するスクリプトは手動で更新する必要があります。

バックグラウンドで、各 Hammer コマンドは最初に API へのバインドを確立し、要求を送信します。大量の Hammer コマンドを順番に実行する場合には、パフォーマンスに影響が出る可能性があります。反対に、API で直接通信するスクリプトではバインドを確立するのは 1 度だけです。詳しい情報は、『Red Hat Satellite API ガイド』を参照してください。

1.1. ヘルプ

`hammer` オプションおよびサブコマンドの完全な一覧を表示するには以下を実行します。

```
$ hammer --help
```

以下のように **`--help`** を使用してサブコマンドを確認します。

```
$ hammer organization --help
```

以下のように **`grep`** を使用して help の出力を検索するか、テキストビューワーにリダイレクトすることができます。

```
$ hammer | less
```

1.2. 認証

デフォルトでは **hammer** は、コマンドをハックするたびに、Satellite の認証情報を求めます。以下のようにコマンド実行時に認証情報を指定できます。

```
$ hammer -u <username> -p <password> <subcommands>
```

または、以下の手順に従い、保存した認証情報を使用してください。

1. **~/.hammer/cli_config.yml** ファイルを作成して、このファイルに以下の内容を追加します。

   ```
   :foreman:
    :host: 'https://satellite.example.com/'
    :username: 'username'
    :password: 'password'
   ```

   例の値は、自分の情報に置き換えます。ファイルではタブを使用せずに常にスペースでインデントを入れるようにしてください。

2. 現在のユーザーのみがファイルを読み取りできるようにして、パスワードを保護します。

   ```
   $ chmod 600 ~/.hammer/cli_config.yml
   ```

3. ファイルを保存して閉じます。以降、Hammer を起動すると **~/.hammer/cli_config.yml** ファイルの認証情報が使用されます。

重要

Hammer の設定ファイルではインデントにスペースのみを使用するようにしてください。Hammer 設定ファイルのインデントに、タブは使用しないでください。

注記

本ガイドの例では、認証情報が保存されていることを前提としています。

1.3. HAMMER のスタンドアロンでの使用

Satellite がインストールされていないサーバーで個別に **hammer** をインストールして、そのサーバーをリモートの Satellite と接続して使用することができます。

このパッケージのインストールには、**rhel-X-server-satellite-6.X-rpms** リポジトリーが必要です。接続にワークステーションを使用するには、リポジトリーを手動でインストールする必要があります。詳細は『Red Hat Satellite インストールガイド』を参照してください。

1. **hammer** をインストールします。

   ```
   # yum install tfm-rubygem-hammer_cli_katello
   ```

2. 希望の Satellite を参照するように **/etc/hammer/cli.modules.d/foreman.yml** を編集します。

1.4. デフォルトの組織の設定

多くの **hammer** コマンドは組織固有のものです。**--organization-id** パラメーターを毎回指定しなくても良いように、Hammer コマンドにデフォルトの組織や、場所を設定することができます。これには以下を実行します。

```
$ hammer defaults add --param-name organization_id --param-value
<org_ID>
```

hammer organization list コマンドの出力で **<org_ID>** を検索します。同様に、以下のようにデフォルトの場所を設定することができます。

```
$ hammer defaults add --param-name location_id --param-value <loc_ID>
```

現在指定しているデフォルトの設定を表示するには、以下のコマンドを実行します。

```
$ hammer defaults list
```

単一の組織を管理することが多い場合には、コマンドを省略できるのでデフォルトの組織を指定すると便利です。ただし、別の組織に切り替える場合には、コマンドラインオプションを使用して指定する必要があります。本ガイドの例は、デフォルトの組織を指定していないことを前提としており、注記のようにシェル変数のアプローチを使用します。

1.5. HAMMER の設定

デフォルトでは、グローバルの **hammer** 設定は以下の場所に配置されています。

- 一般的な **hammer** の設定は **/etc/hammer/cli_config.yml**
- CLI モジュールの設定ファイルは **/etc/hammer/cli.modules.d/**

hammer (**~/.hammer/cli_config.yml**) または CLI モジュール (**~/.hammer/cli.modules.d/** の適切な **.yml** ファイル) に対してユーザー固有のディレクティブを設定できます。

設定ファイルの読み込み順および読み込んだモジュールのバージョンを表示するには、以下を実行します。

```
$ hammer -d --version
```

注記

多くの CLI モジュールの設定を読み込むと、**hammer** コマンドの実行の速度が遅くなる可能性があります。このような場合に、定期的に使用しない CLI モジュールを無効化することを検討してください。

「認証」に記載されているような認証情報を保存する以外に、**~/.hammer/** 設定ディレクトリーに他の複数のオプションを設定できます。たとえば、**~/.hammer/cli_config.yml** でデフォルトのログレベルを変更して、以下のディレクティブを使用してログの回転を設定することができます。これらのディレクティブは、現在のユーザーのみに影響を与え、グローバルには適用されない点に注意してください。

```
:log_level: 'warning'
:log_size: 5 #in MB
```

同様に **hammer** の出力で一度に表示される行数を設定することができます (**--per-page** オプションと同等)。

```
:per-page: 30
```

1.6. HAMMER ロギングの設定

hammer を設定して、さまざまな Satellite コンポーネントのデバッグ情報をロギングすることができます。

全 Satellite コンポーネントに対してデバッグまたは通常の設定オプションを設定できます。

- 全コンポーネントのデバッグレベルを設定するには、以下のコマンドを使用します。

```
# hammer admin logging --all --level-debug
```

- 実稼動レベルのロギングを設定するには、以下のコマンドを使用します。

```
# hammer admin logging --all --level-production
```

- ロギングが設定可能なコンポーネントで現在認識されているものを表示するには、以下を実行します。

```
# hammer admin logging --list
```

- このツールで利用可能なオプションをすべて表示するには、以下を実行します。

```
# hammer admin logging --help

Usage:
    hammer admin logging [OPTIONS]
```

1.7. HAMMER シェルの呼び出し

対話型シェルで **hammer** コマンドを発行することができます。このシェルを呼び出すには、以下のコマンドを発行します。

```
$ hammer shell
```

このシェルで、「hammer」と入力せずに直接サブコマンドを入力できるので、スクリプトで使用する前にコマンドをテストする際に便利です。このシェルを終了するには、**exit** と入力するか、[Ctrl + D] を押してください。

1.8. フォーマット済みの出力の生成

hammer コマンドのデフォルトの出力形式を変更して、他のコマンドラインツールやアプリケーションでの出力の処理を簡素化することができます。たとえば、CSV 形式で、カスタムの区切り文字を利用して (今回はセミコロン) 組織を表示するには、以下のコマンドを実行します。

```
$ hammer --csv --csv-separator ";" organization list
```

CSV 形式での出力は、ID を解析したり **for** ループで使用したりする必要がある場合に便利です (例2.6「ACME 組織の全リポジトリーの同期」または例2.8「複数製品への同期プランの割り当て」を参照してください)。

--output オプションには、他に複数のフォーマットオプションがあります。

```
$ hammer --output <output_format> organization list
```

<output_format> は以下のいずれかに置き換えます。

- **table**: 人間が判読できる表 (デフォルト) 形式で出力を生成します。
- **base**: キーと値のペアの形式で出力を生成します。
- **yaml**: YAML 形式で出力を生成します。
- **csv**: コンマ区切りの値形式で出力を生成します。カスタムの区切り文字を定義するには、代わりに **--csv** および **--csv-separator** オプションを使用してください。
- **json**: JavaScript Object Notation (JSON) 形式の出力を生成します。
- **silent**: 出力を表示しません。

1.9. HAMMER でのトラブルシューティング

hammer ping コマンドを使用して、コアの Satellite サービスのステータスを確認することができます。**katello-service status** コマンドと併用すると、Satellite の問題の診断、トラブルシューティングに役立ちます。すべてのサービスが予想どおりに実行されている場合には、出力は以下のようになります。

```
$ hammer ping
candlepin:
    Status:          ok
    Server Response: Duration: 22ms
candlepin_auth:
    Status:          ok
    Server Response: Duration: 17ms
pulp:
    Status:          ok
    Server Response: Duration: 41ms
pulp_auth:
    Status:          ok
    Server Response: Duration: 23ms
foreman_tasks:
    Status:          ok
    Server Response: Duration: 33ms
```

第2章 組織、場所、リポジトリーの管理

hammer を使用して、組織、場所、リポジトリーの作成、編集、管理をすることができます。Web UI を使用した場合の以下の手順は、**Red Hat Satellite サーバー管理ガイド**「組織、ロケーション、およびライフサイクル環境の設定」を参照してください。

2.1. 組織

Red Hat Satellite の組織は、Satellite デプロイメント内にあるシステム、コンテンツ、その他の機能を分離して集めたものです。本章では、**hammer** を使用した組織の作成、変更の方法を説明します。

2.1.1. 組織の作成

以下のコマンドを使用して組織を作成します。

```
$ hammer organization create \
--name "<org_name>" \
--label "<org_label>" \
--description "<org_description>"
```

ここで、

- **<org_name>** は組織の名前に置き換えます。このパラメーターは必須です。
- **<org_label>** は **subscription-manager** など、コマンドラインアプリケーションで使用する組織のラベルに置き換えます。ラベルには、ホワイトスペースを含めることはできず、後ほど変更することもできません。指定されていない場合は、ラベルは組織名から自動的に生成されます (ホワイトスペースはアンダースコアに置き換えられます)。
- **<org_description>** は、組織の簡単な説明に置き換えます。このパラメーターは必須ではありませんが、多数の組織を管理しやすくなります。

組織の作成時に全設定を行うことができます (**hammer organization create --help** を実行してオプションを表示します)。また、**hammer organization update** コマンドを使用して既存の組織を変更することもできます。

例2.1 ACME の組織の作成および更新

以下の例では、ACME という名前の組織の作成方法を示します。

```
$ ORG="ACME"
$ hammer organization create \
--name $ORG \
--description "Example organization"
```

このコマンドは、組織にコンピューとリソースを割り当てます。

```
$ hammer organization update \
--name $ORG \
--compute-resource-ids 1
```

注記

Satellite Server で実行可能なタスクの多くは、組織固有のものです。Hammer コマンドは、**organization**、**organization-label** または **organization-id** オプションを使用して、3 つの方法で組織を特定することができます。組織 ID を検索するには、以下のコマンドを使用します。

```
$ hammer organization list
```

組織名が長い場合には、シェル変数に保存することを検討してください。Hammer コマンドではこの変数を使用できます。以下に例を示します。

```
$ ORG = "Red Hat Enterprise Linux Developer Team"
$ hammer product list --organization $ORG
```

本ガイドの例ではこのアプローチを使用します。

単一の組織を管理することが多い場合には、以下のように、デフォルトのパラメーターとして ID を保存してください。

```
$ hammer defaults add --param-name organization_id --param-
value 1
```

上記の設定では、組織固有のコマンドは **--organization-id 1** が指定されていることを前提とするため、これ以上入力の必要はありません。

2.1.2. マニフェストのアップロード

サブスクリプションマニフェストを使用して、Red Hat カスタマーポータルから Satellite Server にサブスクリプションを移行します。まず、『Red Hat Satellite Content Management Guide』の説明通りに、Red Hat カスタマーポータルでマニフェストを作成し、そのマニフェストを以下のように組織にアップロードします。

```
$ hammer subscription upload \
--organization-label <org_label> \
--file <path_to_manifest>
```

例2.2 マニフェストの ACME 組織へのアップロード

以下の例では、サブスクリプションマニフェストファイルを ACME 組織にアップロードする方法を説明します (組織名がシェル変数に保存されていることを前提とします)。

```
$ hammer subscription upload --organization $ORG --file
/tmp/manifest.zip
```

マニフェストでインポートしたサブスクリプションを表示するには、以下を実行します。

```
$ hammer subscription list --organization $ORG
```

2.2. 場所

Red Hat Satellite の場所は、物理的な場所を示すデフォルトの設定を集めたものです。本セクションでは、**hammer** を使用した場所の作成方法を説明します。

2.2.1. ロケーションの作成

以下のコマンドを使用して場所を作成します。

```
$ hammer location create --name <location_name>
```

例2.3 スクリプトを使用した複数の場所の作成

以下の Bash スクリプトでは、3 つの場所 (ロンドン、ミュンヘン、ボストン) を作成して、これらの場所を ACME 組織に割り当てます。

```
ORG="ACME"
LOCATIONS="london munich boston"

for LOC in ${LOCATIONS}
do
  hammer location create --name "${LOC}"
  hammer location add-organization --name "${LOC}" --organization
"${ORG}"
done
```

hammer location --help を実行して考えられる場所関連の操作を表示します。

2.3. リポジトリー

リポジトリーは、コンテンツコレクション用のストレージを提供します。本セクションでは、**hammer** を使用してリポジトリーを有効化、同期する方法を説明します。

2.3.1. リポジトリーの有効化

Red Hat リポジトリーを有効化する前に、リポジトリーの名前、リポジトリーが提供する製品の名前、ベースアーキテクチャー、リリースバージョンを知っておく必要があります。以下のコマンドを使用して、リポジトリーを有効化します。

```
$ hammer repository-set enable \
--organization-label <org_label> \
--product "<product_name>" \
--basearch "<base_architecture>" \
--releasever "<release_version>" \
--name "<repository_name>"
```

例2.4 Red Hat Enterprise Linux リポジトリーの有効化

以下のコマンドは、組織用に Red Hat Enterprise Linux 7 Server リポジトリーを有効化します。

```
$ hammer repository-set enable \
--organization $ORG \
--product "Red Hat Enterprise Linux Server" \
--basearch "x86_64" \
--releasever "7Server" \
--name "Red Hat Enterprise Linux 7 Server (RPMs)"
```

hammer repository-set --help を実行して、利用可能なリポジトリー関連の操作を表示します。また、**hammer repository --help** も参照してください。

2.3.2. リポジトリーの同期

リポジトリーを同期すると、Red Hat カスタマーポータルから Satellite Server にコンテンツをプルします。リポジトリーを同期するには、リポジトリーの名前と製品名を指定する必要があります。

```
$ hammer repository synchronize \
--product "<product_name>" \
--name "<repo_name>" \
--organization-label <org_label> \
--async
```

コンテンツビューを作成した場合に、単一の組織内に同じ名前を持つリポジトリーが複数存在する可能性がある点に注意してください。**--id** オプションを使用して、同期するリポジトリーを特定します (**hammer repository list** を実行してリポジトリー ID を検索します)。

例2.5 Red Hat Enterprise Linux リポジトリーの同期

以下のコマンドは、組織内の Red Hat Enterprise Linux 7 Server リポジトリーを 1 度同期します。

```
$ hammer repository synchronize \
--product "Red Hat Enterprise Linux Server" \
--name "Red Hat Enterprise Linux 7 Server (RPMs)" \
--organization $ORG \
--async
```

タスク ID は、上記のコマンドを実行後に表示されます。

```
Repository is being synchronized in task 640bb71f-0ce5-40a3-a675-
425a4acacceb
```

タスクの進捗を表示するには以下を実行します。

```
$ hammer task progress --id 640bb71f-0ce5-40a3-a675-425a4acacceb
```

最初の同期を完了した後に、リポジトリーは Satellite Server にミラーリングされたリポジトリー一覧に追加されます。以下のコマンドを実行して一覧を表示します。

```
$ hammer repository list --organization $ORG
```

以下のように製品に含まれる全リポジトリーを同期することもできます。

```
$ hammer product synchronize \
--organization-label <org_label> \
--name "<product_name>" \
--async
```

--async オプションを指定すると、リポジトリーの同期はバックグラウンドで実行され、並行して複数のリポジトリーを有効化、同期することも可能です。

例2.6 ACME 組織の全リポジトリーの同期

以下の Bash スクリプトは、ACME 組織内の全リポジトリーを同期します。

```
ORG="ACME"

for i in $(hammer --csv repository list --organization $ORG | grep -vi
'^ID' | awk -F, {'print $1'})
do
  hammer repository synchronize --id ${i} --organization $ORG --async
done
```

2.3.3. 同期プランの作成

Red Hat Satellite の製品は、同期プロセスの最小単位として機能するリポジトリーの集まりです。指定の間隔で、選択した製品のリポジトリーを自動的に更新するように同期プランを作成することができます。

同期プランを定義するには、以下のコマンドを実行します。

```
$ hammer sync-plan create \
--name "<sync_plan_name>" \
--enabled=true \
--interval <repetition_interval> \
--organization-label <org_label> \
--sync-date "<initial_sync>"
```

<repetition_interval> は **hourly**、**daily** または **weekly** に置き換えます。また、**<initial_sync>** は、初回同期の日時 (「YYYY-MM-DD HH:MM:SS」形式) に置き換えます。

例2.7 同期プランの作成

以下のコマンドは、2016 年 1 月 15 日から、毎日午前 3 時に実行される ACME 組織の日次の同期スケジュールを作成します。

```
$ hammer sync-plan create \
--name "daily sync at 3 a.m." \
--enabled=true \
--interval daily \
```

```
--organization $ORG \
--sync-date "2016-01-15 03:00:00"
```

同期プランと製品を関連付けるには、以下のコマンドを発行します。

```
$ hammer product set-sync-plan \
--organization-label <org_label> \
--name "<product_name>" \
--sync-plan "<sync_plan_name>"
```

例2.8 複数製品への同期プランの割り当て

以下の Bash スクリプトは、最低でも 1 回同期され、少なくとも 1 つのリポジトリーが含まれる ACME 組織の製品を選択し、「daily sync at 3 a.m」という同期プランに割り当てます。

```
ORG="ACME"
SYNC_PLAN="daily sync at 3 a.m."

for i in $(hammer --csv product list --organization $ORG --per-page 999
| grep -vi '^ID' | grep -vi not_synced | awk -F, {'{ if ($5!=0) print
$1}'})
do
  hammer product set-sync-plan --sync-plan $SYNC_PLAN --organization
$ORG --id $i
done
```

スクリプトの実行後には、以下のコマンドを発行して、どの製品が同期プランに割り当てられたのかを確認します。

```
$ hammer product list --organization $ORG --sync-plan "daily sync at
3 a.m."
```

選択した組織で利用可能な同期プランを表示するには、以下のコマンドを実行します。

```
$ hammer sync-plan list --organization-label <org_label>
```

製品および同期プランとの連携方法に関する詳しい情報は **hammer sync-plan --help** および **hammer product --help** を参照してください。

2.3.4. カスタムリポジトリーの作成

Red Hat のリポジトリーを有効化すると、適切な製品が自動的に作成されます。カスタムのパッケージのリポジトリーを有効化するには、最初にこのリポジトリーの製品を手動で作成する必要があります。

以下のコマンドを使用してカスタムの製品を作成します。

```
$ hammer product create --name "<product_name>" --organization-label
<org_label>
```

以下のコマンドは、カスタムの製品の下に新規リポジトリーを作成します。

```
$ hammer repository create \
--name "<repo_name>" \
--organization-label <org_label> \
--product "<product_name>" \
--content-type <cont_type> --publish-via-http true \
--url "<repo_url>"
```

例で使用している値は独自の値で置き換えてください。

- 特に、**<cont_type>** はリポジトリーのコンテンツタイプを指定しており、**yum**、**puppet**、**docker** の中から 1 つ選択してください。
- **<repo_url>** は、利用可能なリポジトリーの URL を指定します。--publish-via-http が有効化されている場合のみ、有効です。

カスタムのリポジトリーにパッケージをアップロードするには、以下のコマンドを実行します。

```
$ hammer repository upload-content \
--product "<product_name>" \
 --organization-label <org_label> \
 --id "<repo_id>" \
--path <path_to_dir>
```

<path_to_dir> は、カスタムリポジトリーに追加するコンテンツ (RPM パッケージ、Puppet モジュール、Docker イメージ) を含むディレクトリーへのパスに置き換えます。

第3章 コンテンツのライフサイクルの管理

このセクションは、**hammer** を使用してコンテンツビューを作成し、ライフサイクルの環境全体でプロモートする方法を説明します。

3.1. ライフサイクル環境の作成

ライフサイクル環境は、コンテンツライフサイクルのステージを表します。本セクションでは、**hammer** を使用してライフサイクル環境を表示、作成する方法を説明します。デフォルトでは、組織ごとにライブラリー環境が存在します。以下の構文を使用して、新しいライフサイクル環境を作成します。

```
$ hammer lifecycle-environment create \
--name <env_name> \
--description "<env_description>" \
--organization-label <org_label> \
--prior <prior_env_name>
```

例3.1 ライフサイクル環境の作成

以下の例では、ACME 組織のライブラリーをもとに新しい環境を作成する方法を示します (組織名がシェル変数に保存されていることが前提です)。

```
$ hammer lifecycle-environment create \
--name Development \
--description "Initial testing" \
--organization $ORG \
--prior Library
```

--prior を使用して **Development** をもとに別のライフサイクル環境を作成することができます。

既存のライフサイクル環境を表示するには、以下のコマンドを実行します。

```
$ hammer lifecycle-environment list --organization-label <org_label>
```

上記のコマンドの出力は以下のようになります。

```
---|-------------|------------
ID | NAME        | PRIOR
---|-------------|------------
2  | Library     |
5  | Development | Library
6  | Testing     | Development
---|-------------|------------
```

ライフサイクル環境に関するコマンドの情報は、**hammer lifecycle-environment --help** を参照してください。

3.2. コンテンツビューの作成

コンテンツビューとは、ライブラリーからのコンテンツのサブセットで、インテリジェントフィルターで作成されます。コンテンツビューは、別の用途で (通常、開発、QA、実稼働環境) コンテンツを利用できるように、ライフサイクル環境に公開、プロモートすることができます。

```
$ hammer content-view create \
--name <cv_name> \
--repository-ids <repo_ID1>,<repo_ID2>,<repo_ID3> \
--description "<cv_description>" \
--organization-label <org_label>
```

--repository-ids オプションは、選択したリポジトリーをコンテンツビューに追加して、**hammer repository list** コマンドを使用して ID を検索します。このオプションを省略して、空のコンテンツビューを作成し、**update** または **add-repository** サブコマンドを使用して後ほど変更することもできます。

例3.2 コンテンツビューの作成

以下の例は、ACME 組織の下にコンテンツビューを作成して、3 つのリポジトリーに割り当てます。

```
$ hammer content-view create \
--name cv-rhel7-server \
--repository-ids 1,2,3 \
--description "Initial CV for RHEL 7" \
--organization $ORG
```

例3.3 複合コンテンツビューの作成

複合コンテンツビューは、1 つまたは複数のコンテンツビューで構成されます。以下の例は、2 つの既存のコンテンツビューから複合コンテンツビューを作成する方法を示します。

```
$ hammer content-view create \
--name ccv-rhel7-server-scl \
--description "CCV for RHEL7 and Software Collections" \
--organization $ORG \
--composite --component-ids 2,6
```

hammer content-view list を実行して、**--component-ids** オプションの ID を検索します。

コンテンツビューに追加可能なコンテンツビューは、RPM パッケージ、Puppet モジュール、Docker イメージの 3 つです。

3.2.1. リポジトリーのコンテンツビューへの追加

以下のコマンドを使用して既存のコンテンツビューにリポジトリーを追加します。

```
$ hammer content-view update \
--repository-ids <repo_ID1>,<repo_ID2>... \
--name <cv_name> \
--organization-label <org_label>
```

上記のコマンドは、複数のリポジトリーからなる空のコンテンツビューを生成するのに便利です。既存のリポジトリーは上書きされるので、コンテンツビューのリポジトリー数を増やすには以下を使用します。

```
$ hammer content-view add-repository \
--organization-label <org_label> \
--name <cv_name> \
--repository-id <repo_ID>
```

同様に **remove-repository** サブコマンドを使用してコンテンツビューからリポジトリーを削除することができます。コンテンツビューのリポジトリーを検査するには **hammer content-view info** を使用します。

例3.4 コンテンツビューのパッケージのフィルタリング (パッケージの除外)

フィルターを使用すると、リポジトリーからパッケージのサブセットを選択して (除外/包含して) カスタマイズしたコンテンツビューを作成できます。以下の例は、フィルターを作成して **cv-rhel7-server** コンテンツビューから **emacs** パッケージを除外する方法を示します。

まず、組織内にコンテンツビューのフィルターを作成します。

```
$ hammer content-view filter create \
--type rpm \
--name exclude-emacs \
--description "Excluding emacs package" \
--inclusion false \
--organization $ORG \
--repository-ids 1,2,3 \
--content-view cv-rhel7-server
```

hammer repository list を実行してリポジトリー ID を検索します。「emac」で始まる名前のパッケージを除外するルールを作成して、以下のようにフィルターに追加します。

```
$ hammer content-view filter rule create \
--name "emacs*" \
--organization $ORG \
--content-view cv-rhel7-server \
--content-view-filter exclude-emacs
```

上記を実行すると cv-rhel7-server のコンテンツビューを使用したホストは、macs へアクセスできなくなります。複数のルールをフィルターに追加することができます。フィルタリングパラメーターの完全な一覧については **hammer content-view rule create --help** を参照してください。フィルターに存在するルールを確認するには、以下のコマンドを実行します。

```
$ hammer content-view filter rule list \
--content-view cv-rhel7-server \
--content-view-filter exclude-emacs \
--organization $ORG
```

例3.5 コンテンツビュー用のパッケージのフィルタリング (日付別のエラータの絞込)

以下の例では、**cv-rhel7-server** コンテンツビューから特定の日付以前の発表されたエラータを除外するフィルターを作成する方法を示します。エラータ管理の情報は、8章*エラータの管理*を参照してください。以下のようにコンテンツビューのフィルターを作成します。

```
$ hammer content-view filter create \
--type erratum \
--name limit-errata-by-date \
--description "Excluding errata by date" \
--inclusion false \
--organization $ORG \
--repository-ids 1,2,3 \
--content-view cv-rhel7-server
```

以下のように「emac」で始まる名前のエラータを除外するルールを作成し、フィルターに追加します。

```
$ hammer content-view filter rule create \
--end-date <YYYY-MM-DD> \
--organization $ORG \
--content-view cv-rhel7-server \
--content-view-filter limit-errata-by-date \
--types enhancement,bugfix,security
```

3.2.2. コンテンツビューへの Puppet モジュールの追加

Puppet モジュールをコンテンツビューに追加するには、最初にこのモジュールをカスタムの製品内の Puppet リポジトリーにアップロードします。「カスタムリポジトリーの作成」からのコマンドを使用して、リポジトリーに製品を作成し、そのリポジトリーに Puppet モジュールにアップロードします。

Puppet モジュールをコンテンツビューに追加するには、以下のコマンドを実行します。

```
$ hammer content-view puppet-module add \
--content-view <cv_name> \
--name <module_name>
```

例3.6 Puppet モジュールのコンテンツビューへの追加

以下の例では、外部ソースからの Puppet モジュールをcv-rhel7-server コンテンツビューに追加する方法を示します。

1. Puppet Forge から **concat** モジュール (複数のテキストの断片からファイルを構築) をダウンロードします。

```
$ wget -O /tmp/puppetlabs-concat-1.2.5.tar.gz
https://forgeapi.puppetlabs.com
/v3/files/puppetlabs-concat-1.2.3.tar.gz
```

2. ACME-puppet 製品配下に Puppet リポジトリーを作成して、このリポジトリーにモジュールをアップロードします (この例ではリポジトリー ID は 6 という前提です)。

```
$ hammer product create \
--name "ACME-puppet" \
--organization $ORG
```

```
$ hammer repository create \
--organization $ORG \
--product ACME-puppet \
--name "ACME Puppet Repository" \
--content-type puppet \
--url "https://forge.puppetlabs.com/"
```

```
$ hammer repository upload-content \
--organization $ORG \
--product ACME-puppet \
--id 6 \
--path /tmp/puppetlabs-concat-1.2.5.tar.gz
```

3. **id**、**name** または **author** パラメーターを使用してコンテンツビューにモジュールを追加します。正確な値を見つけ出すには、以下を入力します。

```
$ hammer puppet-module list --organization $ORG
---|--------|------------|--------
ID | NAME   | AUTHOR     | VERSION
---|--------|------------|--------
1  | concat | puppetlabs | 1.2.3
---|--------|------------|--------
```

コンテンツビューにモジュールを追加するには、以下を実行します。

```
$ hammer content-view puppet-module add \
--name concat \
--content-view cv-rhel7-server \
--organization $ORG
```

モジュールが正常に追加されたかどうかを検証するには、以下のコマンドを実行します。

```
$ hammer content-view puppet-module list \
--content-view cv-rhel7-server \
--organization $ORG
```

3.2.3. Docker イメージのコンテンツビューへの追加

以下のように、Docker イメージを直接専用のリポジトリーにアップロードできます。

```
$ hammer repository upload-content --path <image_archive> --id
<repo_id>
```

<image_archive> は、Docker イメージを含むアーカイブへのパスに置き換えます。**<repo_id>** を使用して Docker のコンテンツタイプのリポジトリーを特定してから、そのリポジトリーをコンテンツビューに追加できます。

3.3. コンテンツビューの公開

コンテンツビューを公開し、表示してホストが利用できるようにします。以下のコマンドを使用して選択したコンテンツビューを公開します。

```
$ hammer content-view publish \
--id <cv_ID> \
--organization-label <org_label> \
--async
```

コンテンツビューの **<cv_ID>** を検索して、**hammer content-view list** コマンドの出力に公開します。公開されたコンテンツビューはライブラリー環境で利用できます。コンテンツビューのステータスを確認するには、以下のコマンドを実行します。

```
$ hammer content-view info --id <cv_ID>
```

3.4. コンテンツビューのプロモート

プロモーションとは、別のライフサイクル環境にコンテンツビューを移動する行為のことです。これには以下のコマンドを実行します。

```
$ hammer content-view version promote \
--content-view <cv_name> \
--organization-label <org_label> \
--to-lifecycle-environment <env_name>
```

ここでは **env_name** は移行後のライフサイクル環境名を示します。

例3.7 ライフサイクル環境全体でのコンテンツビューのプロモート

以下のバッシュスクリプトは、ACME 組織のすべてのライフサイクル環境でライブラリーから選択したコンテンツビューをプロモートします。

```
ORG="ACME"
CV_ID=1

for i in $(hammer --csv lifecycle-environment list --organization
$ORG | grep -vi '^ID' | awk -F, {'print $1'} | sort -n)
do
   hammer content-view version promote --organization $ORG --to-
lifecycle-environment-id $i --id $CV_ID
done
```

コンテンツビューが正しくプロモートされていることを確認するには、以下のコマンドを実行します。

```
$ hammer content-view version info --id 1
```

3.5. コンテンツビューの増分更新

増分更新では、ライフサイクル環境で新規コンテンツビューのバージョンをプロモートする必要なしに、公開されたコンテンツビューを変更できます。増分更新の結果、新規コンテンツビューのマイナーバージョンが作成されます。増分更新は、緊急時に素早く更新するには有用です。エラータ、パッケージ または Puppet モジュールの追加に、増分更新を使用できます。

コンテンツビューに新規パッケージを追加する増分更新を作成するには以下を実行します。

```
$ hammer content-view version incremental-update \
--content-view-version-id <cv_ID> \
--packages <pkg_name1>,<pkg_name2> \
--lifecycle-environment-ids <env_ID1>, <env_ID2>,...
```

hammer content view version list の出力でコンテンツビューのバージョン ID を検索します。**--packages** オプションでパッケージを指定するのではなく、**--puppet-modules** で Puppet モジュールを、**--errata-ids** でエラータを追加できます (例3.8「増分更新を使用してコンテンツビューにエラータを追加する手順」を参照します)。増分更新との連携についての情報は **hammer content-view version incremental-update --help** を実行します。

例3.8 増分更新を使用してコンテンツビューにエラータを追加する手順

以下の例では、コンテンツビューの増分更新を作成して、ホスト (**auth01.example.com**) にエラータを適用する方法を示します。

```
$ hammer content-view version incremental-update \
--content-view-version-id 4 \
--errata-ids 8c3801f6-12a7-4a62-83f4-addbb1f34ce6 \
--lifecycle-environments Infrastructure
```

上記のコマンドで必要な情報を特定するには、以下の手順を実行します。

1. 以下を実行して、ホストが登録されているコンテンツビューとライフサイクル環境を特定します。

   ```
   $ hammer content-host info --name auth01.example.com --
   organization $ORG
   ```

2. 次にコンテンツビューの現在のバージョンを特定します (コンテンツビューの名前は **RHEL7_Infra** という前提です)。

   ```
   $ hammer content-view info --name "RHEL7_Infra" --organization
   $ORG
   ```

3. **ライブラリー** にあるアプリケーションエラータの一覧に適用するエラータ ID を特定します。

```
$ hammer erratum list --content-view "RHEL7_Infra" --
organization $ORG
$ hammer host errata list --host auth01.example.com
```

第4章 アクティベーションキーの管理

アクティベーションキーはホストのサブスクリプションプロパティーを定義します。アクティベーションキーを使用すると、ホストの登録を迅速化できます。以下の手順を Web UI で行う方法は『Red Hat Satellite Host Configuration Guide』を参照してください。

アクティベーションキーで考えられるユースケースとして、以下の 3 つが挙げられます。

- **サブスクリプションの指定のないアクティベーションキー**: アクティベーションキーを使用するホストが最適なサブスクリプションを検索します。これは、**subscription-manager --auto-attach** を実行するのと同じです。例4.1「空のアクティベーションキーの作成」を参照してください。
- **自動割り当て用にカスタムサブスクリプションプールを指定するアクティベーションキー**: アクティベーションキーを使用するホストはアクティベーションキーで指定された一覧から最適なサブスクリプションを選択します。例4.2「カスタムのサブスクリプションプールを使用したアクティベーションキーの作成」を参照してください。
- **適切なサブスクリプションセットが指定されたアクティベーションキー**: 例4.3「必須のサブスクリプション一覧でのアクティベーションキーの作成」を参照してください。

注記

アクティベーションキーは、ホストが登録されている場合にのみ使用できます。アクティベーションキーに変更が加えられた場合に、今後変更されたアクティベーションキーで登録されたホストにのみ適用されます。その変更は既存のホストには加えられません。

アクティベーションキーを作成するには以下のコマンドを実行します。

```
$ hammer activation-key create --name <ak_name> \
--organization-label <org_label> \
--content-view <cv_name> \
--lifecycle-environment <lc_name>
```

コンテンツビューは公開する必要がある点に注意してください。アクティベーションキーに関する操作の完全一覧を表示するには **hammer activation-key --help** のコマンドを使用します。

アクティベーションキーにサブスクリプションを追加するには、以下のコマンドを実行します。

```
$ hammer activation-key add-subscription \
--id <ak_ID> \
--subscription-id <sub_ID>
```

アクティベーションキー ID を検索するには **hammer activation-key list** を使用し、サブスクリプション ID を検索するには **hammer subscription list** を使用します。

例4.1 空のアクティベーションキーの作成

以下の例では、関連付けられたホストを自動的に最適なサブスクリプションにアタッチするアクティベーションキーを作成する方法を示します。

```
$ hammer activation-key create \
--name "automatically attach key" \
--organization $ORG \
--content-view cv-rhel7-server \
--lifecycle-environment Testing
```

上記のコマンドを実行すると、cv-rhel7-server コンテンツビューに登録されたホストがこのアクティベーションキーに関連付けられます。

例4.2 カスタムのサブスクリプションプールを使用したアクティベーションキーの作成

以下の例では、関連付けられたホストを自動的に、アクティベーションキーに指定された一覧にある最適なサブスクリプションにアタッチするアクティベーションキーを作成する方法を示します。

まず、空のアクティベーションキーを作成します。

```
$ hammer activation-key create \
--name "custom pool key" \
--organization $ORG \
--content-view cv-rhel7-server \
--lifecycle-environment Testing
```

サブスクリプションをアクティベーションキーに追加します。

```
$ hammer activation-key add-subscription \
--name "custom pool key" \
--subscription-id 1
```

必須のサブスクリプションがアクティベーションキーにすべて追加されるまで、この手順を繰り返します。

例4.3 必須のサブスクリプション一覧でのアクティベーションキーの作成

以下の例では、関連付けられたホストを、アクティベーションキーに指定されたサブスクリプションすべてにアタッチするアクティベーションキーを作成する方法を示します。

最初にアクティベーションキーを作成して、必須のサブスクリプションをすべてそのアクティベーションキーに追加します。例4.2「カスタムのサブスクリプションプールを使用したアクティベーションキーの作成」の手順に従ってください。

次に、アクティベーションキーの **auto-attach** プロパティーを無効にします。

```
$ hammer activation-key update \
--organization $ORG \
--name "mandatory subs key" \
--auto-attach false
```

重要

複数のアクティベーションキーを 1 つのコンテンツビューに割り当てることができます。設定が競合した場合には、最後に指定されたキーの値が優先されます。以下のようにホストグループのパラメーターを設定することで、優先順位を指定することができます。

```
$ hammer hostgroup set-parameter \
--name kt_activation_keys \
--value <name_of_first_key>, <name_of_second_key>,... \
--hostgroup <hostgroup_name>
```

第5章 プロビジョニング環境の設定

以下のセクションでは **hammer** を使用してプロビジョニング環境のあらゆる段階を設定する方法を示します。以下の手順を Web UI で行う手順は『Red Hat Satellite Host Configuration Guide』を参照してください。

5.1. ドメイン

Red Hat Satellite のドメインは DNS ゾーンを表します。Satellite は、Red Hat Satellite Capsule Server DNS を使用してドメイン名を割り当てる機能があります。これにより、特定のドメイン内のホストをグループ化し、名前を指定して、パラメーターや Puppet 変数と関連付けることができます。

新規ドメインを作成するには、以下のコマンドを実行します。

```
$ hammer domain create --name <domain_name>
```

hammer organization add-domain または **hammer location add-domain** コマンドを使用して組織と場所に対して、新規作成したドメインを関連付けることができます。ドメインのステータスを表示するには、以下のコマンドを実行します。

```
$ hammer domain info --name <domain_name>
```

5.2. サブネット

Red Hat Satellite のサブネットは、システムのグループを指定するネットワークを定義します。サブネットは、標準の IP アドレス設定を使用して、ネットワークを定義、Red Hat Satellite Capsule Server の DHCP 機能を使用して、サブネット内のシステムに IP アドレスを割り当てます。以下のコマンドには、サブネットの作成に最小限必要なオプションが含まれます。

```
$ hammer subnet create \
--name <subnet_name> \
--organization-ids <org_ID1>,<org_ID2>... \
--location-ids <loc_ID1>,<loc_ID2>... \
--domain-ids <dom_ID1>,<dom_ID2>... \
--boot-mode <boot_mode> \
--network <network_address> \
--mask <netmask> \
--ipam <ipam>
```

ここでは **<boot_mode>** は **Static** または **DHCP** のいずれか、**<ipam>** は **DHCP**, **Internal DB**、**None** のいずれかを指定します。DHCP を使用する場合は **--from** および **--to** オプションを使用して IP 範囲を設定してください。設定可能なオプションの完全な一覧は、**hammer subnet create --help** コマンドの出力を参照してください。

5.3. アーキテクチャー

Satellite のアーキテクチャーはホストおよびオペレーティングシステムの論理グループを表します。アーキテクチャーを表示するには以下のコマンドを実行します。

```
$ hammer architecture list
```

アーキテクチャーは、ホストが Puppet に登録されると自動的に Satellite により登録されるため、(**hammer** でこのオプションが提供されていますが) 手動で作成する必要はほぼありません。

5.4. コンピュートリソース

コンピュートリソースは、仮想化およびクラウドプロバイダーからハードウェアを抽象化したものです。Satellite はコンピュートリソースを使用して仮想マシンとコンテナーをプロビジョニングします。以下のコマンドを使用してコンピュートリソースを作成します。

```
$ hammer compute-resource create \
--name <cr_name> \
--organization-ids <org_ID1>,<org_ID2>... \
--location-ids <loc_ID1>,<loc_ID2>... \
--provider <provider>
```

ここでは **<provider>** は **RHEV**、**RHEL OpenStack Platform**、**Libvirt**、**Docker**、**Rackspace**、**Google**、**EC2** または **VMware** のいずれかを指定します。プロバイダーのタイプに従い、**--url** または **--user** などの他のオプションが必要な場合もあります。詳細は **hammer compute-resource create --help** コマンドの出力を参照してください。

5.5. インストールメディア

インストールメディア (ISO イメージ) では、Red Hat Satellite のキックスタートツリー及び新規ホストインストールのコンテンツを提供します。メディアを表示するには、以下のコマンドを実行します。

```
$ hammer medium list
```

新規メディアを追加するには以下のコマンドを実行します。

```
$ hammer medium create --name <medium_name> --path <path_to_medium>
```

メディアの追加時 (**hammer medium create --help** コマンドの出力参照) または 後ほど **hammer organization add-medium** か **hammer location add-medium** コマンドを使用して、メディアを組織および場所で直接利用できるようにすることが可能です。

5.6. パーティションテーブル

パーティションテーブルは、システムのプロビジョニング時の新規インストール用のパーティションとファイルシステムのレイアウトを定義します。Red Hat Satellite により、オペレーティングシステムファミリーが関連付けられたデフォルトのパーティションテーブルを提供します。パーティションテーブルを表示するには、以下のコマンドを実行します。

```
$ hammer partition-table list
```

新しいパーティションテーブルを作成するには、以下のコマンドを実行します。

```
$ hammer partition-table create \
--name <table_name> \
--file <path_to_layout_file> \
--os-family <os_family>
```

他のサブコマンドについては **hammer partition-table --help** コマンドを参照してください。

5.7. プロビジョニングテンプレート

テンプレートをプロビジョニングすると、無人インストールをシステム的に実行する手段が提供されます。Satellite が提供するプロビジョニングテンプレートを表示するには以下のコマンドを実行します。

```
$ hammer template list
```

新規テンプレートを追加するには以下のコマンドを実行します。

```
$ hammer template create --name <template_name> --file
<path_to_template_file>
```

他のサブオプションについては **hammer template --help** コマンドの出力を参照します。

5.8. オペレーティングシステム

オペレーティングシステムはインストールの方法とメディアの組み合わせを定義して、ファミリーにグループ化します。デフォルトでは、Red Hat Satellite は Red Hat ファミリーを使用します。ファミリーを利用することで、Satellite はホストのプロビジョニング時に特定の動作を変更することができます。オペレーティングシステムを表示するには以下のコマンドを実行します。

```
$ hammer os list
```

新規オペレーティングシステムを作成するには、以下のコマンドを実行します。

```
$ hammer os create --name <os_name> --major <version_number>
```

次に、アーキテクチャー、パーティションテーブル、インストールメディア、設定テンプレートをオペレーティングシステムに追加することができます。詳細は **hammer os --help** の出力を参照してください。

例5.1 複数のオペレーティングシステムの更新

以下のバッシュスクリプトは、各オペレーティングシステムにパーティションテーブル (**Kickstart default**)、設定テンプレート (**Kickstart default PXELinux**) およびプロビジョニングテンプレート(**Satellite Kickstart Default**) を割り当てます。

```
PARTID=$(hammer --csv partition-table list | grep "Kickstart default" |
cut -d, -f1)
PXEID=$(hammer --csv template list --per-page=1000 | grep "Kickstart
default PXELinux" | cut -d, -f1)
SATID=$(hammer --csv template list --per-page=1000 | grep "provision" |
grep "Satellite Kickstart Default" | cut -d, -f1)

for i in $(hammer --csv os list | grep -vi '^ID' | awk -F, {'print
$1'})
```

```
do
   hammer partition-table add-operatingsystem --id="${PARTID}" --
operatingsystem-id="${i}"
   hammer template add-operatingsystem --id="${PXEID}" --
operatingsystem-id="${i}"
   hammer os set-default-template --id="${i}" --config-template-
id="${PXEID}"
   hammer os add-config-template --id="${i}" --config-template-
id="${SATID}"
   hammer os set-default-template --id="${i}" --config-template-
id="${SATID}"
done
```

for ステートメントに **grep** コマンドを追加して、影響を受けるオペレーティングシステムをさらに指定することができます。割当が正しく行われたかどうかを確認するには **hammer os info** コマンドを実行します。

5.9. パラメーター

パラメーターは、プロビジョニング時の Red Hat Satellite の動作を定義します。パラメーターは複数ありますが、詳細は『Red Hat Satellite Host Configuration Guide』を参照してください。以下の例を使用してグローバルパラメーターを設定します。

```
$ hammer global-parameter set --name <param_name> --value <param_value>
```

例5.2 ファイアウォールを無効化するためのグローバルパラメーターの設定

以下のコマンドを実行して **firewall** のグローバルオプションを disabled に設定します。

```
$ hammer global-parameter set --name firewall --value --disabled
```

設定を確認するには、以下のコマンドを実行します。

```
$ hammer global-parameter list
---------|-------------
NAME     | VALUE
---------|-------------
firewall | --disabled
---------|-------------
```

同様に **hammer** を使用して他のパラメータータイプを設定することができます。

- ドメインパラメーターを設定するには以下を使用します。

```
$ hammer domain set-parameter \
--name <param_name> \
--value <param_value> \
--domain <domain_name>
```

- ホストグループのパラメーターを設定するには、以下を使用します。

```
$ hammer hostgroup set-parameter \
--name <param_name> \
--value <param_value> \
--hostgroup <hg_name>
```

- ホストパラメーターを設定するには以下を使用します。

```
$ hammer host set-parameter \
--name <param_name> \
--value <param_value> \
--host <h_name>
```

- スマートクラスパラメーターを更新するには以下を使用します。

```
$ hammer sc-param \
--name <param_name> \
--default-value <param_value>
```

第6章 ホストの管理

ホストは Red Hat Satellite が管理する物理または仮想システムを参照します。以下のセクションでは、**hammer** をしようしたホストとホストのグループの作成、設定方法を説明します。以下の手順を Web UI で行う手順は 『Red Hat Satellite Host Configuration Guide』を参照してください。

6.1. ホストグループの作成

ホストグループは、ホストまたはホストグループの集まりです。共有のホストパラメーターを保持するために、ホストグループを作成することを推奨します。ホストグループのメンバーは、これらのパラメーターを継承するので、ホストの作成じに個別に設定する必要はありません。階層形式でホストグループをネスト化できる点に注意してください。

以下のコマンドはホストグループの作成の基本オプションを表示しています。

```
$ hammer hostgroup create \
--name "<hostgroup_name>" \
--environment "<environment_name>" \
--architecture "<architecture_name>" \
--domain <domain_name> \
--subnet <subnet_name> \
--puppet-proxy <proxy_name> \
--puppet-ca-proxy <ca-proxy_name> \
--operatingsystem "<os_name>" \
--partition-table "<table_name>" \
--medium "<medium_name>" \
--organization-ids <org_ID1>,<org_ID2>... \
--location-ids <loc_ID1>,<loc_ID2>...
```

設定可能なオプションの完全一覧については **hammer hostgroup create --help** を参照してください。ホストグループの作成時に構成できない設定が 2 つあります。

- アクティベーションキーは、ホストグループの作成後に、以下を使用して追加する必要があります。

```
$ hammer hostgroup set-parameter \
--hostgroup "<hostgroup_name>" \
--name "kt_activation_keys" \
--value <key_name>
```

hammer activation-key list を実行してアクティベーションキーの名前を検索します (アクティベーションキーの詳細は4章 *アクティベーションキーの管理*を参照してください)。

- **root** パスワードは、ホストグループにホストを追加する際に指定する必要があります。

例6.1 複数のコンテンツビューに対するホストグループの作成

以下のバッシスクリプトは、ライフサイクル環境ごとにホストグループを作成します。

```
MAJOR="7"
OS=$(hammer --output csv os list | awk -F "," "/RedHat ${MAJOR}/ {print
\$2;exit}")
ARCH="x86_64"
```

```
ORG="ACME"
LOCATIONS="london,munich"
PTABLE_NAME="ptable-acme-os-rhel-server"
DOMAIN="example.com"

hammer lifecycle-environment list --organization "${ORG}" | awk -F "|"
'/[[:digit:]]/ {print $2}' | sed s'/ //' | while read LC_ENV
do
  if [[ ${LC_ENV} == "Library" ]]; then
    continue
  fi

  LC_ENV_LOWER=$(echo ${LC_ENV} | tr '[[:upper:]' '[[:lower:]]')
  ParentID=$(hammer --output csv hostgroup list --per-page 999 | awk -
F"," "(\$3 ~ /^${LC_ENV_LOWER}$/) {print \$1}")

  hammer hostgroup create --name "rhel-${MAJOR}server-${ARCH}" \
    --medium
"${ORG}/Library/Red_Hat_Server/Red_Hat_Enterprise_Linux_${MAJOR}_Server
_Kickstart_${ARCH}_${MAJOR}
    Server" \
    --parent-id ${ParentID} \
    --architecture "${ARCH}" \
    --operatingsystem "${OS}" \
    --partition-table "${PTABLE_NAME}" \
    --subnet "${DOMAIN}" \
    --domain "${DOMAIN}" \
    --organizations "${ORG}" \
    --locations "${LOCATIONS}" \
    --content-view "cv-os-rhel-${MAJOR}Server" \
    --environment-id $(hammer --output csv environment list --per-page
999 | awk -F "," "/KT_$ {ORG}_${LC_ENV}_cv_os_rhel_${MAJOR}Server/
{print \$1}")

  HgID=$(hammer --output csv hostgroup list --per-page 999 | awk -F","
"(\$3 ~ /^${LC_ENV_LOWER}\/rhel-${MAJOR}server-${ARCH}$/) {print \$1}")

  hammer hostgroup set-parameter \
    --hostgroup-id "${HgID}" \
    --name "kt_activation_keys" \
    --value "act-${LC_ENV_LOWER}-os-rhel-${MAJOR}server-${ARCH}"
done
```

6.2. ホストの作成

ホストグループに一般的なパラメーターを設定して、ホスト作成時に必要なオプションの数を減らすことを推奨します。以下のコマンドは、基本的なホストを作成してホストグループに関連付けます。

```
$ hammer host create \
--name "<host_name>" \
--hostgroup "<hostgroup_name>" \
--interface="primary=true, \
            provision=true, \
```

```
            mac=<mac_address>, \
            ip=<ip_address>" \
--organization-id <org_ID> \
--location-id <loc_ID> \
--ask-root-password yes
```

上記のコマンドを実行後に、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ー覧を受け入れます。利用可能なインターフェース設定の一覧については表6.1「--interface オプションの利用可能なキー」を参照してください。

ホストグループのメンバーなしにホストを作成する場合には、「ホストグループの作成」に記載されている追加のオプションを指定します。利用可能なホストのパラメーターは幅広くありますが、詳細は **hammer host create --help** の出力を参照してください。特定のパラメーターの値は、ホストがプロビジョニングするコンピュートリソースの種別により異なります。参考として表6.2「プロビジョニングに固有のホストオプション」を参照してください。

表6.1 --interface オプションの利用可能なキー

キー	説明
type	**Nic::Managed**、**Nic::BMC**、**Nic::Bond** のいずれかのインターフェースタイプを定義します。
name、identifier	インターフェースの ID
mac、ip、domain (または domain_id)、subnet (または subnet_id)	ネットワーク設定、ドメイン、サブネット ID はホストグループから継承できます。
primary、provision、managed、virtual	許容される値は **true** または **false** です。管理ホストは、プライマリーおよびプロビジョニングインターフェースを 1 つ指定する必要があります。
仮想インタフェースに固有のキー	
tag	VLAN タグ。この属性はサブネット VLAN ID より優先されます。

キー	説明
attached_to	このインターフェースが属するインターフェース ID (例: **eth1**)
ボンディングインタフェースに固有のキー	
mode	ボンディングモード。**balance-rr**、**active-backup**、**balance-xor**、**broadcast**、**802.3ad**、**balance-tlb** および **balance-alb** の 1 つ
BMC インターフェースに固有のキー	
provider	BMC プロバイダー。**IPMI** に設定可能
username、password	BMC アクセスの認証情報
Libvirt でプロビジョニングされるホストに固有のキー	
compute_type	インターフェースタイプ。**bridge** または **network** の 1 つ
compute_network または compute_bridge	インターフェース名を指定します。インターフェースタイプに合わせて 1 つ選択します。
compute_model	**virtio**、**rtl8139**、**ne2k_pci**、**pcnet**、**e1000** の 1 つ
RHEV でプロビジョニングされるホストに固有のキー	
compute_name	インターフェース名 (例: **eth0**)

キー	説明
compute_network	クラスターに利用可能なネットワークの中から 1 つ選択します。RHV からの UUID を使用します。
VMware でプロビジョニングされるホストに固有のキー	
compute_type	ネットワークアダプターのタイプ。vSphere のバージョンに合わせて指定します。
compute_network	VMware からのネットワーク ID

表6.2 プロビジョニングに固有のホストオプション

プロバイダー	キー
--compute-attributes オプションのキー	
EC2	flavor_id、image_id、availability_zone、security_group_ids、managed_ip
GCE	machine_type、image_id、network、external_ip
Libvirt	cpus、memory、start
OpenStack	flavor_ref、image_ref、tenant_id、security_groups、network
RHEV	cluster、template、cores、memory、start
VMware	cpus、corespersocket、memory_mb、cluster、path、guest_id、scsi_controller_type、hardware_version、start

プロバイダー	キー
--volume オプションのキー	
Libvirt	poll_name、capacity、format_type
RHEV	size_gb、storage_domain、bootable
VMware	datastore、name、size_gb、thin、eager_zero

例6.2 ボンディングインターフェースペアでのホストの作成

以下の例では、ボンディングインターフェースのペアでホストを作成する方法を示します。インターフェースのボンディングに関する詳しい情報は、『Red Hat Enterprise Linux ネットワークガイド』を参照してください。

```
$ hammer host create --name bondtest \
--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-id 1 \
--location-id 1 \
--ask-root-password yes
```

6.3. ホストコレクションの作成

Red Hat Satellite ホストコレクションは、ホストのグループを指します。以下のコマンドはホストコレクションを作成する際に最低限必要なオプションです。

```
$ hammer host-collection create \
--organization-label <org_label> \
--name <hc_name>
```

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 <ch_ID1>,<ch_ID2>...
```

以下のコマンドを実行して、ホストコレクションとアクティベーションキーを関連付けます (アクティベーションキーの詳細は*4章 アクティベーションキーの管理*を参照してください)。

```
$ hammer activation-key add-host-collection \
--id <ak_ID> \
--host-collection <hc_name>
```

ホストコレクションのグループに含まれるホストは、アクティベーションからの設定を継承します。

6.4. ホストでのリモートジョブの実行

リモートの実行機能により、Satellite Server で任意のコマンドを定義して、リモートホストで実行できます。コマンドは、プロビジョニングテンプレートとよく似たジョブテンプレートで定義されます。デフォルトでジョブテンプレートが複数含まれているので、それを使用することも、リモートホストでのソフトウェアパッケージの操作や Puppet プロセスの起動などのカスタムテンプレートを定義することも可能です。Hammer でこの機能を使用するには、**root** として以下のコマンドを実行してリモートの実行 CLI をインストールします。

```
# yum install tfm-rubygem-hammer_cli_foreman_remote_execution
```

利用可能なジョブテンプレートを表示するには、以下を実行します。

```
$ hammer job-template list
```

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 "<template>" \
--name "<template_name>" \
--provider-type SSH \
--job-category "<category_name>"
```

<template> は、テンプレート定義が含まれるファイルへのパスに置き換えてください。また、カスタムの **<category_name>** を指定するか、既存のカテゴリーの 1 つを選択してください (**Commands**、**Katello**、**Packages**、**Power**、**Puppet** または **Services**)。他に利用可能なパラメーターに関する情報は **hammer job-template create --help** の出力を参照してくださ

い。

カスタムパラメーターでジョブを呼び出すには以下を実行します。

```
$ hammer job-invocation create \
--job-template "<template_name>" \
--inputs <key1>="<value>",<key2>="<value>",... \
--search-query "<query>"
```

リモートジョブに使用するテンプレート名を指定します。キーと値のペアをコンマ区切りのリストとして入力値を指定してください。**hammer job-template info** を実行して、テンプレートに必要なパラメーターが何かを表示します。**<query>** は、どのホストが影響を受けるかを定義するフィルターの表現に置き換えます (例: name ~ rex01)。

例6.3 選択したホストでの httpd サービスの起動

以下の例では、デフォルトの **Service Action - SSH Default** テンプレートをベースにリモートジョブを実行する方法を示します。このテンプレートは、名前に「target」が含まれるホストで **httpd** サービスを機能します。

```
$ hammer job-invocation create \
--job-template "Service Action - SSH Default" \
--inputs service="httpd",action="start" \
--search-query "name ~ target"
```

ジョブの出力を監視するには以下を実行します。

```
$ hammer job-invocation output \
--id <job_ID> \
--host <host_name>
```

hammer job-invocation list の出力で **<job_ID>** を検索します。hammer でのリモートコマンドの実行に関する詳しい情報は **hammer job-template --help** または **hammer job-invocation --help** を参照してください。

第7章 ユーザーおよびパーミッションの管理

Red Hat Satellite では、管理者はユーザーを作成、変更、および削除できます。また、ロールをユーザーに割り当てることによってアクセスパーミッションを設定することもできます。以下のセクションでは、**hammer** を使用してこれらのタスクを実行する方法を説明します。以下の手順を Web UI で行う手順は**『Red Hat Satellite Server 管理ガイド』**の「ユーザーおよびロール」を参照してください。

7.1. ユーザーの作成

Red Hat Satellite ではシステムを使用する個別ユーザーに各種情報を定義できます。**hammer** では、Red Hat Satellite のユーザーを設定するために **user create** および **user update** コマンドがあります。以下のコマンドを使用して新規ユーザーを作成します。

```
$ hammer user create \
--login <user_name> \
--password <user_password> \
--mail <user_mail> \
--auth-source-id 1 \
--organization-ids <org_ID1>,<org_ID2>...
```

--auth-source-id 1 の設定は、ユーザーは内部で認証されることを意味し、外部認証を代わりとして指定することができます。**--admin** オプションを指定して、管理者権限をユーザーに授与します。組織 ID を指定する必要はなく、**update** サブコマンドを使用してユーザーの詳細を変更できます。

ユーザー関連のサブコマンドに関する情報は、**hammer user --help** の出力を参照してください。

7.2. ユーザーグループの作成

複数のユーザーのパーミッションは、ユーザーグループにまとめることで一括して管理できます。また、ユーザーグループ自体をさらにグループ化してパーミッションの階層を作成できます。以下のコマンドを使用して新規ユーザーグループを作成します。

```
$ hammer user-group create --name <usergroup_name>
```

ユーザーグループを追加するには以下のコマンドを実行します。

```
$ hammer user-group add-user --user <user_name> --id <usergroup_id>
```

hammer user-group list を実行してユーザーグループ ID を検索します。同様に、**add-user-group** のサブコマンドを使用するとユーザーグループを追加できます。ユーザーグループ関連の操作に関する情報は **hammer user-group --help** の出力を参照してください。

7.3. ロールの作成

Red Hat Satellite のロールは、パーミッションおよびアクセスレベルを定義します。Satellite には、事前定義済みのロールが複数含まれています。これらのロールを表示するには以下のコマンドを実行します。

```
$ hammer role list
```

ロールに関連付けられたパーミッションを表示するには、以下のコマンドを実行します。

```
$ hammer role filters --id <role_id>
```

<role_id> は、**hammer role list** に出力されたロールの ID です。

カスタムロールを作成するには、以下のコマンドを実行します。

```
$ hammer role create --name <role_name>
```

以下のコマンドでロールにパーミッションフィルターを追加します。

```
$ hammer filter create \
--role <role_name> \
--permission-ids <perm_ID1>,<perm_ID2>...
```

hammer filter available-permissions を使用してロールに追加するパーミッションを検索します。ロールおよびパーミッションの詳細については **hammer role --help** および **hammer filter --help** の出力を参照します。

例7.1 詳細なパーミッションフィルタリング

Red Hat Satellite には、設定済みのユーザーパーミッションを選択したリソースタイプのインスタンスに限定する機能があります。パーミッションフィルターを限定するには、以下のように **--search** オプションを使用します。

```
$ hammer filter create \
--permission-ids 91 \
--search "name ~ ccv*" \
--role qa-user
```

上記のコマンドは **qa-user** ロールに、**ccv** という名前で開始するコンテンツビューにのみ適用されるコンテンツビューを表示、作成、編集、破棄するパーミッションを追加します。詳しい情報は、**『Satellite Server 管理ガイド』**の「詳細なパーミッションフィルタリング」を参照してください。

7.4. ユーザーへのロール割り当て

ユーザーにロールを割り当てるには、以下のコマンドを実行します。

```
$ hammer user add-role --id <user_id> --role <role_name>
```

同様に、ユーザーグループにロールを割り当てることができます。

```
$ hammer user-group add-role --id <usergroup_id> --role <role_name>
```

第8章 エラータの管理

Red Hat 製品のソフトウェアパッケージは、エラータと呼ばれる更新に依存します。これらのソフトウェアパッケージは定期的な間隔、さらに非同期的に発表されます。以下のセクションでは **hammer** を使用してエラータを検出、適用する方法を説明します。以下の手順を Web UI で行う手順は『Red Hat Satellite Host Configuration Guide』を参照してください。

8.1. 利用可能なエラータの検出

全組織で利用可能なエラータを表示するには、以下のコマンドを実行します。

```
$ hammer erratum list
```

例8.1 エラータのフィルタリング

hammer erratum list コマンドは出力一覧のフィルタリングおよび順序付けのオプションが数多く含まれています。たとえば、特定のセキュリティー修正が含まれるエラータを検索するには以下のコマンドを実行します。

```
$ hammer erratum list --cve CVE-2014-0453
```

以下のコマンドにより、指定のバグが含まれる特定の製品に対して適用可能なエラータが表示されますが、この際に対象のセキュリティーエラータが一番上に表示されるように指定のバグが順番に表示されます。

```
$ hammer erratum list \
--product-id 7 \
--search "bug = 1213000 or bug = 1207972" \
--errata-restrict-applicable 1 \
--order "type desc"
```

--search オプションを使用する構文に関する情報は『Red Hat Satellite Host Configuration Guide』を参照してください。また、hammer に実装されるフィルタリングオプションについての詳しい情報は **hammer erratum list --help** の出力を参照してください。

特定のエラータの詳細を表示するには、以下のコマンドを実行します。

```
$ hammer erratum info --id <erratum_ID>
```

<erratum_ID> は、**hammer erratum list** コマンドの出力にあるエラータの一意識別子に置き換えます。名前やリポジトリー名でエラータを特定することも可能です。詳しくは **hammer erratum info --help** の出力を参照してください。

8.2. ホストへのエラータの適用

ホストで利用可能なエラータを表示するには、以下のコマンドを実行します。

```
$ hammer host errata list --host <hostname>
```

選択したエラータをホストに適用するには、以下のコマンドを実行します。

```
$ hammer host errata apply \
--host <hostname> \
--errata-ids <erratum_ID1>,<erratum_ID2>...
```

例8.2 ホストへの全エラータの適用

以下の Bash スクリプトは、ホストで利用可能なエラータすべて適用します (auth01.example.com)。

```
HOST="auth01.example.com"
for i in $(hammer --csv host errata list --host $HOST | grep -vi
'^ID' | awk -F, {'print $1'})
do
  hammer host errata apply --host $HOST --errata-ids $i
done
```

例8.3 セキュリティーアドバイザリーの適用

以下の例は **hammer** を使用してホストにセキュリティー修正を適用する方法を示します。

1. 選択した問題の修正が含まれるエラータ (CVE-2015-3238) を検索します。

```
$ hammer erratum list --cve CVE-2015-3238
-------|----------------|----------|-----------------------
-------
ID     | ERRATA ID      | TYPE     | TITLE
-------|----------------|----------|-----------------------
-------
f30e66 | RHSA-2015:1640 | security | Moderate: pam security
update
-------|----------------|----------|-----------------------
-------
```

2. セキュリティーエラータ (RHSA-2015:1640) がお使いのホスト (auth01.example.com) に適用可能かどうかを確認します。

```
$ hammer host errata list \
--host auth01.example.com \
--search "RHSA-2015:1640"
```

3. ホストにエラータを適用します。

```
$ hammer host errata apply \
--host auth01.example.com \
--errata-ids "RHSA-2015:1640"
```

以下の Bash スクリプトを使用して、該当する場合にはセキュリティーエラータを全ホストに適用することができます (例: RHSA-2015:1640)。

```
ORG="ACME"
RHSA="RHSA-2015:1640"

for i in $(hammer --csv host list --organization $ORG | grep -vi
'^ID' | awk -F, {'print $2'})
do
  hammer host errata apply --host $i --errata-ids $RHSA
done
```

エラータが正しく適用されたことを確認するには、以下のコマンドの出力で適切なタスクを検索します。

```
$ hammer task list
```

選択したタスクの状態を確認するには、以下のコマンドを実行します。

```
$ hammer task progress --id <task_ID>
```

8.3. ホストコレクションへのエラータの適用

ホストコレクションに選択したエラータを適用するには、以下のようにコマンドを入力します。

```
$ hammer host-collection erratum install \
--errata "<erratum_ID1>,<erratum_ID2>,..." \
--name "my-collection" --organization $ORG
```

このコマンドは、Red Hat Satellite 6.2.8 以降で利用できます。

第9章 DOCKER コンテナーの管理

Docker コンテナーは、アプリケーション分離のサンドボックスで、コンテナーイメージにはコンテナーの設定が格納されています。以下のセクションでは **hammer** を使用して Docker コンテナーをプロビジョニングする方法を示します。以下の手順を Web UI で行う手順は『Red Hat Satellite ホスト設定ガイド』を参照してください。

Red Hat Satellite では、コンテナーを Docker プロバイダータイプのコンピュートリソースのみにデプロイできます。コンテナーホストの作成方法については『Red Hat Satellite Host Configuration Guide』を参照してください。このホストをコンピュートリソースとして登録するには、以下のコマンドを実行します。

```
$ hammer compute-resource create
--name <cr_name> \
--organization-ids <org_ID1>,<org_ID2>... \
--location-ids <loc_ID1>,<loc_ID2>... \
--url <cr_url> \
--provider docker
```

以下の構文を使用してコンピュートリソースにコンテナーをプロビジョニングします。

```
$ hammer docker container create \
--name <container_name> \
--compute-resource-id <cr_ID> \
--repository-name <repo_name> \
--tag <tag> \
--command <command>
```

hammer compute-resource list の出力でコンピュートリソース ID を検索します。**<repo_name>** は、Docker イメージが含まれる同期リポジトリーの名前に置き換えます。これには、Docker Hub や内部のレジストリーを参照するカスタムリポジトリー (「カスタムリポジトリーの作成」) または公式の Red Hat イメージリポジトリーを指定することもできます。コンテンツビューからプロビジョニングする場合には、**<repo_name>** はコンテンツビューの名前に置き換えます。コンテンツビューへのイメージの追加に関する情報は「Docker イメージのコンテンツビューへの追加」を参照してください。

コンテナーを起動すると、コンテナーの作成時に **--command** オプションを指定したプロセスが開始されます。コンテナーを起動するには、以下のコマンドを実行します。

```
$ hammer docker container start --id <container_ID>
```

コンテナー関連のオプションの完全一覧は、**hammer docker container --help** コマンドの出力を参照してください。